Pascal Brunot
Somair Kapoor
Olivier Lareynie
Nicolas Simoneau

Mai 2001

APPLICATION DE GESTION DES DONS MOBILIERS POUR L'ASSOCIATION « LES PETITS FRERES DES PAUVRES ».

Rapport d'Analyse - Conception

Table des matières

1. ANALYSE 4
1.1. Rappel du Cahier des Charges – Choix des Classes – Rôle de chaque classe... 4
1.1.1. Module « Gestion des propositions de dons » 4
1.1.2. Module « Gestion des stocks » 6
1.2. Graphique des relations entre les classes 9
1.3. Prévision d'interface graphique 10
1.3.1. Module Fichier 10
1.3.2. Module Gestion des Offres 10
1.3.3. Menu Ventes 12
1.3.4. Module Etat des stocks 12
1.3.5. Module « Rechercher » 14
1.3.6. Module « Statistiques » 15
1.3.7. Module « Fichier » 15
2. CONCEPTION 16
2.1. La base de données 16
2.1.1. Architecture globale 16
2.1.2. Conception des tables 17
2.2. Détails des méthodes et commentaires 19
2.2.1. La classe Offre 19
2.2.2. La classe Entrepot 21
2.2.3. La classe VenteBiannuelle 22
2.2.4. La classe DepotVente 22
2.2.5. La classe Personne 22
2.2.6. La classe Tiers 23
2.2.7. La classe Adherent 23
2.2.8. La classe Objet 24
2.2.9. La classe Dimensionne 24
2.2.10. La classe Adimensionne 25
2.2.11. La classe Table 25
2.2.12. La classe Cuisiniere 26
2.2.13. La classe Meuble 26
2.2.14. La classe Electromenager 27
2.2.15. La classe Base_de_donnees 27
2.3. Résumés UML des méthodes et variables de classe. 28

Préambule

Le rapport d'analyse/conception correspond à une analyse approfondie du cahier des charges qui conduit à des choix de structure du logiciel (classes, interface graphique, persistance).

La partie ANALYSE du rapport rappelle les points important du cahier des charges qui vont justifier nos choix d'organisation du logiciel. De ce fait, cette partie :

1. liste les classes à utiliser ainsi que leur rôle ;
2. donne le graphique des relations entre les classes ;
3. propose une interface graphique ;
4. discute de la gestion de la persistance.

La partie CONCEPTION du rapport approfondit les choix faits dans la partie Analyse. Donc cette partie

1. développe la gestion de l'application avec des bases de données ;
2. liste l'ensemble des méthodes retenues en les commentant.

1. ANALYSE

1.1. Rappel du Cahier des Charges – Choix des Classes – Rôle de chaque classe

L'application comprendra principalement cinq modules accessibles à l'aide de menus. Ces modules sont la base de notre réflexion sur l'analyse et la conception de l'application. Ces modules sont :

- Un module de gestion des propositions de dons, désignées par le terme « offres » ;
- Un module de gestion des stocks de l'association ;
- Un module de recherche rapide dans les stocks;
- Un module proposant des statistiques sur les dons ayant transités par l'association ;
- Un module « Fichier » classique pour la gestion des sauvegardes et des paramètres de l'application.

Les modules « Rechercher », « Statistiques » et « Fichier » feront l'objet d'une étude spéciale dans la section 1.4. Gestion de la persistance.
Pour les modules « Gestion des propositions de dons » et « Gestion des stocks » qui correspondent au cœur de l'implémentation en JAVA, on adoptera le schéma d'analyse suivant :

A. Rappel des besoins de ce module exprimés dans le cahier des charges (cette partie est encadrée dans le rapport)
B. Analyse du cahier des charges et définition de classe(s) adaptée(s)

1.1.1. Module « Gestion des propositions de dons »

A. Rappel des besoins de ce module exprimés dans le cahier des charges

Une offre faite à l'association peut être décrite par les champs suivants :
- Date de réception de l'offre
- Type de matériel (quantité + nature ; exemple : 1 TV couleur) ;
- Référence de l'objet ;
- Nom du donateur potentiel ;
- Numéro de téléphone ;
- Adresse ;

- Description complémentaire ;

Ensuite, l'utilisateur doit pouvoir accepter ou non l'offre.
Un champ Acceptation de l'offre avec les valeurs *Oui* ou *Non* ou *En cours de traitement* est à prévoir.

Si l'offre de don est acceptée, il faut également renseigner :
- Date d'acceptation ;
- Par quel membre de l'association ;
- Date de transport ;
- Les transporteurs ;
- Les véhicules ;
- Le lieu de stockage prévu (*Garde Meuble* ou un *Dépôt Vente*).

L'offre sera alors validée et l'objet sera répertorié dans les stocks.

B. Analyse du cahier des charges et définition de classe(s) adaptée(s)

On définit une classe **Offre** regroupant toutes les informations nécessaires à la gestion des différentes propositions de dons telles que les dates et différentes personnes intervenants (donateur, enquêteur de l'association, transporteur…) La classe Offre contient donc des variables de classes qui référencent des personnes et des objets. Il faut donc créer des classes **Personnes** et **Objet**. Une instance de la classe **Offre** n'agrégera pas une **Personne** (car une personne peut donner, recevoir ou valider plusieurs Offres, cela conduirait à une forte redondance des données en mémoire) mais un **Objet**, puisqu'on a obligatoirement un objet=une offre. On utilisera des numéros de références pour pouvoir connaître les personnes qui agissent sur les offres (Ref_personne).

- La classe **Personne** permet d'avoir toutes les informations utiles sur les différentes personnes.
  On trouve dans la classe **Personne**, le *Nom*, la référence (*ref_personne*) et les coordonnées(*Tel ; Adresse*) de la personne ainsi qu'un champ *Note* pour donner des informations spécifiques sur la personne.
  Les personnes morales (les transporteurs, entrepots, dépôt-ventes, associations tierces) sont considérées comme appartenant à la classe **Personne**.
  Il existe néanmoins d'autres catégories de Personnes, qui peuvent hériter de la classe **Personne** : les adhérents, et les donneurs/receveurs. On choisit donc de créer deux classes **Adherents** et **Tiers** qui spécialisent la classe **Personne.**
  - La classe **Adherents** nous apporte les informations sur le *Prenom, Fonction* et l'A*ge* de la personne
  - La classe **Tiers** nous renseigne sur le *Prenom* et l'*Age* de la personne.

- La classe **Offre** agrège donc une classe **Objet** qui contient les informations pertinentes sur l'objet dont l'association peut accepter le don. Après analyse des propriétés communes ou non aux objets, il ressort deux catégories : les objets dont la connaissance des dimensions exacte est nécessaire pour la redistribution, et les autres pour qui la désignation suffit.
- Dés lors, on peut créer **Dimensionne** et **Adimensionne** qui spécialisent (extends) la classe **Objet :** cela permet de faire la distinction entre les objets dont on veut connaître des informations sur la dimension et les autres objets.
  - La classe **Dimensionne** explicite les dimensions des objets. Elle est spécialisée en 3 différentes classes qui permettent de classer les objets dans différentes catégories.:
    - **Table** ;
    - **Electromenager** ;
    - **Cuisinieres** ;
    - **MeubleLiterie.**
  - La classe **Adimensionne** spécialise les objets dont la connaissance de la dimension n'est pas nécessaire, tels que les couverts ; assiettes ; chaises ; etc.

- Cette construction des classes permet d'éviter l'utilisation et le stockage d'emplacement mémoires inutiles, chaque objet possédant exactement les informations qui le définissent (pas de largeur, hauteur, longueur pour une cuillère par exemple).

1.1.2. Module « Gestion des stocks »

A. Rappel des besoins de ce module exprimés dans le cahier des charges

L'application doit pouvoir permettre de visualiser les objets stockés par entrepôt, et d'enregistrer des sorties de stocks.
Lorsqu'un don quitte l'association, c'est qu'il a été soit vendu, soit donné. Il faut alors l'indiquer dans le menu « Redistribution d'un don ». La fenêtre qui s'ouvrira

Il y a quatre destinations possibles pour les dons reçus :

Garde Meubles des « Petits Frères des Pauvres »
Vente bi-annuelle
Dépôt-vente
Don direct

Objets mis au « Garde Meubles »
On veut connaître :
la date de dépôt au Garde meubles ;
la personne récupérant le don.

Objets destinés à la vente bi-annuelle :
Il s'agit ici d'enregistrer les informations comptables correspondantes :
le montant estimé de l'objet ;
le montant obtenu à la vente.

Objets destinés au dépôt-vente :
Il faut connaître :
la date du dépôt
le montant estimé
la date de la vente par le dépôt-vente
le montant de la vente versé par le dépôt-vente à l'association
Par ailleurs, il est souhaitable de pouvoir gérer plusieurs dépôt-ventes même si l'association ne travaille actuellement qu'avec un seul partenaire. On rajoutera donc un menu déroulant pour préciser le dépôt-vente recevant l'objet.

Objets donnés à une personne en difficulté
Informations générales sur le don :
Référence de l'objet donné
Description
Type de matériel (quantité + nature ; exemple : 1 TV couleur)
Date de réception de la demande
Date d'acceptation
Par quel membre de l'association
Informations sur le transport :
Date de transport
Les transporteurs
Les véhicules
Informations sur l'acquéreur du don :
Nom du demandeur (organisme ou particulier)
Nom du bénéiciaire
Numéro de téléphone
Adresse

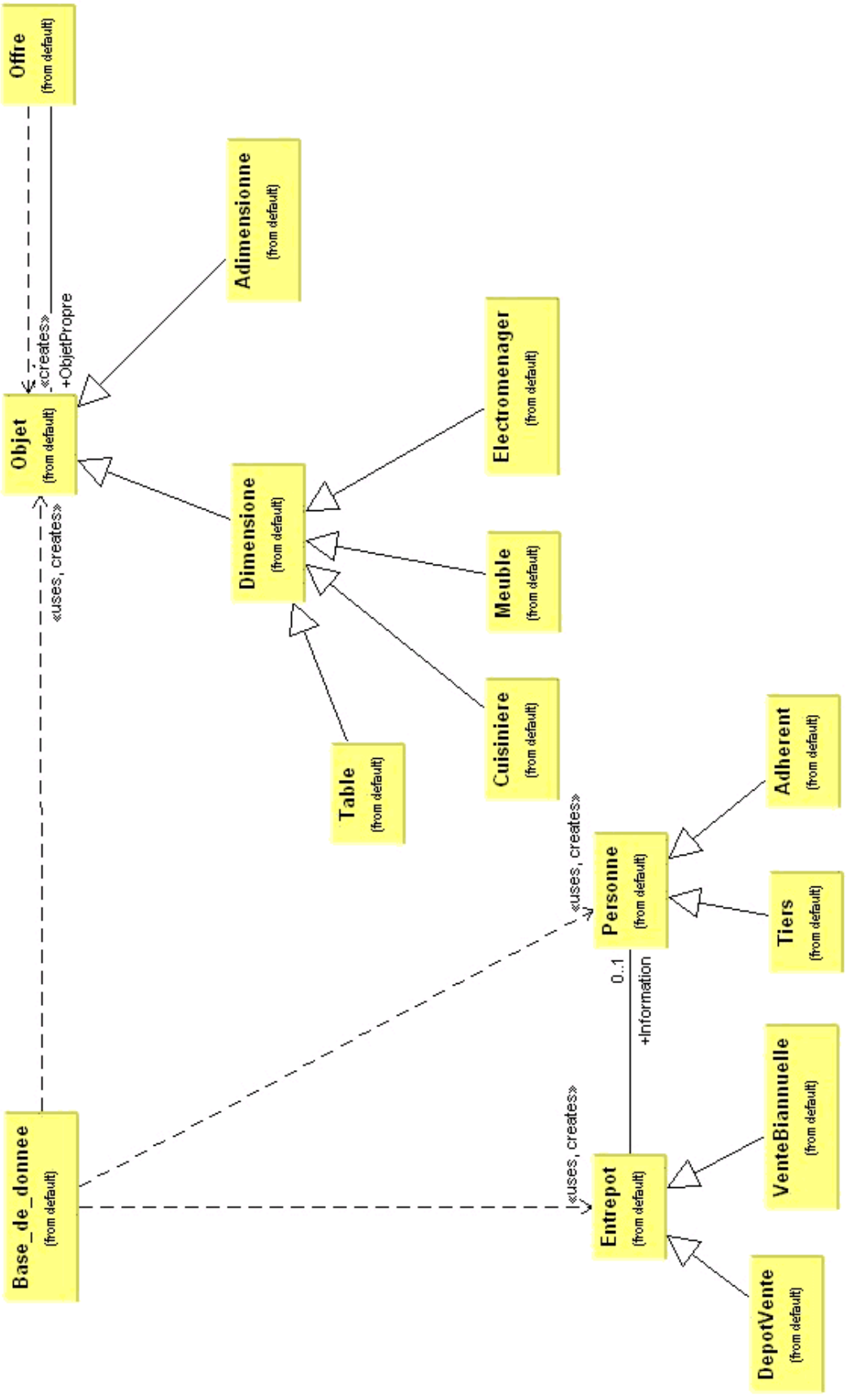
Un objet peut être confié au dépôt-vente puis récupéré par l'association parce qu'une personne âgée en fait la demande. Ce changement se fera au niveau du menu « Saisie d'un don » : il faudra changer le champ « Lieu de stockage actuel ». Le menu « Redistribution d'un don » correspond à un départ ***définitif*** *du don de l'association.*

B. Analyse du cahier des charges et définition de classe(s) adaptée(s)

On définit une classe **Entrepot** permettant de définir un entrepôt et d'effectuer les taches courantes au niveau de celui-ci (Voir si un objet se situe dans l'entrepôt ; Donner un objet à une personne ; Lister les stocks ; Déplacer un objet d'un entrepôt à un autre etc.…)

Il existe deux « entrepôts » particuliers. En effet les dons peuvent être vendus soit lors d'une vente bisannuelle soit par l'intermédiaire d'un dépôt-vente. Deux classes **VenteBiannuelle** et **DepotVente** spécialisent donc la classe **Entrepot**. Ce choix est logique dans la mesure où les objets placés en dépôt vente peuvent être donnés ; les méthodes de **Entrepot** doivent se retrouver dans **DepotVente**.

1.2. Graphique des relations entre les classes

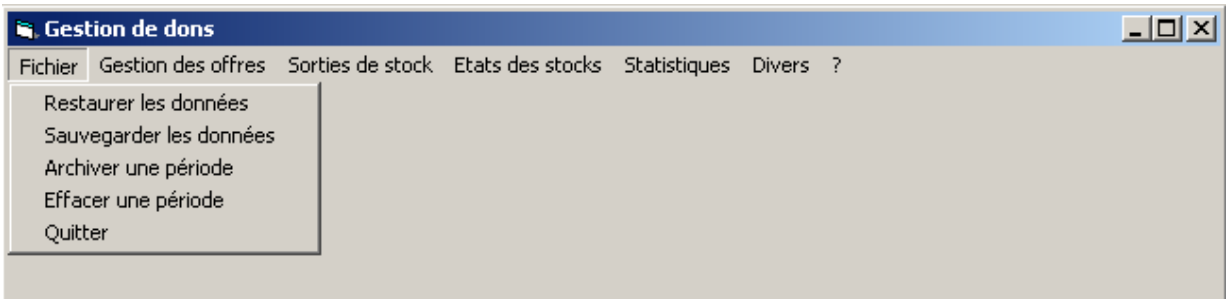


1.3. Prévision d'interface graphique

Chaque menu déroulant correspondra à un des modules décrit dans le Cahier des Charges.

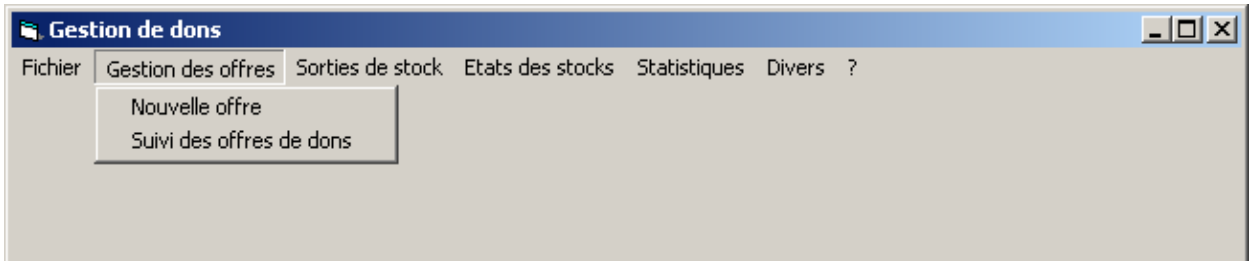
1.3.1. Module Fichier

Ce menu déroulant intégrera les fonctions « classiques » d'un logiciel gérant la persistance de fichier, telles que la sauvegarde, la restauration, la suppresion, etc.

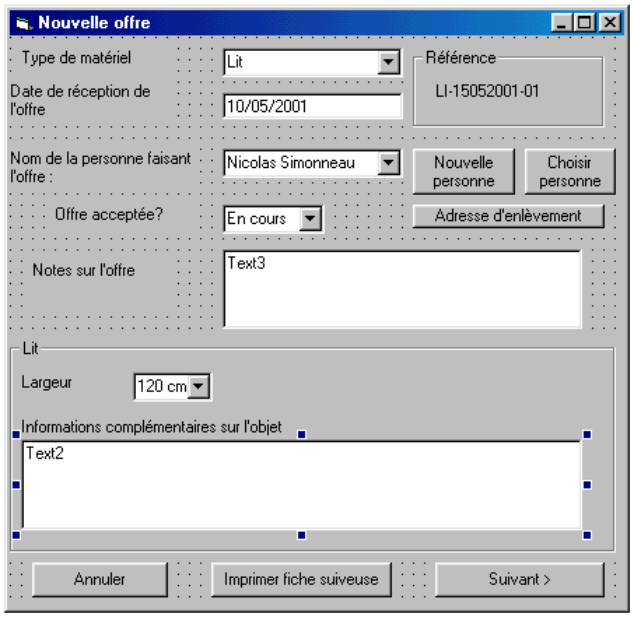


1.3.2. Module Gestion des Offres

Ce menu déroulant constituera le point d'entrée principal dans le logiciel, puisque qu'il est situé au cœur de la fonction même de celui-ci

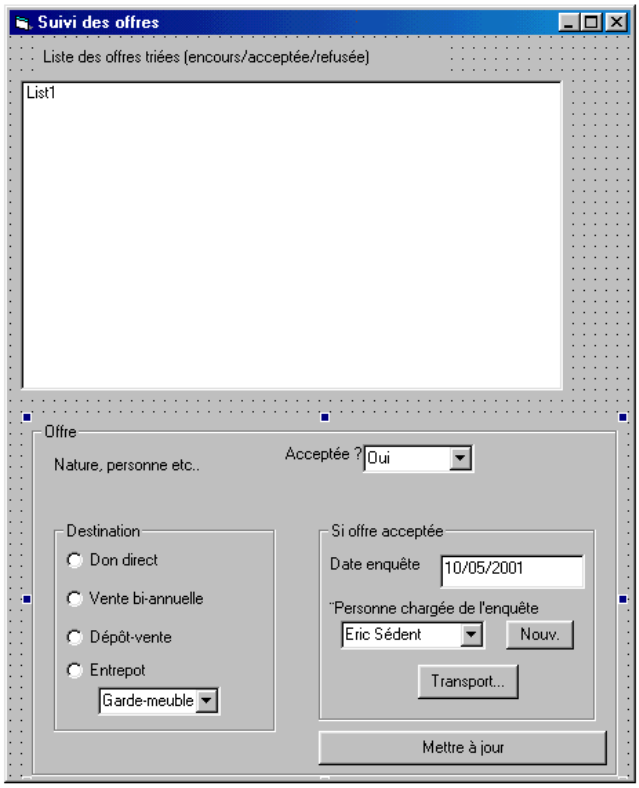


Le champ "nouvelle offre » donnera accés à la fenêtre de description de l'offre ci-dessous. Celle-ci permettra la saisie de tous les champs nécessaires à une première saisie :

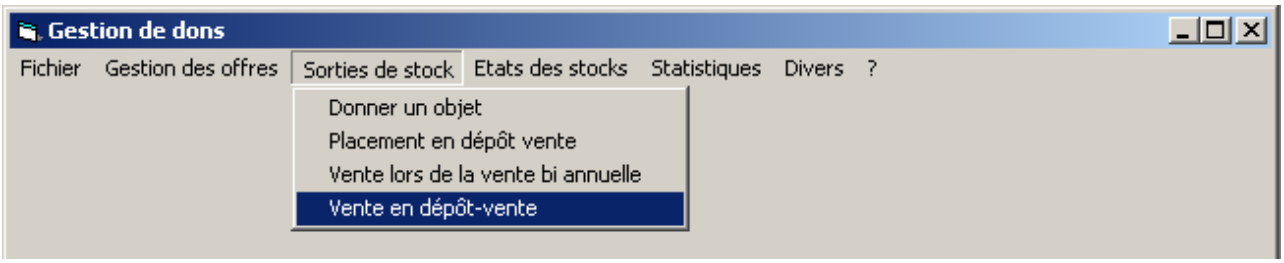


Le bouton suivant de cette fenêtre de dialogue (ou le champ « Suivi des offres » du menu déroulant) permettra l'accés à la fenêtre suivante. Celle assure le suivi du don, et renseigne donc :

- La destination du don : Garde-Meuble, Dépôt-Vente,…
- La personne chargée de l'enquête.
- Et la liste des offres en cours de traitement (attente).



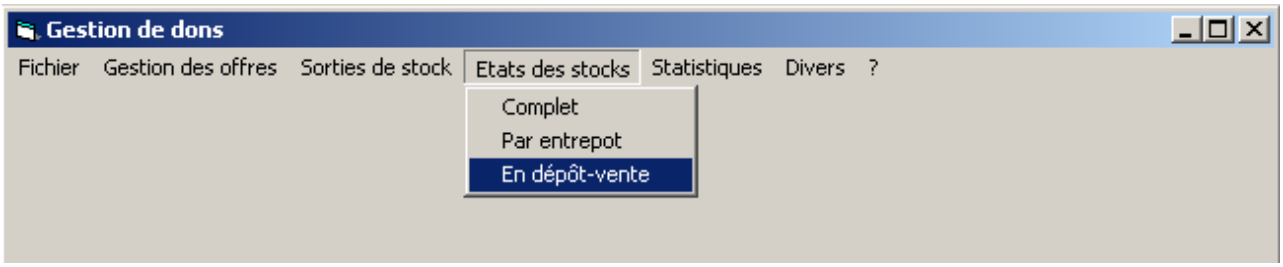
1.3.3. Menu Sorties de stock



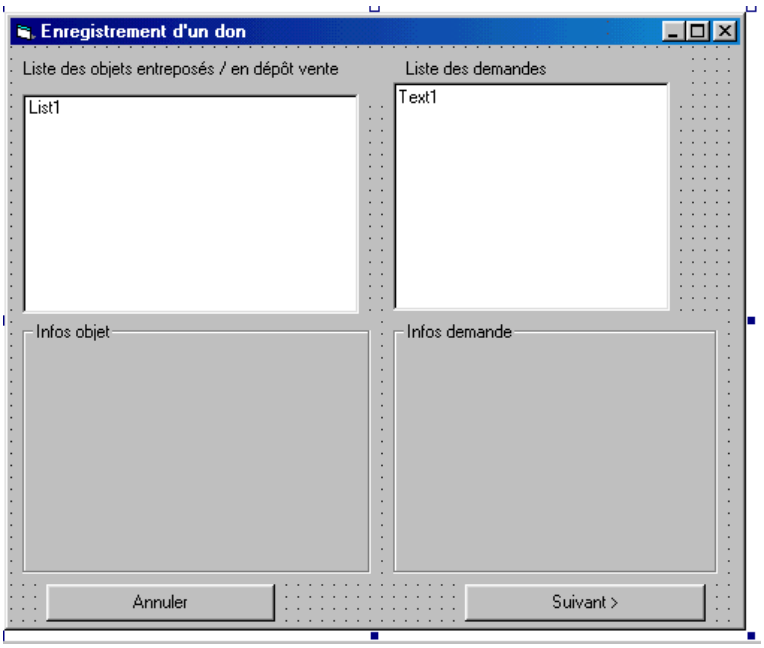
Ce menu permettra l'accés aux outils de gestions des stocks : placer les objets dans un entrepôt précis ou tout simplement les donner à une personne dans le besoin.

1.3.4. Module Etat des stocks

Ce menu donnera l'accès aux principales fonctions de visualisation des différents lieux de stockage.



Il nous semble judicieux de mettre en vis à vis la liste des offres et des demandes pour permettre une affectation rapide.



1.4. Gestion de la persistance et des recherches

L'utilisation d'une base de données, pour gérer la persistance des informations nous est apparue nécessaire pour plusieurs raisons :

- le volume de données à manipuler est important (1000 dons/an) ;
- en cas de corruption des informations, des outils existent pour réparer les bases de données ;
- la rapidité de la recherche est assurée (les algorithmes de BDD étant optimisés pour ça) ;
- la gestion des données est simplifiée grâce à l'utilisation du langage SQL ;
- et enfin le produit est plus facilement adaptable à l'évolution des besoins du client.

1.4.1. Module « Rechercher »

A. Rappel des besoins de ce module exprimés dans le cahier des charges

Ce module à pour but de permettre une réponse rapide face aux demandes qui arrivent à l'association.
Il faut pouvoir rechercher un objet par catégorie, par mots figurant dans la description et suivant les propriétés de l'objet : par exemple, une requête possible sera de chercher toutes les tables d'une largeur supérieure à 1m.
L'application doit lister les objets figurant dans les entrepôts ou dans les dépôts vente répondant à la requête de l'utilisateur.

B. Analyse du cahier des charges et définition de classe(s) adaptée(s)

L'utilisation d'un langage de requêtes standard nous permettra une plus grande flexibilité par rapport aux besoins de l'utilisateur ; un simple « SELECT Ref_objet FROM Meuble_literie WHERE largeur>100 » permet de répondre à la question posée.

1.4.2. Module « Statistiques »

A. Rappel des besoins de ce module exprimés dans le cahier des charges

Les informations statistiques pertinentes sont :

- Nombre de propositions de dons reçues
- Nombre de donateurs, de bénéficiaires
- Nombre de propositions de dons acceptées et ratio reçues/acceptées par catégorie d'objet
- Volume des ventes ventilé suivant les quatre destinations possibles
- Valeur des ventes ventilée suivant les quatre destinations possibles
- Principales catégories d'articles en stock
- Evolution des stocks

Tout ou partie de ces informations doivent pouvoir être imprimées, par un clic sur une icône impression.

B. Analyse du cahier des charges et définition de classe(s) adaptée(s)

La classe **Base_de_donnees** devra permettre l'envoi et le traitement par l'application de requêtes SQL. Les requêtes SQL seront générées par l'application en cours d'exécution, suivant les choix de l'utilisateur par l'intermédiaire de la méthode Requete.
Il n'y a dés lors que de la mise en forme des résultats à faire pour l'application.

1.4.3. Module « Fichier »

A. Rappel des besoins de ce module exprimés dans le cahier des charges

Ce module correspond aux fonctions classiques d'une application :
- La sauvegarde/restauration des données ;
- L'archivage et la purge des données ;
- Quitter l'application.

Mais également l'ajout et la suppression de dépôt-ventes.

B. Analyse du cahier des charges et définition de classe(s) adaptée(s)

Le changement de fichier de la base de données, une opération peu courante, pourra être définie au niveau des drivers ODBC (cf 2.1.1.). Le programme pourra prendre en charge la copie du fichier sur disquettes, par l'intermédiaire d'une méthode *Sauver*.
La classe **Base_de_donnees** devra permettre de changer le nom de la base de données utilisée.

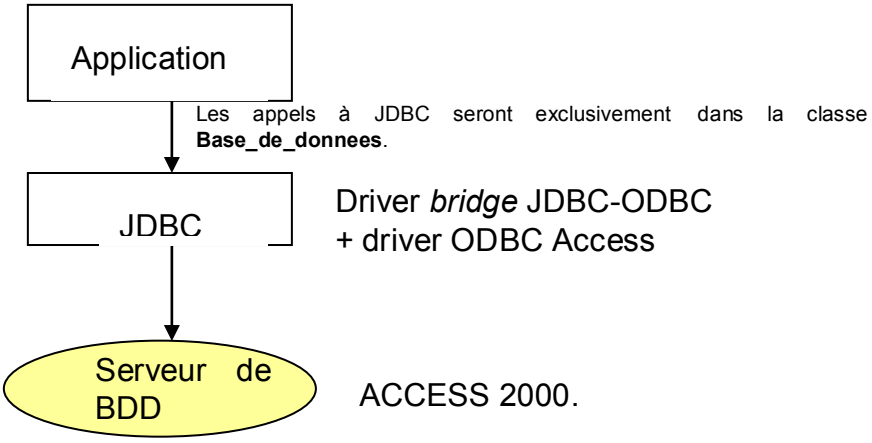
2. CONCEPTION

2.1. La base de données

2.1.1. Architecture globale

L'association possédant Microsoft OFFICE, on peut envisager d'utiliser Access pour la gestion de la base de données (les drivers ODBC étant livrés). MySql est une autre solution envisageable, mais qui requiert une installation sur le poste de l'utilisateur. Il nous semble préférable de limiter les démarches de l'utilisateur pour réinstaller l'application en cas de problème ; aussi nous optons pour ACCESS ; en cas de difficultés techniques en phase de réalisation, nous gardons MySql comme recours.

Nous optons globalement pour une architecture deux-tiers :



Cette architecture se justifie par le fait que nous voulons construire une application monoposte et qui n'est pas en ligne. En cas d'évolution ultérieure (multipostes) le modèle deux-tiers peut être maintenu : nous utiliserons d'ores et déjà des transactions pour les opérations pouvant présenter un risque pour l'intégrité des données.

2.1.2. Conception des tables

Nous avons fait des compromis entre le nombre de tables et la redondance des données, en choisissant de regrouper toutes les personnes dans une table unique.
Les objets, en grand nombre, seront eux stockés dans des tables correspondant à leur type. Les clefs primaires (Ref_objet) des tables des objets seront générés par l'application et non automatiquement incrémentés pour ne pas perdre l'unicité référence/objet.

Les tables sont présentés sur la page suivante.

Adimensionne
Ref_objet int
Type string
Description string
Montant_estime int
Est_livre bool
Date_enlevement date
Date_reponse date
Date_proposition date
Destinataire ref_personne
Bénéficiaire ref_personne
Vehicule string
Electroménager
Ref_objet int
Type string
Largeur int
Hauteur int
Longueur int
Description string
Montant_estime int
Est_livre bool
Date_enlevement date
Date_reponse date
Date_proposition date
Destinataire ref_personne
Bénéficiaire ref_personne
Vehicule string
Capacité string
Puissance string
Meuble_literie
Ref_objet int
Type string
Largeur int
Hauteur int
Longueur int
Description string
Montant_estime int
Est_livre bool
Date_enlevement date
Date_reponse date
Date_proposition date
Destinataire ref_personne
Bénéficiaire ref_personne
Vehicule string
Table
Ref_objet int
Type string
Largeur int
Hauteur int
Longueur int
Description string
Montant_estime int
Est_livre bool
Date_enlevement date
Date_reponse date
Date_proposition date
Destinataire ref_personne
Bénéficiaire ref_personne
Vehicule string
Forme string
Légende:
Clef primaire
Clef étrangère
Nom de table
Stock
Ref_objet int
Ref_entrepot int
Cuisinière
Ref_objet int
Type string
Largeur int
Hauteur int
Longueur int
Description string
Montant_estime int
Est_livre bool
Date_enlevement date
Date_reponse date
Date_proposition date
Destinataire ref_personne
Bénéficiaire ref_personne
Vehicule string
Nb_plaques int
Puissance string
Personne
Ref_personne int
Nom string
Adresse string
Telephone string
Notes string
Age int
Prenom string
Fonction string
Entrepot
Ref_entrepot int
Informations ref_personne

2.2. Détails des méthodes et commentaires

La partie Analyse de ce rapport nous a permis d'envisager l'application à l'aide de 15 classes interdépendantes (agrégation et héritage). Cette partie précise les diagrammes UML de chaque classe selon la norme suivante :

Nom de la Classe
Type1 variable d'instance 1 … Typep variable d'instance p
Signature Méthode1 … Signature Méthoden

D'autre part, nous indiquons les méthodes dans l'ordre suivant : d'abord les méthodes d'instances importantes et les constructeurs, puis les accesseurs.
Pour les arguments des méthodes, nous avons choisi d'indiquer un nom de variable explicite sans en préciser le type. Un autre schéma UML global à la fin de la partie conception de ce rapport indique lui uniquement le type des arguments de chaque méthode.

2.2.1. La classe Offre

Class Offre
Objet ObjetPropre Ref_personne Donnateur Ref_personne Enqueteur String DateReceptionOffre String DateReponse String DateEnlevement String Vehicule Ref_Entrepot ZoneStockage String DateDistribution Ref_personne DestinataireFinal Ref_personne BeneficiaireFinal

Void Creer(InitRefObjet, InitRefDonnateur, InitRefEnqueteur, InitDateReception) Void Refuser(InitDateReponse) Void Accepter(InitDateReponse, InitDateEnlevement, InitVehicule, InitZoneStockage) void Donner(InitDateDistribution, InitDestinataire, InitBeneficiaire) boolean EstLivre() void Etat() void ChangerEntrepot(NewRefEntrepot)
void SetDateEnlevement(NewDate) void SetDateReponse(NewDate) void SetDestinataireFinal(NewDestinataire) void SetEnqueteur(NewEnqueteur) void SetObjetPropre(NewObjet) void SetVehicule(NewVehicule)
Ref_personne GetBeneficaire() String GetDateEnlevement() String GetDateEnlevement() String GetDateReponse() Ref_personne GetDestinataireFinal() Ref_personne GetEnqueteur() Objet GetObjetPropre() String GetVehicule()

Commentaires sur les méthodes importantes :

- Créer permet de créer une offre de don (on ne sait pas encore si l'on va réellement acquérir l'objet) : c'est le constructeur. On initialise donc seulement les variables d'instance concernées : c'est l'étape qui correspond à la réception d'un appel téléphonique par l'association.
- Refuser entraîne la fin du traitement d'une offre de don : l'enquêteur a décidé que l'objet proposé ne pouvait pas être pris en charge par l'association. Accepter fait le contraire.
- Donner permet de donner une Offre acceptée à une personne (physique ou morale)

- EstLivre permet de savoir si une offre est disponible dans un entrepôt ou si elle est a été donnée à une personne
- Etat permet de savoir si l'Offre est Acceptée, Refusée ou En cours de Traitement

2.2.2. La classe Entrepot

class Entrepot
long Ref_entrepot Personne Information
boolean Possede(InitRefObjet) void Donner(InitRefObjet, InitRefBeneficiare, InitRefDestinataire, InitVehicule, InitRefLivreur, InitDateDistribution) void ListerStock() void Deplacer(InitRefOjet,NewRefEntrepot)
void SetEntrepot(NewPersonne)
String GetNom() String GetAdresse() String GetNumTel() long GetRef_entrepot()

Commentaires sur les méthodes importantes :

- Possede permet de savoir si un objet est dans un **Entrepot**
- Donner permet de donner un objet de l'**Entrepot** à une personne (physique ou morale)
- Lister permet d'avoir la liste des Objets présents dans l'**Entrepot**
- Deplacer permet de changer un objet d'**Entrepot**
  ***Ex** : GardeMeuble.Deplacer(203112, DepotVente) deplace le lit référencé 203112 du GardeMeuble vers le DepotVente*

2.2.3. La classe VenteBiannuelle

class VenteBiannuelle extends Entrepot
void Vendre(InitRefObjet, InitMontantObtenu, InitDateVente)

Commentaires sur la méthode

- Vendre permet de vendre un objet, qui n'est alors plus référencé dans le stock des objets présents dans l'Entrepot VenteBiannuelle

2.2.4. La classe DepotVente

class DepotVente extends Entrepot
void Vendre(InitRefObjet, InitMontantObtenu, InitDateVente)

2.2.5. La classe Personne

class Personne
long Ref_personne String Nom String Tel String Adresse String Note
Void SetNom(NewNom) Void SetRef_personne(NewRefPersonne) Void SetTel(NewTel) Void SetAdresse(NewAdresse) Void SetNote(NewNote)
String GetNom() long GetRef_personne() String GetTel() String GetAdresse() String GetNote()

2.2.6. La classe Tiers

class Tiers extends Personne
Int Age String Prénom
void SetPrenom(NewPrenom) void SetFonctionAdh(NewFonction) void SetAge(NewAge)
String GetPrenom() String GetFonctionAdh() Int GetAge()

2.2.7. La classe Adherent

class Adherent extends Personne
String Prenom String FonctionAdh Int Age
void SetPrenom(NewPrenom) void SetFonctionAdh(NewFonction) void SetAge(NewAge)
String GetPrenom() String GetFonctionAdh() Int GetAge()

2.2.8. La classe Objet

class Objet
long Ref_objet String DescriptionSuppl long LieuStockage long MontantEstime
void SetDescription(NewDescription) void SetLieuStockage(NewLieuStockage) void SetMontantEstime(NewMontant)
String GetDescription() long GetLieuStockage() long GetMontantEstime() long GetRef_objet()

2.2.9. La classe Dimensionne

class Dimensionne extends Objet
int longueur int largeur int hauteur
void SetLongueur(NewLongueur) void SetLargeur(NewLargeur) void SetHauteur(NewHauteur)
int GetLongueur() int GetLargeur() int GetHauteur()

2.2.10. La classe Adimensionne

class Adimensionne extends Objet
string type
void SetType(NewType)
void GetType()

2.2.11. La classe Table

class Table extends Dimensionne
String Type String Forme
void SetType(NewType) void SetForme(NewForme)

2.2.12. La classe Cuisiniere

class Cuisiniere extends Dimensione
String Type int NbPlaques int Puissance
void SetType(NewType) void SetNbPlaques(NewNb) void SetPuissance(NewPuissance)
String GetType() String GetNbPlaques() String GetPuissance()

2.2.13. La classe Meuble

class Meuble extends Dimensione
String Type
void SetType(NewType)
String GetType()

Cet objet sera également utilisé pour enregistrer des offres de literie.

2.2.14. La classe Electromenager

class Electromenager extends Dimensione
String Capacite String Puissance String Type
void SetCapacite(NewCapacite) void SetPuissance(NewPuissance) void SetType(NewType)
String GetCapacite() String GetPuissance() String GetType()

2.2.15. La classe Base_de_donnees

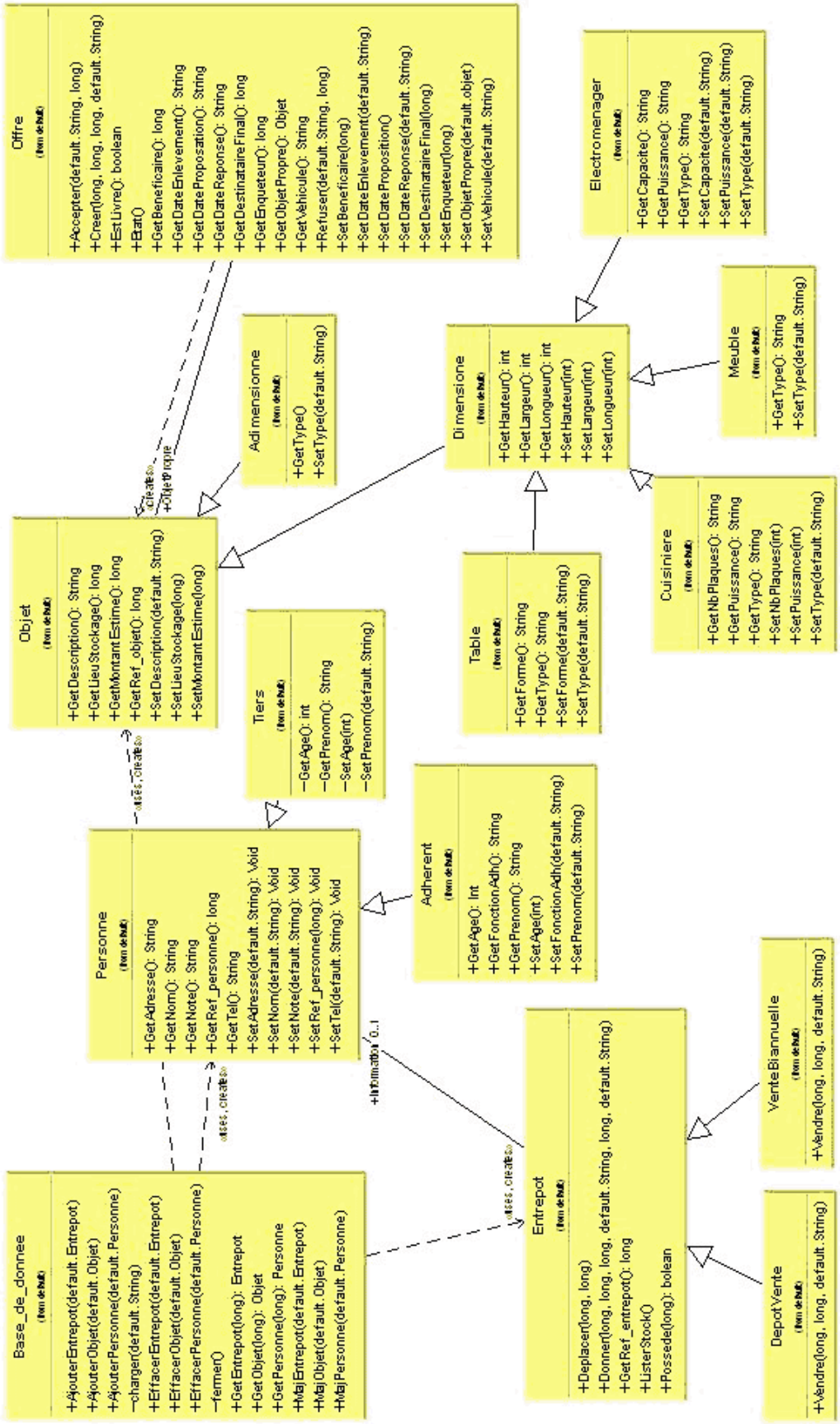
class Cuisiniere extends Dimensione
Void AjouterEntrepot(Entrepot) Void AjouterObjet(Objet) Void AjouterPersonne(Personne) Boolean Ouvrir(String NombaseODBC) Void EffacerEntrepot(Entrepot) Void EffacerObjet(Objet) Void EffacerPersonne(Personne) Void Fermer() Void MajEntrepot(Entrepot) Void MajObjet(Ojet) Void MajPersonne(Personne)

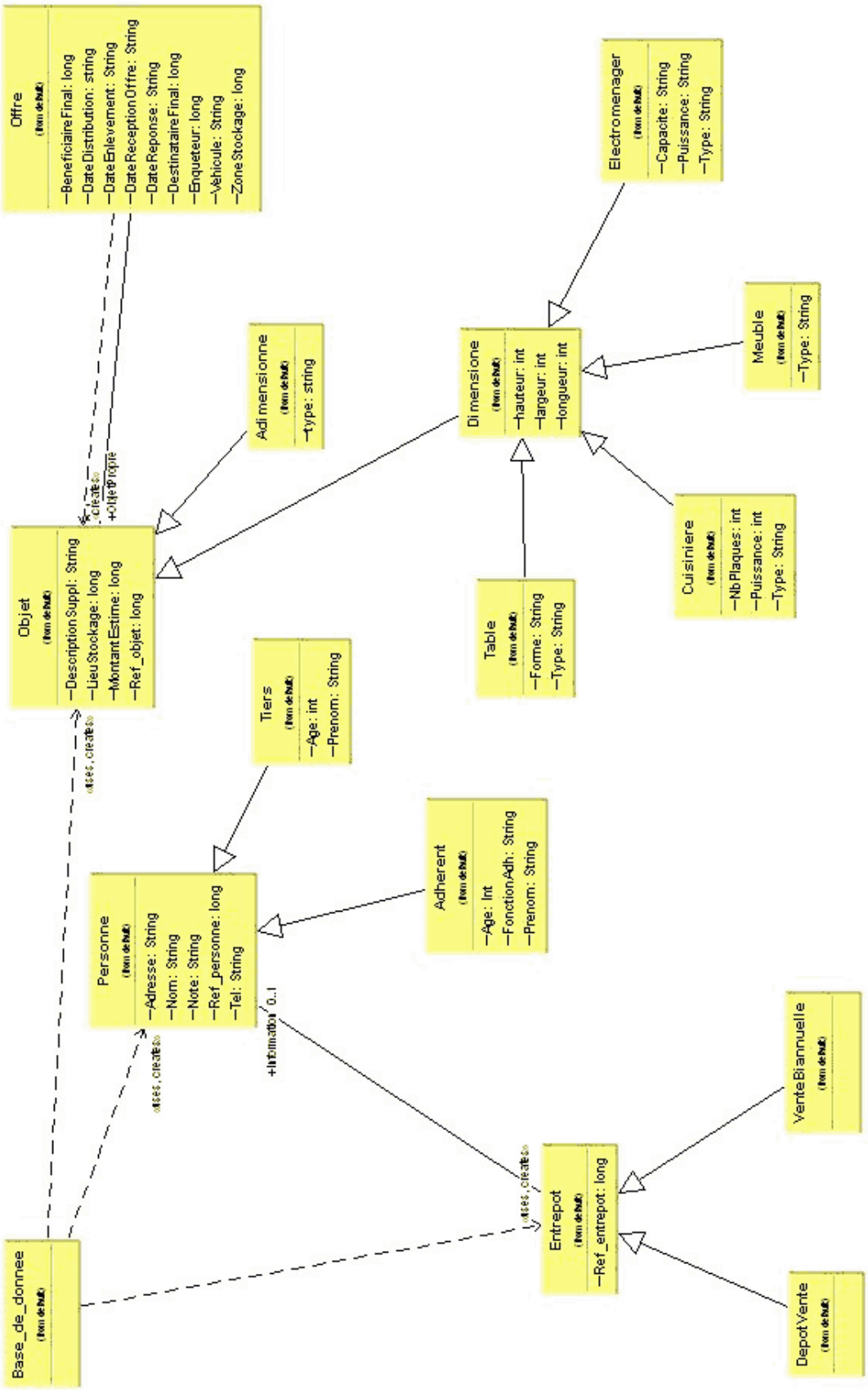
Entrepot GetEntrepot(Ref_Entrepot) Objet GetObjet(Ref_Objet) Personne GetPersonne(Ref_Personne)

Ces méthodes permettront au programme de mettre à jour, de supprimer ou de rajouter des informations dans la base de données.

Deux méthodes particulières, Ouvrir et Fermer seront appelées à l'initialisation et à la fermeture de l'application pour ouvrir l'accès à la base de données.

2.3. Résumés UML des méthodes et variables de classe.


Offre
+Accepter(default.String, long)
+Creer(long, long, long, default.String)
+EstLivre(): boolean
+Etat()
+GetBeneficiaire(): long
+GetDateEnlevement(): String
+GetDateProposation(): String
+GetDateReponse(): String
+GetDestinataireFinal(): long
+GetEnqueteur(): long
+GetObjetPropre(): Objet
+GetVehicule(): String
+Refuser(default.String, long)
+SetBeneficiaire(long)
+SetDateEnlevement(default.String)
+SetDateProposition()
+SetDateReponse(default.String)
+SetDestinataireFinal(long)
+SetEnqueteur(long)
+SetObjetPropre(default.objet)
+SetVehicule(default.String)
Electromenager
+GetCapacite(): String
+GetPuissance(): String
+GetType(): String
+SetCapacite(default.String)
+SetPuissance(default.String)
+SetType(default.String)
Adimensionne
+GetType()
+SetType(default.String)
Dimensione
+GetHauteur(): int
+GetLargeur(): int
+GetLongueur(): int
+SetHauteur(int)
+SetLargeur(int)
+SetLongueur(int)
Meuble
+GetType(): String
+SetType(default.String)
Objet
+GetDescription(): String
+GetLieuStockage(): long
+GetMontantEstime(): long
+GetRef_objet(): long
+SetDescription(default.String)
+SetLieuStockage(long)
+SetMontantEstime(long)
Cuisiniere
+GetNbPlaques(): String
+GetPuissance(): String
+GetType(): String
+SetNbPlaques(int)
+SetPuissance(int)
+SetType(default.String)
Table
+GetForme(): String
+GetType(): String
+SetForme(default.String)
+SetType(default.String)
Tiers
-GetAge(): int
-GetPrenom(): String
-SetAge(int)
-SetPrenom(default.String)
Adherent
+GetAge(): Int
+GetFonctionAdh(): String
+GetPrenom(): String
+SetAge(int)
+SetFonctionAdh(default.String)
+SetPrenom(default.String)
Personne
+GetAdresse(): String
+GetNom(): String
+GetNote(): String
+GetRef_personne(): long
+GetTel(): String
+SetAdresse(default.String): Void
+SetNom(default.String): Void
+SetNote(default.String): Void
+SetRef_personne(long): Void
+SetTel(default.String): Void
Base_de_donnee
+AjouterEntrepot(default.Entrepot)
+AjouterObjet(default.Objet)
+AjouterPersonne(default.Personne)
-charger(default.String)
+EffacerEntrepot(default.Entrepot)
+EffacerObjet(default.Objet)
+EffacerPersonne(default.Personne)
-fermer()
+GetEntrepot(long): Entrepot
+GetObjet(long): Objet
+GetPersonne(long): Personne
+MajEntrepot(default.Entrepot)
+MajObjet(default.Objet)
+MajPersonne(default.Personne)
Entrepot
+Deplacer(long, long)
+Donner(long, long, long, default.String, long, default.String)
+GetRef_entrepot(): long
+ListerStock()
+Possede(long): bolean
VenteBiannuelle
+Vendre(long, long, default.String)
DepotVente
+Vendre(long, long, default.String)
create
+Objet*
+Information
0..1


Offre
-BeneficiaireFinal: long
-DateDistribution: string
-DateEnlevement: String
-DateReceptionOffre: String
-DateReponse: String
-DestinataireFinal: long
-Enqueteur: long
-Vehicule: String
-ZoneStockage: long
Electromenager
-Capacite: String
-Puissance: String
-Type: String
Adimensionne
-type: string
Dimensione
-hauteur: int
-largeur: int
-longueur: int
Meuble
-Type: String
Objet
-DescriptionSuppl: String
-LieuStockage: long
-MontantEstime: long
-Ref_objet: long
Cuisiniere
-NbPlaques: int
-Puissance: int
-Type: String
Table
-Forme: String
-Type: String
Tiers
-Age: int
-Prenom: String
Adherent
-Age: Int
-FonctionAdh: String
-Prenom: String
Personne
-Adresse: String
-Nom: String
-Note: String
-Ref_personne: long
-Tel: String
VenteBiannuelle
Entrepot
-Ref_entrepot: long
Base_de_donnee
DepotVente