📘 Cahier des charges à partir du Tp observabilité

Application Backend – Gestion d'utilisateurs et de produits

---

1. 🎯 Objectif du projet

L'objectif de cette application backend est de fournir une API simple, sécurisée et extensible permettant :

- la gestion d'utilisateurs,
- la gestion d'un catalogue de produits,
- l'exposition d'un ensemble d'opérations CRUD via des endpoints REST,
- la persistance des données dans Firebase.

---

2. 🧱 Périmètre fonctionnel

2.1 Gestion des utilisateurs

L'application doit permettre de **créer un utilisateur** possédant les attributs suivants :

- **[ID](guide://action?prefill=Tell%20me%20more%20about%3A%20ID)** : identifiant unique
- **[Nom](guide://action?prefill=Tell%20me%20more%20about%3A%20Nom)**
- **[Âge](guide://action?prefill=Tell%20me%20more%20about%3A%20%C3%82ge)**
- **[Email](guide://action?prefill=Tell%20me%20more%20about%3A%20Email)**
- **[Mot de passe](guide://action?prefill=Tell%20me%20more%20about%3A%20Mot%20de%20passe)**

✅ La création d'un utilisateur se fait via un endpoint dédié.
✅ Les données sont stockées dans Firebase.

2.2 Gestion des produits

L'application doit fournir à l'utilisateur un ensemble d'opérations permettant de manipuler un **référentiel de produits**.
Chaque produit possède :

- **[ID](guide://action?prefill=Tell%20me%20more%20about%3A%20ID)** : identifiant unique
- **[Nom](guide://action?prefill=Tell%20me%20more%20about%3A%20Nom)**
- **[Prix](guide://action?prefill=Tell%20me%20more%20about%3A%20Prix)**

- **[Date d’expiration](guide://action?prefill=Tell%20me%20more%20about%3A%20Date%20d%E2%80%99expiration)**

Les fonctionnalités attendues :

✅ Afficher tous les produits

- Retourne la liste complète des produits stockés dans Firebase.
- Accessible via un endpoint REST.

✅ Récupérer un produit par ID

- Recherche un produit via son identifiant.
- Si aucun produit ne correspond → **exception / erreur HTTP**.
- **[Gestion stricte des erreurs](guide://action?prefill=Tell%20me%20more%20about%3A%20Gestion%20stricte%20des%20erreurs)**.

✅ Ajouter un produit

- Ajoute un nouveau produit dans Firebase.
- Si un produit avec le même ID existe → **exception / erreur HTTP**.
- **[Validation des données](guide://action?prefill=Tell%20me%20more%20about%3A%20Validation%20des%20donn%C3%A9es)** obligatoire.

✅ Supprimer un produit

- Supprime un produit via son ID.
- Si l’ID n’existe pas → **exception / erreur HTTP**.
- **[Opération irréversible](guide://action?prefill=Tell%20me%20more%20about%3A%20Op%C3%A9ration%20irr%C3%A9versible)**.

✅ Mettre à jour un produit

- Modifie un produit existant.
- Si l’ID n’existe pas → **exception / erreur HTTP**.
- Mise à jour partielle ou complète selon ton implémentation.

---

3. 🏗️ Architecture technique

3.1 Technologies utilisées

- **[Backend REST](guide://action?prefill=Tell%20me%20more%20about%3A%20Backend%20REST)** (Node.js, Java, Python… selon ton choix)
- **[Firebase Realtime Database ou Firestore](guide://action?prefill=Tell%20me%20more%20about%3A%20Firebase%20Realtime%20Database%20ou%20Firestore)** pour la persistance
- **[API RESTful](guide://action?prefill=Tell%20me%20more%20about%3A%20API%20RESTful)** exposée via HTTP
- **[Gestion des erreurs standardisée](guide://action?prefill=Tell%20me%20more%20about%3A%20Gestion%20des%20erreurs%20standardis%C3%A9e)** (codes HTTP 400/404/409/500)

---

4. 🔐 Contraintes et règles métier

4.1 Contraintes sur les utilisateurs

- L'email doit être unique.
- Le mot de passe doit être stocké de manière sécurisée (hash si applicable).
- L'ID utilisateur doit être généré automatiquement ou validé comme unique.

4.2 Contraintes sur les produits

- L'ID produit doit être unique.
- La date d'expiration doit être une date valide.
- Le prix doit être un nombre positif.
- Toute opération sur un ID inexistant doit générer une erreur.

---

5. 🌐 Endpoints API (exemple)

Fonction	Endpoint	Méthode
[Créer un utilisateur](guide://action?prefill=Tell%20me%20more%20about%3A%20Cr%C3%A9er%20un%20utilisateur)	`/users`	POST
[Lister les produits](guide://action?prefill=Tell%20me%20more%20about%3A%20Lister%20les%20produits)	`/products`	GET
[Récupérer un produit](guide://action?prefill=Tell%20me%20more%20about%3A%20R%C3%A9cup%C3%A9rer%20un%20produit)	`/products/{id}`	GET
[Ajouter un produit](guide://action?prefill=Tell%20me%20more%20about%3A%20Ajouter%20un%20produit)	`/products`	POST

|**[Supprimer un produit](guide://action?prefill=Tell%20me%20more%20about%3A%20Supprimer%20un%20produit)**|`/products/{id}`|DELETE|
|**[Mettre à jour un produit](guide://action?prefill=Tell%20me%20more%20about%3A%20Mettre%20%C3%A0%20jour%20un%20produit)**|`/products/{id}`|PUT/PATCH|

---

6. ✅ Critères d’acceptation

- L’application doit fonctionner avec Firebase sans erreurs.
- Toutes les opérations CRUD doivent être testées et validées.
- Les erreurs doivent être renvoyées sous forme de réponses HTTP cohérentes.
- Le code doit être lisible, structuré et documenté.

---

7. 📦 Livrables attendus

- Code source complet du backend
- Documentation des endpoints (Swagger, Postman, ou Markdown)
- Cahier des charges (ce document)
- Instructions d’installation et d’exécution