PokéBoutique

Application E-commerce de Cartes Pokémon

Présentation du Projet et de l'Architecture Technique

Bonjour à tous, et merci de prendre quelques minutes pour cette présentation de PokéBoutique, une application e-commerce dédiée aux cartes Pokémon.

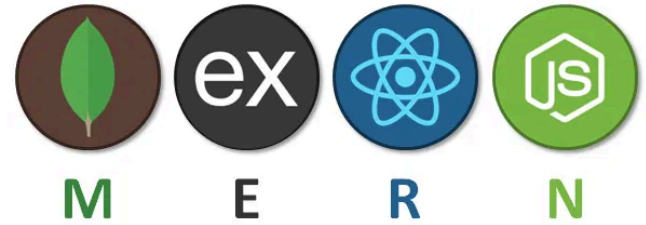
Une Expérience E-commerce Complète sur une Stack MERN Containerisée

Objectif Principal

Proposer une boutique en ligne fonctionnelle pour la vente de cartes Pokémon.

Double Enjeu du Projet

- **Frontend :** Expérience utilisateur élégante (catalogue, panier, paiement simulé).
- **Backend :** API propre, testable et facilement déployable via Docker.



Stack Technique : MERN (MongoDB, Express, React, Node.js) + TailwindCSS**.**

Orchestration : Docker Compose (Mongo, Backend Node.js, Frontend React)**.**

Architecture en Trois Services Docker pour une Isolation Optimale

MongoDB 7.0

Rôle : Base de données NoSQL pour le stockage persistant.

Contenu : Gère les collections pour les cartes Pokémon (catalogue) et les paniers utilisateurs (sessions).

Orchestration : Service dédié dans `docker-compose.yml` avec volume pour la persistance des données.

Backend Node.js 20

Stack : Express 4 et Mongoose.

Architecture : Découpage "Clean" (Routes, Contrôleurs, Services, Modèles) pour une logique métier isolée.

Fonction : Expose l'API REST sous `/api` et gère la logique e-commerce (panier, checkout).

Tests : Couverture assurée par Jest et Supertest.

Frontend React 18

Stack : React 18, Vite, et TailwindCSS pour le style moderne.

Fonction : Application monopage (SPA) gérant la navigation (React Router) et l'interface utilisateur.

Logique Panier : Centralisée dans le hook `useCart`, gérant l'ID de session via `localStorage`.

Tests : Utilise Vitest et React Testing Library (RTL).

Démonstration : Du Catalogue à la Confirmation de Commande

1. Lancement des Services

Démarrage des trois services Docker (Mongo, Backend, Frontend) via `docker compose up --build`. Le frontend est accessible sur **http://localhost:5173**.

2. Catalogue et Ajout au Panier

La page d'accueil affiche le catalogue. Chaque carte propose les actions "Détails" et **"Ajouter"**. L'ajout déclenche un appel à `/api/cart` et met à jour le badge du panier.

3. Gestion du Panier

Sur la page `/cart`, l'utilisateur peut visualiser la liste des articles, ajuster les quantités, et voir le sous-total mis à jour. Chaque modification rafraîchit le total via l'API.

4. Checkout et Confirmation

Le formulaire de paiement fictif sur `/checkout` valide la commande. L'endpoint `/api/cart/checkout` simule le paiement, génère un **orderId**, et redirige vers la page de confirmation.

Qualité et Testabilité : Le Cœur du Backend

Backend & API Standardisée

- **Stack :** Express 4 et Mongoose 8.
- **Logique Métier :** Utilisation de services dédiés (`cardService`, `cartService`) pour isoler les règles métier.
- **Réponses API :** Standardisées. Les succès renvoient `{ data, message? }` et les erreurs `{ error }` avec un code HTTP adapté.

Stratégie de Tests

- **Backend :** Jest + Supertest avec **Mongo Memory Server** pour des tests d'intégration rapides et isolés.
- **Frontend :** Vitest + React Testing Library (RTL) pour valider les composants et les hooks (`useCart`, `useProducts`).
- **Stratégie :** Couverture des scénarios nominaux, validation des entrées, et gestion des erreurs.

Objectif de Couverture : ≥70%

Conclusion & Roadmap

Prochaines Étapes (Roadmap)

1. Intégration d'une **CI** (GitHub Actions) pour automatiser les tests et le build.
2. Ajout de **pagination et filtres** (type, rareté, prix) sur le catalogue pour améliorer la navigation.
3. Renforcement des tests (ex: `msw` pour le frontend) pour une couverture complète des scénarios.
4. Préparation au déploiement Cloud (en gardant Docker comme base).

Bilan du Projet

PokéBoutique offre une démonstration complète d'une stack MERN containerisée : un backend proprement découpé, un frontend moderne et stylé, et une expérience de test/documentation soignée.

Call to Action

Pour découvrir le projet, suivez ces étapes :

```
git clone [REPO_URL]
docker compose up --build
docker compose exec backend npm run seed
```

Merci de votre attention.