CAHIER DES CHARGES - PROJET Perso
CyberVeilleProject
Version 1 du 30/01/2025

● Introduction

A. Objet du document

Ce document définit le cadre du projet CyberVeilleProject, un site web de veille technologique permettant aux utilisateurs de suivre l'actualité sur plusieurs thématiques (cybersécurité, intelligence artificielle, business, etc.). Il sert de référence pour le développement, en détaillant les exigences fonctionnelles, techniques et les contraintes du projet.

B. Contexte projet personnel

Le projet CyberVeilleProject est développé en solo (moi) par un étudiant passionné par la cybersécurité et l'informatique. Ayant pour objectif de reproduire un processus de développement structuré, similaire à celui utilisé en entreprise.
Un site web dédié à la veille technologique et stratégique sur plusieurs domaines comme la cybersécurité, l'intelligence artificielle, le business et plus encore. Il permettra aux utilisateurs de suivre automatiquement les dernières actualités via email, webhook ou API.

C. Enjeux et objectifs

- Permettre aux utilisateurs de suivre des actualités pertinentes grâce à un système de collecte automatisée (scraping, RSS, APIs).
- Offrir une expérience personnalisable avec des abonnements par email, webhook Discord ou API.
- Fournir un dashboard ergonomique permettant de gérer les préférences de veille.
- Assurer une scalabilité et une performance optimales via un système de cache et une architecture optimisée.
- Garantir un développement structuré avec méthodologies professionnelles (Agile, gestion des risques, documentation complète).

● Environnement

A. Description des outils, technologies et méthodologies utilisés

Backend

- Node.js avec NestJS (API REST)
- MongoDB (Base de données NoSQL)
- Puppeteer / Feedparser (Scraping & récupération des flux RSS)
- Gestion des notifications : SendGrid (email), Discord Webhooks, API REST

Frontend

- Next.js (React.js) (Framework moderne et optimisé pour le SEO)
- Tailwind CSS (Design fluide et ergonomique)
- React Query (Gestion d'état avancée)

Infrastructure & Déploiement

- Hébergement Backend : VPS
- Hébergement Frontend : VPS
- CI/CD : GitHub Actions

Méthodologies

- Développement Agile (Scrum/Kanban)
- Documentation API avec Swagger
- Tests automatisés (Jest / Cypress)

B. Environnement de développement

- IDE : Visual Studio Code
- Système de versioning : GitHub
- Base de données en local : MongoDB
- Tests et simulation API : Postman

C. Déploiement prévu

- Séparation frontend/backend pour une meilleure scalabilité
- Docker Compose

● Interlocuteur(s) du projet

- Développeur unique : Responsable de toutes les étapes du projet

● Contexte

A. Présentation du besoin

Le projet vise à combler le besoin d’un outil de veille automatisé et personnalisable, permettant aux utilisateurs de recevoir des informations en temps réel sur des sujets précis.

B. Analyse de l'existant

Les solutions existantes sont souvent limitées :

- Google Alerts : Peu de personnalisation et souvent des résultats non pertinents.
- Feedly : Abonnement requis pour des fonctionnalités avancées.
- Outils de scraping maison : Souvent complexes à mettre en place.

C. Finalité du projet

Créer un outil simple, efficace et personnalisable permettant aux utilisateurs de gérer leur veille de manière flexible, avec plusieurs canaux de diffusion et une interface moderne.

● Limites

A. Limites temporelles

- Projet développé en solo, avec une gestion du temps optimisée.
- Durée estimée : 4 à 6 mois (planification, développement, tests, mise en production).

B. Limites techniques

- Scraping dépendant des changements des sites sources.
- Limitations d'APIs tierces (quotas, restrictions d'usage).

C. Limites fonctionnelles

- Le projet ne couvre pas toutes les thématiques de veille, il évoluera selon les besoins utilisateurs.
- Pas d'application mobile dans un premier temps.

● Exigences fonctionnelles

A. Description globale

- Système d'authentification (inscription, connexion, gestion du compte).
- Choix des catégories de veille et gestion des préférences.
- Affichage des articles de manière claire et structurée.
- Mécanismes de notifications (email, Discord, API REST).

B. User Stories

En tant que..	Je veux..	Afin de..
Visiteur	Voir les différentes catégories de veille	de diversifier ma veille
Visiteur	indiquer mon mail dans une catégories intéressé	de recevoir des news journalière
Visiteur	accéder à une documentation de l'api	afin d'avoir les articles
Visiteur	lire le titre de l'article et la source	de savoir de quoi ça parle et de quel source elle vient
Visiteur	Créer un compte	personnaliser ma veille et recevoir des notifications.
Membre	pouvoir me connecter avec	accéder à mon tableau de

	mon email et mon mot de passe	bord et gérer mes préférences
Membre	pouvoir choisir les thématiques qui m'intéressent (cybersécurité, IA, business…),	recevoir uniquement les actualités pertinentes
Membre	voir une liste d'articles récents correspondant à mes catégories choisies	me tenir informé des dernières actualités
Membre	accéder à un tableau de bord	gérer mes préférences et voir mes alertes récentes
Membre	pouvoir choisir un mode sombre ou clair	personnaliser mon expérience visuelle
administrateur,	Pouvoir approuver ou bannir certaines sources d'articles	garantir la qualité des informations

● Exigences non fonctionnelles

Techniques :

- Scalabilité assurée via MongoDB Atlas et cache Redis.
- API optimisée avec pagination et filtres avancés.

Ergonomie :

- UI/UX moderne et responsive (compatible mobile/tablette/desktop).
- Mode sombre / clair disponible.

Volumétrie :

- Support d'une grande base de données d'articles.
- Optimisation des requêtes avec indexation (et cache peut être)

Sécurité :

- Authentification sécurisée (OAuth, JWT, bcrypt pour mots de passe).
- Protection contre les attaques XSS, CSRF, SQL Injection.

- Gestion des accès API via clé API et rate-limiting.

Développement durable :

- Code optimisé pour réduction de la consommation serveur.

• 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.
Perte de motivations	3/10	Prendre l'air.
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 : Vulnérabilités, failles de	2/10	A priori ça ne devrait pas poser de problèmes car on va s'en occuper dans

sécurité ou mauvaise gestion des données peuvent compromettre la sécurité de l'application.		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	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.

• Conclusion

Le projet CyberVeilleProject vise à proposer une solution moderne et automatisée de veille. Son développement est structuré, avec des contraintes techniques et des objectifs clairs, garantissant scalabilité, sécurité et efficacité.