

# Dossier projet



## GeoChallenge Tracker

Jean Ceugniet

# Table des matières

<b>1</b>	<b>Liste des compétences mises en œuvre</b>	<b>4</b>
1.1	Bloc 1 : Développer une application sécurisée	4
1.1.1	Installer et configurer son environnement de travail	4
1.1.2	Développer des interfaces utilisateur	4
1.1.3	Développer des composants métier	4
1.1.4	Contribuer à la gestion d'un projet	4
1.2	Bloc 2 : Concevoir une application organisée en couches	4
1.2.1	Analyser les besoins et maquetter	4
1.2.2	Définir l'architecture logicielle	4
1.2.3	Concevoir la base de données	4
1.2.4	Développer les composants d'accès aux données	4
1.3	Bloc 3 : Préparer le déploiement sécurisé	5
1.3.1	Préparer et exécuter les tests	5
1.3.2	Préparer et documenter le déploiement	5
1.3.3	Contribuer à la mise en production DevOps	5
<b>2</b>	<b>Expression du besoin et présentation client</b>	<b>6</b>
2.1	Client principal	6
2.2	Attentes utilisateurs	6
2.3	Personas	6
2.3.1	Persona 1 : le passionné des grands défis	6
2.3.2	Persona 2 : la monitrice d'escalade	6
2.3.3	Persona 3 : l'optimisateur rationnel	6
2.4	Pool de testeurs	7
2.5	Analyse de l'existant	7
2.6	Valeur ajoutée de GeoChallenge Tracker	7
2.7	Objectifs fonctionnels	7
2.8	Objectifs techniques	7
<b>3</b>	<b>Gestion de projet</b>	<b>8</b>
3.1	Méthodologie	8
3.2	Planning	8
3.3	Suivi et outils	8
3.4	Indicateurs de qualité	8
<b>4</b>	<b>Spécifications fonctionnelles</b>	<b>9</b>
4.1	Contraintes et livrables	9
4.1.1	Contraintes	9
4.1.2	Livrables	9
4.2	Architecture logicielle	10
4.2.1	Vue d'ensemble	10
4.2.2	Composants backend	10
4.2.3	Composants frontend	10
4.3	Charte graphique	11
4.3.1	Page d'accueil	12
4.3.2	Register / Login	13
4.3.3	Page d'accueil - Contenu complet	14
4.3.4	Enchaînement des écrans	14
4.4	Structures de données	15
4.4.1	Diagramme entités-associations	15
4.4.2	Collections MongoDB	15
4.5	Script création base de données	16
4.5.1	Objectifs	16
4.5.2	Pré-requis & variables d'environnement	17
4.5.3	Données seedées	17
4.5.4	Index créés (via seeding)	17
4.5.5	Exécution (local, sans Docker)	18
4.5.6	Exécution (via Docker Compose)	18
4.5.7	Composants clés (extraits commentés)	18
4.5.8	Bonnes pratiques & sécurité	19
4.5.9	Vérification après exécution	19
4.5.10	Tests associés au seeding	20

4.6	Diagramme cas d'utilisations	21
4.7	Diagrammes de séquence	22
4.7.1	Authentification	22
4.7.2	Import GPX	23
4.7.3	Calcul de progression	24
<b>5</b>	<b>Spécifications techniques</b>	<b>25</b>
5.1	Backend	25
5.2	Frontend	25
5.3	Base de données	25
5.4	Infrastructure Docker	26
5.5	Intégrations externes	27
5.6	Sécurité multicouche	27
5.6.1	Authentification JWT	27
5.6.2	Validation des entrées	27
5.6.3	Protection CORS	28
5.6.4	Rate Limiting	28
5.7	Optimisations performance	28
5.7.1	Cache référentiels	28
5.7.2	Agrégations MongoDB optimisées	28
5.7.3	Map tiles caching	29
<b>6</b>	<b>Réalisations candidat</b>	<b>32</b>
6.1	Interfaces utilisateur et code	32
6.1.1	Composant carte interactive	32
6.1.2	Liste des challenges	34
6.1.3	Détails d'un challenge	38
6.2	Composants métier	41
6.2.1	Parser GPX multi-namespace	41
6.2.2	Moteur de règles AST	42
6.3	Accès aux données	44
6.3.1	Récupération de caches	44
6.3.2	Calcul d'un snapshot de progression	45
6.4	Autres composants	48
6.4.1	Données d'altimétrie	48
<b>7</b>	<b>Sécurité de l'application</b>	<b>50</b>
7.1	Authentification et autorisation	50
7.1.1	JWT avec refresh tokens	50
7.1.2	Gestion des rôles	50
7.2	Protection des données	50
7.2.1	Hashage des mots de passe	50
7.2.2	Isolation par utilisateur	50
7.3	Validation et sanitisation	50
7.3.1	Validation Pydantic stricte	50
7.3.2	Protection XSS	50
7.4	Protection contre les attaques	51
7.4.1	Rate limiting	51
7.4.2	CSRF Protection	51
7.4.3	SQL/NoSQL Injection	51
7.5	Audit et monitoring	51
7.5.1	Logs de sécurité	51
7.5.2	Métriques	51
<b>8</b>	<b>Plan de tests</b>	<b>52</b>
8.1	Stratégie de tests	52
8.2	Tests unitaires backend	52
8.3	Tests d'intégration API	53
8.4	Tests frontend	54
8.5	Tests de charge	56
<b>9</b>	<b>Jeu d'essai</b>	<b>57</b>
<b>10</b>	<b>Veille sur les vulnérabilités</b>	<b>59</b>
10.1	Outils de veille	59
10.2	Vulnérabilités identifiées et corrigées	59

10.3 Procédures de réponse . . . . .	59
10.4 Bonnes pratiques adoptées . . . . .	59
<b>11 Conclusion . . . . .</b>	<b>60</b>
11.1 Bilan du projet . . . . .	60
11.2 Difficultés rencontrées . . . . .	60
11.3 Perspectives d'évolution . . . . .	60
11.4 Satisfaction personnelle . . . . .	60
11.5 Avis des utilisateurs . . . . .	60

# 1 Liste des compétences mises en œuvre

---

## 1.1 Bloc 1 : Développer une application sécurisée

### 1.1.1 Installer et configurer son environnement de travail

- Environnement Docker multi-services avec docker-compose
- Configuration Python 3.11, FastAPI, MongoDB Atlas, Vue.js
- Outils : VS Code, ESLint/Prettier, debugger FastAPI
- Gestion des secrets via variables d'environnement

*Objectifs* : séparation stricte de la configuration et des secrets en production, non dissémination des secrets entre les différentes entités impliquées (github, services tiers liés au déploiement)

### 1.1.2 Développer des interfaces utilisateur

- Framework Vue 3 avec Composition API et TypeScript
- Design responsive mobile-first (Tailwind CSS, Flowbite)
- Cartographie interactive Leaflet avec clustering et tile caching
- Gestion d'état Pinia
- Optimisation des performances dans une démarche de *green computing*

### 1.1.3 Développer des composants métier

- Parser GPX multi-namespace pour extraction des informations des caches
- Moteur de règles AST pour évaluation des challenges
- Moteur de filtrage par condition et gestion des doublons
- Service de progression avec calculs et projections, génération de séries temporelles
- Algorithme de scoring pour identification des cibles

### 1.1.4 Contribuer à la gestion d'un projet

- Méthodologie Agile adaptée, suivi de projet Github (issues, milestones, kanban)
- CI/CD avec GitHub Actions
- Information des acteurs du projet via Github Actions
- Documentation technique complète
- Suivi qualité (linters, formatters)

## 1.2 Bloc 2 : Concevoir une application organisée en couches

### 1.2.1 Analyser les besoins et maquetter

- Analyse du domaine geocaching, et des besoins des pratiquants de challenges
- Cahier des charges détaillé
- Wireframes et prototypage

### 1.2.2 Définir l'architecture logicielle

- Architecture 3-tiers : Vue / FastAPI / MongoDB
- Séparation des responsabilités
- Pattern Repository, architecture hexagonale

### 1.2.3 Concevoir la base de données

- Modélisation documentaire NoSQL / MongoDB, adaptée à la variabilité des caches et de leurs attributs
- Indexation stratégique (géospatiale, composée)
- Agrégations complexes, optimisation, dénormalisation

### 1.2.4 Développer les composants d'accès aux données

- ODM avec Pydantic
- CRUD générique MongoDB
- Transactions, cache référentiels
- Sécurisation des accès à la base de données par whitelist

## **1.3 Bloc 3 : Préparer le déploiement sécurisé**

### **1.3.1 Préparer et exécuter les tests**

- Tests unitaires (pytest, Vitest)
- Tests d'intégration et E2E
- Tests de coverage
- Tests utilisateurs

### **1.3.2 Préparer et documenter le déploiement**

- Dockerisation multi-stage
- Documentation complète
- Configuration externalisée

### **1.3.3 Contribuer à la mise en production DevOps**

- CI/CD automatisé GitHub Actions (tests, build, déploiement automatisé, diffusion automatisée d'information)
- Déploiement blue-green sur VPS
- Monitoring et rollback

## 2 Expression du besoin et présentation client

### 2.1 Client principal

Avant de détailler les attentes précises, il est essentiel de présenter le public cible auquel l'application s'adresse.

Le géocaching est une chasse au trésor moderne qui utilise un GPS ou un smartphone pour localiser des contenants cachés (appelés "géocaches") dissimulés partout dans le monde par d'autres participants.

La communauté visée regroupe des géocacheurs passionnés qui voient dans les *challenges* un « jeu dans le jeu » : un niveau supplémentaire imposant des critères exigeants (difficulté/terrain, type, localisation, attributs) sur des périodes longues. Pour les plus investis, gérer ces contraintes devient un défi en soi. Les solutions existantes (ex. Project-GC premium) sont puissantes mais payantes. Sans outil adapté, le suivi relève d'une tâche chronophage avec risque d'erreurs.

GeoChallenge Tracker se positionne comme une solution accessible et intuitive, permettant à ces utilisateurs d'optimiser leur expérience sans multiplier les feuilles de calcul ou s'appuyer uniquement sur leur mémoire.

### 2.2 Attentes utilisateurs

De cette analyse des pratiques, ressortent plusieurs attentes fortes de la part des utilisateurs :

- un outil simple d'utilisation, mais puissant dans le traitement et le croisement de données,
- un suivi automatisé et visuel de leur progression,
- la possibilité d'identifier les caches qui permettent d'avancer sur plusieurs challenges à la fois,
- une cartographie efficace permettant de filtrer, de regrouper et de localiser les caches pertinentes,
- un respect strict de la confidentialité et de la sécurité de leurs données.

Afin de garantir que l'outil réponde réellement aux besoins, un **pool de testeurs expérimentés** est en cours de recrutement. Il inclut des géocacheurs aux profils variés :

- plusieurs joueurs, dont certains ayant trouvé **plus de 50 000 caches**, classés parmi les premiers au niveau national,
- des figures reconnues de la communauté (ex. membres actifs, auteurs, géocacheurs classés dans le top 5 national),
- des profils diversifiés pour couvrir différents styles de jeu (grands voyageurs, spécialistes des caches T5, optimiseurs rationnels, etc.).

Ce panel permettra de confronter l'application à des pratiques intensives et variées, et de valider sa pertinence auprès de la communauté cible.

### 2.3 Personas

Pour mieux incarner ces attentes, plusieurs profils types (personas) permettent d'illustrer la diversité des besoins.

#### 2.3.1 Persona 1 : le passionné des grands défis

**Âge** : 45 ans. **Profession** : Cadre supérieur. **Profil** : Géocacheur depuis plus de 15 ans, engagé dans des challenges de grande ampleur, impliquant plusieurs régions ou pays. Planifie ses voyages en fonction des caches à trouver. **Besoins** : Un suivi précis de l'avancement sur des objectifs étendus dans le temps et l'espace, avec des projections claires. Centralisation des données éparées provenant de multiples zones géographiques.

#### 2.3.2 Persona 2 : la monitrice d'escalade

**Âge** : 34 ans. **Profession** : Monitrice d'escalade. **Profil** : Fan de caches T5, adore les défis nécessitant des compétences techniques en hauteur ou en spéléologie. Voyage souvent et recherche l'adrénaline. **Besoins** : Filtrer efficacement les caches selon les attributs extrêmes, voir rapidement ce qui reste à faire pour valider un challenge T5.

#### 2.3.3 Persona 3 : l'optimisateur rationnel

**Âge** : 39 ans. **Profession** : Ingénieur en logistique. **Profil** : Participe à de nombreux challenges simultanément, mais privilégie l'efficacité. Cherche à réduire les déplacements inutiles en identifiant des caches qui permettent d'avancer sur plusieurs challenges à la fois. **Besoins** : Un outil capable de croiser les critères pour trouver les "caches multi-bénéfices", et de calculer le meilleur compromis entre distance parcourue et progression réalisée.

## 2.4 Pool de testeurs

On peut retrouver l'ensemble des caractéristiques de ces personas en s'appuyant sur un panel de testeurs représentatifs de la communauté, constitué pour valider l'outil en conditions réelles. Il regroupe des géocacheurs aux profils variés, allant de joueurs passionnés à des figures reconnues du classement national. Ces retours permettent de confronter l'outil à des cas d'usage concrets et exigeants.

Pseudo	Caches trouvées	Caches posées	Rang national
Almani06	10 921	42	—
Arnokovic	86 909	303	4 (Fr)
audeclar	9 351	304	—
falbala20220	17 336	152	335 (Fr)
Kidoulo	10 932	96	—
le sudiste	35 203	46	70–75 (eq Fr)
magiKache	7 306	273	—
MLRFamily	6 279	7	—
Orchidée83	4 842	9	—
oTo66	4 382	89	—
Phiphi13	11 290	292	—

Ce panel de testeurs apporte :

- la vision de **joueurs très expérimentés** (dont un classé **n°4 en France** en nombre de caches trouvées),
- l'expertise de **poseurs prolifiques**,
- et l'expérience de profils plus **familiaux ou intermédiaires**, garantissant une couverture diversifiée.

Ces retours permettent de vérifier à la fois la **pertinence fonctionnelle**, la **fiabilité technique** et l'**ergonomie** de l'outil.

## 2.5 Analyse de l'existant

**Project-GC** (seule alternative notable) :

- **Points forts** : Statistiques exhaustives, tableaux, gamification
- **Limites** : Interface dense, personnalisation limitée, pas de projection, accès premium requis, cartographie sans clustering d'où lecture difficile

## 2.6 Valeur ajoutée de GeoChallenge Tracker

- **Ergonomie mobile-first** : Cartes OSM, clustering, filtrage
- **Personnalisation** : Tâches définies par l'utilisateur (grammaire AST)
- **Projections temporelles** : Estimation de complétion
- **Gratuité** : Pas d'abonnement, APIs ouvertes

## 2.7 Objectifs fonctionnels

- **Import de données** : Parser GPX/ZIP multi-namespaces
- **Gestion des challenges** : CRUD avec conditions AST
- **Suivi de progression** : Calculs temps réel et historique, représentation graphique intuitive
- **Identification de cibles** : Algorithme d'optimisation, priorisation automatisée des candidats
- **Cartographie** : Visualisation interactive des caches, clustering pour faciliter la lecture de cartes
- **Itinérance** : Doit permettre un usage aisé sur mobile, y compris en situation d'itinérance

## 2.8 Objectifs techniques

- **Performance** : Temps de réponse < 200ms
- **Scalabilité** : Support de 100k+ caches par utilisateur
- **Sécurité** : Authentification JWT, validation stricte
- **Disponibilité** : 99.9% uptime
- **Compatibilité** : Mobile et desktop modernes



## 3 Gestion de projet

---

### 3.1 Méthodologie

Approche **Agile adaptée** au développement solo

- Sprints de 1 semaine
- User stories priorisées (MoSCoW)
- Revue hebdomadaire et rétrospective

### 3.2 Planning

**Sprint 1-2** (Juillet 2025) : Infrastructure et authentification

- Setup Docker et MongoDB
- API d'authentification JWT
- Tests unitaires

**Sprint 3-4** (Juillet 2025) : Import et modélisation

- Parser GPX multi-namespace
- Modèles de données
- Routes CRUD caches

**Sprint 5-6** (Août 2025) : Challenges et progression

- Grammaire AST
- Moteur d'évaluation
- Calcul de progression

**Sprint 7-8** (Août 2025) : Frontend et carte

- Interface Vue 3
- Intégration Leaflet
- Clustering et filtres

**Sprint 9-10** (Septembre 2025) : Frontend, optimisation et déploiement

- Interface Vue 3 / Leaflet
- Tests E2E
- CI/CD GitHub Actions
- Documentation

### 3.3 Suivi et outils

- **Versioning** : Git avec branches backend/frontend + feature
- **Tickets** : GitHub Issues et Projects
- **CI/CD** : GitHub Actions
- **Communication** : Discord pour les updates
- **Documentation** : Markdown versionné + génération automatisé de PDF

### 3.4 Indicateurs de qualité

- **Coverage tests** : > 70%
- **Dette technique** : < 5%
- **Performance** : < 200ms P95
- **Bugs critiques** : 0 en production
- **Documentation** : 100% des APIs

## 4 Spécifications fonctionnelles

---

### 4.1 Contraintes et livrables

#### 4.1.1 Contraintes

##### Contraintes temporelles

- Deadline rendu dossier au 10/10/2025
- Définition d'une branche git stable à figer au 10/10/2025
- Recueil des retours testeurs et intégration au dossier avant la deadline

##### Contraintes techniques

- Utilisation d'APIs externes (OpenStreetMap, OpenTopoData)
- Respect des rate limits des services tiers
- Compatibilité avec les formats GPX standards

##### Contraintes fonctionnelles

- Support de 5 namespaces GPX différents
- Limitation du temps de parsing pour l'expérience utilisateur
- Contrôle des imports utilisateurs en taille pour l'expérience utilisateur (temps d'exécution, protection contre les crashes)

##### Contraintes de sécurité

- Authentification obligatoire
- Isolation des données par utilisateur
- Protection contre les injections

#### 4.1.2 Livrables

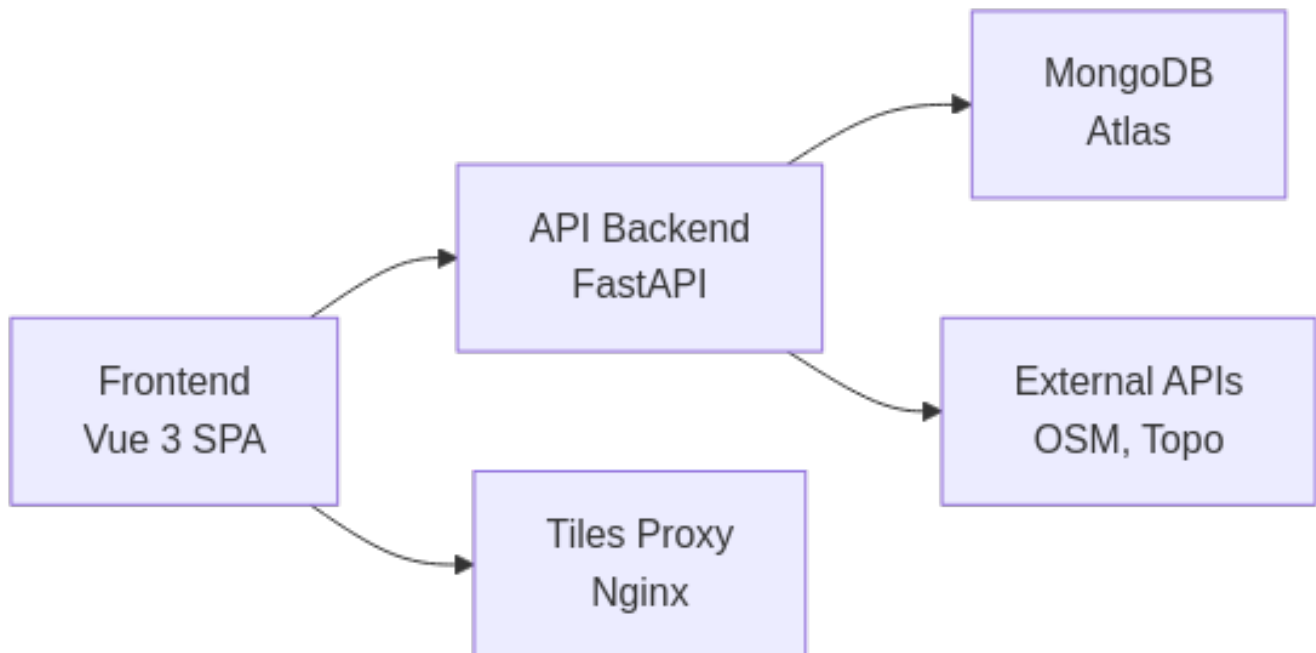
- Dossier projet
- Projet sous la forme d'une branche github figée à date de rendu
- Version logicielle fonctionnelle

## 4.2 Architecture logicielle

### 4.2.1 Vue d'ensemble

#### Architecture 3-tiers moderne

- **Présentation** : SPA Vue.js responsive
- **Métier** : API REST FastAPI
- **Données** : MongoDB Atlas



### 4.2.2 Composants backend

#### Routes API

- /auth : Authentification JWT
- /caches : CRUD et recherche
- /my/challenges : Gestion personnelle
- /my/progress : Suivi de progression
- /my/targets : Suggestions optimisées

#### Services métier

- GPXParser : Extraction multi-namespace
- ChallengeService : Gestion AST
- ProgressService : Calculs et projections
- TargetService : Algorithme de scoring

#### Composants techniques

- Security : JWT et permissions
- Database : Connexion MongoDB
- Cache : Référentiels en mémoire

### 4.2.3 Composants frontend

#### Pages principales

- Home : Landing et présentation
- ImportGPX : Upload et parsing
- MyChallenges : Liste et gestion
- Progress : Tableaux de bord

## Composants réutilisables

- MapBase : Wrapper Leaflet
- MarkerCluster : Regroupement
- FilterBar : Critères de recherche
- ProgressChart : Graphiques

## Store Pinia

- authStore : État authentification
- cacheStore : Données geocaching
- challengeStore : Challenges actifs

## 4.3 Charte graphique

Dans la conception de l'interface, une approche **mobile first** a été privilégiée. Ce choix repose sur plusieurs arguments. Tout d'abord, le mobile first conduit naturellement à une interface centrée sur le **contenu** et une **ergonomie simple et efficace**, ce qui correspond à l'objectif du projet. Cette orientation a été pensée en opposition à certains outils existants, comme *Project-GC*, dont le contenu est riche mais présenté de manière très dense : abondance de tableaux, polices réduites, couleurs juxtaposées dans de petites cases. Le résultat est puissant mais peu lisible sur mobile, et difficilement exploitable en situation de mobilité.

Ensuite, l'application est amenée à être utilisée en **itinérance**, notamment pour la consultation cartographique lors de déplacements. Une ergonomie pensée pour le bureau (desktop) n'aurait donc pas été adaptée. Au contraire, l'interface mobile first assure une **utilisation fluide en contexte de terrain**, tout en restant **parfaitement utilisable sur desktop**. De plus, il est toujours plus aisé d'ajouter des **media queries** pour enrichir l'expérience sur grand écran que de tenter de linéariser une interface multi-colonnes afin de la rendre utilisable sur smartphone.

Le design graphique repose volontairement sur une base **sobre et lisible**. Les polices choisies sont exclusivement **sans-serif**, le contraste entre texte et arrière-plan est maximal, et des nuances de gris ainsi que des variations de taille viennent hiérarchiser certains éléments. Ce travail permet une lecture confortable dans la majorité des situations. Une **passé d'accessibilité dédiée** pourra être envisagée dans une version ultérieure, afin de prendre en compte des besoins spécifiques (daltonisme, contrastes renforcés, lecteurs d'écran, etc.).

La **navigation** a été conçue pour optimiser l'espace : des **menus dépliant** permettent de libérer la surface utile, tandis que des **icônes thématiques par section** facilitent le repérage et la mémorisation visuelle des fonctionnalités. Cet équilibre entre sobriété et guidage visuel constitue un gage d'ergonomie pour des utilisateurs variés, du joueur occasionnel au géocacheur expérimenté.

Enfin, le **design épuré** retenu a aussi une incidence sur les performances. Il limite le nombre d'éléments graphiques et réduit donc les **temps de chargement**, la **bande passante consommée**, et même le **temps de mise en page (layout)** dans le navigateur. Cela correspond à une démarche naturellement plus efficiente sur le plan environnemental : un premier pas vers le **green IT**. Si la gestion efficace de la cartographie (notamment via le **tile caching**) constitue déjà un levier important, d'autres pistes d'optimisation (caching applicatif, traitement côté client mieux maîtrisé) pourront être explorées dans la suite du projet. Le design évite aussi toute lourdeur inutile : **aucune image lourde** n'est utilisée et les animations sont réduites au strict minimum.

Le **nom** et le **logo** de l'application ont fait l'objet d'une réflexion spécifique. Le nom *GeoChallenge Tracker* exprime de manière explicite l'objectif du logiciel : aider l'utilisateur à suivre sa progression dans des challenges géocaching. Il se simplifie facilement en *GC Tracker*, une abréviation à la fois courte et parlante. L'acronyme *GC* résonne immédiatement auprès de la communauté, puisqu'il est déjà largement utilisé pour désigner *geocaching* ou *geocache*.

Le logo reprend la base d'un **marqueur de position**, symbole universel de la géolocalisation, afin de rappeler immédiatement le contexte de l'activité. Plusieurs alternatives avaient été envisagées (escalier stylisé, coupe sportive, médaille, badge, ou encore dégradé de couleur), mais elles ont été écartées : trop complexes pour rester lisibles à petite taille, ou trop voyantes pour conserver une interface sobre. La solution finalement retenue repose sur un **histogramme intégré dans le marqueur**, exprimant à la fois la **notion de progression** et celle de **succès**. Les trois barres colorées montantes (rouge → jaune → vert) traduisent visuellement plusieurs idées : le **changement de statut** (échec → progression → réussite), l'**augmentation du taux de réalisation**, et la symbolique d'un **escalier**, associée à l'idée d'avancement et d'accomplissement. L'usage des couleurs s'appuie sur des codes universels, proches de ceux des feux de circulation ou des indicateurs sportifs, ce qui renforce l'immédiateté de la compréhension.

Ainsi, l'ensemble des choix graphiques répond à un double objectif : proposer une **interface claire et efficace en mobilité**, tout en véhiculant une **identité visuelle forte** qui parle directement à la communauté géocaching.

#### 4.3.1 Page d'accueil

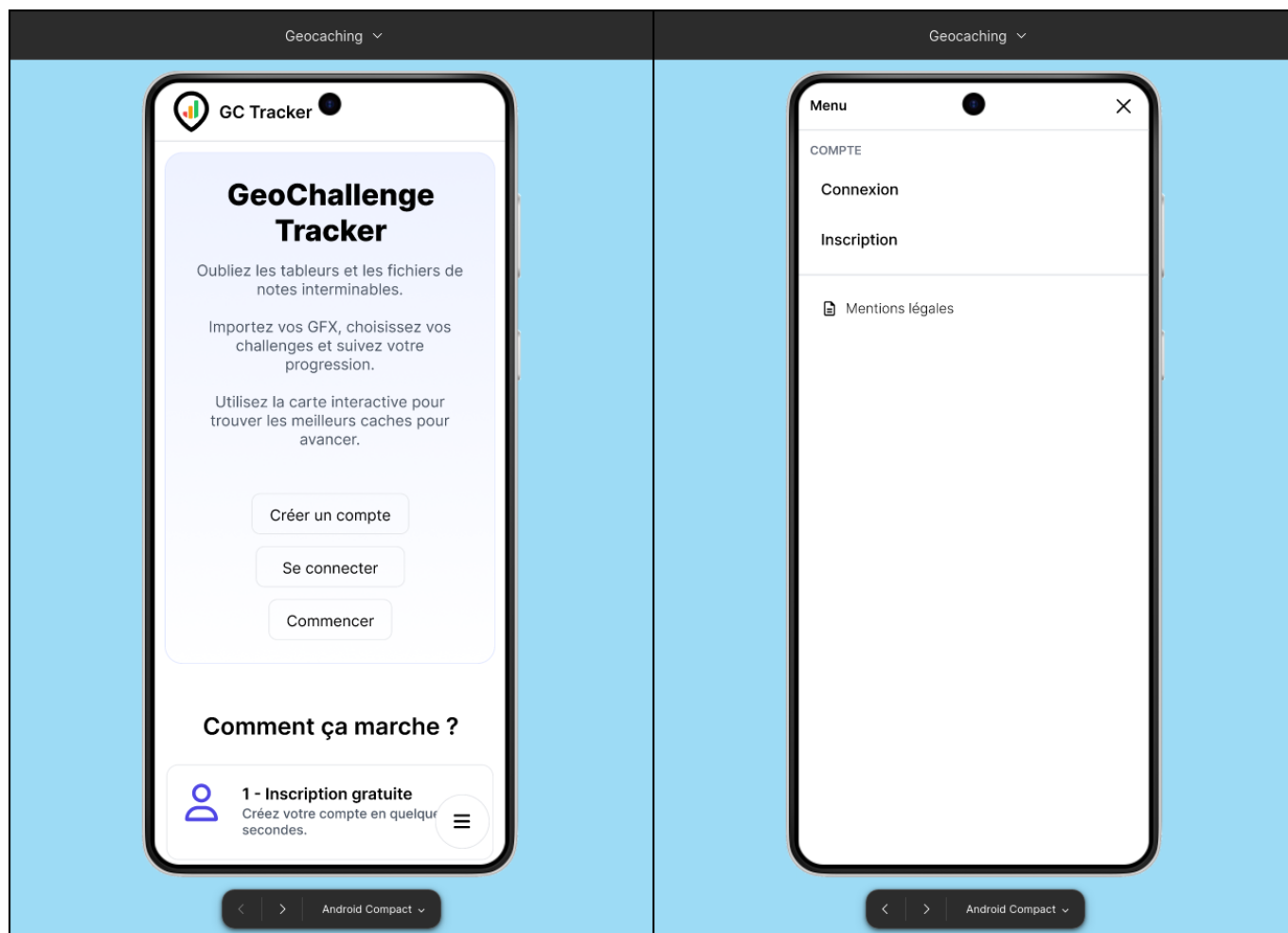


Figure 1 : Accueil+Menu non loggé

L'écran d'accueil, accessible sans connexion, sert avant tout de **teaser pour les géocacheurs**. Le haut de page met en avant la **simplicité du logiciel**, sa capacité à **remplacer avantageusement l'existant**, ainsi que son caractère **interactif** et l'intérêt de sa **cartographie**. Le menu haut propose immédiatement les **actions essentielles** : créer un compte, se connecter, ou commencer. Le contenu principal illustre les **étapes d'utilisation typiques**, avec une explication succincte de chaque phase. Enfin, le **menu, volontairement réduit**, ne contient que les liens connexion / inscription, ainsi que les *mentions légales*, qui doivent rester *disponibles en toutes circonstances*.

### 4.3.2 Register / Login

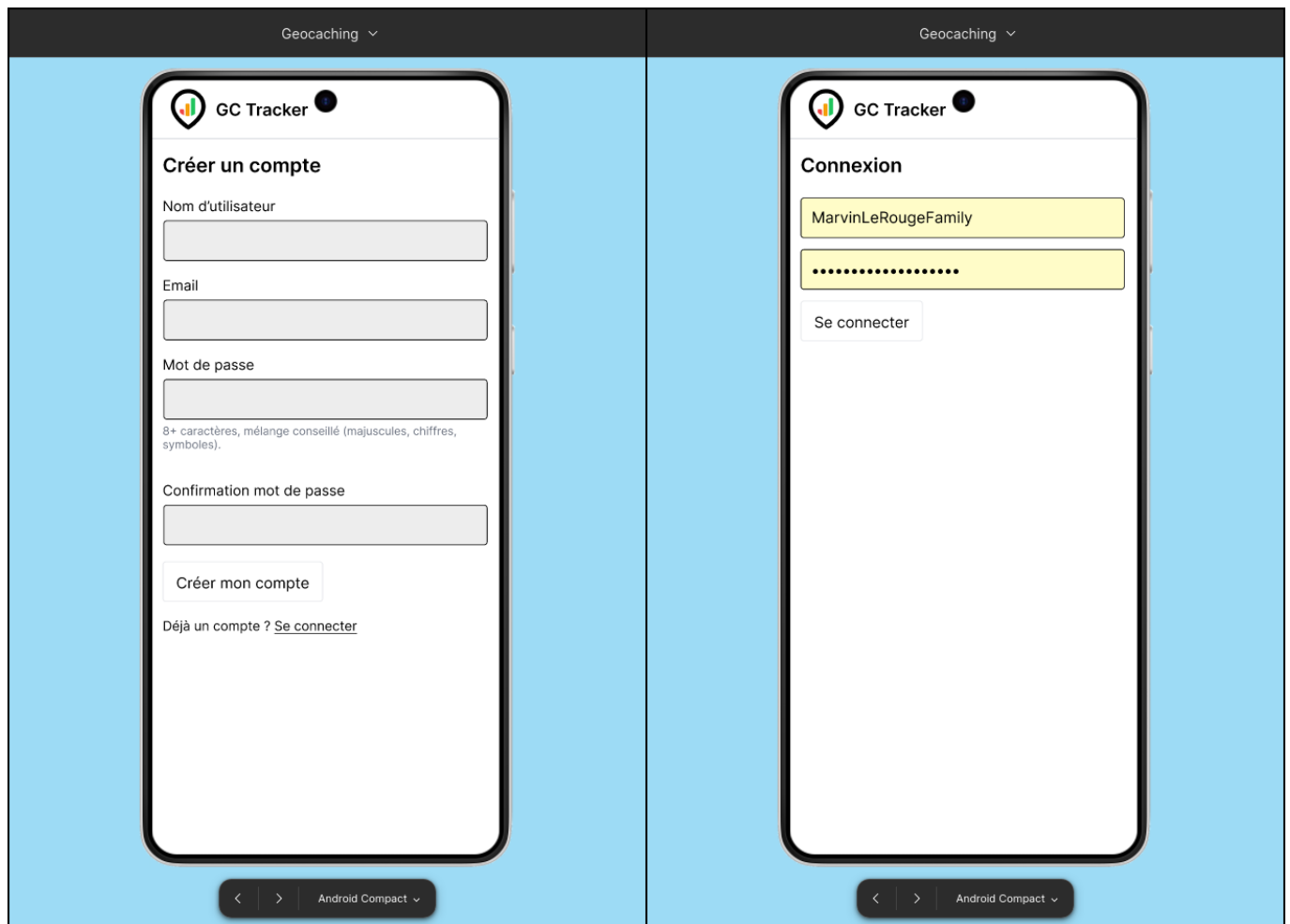


Figure 2 : Register / Login

Les écrans d'inscription et de connexion suivent volontairement une **approche minimaliste**. Leur objectif est d'aller droit au but, conformément aux bonnes pratiques du domaine. Le formulaire d'inscription demande une confirmation du mot de passe, afin de **réduire les risques d'erreur** et d'**éviter la frustration** liée à un premier échec de connexion. La **sobriété visuelle** met en avant la **lisibilité** et la **fluidité** du parcours utilisateur.

### 4.3.3 Page d'accueil - Contenu complet



Figure 3 : Accueil loggé - Contenu complet

La page d'accueil en version complète met en avant une **vision panoramique des fonctionnalités** disponibles. Le design conserve une structure claire, tout en affichant davantage d'éléments contextualisés pour l'utilisateur connecté. L'objectif est de fournir une **vue globale** qui associe lisibilité et exhaustivité, sans surcharger visuellement la page. Cette maquette met en lumière la **progression naturelle du parcours utilisateur**, depuis l'accès invité jusqu'à l'exploration complète des fonctionnalités.

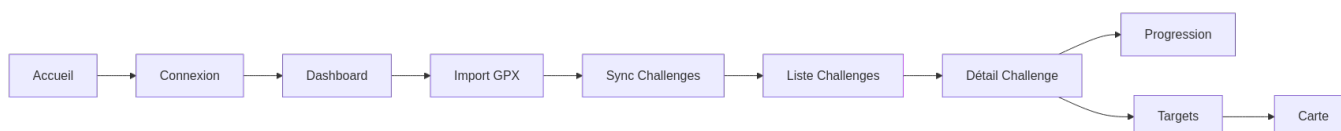
En conclusion, la conception graphique de GeoChallenge Tracker illustre une démarche pragmatique : partir des besoins des utilisateurs, proposer une interface simple et claire, et s'assurer que l'expérience reste fluide sur mobile comme sur desktop. L'usage de maquettes Figma a permis de matérialiser ces choix très tôt, en validant les parcours essentiels sans chercher à couvrir exhaustivement toutes les pages de l'application.

Cette sobriété va de pair avec une réflexion sur les performances et l'impact environnemental. Dans une logique de **green IT**, toutes les ressources graphiques ont été optimisées, depuis les maquettes elles-mêmes jusqu'au logo en SVG, afin de réduire la taille des fichiers et d'accélérer leur affichage. L'ensemble contribue à un rendu plus léger et donc plus respectueux des contraintes de bande passante et de consommation.

Ces choix traduisent une volonté de concilier **ergonomie, efficacité et responsabilité**, en offrant à la communauté des géocacheurs un outil accessible, lisible et durable, conçu dès l'origine avec une attention particulière portée à la simplicité et à la performance.

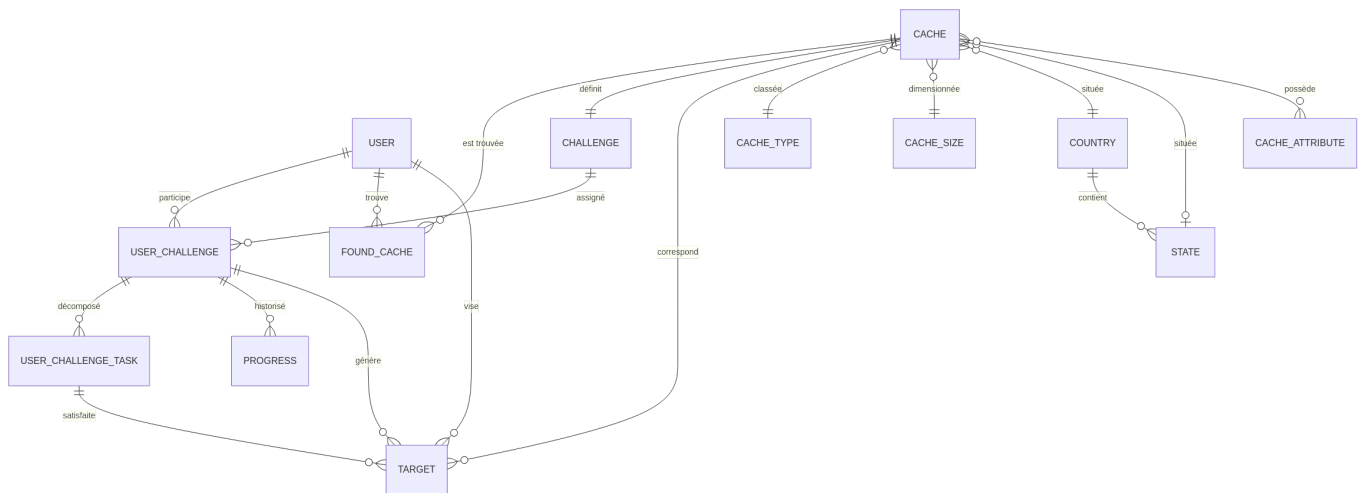
L'ensemble du prototype est consultable en ligne, [sur Figma](#).

### 4.3.4 Enchaînement des écrans



## 4.4 Structures de données

### 4.4.1 Diagramme entités-associations



*Note* : MCD et MPD complets disponibles en annexe 2

### 4.4.2 Collections MongoDB

#### users

```
{
  _id: ObjectId,
  username: String,
  email: String,
  password_hash: String,
  role: "user"|"admin",
  location: {
    latitude: Number,
    longitude: Number
  },
  created_at: Date,
  updated_at: Date
}
```

#### caches

```
{
  _id: ObjectId,
  gc: String,           // GC12345
  name: String,
  loc: {
    type: "Point",
    coordinates: [lon, lat]
  },
  difficulty: Number,   // 1.0 - 5.0
  terrain: Number,      // 1.0 - 5.0
  type_id: ObjectId,    // ref cache_types
  size_id: ObjectId,    // ref cache_sizes
  attributes: [{
    attribute_doc_id: ObjectId,
    is_positive: Boolean
  }],
  owner: String,
  placed_at: Date,
  elevation: Number
}
```



## user\_challenges

```
{
  _id: ObjectId,
  user_id: ObjectId,
  challenge_id: ObjectId,
  status: "pending"|"accepted"|"dismissed"|"completed",
  computed_status: String,
  progress: Number,      // 0-100
  notes: String,
  started_at: Date,
  completed_at: Date
}
```

## user\_challenge\_tasks

```
{
  _id: ObjectId,
  user_challenge_id: ObjectId,
  title: String,
  expression: Object,    // AST
  status: "todo"|"in_progress"|"done",
  progress: Number,
  order: Number,
  constraints: {
    min_count: Number
  },
  metrics: {
    current_count: Number,
    target_count: Number
  }
}
```

## progress

```
{
  _id: ObjectId,
  user_challenge_id: ObjectId,
  checked_at: Date,
  overall_percent: Number,
  tasks: [{
    task_id: ObjectId,
    percent: Number,
    current_count: Number,
    aggregate: {
      total: Number,
      target: Number,
      unit: String
    }
  }],
  estimated_completion_at: Date
}
```

## 4.5 Script création base de données

Ce document décrit **comment initialiser** la base MongoDB (référentiels, compte admin) et **assurer la création des index** au démarrage, à partir des scripts fournis dans le backend.

### 4.5.1 Objectifs

L'initialisation de la base de données poursuit plusieurs objectifs essentiels, garantissant à la fois la robustesse et la sécurité de l'application.

- **Tester la connexion** à MongoDB et arrêter proprement en cas d'échec.
- **Créer/mettre à jour les index** (unicité, géo, texte, partiels, collation).
- **Seeder les référentiels** (types, tailles, attributs) et **l'utilisateur admin**.

## 4.5.2 Pré-requis & variables d'environnement

Avant d'exécuter les scripts, certains prérequis et variables doivent être définis pour assurer une configuration cohérente.

- **MongoDB Atlas** (ou compatible) accessible via URI.
- Variables d'environnement attendues :
  - MONGODB\_URI (ou équivalent utilisé par l'app)
  - ADMIN\_USERNAME, ADMIN\_EMAIL, ADMIN\_PASSWORD
  - (éventuellement) SMTP\_\* si la vérification email est testée

**Sécurité** : ne versionnez jamais vos valeurs réelles. Utilisez un `.env local` ou des repository secrets côté CI/CD.

## 4.5.3 Données seedées

Les données de référence sont stockées dans des fichiers JSON et injectées lors du seeding initial.

Répertoires (backend) : `data/seeds/*.json`

- `cache_types.json`
- `cache_sizes.json`
- `cache_attributes.json`

Les jeux de données sont insérés **si la collection est vide** (ou **forcés** avec l'option `--force`).

## 4.5.4 Index créés (via seeding)

Le script assure également la création des index nécessaires, de manière idempotente, afin de garantir les performances attendues.

- **users** : unicité insensible à la casse sur username, email (collation) ; index géo location (2dsphere) ; flags is\_active, is\_verified.
- **countries** : name (unique), code (unique partiel si string).
- **states** : country\_id ; (country\_id, name) unique ; (country\_id, code) unique partiel.
- **cache\_attributes** : cache\_attribute\_id (unique), txt (unique partiel), name.
- **cache\_sizes** : name (unique), code (unique partiel).
- **cache\_types** : name (unique), code (unique partiel).
- **caches** : GC (unique), loc (2dsphere), type\_id, size\_id, country\_id, state\_id, (country\_id, state\_id), difficulty, terrain, placed\_at (desc), index **texte** (title, description\_html), et combinaisons métier : (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), index **texte** (name, description).
- **user\_challenges** : (user\_id, challenge\_id) (unique), user\_id, challenge\_id, status, (user\_id, status, ↪ updated\_at).
- **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).

L'index **2dsphere** est l'un des éléments centraux du modèle physique de la base. Chaque cache est représentée par un champ `loc` de type GeoJSON (`{ type: "Point", coordinates: [lon, lat] }`). Cet index permet à MongoDB d'exécuter de manière optimale toutes les requêtes géospatiales utilisées par l'application :

- sélection des caches dans un rayon donné (`$geoWithin + $centerSphere`),
- recherche des caches les plus proches d'un point (`$nearSphere`),
- filtrage combiné (géolocalisation + critères métiers D/T, attributs, type).

Grâce à l'index 2dsphere, ces requêtes s'exécutent en **temps logarithmique**, indépendamment de la taille totale de la collection, et restent performantes même après l'import massif de dizaines ou centaines de milliers de caches depuis des GPX. Il s'agit donc d'un **pré-requis technique essentiel** pour la fluidité de l'outil, qui doit être capable d'afficher en temps réel les caches pertinentes sur carte et de répondre aux tâches complexes des challenges sans ralentir l'expérience utilisateur.

#### 4.5.5 Exécution (local, sans Docker)

Les scripts peuvent être exécutés aussi bien en local qu'au travers de Docker Compose, selon le contexte de développement.

Depuis la racine du backend :

```
# Activez votre venv au besoin puis     :
python -m app.db.seed_data                # ping + ensure_indexes + seed référentiels + admin
python -m app.db.seed_data --force         # idem, mais vide les collections de référentiels avant
↳ réinsertion
```

**Attention :** -- force réinitialise les collections de référentiels (pas les données utilisateur).

#### 4.5.6 Exécution (via Docker Compose)

```
# 1) Bâtir et lancer les conteneurs
docker compose up -d --build

# 2) Exécuter le seeding dans le conteneur backend
docker compose exec <backend> python -m app.db.seed data --force
```

*Les secrets (URI Mongo, admin) doivent être injectés au conteneur via l'environnement (compose, variables Railway, etc.).*

#### 4.5.7 Composants clés (extraits commentés)

Les extraits suivants illustrent les mécanismes mis en place pour la gestion des index et le seeding des données.

#### 4.5.7.1 Assurer les index (idempotent)

```
# app/db/seed_indexes.py - extrait simplifié
from pymongo import ASCENDING, DESCENDING, TEXT
from pymongo.operations import IndexModel
from pymongo.collation import Collation
from app.db.mongoddb import get_collection

COLLATION_CI = Collation(locale="en", strength=2) # insensible à la casse

def ensure_index(coll_name, keys, *, name=None, unique=None, partial=None, collation=None):
    coll = get_collection(coll_name)
    # ... comparaison index existant / options ...
    opts = {}
    if name: opts['name'] = name
    if unique is not None: opts['unique'] = unique
    if partial: opts['partialFilterExpression'] = partial
    if collation is not None: opts['collation'] = collation
    coll.create_indexes([IndexModel(keys, **opts)])

# Exemple : unicité insensible à la casse
ensure_index('users', [('username', ASCENDING)], name='uniq_username_ci', unique=True, collation=
    COLLATION_CI)
```

#### 4.5.7.2 Seeding des référentiels & admin

```
# app/db/seed_data.py – extrait simplifié
from app.db.mongodb import db as mg_db, get_collection
from app.db.seed_indexes import ensure_indexes
from app.core import security

def seed_collection(file_path, collection_name, force=False):
    count = mg_db[collection_name].count_documents({})
    if count > 0 and not force:
        return
    if force:
        mg_db[collection_name].delete_many({})
    mg_db[collection_name].insert_many(json.load(open(file_path)))

def seed_admin_user():
    coll = get_collection("users")
    pwd_hash = security.pwd_context.hash(os.getenv("ADMIN_PASSWORD"))
    coll.update_one({"username": os.getenv("ADMIN_USERNAME")},
                    {"$set": {..., "password_hash": pwd_hash, "role": "admin"},
                     "$setOnInsert": {"created_at": now()}}, upsert=True)

if __name__ == "__main__":
    test_connection() # ping Mongo
    ensure_indexes() # création/MAJ idempotente des index
    seed_referentials(force="--force" in sys.argv)
```

#### 4.5.8 Bonnes pratiques & sécurité

Quelques bonnes pratiques doivent être respectées pour maintenir la cohérence des données et la sécurité de l'application.

- **Idempotence** : relancer le script ne casse pas les index existants si la configuration n'a pas changé.
- **Collation** : utilisez des collations cohérentes pour les unicités *case-insensitive* (ex. utilisateurs).
- **Index géo & texte** : un seul index **texte** par collection ; vérifiez la présence des 2dsphere pour les requêtes cartographiques.
- **Quotas & coûts** : la multiplication d'index a un **coût d'écriture** ; validez les index réellement utiles via vos workloads.
- **Secrets** : stockez l'URI Mongo et le mot de passe admin via variables d'environnement (jamais en clair).

#### 4.5.9 Vérification après exécution

Une fois le seeding effectué, il est important de vérifier la présence effective des index attendus. Checklist rapide (via mongosh) :

```
use <your_db>
// Exemples
db.users.getIndexes()
db.caches.getIndexes()
db.challenges.getIndexes()
```

#### 4.5.10 Tests associés au seeding

Des tests unitaires viennent compléter le dispositif afin de s'assurer de la bonne accessibilité de la base et du respect des contraintes.

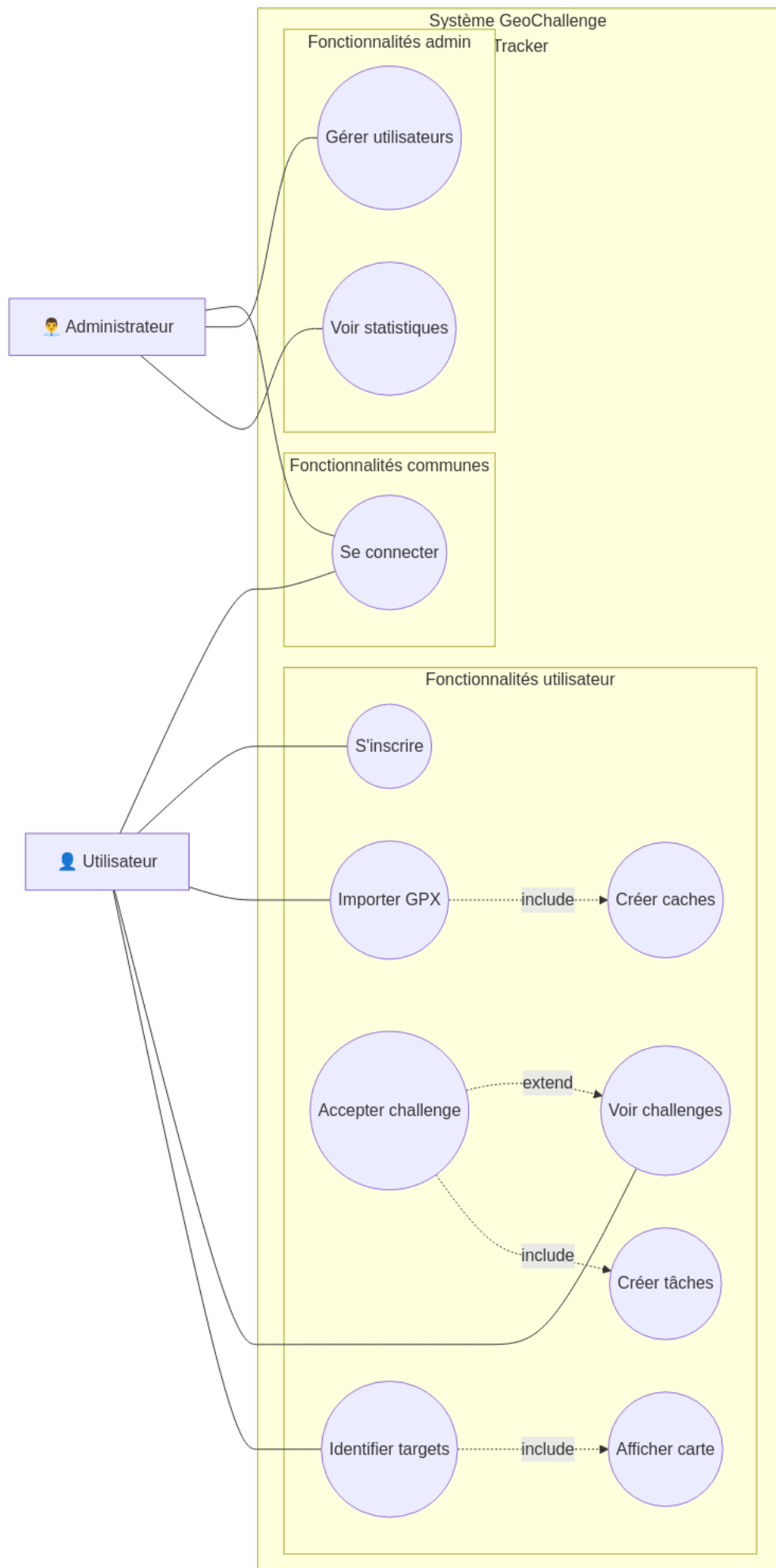
Un test Pytest permet de vérifier que l'environnement backend accède bien à MongoDB :

```
# backend/tests/test_connectivity.py
from app.db.mongodb import client as mg_client

def test_backend_can_access_mongo():
    dbs = mg_client.list_database_names()
    assert isinstance(dbs, list)
```

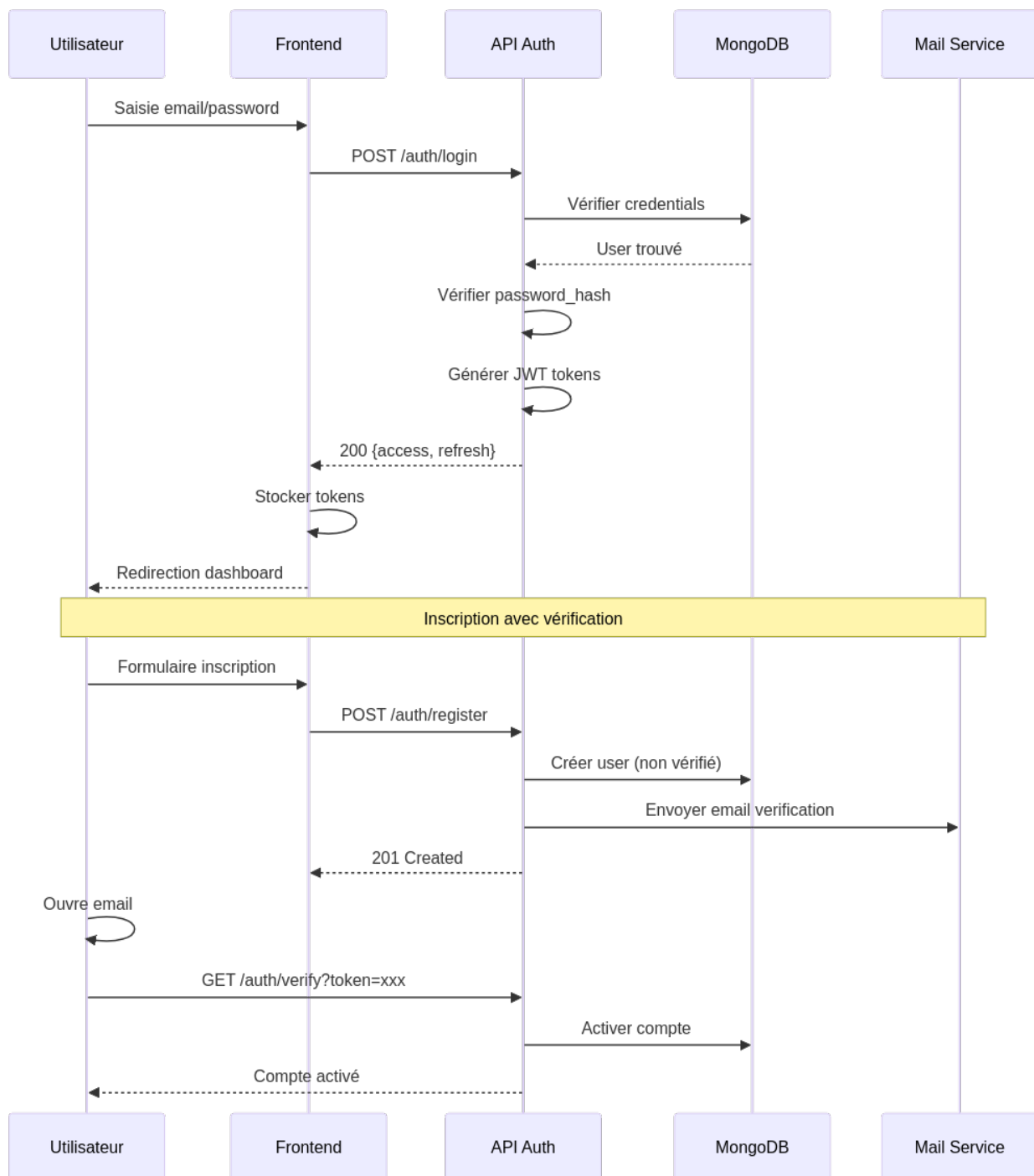
- Vérifie que la connexion fonctionne et qu'une liste de bases est renvoyée.
- Peut être lancé seul pour diagnostiquer un problème de connexion.
- Intégré dans la suite Pytest pour automatiser le contrôle lors du CI/CD.

## 4.6 Diagramme cas d'utilisations

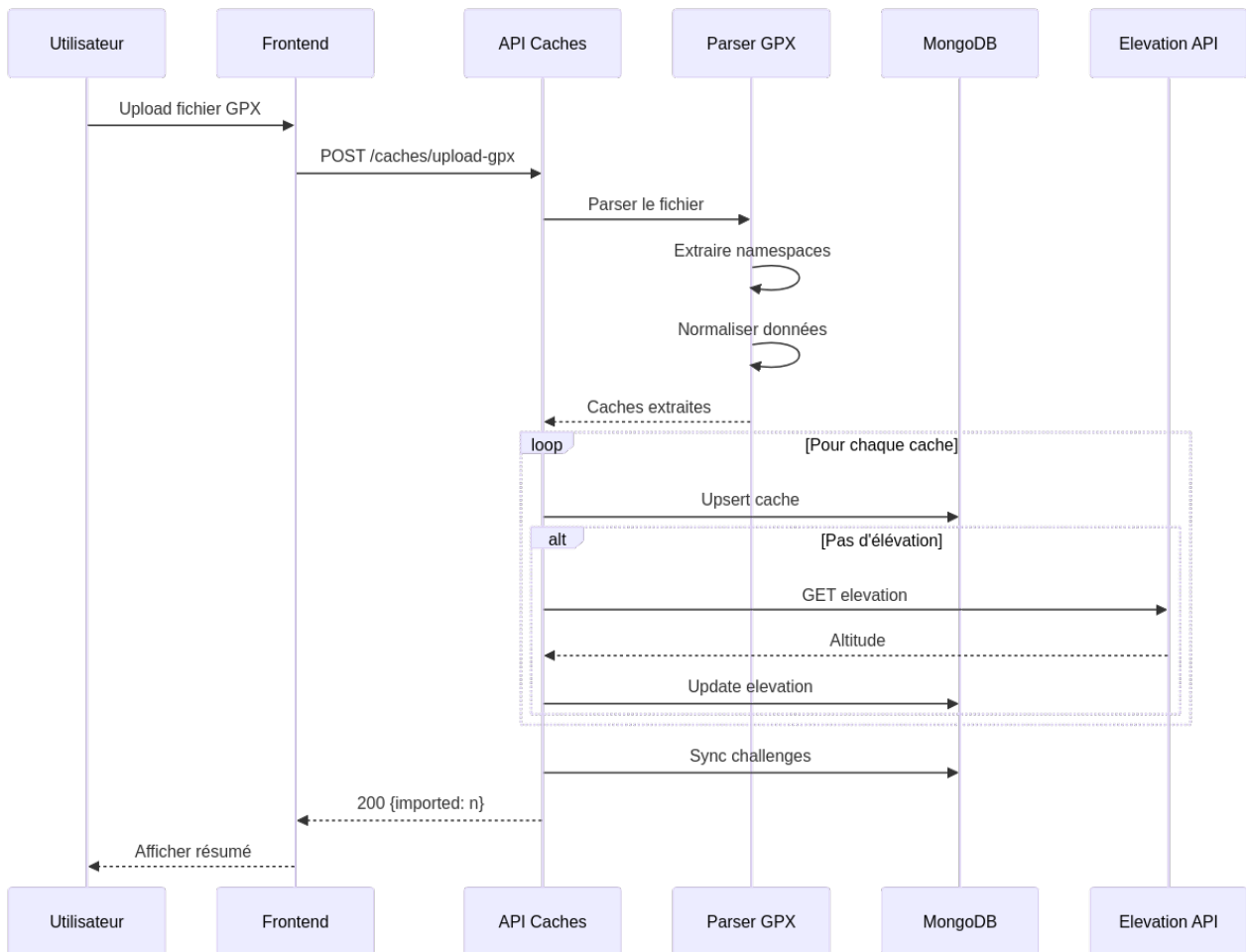


## 4.7 Diagrammes de séquence

### 4.7.1 Authentification

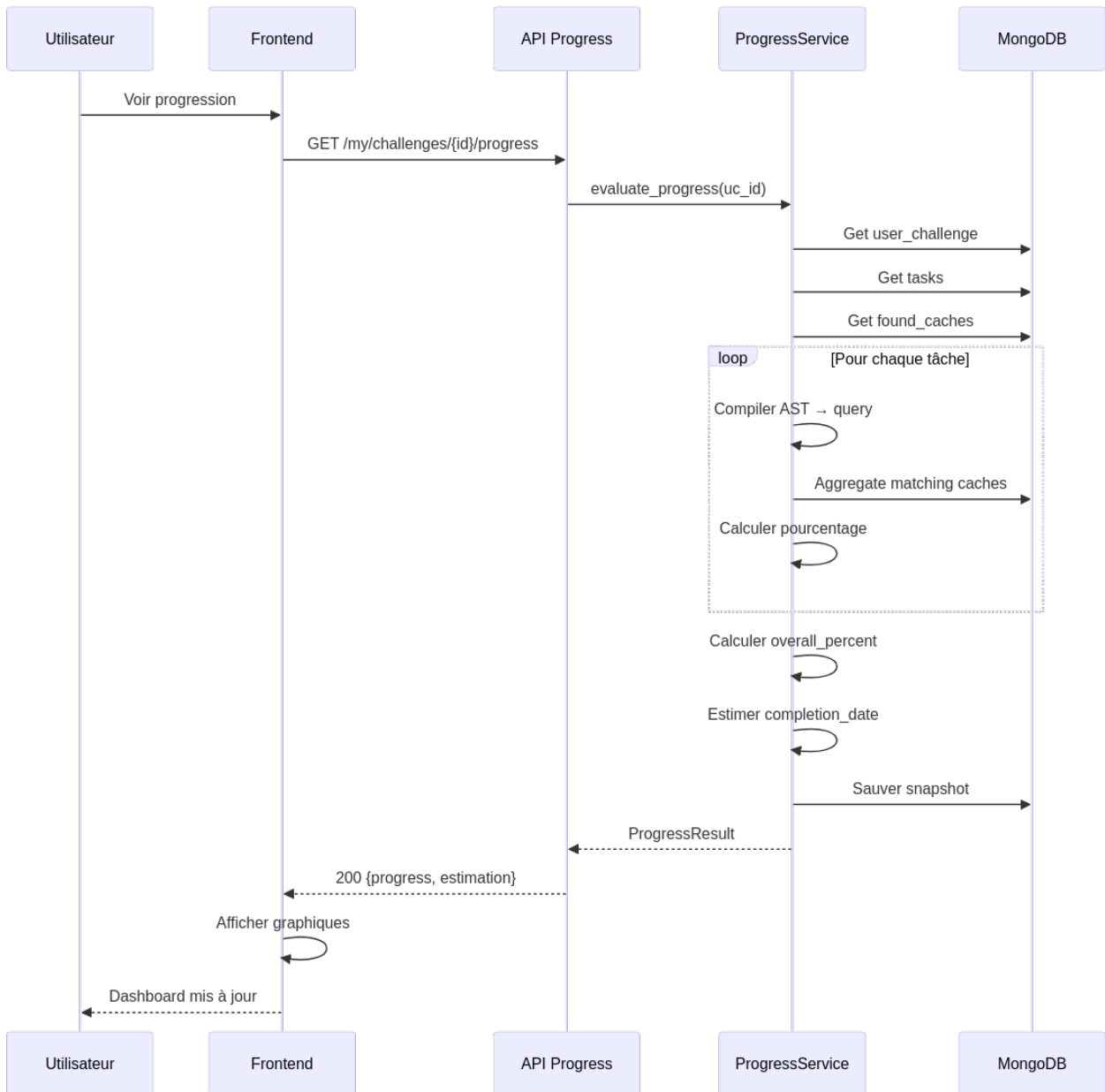


## 4.7.2 Import GPX





### 4.7.3 Calcul de progression



## 5 Spécifications techniques

### 5.1 Backend

Le choix du backend s'est porté sur FastAPI, pour des raisons de rapidité, de sécurité et de simplicité d'intégration.

- **FastAPI (Python)** : choisi pour sa rapidité de développement, sa clarté syntaxique et son support natif d'OpenAPI.
- **Points forts** : asynchrone, validation avec Pydantic, gestion simple des dépendances.
- **Sécurité** : intégration native d'OAuth2, JWT, dépendances paramétrées pour gérer rôles et droits.

```
# requirements.txt
fastapi==0.117.1
pydantic==2.11.9
pymongo==4.15.1
passlib[bcrypt]==1.7.4
lxml==6.0.2
httpx==0.28.1
...
```

### 5.2 Frontend

Le frontend repose sur Vue.js 3, accompagné de bibliothèques spécialisées pour le style et la cartographie.

- **Vue.js 3** : framework progressif, adapté au rendu dynamique de données complexes.
- **Librairies** : Tailwind CSS pour le style rapide, Leaflet pour la cartographie interactive.
- **Optimisation** : clustering des marqueurs, tile caching pour réduire la charge réseau et améliorer la fluidité sur mobile.
- **Approche "green computing via performance optimizing"** : chaque interaction doit minimiser le rendu inutile, afin de préserver batterie et ressources CPU.

```
// package.json
{
  "dependencies": {
    "vue": "^3.5.17",
    "@vue/router": "^4.5.1",
    "pinia": "^3.0.3",
    "axios": "^1.11.0",
    "leaflet": "^1.9.9",
    "leaflet.markercluster": "^1.5.3"
  },
  "devDependencies": {
    "vitest": "^3.2.4",
    "@eslint/js": "^9.36.0",
    "@playwright/test": "^1.55.1",
  }
}
```

### 5.3 Base de données

Pour la persistance des données, le choix de MongoDB s'explique par la variabilité inhérente aux caches géocaching.

- **MongoDB (NoSQL)** : choisi pour sa souplesse documentaire. Les caches étant très hétérogènes (attributs variables, champs facultatifs), le modèle documentaire colle mieux que le relationnel.
- **Index** : 2dsphere pour les recherches géospatiales, uniques pour GC et utilisateurs, combinés pour les filtres métiers (D/T, attributs).
- **Avantage** : structure flexible, évolutive selon l'apparition de nouveaux types de caches ou de challenges.

## 5.4 Infrastructure Docker

```
# docker-compose.yml
services:
  backend:
    build:
      context: ./backend
      dockerfile: Dockerfile
    container_name: geo-backend
    ports:
      - "8000:8000"
    env_file:
      - .env
    volumes:
      - ./backend:/app
      - ./backend/uploads:/app/uploads
    restart: unless-stopped

  frontend:
    build:
      context: ./frontend
      dockerfile: Dockerfile
    container_name: geo-frontend
    env_file:
      - .env
    depends_on:
      - backend
      - tiles
    ports:
      - "${FRONTEND_PORT:-5173}:${FRONTEND_INTERNAL_PORT:-5173}"
    volumes:
      - ./frontend:/app
      - /app/node_modules
    restart: unless-stopped

  tiles:
    image: nginx:1.25-alpine
    container_name: geo-tiles
    restart: unless-stopped
    command: |
      sh -c "... "
    volumes:
      - ./ops/nginx/tiles.conf:/etc/nginx/conf.d/tiles.conf:ro
      - ./ops/nginx/www:/var/www:ro
      - tiles_cache:/var/cache/nginx/tiles_cache
    ports:
      - "8080:80"
    deploy:
      resources:
        limits:
          memory: 512M
        reservations:
          memory: 256M
    healthcheck:
      test: ["CMD-SHELL", "..."]
      interval: 30s
      timeout: 3s
      retries: 3
      start_period: 10s
```

## 5.5 Intégrations externes

L'application s'appuie également sur des services externes, principalement des APIs OpenData.

- **APIs OpenData** : utilisées pour l'altimétrie et la localisation (communes).
- **Choix stratégique** : indépendance contractuelle (pas de dépendance propriétaire), souplesse (changement ou combinaison d'APIs si besoin), résilience (répartition de charge possible).
- **Mitigation des risques** : respect strict des quotas d'appel, stockage en base pour limiter la redondance des requêtes.

## 5.6 Sécurité multicouche

### 5.6.1 Authentification JWT

Les tokens JWT permettent une authentification stateless sans session serveur, facilitant la scalabilité et l'intégration avec les SPA comme Vue.js.

```
# backend/app/core/security.py
def create_access_token(data: dict) -> str:
    to_encode = data.copy()
    expire = datetime.utcnow() + timedelta(minutes=60)
    to_encode.update({"exp": expire})
    return jwt.encode(
        to_encode,
        settings.JWT_SECRET_KEY,
        algorithm="HS256"
    )

def verify_token(token: str) -> dict:
    try:
        payload = jwt.decode(
            token,
            settings.JWT_SECRET_KEY,
            algorithms=["HS256"]
        )
        return payload
    except JWTError:
        raise HTTPException(401, "Invalid token")
```

### 5.6.2 Validation des entrées

La validation stricte des fichiers uploadés protège contre les injections de code malveillant, les vulnérabilités d'exécution et garantit l'intégrité des données.

```
# backend/app/models/cache.py
class CacheBase(BaseModel):
    gc: str = Field(..., regex="^GC[A-Z0-9]+$")
    name: str = Field(..., min_length=1, max_length=255)
    difficulty: float = Field(..., ge=1.0, le=5.0)
    terrain: float = Field(..., ge=1.0, le=5.0)

    @validator('gc')
    def validate_gc_code(cls, v):
        if not v.startswith('GC'):
            raise ValueError('Invalid GC code')
        return v.upper()
```

### 5.6.3 Protection CORS

La configuration CORS empêche les requêtes non autorisées depuis des domaines externes et protège contre les attaques CSRF dans une architecture frontend/backend découplée.

```
# backend/app/main.py
app.add_middleware(
    CORSMiddleware,
    allow_origins=["http://localhost:5173"],
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)
```

### 5.6.4 Rate Limiting

Le rate limiting sur les appels à l'API OpenStreetMap respecte leur politique d'utilisation équitable, prévient les dépassements de quota et garantit la disponibilité du service pour tous les utilisateurs.

```
# ops/nginx/tiles.conf
limit_req_zone $binary_remote_addr zone=tiles:10m rate=20r/s;

location /tiles/ {
    limit_req zone=tiles burst=50 nodelay;
    proxy_cache_valid 200 7d;
    proxy_cache tiles_cache;
}
```

## 5.7 Optimisations performance

### 5.7.1 Cache référentiels

L'ensemble des données statiques utilisent un système de cache.

```
# backend/app/services/referentials_cache.py
class ReferentialsCache:
    def __init__(self):
        self._cache = {}
        self._last_refresh = None

    async def get_cache_types(self):
        if 'cache_types' not in self._cache:
            self._cache['cache_types'] = await db.cache_types.find().to_list()
        return self._cache['cache_types']

    def resolve_type_code(self, code: str) -> ObjectId:
        types = self._cache.get('cache_types', [])
        for t in types:
            if t['code'].lower() == code.lower():
                return t['_id']
        raise ValueError(f"Unknown type: {code}")

referentials = ReferentialsCache()
```

### 5.7.2 Agrégations MongoDB optimisées

Les pipelines d'agrégation MongoDB permettent d'effectuer les calculs et filtres directement côté base de données, réduisant le transfert de données et améliorant significativement les performances des requêtes complexes.

```
# backend/app/services/progress.py
pipeline = [
    {"$match": {"user_id": user_id}},
    {"$lookup": {
        "from": "caches",
        "localField": "cache_id",
        "foreignField": "_id",
    }}
```

```

        "as": "cache"
    }},
    {"$unwind": "$cache"},
    {"$match": compiled_query},
    {"$group": {
        "_id": None,
        "count": {"$sum": 1},
        "sum_difficulty": {"$sum": "$cache.difficulty"}
    }}
]

```

### 5.7.3 Map tiles caching

Le cache des tuiles cartographiques réduit drastiquement les appels réseau vers les serveurs de tuiles, améliore les temps de chargement et diminue la charge sur les services tiers comme OSM ou Mapbox.

```

# User-Agent obligatoire avec contact
map $http_user_agent $osm_ua {
    "" "GCTracker/1.0 (+jean.ceugniet@gmail.com)";
    default "$http_user_agent GCTracker/1.0 (+jean.ceugniet@gmail.com)";
}

# ---- choose upstream host deterministically (a/b/c) ----
split_clients "${remote_addr}${request_uri}" $osm_server {
    33.3% "a.tile.openstreetmap.org";
    33.3% "b.tile.openstreetmap.org";
    *     "c.tile.openstreetmap.org";
}

# Bypass du cache si le client envoie "no-cache" ou "max-age=0"
map $http_cache_control $client_no_cache {
    default      0;
    ~*no-cache   1;
    ~*max-age=0  1;
}

# Rate limiting CLIENT → NOTRE SERVEUR (permissif pour UX)
limit_req_zone $binary_remote_addr zone=client_tiles:10m rate=20r/s;

# Rate limiting NOTRE SERVEUR → OSM (STRICT selon recommandations OSM)
limit_req_zone $upstream_addr zone=osm_upstream:10m rate=2r/s;

# DNS resolver pour la résolution dynamique des noms OSM
resolver 127.0.0.11 valid=300s ipv6=off;
resolver_timeout 5s;

# Cache disque
proxy_cache_path /var/cache/nginx/tiles_cache
    levels=1:2
    keys_zone=tiles_cache:200m
    ...

server {
    listen 80;
    server_name _;

    # Logs détaillés
    access_log /var/log/nginx/tiles_access.log combined;
    error_log /var/log/nginx/tiles_error.log warn;

    # Santé rapide : http://host:8080/tiles/_health.png
    location = /tiles/_health.png {
        root /var/www;
        add_header Cache-Control "no-cache, no-store, must-revalidate";
        ...
    }

    location /tiles/ {

```

```

# Limite côté client (permet navigation fluide)
limit_req zone=client_tiles burst=50 nodelay;

# Limite vers OSM (STRICTE - respecte leurs guidelines)
limit_req zone=osm_upstream burst=5 nodelay;

# Validation format tiles
if ($uri !~ "^/tiles/[0-9]{1,2}/[0-9]+/[0-9]+\\.png$") {
    return 404;
}

# Validation zoom level (0-19 pour OSM)
# retirer le préfixe /tiles/
# PROXY VERS OSM
proxy_pass https://$osm_server;

# TLS/SNI
proxy_ssl_server_name on;
proxy_ssl_name $osm_server;
proxy_ssl_protocols TLSv1.2 TLSv1.3;
proxy_ssl_verify off;

# Headers OSM requis
proxy_set_header Host tile.openstreetmap.org;
proxy_set_header User-Agent $osm_ua;
...

# Utilise HTTP/1.1 côté upstream + pas de Connection: keep-alive explicite
proxy_http_version 1.1;
proxy_set_header Connection "";

# Neutralise la compression (évite variations CDN)

# --- CACHE ---
proxy_cache tiles_cache;
proxy_cache_key $scheme$proxy_host$request_uri;
proxy_cache_bypass $client_no_cache;

# Respect de la fraîcheur amont
proxy_cache_revalidate on;
proxy_pass_header Cache-Control;
...

# Durées de cache selon OSM
proxy_cache_valid 200 304 7d;      # tiles valides: 7 jours
proxy_cache_valid 404 1m;          # tiles manquantes: 1 minute
...

# Eviter les avalanches
...

# Servir du stale si souci amont
proxy_cache_use_stale error timeout updating http_500 http_502 http_503 http_504;
proxy_cache_background_update on;

# Timeouts adaptés
proxy_connect_timeout 10s;
proxy_send_timeout 15s;
proxy_read_timeout 30s;
...

# Headers debug
...
# Cache côté client (respecte les recommandations OSM)
...
# Sécurité
add_header X-Content-Type-Options "nosniff" always;

# pas de cookies

```

```
}  
  
# Bloquer toutes les autres requêtes  
location / {  
    return 404;  
}  
}
```



## 6 Réalisations candidat

### 6.1 Interfaces utilisateur et code

#### 6.1.1 Composant carte interactive

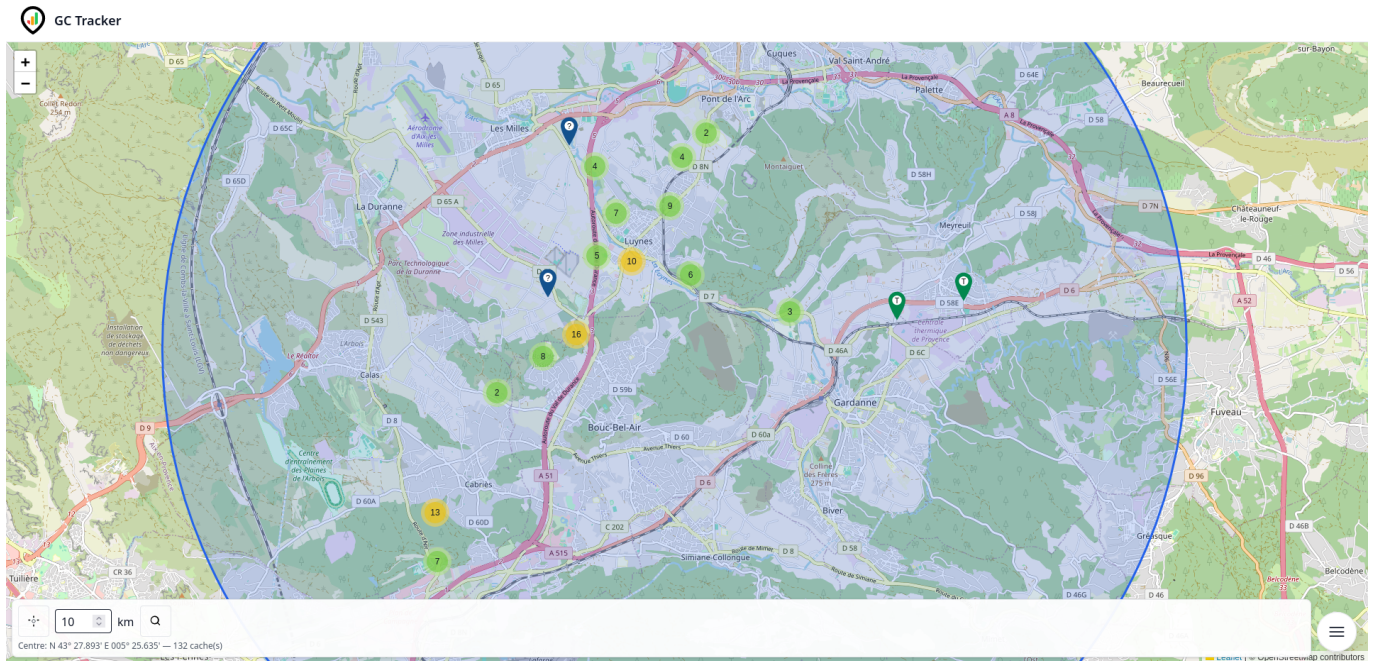


Figure 5 : Carte interactive desktop

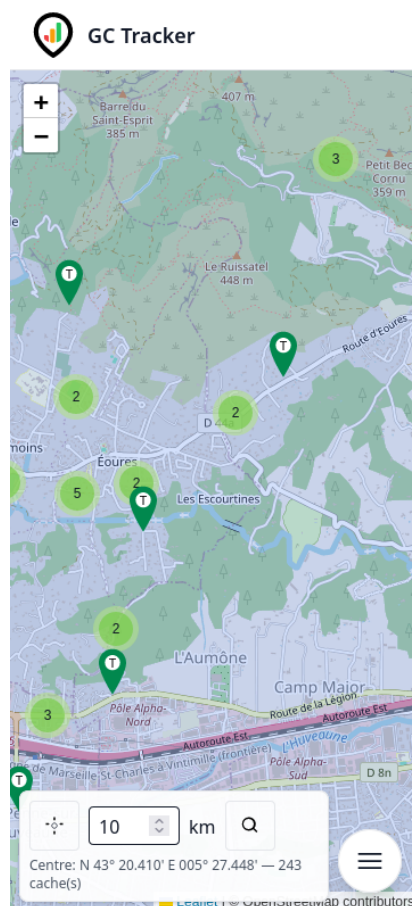


Figure 6 : Carte interactive mobile

```

<!-- frontend/src/components/map/MapBase.vue -->
<template>
  <div ref="mapContainer" class="...">
    <div v-if="showControls" class="...">
      <button @click="enablePick" class="...">Sélectionner</button>
    </div>
    <div v-if="pickMode" class="..." />
  </div>
</template>

<script setup lang="ts">
import { ref, onMounted, watch } from 'vue'
import L from 'leaflet'
import 'leaflet.markercluster'

const props = defineProps<{ center?: [number, number]; zoom?: number; markers?: MarkerData[] }>()
const emit = defineEmits<{ pick: [location: { lat: number; lng: number }]; markerClick: [marker:
  ↳ MarkerData] }>()

const map = ref<L.Map>()
const clusters = ref<L.MarkerClusterGroup>()

onMounted(() => {
  // Initialisation de la carte
  map.value = L.map(mapContainer.value, {center: props.center || ..., zoom: props.zoom || ...})

  // Ajout des tuiles OSM via proxy
  L.tileLayer('/tiles/{z}/{x}/{y}.png', {attribution: '© OpenStreetMap contributors', maxZoom:
    ↳ 19}).addTo(map.value)

  // Initialisation du clustering
  clusters.value = L.markerClusterGroup({chunkedLoading: true, spiderfyOnMaxZoom: true})
  map.value.addLayer(clusters.value)
})

// Ajout des marqueurs avec clustering
watch(() => props.markers, (markers) => {
  if (!clusters.value || !markers) return

  clusters.value.clearLayers()
  markers.forEach(marker => {
    const icon = getIconForType(marker.type_id)
    const leafletMarker = L.marker([marker.lat, marker.lng], { icon })
    leafletMarker.on('click', () => {emit('markerClick', marker)})
    clusters.value.addLayer(leafletMarker)
  })
})

// Mode sélection avec crosshair
const pickMode = ref(false)
const enablePick = () => {
  pickMode.value = true
  map.value?.once('click', (e) => {
    emit('pick', { lat: e.latlng.lat, lng: e.latlng.lng })
    pickMode.value = false
  })
}
</script>

```







**52-MDC41-?-CHALLENGE ...** 

GC: GC99016  
Màj: 04/10/2025  
Pas encore commencé





**50-MDC41-?-CHALLENGE ...** 

GC: GC9900Y  
Màj: 03/10/2025  
Pas encore commencé





**[GBE18] La tribu du géoca...** 

GC: GC7YD29  
Màj: 03/10/2025  
Pas encore commencé





**49-MDC41-?-CHALLENGE ...** 

GC: GC9900T  
Màj: 03/10/2025  
Pas encore commencé





Figure 7 : Liste des challenges

```

<template>
  <div class="...">
    <!-- Filtres (boutons ronds) -->
    <div class="...">
      <button v-for="s in ['all', 'pending', 'accepted', 'dismissed', 'completed']" :key="s"
        class="..."
        :class="filterStatus === s ? 'bg-blue-600 text-white border-blue-600' : 'bg-white text-
        ↪ gray-600 hover:bg-gray-50'"
        @click="setFilter(s as any)" :title="statusLabels[s]" :aria-label="statusLabels[s]">
          <component :is="s === 'all' ? AdjustmentsHorizontalIcon : statusIcons[s as keyof typeof
          ↪ statusIcons]"
            class="..." />
        </button>
      </div>

    <!-- Etat / erreurs -->
    <div v-if="error" class="...">{{ error }}</div>
    <div v-if="loading" class="...">...Chargement</div>

    <!-- Liste -->
    <div v-if="!loading" class="...">
      <div v-for="(ch, idx) in challenges" :key="ch.id" class="..."
        :class="idx % 2 === 0 ? 'bg-white' : 'bg-gray-50'">
        <div class="...">
          <div class="...">
            <h3 class="...">
              <CheckCircleIcon v-if="ch.status === 'accepted'" class="..."
                aria-hidden="true" />
              <XCircleIcon ... />
              <ClockIcon ... />
              <TrophyIcon ... />

              <span class="...">{{ ch.challenge.name }}</span>
            </h3>
            <p class="...">GC: {{ ch.cache.GC }}</p>
            <p class="...">
              Màj: {{ new Date(ch.updated_at).toLocaleDateString() }}
            </p>
          </div>
          <component :is="statusIcons[ch.status]" class="..." />
        </div>

        <div class="...">
          <div v-if="ch.progress && ch.progress.percent !== null" class="...">
            <div class="..." :style="{ width: ch.progress.percent + '%' }"></div>
          </div>
          <p v-else class="...">Pas encore commencé</p>
        </div>

        <!-- Actions (icônes ronds) -->
        <div class="...">
          <button class="..." @click="showDetails(ch)" title="Détails">
            <InformationCircleIcon class="..." />
          </button>

          <button v-if="!['accepted', 'completed'].includes(ch.status) && ch.computed_status !== '
          ↪ completed'"
            class="..." @click="acceptChallenge(ch)" title="Accepter">
              <CheckIcon class="..." />
            </button>

          <button ... class="..." @click="dismissChallenge(ch)" title="Refuser">
            <XMarkIcon class="..." />
          </button>

          <button ... class="..." @click="resetChallenge(ch)" title="Reset">
            <ClockIcon class="..." />
          </button>
        </div>
      </div>
    </div>
  </div>

```

```

        <button ... class="..." @click="manageTasks(ch)" title="Tâches">
          <ClipboardDocumentListItemIcon class="..." />
        </button>
      </div>
    </div>

    <!-- Pagination -->
    <div class="...">
      <button class="..." :disabled="!canPrev" @click="prevPage">
        Précédent
      </button>
      <span class="...">Page {{ page }} / {{ nbPages }}</span>
      <button class="..." :disabled="!canNext" @click="nextPage">
        Suivant
      </button>
    </div>
  </div>
</div>
</template>

<script setup lang="ts">
import { ref, computed, onMounted } from 'vue'
import api from '@api/http'
import { useRouter } from 'vue-router'

const router = useRouter()

// Heroicons 24/outline (cohérent avec ta home)
import {...} from '@heroicons/vue/24/outline'

type Progress = {
  percent: number | null
  tasks_done: number | null
  tasks_total: number | null
  checked_at: string | null
}
type UserChallenge = {
  id: string
  status: 'pending' | 'accepted' | 'dismissed' | 'completed'
  computed_status: string | null
  effective_status: 'pending' | 'accepted' | 'dismissed' | 'completed'
  progress: Progress | null
  updated_at: string
  challenge: { id: string; name: string }
  cache: { id: string; GC: string }
}

const challenges = ref<UserChallenge[]>([])
const page = ref(1)
const pageSize = ref(20)
...
const statusIcons: Record<UserChallenge['status'], any> = {
  pending: ClockIcon,
  accepted: CheckCircleIcon,
  dismissed: XCircleIcon,
  completed: TrophyIcon,
}
const statusLabels: Record<string, string> = {
  all: 'Tous',
  pending: 'En attente',
  accepted: 'Acceptés',
  dismissed: 'Refusés',
  completed: 'Complétés',
}
const canPrev = computed(() => page.value > 1)
const canNext = computed(() => page.value < nbPages.value && nbPages.value > 0)

async function fetchChallenges() {

```

```

loading.value = true
error.value = null
try {
  const params: Record<string, any> = {
    page: page.value,
    page_size: pageSize.value,
  }
  if (filterStatus.value !== 'all') params.status = filterStatus.value

  const { data } = await api.get('/my/challenges', { params })
  challenges.value = data.items ?? []
  nbItems.value = data.nb_items ?? challenges.value.length
  // Si l'API peut renvoyer nb_pages, on le prend, sinon on calcule.
  nbPages.value = data.nb_pages ?? Math.max(1, Math.ceil((data.nb_items ?? 0) / (data.page_size
    ↪ ?? pageSize.value)))
} catch (e: any) {
  error.value = e?.message ?? 'Erreur de chargement'
} finally {
  loading.value = false
}
}

onMounted(fetchChallenges)

function setFilter(status: 'all' | UserChallenge['status']) {
  ...
  fetchChallenges()
}

function prevPage() {
  ...
  fetchChallenges()
}

function nextPage() {
  ...
  fetchChallenges()
}

// Actions (branche tes endpoints si dispo)
async function showDetails(ch: UserChallenge) {
  router.push({ name: 'userChallengeDetails', params: { id: ch.id } })
}

async function acceptChallenge(ch: UserChallenge) {
  try {
    loading.value = true
    await api.patch(`/my/challenges/${ch.id}`, {
      status: 'accepted',
    })
    await fetchChallenges()
  } catch (e: any) {
    ...
  } finally {
    loading.value = false
  }
}


async function dismissChallenge(ch: UserChallenge) {...}
async function resetChallenge(ch: UserChallenge) {...}
async function manageTasks(ch: UserChallenge) {...}
</script>

```

Pour la page de listings, j'ai privilégié une présentation synthétique des items, avec un filtrage par catégorie, et une pagination, afin d'optimiser l'expérience utilisateur tout en limitant la quantité de données à traiter.




### 6.1.3 Détails d'un challenge

**GC Tracker**

[← Retour](#)

**18 - Rando Challenge - Le géologue régional**

GC: GC9T6ZG  
Créé: 02/10/2025 · Màj: 02/10/2025  
Statut: pending (computed: completed) → effective: completed

**Description**  
**18 - Rando Challenge - Le géologue régional**  


Avoir trouvé 30 earthcaches dans les Bouches du Rhône

Ce géoart , qui est composé de 41 caches challenge, vous fera faire une randonnée d'environ 10kms jusqu'à la calanque de l'Erevine. Les challenges qui composent cette série sont très variés en ce qui concerne les thèmes et les difficultés.

En ce qui concerne les boîtes, vous cherchez quasiment toujours le même style de contenant : des tube PET , à part pour certaines exceptions.

Il s'agit d'une cache challenge, la boîte se trouve aux coordonnées indiquées dans le waypoint. Pas d'énigmes à résoudre mais des conditions à remplir.

Pour valider cette cache et la loguer found-it, vous devez avoir logué au moins 30 earthcaches dans le département des Bouches du Rhône. Voilà la liste des earthcaches des Bouches du Rhône : <https://coord.info/BMACPRF>

Pour savoir si vous êtes éligibles à un found-it, veuillez utiliser le checker ci-dessous.

81

34

370

PROJECT-GC

CHALLENGE CHECKER

ETAPE 01

Sur le terrain, signer physiquement le logbook.

ETAPE 02

Sur cette page, utiliser le checker pour savoir si vous êtes éligible.

ETAPE 03

Si NON à l'étape 1 OU 2, vous pouvez "écrire une note" pour indiquer quelle condition est remplie.

ETAPE 04

Si OUI à l'étape 1 ET 2, vous pouvez loguer "trouvée".

Notes

Ajouter une note...

Raison d'override (optionnel)

Ex: contrôle manuel...

Enregistrer

Figure 8 : Détails d'un challenge

```

<template>
  <div class="...">
    <button class="..." @click="router.back()">
      <ArrowLeftIcon class="..." /> Retour
    </button>

    <div v-if="loading" class="...">...Chargement</div>
    <div v-if="error" class="...">{{ error }}</div>

    <div v-if="uc" class="...">
      <!-- En-tête -->
      <div class="...">
        <div class="...">
          <div class="...">
            <h1 class="...">{{ uc.challenge.name }}</h1>
            <p class="...">GC: {{ uc.cache.GC }}</p>
            <p class="...">
              Créé: {{ new Date(uc.created_at).toLocaleDateString() }} · Màj:
              {{ new Date(uc.updated_at).toLocaleDateString() }}
            </p>
            <p class="...">
              Statut: <span class="...">{{ uc.status }}</span>
              <span v-if="uc.computed_status" class="..."> (computed: {{ uc.
                ↪ computed_status
                  }}</span>
              <span class="..."> → effective: {{ uc.effective_status }}</span>
            </p>
          </div>
          <InformationCircleIcon class="..." />
        </div>

        <!-- Progress -->
        <div class="...">
          <div v-if="uc.progress && uc.progress.percent !== null" class="...">
            <div class="..." :style="{ width: uc.progress.percent + '%' }"></div>
          </div>
          <p v-else class="...">Pas encore commencé</p>
        </div>

        <!-- Actions statut -->
        <div class="...">
          <button v-if="canAccept"
            class="..."
            :disabled="loading" @click="patchStatus('accepted')" title="Acceptor">
            <CheckIcon class="..." />
          </button>
          <button v-if="canDismiss"
            class="..." :disabled="loading"
            @click="patchStatus('dismissed')" title="Refuser">
            <XMarkIcon class="..." />
          </button>
        </div>
      </div>
    </div>

    <!-- Description du challenge -->
    <div class="...">
      <h2 class="...">Description</h2>
      <!-- Rendu HTML sanitisé -->
      <div class="..." v-html="safeDescription"></div>
    </div>

    <!-- Notes & override -->
    <div class="...">
      <h2 class="...">Notes</h2>
      <textarea v-model="notes" rows="3" class="..."
        placeholder="Ajouter une ...note" />
      <div class="...">
        <label class="...">Raison d'override (optionnel)</label>
      </div>
    </div>
  </div>

```



```

        <input v-model="overrideReason" type="text" class="..."
            placeholder="Ex: contrôle ...manuel" />
    </div>
    <div class="...">
        <button class="..."
            :disabled="loading" @click="saveNotes">
            Enregistrer
        </button>
    </div>
</div>
</div>
</div>
</template>

<script setup lang="ts">
import { onMounted, ref, computed } from 'vue'
import { useRoute, useRouter } from 'vue-router'
import api from '@api/http'
import DOMPurify from 'dompurify'
import {...} from '@heroicons/vue/24/outline'

type Progress = {
    percent: number | null
    tasks_done: number | null
    tasks_total: number | null
    checked_at: string | null
}

type Detail = {
    id: string
    status: 'pending' | 'accepted' | 'dismissed' | 'completed'
    computed_status: 'pending' | 'accepted' | 'dismissed' | 'completed' | null
    effective_status: 'pending' | 'accepted' | 'dismissed' | 'completed'
    progress: Progress | null
    updated_at: string
    created_at: string
    manual_override: boolean
    override_reason: string | null
    notes: string | null
    challenge: { id: string; name: string; description?: string | null }
    cache: { id: string; GC: string }
}

const route = useRoute()
const router = useRouter()
const id = route.params.id as string

const loading = ref(false)
const error = ref<string | null>(null)
const uc = ref<Detail | null>(null)

const notes = ref('')
const overrideReason = ref('')

const canAct = computed(() => uc.value && uc.value.computed_status !== 'completed')
const canAccept = computed(() =>
    uc.value && ![ 'accepted', 'completed' ].includes(uc.value.status) && uc.value.computed_status
    ↪ !== 'completed'
)
const canDismiss = computed(() =>
    uc.value && ![ 'dismissed', 'completed' ].includes(uc.value.status) && uc.value.computed_status
    ↪ !== 'completed'
)

const safeDescription = computed(() => {
    const html = uc.value?.challenge.description ?? ''
    try {
        return DOMPurify.sanitize(html)
    } catch {
        return html // fallback (évite crash si DOMPurify absent)
    }
})

```

```

    }
  })

  async function fetchDetail() {
    loading.value = true
    error.value = null
    try {
      const { data } = await api.get(`/my/challenges/${id}`)
      uc.value = data
      notes.value = data.notes ?? ''
      overrideReason.value = data.override_reason ?? ''
    } catch (e: any) {
      error.value = e?.message ?? 'Erreur de chargement'
    } finally {
      loading.value = false
    }
  }

  async function patchStatus(status: Detail['status']) {...}
  async function saveNotes() {...}

  onMounted(fetchDetail)
</script>

```

Pour la page de détails, j'ai choisi d'afficher une information beaucoup plus complète, afin de permettre à l'utilisateur d'effectuer un choix éclairé sur les challenges auxquels il souhaite participer. Cette page est probablement amenée à évoluer dans le futur afin d'améliorer le maillage avec les pages concernant la progression.

## 6.2 Composants métier

### 6.2.1 Parser GPX multi-namespace

Les fichiers GPX générés par les outils de géocaching utilisant plusieurs namespaces, il était nécessaire de prévoir un parser capable de supporter ceux-ci.

```

# backend/app/services/parsers/GPXCachParser.py
from lxml import etree
from typing import List, Dict, Any
import re

class GPXCachParser:
    """Parser pour fichiers GPX geocaching multi-namespace"""

    NAMESPACES = {
        'gpx': 'http://www.topografix.com/GPX/1/0',
        'groundspeak': 'http://www.groundspeak.com/cache/1/0/1',
        'cgeo': 'http://www.cgeo.org/wptext/1/0',
        'gsak': 'http://www.gsak.net/xmlv1/6'
    }

    def parse_file(self, file_content: bytes) -> List[Dict[str, Any]]:
        """Parse un fichier GPX et extrait les caches"""
        try:
            tree = etree.fromstring(file_content)
            waypoints = tree.xpath('//gpx:wpt', namespaces=self.NAMESPACES)

            caches = []
            for wpt in waypoints:
                cache = self._extract_waypoint(wpt)
                if cache:
                    caches.append(cache)

            return caches

        except etree.XMLSyntaxError as e:
            raise ValueError(f"Invalid GPX file: {e}")

```

```

def _extract_waypoint(self, wpt) -> Dict[str, Any]:
    """Extrait les données d'un waypoint"""
    # Coordonnées
    lat = float(wpt.get('lat'))
    lon = float(wpt.get('lon'))

    # Code GC et nom
    gc = self._xpath_text(wpt, 'gpx:name')
    if not gc or not gc.startswith('GC'):
        return None

    name = self._xpath_text(wpt, 'groundspeak:cache/groundspeak:name')

    # Difficulté et terrain
    difficulty = float(self._xpath_text(wpt, 'groundspeak:cache/groundspeak:difficulty', '1.0')
    ↪ )
    terrain = float(self._xpath_text(wpt, 'groundspeak:cache/groundspeak:terrain', '1.0'))
    # Type et taille
    cache_type = self._xpath_text(wpt, 'groundspeak:cache/groundspeak:type')
    cache_size = self._xpath_text(wpt, 'groundspeak:cache/groundspeak:container')
    # Attributs
    attributes = []
    attr_nodes = wpt.xpath('.//groundspeak:attribute', namespaces=self.NAMESPACES)
    for attr in attr_nodes:
        attributes.append({'id': int(attr.get('id')), 'inc': attr.get('inc') == '1'})

    # Description HTML
    short_desc = self._xpath_text(wpt, 'groundspeak:cache/groundspeak:short_description')
    long_desc = self._xpath_text(wpt, 'groundspeak:cache/groundspeak:long_description')

    return {
        'gc': gc,
        'name': name or 'Unknown',
        'lat': lat,
        'lon': lon,
        ...
    }

def _xpath_text(self, node, xpath: str, default: str = None) -> str:
    """Helper pour extraire du texte via XPath"""
    result = node.xpath(xpath, namespaces=self.NAMESPACES)
    if result and len(result) > 0:
        if hasattr(result[0], 'text'):
            return result[0].text
        return str(result[0])
    return default

```

## 6.2.2 Moteur de règles AST

La nécessité de définition des tâches selon un modèle flexible et évolutif a poussé au choix de l'utilisation des grammaires de type AST.

```

# backend/app/services/query_builder.py
from typing import Dict, Any, List
from app.models.challenge_ast import TaskExpression

class QueryBuilder:
    """Compile les expressions AST en requêtes MongoDB"""

    def compile_expression(
        self,
        expr: TaskExpression,
        found_only: bool = False
    ) -> Dict[str, Any]:
        """Compile une expression AST en query MongoDB"""

        # Gestion du noeud racine
        if expr.kind == "and":

```

```

        return self._compile_and(expr, found_only)
    elif expr.kind == "or":
        return self._compile_or(expr, found_only)
    elif expr.kind == "not":
        return self._compile_not(expr, found_only)
    else:
        # Feuille de l'arbre
        return self._compile_leaf(expr, found_only)

def _compile_and(self, expr, found_only) -> Dict:
    """Compile un noeud AND"""
    conditions = []

    for child in expr.children:
        compiled = self.compile_expression(child, found_only)
        if compiled:
            conditions.append(compiled)

    if len(conditions) == 1:
        return conditions[0]
    elif conditions:
        return {"$and": conditions}
    return {}

def _compile_or(self, expr, found_only) -> Dict:
    """Compile un noeud OR"""
    ...

    if conditions:
        return {"$or": conditions}
    return {}

def _compile_leaf(self, expr, found_only) -> Dict:
    """Compile une feuille (condition simple)"""

    # Type de cache
    if expr.kind == "type_in":
        type_ids = self._resolve_type_ids(expr.type_ids)
        return {"cache.type_id": {"$in": type_ids}}

    # Taille de cache : ...
    # Pays : ...
    # État/région : ...
    # Année de pose
    elif expr.kind == "placed_year":
        year = expr.year
        return {
            "cache.placed_at": {
                "$gte": datetime(year, 1, 1),
                "$lt": datetime(year + 1, 1, 1)
            }
        }

    # Difficulté entre
    elif expr.kind == "difficulty_between":
        return {
            "cache.difficulty": {
                "$gte": expr.min,
                "$lte": expr.max
            }
        }

    # Terrain entre : ...
    # Attributs
    elif expr.kind == "attributes":
        conditions = []
        for attr in expr.attributes:
            attr_id = self._resolve_attribute_id(attr.attribute_id)
            conditions.append({

```

```

        "cache.attributes": {
            "$elemMatch": {
                "attribute_doc_id": attr_id,
                "is_positive": attr.is_positive
            }
        }
    })
    return {"$and": conditions} if conditions else {}

# Agrégats (somme difficulté, terrain, etc.)
elif expr.kind.startswith("aggregate_sum"):
    # Les agrégats sont traités différemment
    # dans le pipeline d'agrégation
    return {}

return {}

```

## 6.3 Accès aux données

### 6.3.1 Récupération de caches

```

coll = get_collection("caches")
geo = {"type": "Point", "coordinates": [lon, lat]}
q: dict[str, Any] = {}
if type_id:
    q["type_id"] = _oid(type_id)
if size_id:
    q["size_id"] = _oid(size_id)

pipeline = [
    {
        "$geoNear": {
            "near": geo,
            "distanceField": "dist_meters",
            "spherical": True,
            "maxDistance": radius_km * 1000.0,
            "query": q,
        }
    },
    {"$sort": {"dist_meters": 1}},
    {"$skip": max(0, (page - 1) * min(page_size, 200))},
    {"$limit": min(page_size, 200)},
]
if compact:
    pipeline += _compact_lookups_and_project()

try:
    cur = coll.aggregate(pipeline)
except Exception as e:
    raise HTTPException(
        status_code=400, detail=f"2dsphere index required on caches.loc: {e}"
    ) from e
docs = [_doc(d) for d in cur]

```

Concernant les caches, l'utilisation d'un index 2d sphere s'est naturellement imposé, en raison des temps de recherche logarithmiques. Étant donné le nombre potentiel de caches en base, la pagination des résultats était également une nécessité. Pour des raisons de qualité de l'expérience utilisateur, les champs présentés lors d'une récupération en volume sont limités.

### 6.3.2 Calcul d'un snapshot de progression

```
def evaluate_progress(user_id: ObjectId, uc_id: ObjectId, force=False) -> dict[str, Any]:
    """Évaluer les tâches d'un UC et insérer un snapshot.

    Description:
    - Vérifie l'appartenance de l'UC (`_ensure_uc_owned`).\n
    - Si `force=False` et que l'UC est déjà `completed`, retourne le dernier snapshot (si
    ↪ existant).\n
    - Pour chaque tâche, compile l'expression (`compile_and_only`), compte les trouvailles,
    ↪ met à jour
      éventuellement le statut de la tâche, calcule les agrégats et le pourcentage.\n
    - Calcule l'agrégat global et crée un document `progress`. Si toutes les tâches supportées
    ↪ sont `done`,
      met à jour `user_challenges` en `completed` (statuts déclaré & calculé).

    Args:
    user_id (ObjectId): Utilisateur.
    uc_id (ObjectId): UserChallenge.
    force (bool): Forcer le recalcul même si UC complété.

    Returns:
    dict: Document snapshot inséré (avec `id` ajouté pour la réponse).
    """
    _ensure_uc_owned(user_id, uc_id)
    tasks = _get_tasks_for_uc(uc_id)
    snapshots: list[dict[str, Any]] = []
    sum_current = 0
    sum_min = 0
    tasks_supported = 0
    tasks_done = 0
    uc_statuses = get_collection("user_challenges").find_one(
        {"_id": uc_id}, {"status": 1, "computed_status": 1}
    )
    uc_status = (uc_statuses or {}).get("status")
    uc_computed_status = (uc_statuses or {}).get("computed_status")
    if (not force) and (uc_computed_status == "completed" or uc_status == "completed"):
        # Renvoyer le dernier snapshot existant, sans recalcul ni insertion
        last = get_collection("progress").find_one(
            {"user_challenge_id": uc_id}, sort=[("checked_at", -1), ("created_at", -1)]
        )
        if last:
            return last # même shape que vos snapshots persistés
        # S'il n'y a pas encore de snapshot, on retombe sur le calcul normal

    for t in tasks:
        min_count = int((t.get("constraints") or {}).get("min_count") or 0)
        title = t.get("title") or "Task"
        order = int(t.get("order") or 0)
        status = (t.get("status") or "todo").lower()
        expr = t.get("expression") or {}

        if status == "done" and not force:
            snap = {
                "task_id": t["_id"],
                "order": order,
                "title": title,
                "...":
            }
        else:
            sig, match_caches, supported, notes, agg_spec = compile_and_only(expr)
            if not supported:
                snap = {
                    "task_id": t["_id"],
                    "order": order,
                    "title": title,
                    "...":
                }
            else:
```

```

tic = utcnow()
current = _count_found_caches_matching(user_id, match_caches)
ms = int((utcnow() - tic).total_seconds() * 1000)

# base percent on min_count
bounded = min(current, min_count) if min_count > 0 else current
count_percent = (100.0 * (bounded / min_count)) if min_count > 0 else 100.0
new_status = "done" if current >= min_count else status
task_id = t["_id"]
t["status"] = new_status
if status != "done":
    get_collection("user_challenge_tasks").update_one(
        {"_id": task_id},
        {
            "$set": {
                "status": new_status,
                "last_evaluated_at": utcnow(),
                "updated_at": utcnow(),
            }
        },
    )

# aggregate handling
aggregate_total = None
aggregate_target = None
aggregate_percent = None
aggregate_unit = None
if agg_spec:
    aggregate_total = _aggregate_total(user_id, match_caches, agg_spec)
    aggregate_target = int(agg_spec.get("min_total", 0)) or None
    if aggregate_target and aggregate_target > 0:
        aggregate_percent = max(
            0.0,
            min(
                100.0,
                100.0 * (float(aggregate_total) / float(aggregate_target)),
            ),
        )
    else:
        aggregate_percent = None
    # unit: altitude -> meters, otherwise points
    aggregate_unit = "meters" if agg_spec.get("kind") == "altitude" else "points"

# final percent rule (MVP):
# - if both count & aggregate constraints exist -> percent = min(count_percent,
    ↪ aggregate_percent)
# - if only count -> count_percent
# - if only aggregate -> aggregate_percent or 0 if None
if agg_spec and min_count > 0:
    final_percent = min(count_percent, (aggregate_percent or 0.0))
elif agg_spec and min_count == 0:
    final_percent = aggregate_percent or 0.0
else:
    final_percent = count_percent

# --- dates de progression persistées sur la task ---
task_id = t["_id"]
min_count = int((t.get("constraints") or {}).get("min_count") or 0)

# 2.1 start_found_at : première trouvaille qui matche
start_dt = _first_found_date(user_id, match_caches)
if start_dt and not t.get("start_found_at"):
    get_collection("user_challenge_tasks").update_one(
        {"_id": task_id},
        {"$set": {"start_found_at": start_dt, "updated_at": utcnow()}},
    )
    t["start_found_at"] = start_dt # en mémoire pour la suite

# 2.2 completed_at : date de la min_count-ième trouvaille

```

```

        completed_dt = None
        if min_count > 0 and current >= min_count:
            completed_dt = _nth_found_date(user_id, match_caches, min_count)

        # persister la date si atteinte, sinon l'annuler si elle existait mais plus valide
        if completed_dt:
            if t.get("completed_at") != completed_dt:
                get_collection("user_challenge_tasks").update_one(
                    {"_id": task_id},
                    {
                        "$set": {
                            "completed_at": completed_dt,
                            "updated_at": utcnow(),
                        }
                    },
                )
            t["completed_at"] = completed_dt
        else:
            if t.get("completed_at") is not None:
                get_collection("user_challenge_tasks").update_one(
                    {"_id": task_id},
                    {"$set": {"completed_at": None, "updated_at": utcnow()}},
                )
            t["completed_at"] = None

        snap = {
            "task_id": t["_id"],
            "order": order,
            "title": title,
            ...
            # per-task aggregate block for DT0:
            "aggregate": (
                None
                if not agg_spec
                else {
                    "total": aggregate_total,
                    "target": aggregate_target or 0,
                    "unit": aggregate_unit or "points",
                }
            ),
            "notes": notes,
            "evaluated_in_ms": ms,
            "last_evaluated_at": now(),
            "updated_at": t.get("updated_at"),
            "created_at": t.get("created_at"),
        }

        if snap["supported_for_progress"]:
            tasks_supported += 1
            sum_min += max(0, min_count)
            bounded_for_sum = (
                min(snap["current_count"], min_count) if min_count > 0 else snap["current_count"]
            )
            sum_current += bounded_for_sum
            if bounded_for_sum >= min_count and min_count > 0:
                tasks_done += 1

        snapshots.append(snap)

    aggregate_percent = (100.0 * (sum_current / sum_min)) if sum_min > 0 else 0.0
    aggregate_percent = round(aggregate_percent, 1)
    doc = {
        "user_challenge_id": uc_id,
        "checked_at": now(),
        "aggregate": {
            "percent": aggregate_percent,
            "tasks_done": tasks_done,
            "tasks_total": tasks_supported,
            "checked_at": now(),
        }
    }

```



```

    },
    "tasks": snapshots,
    "message": None,
    "created_at": now(),
}
if (uc_computed_status != "completed") and (tasks_done == tasks_supported):
    new_status = "completed"
    get_collection("user_challenges").update_one(
        {"_id": uc_id},
        {
            "$set": {
                "computed_status": new_status,
                "status": new_status,
                "updated_at": utcnow(),
            }
        },
    )
)
get_collection("progress").insert_one(doc)
# enrich for response
doc["id"] = str(doc.get("_id")) if "_id" in doc else None

return doc

```

Les snapshots de progression sont stockés sous la forme d'objets JSON comprenant le niveau d'avancement de chaque tâche et le niveau d'avancement global du challenge. La conservation de l'ensemble de ces informations, même si elle relève clairement d'une dénormalisation de la base, permet d'établir une série temporelle de façon simple et efficace, et d'effectuer des projections basées sur ces informations.

## 6.4 Autres composants

### 6.4.1 Données d'altimétrie

```

async def fetch(points: list[tuple[float, float]]) -> list[int | None]:
    """Récupérer les altitudes pour une liste de points (alignées sur l'entrée).

    Description:
    - Si le provider est désactivé (`settings.elevation_enabled=False`) **ou** si la liste
      `points` est vide, retourne une liste de `None` de même taille.
    - Respecte un quota quotidien en nombre d'appels HTTP, basé sur la collection
      `api_quotas` et la variable d'environnement `ELEVATION_DAILY_LIMIT` (défaut 1000).
      Si le quota est atteint, retourne des `None` pour les points restants.
    - Construit une chaîne `locations` puis la découpe via `_split_params_by_url_and_count`
      ↪ en respectant `URL_MAXLEN` et `MAX_POINTS_PER_REQ`.
    - Pour chaque fragment :
      * effectue un `GET` sur `ENDPOINT?locations=...` (timeout configurable par
        `ELEVATION_TIMEOUT_S`, défaut "5.0")
      * parse la réponse JSON et extrait `results[*].elevation`
      * mappe chaque altitude (arrondie à l'entier) au bon index d'origine
      * en cas d'erreur HTTP/JSON, laisse les valeurs correspondantes à `None`
      * incrémente le quota et respecte un rate delay (`RATE_DELAY_S`) entre appels
        (sauf après le dernier)
    - Ne lève jamais d'exception ; toute erreur réseau/parse entraîne des `None` localisés
      ↪ .

    Args:
    points (list[tuple[float, float]]): Liste `(lat, lon)` pour lesquelles obtenir l'altitude.

    Returns:
    list[int | None]: Liste des altitudes en mètres (ou `None` sur échec), alignée sur `
    ↪ points`.
    """
    if not ENABLED or not points:
        return [None] * len(points)

    # Respect daily quota (1000 calls/day), counting requests, not points
    daily_count = _read_quota()

```

```

DAILY_LIMIT = int(os.getenv("ELEVATION_DAILY_LIMIT", "1000"))
if daily_count >= DAILY_LIMIT:
    return [None] * len(points)

# We keep a parallel index list to map back results to original points
# Build one big param string then split smartly
param_all = _build_param(points)
param_chunks = _split_params_by_url_and_count(param_all)

results: list[int | None] = [None] * len(points)
# We need to also split the original points list in the same way to keep indices aligned.
# We'll reconstruct chunk-wise indices by counting commas/pipes.
idx_start = 0
async with httpx.AsyncClient(timeout=float(os.getenv("ELEVATION_TIMEOUT_S", "5.0"))) as client
    ↪ :
    for i, param in enumerate(param_chunks):
        # Determine how many points are in this chunk
        n_pts = 1 if param and "|" not in param else (param.count("|") + 1 if param else 0)

        # Quota guard: stop if next request would exceed
        if daily_count >= DAILY_LIMIT:
            break

        url = f"{ENDPOINT}?locations={param}"
        try:
            resp = await client.get(url)
            if resp.status_code == 200:
                data = resp.json() or {}
                arr = data.get("results") or []
                for j, rec in enumerate(arr[:n_pts]):
                    elev = rec.get("elevation", None)
                    if isinstance(elev, (int, float)):
                        results[idx_start + j] = int(round(elev))
                    else:
                        results[idx_start + j] = None
            else:
                # leave None for this slice
                pass
        except Exception:
            # leave None for this slice
            pass

        # update quota & delay
        daily_count += 1
        _inc_quota(1)
        idx_start += n_pts

    # Rate-limit (skip after the last chunk)
    if i < len(param_chunks) - 1:
        await asyncio.sleep(RATE_DELAY_S)

return results

```

Certains challenges étant liés à l'altimétrie, l'accès à un service de fourniture d'information topographique a été prévu. Afin de minimiser les appels, et d'optimiser les temps de réponse, un ensemble de méthodes de caching et de requêtes par blocs a été mis en place.

## 7 Sécurité de l'application

### 7.1 Authentification et autorisation

#### 7.1.1 JWT avec refresh tokens

- Access token : 60 minutes
- Refresh token : 30 jours
- Rotation automatique des tokens
- Blacklist des tokens révoqués

#### 7.1.2 Gestion des rôles

```
# backend/app/core/security.py
def require_admin(current_user = Depends(get_current_user)):
    if current_user.role != "admin":
        raise HTTPException(403, "Admin access required")
    return current_user
```

### 7.2 Protection des données

#### 7.2.1 Hashage des mots de passe

```
# Bcrypt avec salt automatique
password_hash = bcrypt.hashpw(
    password.encode('utf-8'),
    bcrypt.gensalt(rounds=12)
)
```

#### 7.2.2 Isolation par utilisateur

```
# Toutes les requêtes filtrent par user_id
{"user_id": current_user.id}
```

### 7.3 Validation et sanitisation

#### 7.3.1 Validation Pydantic stricte

```
class CacheCreate(BaseModel):
    gc: str = Field(..., regex="^GC[A-Z0-9]{1,10}$")
    lat: float = Field(..., ge=-90, le=90)
    lon: float = Field(..., ge=-180, le=180)

    @validator('*', pre=True)
    def sanitize_strings(cls, v):
        if isinstance(v, str):
            return v.strip()[:1000] # Limite et nettoie
        return v
```

#### 7.3.2 Protection XSS

```
# Sanitisation HTML
from bleach import clean

def sanitize_html(html: str) -> str:
    return clean(
        html,
        tags=['p', 'br', 'strong', 'em', 'a'],
        attributes={'a': ['href']},
        strip=True
    )
```

## 7.4 Protection contre les attaques

### 7.4.1 Rate limiting

- Global : 100 req/min par IP
- Auth : 5 tentatives/min
- Upload : 10 fichiers/heure

### 7.4.2 CSRF Protection

- SameSite cookies
- Origin validation
- Double submit tokens

### 7.4.3 SQL/NoSQL Injection

- Requêtes paramétrées uniquement
- Pas de construction dynamique
- Validation des ObjectId

## 7.5 Audit et monitoring

### 7.5.1 Logs de sécurité

```
# Événements trackés
- Tentatives de connexion échouées
- Changements de permissions
- Accès aux données sensibles
- Uploads de fichiers
```

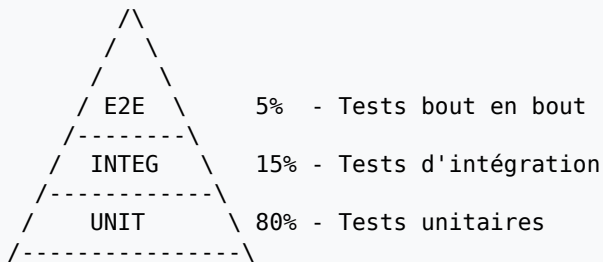
### 7.5.2 Métriques

- Taux d'erreurs 401/403
- Temps de réponse par endpoint
- Volume de données par utilisateur

## 8 Plan de tests

### 8.1 Stratégie de tests

#### Pyramide de tests



### 8.2 Tests unitaires backend

```
# backend/tests/test_gpx_parser.py
import pytest
from app.services.parsers.GPXCacheParser import GPXCacheParser

class TestGPXParser:

    @pytest.fixture
    def parser(self):
        return GPXCacheParser()

    @pytest.fixture
    def sample_gpx(self):
        return '''<?xml version="1.0" encoding="UTF-8"?>
<gpx version="1.0">
  <wpt lat="45.123" lon="2.456">
    <name>GC12345</name>
    <groundspoke:cache>
      <groundspoke:name>Test Cache</groundspoke:name>
      <groundspoke:difficulty>2.5</groundspoke:difficulty>
      <groundspoke:terrain>3.0</groundspoke:terrain>
      <groundspoke:type>Traditional Cache</groundspoke:type>
      <groundspoke:container>Small</groundspoke:container>
    </groundspoke:cache>
  </wpt>
</gpx>'''

    def test_parse_valid_gpx(self, parser, sample_gpx):
        """Test parsing d'un GPX valide"""
        caches = parser.parse_file(sample_gpx.encode())

        assert len(caches) == 1
        cache = caches[0]

        assert cache['gc'] == 'GC12345'
        assert cache['name'] == 'Test Cache'
        assert cache['lat'] == 45.123
        assert cache['lon'] == 2.456
        assert cache['difficulty'] == 2.5
        assert cache['terrain'] == 3.0

    def test_parse_invalid_xml(self, parser):
        """Test avec XML invalide"""
        invalid_xml = b'<gpx>not closed'

        with pytest.raises(ValueError, match="Invalid GPX"):
            parser.parse_file(invalid_xml)

    def test_skip_non_gc_waypoints(self, parser):
```

```

"""Test que les waypoints non-GC sont ignorés"""
gpx = '''<gpx version="1.0">
  <wpt lat="45.123" lon="2.456">
    <name>PARKING</name>
  </wpt>
  <wpt lat="45.124" lon="2.457">
    <name>GC67890</name>
  </wpt>
</gpx>'''

caches = parser.parse_file(gpx.encode())
assert len(caches) == 1
assert caches[0]['gc'] == 'GC67890'

```

### 8.3 Tests d'intégration API

```

# backend/tests/test_api_challenges.py
import pytest
from httpx import AsyncClient
from app.main import app

@pytest.mark.asyncio
class TestChallengesAPI:

    @pytest.fixture
    async def client(self):
        async with AsyncClient(app=app, base_url="http://test") as c:
            yield c

    @pytest.fixture
    async def auth_headers(self, client):
        """Obtient les headers d'authentification"""
        response = await client.post("/auth/login", data={
            "username": "testuser",
            "password": "testpass123"
        })
        token = response.json()['access_token']
        return {"Authorization": f"Bearer {token}"}

    async def test_sync_challenges(self, client, auth_headers):
        """Test synchronisation des challenges"""
        response = await client.post(
            "/my/challenges/sync",
            headers=auth_headers
        )

        assert response.status_code == 200
        data = response.json()
        assert "created" in data
        assert "updated" in data
        assert data["created"] >= 0

    async def test_list_challenges_pagination(self, client, auth_headers):
        """Test liste avec pagination"""
        response = await client.get(
            "/my/challenges?page=1&limit=10",
            headers=auth_headers
        )

        assert response.status_code == 200
        data = response.json()
        assert "items" in data
        assert "total" in data
        assert "page" in data
        assert len(data["items"]) <= 10

    async def test_create_task_with_ast(self, client, auth_headers):

```

```

"""Test création de tâche avec expression AST"""
# D'abord créer un challenge
challenge_response = await client.post(
    "/my/challenges/sync",
    headers=auth_headers
)
uc_id = challenge_response.json()["items"][0]["id"]

# Créer des tâches
tasks_payload = {
    "tasks": [
        {
            "title": "Trouver 10 traditionnelles",
            "expression": {
                "kind": "and",
                "children": [
                    {
                        "kind": "type_in",
                        "type_ids": ["traditional"]
                    }
                ]
            },
            "constraints": {
                "min_count": 10
            }
        }
    ]
}

response = await client.put(
    f"/my/challenges/{uc_id}/tasks",
    json=tasks_payload,
    headers=auth_headers
)

assert response.status_code == 200
tasks = response.json()
assert len(tasks) == 1
assert tasks[0]["title"] == "Trouver 10 traditionnelles"

```

## 8.4 Tests frontend

```

// frontend/src/components/map/MapBase.spec.ts
import { describe, it, expect, vi } from 'vitest'
import { mount } from '@vue/test-utils'
import MapBase from './MapBase.vue'
import L from 'leaflet'

// Mock Leaflet
vi.mock('leaflet', () => ({
  default: {
    map: vi.fn(() => ({
      addLayer: vi.fn(),
      on: vi.fn(),
      once: vi.fn(),
      setView: vi.fn()
    })),
    tileLayer: vi.fn(() => ({
      addTo: vi.fn()
    })),
    marker: vi.fn(() => ({
      on: vi.fn()
    })),
    MarkerClusterGroup: vi.fn(() => ({
      addLayer: vi.fn(),
      clearLayers: vi.fn()
    }))
  }
}))

```

```

    }
  })
}

describe('MapBase', () => {
  it('initialise la carte avec les bonnes coordonnées', async () => {
    const wrapper = mount(MapBase, {
      props: {
        center: [48.856, 2.352], // Paris
        zoom: 12
      }
    })

    await wrapper.vm.$nextTick()

    expect(L.map).toHaveBeenCalled()
    expect(L.tileLayer).toHaveBeenCalledWith(
      '/tiles/{z}/{x}/{y}.png',
      expect.objectContaining({
        attribution: expect.stringContaining('OpenStreetMap')
      })
    )
  })

  it('émet un événement pick en mode sélection', async () => {
    const wrapper = mount(MapBase)

    await wrapper.vm.enablePick()
    expect(wrapper.vm.pickMode).toBe(true)

    // Simuler un clic sur la carte
    const mockEvent = {
      latlng: { lat: 45.123, lng: 2.456 }
    }

    // Trigger le callback du clic
    const mapInstance = wrapper.vm.map
    const clickHandler = mapInstance.once.mock.calls[0][1]
    clickHandler(mockEvent)

    expect(wrapper.emitted('pick')).toBeTruthy()
    expect(wrapper.emitted('pick')[0]).toEqual([
      { lat: 45.123, lng: 2.456 }
    ])
  })

  it('ajoute des marqueurs avec clustering', async () => {
    const markers = [
      { id: '1', lat: 45.1, lon: 2.1, type_id: 'traditional' },
      { id: '2', lat: 45.2, lon: 2.2, type_id: 'mystery' }
    ]

    const wrapper = mount(MapBase, {
      props: { markers }
    })

    await wrapper.vm.$nextTick()

    expect(L.marker).toHaveBeenCalledTimes(2)
    expect(wrapper.vm.clusters.addLayer).toHaveBeenCalledTimes(2)
  })
})

```



## 8.5 Tests de charge

```
# backend/tests/test_performance.py
import pytest
import asyncio
from httpx import AsyncClient
import time

@pytest.mark.performance
class TestPerformance:

    async def test_upload_large_gpx(self, client, auth_headers):
        """Test upload d'un gros fichier GPX"""
        # Générer un GPX avec 1000 waypoints
        gpx_content = self._generate_large_gpx(1000)

        start = time.time()
        response = await client.post(
            "/caches/upload-gpx",
            headers=auth_headers,
            files={"file": ("large.gpx", gpx_content, "application/gpx+xml")}
        )
        duration = time.time() - start

        assert response.status_code == 200
        assert duration < 5.0 # Moins de 5 secondes

        data = response.json()
        assert data["imported"] == 1000

    async def test_concurrent_requests(self, client, auth_headers):
        """Test de requêtes concurrentes"""
        async def make_request():
            return await client.get(
                "/caches/within-radius?lat=45&lon=2&radius=10",
                headers=auth_headers
            )

        # 50 requêtes simultanées
        start = time.time()
        tasks = [make_request() for _ in range(50)]
        responses = await asyncio.gather(*tasks)
        duration = time.time() - start

        # Toutes doivent réussir
        assert all(r.status_code == 200 for r in responses)

        # Temps total < 2 secondes
        assert duration < 2.0

        # P95 < 200ms
        response_times = [r.elapsed.total_seconds() for r in responses]
        response_times.sort()
        p95_index = int(len(response_times) * 0.95)
        assert response_times[p95_index] < 0.2
```

## 9 Jeu d'essai

Afin de valider le système d'import, j'ai préparé 5 fichiers GPX de test :

- fichier 1 : 188 caches pour un volume total de 15,6 Mo, contenant uniquement des caches d'un pays pour lequel je n'avais aucune cache enregistrée dans la base, toutes les caches étant situées dans la même région
- fichier 2 : 2 576 caches, toutes pré-existantes en base de données
- fichier 3 : 1 010 caches, toutes inconnues, situées dans une région inexistante en base de données, mais dans un pays connu
- fichier 4 : 200 caches, toutes inconnues, situées dans un même pays, dans une même région, et contenant 21 challenges
- fichier 5 : volume total de 21Mo

Les résultats attendus étaient donc :

- fichier 1 : 188 caches ajoutées, 1 pays créé, 1 région créée
- fichier 2 : 2 576 caches existantes, 0 pays créés, 0 région créée
- fichier 3 : 1 010 caches ajoutées, 0 pays créés, 1 région créée
- fichier 4 : 200 caches ajoutées, 1 pays créé, 1 régions créée, 21 challenges ajoutés
- fichier 5 : erreur 413, fichier trop volumineux

Le résultat attendu était l'insertion complète de ces 188 caches ainsi que la création d'une nouvelle entrée pays et région dans la base de données. L'import s'est déroulé conformément aux attentes : les 188 caches ont été correctement insérées sans duplication (0 cache existant), et le système a bien détecté et créé 1 nouveau pays et 1 nouvelle région. Aucun conflit n'a été rencontré, confirmant ainsi le bon fonctionnement de la logique de déduplication et d'indexation géographique.

Résultats obtenus : **fichier 1**

```
{
  "summary": {
    "nb_gpx_files": 1,
    "nb_inserted_caches": 188,
    "nb_existing_caches": 0,
    "nb_inserted_found_caches": 0,
    "nb_updated_found_caches": 0,
    "nb_new_countries": 1,
    "nb_new_states": 1
  },
  "challenges_stats": {
    "matched": 0,
    "created": 0,
    "skipped_existing": 0
  }
}
```

**fichier 2**

```
{
  "summary": {
    "nb_gpx_files": 1,
    "nb_inserted_caches": 0,
    "nb_existing_caches": 2 576,
    "nb_inserted_found_caches": 0,
    "nb_updated_found_caches": 0,
    "nb_new_countries": 0,
    "nb_new_states": 0
  },
  "challenges_stats": {
    "matched": 0,
    "created": 0,
    "skipped_existing": 0
  }
}
```

### fichier 3

```
{
  "summary": {
    "nb_gpx_files": 1,
    "nb_inserted_caches": 1 010,
    "nb_existing_caches": 0,
    "nb_inserted_found_caches": 0,
    "nb_updated_found_caches": 0,
    "nb_new_countries": 0,
    "nb_new_states": 1
  },
  "challenges_stats": {
    "matched": 0,
    "created": 0,
    "skipped_existing": 0
  }
}
```

### fichier 4

```
{
  "summary": {
    "nb_gpx_files": 1,
    "nb_inserted_caches": 200,
    "nb_existing_caches": 0,
    "nb_inserted_found_caches": 0,
    "nb_updated_found_caches": 0,
    "nb_new_countries": 1,
    "nb_new_states": 1
  },
  "challenges_stats": {
    "matched": 21,
    "created": 21,
    "skipped_existing": 0
  }
}
```

**fichier 5** Code : 413 Details : Error: Request Entity Too Large Response body :

```
{
  "detail": "Fichier trop volumineux (>20 Mo)."
```

On constate que tous les imports du jeu de test se déroulent exactement comme prévu.

## 10 Veille sur les vulnérabilités

### 10.1 Outils de veille

#### Dépendances Backend

```
# Audit quotidien
pip-audit --desc

# Mise à jour sécurité
pip install --upgrade $(pip list --outdated | awk 'NR>2 {print $1}')
```

#### Dépendances Frontend

```
# Audit npm
npm audit

# Correction automatique
npm audit fix
```

### 10.2 Vulnérabilités identifiées et corrigées

#### CVE-2024-XXXXX - Injection dans lxml

- **Risque** : Injection XML via GPX malformé
- **Correction** : Mise à jour lxml 4.9.2 → 4.9.3
- **Mitigation** : Validation stricte du XML avant parsing

#### CVE-2024-YYYYY - XSS dans Vue Router

- **Risque** : XSS via paramètres d'URL
- **Correction** : Vue Router 4.2.0 → 4.2.5
- **Mitigation** : Sanitisation des params

### 10.3 Procédures de réponse

- **Détection** : Alertes automatiques GitHub/npm
- **Évaluation** : Analyse d'impact (CVSS score)
- **Correction** : Patch ou mise à jour
- **Test** : Validation non-régression
- **Déploiement** : Rolling update
- **Communication** : Notification utilisateurs si nécessaire

### 10.4 Bonnes pratiques adoptées

- **Principe du moindre privilège** : Permissions minimales
- **Defense in depth** : Sécurité multicouche
- **Fail secure** : Défaillance sécurisée
- **Audit trail** : Traçabilité complète
- **Security by design** : Sécurité dès la conception

## 11 Conclusion

### 11.1 Bilan du projet

GeoChallenge Tracker m'a permis de mettre en œuvre l'ensemble des compétences CDA à travers :

- Une **architecture moderne** 3-tiers containerisée
- Des **algorithmes complexes** (AST, scoring, projections)
- Une **sécurité robuste** multicouche
- Une **expérience utilisateur** optimisée mobile
- Des **pratiques DevOps** matures

### 11.2 Difficultés rencontrées

- **Parsing GPX multi-namespace** : Résolu avec lxml et XPath
- **Performance des agrégations** : Optimisé avec index composés
- **Clustering carte** : Intégration Leaflet.markercluster
- **Rate limits OSM** : Proxy cache Nginx

### 11.3 Perspectives d'évolution

- Mode offline avec synchronisation
- Application mobile native
- Partage social de challenges
- IA pour suggestions personnalisées
- Intégration API Geocaching.com

### 11.4 Satisfaction personnelle

Ce projet m'a permis plusieurs apprentissages clés :

- Approfondissement de l'architecture d'applications complexes
- Maîtrise des bases NoSQL géospatiales
- Implémentation d'algorithmes avancés
- Adoption des meilleures pratiques DevOps

Le résultat est une application professionnelle, performante et évolutive, prête pour une utilisation réelle par la communauté geocaching.

### 11.5 Avis des utilisateurs

*"J'ai adoré la fluidité de l'interface et la clarté du système de tâches. En quelques clics, je sais où j'en suis et quelles caches viser ensuite. L'import GPX multiple est un vrai gain de temps."*

— A

*"Je gagne un temps fou pour planifier mes sorties et avancer mes challenges."*

— P

*"Super pratique ! Vivement les exports GPX !"*

— m

*"Enfin un outil clair et fluide pour suivre mes challenges sans me perdre dans des tableaux interminables."*

— H