Base de Connaissance - Documentation Technique

Numéro de révision : 1
Auteur(s) : Neil ANDRÉ

Vérificateur : Marc PARENTHOËN
Approbateur : Alain MOUSSEAU

Date création : Septembre 2025

9 pages

1. Objectif

Développer une application web permettant de :

- **Indexer des fichiers (.txt, .docx, .pptx, .pdf) dans une base de données MySQL.**
- **Rechercher du contenu dans ces fichiers via une interface web moderne.**
- **Gérer les fichiers (upload, ouverture, enregistrement, téléchargement).**
- **Authentifier les utilisateurs pour sécuriser les uploads.**

---

2. Prérequis

Bibliothèques Python nécessaires

pip install flask mysql-connector-python python-docx python-pptx pdfplumber werkzeug flask-login

Base de données MySQL

```
CREATE DATABASE knowledge_base;

USE knowledge_base;


-- Table pour stocker les documents indexés
CREATE TABLE documents (
    id INT AUTO_INCREMENT PRIMARY KEY,
    title VARCHAR(255) NOT NULL,
    content LONGTEXT,
    file_type VARCHAR(10) NOT NULL,
    import_date DATETIME DEFAULT CURRENT_TIMESTAMP
);



-- Table pour gérer les utilisateurs
CREATE TABLE users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(50) UNIQUE NOT NULL,
    password VARCHAR(255) NOT NULL
);
```

Répertoires nécessaires

Répertoire	Rôle
uploads/	**Stocke temporairement les fichiers uploadés.**
saved_files/	**Stocke les fichiers enregistrés définitivement.**
documents/	**Contient les fichiers à indexer automatiquement au démarrage.**
static/css/	**Contient le fichier style.css pour le design moderne.**
templates/	**Contient les templates HTML (base.html, search.html, etc.).**

3. Fonctionnalités Clés

A. Indexation des Fichiers

1. **Indexation automatique :**
    - **Au démarrage, le script parcourt documents/ et indexe tous les fichiers autorisés.**
    - **Utilise extract_content() pour extraire le texte selon le type de fichier.**
2. **Upload manuel :**
    - **Via /upload, les utilisateurs authentifiés peuvent uploader des fichiers.**
    - **Le contenu est extrait et inséré dans la base avec insert_into_db().**

B. Recherche de Contenu

Fonctionnalité	Description
Recherche par mots-clés	L'utilisateur saisit un mot-clé → requête SQL pour trouver les documents.
Filtrage par type	Filtre les résultats par type (TXT, DOCX, PPTX, PDF).
Surlignage des résultats	Mots-clés mis en évidence en jaune (`highlight_text()`).
Pagination	5 résultats par page.
Statistiques	Nombre de résultats par type de fichier.

C. Gestion des Fichiers

Action	Description
Ouverture des fichiers	Cliquez sur un nom de fichier pour l'ouvrir dans le navigateur (MIME correct).
Enregistrement	Après upload, option pour enregistrer le fichier dans saved_files/.
Téléchargement	Les fichiers peuvent être téléchargés depuis les résultats.

D. Authentification

- **Système de login** :
  - Un utilisateur admin est créé automatiquement (mot de passe : admin).
  - Seuls les utilisateurs connectés peuvent uploader des fichiers.

---

4. Flux de Fonctionnement

Étapes après le démarrage de Bdc.py

1. **Indexation automatique** :
   - Le script indexe tous les fichiers de documents/.
2. **Accès à l'interface** (http://localhost:5000) :
   - Recherche de mots-clés.
   - Connexion pour uploader des fichiers.
3. **Upload d'un fichier** :
   - Sélection → Upload → Page intermédiaire avec options :
     - Ouvrir le fichier.
     - Enregistrer dans saved_files/.
4. **Recherche et consultation** :
   - Saisie d'un mot-clé → résultats avec surlignage.
   - Cliquez sur un fichier pour l'ouvrir/télécharger.

---

5. Structure du Code

Fonctions Principales

Fonction	Rôle
allowed_file()	Vérifie si le fichier est d'un type autorisé.
save_file()	Sauvegarde le fichier uploadé dans uploads/.
extract_content()	Extrait le texte selon le type de fichier.
insert_into_db()	Insère le contenu extrait dans la base de données.

Fonction	Rôle
highlight_text()	Surligne les mots-clés dans les résultats.
uploaded_file()	Gère l'ouverture des fichiers dans le navigateur.
save_file_permanently()	Enregistre le fichier dans saved_files/.
process_directory()	Indexe tous les fichiers d'un répertoire.

Routes Flask

Route	Description
/	Page de recherche.
/login	Page de connexion.
/logout	Déconnexion.
/upload	Upload d'un fichier.
/post_upload/	Page intermédiaire après upload.
/save_file/	Enregistre le fichier dans saved_files/.
/uploads/	Ouvre le fichier dans le navigateur.

6. Exemple d'Utilisation

Cas d'usage : Rechercher et uploader un fichier

1. **Rechercher un mot-clé** :
    - Saisir "installation" → documents contenant ce mot s'affichent (mots surlignés).
2. **Uploader un fichier** :
    - Connexion avec admin/admin → cliquer sur **"Uploader un fichier"** → sélectionner un fichier.
    - Après upload : choisir d'ouvrir ou d'enregistrer le fichier.
3. **Consulter un fichier** :
    - Dans les résultats, cliquer sur le nom du fichier pour l'ouvrir.

---

7. Points Forts

✅ **Interface moderne** :

- Design épuré avec CSS intégré (style.css).
- Zone de recherche **visible et ergonomique**.
- Boutons et alertes stylisés.

✅ **Gestion des erreurs** :

- Fichiers introuvables, extractions échouées.

✅ **Sécurité** :

- Authentification requise pour les uploads.

✅ **Flexibilité** :

- Indexation automatique ou manuelle.

---

8. Commandes Utiles

Lancer le script

python Bdc.py

Vérifier les fichiers indexés

SELECT * FROM documents;

Ajouter un utilisateur (mot de passe haché requis)

INSERT INTO users (username, password) VALUES ('nouvel_utilisateur', '$2b$12$hashed_password');

9. Structure du Projet

```
appliBDC/
├── Bdc.py                  # Script principal Flask
├── static/
│   ├── style.css           # Feuille de style CSS
│   └── images/             # Dossier pour les icônes et logos
├── templates/              # Templates HTML
│   ├── about.html          # Template à propos du site (héritage)
│   ├── base.html           # Template de base (héritage)
│   ├── login.html          # Page de connexion
│   ├── register.html       # Page d'inscription
│   ├── search.html         # Page de recherche
│   └── ...                 # Autres templates
├── documents/              # Fichiers à indexer
├── saved_files/            # Fichiers sauvegardés
└── uploads/                # Fichiers téléchargés temporairement
```

10. Vue de l'Interface Web

Page de Recherche

- **Barre de recherche** claire et visible.
- **Résultats** avec surlignage des mots-clés.
- **Pagination** et filtres par type.
- **Boutons** pour uploader/se connecter.

Exemple de code HTML (extrait de search.html)

```
<div class="search-container">
  <form class="search-form" method="get">
    <input class="search-input" type="text" name="search" placeholder="Rechercher un mot-clé...">
    <select class="search-select" name="type">
      <option value="pdf">PDF</option>
      <option value="docx">Word</option>
    </select>
    <button class="btn btn-primary">Rechercher</button>
  </form>
</div>
```

CSS (extrait de style.css)

```
:root {
  --bg-color: #f5f5f5;        /* Fond gris clair */
  --nav-color: #2c3e50;       /* Couleur de la barre de navigation */
  --primary-color: #3498db;    /* Bleu primaire */
  --secondary-color: #2980b9;  /* Bleu secondaire (survol) */
  --text-color: #333;         /* Couleur du texte */
  --light-gray: #ecf0f1;      /* Gris clair pour les cartes */
  --highlight-color: #fff2a8;  /* Jaune clair pour la surbrillance */
  --border-radius: 8px;       /* Bordures arrondies */
  --box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);  /* Ombre légère */
}
```

```
body {
  font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
  background-color: var(--bg-color);
  color: var(--text-color);
  margin: 0;
  padding: 0;
  line-height: 1.6;
}

/* ===== Barre de navigation ===== */
nav {
  background-color: var(--nav-color);
  padding: 1rem 2rem;
  display: flex;
  justify-content: space-between;
  align-items: center;
  box-shadow: var(--box-shadow);
  position: sticky;
  top: 0;
  z-index: 100;
}
```

11. Améliorations Récentes

- **Design moderne** :
  - Couleurs harmonieuses (--primary-color: #4a6fa5).
  - Ombres et bordures arrondies pour un rendu professionnel.
  - Effets au survol (ex: bordure bleue au focus).
- **Zone de recherche optimisée** :
  - Taille agrandie, placeholder descriptif, fond blanc pour un meilleur contraste.
- **Responsive** :
  - Adapté aux mobiles et écrans larges.

---

12. Prochaines Étapes (Optionnel)

- **Ajouter un logo** dans static/images/.
- **Intégrer Font Awesome** pour des icônes.
- **Ajouter un système de tags** pour les documents.

Résumé Visuel

Base de Connaissance

├— Backend (Python/Flask)

├— Frontend (HTML/CSS)

├— Base de données (MySQL)

└— Gestion des fichiers (Upload/Indexation)

13. Points clés du code

Élément	Description	Fichier
Routes Flask	Définissent les URLs et les fonctions associées.	Bdc.py
Flask-Login	Gère l'authentification des utilisateurs	Bdc.py
Base de données	Connexion et requêtes SQL pour les documents et utilisateurs.	Bdc.py
Templates	Structure HTML avec héritage (base.html).	templates/
CSS	Styles pour l'affichage (couleurs, polices, effets).	static/css/style.css
Logs	Journalisation des erreurs et actions.	knowledge_base.log

---

3. Maintenance et dépannage

Problèmes courants et solutions :

Problème	Cause possible	Solution
CSS non appliqué dans base.html.	Fichier manquant ou chemin incorrect.	Vérifiez static/style.css et le lien
Erreur 404 pour les images	Fichiers manquants dans static/images/.	Ajoutez les icônes (docx-icon.png, etc.).
Erreur BuildError	Route manquante (ex: /register).	Ajoutez la route dans Bdc.py ou supprimez le lien.
Base de données inaccessible	Mauvais identifiants ou serveur MySQL arrêté.	Vérifiez get_db_connection() dans Bdc.py.
Fichiers non indexés	Dossier documents/ vide ou permissions incorrectes.	Vérifiez les chemins et permissions.