#### **Tableau des Interacteurs**

| **Interacteur** | **Sections concernées** | **Description** |
| --- | --- | --- |
| Stakeholders et décideurs | Vue d’ensemble du projet | Vision globale et objectifs stratégiques. |
| Développeurs | Guide d’installation, Normes de codage | Instructions techniques pour la mise en place et le développement. |
| Développeurs intégrateurs | Documentation API et Intégration | Endpoints et méthodes pour intégrer Finna avec d'autres systèmes. |
| Data Engineers | Guide du scraping et collecte des données | Collecte et traitement des données via des scripts automatisés. |
| Utilisateurs finaux | Guide utilisateur pour l’extension | Installation et utilisation de l’extension Finna. |
| Développeurs QA | Tests et assurance qualité | Procédures de tests pour garantir la fiabilité et la robustesse. |
| DevOps | CI/CD et automatisation | Configuration des pipelines de déploiement et gestion des services Docker. |
| Équipe marketing | Stratégie marketing et engagement | Stratégies de communication et implication des utilisateurs. |
| Administrateurs de bases de données | Documentation base de données et gestion des données | Structure, configuration et maintenance de la base de données PostgreSQL. |

#### 

#### **1. Vue d’ensemble du projet (Pour les stakeholders et décideurs)**

Finna vise à résoudre le problème croissant de désinformation en ligne. En France, environ 30 % des citoyens ont des difficultés à identifier les fausses informations. Finna répond à ce besoin avec une extension web qui simplifie la vérification des informations.

Cette extension permet aux utilisateurs de soumettre des textes, comme des extraits d’articles ou des tweets, afin de vérifier leur authenticité. Grâce à une IA, Finna génère un résumé des informations trouvées, accompagnées de leurs sources fiables.

Le projet s’appuie sur des technologies modernes : React pour le frontend, Node.js pour le backend, PostgreSQL pour la gestion des données, et des scripts Python pour le scraping des sources. Une roadmap bien définie assure le développement structuré du produit. Parmi les étapes clés figurent la création d’une interface utilisateur intuitive et le développement d’un algorithme de recherche performant.

#### **2. Guide d’installation et de configuration (Pour les développeurs)**

L’installation de Finna suit un processus clair et rapide pour permettre aux développeurs de lancer le projet en local ou sur un serveur.

**Prérequis** :

* Node.js v16+
* PostgreSQL v8+
* Python v3.8+
* Git

**Étapes détaillées** :

1. Clonez le dépôt Git :  
    git clone <https://github.com/eipFinna/FINNA_MIRROR.git>  
   cd finna
2. Installez les dépendances :  
    cd backend && npm install  
   cd ../frontend && npm install
3. Configurez la base de données :
   * Modifiez le fichier .env avec vos identifiants PostgreSQL.
   * Initialiser les tables via Docker Compose et le script dbInit.sql.
4. Lancez l’application :
   * Backend : npm run dev
   * Frontend : npm start
5. Lancez le script de scraping :  
    python scripts/scraper.py

#### **3. Documentation API et Intégration (Pour les développeurs intégrateurs)**

L’API de Finna permet une communication fluide entre les différents composants du projet et les applications tierces.

**Endpoints principaux** :

* **POST /verify** : Soumet un texte à analyser. Retourne un résumé et les sources associées.
* **GET /sources** : Récupère toutes les sources disponibles dans la base de données.

**Exemple d’appel API** :

curl -X POST -H "Content-Type: application/json" -d '{"text": "exemple"}' http://localhost:3000/verify

Les codes de réponse HTTP :

* 200 : Succès
* 404 : Aucune source trouvée
* 500 : Erreur interne

#### **4. Guide du scraping et collecte des données (Pour les data engineers)**

Les scripts de scraping collectent automatiquement des articles et des informations provenant de diverses sources fiables. Ces données alimentent la base de données utilisée par Finna.

**Étapes pour configurer le scraping** :

1. Installez les dépendances nécessaires :  
    pip install -r requirements.txt
2. Configurez les sources à scraper dans config.yaml.
3. Lancez le script :  
    python scraper.py
4. Vérifiez les logs pour identifier et résoudre les erreurs potentielles. Les journaux détaillent les articles récupérés et les échecs éventuels.

#### **5. Guide utilisateur pour l’extension (Pour les utilisateurs finaux)**

Finna propose une interface simple et intuitive qui s’intègre directement dans le navigateur des utilisateurs. Voici comment en tirer le meilleur parti :

**Installation** :

* Rendez-vous sur le site officiel de Finna.
* Téléchargez et ajoutez l’extension à votre navigateur via les paramètres.

**Utilisation** :

1. Saisissez ou copiez un texte dans la barre de recherche de l’extension.
2. Cliquez sur "Vérifier" pour lancer l’analyse.
3. Consultez les résultats affichés :
   * Sources fiables liées au texte.
   * Résumé généré par l’IA.
   * Message explicatif en cas d’absence de sources.

#### **6. Normes de codage et bonnes pratiques (Pour les développeurs)**

Pour maintenir un code de haute qualité et facilement maintenable :

* Respectez les conventions :
  + JavaScript : Utilisez const et let, limitez les variables globales.
  + Python : Suivez PEP 8, documentez vos fonctions avec des docstrings.
* Utilisez des outils d’analyse statique : ESLint pour JavaScript, flake8 pour Python.
* Suivez des workflows Git structurés :
  + Nommez vos branches selon leur fonction (feature/nouvelle-fonctionnalité).
  + Rédigez des messages de commit clairs et descriptifs.

#### **7. Tests et assurance qualité (Pour les développeurs QA)**

Les tests sont essentiels pour garantir la robustesse du système. Finna utilise des outils modernes pour assurer une couverture complète.

**Types de tests** :

* Tests unitaires : Vérifient les fonctions isolées.
* Tests d’intégration : Valident les interactions entre les modules.

**Outils utilisés** :

* Backend : Jest
* Frontend : React Testing Library

**Exécution des tests** :

npm test

#### **8. CI/CD et automatisation (Pour les DevOps)**

Pour assurer des déploiements rapides et fiables, Finna utilise des pipelines CI/CD configurés avec Git et Docker Compose.

**Étapes principales** :

1. Chaque commit déclenche l’exécution automatique des tests unitaires.
2. Les pipelines gèrent la livraison vers un environnement de staging.
3. Docker Compose orchestre les services nécessaires : backend, base de données, etc.

#### **9. Stratégie marketing et engagement (Pour l’équipe marketing)**

La stratégie marketing de Finna repose sur une communication claire et engageante :

* Publiez régulièrement sur Twitter/X et Instagram pour éduquer les utilisateurs sur la vérification des informations.
* Organisez des sessions de feedback via Discord.
* Montrez l’efficacité de Finna en répondant à des cas d’actualité en temps réel.

#### **10. Documentation base de données et gestion des données (Pour les administrateurs de bases de données)**

La base de données PostgreSQL constitue le cœur de Finna. Elle stocke les utilisateurs, les sources et les résultats d’analyse.

**Structure principale** :

* users : Table contenant les informations des utilisateurs.
* sources : Articles et métadonnées collectées.

**Scripts d’initialisation** :

* Le fichier dbInit.sql configure les tables et insère les données initiales.

**Outils et configuration** :

* Docker Compose simplifie le lancement des services PostgreSQL.
* Surveillez les performances avec des outils comme pgAdmin ou des requêtes analytiques.