Rapport d’environnement technique pour migration web de l’application

Unai DURANONA
Laurent Jaurey

10 juillet 2025

Table des matières

Analyse du Code VBA et des Liens Externes

Ce document détaille l'analyse du code VBA extrait du fichier XLSM fourni, en se concentrant sur la structure du projet, les logiques de calcul, et les interactions avec des applications externes, notamment COCO.

1. Structure Générale du Projet VBA

Le projet VBA est composé de plusieurs modules et formulaires, chacun ayant un rôle spécifique dans la fonctionnalité globale du fichier Excel. Les principaux composants identifiés sont :

•COFEWorkbook.cls (Module de Classe) : Gère les événements au niveau du classeur (ouverture, activation, désactivation de feuilles) et l'intégration du menu COFE dans l'interface Excel. Il semble être le point d'entrée principal pour l'initialisation de l'intégration avec COFE.

•KeepExamples.frm (Formulaire Utilisateur) : Un formulaire simple qui permet à l'utilisateur de sélectionner les feuilles d'exemple à conserver ou à supprimer lors de l'ouverture du classeur.

•FlowsheetObject.cls (Module de Classe) : Ce module est crucial car il interagit directement avec l'objet COCO_COFE.Document. Il contient des fonctions pour récupérer l'objet COFE intégré, ajuster sa taille sur la feuille de calcul, et scanner les objets OLE présents dans le classeur.

•PromptFlowsheet.frm (Formulaire Utilisateur) : Un formulaire utilisé pour inviter l'utilisateur à choisir entre créer un nouveau document flowsheet ou en charger un existant à partir d'un fichier .fsd.

•COFEActions.bas (Module Standard) : Contient des procédures et fonctions liées aux actions utilisateur, telles que la configuration du flowsheet, le remplacement du document flowsheet intégré, la sauvegarde d'une copie du document intégré, et les fonctions de calcul.

•COFEUtilties.bas (Module Standard) : Fournit des fonctions utilitaires génériques pour la manipulation des plages Excel, la gestion des erreurs, et la récupération de l'objet COFE_Document.

•COFEDimensionality.bas (Module Standard) : Gère la mise en forme des unités et des dimensions, probablement pour l'affichage des résultats de calculs.

•COFEStreams.bas (Module Standard) : Contient des fonctions pour interagir avec les flux (streams) de données au sein de l'application COFE, permettant de récupérer ou de définir des propriétés comme la pression, la température, le débit et la composition.

•COFEThermo.bas (Module Standard) : Gère les aspects thermodynamiques, y compris la récupération des modèles de matériaux, des noms de composés, et le calcul des propriétés thermodynamiques et des équilibres.

•COFEUnitOperations.bas (Module Standard) : Fournit des fonctions pour interagir avec les opérations unitaires (unit operations) définies dans COFE, permettant de récupérer des paramètres, des ports et leurs propriétés.

•Feuil1.cls (Module de Classe de Feuille) : Contient la procédure ExportPDF qui gère l'exportation de certaines feuilles Excel (ZOE, Computation, Mass Balance) en format PDF.

•Feuil2.cls, Feuil3.cls, Feuil4.cls, Feuil5.cls, Feuil6.cls (Modules de Classe de Feuilles) : Ces modules sont actuellement vides, ce qui indique qu'ils pourraient être des placeholders ou des restes de développements antérieurs.

2. Logique et Calculs

Le fichier XLSM est fortement orienté vers des calculs complexes, en particulier ceux liés à la modélisation de procédés chimiques ou physiques, comme en témoigne l'intégration avec l'application COCO. Les calculs sont orchestrés par le code VBA qui :

•Lit les données des feuilles Excel : Des plages nommées comme CH4Stream, O2Stream, InletT, PressLP, minTrange, CityLocation sont utilisées pour récupérer les entrées utilisateur ou les paramètres de calcul.
•Met à jour les paramètres de COFE : Les fonctions COFE_SetStreamItemValue sont utilisées pour envoyer des valeurs (composition, débit, température, pression) à l'objet COCO_COFE.Document.
•Déclenche les calculs dans COFE : La fonction COFE_CalculateFlowsheet appelle la méthode Solve de l'objet COFEDocument, ce qui déclenche les calculs au sein de l'application COCO.
•Récupère les résultats de COFE : Des fonctions comme COFE_GetStreamItemValueShort, COFE_GetStreamPressure, COFE_GetStreamTemperature, COFE_GetStreamFlow, COFE_GetStreamComposition sont utilisées pour récupérer les résultats des calculs effectués par COCO et les afficher dans les feuilles Excel.
•Effectue des études paramétriques : La procédure COFE_EtudeParametrique_TP est un exemple clair d'automatisation d'une étude de sensibilité. Elle itère sur une plage de températures, exécute les calculs COFE pour chaque température, et enregistre les résultats dans la feuille SimulationResults. Cela implique une boucle de calculs intensifs.
•Génère des graphiques : Les procédures COFE_CreerGraphique et COFE_UpdateGraphique sont responsables de la création et de la mise à jour de graphiques basés sur les résultats des simulations, notamment les coûts énergétiques en fonction de la température et les données météorologiques.
•Gère les bilans massiques : La procédure COFE_MassBalance effectue des calculs de bilan massique en interagissant avec COFE pour obtenir les propriétés des flux (pression, température, débit, composition) et les affiche dans la feuille Mass Balance.

La logique de calcul est complexe et dépend fortement de l'interaction bidirectionnelle entre Excel et l'application COCO. Les calculs ne sont pas effectués directement dans Excel mais délégués à COCO, qui est une application externe spécialisée dans la modélisation de procédés.

3. Liens et Appels Externes (Application COCO)

Le fichier XLSM est intrinsèquement lié à l'application COCO. Cette dépendance est manifeste à travers plusieurs éléments du code VBA :

•ProgID COCO_COFE.Document : C'est l'identifiant programmatique de l'objet OLE (Object Linking and Embedding) qui représente l'intégration de COFE dans Excel. Chaque fois que le code VBA interagit avec l'application COCO, il le fait via cet objet. Par exemple, obj.progID = "COCO_COFE.Document" est utilisé pour identifier et manipuler l'objet COFE intégré.
•Interfaces COM (Component Object Model) : Le code VBA utilise des interfaces COM exposées par COCO pour accéder à ses fonctionnalités. Bien que les types exacts des interfaces ne soient pas directement visibles dans le code VBA (ils sont souvent définis dans des bibliothèques de types référencées), les noms de fonctions et de propriétés comme ICOFEDocument, ICOFEMaterial, ICapeThermoMaterialObject, ICapeCollection, ICapeParameter, ICapeUnit, ICapeUnitPort, ICapeIdentification,

ICapeParameterSpec, ICapeMaterialTemplateSystem indiquent clairement l'utilisation de l'API de COCO (ou plus précisément, de l'architecture CAPE-OPEN, dont COCO est une implémentation).
•Méthodes et Propriétés Spécifiques à COFE :
•COFEDocument.Solve : Déclenche le moteur de calcul de COCO.
•COFEDocument.ShowMainConfiguration : Ouvre la boîte de dialogue de configuration de COFE.
•COFEDocument.GetStream, COFEDocument.GetUnit, COFEDocument.GetSimulationContext : Permettent d'accéder aux objets internes de COCO (flux, opérations unitaires, contexte de simulation).
•GetProp, SetProp : Utilisées pour lire et écrire des propriétés (pression, température, composition, etc.) des flux et des matériaux gérés par COCO.
•CalcEquilibrium : Déclenche les calculs d'équilibre thermodynamique au sein de COCO.
•Déclarations Declare PtrSafe Function GetSaveFileName Lib "comdlg32.dll" : Cette déclaration indique l'utilisation de fonctions de l'API Windows (comdlg32.dll) pour afficher des boîtes de dialogue de sauvegarde de fichiers. Bien que ce ne soit pas directement lié à COCO, cela montre une dépendance à l'environnement Windows et aux bibliothèques système.
•Fichiers .fsd : Le formulaire PromptFlowsheet et la procédure COFE_ReplaceFlowsheetDocument font référence à des fichiers .fsd (Flowsheet Document). Il est très probable que ce format de fichier soit le format natif de COCO pour sauvegarder et charger des modèles de procédés.
En résumé, l'application COCO n'est pas seulement une source de données, mais le moteur de calcul principal du fichier XLSM. Le code VBA agit comme une interface utilisateur et un orchestrateur, préparant les données pour COCO, déclenchant les calculs, et affichant les résultats. La conversion en application web nécessitera de remplacer ou de réimplémenter cette intégration profonde avec COCO.

4. Autres Liens et Dépendances

Outre l'intégration avec COCO, le fichier XLSM présente les dépendances suivantes :
•Fichiers .fsd : Comme mentionné, ces fichiers sont des documents de flowsheet spécifiques à COCO. Leur gestion (ouverture, sauvegarde) est intégrée au VBA.
•Feuilles Excel : Les données d'entrée et de sortie, ainsi que les paramètres de configuration, sont stockés directement dans les feuilles Excel (ZOE, Computation, Mass Balance, MeteoData, SimulationResults). La structure de ces feuilles est essentielle au fonctionnement du classeur.
•Fonctions Excel natives : Le code VBA utilise des fonctions Excel natives comme Application.WorksheetFunction.Transpose, Application.Max, Evaluate, Range, Sheets, ThisWorkbook, OLEObjects, CommandBars, FileDialog, etc. Ces fonctions devront être réimplémentées ou remplacées par des équivalents web.
•API Windows (comdlg32.dll) : Utilisée pour les boîtes de dialogue de fichiers, ce qui est une dépendance au système d'exploitation Windows.
•Objets Scripting.FileSystemObject : Utilisé pour la manipulation de fichiers (vérification de l'existence de fichiers), ce qui est également une dépendance à l'environnement Windows.
Cette analyse met en évidence la complexité de la migration vers une application web, principalement en raison de la dépendance critique à l'application COCO et à l'environnement Excel/Windows.

5. Identification des Appels Externes Spécifiques à COCO et Autres Dépendances

L'analyse approfondie du code VBA révèle plusieurs points d'intégration et d'appels spécifiques à l'application COCO, ainsi que d'autres dépendances externes. Ces éléments sont cruciaux pour comprendre l'étendue de la migration nécessaire vers une application web.

5.1. Intégration COCO via OLE/COM

L'interaction principale avec COCO se fait via l'objet OLE COCO_COFE.Document. Les méthodes et propriétés utilisées sont les suivantes :
•COFEWorkbook.cls :
•obj.progID = "COCO_COFE.Document" : Utilisé pour identifier l'objet COFE intégré dans le classeur. C'est le point d'ancrage de l'intégration.
•obj.AutoLoad = True : Indique que l'objet COFE doit être chargé automatiquement à l'ouverture du classeur.
•Call COFE_CompleteCOFERecalc : Cette fonction, bien que définie dans COFEUtilties.bas, est appelée ici pour s'assurer que tous les objets COFE sont recalculés. Elle parcourt les feuilles et force le recalcul des formules contenant "COFE_".
•FlowsheetObject.cls :
•Function GetCOFEObject() As OLEObject : Cette fonction est chargée de trouver et de retourner l'objet COCO_COFE.Document parmi les objets OLE de la feuille. Elle est fondamentale pour toutes les interactions ultérieures avec COCO.
•obj.Activate : Active l'objet COFE intégré, le rendant accessible pour les opérations.
•fixsize : Ajuste la taille de l'objet COFE sur la feuille de calcul. Cela implique une manipulation directe de l'interface utilisateur de l'objet embarqué.
•obj.progID = "COCO_COFE.Document" : Vérification constante de l'identifiant de l'objet pour s'assurer qu'il s'agit bien de l'objet COFE.
•ScannerObjetsOLE : Cette procédure parcourt tous les objets OLE dans toutes les feuilles pour identifier leur progID, leur nom, leur visibilité et leur position. Elle confirme la présence de COCO_COFE.Document.
•COFEActions.bas :
•Set COFEDocument = COFE_GetCOFEDoc : Récupère l'instance du document COFE. Cette ligne est présente dans presque toutes les fonctions qui interagissent avec COCO.
•Call COFEDocument.ShowMainConfiguration : Ouvre la boîte de dialogue de configuration de l'application COFE. C'est une interaction directe avec l'interface utilisateur de COCO.
•ws.OLEObjects.Add("COCO_COFE.Document") ou ws.OLEObjects.Add(, prompt.filebox.Text) : Permet d'insérer un nouvel objet COFE dans la feuille de calcul, soit un document vide, soit un document à partir d'un fichier .fsd existant. Cela montre la capacité du VBA à instancier et manipuler des objets COM de COCO.
•oldObject.Delete : Supprime un objet COFE existant, ce qui est nécessaire lors du remplacement d'un flowsheet.
•Set doc = obj.object : Accède à l'objet COM sous-jacent de l'objet OLE, permettant d'appeler ses méthodes et propriétés spécifiques.
•COFEDocument.SaveCopy(OFN.strFile) : Sauvegarde une copie du document flowsheet intégré dans un fichier .fsd. Cela confirme la gestion des fichiers .fsd par COCO.
•COFEDocument.Solve : La fonction la plus critique, elle déclenche le moteur de calcul de COCO. C'est l'appel qui exécute les simulations complexes.

•COFEStreams.bas :
•COFEDocument.GetStream(streamID) : Récupère un objet ICOFEStream ou ICapeThermoMaterialObject à partir de son identifiant. Ces objets représentent les flux de matière ou d'énergie dans le flowsheet COCO.
•str1.GetProp("pressure", "overall", Empty, "", ""), str1.GetProp("temperature", "overall", Empty, "", ""), str1.GetProp("totalFlow", "overall", Empty, "", basis), str1.GetProp("fraction", "overall", Empty, "", basis) : Ces appels sont utilisés pour récupérer diverses propriétés des flux (pression, température, débit, composition) depuis COCO. Les paramètres overall, Empty, basis et les noms de propriétés (pressure, temperature, totalFlow, fraction) sont spécifiques à l'API CAPE-OPEN/COCO.
•str1.SetProp("pressure", "overall", Empty, "", "", v), str1.SetProp("temperature", "overall", Empty, "", "", v), str1.SetProp("fraction", "overall", Empty, "", basis, X()) : Utilisés pour définir les propriétés des flux, permettant à Excel de modifier les données d'entrée de COCO.
•COFEThermo.bas :
•COFEDocument.GetSimulationContext : Récupère le contexte de simulation de COCO, qui contient des informations sur les modèles thermodynamiques et les matériaux disponibles.
•SimContext.CreateMaterialTemplate(templateName) : Crée un modèle de matériau basé sur un nom de template (par exemple, "default").
•templ.CreateMaterialObject() : Crée un objet matériau à partir du template, permettant de manipuler ses propriétés.
•mat.ComponentIds() : Récupère la liste des composants chimiques gérés par le modèle thermodynamique.
•mat.SetProp("fraction", phaseName, Empty, "", compositionBasis, val) : Définit la composition d'une phase pour un matériau.
•mat.SetProp("temperature", "overall", Empty, "", "", val) et mat.SetProp("pressure", "overall", Empty, "", "", val) : Définissent la température et la pression globales d'un matériau.
•mat.CalcProp(val, val1, calcType) : Calcule une propriété spécifique d'un matériau (par exemple, enthalpie, entropie).
•mat.GetProp(propertyName, phaseName, Empty, calcType, basis) : Récupère la valeur d'une propriété calculée.
•mat.GetComponentConstant(props, comps) : Récupère une constante de composant (par exemple, masse molaire).
•mat.CalcEquilibrium(eqType, Empty) : Effectue un calcul d'équilibre thermodynamique (par exemple, flash TP, flash PH) pour un matériau.
•mat1.GetSupportedPhaseList : Récupère la liste des phases supportées par le modèle thermodynamique.
•mat.PhaseIds : Récupère les identifiants des phases présentes dans le matériau.
•COFEUnitOperations.bas :
•COFEDocument.GetUnit(unitID) : Récupère un objet ICapeUnit à partir de son identifiant. Ces objets représentent les opérations unitaires (réacteurs, échangeurs, pompes, etc.) dans le flowsheet COCO.
•unit.Parameters : Accède à la collection de paramètres d'une opération unitaire.
•collection.Item(i) ou collection.Item(parameterID) : Récupère un paramètre spécifique de l'opération unitaire.
•param.Value : Lit ou écrit la valeur d'un paramètre de l'opération unitaire.
•param.Specification : Accède aux spécifications d'un paramètre, y compris sa dimensionnalité.
•unit.ports : Accède à la collection de ports d'une opération unitaire (points de connexion pour les flux).
•port.portType : Récupère le type de port (matériau, énergie, information).

•port.Direction : Récupère la direction du port (entrée, sortie, entrée/sortie).
•port.connectedObject : Récupère l'objet (flux) connecté à un port.

5.2. Dépendances Windows et Fichiers

•comdlg32.dll (API Windows) : Utilisée dans COFEActions.bas pour la fonction GetSaveFileName, qui ouvre une boîte de dialogue standard de Windows pour la sélection d'un chemin de sauvegarde. Cette dépendance est spécifique à l'environnement Windows.
•Scripting.FileSystemObject : Utilisé dans PromptFlowsheet.frm pour vérifier l'existence d'un fichier (fs.FileExists(filebox.Text)). C'est un objet COM de Windows pour la manipulation du système de fichiers.
•Fichiers .fsd : Ces fichiers sont le format natif de COCO pour les documents de flowsheet. Le code VBA les manipule directement pour charger ou sauvegarder des configurations de simulation. La migration nécessitera une gestion de ces fichiers côté serveur ou une conversion de leur contenu.

5.3. Dépendances Excel Natives

De nombreuses fonctions et objets Excel sont utilisés, ce qui nécessitera une réimplémentation ou une adaptation dans un environnement web :
•Manipulation de l'interface utilisateur Excel : Application.CommandBars, Application.FileDialog, MsgBox, Application.DisplayAlerts, Application.ScreenUpdating, Application.StatusBar sont tous des éléments spécifiques à l'interface utilisateur d'Excel.
•Objets de classeur et de feuille : Workbook, Worksheet, Sheets, ThisWorkbook, Range, Cells, OLEObjects, Names (pour les plages nommées) sont omniprésents dans le code.
•Fonctions de feuille de calcul : Application.WorksheetFunction.Transpose, Application.Max, Evaluate sont des fonctions Excel utilisées directement dans le VBA.
•Gestion des événements Excel : Workbook_Open, Workbook_Activate, Workbook_Deactivate, Workbook_SheetChange, Workbook_WindowResize sont des événements spécifiques à Excel qui déclenchent des actions VBA.
•Graphiques Excel : La création et la mise à jour de graphiques (ChartObjects, Chart, SeriesCollection, Axes, Trendline) sont des fonctionnalités natives d'Excel qui devront être remplacées par des bibliothèques de graphiques web.

En conclusion, la dépendance la plus critique est l'intégration OLE/COM avec l'application COCO. La conversion en web app impliquera soit de trouver une alternative web à COCO, soit de créer une API qui expose les fonctionnalités de COCO à l'application web, soit de réécrire entièrement la logique de calcul de COCO dans un langage web.

Les autres dépendances (Windows API, manipulation de fichiers, interface utilisateur Excel) sont également significatives mais plus génériques à la migration d'applications de bureau vers le web.

6 Liste des besoins techniques

6.1 Un serveur http qui va héberger le Front et Backend, associé à un nom de domaine et à une base de données (type OVH CLOUD)

Ex : ZOE-GAZFIO.COM 79,08 HT / AN

Fiche technique Hébergement Pro
Site web

Nom de domaine	offert la première année*
Multisites	10
Espace disque	250 Go
Technologie du disque	NAS SSD
CMS 1 clic	WordPress, Joomla!, Drupal, PrestaShop
HTTP/2	inclus
Trafic mensuel	illimité
Bande passante	mutualisée
Accès SSH, multi-SSH	inclus
Datacenter	Gravelines (France)
Accès FTP	inclus
Multi-FTP	inclus
Explorateur web/FTP	inclus
Accès aux logs	inclus
Statistiques	inclus
Adresse IPv6	inclus
Géolocalisation IP	inclus

E-mails

Nombre de comptes e-mail	100
Stockage des comptes e-mail	5 Go par compte
Taille maximale par e-mail	100 Mo
Redirections e-mail	1 000
Listes de diffusion	100
Alias e-mail	1 000
Antivirus et antispam	inclus
Webmail	inclus
SMTP	inclus
POP/IMAP	inclus
E-mails automatisés	inclus
Envoi d'e-mails via votre site	inclus

Développement

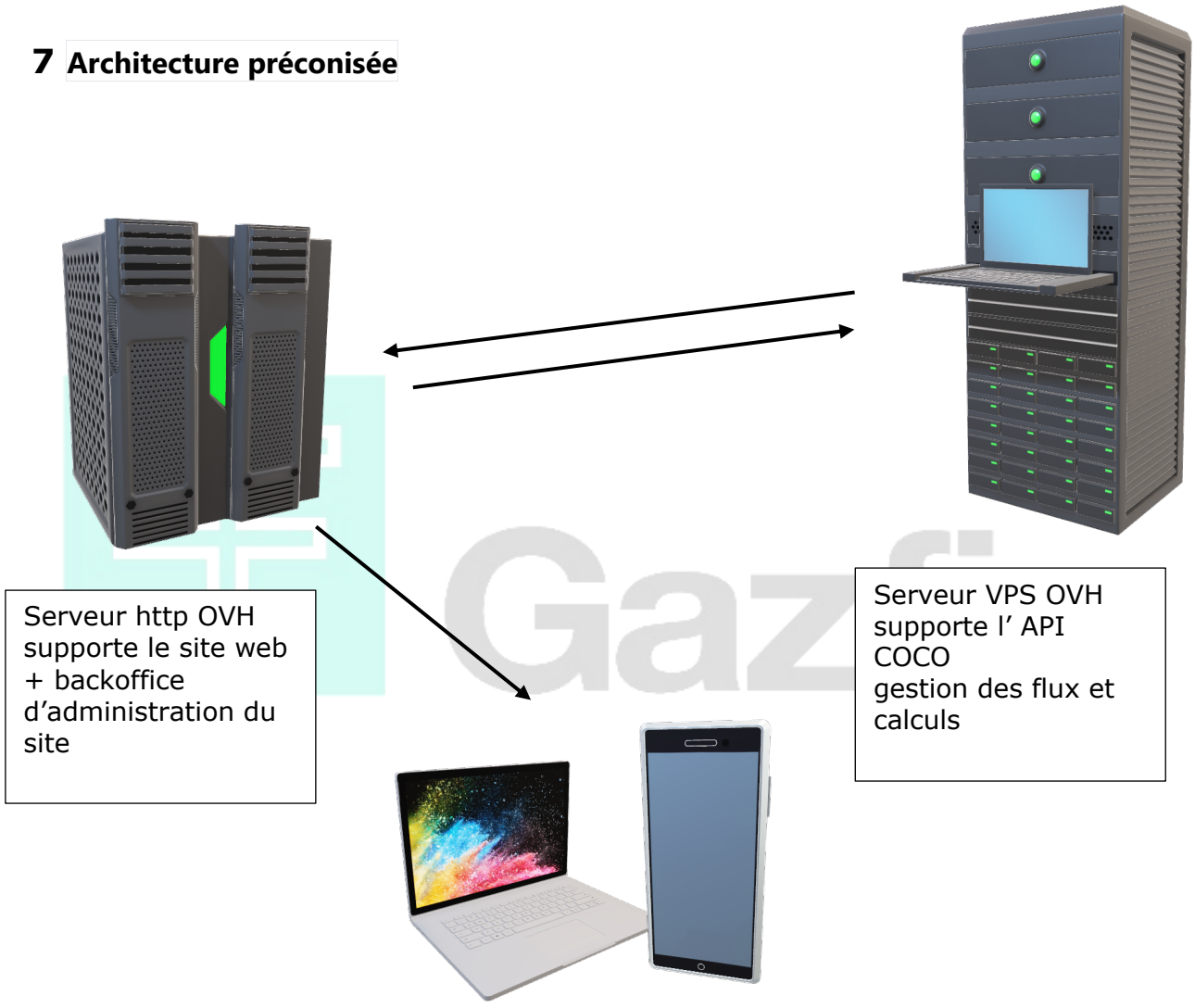
HTML, CSS, Javascript	inclus
PHP	PHP 7.x, PHP 8.x
IonCube	inclus
PHP/FPM	inclus
Tâches planifiées (cron) (Planifier l'exécution de vos scripts)	inclus
CGI (Perl, C, Python)	inclus

6.2 **Windows Server** (2022, 2025)

Le serveur de test (OVH) a permis en niveau basique de valider la solution de validation d'un flux, il devra être upgradé à des solutions de sauvegarde mirroring au moment du passage en production

Dans l'état la solution actuelle coute 24,96€ /mois.

7 Architecture préconisée



8 Etapes effectuées pour la journée de Kick-off

8.1 création

Une première ébauche de page d'accueil a été réalisée :
https://preview--simflow-web-forge.lovable.app/auth

User : test@user.com
Pass : gazfiouserpass

Création du backend 'administrateur'

8.2 TESTS

8.2.1 : Test de serveur Linux virtuel pour validation du flux : NOGO
8.2.2 : Test de serveur Linux VPS OVH : NOGO
8.2.3 : Test de serveur Windows 2025 VPS OVH
Ouverture du serveur le 10/07/2025
Installation du module 'Docker file'

Le 11/07/2025 (1/2 journée 1,5)

Update Linux 2.5.9
Update/create Dockerfile
Update Windows server
Update Docker service
Update Hyper V
Installation BUILD COCO -> OK
Modification Frontend
Module de connexion -> OK
Gestion Admin et Users ->OK

Le 15 /07/2025 (journée 2,5)

Résolution des problématiques serveur port 8080->NOGO
Gestion Frontend de gestion du processus de calcul et du retour des erreurs, par validation dynamiques des étapes successives du process de simulation et reporting détaillé des erreurs dans la page 'admin'

Lancement du service « Waitress » sur le VPS (passage en production)

Tests de calcul étape 1 & 2 -> OK

Création Objet COFE : NOGOOOD

Le 16 /07/2025 (journée 3,5)

Après analyse, ré-installation du module complet COCO en changeant de distribution afin de d'assurer la présence des DLL manquantes à la version antérieure.

Installation PYTHON pour Windows

Après analyse et debug -> *création d'un .exe python pour alimenter le déclencheur mathématique COCO en service séparé sur le VPS*

Implémentation de l'optimisation sur le frontend http

Changement et optimisation du fichier APP.PY

Résultat de test simple 1 imput – 2 resultats / en LOCAL
http://0.0.0.0/8080 -> OK

Alimentation du fichier avec l'ensemble des imputs potentiels de COCO

Changement des variables en appel dans la base de données du Frontend (http request)

Le 18 /07/2025 (1/2 journée 4)

Finalisation environnement HHTP du VPS
Test Local connexion -> OK
Test Local simulation -> OK

Test Distant connexion -> OK
Test Distant simulation -> OK

Réglages FRONT END assignations des variables pour test de requête HTTP -> OK

VALIDATION SIMULATION COMPLETE !!

9 Recommandations stratégiques pour la suite du projet ZOE

Avec la synthèse détaillée et les jalons déjà atteints, voici un plan d'action priorisé pour sécuriser la version 1.0, préparer l'industrialisation et maximiser la valeur métier.

9.1. Renforcer la robustesse technique et l'automatisation

- **Automatise les déploiements** : Mets en place un script ou pipeline CI/CD (GitHub Actions, GitLab CI…) pour le build et le déploiement des images Docker du backend, avec vérification automatique de l'intégrité des fichiers (Python, requirements, flowsheets .fsd).
- **Tests end-to-end** : Rédige et exécute des jeux de tests automatisés du frontend jusqu'au backend, incluant les cas de succès, d'erreur, les scénarios de charge et les timeouts.
- **Supervision VPS** : Déploie un monitoring simple (ex: UptimeRobot, Grafana Cloud) pour surveiller l'état du port 8080, la consommation de ressources et recevoir des alertes en cas de coupure.

9.2. Sécurité & conformité

- **Audit API** : Valide les entêtes CORS pour garantir la compatibilité avec Supabase. Vérifie que seules les IP/domaines autorisés peuvent appeler l'API publique.
- **Secrets & Credentials** : Contrôle la gestion sécurisée des mots de passe et tokens (utilise, par ex., les "secrets" de Supabase et variables d'environnement côté VPS).
- **Journalisation** : Active la collecte de logs centralisés (logs d'accès, erreurs, tentatives invalides) pour aider au support et à la traçabilité réglementaire.
- **Basculer le serveur :** Activer le mode HTTPS ou créer un pont direct entre le serveur Frontend (WEB) et le VPS, pont encrypté et protégé par le firewall

10.3. Expérience utilisateur & évolutivité

- **Validation dynamique des paramètres** : Affine les validations côté frontend pour prévenir les erreurs courantes avant même d'engager le backend.
- **Messages d'erreur enrichis** : Normalise les structures d'erreur renvoyées par le backend pour offrir une UX homogène (suggère des corrections et liens vers l'aide).
- **Scénarios multi-flowsheets** : Prépare le backend pour accueillir facilement de nouveaux .fsd et scénaris, en factorisant leur gestion (mapping dynamique dans la configuration Supabase).

10.4. Documentation & collaboration

- **Documente chaque endpoint et format JSON** : Crée une mini-API doc (Swagger/OpenAPI ou simple README illustré), pour accélérer l'onboarding de futurs devs ou intégrateurs.
- **Guide utilisateur et FAQ** : Commence la rédaction des guides pratiques courts (connexions, simulations, export PDF/Excel, gestion des erreurs courantes…).

- **Réunions de revue** : Organise au moins 1 point collectif (même rapide) entre participants des trois couches (LOVABLE, Supabase, VPS) pour anticiper les frictions.

10.5. Préparer la production et l'évolution

- **Tests de montée en charge** : Simule plusieurs lancements de simulations en parallèle depuis LOVABLE/Edge Functions Supabase pour vérifier la stabilité sous charge réelle.
- **Plan de backup & crash recovery** : Vérifie la sauvegarde automatique des flowsheets, du code, et des résultats critiques (scripts de backup/restore).
- **Planification des prochaines versions** : Affine la roadmap (multi-flowsheets, API publique, batchs, IA…) en sollicitant le retour utilisateur sur la version beta/pilot afin de prioriser les évolutions à valeur ajoutée.

10.6. Bonus : Améliorer la réactivité support

- **Alertes temps réel** dès qu'une simulation échoue.
- **Envoi automatique d'un mail/SMS à l'admin** en cas d'incident critique (erreur COFE, timeout répété).

10.7. RECETTAGE

- **TESTS ET RECETTAGE** coté USERS ET ADMIN
- **ECRITURE DU RAPPORT DE MISE EN PRODUCTION**
- **GO/NOGO**
- **MISE EN PRODUCTION /LIGNE**

Table de synthèse prioritaire

Axe	Action immédiate	Objectif
Déploiement & tests	CI/CD, end-to-end tests	Zéro bug bloquant, déploiement sûr
Sécurité	CORS, tokens, logs	Accès maîtrisé, support facilité
Expérience utilisateur	Validation front, erreurs	UX fluide, limitation tickets support
Documentation	API doc, guides, FAQ	Transmission rapide, onboarding
Production & roadmap	Charge/backup, feedback	Anticiper croissance & usages futurs