SENTIMENT ANALYZER

Documentation Technique Complète

Projet personelle — Analyse de sentiment par IA

Technologie : FastAPI · Vue.js · Hugging Face · Tailwind CSS

Version 1.0 — Mars 2026

1. Présentation du Projet

Sentiment Analyzer est une application web full-stack permettant d'analyser automatiquement le sentiment d'un texte (avis client, commentaire, tweet...) grâce à l'intelligence artificielle.

L'application classe le texte en trois catégories : Positif, Négatif ou Neutre, avec un score de confiance en pourcentage.

1.1 Objectifs

- Fournir une interface simple et intuitive pour l'analyse de sentiment
- Exploiter un modèle NLP pré-entraîné via l'API Hugging Face
- Exposer une API REST robuste avec gestion des erreurs
- Respecter les bonnes pratiques de sécurité (clé API côté serveur)

1.2 Technologies utilisées

Composant	Technologie	Rôle
Backend	FastAPI (Python)	API REST, validation, logique métier
Frontend	Vue.js 3 + Tailwind	Interface utilisateur réactive
IA / NLP	Hugging Face API	Modèle de classification de sentiment
Modèle IA	distilbert-sst-2	Analyse POSITIVE / NEGATIVE
Serveur HTTP	Uvicorn	Serveur ASGI haute performance

2. Architecture Logicielle

L'application suit une architecture Client-Serveur avec appel à une API externe :

```
[Navigateur - Vue.js] → HTTP POST /api/analyze → [Backend FastAPI] → HTTPS → [Hugging Face API]
```

2.1 Flux de données

- L'utilisateur saisit un texte dans l'interface
- Le frontend envoie une requête POST /api/analyze avec le texte
- Le backend valide l'entrée (3 à 1000 caractères)
- Le backend appelle l'API Hugging Face avec la clé secrète
- L'API retourne un label (POSITIVE/NEGATIVE) et un score
- Le backend normalise et renvoie la réponse au frontend
- L'interface affiche le résultat avec code couleur

2.2 Arborescence du projet

```
sentiment-analyzer/
├── backend/
│   ├── app/
│   │   ├── main.py              # Entrée FastAPI + CORS
│   │   ├── routes/analyze.py    # POST /api/analyze
│   │   ├── services/huggingface.py  # Appel IA
│   │   ├── schemas/sentiment.py     # Validation Pydantic
│   │   ├── core/config.py           # Variables d'env
│   │   └── utils/validators.py      # Sanitization
│   ├── requirements.txt
│   └── .env
└── frontend/
    ├── src/
    │   ├── components/          # TextInput, Button, Result, Loader
    │   ├── views/Home.vue       # Vue principale
    │   └── services/api.js      # Appels HTTP
    └── package.json
```

3. Documentation de l'API

3.1 Endpoint principal

Méthode	**POST**
URL	`/api/analyze`
Description	Analyse le sentiment d'un texte utilisateur via l'IA

3.2 Requête

```
{ "text": "Ce produit est incroyable !" }
```

Contraintes de validation :

- Champ obligatoire
- Longueur minimale : 3 caractères
- Longueur maximale : 1000 caractères
- Texte nettoyé automatiquement (trim + anti-injection)

3.3 Réponses

Succès (200)

```
{ "sentiment": "POSITIVE", "confidence": 0.9998 }
```

Erreur 400 — Texte invalide

```
{ "detail": "Text must be between 3 and 1000 characters." }
```

Erreur 503 — Service indisponible

```
{ "detail": "Sentiment service unavailable." }
```

4. Composants de l'Interface

Composant	Fichier	Description
TextInput	TextInput.vue	Textarea avec compteur de caractères et validation live
AnalyzeButton	AnalyzeButton.vue	Bouton avec état loading (spinner) et désactivation auto
Loader	Loader.vue	Spinner animé affiché pendant l'appel API
ResultDisplay	ResultDisplay.vue	Carte de résultat colorée + barre de confiance animée

4.1 Code couleur des résultats

- □POSITIF → Fond vert, bordure verte
- □ NÉGATIF → Fond rouge, bordure rouge
- ○ NEUTRE → Fond gris, bordure grise

5. Sécurité

5.1 Protection de la clé API

- La clé Hugging Face est stockée dans un fichier .env côté serveur
- Elle n'est jamais exposée au frontend ni dans le code source
- Le fichier .env est exclu du dépôt Git via .gitignore
- Un fichier .env.example sans clé est fourni comme modèle

5.2 Validation des entrées

- Trim automatique du texte (suppression espaces superflus)
- Vérification de la longueur (3 à 1000 caractères)
- Suppression des caractères de contrôle
- Détection de patterns suspects (XSS basique)

5.3 Gestion des erreurs

- Timeout de 5 secondes sur les appels Hugging Face
- 1 retry automatique en cas d'échec
- Réponses d'erreur standardisées (400, 503)
- Logs détaillés côté serveur (INFO / ERROR)

6. Déploiement et Lancement

6.1 Prérequis

- Python 3.10 ou supérieur
- Node.js 18 ou supérieur
- Clé API Hugging Face (gratuite sur huggingface.co)

6.2 Lancement en développement

Backend

```
cd backend
python -m venv venv
.\venv\Scripts\activate
pip install -r requirements.txt
uvicorn app.main:app --reload --port 8000
```

Frontend

```
cd frontend
npm install
npm run dev
```

URLs d'accès :

- Frontend : http://localhost:5173
- Backend API : http://localhost:8000
- Documentation Swagger : http://localhost:8000/docs

6.3 Variables d'environnement (.env)

```
HUGGINGFACE_API_KEY=hf_xxxxxxxxxxxxxxxxxxx
HF_MODEL_URL=https://router.huggingface.co/hf-inference/models/...
HF_TIMEOUT_SECONDS=5
HF_MAX_RETRIES=1
RATE_LIMIT_PER_MINUTE=10
```

7. Améliorations Futures

- Cache Redis pour éviter les appels redondants à l'API
- Support multilingue (français, espagnol, arabe...)
- Historique des analyses avec base de données
- Dashboard analytics avec graphiques
- Fine-tuning du modèle NLP sur données spécifiques
- Authentification utilisateur
- Export des résultats en CSV/Excel

— Fin de la documentation —

Projet réalisé dans un cadre personnelle — 2026