Cahier des Charges

Application de Reporting

Version 1.0

Projet :
Optimisation des processus décisionnels chez GlobalTrade Inc.

Responsable :
Marie Challet

Entreprise :
GlobalTrade Inc.

9 janvier 2025

Document confidentiel.

Table des matières

Introduction

Ce projet s'inscrit dans le cadre de mon cursus académique, où je suis amené à développer des compétences en conception et développement d'outils décisionnels. L'objectif de cette réalisation est double : répondre aux exigences pédagogiques tout en appliquant concrètement les méthodes apprises. En tant qu'étudiant en alternance, le temps que je peux consacrer au développement est limité à 5 semaines, une contrainte intégrée dans la planification et le déroulement du projet.

Dans ce cadre, j'ai élaboré un scénario fictif, celui de l'entreprise GlobalTrade Inc., pour donner un contexte concret et structuré à cette étude. Cette startup, active dans le domaine du commerce international, est confrontée à des problématiques courantes de gestion et d'optimisation des données, ce qui en fait un cas d'application pertinent pour développer une solution décisionnelle. Ce choix permet de combiner l'approche théorique avec des défis pratiques réalistes.

1 Contexte et Objectifs

L'entreprise GlobalTrade Inc., une startup en pleine croissance dans le domaine du commerce international, cherche à améliorer ses processus décisionnels grâce à une exploitation optimisée de ses données. GlobalTrade Inc. fait appel à un développeur en alternance pour concevoir une solution sur mesure, capable de répondre à ses besoins stratégiques et opérationnels.

GlobalTrade Inc. gère des données issues de multiples sources : suivi des transporteurs, outils de gestion logistique, et fichiers financiers. Cependant, cette dispersion des informations limite sa capacité à analyser efficacement ses performances et à optimiser ses opérations.

1.1 Problématiques Identifiées

Lors des discussions avec les fondateurs et les responsables de GlobalTrade Inc., plusieurs problématiques critiques ont été mises en évidence :

— **Données fragmentées :** Les données sont stockées dans différents outils et formats, rendant difficile leur centralisation pour des analyses globales.

— **Visibilité limitée :** Le manque d'indicateurs consolidés ralentit les prises de décision, en particulier dans les domaines financiers et logistiques.

— **Efforts manuels conséquents :** La génération des rapports repose sur des processus manuels, exposant l'équipe à des erreurs fréquentes et à une charge de travail élevée.

— **Opportunités manquées :** L'absence d'outils analytiques performants empêche d'identifier rapidement les inefficacités et d'exploiter les opportunités d'économie.

1.2 Objectifs du Projet

Pour répondre à ces problématiques, le projet vise à développer une application de reporting accessible et adaptée aux besoins spécifiques de GlobalTrade Inc. Les principaux objectifs sont les suivants :

— **Centralisation des données :** Intégrer et unifier les données provenant de différentes sources dans un entrepôt de données simple et efficace.

— **Visualisations interactives :** Fournir des tableaux de bord clairs et dynamiques, permettant de suivre facilement les indicateurs clés.

— **Automatisation des rapports :** Réduire le travail manuel en générant automatiquement des rapports exploitables.

— **Amélioration des décisions :** Aider les fondateurs et responsables à identifier rapidement les problèmes et à prioriser les actions stratégiques.

1.3 Enjeux Stratégiques

Pour une startup comme GlobalTrade Inc., ce projet revêt une importance particulière dans le cadre de sa croissance et de sa structuration. Les principaux enjeux sont les suivants :

— **Gagner en agilité :** Accélérer les prises de décisions grâce à des indicateurs fiables et consolidés en temps quasi réel.

— **Réduire les coûts :** Identifier et corriger les inefficacités opérationnelles pour optimiser les dépenses.

— **Satisfaire les clients :** Réduire les délais de livraison et offrir une visibilité accrue sur les performances internes.

— **Structurer les données :** Mettre en place une base solide pour la gestion et l'analyse des données, essentielle à la croissance future.

1.4 Valeur Ajoutée du Projet

En tant que développeur, mon rôle est de concevoir une solution intuitive et accessible, capable de transformer les données de GlobalTrade Inc. en un levier de performance. En travaillant étroitement avec l'équipe fondatrice, l'objectif est de fournir un outil évolutif et aligné avec les priorités de la startup.

Perspectives futures : L'application est conçue pour être évolutive, permettant l'ajout ultérieur de fonctionnalités avancées. Parmi celles-ci, l'intégration d'un module prédictif pourrait être envisagée pour anticiper les tendances et améliorer les décisions stratégiques, telles que la prévision des ventes, l'optimisation logistique, et la détection proactive des inefficacités.

Exemple d'impact concret : Avec une centralisation efficace des données, GlobalTrade Inc. pourra réduire les délais liés à la préparation des rapports de 50% et identifier rapidement les leviers d'amélioration opérationnelle, contribuant ainsi à sa compétitivité.

Le succès de ce projet permettra à GlobalTrade Inc. d'optimiser ses processus internes tout en renforçant sa capacité à innover dans un marché compétitif. En mettant en œuvre une infrastructure légère mais robuste, la startup disposera des outils nécessaires pour accompagner sa croissance.

2 Périmètre du Projet

2.1 Inclusions

Le projet couvre toutes les étapes nécessaires à la conception, au développement, et au déploiement d'une application de reporting adaptée aux besoins spécifiques d'une startup, en mettant l'accent sur la simplicité, la rapidité de mise en œuvre, et l'efficacité. Les éléments suivants sont inclus dans le périmètre :

- **Conception et développement de l'application :** Le développement de l'architecture complète de l'application, incluant :
  - Un entrepôt de données simple (Datawarehouse) consolidant les données essentielles des bases sources.
  - Un back-end géré via RShiny, offrant un traitement des données performant et une intégration native avec PostgreSQL.
  - Un front-end interactif généré automatiquement par RShiny, garantissant une expérience utilisateur intuitive.
  - Des tableaux de bord dynamiques et des visualisations interactives pour une analyse claire et rapide.
- **Intégration des bases de données existantes :** Les données seront extraites, transformées et chargées (ETL) à partir des bases PostgreSQL existantes vers le Datawarehouse. Le processus sera optimisé pour garantir une unification rapide et fiable des données.
- **Fonctionnalités clés :** L'application proposera :
  - Des filtres de base pour explorer les données selon des critères essentiels (période, catégorie, région).
  - L'export des rapports en format HTML, avec une possibilité d'ajout ultérieur des formats PDF, CSV et XLSX.
  - Des graphiques interactifs (histogrammes, barres empilées, cartes thermiques) générés à partir des bibliothèques R.
- **Déploiement et formation :** La mise en production sera suivie d'une formation rapide (session de 1 heure) pour garantir une adoption immédiate par les utilisateurs finaux.

2.2 Exclusions

Certaines activités et responsabilités ne sont pas incluses dans le périmètre du projet, notamment :

- **Modification des bases de données sources :** Toute intervention sur le schéma ou la structure des bases de données existantes utilisées par GlobalTrade Inc. est exclue.
- **Maintenance continue :** La maintenance évolutive ou corrective après le suivi post-déploiement devra faire l'objet d'un contrat séparé.
- **Fonctionnalités avancées de gestion des données sources :** Les outils pour la correction ou la validation des données brutes dans les systèmes sources ne font pas partie de ce projet.

2.3 Limitations

Certaines contraintes devront être prises en compte pour garantir la faisabilité et la qualité du projet :

- **Volume de données :** Le Datawarehouse sera dimensionné pour traiter les volumes actuels avec une marge de croissance de 20% sur trois ans. Toute augmentation majeure au-delà de cette prévision nécessitera des ajustements futurs.
- **Technologies utilisées :** Les outils sélectionnés (RShiny, PostgreSQL) devront s'intégrer à l'écosystème technologique existant de la startup, limitant l'introduction de nouvelles technologies complexes.
- **Qualité des données sources :** La qualité et l'exactitude des rapports générés dépendront directement de la qualité des données fournies par les bases sources.
- **Absence d'analyse prédictive :** Bien que le projet actuel se concentre sur l'analyse descriptive et visuelle des données, l'intégration de fonctionnalités prédictives pourra être étudiée dans le cadre d'une future évolution, si nécessaire.

2.4 Bénéfices attendus

Le périmètre défini permettra d'obtenir des bénéfices concrets pour GlobalTrade Inc., contribuant directement à sa croissance :

- **Réduction du temps d'analyse :** L'accès rapide à des tableaux de bord consolidés permettra de passer de plusieurs heures de traitement manuel à une consultation instantanée.
- **Amélioration de la prise de décision :** Les visualisations interactives fourniront une compréhension claire et immédiate des données critiques, facilitant des décisions éclairées.
- **Optimisation des performances opérationnelles :** L'identification rapide des inefficacités dans les processus logistiques contribuera à améliorer la rentabilité et la satisfaction client.

3 Calendrier

Le projet sera réalisé en suivant un calendrier structuré en 3 phases principales, chacune ayant des objectifs clairs et des livrables spécifiques. Le calendrier global s'étend sur **5 semaines**.

3.1 Détail des Phases

- **Phase 1 : Développement (3 semaines)**
  - **Objectifs :**
    - Développer les fonctionnalités essentielles de l'application.
    - Intégrer les bases de données et construire les tableaux de bord interactifs.
  - **Activités :**
    - Développement du back-end (RShiny) avec intégration PostgreSQL.
    - Développement des tableaux de bord et filtres interactifs.

- Mise en place de l'export CSV et des visualisations dynamiques.
- **Livrables :**
  - Version bêta fonctionnelle de l'application.

— **Phase 2 : Tests et Ajustements (1 semaines)**
- **Objectifs :**
  - Valider la conformité de l'application aux spécifications du cahier des charges.
  - Garantir la stabilité et corriger les anomalies critiques.
- **Activités :**
  - Tests unitaires et fonctionnels sur les fonctionnalités principales.
  - Tests utilisateurs pour valider l'ergonomie et l'expérience utilisateur.
  - Ajustements des fonctionnalités critiques.
- **Livrables :**
  - Version prête pour le déploiement.

— **Phase 3 : Déploiement et Formation (1 semaines)**
- **Objectifs :**
  - Déployer l'application dans l'environnement de production.
  - Former les utilisateurs finaux à l'utilisation de l'outil.
- **Activités :**
  - Formation des utilisateurs finaux (session rapide d'1 heure).
- **Livrables :**
  - Application en production et accessible.
  - Guide utilisateur succinct.

3.2 Chronogramme Global

Le projet sera organisé selon le calendrier suivant :

— **Semaine 1-3 :** Développement de l'application.

— **Semaine 4 :** Tests et ajustements.

— **Semaine 5 :** Déploiement et formation.

Des points de suivi hebdomadaires avec la startup permettront de valider l'avancement et d'ajuster le planning si nécessaire.

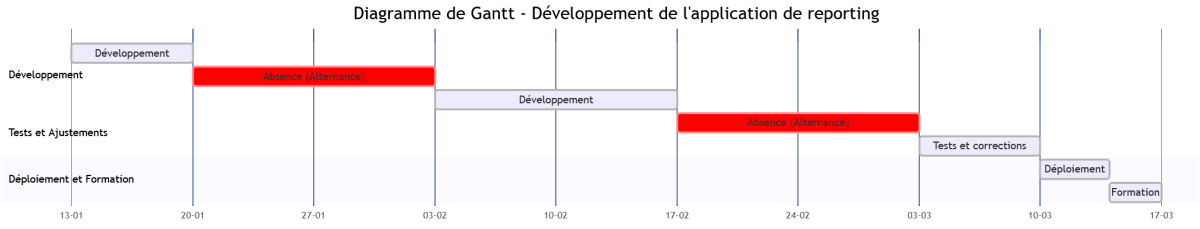


Diagramme de Gantt

4 Exigences Fonctionnelles et Techniques

Cette section détaille les exigences fonctionnelles et techniques de l'application de reporting, établies pour répondre aux besoins spécifiques identifiés lors de l'analyse initiale.

Le diagramme ci-dessous présente les principaux cas d'utilisation de l'application de reporting, mettant en évidence les interactions clés entre les acteurs (utilisateur et administrateur) et les fonctionnalités de l'application.

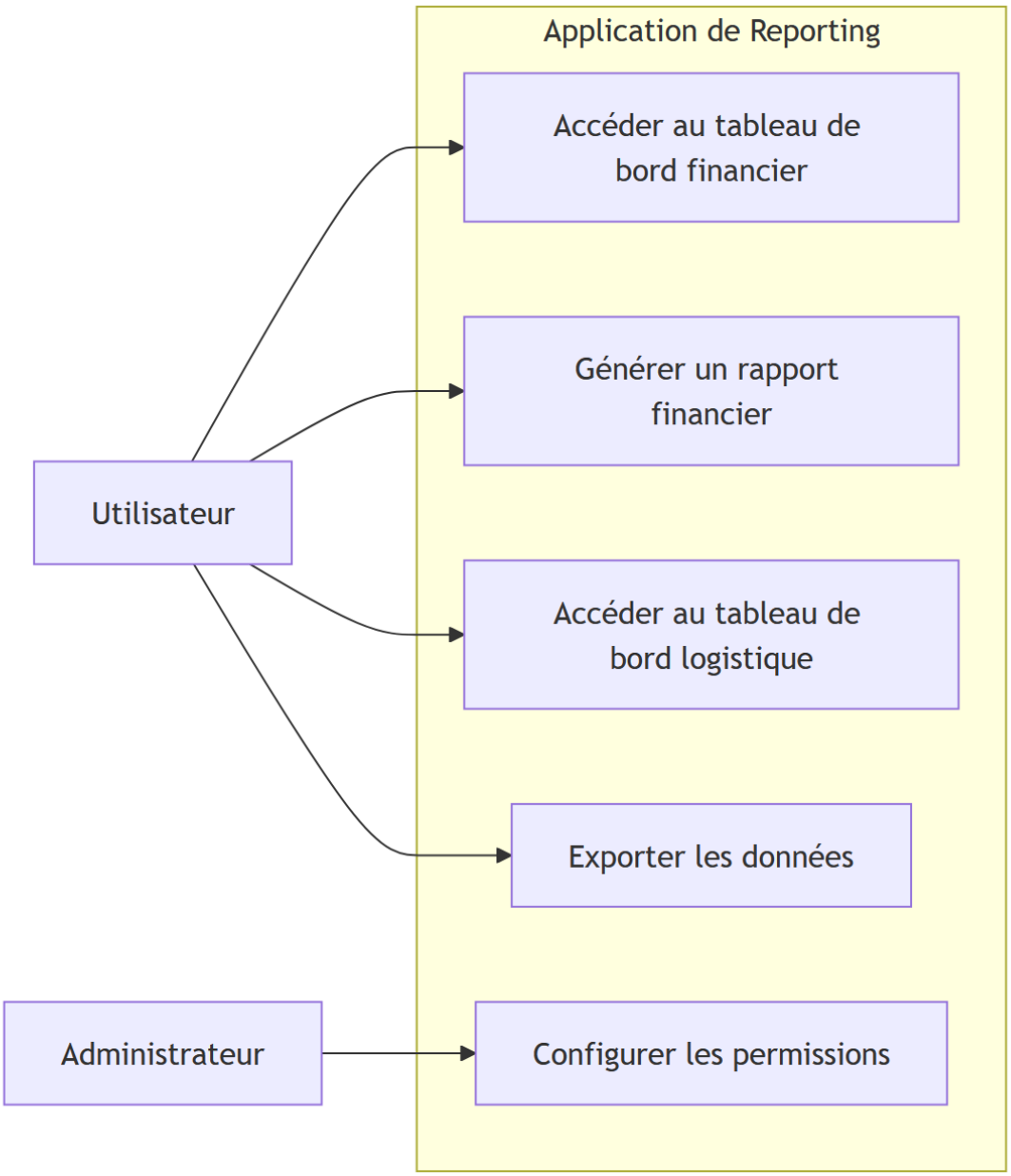


Diagramme de cas d'utilisation

4.1 Exigences Fonctionnelles

4.1.1 Module Financier

Les fonctionnalités suivantes sont nécessaires pour permettre une analyse efficace des données financières :

- **F1 :** Identifier les produits bénéficiant des plus grosses remises. Le rapport inclura :
  - Le client concerné.
  - Le nom de l'employé ayant pris la commande.
  - Un graphique comparatif des remises par produit.
- **F2 :** Générer un tableau détaillé du chiffre d'affaires par produit, avec sous-totaux par catégorie et un histogramme visuel.
- **F3 :** Visualiser le chiffre d'affaires par pays, accompagné d'une carte thermique des frais de port associés.
- **F4 :** Produire un rapport des chiffres d'affaires transportés par transporteur, accompagné d'un graphique sectoriel.
- **F5 :** Afficher la liste des commandes par employé, incluant les remises moyennes accordées par client.
- **F6 :** Consolider les chiffres d'affaires et les quantités vendues par fournisseur, avec une analyse comparative.
- **F7 :** Générer un tableau des frais de port cumulés par transporteur, avec le nombre total d'expéditions.
- **F8 :** Fournir un rapport personnalisé sur les marges bénéficiaires par région, période, et catégorie de produit.

4.1.2 Module Logistique

Pour améliorer la visibilité et l'efficacité des opérations logistiques, les fonctionnalités suivantes sont requises :

- **L1 :** Lister les commandes présentant les plus grands écarts entre la date d'objectif de livraison et la date d'envoi.
- **L2 :** Identifier les produits nécessitant un réapprovisionnement, basés sur les seuils de stock et les commandes en cours.
- **L3 :** Cartographier les quantités de produits livrées par région et par ville, avec des visualisations interactives.
- **L4 :** Mettre en évidence les commandes à livrer dont les fournisseurs sont situés dans la même région que le lieu de livraison, afin d'optimiser la logistique.
- **L5 :** Fournir un rapport personnalisé pour analyser les coûts logistiques par transporteur et par région.

Le diagramme ci-dessous détaille les interactions entre les différentes couches du système lors de la génération d'un rapport, depuis la demande utilisateur jusqu'à la récupération des données dans le datawarehouse.

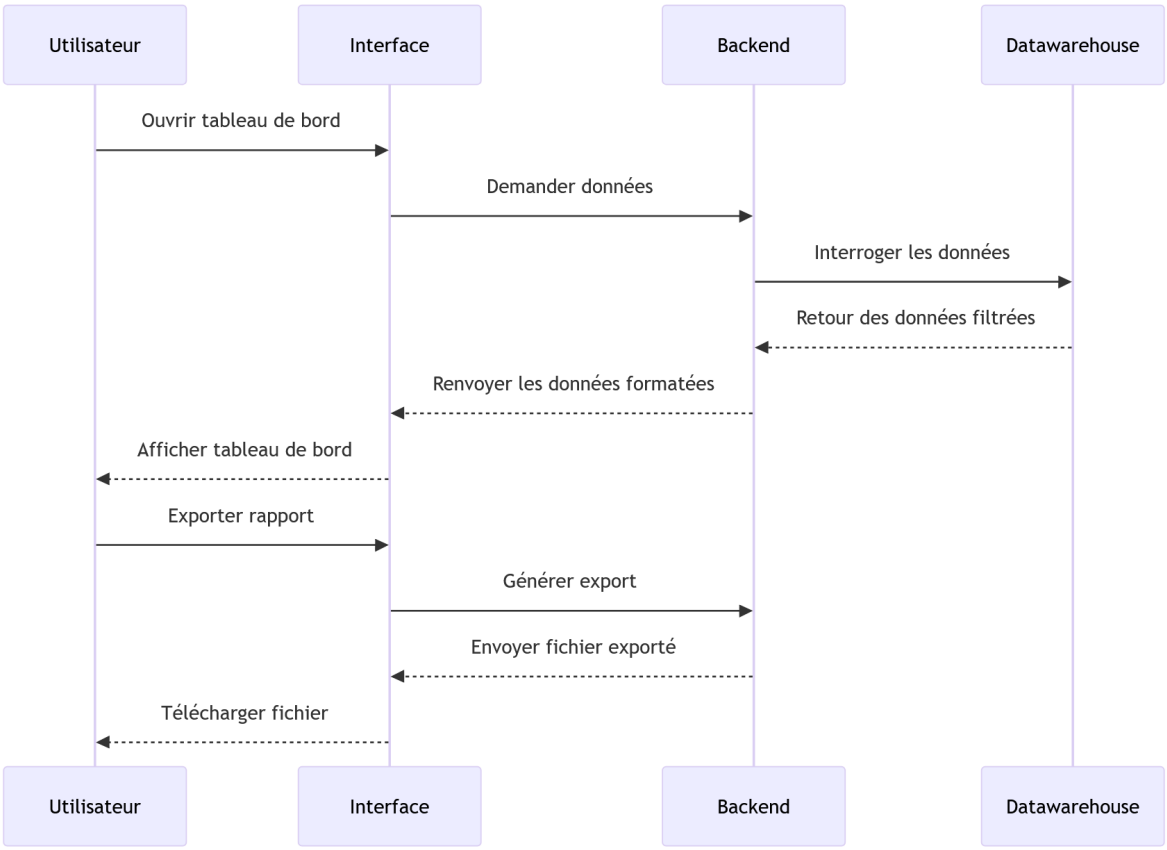


Diagramme de séquence

4.2 Exigences Techniques

4.2.1 Technologies Requises

- **Back-end :**
  - Langage : R (via RShiny) pour le traitement des données et la génération des tableaux de bord interactifs.
  - Base de données : PostgreSQL pour sa robustesse, sa scalabilité, et sa gestion des transactions complexes.
- **Front-end :**
  - Technologie : HTML5 et CSS3
  - Framework : RShiny pour simplifier le développement d'interfaces interactives et fonctionnelles.

4.2.2 Performance et Scalabilité

- Les temps de chargement des tableaux de bord ne devront pas excéder 10 secondes pour les volumes de données actuels.
- L'application devra supporter une augmentation de 20% des volumes de données sans impact significatif sur les performances.

4.2.3 Sécurité et Gestion des Accès

- **Gestion des droits d'accès :**

  - Les droits d'accès seront définis en fonction des départements de l'entreprise, garantissant que chaque utilisateur ne pourra accéder qu'aux données pertinentes à son rôle.
  - Exemple : le département Finance pourra accéder aux rapports financiers (F1-F8), tandis que le département Logistique sera limité aux rapports logistiques (L1-L5).
- Toutes les communications entre le client et le serveur seront sécurisées via HTTPS.
- Les données sensibles, telles que les informations clients et financières, seront chiffrées dans la base de données à l'aide d'un algorithme robuste (par exemple, AES-256).

4.2.4 Compatibilité et Accessibilité

- L'application sera compatible avec les navigateurs modernes (Chrome, Firefox, Edge).
- L'interface sera conçue pour une utilisation intuitive, même sans formation approfondie, avec un respect des standards d'ergonomie (UI/UX).

5 Comparaison des Technologies

Dans le cadre de ce projet, plusieurs technologies ont été envisagées pour répondre aux besoins fonctionnels et techniques identifiés, tout en prenant en compte les contraintes d'un projet mené par un alternant pour une startup en développement. Chaque technologie a été évaluée en fonction de ses avantages, de ses inconvénients, et de sa pertinence dans ce contexte.

5.1 Évaluation des Technologies

Back-end : Python (Django/Flask) Python, via les frameworks Django ou Flask, est réputé pour sa flexibilité et son écosystème riche. Il est particulièrement adapté aux applications nécessitant une architecture modulaire, et il bénéficie d'une large communauté pour le support et les bibliothèques tierces. Cependant, dans un contexte où le développeur doit gérer seul le projet, l'intégration des différentes couches (API, base de données, interface) peut augmenter la complexité et le temps de développement. De plus, Python peut nécessiter des efforts supplémentaires pour implémenter des visualisations avancées.

Back-end : R (Shiny) RShiny est particulièrement efficace pour les projets nécessitant des analyses de données et des visualisations interactives. Il offre une intégration native entre le backend et le frontend, réduisant ainsi la charge de travail. Son approche unifiée simplifie le développement, mais il est moins adapté pour des applications nécessitant une interaction utilisateur très complexe ou une scalabilité massive. Pour un projet où les visualisations sont centrales et les utilisateurs limités, Shiny est un choix optimal.

Front-end : React React est puissant et flexible, mais il nécessite une courbe d'apprentissage plus élevée et une structuration rigoureuse. Il est idéal pour des projets impliquant une forte interaction utilisateur et un frontend sophistiqué. Toutefois, son adoption dans

ce projet ajouterait une couche de complexité supplémentaire, ce qui peut être un frein dans notre contexte.

Front-end : Vue.js Vue.js est plus facile à prendre en main que React, ce qui en fait une option intéressante pour des projets de taille moyenne. Cependant, il reste moins adapté au projet que RShiny.

Base de Données : PostgreSQL PostgreSQL est un choix solide pour ce projet grâce à sa robustesse, sa compatibilité avec R, et ses fonctionnalités avancées pour la gestion des transactions et l'intégration des données complexes. Bien qu'il nécessite une certaine expertise pour une optimisation à grande échelle, il répond parfaitement aux besoins d'une startup avec un volume de données limité mais en croissance.

5.2 Choix des Technologies

À partir des évaluations ci-dessus, les technologies suivantes sont retenues comme les plus adaptées au projet :

— **Back-end et Front-end : R (Shiny).** RShiny est privilégié pour sa capacité à gérer à la fois le backend et le frontend, offrant une solution simplifiée et efficace pour développer des visualisations interactives adaptées aux besoins de la startup.

— **Base de Données : PostgreSQL.** PostgreSQL garantit la stabilité, la fiabilité, et l'évolutivité nécessaires pour consolider les données et les exploiter dans l'application.

5.3 Justification des Choix

Ces technologies ont été sélectionnées en fonction des besoins spécifiques du projet et des contraintes de ressources disponibles :

— **Efficacité pour un développeur unique :** RShiny permet de gérer à la fois le backend et le frontend, réduisant le besoin de coordonner plusieurs technologies.

— **Adapté aux besoins de la startup :** PostgreSQL et RShiny offrent des solutions robustes et évolutives pour gérer des données complexes et produire des visualisations interactives.

— **Coût optimisé :** Toutes les technologies retenues sont open source, limitant les dépenses liées aux licences ou aux infrastructures.

— **Scalabilité raisonnable :** Les solutions choisies permettent une croissance des volumes de données de 20% à court terme, avec la possibilité d'ajuster l'architecture si nécessaire.

— **Focalisation sur les visualisations :** RShiny est conçu pour produire rapidement des visualisations interactives, alignées sur les besoins du projet.

Le diagramme suivant présente l'architecture logicielle de l'application, décrivant les composants clés et leurs interactions.

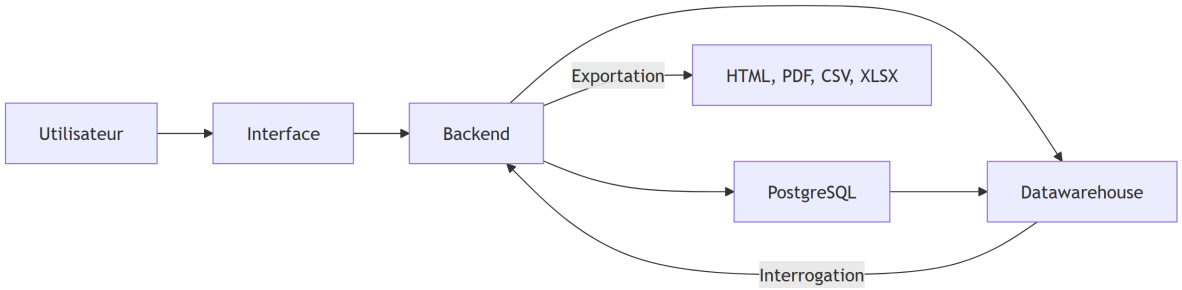


Diagramme de composant

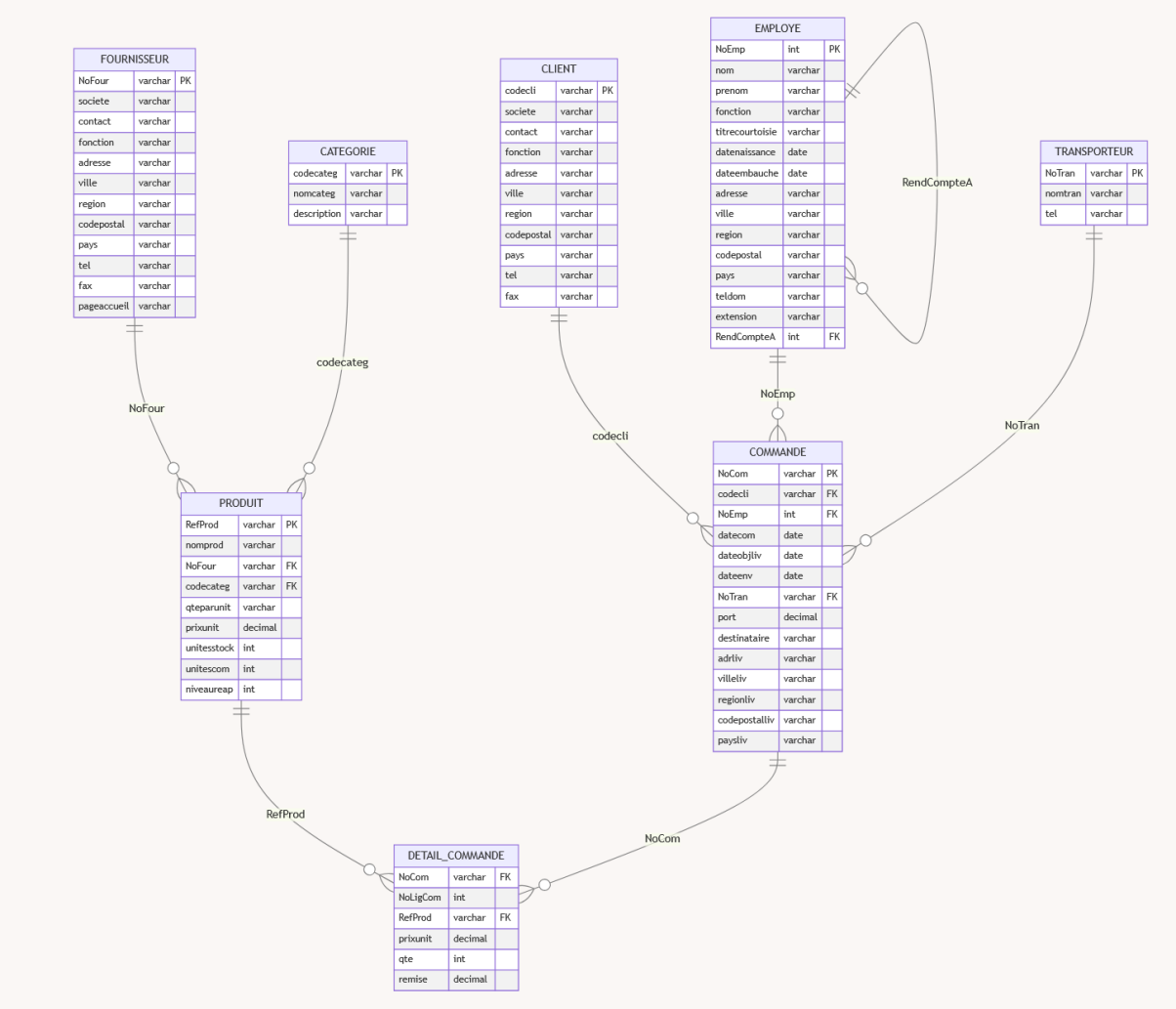
6 Modèles Conceptuels de Données (MCD)

Cette section présente les modèles conceptuels de données élaborés pour structurer les données existantes et celles du Datawarehouse. Ces modèles assurent une organisation logique et facilitent les analyses dans le cadre de l'application de reporting.

6.1 MCD des Données Existantes

Le modèle conceptuel des données existantes reflète l'architecture actuelle utilisée par GlobalTrade Inc. Il représente les relations entre les entités principales extraites des bases de données sources (PostgreSQL) et leurs attributs. Voici les entités clés :

- **CLIENT :** Informations sur les clients (nom, adresse, contact).
- **FOURNISSEUR :** Données sur les fournisseurs (nom, localisation, contact).
- **EMPLOYE :** Informations sur les employés (nom, rôle, hiérarchie).
- **COMMANDE :** Commandes effectuées avec dates et transporteurs associés.
- **DETAIL_COMMANDE :** Produits et quantités incluses dans chaque commande.
- **PRODUIT :** Détails des produits (nom, catégorie, fournisseur).
- **TRANSPORTEUR :** Informations sur les transporteurs (nom, contact).
- **CATEGORIE :** Catégories auxquelles les produits appartiennent.

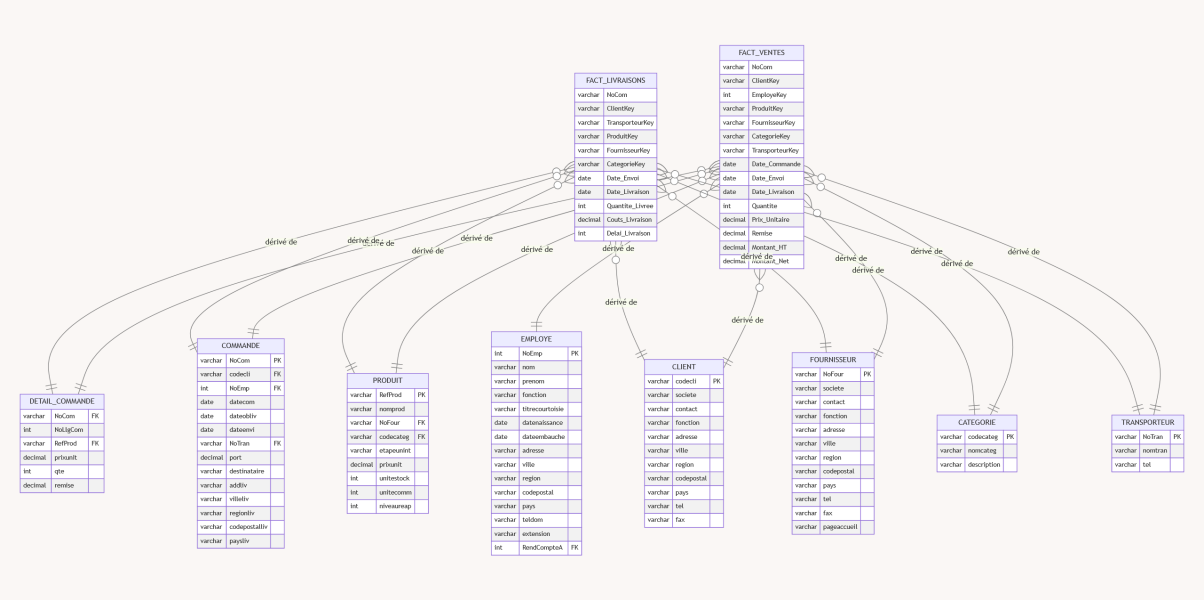


Modèle Conceptuel des Données Existantes

6.2 MCD des Données du Datawarehouse

Le modèle conceptuel du Datawarehouse est conçu pour répondre aux besoins de reporting et d'analyse, en consolidant les données hétérogènes en un système unifié. Les entités principales incluent :

- **FACT_VENTES :** Table factuelle centralisant les ventes, incluant les montants, quantités, remises, et dates.
- **DIM_CLIENT :** Dimension contenant les informations des clients.
- **DIM_FOURNISSEUR :** Dimension décrivant les fournisseurs.
- **DIM_PRODUIT :** Dimension regroupant les caractéristiques des produits.
- **DIM_CATEGORIE :** Catégories pour structurer les produits.
- **DIM_TRANSPORTEUR :** Dimension pour les transporteurs associés aux livraisons.
- **DIM_TEMPS :** Table dimensionnelle pour organiser les données temporelles (jours, mois, années).



Modèle Conceptuel des Données du Datawarehouse.

6.3 Avantages du Modèle Datawarehouse

Le Datawarehouse apporte plusieurs avantages dans le cadre de ce projet :

— **Centralisation des données :** Les informations provenant de multiples bases sont consolidées pour offrir une vue d'ensemble.

— **Optimisation des requêtes :** Les tables dimensionnelles et factuelles sont conçues pour permettre des analyses rapides et efficaces.

— **Adaptabilité :** Le modèle est extensible pour intégrer de nouvelles dimensions ou de nouveaux faits en fonction des besoins futurs.

6.4 Schématisation et Navigation dans les Données

Le Datawarehouse suit un modèle en étoile (*Star Schema*) où les tables dimensionnelles sont connectées à une ou plusieurs tables factuelles. Ce schéma est choisi pour :

— Simplifier les requêtes pour le reporting.

— Minimiser la redondance tout en maintenant des performances élevées.

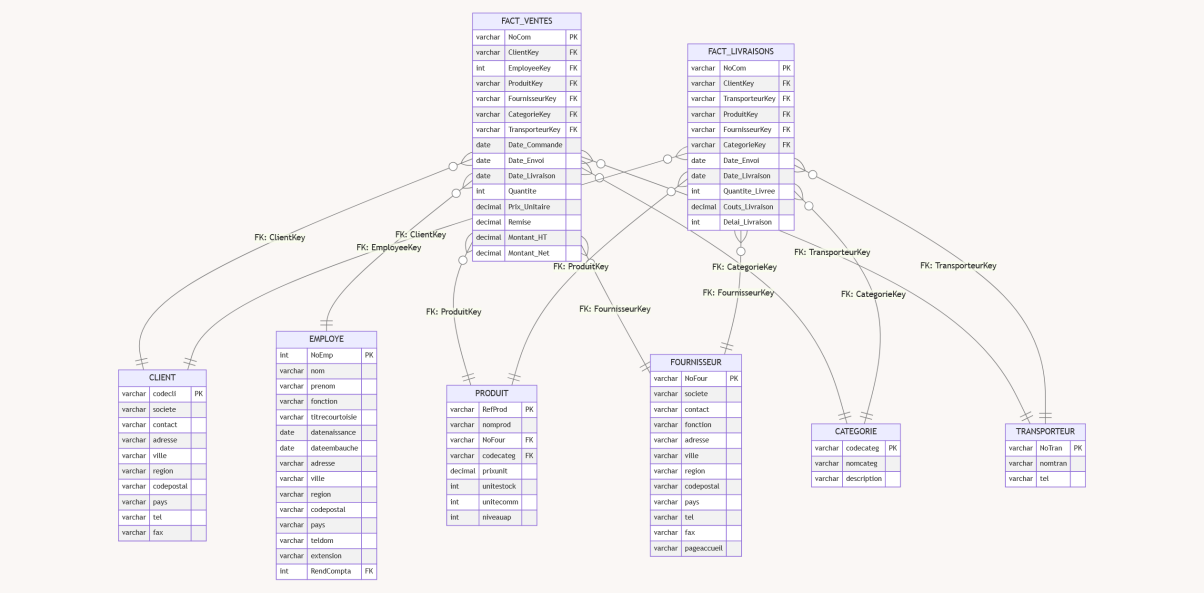


Schéma en étoile du Datawarehouse.

7 Scripts SQL

Cette section détaille les scripts SQL nécessaires à la création des tables, des index, et des vues pour le Datawarehouse. Ces scripts assurent une implémentation robuste et optimisée des modèles conceptuels de données décrits précédemment.

7.1 Création des Tables

Les scripts ci-dessous créent les tables de base nécessaires à la gestion des entités principales :

```
-- Suppression des tables si elles existent
DROP TABLE IF EXISTS DETAIL_COMMANDE;
DROP TABLE IF EXISTS COMMANDE;
DROP TABLE IF EXISTS PRODUIT;
DROP TABLE IF EXISTS TRANSPORTEUR;
DROP TABLE IF EXISTS EMPLOYE;
DROP TABLE IF EXISTS CLIENT;
DROP TABLE IF EXISTS FOURNISSEUR;
DROP TABLE IF EXISTS CATEGORIE;

-- Création de la table CATEGORIE
CREATE TABLE CATEGORIE (
    codecateg VARCHAR(50) PRIMARY KEY,
    nomcateg VARCHAR(100) NOT NULL,
    description VARCHAR(255)
);

-- Création de la table FOURNISSEUR
CREATE TABLE FOURNISSEUR (
```

```
    NoFour VARCHAR(50) PRIMARY KEY,
    societe VARCHAR(100) NOT NULL,
    contact VARCHAR(100),
    fonction VARCHAR(100),
    adresse VARCHAR(200),
    ville VARCHAR(100),
    region VARCHAR(100),
    codepostal VARCHAR(20),
    pays VARCHAR(100),
    tel VARCHAR(20),
    fax VARCHAR(20),
    pageaccueil VARCHAR(200)
);

-- Création de la table CLIENT
CREATE TABLE CLIENT (
    codecli VARCHAR(50) PRIMARY KEY,
    societe VARCHAR(100) NOT NULL,
    contact VARCHAR(100),
    fonction VARCHAR(100),
    adresse VARCHAR(200),
    ville VARCHAR(100),
    region VARCHAR(100),
    codepostal VARCHAR(20),
    pays VARCHAR(100),
    tel VARCHAR(20),
    fax VARCHAR(20)
);

-- Création de la table EMPLOYE
CREATE TABLE EMPLOYE (
    NoEmp SERIAL PRIMARY KEY,
    nom VARCHAR(100) NOT NULL,
    prenom VARCHAR(100) NOT NULL,
    fonction VARCHAR(100),
    titrecourtoisie VARCHAR(50),
    datenaissance DATE,
    dateembauche DATE,
    adresse VARCHAR(200),
    ville VARCHAR(100),
    region VARCHAR(100),
    codepostal VARCHAR(20),
    pays VARCHAR(100),
    teldom VARCHAR(20),
    extension VARCHAR(20),
    RendCompteA INTEGER,
    FOREIGN KEY (RendCompteA) REFERENCES EMPLOYE(NoEmp) ON DELETE SET NULL
);
```

```
-- Création de la table TRANSPORTEUR
CREATE TABLE TRANSPORTEUR (
    NoTran VARCHAR(50) PRIMARY KEY,
    nomtran VARCHAR(100) NOT NULL,
    tel VARCHAR(20)
);

-- Création de la table PRODUIT
CREATE TABLE PRODUIT (
    RefProd VARCHAR(50) PRIMARY KEY,
    nomprod VARCHAR(100) NOT NULL,
    NoFour VARCHAR(50),
    codecateg VARCHAR(50),
    qteparunit VARCHAR(100),
    prixunit DECIMAL(10,2),
    unitesstock INTEGER,
    unitescom INTEGER,
    niveaureap INTEGER,
    FOREIGN KEY (NoFour) REFERENCES FOURNISSEUR(NoFour) ON DELETE SET NULL,
    FOREIGN KEY (codecateg) REFERENCES CATEGORIE(codecateg) ON DELETE SET NULL
);

-- Création de la table COMMANDE
CREATE TABLE COMMANDE (
    NoCom VARCHAR(50) PRIMARY KEY,
    codecli VARCHAR(50),
    NoEmp INTEGER,
    datecom DATE,
    dateobjliv DATE,
    dateenv DATE,
    NoTran VARCHAR(50),
    port DECIMAL(10,2),
    destinataire VARCHAR(100),
    adrliv VARCHAR(200),
    villeliv VARCHAR(100),
    regionliv VARCHAR(100),
    codepostalliv VARCHAR(20),
    paysliv VARCHAR(100),
    FOREIGN KEY (codecli) REFERENCES CLIENT(codecli),
    FOREIGN KEY (NoEmp) REFERENCES EMPLOYE(NoEmp),
    FOREIGN KEY (NoTran) REFERENCES TRANSPORTEUR(NoTran)
);

-- Création de la table DETAIL_COMMANDE
CREATE TABLE DETAIL_COMMANDE (
    NoCom VARCHAR(50),
    NoLigCom INTEGER,
```

```
    RefProd VARCHAR(50),
    prixunit DECIMAL(10,2),
    qte INTEGER,
    remise DECIMAL(4,2),
    PRIMARY KEY (NoCom, NoLigCom),
    FOREIGN KEY (NoCom) REFERENCES COMMANDE(NoCom),
    FOREIGN KEY (RefProd) REFERENCES PRODUIT(RefProd)
);

-- Ajout d'index pour optimiser les performances
CREATE INDEX idx_commande_client ON COMMANDE(codecli);
CREATE INDEX idx_commande_employe ON COMMANDE(NoEmp);
CREATE INDEX idx_commande_transporteur ON COMMANDE(NoTran);
CREATE INDEX idx_produit_fournisseur ON PRODUIT(NoFour);
CREATE INDEX idx_produit_categorie ON PRODUIT(codecateg);
CREATE INDEX idx_detail_commande_produit ON DETAIL_COMMANDE(RefProd);
```

7.2 Création des Vues pour les Rapports

Les vues ci-dessous permettent d'extraire les données nécessaires pour les tableaux de bord et les rapports analytiques :

```
CREATE VIEW FACT_VENTES AS
SELECT
    dc.NoCom,
    cmd.codecli AS ClientKey,
    cmd.NoEmp AS EmployeKey,
    p.RefProd AS ProduitKey,
    p.NoFour AS FournisseurKey,
    p.codecateg AS CategorieKey,
    cmd.NoTran AS TransporteurKey,
    cmd.datecom AS Date_Commande,
    cmd.dateenvi AS Date_Envoi,
    cmd.datebliv AS Date_Livraison,
    dc.qte AS Quantite,
    dc.prixunit AS Prix_Unitaire,
    dc.remise AS Remise,
    (dc.qte * dc.prixunit) AS Montant_HT,
    (dc.qte * dc.prixunit) - dc.remise AS Montant_Net
FROM DETAIL_COMMANDE dc
JOIN COMMANDE cmd ON dc.NoCom = cmd.NoCom
JOIN PRODUIT p ON dc.RefProd = p.RefProd
JOIN CLIENT c ON cmd.codecli = c.codecli
JOIN FOURNISSEUR f ON p.NoFour = f.NoFour
JOIN CATEGORIE cat ON p.codecateg = cat.codecateg
LEFT JOIN EMPLOYE e ON cmd.NoEmp = e.NoEmp
LEFT JOIN TRANSPORTEUR t ON cmd.NoTran = t.NoTran;

CREATE VIEW FACT_LIVRAISONS AS
```

```
SELECT
    dc.NoCom,
    cmd.codecli AS ClientKey,
    cmd.NoTran AS TransporteurKey,
    p.RefProd AS ProduitKey,
    p.NoFour AS FournisseurKey,
    p.codecateg AS CategorieKey,
    cmd.dateenvi AS Date_Envoi,
    cmd.datebliv AS Date_Livraison,
    dc.qte AS Quantite_Livree,
    cmd.port AS Couts_Livraison,
    DATEDIFF(day, cmd.dateenvi, cmd.datebliv) AS Delai_Livraison
FROM DETAIL_COMMANDE dc
JOIN COMMANDE cmd ON dc.NoCom = cmd.NoCom
JOIN PRODUIT p ON dc.RefProd = p.RefProd
JOIN CLIENT c ON cmd.codecli = c.codecli
JOIN FOURNISSEUR f ON p.NoFour = f.NoFour
JOIN CATEGORIE cat ON p.codecateg = cat.codecateg
LEFT JOIN TRANSPORTEUR t ON cmd.NoTran = t.NoTran;
```

8 Prototypage de l'Interface Utilisateur

Pour garantir une expérience utilisateur (UX) intuitive et une interface utilisateur (UI) efficace, un prototypage fonctionnel a été réalisé. Ce prototypage a permis de valider les concepts visuels et fonctionnels avant de passer à la phase de développement, tout en restant adapté au contexte d'un projet géré par un alternant.

8.1 Objectifs du Prototypage

Le prototypage vise à atteindre les objectifs suivants :

- **Validation de l'ergonomie :** S'assurer que les utilisateurs finaux peuvent naviguer facilement dans l'application et accéder rapidement aux informations essentielles.
- **Test des interactions clés :** Simuler les principales actions utilisateur (navigation, filtrage des données, export des rapports) pour identifier les éventuelles améliorations à apporter.
- **Amélioration de la communication :** Fournir une base visuelle claire pour aligner les attentes des parties prenantes et des équipes techniques.

8.2 Structure du Prototype

Le prototype couvre les écrans essentiels à l'application, conçus pour répondre aux besoins fonctionnels identifiés :

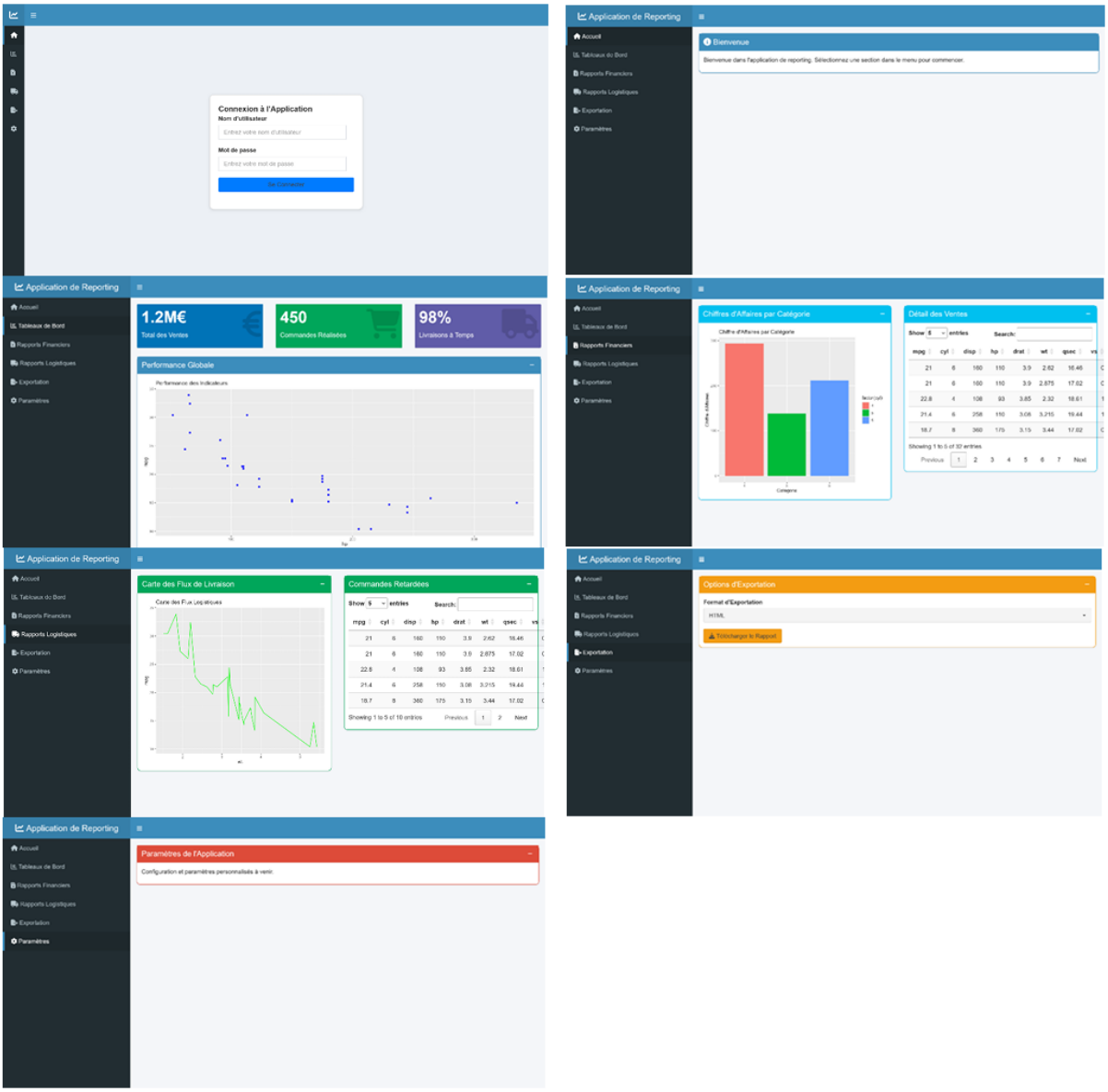
- **Page d'accueil :**
  - Tableau de bord affichant les indicateurs clés de performance (KPI) spécifiques à chaque département.
  - Navigation simplifiée vers les modules Financier et Logistique.

- **Rapports financiers :**
  - Graphiques interactifs pour visualiser les chiffres d'affaires, remises, et marges bénéficiaires.
  - Tableau détaillé des ventes par produit, catégorie, et région.
- **Rapports logistiques :**
  - Carte interactive montrant les flux de livraison par région.
  - Liste des commandes avec indicateurs de performance (retards, coûts logistiques).
- **Page d'exportation :**
  - Options pour exporter les rapports
  - Aperçu du fichier avant export.

8.3 Caractéristiques du Prototype

Le prototype inclut des fonctionnalités interactives pour simuler une expérience utilisateur réaliste :

- **Navigation simulée :** Les flux de navigation entre les écrans sont représentés via des liens simples ou des dessins explicatifs.
- **Éléments graphiques basiques :** Les graphiques et tableaux sont illustrés pour montrer les informations principales à afficher.
- **Compatibilité mobile :** Les maquettes incluent des vues adaptées aux écrans de tablettes pour les consultations en mobilité.



Prototype de l'UI

9 Plan de Tests

Un plan de tests rigoureux est essentiel pour garantir la fiabilité, la performance et la conformité de l'application avec les exigences spécifiées. Cette section détaille les étapes et types de tests qui seront réalisés pour valider l'application, tout en prenant en compte les ressources limitées du projet.

9.1 Objectifs des Tests

Les objectifs principaux des tests sont :

— Identifier et corriger les anomalies avant la mise en production.

— Valider la conformité des fonctionnalités aux spécifications définies.

— Assurer la robustesse, la performance et la sécurité de l'application.

— Garantir une expérience utilisateur intuitive et fluide.

9.2 Types de Tests

Différents types de tests seront effectués pour couvrir les aspects critiques de l'application.

9.2.1 Tests Unitaires

- **Objectif :** Vérifier que chaque composant de l'application (fonctions, modules) fonctionne correctement en isolation.
- **Exemples de cas de test :**
  - Calcul des indicateurs clés (chiffre d'affaires, remises) dans les modules financiers.
  - Génération des graphiques interactifs via RShiny.
  - Connexion et interrogation des bases de données PostgreSQL.
- **Outils :** Tests manuels ou automatisés via des scripts R pour valider les fonctions critiques.

9.2.2 Tests Intégrés

- **Objectif :** Vérifier que les différents modules de l'application interagissent correctement.
- **Exemples de cas de test :**
  - Intégration des filtres avec les tableaux de bord.
  - Génération d'un rapport consolidé basé sur les données PostgreSQL.
  - Exportation des rapports.
- **Outils :** Scénarios de tests manuels pour simuler des flux utilisateurs.

9.2.3 Tests Fonctionnels

- **Objectif :** Valider que les fonctionnalités principales répondent aux attentes des utilisateurs.
- **Exemples de cas de test :**
  - Affichage des KPI financiers (chiffre d'affaires, remises).
  - Navigation entre les tableaux de bord Financier et Logistique.
  - Filtrage des données par région, période ou catégorie.
- **Outils :** Tests fonctionnels guidés par les spécifications.

9.2.4 Tests de Performance

- **Objectif :** Mesurer la rapidité et l'efficacité de l'application sous différentes charges.
- **Exemples de cas de test :**
  - Temps de réponse pour des requêtes volumineuses.
  - Temps de chargement des tableaux de bord pour 5-10 utilisateurs simultanés.
- **Outils :** Simulations via des scripts R pour évaluer la latence des calculs et requêtes.

9.2.5 Tests de Sécurité

- **Objectif :** Identifier les vulnérabilités et garantir la confidentialité des données.
- **Exemples de cas de test :**
  - Validation des droits d'accès selon les rôles (par département).
  - Vérification des connexions sécurisées via HTTPS.
  - Tests d'intégrité des données sensibles chiffrées.
- **Outils :** OWASP ZAP ou audits manuels pour vérifier la sécurité basique.

9.2.6 Tests d'Utilisabilité

- **Objectif :** S'assurer que l'interface utilisateur est intuitive et répond aux attentes des utilisateurs finaux.
- **Exemples de cas de test :**
  - Facilité d'utilisation des graphiques interactifs et des filtres.
  - Navigation entre les modules et compréhension immédiate des informations affichées.
- **Outils :** Retours des utilisateurs testeurs basés sur des scénarios pratiques.

9.3 Planification des Tests

Les tests seront organisés en cinq phases principales se déroulant simultanément sur 1 semaine :

- **Phase 1 : Tests Unitaires** Valider les fonctions et composants critiques.
- **Phase 2 : Tests Intégrés** Vérifier l'interopérabilité entre les modules.
- **Phase 3 : Tests Fonctionnels** S'assurer que les fonctionnalités principales répondent aux spécifications.
- **Phase 4 : Tests de Performance et de Sécurité** Évaluer les temps de réponse et la sécurité.
- **Phase 5 : Tests d'Utilisabilité avec utilisateurs finaux** Valider l'expérience utilisateur.

9.4 Critères d'Acceptation

Le projet sera validé uniquement si les critères suivants sont remplis :

- Toutes les fonctionnalités principales sont opérationnelles, sans anomalies critiques.
- Les temps de réponse ne dépassent pas 7 secondes pour les tableaux de bord.
- Aucun problème de sécurité majeur n'est détecté.
- Les utilisateurs finaux valident l'intuitivité et la fluidité de la navigation.

9.5 Outils Utilisés

Les outils utilisés pour la réalisation des tests sont adaptés aux contraintes du projet :

— **Tests unitaires et intégrés :** Scripts R pour valider les calculs et les intégrations.

— **Tests de performance :** Simulations via des scripts manuels pour analyser les temps de réponse.

— **Tests de sécurité :** OWASP ZAP pour identifier les vulnérabilités basiques.

— **Tests d'utilisabilité :** Retours qualitatifs des utilisateurs finaux.

10 Maintenance et Évolution

Une fois l'application mise en production, un plan de maintenance et d'évolution est essentiel pour garantir la pérennité et l'efficacité du système à long terme. Cette partie sera gérée par un nouveau contrat ou par l'équipe technique de l'entreprise.

10.1 Objectifs de la Maintenance

Les objectifs principaux de la maintenance sont :

— **Assurer la disponibilité :** Garantir un fonctionnement optimal de l'application en production.

— **Corriger les anomalies :** Résoudre rapidement les éventuels bugs ou problèmes détectés.

— **Répondre aux évolutions :** Adapter l'application aux nouveaux besoins des utilisateurs et aux changements métiers.

— **Optimiser les performances :** Effectuer des ajustements pour garantir des temps de réponse rapides et une gestion efficace des ressources.

10.2 Types de Maintenance

10.2.1 Maintenance Corrective

— **Description :** Identification et résolution des anomalies signalées par les utilisateurs ou détectées via le monitoring.

— **Délai d'intervention :**

  — Anomalies critiques : résolution dans un délai maximum de 24 heures.

  — Anomalies mineures : correction planifiée lors des mises à jour périodiques.

10.2.2 Maintenance Préventive

— **Description :** Surveillance proactive pour minimiser les risques d'incidents futurs.

— **Actions prévues :**

  — Surveillance des performances système (temps de réponse, utilisation des ressources).

  — Analyse régulière des journaux pour détecter des anomalies potentielles.

  — Mise à jour périodique des bibliothèques et des dépendances.

10.2.3 Maintenance Évolutive

- **Description :** Ajout ou modification des fonctionnalités pour répondre aux besoins évolutifs des utilisateurs.
- **Exemples d'évolutions possibles :**
  - Ajout de nouveaux filtres ou indicateurs (KPI) dans les tableaux de bord.
  - Amélioration des visualisations pour une meilleure clarté et interactivité.
  - Extension de l'application pour inclure de nouvelles dimensions d'analyse ou de nouveaux utilisateurs.

10.3 Organisation de la Maintenance

10.3.1 Responsabilité de la Maintenance

La maintenance sera assurée par le développeur principal ou par une nouvelle équipe avec une organisation légère mais efficace :

- **Développeur/équipe :** Responsable des correctifs, des évolutions et de la surveillance.
- **Parties prenantes :** Les utilisateurs remontent les incidents et proposent des évolutions via un système de ticketing simple.

10.3.2 Processus de Maintenance

Le processus suivant sera appliqué pour chaque intervention :

- **Identification :** Les incidents sont signalés via un outil de ticketing (par exemple, GitHub Issues ou Trello).
- **Analyse :** Les incidents sont priorisés en fonction de leur impact sur l'application.
- **Intervention :** Les correctifs ou évolutions sont développés et testés dans un environnement de staging.
- **Validation :** Les changements sont validés avant déploiement en production.
- **Documentation :** Les modifications sont documentées pour assurer un suivi clair.

10.4 Plan d'Évolution à Long Terme

Pour garantir l'évolutivité de l'application, un plan sur 2 ans est proposé :

- **Phase 1 : Stabilisation (6 premiers mois).**
  - Correction des anomalies détectées après la mise en production.
  - Optimisation des performances et de l'expérience utilisateur.
  - Collecte des retours des utilisateurs pour prioriser les améliorations.
- **Phase 2 : Ajout de fonctionnalités (6 mois - 1 an).**
  - Développement de nouveaux tableaux de bord ou filtres avancés.
  - Intégration de fonctionnalités analytiques supplémentaires.
  - Amélioration des visualisations pour une interactivité accrue.
- **Phase 3 : Extension et innovation (1-2 ans).**
  - Adaptation pour gérer des volumes de données accrus.

— Intégration de nouveaux modules ou dimensions d'analyse.
— **Exploration des fonctionnalités prédictives :** Une piste d'amélioration future inclut l'intégration de modèles analytiques prédictifs pour anticiper des tendances clés (ventes, logistique) et renforcer les capacités décisionnelles. Ces analyses, basées sur les données historiques, pourraient transformer l'application en un véritable outil proactif.

10.5 Outils de Maintenance

Les outils suivants seront utilisés pour assurer une maintenance efficace :

— **Monitoring :** Grafana pour surveiller les performances système.
— **Ticketing :** Trello ou GitHub Issues pour le suivi des incidents et des tâches.
— **Contrôle de version :** Git pour gérer les modifications de code.
— **Staging :** Utilisation d'un environnement de test pour valider les modifications avant déploiement.

11 Coûts et Budget

Le budget global du projet a été réévalué afin de limiter les coûts, en prenant en compte les contraintes identifiées : un seul développeur polyvalent porte le projet, l'ensemble des outils utilisés sont open source, et l'hébergement se fera sur un serveur dédié déjà acquis par l'entreprise. Cette approche permet une gestion financière optimisée tout en garantissant la faisabilité technique du projet.

11.1 Résumé du Budget

Le tableau ci-dessous présente une répartition des coûts estimés par catégorie :

Catégorie	**Description**	**Coût estimé (€)**
Ressources humaines	Développeur en alternance (3 mois)	5,000
Outils et logiciels	Outils open source (RShiny, PostgreSQL, Grafana)	0
Infrastructure	Hébergement sur un serveur dédié OVH (déjà acquis)	0
Total		5,000

TABLE 1 – Répartition des coûts estimés selon une approche optimisée.

11.2 Détails des Coûts

11.2.1 Ressources Humaines

Le projet sera pris en charge par un développeur polyvalent en alternance, qui assurera l'ensemble des tâches : gestion de projet, développement back-end et front-end, ainsi que l'intégration. Le coût estimé repose sur un salaire équivalent à 78% du SMIC, conformément aux exigences légales, avec un léger ajustement pour prendre en compte la charge de travail.

— **Développeur unique (3 mois) :** Responsable de l'intégralité du cycle de développement, y compris l'analyse des besoins, le développement, et les tests.

— Le salaire estimé reflète la polyvalence du développeur et la charge de travail nécessaire à la réalisation du projet.

11.2.2 Outils et Logiciels

Tous les outils utilisés pour le développement et l'analyse sont des solutions open source, ce qui permet d'éliminer les coûts liés aux licences.

— **Back-end et Front-end :** RShiny pour le développement de l'application web, intégrant à la fois l'interface utilisateur et la logique serveur.

— **Base de données :** PostgreSQL, choisi pour sa robustesse et sa parfaite compatibilité avec R.

— **Monitoring et Sécurité :** Des outils comme Grafana pour la surveillance et OWASP ZAP pour les tests de sécurité.

— **Visualisations :** Les bibliothèques R et Shiny offrent des outils puissants pour créer des graphiques interactifs sans coût additionnel.

11.2.3 Infrastructure

Le projet sera hébergé sur un serveur dédié OVH déjà acquis par l'entreprise, ce qui annule les coûts d'hébergement et d'infrastructure.

— Utilisation du serveur existant pour déployer l'application RShiny et la base de données PostgreSQL.

— La maintenance de l'infrastructure sera assurée en interne par l'équipe technique actuelle.

11.3 Estimation des Risques Financiers

Bien que les risques financiers soient réduits grâce à cette approche optimisée, certains facteurs peuvent avoir un impact sur le coût final :

— **Risque de surcharge pour le développeur :** Des retards peuvent survenir en cas de tâches imprévues ou de complexité technique élevée.

— **Risque technique :** La dépendance à un seul développeur pourrait entraîner des retards si celui-ci devient indisponible ou rencontre des difficultés majeures.

— **Risque d'évolutivité :** Des besoins non anticipés pourraient nécessiter des ajustements importants après la mise en production.

11.4 Plan de Contingence

Afin de minimiser ces risques, les mesures suivantes seront mises en place :

— Des revues hebdomadaires pour suivre l'avancement du projet et ajuster les priorités si nécessaire.

— Une documentation complète pour faciliter la transition en cas de changement de développeur.

— Une phase de test approfondie pour limiter les efforts de maintenance post-déploiement.

11.5 Conclusion sur le Budget

Avec un budget total estimé à 5,000 €, ce projet repose sur une approche réaliste et optimisée, en s'appuyant sur des outils open source, un hébergement déjà disponible, et un seul développeur. Cette stratégie permet de maîtriser les coûts tout en répondant efficacement aux objectifs fonctionnels et techniques fixés.

12 Sécurité

La sécurité des données et des accès est une priorité absolue pour ce projet, compte tenu de la sensibilité des informations financières et logistiques traitées. Cette section décrit les mesures prévues pour protéger l'application et ses données contre les menaces internes et externes, tout en tenant compte des ressources limitées.

12.1 Objectifs de Sécurité

Les principaux objectifs de sécurité sont :

— **Confidentialité :** Garantir que seules les personnes autorisées ont accès aux données sensibles.

— **Intégrité :** Prévenir les modifications non autorisées ou accidentelles des données.

— **Disponibilité :** Assurer un accès continu à l'application, même en cas d'incidents techniques.

— **Traçabilité :** Enregistrer et analyser les actions pour détecter tout comportement suspect.

12.2 Mesures de Sécurité

12.2.1 Contrôle d'Accès

— **Authentification :** Mise en place d'une authentification avec des mots de passe forts. L'authentification multi-facteurs (MFA) pourra être ajoutée en option future.

— **Gestion des rôles :** Définition de rôles basés sur les départements (développeur, administration, Finance, Logistique) avec des accès limités aux données et fonctionnalités pertinentes.

— **Sessions sécurisées :** Implémentation de sessions avec expiration automatique après une période d'inactivité.

12.2.2 Audit et Monitoring

— **Journaux d'activité :** Enregistrement des connexions et des actions critiques dans des logs horodatés.

— **Surveillance continue :** Utilisation d'outils comme Grafana pour détecter des anomalies dans les performances ou les accès.

12.2.3 Formation et Sensibilisation

— **Utilisateurs finaux :** Sensibilisation aux bonnes pratiques (mots de passe forts, vigilance face au phishing).

— **Développeur :** Adoption des normes de développement sécurisé (OWASP).

12.3 Plan de Réponse aux Incidents

Un plan de réponse aux incidents sera mis en place pour réagir efficacement en cas de problème de sécurité :

— **Identification :** Détection des incidents via logs et outils de surveillance.

— **Isolation :** Contenir l'incident (par exemple, désactivation temporaire des comptes affectés).

— **Résolution :** Correction rapide des vulnérabilités identifiées.

— **Amélioration :** Mise à jour des pratiques et procédures pour prévenir des incidents similaires.

12.4 Conformité aux Réglementations

L'application respectera les normes et réglementations applicables :

— **RGPD :** Respect des droits des utilisateurs (accès, rectification, suppression) et transparence sur le traitement des données.

— **Bonnes pratiques :** Adoption des recommandations OWASP pour sécuriser le développement.

12.5 Validation de la Sécurité

Avant la mise en production, des tests de sécurité seront réalisés :

— **Tests de pénétration :** Simulations pour identifier les vulnérabilités critiques.

— **Vérification de conformité :** Validation que toutes les mesures de sécurité sont correctement implémentées.

12.6 Résumé des Mesures de Sécurité

Les mesures prévues sont adaptées au contexte d'une startup avec des ressources limitées, tout en garantissant un niveau élevé de protection pour les données et les utilisateurs de l'application. Elles incluent un chiffrement robuste, une gestion fine des accès, et une surveillance proactive pour prévenir et répondre aux menaces.

13 Déploiement et Hébergement

Le déploiement et l'hébergement de l'application sont adaptés aux ressources limitées du projet. L'approche retenue s'appuie sur un seul développeur polyvalent, l'utilisation d'un serveur dédié OVH déjà acquis par l'entreprise, et des outils open source pour garantir une infrastructure performante et économique. Cependant, dans un premier temps, l'outil sera hébergé localement.

13.1 Environnements de Déploiement

Deux environnements distincts seront mis en place pour assurer le développement et la production :

- **Environnement de développement :**
  - Utilisé par le développeur pour implémenter et tester les fonctionnalités.
  - En local pour ne pas utiliser de ressources non nécessaires.
- **Environnement de production :**
  - Hébergement de l'application en ligne pour les utilisateurs finaux.
  - Configuré sur le serveur dédié OVH existant pour optimiser les coûts.

13.2 Étapes du Déploiement

Le déploiement de l'application suivra un processus simplifié :

- **Préparation :**
  - Validation manuelle du code et exécution de tests unitaires.
  - Configuration du serveur OVH avec les dépendances nécessaires (R, RShiny, PostgreSQL).
- **Migration des données :**
  - Chargement des données initiales dans PostgreSQL à partir des fichiers ou sources existantes.
  - Validation de l'intégrité des données.
- **Mise en production :**
  - Déploiement de l'application RShiny sur le serveur OVH en utilisant shiny server.
  - Configuration des sauvegardes automatisées et des outils de monitoring.

13.3 Infrastructure d'Hébergement

L'infrastructure sera basée sur le serveur dédié OVH existant pour minimiser les coûts. Les principaux composants incluent :

- **Serveur dédié OVH :** Hébergement de l'application RShiny et de la base PostgreSQL sur une même machine, configurée pour supporter jusqu'à 20 utilisateurs simultanés.
- **Base de données :** PostgreSQL installé sur le serveur pour stocker et interroger les données de l'application.
- **Monitoring :** Installation d'outils open source tels que Grafana pour surveiller l'utilisation des ressources.
- **Sauvegardes locales et distantes :** Configuration de sauvegardes régulières des bases de données et des fichiers critiques.

13.4 Outils et Automatisation

Pour simplifier la gestion et le déploiement, les outils suivants seront utilisés :

— **CI/CD :** Automatisation minimale avec des scripts Bash pour déployer les mises à jour directement sur le serveur OVH.

— **Monitoring :** Grafana et Prometheus pour surveiller les performances et détecter les anomalies.

— **Alertes :** Configuration de notifications par email pour signaler les incidents critiques.

13.5 Sauvegardes et Récupération en Cas de Panne

Un plan de sauvegarde simple mais efficace sera mis en place pour garantir la continuité des services :

— **Fréquence des sauvegardes :** Quotidienne pour les bases de données et hebdomadaire pour les configurations.

— **Emplacement des sauvegardes :** Sauvegardes locales sur le serveur et copies déportées sur un stockage distant sécurisé.

— **Tests de restauration :** Simulations régulières pour vérifier la capacité à restaurer rapidement l'application et les données en cas de panne.

13.6 Scalabilité et Optimisation

L'infrastructure actuelle est dimensionnée pour un usage modéré. Toutefois, des mesures simples permettront de répondre à une augmentation de la charge :

— **Optimisation des requêtes SQL :** Validation et indexation des requêtes pour réduire les temps de réponse.

— **Mise en cache :** Utilisation de solutions comme RCache pour améliorer les performances des visualisations répétitives.

— **Migration future :** Possibilité de passer à une architecture cloud si l'utilisation dépasse les capacités du serveur dédié.

13.7 Plan de Suivi Post-Déploiement

Un suivi post-déploiement sera effectué sur une durée d'un mois :

— **Surveillance :** Monitoring continu pour détecter les éventuels problèmes de performances ou de sécurité.

— **Support utilisateur :** Résolution rapide des anomalies signalées par les utilisateurs.

— **Collecte de feedback :** Intégration des retours utilisateurs pour ajuster ou améliorer l'application.

13.8 Conclusion sur le Déploiement et l'Hébergement

Grâce à l'utilisation du serveur dédié OVH et à des outils open source, le déploiement de l'application sera économique tout en répondant aux exigences de performance et de sécurité. Cette approche garantit une mise en production rapide et un fonctionnement fiable, tout en restant évolutive pour répondre à d'éventuels besoins futurs.

14 Critères d'Acceptation

Les critères d'acceptation sont les conditions que l'application doit remplir pour être considérée comme achevée et prête à être livrée. Ces critères garantissent que toutes les fonctionnalités essentielles sont conformes aux spécifications, que les performances sont satisfaisantes, et que l'expérience utilisateur est intuitive.

14.1 Critères Fonctionnels

- **Accès aux tableaux de bord :** Les tableaux de bord financiers et logistiques doivent être accessibles et afficher correctement les données consolidées à partir de la base PostgreSQL.
- **Filtrage des données :** Les utilisateurs doivent pouvoir filtrer les données par :
  - Période (mois, trimestre, année).
  - Région géographique.
  - Catégorie de produit ou de transporteur.
- **Exportation des rapports :** Les rapports doivent être exportables au format HTML. Les formats PDF, CSV et XLSX sont optionnels et peuvent être ajoutés ultérieurement.
- **Visualisations interactives :** Les graphiques (histogrammes, cartes thermiques, etc.) doivent être dynamiques et permettre un survol interactif des données.
- **Authentification :** Un système simple d'authentification doit restreindre l'accès aux tableaux de bord aux utilisateurs autorisés.

14.2 Critères Techniques

- **Performance :** Le temps de chargement des tableaux de bord ne doit pas excéder 10 secondes pour les volumes de données actuels.
- **Sécurité :**
  - Toutes les connexions doivent être sécurisées via HTTPS.
  - Les mots de passe doivent être hashés dans la base de données.
- **Intégration :** L'application doit s'intégrer parfaitement avec PostgreSQL, garantissant une synchronisation des données.
- **Compatibilité :** L'application doit fonctionner sur les navigateurs modernes (Chrome, Firefox).

14.3 Critères d'Utilisabilité

— **Interface intuitive :** Les utilisateurs doivent pouvoir naviguer facilement dans les tableaux de bord et comprendre rapidement les informations affichées.

— **Clarté des visualisations :** Les graphiques et tableaux doivent être lisibles et permettre une interprétation immédiate des données.

— **Formation :** Les utilisateurs finaux doivent pouvoir utiliser l'application après une session de formation de 1 heure maximum, avec une documentation fournie (manuel ou vidéo).

14.4 Critères de Validation des Tests

— **Tests fonctionnels :** Tous les scénarios d'utilisation définis doivent être validés avec succès (aucune anomalie critique ou bloquante).

— **Tests de performance :** Les temps de réponse doivent respecter les seuils définis.

— **Tests de sécurité :** Les connexions non autorisées doivent être bloquées et enregistrées dans des journaux pour audit.

14.5 Critères de Satisfaction Client

Les parties prenantes doivent valider les livrables selon les critères suivants :

— **Conformité aux besoins :** Les fonctionnalités développées répondent aux cas d'usage définis dans le cahier des charges.

— **Adoption par les utilisateurs :** Les utilisateurs finaux doivent confirmer que l'application améliore leur efficacité.

14.6 Processus d'Acceptation

Le processus d'acceptation suivra les étapes suivantes :

— **Phase de validation technique :** Vérification des critères fonctionnels et techniques par le développeur et les parties prenantes.

— **Phase de validation utilisateur :** Tests avec un groupe d'utilisateurs finaux pour valider la simplicité d'utilisation.

— **Validation finale :** Réunion avec les parties prenantes pour signer l'accord de livraison.

Ces critères garantissent que l'application sera fonctionnelle, performante, et adaptée aux besoins essentiels des utilisateurs finaux, tout en respectant les contraintes de ressources. La validation de ces critères marquera la fin officielle du projet.

Conclusion et Récapitulatif

Ce cahier des charges pour le projet **Application de Reporting** vise à répondre aux besoins spécifiques de *GlobalTrade Inc.* en améliorant ses processus décisionnels par la centralisation, l'automatisation, et l'analyse avancée de données.

Résumé des Points Clés

- **Objectifs du projet :**
  - Centralisation des données provenant de sources diverses.
  - Création de tableaux de bord interactifs et automatisation des rapports.
  - Facilitation des prises de décision stratégiques et opérationnelles.
- **Périmètre :** Conception et déploiement d'une application décisionnelle utilisant RShiny et PostgreSQL, excluant la maintenance post-projet et les modifications des bases de données sources.
- **Exigences techniques :**
  - Performance optimale pour un volume modéré de données.
  - Sécurité renforcée avec chiffrement des données sensibles et gestion des accès.
  - Compatibilité avec les navigateurs modernes.
- **Calendrier :** Réalisation en trois phases principales sur 5 semaines, incluant le développement, les tests, et le déploiement.
- **Coûts :** Budget optimisé à 5,000 €, grâce à l'utilisation d'outils open source et de ressources internes.

Appel à l'Action et Prochaines Étapes

Nous invitons les parties prenantes à valider ce document pour lancer la mise en œuvre du projet. Les prochaines étapes incluent :

1. Validation finale du cahier des charges.
2. Début des phases de développement selon le calendrier défini.
3. Points de suivi hebdomadaires pour garantir l'alignement avec les objectifs.

Coordonnées des Responsables

Pour toute question ou clarification concernant ce document, veuillez contacter :

- **Responsable du projet :** Marie Challet (`marie.challet@globaltrade.com`).

Nous sommes convaincus que ce projet constituera un levier stratégique pour *GlobalTrade Inc.*, permettant d'améliorer ses performances et de renforcer sa compétitivité.

Le 9 janvier 2025
L'équipe projet