# Architecture du projet de résumé de texte

## Structure du projet

```

resume\_app/

│

├── app/

│ ├── \_\_init\_\_.py

│ ├── models/

│ │ ├── \_\_init\_\_.py

│ │ └── document.py

│ ├── services/

│ │ ├── \_\_init\_\_.py

│ │ ├── text\_extractor.py

│ │ └── summarizer.py

│ ├── api/

│ │ ├── \_\_init\_\_.py

│ │ └── routes.py

│ └── db.py

│

├── ui/

│ ├── \_\_init\_\_.py

│ └── streamlit\_app.py

│

├── tests/

│ ├── \_\_init\_\_.py

│ ├── test\_text\_extractor.py

│ └── test\_summarizer.py

│

├── config.py

├── requirements.txt

└── run.py

```

## Composants principaux

1. \*\*app/\*\* : Contient la logique principale de l'application

- \*\*models/\*\* : Définitions des modèles de données (SQLAlchemy)

- \*\*services/\*\* : Logique métier (extraction de texte, résumé)

- \*\*api/\*\* : Routes API Flask

- \*\*db.py\*\* : Configuration de la base de données

2. \*\*ui/\*\* : Interface utilisateur Streamlit

3. \*\*tests/\*\* : Tests unitaires et d'intégration

4. \*\*config.py\*\* : Configuration de l'application

5. \*\*requirements.txt\*\* : Dépendances du projet

6. \*\*run.py\*\* : Point d'entrée de l'application

## Flux de travail

1. L'utilisateur télécharge un fichier via l'interface Streamlit

2. Le fichier est envoyé à l'API Flask

3. Le service d'extraction de texte extrait le contenu

4. Le service de résumé génère un résumé à l'aide d'un modèle d'IA

5. Le résumé est stocké dans la base de données et renvoyé à l'utilisateur

## Implémentation Agile

1. \*\*Sprint 1\*\* : Mise en place de l'architecture de base et de l'extraction de texte

2. \*\*Sprint 2\*\* : Implémentation du service de résumé avec un modèle d'IA simple

3. \*\*Sprint 3\*\* : Développement de l'interface utilisateur Streamlit

4. \*\*Sprint 4\*\* : Intégration de la base de données et stockage des résumés

5. \*\*Sprint 5\*\* : Amélioration du modèle de résumé et optimisation des performances

6. \*\*Sprint 6\*\* : Tests, débogage et préparation au déploiement

## Technologies clés

- Flask : Framework web backend

- Streamlit : Interface utilisateur frontend

- SQLAlchemy : ORM pour la gestion de la base de données

- Modèles d'IA pré-entraînés : Pour la génération de résumés (ex: BART, T5)

- PyPDF2 / python-docx / python-pptx : Pour l'extraction de texte des différents formats de fichiers