Table des matières

I. Introduction Générale

1. Contexte et justification du projet

Avec la croissance rapide des applications mobiles, les plateformes de distribution comme Google Play Store génèrent des volumes très importants d'avis utilisateurs. Ces avis constituent une source précieuse d'information pour les équipes produit, techniques et métiers, mais leur exploitation manuelle devient rapidement complexe, chronophage et peu fiable lorsque le volume dépasse quelques centaines de commentaires.

Les équipes sont confrontées à plusieurs difficultés :

- analyser des milliers d'avis hétérogènes (langue, qualité, longueur),
- identifier rapidement les problèmes critiques (bugs, crashs, régressions),
- suivre l'évolution du sentiment utilisateur après chaque mise à jour,
- transformer des retours textuels non structurés en indicateurs décisionnels.

Les solutions existantes sur le marché sont souvent :

- coûteuses et peu accessibles pour les petites équipes,
- limitées en personnalisation des analyses,
- insuffisamment intégrées aux outils métiers (Jira, Confluence),
- peu flexibles pour une exploitation avancée de l'IA conversationnelle.

Dans ce contexte, le projet Feedly s'inscrit comme une opportunité produit et technique visant à proposer une plateforme intelligente, modulaire et scalable, capable de collecter, analyser et valoriser automatiquement les avis Google Play, tout en fournissant une interface moderne de visualisation et un assistant IA orienté prise de décision.

2. Objectifs globaux

Objectifs business

- Améliorer la compréhension des retours utilisateurs afin d'optimiser la qualité des applications mobiles.
- Réduire le temps nécessaire à l'identification des problèmes critiques.
- Centraliser l'analyse des avis dans un outil unique et exploitable par les équipes produit, techniques et management.
- Faciliter la prise de décision grâce à des indicateurs clairs et des rapports exploitables.

Objectifs techniques

- Mettre en place une architecture modulaire et évolutive permettant de traiter de grands volumes de données.
- Automatiser la collecte, le traitement et l'analyse des avis via une pipeline asynchrone.
- Exploiter des techniques d'intelligence artificielle et de NLP pour l'analyse de sentiment et la catégorisation thématique.
- Fournir une API sécurisée et performante pour l'interfaçage avec un frontend moderne et des services tiers.

Objectifs utilisateurs

- Offrir une interface intuitive permettant de visualiser rapidement l'état de santé d'une application.
- Permettre l'exploration fine des avis via des filtres et des graphiques interactifs.
- Mettre à disposition un chatbot conversationnel capable de répondre à des questions métier en langage naturel.
- Faciliter l'export et le partage des analyses sous forme de rapports.

3. Définitions et acronymes

- **MVP (Minimum Viable Product) :** Version initiale du produit intégrant uniquement les fonctionnalités essentielles permettant de valider la valeur du projet.
- **ETL (Extract, Transform, Load) :** Processus de collecte des données, de transformation (nettoyage, enrichissement) et de chargement dans une base de données.
- **NLP (Natural Language Processing) :** Ensemble de techniques permettant l'analyse et la compréhension automatique du langage naturel.
- **RAG (Retrieval-Augmented Generation) :** Approche combinant recherche d'informations dans une base de données et génération de réponses par un modèle de langage.
- **KPI (Key Performance Indicator) :** Indicateur clé permettant de mesurer la performance et l'évolution d'un système ou d'un produit.

II. Vision Produit et Stratégie MVP

1. Vision globale du produit

Feedly a pour vocation de devenir une **plateforme d'intelligence produit** centrée sur l'analyse des retours utilisateurs des applications mobiles.
Le produit doit permettre aux équipes de passer :

d'une lecture manuelle et réactive des avis → à une analyse automatisée, priorisée et actionnable.

La plateforme s'appuie sur trois piliers :

1. **Automatisation** de la collecte et du traitement des avis.
2. **Intelligence artificielle** pour transformer du texte non structuré en insights exploitables.
3. **Restitution claire** via dashboard, chatbot et intégrations métiers.

2. Philosophie MVP

Le **MVP (Minimum Viable Product)** a pour objectif principal de :

- valider la **valeur métier** du produit,
- démontrer la faisabilité technique,
- obtenir des retours utilisateurs rapides.

Le MVP se concentre volontairement sur :

- **une seule source** : Google Play Store,
- **un périmètre fonctionnel maîtrisé**,
- des analyses IA **simples mais fiables**,
- une interface claire orientée décision.

Note : Toute fonctionnalité non critique à la validation du produit est **reportée en phase future**.

3. Fonctionnalités incluses dans le MVP

a) Collecte des avis (MVP)

- Ajout d'une application Google Play via son identifiant ou son URL.
- Récupération automatique des avis avec :
  - texte,
  - note,
  - date,
  - version de l'application.
- Paramétrage simple :
  - période d'analyse,
  - volume d'avis.
- Lancement manuel de l'analyse depuis l'interface.

b) Analyse IA (MVP)

- Analyse de **sentiment** :
    - positif,
    - neutre,
    - négatif.
- Première **catégorisation thématique** des avis négatifs :
    - bugs,
    - performance,
    - UX/UI,
    - fonctionnalités.
- Calcul d'indicateurs globaux :
    - score moyen,
    - répartition des sentiments,
    - top problèmes détectés.

c) Interface utilisateur (MVP)

- Dashboard principal avec :
    - KPIs clés,
    - graphiques de répartition des sentiments,
    - évolution des notes.
- Page "Applications surveillées".
- Vue détaillée des avis avec filtres basiques.
- Chatbot IA :
    - questions simples sur les tendances et problèmes majeurs.
- Export des données au format CSV.

d) Architecture et technique (MVP)

- Backend API avec FastAPI.
- Pipeline asynchrone (scraping + analyse).
- Base de données centralisée.
- Authentification basique (JWT).

- Frontend Next.js avec composants réutilisables.

4. Fonctionnalités hors MVP (Roadmap future)

Ces fonctionnalités sont **volontairement exclues du MVP**, mais prévues à moyen / long terme.

I. Analyse avancée

- Analyse multilingue avancée.
- Détection automatique des régressions après mise à jour.
- Analyse comparative entre plusieurs applications.
- Score de criticité avancé basé sur l'impact utilisateur.

II. Chatbot & IA avancée

- RAG complet avec historique long terme.
- Génération automatique de rapports mensuels.
- Suggestions proactives (alertes intelligentes).
- Recommandations produit basées sur les tendances.

III. Intégrations métiers

- Intégration complète Jira :
    - création automatique de tickets,
    - synchronisation des statuts.
- Génération de pages Confluence.
- Notifications (email, Slack, Teams).

IV. Plateforme & SaaS

- Gestion multi-utilisateurs et rôles.
- Gestion d'abonnement et facturation.
- Monitoring avancé et observabilité.
- Scalabilité horizontale (multi-tenant).

5. Critères de succès du MVP

Le MVP sera considéré comme **réussi** si :

- un utilisateur peut analyser une application en moins de quelques minutes,
- les résultats sont compréhensibles sans expertise technique,
- les insights produits sont exploitables pour la prise de décision,
- la plateforme est stable et performante sur des volumes raisonnables.

III. Périmètre Fonctionnel Détaillé du MVP

1. Typologie des utilisateurs (MVP)

Dans le cadre du MVP, la plateforme distingue **un seul type d'utilisateur fonctionnel** afin de limiter la complexité.

Utilisateur standard (Product / Tech / Manager)

- Accès à toutes les fonctionnalités du MVP.
- Peut :
    - ajouter une application,
    - lancer une analyse,
    - consulter les résultats,
    - interagir avec le chatbot,
    - exporter les données.

Les rôles avancés (admin, viewer, billing) sont reportés en phase future.

2. Parcours utilisateur global (User Flow)

Flux principal du MVP :

1. Création de compte / connexion
2. Accès au dashboard
3. Ajout d'une application Google Play
4. Paramétrage de l'analyse
5. Lancement du scraping et de l'analyse
6. Consultation des résultats
7. Interaction avec le chatbot
8. Export des données

3. Authentification et accès (MVP)

a) Inscription / Connexion

- Inscription via :
    - email + mot de passe.
- Connexion sécurisée par JWT.
- Gestion de session côté frontend.

b) Sécurité minimale

- Hashage des mots de passe.
- Protection des routes API.
- Accès aux données limité à l'utilisateur connecté.

4. Gestion des applications surveillées

a) Ajouter une application

- Champs requis :
    - Nom ou URL Google Play.
- Vérification de validité de l'application.
- Enregistrement de l'application en base.

b) Liste des applications

- Affichage sous forme de cartes :
    - logo,
    - nom,
    - score global,
    - date de dernière analyse.
- Actions disponibles :
    - "Analyser maintenant",
    - "Voir détails".

5. Paramétrage et lancement de l'analyse

a) Paramètres d'analyse (MVP)

- Période :
    - 30 jours,

    - 3 mois,
    - 6 mois.
- Volume d’avis :
    - 100,
    - 1000,
    - 5000

b) Lancement de l’analyse

- Bouton “Lancer l’analyse”.
- Traitement asynchrone.
- Indicateur d’état :
    - en cours,
    - terminé,
    - erreur.

6. Consultation des résultats

a) Dashboard principal

- KPIs affichés :
    - score moyen,
    - % avis positifs / neutres / négatifs,
    - nombre total d’avis analysés.
- Visualisations :
    - graphique de répartition des sentiments,
    - évolution des notes dans le temps.
- Top problèmes détectés (Top 5).

b) Vue détaillée des avis

- Liste paginée des avis :
    - texte,
    - note,
    - date,
    - sentiment,

    - catégorie.
- Filtres disponibles :
    - sentiment,
    - période,
    - catégorie.
- Recherche textuelle simple.

7. Chatbot IA (MVP)

a) Objectif

Permettre à l'utilisateur d'obtenir rapidement des **insights synthétiques** sans parcourir manuellement les données.

b) Fonctionnalités

- Interface conversationnelle.
- Questions prédéfinies :
    - "Top problèmes détectés"
    - "Tendance générale des avis"
- Réponses basées sur les données analysées.
- Historique de conversation limité à la session.

8. Export des données

a) Formats disponibles

- CSV uniquement (MVP).

b) Contenu exporté

- Avis analysés.
- Méta-données associées.
- Résultats d'analyse (sentiment, catégorie).

9. Fonctionnalités explicitement exclues du MVP

- Intégration Jira / Confluence.
- Génération de rapports PDF.
- Notifications automatiques.

- Multi-utilisateurs et rôles.
- Analyse comparative multi-apps.
- Monitoring avancé.

IV. Architecture Fonctionnelle et Technique du MVP

1. Vue d'ensemble de l'architecture

L'architecture du MVP repose sur une **séparation claire des responsabilités**, afin de garantir :

- la maintenabilité,
- la scalabilité future,
- l'évolutivité vers des fonctionnalités avancées (Jira, Confluence, SaaS).

Composants principaux

1. **Frontend Web**
2. **Backend API**
3. **Pipeline de collecte et d'analyse**
4. **Moteur IA**
5. **Base de données**
6. **Services d'infrastructure**

2. Frontend (MVP)

a) Rôle

- Interface utilisateur principale.
- Visualisation des données et interaction avec le chatbot.
- Communication avec le backend via API REST.

b) Technologies retenues

- **Framework** : Next.js 16 (App Router)
- **Langage** : TypeScript
- **UI / Styling** :
    - Tailwind CSS
    - Shadcn UI
- **State management** : Zustand

- **Charts** : Recharts

c) Responsabilités

- Authentification utilisateur.
- Gestion des pages :
    - Login / Signup
    - Dashboard
    - Applications
    - Analyse détaillée
    - Chatbot
- Gestion des états :
    - chargement,
    - erreurs,
    - statuts d'analyse.

3. Backend API (MVP)

a) Rôle

- Point d'entrée central de la plateforme.
- Gestion de la logique métier.
- Exposition des données au frontend.
- Sécurisation des accès.

b) Technologie retenue

- **FastAPI (Python)**

Choix motivé par :

- performance,
- intégration native avec IA/NLP,
- support asynchrone,
- documentation automatique (OpenAPI).

c) Responsabilités

- Authentification (JWT).
- Gestion des utilisateurs.
- Gestion des applications surveillées.

- Lancement et suivi des analyses.
- Exposition des résultats analytiques.
- Interface avec le pipeline IA.

4. Pipeline de collecte et d'analyse

1. **Collecte (Extract)**

- Source : Google Play Store.
- Outil : google-play-scraper (Python).
- Fréquence :
    - déclenchée manuellement (MVP).

2. **Transformation (Transform)**

- Nettoyage du texte.
- Normalisation des champs.
- Détection langue (si nécessaire).

3. **Chargement (Load)**

- Stockage structuré en base de données.
- Association des avis à une application et un utilisateur.

5. Moteur IA (MVP)

a) Rôle

- Analyser les avis collectés.
- Enrichir les données stockées.

b) Fonctionnalités IA

- Analyse de sentiment :
    - positif,
    - neutre,
    - négatif.
- Classification thématique des avis négatifs.
- Extraction de mots-clés simples.

c) Technologies

- Google Generative AI (Gemini).

- Logique asynchrone (worker).

6. Base de données

a) Type

- **PostgreSQL**

b) Données stockées

- Utilisateurs.
- Applications.
- Avis bruts.
- Résultats d'analyse.
- Historique des analyses.

7. Gestion asynchrone et performances

a) Objectif

- Éviter le blocage des requêtes API.
- Permettre le traitement de volumes importants.

b) Technologies

- **Celery** (workers).
- **Redis** (broker).

8. Sécurité (MVP)

- Authentification JWT.
- Protection des endpoints.
- Séparation des données par utilisateur.
- Gestion des erreurs côté API.

9. Évolutivité prévue (post-MVP)

- Microservices dédiés (IA, intégrations).
- Ajout Jira / Confluence.
- Multi-tenancy.
- Facturation et abonnements.

- Notifications temps réel.

V. Modèle de Données (MVP)

1. Objectifs du modèle de données

Le modèle de données du MVP doit :

- Stocker les avis Google Play de manière structurée.
- Conserver l'historique des analyses IA.
- Permettre des filtres performants (date, sentiment, catégorie, version).
- Supporter la gestion multi-applications et multi-utilisateurs.
- Être évolutif vers des fonctionnalités futures (Jira, Confluence, SaaS).

2. Entités principales

1. Utilisateur (User)

Description
Représente un utilisateur de la plateforme.

Champs clés

- id
- email
- password_hash
- full_name
- role (admin / user)
- created_at
- updated_at

Relations

- Un utilisateur peut surveiller plusieurs applications.
- Un utilisateur peut lancer plusieurs analyses.

2. Application surveillée (Application)

Description
Application Google Play suivie par un utilisateur.

Champs clés

- id
- package_name (ex: com.whatsapp)
- app_name
- developer
- icon_url
- created_at
- last_analysis_at

Relations

- Une application appartient à un utilisateur.
- Une application possède plusieurs avis.

c) Avis (Review)

Description
Avis brut collecté depuis Google Play.

Champs clés

- id
- application_id
- author_name
- rating (1 à 5)
- content
- language
- review_date
- app_version

Relations

- Un avis appartient à une application.
- Un avis possède une analyse IA.

d) Analyse IA (ReviewAnalysis)

Description
Résultat de l'analyse IA d'un avis.

Champs clés

- id

- review_id
- sentiment (positive / neutral / negative)
- category (bug, performance, ux, feature_request…)
- priority_score
- keywords
- analyzed_at

Relations

- Une analyse est liée à un seul avis.

e) Analyse globale (AnalysisRun)

Description
Représente une session d'analyse déclenchée par l'utilisateur.

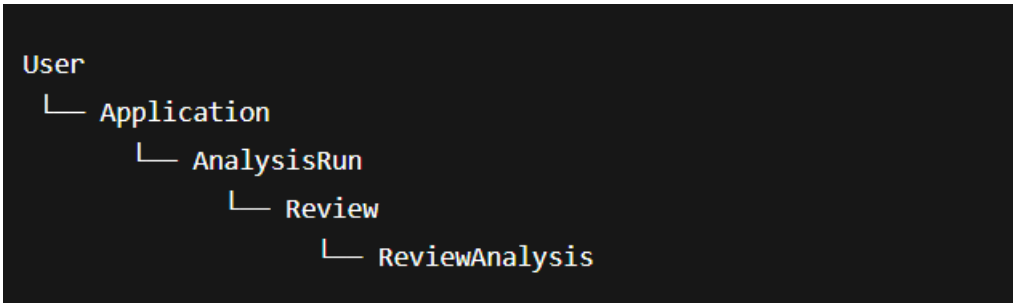
Champs clés

- id
- application_id
- period_start
- period_end
- review_limit
- status (pending / running / completed / failed)
- created_at
- completed_at

Relations

- Une analyse globale est liée à une application.
- Une analyse globale produit plusieurs avis analysés.

3. Relations (vue synthétique)



4. Indexation et performances (MVP)

Pour garantir de bonnes performances :

- Index sur :
    - application_id
    - review_date
    - sentiment
    - category
- Pagination obligatoire sur les listes d'avis.
- Agrégations calculées côté backend (KPIs).

5. Évolutions futures prévues

- Table JiraTicket
- Table ConfluenceDocument
- Historique des alertes critiques
- Multi-organisation / équipes
- Logs d'audit

VI. Fonctionnalités du MVP (Spécifications détaillées)

Cette section décrit **exactement ce qui doit être implémenté pour le MVP**, ni plus, ni moins.
Chaque fonctionnalité est exprimée sous forme de **User Stories** avec des **critères d'acceptation**.

1. Authentification & Gestion des utilisateurs

Inscription / Connexion

User Story

En tant qu'utilisateur, je veux créer un compte et me connecter afin d'accéder à mon espace privé.

Fonctionnalités

- Inscription par email + mot de passe
- Connexion sécurisée (JWT)
- Déconnexion
- Protection des routes privées

Critères d'acceptation

- Un utilisateur non authentifié ne peut pas accéder au dashboard
- Le token JWT est stocké de manière sécurisée
- Les erreurs d'authentification sont clairement affichées

2. Gestion des applications surveillées

a) Ajouter une application Google Play

User Story

En tant qu'utilisateur, je veux ajouter une application Google Play afin d'analyser ses avis.

Fonctionnalités

- Ajout par :
    - URL Google Play
    - ou Package name (com.example.app)
- Récupération automatique des métadonnées (nom, icône, développeur)

Critères d'acceptation

- L'application est visible dans la liste après ajout
- Une application ne peut pas être ajoutée deux fois
- Validation des entrées utilisateur

b) Liste des applications surveillées

User Story

En tant qu'utilisateur, je veux voir toutes mes applications surveillées pour suivre leur état.

Fonctionnalités

- Liste des applications
- Dernière date d'analyse
- Score global
- Boutons d'action :
    - "Analyser maintenant"
    - "Voir détails"

Critères d'acceptation

- Chargement rapide (<2s)

- Données cohérentes avec la dernière analyse

3. Collecte des avis (Scraping)

1. Paramétrage de la collecte

User Story

En tant qu'utilisateur, je veux configurer la période et le volume d'avis afin de contrôler l'analyse.

Fonctionnalités

- Sélecteur de période :
    - 30 jours
    - 3 mois
    - 6 mois
    - personnalisé
- Volume :
    - 100 / 1000 / 5000 / illimité

Critères d'acceptation

- Les paramètres sont pris en compte par le pipeline
- Une analyse peut être relancée avec d'autres paramètres

b) Lancement de l'analyse

User Story

En tant qu'utilisateur, je veux lancer une analyse afin de récupérer et analyser les avis.

Fonctionnalités

- Lancement asynchrone
- Statut visible :
    - En attente
    - En cours
    - Terminé
    - Erreur
- Logs internes (non visibles utilisateur)

Critères d'acceptation

- L'UI reste réactive

- Les erreurs sont gérées proprement

4. Analyse IA / NLP

a) Analyse de sentiment

User Story

En tant qu'utilisateur, je veux connaître le sentiment global des avis afin d'évaluer la satisfaction.

Fonctionnalités

- Classification :
    - Positif
    - Neutre
    - Négatif
- Calcul des pourcentages globaux

Critères d'acceptation

- Chaque avis possède un sentiment
- Les KPIs correspondent aux données stockées

b) Classification thématique

User Story

En tant qu'utilisateur, je veux comprendre les causes principales des avis négatifs.

Fonctionnalités

- Catégories principales :
    - Bugs
    - Performance
    - UI/UX
    - Fonctionnalités manquantes
- Extraction de mots-clés
- Score de priorité

Critères d'acceptation

- Les avis négatifs sont catégorisés
- Les catégories sont filtrables dans l'UI

5. Dashboard & Visualisation

a) Dashboard principal

User Story

En tant qu'utilisateur, je veux un dashboard synthétique pour prendre des décisions rapidement.

Fonctionnalités

- KPIs :
    - Note moyenne
    - % sentiments
    - Nombre d'avis
    - Top problèmes
- Graphiques :
    - Camembert sentiment
    - Courbe évolution notes
    - Histogramme catégories

Critères d'acceptation

- Les graphiques se mettent à jour après analyse
- Temps de chargement raisonnable

6. Consultation des avis détaillés

User Story

En tant qu'utilisateur, je veux consulter les avis bruts et analysés pour approfondir l'analyse.

Fonctionnalités

- Liste paginée
- Champs affichés :
    - Texte
    - Note
    - Date
    - Version
    - Sentiment
    - Catégorie
- Filtres :

    - Sentiment
    - Catégorie
    - Période
    - Version

Critères d'acceptation

- Pagination fonctionnelle
- Filtres combinables

7. Chatbot IA (RAG – MVP)

User Story

En tant qu'utilisateur, je veux poser des questions en langage naturel pour obtenir des insights rapides.

Fonctionnalités

- Interface conversationnelle
- Questions types :
    - "Top problèmes cette semaine"
    - "Avis négatifs sur la dernière version"
- Réponses basées sur les données collectées

Critères d'acceptation

- Réponses cohérentes avec la base de données
- Temps de réponse acceptable

8. Export & Reporting

User Story

En tant qu'utilisateur, je veux exporter les données pour les partager.

Fonctionnalités

- Export CSV (MVP)
- Export PDF (optionnel MVP+)

Critères d'acceptation

- Les fichiers exportés sont exploitables
- Respect des filtres actifs

VII. Architecture Technique du MVP

Cette section décrit l'architecture cible du MVP, les choix technologiques retenus et les flux entre les différents composants.

1. Vue d'ensemble de l'architecture

Le MVP repose sur une **architecture modulaire orientée services**, permettant une montée en charge progressive et une évolution vers des fonctionnalités avancées.

Composants principaux :

1. Frontend Web (Next.js)
2. Backend API (FastAPI)
3. Pipeline de collecte & traitement (ETL)
4. Moteur IA / NLP
5. Base de données
6. Services tiers (Google Play, Jira, Confluence)

Principe clé :

- Le frontend ne communique **que** avec l'API backend
- Les traitements lourds sont exécutés **asynchroniquement**
- L'IA est découplée du flux utilisateur

2. Architecture Frontend (MVP)

Technologies

- Next.js 16 (App Router)
- React 19
- Tailwind CSS + shadcn/ui
- Zustand (state management)
- Recharts (visualisation)

Responsabilités

- Authentification utilisateur
- Navigation et routing
- Visualisation des KPIs et graphiques
- Gestion des formulaires (apps, paramètres)

- Interface chatbot
- Gestion des thèmes (light/dark)

Contraintes

- Aucun accès direct à la base de données
- Toutes les données transitent par l'API backend
- Composants découplés et réutilisables

3. Architecture Backend (MVP)

Technologies

- FastAPI (Python)
- SQLAlchemy (ORM)
- PostgreSQL
- JWT pour l'authentification
- Celery + Redis pour les tâches asynchrones

Responsabilités

- Exposition des APIs REST
- Gestion des utilisateurs et permissions
- Orchestration du pipeline de scraping
- Stockage et agrégation des données
- Sécurité et contrôle d'accès
- Intégration Jira / Confluence (MVP partiel)

Exemples d'APIs

- POST /auth/login
- POST /apps
- POST /apps/{id}/analyze
- GET /apps/{id}/dashboard
- GET /reviews
- POST /chat/query
- POST /jira/ticket

4. Pipeline ETL (Collecte & Traitement)

Étapes du pipeline

1. **Extract**
    - Scraping Google Play Store
    - Récupération des avis et métadonnées
2. **Transform**
    - Nettoyage des textes
    - Normalisation des champs
3. **Load**
    - Insertion en base de données
    - Marquage des avis “non analysés”

Caractéristiques

- Asynchrone (Celery)
- Tolérant aux erreurs
- Rejouable (re-run possible)

5. Moteur IA / NLP

Fonctionnement

- Service indépendant ou worker
- Polling ou déclenchement par événement
- Traitement batch des avis non analysés

Fonctionnalités MVP

- Analyse de sentiment
- Classification thématique
- Extraction de mots-clés
- Score de criticité

Technologies

- Google Gemini API
- (Optionnel) HuggingFace pour modèles locaux
- LangChain (préparation RAG)

6. Base de données

Technologie

- PostgreSQL

Principales entités

- Users
- Applications
- Reviews
- Analyses
- Categories
- Tickets (Jira)
- ChatLogs

Contraintes

- Historisation complète
- Aucune suppression définitive en MVP
- Indexation pour performance (reviews, dates)

7. Intégration Jira & Confluence (MVP)

Jira

- Création automatique de tickets pour avis critiques
- Mapping :
    - Titre ← catégorie + résumé
    - Description ← avis + analyse IA
    - Priorité ← score de criticité

Confluence

- Génération de pages :
    - Rapport hebdomadaire
    - Synthèse des tendances
- Historisation par application

8. Sécurité & conformité

Sécurité

- JWT + refresh tokens
- Validation des entrées (Pydantic)
- Protection des endpoints sensibles
- Rate limiting (basique MVP)

Conformité

- Pas de données personnelles sensibles
- Anonymisation des avis
- Respect des CGU Google Play

9. Scalabilité & évolutivité

Prévu dès le MVP

- Services découplés
- Workers IA indépendants
- Scaling horizontal du scraping

Préparé pour le futur

- Passage microservices
- Event-driven architecture
- Cache Redis pour analytics

VIII. Périmètre du MVP et Fonctionnalités Futures (Roadmap)

1. Définition du MVP

Le **MVP (Minimum Viable Product)** a pour objectif de :

- Valider la **valeur produit**
- Tester l'adoption utilisateur
- Confirmer la **faisabilité technique**
- Préparer l'industrialisation

Note : Le MVP se concentre sur **une seule source (Google Play)**, **un nombre limité d'applications**, et **des analyses IA ciblées**.

2. Fonctionnalités incluses dans le MVP

a) Collecte & Données

- Scraping Google Play Store
- Collecte des avis avec métadonnées :
    - Texte
    - Note
    - Date
    - Version de l'application
- Planification de collecte manuelle ou périodique

b) Analyse IA / NLP

- Analyse de sentiment (Positif / Neutre / Négatif)
- Classification thématique des avis négatifs :
    - Bugs
    - Performance
    - UX/UI
    - Fonctionnalités
- Extraction de mots-clés
- Calcul d'un score de criticité

c) Dashboard (Frontend)

- Vue globale par application
- KPIs principaux :
    - Note moyenne
    - Répartition des sentiments
    - Nombre d'avis analysés
    - Top problèmes détectés
- Graphiques :
    - Évolution des notes
    - Répartition sentimentale

    - Histogramme des catégories
- Filtres :
    - Date
    - Sentiment
    - Catégorie
    - Version app

d) Chatbot IA (MVP)

- Interface conversationnelle
- Questions préconfigurées :
    - “Top problèmes de la semaine”
    - “Évolution globale”
- Réponses générées à partir des données analysées
- Historique de conversation simple

e) Intégration Jira (MVP)

- Création manuelle ou semi-automatique de tickets
- Liaison avis → ticket Jira
- Définition de priorité basée sur criticité
- Suivi du statut du ticket

f) Gestion utilisateur

- Authentification (login / signup)
- Gestion de profil basique
- Gestion multi-applications (limité)

3. Fonctionnalités explicitement exclues du MVP

Ces fonctionnalités sont **hors périmètre MVP** :

- Support App Store (Apple)
- Analyse multilingue avancée
- IA explicative avancée (why / root cause)

- Auto-clustering automatique sans règles
- Recommandations produit automatisées
- Billing / facturation
- Rôles utilisateurs avancés
- Monitoring temps réel
- Notifications temps réel (email, Slack)
- Custom dashboards

4. Fonctionnalités prévues post-MVP (Phase 2 & 3)

Phase 2 – Amélioration Produit

- Analyse multilingue
- Détection de tendances temporelles
- Alertes automatiques
- Chatbot RAG avancé
- Création automatique de tickets Jira
- Templates de rapports Confluence

Phase 3 – Industrialisation SaaS

- Support multi-sources (App Store, réseaux sociaux)
- Facturation & abonnements
- RBAC (roles & permissions)
- SLA & monitoring avancé
- API publique
- Marketplace d'intégrations

5. Critères de succès du MVP

Le MVP est considéré comme réussi si :

- Les avis sont collectés sans interruption
- L'analyse IA atteint une précision acceptable (>75%)
- Le dashboard est exploitable pour la décision
- Les tickets Jira permettent une action concrète

- Les utilisateurs comprennent et utilisent le chatbot

IX. Exigences Fonctionnelles Détaillées (User Stories)

Cette section décrit les fonctionnalités du MVP sous forme de **user stories**, afin de :

- Clarifier les besoins métiers
- Faciliter la planification Agile (backlog, sprints)
- Servir de base aux tests fonctionnels

1. Gestion des utilisateurs

US-01 — Inscription utilisateur

En tant que nouvel utilisateur
Je veux créer un compte avec mon email et un mot de passe
Afin de accéder à la plateforme Feedly

Critères d'acceptation

- Formulaire signup avec validation
- Mot de passe sécurisé
- Message de confirmation
- Redirection vers le dashboard

US-02 — Authentification

En tant que utilisateur
Je veux me connecter à mon compte
Afin de accéder à mes applications et analyses

Critères d'acceptation

- Login email / mot de passe
- Gestion des erreurs (identifiants invalides)
- Session sécurisée (JWT)

2. Gestion des applications Google Play

US-03 — Ajouter une application

En tant que utilisateur
Je veux ajouter une application Google Play à surveiller
Afin de analyser les avis utilisateurs

Critères d'acceptation

- Champ URL ou package name
- Vérification de validité
- Lancement du scraping
- Indication d'état (en cours / terminé / erreur)

US-04 — Visualiser les applications surveillées

En tant que utilisateur
Je veux voir la liste de mes applications
Afin de suivre leur état d'analyse

Critères d'acceptation

- Liste des apps ajoutées
- Score global par app
- Date de dernière analyse
- Bouton "Analyser maintenant"

3. Collecte et pipeline de données

US-05 — Lancer la collecte des avis

En tant que utilisateur
Je veux lancer une collecte d'avis
Afin de analyser une période donnée

Critères d'acceptation

- Choix de période (30j, 3 mois, 6 mois)
- Choix du volume
- Lancement asynchrone
- Suivi de progression

US-06 — Nettoyage et stockage

En tant que système
Je veux nettoyer et stocker les avis
Afin de garantir des données exploitables

Critères d'acceptation

- Suppression des doublons
- Normalisation du texte
- Stockage en base de données

4. Analyse IA / NLP

US-07 — Analyse de sentiment

En tant que utilisateur
Je veux connaître le sentiment global des avis
Afin de mesurer la satisfaction utilisateur

Critères d'acceptation

- Classification Positif / Neutre / Négatif
- Score agrégé
- Visualisation graphique

US-08 — Classification thématique

En tant que utilisateur
Je veux identifier les types de problèmes récurrents
Afin de prioriser les corrections

Critères d'acceptation

- Catégories automatiques
- Filtres par thème
- Liste des avis associés

5. Dashboard & visualisation

US-09 — Consulter le dashboard principal

En tant que utilisateur
Je veux visualiser les KPIs clés
Afin de prendre des décisions rapidement

Critères d'acceptation

- KPIs visibles en haut de page
- Graphiques interactifs
- Filtres globaux

US-10 — Analyse détaillée des avis

En tant que utilisateur
Je veux consulter les avis individuellement
Afin de comprendre les retours précis

Critères d'acceptation

- Liste paginée
- Filtres avancés
- Accès aux métadonnées

6. Chatbot IA (MVP)

US-11 — Interroger les données via le chatbot

En tant que utilisateur
Je veux poser des questions en langage naturel
Afin de obtenir des synthèses rapides

Critères d'acceptation

- Interface conversationnelle
- Réponses basées sur les données collectées
- Questions préconfigurées disponibles

7. Intégration Jira

US-12 — Créer un ticket Jira

En tant que utilisateur
Je veux créer un ticket Jira depuis un avis critique
Afin de assurer le suivi des problèmes

Critères d'acceptation

- Bouton "Créer ticket"
- Remplissage automatique (titre, description)
- Définition de la priorité
- Confirmation de création

Export & reporting

US-13 — Exporter les données

En tant que utilisateur
Je veux exporter les analyses
Afin de les partager avec mon équipe

Critères d'acceptation

- Export CSV
- Export PDF (basique)
- Données filtrées respectées

X. Exigences Non Fonctionnelles

Cette section définit les **critères de qualité** du MVP. Elle est essentielle pour garantir une plateforme **fiable, sécurisée et évolutive**, même si certaines optimisations avancées seront prévues pour les phases futures.

1. Sécurité

1. **Authentification & Autorisation**

- Authentification basée sur **JWT**
- Expiration et renouvellement des tokens
- Séparation des rôles (MVP) :
    - Utilisateur standard
    - Administrateur
- Accès aux données limité à l'utilisateur propriétaire (multi-tenant)

2. **Protection des données**

- Chiffrement des mots de passe (bcrypt / argon2)
- Variables sensibles stockées dans des fichiers .env
- Aucune clé API exposée côté frontend
- Connexions sécurisées (HTTPS recommandé)

3. **Sécurité applicative**

- Protection contre :
    - Injection SQL
    - XSS
    - CSRF
- Validation systématique des entrées utilisateur

- Journalisation des erreurs critiques

2. Performance

- Temps de réponse API (objectif MVP) :
    - ≤ 500 ms pour 90 % des requêtes
- Traitements lourds exécutés **asynchroniquement** (scraping, IA)
- Pagination obligatoire pour :
    - Avis utilisateurs
    - Logs
    - Historique d'analyses
- Mise en cache possible (Redis) pour :
    - KPIs
    - Résumés IA

3. Scalabilité

- Architecture **modulaire et découplée**
- Possibilité d'ajouter :
    - De nouveaux workers Celery
    - De nouvelles sources de données (post-MVP)
- Support du **multi-applications** par utilisateur
- Séparation claire :
    - API Backend
    - Pipeline de données
    - Moteur IA

4. Disponibilité & Fiabilité

- Tolérance aux erreurs de scraping (retry, fallback)
- Gestion des échecs IA sans bloquer le pipeline global
- Logs applicatifs centralisés
- Monitoring basique de l'état :
    - Scraping

    - Analyse IA
    - API

5. Maintenabilité & Qualité du code

- Architecture en couches :
    - API
    - Services métier
    - Accès données
- Respect des principes :
    - SOLID
    - DRY
- Documentation minimale :
    - README technique
    - Commentaires sur les modules critiques
- Convention de nommage homogène

6. Compatibilité & Accessibilité

- Compatibilité navigateurs modernes :
    - Chrome
    - Firefox
    - Edge
- Interface responsive (desktop en priorité MVP)
- Accessibilité basique :
    - Lisibilité des contrastes
    - Navigation clavier partielle

7. Conformité & Légalité (MVP)

- Respect des conditions d'utilisation publiques de Google Play
- Données collectées uniquement à des fins d'analyse
- Pas de stockage de données personnelles sensibles
- Mentions légales et politique de confidentialité (version simplifiée)

XI. Architecture Technique & Choix Technologiques (MVP)

1. Vue d'ensemble de l'architecture

Le MVP de **Feedly** repose sur une architecture **modulaire, découplée et scalable**, permettant :

- une montée en charge progressive,
- une évolution fonctionnelle sans refonte majeure,
- une séparation claire des responsabilités (scraping, IA, API, UI).

L'architecture est organisée autour de **4 blocs principaux** :

1. **Frontend Web**
2. **Backend API**
3. **Pipeline de collecte et d'analyse**
4. **Services externes (Google Play, IA, Jira, Confluence)**

2. Frontend (MVP)

Rôle :

- Interface utilisateur principale
- Visualisation des données
- Interaction avec le chatbot IA
- Paramétrage des analyses

Choix technologiques :

- **Framework** : Next.js (App Router)
- **UI** : Tailwind CSS + Shadcn UI
- **State management** : Zustand
- **Charts** : Recharts
- **Auth côté client** : JWT (via cookies sécurisés)

Pages clés (MVP) :

- Landing page (public)
- Login / Signup
- Dashboard principal
- Applications surveillées

- Vue analyse détaillée
- Chatbot IA
- Paramètres utilisateur

3. Backend API (MVP)

Rôle :

- Exposer les données au frontend
- Gérer l'authentification et les utilisateurs
- Orchestrer le pipeline scraping / IA
- Centraliser la logique métier

Choix technologiques :

- **Framework** : FastAPI (Python)
- **Auth** : JWT (python-jose)
- **ORM** : SQLAlchemy
- **Validation** : Pydantic
- **Base de données** : PostgreSQL

Principaux endpoints :

- Authentification (login, signup)
- Gestion des applications
- Lancement d'analyses
- Consultation des avis
- KPIs & statistiques
- Chatbot (RAG)

4. Pipeline Scraping & Traitement IA

Scraping :

- Récupération des avis Google Play
- Métadonnées : note, date, version, langue, localisation
- Fréquence configurable

Analyse IA :

- Détection du sentiment

- Classification thématique
- Scoring de criticité
- Stockage des résultats enrichis en base

Choix technologiques :

- Python
- google-play-scraper
- Google Gemini (via API)
- Celery + Redis (traitement asynchrone)

5. Chatbot IA (RAG – MVP)

Fonctionnement :

- Questions en langage naturel
- Recherche dans la base de données (avis + analyses)
- Génération de réponses synthétiques

Technos :

- Gemini API
- Vectorisation (phase MVP simple : SQL + filtres)
- Évolution future vers base vectorielle dédiée

6. Intégrations Externes (MVP)

- **Google Play Store** : source des données
- **Jira** : création automatique de tickets pour bugs critiques
- **Confluence** : génération de rapports et documentation
- **Exports** : CSV / PDF

7. Séparation MVP / Évolutions futures

Élément	MVP	Post-MVP
Scraping	Google Play uniquement	Multi-stores
IA	Sentiment + catégories	Résumés, prédictions
Chatbot	Q&A simple	Agents multi-tâches
Jira	Création tickets	Workflow avancé
Confluence	Rapports simples	Documentation intelligente

XII. Plan de Réalisation du MVP

Cette section décrit **l'approche méthodologique**, le **découpage des tâches** et les **jalons** nécessaires à la mise en œuvre du MVP de Feedly.

1. Méthodologie de développement

Le projet adopte une approche **itérative et incrémentale**, inspirée de **l'Agile / Scrum** :

- Découpage en **sprints courts** (1 à 2 semaines)
- Livraison continue de fonctionnalités utilisables
- Validation rapide avec des utilisateurs pilotes
- Priorité donnée à la **valeur métier**

2. Découpage en lots fonctionnels

Lot 1 – Fondations techniques

Objectif : mettre en place une base stable.

- Initialisation des dépôts (frontend / backend)
- Configuration CI/CD de base
- Mise en place PostgreSQL & Redis
- Authentification (JWT)
- Structure globale API / Frontend

✅ **Livrable** : plateforme accessible avec login fonctionnel

Lot 2 – Collecte des avis (Scraping)

- Intégration Google Play Scraper
- Modèle de données avis / applications
- Lancement manuel de collecte
- Gestion des erreurs et doublons

✅ **Livrable** : avis stockés en base

Lot 3 – Analyse IA (Sentiment & Catégories)

- Intégration Gemini
- Pipeline de traitement asynchrone
- Stockage des résultats IA
- Score de criticité

✅ **Livrable** : avis enrichis automatiquement

Lot 4 – Dashboard & Visualisations

- KPIs principaux
- Graphiques (sentiment, notes, catégories)
- Filtres (date, version, sentiment)

✅ **Livrable** : dashboard exploitable

Lot 5 – Chatbot IA (RAG MVP)

- Interface conversationnelle
- Requêtes simples sur données analysées
- Réponses synthétiques

✅ **Livrable** : chatbot fonctionnel

Lot 6 – Intégration Jira / Confluence

- Création automatique de tickets Jira
- Génération de rapports Confluence
- Paramétrage des seuils

✅ **Livrable** : boucle actionnable complète

3. Planning indicatif (MVP)

Phase	Durée estimée
Fondations	1 semaine
Scraping	1 semaine
IA / NLP	2 semaines
Dashboard	2 semaines
Chatbot	1 semaine
Jira / Confluence	1 semaine
Total MVP	**8 semaines**

4. Dépendances et risques

Risques identifiés :

- Limitation API Google Play
- Coût et latence des appels IA
- Qualité variable des avis
- Complexité intégration Jira/Confluence

Mesures :

- Batch processing
- Caching
- Feature flags
- Monitoring actif

XIII. Critères de Validation & Indicateurs de Succès (MVP)

- Avis correctement collectés et stockés
- Analyse IA cohérente (sentiment + catégories)
- Dashboard fonctionnel avec KPIs et filtres
- Chatbot répond aux questions simples

- Tickets Jira et rapports Confluence créés automatiquement

XIV. Plan de Déploiement & Production

- Conteneurisation (Docker)
- CI/CD GitHub Actions
- Monitoring (Prometheus/Grafana)
- Sauvegarde base de données et logs
- Environnement dev / staging / prod

XV. Maintenance & Evolution

- Surveillance pipeline Scraping et IA
- Corrections bugs et mises à jour
- Ajout fonctionnalités futures (multi-langue, export amélioré, notifications)

XVI. Annexes

- Schémas architecture
- Modèles données
- Diagrammes UX/UI
- Liste API et clés nécessaires