1. Architecture de l'Application

Notre application est une application web qui repose sur une architecture modulaire en trois couches : interface utilisateur, backend, et moteur de modèles d'apprentissage automatique pour analyser la consommation énergétique des bâtiments.

- Frontend (Interface Utilisateur) : Géré par Dash, un framework Python pour les applications interactives. Cette couche gère l'affichage des visualisations de données et les interactions utilisateur.
- Backend (Serveur Flask/Gunicorn) : Reçoit les requêtes de l'interface utilisateur et utilise des modèles de machine learning pour les prédictions énergétiques. Le serveur traite les demandes, exécute les calculs de prédiction, et retourne les résultats à l'interface.
- Modèles de Machine Learning : Comprend des modèles de classification et de régression, entraînés avec des données historiques. Utilisés pour prédire la consommation énergétique d'après les diagnostics de performance énergétique (DPE) et les caractéristiques des bâtiments.
- Base de Données : Regroupe les données DPE et les données de consommation énergétique réelle, stockées localement ou récupérées via l'API Ademe.

2. Étapes d'Installation de l'Application sur une Machine Locale

1. Prérequis :
    - Installer Python 3.9 ou une version plus récente.
    - Installer Git pour cloner le dépôt.
    - Installer Visual Studio Code ou configurer un environnement Python pour installer les bibliothèques/packages requis.

2. Installation :
    - Clonage du Dépôt : Cloner le projet depuis GitHub sur votre machine à l'aide de cette commande : « git clone https://github.com/hugocollin/m2_enedis ».
    - Installation des Bibliothèques : Une fois l'environnement activé, installer les bibliothèques nécessaires listées dans le fichier requirements.txt du projet.

3. Exécution de l'Application :
    - Lancer l'application depuis le terminal.
    - Une fois démarrée, l'application sera accessible dans un navigateur à une adresse locale.

3. Packages et Bibliothèques Utilisés

Voici les principales bibliothèques Python nécessaires :

- **Dash** : Un framework Python pour créer des applications web analytiques. Dash simplifie la création d'interfaces interactives avec des composants de visualisation et de contrôle, parfait pour des applications de data science.
- **Gunicorn** : Un serveur HTTP Python WSGI, performant et adapté aux environnements de production. Il est utilisé pour déployer des applications web en assurant la gestion des connexions et de la scalabilité.
- **Joblib** : Une bibliothèque pour la sérialisation (sauvegarde) efficace des objets Python, en particulier pour les modèles de machine learning. Joblib permet de sauvegarder des modèles pour les réutiliser sans les entraîner à nouveau.
- **Kaleido** : Outil pour exporter des visualisations créées avec Plotly sous forme d'images statiques (PNG, PDF). Utile pour générer des rapports ou partager des graphiques en dehors de l'application.
- **Pandas** : Une bibliothèque de manipulation de données largement utilisée, permettant des opérations avancées sur des structures de données (DataFrames). Elle facilite le nettoyage, la transformation et l'analyse des données de consommation.
- **Plotly** : Une bibliothèque de visualisation qui permet de créer des graphiques interactifs, intégrée avec Dash pour des visualisations avancées de données. Elle propose des types de graphiques variés pour représenter des données complexes.
- **Requests** : Utilisée pour envoyer des requêtes HTTP, principalement pour récupérer des données depuis des API externes (comme l'API Ademe). Requests simplifie l'interaction avec des services web RESTful.
- **Scikit-learn** : Une bibliothèque de machine learning offrant de nombreux algorithmes pour la classification, la régression, le clustering, et plus encore. Scikit-learn est utilisé ici pour entraîner des modèles de prédiction basés sur les données de DPE et de consommation.