Rapport de Projet : Plateforme Feedly (MVP)

1. Introduction et Vision

Le projet **Feedly** répond à la complexité croissante de l'exploitation manuelle des milliers d'avis utilisateurs sur le Google Play Store. L'objectif est de transformer ces retours non structurés en indicateurs décisionnels automatisés pour les équipes produit et techniques.

Objectifs du MVP

- **Business :** Réduire le temps d'identification des problèmes critiques et faciliter la prise de décision.
- **Techniques :** Automatiser la collecte (ETL) et l'analyse (NLP/IA) via une architecture asynchrone.
- **Philosophie :** Se concentrer sur une source unique (Google Play) avec des analyses simples mais fiables pour valider la valeur métier rapidement.

2. Périmètre Fonctionnel Détaillé

Le MVP s'articule autour de six lots fonctionnels majeurs :

- **Gestion des Applications :** Ajout par URL ou package name, avec récupération automatique des métadonnées (nom, icône, développeur).
- **Collecte (Scraping) :** Paramétrage de la période (30j à 6 mois) et du volume (100 à 5000 avis) pour un lancement asynchrone.
- **Moteur d'Analyse IA :** Classification des sentiments (Positif, Neutre, Négatif).
- Catégorisation thématique des avis négatifs (Bugs, Performance, UI/UX, Fonctionnalités manquantes).
- Calcul d'un score de priorité et extraction de mots-clés.
- **Dashboard de Visualisation :** KPIs (note moyenne, % sentiments), graphiques d'évolution et histogrammes des catégories.
- **Chatbot IA (RAG) :** Interface conversationnelle répondant à des questions types comme "Top problèmes de la semaine".
- **Intégration Jira :** Création semi-automatique de tickets depuis un avis critique avec remplissage automatique du titre et de la description.

3. Architecture Technique (MVP)

L'architecture est modulaire et découplée en quatre blocs principaux.

- **Frontend :** Next.js 16 (App Router), TypeScript, Tailwind CSS, et Recharts pour les visuels.

- **Backend API :** FastAPI (Python) pour ses performances et son intégration native avec l'IA. Authentification sécurisée via JWT.
- **Données & Traitement : Base de données :** PostgreSQL pour le stockage structuré (Utilisateurs, Apps, Avis, Analyses).
- **Asynchronisme :** Celery et Redis pour gérer le scraping et les appels IA sans bloquer l'interface.
- **Moteur IA :** API Google Gemini pour l'analyse de sentiment et la classification.

4. Plan de Réalisation et Planning

Le développement est estimé à **8 semaines** au total:

Phase	Durée	Livrables clés
1. Fondations	1 sem.	Authentification JWT, CRUD utilisateurs/apps.
2. Scraping	1 sem.	Pipeline ETL Google Play fonctionnel.
3. IA / NLP	2 sem.	Analyse de sentiment et classification thématique.
4. Dashboard	2 sem.	Visualisations graphiques et filtres avancés.
5. Chatbot	1 sem.	Interface de Q&A basée sur les données.
6. Intégrations	1 sem.	Connecteur Jira et export CSV.

5. Risques et Critères de Succès

Risques identifiés:

- Limitations ou blocages de l'API Google Play.
- Coûts et latence des appels API IA (Gemini).
- Complexité de l'intégration avec les outils tiers (Jira/Confluence).

Indicateurs de Succès (KPIs MVP):

- Analyse d'une application en moins de quelques minutes.
- Précision de l'analyse IA supérieure à **75%**.
- Adoption du chatbot pour obtenir des synthèses rapides.

6. Roadmap Future (Post-MVP)

Les phases suivantes incluront le support de l'App Store (Apple), l'analyse multilingue avancée, les notifications en temps réel (Slack/Teams) et la gestion multi-organisationnelle (SaaS).