APPROCHE de Devis N° : 0125-018	
Client	GAZFIO
N° réf. client	016
Adresse	45 Av. du Président J F Kennedy, 64200 Biarritz
Date limite de paiement	
Modalité de paiement	Virement bancaire

Voici, étape par étape, la démarche pour transformer le classeur Excel + VBA en une véritable application web, ainsi que l'option de développement optimale.

---

1. Cartographie du fichier existant

- Extraire le code VBA : VBE7.dll → export `.bas`, `.cls`, `.frm`.
- Inventorier :
  - Feuilles et plages nommées utilisées par le code.
  - Modules standards, objets ThisWorkbook & Sheets.
  - Références COM/VBA : appels à l'Add-In « COCO » (probablement un add-in COM ou une DLL).
  - Fonctions Worksheet personnalisées (UDF).
  - Tableaux croisés, graphiques, MFC.
  - Connexions externes éventuelles (PowerQuery, ODBC, fichiers texte…).

---

2. Analyse fonctionnelle et technique

- Classer les macro-processus : import de données, calculs « complexes » (délégués à COCO), génération de rapports, export.
- Quantifier :
  - ⧉ Nombre de lignes de code VBA.
  - ⧉ Nombre de points d'entrée (boutons, événements Workbook / Worksheet).
  - ⧉ Dépendances COCO : nom des fonctions, signatures, retours d'erreur.
- Identifier les blocs « purs calculs » et ceux « UI Excel ».

---

3. Décision d'architecture cible

Option A – Web app 100 % custom (recommandé)
- Front-end : React + TypeScript (+ MUI ou AntD).
- Back-end : Python FastAPI (ou .NET Core C# si tu veux rester Microsoft) qui encapsule :
  - Reprise des algorithmes VBA → modules Python/C#.
  - Appels à COCO :
    • Si COCO expose déjà une DLL/.NET : interop C# dans un micro-service.
    • Sinon, wrapper la DLL COM via ctypes/pywin32 ou via un service Windows isolé, exposé en HTTP (façade REST).
- Persistance : PostgreSQL (tables + JSONB pour stocker l'équivalent des feuilles dynamiques).
- Reporting : PDFKit (Node) ou WeasyPrint pour les rendus PDF, ou bien BIRT/Power BI pour dashboards externes.

Pourquoi cette option :
- Séparation claire UI / logique.
- Facile à dockeriser et déployer (Azure App Service, AWS ECS, on-prem).
- Performance côté calcul : FastAPI async + uWSGI/Gunicorn.
- Écosystème riche pour la science des données (NumPy, pandas) ⇒ remplace aisément les formules Excel.

Option B – Excel-online (+ Office JS)
- Garder le classeur, migrer la logique en TypeScript Office Script + Azure Functions.
- Moins disruptif pour les utilisateurs, mais limité si COCO n'a pas d'API web.

---

4. Feuille de route projet

Phase 0 – Kick-off (1 semaine)
- Export complet du VBA, réunion de cadrage, validation des flux.

Phase 1 – Rétro-ingénierie (2-3 semaines)
- Commenter chaque routine.
- Produire la matrice « fonction → feuille → dépendances ».
- POCs : appel COCO depuis un service Python/C#.

Phase 2 – Conception applicative (2 semaines)
- Modélisation BD.
- Maquettes Figma du front-end.
- Spécs API REST & events.

Phase 3 – Dév cœur métier (4-6 semaines)
- Portage des algorithmes (tests unitaires VS Excel).
- Service wrapper COCO.
- Génération rapports PDF.

Phase 4 – Front-end (3-4 semaines)
- Écrans saisie, tableaux, téléchargements CSV/XLSX.
- Authentification (Azure AD ou Keycloak).

Phase 5 – Recette & migration (2 semaines)
- Comparaison sorties Excel vs Web (±0,1 %).
- Formation utilisateurs.

Phase 6 – Déploiement & support (continu)

---

5. Chiffrage indicatif

- Analyse / Conception : 15 j/h
- Développement back-end : 30 j/h
- Front-end : 20 j/h
- QA / Recette / Docs : 10 j/h
- PMO + Buffer 20 % : 15 j/h

Total Indicatif : 90 j/h

Voici une estimation réaliste – et plutôt agressive – du temps qu'un développeur polyvalent (back + front) peut consacrer au projet s'il :

- ré-utilise massivement l'IA générative pour rédiger le code boilerplate, la documentation et les tests ;
- dispose d'une API documentée pour COCO (pas de rétro-ingénierie DLL) ;
- un rythme concentré "full-time" (≈ 7 h productives / jour).

Estimation par phase (1 seul dev)

- 1. Kick-off & prise en main : **1 jour**
  Importer le matériel Excel, lire la doc COCO, configurer dépôt Git + hébergement

- 2. Rétro-conception rapide : **2 jours**
  - Extraction automatique du VBA (VBA-Extractor + ChatGPT pour commentaires).
  - Mapping fonctions ↔️ API COCO.
  - Tableur de dépendances généré avec Python.

- 3. Architecture & maquettes : **2 jours**
  - Choix techno (FastAPI + React).
  - Schéma BD avec SQLModel/SQLAlchemy.
  - Figma basse fidélité (assistée IA).

- 4. Back-end (API + wrapper COCO) : **5 jours**
  - Génération squelette FastAPI (Copilot/ChatGPT).
  - Endpoints CRUD, sécurité JWT.
  - Client Python pour COCO (swagger → codegen).
  - Tests unitaires (pytest-async, générés en partie par IA).

- 5. Portage logique métier : **4 jours**
  - Conversion des algorithmes Excel en fonctions Python/NumPy.
  - Validation vs. résultats XLSM (scripts de diff auto).

- 6. Front-end React + charts : **4 jours**
  - Vite + TypeScript + Mantine/Material UI.
  - Formulaires, tableaux (AG-Grid), graphiques (ECharts).
  - Appels API avec React-Query.
  - Storybook assisté IA.

- 7. Génération de rapports (PDF/Excel) : **2 jours**
  - Templates Jinja2 + WeasyPrint / ExcelWriter.

- 8. Recette & optimisation : **3 jours**
  - Tests E2E (Playwright, prompts IA pour scripts).
  - Ajustements perf & UX.

- 9. Déploiement CI/CD & docs : **2 jours**
  - Docker compose, GitHub Actions, déploiement sur VPS ou Azure Web App.
  - Rédaction docs Markdown + screencasts (aides IA).

Total estimé : **25 jours ouvrés**
≈ 5 semaines calendaires pour un développeur unique.

Facteurs de réussite

- Prompts précis : plus les instructions à l'IA sont granulaires, plus le temps "gagné" est élevé.
- Tests automatisés : indispensables pour sécuriser la rapidité.
- Discipline Git (petits commits + revues IA).
- Gestion des "unknown unknowns" : prévoir 2-3 jours tampon si l'API COCO a des zones grises.

Conseil final

Avec 25 jours, on table sur une vélocité x2 à x2,5 par rapport à un cycle classique (80-90 j/h) grâce à l'IA générative et à une API COCO disponible. On garde néanmoins un buffer (10 %) pour l'imprévu, surtout en prod.

Estimation Optimisée : 30 j/h x 600€ = 18 000€ HT