Cahier des charges API Strava => Notion

Version : 1.0 **Date :** 10 Décembre 2025 **Stack :** Python / Notion API / Strava API

1. Présentation du Projet

1.1. Objectif

Développer un micro-service (middleware) permettant de synchroniser automatiquement les nouvelles activités sportives depuis **Strava** vers une base de données **Notion**. L'objectif est de centraliser les données d'entraînement pour analyse, sans saisie manuelle.

1.2. Périmètre

- **Inclus :**
  - Import des **nouvelles** activités (postérieures à la mise en production).
  - Prise en charge de **tous les types** d'activités (Course, Vélo, Yoga, Musculation, etc.).
  - Gestion de la **création** (nouvelle activité) et de la **mise à jour** (modification du titre/description sur Strava).
  - Archivage : Si une activité est supprimée sur Strava, elle est **conservée** dans Notion (pas de suppression).
- **Exclu :**
  - Import de l'historique passé (backlog).
  - Génération de cartes/images du tracé GPS (pour l'instant).

2. Spécifications Fonctionnelles

2.1. Flux de données (User Flow)

1. L'utilisateur termine une activité et la synchronise sur Strava.
2. Strava envoie une notification (Webhook) au backend Python.
3. Le backend analyse l'événement :
   a. Si `create` : Récupération des détails et insertion dans Notion.
   b. Si `update` : Récupération des mises à jour et modification de la page Notion existante.
   c. Si `delete` : Ignorer l'événement.

2.2. Règles de Gestion

- **Format du Titre Notion :** `YYYY-MM-DD - [Nom de l'activité Strava]`
  - *Exemple :* `2025-10-12 - Sortie longue dimanche`
- **Identification unique :** L'ID de l'activité Strava (`activity_id`) sera stocké dans une propriété Notion dédiée pour permettre les mises à jour futures (clé primaire).
- **Sécurité des Tokens :**
  - Les identifiants statiques (Client ID, Client Secret) sont stockés en variables d'environnement.
  - Le `refresh_token` (qui change) sera stocké de manière sécurisée et persistante (voir Architecture).

3. Architecture Technique

3.1. Schéma Global

Le système repose sur une architecture événementielle (Event-Driven).

3.2. Stack Technique

- **Langage :** Python 3.x
- **Framework Web :** FastAPI (pour sa rapidité et sa gestion native de l'asynchrone) ou Flask.
- **Hébergement (Free Tier) :** Render, Railway ou Vercel.
- **Tunneling (Dev Local) : ngrok** (Outil requis pour exposer ton serveur local à Strava pendant le développement).

3.3. Stratégie de Stockage des Secrets (Sécurité)

Puisque le code sera sur un GitHub Public, aucun secret ne doit être "en dur" dans le code. Comme les hébergeurs gratuits ont souvent un système de fichiers éphémère (un fichier JSON local serait effacé à chaque redémarrage), nous utiliserons une approche hybride :

1. **Variables d'environnement (.env) :**
   a. `STRAVA_CLIENT_ID`
   b. `STRAVA_CLIENT_SECRET`
   c. `NOTION_API_KEY`

d. `NOTION_DB_ID` (ID de la base de données de sport)

2. **Stockage Persistant du Token ("The Config Vault") :**
   a. Nous créerons une **seconde petite base de données Notion** (privée) pour servir de stockage au script.
   b. Le script Python ira lire/écrire le `refresh_token` Strava directement dans cette base Notion.
   c. *Avantage :* Gratuit, sécurisé, persistant, et pas de fichier JSON à gérer dans le repo.

4. Structure des Données (Mapping)

4.1. Base de données Notion ("Activités")

Nom de la propriété	Type Notion	Description
Name	Title	`Date - Nom Strava`
Date	Date	Date et heure de début
Activity ID	Number	ID unique Strava (Clef de liaison)
Type	Select	Run, Ride, WeightTraining, etc.
Distance (km)	Number	Converti depuis les mètres
Durée (min)	Number	`moving_time` converti en minutes
D+ (m)	Number	Dénivelé positif (`total_elevation_gain`)
Calories	Number	`calories`

4.2. Base de données Notion ("Config") - *Pour la sécurité*

Une base simple à 2 colonnes pour stocker les tokens dynamiques :

- **Key** (Title) : ex: `strava_refresh_token`
- **Value** (Text) : *Le token cryptique...*

5. Plan de Développement (Roadmap)

Phase 1 : Setup & Outils

- Créer l'application sur le site développeur Strava (récupérer Client ID/Secret).
- Créer l'intégration Notion (récupérer Token).
- Mettre en place l'environnement Python local + Git.
- **Action clé :** Installer et comprendre `ngrok` (pour tester les webhooks).

Phase 2 : Authentification (Le plus dur d'abord)

- Script Python pour générer le premier token manuellement.
- Implémenter la logique : "Lire token dans Notion" -> "Si expiré, rafraîchir via Strava" -> "Sauvegarder nouveau token dans Notion".

Phase 3 : Le Webhook

- Créer une route `GET /webhook` pour valider le "Challenge" de Strava (vérification initiale).
- Créer une route `POST /webhook` pour recevoir les événements.

Phase 4 : Traitement & Écriture

- Coder la logique de récupération des détails de l'activité (GET activity).
- Coder le mapping JSON vers Notion.
- Gérer le cas "Update" (Rechercher la page Notion avec l'`Activity ID` et la mettre à jour).

Phase 5 : Déploiement

- Pousser le code sur GitHub (sans fichiers sensibles).
- Connecter à Render/Railway.
- Mettre à jour l'URL du webhook Strava vers l'URL de production.