CAHIER DES CHARGES - PROJET SAE
Groupe PAQUETA
Version 2 du 08/12/2024

1. Introduction

A. Objet du document

Ce cahier des charges a pour but de définir précisément les attentes, les objectifs, et les contraintes liés à la réalisation du projet **MyPokemonList**. Ce projet est développé dans le cadre d'une Situation d'Apprentissage et d'Évaluation (SAÉ) en troisième année du BUT Informatique. Le livrable attendu est une application web permettant de gérer et de suivre une collection de cartes Pokémon, en proposant une interface conviviale et intuitive pour les utilisateurs. Ce document formalise les besoins exprimés, le contexte du projet, ainsi que les enjeux techniques et organisationnels, tout en s'assurant du respect des contraintes définies dans le cadre pédagogique.

B. Contexte SAE

Dans le cadre de cette SAÉ, l'équipe Paqueta est amenée à développer une application web innovante et fonctionnelle tout en respectant des exigences pédagogiques strictes. L'objectif est de fournir un outil capable de répondre à un besoin spécifique dans un domaine défini.

Le projet choisi par l'équipe est une plateforme de carte Pokemon, enrichie de fonctionnalités personnalisées. Les utilisateurs pourront consulter et rechercher des cartes Pokémon par génération, rareté ou autres critères, tout en ayant la possibilité de gérer leur propre collection virtuelle. Ce projet s'inscrit dans une volonté d'apprentissage approfondi des technologies modernes telles que MongoDB, Node.js (Express), et React.js, tout en respectant les méthodologies de développement collaboratif.

L'équipe projet doit concevoir une solution complète et optimisée, allant du backend (API RESTful) à un frontend interactif. Les livrables incluront une interface web fonctionnelle, un dépôt Git bien structuré, et un rapport décrivant les choix techniques et organisationnels. Le projet est à réaliser dans un cadre temporel contraint, avec une date butoir fixée au **01/01/2025**.

C. Enjeux et objectifs

Le projet **MyPokemonList** vise plusieurs objectifs, répondant à la fois aux besoins des utilisateurs et aux exigences académiques :

1. **Faciliter la gestion de collections personnelles** :
    - Permettre aux utilisateurs de suivre les cartes qu'ils possèdent.
    - Proposer une interface simple pour rechercher et filtrer les cartes par génération, rareté ou extension.
2. **Apprentissage des technologies modernes** :
    - Maîtriser des outils tels que MongoDB pour gérer une base de données NoSQL complexe.
    - Développer une API performante avec Express.js pour la gestion des données.
    - Créer un frontend dynamique et ergonomique avec React.js.
3. **Respect des contraintes pédagogiques** :
    - Réaliser un projet collaboratif dans un environnement structuré (Git).
    - Livrer une application optimisée avec des métriques de performance.
    - Produire un rapport détaillé en respectant les consignes de la SAÉ.
4. **Offrir une expérience utilisateur fluide et intuitive** :
    - Concevoir une interface ergonomique adaptée à tous les supports.
    - Assurer des performances optimales pour la recherche et la gestion des collections.

En résumé, **MyPokemonList** représente une opportunité unique pour l'équipe projet de combiner apprentissage technique et créativité, tout en répondant à un besoin réel pour une communauté passionnée par les cartes Pokémon. Le projet vise également à démontrer les compétences acquises en développement, en optimisation des performances, et en travail collaboratif.

2. Environnement

A. Description des outils, technologies et méthodologies utilisés

Le projet **MyPokemonList** s'appuie sur des outils modernes et des technologies éprouvées pour garantir un développement efficace et une performance optimale. Voici les principaux éléments utilisés :

1) Base de données :
    a) **MongoDB** : Base NoSQL permettant de gérer les relations complexes entre utilisateurs, collections, cartes, blocs et extensions.

2) Backend :

a) Node.js avec Express : Framework léger et rapide pour développer l'API RESTful du projet.

3) Frontend :
    a) **React.js** : Bibliothèque JavaScript pour construire une interface utilisateur dynamique et réactive.
    b) **Tailwind CSS** : Framework CSS pour concevoir une interface moderne et responsive.

4) Outils de gestion de projet et de collaboration :
    a) **Git** et **GitLab** : Pour le contrôle de version et la collaboration entre membres de l'équipe.
    b) **Google Drive** : Stockage et partage des documents liés au projet.

5) Virtualisation et conteneurisation :
    a) **Docker** : Utilisé pour standardiser l'environnement de développement et de production, garantissant une compatibilité entre les environnements.

B. Environnement de développement

1) Système d'exploitation (OS) :
    a) Machines de développement sous Windows, macOS ou Linux.
    b) Déploiement en environnement Linux (via le VPS).
2) IDE et éditeurs de code :
    a) **Visual Studio Code** : Choisi pour son intégration avec Git, ses extensions, et sa facilité d'utilisation.
3) Frameworks et librairies :
    a) **Express.js** pour le backend.
    b) **React.js** et **Tailwind CSS** pour le frontend.

C. Déploiement prévu

1) Serveurs :
    a) L'application sera hébergée sur un **VPS privé**, garantissant un contrôle total sur le serveur et une flexibilité pour les besoins futurs.
    b) Deux sous-domaines dédiés seront configurés :

i) **Frontend** : `MyPokemonList.but-info.xyz`
ii) **API Backend** : `api.MyPokemonList.but-info.xyz`

2) Hébergement et infrastructure :
    a) Utilisation de Docker pour déployer l'application dans un environnement isolé, minimisant les risques de conflits et facilitant les mises à jour.
    b) Configuration du VPS pour assurer un accès sécurisé (via HTTPS) et des performances optimales.

En combinant ces outils et technologies, l'équipe garantit un environnement de développement cohérent, une collaboration efficace, et un déploiement fiable, tout en respectant les contraintes et les bonnes pratiques de développement.

3. Interlocuteur(s) du projet

A. Equipe Paqueta

- VERCHERE Mathis
- MERI Quentin
- ARUMUGADAS Ravichandran
- AKKOU Layn
- DAO Viet Long

B. Evaluateurs

- Responsable du projet : M. Gaël Guibon
- Équipe d'évaluation :
    - Développement : M. Jean-Christophe Dubacq, M. Gaël Guibon
    - Optimisation : M. David Hébert
    - Communication : Mme Marie-Eva Lesaunier

4. Contexte

A. Présentation du besoin

Présentation du besoin

Le projet **MyPokemonList** répond à une demande croissante des amateurs de cartes Pokémon, qui souhaitent un outil numérique simple et efficace pour gérer leur collection personnelle. L'objectif principal est de fournir une plateforme permettant aux utilisateurs de :

- Rechercher des cartes Pokémon selon divers critères, tels que le nom, la génération, la rareté, ou les extensions.
- Enregistrer et organiser leurs cartes pour constituer une collection virtuelle.
- Obtenir des informations détaillées sur chaque carte, comme son type, ses caractéristiques et son bloc d'appartenance.

En plus de répondre à ce besoin spécifique, l'application vise à offrir une expérience utilisateur intuitive et rapide, accessible à la fois sur ordinateur et mobile.

B. Analyse de l'existant

Le projet s'inscrit dans un domaine où peu de solutions personnalisées et interactives existent pour la gestion des collections personnelles de cartes Pokémon. Bien que des bases de données ou des catalogues soient disponibles pour consulter les informations des cartes, ces solutions sont souvent limitées aux fonctionnalités de consultation, sans permettre de gestion active et individuelle des collections.

MyPokemonList se distingue par :

- Une interface utilisateur moderne et personnalisable.
- La possibilité pour chaque utilisateur de créer et gérer sa propre collection virtuelle.
- Des fonctionnalités de recherche avancée et rapide, adaptées aux besoins spécifiques des collectionneurs.

C. Finalité du projet

La réalisation de **MyPokemonList** s'inscrit dans un double objectif académique et personnel :

1. **Apprentissage et mise en pratique des compétences** :

    - Maîtriser les technologies modernes, telles que MongoDB, Node.js (Express), React.js et Docker.
    - Appliquer les méthodologies de développement collaboratif avec Git et GitLab.
    - Comprendre les concepts avancés liés à la conception et à l'optimisation d'applications web.
2. **Production de livrables pour le SAE** :
    - Développer une application web fonctionnelle et innovante.
    - Répondre aux exigences techniques et organisationnelles définies dans le cadre du projet.
    - Fournir un rapport documentant les choix technologiques, les étapes de développement et les résultats obtenus.

En somme, **MyPokemonList** représente une opportunité d'allier passion pour les cartes Pokémon et défis techniques, tout en valorisant les compétences en développement informatique dans un cadre structurant et formateur.

5. Limites

A. Limites temporelles

- La date butoir pour la livraison finale du projet est fixée au **01/01/2025**.
- Le calendrier est contraint par les exigences du SAE et laisse peu de marge pour des retards ou des modifications majeures en fin de parcours.
- Les jalons intermédiaires doivent être respectés pour assurer une progression constante et éviter les blocages en fin de projet.

B. Limites techniques

Budget limité :

- En tant qu'étudiants, nous disposons de moyens financiers modestes. Les choix technologiques privilégient des solutions open-source et gratuites.

Ressources matérielles :

- Chaque membre de l'équipe utilise un **ordinateur fixe personnel**, sans possibilité immédiate d'accès à des environnements professionnels ou hautement performants.
- Le projet repose sur un **petit VPS** pour l'hébergement, offrant des capacités limitées mais suffisantes pour le projet.

Ressources disponibles :

- Un nom de domaine a été acquis pour le projet, permettant d'héberger les différentes parties de l'application sous les sous-domaines :
    - `mypokemonlist.but-info.xyz` pour le frontend.
    - `api.mypokemonlist.but-info.xyz` pour le backend.
- Bien que le VPS soit fonctionnel, il reste soumis à des contraintes de bande passante, de stockage, et de performance.

Compétences :

C. Les membres de l'équipe sont encore en phase d'apprentissage, ce qui peut engendrer des défis techniques liés aux technologies modernes comme Docker, Tailwind CSS ou MongoDB.

D. Limites fonctionnelles

- **Pas de fonctionnalités de marketplace** : Le projet n'inclut pas d'échange ou de vente de cartes entre utilisateurs.
- **Fonctionnalités simplifiées pour les recommandations** : Bien que des recommandations soient envisagées, elles seront limitées à des filtres statiques (par exemple, génération ou rareté) et non à des algorithmes d'intelligence artificielle.
- **Échelle réduite** : L'application vise principalement une audience réduite pour les tests utilisateurs, en raison des limitations du VPS et des contraintes de charge.

6. Exigences fonctionnelles

A. Description globale

Le projet **MyPokemonList** vise à développer une application web permettant aux utilisateurs de gérer leur collection de cartes Pokémon, tout en explorant les séries et générations disponibles. Voici les pages et fonctionnalités principales attendues :

- Page de connexion :
    - Authentification des utilisateurs via e-mail et mot de passe.
    - Création de compte pour les nouveaux utilisateurs.
    - Fonction de réinitialisation de mot de passe en cas d'oubli.

- Page d’accueil :
    - Présentation des fonctionnalités principales et des dernières mises à jour.
    - Accès rapide à la collection personnelle, à la recherche de cartes et aux paramètres.

- Page de séries :
    - Liste des séries et extensions Pokémon organisées par génération.
    - Recherche et filtrage des cartes par rareté, type ou extension.
    - Informations détaillées pour chaque série (date de sortie, nombre de cartes, etc.).

- Page de recherche avancée :
    - Recherche globale avec filtres multicritères (nom, génération, rareté, type).
    - Affichage rapide des résultats sous forme de liste ou de vignettes.

- Page de collection personnelle :
    - Visualisation des cartes ajoutées à la collection d’un utilisateur.
    - Statistiques sur la collection (cartes possédées par série, pourcentage complété).
    - Ajout ou suppression de cartes via une interface intuitive.

- Page des paramètres utilisateur :
    - Modification des informations personnelles (nom, avatar, e-mail).
    - Choix des préférences de langue, thème (clair/sombre), et notifications.

- Page d'administration (optionnelle) :
    - Accessible uniquement aux administrateurs pour gérer le contenu, les séries, et les utilisateurs (modération).

Fonctionnalités principales

- Gestion des collections :
    - Permettre aux utilisateurs d’ajouter, modifier ou supprimer des cartes de leur collection.
    - Génération automatique de statistiques basées sur les données de la collection.

- Exploration des séries :
    - Consultation des cartes pour chaque série ou génération.
    - Détails et illustrations des cartes disponibles.

- Recherche intuitive :
    - Barre de recherche rapide avec suggestions automatiques.
    - Filtres par type, rareté, génération, et bloc.

- Personnalisation de l’expérience utilisateur :
    - Interface adaptable aux préférences des utilisateurs (thème, langue).
    - Notifications pour les mises à jour des séries ou nouvelles cartes ajoutées.

B. User Stories

En tant que..	Je veux..	Afin de..
Visiteur	Pouvoir voir les différentes cartes qui existes	connaître ce qui existe
Visiteur	Consulter les séries Pokémon	Découvrir le contenu sans avoir à m'inscrire
Visiteur	Tester la recherche avancée sans créer de compte	Découvrir les fonctionnalités avant de m'inscrire
Visiteur	Consulter une carte détaillée avec son illustration et ses caractéristiques	Découvrir les informations précises sur les cartes
Visiteur	Accéder aux mentions légales et politiques de confidentialité	M'assurer que mes données sont protégées
Membre	pour me connecter	de faire une collection
Membre	Modifier mes informations personnelles	Personnaliser mon profil
Membre	Rechercher des cartes par génération ou rareté	Trouver facilement les cartes que je souhaite voir
Membre	Me connecter	Faire une collection
Membre	Ajouter des cartes à ma collection	Suivre les cartes que je possède
Membre	Partager ma collection avec d'autres utilisateurs via un lien	Montrer mes cartes à mes amis
Membre	Exporter ma collection en format CSV ou JSON	Garder une copie de ma collection en dehors du site
Membre	Trier les cartes dans ma collection par rareté, type, ou date d'ajout	Organiser facilement mes cartes
Modérateur	voir les pseudo des utilisateurs	modérer, en cas d'abus
Modérateur	Vérifier les profils utilisateurs douteux (ex. : comptes multiples)	Assurer l'intégrité de la communauté

Modérateur	Supprimer des commentaires ou comptes abusifs	Maintenir un environnement sain
Admin	Ajouter de nouvelles séries ou cartes dans la base de données	Mettre à jour le contenu du site

7. Exigences non fonctionnelles

Techniques :

Utilisation de frameworks modernes :

- Backend : **Express.js** pour la création de l'API RESTful.
- Frontend : **React.js** pour le développement d'une interface utilisateur interactive.
- **Tailwind CSS** pour la gestion du style et de la mise en page.
- **MongoDB** comme base de données NoSQL, adaptée aux grandes collections.
- **Docker** pour assurer la portabilité et l'uniformité des environnements de développement et de production.

Respect des conventions de code :

- Utilisation des bonnes pratiques en JavaScript (ES6+).
- Respect des conventions de nommage, des structures de fichiers, et du formatage de code standardisé via un linter (ESLint, Prettier).
- Versionnement du code sur **GitLab**.

Performance :

Temps de réponse :

- Le système doit afficher les résultats de recherche en moins de **2 secondes**, même sur des requêtes complexes.

Capacité d'accueil :

- L'application doit pouvoir supporter jusqu'à **1 000 utilisateurs simultanés**, grâce à un backend optimisé et une base de données bien structurée.

Ergonomie :

Interface claire et intuitive :

- Conception d'une interface utilisateur simple, organisée et facile à prendre en main, avec des éléments bien définis.

Compatibilité mobile :

- L'application doit être **responsive** et compatible avec les écrans d'ordinateurs, tablettes et smartphones.

Volumétrie :

Données à gérer :

- La base de données doit pouvoir stocker **plusieurs milliers de cartes**, y compris leurs images associées, descriptions, types, raretés et autres métadonnées.
- Le système doit permettre un accès rapide à ces données via des recherches indexées.

Sécurité :

Stockage sécurisé des mots de passe :

- Utilisation d'un algorithme de **hashage sécurisé (bcrypt)** pour protéger les mots de passe utilisateurs.
- Mise en place de **tokens JWT** pour la gestion des sessions utilisateurs.

Protection contre les attaques :

- Prévention des attaques par injection **SQL/NoSQL**.
- Sécurisation de l'API via **HTTPS** et validation des entrées utilisateurs.

Développement durable :

Optimisation du code :

- Réduction des charges inutiles sur le serveur en optimisant les requêtes API et les accès à la base de données.
- Minimisation des fichiers et du chargement côté client grâce à des outils comme **Webpack** ou **Vite.js**.

Technologies open source :

8. Utilisation exclusive de technologies **open source** (Node.js, React.js, MongoDB, Docker, GitLab) pour limiter les coûts et favoriser le partage de connaissances.

9. Risques et gestion des risques

Risques	Probabilité	Ce qu'on compte faire pour y remédier
Retards dans les délais : Des problèmes de planification, des complications techniques inattendues ou des exigences changeantes peuvent causer des retards.	7/10	C'est le plus probable mais c'est également un risque qui peut être remédié facilement si on en est conscient. Pour cela il est important de prendre de l'avance quand on peut et également surestimer le temps que chaque tâche va nous prendre pour être préparé face à des imprévus.
Failles de performance sous forte charge	5/10	Effectuer des tests de charge pour identifier les points faibles. Optimiser les requêtes MongoDB et réduire les appels API inutiles. Augmenter les capacités du VPS si nécessaire en prévoyant un budget pour une montée en gamme.
Conflits dans l'équipe ou mauvaise coordination	3/10	Organiser des réunions hebdomadaires pour suivre l'avancement. Utiliser un outil comme Trello ou Jira pour attribuer et suivre les tâches. S'assurer que chaque membre connaît ses responsabilités pour éviter les chevauchements et les malentendus.
Pannes matérielles ou dysfonctionnement du VPS	3/10	Utiliser des sauvegardes régulières des données et des configurations du serveur. Mettre en place une stratégie de restauration rapide en cas de panne, incluant une documentation claire pour réinstaller le projet sur un autre serveur si nécessaire.
Problèmes d'intégration frontend/backend	3/10	Tester régulièrement l'intégration des composants frontend avec l'API backend dès les premières phases de développement. Utiliser des outils comme Postman pour vérifier les endpoints et éviter les incompatibilités.
Problèmes de sécurité :	2/10	A priori ça ne devrait pas poser de

Vulnérabilités, failles de sécurité ou mauvaise gestion des données peuvent compromettre la sécurité de l'application.		problèmes car on va s'en occuper dans le code pour que ça soit le plus sécurisé possible. On va faire plusieurs tests à travers le projet pour voir si tout est bien sécurisé.
Changements dans les exigences : Les exigences du client ou de l'utilisateur final pourraient évoluer pendant le processus de développement, conduisant à des révisions fréquentes et des changements de direction.	2/10	Même si c'est peu probable, pour y remédier il est important d'être très clair dans le cahier des charges pour qu'il n'y est aucune ambiguïté. Ainsi, si les exigences du client viennent à évoluer on pourra dire que ce n'était pas prévu.

10. Principaux jalons

Jalon 1 : Étude préalable et schéma de la base de données (fin novembre 2024).
Jalon 2 : Développement du backend avec un POC fonctionnel (mi-décembre 2024).
Jalon 3 : Développement du frontend avec l'intégration API (fin décembre 2024).
Jalon 4 : Tests utilisateurs et ajustements finaux (semaine avant 01/01/2025).