

DOCUMENT DE SPÉCIFICATIONS FONCTIONNELLES, TECHNIQUES ET EXPÉRIENTIELLES DÉTAILLÉES

PROJET : Plateforme Web de Gestion Intégrée COMPTA FREELANCE (SYSCohada) - Version Complète

FOCUS : Couverture Exhaustive : Flux Utilisateur, Configuration, Fonctionnalités, UI/UX, Technique

VERSION : 2.1 (Workflow-Centric)

DATE : 25 mai 2024

RÉSUMÉ : Ce document fournit une description exhaustive et centrée sur l'utilisateur de la plateforme web ERP interne pour COMPTA FREELANCE. Il détaille les flux de travail typiques des collaborateurs, la configuration initiale du système et des dossiers clients, la gestion des utilisateurs, les spécifications détaillées de chaque module fonctionnel (conformité SYSCohada) en liant les fonctionnalités aux étapes du workflow, les exigences UI/UX détaillées découlant de ces flux (inspirées du mix Odoo/Sage, utilisant HTMX/Alpine.js), l'architecture technique (Django), les exigences non fonctionnelles, et les considérations de déploiement. Ce document sert de référence principale pour le développement en mettant l'accent sur l'expérience utilisateur finale.

TABLE DES MATIÈRES

1. Introduction et Contexte Général

- 1.1. Objectif Final du Produit
- 1.2. Portée Détaillée (Inclusions & Exclusions Claires)
- 1.3. Public Cible et Personas Utilisateurs
 - 1.3.1. Administrateur Système / Superviseur
 - 1.3.2. Comptable / Gestionnaire de Dossier
 - 1.3.3. Assistant Comptable
- 1.4. Définitions, Acronymes et Références (Incluant SYSCohada)
- 1.5. Environnement de Développement (Rappel Codespaces)

2. Flux Utilisateur Global et Navigation Principale

- 2.1. Vision d'Ensemble du Parcours Utilisateur
- 2.2. Éléments de Navigation Globale (UI/UX)
 - 2.2.1. Barre Latérale Gauche (Navigation Principale)
 - 2.2.2. Barre Supérieure (Contexte et Actions Globales)
 - 2.2.3. Fil d'Ariane (Breadcrumbs)

3. Configuration Initiale du Système (Flux Administrateur)

- 3.1. Flux : Première Connexion et Configuration Globale
- 3.2. Flux : Crédit des Rôles et des Premiers Utilisateurs

4. Gestion des Dossiers Clients (Onboarding Client - Flux Administrateur/Superviseur)

4.1. Flux : Création d'un Nouveau Dossier Client

4.1.1. Accès et Lancement

4.1.2. Étape 1 : Informations Générales Client (Sélection/Création Tiers)

4.1.3. Étape 2 : Paramètres Comptables Initiaux (Exercice, Plan, Journaux, TVA, Séquences)

4.1.4. Étape 3 : Assignation Utilisateurs

4.1.5. Finalisation et Feedback

4.2. Flux : Configuration/Modification d'un Dossier Existant

4.3. Flux : Sélection du Dossier Actif par l'Utilisateur (Tous Rôles)

5. Philosophie Visuelle & Expérience Utilisateur (UI/UX) Détaillée

5.1. Vision Globale : Synthèse Odoo/Sage pour Efficacité et Modernité Web

5.2. Grille de Mise en Page et Responsive Design (Objectifs Détaillés)

5.3. Palette de Couleurs, Typographie, Iconographie (Spécifications)

5.4. Composants d'Interface Standards (Design System Interne)

5.4.1. Boutons

5.4.2. Champs de Formulaire

5.4.3. Tableaux de Données

5.4.4. Cartes (Cards) / Widgets

5.4.5. Modales / Pop-ups

5.4.6. Notifications / Messages Toast / Alertes Contextuelles

5.4.7. Indicateurs de Chargement / Progression

5.5. Principes de Navigation Globale et Contextuelle

5.6. Accessibilité (Considérations de base)

6. Module Cœur Comptable (SYSCohada) - Flux et Spécifications

6.1. Flux : Paramétriser la Comptabilité d'un Dossier Client (Admin/Sup)

6.1.1. Gestion des Exercices Comptables (UI et Logique)

6.1.2. Gestion des Journaux Comptables (UI et Logique)

6.1.3. Gestion du Plan Comptable Spécifique (Consultation, Auxiliaires)

6.2. Flux : Saisir une Écriture Manuelle (Assistant/Comptable)

6.2.1. Accès et Sélection Journal

6.2.2. Interface de Saisie Détaillée (Layout, En-tête, Lignes)

6.2.3. Interaction Lignes (Ajout/Suppression/Validation Compte - HTMX/Alpine)

6.2.4. Calcul et Affichage Solde Temps Réel (HTMX/Alpine)

6.2.5. Workflow de Validation et Sauvegarde (HTMX/Django)

6.3. Flux : Consulter le Grand Livre d'un Compte (Tous Rôles avec Accès)

6.3.1. Accès et Filtres (Compte, Période)

6.3.2. Affichage des Résultats (Tableau GL - HTMX)

6.3.3. Options d'Export

6.4. Flux : Consulter une Balance (Générale/Auxiliaire) (Comptable/Sup)

6.4.1. Accès et Filtres (Période, Type, Niveau)

6.4.2. Affichage des Résultats (Tableau Balance - HTMX)

6.4.3. Options d'Export

6.5. Flux : Lettrer un Compte Tiers (Assistant+/Comptable)

6.5.1. Accès et Sélection Compte Tiers

6.5.2. Interface de Lettrage (Tableau Non Lettrés, Zone Sélection)

6.5.3. Workflow de Sélection et Validation (HTMX/Alpine/Django)

7. Module Gestion des Tiers - Flux et Spécifications

- 7.1. Flux : Consulter la Liste des Tiers (Tous Rôles avec Accès)
- 7.2. Flux : Créer un Nouveau Tiers (Assistant+/Comptable/Sup)
- 7.3. Flux : Modifier un Tiers Existant
- 7.4. Flux : Consulter la Fiche Détail d'un Tiers (Vue 360°)

8. Module Gestion des Ventes - Flux et Spécifications

- 8.1. Flux : Consulter la Liste des Factures Ventes (Assistant+/Comptable/Sup)
- 8.2. Flux : Créer une Facture de Vente (Assistant+/Comptable)
 - 8.2.1. Initialisation et En-tête
 - 8.2.2. Ajout/Gestion des Lignes de Facture (Dynamique HTMX/Alpine)
 - 8.2.3. Validation et Génération Écriture Comptable
- 8.3. Flux : Modifier une Facture (si Brouillon)
- 8.4. Flux : Visualiser une Facture Validée (Aperçu/PDF)
- 8.5. Flux : Enregistrer un Encaissement Client (Assistant+/Comptable)
- 8.6. Flux : Affecter/Letturer un Encaissement à une/des Facture(s)
- 8.7. Flux : Gérer un Avoir/Note de Crédit Vente

9. Module Gestion des Achats - Flux et Spécifications

- 9.1. Flux : Consulter la Liste des Factures Achats (Assistant+/Comptable/Sup)
- 9.2. Flux : Saisir une Facture d'Achat (Assistant/Comptable)
 - 9.2.1. Saisie des Informations Fournisseur et Facture
 - 9.2.2. Ajout/Gestion des Lignes d'Achat (Imputation Charges/TVA)
 - 9.2.3. Validation et Génération Écriture Comptable
- 9.3. Flux : Suivre les Échéances Fournisseurs
- 9.4. Flux : Enregistrer un Paiement Fournisseur (Assistant+/Comptable)
- 9.5. Flux : Affecter/Letturer un Paiement à une/des Facture(s)
- 9.6. Flux : Gérer un Avoir/Note de Crédit Achat

10. Module Gestion des Stocks - Flux et Spécifications (Simplifié)

- 10.1. Flux : Gérer le Catalogue Articles (CRUD Articles)
- 10.2. Flux : Saisir un Mouvement de Stock Manuel
- 10.3. Flux : Consulter l'État des Stocks
- 10.4. (Optionnel) Flux : Intégration avec Achats/Ventes
- 10.5. (Optionnel) Flux : Réaliser un Inventaire

11. Module Gestion de Trésorerie - Flux et Spécifications

- 11.1. Flux : Paramétrier les Comptes de Trésorerie (Admin/Sup)
- 11.2. Flux : Effectuer un Rapprochement Bancaire (Comptable)
 - 11.2.1. Sélection Compte et Période
 - 11.2.2. Interface de Pointage (Comparaison Compta/Relevé)
 - 11.2.3. Validation et Génération État de Rapprochement
- 11.3. Flux : Consulter les Soldes et Mouvements de Trésorerie

12. Module Gestion Budgétaire - Flux et Spécifications (Simplifié)

- 12.1. Flux : Définir un Budget (Comptable/Sup)
- 12.2. Flux : Consulter le Suivi Budgétaire (Comparatif Réel/Prévu)

13. Module Reporting et États Financiers - Flux et Spécifications

- 13.1. Flux : Générer un État Financier SYSCohada (TAFIRE) (Comptable/Sup)

- 13.2. Interface d'Affichage des États Générés
- 13.3. Flux : Exporter un Rapport (PDF/CSV/Excel)

14. Import / Export de Données - Flux et Spécifications

- 14.1. Flux : Importer des Données Initiales (Admin/Sup)
- 14.1.1. Import Plan Comptable (si besoin)
- 14.1.2. Import Tiers
- 14.1.3. Import Balance d'Ouverture
- 14.2. Fonctionnalités d'Export de Données

15. Exigences Non Fonctionnelles - Approfondissement

- 15.1. Sécurité Détailée (Authentification, Autorisations, Protections OWASP...)
- 15.2. Performance et Scalabilité (Objectifs Temps Réponse, Optimisations DB/Cache...)
- 15.3. Maintenabilité et Qualité du Code (Conventions, Modularité, Tests Automatisés...)
- 15.4. Fiabilité et Disponibilité (Gestion Erreurs, Transactions DB, Objectifs Uptime...)

16. Déploiement et Exploitation - Considérations

- 16.1. Infrastructure Cible (Production) (Serveur, Nginx, Gunicorn, DB...)
- 16.2. Processus de Déploiement (Automatisation, CI/CD...)
- 16.3. Sauvegarde et Restauration (Stratégies DB, Code, Fichiers)
- 16.4. Monitoring et Logging (Outils Applicatifs, Système, Centralisation)

17. Conclusion et Prochaines Étapes

- 17.1. Résumé des Objectifs Atteints par cette Spécification
- 17.2. Facteurs Clés de Succès
- 17.3. Prochaines Étapes Logiques (Validation, Priorisation MVP, Développement...)

18. Annexes (Description du Contenu)

- 18.1. Annexe A : Plan Comptable SYSCohada Révisé de Référence (Description du contenu attendu)
 - 18.2. Annexe B : Wireframes / Maquettes Visuelles Détaillées (Description du contenu attendu)
 - 18.3. Annexe C : Modèle de Données Physique (Description du contenu attendu)
 - 18.4. Annexe D : Règles de Validation Métier SYSCohada Détaillées (Description du contenu attendu)
-

1. Introduction et Contexte Général

1.1. Objectif Final du Produit

L'objectif principal de ce projet est de développer une application web interne complète et robuste, désignée comme la "Plateforme Web de Gestion Intégrée COMPTA FREELANCE". Cette plateforme servira d'outil centralisé pour les collaborateurs de l'entreprise COMPTA FREELANCE, leur permettant de gérer l'intégralité des prestations comptables et financières fournies à leurs divers clients. Un impératif absolu et une caractéristique fondamentale du produit est l'adhésion stricte et la conformité démontrable aux normes et principes du Système Comptable

OHADA Révisé (SYSCohada Révisé) pour toutes les opérations, processus et rapports générés. L'application vise également à rationaliser les flux de travail internes, améliorer l'efficacité opérationnelle, renforcer la collaboration entre les collaborateurs, garantir la traçabilité des informations et, in fine, éléver la qualité globale des services rendus par COMPTA FREELANCE.

1.2. Portée Détaillée (Inclusions & Exclusions Claires)

Inclusions :

- Le développement complet d'une application web accessible via les navigateurs web modernes standards.
- L'implémentation d'un système d'authentification sécurisé et d'une gestion fine des droits basée sur les rôles pour les utilisateurs internes (collaborateurs COMPTA FREELANCE).
- La mise en œuvre d'une gestion multi-dossiers permettant de séparer et de gérer les données de chaque client de COMPTA FREELANCE de manière étanche.
- Le développement des modules fonctionnels suivants, avec conformité SYSCohada intégrée :
 - Cœur Comptable (Plan comptable, journaux, saisie écritures, consultations GL/Balances, lettrage).
 - Gestion des Tiers (Clients et Fournisseurs).
 - Gestion du Cycle de Vente (Facturation client, suivi encaissements).
 - Gestion du Cycle d'Achat (Saisie factures fournisseurs, suivi paiements).
 - Gestion des Stocks (Fonctionnalité de base : articles, mouvements, quantités).
 - Gestion de Trésorerie (Suivi comptes Banque/Caisse, aide au rapprochement).
 - Gestion Budgétaire (Saisie prévisions, suivi simplifié).
 - Reporting (Génération états comptables et financiers SYSCohada TAFIRE de base).
- La conception et l'implémentation d'une interface utilisateur (UI) et d'une expérience utilisateur (UX) s'inspirant d'un mélange Odoo/Sage, privilégiant la clarté, l'efficacité et la réactivité (via HTMX/Alpine.js).
- La mise en place de fonctionnalités d'administration système de base (gestion utilisateurs, configuration dossiers clients).
- Le développement sera effectué dans l'environnement GitHub Codespaces en utilisant la stack technologique définie (Python/Django/HTMX/Alpine.js/SQLite-PostgreSQL).
- La production de tests automatisés (unitaires et intégration) pour les logiques métier critiques.

Exclusions :

- Un portail ou espace client accessible par les clients finaux de COMPTA FREELANCE.
- Des fonctionnalités avancées de CRM (Gestion de la Relation Client) au-delà de la gestion de base des tiers.
- Des modules de gestion de projet interne, de gestion du temps ou de

ressources humaines pour COMPTA FREELANCE.

- Des fonctionnalités de paie.
- Des intégrations automatisées complexes avec des systèmes externes (ex: API bancaires pour relevés automatiques, connexion à d'autres logiciels via API tierces). L'import/export manuel de fichiers (CSV/Excel) sera privilégié pour les échanges de données initiaux.
- Des fonctionnalités analytiques ou de Business Intelligence très poussées au-delà des rapports standards définis.
- La migration automatisée des données historiques depuis d'éventuels systèmes existants (seules des fonctionnalités d'import initial pour les données de base seront prévues).
- Une application mobile native distincte (l'accès se fait via le navigateur web).

1.3. Public Cible et Personas Utilisateurs

L'application est conçue exclusivement pour les collaborateurs internes de COMPTA FREELANCE. Trois personas principaux sont identifiés :

- **1.3.1. Administrateur Système / Superviseur :**
 - *Profil* : Responsable de la plateforme, potentiellement un associé ou un responsable informatique interne. Possède une vue d'ensemble des opérations et des droits maximums.
 - *Besoins / Objectifs* : Configurer le système global, gérer les utilisateurs et leurs accès, créer et configurer les dossiers clients, superviser l'activité générale, accéder à tous les modules et rapports, assurer la maintenance de base des données de référence.
 - *Compétences Techniques* : Bonne compréhension fonctionnelle de l'ensemble de l'application, à l'aise avec les tâches d'administration.
- **1.3.2. Comptable / Gestionnaire de Dossier :**
 - *Profil* : Utilisateur principal de l'application. Gère un portefeuille de dossiers clients. Maîtrise la comptabilité et SYSCohada.
 - *Besoins / Objectifs* : Effectuer l'ensemble des tâches comptables pour ses clients (saisie, contrôle, lettrage, rapprochement), générer les factures de vente, suivre les achats, préparer les déclarations de TVA, générer les états financiers, analyser les données, répondre aux demandes des clients (via les données de l'outil). Recherche l'efficacité, la fiabilité et la conformité.
 - *Compétences Techniques* : Très bonne maîtrise des outils comptables, peut être familier avec Sage ou Odoo. S'adaptera à l'interface web si elle est efficace.
- **1.3.3. Assistant Comptable :**
 - *Profil* : Effectue principalement des tâches de saisie de données sous la supervision d'un comptable.
 - *Besoins / Objectifs* : Saisir rapidement et précisément les écritures (achats, ventes, trésorerie), enregistrer les pièces, effectuer des tâches de classement ou de préparation simples. Nécessite une interface de saisie très claire et rapide.
 - *Compétences Techniques* : Maîtrise la saisie de données, bonne connaissance des pièces comptables. Moins de besoins en analyse ou reporting avancé. Accès potentiellement limité à certains modules ou

actions.

1.4. Définitions, Acronymes et Références

- **API** : Application Programming Interface / Interface de Programmation Applicative.
- **CRUD** : Create, Read, Update, Delete - Opérations de base sur les données.
- **CSS** : Cascading Style Sheets - Langage de style pour le web.
- **CSV** : Comma-Separated Values - Format de fichier texte pour les données tabulaires.
- **Django** : Framework web de haut niveau en Python utilisé pour le backend.
- **DTL** : Django Template Language - Système de templating de Django.
- **ERP** : Enterprise Resource Planning / Logiciel de Gestion Intégré.
- **Git** : Système de contrôle de version distribué.
- **GitHub Codespaces** : Environnement de développement basé sur le cloud.
- **HTML** : HyperText Markup Language - Langage de structure des pages web.
- **HTMX** : Bibliothèque JavaScript permettant des interactions AJAX et des mises à jour partielles de page directement depuis les attributs HTML.
- **HTTP(S)** : Hypertext Transfer Protocol (Secure) - Protocole de communication web.
- **IDE** : Integrated Development Environment / Environnement de Développement Intégré.
- **JavaScript (JS)** : Langage de programmation côté client (et serveur via Node.js).
- **JSON** : JavaScript Object Notation - Format d'échange de données léger.
- **LTS** : Long-Term Support - Version d'un logiciel bénéficiant d'un support étendu.
- **MPA** : Multi-Page Application - Application web traditionnelle où la navigation charge de nouvelles pages.
- **MVP** : Minimum Viable Product / Produit Minimum Viable.
- **MySQL** : Système de Gestion de Base de Données Relationnelle populaire.
- **Nginx** : Serveur web et reverse proxy populaire.
- **OHADA** : Organisation pour l'Harmonisation en Afrique du Droit des Affaires.
- **ORM** : Object-Relational Mapper - Outil permettant de manipuler la base de données via des objets (ex: l'ORM Django).
- **OWASP** : Open Web Application Security Project - Organisation travaillant sur la sécurité web.
- **PDF** : Portable Document Format - Format de fichier pour documents.
- **PEP 8** : Guide de style officiel pour le code Python.
- **PostgreSQL (ou Postgres)** : Système de Gestion de Base de Données Relationnelle open-source avancé.
- **REST(ful)** : Representational State Transfer - Style d'architecture pour les APIs web.
- **SQL** : Structured Query Language - Langage de requête pour bases de données relationnelles.
- **SQLite** : Moteur de base de données relationnelle simple, basé sur fichier.
- **SSD** : Software Specification Document (ce document).
- **SSR** : Server-Side Rendering - Rendu HTML effectué côté serveur.
- **SSL/TLS** : Secure Sockets Layer / Transport Layer Security - Protocoles de chiffrement pour sécuriser les communications (utilisé dans HTTPS).

- **SYSCohada Révisé** : Système Comptable OHADA Révisé - Le référentiel comptable obligatoire pour ce projet. Fait référence à l'Acte Uniforme relatif au Droit Comptable et à l'Information Financière révisé.
- **TAFIRE** : Tableau Financier et Récapitulatif des États - Format standardisé des états financiers SYSCohada (Bilan, Résultat, Flux de Trésorerie, Notes Annexes).
- **UI** : User Interface / Interface Utilisateur.
- **UX** : User Experience / Expérience Utilisateur.
- **WSGI** : Web Server Gateway Interface - Interface standard entre les serveurs web et les applications Python.

1.5. Environnement de Développement (Rappel Codespaces)

Le développement de l'application sera entièrement réalisé au sein de l'environnement de développement cloud GitHub Codespaces. Cet environnement fournit un conteneur Linux avec un éditeur VS Code accessible via navigateur web. Cela implique un accès potentiel depuis divers appareils, le test de l'application web via lancement du serveur de développement Django et accès à l'URL générée par le port forwarding. Les limitations potentielles de cet environnement ont influencé le choix d'une stack technologique web (Django+HTMX+Alpine).

2. Flux Utilisateur Global et Navigation Principale

2.1. Vision d'Ensemble du Parcours Utilisateur

Un collaborateur de COMPTA FREELANCE interagira typiquement avec la plateforme en suivant un schéma général :

1. **Accès et Authentification** : Connexion sécurisée à la plateforme via un écran de login dédié.
2. **Sélection du Contexte de Travail** : Choix explicite du dossier client spécifique sur lequel les actions vont porter. Ce contexte doit être clairement visible et modifiable à tout moment.
3. **Navigation vers un Module Fonctionnel** : Utilisation des éléments de navigation principaux (barre latérale) pour accéder à la section fonctionnelle désirée (Comptabilité, Ventes, Achats, Tiers, Reporting, etc.).
4. **Exécution de Tâches Spécifiques** : Interaction avec les écrans du module choisi pour réaliser des actions précises : consultation de listes, remplissage de formulaires de saisie, déclenchement de processus (validation, génération), consultation de détails. L'interface doit guider l'utilisateur à travers les étapes logiques de chaque tâche.
5. **Changement de Contexte (Optionnel)** : Possibilité de basculer facilement vers un autre dossier client via le sélecteur global, ou de naviguer vers un autre module fonctionnel via la barre latérale.
6. **Déconnexion Sécurisée** : Clôture de la session de travail via une action explicite.

L'interface utilisateur et l'architecture technique sont conçues pour rendre ce

parcours fluide, logique et efficace, en minimisant les rechargements de page inutiles grâce à HTMX pour les interactions clés.

2.2. Éléments de Navigation Globale (UI/UX)

Ces éléments sont présents sur quasiment toutes les pages après la connexion et assurent la cohérence de la navigation et la perception du contexte.

- **2.2.1. Barre Latérale Gauche (Navigation Principale) :**
 - *Visuel* : Barre verticale fixe sur le côté gauche de l'écran (sur les écrans larges/moyens), potentiellement repliable en une barre d'icônes via un bouton dédié. Sur mobile, elle est cachée par défaut et s'ouvre via un bouton "burger". Contient une liste verticale d'items, chacun composé d'une icône et d'un libellé texte clair.
 - *Contenu* : Logo COMPTA FREELANCE (ou nom) en haut. Liens vers les modules principaux accessibles à l'utilisateur selon ses droits : Tableau de Bord, Comptabilité (peut se déplier en : Saisie, Grand Livre, Balance, Lettrage...), Tiers (peut se déplier en : Clients, Fournisseurs), Ventes (Factures, Encaissements...), Achats (Factures, Paiements...), Stocks (Articles, Mouvements...), Trésorerie (Rapprochement...), Budget, Reporting, Paramètres (si droits suffisants).
 - *Interaction* : L'item correspondant à la page/module actif est visuellement mis en évidence (fond différent, couleur de texte/icône primaire). Cliquer sur un item charge la vue principale de ce module dans la zone de contenu à droite. Cliquer sur un sous-item (s'il existe) charge directement cette sous-section.
- **2.2.2. Barre Supérieure (Contexte et Actions Globales) :**
 - *Visuel* : Barre horizontale fixe en haut de la page, au-dessus de la zone de contenu principal et à droite de la barre latérale (si dépliée).
 - *Contenu (Gauche à Droite)* :
 - Bouton "Burger" ou icône pour afficher/masquer la barre latérale sur les écrans où elle est repliable/cachée.
 - **Sélecteur de Dossier Client Actif** : Affiche clairement le nom du dossier client actuellement sélectionné (ex: "Dossier: Société ABC SARL"). Cliquer sur ce nom ouvre un menu déroulant ou une modale permettant de rechercher et de sélectionner un autre dossier parmi ceux auxquels l'utilisateur a accès. Cette sélection est une action critique qui met à jour le contexte de toute l'application.
 - **Menu Utilisateur/Profil** : Affiche le nom de l'utilisateur connecté. Cliquer dessus ouvre un petit menu déroulant avec des options comme "Mon Profil" (si applicable), "Paramètres Utilisateur" (si applicable), et impérativement "Se Déconnecter".
 - *Interaction* : Le changement de dossier client via le sélecteur doit recharger les données affichées dans la zone de contenu principal pour refléter le nouveau contexte, idéalement sans rechargement complet de page si la structure le permet (possible via HTMX sur le conteneur principal ou via rechargement classique si plus simple).
- **2.2.3. Fil d'Ariane (Breadcrumbs) :**
 - *Visuel* : Ligne de texte située directement sous la Barre Supérieure, au

- début de la Zone de Contenu Principal. Affiche le chemin hiérarchique de la page actuelle.
- o *Exemple* : Accueil > Comptabilité > Journaux de Saisie > Journal des Achats (AC)
 - o *Interaction* : Chaque segment du chemin (sauf le dernier, qui est la page actuelle) est un lien hypertexte permettant à l'utilisateur de remonter facilement aux niveaux supérieurs de la navigation. Il est généré dynamiquement par chaque vue Django en fonction de sa position dans l'arborescence de l'application.
-

3. Configuration Initiale du Système (Flux Administrateur)

Ces flux décrivent les étapes nécessaires pour configurer l'application après sa première installation, actions généralement réservées aux utilisateurs avec les droits d'administrateur système ou de superviseur.

3.1. Flux : Première Connexion et Configuration Globale

1. **Prérequis** : L'application a été déployée, les migrations initiales Django appliquées, et un premier compte super-utilisateur a été créé via la commande `python manage.py createsuperuser`.
2. **Action** : L'administrateur accède à l'URL de l'application et se connecte avec les identifiants du super-utilisateur.
3. **Interface** : Après connexion, il est probable qu'il soit redirigé vers une page d'accueil minimale ou directement vers la section d'administration (si aucune configuration n'est détectée). La sélection de dossier client n'est pas pertinente à ce stade.
4. **Action** : L'administrateur navigue vers la section dédiée aux Paramètres Généraux du Système (via la barre latérale "Paramètres" > "Configuration Globale" ou via l'interface /admin/ de Django).
5. **Action** : Dans la sous-section "Informations Entreprise", il remplit le formulaire avec les données de COMPTA FREELANCE (Nom, Adresse, N° Fiscal, etc.) et télécharge le logo si l'option est disponible. Il enregistre les modifications.
6. **Feedback** : Message confirmant l'enregistrement des informations globales.
7. **Action** : Il vérifie/configure les Paramètres de Sécurité Globaux (via `settings.py` principalement pour les validateurs de mot de passe).
8. **Action** : Il configure les paramètres SMTP dans `settings.py` ou les variables d'environnement pour permettre l'envoi d'emails. Il peut exister une option dans l'interface admin pour envoyer un email de test afin de valider la configuration.
9. **Action** : Il accède à la section (potentiellement dans l'Admin Django) pour visualiser le Plan Comptable SYSCohada standard et confirme qu'il a été correctement chargé par les migrations/fixtures initiales. Il note qu'il ne peut pas modifier ce plan standard directement.

3.2. Flux : Création des Rôles et des Premiers Utilisateurs

1. **Prérequis** : L'administrateur est connecté.
2. **Action** : L'administrateur accède à l'interface d'administration Django

- (/admin/auth/group/).
- 3. **Interface** : Liste des groupes existants (initialement vide ou avec des groupes par défaut). Bouton "Ajouter Groupe".
 - 4. **Action** : Clique "Ajouter Groupe". Saisit "Administrateurs" comme nom. Sélectionne **toutes** les permissions disponibles (ou un sous-ensemble pertinent pour les admins non-superusers). Enregistre.
 - 5. **Action** : Répète l'étape 4 pour créer le groupe "Comptables", en sélectionnant soigneusement les permissions nécessaires pour la gestion complète des dossiers clients (CRUD sur écritures, factures, tiers, accès aux rapports, etc., mais potentiellement pas la gestion des utilisateurs ou la configuration globale).
 - 6. **Action** : Répète l'étape 4 pour créer le groupe "Assistants", en sélectionnant des permissions plus limitées (ex: add et view sur écritures et factures, view sur tiers, pas de suppression, pas de validation finale, pas d'accès aux rapports financiers complexes ou à la configuration).
 - 7. **Action** : L'administrateur navigue vers la section de gestion des Utilisateurs (soit /admin/auth/user/, soit l'interface dédiée si créée).
 - 8. **Interface** : Liste des utilisateurs (contient au moins le super-utilisateur initial). Bouton "Ajouter Utilisateur".
 - 9. **Action** : Clique "Ajouter Utilisateur".
 - 10. **Interface** : Affichage du formulaire de création (Username/Email, Prénom, Nom, Assignation aux Groupes/Rôles).
 - 11. **Action** : L'admin remplit les informations pour un premier collaborateur (ex: un comptable). Il coche la case correspondant au groupe "Comptables". Il clique sur "Enregistrer" ou "Inviter".
 - 12. **Système** : Le compte utilisateur est créé. Un email est envoyé au collaborateur (si SMTP configuré) avec un lien pour définir son mot de passe ou un mot de passe temporaire.
 - 13. **Feedback** : Message de succès. L'utilisateur apparaît dans la liste avec le rôle assigné.
 - 14. **Action** : L'admin répète les étapes 10 à 13 pour créer les comptes des autres collaborateurs initiaux (autres comptables, assistants...).

4. Gestion des Dossiers Clients (Onboarding Client - Flux Administrateur/Superviseur)

Ce chapitre décrit le processus par lequel un administrateur ou un superviseur enregistre un nouveau client de COMPTA FREELANCE dans la plateforme, configure son environnement comptable initial, et assigne les collaborateurs qui y auront accès.

4.1. Flux : Crédation d'un Nouveau Dossier Client

Ce flux est essentiel pour commencer à travailler pour un nouveau client. Il est typiquement réalisé par un Administrateur ou un Superviseur disposant des droits nécessaires.

- 4.1.1. Accès et Lancement :
 1. **Action** : L'utilisateur (Admin/Sup) se connecte à la plateforme.

2. **Action** : Il navigue vers la section dédiée à la gestion des dossiers clients, accessible via la barre latérale (ex: "Dossiers Clients" ou "Paramètres > Gestion des Clients").
 3. **Interface** : La page affiche une liste (tableau) des dossiers clients déjà existants, avec des informations clés (Nom, Statut...) et des options de filtrage/recherche. Un bouton proéminent "Créer Nouveau Dossier" est visible.
 4. **Action** : L'utilisateur clique sur "Créer Nouveau Dossier".
- **4.1.2. Étape 1 : Informations Générales Client (Sélection/Création Tiers) :**
 1. **Interface** : Un formulaire, potentiellement la première étape d'un assistant (wizard), s'affiche. Il est intitulé "Étape 1: Informations du Client".
 2. **Champ Principal** : Un champ "Client" (type Autocomplete ou Select avec recherche) permet de rechercher un Tiers de type "Client" déjà enregistré dans la base de données globale des Tiers.
 3. **Action** : L'utilisateur commence à taper le nom du client.
 - *Cas A : Le Client existe.* Le système propose une liste de clients correspondants. L'utilisateur sélectionne le client approprié. Les informations du client (Nom, Num Contribuable...) peuvent pré-remplir certains champs du formulaire du dossier.
 - *Cas B : Le Client n'existe pas.* Aucun résultat correspondant n'est trouvé. Un bouton "Créer Nouveau Client" apparaît à côté du champ de recherche. L'utilisateur clique dessus.
 - **Interface (Création Rapide Tiers)** : Une fenêtre modale ou une section de formulaire s'affiche, demandant les informations minimales pour créer le Tiers Client : Nom de la société (*requis*), Numéro Contribuable (*requis ou fortement recommandé*), Adresse (optionnel à ce stade), Email (optionnel).
 - **Action** : L'utilisateur remplit les informations et clique "Enregistrer le Client". Le système crée l'enregistrement Tiers en base et sélectionne automatiquement ce nouveau client dans le champ "Client" du formulaire principal. La modale se ferme.
 4. **Interface (Suite Étape 1)** : Une fois un client sélectionné/créé, d'autres champs apparaissent/sont actifs :
 - Nom du Dossier: (Pré-rempli avec le nom du client, modifiable).
 - Statut du Dossier: (Liste déroulante, par défaut "Actif").
 - Autres champs descriptifs optionnels (ex: Secteur d'activité, contact principal...).
 5. **Action** : L'utilisateur vérifie/complète ces informations et clique sur le bouton "Suivant" ou "Passer aux Paramètres Comptables".
 - **4.1.3. Étape 2 : Paramètres Comptables Initiaux (Exercice, Plan, Journaux, TVA, Séquences) :**
 1. **Interface** : Affichage de la deuxième étape de l'assistant : "Étape 2: Configuration Comptable Initiale". Le formulaire est divisé en sous-sections logiques.
 2. **Section "Premier Exercice Comptable"** :
 - Champs obligatoires Date de Début et Date de Fin. Validation pour s'assurer que Fin > Début.

3. **Section "Plan Comptable"** :
 - Option (probablement un affichage simple ou checkbox cochée par défaut) indiquant "Utiliser le Plan Comptable SYSCohada Révisé Standard". Aucune action requise normalement. Une option avancée (cachée/désactivée par défaut) pour lier un plan personnalisé pourrait exister mais est hors scope standard.
 4. **Section "Journaux Comptables"** :
 - Une liste pré-cochée des journaux standards recommandés (AC, VE, BQ, CS, OD) est affichée avec leurs codes et libellés par défaut.
 - L'utilisateur peut décocher un journal s'il n'est pas pertinent, ou modifier légèrement le code (ex: BQ1, BQ2 si multi-banques) ou le libellé. Un bouton "Ajouter un Journal" permet de créer manuellement un journal supplémentaire si besoin (en spécifiant code, libellé, type).
 5. **Section "Paramètres TVA"** :
 - Champ Régime de TVA: (Liste déroulante : Exonéré, Réel Normal, Réel Simplifié, etc. - liste configurable globalement).
 - Champ Taux de TVA par défaut: (Champ numérique ou sélection parmi des taux préconfigurés).
 6. **Section "Séquences de Numérotation"** :
 - Lignes pour configurer les séquences automatiques (préfixe, suffixe, nombre de chiffres, prochain numéro) pour au moins : Factures de Vente. D'autres documents (Avoirs, etc.) peuvent être ajoutés.
 7. **Action** : L'utilisateur remplit/vérifie tous les champs requis et clique "Suivant" ou "Passer à l'Assignation".
- 4.1.4. **Étape 3 : Assignation Utilisateurs** :
 1. **Interface** : Affichage de la troisième étape : "Étape 3: Collaborateurs Assignés". Une liste de tous les utilisateurs internes (Collaborateurs COMPTA FREELANCE avec rôles Comptable, Assistant...) est affichée.
 2. **Liste Utilisateurs** : Chaque ligne contient le nom de l'utilisateur et une case à cocher (checkbox).
 3. **Action** : L'administrateur coche les cases correspondant aux collaborateurs qui sont autorisés à accéder et à travailler sur ce dossier client spécifique.
 4. **Action** : L'utilisateur clique sur le bouton "Terminer la Création" ou "Créer le Dossier".
 - 4.1.5. **Finalisation et Feedback** :
 1. **Système (Backend - Django)** :
 - Valide l'ensemble des données collectées lors des étapes précédentes.
 - Crée l'enregistrement du Dossier Client.
 - Crée l'enregistrement du Tiers Client si nécessaire (via Étape 1).
 - Crée le premier enregistrement ExerciceComptable associé.
 - Crée les enregistrements JournalComptable sélectionnés/définis.
 - Enregistre les paramètres TVA et les séquences de numérotation associés au dossier.
 - Crée les enregistrements liant les utilisateurs sélectionnés à ce dossier.

- dossier (gestion des permissions spécifiques au dossier).
- Le tout idéalement dans une transaction de base de données pour garantir l'atomicité (tout ou rien).

2. Interface (Feedback) :

- Si succès : Affichage d'un message clair "Le dossier pour [Nom Client] a été créé avec succès." Redirection vers la liste des dossiers clients (avec le nouveau dossier visible) ou vers la page de configuration de ce nouveau dossier.
- Si échec (validation ou erreur serveur) : Affichage d'un message d'erreur explicite, en essayant de conserver les données déjà saisies pour correction.

4.2. Flux : Configuration/Modification d'un Dossier Existant

- Action** : L'Admin/Superviseur navigue vers la "Liste des Dossiers Clients".
- Action** : Il localise le dossier à modifier et clique sur l'action "Configurer" (icône roue dentée ou bouton).
- Interface** : La page de configuration du dossier s'affiche. Elle reprend les informations des étapes de création, mais organisées potentiellement par onglets ou sections pour une meilleure clarté : "Informations Générales", "Exercices Comptables", "Journaux", "Paramètres TVA & Séquences", "Utilisateurs Assignés".
- Action** : L'utilisateur navigue vers la section/onglet pertinent(e).
- Action** : Il effectue les modifications nécessaires (ex: ajouter un nouvel exercice annuel, modifier un taux de TVA par défaut, ajouter/retirer un collaborateur de l'accès au dossier).
- Interface** : Chaque section/onglet a son propre bouton "Enregistrer les modifications" ou la sauvegarde est globale pour la page.
- Système** : Valide les modifications et les enregistre en base de données. Des contrôles stricts doivent empêcher certaines modifications si elles ont un impact comptable majeur sur des données existantes (ex: modifier les dates d'un exercice clôturé).
- Feedback** : Message de succès ou d'erreur affiché via notification ou alerte contextuelle.

4.3. Flux : Sélection du Dossier Actif par l'Utilisateur (Tous Rôles)

Ce flux est fondamental pour l'utilisation quotidienne de l'application.

- Contexte** : L'utilisateur est connecté à la plateforme.
- Interface** : La Barre Supérieure affiche le nom du dossier client actuellement actif (ou un message invitant à sélectionner un dossier si aucun n'est actif). Ex: "Dossier Actif: [Nom Dossier]".
- Action** : L'utilisateur clique sur le nom du dossier actif (ou sur le message d'invite).
- Interface** : Un menu déroulant ou une liste s'affiche, contenant **uniquement** les noms des dossiers clients auxquels cet utilisateur spécifique a été assigné (cf. étapes 4.1.4 / 4.2). Une barre de recherche peut être présente si la liste est longue.
- Action** : L'utilisateur clique sur le nom du dossier qu'il souhaite activer.

6. **Système (Backend/Session)** : L'identifiant du dossier sélectionné est enregistré dans la session de l'utilisateur côté serveur (session Django).
 7. **Interface (Mise à Jour)** :
 - o Le nom affiché dans le Sélecteur de Dossier Actif dans la barre supérieure est mis à jour.
 - o La zone de contenu principal de la page est **immédiatement rafraîchie** pour afficher les données correspondant au nouveau dossier sélectionné. Ceci peut être réalisé par :
 - Un rechargement complet de la page actuelle.
 - Une requête HTMX qui cible le conteneur principal de la page (body ou un div#content) et remplace son contenu par la vue rendue pour le nouveau dossier (HX-Refresh: true ou ciblage explicite). Cette option offre une meilleure expérience utilisateur si elle est techniquement gérable sur l'ensemble des vues.
 8. **Conséquence** : Toutes les actions et consultations effectuées par l'utilisateur (saisie, listes, rapports) s'appliqueront désormais au contexte de ce nouveau dossier actif, jusqu'à ce qu'il en sélectionne un autre. Les vues Django doivent systématiquement utiliser l'ID du dossier actif (récupéré depuis la session) pour filtrer toutes les requêtes de base de données.
-

5. Philosophie Visuelle & Expérience Utilisateur (UI/UX) Détaillée

5.1. Vision Globale : Synthèse Odoo/Sage pour Efficacité et Modernité Web

L'interface utilisateur (UI) et l'expérience utilisateur (UX) de la plateforme COMPTA FREELANCE chercheront à incarner une synergie entre deux paradigmes : l'esthétique moderne, l'accessibilité et la fluidité des applications web contemporaines (inspiration Odoo), et la densité informationnelle, la rapidité de saisie et la robustesse fonctionnelle souvent associées aux logiciels comptables de bureau classiques (inspiration Sage Saari). L'objectif n'est pas une imitation, mais la création d'une interface unique, professionnelle, intuitive et optimisée pour le travail comptable quotidien dans un environnement web. L'utilisation ciblée de HTMX et Alpine.js sera fondamentale pour atteindre la réactivité souhaitée sans la charge d'une SPA traditionnelle.

5.2. Grille de Mise en Page et Responsive Design (Objectifs Détaillés)

Une structure de grille cohérente (probablement basée sur 12 colonnes, via Bootstrap ou Tailwind) sera employée pour organiser les éléments sur la page et faciliter l'adaptabilité aux différentes tailles d'écran.

- **Grand Écran (Desktop)** : Optimisation pour l'affichage d'informations denses. Layout principal à deux ou trois colonnes (Sidebar fixe + Contenu principal fluide ; ou Sidebar + Contenu + Barre d'actions contextuelle si nécessaire). Largeur maximale du contenu potentiellement limitée pour la lisibilité sur écrans très larges.
- **Écran Moyen (Tablette)** : La barre latérale devient repliable (icônes seules) pour gagner de l'espace. La grille s'adapte, les éléments peuvent passer sur moins de colonnes. Le défilement vertical est normal.

- **Petit Écran (Mobile)** : Approche "Mobile First" conceptuelle mais réalisation pragmatique. Barre latérale cachée par défaut, accessible via bouton "burger". Contenu principal sur une seule colonne. Les formulaires s'empilent verticalement. Les tableaux de données complexes sont le défi majeur : ils nécessiteront un défilement horizontal ou une refonte de l'affichage (ex: afficher moins de colonnes, détails sur clic). L'objectif est de maintenir la **consultation possible et la saisie simple réalisable**, même si l'efficacité optimale reste sur écran plus large.

5.3. Palette de Couleurs, Typographie, Iconographie (Spécifications)

- **Palette :**
 - Fond : Blanc (#FFFFFF), Gris Très Clair (#F8F9FA).
 - Texte : Gris Foncé (#212529).
 - Bordures/Lignes : Gris Clair (#DEE2E6, #CED4DA).
 - Primaire : Bleu Professionnel (#0D6EFD ou couleur corporate COMPTA FREELANCE). Utilisé pour liens, boutons primaires, indicateurs actifs, focus.
 - Secondaire : Gris Moyen (#6C757D). Pour boutons secondaires, textes d'aide, éléments moins importants.
 - Accents : Vert (#198754) pour succès/validation. Rouge (#DC3545) pour erreur/suppression. Jaune (#FFC107) pour avertissements. Bleu Info (#0DCAF0).
- **Typographie :**
 - Police : Sans-serif, haute lisibilité, disponible (Inter, Roboto, Lato, system-ui). Weights variés (Regular, Medium, Bold).
 - Tailles : Base 1rem (16px) pour le corps de texte. Hiérarchie claire : h1 (~2rem), h2 (~1.75rem), h3 (~1.5rem), h4 (~1.25rem). Labels/Textes petits ~0.875rem.
 - Lisibilité : Interligne ~1.5. Marges suffisantes entre paragraphes/blocs.
- **Iconographie :**
 - Set d'icônes SVG cohérent (Bootstrap Icons, Tabler Icons, Feather Icons, ou FontAwesome Free). Utilisation pour : Navigation principale, Boutons d'action (Ajouter, Modifier, Supprimer, Valider, Exporter...), Indicateurs de statut, Champs de formulaire (Calendrier pour date...). Doit renforcer la compréhension, pas juste décorer. Taille et alignement cohérents avec le texte.

5.4. Composants d'Interface Standards (Design System Interne)

La définition et l'utilisation cohérente de ces composants sont essentielles pour une UI/UX réussie.

- **5.4.1. Boutons :**
 - *Styles* : Primaire (fond plein couleur primaire), Secondaire (contour primaire ou fond gris), Danger (fond rouge), Succès (fond vert), Info (fond bleu info), Texte/Lien (pas de bordure/fond).
 - *Tailles* : Small, Normal, Large.
 - *États* : Hover (léger assombrissement/éclaircissement), Focus (outline visible), Active (assombrissement plus prononcé), Disabled (opacité

- réduite, curseur not-allowed).

 - o *Usage* : Bouton Primaire pour l'action principale du formulaire/vue. Secondaire pour actions alternatives ou moins importantes (ex: Annuler). Danger pour suppressions (souvent avec modale de confirmation). Icône optionnelle à gauche du texte.
- **5.4.2. Champs de Formulaire :**
 - o *Apparence* : Bordure claire, fond blanc/très clair. Hauteur standardisée. Effet visuel subtil au focus (ex: bordure couleur primaire).
 - o *Types HTML5* : Utilisation appropriée (text, email, number, date, password, search).
 - o *Labels* : Toujours présents, clairement associés (balise <label for="...">). Positionnés au-dessus ou à gauche du champ. Texte concis. Astérisque (*) pour les champs obligatoires.
 - o *Aide Contextuelle* : Texte court sous le champ si nécessaire pour clarifier le format attendu ou la signification.
 - o *Validation* :
 - Côté Client (via Alpine.js/HTML5) : Validation basique (obligatoire, format email/nombre) possible pour feedback immédiat.
 - Côté Serveur (Django Forms + HTMX) : Validation métier complète. Si erreur, la réponse HTMX met à jour la section du formulaire : bordure rouge sur le champ invalide + message d'erreur spécifique affiché sous le champ.
 - o *Select / Autocomplete* : Style cohérent. Pour Autocomplete (Comptes, Tiers), la liste déroulante des suggestions (chargée via HTMX) doit être claire et facile à naviguer.
- **5.4.3. Tableaux de Données :**
 - o *Structure* : Balise <table> sémantique (<thead>, <tbody>, <tfoot>).
 - o *Style* : Option "Dense" pour les vues comptables : padding réduit dans les cellules, taille de police légèrement réduite. Style "Standard" pour autres listes. Lignes zébrées (striped) pour améliorer la lisibilité horizontale. Bordures fines entre lignes et/ou colonnes.
 - o *En-têtes (<thead>)* : Fond léger, texte en gras, aligné avec le contenu de la colonne (gauche pour texte, droite pour nombres). Icônes de tri cliquables si le tri est implémenté.
 - o *Pagination* : Si nécessaire (> 25-50 lignes), contrôles clairs sous le tableau (Précédent, Suivant, Numéros de page). Interactions via HTMX pour charger la page suivante sans recharger toute la vue.
 - o *Filtres/Recherche* : Barre dédiée au-dessus du tableau. Champs de formulaire standards. Bouton "Appliquer Filtres" / "Réinitialiser" déclenchant une mise à jour du tableau via HTMX.
 - o *Actions par Ligne* : Boutons (souvent petits, avec icône seule) dans une dernière colonne ou visibles au survol de la ligne.
- **5.4.4. Cartes (Cards) / Widgets :**
 - o *Structure* : Conteneur div avec bordure légère, box-shadow discrète, padding interne. En-tête optionnel (card-header) avec titre. Corps (card-body) pour le contenu. Pied de page optionnel (card-footer) pour actions/infos supplémentaires.
 - o *Usage* : Tableau de bord, regroupement de sections dans des formulaires complexes, affichage de détails synthétiques.

- **5.4.5. Modales / Pop-ups :**
 - *Apparence* : Fenêtre centrée (ou positionnée), avec un fond semi-transparent (backdrop) pour masquer l'arrière-plan. Bordure claire, en-tête avec titre et bouton de fermeture (X). Zone de contenu. Zone de pied de page avec boutons d'action (Confirmer, Annuler...).
 - *Déclenchement* : Clic sur un bouton. Peut être chargée dynamiquement via HTMX si le contenu dépend du contexte.
- **5.4.6. Notifications / Messages Toast / Alertes Contextuelles :**
 - *Toasts* : Petites boîtes apparaissant brièvement dans un coin de l'écran (ex: en haut à droite). Fond coloré selon le type (Succès, Erreur...). Disparaissent automatiquement après quelques secondes. Déclenchés via réponse HTMX (ex: header HX-Trigger avec événement JS) ou Alpine.js.
 - *Alertes Contextuelles* : Boîte colorée (Succès, Erreur...) affichée en haut de la zone de contenu principal ou près de l'élément concerné. Doit pouvoir être fermée par l'utilisateur (bouton 'X'). Reste visible jusqu'à fermeture ou navigation. Peut être injectée via HTMX.
- **5.4.7. Indicateurs de Chargement / Progression :**
 - *Pour requêtes HTMX* : Utilisation des classes CSS fournies par HTMX (ex: htmx-request) pour afficher un spinner à côté du bouton déclencheur, ou pour griser/mettre en overlay la zone cible (hx-indicator).
 - *Pour actions longues* : Barre de progression si la durée peut être estimée (rare en web), sinon spinner animé clair.

5.5. Principes de Navigation Globale et Contextuelle

La navigation doit être intuitive et prévisible.

- **Cohérence** : La barre latérale et la barre supérieure sont les points d'ancrage constants.
- **Découvrbilité** : Les modules et sous-sections doivent être clairement libellés et logiquement groupés dans la navigation.
- **Contexte Clair** : L'utilisateur doit toujours savoir sur quel Dossier Client il travaille (via barre supérieure) et où il se trouve dans l'application (via fil d'ariane et mise en évidence dans la navigation).
- **Retour Facile** : Le fil d'ariane et la structure logique permettent de remonter facilement. L'utilisation de HTMX doit préserver l'historique du navigateur autant que possible (hx-push-url).

5.6. Accessibilité (Considérations de base)

Des efforts seront faits pour respecter les principes de base :

- Contrastes suffisants (vérifiables avec des outils en ligne).
- Navigation clavier fonctionnelle (ordre de tabulation logique, focus visible sur les éléments interactifs).
- Utilisation correcte des balises sémantiques HTML5.
- Labels associés aux champs de formulaire via for.
- Attributs alt pour les images significatives.

- Utilisation prudente de JavaScript (Alpine/HTMX) pour ne pas casser l'accessibilité de base.

6. Module Cœur Comptable (SYSCohada) - Flux et Spécifications

Ce module constitue la fondation de l'application, gérant la structure comptable de base, la saisie des écritures, et les consultations fondamentales, tout en assurant une conformité rigoureuse avec le SYSCohada Révisé.

6.1. Flux : Paramétriser la Comptabilité d'un Dossier Client (Admin/Sup)

Ce flux permet de personnaliser et de gérer les éléments comptables spécifiques à un dossier client après sa création initiale.

1. **Action** : L'Admin/Superviseur se connecte et sélectionne le Dossier Client à configurer via le sélecteur global.
 2. **Action** : Il navigue vers la section "Paramètres" ou "Configuration" (via la barre latérale), puis choisit la sous-section "Paramètres Comptables" (ou accède via l'action "Configurer" de la liste des dossiers).
 3. **Interface** : Affichage de la page de configuration comptable du dossier, organisée en onglets ou sections : "Exercices Comptables", "Journaux Comptables", "Plan Comptable (Auxiliaires)".
- **6.1.1. Gestion des Exercices Comptables (UI et Logique) :**
 - *Interface (Onglet/Section "Exercices")* : Tableau listant les exercices créés pour ce dossier. Colonnes : Date Début, Date Fin, Statut (Ouvert, Clôturé). Bouton "Ajouter Exercice" en haut. Pour chaque ligne d'exercice : Actions contextuelles (ex: "Clôturer" pour le plus ancien ouvert, "Réouvrir" pour le dernier clos).
 - *Flux Ajout Exercice* :
 1. Clic "Ajouter Exercice".
 2. Affichage d'un formulaire (modal ou section) : Champs Date Début (pré-rempli si possible au lendemain de la fin du précédent), Date Fin.
 3. Validation : Dates cohérentes (Fin > Début), pas de chevauchement avec exercices existants.
 4. Clic "Enregistrer". Système crée l'enregistrement ExerciceComptable lié au dossier. Tableau mis à jour (via HTMX).
 - *Flux Clôture Exercice* :
 1. Clic "Clôturer" sur l'exercice concerné (le plus ancien ouvert).
 2. Affichage d'une modale de confirmation ("Voulez-vous vraiment clôturer l'exercice XXXX ? Cette action est difficilement réversible et empêchera toute saisie."). Potentiellement, affichage de contrôles préalables (ex: Toutes écritures validées ? Balance équilibrée ? - à définir).
 3. Clic "Confirmer la Clôture".
 4. Système met à jour le statut de l'exercice à "Clôturé". Exécute potentiellement des tâches de clôture (génération écriture d'à-nouveaux pour l'exercice suivant si existant et ouvert - logique complexe à définir).

- 5. Tableau mis à jour. Saisie/Modification interdite dans cet exercice via les autres modules.
- o *Flux Réouverture Exercice :*
 1. Clic "Réouvrir" sur le dernier exercice clôturé (action à utiliser avec extrême précaution).
 2. Modale de confirmation très explicite sur les risques.
 3. Clic "Confirmer la Réouverture".
 4. Système met à jour le statut à "Ouvert". Annule potentiellement les écritures d'à-nouveaux générées si applicable.
 5. Tableau mis à jour. Saisie/Modification à nouveau possible.
- **6.1.2. Gestion des Journaux Comptables (UI et Logique) :**
 - o *Interface (Onglet/Section "Journaux") :* Tableau listant les journaux spécifiques à ce dossier. Colonnes : Code, Libellé, Type (Achat, Vente, Banque...). Bouton "Ajouter Journal". Actions par ligne : Modifier, Supprimer (seulement si aucune écriture n'existe pour ce journal).
 - o *Flux Ajout Journal :*
 1. Clic "Ajouter Journal".
 2. Formulaire (modal/section) : Champ Code (texte court, unique pour ce dossier), Libellé (texte), Type (liste déroulante Achat/Vente/Banque/Caisse/OD).
 3. Validation : Code unique.
 4. Clic "Enregistrer". Système crée JournalComptable lié au dossier. Tableau mis à jour (HTMX).
 - o *Flux Modification Journal :*
 1. Clic "Modifier" sur une ligne.
 2. Formulaire pré-rempli. Seuls Libellé et Type sont modifiables (Code est clé).
 3. Clic "Enregistrer". Système met à jour. Tableau mis à jour.
 - o *Flux Suppression Journal :*
 1. Clic "Supprimer" sur une ligne (bouton désactivé si écritures existent).
 2. Modale de confirmation.
 3. Clic "Confirmer". Système supprime l'enregistrement. Tableau mis à jour.
- **6.1.3. Gestion du Plan Comptable Spécifique (Consultation, Auxiliaires) :**
 - o *Interface (Onglet/Section "Plan Comptable") :*
 - Zone affichant que le dossier utilise le "Plan Comptable SYSCohada Révisé Standard". Lien pour consulter ce plan standard (lecture seule).
 - Section dédiée à la "Gestion des Comptes Auxiliaires" (ou Comptes Tiers).
 - o *Flux Consultation Plan Standard :* Clic sur le lien -> Ouvre une modale ou une page affichant le plan SYSCohada général (Numéro, Intitulé). Non modifiable. Fonction de recherche intégrée.
 - o *Flux Gestion Auxiliaires :*
 1. *Visualisation :* Tableau listant les comptes auxiliaires déjà créés pour ce dossier, groupés par compte général Tiers (ex: 401 - Fournisseurs, 411 - Clients). Colonnes : N° Compte Général, N°

- Compte Auxiliaire, Intitulé (Nom Tiers).
- 2. *Création* : La création se fait **implicitement** lors de la création d'un Tiers (Client/Fournisseur) via le Module Tiers (Section 7). Un compte auxiliaire est automatiquement généré et lié au compte collectif défini pour le tiers. Le numéro auxiliaire peut être le code du tiers ou une séquence automatique.
- 3. *Modification/Consultation* : La modification de l'intitulé se fait via la fiche Tiers. Cette section permet surtout de voir le lien entre Tiers et Comptes Auxiliaires.

6.2. Flux : Saisir une Écriture Manuelle (Assistant/Comptable)

Ce flux décrit la saisie d'une écriture non générée automatiquement par les modules Ventes/Achats (ex: OD, opérations de trésorerie manuelles).

- **6.2.1. Accès et Sélection Journal :**
 - 1. **Action** : Utilisateur se connecte, sélectionne le Dossier Client.
 - 2. **Action** : Navigue vers "Comptabilité" > "Saisie des Écritures".
 - 3. **Interface** : Si plusieurs journaux sont saisissables manuellement, une première étape peut demander de sélectionner le journal (ex: liste déroulante ou page de sélection). Sinon, accès direct à l'écran de saisie pour un journal par défaut (ex: OD). Le journal sélectionné est clairement indiqué en titre.
- **6.2.2. Interface de Saisie Détaillée (Layout, En-tête, Lignes) :**
 - *Interface* : Écran principal divisé en :
 - Zone d'En-tête : Champs Date Écriture, N° Pièce, Libellé Général.
 - Zone des Lignes : Tableau dynamique (Colonnes: Compte, Libellé Ligne, Débit, Crédit, [Act: Suppr]).
 - Zone Totaux : Affichage Totaux Débit, Crédit, Solde.
 - Zone Actions : Boutons "Enregistrer", "Annuler".
 - *Workflow Saisie En-tête* : Utilisateur remplit les champs. Date Écriture doit être dans un exercice ouvert.
- **6.2.3. Interaction Lignes (Ajout/Suppression/Validation Compte - HTMX/Alpine) :**
 1. **Action (Ajout)** : Clic sur "(+ Ajouter Ligne)". Requête hx-get -> Vue Django renvoie HTML nouvelle ligne -> HTMX insère dans tbody. Focus mis sur champ Compte (Alpine.js).
 2. **Action (Saisie Compte)** : Utilisateur tape dans champ N° Compte. Requête hx-get (avec trigger keyup delay) -> Vue Django cherche comptes correspondants (plan général + auxiliaires si journal Tiers) -> Renvoie liste suggestions HTML -> HTMX affiche sous le champ. Sélection remplit le champ.
 3. **Action (Saisie Ligne)** : Utilisateur remplit Libellé Ligne, Débit ou Crédit. Alpine.js assure que si Débit > 0, Crédit = 0 et vice-versa.
 4. **Système (Calcul Solde)** : Chaque modification de Débit/Crédit, ajout ou suppression de ligne déclenche (via input ou htmx:afterSwap) une requête hx-post (hx-trigger="input delay:500ms, line-changed from:body") envoyant les données des lignes -> Vue Django recalcule ->

Renvoie fragment HTML mettant à jour la zone Totaux et l'indicateur visuel d'équilibre.

5. **Action (Suppression)** : Clic sur "(X)" d'une ligne. Requête hx-delete -> Vue Django ne fait rien côté serveur (la ligne n'existe pas encore) ou marque pour suppression si modification -> Réponse vide ou indicateur -> HTMX retire la ligne du DOM (hx-target="closest tr"). Le recalculation du solde est déclenché.
- **6.2.4. Calcul et Affichage Solde Temps Réel (HTMX/Alpine) :**
 - *Interface* : Zone dédiée sous le tableau des lignes : Total Débit: [valeur], Total Crédit: [valeur], Solde: [différence]. Un indicateur visuel (ex: pastille verte/rouge, texte) à côté de "Solde".
 - *Mise à jour* : Effectuée via la réponse HTMX déclenchée par les modifications des lignes (cf. 6.2.3 étape 4).
- **6.2.5. Workflow de Validation et Sauvegarde (HTMX/Django) :**
 1. **Action** : Utilisateur clique "Enregistrer l'Écriture" (bouton actif seulement si écriture équilibrée? - à décider).
 2. **Système (HTMX/Django)** : Requête hx-post avec toutes les données du formulaire (en-tête + lignes sérialisées).
 3. **Validation Serveur** : La vue Django effectue toutes les validations :
 - Exercice ouvert ?
 - Journal valide ?
 - Écriture équilibrée ?
 - Chaque ligne : Compte valide et autorisé ? Montants valides ? Autres règles SYSCohada ?
 4. *Si Erreur :*
 - **Réponse HTMX** : Renvoie le formulaire (ou des fragments ciblés) avec les erreurs mises en évidence (messages sous les champs, is-invalid classes CSS). La cible de hx-post est le formulaire lui-même (hx-target="#form-ecriture").
 - **Interface** : Le formulaire s'affiche avec les erreurs. L'utilisateur corrige.
 5. *Si Succès :*
 - **Action Serveur** : Crée l'enregistrement EcritureComptable et les LigneEcriture associées en base, dans une transaction.
 - **Réponse HTMX** : Envoie une réponse qui déclenche plusieurs actions côté client (via headers HX-Trigger ou JS simple dans la réponse) :
 - Afficher un message "Toast" de succès ("Écriture enregistrée").
 - Vider le formulaire de saisie pour permettre une nouvelle saisie (hx-target="#form-ecriture", la vue renvoie un formulaire vide).
 - (Optionnel) Mettre à jour une liste d'écritures récentes affichée ailleurs sur la page (si présente, via hx-trigger avec événement personnalisé).

6.3. Flux : Consulter le Grand Livre d'un Compte (Tous Rôles avec Accès)

- **6.3.1. Accès et Filtres (Compte, Période) :**
 - Action** : Naviguer vers "Comptabilité" > "Grand Livre".
 - Interface** : Section de filtres en haut :
 - Champ Compte: (Autocomplete, recherche par N°/Intitulé sur Plan Général + Auxiliaires).
 - Champ Date Début: (Datepicker).
 - Champ Date Fin: (Datepicker).
 - Bouton "Afficher" (style primaire).
 - Action** : L'utilisateur saisit/sélectionne le compte, les dates, clique "Afficher".
- **6.3.2. Affichage des Résultats (Tableau GL - HTMX) :**
 - Système (HTMX/Django)** : Le clic sur "Afficher" déclenche une requête hx-get (ou hx-post si beaucoup de filtres) avec les paramètres vers une vue Django. La vue récupère le solde initial (report à nouveau de l'exercice précédent si pertinent), puis toutes les LigneEcriture pour ce compte et cette période, ordonnées par date. Elle calcule le solde progressif. Elle renvoie le fragment HTML contenant le tableau de résultat.
 - Interface** : Une zone "Résultats" sous les filtres est mise à jour (hx-target="#gl-results", hx-swap="innerHTML") avec le tableau :
 - Ligne Solde Initial.
 - Tableau (<thead> : Date, Journal, N° Pièce, Libellé, Mouvement Débit, Mouvement Crédit, Solde Progressif).
 - Lignes de <tbody> pour chaque mouvement.
 - Ligne <tfoot> avec Totaux Mouvements D/C et Solde Final.
 - Le tableau est dense, formaté pour la lisibilité comptable.
- **6.3.3. Options d'Export :**
 - **Interface** : Des boutons "Exporter CSV" / "Exporter PDF" apparaissent près du tableau de résultat.
 - **Action** : Clic sur un bouton -> Déclenche une requête GET vers une URL spécifique (ex: /comptabilite/grand-livre/export/csv/?compte=...&debut=...&fin=...).
 - **Système (Django)** : Une vue dédiée génère le fichier CSV ou PDF (avec une librairie comme ReportLab ou WeasyPrint) et le renvoie avec les bons headers HTTP pour déclencher le téléchargement dans le navigateur.

6.4. Flux : Consulter une Balance (Générale/Auxiliaire) (Comptable/Sup)

- **6.4.1. Accès et Filtres (Période, Type, Niveau) :**
 - Action** : Naviguer vers "Comptabilité" > "Balance".
 - Interface** : Section de filtres :
 - Date Début, Date Fin.
 - Type de Balance: (Radio/Select : Générale, Auxiliaire).
 - (Si Générale) Niveau de détail: (Radio/Select : Tous les comptes, Comptes mouvementés seulement, Niveau des classes...).
 - (Si Auxiliaire) Compte Collectif Tiers: (Select/Autocomplete pour choisir 401, 411...).
 - Bouton "Afficher".

- **6.4.2. Affichage des Résultats (Tableau Balance - HTMX) :**
 1. **Action** : Sélectionner options, cliquer "Afficher".
 2. **Système (HTMX/Django)** : Requête hx-get/post -> Vue Django calcule la balance demandée (agrégation des soldes et mouvements par compte sur la période). Renvoie fragment HTML du tableau.
 3. **Interface** : Zone "Résultats" mise à jour (hx-target, hx-swap) avec le tableau :
 - **Balance Générale** : Colonnes : N° Compte, Intitulé, Solde Initial D/C, Mouvements Période D/C, Solde Final D/C.
 - **Balance Auxiliaire** : Colonnes : N° Compte Aux, Intitulé Tiers, Soldes/Mouvements D/C...
 - Lignes pour chaque compte/tiers. Lignes de totaux généraux. Format SYSCohada.
- **6.4.3. Options d'Export :**
 - Similaire au Grand Livre : Boutons "Exporter CSV/PDF" déclenchant des vues Django dédiées à la génération de fichiers.

6.5. Flux : Lettrer un Compte Tiers (Assistant+/Comptable)

- **6.5.1. Accès et Sélection Compte Tiers :**
 1. **Action** : Naviguer vers "Comptabilité" > "Lettrage Tiers".
 2. **Interface** : Filtres : Champ Compte Tiers: (Autocomplete sur comptes généraux 401/411 + comptes auxiliaires), Options (Afficher Non Lettrées seulement, Afficher déjà lettrées).
 3. **Action** : Sélectionner un compte tiers spécifique (ex: "401SUP001 - Fournisseur DUPONT").
- **6.5.2. Interface de Lettrage (Tableau Non Lettrés, Zone Sélection) :**
 1. **Système (HTMX/Django)** : Charge (via hx-get sur changement de compte) les écritures non lettrées pour ce compte. Renvoie le fragment HTML.
 2. **Interface** : Affiche un tableau optimisé pour le lettrage :
 - Colonnes : Checkbox, Date, Jnl, N° Pièce, Libellé, Débit, Crédit.
 - Peut être séparé en deux tableaux (Débits / Crédits) ou un seul trié différemment.
 3. **Interface** : Zone "Sélection pour Lettrage" en bas ou sur le côté :
 - Affiche : "X lignes sélectionnées", "Total Débit Sélectionné:", "Total Crédit Sélectionné:", "Solde Sélectionné:".
 - Bouton "Lettrer Sélection" (initialement désactivé).
- **6.5.3. Workflow de Sélection et Validation (HTMX/Alpine/Django) :**
 1. **Action** : L'utilisateur coche les checkboxes des lignes Débit et Crédit qu'il souhaite rapprocher.
 2. **Système (Alpine.js)** : À chaque coche/décoche, un script Alpine recalcule les totaux Débit/Crédit de la sélection et met à jour les valeurs dans la zone "Sélection pour Lettrage". Il active/désactive le bouton "Lettrer Sélection" si Solde == 0.
 3. **Action** : Quand la sélection est équilibrée et correcte, l'utilisateur clique "Lettrer Sélection".
 4. **Système (HTMX/Django)** : Requête hx-post avec les IDs des LigneEcriture sélectionnées.

5. **Validation Serveur** : La vue Django vérifie : sélection équilibrée ? Lignes appartiennent au bon compte ? Lignes non déjà lettrées ?
6. *Si Succès :*
 - **Action Serveur** : Génère un identifiant de lettrage unique (ex: 'LET' + timestamp ou UUID court). Met à jour le champ lettrage_id de toutes les lignes sélectionnées avec cet ID.
 - **Réponse HTMX** : Renvoie une réponse qui déclenche : Affichage Toast "Lettrage effectué", et rafraîchissement du tableau des écritures non lettrées (les lignes lettrées disparaissent, via HX-Refresh: true ou re-requête hx-get sur le tableau).
7. *Si Échec* : Réponse HTMX affiche un message d'erreur clair.

7. Module Gestion des Tiers - Flux et Spécifications

Ce module centralise la gestion des informations relatives aux clients et aux fournisseurs des entreprises gérées par COMPTA FREELANCE. Il est étroitement lié au module Cœur Comptable via les comptes auxiliaires.

7.1. Flux : Consulter la Liste des Tiers (Tous Rôles avec Accès)

1. **Action** : Utilisateur se connecte, sélectionne le Dossier Client.
2. **Action** : Navigue vers "Tiers" dans la barre latérale. Il peut y avoir des sous-menus "Clients" et "Fournisseurs". Sélectionne l'un ou l'autre, ou une vue combinée avec filtre.
3. **Interface** : Affichage de la liste des tiers (clients ou fournisseurs selon la vue).
 - o *Filtres* : Champ de recherche (par Nom, N° Compte Auxiliaire, Num Contribuable), Filtre par Type (Client/Fournisseur si vue combinée), Filtre par Statut (Actif/Inactif). Bouton "Appliquer Filtres". Bouton "Ajouter Client" / "Ajouter Fournisseur".
 - o *Tableau* : Colonnes : Nom/Raison Sociale, N° Compte Auxiliaire, Type (Client/F.), Num Contribuable, Téléphone Principal, Email Principal, Solde Comptable (optionnel, calculé), Statut, Actions (Voir Détail, Modifier, Activer/Désactiver).
 - o *Pagination* : Contrôles de pagination si la liste est longue.
4. **Interaction** : La recherche et le filtrage mettent à jour le tableau via des requêtes HTMX (hx-get sur le conteneur du tableau). La pagination fonctionne de même. Cliquer sur une ligne ou "Voir Détail" mène au flux 7.4. Cliquer sur "Ajouter..." mène au flux 7.2.

7.2. Flux : Créer un Nouveau Tiers (Assistant+/Comptable/Sup)

1. **Action** : Depuis la liste des Tiers, cliquer sur "Ajouter Client" ou "Ajouter Fournisseur".
2. **Interface** : Affichage d'un formulaire de création (pleine page ou grande modale). Titre : "Nouveau Client" ou "Nouveau Fournisseur". Formulaire organisé par sections (Informations Générales, Adresses, Contacts, Paramètres Comptables).
3. **Section "Informations Générales"** :

- o Champs : Nom / Raison Sociale (*requis*), Type (pré-rempli Client/Fournisseur, non modifiable), Numéro Contribuable (validation format si possible), Téléphone Principal, Email Principal, Site Web (optionnel).
4. **Section "Adresse Principale"** :
 - o Champs : Adresse Ligne 1, Ligne 2, Code Postal, Ville, Pays (Select).
 5. **Section "Contacts" (Optionnel)** :
 - o Possibilité d'ajouter dynamiquement des contacts liés (Nom, Prénom, Fonction, Email, Téléphone). Géré via HTMX/Alpine pour ajouter/supprimer des lignes de contact.
 6. **Section "Paramètres Comptables"** :
 - o Champ Compte Collectif Général : (Select/Autocomplete filtré sur les comptes de classe 40 ou 41 du Plan Comptable SYSCohada). *Requis*.
 - o Champ Numéro Compte Auxiliaire : (Soit généré automatiquement selon une séquence, soit saisie manuelle avec validation d'unicité pour ce compte collectif).
 - o Champ Conditions de Paiement par Défaut : (Select : ex: 30 jours nets, Comptant, etc. - liste configurable).
 - o Champ Taux de TVA par défaut (si applicable spécifiquement à ce tiers).
 7. **Action** : L'utilisateur remplit les informations requises et clique sur "Enregistrer le Tiers".
 8. **Système (Django)** :
 - o Valide toutes les données (champs obligatoires, format email, unicité Num Compte Auxiliaire si manuel...).
 - o Crée l'enregistrement Tiers en base.
 - o Crée automatiquement l'enregistrement CompteAuxiliaire correspondant et le lie au Tiers et au Compte Collectif sélectionné.
 - o Crée les enregistrements Adresse, Contact si saisis.
 9. **Feedback** :
 - o *Si Succès* : Message "Toast" succès. Redirection vers la Fiche Détail du Tiers nouvellement créé (Flux 7.4) ou retour à la liste des Tiers avec le nouveau tiers visible.
 - o *Si Erreur* : Réaffichage du formulaire avec les messages d'erreur mis en évidence (via réponse HTMX).

7.3. Flux : Modifier un Tiers Existant

1. **Action** : Depuis la Liste des Tiers ou la Fiche Détail, cliquer sur l'action "Modifier".
2. **Interface** : Affichage du formulaire de modification, pré-rempli avec les données existantes du Tiers. Structure identique au formulaire de création. Certains champs peuvent être non modifiables (ex: Type Client/Fournisseur si des transactions existent, Numéro Compte Auxiliaire).
3. **Action** : L'utilisateur modifie les informations souhaitées.
4. **Action** : Clique sur "Enregistrer les Modifications".
5. **Système (Django)** : Valide les données modifiées. Met à jour l'enregistrement Tiers et potentiellement les enregistrements liés (Adresse, Contacts...).
6. **Feedback** : Message succès. Redirection vers la Fiche Détail mise à jour ou retour à la liste. Si erreur, réaffichage du formulaire avec erreurs.

7.4. Flux : Consulter la Fiche Détail d'un Tiers (Vue 360°)

1. **Action** : Depuis la Liste des Tiers, cliquer sur le nom du tiers ou sur l'action "Voir Détail".
 2. **Interface** : Affichage de la fiche détaillée du Tiers. Mise en page organisée, potentiellement avec des onglets ou des sections claires :
 - o **En-tête** : Nom du Tiers, Type, Numéro Compte Auxiliaire, Statut. Boutons d'action principaux (Modifier, Archiver/Activer, Créer Facture/Paiement Lié...).
 - o **Onglet/Section "Informations Générales"** : Affiche toutes les coordonnées (Adresse, Téléphone, Email, Num Contribuable...).
 - o **Onglet/Section "Historique Comptable"** : Affiche un aperçu rapide des dernières écritures concernant ce tiers dans le Grand Livre Auxiliaire. Solde actuel du compte. Lien vers le GL complet de ce tiers.
 - o **Onglet/Section "Factures Ventes" (si Client) / "Factures Achats" (si Fournisseur)** : Tableau listant les factures associées à ce tiers, avec leur statut (Payée, En retard...). Liens vers les détails des factures.
 - o **Onglet/Section "Paiements" (Encaissements/Décaissements)** : Tableau listant les paiements liés à ce tiers.
 - o **Onglet/Section "Contacts" (si géré)** : Liste des contacts associés.
 - o **Onglet/Section "Paramètres"** : Affiche les paramètres comptables (Compte Collectif, Conditions Paiement...).
 3. **Interaction** : Les tableaux dans les onglets peuvent avoir leur propre pagination/filtrage simple (via HTMX). Cliquer sur des liens (ex: numéro de facture) navigue vers l'écran de détail correspondant.
-

8. Module Gestion des Ventes - Flux et Spécifications

Ce module couvre le processus allant de la création de la facture client à l'enregistrement de l'encaissement et au suivi du paiement.

8.1. Flux : Consulter la Liste des Factures Ventes (Assistant+/Comptable/Sup)

1. **Action** : Utilisateur se connecte, sélectionne Dossier Client.
2. **Action** : Navigue vers "Ventes" > "Factures Clients".
3. **Interface** : Affichage de la liste des factures de vente pour le dossier actif.
 - o **Filtres** : Recherche (par N° Facture, Nom Client), Filtre par Statut (Brouillon, Validée, Payée Partiellement, Payée, Annulée, En Retard), Filtre par Période (Date Facture). Bouton "Appliquer". Bouton "Créer Facture".
 - o **Tableau** : Colonnes : N° Facture, Date Facture, Client, Date Échéance, Montant TTC, Montant Dû, Statut. Actions (Voir, Modifier si Brouillon, Dupliquer, Enregistrer Paiement, Annuler si possible...).
 - o **Pagination**.
4. **Interaction** : Filtrage/Pagination via HTMX. Clic sur N° Facture ou "Voir" -> Flux 8.4. Clic "Créer Facture" -> Flux 8.2.

8.2. Flux : Crée une Facture de Vente (Assistant+/Comptable)

- **8.2.1. Initialisation et En-tête :**
 1. **Action** : Clic sur "Créer Facture" depuis la liste ou autre raccourci.
 2. **Interface** : Affichage du formulaire "Nouvelle Facture de Vente".
 - Champ Client: (Autocomplete/Recherche sur Tiers Clients). La sélection peut pré-remplir Adresse de facturation, Conditions Paiement, Taux TVA par défaut.
 - Champ Date Facture: (Datepicker, défaut: aujourd'hui).
 - Champ Date Échéance: (Calculée automatiquement basée sur Conditions Paiement du client, modifiable).
 - Champ N° Facture: (Affiché comme "PROVISOIRE" ou vide, sera généré à la validation).
 - Champs pour Objet ou Référence Commande Client (optionnels).
- **8.2.2. Ajout/Gestion des Lignes de Facture (Dynamique HTMX/Alpine) :**
 1. **Interface** : Section "Lignes de Facture" sous forme de tableau éditable. Colonnes : Description/Article, Compte Produit (SYSCohada Classe 7), Quantité, Unité (optionnel), Prix Unitaire HT, Taux TVA (%), Montant HT, Montant TVA, Montant TTC, [Action: Suppr Ligne].
 2. **Action** : Clic sur "(+) Ajouter Ligne". Requête hx-get → Vue renvoie HTML ligne vierge → HTMX insère.
 3. **Action** : Utilisateur remplit la ligne :
 - Description/Article: Texte libre ou sélection d'un Article si Module Stocks est utilisé/lié (Autocomplete).
 - Compte Produit: Autocomplete>Select sur comptes de Classe 7.
 - Quantité, Prix Unitaire HT.
 - Taux TVA: Select/Input (défaut du client ou global).
 4. **Système (Alpine.js local ou réponse HTMX partielle)** : Calcule automatiquement Montant HT, Montant TVA, Montant TTC pour la ligne dès que Qté, PU, Taux TVA sont saisis. Met à jour les totaux généraux de la facture (Total HT, Total TVA par taux, Total TTC) affichés sous le tableau.
 5. **Action** : Ajoute/Supprime d'autres lignes si nécessaire (interactions HTMX/Alpine comme pour saisie écriture).
- **8.2.3. Validation et Génération Écriture Comptable :**
 1. **Interface** : Sous les lignes et totaux, boutons "Enregistrer Brouillon" et "Valider la Facture".
 2. **Action (Enregistrer Brouillon)** : Clic "Enregistrer Brouillon". Requête hx-post. Django valide les données saisies (client, dates, montants > 0...) mais **ne génère pas** d'écriture comptable. Enregistre la facture avec statut "Brouillon". Feedback succès, retour à la liste ou reste sur le formulaire éditable.
 3. **Action (Valider la Facture)** : Clic "Valider la Facture". Requête hx-post.
 4. **Système (Django - Backend)** :
 - Validation complète et finale des données.
 - Assignation d'un numéro de facture définitif basé sur la séquence configurée.
 - Mise à jour du statut de la facture à "Validée".
 - **Génération de l'écriture comptable correspondante dans le Journal des Ventes (JV)** :
 - Débit : Compte Client (411xxxx) pour le Montant TTC.
 - Crédit : Compte(s) de Produit (7xxxxxxxx) pour les Montants

HT par ligne/compte.

- Crédit : Compte TVA Collectée (443xxx) pour le Total TVA.
- L'écriture est générée avec la date de la facture, le N° de facture comme pièce/libellé.
- Enregistrement de la facture et de l'écriture en base dans une transaction.

5. Feedback :

- *Si Succès* : Message "Toast" succès ("Facture FV-XXXX validée et comptabilisée"). Redirection vers la page de visualisation de la facture validée (Flux 8.4) via HX-Redirect.
- *Si Erreur* : Réponse HTMX affichant les erreurs sur le formulaire. Statut reste "Brouillon".

8.3. Flux : Modifier une Facture (si Brouillon)

1. **Action** : Depuis la liste des factures, cliquer "Modifier" sur une facture au statut "Brouillon".
2. **Interface** : Affichage du formulaire de création/édition pré-rempli avec les données du brouillon.
3. **Action** : Modifier les informations ou les lignes.
4. **Action** : Cliquer "Enregistrer Brouillon" ou "Valider la Facture" (cf. flux 8.2.3).

8.4. Flux : Visualiser une Facture Validée (Aperçu/PDF)

1. **Action** : Depuis la liste, cliquer sur le N° de Facture ou l'action "Voir".
2. **Interface** : Affichage de la page de détail de la facture (non éditable).
 - o Affiche clairement toutes les informations de la facture (Client, Dates, Numéro, Lignes, Totaux).
 - o Affiche le statut ("Validée", "Payée Partiellement", "Payée").
 - o Affiche le Montant Dû.
 - o Affiche les paiements déjà enregistrés et liés à cette facture.
 - o Affiche un lien ou des informations sur l'écriture comptable générée (Journal, N° Écriture).
 - o Boutons d'actions contextuelles : "Enregistrer un Paiement", "Générer PDF", "Dupliquer", "Créer Avoir".
3. **Action (Générer PDF)** : Clic -> Requête GET vers une vue Django dédiée -> Génération du PDF de la facture (avec un template PDF simple mais professionnel) -> Téléchargement proposé au navigateur.

8.5. Flux : Enregistrer un Encaissement Client (Assistant+/Comptable)

1. **Action** : Soit depuis la fiche Client, soit depuis la vue d'une Facture, soit via un menu "Ventes > Encaissements > Enregistrer".
2. **Interface** : Formulaire "Nouvel Encaissement Client".
 - o Champ Client: (Sélection/Autocomplete).
 - o Champ Date Encaissement: (Datepicker).
 - o Champ Montant Encaissé: (Numérique).
 - o Champ Compte de Trésorerie: (Select/Autocomplete sur comptes Banque/Caisse).
 - o Champ Mode de Paiement: (Select: Virement, Chèque, Espèces...).

- o Champ Référence Paiement: (Texte libre : N° Chèque, Ref Virement...).
 - o (Optionnel) Section pour affecter immédiatement ce paiement à une ou plusieurs factures ouvertes de ce client.
3. **Action** : Remplir le formulaire, cliquer "Enregistrer l'Encaissement".
 4. **Système (Django)** : Valide les données. Crée l'enregistrement EncaissementClient. Génère l'écriture comptable dans le journal de trésorerie approprié (Débit: Banque/Caisse, Crédit: Compte Client 411).
 5. **Feedback** : Message Succès. Redirection vers la liste des encaissements ou la fiche client.

8.6. Flux : Affecter/Letter un Encaissement à une/des Facture(s)

- **Option 1 (Lors de la saisie encaissement)** : Le formulaire d'encaissement peut inclure une section (chargée via HTMX après sélection client) listant les factures ouvertes du client avec des champs pour saisir le montant affecté à chacune. La validation vérifie que le total affecté <= montant encaissé. Le lettrage peut être déclenché si une affectation solde une facture et l'encaissement.
- **Option 2 (Via Module Lettrage)** : L'encaissement crée une ligne au crédit du compte 411. La facture crée une ligne au débit. L'utilisateur utilise le module de lettrage (Flux 6.5) pour rapprocher manuellement la/les ligne(s) d'encaissement avec la/les ligne(s) de facture.

8.7. Flux : Gérer un Avoir/Note de Crédit Vente

1. **Action** : Depuis une facture validée, cliquer "Créer Avoir". Ou via "Ventes > Avoirs > Créer".
2. **Interface** : Formulaire similaire à la création de facture, mais pour un avoir. Permet de sélectionner la facture d'origine (pré-rempli infos), de saisir les lignes à annuler/rembourser (quantités négatives ou montants négatifs), raison de l'avoir.
3. **Action** : Valider l'avoir.
4. **Système (Django)** : Crée l'enregistrement Avoir. Génère l'écriture comptable inverse de la vente dans le Journal des Ventes (ou un journal d'avoir dédié) : Crédit Compte Client, Débit Comptes Produits/TVA. L'avoir peut ensuite être lettré avec la facture initiale ou un remboursement.

9. Module Gestion des Achats - Flux et Spécifications

Ce module est le symétrique du module Ventes et couvre la réception et la comptabilisation des factures fournisseurs, ainsi que le suivi et l'enregistrement des paiements effectués.

9.1. Flux : Consulter la Liste des Factures Achats (Assistant+/Comptable/Sup)

1. **Action** : Utilisateur se connecte, sélectionne Dossier Client.
2. **Action** : Navigue vers "Achats" > "Factures Fournisseurs".
3. **Interface** : Affichage de la liste des factures d'achat enregistrées pour le

dossier actif.

- o *Filtres* : Recherche (par N° Facture Fournisseur, Nom Fournisseur), Filtre par Statut (À Payer, Payée Partiellement, Payée, Annulée, En Retard d'échéance), Filtre par Période (Date Facture ou Date Réception). Bouton "Appliquer". Bouton "Saisir Facture Achat".
 - o *Tableau* : Colonnes : N° Facture Fournisseur, Date Facture F., Fournisseur, Date Échéance, Montant TTC, Montant Dû, Statut. Actions (Voir Détail, Modifier si possible, Enregistrer Paiement, Annuler si possible...).
 - o *Pagination*.
4. **Interaction** : Filtrage/Pagination via HTMX. Clic sur N° Facture ou "Voir" -> Vue détail (similaire à Ventes, mais pour Achat). Clic "Saisir Facture Achat" -> Flux 9.2.

9.2. Flux : Saisir une Facture d'Achat (Assistant/Comptable)

Ce flux est crucial car il implique la saisie des informations de la pièce fournisseur et l'imputation comptable correcte.

- **9.2.1. Saisie des Informations Fournisseur et Facture :**
 1. **Action** : Clic sur "Saisir Facture Achat" depuis la liste ou autre raccourci.
 2. **Interface** : Affichage du formulaire "Nouvelle Facture Fournisseur".
 - Champ Fournisseur: (Autocomplete/Recherche sur Tiers Fournisseurs). La sélection peut pré-remplir compte collectif, conditions paiement par défaut. Bouton "Créer Nouveau Fournisseur" si besoin (cf. Flux 7.2). *Requis*.
 - Champ N° Facture Fournisseur: (Texte libre, tel qu'indiqué sur la pièce). *Requis*. Doit idéalement vérifier l'unicité pour ce fournisseur pour éviter les doublons.
 - Champ Date Facture Fournisseur: (Datepicker). *Requis*.
 - Champ Date de Réception: (Datepicker, défaut: aujourd'hui). Utile pour la date de prise en compte de la TVA déductible.
 - Champ Date Échéance: (Calculée automatiquement basée sur Date Facture F. et Conditions Paiement du fournisseur, modifiable).
 - Champ Objet / Référence Commande: (Texte libre).
- **9.2.2. Ajout/Gestion des Lignes d'Achat (Imputation Charges/TVA) :**
 1. **Interface** : Section "Lignes de la Facture" sous forme de tableau éditable. Colonnes : Description Achat, Compte Charge (SYSCohada Classe 6 ou Immobilisation Classe 2), Quantité (optionnel), Prix Unitaire HT, Taux TVA (%), Montant HT, Montant TVA, Montant TTC, [Action: Suppr Ligne].
 2. **Action** : Clic sur "(+) Ajouter Ligne". Requête hx-get -> Vue renvoie HTML ligne vierge -> HTMX insère.
 3. **Action** : Utilisateur remplit la ligne :
 - Description Achat: Texte libre (*requis*).
 - Compte Charge: Autocomplete>Select sur comptes de Classe 6 (ou 2 si immo). *Requis*. C'est l'imputation comptable principale.
 - Quantité, Prix Unitaire HT (ou directement Montant HT).
 - Taux TVA: Select/Input (Taux standard, réduit, etc.). Permet le

- calcul de la TVA déductible.
4. **Système (Alpine.js / HTMX)** : Calcule automatiquement Montant HT (si Qté/PU saisis), Montant TVA, Montant TTC pour la ligne. Met à jour les totaux généraux de la facture (Total HT, Total TVA par taux, Total TTC) affichés sous le tableau.
 5. **Action** : Ajoute/Supprime d'autres lignes si nécessaire.
- **9.2.3. Validation et Génération Écriture Comptable :**
 1. **Interface** : Sous les lignes et totaux, bouton "Enregistrer la Facture". (Il n'y a généralement pas de statut "Brouillon" pour les factures d'achat saisies, elles sont enregistrées directement). Bouton "Annuler".
 2. **Action** : Clic "Enregistrer la Facture". Requête hx-post.
 3. **Système (Django - Backend)** :
 - Validation complète : Fournisseur sélectionné, N° Facture F. unique pour ce fournisseur, Dates cohérentes, Lignes présentes, Comptes de charge valides, Calculs totaux cohérents.
 - **Génération de l'écriture comptable correspondante dans le Journal des Achats (JA) :**
 - Débit : Compte(s) de Charge (6xxxxxx) ou Immo (2xxxxxx) pour les Montants HT par ligne/compte.
 - Débit : Compte TVA Déductible (445xxx) pour le Total TVA.
 - Crédit : Compte Fournisseur (401xxxx) pour le Montant TTC.
 - L'écriture est générée avec la date de réception ou date de facture (selon paramétrage/règle TVA), le N° Facture F. comme pièce/libellé.
 - Enregistrement de la FactureAchat et de l'écriture comptable en base dans une transaction.
 4. **Feedback** :
 - *Si Succès* : Message "Toast" succès ("Facture Achat enregistrée et comptabilisée"). Redirection vers la liste des factures achats ou affichage du détail de la facture saisie.
 - *Si Erreur* : Réponse HTMX affichant les erreurs sur le formulaire. L'utilisateur corrige.

9.3. Flux : Suivre les Échéances Fournisseurs

1. **Action** : Naviguer vers "Achats" > "Échéancier Fournisseurs" (ou une vue spécifique dans la liste des factures).
2. **Interface** : Tableau listant les factures fournisseurs **non soldées**, triées par date d'échéance (les plus proches en premier).
 - o *Filtres* : Par Fournisseur, Par Période d'Échéance (Ex: Prochains 7 jours, 30 jours, En retard).
 - o *Tableau* : Colonnes : Date Échéance, Fournisseur, N° Facture F., Montant Dû, Jours de Retard (si échéance passée). Actions (Enregistrer Paiement, Voir Facture).
3. **Interaction** : Permet d'identifier rapidement les factures à payer. Le filtrage/tri via HTMX.

9.4. Flux : Enregistrer un Paiement Fournisseur (Assistant+/Comptable)

1. **Action** : Soit depuis l'Échéancier, soit depuis la Fiche Fournisseur, soit depuis la vue d'une Facture Achat, soit via "Achats > Paiements Fournisseurs > Enregistrer".
2. **Interface** : Formulaire "Nouveau Paiement Fournisseur".
 - o Champ Fournisseur: (Sélection/Autocomplete).
 - o Champ Date Paiement: (Datepicker).
 - o Champ Montant Payé: (Numérique).
 - o Champ Compte de Trésorerie: (Select/Autocomplete sur comptes Banque/Caisse débités).
 - o Champ Mode de Paiement: (Select: Virement, Chèque...).
 - o Champ Référence Paiement: (Texte libre : Ref Virement...).
 - o (Optionnel) Section pour affecter immédiatement ce paiement à une ou plusieurs factures ouvertes de ce fournisseur (similaire à Ventes).
3. **Action** : Remplir le formulaire, cliquer "Enregistrer le Paiement".
4. **Système (Django)** : Valide les données. Crée l'enregistrement DecaissementFournisseur. Génère l'écriture comptable dans le journal de trésorerie approprié (Débit: Compte Fournisseur 401, Crédit: Banque/Caisse 5xx).
5. **Feedback** : Message Succès. Redirection vers la liste des paiements ou la fiche fournisseur.

9.5. Flux : Affecter/Letter un Paiement à une/des Facture(s)

- **Option 1 (Lors de la saisie paiement)** : Comme pour les encaissements, le formulaire de paiement peut inclure une section pour affecter le montant payé à des factures spécifiques. La validation vérifie Total affecté <= Montant Payé. Peut déclencher lettrage si solde.
- **Option 2 (Via Module Lettrage)** : Le paiement crée une ligne au débit du compte 401. La facture crée une ligne au crédit. L'utilisateur utilise le module de lettrage (Flux 6.5) pour rapprocher manuellement la/les ligne(s) de paiement avec la/les ligne(s) de facture.

9.6. Flux : Gérer un Avoir/Note de Crédit Achat

1. **Action** : Via "Achats > Avoirs Fournisseurs > Saisir". Ou lié à une facture d'achat existante.
2. **Interface** : Formulaire similaire à la saisie de facture achat, mais pour un avoir. Sélection Fournisseur, Référence Avoir Fournisseur, Date Avoir. Saisie des lignes de produits/services retournés ou remises (montants négatifs ou comptes spécifiques). Imputation comptable inverse (Crédit Comptes Charges/TVA, Débit Compte Fournisseur).
3. **Action** : Enregistrer l'avoir.
4. **Système (Django)** : Crée l'enregistrement Avoir Achat. Génère l'écriture comptable correspondante dans le Journal des Achats (ou journal dédié). L'avoir peut ensuite être lettré avec la facture initiale ou un paiement.

10. Module Gestion des Stocks - Flux et Spécifications (Simplifié)

Ce module offre un suivi basique des quantités en stock. L'intégration avec

Achats/Ventes peut être limitée dans la version initiale. La valorisation et la comptabilisation automatique des variations de stock peuvent être des évolutions futures.

10.1. Flux : Gérer le Catalogue Articles (CRUD Articles)

1. **Action** : Naviguer vers "Stocks" > "Articles".
2. **Interface (Liste Articles)** : Tableau : Référence Article, Désignation, Unité de Mesure, Quantité en Stock Actuelle, Compte de Stock Associé. Filtres/Recherche. Bouton "Ajouter Article". Actions (Modifier, Voir Mouvements, Supprimer si non utilisé).
3. **Interface (Formulaire Ajout/Modif Article)** :
 - o Champs : Référence (unique), Désignation (*requis*), Unité de Mesure (texte libre ou select), Compte de Stock Général (Select/Autocomplete sur Classe 3 SYSCohada, *requis*), Description longue (optionnel), Prix d'achat/vente par défaut (optionnels).
 - o Boutons "Enregistrer", "Annuler".
4. **Workflow** : CRUD standard. La création/modification enregistre les informations de l'article. La suppression n'est possible que si aucun mouvement de stock n'existe pour cet article.

10.2. Flux : Saisir un Mouvement de Stock Manuel

Utile pour les entrées initiales, les sorties pour consommation interne, ou les corrections d'inventaire simples.

1. **Action** : Naviguer vers "Stocks" > "Mouvements de Stock" > "Nouveau Mouvement".
2. **Interface** : Formulaire "Nouveau Mouvement de Stock".
 - o Champ Article: (Autocomplete/Recherche sur catalogue articles). *Requis*.
 - o Champ Date Mouvement: (Datepicker, défaut: aujourd'hui).
 - o Champ Type de Mouvement: (Select : Entrée (Achat/Inventaire+/Autre), Sortie (Vente/Inventaire-/Autre)). *Requis*.
 - o Champ Quantité: (Numérique, positive). *Requis*.
 - o Champ Coût Unitaire: (Numérique, optionnel - pour info, pas de valorisation comptable auto initialement).
 - o Champ Libellé/Référence: (Texte libre).
3. **Action** : Remplir, cliquer "Enregistrer".
4. **Système (Django)** : Valide les données. Crée l'enregistrement MouvementStock. Met à jour la quantité en stock sur la fiche Article. Ne génère PAS d'écriture comptable dans cette version simplifiée (pas d'inventaire permanent comptabilisé automatiquement).
5. **Feedback** : Message succès. Redirection vers la liste des mouvements.

10.3. Flux : Consulter l'État des Stocks

1. **Action** : Naviguer vers "Stocks" > "État des Stocks". Ou consulter la quantité sur la fiche Article.
2. **Interface** : Tableau listant les articles avec leur quantité en stock actuelle.

- o *Filtres* : Recherche par Référence/Désignation. Option pour afficher articles avec stock nul ou négatif.
- o *Tableau* : Colonnes : Référence, Désignation, Unité, Quantité en Stock. Lien vers détail article ou historique mouvements.

10.4. (Optionnel) Flux : Intégration avec Achats/Ventes

- *Si Implémenté* : Lors de la saisie d'une ligne de facture d'achat ou de vente contenant un article géré en stock, le système pourrait :
 - o Proposer une validation basée sur le stock disponible (pour les ventes).
 - o Générer **automatiquement** un mouvement de stock (Entrée Achat / Sortie Vente) lors de la validation de la facture (ou d'un bon de livraison/réception si ces documents sont gérés). Cela nécessite une configuration claire pour lier les lignes de facture aux articles.

10.5. (Optionnel) Flux : Réaliser un Inventaire

- *Interface* : Écran dédié "Inventaire Physique". Permet de sélectionner des articles (ou tous), d'afficher la quantité théorique (calculée par le système), et de saisir la quantité réelle constatée.
- *Système* : Calcule les écarts. Génère des mouvements de stock de type "Inventaire+" ou "Inventaire-" pour ajuster la quantité théorique à la quantité réelle. Ne génère pas d'écriture comptable pour la valeur de l'écart dans cette version simplifiée.

11. Module Gestion de Trésorerie - Flux et Spécifications

Ce module se concentre sur le suivi des soldes des comptes bancaires et des caisses, ainsi que sur le processus essentiel de rapprochement bancaire. Les mouvements sont principalement issus des écritures enregistrées dans les journaux de type Banque (BQ) et Caisse (CS).

11.1. Flux : Paramétriser les Comptes de Trésorerie (Admin/Sup)

1. **Action** : Utilisateur (Admin/Sup) se connecte, sélectionne le Dossier Client.
2. **Action** : Navigue vers "Paramètres" > "Paramètres Comptables" > Onglet/Section "Comptes de Trésorerie".
3. **Interface** : Tableau listant les comptes de trésorerie déjà créés pour ce dossier. Colonnes : Libellé (ex: BICICI Compte Courant, Caisse Principale), Type (Banque/Caisse), N° Compte Bancaire (si Banque), Compte Comptable Associé (ex: 521100, 571000). Bouton "Ajouter Compte Trésorerie". Actions par ligne (Modifier, Désactiver/Supprimer si non utilisé).
4. **Flux Ajout Compte** :
 - o Clic "Ajouter..." .
 - o Formulaire (modal/section) :
 - Libellé: (Texte, ex: SGBCI Compte XXXXX). *Requis*.
 - Type: (Select: Banque, Caisse). *Requis*.
 - N° Compte Bancaire / IBAN: (Texte, visible si Type=Banque).

- Compte Comptable Général: (Autocomplete>Select sur comptes Classe 5 SYSCohada - 521, 571...). *Requis*. Doit être unique par compte de trésorerie.
 - (Optionnel) Journal Comptable Associé: (Select sur Journaux de type Banque/Caisse définis pour le dossier). Utile pour pré-sélectionner le journal lors de saisies manuelles de trésorerie.
 - o Clic "Enregistrer". Système crée l'enregistrement CompteTresorerie. Tableau mis à jour.
5. **Flux Modification/Désactivation** : Actions standards via le tableau et formulaire de modification. La suppression n'est possible que si aucune écriture n'est liée au compte comptable associé.

11.2. Flux : Effectuer un Rapprochement Bancaire (Comptable)

Ce flux permet de comparer les écritures comptables enregistrées dans un journal de banque avec les opérations figurant sur le relevé bancaire correspondant pour identifier les écarts et valider les soldes.

- **11.2.1. Sélection Compte et Période :**
 1. **Action** : Utilisateur (Comptable) se connecte, sélectionne le Dossier Client.
 2. **Action** : Navigue vers "Trésorerie" > "Rapprochement Bancaire".
 3. **Interface** : Écran initial de sélection :
 - Champ Compte Bancaire: (Select/Autocomplete listant les CompteTresorerie de type "Banque" paramétrés). *Requis*.
 - Champ Date du Relevé Bancaire: (Datepicker, correspond à la date de fin du relevé). *Requis*.
 - Champ Solde Final Relevé Bancaire: (Numérique). *Requis*.
 - Bouton "Démarrer le Rapprochement".
 4. **Action** : Sélectionne le compte, saisit la date et le solde du relevé, clique "Démarrer..." .
- **11.2.2. Interface de Pointage (Comparaison Compta/Relevé) :**
 1. **Système (Django)** : Récupère :
 - Le solde comptable du compte sélectionné à la date du relevé.
 - Toutes les LigneEcriture non encore rapprochées pour ce compte comptable jusqu'à la date du relevé.
 - Les informations du dernier état de rapprochement validé (s'il existe) pour ce compte.
 2. **Interface** : Affichage de l'écran de rapprochement principal :
 - **En-tête** : Rappel du Compte Bancaire, Date du Relevé, Solde Relevé Bancaire saisi. Affichage du Solde Comptable calculé à cette date. Calcul et affichage de l'Écart initial (Solde Relevé - Solde Comptable).
 - **Zone Principale (Layout côté-à-côte ou tableau unifié)** :
 - **Colonne/Tableau "Écritures Comptables Non Rapprochées"** : Liste les lignes d'écriture (Date, Libellé, Montant Débit, Montant Crédit) avec une Checkbox "Pointer" à côté de chaque ligne.
 - **Colonne/Tableau "Opérations Bancaires (Importées/Saisies)" (Simplifié initialement)** : Dans un

premier temps, cette partie peut être absente si l'on ne fait qu'un pointage manuel côté compta. Si un import de relevé (CSV/OFX) est implémenté ultérieurement, ce tableau affichera les lignes du relevé avec une checkbox. Pour la version manuelle, on peut juste avoir une zone de totaux.

- **Zone de Synthèse/Contrôle (Mise à jour dynamique via HTMX/Alpine) :**
 - Solde Comptable Initial (non rapproché).
 - Total Débits Comptables Pointés.
 - Total Crédits Comptables Pointés.
 - Solde Comptable Rapproché Théorique (Solde Initial + Débits Pointés - Crédits Pointés).
 - Écart Final (Solde Relevé Bancaire - Solde Comptable Rapproché Théorique). Doit tendre vers 0.
- **Boutons d'Action** : "Enregistrer l'État de Rapprochement" (actif seulement si Écart Final = 0 ou proche de 0 avec justification ?), "Sauvegarder en Brouillon", "Annuler".
- **11.2.3. Workflow de Pointage et Validation :**
 1. **Action** : L'utilisateur coche les checkboxes "Pointer" à côté des écritures comptables qui apparaissent sur son relevé bancaire papier/PDF.
 2. **Système (HTMX/Alpine)** : À chaque coche/décoche, la Zone de Synthèse/Contrôle est recalculée et mise à jour (Totaux Pointés, Solde Rapproché, Écart Final).
 3. **Action** : L'utilisateur continue le pointage jusqu'à ce que l'Écart Final soit nul (idéalement) ou expliqué.
 4. **Action** : Si l'écart est nul, il clique sur "Enregistrer l'État de Rapprochement".
 5. **Système (Django)** :
 - Crée un enregistrement EtatRapprochementBancaire (avec compte, date relevé, solde relevé).
 - Pour chaque ligne d'écriture pointée (checkbox cochée), met à jour un flag rapprochee = True ou lie la ligne à l'enregistrement EtatRapprochementBancaire.
 - Enregistre le tout en base.
 6. **Feedback** : Message "Toast" succès ("Rapprochement enregistré pour le compte XXX au JJ/MM/AAAA"). Redirection vers une vue de consultation des rapprochements ou retour à l'écran de sélection. Les écritures rapprochées ne réapparaîtront pas lors du prochain rapprochement pour ce compte.
 7. **Option "Sauvegarder en Brouillon"** : Permet d'enregistrer l'état du pointage (quelles lignes sont cochées) sans marquer les écritures comme définitivement rapprochées, pour reprendre plus tard.

11.3. Flux : Consulter les Soldes et Mouvements de Trésorerie

1. **Action** : Naviguer vers "Trésorerie" > "Situation de Trésorerie" ou "Consultation des Comptes".
2. **Interface** :

- o *Filtres* : Sélection du/des Compte(s) de Trésorerie (Banque/Caisse), Période (Date Début/Fin).
 - o *Affichage* :
 - Soit un tableau de bord simple affichant le solde actuel de chaque compte de trésorerie.
 - Soit un tableau type "Relevé de Compte" (similaire au Grand Livre mais pour un compte 5xx) affichant le solde initial et tous les mouvements (débit/crédit) sur la période sélectionnée, avec solde progressif.
3. **Interaction** : Sélection des filtres -> Mise à jour de l'affichage via HTMX.
-

12. Module Gestion Budgétaire - Flux et Spécifications (Simplifié)

Ce module permet la saisie de prévisions budgétaires et une comparaison simple avec les dépenses/recettes réelles enregistrées en comptabilité.

12.1. Flux : Définir un Budget (Comptable/Sup)

1. **Action** : Naviguer vers "Budget" > "Définition des Budgets".
2. **Interface** :
 - o *Sélection Contexte* : Choisir l'Exercice Comptable pour lequel définir le budget. Bouton "Créer un Nouveau Budget" (si plusieurs budgets possibles par exercice, ex: Initial, Révisé).
 - o *Grille de Saisie Budgétaire* : Tableau affichant les comptes de Charges (Classe 6) et de Produits (Classe 7) pertinents (possibilité de filtrer/sélectionner les comptes à budgéter). Colonnes : N° Compte, Intitulé Compte, Champ de saisie "Montant Prévu Annuel". (Optionnel) Possibilité de détailler par mois ou trimestre.
3. **Action** : L'utilisateur saisit les montants prévus pour chaque compte pertinent dans la grille.
4. **Action** : Clic sur "Enregistrer le Budget".
5. **Système (Django)** : Enregistre les données dans les modèles Budget et LigneBudgetaire.
6. **Feedback** : Message succès.

12.2. Flux : Consulter le Suivi Budgétaire (Comparatif Réel/Prévu)

1. **Action** : Naviguer vers "Budget" > "Suivi Budgétaire".
2. **Interface** :
 - o *Filtres* : Sélection du Budget à consulter, Période d'analyse (Mois, Trimestre, Année en cours), Niveau de détail (par compte, par groupe de comptes?). Bouton "Afficher Rapport".
 - o *Zone de Résultat* : Tableau comparatif.
 - o *Tableau* : Colonnes : N° Compte, Intitulé Compte, Montant Budgété (pour la période), Montant Réalisé (calculé depuis les écritures comptables), Écart (Montant ou %), Taux de Réalisation (%). Lignes pour chaque compte budgéte. Totaux par classe/global.
3. **Action** : Sélectionner filtres, cliquer "Afficher".
4. **Système (HTMX/Django)** : Requête -> Vue Django récupère les prévisions du

budget sélectionné, agrège les montants réels depuis les écritures comptables pour la période et les comptes concernés, calcule les écarts/taux. Renvoie le fragment HTML du tableau.

5. **Interface** : Mise à jour de la zone de résultat avec le tableau comparatif. Options d'export basiques (CSV).

13. Module Reporting et États Financiers - Flux et Spécifications

Ce module est dédié à la génération et à la consultation des rapports comptables et financiers, avec un accent particulier sur la production des états financiers conformes au format SYSCohada Révisé (TAFIRE).

13.1. Flux : Générer un État Financier SYSCohada (TAFIRE) (Comptable/Sup)

Ce flux permet de produire les états officiels (Bilan, Compte de Résultat, Tableau de Flux de Trésorerie) pour un exercice donné.

1. **Action** : Utilisateur (Comptable/Sup) se connecte, sélectionne le Dossier Client.
2. **Action** : Navigue vers "Reporting" > "États Financiers SYSCohada".
3. **Interface** : Page de génération des états.
 - o *Section Paramètres* :
 - Champ Exercice Comptable: (Select listant les exercices clôturés pour ce dossier client). La génération sur exercice non clos peut être permise mais avec un avertissement clair "PROVISOIRE". *Requis*.
 - Champ État(s) à Générer: (Checkboxes ou Multi-select : Bilan, Compte de Résultat, Tableau Flux de Trésorerie). Sélection multiple possible. *Requis*.
 - (Optionnel) Comparer avec Exercice N-1: (Checkbox, si les données N-1 sont disponibles et importées/calculées).
 - (Optionnel) Niveau de Détail: (Select : Système de base, Système allégé - si pertinent et géré).
 - o *Bouton* : "Générer les États Sélectionnés".
4. **Action** : L'utilisateur sélectionne l'exercice, les états souhaités, et clique sur "Générer...".
5. **Système (Django - Backend)** :
 - o Déclenche une tâche potentiellement longue (surtout pour le TFT). Utilisation d'un indicateur de chargement visible côté client (HTMX peut afficher un message "Génération en cours..."). Pourrait idéalement utiliser une tâche asynchrone (Celery) pour les très gros dossiers, avec notification à l'utilisateur une fois terminé.
 - o **Validation** : Vérifie que l'exercice sélectionné est valide et que les données comptables nécessaires existent et sont cohérentes (ex: balance équilibrée).
 - o **Calculs** : Pour chaque état demandé :
 - **Bilan** : Agrège les soldes finaux des comptes de classe 1 à 5 selon la structure Actif/Passif du modèle SYSCohada TAFIRE. Si N-1 demandé, récupère aussi les soldes N-1.

- **Compte de Résultat** : Agrège les soldes des comptes de classe 6 et 7 selon la structure du modèle SYSCohada TAFIRE (par nature). Calcule les soldes intermédiaires de gestion et le résultat net. Si N-1 demandé, idem.
- **Tableau Flux de Trésorerie (TFT)** : C'est le plus complexe. Implémente la méthode indirecte (recommandée) ou directe si spécifiée. Nécessite les soldes N et N-1 du Bilan, le Compte de Résultat N, et potentiellement des informations détaillées sur les variations de certains postes (immos, stocks, créances, dettes, capitaux propres) extraites des mouvements du Grand Livre ou d'informations complémentaires.
- **Stockage/Format** : Les états générés peuvent être stockés en base (modèle EtatFinancierGenere) ou générés à la volée. Le format de sortie principal est une **vue HTML** structurée respectant le modèle TAFIRE. Une option d'export PDF/Excel doit utiliser ces données calculées.

6. Feedback & Affichage :

- *Si Erreur (Validation ou Calcul)* : Affiche un message d'erreur détaillé.
- *Si Succès* :
 - Redirection vers une nouvelle page affichant les états générés (Flux 13.2) OU
 - Mise à jour de la zone de contenu (via HTMX) pour afficher directement les états générés (peut être lourd si plusieurs états).
 - Affichage d'un message "Toast" de succès.

13.2. Interface d'Affichage des États Générés

- **Visuel** : Page dédiée affichant un ou plusieurs états financiers.
 - **En-tête** : Rappel du Dossier Client, Exercice concerné, Date de génération. Options "Exporter PDF", "Exporter Excel".
 - **Contenu** : Chaque état (Bilan, Résultat, TFT) est présenté dans une section distincte, sous forme de **tableau(x)** HTML respectant scrupuleusement la structure et les intitulés des lignes du modèle SYSCohada TAFIRE.
 - **Formatage** : Utilisation de styles CSS pour une présentation claire et professionnelle (indentation pour les sous-totaux, lignes de totaux en gras, formatage des nombres avec séparateur de milliers, gestion des signes +/-). Comparaison N/N-1 affichée dans des colonnes adjacentes si l'option a été sélectionnée.

13.3. Flux : Exporter un Rapport (PDF/CSV/Excel)

Ce flux s'applique aux états financiers mais aussi aux autres rapports (GL, Balances...).

1. **Action** : Depuis une vue affichant un rapport (ex: Balance Générale, État Financier), l'utilisateur clique sur un bouton "Exporter PDF" ou "Exporter CSV/Excel".
2. **Système (Django)** :
 - Une requête GET est envoyée à une vue Django spécifique à l'exportation, contenant les mêmes paramètres de filtrage/sélection

- que la vue d'affichage (ex: compte, période, exercice...).
 - o La vue Django **recalcule** les données du rapport (ou les récupère si déjà calculées et stockées).
 - o Elle utilise ensuite une bibliothèque pour générer le fichier dans le format demandé :
 - **PDF** : ReportLab, WeasyPrint (rendu HTML vers PDF), ou autre. Nécessite la création de templates PDF ou la conversion de templates HTML. La mise en page doit être soignée et professionnelle.
 - **CSV/Excel** : Utilisation du module csv de Python ou de bibliothèques comme openpyxl ou pandas pour générer un fichier tabulaire propre.
 - o La vue Django renvoie une HttpResponseRedirect avec le contenu du fichier généré et les **headers HTTP appropriés** (Content-Type, Content-Disposition: attachment; filename="nom_rapport.pdf").
3. **Navigateur Client** : Interprète les headers et propose automatiquement le téléchargement du fichier à l'utilisateur.

13.4. Autres Rapports de Gestion (Spécifications Détailées)

Au-delà des états financiers SYSCohada obligatoires et des consultations comptables de base (GL, Balances), la plateforme peut intégrer des rapports de gestion fournissant des éclairages supplémentaires sur l'activité et la santé financière des clients gérés. Ces rapports exploitent les données déjà présentes dans le système (écritures comptables, factures, tiers).

- **13.4.1. Rapport : Suivi Détailé de la TVA**
 - o *Objectif / Utilité* : Faciliter la préparation des déclarations de TVA périodiques (mensuelles ou trimestrielles) en fournissant un état récapitulatif de la TVA collectée et déductible sur la période. Essentiel pour la conformité fiscale.
 - o *Données d'entrée / Filtres* :
 - Dossier Client: (Contexte implicite via session).
 - Période de Déclaration: (Mois ou Trimestre spécifique, ou Plage de dates). *Requis*.
 - o *Structure du Rapport (Affichage HTML & Export CSV/Excel)* :
 - **En-tête** : Nom du Client, Période de Déclaration.
 - **Section 1 : TVA Collectée (Ventes)**
 - Tableau listant les opérations ayant généré de la TVA collectée sur la période (basé sur écritures au crédit des comptes 443xxx dans le Journal de Ventes).
 - Colonnes : Date Écriture, N° Pièce (Facture Vente), Compte Client, Base HT, Taux TVA, Montant TVA Collectée.
 - Sous-totaux par Taux de TVA.
 - Total Général TVA Collectée sur la période.
 - **Section 2 : TVA Déductible (Achats & Charges)**
 - Tableau listant les opérations ayant généré de la TVA déductible sur la période (basé sur écritures au débit des comptes 445xxx dans les Journaux Achats, OD, etc.).

- Colonnes : Date Écriture, N° Pièce (Facture Achat/Autre), Compte Fournisseur/Tiers, Base HT, Taux TVA, Montant TVA Déductible.
 - Sous-totaux par Taux de TVA.
 - Total Général TVA Déductible sur la période.
- **Section 3 : Synthèse**
 - Total TVA Collectée.
 - Total TVA Déductible.
 - **TVA Nette Due (ou Crédit de TVA)** = TVA Collectée - TVA Déductible.
 - (Optionnel) Report Crédit de TVA période précédente.
 - Montant Final à Payer (ou Crédit à Reporter).
- *Flux Utilisateur* : Navigation vers "Reporting" > "Déclarations Fiscales" > "Suivi TVA". Sélectionner la période, cliquer "Générer". Affichage HTML + Options Export.
- **13.4.2. Rapport : État des Immobilisations et Tableau d'Amortissement**
 - *Objectif / Utilité* : Suivre le parc d'immobilisations de l'entreprise cliente et calculer/vérifier les dotations aux amortissements annuelles conformément aux règles SYSCohada. Nécessite la saisie correcte des acquisitions dans les comptes de Classe 2 et potentiellement des informations complémentaires.
 - *Données d'entrée / Filtres* :
 - Dossier Client: (Contexte implicite).
 - Exercice Comptable: (Sélection de l'exercice pour lequel calculer les amortissements). *Requis*.
 - *Structure du Rapport (Affichage HTML & Export CSV/Excel)* :
 - **En-tête** : Nom du Client, Exercice Comptable concerné.
 - **Tableau des Immobilisations** : Liste des immobilisations enregistrées (idéalement via un sous-module ou des écritures d'achat bien identifiées en Classe 2).
 - Colonnes :
 - Compte d'Immobilisation (ex: 24xxxx).
 - Libellé / Désignation de l'Immo.
 - Date d'Acquisition / Mise en Service.
 - Valeur d'Origine (Brute).
 - Mode d'Amortissement (Linéaire - par défaut SYSCohada, autres modes si gérés).
 - Durée d'Amortissement (Années).
 - Taux d'Amortissement (%).
 - Amortissements Antérieurs Cumulés (au début de l'exercice).
 - **Dotation de l'Exercice (calculée)**.
 - Amortissements Cumulés Fin d'Exercice.
 - Valeur Nette Comptable (VNC) Fin d'Exercice.
 - *Logique de Calcul* : La vue Django associée à ce rapport doit implémenter la logique de calcul de l'annuité d'amortissement (prorata temporis la première année pour mode linéaire) en fonction de la date d'acquisition/mise en service, de la durée et de la valeur brute, tout en tenant compte des amortissements antérieurs.
 - *Flux Utilisateur* : Navigation vers "Reporting" > "Immobilisations &

Amortissements". Sélectionner l'exercice, cliquer "Générer". Affichage HTML + Options Export. Ce rapport est essentiel pour passer l'écriture de Dotation aux Amortissements en fin d'exercice.

- **13.4.3. Rapport : Analyse de Marge Brute (Simplifiée)**
 - *Objectif / Utilité* : Donner une indication de la rentabilité brute générée par les ventes de produits ou services, en comparant les revenus (Classe 7) aux coûts directs associés (certains comptes de Classe 6). Nécessite une bonne imputation des charges.
 - *Données d'entrée / Filtres* :
 - Dossier Client: (Contexte implicite).
 - Période d'Analyse: (Date Début, Date Fin). *Requis*.
 - (Optionnel) Filtre par Produit/Service ou Catégorie Analytique (si une dimension analytique simple est implémentée).
 - *Structure du Rapport (Affichage HTML & Export CSV/Excel)* :
 - **En-tête** : Nom du Client, Période d'Analyse.
 - **Tableau Synthétique** :
 - Ligne 1 : **Chiffre d'Affaires (Ventes - Classe 70)** - Montant Total sur la période.
 - Ligne 2 : **Coût des Marchandises Vendues / Coût des Prestations** (Agrégat de comptes de Classe 6 pertinents, ex: 601 Achats Marchandises, 607 Achats Prestations, + Variation de Stock si gérée). - Montant Total sur la période.
 - Ligne 3 : **Marge Brute** (Ligne 1 - Ligne 2) - Montant et Pourcentage du CA.
 - (Optionnel) **Tableau Détailé par Produit/Catégorie** : Si une dimension analytique ou un lien entre lignes de vente et lignes d'achat/coût est possible, le tableau peut décliner CA, Coût Direct, Marge Brute par item.
 - *Flux Utilisateur* : Navigation vers "Reporting" > "Analyse de Gestion" > "Marge Brute". Sélectionner période, (options), cliquer "Générer". Affichage HTML + Options Export.
- **1.3.4.4. Rapport : Suivi des Postes Clés (Tableau de Bord Financier Personnalisable - Concept Avancé)**
 - *Objectif / Utilité* : Offrir une vue personnalisable de l'évolution de certains postes clés du Bilan ou du Compte de Résultat sur plusieurs périodes (mois, trimestres).
 - *Interface de Configuration (Admin/Sup ou Comptable)* : Permettre de sélectionner des comptes ou groupes de comptes pertinents (ex: Trésorerie nette, Créances clients, Dettes fournisseurs, Chiffre d'affaires, Charges de personnel, Résultat d'exploitation) à inclure dans le tableau de bord.
 - *Données d'entrée / Filtres (Utilisateur)* :
 - Dossier Client: (Contexte implicite).
 - Période d'Affichage: (Ex: 12 derniers mois, Année en cours par mois, 4 derniers trimestres...).
 - *Structure du Rapport (Affichage HTML, potentiellement avec graphiques simples via Chart.js ou similaire)* :
 - **En-tête** : Nom du Client, Période Affichée.
 - **Tableau** :
 - Lignes : Chaque poste clé sélectionné lors de la

- configuration.
- Colonnes : Chaque période (mois/trimestre) de la plage sélectionnée.
- Cellules : Valeur du poste clé (solde ou flux) pour cette période.
- **Graphiques (Optionnel)** : Graphiques linéaires simples montrant l'évolution de certains postes clés sur la période.
- *Flux Utilisateur* : Navigation vers "Reporting" > "Tableau de Bord Financier". La vue charge les données pour les postes configurés et la période par défaut. L'utilisateur peut changer la période. Affichage HTML/Graphique + Options Export (CSV/Excel des données tabulaires).

14. Import / Export de Données - Flux et Spécifications

Ce module décrit les fonctionnalités permettant d'intégrer des données externes dans l'application (principalement lors de la configuration initiale d'un dossier) et d'extraire des données de l'application pour d'autres usages.

14.1. Import Initial des Données (Fonctionnalités Critiques)

Ces fonctionnalités sont essentielles pour démarrer la gestion d'un dossier client qui a déjà une existence comptable. Elles sont typiquement accessibles par les Administrateurs ou Superviseurs via les paramètres du dossier client.

- **14.1.1. Import du Plan Comptable (Si personnalisations spécifiques au client) :**
 - *Contexte* : Uniquement si le Plan Comptable SYSCohada standard ne suffit pas et qu'une structure auxiliaire très spécifique ou des comptes généraux supplémentaires (non standards) sont nécessaires et autorisés (à utiliser avec extrême prudence pour rester conforme). La norme est d'utiliser le plan standard et des comptes auxiliaires.
 - *Interface* : Section dédiée dans les Paramètres Comptables du dossier. Bouton "Importer Plan Personnalisé".
 - *Format Attendu* : Fichier CSV ou Excel (.xlsx) avec colonnes prédéfinies : Numero_Compte, Intitule, Type_Compte (ex: Actif, Passif, Charge, Produit, TiersClient, TiersFournisseur), Compte_Rattachement (pour auxiliaires). Un modèle de fichier doit être téléchargeable.
 - *Workflow* :
 1. Clic "Importer...". Sélection du fichier.
 2. Système (Django) : Lecture du fichier, validation préliminaire du format et des en-têtes.
 3. Interface : Affichage d'une prévisualisation des données lues, avec indication des erreurs potentielles par ligne (ex: N° compte invalide, type inconnu, doublon).
 4. Action : L'utilisateur corrige le fichier source ou confirme l'import (si erreurs non bloquantes).
 5. Système : Tente d'importer les données valides en base (PlanComptableGeneral, CompteAuxiliaire). Reporte les erreurs

détaillées.

6. Feedback : Message récapitulatif (X lignes importées, Y erreurs).

- 14.1.2. Import des Tiers (Clients / Fournisseurs) :

- *Contexte* : Pour créer rapidement la base de données clients/fournisseurs lors du démarrage d'un dossier.
- *Interface* : Section dédiée dans les Paramètres du dossier ou dans le module "Tiers". Bouton "Importer Tiers".
- *Format Attendu* : Fichier CSV/Excel. Colonnes : Type (Client/Fournisseur), Nom_Societe (*requis*), Num_Contribuable, Adresse_L1, Adresse_L2, Code_Postal, Ville, Pays, Telephone, Email, Compte_Collectif_General (*requis*, doit exister dans le plan), Num_Compte_Auxiliaire (optionnel, sinon généré), Conditions_Paiement. Modèle téléchargeable.
- *Workflow* :
 1. Sélection fichier.
 2. Validation format + Prévisualisation avec erreurs (Compte Collectif invalide, Type inconnu...).
 3. Confirmation Import.
 4. Système : Crée les enregistrements Tiers et CompteAuxiliaire associés pour les lignes valides. Reporte les erreurs.
 5. Feedback : Message récapitulatif.

- 14.1.3. Import de la Balance d'Ouverture / Écritures d'À-Nouveaux :

- *Contexte* : Pour enregistrer les soldes initiaux des comptes au début du premier exercice créé dans l'application. C'est une étape comptable fondamentale.
- *Interface* : Section dédiée dans les Paramètres Comptables du dossier, liée au premier exercice ouvert. Bouton "Importer Balance d'Ouverture".
- *Format Attendu* : Fichier CSV/Excel très simple. Colonnes : Numero_Compte (Général ou Auxiliaire si généré), Intitule_Compte (pour vérification), Solde_Debit_Ouverture, Solde_Credit_Ouverture. Une seule des deux colonnes de solde doit être remplie par ligne.
- *Workflow* :
 1. Sélection fichier.
 2. Validation format + Prévisualisation avec erreurs (Compte inconnu, Ligne avec Débit ET Crédit...). Vérification cruciale que **Total Débit Ouverture = Total Crédit Ouverture** dans le fichier.
 3. Confirmation Import (uniquement si le fichier est équilibré).
 4. Système (Django) :
 - Génère une seule écriture comptable dans le journal des Opérations Diverses (OD) ou un journal spécifique "AN - À Nouveaux".
 - Date de l'écriture : Premier jour de l'exercice.
 - Libellé : "Balance d'Ouverture Exercice XXXX".
 - Pour chaque ligne du fichier importé, crée une LigneEcriture associée à cette écriture globale, avec le compte, le montant au débit ou au crédit.
 5. Feedback : Message succès ou échec (si déséquilibre ou autre erreur bloquante). L'écriture d'AN est visible dans les consultations comptables.

14.2. Export des Données (Fonctionnalités Utiles)

Ces fonctionnalités permettent d'extraire les données pour analyse externe, archivage, ou migration future.

- **14.2.1. Export des Listes Principales (Clients, Fournisseurs, Articles, Écritures...) en CSV/Excel :**
 - *Interface* : Sur chaque écran affichant un tableau de données principal (Liste Clients, Liste Factures, Grand Livre, Balance...), un bouton "Exporter CSV" ou "Exporter Excel" est disponible.
 - *Action* : Clic sur le bouton.
 - *Système (Django)* : La vue associée prend en compte les **filtres actuellement appliqués** sur la liste affichée. Elle récupère les données correspondantes de la base. Elle génère le fichier CSV (simple) ou Excel (.xlsx, utilisant openpyxl pour un meilleur formatage) avec les colonnes visibles dans le tableau. Elle renvoie le fichier via HttpResponseRedirect pour téléchargement.
 - *Format* : Encodage UTF-8 pour CSV. Formatage de base pour Excel (en-têtes, types de données).
- **14.2.2. Export des Rapports Comptables :**
 - *Interface* : Sur les écrans affichant les rapports (GL, Balances, États Financiers...), boutons "Exporter PDF" et "Exporter CSV/Excel".
 - *Workflow* : Décrit en Section 13.3. Le PDF vise une présentation formelle, tandis que CSV/Excel fournit les données brutes du rapport.
- **14.2.3. (Optionnel) Export Complet des Données d'un Dossier :**
 - *Contexte* : Fonctionnalité avancée pour administrateurs, utile pour archivage légal ou migration vers un autre système.
 - *Interface* : Option spécifique dans les paramètres avancés du dossier client.
 - *Format* : Pourrait être un ensemble de fichiers CSV (un par modèle de données principal : tiers, écritures, lignes, factures...), un fichier JSON contenant toutes les données sérialisées, ou un dump SQL spécifique à ce dossier (plus complexe à gérer en multi-dossiers). Le choix dépend de la complexité et de l'usage visé.
 - *Workflow* : Action déclenchée par l'admin -> Tâche potentiellement longue (asynchrone ?) -> Génération du/des fichier(s) -> Mise à disposition pour téléchargement.

15. Exigences Non Fonctionnelles - Approfondissement

Ces exigences définissent les qualités du système au-delà de ses fonctionnalités directes (sécurité, performance, etc.). Elles sont cruciales pour la fiabilité et l'acceptation de l'application.

15.1. Sécurité Détaillée

La sécurité est primordiale pour une application manipulant des données financières.

- **Authentification Robuste :**

- Utilisation du système d'authentification de Django avec hachage sécurisé des mots de passe (algorithme par défaut : PBKDF2, Argon2 recommandé si possible).
 - Application stricte de la politique de complexité des mots de passe définie globalement (via AUTH_PASSWORD_VALIDATORS).
 - Protection contre les attaques de type brute-force sur la page de connexion (ex: limitation du nombre de tentatives par IP/utilisateur via django-axes ou équivalent).
 - Gestion sécurisée des sessions (cookies HttpOnly, Secure en production HTTPS, durée de session configurable).
- **Autorisations Granulaires :**
 - Utilisation systématique du framework de permissions de Django. Chaque vue doit vérifier les permissions de l'utilisateur (@permission_required decorator, user.has_perm() method) avant d'autoriser l'accès ou l'action.
 - Les requêtes de base de données doivent systématiquement être filtrées par le dossier_client_id actif stocké en session pour garantir le cloisonnement des données.
 - Les permissions doivent être définies au niveau le plus fin possible (permissions par modèle par défaut, potentiellement permissions au niveau objet si nécessaire pour des cas très spécifiques).
- **Protection contre les Vulnérabilités Web (OWASP Top 10) :**
 - **Injection** : Prévenue par l'utilisation systématique de l'ORM Django (qui paramètre les requêtes SQL) et de l'échappement automatique des templates Django (DTL). Toute utilisation de SQL brut (raw()) doit être scrutée.
 - **Broken Authentication** : Adressée par les points d'authentification robuste ci-dessus.
 - **Sensitive Data Exposure** : Pas de stockage d'informations sensibles non nécessaires. HTTPS impératif en production. Pas de logs contenant des données sensibles.
 - **XML External Entities (XXE)** : Non applicable si aucun XML n'est parsé depuis une source non sûre.
 - **Broken Access Control** : Adressé par l'application rigoureuse des permissions Django dans chaque vue.
 - **Security Misconfiguration** : Configuration correcte du serveur web (Nginx), de Django (ex: DEBUG=False en production, SECRET_KEY sécurisée et hors du code source), des headers de sécurité HTTP.
 - **Cross-Site Scripting (XSS)** : Prévenu par l'échappement automatique de DTL. Toute injection de HTML via des variables doit utiliser |safe avec une extrême prudence et seulement sur du contenu entièrement contrôlé ou nettoyé. Utilisation de Content-Security-Policy header.
 - **Insecure Deserialization** : Éviter la déserialisation de données provenant de sources non sûres.
 - **Using Components with Known Vulnerabilities** : Surveillance et mise à jour régulière de Django et de toutes les dépendances Python (requirements.txt) et JavaScript (HTMX, Alpine). Utilisation d'outils de scan de dépendances (ex: pip-audit, safety, npm audit si des dépendances JS sont gérées via npm).
 - **Server-Side Request Forgery (SSRF)** : Non applicable directement sauf

si l'application fait des requêtes vers des URLs externes fournies par l'utilisateur.

- **Protection CSRF (Cross-Site Request Forgery)** : Utilisation systématique du middleware CSRF de Django et du tag {% csrf_token %} dans tous les formulaires soumis via POST. Vérification pour les requêtes HTMX utilisant POST/PUT/DELETE.
- **Headers de Sécurité HTTP** : Configuration via Nginx ou middleware Django pour inclure : Strict-Transport-Security (HSTS), Content-Security-Policy (CSP - à définir restrictivement), X-Content-Type-Options: nosniff, X-Frame-Options: DENY ou SAMEORIGIN, Referrer-Policy: same-origin ou strict-origin-when-cross-origin.
- **Logs de Sécurité** : Configuration du logging Django pour enregistrer les événements de sécurité : tentatives de connexion réussies/échouées, accès refusés (403), erreurs serveur (500), actions administratives critiques.

15.2. Performance et Scalabilité

L'application doit être réactive pour l'utilisateur et capable de gérer un nombre croissant de dossiers clients et de données.

- **Objectifs de Temps de Réponse Indicatifs :**
 - Affichage Vues Listes (Tableaux) : < 1.5s (pour ~50 lignes).
 - Affichage Formulaires Simples : < 0.8s.
 - Réponses Requêtes HTMX (hors réseau) : Objectif < 400ms pour validation/mise à jour simple.
 - Génération Rapports Simples (Balance) : < 10s (pour volume données standard).
 - Génération États Financiers : < 1 min (doit rester acceptable).
- **Optimisation Base de Données :**
 - Utilisation judicieuse de select_related (pour les relations ForeignKey) et prefetch_related (pour les relations ManyToMany ou Reverse ForeignKey) dans les vues Django pour minimiser le nombre de requêtes SQL.
 - Ajout d'index de base de données (db_index=True sur les modèles) sur les champs fréquemment utilisés dans les clauses WHERE (filtres) ou ORDER BY (tris), notamment les clés étrangères, les dates, les statuts.
 - Analyse périodique des requêtes lentes (ex: via django-debug-toolbar en développement, outils de monitoring PostgreSQL en production).
- **Mise en Cache :**
 - Utilisation du système de cache de Django (Memcached ou Redis recommandé en production) pour :
 - Mettre en cache des requêtes de base de données coûteuses et rarement modifiées (ex: Plan Comptable).
 - Mettre en cache des fragments de templates coûteux à rendre.
 - Mettre en cache des résultats de calculs de rapports intermédiaires.
 - Configuration agressive du cache navigateur pour les assets statiques (CSS, JS, images) via Nginx (expires, Cache-Control).
- **Optimisation Frontend :**
 - Minification et concaténation des fichiers CSS et JS en production.

- Chargement efficace des ressources (ex: defer pour les scripts non critiques).
- Optimisation des images (format, compression).
- **Requêtes Asynchrones** : Pour les tâches > 10-15 secondes (génération gros PDF, imports/exports volumineux), implémentation avec Celery et un broker (Redis/RabbitMQ) pour les exécuter en arrière-plan sans bloquer la requête HTTP. L'utilisateur est notifié quand la tâche est terminée (ex: via WebSockets/Server-Sent Events ou simple rafraîchissement/polling).
- **Scalabilité Serveur** : L'architecture Django/Gunicorn/Nginx permet une scalabilité horizontale (ajout d'instances de serveur d'application) si la charge augmente, à condition que la base de données et le cache puissent suivre.

15.3. Maintenabilité et Qualité du Code

Assurer la maintenabilité de l'application est crucial pour son évolution future, la correction de bugs et l'ajout de nouvelles fonctionnalités.

- **Conventions de Codage :**
 - Respect impératif du guide de style **PEP 8** pour tout le code Python. Utilisation d'outils de linting (comme Flake8 ou Pylint) et de formatage automatique (comme Black) intégrés à l'environnement de développement (Codespaces/VS Code) pour garantir la cohérence.
 - Adoption de conventions claires et cohérentes pour le nommage des variables, fonctions, classes, modèles Django, et URLs (ex: noms de modèles au singulier, noms d'URLs descriptifs, etc.).
 - Conventions similaires pour HTML (sémantique), CSS (ex: méthodologie BEM ou approche utilitaire cohérente si Tailwind est utilisé), et JavaScript (si du JS spécifique est écrit au-delà d'Alpine/HTMX).
- **Modularité et Couplage :**
 - Organisation logique du code en **applications Django distinctes** pour chaque module fonctionnel majeur (ex: comptabilité, tiers, ventes, achats, core pour les éléments partagés...).
 - Viser un **couplage faible** entre les applications : une application ne devrait pas dépendre excessivement des détails d'implémentation d'une autre. Communication via des signaux Django ou des appels de fonctions/méthodes bien définis si nécessaire.
 - Utilisation de **Managers de Modèles** (`objects = MyManager()`) pour encapsuler la logique de requête complexe liée à un modèle.
 - Utilisation de **Services** (classes Python simples) pour encapsuler la logique métier complexe ou les processus multi-étapes qui ne relèvent pas directement d'un modèle ou d'une vue spécifique.
- **Commentaires et Documentation :**
 - **Code Commenté** : Ajouter des commentaires clairs et concis pour expliquer le *pourquoi* d'un code complexe ou non évident, pas seulement le *comment*. Éviter les commentaires superflus sur du code simple.
 - **Docstrings** : Utiliser des docstrings (conformes PEP 257) pour

- documenter le but, les arguments et les retours des fonctions, méthodes et classes publiques.
- o **Documentation Externe** : Maintenir ce document de spécifications (SSD) à jour au fur et à mesure des évolutions. Rédiger et maintenir un fichier README.md clair à la racine du projet expliquant comment installer, configurer, lancer et tester l'application.
- **Tests Automatisés** : Essentiels pour la fiabilité comptable.
 - o **Stratégie** : Viser une bonne couverture de tests, en particulier pour les modèles (validations, méthodes personnalisées), les formulaires (validation), les vues (logique métier, réponses HTTP), et les services métier.
 - o **Outils** : unittest (intégré à Django) ou pytest (plus puissant, avec pytest-django). Utilisation de factory-boy pour générer facilement des données de test. Utilisation de coverage.py pour mesurer la couverture de tests.
 - o **Types de Tests** :
 - *Unitaires* : Tester des fonctions/méthodes isolées avec des données mockées.
 - *Intégration* : Tester l'interaction entre plusieurs composants (ex: une vue qui utilise un formulaire et un modèle) en utilisant le client de test Django (self.client.get/post). Tester les flux critiques.
 - o **Exécution** : Intégration des tests dans le pipeline de CI/CD pour une exécution automatique à chaque commit/push.
- **Gestion de Version (Git/GitHub)** :
 - o Utilisation systématique de Git pour tout le code source. Dépôt centralisé sur GitHub.
 - o **Workflow de Branches** : Adopter un workflow comme Gitflow ou GitHub Flow (ex: branche main pour la production, branche develop pour l'intégration, branches de fonctionnalités feature/xxx, branches de correction fix/xxx).
 - o **Pull Requests (PR)** : Utiliser les PR pour intégrer le code des branches de fonctionnalités/corrections dans develop (ou main). Nécessite une revue de code par un autre membre de l'équipe (si applicable) et la réussite des tests automatisés (via CI) avant la fusion.
 - o **Tags de Release** : Marquer les versions déployées en production avec des tags Git (ex: v1.0.0).
- **Gestion des Dépendances** :
 - o Utiliser un fichier requirements.txt (ou pyproject.toml avec Poetry/PDM) pour lister **toutes** les dépendances Python avec leurs versions exactes (pip freeze > requirements.txt).
 - o Utiliser des **environnements virtuels** Python (venv) systématiquement pour isoler les dépendances du projet. Codespaces le fait généralement automatiquement.
 - o Surveillance régulière des vulnérabilités connues dans les dépendances (ex: via pip-audit, safety, ou les alertes de sécurité GitHub Dependabot). Planifier les mises à jour des dépendances.

15.4. Fiabilité et Disponibilité

L'application doit être digne de confiance et accessible lorsque les utilisateurs en ont

besoin.

- **Gestion Robuste des Erreurs :**
 - Utiliser les blocs try...except de manière appropriée dans le code Python pour gérer les erreurs attendues (ex: ObjectDoesNotExist, erreurs de validation, erreurs d'API externes si pertinent) sans faire planter l'application.
 - Configurer Django pour afficher des pages d'erreur personnalisées et conviviales (mais non verbeuses sur les détails techniques) pour les erreurs 403 (Interdit), 404 (Non Trouvé), et 500 (Erreur Serveur) en mode production (DEBUG=False).
 - **Logging Détailé des Erreurs** : Configurer le système de logging de Django pour enregistrer toutes les exceptions non interceptées (niveau ERROR ou CRITICAL), y compris la traceback complète, dans un fichier de log ou un système de logging centralisé. Intégration avec des outils de suivi d'erreurs comme Sentry fortement recommandée.
- **Transactions de Base de Données :**
 - Utiliser systématiquement les **transactions atomiques** de Django pour toutes les opérations qui modifient plusieurs enregistrements liés et doivent réussir ou échouer en bloc. Le comportement par défaut de Django (ATOMIC_REQUESTS=True dans les BDD supportées comme PostgreSQL) est souvent suffisant pour les vues standard, mais pour les logiques plus complexes dans les services ou commandes, utiliser transaction.atomic() explicitement.
 - Exemple critique : La validation d'une facture doit créer la facture ET l'écriture comptable associée dans la même transaction. Si l'un échoue, l'autre doit être annulé (rollback).
- **Disponibilité (Objectifs en Production) :**
 - Viser un taux de disponibilité élevé (ex: > 99.9%, "trois neuf"). Cela dépend fortement de la qualité de l'infrastructure d'hébergement et des processus de déploiement/maintenance.
 - Mettre en place un monitoring de disponibilité (uptime checks) depuis un service externe pour être alerté rapidement en cas d'indisponibilité.
 - Prévoir des procédures de maintenance planifiée avec communication aux utilisateurs pour minimiser l'impact.

16. Déploiement et Exploitation - Considérations

Cette section détaille les aspects pratiques liés à la mise en production de l'application et à sa gestion opérationnelle au quotidien.

16.1. Infrastructure Cible (Production)

L'environnement de production recommandé pour assurer la fiabilité, la performance et la sécurité de l'application Django est le suivant :

- **Serveur(s) d'Application** : Un ou plusieurs serveurs virtuels ou physiques sous un système d'exploitation Linux stable et maintenu (Distribution

- recommandée : Ubuntu LTS ou Debian Stable).
- **Serveur Web Frontal (Reverse Proxy)** : Nginx est fortement recommandé. Ses rôles incluront :
 - Terminaison des connexions SSL/TLS (gestion des certificats HTTPS).
 - Service direct et efficace des fichiers statiques (/static/, /media/).
 - Proxy inverse transmettant les requêtes dynamiques au(x) serveur(s) d'application WSGI.
 - Load Balancing (répartition de charge) si plusieurs instances du serveur d'application sont déployées.
 - Mise en cache éventuelle de certaines réponses HTTP.
 - Application de limitations de débit (rate limiting) ou de règles de sécurité de base.
 - **Serveur d'Application WSGI** : Gunicorn est le standard de facto pour exécuter des applications Django en production. Il sera configuré pour lancer plusieurs processus workers synchrones (ou asynchrones si uvicorn/ASGI est utilisé) afin de gérer les requêtes concurrentes de manière efficace.
 - **Base de Données** : Un serveur PostgreSQL dédié, distinct des serveurs d'application, est le choix privilégié. Il doit être correctement dimensionné (CPU, RAM, Disque IOPS) et configuré pour la performance et la sécurité. L'utilisation d'un service de base de données managé dans le cloud (AWS RDS, Google Cloud SQL, Azure Database for PostgreSQL) est une option très attractive pour déléguer la gestion de l'infrastructure de la base de données (sauvegardes, mises à jour, haute disponibilité). MySQL/MariaDB sont des alternatives viables si l'expertise ou l'infrastructure existante le justifie.
 - **Serveur de Cache** : Un serveur Redis ou Memcached dédié est recommandé pour décharger la base de données et accélérer les temps de réponse via le système de cache de Django. Un service managé (Elasticache, Memorystore...) est également une option pertinente.
 - **Broker de Tâches (si nécessaire)** : Si des tâches asynchrones sont implémentées avec Celery, un broker comme Redis (souvent partagé avec le cache) ou RabbitMQ (plus robuste pour des files d'attente complexes) est requis.
 - **Serveur(s) Worker(s) Celery (si nécessaire)** : Un ou plusieurs serveurs dédiés à l'exécution des processus workers Celery qui consomment les tâches du broker.
 - **Hébergement Physique/Virtuel** : Le choix entre des serveurs dédiés, des Machines Virtuelles (VPS), ou une infrastructure Cloud (IaaS comme EC2/Compute Engine, ou PaaS comme Heroku/Render) dépendra du budget, des compétences internes, des besoins de scalabilité et du niveau de contrôle souhaité. Une approche Cloud IaaS offre généralement le meilleur équilibre entre flexibilité et contrôle pour cette stack.

16.2. Processus de Déploiement

Un processus automatisé et fiable est essentiel pour minimiser les erreurs et les temps d'arrêt lors des mises à jour.

- **Gestion des Environnements** : Maintenir au minimum trois environnements isolés : Développement (local/Codespaces), Staging (pré-production, réplique de la prod), Production (live).

- **Automatisation du Déploiement :**
 - *Infrastructure as Code (IaC)* : Utiliser Terraform ou Pulumi pour définir et provisionner l'infrastructure Cloud de manière reproductible.
 - *Configuration Management* : Utiliser Ansible ou SaltStack pour configurer les serveurs (installation paquets, configuration Nginx/Gunicorn, déploiement code).
 - *Conteneurisation (Optionnel mais recommandé)* : Utiliser Docker pour packager l'application Django et ses dépendances. Utiliser Docker Compose (pour déploiements simples) ou Kubernetes (pour déploiements complexes et scalables) pour orchestrer les conteneurs (application, BDD, cache, workers...).
- **Pipeline CI/CD (Intégration et Déploiement Continus) :**
 - *Outil* : GitHub Actions, GitLab CI, Jenkins, CircleCI...
 - *Déclencheurs* : Push/Merge sur branches develop (déploiement Staging), main ou tags Git (déploiement Production).
 - *Étapes Clés du Pipeline* :
 1. Récupération du code (git checkout).
 2. Installation des dépendances (pip install -r requirements.txt).
 3. Vérification de la qualité du code (Linting flake8, Formatage black --check).
 4. Exécution des tests automatisés (python manage.py test). Un échec ici bloque le pipeline.
 5. Construction de l'artefact (ex: Build image Docker). Push vers un registre d'images (Docker Hub, ECR, GCR).
 6. Déploiement sur l'environnement cible (via SSH/Ansible, docker-compose up, kubectl apply...).
 7. Exécution des commandes post-déploiement sur le serveur cible :
 - Application des migrations de base de données (python manage.py migrate --noinput).
 - Collecte des fichiers statiques (python manage.py collectstatic --noinput).
 8. Redémarrage du/des serveur(s) d'application WSGI (ex: systemctl reload gunicorn).
 9. (Optionnel) Exécution de tests de fumée (smoke tests) pour vérifier que l'application répond correctement.
- **Stratégies de Déploiement Avancées (Zero Downtime) :**
 - *Blue/Green Deployment* : Déployer la nouvelle version sur un environnement "Green" identique à la production "Blue". Après validation, basculer le trafic réseau (via load balancer) vers Green. Permet un rollback instantané.
 - *Rolling Updates* : Mettre à jour les instances de serveur d'application une par une (si plusieurs existent derrière un load balancer), en s'assurant qu'il y a toujours des instances actives pour servir les requêtes pendant la mise à jour.

16.3. Sauvegarde et Restauration

La stratégie de sauvegarde doit garantir la possibilité de récupérer les données en cas d'incident majeur.

- **Base de Données (PostgreSQL) :**
 - *Fréquence* : Sauvegardes complètes au minimum quotidiennes (via pg_dump).
 - *PITR (Point-in-Time Recovery)* : Activation de l'archivage continu des WAL (Write-Ahead Logs) pour permettre une restauration à n'importe quel point dans le temps entre deux sauvegardes complètes.
 - *Stockage* : Externe, sécurisé, redondant géographiquement (ex: AWS S3 avec versioning et lifecycle policies, GCP Cloud Storage...). Chiffrement des sauvegardes au repos et en transit.
 - *Rétention* : Politique claire (ex: 7 derniers jours quotidiens, 4 dernières semaines hebdomadaires, 12 derniers mois mensuels).
 - *Tests* : Procédure de restauration documentée et testée régulièrement (ex: trimestriellement) sur un environnement isolé pour valider l'intégrité et le temps de récupération (RTO).
- **Code Source** : Géré et versionné via Git/GitHub. La sauvegarde est intrinsèque au système Git distribué et à l'hébergement sur GitHub.
- **Fichiers Statiques/Médias** : Si des fichiers sont uploadés par les utilisateurs (MEDIA_ROOT), ce répertoire doit être sauvegardé régulièrement (ex: via rsync vers un stockage externe ou utilisation d'un stockage objet type S3 directement pour les uploads via django-storages).
- **Configuration** : La configuration de l'infrastructure (Terraform/Ansible/Dockerfiles) et de l'application (variables d'environnement, settings.py) doit être versionnée dans Git.

16.4. Monitoring et Logging

Une surveillance proactive est nécessaire pour maintenir la performance et la disponibilité.

- **Monitoring Applicatif (APM) :**
 - *Outil* : Sentry (fortement recommandé pour Django), Datadog APM, New Relic APM...
 - *Fonctionnalités Clés* : Suivi des erreurs/exceptions en temps réel avec contexte complet, mesure des temps de réponse des requêtes web et des requêtes de base de données, identification des goulets d'étranglement (N+1 queries, code lent), suivi des performances des tâches asynchrones (Celery).
- **Monitoring Système/Infrastructure :**
 - *Métriques* : CPU (utilisation, load average), Mémoire RAM (utilisée, swap), Disque (utilisation, IOPS), Réseau (bande passante, latence).
 - *Outils* : Agents systèmes (Datadog Agent, Prometheus Node Exporter...), services managés du fournisseur cloud.
 - *Tableaux de Bord* : Visualisation des métriques via Grafana, Datadog Dashboards, CloudWatch Dashboards...
 - *Alertes* : Configuration d'alertes sur les seuils critiques (CPU > 90% pendant X min, Espace disque < 10%, Taux d'erreur 5xx > Y%).
- **Logging Centralisé :**
 - *Configuration* : Configurer Django (LOGGING setting), Gunicorn, Nginx, PostgreSQL pour envoyer leurs logs vers un système centralisé. Utiliser des formats structurés (JSON) si possible.

- *Outils* : ELK Stack (Elasticsearch, Logstash, Kibana), Graylog, Loki+Grafana, ou services cloud (CloudWatch Logs, Google Cloud Logging, Datadog Logs...).
 - *Bénéfices* : Recherche et analyse faciles de tous les logs, corrélation d'événements entre services, création de tableaux de bord basés sur les logs, alertes sur motifs spécifiques dans les logs.
 - **Monitoring de Disponibilité (Uptime) :**
 - *Outil Externe* : UptimeRobot, Pingdom, Better Uptime, services de monitoring cloud...
 - *Configuration* : Vérifications HTTP(S) régulières (toutes les 1-5 minutes) sur l'URL principale de l'application et potentiellement sur des endpoints de health check spécifiques.
 - *Alertes* : Notifications immédiates (Email, SMS, Slack...) en cas d'indisponibilité détectée.
-

17. Conclusion et Prochaines Étapes

17.1. Résumé des Objectifs Atteints par cette Spécification

Ce document a établi une base de référence détaillée et exhaustive pour le développement de la Plateforme Web de Gestion Intégrée COMPTA FREELANCE. Il précise les objectifs métier, la portée fonctionnelle avec un accent sur la conformité SYSCohada Révisé, les flux de travail utilisateurs anticipés, les directives UI/UX visant une synergie entre modernité et efficacité, l'architecture technique basée sur Django/HTMX/Alpine.js adaptée aux contraintes de développement, ainsi que les exigences non fonctionnelles critiques et les considérations pour le déploiement et l'exploitation. Il sert de guide commun pour toutes les parties prenantes du projet.

17.2. Facteurs Clés de Succès

Le succès de ce projet repose sur plusieurs piliers fondamentaux :

- **Exactitude Métier** : Validation continue et rigoureuse de la logique comptable et de la conformité SYSCohada par des experts du domaine.
- **Approche Itérative** : Développement par modules/fonctionnalités, permettant des livraisons fréquentes et l'intégration rapide du feedback des utilisateurs finaux (collaborateurs COMPTA FREELANCE).
- **Qualité et Tests** : Une stratégie de tests automatisés robuste est indispensable pour garantir la fiabilité des calculs et des processus, minimisant les régressions.
- **Collaboration** : Une communication transparente et régulière entre l'équipe de développement, les experts métier et les futurs utilisateurs.
- **Gestion des Contraintes** : Adaptation pragmatique aux limitations potentielles de l'environnement de développement (Codespaces via mobile) sans compromettre la qualité finale du produit déployé.
- **Adoption Utilisateur** : Une interface utilisateur bien conçue (claire, efficace, cohérente) et une formation adéquate seront nécessaires pour assurer l'adoption par les collaborateurs.

17.3. Prochaines Étapes Logiques (Séquence indicative)

- 1. Revue et Validation Finale** : Ce document doit être revu et formellement approuvé par les représentants métier et techniques de COMPTA FREELANCE.
- 2. Définition du MVP et Priorisation du Backlog** : Identifier le sous-ensemble minimal de fonctionnalités (MVP) absolument nécessaires pour une première version utilisable (ex: Cœur Compta de base, Tiers, Saisie Achats/Ventes manuelles). Créer un backlog détaillé et priorisé des autres fonctionnalités/modules.
- 3. Mise en Place de l'Environnement de Développement** : Finaliser la configuration de l'environnement Codespaces (inclure Python, Django, Node/npm si besoin pour outils frontend, potentiellement conteneur DB PostgreSQL pour tests plus réalistes). Mettre en place la structure initiale du projet Django.
- 4. Développement du Sprint 0 / Itération 1** : Commencer l'implémentation du MVP, en se concentrant sur les fondations (modèles de base, authentification, structure de navigation, configuration dossier client) et le premier module fonctionnel prioritaire, en suivant les spécifications UI/UX et fonctionnelles. Mettre en place les bases des tests automatisés et du pipeline CI simple.
- 5. Démonstrations et Feedback** : Organiser des démonstrations régulières (fin de sprint/itération) aux parties prenantes pour recueillir du feedback et ajuster les priorités du backlog.
- 6. Itérations Suivantes** : Continuer le développement itératif en ajoutant les modules et fonctionnalités selon la priorisation du backlog.

17.4. Évolution Future (Post-Lancement Initial)

Une fois la version initiale déployée et utilisée, des évolutions possibles incluent :

- Fonctionnalités comptables avancées (gestion analytique, rapprochement bancaire automatisé via import de relevés...).
- Améliorations du reporting (plus de rapports de gestion, tableaux de bord plus interactifs).
- Intégrations avec des outils tiers.
- Optimisations de performance continues.
- (Hors scope initial) Un portail client sécurisé.

18. Annexes (Description du Contenu)

Cette section décrit le contenu attendu pour les annexes qui devraient idéalement accompagner ce document de spécifications pour fournir des détails supplémentaires et des références concrètes. Ces annexes ne sont pas générées ici mais leur contenu est défini pour être produit séparément.

18.1. Annexe A : Plan Comptable SYSCohada Révisé de Référence

- **Objectif** : Fournir la liste complète et officielle des comptes du Système Comptable OHADA Révisé, qui sert de base obligatoire pour la structuration

comptable dans l'application.

- **Contenu Attendu :**
 - Un tableau listant tous les comptes généraux définis par l'Acte Uniforme relatif au Droit Comptable et à l'Information Financière.
 - **Colonnes du Tableau :**
 - Classe: (1 à 9)
 - Numéro de Compte: (Numérotation standard SYSCohada, ex: 1011, 4011, 4111, 6011, 7011, 521, 571...).
 - Intitulé du Compte: (Libellé officiel du compte, ex: Capital souscrit - non appelé, Fournisseurs - Achats de biens et services, Clients - Ventes de biens et services, Achats de marchandises A, Ventes de marchandises A, Banques locales, Caisse...).
 - Type de Compte (Indicatif) : Mention si le compte est typiquement utilisé comme compte collectif de tiers (ex: Client, Fournisseur), compte de trésorerie, compte de stock, compte d'immobilisation, compte de charge, compte de produit, compte de capitaux propres.
 - Commentaires / Notes SYSCohada (Optionnel) : Brèves indications sur l'utilisation spécifique du compte selon le référentiel.
- **Format :** Idéalement un fichier séparé (PDF ou Excel/CSV) facilement consultable, ou une section formatée directement dans le document principal si la longueur le permet.
- **Source :** Doit être basée sur la version la plus récente de l'Acte Uniforme OHADA pertinent.

18.2. Annexe B : Wireframes / Maquettes Visuelles Détaillées

- **Objectif :** Illustrer visuellement la structure et la disposition des éléments d'interface pour les écrans clés de l'application, en complément des descriptions textuelles de la section UI/UX.
- **Contenu Attendu :** Une série de schémas basse fidélité (wireframes) ou haute fidélité (maquettes/mockups) pour au moins les écrans suivants :
 - Écran de Connexion.
 - Tableau de Bord Principal (avec exemples de widgets).
 - Layout Général (avec Barre Latérale, Barre Supérieure, Zone de Contenu, Fil d'Ariane).
 - Écran de Liste Type (ex: Liste des Factures Ventes, avec filtres et tableau).
 - Écran de Formulaire de Saisie Complexe (ex: Saisie d'Écriture Comptable, avec en-tête et tableau de lignes dynamique).
 - Écran de Formulaire de Création/Édition Simple (ex: Création d'un Tiers).
 - Écran de Consultation Détailée (ex: Fiche Tiers Vue 360°, Visualisation Facture).
 - Écran de Rapport Type (ex: Affichage Balance Générale).
 - (Optionnel) Écran de Rapprochement Bancaire.
- **Format :** Fichiers images (PNG, JPG) ou PDF exportés depuis un outil de maquettage (ex: Figma, Balsamiq, Sketch, ou même des dessins clairs scannés). Chaque maquette doit être légendée pour indiquer à quel écran/flux elle correspond. L'inclusion directe dans le document principal est préférable

si possible.

18.3. Annexe C : Modèle de Données Physique (Description ou Diagramme)

- **Objectif** : Fournir une vue d'ensemble de la structure de la base de données sous-jacente, montrant les tables principales (correspondant aux modèles Django) et leurs relations.
- **Contenu Attendu** :
 - **Option 1 (Préférable) : Diagramme Entité-Relation (ERD)**
 - Représentation graphique des modèles Django clés (User, Group, DossierClient, ExerciceComptable, JournalComptable, PlanComptableGeneral, CompteAuxiliaire, Tiers, EcritureComptable, LigneEcriture, FactureVente, LigneFactureVente, FactureAchat, LigneFactureAchat, Article, MouvementStock, CompteTresorerie, etc.).
 - Montre les relations entre les modèles (OneToOne, ForeignKey/ManyToOne, ManyToMany) avec indication des clés primaires et étrangères.
 - Peut être généré avec des outils comme django-extensions (graph_models) ou des outils de modélisation de BDD.
 - **Option 2 (Alternative Textuelle) : Description des Modèles Clés**
 - Pour chaque modèle Django principal : Lister ses champs (fields), leur type de données Django (CharField, IntegerField, DateField, ForeignKey, etc.), et leurs contraintes principales (unique=True, null=True, blank=True, default=...).
 - Décrire brièvement le but du modèle et les relations clés avec d'autres modèles.
- **Format** : Image (pour le diagramme) ou section texte structurée.

18.4. Annexe D : Règles de Validation Métier SYSCohada Détaillées

- **Objectif** : Centraliser et détailler les règles de validation spécifiques au SYSCohada qui doivent être implémentées dans l'application, au-delà des validations de format de données classiques.
- **Contenu Attendu** : Une liste structurée de règles, potentiellement groupées par module :
 - **Comptabilité Générale** :
 - Règle d'équilibre stricte Débit = Crédit pour chaque écriture.
 - Utilisation des comptes généraux conformément au plan SYSCohada (ex: pas de mouvement direct sur certains comptes de regroupement).
 - Règles de contrepartie implicites ou obligatoires pour certains types d'opérations.
 - Contraintes sur les dates d'écriture (doivent appartenir à un exercice ouvert).
 - Règles de lettrage (ne lettrer que des écritures sur le même compte tiers, sélection équilibrée).
 - **TVA** :
 - Règles de calcul de la TVA collectée et déductible selon les taux applicables.

- Règles d'exigibilité de la TVA (sur les débits ou les encaissements/décaissements - selon régime).
 - Comptes de TVA à utiliser selon la nature de l'opération.
- **Facturation :**
 - Mentions obligatoires sur les factures (selon législation locale + OHADA).
 - Règles de numérotation séquentielle et sans trou.
- **Amortissements :**
 - Méthodes d'amortissement autorisées (Linéaire principalement).
 - Règles de calcul de l'annuité (prorata temporis).
 - Comptes d'amortissement et de dotation à utiliser.
- **Stocks :**
 - Méthodes de valorisation autorisées (CUMP, FIFO...).
 - Règles de comptabilisation des variations de stock (si inventaire permanent).
- **États Financiers :**
 - Correspondance exacte entre les soldes des comptes et les postes des états TAFIRE.
 - Règles de calcul des soldes intermédiaires de gestion, du résultat, des flux de trésorerie.
- **Format :** Liste textuelle structurée, potentiellement avec des références aux articles pertinents de l'Acte Uniforme SYSCohada. Cette annexe est cruciale et nécessite une validation par un expert comptable SYSCohada.