Fiche d'installation d'un projet Streamlit

1. Configuration de l'environnement Anaconda

Création de l'environnement

1. Ouvrez **Anaconda Prompt** ou **Terminal**.
2. Créez un nouvel environnement Python :
3. conda create -n mon_projet_streamlit python=3.9
4. Activez l'environnement :
5. conda activate mon_projet_streamlit

2. Installation des bibliothèques nécessaires

Installez les bibliothèques suivantes dans l'environnement activé :

pip install streamlit pandas matplotlib

Si vous utilisez **Git** ou **CSS** :

pip install gitpython

3. Structure du projet Streamlit

Voici la structure recommandée pour un projet Streamlit :

```
mon_projet_streamlit/
├── app.py                # Page principale
├── pages/                # Sous-pages (si multi-pages)
│   ├── detail.py
│   └── reco.py
├── utils/                # Fonctions utilitaires (comme le data_loader)
│   └── data_loader.py
├── assets/               # Fichiers statiques (CSS, images)
│   └── style.css         # Fichier CSS personnalisé
├── .streamlit/           # Configuration de Streamlit
│   └── config.toml
├── data/                 # Fichiers de données (CSV, Excel, JSON)
├── requirements.txt      # Bibliothèques nécessaires
```

└── README.md # Documentation du projet

4. Configuration de Git et GitHub

Initialisation du projet Git

1. Ouvrez votre terminal ou VS Code dans le dossier du projet.
2. Initialisez Git :
3. git init
4. Créez un fichier .gitignore pour exclure les fichiers inutiles :

echo "venv/" >> .gitignore echo "**pycache**/" >> .gitignore echo ".DS_Store" >> .gitignore

4. Connectez votre projet à GitHub :
```bash
git remote add origin
https://github.com/votre_nom_utilisateur/mon_projet_streamlit.git
```

5. Faites un premier commit et push :
6. Git add .
7. git commit -m "Initialisation du projet Streamlit"
8. git push -u origin main

5. Configuration de VS Code

Extensions nécessaires :

- **Python** (support du langage Python)
- **GitHub** (gestion Git)
- **Streamlit Runner for VS Code** (pour exécuter des apps Streamlit)
- **GitLens** (visualisation des commits Git)

Configuration de l'interpréteur Python

1. Ouvrez votre projet dans VS Code.
2. Accédez à **Ctrl + Shift + P** > **Python: Select Interpreter**.
3. Sélectionnez l'environnement créé (**mon_projet_streamlit**).

6. Étapes pour créer et configurer le dossier **.streamlit**

1. Créez un dossier **.streamlit** à la racine du projet.
2. Ajoutez un fichier **config.toml** à l'intérieur.
3. Exemple de configuration dans config.toml :
4. [server]
5. headless = false
6. port = 8501
7. enableCORS = false
8. enableXsrfProtection = false

7. Comment appliquer le CSS dans Streamlit ?

Explication rapide

Streamlit ne supporte pas directement les fichiers CSS, mais vous pouvez **injecter du CSS personnalisé** en utilisant **st.markdown()**.

Utilisation du fichier **style.css**

1. **Créez un fichier CSS** dans assets/style.css :
2. .title {
3. font-size: 36px;
4. color: #4CAF50;
5. text-align: center;
6. }
7.
8. .image-grid img {
9. width: 200px;
10. height: 300px;
11. border-radius: 8px;
12. }
13. **Chargez le fichier CSS** dans app.py :
14. def load_css(file_path):
15. with open(file_path) as f:
16. st.markdown(f"<style>{f.read()}</style>", unsafe_allow_html=True)
17.
18. load_css("assets/style.css")
19. **Utilisez des classes CSS** dans votre app :
20. st.markdown('<h1 class="title">Mon Projet Streamlit</h1>', unsafe_allow_html=True)
21. Le CSS peut être modifié dans style.css et sera appliqué dynamiquement dans Streamlit.

---

8. C'est quoi le data_loader ?

Le **data_loader** est un module Python qui centralise les fonctions pour **charger** et **pré-traiter** vos données. Cela évite de répéter le même code dans plusieurs fichiers.

Exemple de data_loader.py :

```
import pandas as pd

def load_csv(file_path):
    """Charge un fichier CSV en DataFrame."""
    return pd.read_csv(file_path)

def clean_data(df):
    """Nettoie les données en supprimant les valeurs manquantes et doublons."""
    return df.dropna().drop_duplicates()

def describe_data(df):
    """Retourne un résumé des données."""
    return df.describe()
```

Utilisation dans app.py :

```
from utils.data_loader import load_csv, clean_data

data = load_csv("data/mon_fichier.csv")
cleaned_data = clean_data(data)
st.dataframe(cleaned_data)
```