Rapport de démarche – Outil de suivi de consommation énergétique

1. Contexte et objectifs du projet

L'objectif de ce projet était de concevoir un **outil web de suivi et de visualisation des consommations énergétiques**, permettant :

- la saisie de données de consommation,
- leur consultation sous forme de liste,
- un tableau de bord graphique

---

2. Démarche de conception

2.1 Approche générale

La démarche suivie a été itérative :

1. **Définition du modèle de données** (catégories, consommations, années...)
2. **Mise en place de l'API et de la base de données**
3. **Création des interfaces principales** (formulaire, liste, dashboard)
4. **Amélioration progressive de l'expérience utilisateur** (filtres, suppression, validations, erreurs, accesibilité numérique...)
5. **Identification des limites et ajustements nécessaires**

---

3. Choix techniques

3.1 Backend

- **Framework** : FastAPI
- **ORM** : SQLAlchemy
- **Base de données** : PostgresSQL

Justification :

- SQLAlchemy permet une bonne séparation entre la logique et la base de données.

3.2 Frontend

- **Templates HTML** avec Jinja2
- **CSS** pour définir un style personnalisé.
- **JavaScript** pour la gestion des filtres et des interactions utilisateur

Justification :

- Choix volontairement simple afin de privilégier la lisibilité et la maintenabilité.

- Évite la complexité d'un framework frontend lourd pour un périmètre fonctionnel maîtrisé.

---

4. Ingénierie IHM (Interface Homme-Machine)

4.1 Principes d'ingénierie retenus

La conception de l'interface utilisateur s'est appuyée sur plusieurs principes fondamentaux de l'ingénierie IHM :

- **Simplicité et lisibilité** : limiter la charge cognitive de l'utilisateur.
- **Cohérence** : mêmes logiques d'interaction sur l'ensemble de l'application.
- **Feedback utilisateur** : chaque action importante génère un retour visible. (Feedback)
- **Prévention des erreurs** plutôt que correction a posteriori. (Feedback)
- **Progressivité** : affichage des informations selon le besoin réel.

Ces principes ont guidé aussi bien la structure des pages que le comportement dynamique des formulaires et filtres.

4.2 Conception du formulaire de saisie

Le formulaire de saisie a été conçu pour :

- guider l'utilisateur étape par étape,
- réduire les erreurs de saisie,
- assurer la cohérence des données enregistrées.

Choix IHM notables :

- Sélection par **listes déroulantes** (catégorie, sous-catégorie) plutôt que saisie libre.
- Champs obligatoires clairement identifiés.
- Validation côté backend pour garantir l'intégrité des données.
- Organisation verticale logique (année → catégorie → sous-catégorie → valeur).

L'ajout des **sous-catégories** a amélioré la précision des données (ce que je pense nécessaire dans le domaine des énergies), mais a introduit une complexité supplémentaire dans l'interface.

4.3 Conception de la liste et des filtres

La page liste a été pensée comme un **outil de consultation et de contrôle** des données.

Choix d'ingénierie IHM :

- Tableau lisible avec hiérarchisation visuelle des informations et tags de catégorisation.
- Filtres combinables (catégorie / année).
- Conservation des filtres après action (suppression).
- Bouton de suppression avec **confirmation explicite** afin d'éviter toute erreur critique.

Un travail spécifique a été mené pour :

- empêcher les états incohérents (ex. filtres invalides),
- garantir des URLs propres,
- éviter les erreurs techniques visibles par l'utilisateur.

4.5 Gestion des erreurs et prévention

Un soin particulier a été apporté à la **prévention des erreurs utilisateur** :

- Distinction claire entre :
  - *"Toutes les années"* (choix par défault).
  - *"Une année en particulier"* (choix explicite).
- Messages d'erreur visibles et compréhensibles.
- Aucune action destructrice sans confirmation.

Ces choix visent à améliorer la **fiabilité perçue** de l'outil.

---

5. Écarts fonctionnels et impact IHM

5.1 Ajout des sous-catégories

L'ajout des sous-catégories, non prévu initialement, a eu un impact direct sur l'IHM :

- Augmentation du nombre de champs dans les formulaires.
- Nécessité de clarifier la relation catégorie / sous-catégorie.
- Risque accru de confusion pour un utilisateur novice.

Ce décalage a mis en évidence l'importance :

- d'une **analyse des besoins approfondie en amont**,
- et de la prise en compte des impacts IHM lors de toute évolution fonctionnelle.

---

6. Problèmes identifiés et pistes d'amélioration IHM

6.1 Problèmes identifiés

- Complexité croissante du formulaire avec l'ajout de sous-catégories.
- Absence de messages contextuels explicatifs pour certains choix.
- Navigation perfectible entre les différentes pages.
- Pas de gestion avancée des profils utilisateurs (débutant / expert).
- Pas d'import de fichiers externes pour provisionner la BDD.

6.2 Améliorations envisagées

- Aide contextuelle (info-bulles, descriptions dynamiques).
- Regroupement ou masquage conditionnel de champs.
- Bouton explicite **"Réinitialiser les filtres"**.
- Amélioration de l'accessibilité (navigation clavier, contrastes).

- Tests utilisateurs pour valider les choix IHM.

---

7. Conclusion

L'ingénierie IHM a joué un rôle central dans ce projet, en cherchant à concilier :

- simplicité d'usage,
- précision des données,
- et évolutivité fonctionnelle.

Les écarts entre le périmètre initial et la solution finale illustrent une problématique courante en développement :
l'évolution des besoins entraîne une complexification de l'IHM, qui doit être maîtrisée par des choix de conception rigoureux.

Ce projet constitue un exemple de mes compétences, tant sur le plan technique que sur celui de l'ingénierie des interfaces.

---

8. Références

8.1 Environnement de travail

Le développement de l'application a été réalisé à l'aide de l'environnement suivant :

- **Visual Studio Code** : éditeur principal pour le développement backend et frontend.
- **Docker Desktop** : exécution de l'application et de la base de données PostgreSQL dans des conteneurs, garantissant un environnement reproductible.
- **PostgreSQL** : base de données relationnelle utilisée pour le stockage des données énergétiques.

Ces outils ont permis de travailler dans un contexte proche des conditions réelles de déploiement.

8.2 Utilisation des IA génératives

Des outils d'intelligence artificielle générative ont été utilisés comme **assistance au développement**, de manière encadrée :

- **GitHub Copilot (Claude Haiku 4.5)** : aide à l'auto-complétion du code et à la génération de structures de code (CRUD, modèles...).
- **ChatGPT (version 5.2)** : support à la réflexion technique, à la compréhension des technologies utilisées et à la rédaction de documentation.

Les IA n'ont pas produit de solution finale autonome :
le code a été **analysé, adapté, testé et validé manuellement**.
Elles ont servi à améliorer la productivité et la qualité, sans remplacer le travail de conception.