

## Analyse détaillée des technologies du projet

### Backend - Framework & Serveur

#### FastAPI 0.124.4

- **Rôle :** Framework web Python pour créer des API REST
- **Avantages :**
  - **Performance** : Plus rapide que Django/Flask (basé sur Starlette + Pydantic)
  - **Documentation auto** : Génère OpenAPI/Swagger automatiquement (/docs)
  - **Validation** : Vérifie automatiquement les données entrantes avec Pydantic
  - **Async natif** : Support async/await pour hautes performances
  - **Autocomplétion** : Typage fort → meilleur support IDE
- **Alternatives** : Django REST (plus lourd), Flask (moins moderne), Express.js (Node.js)

#### Uvicorn 0.34.0

- **Rôle :** Serveur ASGI pour exécuter FastAPI
- **Avantages :**
  - Ultra-rapide (basé sur uvloop)
  - Hot reload en développement
  - Support async natif
- **Alternatives** : Gunicorn (WSGI, moins performant), Hypercorn

---

### Backend - Base de données

#### SQLModel 0.0.27

- **Rôle :** ORM (Object-Relational Mapping) pour PostgreSQL
- **Avantages :**
  - **Best of both worlds** : Combine SQLAlchemy (ORM puissant) + Pydantic (validation)
  - **Moins de code** : Une seule classe = modèle DB + schéma API
  - **Type safety** : Détection d'erreurs avant exécution
  - Crée par le même auteur que FastAPI (intégration parfaite)
- **Alternatives** : SQLAlchemy pur (plus verbeux), Django ORM (lié à Django), Tortoise ORM

## psycopg2-binary 2.9.10

- **Rôle :** Driver PostgreSQL pour Python
- **Avantages :**
  - 🏆 Standard de facto pour PostgreSQL en Python
  - ⚡ Version binary = pré-compilée (pas besoin de compiler)
  - 🔒 Très stable et mature
- **Alternatives :** asyncpg (async pur mais plus complexe), psycopg3

## PostgreSQL 15

- **Rôle :** Base de données relationnelle
- **Avantages :**
  - 🏆 **Open source** leader en fonctionnalités avancées
  - 🔗 **Relations complexes** : Foreign keys, joins performants
  - 📊 **Types riches** : JSON, arrays, dates avancées
  - 🔒 ACID compliance (transactions fiables)
  - 🚀 Excellent pour analytics (vs MySQL)
- **Alternatives :** MySQL (moins de features), SQLite (fichier simple), MongoDB (NoSQL)

---

## Backend - Traitement de données

### Pandas 2.3.3

- **Rôle :** Analyse et manipulation de données tabulaires
- **Avantages :**
  - 📊 **Standard industrie** pour data science
  - 🛠️ **Parsing CSV** puissant avec détection automatique de types
  - 🎯 **Groupby/Aggregate** : Facile de grouper par joueur, calculer moyennes
  - 📈 Compatible avec Matplotlib/Seaborn
- **Exemple :** Votre catapult\_parser.py lit les CSV Catapult et groupe par joueur
- **Alternatives :** Polars (plus rapide mais moins mature), CSV stdlib (trop basique)

### Matplotlib 3.10.8

- **Rôle :** Génération de graphiques et visualisations

- **Avantages :**
  - **Contrôle total** : Personnalisation pixel par pixel
  - Tous types de graphiques (gauges semi-circulaires custom)
  - Export PNG/PDF/SVG
  - Bibliothèque la plus utilisée en Python
- **Votre usage** : Rapports professionnels avec header, logo, jauge, tableaux
- **Alternatives** : Plotly (interactif HTML), Seaborn (simplifie Matplotlib)

## Seaborn 0.13.2

- **Rôle** : Visualisations statistiques élégantes
- **Avantages :**
  - **Thèmes pro** par défaut (vs Matplotlib brut)
  - Graphiques statistiques en 1 ligne
  - S'intègre avec Matplotlib (même backend)
- **Votre usage** : Probablement pour le color scheme et l'esthétique
- **Alternatives** : Matplotlib pur (moins joli), Plotly Express

## Pillow 12.0.0

- **Rôle** : Manipulation d'images (PIL modernisé)
- **Avantages :**
  - Ouvre/redimensionne/affiche images
  - Intégration Matplotlib (via [OffsetImage](#))
  - Formats multiples (PNG, JPG, etc.)
- **Votre usage** : Afficher le logo Thonon Evian dans les rapports
- **Alternatives** : OpenCV (overkill), imageio

## python-multipart 0.0.21

- **Rôle** : Parser les uploads de fichiers (multipart/form-data)
- **Avantages :**
  - Requis par FastAPI pour UploadFile
  - Gère les gros fichiers en streaming
- **Votre usage** : Upload des CSV Catapult via [/catapult/upload](#)
- **Alternatives** : Aucune (dépendance obligatoire FastAPI)

---

## Frontend - Framework

### React 19.2.0

- **Rôle :** Framework JavaScript pour interfaces utilisateur
- **Avantages :**
  -  **Composants réutilisables** : Découpage modulaire de l'UI
  -  **Virtual DOM** : Mises à jour ultra-rapides
  -  **Écosystème énorme** : Millions de bibliothèques
  -  **Standard industrie** : Utilisé par Facebook, Netflix, etc.
  -  **Hooks** : useState, useEffect pour logique moderne
- **Alternatives :** Vue.js (plus simple), Angular (plus verbeux), Svelte (compile)

### Vite 7.2.4

- **Rôle :** Build tool et serveur de développement
- **Avantages :**
  -  **Ultra-rapide** : Hot Module Replacement instantané
  -  Dev server démarre en <1s (vs 30s webpack)
  -  Build optimisé avec Rollup
  -  Configuration minimale
  -  Support TypeScript/JSX natif
- **Alternatives :** Create React App (obsolète), Webpack (lent), Next.js (overkill pour SPA)

### ESLint 9.39.1

- **Rôle :** Linter JavaScript/TypeScript
  - **Avantages :**
    -  Déetecte erreurs avant exécution
    -  Force conventions de code
    -  Auto-fix pour beaucoup d'erreurs
    -  Plugins React spécifiques (hooks rules)
  - **Alternatives :** Prettier seul (formatte mais ne vérifie pas logique), TSLint (déprécié)
-

## Infrastructure

### Docker Compose 3.8

- **Rôle :** Orchestration de conteneurs
  - **Avantages :**
    - **Environnement reproductible** : Même config partout
    - **Un fichier** définit toute l'infra
    - docker-compose up lance tout
    - Volumes pour persistance PostgreSQL
  - **Votre usage :** PostgreSQL isolé du système, facile à reset
  - **Alternatives :** Installation locale (pollution du système), Kubernetes (overkill)
- 

### Pourquoi cette stack ensemble ?

1. **FastAPI + SQLAlchemy** : Cohérence Pydantic partout (validation uniforme)
2. **Pandas + Matplotlib** : Standard data science (compatibilité maximale)
3. **React + Vite** : Combo moderne ultra-rapide pour SPA
4. **PostgreSQL + Docker** : Base pro en dev isolé
5. **Python 3.12** : Dernières features (type hints améliorés, performances)