Système de Facturation Informatisé

pour Station-Service

Analyse Fonctionnelle et Conception UML

Documentation Technique et Fonctionnelle

16 décembre 2025

Table des matières

Chapitre 1

Introduction

1.1 Contexte du projet

Le présent document décrit l'analyse, la conception et l'architecture d'un système de facturation informatisé destiné à une station-service. Ce système a pour objectif de moderniser et d'automatiser l'ensemble des processus de gestion commerciale, depuis la gestion des clients jusqu'au suivi des paiements.

1.1.1 Problématique

Les stations-service modernes offrent une diversité de services qui va bien au-delà de la simple distribution de carburant. Elles proposent des services d'entretien automobile, de lavage, de vidange, ainsi que la vente de produits annexes. La gestion manuelle de ces activités entraîne plusieurs problèmes :

- Risques d'erreurs dans la facturation
- Difficultés de traçabilité des transactions
- Temps de traitement important des devis et factures
- Gestion complexe des paiements partiels
- Absence de vision claire sur l'état des créances
- Difficultés dans le suivi des statuts des documents

1.1.2 Objectifs du système

Le système de facturation doit permettre de :

1. Gérer efficacement les informations clients et leur historique
2. Créer et gérer des devis avec transformation en factures
3. Automatiser le calcul des montants (HT, TVA, TTC)
4. Assurer la numérotation automatique des documents
5. Gérer les paiements (partiels ou complets)
6. Suivre les statuts des documents (devis, factures)
7. Garantir l'intégrité et la cohérence des données
8. Générer des rapports et statistiques

1.2 Périmètre du système

1.2.1 Inclus dans le périmètre

— Gestion des clients (création, modification, consultation)
— Catalogue des services et tarification
— Création et gestion des devis
— Transformation devis vers facture
— Facturation (création, modification, consultation)
— Gestion des paiements multiples par facture
— Suivi des statuts (devis : en attente/accepté/refusé, factures : impayée/partiellement payée/payée)
— Calculs automatiques (montants HT, TVA, TTC, soldes)
— Consultation de l'historique client

1.2.2 Exclus du périmètre

— Gestion des stocks de carburant et produits
— Gestion du personnel et planning
— Comptabilité générale avancée
— Gestion de la pompe à essence (automatisation)
— Système de caisse enregistreuse intégré

1.3 Méthodologie

Le projet suit une approche structurée en plusieurs phases :

1. **Analyse fonctionnelle** : Identification des acteurs, des besoins et des règles de gestion
2. **Modélisation UML** : Création des diagrammes de cas d'utilisation, de classes, de séquence et d'activités
3. **Conception de la base de données** : Modélisation relationnelle sous PostgreSQL avec respect des formes normales
4. **Implémentation** : Développement du système et de la base de données
5. **Tests et validation** : Vérification du bon fonctionnement selon les spécifications

1.4 Outils et technologies

1.4.1 Outils de modélisation

— **StarUML** ou **PlantUML** : Pour la création des diagrammes UML
— **Draw.io** : Pour les schémas d'architecture
— **pgModeler** ou **DBeaver** : Pour la modélisation de la base de données PostgreSQL

1.4.2 Technologies de développement

— **PostgreSQL 15+** : Système de gestion de base de données relationnelle
— **Python 3.10+** avec frameworks web (Flask/Django) ou Java Spring Boot
— **HTML5/CSS3/JavaScript** pour l'interface utilisateur
— **Git** pour la gestion de versions

1.4.3 Environnement de développement

— **Visual Studio Code** ou **PyCharm / IntelliJ IDEA**
— **pgAdmin 4** : Administration de PostgreSQL
— **Postman** : Tests des API
— **Docker** : Conteneurisation (optionnel)

Chapitre 2

Analyse Fonctionnelle

2.1 Identification des acteurs

2.1.1 Acteurs principaux

Le Gérant

Rôle : Responsable principal de la station-service
Responsabilités :

- Gestion complète du système
- Supervision de toutes les opérations
- Validation des devis importants
- Consultation des rapports et statistiques
- Configuration des services et tarifs
- Gestion des utilisateurs du système

Droits d'accès : Accès complet à toutes les fonctionnalités

L'Employé/Caissier

Rôle : Personnel de la station chargé des opérations quotidiennes
Responsabilités :

- Accueil et enregistrement des clients
- Création de devis pour les services demandés
- Création et édition des factures
- Enregistrement des paiements
- Consultation des informations clients
- Impression des documents

Droits d'accès : Accès limité aux fonctionnalités opérationnelles courantes

Le Client

Rôle : Bénéficiaire des services de la station
Interactions avec le système :

— Fourniture d'informations personnelles (nom, contact, véhicule)
— Demande de services
— Réception de devis
— Acceptation ou refus de devis
— Réception de factures
— Effectuation de paiements
— Consultation de son historique (via interface optionnelle)

Droits d'accès : Consultation uniquement de ses propres données (si portail client implémenté)

2.1.2 Acteurs secondaires

Le Système de Numérotation

Rôle : Composant système automatique
Fonction :

— Génération automatique des numéros de devis (format : DEV-YYYY-NNNN)
— Génération automatique des numéros de factures (format : FAC-YYYY-NNNN)
— Garantie de l'unicité des numéros
— Séquençage chronologique

Le Système de Calcul

Rôle : Composant système automatique
Fonction :

— Calcul des montants HT (somme des lignes)
— Application des taux de TVA
— Calcul des montants TTC
— Calcul des soldes restants dus
— Mise à jour automatique des statuts de paiement

2.2 Cas d'utilisation du système

2.2.1 Diagramme de cas d'utilisation global

Le système comporte les cas d'utilisation suivants organisés par domaine fonctionnel :

Gestion des clients

- **CU01 - Créer un client** : Enregistrement d'un nouveau client dans le système
- **CU02 - Modifier un client** : Mise à jour des informations d'un client existant
- **CU03 - Consulter un client** : Affichage des détails et de l'historique d'un client
- **CU04 - Rechercher un client** : Recherche par nom, téléphone ou véhicule

Gestion des services

- **CU05 - Créer un service** : Ajout d'un nouveau service au catalogue
- **CU06 - Modifier un service** : Mise à jour d'un service (tarif, description)
- **CU07 - Consulter les services** : Affichage du catalogue des services
- **CU08 - Désactiver un service** : Marquage d'un service comme non disponible

Gestion des devis

- **CU09 - Créer un devis** : Création d'un nouveau devis pour un client
- **CU10 - Modifier un devis** : Modification d'un devis en statut "en attente"
- **CU11 - Consulter un devis** : Affichage des détails d'un devis
- **CU12 - Valider un devis** : Changement du statut en "accepté"
- **CU13 - Refuser un devis** : Changement du statut en "refusé"
- **CU14 - Transformer devis en facture** : Création automatique d'une facture depuis un devis accepté
- **CU15 - Imprimer un devis** : Génération d'un document PDF

Gestion des factures

- **CU16 - Créer une facture** : Création d'une nouvelle facture
- **CU17 - Modifier une facture** : Modification d'une facture non payée
- **CU18 - Consulter une facture** : Affichage des détails d'une facture
- **CU19 - Annuler une facture** : Annulation d'une facture avec création d'un avoir
- **CU20 - Imprimer une facture** : Génération d'un document PDF

Gestion des paiements

- **CU21 - Enregistrer un paiement** : Ajout d'un paiement pour une facture
- **CU22 - Consulter les paiements** : Affichage de l'historique des paiements d'une facture
- **CU23 - Annuler un paiement** : Annulation d'un paiement erroné

Consultation et rapports

- **CU24 - Consulter l'historique client** : Vue complète des devis, factures et paiements d'un client
- **CU25 - Générer rapport des ventes** : Statistiques sur une période donnée
- **CU26 - Consulter les impayés** : Liste des factures non soldées
- **CU27 - Générer rapport TVA** : Récapitulatif pour déclaration fiscale

2.2.2 Description détaillée des cas d'utilisation principaux

CU09 - Créer un devis

Cas d'utilisation	Créer un devis
Acteur principal	Employé, Gérant
Acteurs secondaires	Système de numérotation, Système de calcul
Préconditions	— L'utilisateur est authentifié — Le client existe dans le système — Au moins un service est disponible
Scénario nominal	1. L'employé sélectionne le client 2. Le système affiche les informations du client 3. L'employé ajoute des services au devis (service, quantité) 4. Le système calcule automatiquement les montants HT, TVA et TTC 5. L'employé saisit les remarques éventuelles 6. L'employé valide la création 7. Le système génère un numéro de devis unique 8. Le système enregistre le devis avec le statut "en attente" 9. Le système affiche le devis créé
Scénarios alternatifs	**3a.** Le client demande un service non disponible — L'employé informe le client — Retour à l'étape 3 ou fin du cas **6a.** Erreur de validation — Le système affiche les erreurs — Retour à l'étape concernée
Postconditions	— Un nouveau devis est créé avec statut "en attente" — Le devis a un numéro unique — Les montants sont calculés et cohérents

CU14 - Transformer devis en facture

Cas d'utilisation	Transformer devis en facture
Acteur principal	Employé, Gérant
Acteurs secondaires	Système de numérotation, Système de calcul
Préconditions	— L'utilisateur est authentifié — Le devis existe et a le statut "accepté" — Le devis n'a pas déjà été transformé en facture
Scénario nominal	1. L'employé recherche et sélectionne le devis accepté 2. Le système affiche les détails du devis 3. L'employé demande la transformation en facture 4. Le système vérifie que le devis est transformable 5. Le système génère un numéro de facture unique 6. Le système crée une nouvelle facture avec : — Les mêmes lignes de services — Les mêmes montants — Une référence au devis d'origine — Le statut "impayée" — La date du jour 7. Le système met à jour le devis (marqué comme transformé) 8. Le système affiche la facture créée
Scénarios alternatifs	**4a.** Le devis n'est pas dans un état transformable — Le système affiche un message d'erreur — Fin du cas **4b.** Le devis a déjà été transformé — Le système affiche la facture existante — Fin du cas
Postconditions	— Une nouvelle facture est créée — La facture reprend toutes les informations du devis — Le lien devis-facture est établi — La facture a le statut "impayée"

CU21 - Enregistrer un paiement

Cas d'utilisation	Enregistrer un paiement
Acteur principal	Employé, Gérant
Acteurs secondaires	Système de calcul
Préconditions	— L'utilisateur est authentifié — La facture existe et n'est pas entièrement payée
Scénario nominal	1. L'employé recherche et sélectionne la facture 2. Le système affiche les détails de la facture et le solde restant 3. L'employé saisit : — Le montant du paiement — Le mode de paiement (espèces, carte, chèque, virement) — La date du paiement — Une référence (optionnel, ex : numéro chèque) 4. Le système vérifie que le montant ne dépasse pas le solde 5. L'employé valide l'enregistrement 6. Le système enregistre le paiement 7. Le système recalcule le solde restant 8. Le système met à jour le statut de la facture : — "partiellement payée" si solde ¿ 0 — "payée" si solde = 0 9. Le système affiche la confirmation
Scénarios alternatifs	**4a.** Le montant dépasse le solde restant — Le système affiche un avertissement — L'employé peut ajuster ou annuler — Retour à l'étape 3 **4b.** Montant négatif ou nul — Le système affiche une erreur — Retour à l'étape 3
Postconditions	— Un nouveau paiement est enregistré — Le solde de la facture est mis à jour — Le statut de la facture est actualisé — L'historique des paiements est mis à jour

2.3 Règles de gestion

2.3.1 Règles générales

1. **RG01 - Unicité des numéros** : Chaque devis et chaque facture doit avoir un numéro unique dans le système.
2. **RG02 - Format de numérotation** :
   - Devis : DEV-YYYY-NNNN (ex : DEV-2024-0001)
   - Factures : FAC-YYYY-NNNN (ex : FAC-2024-0001)
3. **RG03 - Chronologie** : Les numéros doivent être attribués de manière strictement chronologique.
4. **RG04 - Intégrité référentielle** : Toute suppression doit respecter les dépendances entre entités.

2.3.2 Règles sur les clients

5. **RG05 - Information minimale** : Un client doit avoir au minimum un nom et un moyen de contact.
6. **RG06 - Historique conservé** : L'historique d'un client (devis, factures) doit être conservé même après suppression logique du client.
7. **RG07 - Unicité du contact** : Un numéro de téléphone ne peut être associé qu'à un seul client actif.

2.3.3 Règles sur les services

8. **RG08 - Tarif positif** : Le prix d'un service doit être strictement positif.
9. **RG09 - Taux de TVA valide** : Le taux de TVA doit être parmi les taux légaux (0%, 5.5%, 10%, 20%).
10. **RG10 - Service actif** : Seuls les services actifs peuvent être ajoutés à un nouveau devis ou facture.

2.3.4 Règles sur les devis

11. **RG11 - Statuts possibles** : Un devis peut avoir les statuts : "en attente", "accepté", "refusé".
12. **RG12 - Modification limitée** : Un devis ne peut être modifié que s'il est "en attente".
13. **RG13 - Transformation** : Seul un devis "accepté" peut être transformé en facture.
14. **RG14 - Unicité de transformation** : Un devis ne peut être transformé qu'une seule fois en facture.
15. **RG15 - Lignes obligatoires** : Un devis doit contenir au moins une ligne de service.
16. **RG16 - Quantité positive** : La quantité d'un service dans une ligne doit être strictement positive.

2.3.5 Règles sur les factures

17. **RG17 - Statuts possibles** : Une facture peut avoir les statuts : "impayée", "partiellement payée", "payée", "annulée".
18. **RG18 - Modification limitée** : Une facture ne peut être modifiée que si elle est "impayée" et n'a aucun paiement.
19. **RG19 - Annulation avec avoir** : L'annulation d'une facture payée ou partiellement payée nécessite la création d'un avoir.
20. **RG20 - Lignes obligatoires** : Une facture doit contenir au moins une ligne de service.
21. **RG21 - Référence devis optionnelle** : Une facture peut être créée directement ou depuis un devis.

2.3.6 Règles sur les paiements

22. **RG22 - Montant positif** : Un paiement doit avoir un montant strictement positif.
23. **RG23 - Contrôle du solde** : La somme des paiements ne peut pas dépasser le montant TTC de la facture.
24. **RG24 - Mode de paiement obligatoire** : Chaque paiement doit avoir un mode de paiement défini.
25. **RG25 - Date cohérente** : La date d'un paiement ne peut pas être antérieure à la date de la facture.
26. **RG26 - Mise à jour automatique du statut** : Le statut de la facture doit être automatiquement mis à jour après chaque paiement.

2.3.7 Règles de calcul

27. **RG27 - Calcul ligne HT** : Montant HT ligne = Prix unitaire × Quantité
28. **RG28 - Calcul total HT** : Total HT = Somme des montants HT de toutes les lignes
29. **RG29 - Calcul TVA** : Montant TVA = Total HT × (Taux TVA / 100)
30. **RG30 - Calcul TTC** : Montant TTC = Total HT + Montant TVA
31. **RG31 - Calcul solde** : Solde restant = Montant TTC - Somme des paiements
32. **RG32 - Arrondi** : Tous les montants sont arrondis à 2 décimales.

Chapitre 3

Modélisation UML

3.1 Diagramme de classes

Le diagramme de classes représente la structure statique du système et les relations entre les différentes entités.

3.1.1 Classes principales

Classe Client

Client
- id_client : INTEGER (PK) - nom : VARCHAR(100) - prenom : VARCHAR(100) - telephone : VARCHAR(20) - email : VARCHAR(100) - adresse : TEXT - ville : VARCHAR(50) - code_postal : VARCHAR(10) - immatriculation_vehicule : VARCHAR(20) - modele_vehicule : VARCHAR(100) - date_creation : TIMESTAMP - actif : BOOLEAN
+ creerClient() : Client + modifierClient() : void + consulterHistorique() : List¡Document¿ + desactiverClient() : void

Classe Service

Service
- id_service : INTEGER (PK) - code : VARCHAR(20) - designation : VARCHAR(200) - description : TEXT - prix_unitaire_ht : DECIMAL(10,2) - taux_tva : DECIMAL(5,2) - unite : VARCHAR(20) - categorie : VARCHAR(50) - actif : BOOLEAN
+ creerService() : Service + modifierService() : void + calculerPrixTTC() : DECIMAL + desactiverService() : void

Classe Devis

Devis
- id_devis : INTEGER (PK) - numero : VARCHAR(20) (UNIQUE) - date_emission : DATE - date_validite : DATE - id_client : INTEGER (FK) - montant_ht : DECIMAL(10,2) - montant_tva : DECIMAL(10,2) - montant_ttc : DECIMAL(10,2) - statut : VARCHAR(20) - remarques : TEXT - id_utilisateur : INTEGER (FK)
+ creerDevis() : Devis + modifierDevis() : void + validerDevis() : void + refuserDevis() : void + transformerEnFacture() : Facture + calculerMontants() : void + genererPDF() : File

Classe LigneDevis

LigneDevis
- id_ligne_devis : INTEGER (PK) - id_devis : INTEGER (FK) - id_service : INTEGER (FK) - quantite : DECIMAL(10,2) - prix_unitaire_ht : DECIMAL(10,2) - taux_tva : DECIMAL(5,2) - montant_ht : DECIMAL(10,2) - montant_tva : DECIMAL(10,2) - montant_ttc : DECIMAL(10,2)
+ ajouterLigne() : LigneDevis + modifierLigne() : void + supprimerLigne() : void + calculerMontants() : void

Classe Facture

Facture
- id_facture : INTEGER (PK) - numero : VARCHAR(20) (UNIQUE) - date_emission : DATE - date_echeance : DATE - id_client : INTEGER (FK) - id_devis : INTEGER (FK, nullable) - montant_ht : DECIMAL(10,2) - montant_tva : DECIMAL(10,2) - montant_ttc : DECIMAL(10,2) - montant_paye : DECIMAL(10,2) - solde_restant : DECIMAL(10,2) - statut : VARCHAR(20) - remarques : TEXT - id_utilisateur : INTEGER (FK)
+ creerFacture() : Facture + modifierFacture() : void + annulerFacture() : void + calculerMontants() : void + mettreAJourStatut() : void + genererPDF() : File

Classe LigneFacture

LigneFacture
- id_ligne_facture : INTEGER (PK) - id_facture : INTEGER (FK) - id_service : INTEGER (FK) - quantite : DECIMAL(10,2) - prix_unitaire_ht : DECIMAL(10,2) - taux_tva : DECIMAL(5,2) - montant_ht : DECIMAL(10,2) - montant_tva : DECIMAL(10,2) - montant_ttc : DECIMAL(10,2)
+ ajouterLigne() : LigneFacture + modifierLigne() : void + supprimerLigne() : void + calculerMontants() : void

Classe Paiement

Paiement
- id_paiement : INTEGER (PK) - id_facture : INTEGER (FK) - date_paiement : DATE - montant : DECIMAL(10,2) - mode_paiement : VARCHAR(20) - reference : VARCHAR(50) - remarques : TEXT - id_utilisateur : INTEGER (FK)
+ enregistrerPaiement() : Paiement + annulerPaiement() : void + consulterPaiement() : void

Classe Utilisateur

Utilisateur
- id_utilisateur : INTEGER (PK) - nom : VARCHAR(100) - prenom : VARCHAR(100) - email : VARCHAR(100) (UNIQUE) - mot_de_passe_hash : VARCHAR(255) - role : VARCHAR(20) - actif : BOOLEAN - date_creation : TIMESTAMP
+ seConnecter() : Session + seDeconnecter() : void + changerMotDePasse() : void + verifierPermission() : boolean

3.1.2 Relations entre classes

1. **Client - Devis** : Association 1..* (Un client peut avoir plusieurs devis)
2. **Client - Facture** : Association 1..* (Un client peut avoir plusieurs factures)
3. **Devis - LigneDevis** : Composition 1..* (Un devis contient au moins une ligne)
4. **Service - LigneDevis** : Association 1..* (Un service peut apparaître dans plusieurs lignes)
5. **Devis - Facture** : Association 0..1 - 0..1 (Un devis peut être transformé en une facture)
6. **Facture - LigneFacture** : Composition 1..* (Une facture contient au moins une ligne)
7. **Service - LigneFacture** : Association 1..* (Un service peut apparaître dans plusieurs lignes)
8. **Facture - Paiement** : Composition 1..* (Une facture peut avoir plusieurs paiements)
9. **Utilisateur - Devis** : Association 1..* (Un utilisateur crée plusieurs devis)
10. **Utilisateur - Facture** : Association 1..* (Un utilisateur crée plusieurs factures)
11. **Utilisateur - Paiement** : Association 1..* (Un utilisateur enregistre plusieurs paiements)

3.2 Diagrammes de séquence

3.2.1 Séquence : Création d'un devis

Acteurs : Employé, Système
Description : Ce diagramme illustre les interactions lors de la création d'un devis.
Flux :

1. L'employé demande la création d'un nouveau devis
2. Le système affiche le formulaire de sélection client
3. L'employé sélectionne le client
4. Le système affiche les informations du client
5. L'employé ajoute une ligne de service (service, quantité)
6. Le système calcule les montants de la ligne (HT, TVA, TTC)
7. L'employé peut ajouter d'autres lignes (retour à 5)
8. L'employé valide le devis
9. Le système génère le numéro de devis
10. Le système calcule les totaux (HT, TVA, TTC)
11. Le système enregistre le devis en base de données
12. Le système affiche le récapitulatif du devis créé

3.2.2 Séquence : Transformation devis en facture

Acteurs : Employé, Système

Description : Ce diagramme montre le processus de transformation d'un devis accepté en facture.

Flux :

1. L'employé recherche un devis
2. Le système affiche la liste des devis
3. L'employé sélectionne un devis accepté
4. Le système affiche les détails du devis
5. L'employé demande la transformation en facture
6. Le système vérifie le statut du devis (doit être "accepté")
7. Le système vérifie qu'aucune facture n'existe déjà pour ce devis
8. Le système génère un numéro de facture
9. Le système crée la facture avec :
   - Les informations du client
   - La référence au devis
   - Les mêmes lignes de services
   - Les mêmes montants
   - Le statut "impayée"
10. Le système copie toutes les lignes du devis vers la facture
11. Le système marque le devis comme transformé
12. Le système enregistre la facture
13. Le système affiche la nouvelle facture

3.2.3 Séquence : Enregistrement d'un paiement

Acteurs : Employé, Système, Client

Description : Ce diagramme détaille l'enregistrement d'un paiement pour une facture.

Flux :

1. Le client effectue un paiement
2. L'employé recherche la facture
3. Le système affiche les détails de la facture et le solde restant
4. L'employé saisit les informations du paiement (montant, mode, date, référence)
5. Le système vérifie que le montant est positif
6. Le système vérifie que le montant ne dépasse pas le solde
7. L'employé valide l'enregistrement
8. Le système enregistre le paiement
9. Le système recalcule le montant payé total
10. Le système recalcule le solde restant
11. Le système met à jour le statut de la facture :
    - "partiellement payée" si solde ¿ 0
    - "payée" si solde = 0
12. Le système enregistre les modifications
13. Le système affiche la confirmation avec le nouveau solde

3.3 Diagrammes d'activités

3.3.1 Activité : Processus complet de devis à paiement

Description : Ce diagramme montre le flux complet depuis la demande de devis jusqu'au paiement final.

Étapes :

1. **Début** : Client demande un service
2. **Créer le devis**
    - Sélectionner le client
    - Ajouter les services
    - Calculer les montants
    - Générer le numéro
    - Enregistrer (statut : "en attente")
3. **Décision client**
    - Si accepté → continuer à l'étape 4
    - Si refusé → marquer statut "refusé" → Fin
    - Si en attente → attendre décision

4. **Valider le devis**
   — Changer statut à "accepté"
5. **Transformer en facture**
   — Générer numéro facture
   — Copier les lignes
   — Créer lien devis-facture
   — Statut facture : "impayée"
6. **Service rendu**
   — Exécution des prestations
7. **Paiement(s)**
   — Enregistrer paiement
   — Recalculer solde
   — Mettre à jour statut
8. **Vérification solde**
   — Si solde ¿ 0 → statut "partiellement payée" → retour à étape 6
   — Si solde = 0 → statut "payée" → Fin

3.3.2 Activité : Gestion d'un paiement

Description : Détail du processus d'enregistrement d'un paiement avec validations.
Flux :

1. **Début** : Sélectionner la facture
2. **Afficher informations facture**
   — Montant TTC
   — Montant déjà payé
   — Solde restant
3. **Saisir informations paiement**
   — Montant
   — Mode de paiement
   — Date
   — Référence (optionnel)
4. **Validation montant**
   — Montant ¿ 0 ? Non → Erreur → retour étape 3
   — Montant ¡= solde ? Non → Avertissement → Choix : Ajuster ou Annuler
   — Si valide → continuer
5. **Enregistrer paiement**
   — Insertion en base
   — Transaction BEGIN
6. **Recalculer totaux**

   - Total payé = Somme(paiements)
   - Solde = TTC - Total payé
7. **Mettre à jour statut facture**
   - Si solde = 0 → "payée"
   - Si solde ¿ 0 → "partiellement payée"
8. **Validation transaction**
   - COMMIT
9. **Confirmation**
   - Afficher message de succès
   - Afficher nouveau solde
10. **Fin**

3.4 Diagramme d'états-transitions

3.4.1 États d'un Devis

États possibles :

- **En attente** : État initial après création
- **Accepté** : Le client a validé le devis
- **Refusé** : Le client a refusé le devis
- **Transformé** : Le devis a été converti en facture (sous-état d'Accepté)

Transitions :

1. Création → **En attente**
2. En attente → **Accepté** [validation client]
3. En attente → **Refusé** [refus client]
4. Accepté → **Transformé** [transformation en facture]
5. En attente → **En attente** [modification]

États terminaux : Refusé, Transformé

3.4.2 États d'une Facture

États possibles :

- **Impayée** : État initial, aucun paiement enregistré
- **Partiellement payée** : Au moins un paiement, mais solde ¿ 0
- **Payée** : Solde = 0, facture totalement réglée
- **Annulée** : Facture annulée (avec avoir)

Transitions :

1. Création → **Impayée**

2. Impayée → **Partiellement payée** [enregistrement paiement partiel]
3. Impayée → **Payée** [enregistrement paiement total]
4. Partiellement payée → **Partiellement payée** [autre paiement partiel]
5. Partiellement payée → **Payée** [paiement du solde]
6. Impayée → **Annulée** [annulation]
7. Partiellement payée → **Annulée** [annulation avec avoir]
8. Payée → **Annulée** [annulation avec avoir]

États terminaux : Payée, Annulée

Chapitre 4

Conception de la Base de Données

4.1 Modèle Conceptuel de Données (MCD)

Le MCD représente les entités du système et leurs associations sans considérations techniques.

4.1.1 Entités principales

1. **CLIENT** (id_client, nom, prenom, telephone, email, adresse, ville, code_postal, immatriculation_vehicule, modele_vehicule, date_creation, actif)
2. **SERVICE** (id_service, code, designation, description, prix_unitaire_ht, taux_tva, unite, categorie, actif)
3. **DEVIS** (id_devis, numero, date_emission, date_validite, montant_ht, montant_tva, montant_ttc, statut, remarques)
4. **FACTURE** (id_facture, numero, date_emission, date_echeance, montant_ht, montant_tva, montant_ttc, montant_paye, solde_restant, statut, remarques)
5. **PAIEMENT** (id_paiement, date_paiement, montant, mode_paiement, reference, remarques)
6. **UTILISATEUR** (id_utilisateur, nom, prenom, email, mot_de_passe_hash, role, actif, date_creation)

4.1.2 Associations

1. **POSSEDE** : CLIENT (1,n) (1,1) DEVIS
2. **POSSEDE** : CLIENT (1,n) (1,1) FACTURE
3. **CONTIENT** : DEVIS (1,n) (1,n) SERVICE [avec attributs : quantite, prix_unitaire_ht, taux_tva, montant_ht, montant_tva, montant_ttc]
4. **CONTIENT** : FACTURE (1,n) (1,n) SERVICE [avec attributs : quantite, prix_unitaire_ht, taux_tva, montant_ht, montant_tva, montant_ttc]
5. **GENERE** : DEVIS (0,1) (0,1) FACTURE
6. **REGLEE_PAR** : FACTURE (1,1) (0,n) PAIEMENT
7. **CREE** : UTILISATEUR (1,n) (1,1) DEVIS

8. **CREE** : UTILISATEUR (1,n) (1,1) FACTURE
9. **ENREGISTRE** : UTILISATEUR (1,n) (1,1) PAIEMENT

4.2 Modèle Logique de Données (MLD)

Le MLD traduit le MCD en tables relationnelles avec clés primaires et étrangères.

4.2.1 Tables

CLIENT(id_client, nom, prenom, telephone, email, adresse, ville, code_postal, immatriculation_vehicule, modele_vehicule, date_creation, actif)

SERVICE(id_service, code, designation, description, prix_unitaire_ht, taux_tva, unite, categorie, actif)

UTILISATEUR(id_utilisateur, nom, prenom, email, mot_de_passe_hash, role, actif, date_creation)

DEVIS(id_devis, numero, date_emission, date_validite, montant_ht, montant_tva, montant_ttc, statut, remarques, #id_client, #id_utilisateur)

LIGNE_DEVIS(id_ligne_devis, quantite, prix_unitaire_ht, taux_tva, montant_ht, montant_tva, montant_ttc, #id_devis, #id_service)

FACTURE(id_facture, numero, date_emission, date_echeance, montant_ht, montant_tva, montant_ttc, montant_paye, solde_restant, statut, remarques, #id_client, #id_devis, #id_utilisateur)

LIGNE_FACTURE(id_ligne_facture, quantite, prix_unitaire_ht, taux_tva, montant_ht, montant_tva, montant_ttc, #id_facture, #id_service)

PAIEMENT(id_paiement, date_paiement, montant, mode_paiement, reference, remarques, #id_facture, #id_utilisateur)

4.3 Modèle Physique de Données (MPD) - PostgreSQL

4.3.1 Script de création complet

Listing 4.1 – Création de la base de données

```
-- ================================================
-- SYST ME DE FACTURATION STATION-SERVICE
-- Script de cr ation PostgreSQL
-- ================================================

-- Suppression des tables existantes (d veloppement uniquement)
DROP TABLE IF EXISTS paiement CASCADE;
DROP TABLE IF EXISTS ligne_facture CASCADE;
DROP TABLE IF EXISTS facture CASCADE;
DROP TABLE IF EXISTS ligne_devis CASCADE;
DROP TABLE IF EXISTS devis CASCADE;
DROP TABLE IF EXISTS service CASCADE;
DROP TABLE IF EXISTS client CASCADE;
DROP TABLE IF EXISTS utilisateur CASCADE;
```

```
15
16  -- ================================================
17  -- TABLE: UTILISATEUR
18  -- ================================================
19  CREATE TABLE utilisateur (
20      id_utilisateur SERIAL PRIMARY KEY,
21      nom VARCHAR(100) NOT NULL,
22      prenom VARCHAR(100) NOT NULL,
23      email VARCHAR(100) NOT NULL UNIQUE,
24      mot_de_passe_hash VARCHAR(255) NOT NULL,
25      role VARCHAR(20) NOT NULL CHECK (role IN ('GERANT', 'EMPLOYE'
            )),
26      actif BOOLEAN NOT NULL DEFAULT TRUE,
27      date_creation TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP
28  );
29
30  -- Index pour recherche par email
31  CREATE INDEX idx_utilisateur_email ON utilisateur(email);
32
33  -- ================================================
34  -- TABLE: CLIENT
35  -- ================================================
36  CREATE TABLE client (
37      id_client SERIAL PRIMARY KEY,
38      nom VARCHAR(100) NOT NULL,
39      prenom VARCHAR(100),
40      telephone VARCHAR(20) NOT NULL,
41      email VARCHAR(100),
42      adresse TEXT,
43      ville VARCHAR(50),
44      code_postal VARCHAR(10),
45      immatriculation_vehicule VARCHAR(20),
46      modele_vehicule VARCHAR(100),
47      date_creation TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
48      actif BOOLEAN NOT NULL DEFAULT TRUE,
49
50      -- Contraintes
51      CONSTRAINT chk_client_telephone CHECK (LENGTH(TRIM(telephone)
            ) >= 10)
52  );
53
54  -- Index pour recherches fr quentes
55  CREATE INDEX idx_client_nom ON client(nom);
56  CREATE INDEX idx_client_telephone ON client(telephone);
57  CREATE INDEX idx_client_actif ON client(actif);
58
59  -- ================================================
60  -- TABLE: SERVICE
61  -- ================================================
```

```sql
CREATE TABLE service (
    id_service SERIAL PRIMARY KEY,
    code VARCHAR(20) NOT NULL UNIQUE,
    designation VARCHAR(200) NOT NULL,
    description TEXT,
    prix_unitaire_ht DECIMAL(10,2) NOT NULL,
    taux_tva DECIMAL(5,2) NOT NULL,
    unite VARCHAR(20) NOT NULL DEFAULT 'unit ',
    categorie VARCHAR(50) NOT NULL,
    actif BOOLEAN NOT NULL DEFAULT TRUE,

    -- Contraintes
    CONSTRAINT chk_service_prix_positif CHECK (prix_unitaire_ht > 0),
    CONSTRAINT chk_service_tva_valide CHECK (taux_tva IN (0, 5.5, 10, 20))
);

-- Index pour recherches
CREATE INDEX idx_service_code ON service(code);
CREATE INDEX idx_service_categorie ON service(categorie);
CREATE INDEX idx_service_actif ON service(actif);

-- ================================================
-- TABLE: DEVIS
-- ================================================
CREATE TABLE devis (
    id_devis SERIAL PRIMARY KEY,
    numero VARCHAR(20) NOT NULL UNIQUE,
    date_emission DATE NOT NULL DEFAULT CURRENT_DATE,
    date_validite DATE NOT NULL,
    id_client INTEGER NOT NULL,
    montant_ht DECIMAL(10,2) NOT NULL DEFAULT 0,
    montant_tva DECIMAL(10,2) NOT NULL DEFAULT 0,
    montant_ttc DECIMAL(10,2) NOT NULL DEFAULT 0,
    statut VARCHAR(20) NOT NULL DEFAULT 'EN_ATTENTE',
    remarques TEXT,
    id_utilisateur INTEGER NOT NULL,
    date_creation TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,

    -- Cl s    trangres
    CONSTRAINT fk_devis_client FOREIGN KEY (id_client)
        REFERENCES client(id_client) ON DELETE RESTRICT,
    CONSTRAINT fk_devis_utilisateur FOREIGN KEY (id_utilisateur)
        REFERENCES utilisateur(id_utilisateur) ON DELETE RESTRICT
        ,

    -- Contraintes
```

```
    CONSTRAINT chk_devis_statut CHECK (statut IN ('EN_ATTENTE', '
       ACCEPTE', 'REFUSE')),
    CONSTRAINT chk_devis_dates CHECK (date_validite >=
       date_emission),
    CONSTRAINT chk_devis_montants_positifs CHECK (
        montant_ht >= 0 AND montant_tva >= 0 AND montant_ttc >= 0
    )
);

-- Index pour recherches et performances
CREATE INDEX idx_devis_numero ON devis(numero);
CREATE INDEX idx_devis_client ON devis(id_client);
CREATE INDEX idx_devis_statut ON devis(statut);
CREATE INDEX idx_devis_date ON devis(date_emission DESC);

-- ================================================
-- TABLE: LIGNE_DEVIS
-- ================================================
CREATE TABLE ligne_devis (
    id_ligne_devis SERIAL PRIMARY KEY,
    id_devis INTEGER NOT NULL,
    id_service INTEGER NOT NULL,
    quantite DECIMAL(10,2) NOT NULL,
    prix_unitaire_ht DECIMAL(10,2) NOT NULL,
    taux_tva DECIMAL(5,2) NOT NULL,
    montant_ht DECIMAL(10,2) NOT NULL,
    montant_tva DECIMAL(10,2) NOT NULL,
    montant_ttc DECIMAL(10,2) NOT NULL,

    -- Cl  s    trangres
    CONSTRAINT fk_ligne_devis_devis FOREIGN KEY (id_devis)
        REFERENCES devis(id_devis) ON DELETE CASCADE,
    CONSTRAINT fk_ligne_devis_service FOREIGN KEY (id_service)
        REFERENCES service(id_service) ON DELETE RESTRICT,

    -- Contraintes
    CONSTRAINT chk_ligne_devis_quantite CHECK (quantite > 0),
    CONSTRAINT chk_ligne_devis_prix CHECK (prix_unitaire_ht >= 0)
);

-- Index pour performances
CREATE INDEX idx_ligne_devis_devis ON ligne_devis(id_devis);
CREATE INDEX idx_ligne_devis_service ON ligne_devis(id_service);

-- ================================================
-- TABLE: FACTURE
-- ================================================
CREATE TABLE facture (
    id_facture SERIAL PRIMARY KEY,
```

```
154     numero VARCHAR(20) NOT NULL UNIQUE,
155     date_emission DATE NOT NULL DEFAULT CURRENT_DATE,
156     date_echeance DATE NOT NULL,
157     id_client INTEGER NOT NULL,
158     id_devis INTEGER,
159     montant_ht DECIMAL(10,2) NOT NULL DEFAULT 0,
160     montant_tva DECIMAL(10,2) NOT NULL DEFAULT 0,
161     montant_ttc DECIMAL(10,2) NOT NULL DEFAULT 0,
162     montant_paye DECIMAL(10,2) NOT NULL DEFAULT 0,
163     solde_restant DECIMAL(10,2) NOT NULL DEFAULT 0,
164     statut VARCHAR(20) NOT NULL DEFAULT 'IMPAYEE',
165     remarques TEXT,
166     id_utilisateur INTEGER NOT NULL,
167     date_creation TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
168
169     -- Cl  s     trangres
170     CONSTRAINT fk_facture_client FOREIGN KEY (id_client)
171         REFERENCES client(id_client) ON DELETE RESTRICT,
172     CONSTRAINT fk_facture_devis FOREIGN KEY (id_devis)
173         REFERENCES devis(id_devis) ON DELETE RESTRICT,
174     CONSTRAINT fk_facture_utilisateur FOREIGN KEY (id_utilisateur
        )
175         REFERENCES utilisateur(id_utilisateur) ON DELETE RESTRICT
            ,
176
177     -- Contraintes
178     CONSTRAINT chk_facture_statut CHECK (statut IN (
179         'IMPAYEE', 'PARTIELLEMENT_PAYEE', 'PAYEE', 'ANNULEE'
180     )),
181     CONSTRAINT chk_facture_dates CHECK (date_echeance >=
        date_emission),
182     CONSTRAINT chk_facture_montants CHECK (
183         montant_ht >= 0 AND
184         montant_tva >= 0 AND
185         montant_ttc >= 0 AND
186         montant_paye >= 0 AND
187         montant_paye <= montant_ttc AND
188         solde_restant >= 0 AND
189         solde_restant = montant_ttc - montant_paye
190     )
191 );
192
193 -- Index pour recherches et performances
194 CREATE INDEX idx_facture_numero ON facture(numero);
195 CREATE INDEX idx_facture_client ON facture(id_client);
196 CREATE INDEX idx_facture_statut ON facture(statut);
197 CREATE INDEX idx_facture_date ON facture(date_emission DESC);
198 CREATE INDEX idx_facture_devis ON facture(id_devis);
199
```

```
200 -- ================================================
201 -- TABLE: LIGNE_FACTURE
202 -- ================================================
203 CREATE TABLE ligne_facture (
204     id_ligne_facture SERIAL PRIMARY KEY,
205     id_facture INTEGER NOT NULL,
206     id_service INTEGER NOT NULL,
207     quantite DECIMAL(10,2) NOT NULL,
208     prix_unitaire_ht DECIMAL(10,2) NOT NULL,
209     taux_tva DECIMAL(5,2) NOT NULL,
210     montant_ht DECIMAL(10,2) NOT NULL,
211     montant_tva DECIMAL(10,2) NOT NULL,
212     montant_ttc DECIMAL(10,2) NOT NULL,
213 
214     -- Cl s    trangres
215     CONSTRAINT fk_ligne_facture_facture FOREIGN KEY (id_facture)
216         REFERENCES facture(id_facture) ON DELETE CASCADE,
217     CONSTRAINT fk_ligne_facture_service FOREIGN KEY (id_service)
218         REFERENCES service(id_service) ON DELETE RESTRICT,
219 
220     -- Contraintes
221     CONSTRAINT chk_ligne_facture_quantite CHECK (quantite > 0),
222     CONSTRAINT chk_ligne_facture_prix CHECK (prix_unitaire_ht >=
        0)
223 );
224 
225 -- Index pour performances
226 CREATE INDEX idx_ligne_facture_facture ON ligne_facture(
    id_facture);
227 CREATE INDEX idx_ligne_facture_service ON ligne_facture(
    id_service);
228 
229 -- ================================================
230 -- TABLE: PAIEMENT
231 -- ================================================
232 CREATE TABLE paiement (
233     id_paiement SERIAL PRIMARY KEY,
234     id_facture INTEGER NOT NULL,
235     date_paiement DATE NOT NULL DEFAULT CURRENT_DATE,
236     montant DECIMAL(10,2) NOT NULL,
237     mode_paiement VARCHAR(20) NOT NULL,
238     reference VARCHAR(50),
239     remarques TEXT,
240     id_utilisateur INTEGER NOT NULL,
241     date_creation TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
242 
243     -- Cl s    trangres
244     CONSTRAINT fk_paiement_facture FOREIGN KEY (id_facture)
245         REFERENCES facture(id_facture) ON DELETE RESTRICT,
```

```
246     CONSTRAINT fk_paiement_utilisateur FOREIGN KEY (
            id_utilisateur)
247         REFERENCES utilisateur(id_utilisateur) ON DELETE RESTRICT
                ,
248
249     -- Contraintes
250     CONSTRAINT chk_paiement_montant CHECK (montant > 0),
251     CONSTRAINT chk_paiement_mode CHECK (mode_paiement IN (
252         'ESPECES', 'CARTE', 'CHEQUE', 'VIREMENT', 'AUTRE'
253     ))
254 );
255
256 -- Index pour recherches
257 CREATE INDEX idx_paiement_facture ON paiement(id_facture);
258 CREATE INDEX idx_paiement_date ON paiement(date_paiement DESC);
259 CREATE INDEX idx_paiement_mode ON paiement(mode_paiement);
260
261 -- ================================================
262 -- VUES UTILES
263 -- ================================================
264
265 -- Vue: Historique complet client
266 CREATE OR REPLACE VIEW v_historique_client AS
267 SELECT
268     c.id_client,
269     c.nom || ' ' || COALESCE(c.prenom, '') AS nom_complet,
270     'DEVIS' AS type_document,
271     d.numero AS numero_document,
272     d.date_emission,
273     d.montant_ttc,
274     d.statut,
275     NULL AS solde_restant
276 FROM client c
277 JOIN devis d ON c.id_client = d.id_client
278 UNION ALL
279 SELECT
280     c.id_client,
281     c.nom || ' ' || COALESCE(c.prenom, '') AS nom_complet,
282     'FACTURE' AS type_document,
283     f.numero AS numero_document,
284     f.date_emission,
285     f.montant_ttc,
286     f.statut,
287     f.solde_restant
288 FROM client c
289 JOIN facture f ON c.id_client = f.id_client
290 ORDER BY date_emission DESC;
291
292 -- Vue: Factures impay es ou partiellement pay es
```

```sql
CREATE OR REPLACE VIEW v_factures_impayees AS
SELECT
    f.id_facture,
    f.numero,
    f.date_emission,
    f.date_echeance,
    c.nom || ' ' || COALESCE(c.prenom, '') AS nom_client,
    c.telephone,
    f.montant_ttc,
    f.montant_paye,
    f.solde_restant,
    f.statut,
    CASE
        WHEN f.date_echeance < CURRENT_DATE THEN 'EN_RETARD'
        ELSE 'A_JOUR'
    END AS etat_echeance
FROM facture f
JOIN client c ON f.id_client = c.id_client
WHERE f.statut IN ('IMPAYEE', 'PARTIELLEMENT_PAYEE')
ORDER BY f.date_echeance ASC;

-- Vue: Statistiques des services
CREATE OR REPLACE VIEW v_statistiques_services AS
SELECT
    s.id_service,
    s.code,
    s.designation,
    s.categorie,
    COUNT(DISTINCT lf.id_facture) AS nb_factures,
    SUM(lf.quantite) AS quantite_totale,
    SUM(lf.montant_ttc) AS ca_total
FROM service s
LEFT JOIN ligne_facture lf ON s.id_service = lf.id_service
LEFT JOIN facture f ON lf.id_facture = f.id_facture
WHERE f.statut != 'ANNULEE' OR f.statut IS NULL
GROUP BY s.id_service, s.code, s.designation, s.categorie
ORDER BY ca_total DESC NULLS LAST;

-- ================================================
-- FONCTIONS ET TRIGGERS
-- ================================================

-- Fonction: G n ration automatique num ro devis
CREATE OR REPLACE FUNCTION generer_numero_devis()
RETURNS TRIGGER AS $
DECLARE
    annee INTEGER;
    prochain_numero INTEGER;
    nouveau_numero VARCHAR(20);
```

```
342 BEGIN
343     annee := EXTRACT(YEAR FROM CURRENT_DATE);
344
345     SELECT COALESCE(MAX(CAST(SUBSTRING(numero FROM 10) AS INTEGER
        )), 0) + 1
346     INTO prochain_numero
347     FROM devis
348     WHERE SUBSTRING(numero FROM 5 FOR 4) = annee::TEXT;
349
350     nouveau_numero := 'DEV-' || annee || '-' || LPAD(
        prochain_numero::TEXT, 4, '0');
351
352     NEW.numero := nouveau_numero;
353     RETURN NEW;
354 END;
355 $ LANGUAGE plpgsql;
356
357 CREATE TRIGGER trg_generer_numero_devis
358 BEFORE INSERT ON devis
359 FOR EACH ROW
360 WHEN (NEW.numero IS NULL OR NEW.numero = '')
361 EXECUTE FUNCTION generer_numero_devis();
362
363 -- Fonction: G n ration automatique num ro facture
364 CREATE OR REPLACE FUNCTION generer_numero_facture()
365 RETURNS TRIGGER AS $
366 DECLARE
367     annee INTEGER;
368     prochain_numero INTEGER;
369     nouveau_numero VARCHAR(20);
370 BEGIN
371     annee := EXTRACT(YEAR FROM CURRENT_DATE);
372
373     SELECT COALESCE(MAX(CAST(SUBSTRING(numero FROM 10) AS INTEGER
        )), 0) + 1
374     INTO prochain_numero
375     FROM facture
376     WHERE SUBSTRING(numero FROM 5 FOR 4) = annee::TEXT;
377
378     nouveau_numero := 'FAC-' || annee || '-' || LPAD(
        prochain_numero::TEXT, 4, '0');
379
380     NEW.numero := nouveau_numero;
381     RETURN NEW;
382 END;
383 $ LANGUAGE plpgsql;
384
385 CREATE TRIGGER trg_generer_numero_facture
386 BEFORE INSERT ON facture
```

```
387 FOR EACH ROW
388 WHEN (NEW.numero IS NULL OR NEW.numero = '')
389 EXECUTE FUNCTION generer_numero_facture();
390
391 -- Fonction: Calcul automatique montants ligne devis
392 CREATE OR REPLACE FUNCTION calculer_montants_ligne_devis()
393 RETURNS TRIGGER AS $
394 BEGIN
395     NEW.montant_ht := NEW.quantite * NEW.prix_unitaire_ht;
396     NEW.montant_tva := NEW.montant_ht * (NEW.taux_tva / 100);
397     NEW.montant_ttc := NEW.montant_ht + NEW.montant_tva;
398
399     -- Arrondi    2 d cimales
400     NEW.montant_ht := ROUND(NEW.montant_ht, 2);
401     NEW.montant_tva := ROUND(NEW.montant_tva, 2);
402     NEW.montant_ttc := ROUND(NEW.montant_ttc, 2);
403
404     RETURN NEW;
405 END;
406 $ LANGUAGE plpgsql;
407
408 CREATE TRIGGER trg_calculer_montants_ligne_devis
409 BEFORE INSERT OR UPDATE ON ligne_devis
410 FOR EACH ROW
411 EXECUTE FUNCTION calculer_montants_ligne_devis();
412
413 -- Fonction: Calcul automatique montants ligne facture
414 CREATE OR REPLACE FUNCTION calculer_montants_ligne_facture()
415 RETURNS TRIGGER AS $
416 BEGIN
417     NEW.montant_ht := NEW.quantite * NEW.prix_unitaire_ht;
418     NEW.montant_tva := NEW.montant_ht * (NEW.taux_tva / 100);
419     NEW.montant_ttc := NEW.montant_ht + NEW.montant_tva;
420
421     -- Arrondi    2 d cimales
422     NEW.montant_ht := ROUND(NEW.montant_ht, 2);
423     NEW.montant_tva := ROUND(NEW.montant_tva, 2);
424     NEW.montant_ttc := ROUND(NEW.montant_ttc, 2);
425
426     RETURN NEW;
427 END;
428 $ LANGUAGE plpgsql;
429
430 CREATE TRIGGER trg_calculer_montants_ligne_facture
431 BEFORE INSERT OR UPDATE ON ligne_facture
432 FOR EACH ROW
433 EXECUTE FUNCTION calculer_montants_ligne_facture();
434
435 -- Fonction: Mise    jour totaux devis
```

```
436 CREATE OR REPLACE FUNCTION maj_totaux_devis()
437 RETURNS TRIGGER AS $
438 DECLARE
439     v_id_devis INTEGER;
440 BEGIN
441     IF TG_OP = 'DELETE' THEN
442         v_id_devis := OLD.id_devis;
443     ELSE
444         v_id_devis := NEW.id_devis;
445     END IF;
446 
447     UPDATE devis
448     SET montant_ht = (
449             SELECT COALESCE(SUM(montant_ht), 0)
450             FROM ligne_devis
451             WHERE id_devis = v_id_devis
452         ),
453         montant_tva = (
454             SELECT COALESCE(SUM(montant_tva), 0)
455             FROM ligne_devis
456             WHERE id_devis = v_id_devis
457         ),
458         montant_ttc = (
459             SELECT COALESCE(SUM(montant_ttc), 0)
460             FROM ligne_devis
461             WHERE id_devis = v_id_devis
462         )
463     WHERE id_devis = v_id_devis;
464 
465     RETURN NULL;
466 END;
467 $ LANGUAGE plpgsql;
468 
469 CREATE TRIGGER trg_maj_totaux_devis
470 AFTER INSERT OR UPDATE OR DELETE ON ligne_devis
471 FOR EACH ROW
472 EXECUTE FUNCTION maj_totaux_devis();
473 
474 -- Fonction: Mise    jour totaux facture
475 CREATE OR REPLACE FUNCTION maj_totaux_facture()
476 RETURNS TRIGGER AS $
477 DECLARE
478     v_id_facture INTEGER;
479 BEGIN
480     IF TG_OP = 'DELETE' THEN
481         v_id_facture := OLD.id_facture;
482     ELSE
483         v_id_facture := NEW.id_facture;
484     END IF;
```

```
485
486     UPDATE facture
487     SET montant_ht = (
488             SELECT COALESCE(SUM(montant_ht), 0)
489             FROM ligne_facture
490             WHERE id_facture = v_id_facture
491         ),
492         montant_tva = (
493             SELECT COALESCE(SUM(montant_tva), 0)
494             FROM ligne_facture
495             WHERE id_facture = v_id_facture
496         ),
497         montant_ttc = (
498             SELECT COALESCE(SUM(montant_ttc), 0)
499             FROM ligne_facture
500             WHERE id_facture = v_id_facture
501         ),
502         solde_restant = (
503             SELECT COALESCE(SUM(montant_ttc), 0)
504             FROM ligne_facture
505             WHERE id_facture = v_id_facture
506         ) - (
507             SELECT COALESCE(SUM(montant), 0)
508             FROM paiement
509             WHERE id_facture = v_id_facture
510         )
511     WHERE id_facture = v_id_facture;
512
513     RETURN NULL;
514 END;
515 $ LANGUAGE plpgsql;
516
517 CREATE TRIGGER trg_maj_totaux_facture
518 AFTER INSERT OR UPDATE OR DELETE ON ligne_facture
519 FOR EACH ROW
520 EXECUTE FUNCTION maj_totaux_facture();
521
522 -- Fonction: Mise    jour statut facture apr s paiement
523 CREATE OR REPLACE FUNCTION maj_statut_facture_paiement()
524 RETURNS TRIGGER AS $
525 DECLARE
526     v_id_facture INTEGER;
527     v_montant_ttc DECIMAL(10,2);
528     v_montant_paye DECIMAL(10,2);
529     v_solde DECIMAL(10,2);
530 BEGIN
531     IF TG_OP = 'DELETE' THEN
532         v_id_facture := OLD.id_facture;
533     ELSE
```

```
534         v_id_facture := NEW.id_facture;
535     END IF;
536 
537     -- Calcul du montant pay  total
538     SELECT
539         f.montant_ttc,
540         COALESCE(SUM(p.montant), 0)
541     INTO v_montant_ttc, v_montant_paye
542     FROM facture f
543     LEFT JOIN paiement p ON f.id_facture = p.id_facture
544     WHERE f.id_facture = v_id_facture
545     GROUP BY f.montant_ttc;
546 
547     v_solde := v_montant_ttc - v_montant_paye;
548 
549     -- Mise    jour de la facture
550     UPDATE facture
551     SET montant_paye = v_montant_paye,
552         solde_restant = v_solde,
553         statut = CASE
554             WHEN v_solde = 0 THEN 'PAYEE'
555             WHEN v_solde > 0 AND v_montant_paye > 0 THEN '
                    PARTIELLEMENT_PAYEE'
556             ELSE 'IMPAYEE'
557         END
558     WHERE id_facture = v_id_facture;
559 
560     RETURN NULL;
561 END;
562 $ LANGUAGE plpgsql;
563 
564 CREATE TRIGGER trg_maj_statut_facture_paiement
565 AFTER INSERT OR UPDATE OR DELETE ON paiement
566 FOR EACH ROW
567 EXECUTE FUNCTION maj_statut_facture_paiement();
568 
569 -- ================================================
570 -- DONN ES DE TEST
571 -- ================================================
572 
573 -- Insertion utilisateurs
574 INSERT INTO utilisateur (nom, prenom, email, mot_de_passe_hash,
    role) VALUES
575 ('ADMIN', 'G rant', 'gerant@station.tg', '
    $2b$12$LQv3c1yqBWVHxkd0LHAkCOYz6TtxMQJqhN8/LewY5GyYIj.7', '
    GERANT'),
576 ('MENSAH', 'Kofi', 'kofi.mensah@station.tg', '
    $2b$12$LQv3c1yqBWVHxkd0LHAkCOYz6TtxMQJqhN8/LewY5GyYIj.7', '
    EMPLOYE');
```

```
577
578 -- Insertion clients
579 INSERT INTO client (nom, prenom, telephone, email, ville,
       immatriculation_vehicule, modele_vehicule) VALUES
580 ('AGBODJAN', 'Edem', '90123456', 'edem.agbodjan@email.tg', 'Lom
       ', 'TG-1234-AA', 'Toyota Corolla 2018'),
581 ('KOUASSI', 'Ama', '91234567', 'ama.kouassi@email.tg', 'Lom ', '
       TG-5678-BB', 'Honda Civic 2020'),
582 ('ATTIOGBE', 'Koffi', '92345678', NULL, 'Kara', 'TG-9012-CC', '
       Peugeot 208 2019');
583
584 -- Insertion services
585 INSERT INTO service (code, designation, prix_unitaire_ht,
       taux_tva, unite, categorie) VALUES
586 ('CARB-ESS', 'Essence Super', 650.00, 20.00, 'litre', 'Carburant'
       ),
587 ('CARB-GAZ', 'Gasoil', 580.00, 20.00, 'litre', 'Carburant'),
588 ('MAINT-VID', 'Vidange compl te', 25000.00, 20.00, 'unit ', '
       Entretien'),
589 ('MAINT-FIL', 'Changement filtres', 8000.00, 20.00, 'unit ', '
       Entretien'),
590 ('LAV-EXT', 'Lavage ext rieur', 3000.00, 20.00, 'unit ', '
       Lavage'),
591 ('LAV-COMP', 'Lavage complet + int rieur', 6000.00, 20.00, '
       unit ', 'Lavage'),
592 ('PNEU-MNT', 'Montage/D montage pneus', 4000.00, 20.00, 'pneu',
       'Pneumatique'),
593 ('DIAG-ELEC', 'Diagnostic  lectronique ', 15000.00, 20.00, '
       unit ', 'Diagnostic');
594
595 -- Message de confirmation
596 DO $
597 BEGIN
598     RAISE NOTICE 'Base de donn es cr  e avec succ s!';
599     RAISE NOTICE 'Utilisateurs: 2 (1 g rant , 1 employ )';
600     RAISE NOTICE 'Clients: 3';
601     RAISE NOTICE 'Services: 8';
602 END $;
```