TP: Création d'une API REST sécurisée pour les données de parkings de Nantes Métropole.

Objectifs pédagogiques

Ce TP a pour but de :

- **Apprendre à consommer une API Open Data.**
- **Traiter et transformer les données avant exposition.**
- **Concevoir une API REST sécurisée avec une API Key.**
- **Documenter et tester son API.**

Prérequis

Connaissance de base en développement.
Notions sur les API REST et l'utilisation de Postman ou curl.
Accès à l'open data Nantes Métropole

Énoncé du TP

Contexte :

Vous travaillez pour une startup qui souhaite proposer une **API** optimisée pour afficher la disponibilité des parkings en temps réel à Nantes. L'API doit récupérer les données depuis **l'Open Data de la ville**, effectuer un traitement simple, puis les exposer via un endpoint sécurisé.

Étapes du TP

Étape 1 : Récupération des données depuis l'Open Data

Consommer **l'API Open Data** de Nantes qui fournit les disponibilités des parkings en temps réel.

URL Open Data : cliquez-ici

Étape 2 : Transformation des données

Avant d'exposer les données, on va les nettoyer et les enrichir.

Tâches :

- **Filtrer uniquement les parkings ouverts.**
- **Calculer le taux d'occupation :**
- **Ajouter une classification de la disponibilité (élevée, moyenne, faible).**

Format de sortie attendu :

```
{
  "nom": "Gare Sud",
  "places_disponibles": 120,
  "capacite": 200,
  "taux_occupation": 40,
  "disponibilite": "élevée"
}
```

Indice : Définir la disponibilité comme suit :

- **disponibilité = "élevée" si grp_disponible > 50% de la capacité.**
- **disponibilité = "moyenne" si 20% < grp_disponible ≤ 50%.**
- **disponibilité = "faible" si grp_disponible ≤ 20%.**

Étape 3 : Création de l'API REST

Concevoir une API REST qui expose ces données transformées.

Tâches :

- **Ajouter une route GET /parkings qui retourne les données transformées.**
- **Tester l'API en local avec Postman ou curl.**

Étape 4 : Sécurisation avec une API Key

Restreindre l'accès à l'API en demandant une clé d'authentification.

Tâches :

- **Générer une API Key unique.**
- **Vérifier la clé dans chaque requête entrante (GET /parkings).**
- **Retourner une erreur 403 Forbidden si la clé est absente ou incorrecte.**

Étape 5 : Documentation et tests

Rédiger une documentation simple et tester l'API.

Tâches :

- **Ajouter une page README.md expliquant comment utiliser l'API.**
- **Tester l'API avec Postman**
- **Bonus : Ajouter une documentation Swagger.**

Etape 6 : Ecriture en base de données

Stocker sur une table de base de données les noms et capacité d'accueil des parkings, pour un traitement ultérieur.

Critères d'évaluation

- L'API récupère correctement les données Open Data.
- Les transformations sont bien appliquées (filtrage, calculs, formatage).
- L'API REST est fonctionnelle et expose les données sous /parkings.
- L'authentification API Key est en place et protège l'accès.
- La documentation est claire et permet de comprendre l'utilisation de l'API.