

Spécialité Informatique
2^e année

ENSICAEN

Rapport de projet

Gestpharma

Logiciel de gestion pour pharmacie

BENNADY Achraf
JRAIF Tarik

Suivi :
Clouard Régis

Année 2004-2005

Table des matières

| | |
|---|----|
| 1. Introduction | 3 |
| 2. Client et cahier de charge | 4 |
| 2.1) Notre client | 4 |
| 2.2) Le cahier de charge..... | 4 |
| 2.2.1) Introduction | 4 |
| 2.2.2) L'interface du logiciel..... | 4 |
| 2.2.3) Le diagramme des cas d'utilisation | 5 |
| 2.2.4) Les fonctionnalités du logiciel | 5 |
| 3. Conception de la base de données | 8 |
| 3.1) Pourquoi une base de données ? | 8 |
| 3.2) Conception de la base de données..... | 8 |
| 3.2.1) Dictionnaire des données | 8 |
| 3.2.2) Graphe des dépendances fonctionnelles | 9 |
| 3.2.3) Modèle conceptuel final..... | 10 |
| 3.2.4) Représentation physique de la base de données..... | 10 |
| 3.2.4.1) Pourquoi Access ?..... | 10 |
| 3.2.4.2) Gestpharma et Access | 11 |
| 4. Développement du logiciel | 12 |
| 4.1) L'environnement de développement | 12 |
| 4.1.1) Borland C++ Builder | 12 |
| 4.1.2) Les composants utilisés | 12 |
| 4.2) Développement du logiciel | 13 |
| 4.2.1) Analyse et Modélisation | 13 |
| 4.2.2) Conception du logiciel..... | 15 |
| 4.2.2.1) La fenêtre principale | 15 |
| 4.2.2.2) Les pages du logiciel..... | 15 |
| 4.2.2.3) Fonctionnalités globales..... | 16 |
| 4.2.3) Installation du logiciel | 16 |
| 5. Conclusion | 17 |
| 6. Annexe | 18 |
| 6.1) Documentation du logiciel..... | 18 |
| 6.1.1) La page d'identification et les utilisateurs | 18 |
| 6.1.2) Les pages des fonctionnalités liées au stock | 18 |
| 6.1.3) Les pages des fonctionnalités liées à la vente | 20 |
| 6.1.4) Les pages des fonctionnalités liées aux commandes | 21 |
| 6.1.5) Les pages des fonctionnalités liées aux échanges | 22 |
| 6.1.6) Les pages des fonctionnalités liées aux crédits..... | 24 |
| 6.1.7) Les pages des fonctionnalités liées aux fournisseurs et bons de livraison..... | 25 |

1. Introduction

Ce travail s'intègre dans le cadre du programme de formation de la 2ème année de l'ENSICAEN filière Informatique.

Nous avons choisi pour notre projet de concevoir un logiciel de gestion pour pharmacie. Notre choix a été motivé par plusieurs points et notamment le fait d'avoir un client pour notre logiciel ce qui nous permet de nous initier à l'approche client développeur et au cycle de vie du logiciel, et nous engage à concevoir un produit fiable, robuste et répondant complètement au besoins du client. Notre choix a aussi été influé par les outils informatiques que nous allons mettre en œuvre pour ce logiciel à savoir la conception d'une base de données, d'une GUI (Graphical User Interface) et de toute la programmation qu'il y a derrière pour satisfaire le cahier de charge du client et aboutir à une application simple, utile, performante, ergonomique et fiable.

La conception et la mise en œuvre des bases de données constituent un volet très important de l'informatique car elles sont aujourd'hui au cœur des applications quotidiennes et du système d'information des entreprises. Les GUI constituent aussi une partie primordiale de l'informatique moderne car ils permettent la vulgarisation des applications pour le grand public et l'augmentation de l'interaction des utilisateurs avec le logiciel permettant ainsi une prise en main facile et une organisation visuelle efficace surtout quant il s'agit de la manipulation des bases de données.

C'est pour toutes ces raisons que nous avons choisi ce projet, et nous espérons que ce travail satisfera notre client et notre école et nous permettra d'enrichir notre savoir.

Pour ce projet nous avons choisit de suivre le cycle de développement logiciel complet pour nous initier aux méthodes de développement professionnelles et aboutir à un produit fini. Le travail que nous avons effectué se décompose en trois grandes partie : premièrement, l'analyse du cahier de charge et des besoins du client. Deuxièmement, la conception de la base de données du logiciel en utilisant le SGBD Ms Access. Et troisièmement, le conception de l'application avec sa GUI, ses fonctionnalités et ses tests d'intégration et de validation en utilisant le langage C++ et l'environnement de développement (IDE) Borland C++ Builder.

Dans ce rapport nous allons détailler toutes les étapes de conception de notre logiciel, et il se composera des trois parties citées dans le paragraphe précédent.

2. Client et cahier de charge

2.1) Notre client

Notre client est un pharmacien marocain dont la pharmacie, qui porte le nom de « Pharmacie Chorfa », se situe dans un village au sud de la capitale économique marocaine Casablanca. Sa pharmacie a ouvert en juin 2000, et près de 5 ans après, son activité a évolué et le pharmacien a ressenti le besoin d'informatiser la gestion de sa pharmacie. Les logiciels de gestion présents dans le marché marocain coûtent en moyenne 500 euros ce qui représente une somme importante par rapport au bénéfice de cette pharmacie. De plus les fonctionnalités qu'ils offrent ne correspondent pas au besoin de notre client car les logiciels disponibles regroupent la gestion du stock et la gestion comptable. Le pharmacien a trouvé que c'était trop compliqué à utiliser et avec trop de fonctionnalités inutiles. Il a donc voulu soumettre un cahier de charge pour l'application qu'il désire avoir à quelques entreprises de développement logiciel mais le prix a rapidement doublé et le pharmacien a préféré renoncer. Après avoir pris connaissance du cahier de charge et de la présence d'un projet dans le cadre de notre deuxième année à l'école nous lui avons proposé de concevoir ce logiciel bénévolement.

Voilà, ceci était l'histoire de notre rencontre avec notre premier client et nous espérons qu'il y en aura d'autres tout au long de notre carrière professionnelle.

2.2) Le cahier de charge

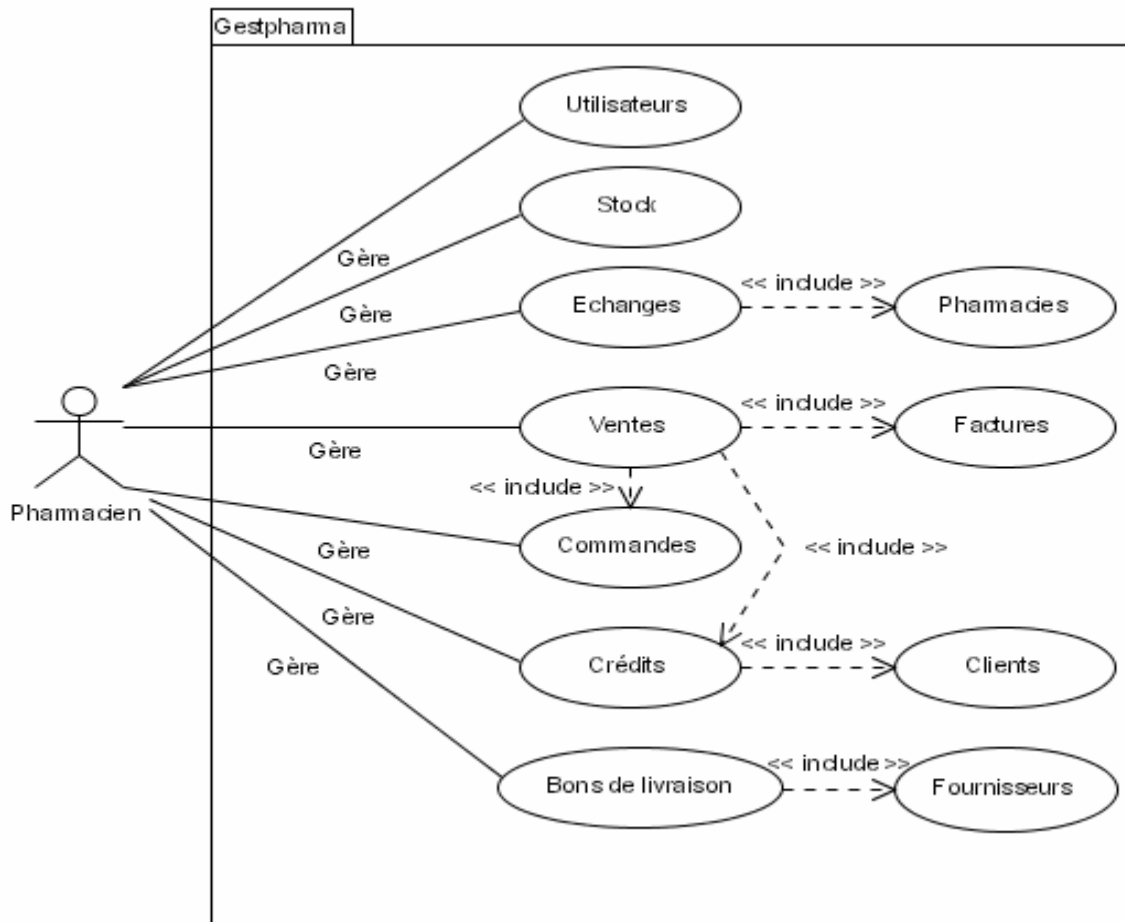
2.2.1) Introduction

Le cahier de charge au départ n'était pas bien mis en forme pour être directement exploitable vu que notre client, comme la plupart des clients, est un novice en informatique et n'arrivait pas à cerner précisément toutes les fonctionnalités qu'il voulait avoir dans son logiciel. Mais après plusieurs rencontres de quelques heures avec notre client pendant les vacances de la Toussaint nous avons pu discuter avec lui de toutes les fonctionnalités et les caractéristiques du logiciel. Cela nous a permis de comprendre précisément ce que le pharmacien attendait du logiciel et le fonctionnement de la pharmacie.

2.2.2) L'interface du logiciel

Le client désire une interface ergonomique, claire, sobre et surtout très facile à utiliser. Il veut une organisation des différentes fonctionnalités du logiciel sous forme d'une barre de menu déroulante. L'interface graphique doit être de type SDI (Single Document Interface c'est à dire une fenêtre principale et plusieurs pages à l'intérieur) car l'utilisateur n'aime pas les interfaces MDI (Multiple Document Interface) qu'il trouve confuse et difficile à manipuler. Le client veut une utilisation importante des tableaux pour organiser les données car il a l'habitude de les manipuler. Il veut aussi que chaque fonctionnalité accessible par le menu soit représentée par une page (frame) dans la fenêtre principale qui doit s'afficher plein écran. Il veut une couleur de fond simple et claire de préférence verte pour coïncider avec les couleurs de la pharmacie. Les pages doivent comporter des labels pour indiquer leur rôle.

2.2.3) Le diagramme des cas d'utilisation



2.2.4) Les fonctionnalités du logiciel

- **Les utilisateurs :** Le logiciel doit demander au démarrage une identification de l'utilisateur pour assurer la confidentialité et l'intégrité des données. Le pharmacien et ses collègues doivent pouvoir consulter et manipuler la liste des utilisateurs (pharmaciens) qui seront identifiés par un identifiant et un mot de passe.
- **Le Stock :** Le logiciel doit gérer complètement le stock lors des différentes opérations d'entrée et de sortie de médicaments. L'utilisateur doit pouvoir ajouter des médicaments au stock avec leurs données (désignation, forme c'est à dire le type du médicament, le prix unitaire de vente, le prix unitaire d'achat et la quantité présente initialement). Le prix unitaire d'achat est calculé grâce à un coefficient sur le prix de vente qui est de 70% du prix de vente pour les médicaments, 50% pour le paramédical et 85,16% pour les laits. Le client doit pouvoir supprimer un médicament s'il n'est pas utilisé dans des opérations ou modifier ses données. Les formes des médicaments ne doivent pas être tapées pour chaque médicament car il n'y en a pas beaucoup. L'utilisateur doit pouvoir voir son stock complet sous forme d'un tableau avec une indication sur le nombre de produits, de médicaments différents et sa valeur en s'appuyant sur le prix de vente. Il faut aussi pouvoir imprimer le tableau représentant le stock pour les besoins d'inventaire.

- **Les échanges :** Le logiciel doit gérer les échanges de médicaments avec les pharmacies (c'est une pratique courante entre pharmaciens dans les zones éloignées car les fournisseurs ne font que 2 livraisons par jour au maximum et il faut répondre aux demandes des clients). L'utilisateur doit pouvoir manipuler une liste des pharmacies avec lesquels il a eu des échanges et il veut enregistrer le nom de la pharmacie, son adresse et son n° de téléphone. Il doit pouvoir ajouter un échange avec ses médicaments et la pharmacie concernée. A la sélection de la forme d'un médicament il veut saisir de façon intuitive la désignation du médicament ou avec une liste déroulante, puis sa quantité. Et il veut aussi voir la valeur de l'échange en même temps. Il y a deux type d'échange, en sortie (les médicaments seront retirés du stock) et en entrée (les médicaments seront ajoutés au stock). Toutes les données concernant les échanges doivent être enregistrées pour pouvoir être consultées. Le client veut pouvoir consulter les échanges avec une pharmacie précise, à une date précise ou dans une période entre deux dates (mois, année...) avec la possibilité de voir tout le détail des médicaments d'un échange précis.

- **Les ventes :** Le logiciel doit enregistrer les ventes et les opérations qui leurs sont liées. L'utilisateur doit pouvoir saisir la liste des médicaments de la vente de la même façon que celle des médicaments d'un échange et voir en même temps la valeur de la vente. Il y a trois types de vente :
 - La vente normale, et il faut enregistrer la date et l'heure de la vente, le nombre de médicaments et le montant de la vente sans le détail complet.
 - Une vente avec facture, et dans ce cas il faut enregistrer la vente de la même façon qu'une vente normale en ajoutant le nom du client. Il faut en plus générer une facture pour l'imprimer avec un numéro de facture ordonné croissant et l'enregistrer avec le n° de facture, la date, l'heure et le nom du client. L'utilisateur doit pouvoir manipuler une liste des factures qu'il a édité pour changer le client ou la date ou supprimer une facture. Il doit pouvoir consulter chaque facture et l'imprimer à n'importe quel moment. Un exemple de facture nous a été fourni par le pharmacien.
 - Une vente à crédit, et dans ce cas il faut enregistrer la vente de la même façon qu'une vente normale en sélectionnant le client de la vente dont on doit avoir enregistré plusieurs informations que nous allons voir dans la partie des crédits, et il faut aussi saisir le montant réglé. Le montant de la vente moins le montant réglé donnera la somme à ajouter au montant du crédit du client avec comme date de crédit la date de la vente. On doit aussi enregistrer le détail de la vente qui sera ajouté au détail du client pour pouvoir être consulté. Après chaque vente validée, les médicaments doivent être retirés du stock et ajouter à la commande courante que nous allons voir plus bas dans la partie des commandes.

L'utilisateur doit pouvoir consulter les ventes effectuées à une date ou une période entre deux dates avec un calcul de la recette et du nombre de vente et de médicaments.

- **Les commandes :** Pour assurer l'approvisionnement du stock, tous les médicaments vendus doivent être ajouté à la commande courante qui est la commande contenant tous les médicaments vendus après la dernière commande effectuée. L'utilisateur doit pouvoir consulter cette commande courante et la modifier. Quand la commande courante sera effectuée, une nouvelle commande vide prendra sa place et l'ancienne sera enregistrée avec l'heure pour être consultée. On doit pouvoir consulter les commandes avec leur valeur et nombre de médicaments à une date ou un intervalle entre deux dates avec la possibilité de voir le détail des médicaments.

- **Les crédits :** Avant d'effectuer une vente à crédit il faut saisir des données sur le client si c'est un nouveau client (Nom du client, son n° de carte d'identité nationale CIN et son n° de téléphone). Après la vente on ajoute le montant du crédit au client et on met à jour le montant qu'il lui reste à payer. A chaque client correspond un seul crédit dont le montant est la somme de tous les crédits qu'il a pris, son détail contient tous les médicaments qu'il a pris et sa date celle du dernier crédit pris. Il faut pouvoir ajouter les paiements des clients avec leur montant et leur date et mettre à jour le montant restant à payer. Il faut aussi pouvoir consulter les paiements pour modifier leur montant ou les supprimer. L'utilisateur doit pouvoir consulter les crédits d'un client ou d'une date ou d'une période entre deux dates avec la possibilité de voir le détail des médicaments et des paiements de chaque client ainsi que le montant du crédit et le reste à payer.
- **Les bons de livraison :** Le logiciel doit permettre la saisie des données d'un bon de livraison d'un fournisseur et mettre à jour le stock. L'utilisateur doit pouvoir manipuler une liste contenant ses fournisseurs (nom, adresse, n° téléphone). Un exemple de bon de livraison nous a été fourni par le client. Il faut aussi pouvoir consulter les bons de livraison par fournisseur, par date et période entre deux dates avec la possibilité de voir tout le détail du bon de livraison.
- **Fonctionnalités ajoutée :** Après accord du client nous avons ajouté une page avec les statistiques des ventes sous forme d'histogrammes et plusieurs informations calculées sur les résultats de consultation des différentes pages.

3. Conception de la base de données

3.1) Pourquoi une base de données ?

Gestpharma est un logiciel de gestion de stock qui offre à son utilisateur plusieurs fonctionnalités dont la consultation, l'enregistrement, la modification ou même la suppression de données relatives à son commerce. Ces informations devraient donc être stockées sous un format particulier qui pourrait offrir une souplesse en lecture et écriture tout en assurant la non-volatilité des données. La meilleure solution – si ce n'est la seule - était donc de représenter toutes ces informations sous la forme d'une base de données. Or cette base de données se doit d'être normalisée afin de supprimer la redondance logique, éviter les anomalies de stockage et résoudre les problèmes de reconnexion. Pour cela, nous avons opté pour la méthode de synthèse Merise que nous allons détailler dans les paragraphes suivants.

3.2) Conception de la base de données

Méthode de synthèse MERISE :

3.2.1) Dictionnaire des données

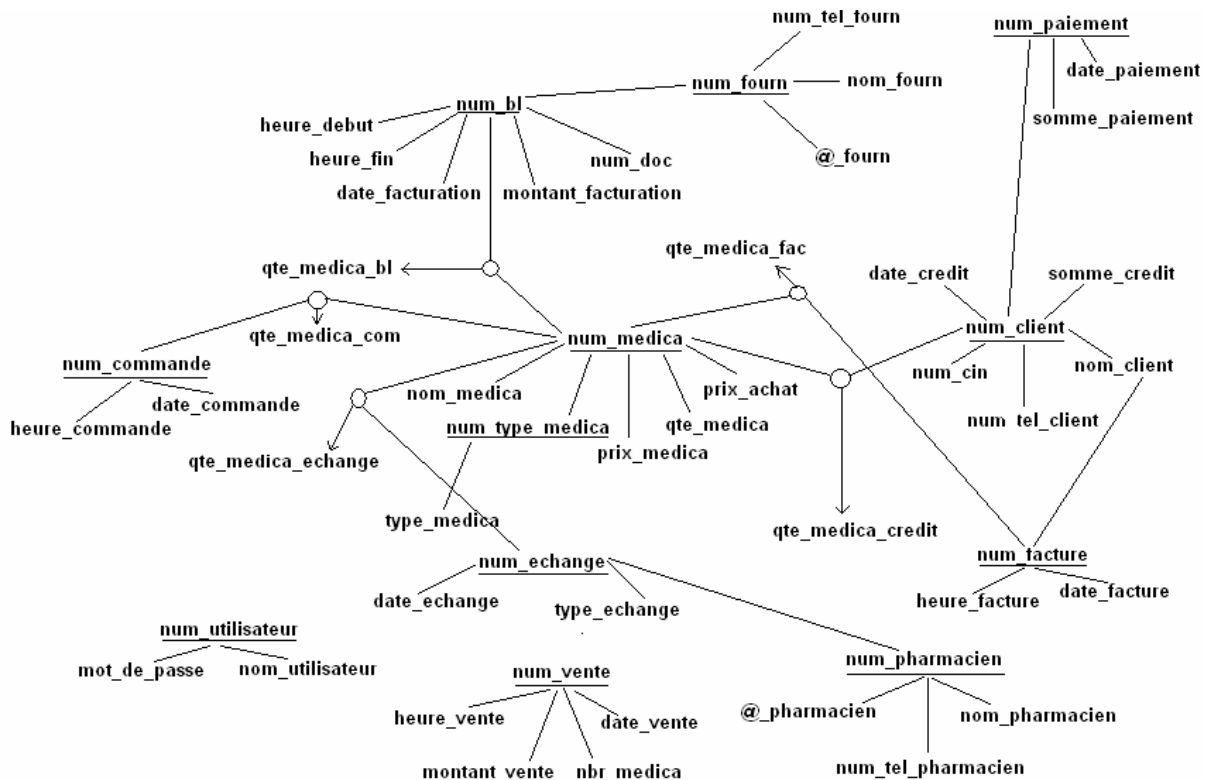
| Données | Type | Unité | Type de données | Opération de calcul | Règles intégrité | Document |
|----------------------|------|------------------|-----------------|---------------------|------------------|--------------|
| @_fourn ¹ | AN | | E | | | Fournisseur |
| @_pharmacien | AN | | E | | | Pharmacien |
| date_commande | AN | | E | | | Commande |
| date_credit | AN | | E | | | Client |
| date_echange | AN | | E | | | Echange |
| date_facturation | AN | | E | | | BonLivraison |
| date_facture | AN | | E | | | Facture |
| date_paiement | AN | | E | | | Paiement |
| date_vente | AN | | E | | | Vente |
| heure_commande | AN | | E | | | Commande |
| heure_debut | AN | | E | | | BonLivraison |
| heure_facture | AN | | E | | | Facture |
| heure_fin | AN | | E | | | BonLivraison |
| heure_vente | AN | | E | | | Vente |
| montant_facturation | N | Dhs ² | E | | | BonLivraison |
| montant_vente | N | Dhs | E | | | Vente |
| mot_de_passe | AN | | E | | | Utilisateurs |
| nbr_medica | N | | E | | | Vente |
| nom_client | A | | E | | | Client |
| nom_fourn | A | | E | | | Fournisseur |
| nom_medica | AN | | E | | | Stock |
| nom_pharmacien | A | | E | | | Pharmacien |
| nom_utilisateur | A | | E | | | Utilisateurs |
| num_bl ³ | N | | E | | | BonLivraison |
| num_cin ⁴ | AN | | E | | | Client |
| num_client | N | | E | | | Client |
| num_commande | N | | E | | | Commande |
| num_doc | N | | E | | | BonLivraison |
| num_echange | N | | E | | | Echange |
| num_facture | N | | E | | | Facture |

| | | | | | |
|----------------------------|----|-----|---|--|----------------|
| num_fourn | N | | E | | Fournisseur |
| num_medica | N | | E | | Stock |
| num_paiement | N | | E | | Paiement |
| num_pharmacien | N | | E | | Pharmacien |
| num_tel_client | N | | E | | Client |
| num_tel_fourn | N | | E | | Fournisseur |
| num_tel_pharmacien | N | | E | | Pharmacien |
| num_type_medica | N | | E | | Forme |
| num_utilisateur | N | | E | | Utilisateurs |
| num_vente | N | | E | | Vente |
| prix_achat | N | Dhs | E | | Stock |
| prix_medica | N | Dhs | E | | Stock |
| qte_medica | N | | E | | Stock |
| qte_medica_bl ³ | N | | E | | DetailBl |
| qte_medica_com | N | | E | | DetailCommande |
| qte_medica_credit | N | | E | | DetailClient |
| qte_medica_echange | N | | E | | DetailEchange |
| qte_medica_fac | N | | E | | DetailFacture |
| somme_credit | N | Dhs | E | | Client |
| somme_paiement | N | Dhs | E | | Paiement |
| type_echange | AN | | E | | Echange |
| type_medica | AN | | E | | Forme |

1 : @ = Adresse. 2 : Dirhams (monnaie du Maroc). 3: bl = Bon de Livraison.

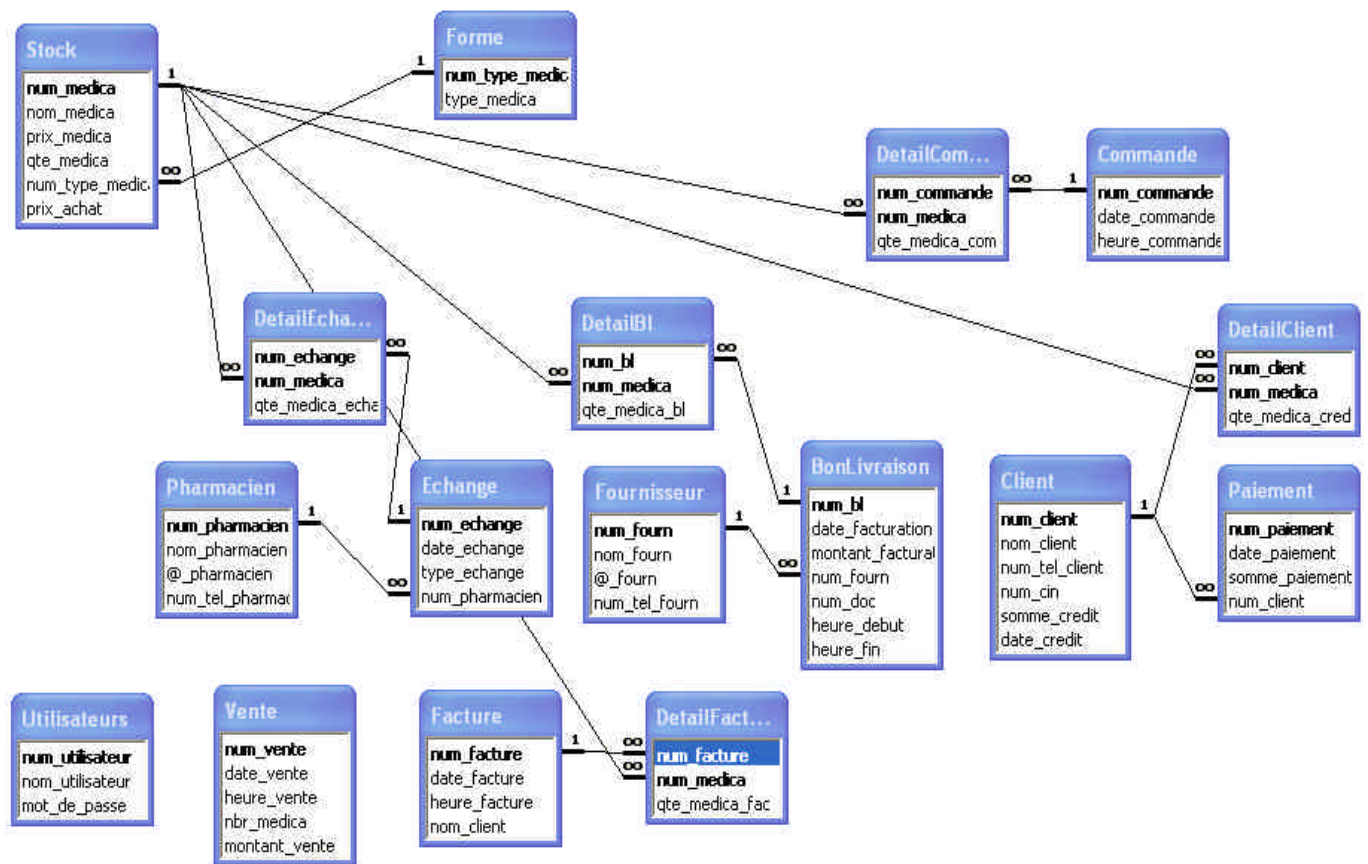
3.2.2) Graphe des dépendances fonctionnelles

Le graphe ci-dessus représente toutes les données et les relations qui les lient. Seules les dépendances fonctionnelles élémentaires et directes sont représentées.



- Dépendances fonctionnelles simples
- ⊙ → Dépendances fonctionnelle à partie gauche composée

3.2.3) Modèle conceptuel final



3.2.4) Représentation physique de la base de données

Après avoir établi une représentation logique de la base de données, nous devons choisir un outil informatique qui puisse permettre une représentation physique de ladite base.

Parmi les logiciels de gestion de BDD présents sur le marché, nous avons opté pour le plus classique et le plus utilisé : Microsoft Access.

3.2.4.1) Pourquoi Access ?

Le client, auquel est destiné ce logiciel, travaille sur une machine qui tourne sous Windows. Il fallait donc impérativement trouver un gestionnaire de base de données qui puisse s'adapter parfaitement avec la plateforme de la machine. Microsoft Access nous paraissait donc le mieux adapté à cette situation vue son utilisation massive dans le domaine de la gestion de base de données (grande quantité d'informations, de tutoriaux et de F.A.Q sur Internet) et le fait qu'il fait partie des programmes livrés à l'achat d'une machine tournant sous Windows (Microsoft Office).

Aux qualités citées ci-dessus s'ajoutent aussi une grande souplesse et une facilité d'utilisation.

3.2.4.2) Gestpharma et Access

Quelques points concernant la conception de la base de données sous Access :

- Les entités et les associations sont représentées par des tables.
- Ces tables sont initialement vides.
- Les données de type alphanumérique sont de type 'Texte' sous Access.
- Les données numériques sont de type 'Entier' (ex. qte_medica) ou 'Réel Double' (ex. prix_achat).
- Les identifiants des tables Client, Fournisseur, Forme, Paiement, Pharmacien Et Utilisateurs sont de type 'NuméroAuto' sous Access. Leur valeur s'incrémente automatiquement après chaque enregistrement, **mais ne se décrémente pas après une suppression**. Cette propriété permet d'insérer une donnée sans se soucier du numéro qu'on doit attribuer à son identifiant.
- Les dates ont été déclarées comme étant de type 'Texte' au lieu de 'Date'. Ceci est dû au fait que Access interprète les dates à la fois en français et en anglais (02/03/2005 signifie pour lui à la fois le 02 Mars2005 et le 03 Février 2005) ce qui pourrait nuire à la recherche d'informations par intervalle de dates.

❗ Concernant les cardinalités, représentées sur le modèle conceptuel, Access suit une logique inverse à celle utilisée par l'approche Merise (logique UML) !

Exemple : *Relation Stock - Forme*

Une forme peut être affectée à plusieurs médicaments et un médicament ne peut avoir qu'une seule forme précise.

4. Développement du logiciel

4.1) L'environnement de développement

4.1.1) Borland C++ Builder

C++Builder est un **IDE** (environnement de développement intégré). Il regroupe tout un ensemble d'outils permettant d'effectuer un maximum de tâches de développement au sein du même environnement de travail. C++ Builder est de plus un environnement de développement visuel C++ RAD (Rapid Application Development). Il permet de construire rapidement des applications basées sur une GUI et interagissant avec une base de donnée en utilisant des composants et simplifie au maximum l'écriture du code et la réalisation de l'interface. On peut ainsi très rapidement se consacrer à la partie "métier" du code (le code réellement utile de l'application).

4.1.2) Les composants utilisés

Pour réaliser notre logiciel nous avons utilisé plusieurs composants offerts par l'environnement de développement dans deux bibliothèques : la bibliothèque de composants visuels (VCL) et la bibliothèque de composants multiplate-forme (CLX). Ces composants se divisent en deux groupes, ceux liés à l'utilisation de la base de données et ceux des éléments de notre interface graphique.

Les composants liés à la base de données :

Nous avons choisi de travailler avec des composants ADO (ActiveX Data Object) permettant d'accéder aux bases de données de façon beaucoup plus facile sans se soucier de tout ce qui est allocation des environnements de travail (cf. programmation avec la couche basse d'ODBC). ADO fournit des objets qui permettent de se connecter à une base et de réaliser des requêtes SQL sur cette base et voici ceux que nous avons utilisés :

- **TADOConnection** : Se connecte à un stockage de données ADO et les autres composants accèdent à la base de données à travers lui.
- **TADOCommand** : Permet d'exécuter des requêtes SQL qui ne retournent pas de résultats (INSERT, UPDATE, DELETE).
- **TADODataset** : Représente un ensemble de données récupéré dans une base de données avec une requête SQL et il permet de manipuler cet ensemble.
- **TADOTable** : représente une image d'une table dans une base de données et permet de la manipuler.
- **TADOQuery** : Permet d'exécuter des requêtes SQL sur une base de données et de manipuler le résultat.
- **TDataSource** : Sert d'interface entre un composant d'ensemble de données et les contrôles orientés données d'une fiche comme les tableaux ou les listes déroulantes.
- **TDBGrid** : Affiche et manipule les enregistrements d'un ensemble de données dans une grille tabulaire.
- **TDBNavigator** : Est utilisé pour se déplacer dans un ensemble de données et effectuer des actions sur les données (par exemple, insérer un nouvel enregistrement ou expédier un enregistrement).
- **TDBLookupComboBox** : Représente une liste déroulante qui identifie un ensemble de valeurs de champ d'un ensemble de données à l'aide d'un ensemble de valeurs correspondantes d'un autre ensemble de données (utilisé pour les jointure naturelles).

Les composants de l'interface graphique :

Nous avons choisi d'utiliser les composants de la VCL qui sont des objets visuels qu'on peut manipuler lors de la conception et gérer leurs propriétés, leurs méthodes et leurs événements.

- **TForm** : représente une fenêtre (fiche) standard d'une application (la fenêtre principale).
- **TFrame** : C'est un conteneur de composants qui peut être imbriqué dans des fenêtres, ou dans d'autres cadres (les pages insérées dans la fenêtre principale pour l'interface SDI).
- **TMainMenu** : Encapsule dans une fenêtre une barre des menus et ses menus déroulants.
- **TLabel** : C'est un contrôle non-fenêtré qui affiche du texte ou un bitmap dans une fiche.
- **TEdit** : C'est un contrôle de saisie monoligne.
- **TRadioGroup** : Représente un groupe de boutons radio qui fonctionnent ensemble.
- **TGroupBox** : représente une boîte groupe contenant plusieurs composants.
- **TComboBox** : Combine une boîte de saisie et une liste déroulante.
- **TPanel** : implémente un contrôle volet générique contenant plusieurs composants.
- **TBitBtn** : C'est un contrôle bouton poussoir qui peut contenir une image sur sa face.
- **TStringGrid** : C'est un contrôle grille (tableau) conçu pour simplifier la gestion de chaînes et d'objets associés et leur affichage.
- **TImage** : Affiche une image graphique (bmp, jpeg...).
- **TRichEdit** : Encapsule un contrôle éditeur de texte formaté de Windows.
- **TDateTimePicker** : Affiche une boîte liste permettant la saisie de dates ou d'heures.
- **TChart** : Permet de construire et d'afficher des graphes (histogrammes, camembert...).
- **TPrintDialog** : Affiche une boîte de dialogue d'impression.
- **RXGifAnimator** : Affiche une image graphique animée de type GIF.
- **TPrintDialog** : Affiche une boîte de dialogue pour les options d'impression.

4.2) Développement du logiciel

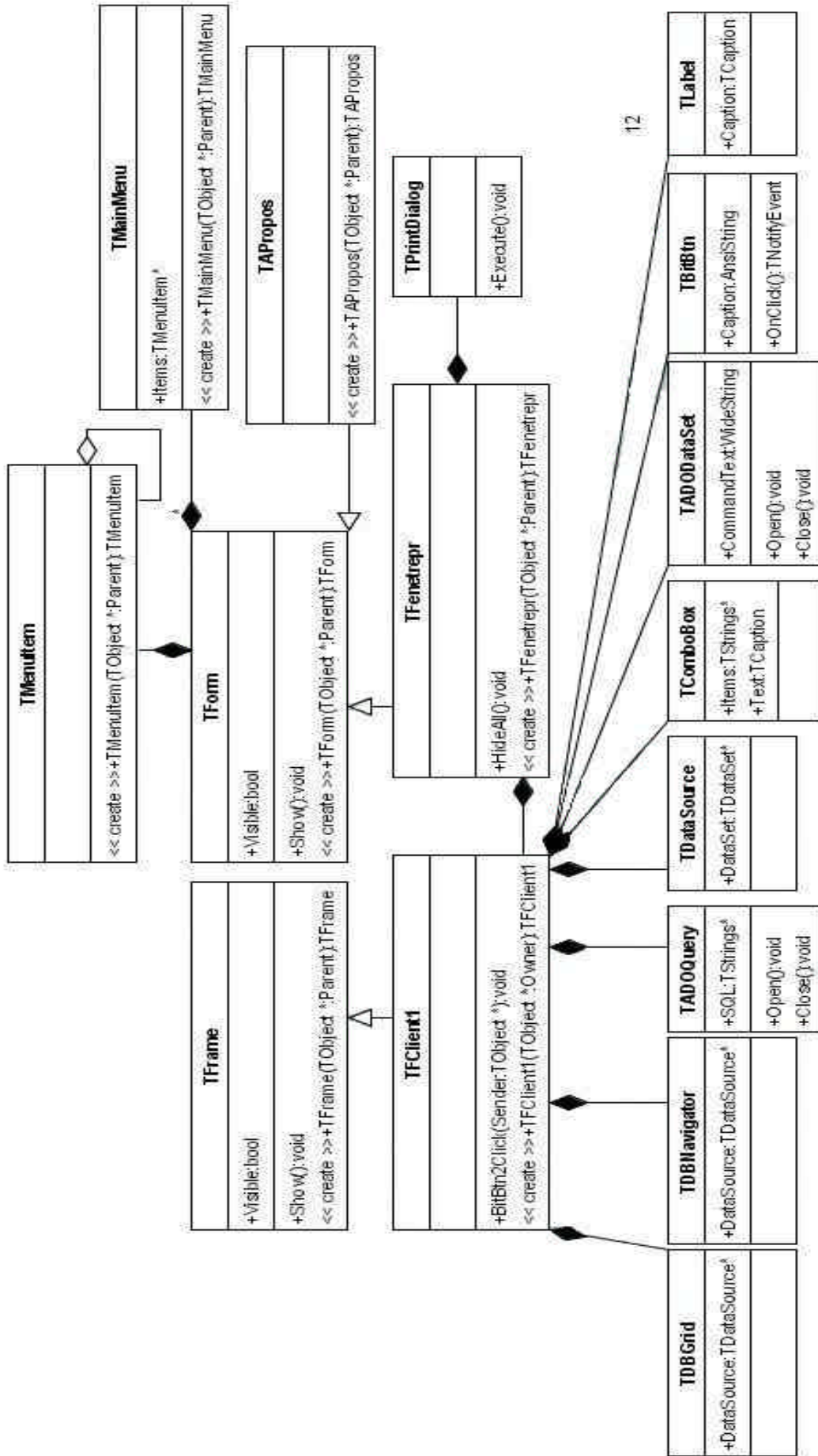
4.2.1) Analyse et Modélisation

Le point central du logiciel est sa GUI, donc toutes les fonctionnalités seront intégrées à l'aide de composants visuels intégrés à l'interface et interagissant avec des composants de bases de données.

La modélisation du logiciel suit le modèle de son interface. Cette interface est de type SDI avec une fenêtre principale (TForm) et plusieurs frames (TFrame) à l'intérieur accessibles grâce à une barre de menu (TMainMenu). Les frames sont composées de plusieurs composants visuels et de composants orientés base de données. Certains composants sont écouteurs d'événements et exécutent des méthodes en réponse.

Dans la page qui suit, on peut voir une partie de la modélisation UML de l'application. Pour des soucis de place et de lisibilité, nous avons choisi de modéliser une seule frame, celle de la liste des clients « TClient1 », et nous n'avons pas noté tous les attributs et méthodes des objets. Les autres frames suivent le même modèle. Le menu de la fenêtre principale contient plusieurs items ordonnés suivant leur fonctionnalité, et ils serviront à afficher les différentes frames.

Diagramme de classes UML :



12

4.2.2) Conception du logiciel

4.2.2.1) La fenêtre principale

La fenêtre principale est un objet héritant de la classe TForme. Elle représente une fenêtre standard affichée en plein écran. Elle se compose d'une barre de menu (TMainMenu), elle-même composée de plusieurs items (TMenuItem). A cette fenêtre viendront s'ajouter toutes les frames (TFrame) du logiciel. Le menu permettra de naviguer à travers les pages en utilisant une méthode HideAll() de la fenêtre principale qui cache toutes les frames puis en appelant la méthode Show() de la frame qu'on désire afficher avec des opérations de mise à jour des composants de la frame liés à la base de données. La fenêtre principale permet aussi d'imprimer la page visible avec une boîte de dialogue d'option d'impression (TPrintDialog) et permet aussi d'afficher l'aide du logiciel qui est sous forme d'un fichier .hlp conçu avec le logiciel « Help&Manual » et compilé avec Help Workshop.

4.2.2.2) Les pages du logiciel

⇒ La page d'identification et la page de gestion des utilisateurs :

La page d'identification est la première page qui s'affiche au démarrage du logiciel qui est protégé par mot de passe. Il y a aussi une page pour gérer les utilisateurs avec leur identifiant et leur mot de passe.

⇒ Les pages des fonctionnalités liées au stock :

Il y a 4 pages pour gérer le stock. Une pour ajouter de nouveaux médicaments au stock. Une autre pour éditer le stock et modifier les données concernant un médicament. Une troisième page pour la liste de formes des médicaments présents dans le stock. Et enfin une page pour voir le stock complètement et pouvoir l'imprimer.

⇒ Les pages des fonctionnalités liées à la vente :

Il y a 4 pages pour gérer les ventes. Une pour effectuer une vente (normale, à crédit ou avec facture). Une autre pour consulter les ventes. Une troisième page pour consulter les factures et pouvoir les imprimer. Et enfin une page pour consulter les statistiques des ventes sous forme d'histogrammes.

⇒ Les pages des fonctionnalités liées aux commandes :

Il y a 2 pages pour gérer les commandes. Une pour consulter ou modifier la commande courante. Et une deuxième pour consulter les commandes effectuées.

⇒ Les pages des fonctionnalités liées aux échanges :

Il y a 3 pages pour gérer les échanges. Une pour la liste des pharmacies (consultation et modification). Une autre pour ajouter un échange avec une pharmacie. Et enfin une page pour consulter les échanges effectués avec leurs données.

⇒ Les pages des fonctionnalités liées aux crédits :

Il y a 3 pages pour gérer les crédits. Une pour la liste des clients (consultation et modification). Une autre pour ajouter les paiements d'un client. Et enfin une page pour consulter les crédits des clients avec leurs données.

⇒ Les pages des fonctionnalités liées aux fournisseurs et leurs bons de livraison :

Il y a une page permettant de consulter les bons de livraisons enregistrés et visualiser leurs données. Cette page permet aussi de voir la liste des fournisseurs (consultation et

modification). Une autre page permet d'ajouter les bons de livraisons avec une interface proche du document fournit par le client.

4.2.2.3) Fonctionnalités globales

Toutes les actions de l'utilisateur sont contrôlées et toutes les erreurs écartées. Les zones de saisie dispose d'une validation de la saisie suivant le type de caractères qu'ils peuvent contenir. Les opérations incohérentes sont notifiées à l'utilisateur. Toute opération enregistrée est réversible par modification ou suppression.

4.2.3) Installation du logiciel

Nous avons utilisé InstallShieldExpress pour fabriquer un fichier d'installation automatique qui installe le logiciel sur la machine cible. Le logiciel est composé d'un exécutable « Gestpharma.exe » et d'un dossier Data contenant la base de données « Gestpharma.mdb », l'aide « Gestpharma.hlp » et les différentes images insérées dans le logiciel. L'exécutable a besoin de 2 DLL (borlndmm.dll et cc3260mt.dll) et 6 packages .bpl (adortl60.bpl, tee60.bpl, dbrtl60.bpl, rtl60.bpl, vcl60.bpl, vcldb60.bpl). Nous avons choisi de fournir le logiciel sur CD-ROM avec exécution automatique de l'installation. Le fichier d'installation « Setup.exe » installe le logiciel avec ses données, place les DLLs et les packages dans le répertoire système de Windows et ajoute un raccourci dans le menu démarrer.

5. Conclusion

Ce projet nous a permis d'avoir une approche complète du développement logiciel. Il nous a permis de nous initier au contact avec le client et à l'analyse des besoins et du cahier de charge. Nous avons aussi pu nous initier au cycle complet du développement logiciel de la conception à la validation en passant par les différentes étapes incrémentales de codage et de tests. Ce travail nous a appris à concevoir une base de données complète en utilisant la méthode merise et en se basant sur les besoins de l'application à réaliser. On a aussi appris à utiliser le SGBD Microsoft Access et l'environnement de développement très performant Borland C++ Builder. On a aussi pu découvrir le fonctionnement d'une pharmacie et les besoins de gestion informatique qu'ils peuvent avoir.

Notre client a beaucoup apprécié l'application et il la trouve ergonomique, simple et complète. Il nous a proposé de jouer le rôle de commercial pour essayer de distribuer le logiciel auprès des différentes pharmacies du royaume. Nous avons aussi reçu plusieurs demandes pour concevoir des logiciels de gestion de stock pour d'autres commerces. Et avec le savoir-faire que nous avons acquis dans ce projet, nous sommes capable de concevoir ce type de logiciel en moins d'un mois ce qui nous permet de répondre rapidement aux clients potentiels.

Ce travail nous a donné un avant-goût du métier de développeur et il nous a permis de concevoir pour la première fois une vraie application pour un vrai client et cela constitue une grande satisfaction personnelle et professionnelle qui signe le début d'aboutissement de notre formation.

6. Annexe

6.1) Documentation du logiciel

6.1.1) La page d'identification et les utilisateurs

La page d'identification est la première page qui s'affiche au démarrage du logiciel. Elle comporte un champs de saisie pour l'identifiant de l'utilisateur et un champs pour son mot de passe. Vous validez l'identification en cliquant sur le bouton OK. Si l'identification réussie la barre de menu de la fenêtre principale du logiciel devient active, la page de vente s'affiche et vous pouvez naviguer grâce à la barre de menu dans les différentes rubriques du logiciel.

Cette page est liée à la table Utilisateur de la base de données pour vérifier l'identification grâce à un composant TADODataSet qui exécute des requêtes SQL et récupère le résultat.

Pour gérer les utilisateurs et leur mot de passe, une liste des utilisateurs est dans la page accessible par le menu "Utilisateurs" puis "Liste des Utilisateurs".

Cette page comporte un tableau (TDBGrid) représentant une image de la table Utilisateur, grâce à un composant TDataSource lié à un TADOQuery qui exécute la requête « SELECT * FROM Utilisateur » et stock le résultat. On peut insérer un nouvel utilisateur, modifier son identifiant et/ou mot de passe ou supprimer un utilisateur en utilisant le tableau éditable et le composant TDBNavigator qui est lié au TADOQuery à travers un composant TDataSource. Les composants liés à la base de données sont connectés grâce à leur propriété « ConnexionString » qui spécifie tous les paramètres.

6.1.2) Les pages des fonctionnalités liées au stock

⇒ La page d'ajout de médicaments au stock :

La page d'ajout au stock est accessible par le menu "Stock" puis "Ajouter au stock".

Cette page comporte un tableau (TDBGrid) qui représente une image de la table Stock avec une jointure naturelle avec la table Forme grâce à un composant TDataSource lié à un TADOQuery. Ce tableau montre tous les médicaments présents dans le stock avec leur forme, leur prix unitaire d'achat et de vente et leur quantité dans le stock. En haut à gauche du tableau, vous avez un bouton "Actualiser" qui sert à actualiser l'affichage du tableau en cas de ralentissement à l'insertion d'un nouveau médicament qui doit normalement s'afficher instantanément en haut du tableau (on utilise la méthode Requery() de l'objet TADOQuery). La liste déroulante en dessous du bouton "Actualiser" sert à changer l'ordre des médicaments affichés dans le tableau, soit par ordre d'ajout soit par ordre alphabétique de la désignation du médicament (c'est un changement de l'ORDER BY de la requête SQL du TADOQuery).

La barre en dessous du tableau sert à ajouter un nouveau médicament dans le stock avec un champs de saisie pour la désignation, une liste déroulante pour la forme qui est rempli avec les données de la table Forme à chaque affichage de la page (clique sur le menu "Ajouter au stock".), un champs de saisie pour le prix d'achat et le prix de vente est automatiquement calculé suivant le forme et saisie avec possibilité de le modifier puis un champs de saisie pour la quantité. Le bouton ajouter vérifie la validité des champs et ajoute le nouveau médicament dans le stock grâce à une requête INSERT exécutée par un composant TADOCCommand. La connexion avec la base de données est assurée avec un

composant TADOConnection. Les différentes lectures de la base de données sont effectuées avec un composant TADODataset.

⇒ **La page de la liste des formes :**

La liste des formes est dans la page accessible par le menu "Stock" puis "Liste des Formes". Cette page comporte un tableau (TDBGrid) représentant une image de la table Forme, grâce à un composant TDataSource lié à un TADOQuery qui exécute la requête « SELECT * FROM Forme » et stock le résultat. On peut insérer une nouvelle forme, modifier son nom ou la supprimer en utilisant le tableau éditable et le composant TDBNavigator qui est lié au TADOQuery à travers un composant TDataSource. Les composants liés à la base de données sont connectés grâce à leur propriété « ConnexionString » qui spécifie tous les paramètres.

⇒ **La page d'édition du stock :**

Pour éditer le stock, c'est à dire modifier les données concernant un médicament (Désignation, Prix, quantité ou forme), il faut aller à la page d'édition du stock en cliquant sur le menu "Stock" puis "Editer le stock".

Cette page comporte un tableau (TDBGrid) qui représente une image de la table Stock grâce à un composant TDataSource lié à un TADOQuery. La jointure naturelle avec la table Forme est réalisée avec un objet TDBLookupComboBox qui met dans la table Stock le n° de forme correspondant à celle sélectionnée dans la liste et qui est connecté à la table Forme avec un composant TADOTable. On peut modifier les données d'un médicament en utilisant le tableau éditable et le composant TDBNavigator qui est lié au TADOQuery à travers un composant TDataSource. Pour modifier la forme d'un médicament, il faut utiliser la liste déroulante. Cette liste donne la forme du médicament courant sélectionné dans le tableau. La liste déroulante en haut à droite du tableau sert à changer l'ordre des médicaments affichés dans le tableau (par ordre d'ajout, ordre alphabétique, prix croissant, quantité croissante ou forme, c'est un changement de l'ORDER BY de la requête SQL du TADOQuery).

La liste déroulante en dessous permet de rechercher un médicament dans le tableau et de sélectionner sa ligne (utilise la méthode Locate() de TADOQuery). La liste est à écriture intuitive de la désignation des médicaments. Pour rechercher le médicament il faut cliquer sur le bouton « Rechercher » après avoir tapé le nom du médicament ou une partie du nom dans la liste. La connexion avec la base de données est assurée avec un composant TADOConnection. Les différentes lectures de la base de données sont effectuées avec un composant TADODataset.

⇒ **La page pour voir le stock :**

Pour voir le stock il faut cliquer sur le menu "Stock" puis "Voir le stock". Cette page comporte un tableau (TDBGrid) qui représente une image de la table Stock avec une jointure naturelle avec la table Forme grâce à un composant TDataSource lié à un TADOQuery. Ce tableau montre tous les médicaments présents dans le stock avec leur forme, leur prix unitaire d'achat et de vente et leur quantité dans le stock. A droite du tableau, il y a un panel avec des labels représentant des informations calculées sur le stock (valeur, nombre de médicaments, nombre de produits).

On peut l'imprimer en cliquant sur le menu "Fichier" puis "Imprimer". Le tableau du stock peut tenir sur plusieurs pages, dans ce cas on a une indication "Page" au dessus du tableau qui donne le nombre de pages totales à imprimer, pour changer de page il faut cliquer sur les boutons "Précédente" ou "Suivante" qui changeront automatiquement la page affichée du tableau. La connexion avec la base de données est assurée avec un composant TADOConnection.

6.1.3) Les pages des fonctionnalités liées à la vente

⇒ La page pour effectuer une vente :

Pour effectuer une vente il faut cliquer sur le menu "Vente" puis "Effectuer une vente", ou sur la touche F1 du clavier.

Pour effectuer une vente il faut suivre deux étapes: remplir la liste de médicaments de la vente, puis valider la vente suivant son type (vente normale, avec facture ou avec crédit).

La liste des médicaments est un tableau (StringGrid) de 5 colonnes. Pour la remplir il faut sélectionner une forme dans la liste déroulante Forme (remplit avec les données de la table Forme à chaque affichage de la page) ce qui remplit la liste des médicaments avec ceux de la forme sélectionnée (table Stock), puis sélectionner le nom d'un médicament avec saisie intuitive dans la liste Médicament, puis saisir sa quantité dans la vente. Il est possible de supprimer une ligne de la liste en cliquant sur le bouton « Supprimer Ligne ». Le panel sous le tableau affiche en permanence le montant de la vente. On peut valider 3 types de ventes :

Vente normale : pour cela il suffit de cliquer sur le bouton "Valider vente" après avoir rempli la liste des médicaments. Les médicaments de la vente seront retirés du stock (table Stock) et ajouté à la commande courante (table Commande et DetailCommande). La recette, le nombre de médicaments, la date et l'heure seront enregistrés dans la table Vente et une confirmation du succès de l'opération apparaît.

Vente à crédit : pour cela il faut cliquer sur le bouton "Vente à crédit", choisir le client auquel vous allez accorder le crédit (si c'est un nouveau client, il faut cliquer sur le bouton nouveau qui affiche la page de la liste des clients, et insérer le nouveau client avec ses données puis revenir à la page des ventes pour terminer la vente). Vous devez saisir aussi le montant réglé lors de la vente qui sera soustrait du montant de la vente pour donner la valeur du crédit accordé à ce client. La valeur de la vente sera ajoutée à la recette et le montant du crédit sera ajouté au crédit du client et une confirmation du succès de l'opération apparaît. Enfin, les mêmes opérations qu'une vente normale seront enregistrées.

Vente avec facture : pour cela il faut cliquer sur le bouton "Vente avec facture" après avoir saisi le nom du client dans le champ "Nom client". Une confirmation du succès de l'opération apparaît, puis la facture est affichée. Les mêmes opérations qu'une vente normale seront enregistrées. Sur la facture on peut changer les coordonnées de la pharmacie en haut à gauche de la facture et l'identité du client à droite de la facture en dessous de la date grâce à deux TRichEdit à droite de la facture. On peut imprimer la facture grâce au menu "Fichier" puis "Imprimer". La facture sera enregistrée dans la base de données (table Facture et DetailFacture) et vous pourrez la consulter pour changer sa date ou son client et la réimprimer. Les mêmes opérations qu'une vente normale seront enregistrées.

La connexion avec la base de données est assurée avec un composant TADODConnection. Les différentes lectures de la base de données sont effectuées avec un composant TADODDataSet est les insertion par un TADODCommand.

⇒ La page pour consulter les ventes :

Pour consulter les ventes il faut aller à la page de consultation des ventes grâce au menu "Vente" puis "Consulter les ventes".

Il y a plusieurs critères applicables à la consultation et le cadre Vente de la page fournit plusieurs boutons pour la consultation des ventes.

Le bouton "Consulter pour cette date" permet de consulter toutes les ventes effectuées à la date sélectionnée dans le champ "Date" (TDateTimePicker) qui représente un calendrier déroulant. Le bouton "Consulter pour cet intervalle" permet de consulter toutes les ventes effectuées dans l'intervalle de date entre le champ "Date" et le champ "Date intervalle".

Le bouton "Tout consulter" permet de consulter toutes les ventes.

Le résultat de la consultation s'affiche sous forme d'un tableau (TDBGrid connecté à un TDataSource lié à un TADODataset) comportant les informations (table Vente) sur chaque vente avec des données calculées sur le résultat en dessous du tableau. La connexion avec la base de données est assurée avec un composant TADOConnection.

⇒ **La page pour consulter les factures :**

Pour voir les factures que ont été édité lors des ventes, il faut aller à la page de consultation des factures en cliquant sur le menu "Vente" puis "Consulter les factures".

Cette page comporte un tableau (TDBGrid connecté à un TDataSource lié à un TADOQuery) représentant une image de la table Facture. On peut modifier leur date et le nom de leur client ou les supprimer grâce au tableau et au TDBNavigator. Et on peut aussi les afficher pour les imprimer en sélectionnant une facture dans le tableau et en cliquant sur le bouton "Voir cette facture". La facture s'affiche et vous pouvez l'imprimer en cliquant sur le menu "Fichier" puis "Imprimer". Les factures ont la mise en forme du model fournit par le client avec des labels et un tableau (StringGrid) représentant la listes de médicaments.

⇒ **La page pour consulter les statistiques des ventes :**

Pour consulter les statistiques des ventes sous forme d'histogramme représenté par un composant TChart il faut cliquer sur le menu "Vente" puis "Statistique des ventes". Les données sont regroupées par mois et concernent une année précise qu'on doit saisir. On peut consulter l'histogramme des recettes, du nombre de ventes, du nombre de médicaments ou les 3 sur un même graphe (liste déroulante) en cliquant sur le bouton « afficher ». On utilise la methode Add() du composant Chart pour ajouter les couples de données extraits de la table Vente avec un composant TADODataset connecté avec un TADOConnection.

6.1.4) Les pages des fonctionnalités liées aux commandes

⇒ **La page pour de la commande courante :**

La commande courante, menu « Commande » puis « Commande courante », contient la liste des médicaments vendus après la dernière commande effectuée. La liste des médicaments est un tableau StringGrid de 4 colonnes. Le cadre en dessus du tableau présente quelques information sur la commande notamment sa valeur et le nombre de médicaments. On peut ajouter des médicaments à la commande ou augmenter la quantité d'un médicament déjà présent dans la commande, et cela en utilisant la barre en dessous du tableau. La liste déroulante "Forme" (remplit avec les données de la table Forme à chaque affichage de la page) permet de sélectionner la forme d'un médicament que vous pouvez choisir ensuite dans la liste Médicament. Après cela il faudra saisir sa quantité à ajouter à la commande. Ensuite, il faut cliquer sur le bouton ajouter pour ajouter ce médicament à la commande représentée par le tableau du milieu. Pour effectuer la commande il faut cliquer sur le bouton "Effectuer commande" en dessous du tableau. Après cela la commande est enregistrée (table Commande et DetailCommande). Après avoir effectuer une commande une nouvelle commande courante vide au départ la remplacera et contiendra tous les médicaments vendus après. La connexion avec la base de données est assurée avec un composant TADOConnection. Les différentes lectures de la base de données sont effectuées avec un composant TADODataSet est les insertion par un TADOCommand.

⇒ **La page pour consulter les commandes :**

Pour consulter les commandes effectuées il faut aller à la page de consultation des commandes grâce au menu "Commande" puis "Consulter les commandes".

Il y a plusieurs critères applicables à la consultation et le cadre Commande de la page fournit plusieurs boutons pour la consultation.

Le bouton "Consulter pour cette date" permet de consulter toutes les commandes effectuées à la date sélectionnée dans le champ "Date" (TDateTimePicker) qui représente un calendrier déroulant. Le bouton "Consulter pour cet intervalle" permet de consulter toutes les commandes effectuées dans l'intervalle de date entre le champ "Date" et le champ "Date intervalle". Le bouton "Tout consulter" permet de consulter toutes les commandes.

Le résultat de la consultation s'affiche sous forme d'un tableau (TStringGrid) à 4 colonnes comportant les informations (table Commande et DetailCommande) sur chaque commande avec des données calculées sur le résultat en dessous du tableau. Le bouton "Voir le détail" affiche un deuxième tableau (TStringGrid) contenant les médicaments de la commande sélectionnée dans le premier tableau. Le bouton "Cacher le détail" permet de cacher le tableau de détail. Le bouton "Commande courante" affiche la page de la commande courante.

La connexion avec la base de données est assurée avec un composant TADOConnection. Les différentes lectures de la base de données sont effectuées avec un composant TADODataset.

6.1.5) Les pages des fonctionnalités liées aux échanges

⇒ La page de la liste des pharmacies :

La liste des pharmaciens est dans la page accessible par le menu " Echanges" puis "Liste des Pharmacies". Cette page comporte un tableau (TDBGrid) représentant une image de la table Pharmacien, grâce à un composant TDataSource lié à un TADOQuery qui exécute la requête « SELECT * FROM Pharmacien » et stock le résultat. On peut insérer une nouvelle pharmacie, modifier ses données ou la supprimer en utilisant le tableau éditable et le composant TDBNavigator qui est lié au TADOQuery à travers un composant TDataSource.

La liste déroulante à droite permet de rechercher une pharmacie dans le tableau et de sélectionner sa ligne (utilise la méthode Locate() de TADOQuery). La liste est à écriture intuitive du nom de la pharmacie (remplit avec les données la table Pharmacien à chaque affichage de la page). Pour rechercher la pharmacie il faut cliquer sur le bouton « Rechercher » après avoir tapé le nom de la pharmacie ou une partie du nom dans la liste.

Les composants liés à la base de données sont connectés grâce à leur propriété « ConnexionString » qui spécifie tous les paramètres.

⇒ La page pour ajouter un échange :

Pour ajouter un échange à la base de donnée il faut aller à la page d'ajout d'échange en cliquant sur "Echange" puis "Ajouter un échange".

Pour ajouter un échange il faut suivre 3 étapes: remplir la liste de médicament de l'échange puis sélectionner le nom de pharmacien et le type de l'échange et enfin valider l'échange.

La liste au milieu (tableau TStringGrid à 4 colonnes) doit contenir tous les médicaments de l'échange. La liste déroulante "Forme" (remplit avec les données de la table Forme à chaque affichage de la page) permet de sélectionner la forme d'un médicament que vous pouvez choisir ensuite dans la liste Médicament. Après cela il faudra saisir sa quantité dans l'échange que vous voulez ajouter. Ensuite, il faut cliquer sur le bouton ajouter pour ajouter ce médicament à la liste du milieu. Si ce médicament n'est pas présent dans le stock, il faut choisir sa forme, taper sa désignation à la main dans le champ médicament puis cliquer sur le bouton "Nouveau" qui affichera la page d'ajout dans le stock avec la forme et la désignation du médicament que vous avez tapé, et il ne vous reste plus

qu'à saisir son prix et sa quantité initiale (qui est normalement nulle) et cliquer sur "Ajouter" pour l'ajouter au stock, puis revenir à la page d'ajout d'échange pour l'ajouter à la liste.

Le bouton "Supprimer ligne" permet de supprimer la ligne sélectionnée dans le tableau des médicaments en cas d'erreur de saisie. La valeur de l'échange est affichée instantanément à l'ajout des médicaments dans la zone en haut à droite de la quantité.

Avant de valider l'échange il faut choisir la pharmacie dans la liste déroulante (remplit avec les données de la table Pharmacien à chaque affichage de la page), puis choisir le type de l'échange. Enfin il faut cliquer sur le bouton "Valider Echange". Une boîte de dialogue confirmera le succès de l'opération. Si la pharmacie de l'échange n'est pas présente dans la liste il faut aller à la liste des pharmacies, ajouter la nouvelle pharmacie avec ses informations puis revenir à la page d'ajout d'échange pour continuer l'enregistrement de l'échange. Une fois l'enregistrement validé avec succès, les quantités des médicaments seront ajoutées au stock si c'est un échange en entrée et soustraites si c'est un échange en sortie et les données seront enregistrées dans la table Echange et DetailEchange. La connexion avec la base de données est assurée avec un composant TADODataSet. Les différentes lectures de la base de données sont effectuées avec un composant TADODataSet est les insertion par un TADODCommand.

⇒ **La page pour consulter les échanges :**

Pour consulter les échanges il faut aller à la page de consultation des échanges grâce au menu "Echange" puis "Consulter les échanges". Il y a plusieurs critères applicables à la consultation et le cadre Echange de la page fournit plusieurs boutons pour la consultation.

Le bouton "Consulter pour cette pharmacie" permet de consulter tous les échanges effectués avec la pharmacie sélectionnée dans la liste déroulante "Pharmacie" (remplit avec les données la table Pharmacien à chaque affichage de la page).

Le bouton "Consulter pour cette date" permet de consulter tous les échanges effectués à la date sélectionnée dans le champ "Date" (TDateTimePicker) qui représente un calendrier déroulant. Le bouton "Consulter pour cet intervalle" permet de consulter tous les échanges effectués dans l'intervalle de date entre le champ "Date" et le champ "Date intervalle". Le bouton "Consulter pour cette pharmacie à cette date" permet de consulter tous les échanges effectués à la date sélectionnée dans le champ "Date" avec la pharmacie sélectionnée dans la liste déroulante "Pharmacie". Le bouton "Consulter pour cette pharmacie à cet intervalle" permet de consulter tous les échanges effectués dans l'intervalle de date entre le champ "Date" et le champ "Date intervalle" avec la pharmacie sélectionnée dans la liste déroulante "Pharmacie". Le bouton "Tout consulter" permet de consulter tous les échanges.

Le bouton "Supprimer Echange" permet de supprimer l'échange sélectionné dans le tableau de résultat de la consultation avec tout son détail (table Echange et DetailEchange).

Le résultat de la consultation s'affiche sous forme d'un tableau (StringGrid à 5 colonnes) comportant les informations sur chaque échange avec des données calculées sur le résultat en dessous du tableau (e nombre d'échanges, le nombre de médicaments total et la valeur totale). On peut voir le détail concernant un échange du résultat en cliquant sur un échange dans le tableau résultat puis en cliquant sur le bouton "Voir le détail". Le détail s'affiche sous forme d'un deuxième tableau (StringGrid) à droite du celui du résultat avec l'ensemble des médicaments de l'échange et leur quantité. Le bouton "Cacher le détail" permet de cacher le tableau de détail. La connexion avec la base de données est assurée avec un composant TADODataSet. Les différentes lectures de la base de données sont effectuées avec un composant TADODataSet est les insertion par un TADODCommand.

6.1.6) Les pages des fonctionnalités liées aux crédits

⇒ La page de la liste des clients :

La liste des clients est dans la page accessible par le menu "Crédit" puis "Liste des Clients". La page comporte un tableau (TDBGrid) représentant une image de la table Client, grâce à un composant TDataSource lié à un TADOQuery qui exécute la requête « SELECT * FROM Client » et stock le résultat. On peut insérer un nouveau client, modifier ses données ou la supprimer en utilisant le tableau éditable et le composant TDBNavigator qui est lié au TADOQuery à travers un composant TDataSource.

La liste déroulante à droite permet de rechercher un client dans le tableau et de sélectionner sa ligne (utilise la méthode Locate() de TADOQuery). La liste est à écriture intuitive du nom du client (remplit avec les données de la table Client à chaque affichage de la page). Pour rechercher le client il faut cliquer sur le bouton « Rechercher » après avoir tapé le nom du client ou une partie du nom dans la liste.

Les composants liés à la base de données sont connectés grâce à leur propriété « ConnexionString » qui spécifie tous les paramètres.

⇒ La page pour ajouter les paiements :

Pour ajouter les paiements d'un client qui a pris un crédit, il faut cliquer sur le menu "Crédit" puis "Ajouter un paiement". Cette page permet de voir le détail des paiements d'un client et la somme restant due. On peut aussi ajouter un nouveau paiement du client qui fera diminuer le montant dû et cela grâce au cadre "Paiement" de la page. La liste déroulante client (remplit avec les données de la table Client à chaque affichage de la page) permet de sélectionner un client dont le détail apparaîtra instantanément en dessous de ce cadre sous forme d'un tableau (TDBGrid représentant une image de la table Paiement pour le client sélectionné, grâce à un composant TDataSource lié à un TADODataset). Il faut ensuite saisir le montant de son paiement et cliquer sur « Valider » pour l'ajouter aux paiements de ce client et diminuer ainsi son crédit. Vous pouvez supprimer un paiement du client ou modifier sa somme directement sur le tableau TDBGrid en cliquant sur la cellule correspondante ou en utilisant le composant TDBNavigator qui est lié au TADODataset à travers un composant TDataSource. Un label en dessus du tableau informe si le crédit de ce client est soldé ou pas. Le cadre en bas de la page regroupe quelques informations utiles comme le total du montant des crédits de ce client, la date du crédit qui est la date du dernier crédit que ce client a pris, le nombre de paiements et le plus important le montant qui reste à payer qui est la différence entre le montant total du crédit et la somme des paiements du client (table Paiement et Client). La connexion avec la base de données est assurée avec un composant TADOConnection. Les différentes lectures de la base de données sont effectuées avec un composant TADODataset et les insertions par un TADOCommand.

⇒ La page pour consulter les crédits :

Pour consulter les crédits il faut aller à la page de consultation des crédits grâce au menu "Crédit" puis "Consulter les crédits".

Il y a plusieurs critères applicables à la consultation et le cadre Crédit de la page fournit plusieurs boutons pour la consultation. Le bouton "Consulter ce client" permet de consulter les informations sur le crédit du client sélectionné dans la liste déroulante "Client" (remplit avec les données de la table Client à chaque affichage de la page). Le bouton "Consulter cette date" permet de consulter toutes les informations sur le crédit des clients ayant pris leur crédit à la date sélectionnée dans le champ "Date" (TDateTimePicker) qui représente un calendrier déroulant. Le bouton "Consulter cet intervalle" permet de consulter toutes les informations sur le crédit des clients ayant pris leur crédit dans l'intervalle de date entre le champ "Date" et le champ "Date intervalle". Le bouton "Tout consulter" permet de consulter les informations sur le crédit de tous les clients. Le résultat de la consultation

s'affiche sous forme d'un tableau (StringGrid à 4 colonnes) comportant les informations sur chaque client (table Client Paiement et DetailClient) avec des données calculées sur le résultat en dessous du tableau (le total des montants, le nombre de crédits et le total de la somme du sur cette l'ensemble du résultat de la consultation). On peut voir le détail concernant un client du résultat en cliquant sur un client dans le tableau résultat puis en cliquant sur le bouton "Voir le détail". Le détail s'affiche sous forme de deux tableaux (StringGrid) à droite de celui du résultat avec l'ensemble des médicaments qu'a pris ce client en payant par crédit (table DetailClient), le détail des ses paiements (table Paiement) et des informations complémentaire sur son crédit en bas de la page (la somme total des crédits que vous lui avez accordé, la date de son dernier crédit, le nombre de paiements de ce client et le plus important la somme qu'il lui reste à payer). Le label en dessus du tableau informe si le crédit de ce client est soldé ou pas. Le bouton "Cacher le détail" permet de cacher le tableau de détail.

6.1.7) Les pages des fonctionnalités liées aux fournisseurs et bons de livraison

⇒ La page de la liste des fournisseurs :

La liste des fournisseurs est dans la page accessible par le menu "Bon livraison" puis "Consulter BI et fournisseurs". La page comporte un tableau (TDBGrid) représentant une image de la table Fournisseur, grâce à un composant TDataSource lié à un TADOQuery qui exécute la requête « SELECT * FROM Fournisseur » et stock le résultat. On peut insérer un nouveau fournisseur, modifier ses données ou le supprimer en utilisant le tableau éditable et le composant TDBNavigator qui est lié au TADOQuery à travers un composant TDataSource.

⇒ La page de consultation des bons de livraison :

La consultation des BI se fait sur la page contenant la liste des fournisseurs. Le cadre « Recherche Bons de livraison » fournit un ensemble de boutons permettant une recherche selon plusieurs critères.

La liste déroulante des fournisseurs permet d'affiner la recherche en sélectionnant un fournisseur précis ou en choisissant l'option 'tous les fournisseurs'.

Le bouton « Recherche » permet de rechercher tous les bons de livraison dont la date de facturation est celle sélectionnée sur la liste déroulante des dates (TDateTimePicker).

Le bouton « Rechercher par période » permet une recherche par intervalle de dates (les deux dates représentant les bornes de l'intervalle sont sélectionnées sur les deux TDateTimePicker).

Le résultat de la recherche s'affiche sous la forme d'un tableau (TStringGrid). Sous ce tableau se trouvent deux boutons : Un bouton « Voir détails du on de livraison » qui permet d'afficher les détails du BI sélectionné, et un bouton « Supprimer BI » qui supprime le BI sélectionné.

Le Panel « Cumul Période T.T.C » situé au-dessous de la table affiche la somme des montants de facturation des BI présents dans la table.

⇒ La page d'ajout des bons de livraison :

Cette page est accessible par le menu "Bon livraison" puis "Ajouter un BL". Elle permet à l'utilisateur d'ajouter un nouveau bon de livraison à sa liste de BI en entrant tous les paramètres de ce dernier.

Cette page comporte :

- Une zone « En-tête » où l'utilisateur peut entrer le nom du fournisseur (dans une liste déroulante), le numéro de document (dans un champ TEdit) ainsi que la date de facturation (dans une liste déroulante de type TDateTimePicker).
- Une zone « Corps » qui sert à ajouter les médicaments dans le bon de livraison en entrant la Quantité (dans un champ TEdit), la forme (dans une liste déroulante) et le nom du médicament à ajouter (dans une liste déroulante).
Le bouton « Ajouter » (resp. « Supprimer ») sert à ajouter (resp. supprimer) un médicament à une table située à gauche et contenant la liste des médicaments présents dans le BI.
Si le médicament à ajouter n'existe pas, l'utilisateur devra à ce moment l'ajouter à la table des stocks (en cliquant sur le bouton « Nouveau ») puis revenir à la page "Ajouter un bon de livraison" pour pouvoir insérer le nouveau médicament.
- Les deux TEdit (H. Début et H. Fin) situés au-dessous de la table permettent d'entrer l'heure de début et de fin de la facturation du BI.
Un Label "Total P. Net T.T.C" situé au-dessous de la table affiche la montant du BI.
- Enfin un bouton « Valider » permet d'enregistrer le BI dans la table des bons de livraisons.