

**2011**

Cindy Joseph & Guillaume Leroy

Polytech’Montpellier pour Neorizon représenté par M. Baïla

26/05/2011

Développement d’une application de gestion des titres restaurant



Rapport technique

Remerciements

Nous tenons à remercier toutes les personnes qui nous ont permis de mener à bien ce projet pendant ces deux mois :

* M. Sébastian Baïla, notre demandeur, pour sa disponibilité et son aide au bon déroulement du projet
* M. Jacques Ruiz, notre responsable pédagogique, qui nous a suivis régulièrement et conseillés tout au long de ce projet
* L’administration de Polytech’ Montpellier pour nous avoir fourni le matériel nécessaire à la bonne réalisation de ce projet.
* M. Marc Dumas, spécialiste de la communication, pour nous avoir aidés à préparer notre soutenance.

Sommaire

[1 Introduction 6](#_Toc294113827)

[2 Présentation du projet 7](#_Toc294113828)

[2.1 Le problème de gestion 7](#_Toc294113829)

[2.2 Les besoins fonctionnels 7](#_Toc294113830)

[2.3 La mission 7](#_Toc294113831)

[2.4 Contraintes techniques 8](#_Toc294113832)

[3 Déroulement du projet 9](#_Toc294113833)

[3.1.1 Méthodes utilisées 9](#_Toc294113834)

[3.1.2 Choix technologiques 9](#_Toc294113835)

[4 Travail Réalisé 10](#_Toc294113836)

[4.1 Présentation du domaine 10](#_Toc294113837)

[4.2 Modèle de fonctionnement global (MCC) 10](#_Toc294113838)

[4.3 Description des données 11](#_Toc294113839)

[4.3.1 Inventaire des documents 11](#_Toc294113840)

[4.3.2 Inventaire des rubriques 12](#_Toc294113841)

[4.4 Dictionnaire de données 13](#_Toc294113842)

[4.5 Analyse préalable 15](#_Toc294113843)

[4.5.1 Recueil des données 15](#_Toc294113844)

[4.5.2 Modèle conceptuel de traitements 16](#_Toc294113845)

[4.6 Conception 17](#_Toc294113846)

[4.6.1 Diagramme états-transitions 17](#_Toc294113847)

[4.6.2 Diagramme d’activité 18](#_Toc294113848)

[4.6.3 Diagramme de classes 20](#_Toc294113849)

[4.6.4 Architecture 22](#_Toc294113850)

[4.7 Conception détaillée du logiciel 23](#_Toc294113851)

[4.7.1 Diagramme de classes logiciel 23](#_Toc294113852)

[4.7.2 Schéma de la base de données 24](#_Toc294113853)

[4.7.3 Cas d’utilisation 25](#_Toc294113854)

[4.7.4 Hiérarchie des fonctionnalités 26](#_Toc294113855)

[4.7.5 Gestion des titres restaurant 27](#_Toc294113856)

[4.7.6 Gestion des bordereaux 35](#_Toc294113857)

[4.7.7 Diagramme de séquence global 36](#_Toc294113858)

[4.7.8 Gestion des établissements 46](#_Toc294113859)

[4.7.9 Statistiques 53](#_Toc294113860)

[4.8 Développement 56](#_Toc294113861)

[4.8.1 Fenêtre principale 56](#_Toc294113862)

[4.8.2 Schéma d’enchaînement des vues 57](#_Toc294113863)

[4.8.3 Ensemble des classes 58](#_Toc294113864)

[4.8.4 Connexion à la base de données 59](#_Toc294113865)

[4.8.5 Initialisation de la base de données 59](#_Toc294113866)

[4.8.6 Échanges avec la base de données 60](#_Toc294113867)

[4.8.8 Lecture des codes-barres 62](#_Toc294113868)

[4.8.9 Gestion des établissements 65](#_Toc294113869)

[4.8.10 Gestion des bordereaux 70](#_Toc294113870)

[4.8.11 Statistiques 83](#_Toc294113871)

[4.9 Tests 86](#_Toc294113872)

[4.10 Limites 86](#_Toc294113873)

[4.11 Améliorations possibles 87](#_Toc294113874)

[4.11.1 Impression d’un bordereau complet 87](#_Toc294113875)

[4.11.2 Sauvegarde de la BD 87](#_Toc294113876)

[4.11.3 Design des fenêtres 87](#_Toc294113877)

[5 Conclusion 88](#_Toc294113878)

[6 Annexes 89](#_Toc294113879)

[Résumé 90](#_Toc294113880)

[Summary 90](#_Toc294113881)

Table des illustrations

[Image 3.1.2.1 : Modèle de fonctionnement global 10](#_Toc294113782)

[Image 4.5.2.1 : Modèle Conceptuel des Traitements 16](#_Toc294113783)

[Image 4.6.1.1 : Diagramme états-transitions 17](#_Toc294113784)

[Image 4.6.2.1 : Diagramme d’activité 18](#_Toc294113785)

[Image 4.6.3.1 : Diagramme de classes 20](#_Toc294113786)

[Image 4.6.4.1 : Schéma de l'architecture 22](#_Toc294113787)

[Image 4.7.1.1 : Diagramme de classes logiciel 23](#_Toc294113788)

[Image 4.7.3.1 : Use Case global 25](#_Toc294113789)

[Image 4.7.4.1 : Schéma hiérarchique des fonctionnalités 26](#_Toc294113790)

[Image 4.7.5.1 : Use Case Gestion des Titres 27](#_Toc294113791)

[Image 4.7.5.2 : Maquette Scan des titres 29](#_Toc294113792)

[Image 4.7.5.3 : Diagramme de séquence Insertion de Titres Restaurant 30](#_Toc294113793)

[Image 4.7.5.4 : Maquette Consultation de Titres Restaurant 32](#_Toc294113794)

[Image 4.7.5.5 : Diagramme de séquence Consultation des Titres Restaurant 32](#_Toc294113795)

[Image 4.7.5.6 : Maquette Suppression de Titres Restaurant 34](#_Toc294113796)

[Image 4.7.5.7 : Diagramme de séquence Suppression de Titres Restaurant 34](#_Toc294113797)

[Image 4.7.6.1 : use case Gestion des bordereaux 35](#_Toc294113798)

[Image 4.7.7.1 : Diagramme de Séquence Gestion des bordereaux 36](#_Toc294113799)

[Image 4.7.7.2 : Maquette Liste des Bordereaux 37](#_Toc294113800)

[Image 4.7.7.3 : Maquette Détails d’un Bordereaux 38](#_Toc294113801)

[Image 4.7.7.4 : Maquette Liste des Bordereaux 40](#_Toc294113802)

[Image 4.7.7.5 : Maquette Détails d'un Bordereaux 41](#_Toc294113803)

[Image 4.7.7.6 : Maquette Détails d'un bordereau 43](#_Toc294113804)

[Image 4.7.7.7 : Maquette Liste Bordereaux 45](#_Toc294113805)

[Image 4.7.8.1 : Maquette Informations Établissement 47](#_Toc294113806)

[Image 4.7.8.2 : Maquette Informations Gérant 47](#_Toc294113807)

[Image 4.7.8.3 : Maquette Modification Assurance 48](#_Toc294113808)

[Image 4.7.8.4 : Diagramme de Séquence Consultation et modification d’un Etablissement 49](#_Toc294113809)

[Image 4.7.8.5 : Maquette Informations Établissement 51](#_Toc294113810)

[Image 4.7.8.6 : Maquette Ajout d'un Établissement 51](#_Toc294113811)

[Image 4.7.8.7 : Maquette Informations Gérant 52](#_Toc294113812)

[Image 4.7.8.8 : Diagramme de Séquence Ajout d'un Établissement 52](#_Toc294113813)

[Image 4.7.9.1 : Maquette Statistiques 54](#_Toc294113814)

[Image 4.7.9.2 : Diagramme de Séquence Statistiques 55](#_Toc294113815)

[Image 4.8.2.1 : Schéma d’enchaînement des vues 57](#_Toc294113816)

[Image 4.8.2.2 Schéma de dépendance de la fenêtre « FormListBord » 58](#_Toc294113817)

[Image 4.8.2.3 : Schéma de dépendance de la fenêtre « FormChoixAjoutTitre » 58](#_Toc294113818)

[Image 4.8.8.1 : Fenêtre du clavier tactile 63](#_Toc294113819)

[Image 4.8.9.1 : Vue Etablissement 65](#_Toc294113820)

[Image 4.8.10.1 : Vue Bordereau 71](#_Toc294113821)

[Image 4.8.10.2 : Fenêtre ajout bordereau 72](#_Toc294113822)

[Image 4.8.10.3 : Fenêtre choix d’ajout de titres 77](#_Toc294113823)

[Image 4.8.10.4 : Importation des titres de NeoResto 78](#_Toc294113824)

[Image 4.8.10.5 : Fenêtre Edition du remboursement 81](#_Toc294113825)

[Image 4.8.11.1 : Vue Statistiques 83](#_Toc294113826)

# Introduction

Dans le cadre de notre formation d’ingénieur en informatique et gestion, nous devons réaliser à la fin de la quatrième année un projet industriel en collaboration avec une entreprise. Ce projet a pour but de nous placer dans les conditions du monde professionnel.

Dans cette optique, nous avons choisi de réaliser le projet proposé par l’entreprise Neorizon. Notre choix s’est porté sur ce projet, car il nous permettait de réaliser un projet dans son intégralité. L’objectif général était de concevoir une solution permettant de gérer les titres restaurant qui étaient jusque-là traités manuellement.

Ce rapport vise à présenter de manière synoptique le travail accompli durant ce projet. Dans un premier temps nous présenterons le contexte général dans lequel nous avons évolué. Puis nous aborderons les objectifs attendus et le travail réalisé pour les atteindre. Nous décrirons ensuite la démarche suivie et les problèmes que nous avons rencontrés. Enfin, nous ferons un bilan sur le travail que nous avons effectué.

# Présentation du projet

## Le problème de gestion

Actuellement il n’y a pas d’outil approprié pour gérer les différents types de titres restaurant. En effet, pour se faire rembourser les titres restaurant perçus, le restaurateur doit remplir un bordereau qu’il envoie à la CRT. Pour remplir ce bordereau, il doit donc trier, compter et classer les titres perçus manuellement afin de pouvoir être remboursé, ce qui peut être source d’erreurs et nécessite beaucoup de temps. La gestion de ces titres est donc aujourd’hui une réelle perte de temps pour la profession.

## Les besoins fonctionnels

L’objectif principal de ce projet était donc de créer une application permettant aux restaurateurs d'automatiser et de simplifier cette procédure de gestion des titres.

La solution que nous devions réaliser devait atteindre ces objectifs :

* Élaborer les bordereaux pour le remboursement de façon semi-automatique (en utilisant les informations saisies par l’utilisateur) et les imprimer
* Gérer les bordereaux : consultation, modification de leur état, archivage
* Présenter des statistiques simples concernant les données enregistrées à propos des titres

## La mission

Notre mission était donc d’analyser, de concevoir et de développer une application qui permettrait d’atteindre les objectifs cités dans le paragraphe précédent.

L’application devrait offrir les fonctionnalités suivantes :

* Saisie des titres restaurant à l’aide d’une douchette ou en saisissant le code à la main
* Elaboration de bordereaux pour le remboursement des titres restaurant
* Consultation et édition des bordereaux
* Gestion de l’état des bordereaux (En cours, imprimé, envoyé ou remboursé)
* Impression des bordereaux selon les standards de la CRT
* Historique des bordereaux et des titres
* Statistiques sur les titres restaurant (chiffre d’affaire, quantité, répartition), les bordereaux (suivi de l’état, écarts de remboursement)
* Paramétrage de l’application (ajout, modification ou suppression des établissements)

## Contraintes techniques

Pour réaliser ce projet, il était demandé d’utiliser le Framework[[1]](#footnote-1) .NET dans sa version 2.0 avec Visual Studio 2008 et le langage de programmation objet C#, afin de garder une cohérence avec NeoResto.

Le logiciel est destiné à être utilisé sur des écrans tactiles, donc sans utilisation d’une souris. Il faut donc adapter l’interface à ce type d’utilisation, en fournissant des éléments graphiques plus gros, plus accessibles.

# Déroulement du projet

### Méthodes utilisées

Nous devions dans un premier temps réaliser l’analyse du système lié au titre restaurant afin de réaliser la base de données relationnelle qui allait contenir toutes les informations nécessaires à l’élaboration de bordereaux. Afin de réaliser cette étude nous avons décidé d’utiliser certains éléments de la méthode Merise car elle contient tous les éléments dont nous avions besoin afin d’aboutir à un bon résultat.

Afin de modéliser notre application nous avons utilisé l’Unified Modelling Language (UML) car le langage de programmation imposé par le demandeur C# est orienté objet.

Pendant la phase de développement nous avons organisé plusieurs entretiens avec le demandeur afin de lui montrer l’avancée de notre travail. Ces rendez-vous réguliers nous ont permis d’être assez réactifs sur les modifications à apporter, notamment au niveau des interfaces utilisateurs qui ont beaucoup évoluées entre l’étape de maquettage et le développement.

### Choix technologiques

Si le langage de programmation et le Framework nous ont été imposés nous avons été chargés par le demandeur de choisir un système de gestion de base de données (SGBD). Pour les besoins de l’application à réaliser nous devions choisir un outil offrant les mêmes possibilités qu’un SGBD ordinaire, mais disponible dans une version embarquée. Nous nous sommes intéressés à plusieurs technologies et nous en avons retenues trois : PostGreSQL, Firebird et SQLite. Nous avons choisi Firebird car ce SGBD était disponible sous une version embarquée et en même temps offrait une plus grande souplesse que PostGreSQL. SQLite, bien que compacte, n’offre que des fonctionnalités minimales et il ne manque donc d’importantes, telle l’intégrité référentielle[[2]](#footnote-2).

Nous avons utilisé la bibliothèque ZedGraph afin de dessiner les diagrammes nécessaires pour la partie statistiques.

# Travail Réalisé

## Présentation du domaine

Le domaine étudié n’est pas une organisation en particulier, mais un type d’organisation ayant des règles de fonctionnement communes. En effet, tous les bars, restaurant et brasserie sont articulés autour de plusieurs parties (que l’on pourrait comparer à des services) :

* La salle, où sont accueillis les clients et où les serveurs prennent les commandes
* La caisse, établissant la facturation pour les clients
* La cuisine, répondant aux commandes des clients indiqués par les serveurs
* Le bureau, dans lequel le gérant établi notamment les bordereaux de remboursement des titres restaurant

C’est ce dernier point qui nous intéresse tout particulièrement dans le cadre de ce projet. Le gérant rassemble les titres perçus en liasse par période (de son choix) et établi un bordereau qu’il envoie à la CRT pour être remboursé.

En salle, les serveurs peuvent également scanner les titres reçus dans un logiciel du type NeoResto.

## Modèle de fonctionnement global (MCC)

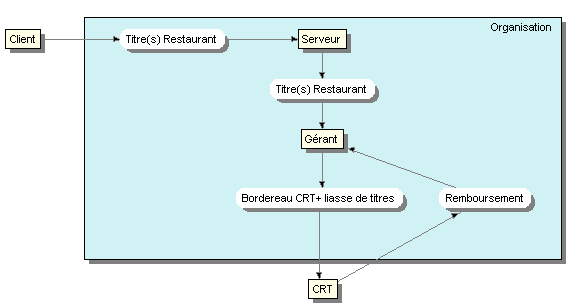


Image . : Modèle de fonctionnement global

Le modèle de fonctionnement global permet de représenter les flux d’information s’échangeant entre les différents acteurs du système étudié.

Le domaine étudié se restreint à la gestion des titres restaurant.

Le client s’acquitte de sa facture en payant avec des titres restaurant. Le serveur enregistre ces titres à partir de la caisse, puis le gérant les comptabilise et les enregistre sur un bordereau qu’il envoie, accompagné de la liasse de titres qu’il souhaite se faire rembourser, à la CRT (Centrale de Règlement des Titres). Après vérification, la CRT rembourse l’établissement concerné.

## Description des données

### Inventaire des documents

* Titres restaurant
  + Chèque de Table
  + Chèque Déjeuner
  + Ticket Restaurant
  + Chèque Restaurant
* Bordereaux de la CRT

### Inventaire des rubriques

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Rubriques | Chèque de table | Chèque Déjeuner | Ticket Restaurant | Chèque Restaurant | Bordereau de la CRT |
| Valeur faciale | X | X | X | X |  |
| Code sécurité | X | X | X | X |  |
| Code Barre | X | X | X | X |  |
| Nom de l'entreprise du bénéficiaire | X | X | X |  |  |
| Adresse de l'entreprise du bénéficiaire |  |  |  | X |  |
| Date limite de validité | X | X | X |  |  |
| Année du titre | X | X |  | X |  |
| Ligne OCR | X | X | X | X |  |
| Date de début de validité |  |  | X |  |  |
| Nom du bénéficiaire | X | X | X | X |  |
| Adresse du bénéficiaire |  |  | X |  |  |
| Fonction du bénéficiaire | X |  |  | X |  |
| Pays de validité | X |  |  | X |  |
| Numéro de titre | X | X |  |  |  |
| Type du titre | X | X | X | X |  |
| Nationalité du titre (drapeau) | X | X |  | X |  |
| Montant total |  |  |  |  | X |
| Quantité de titres |  |  |  |  | X |
| Numéro de remise |  |  |  |  | X |
| Date de remise |  |  |  |  | X |
| Commande 5 enveloppes CRT |  |  |  |  | X |
| Code affilié |  |  |  |  | X |
| Numéro Chèque Déjeuner /Services |  |  |  |  | X |
| Numéro Chèque de Table/Interservices |  |  |  |  | X |
| Numéro Chèque Restaurant/Multi-Services |  |  |  |  | X |
| Numéro Ticket Restaurant/Services |  |  |  |  | X |
| Nom de l'établissement |  |  |  |  | X |

## 

## Dictionnaire de données

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Nom symbolique** | **Désignation** | **Type** | **Taille** | **Contrôles** | **Remarques** |
| tot | Montant total | numérique | illimitée,2 | non nul |  |
| nbTitre | Quantité de titres | numérique | illimitée | non nul |  |
| numRemise | Numéro de remise | alphanumérique | 9 | non nul |  |
| dateRemise | Date de remise | date | JJMMAAAA | non nul |  |
| com | Commande 5 enveloppes CRT | booléen | 1 | non nul |  |
| codeAfilié | Code affilié | alphanumérique | 9 | non nul | étoile avant le dernier caractère |
| numCD | Numéro Chèque Déjeuner /Services | numérique | 9 | non nul |  |
| numCT | Numéro Chèque de Table/Interservices | numérique | 9 | non nul |  |
| numCR | Numéro Chèque Restaurant/Multi-Services | numérique | 9 | non nul |  |
| numTR | Numéro Ticket Restaurant/Services | numérique | 9 | non nul |  |
| nomEtablissement | Nom de l'établissement | alphanumérique | illimité | non nul |  |
| valeurFaciale | Valeur faciale du chèque | Numérique | 3,2 | Strictement supérieur à 0 |  |
| codeSecu | Code de sécurité | Alphanumérique | Variable |  |  |
| numCodeBarre | Code barre | Numérique | 24 | 1-9 : numéro de titre, 10-11 : clé de cryptage, 12-16 : Valeur faciale, 17 : émetteur, 18-19 : clé de contrôle = (1:9+10:11+12:16+17+20:22+23+24)mod97, 20-22 : code famille, 23 : Produit, 24 : Millésime | 24 digits pour le millésime 2011 |
| nomEnt | Nom de l'entreprise du bénéficiaire | Alphanumérique | Illimité |  |  |
| adrEnt | Adresse del'entreprise du bénéficiaire | Alphanumérique | Illimité |  |  |
| dateLimite | Date limite de validité | Date | JJ MM AAAA | dateLimite > dateDebutValidite |  |
| annee | Année du titre | Numérique | 4 |  |  |
| OCR | Ligne OCR | Numérique | 32 |  | Cette chaîne peut être séparée par des symboles < ou > (exactement 4) |
| dateDebutValidite | Date de début de validité | Date | JJ MM AAAA | dateLimite > dateDebutValidite |  |
| nomBeneficiaire | Nom du bénéficiaire | Alphanumérique |  |  |  |
| adrBeneficiaire | Adresse du bénéficiaire | Alphanumérique |  |  |  |
| fonctionBeneficiaire | Fonction du bénéficiaire | Alphanumérique |  |  |  |
| pays | Pays de validité | Alphanumérique |  |  |  |
| numTitre | Numéro de titre | Numérique |  |  |  |
| typeTitre | Type du titre | Alphanumérique |  | Chèque déjeuner, Chèque de table, Chèque restaurant ou Ticket restaurant |  |
| nat | Nationalité du titre (drapeau) | Image |  |  | Le drapeau du pays d'origine du titre |

## Analyse préalable

Nous avons utilisé quelques éléments de la méthode Merise dans la phase d’analyse.

### Recueil des données

À partir des informations disponibles sur les titres restaurant et les bordereaux de la CRT, nous avons établi le modèle conceptuel de données suivant.

Chaque établissement est enregistré auprès de la CRT par un code affilié, il s’agit de leur identifiant. Chaque établissement (restaurant) peut recevoir des titres restaurant en guise de paiement. Un titre restaurant est identifié par son code-barres. Il contient d’autres informations comme sa valeur faciale et sa date limite. Il existe quatre types de titre : les chèques déjeuner, les chèques de table, les tickets restaurant et les chèques de table.

Lorsqu’un établissement demande le remboursement des titres qu’il a perçu, il envoie à la CRT un bordereau accompagné d’une liasse constituée des titres qu’il souhaite se faire rembourser. Le bordereau contient les informations suivantes : le numéro de remise, la date de remise (date à laquelle le bordereau est envoyé), le montant total du bordereau, le nombre de titres pour chaque type et les neufs derniers caractères du code-barres d’un titre tiré au hasard dans la liasse par type de titre.

### Modèle conceptuel de traitements

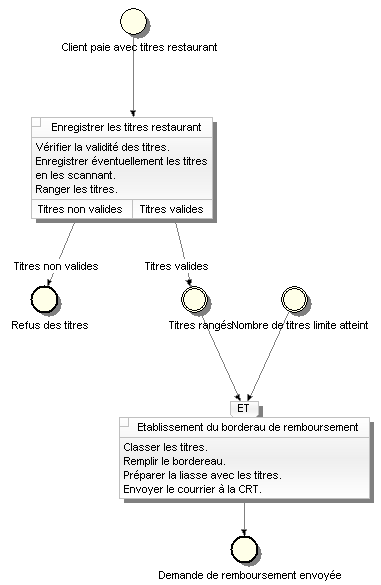


Image . : Modèle Conceptuel des Traitements

Lorsqu’un client paie avec un ou plusieurs titres restaurant, l’employé qui l’encaisse doit vérifier que le titre est valide (par exemple la date limite de validité). S’il a la possibilité de le faire il scanne les titres afin que les données soient enregistrées dans le logiciel de gestion de l’établissement (NeoResto par exemple). Enfin il range les titres.

Lorsqu’un certain nombre de titre est atteint, un employé se charge de remplir le bordereau de remboursement des titres. Il doit classer les titres par type afin de préparer la liasse qui sera envoyée avec le bordereau. Il complète le bordereau pré-rempli. Enfin il envoie la liasse et le bordereau à la CRT.

## Conception

### Diagramme états-transitions

Le diagramme d’états-transitions permet de décrire les changements d’états d’un objet ou d’un composant en réponse aux interactions avec d’autres objets ou composants ou d’acteurs.

Ici, on s’intéresse uniquement au diagramme concernant le bordereau.

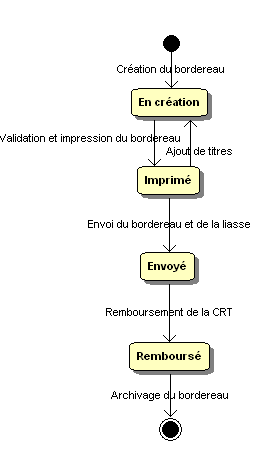


Image . : Diagramme états-transitions

Lorsqu’un bordereau est créé son état est par défaut en création. Une fois le bordereau terminé, l’utilisateur peut imprimer le bordereau. L’état du bordereau se transforme donc automatiquement en « Imprimé ». Si l’utilisateur veut ajouter des titres au bordereau qu’il vient d’imprimer cela reste possible et le bordereau repasse à l’état en « En Création ». Quand l’utilisateur a envoyé le bordereau à la CRT il doit indiquer dans l’application qu’il a envoyé le bordereau afin que l’état du bordereau passe à « Envoyé ». Aucun retour en arrière n’est alors possible. Enfin lorsqu’il reçoit le remboursement de la CRT il doit également l’indiquer dans l’application.

### Diagramme d’activité

Le diagramme d’activité est une vue dynamique d’UML qui permet de représenter le déroulement d’un cas d’utilisation.  
Ici le diagramme d’activité décrit les Use Cases « Gestion des bordereaux » et « Gestion des états des bordereaux » décrit respectivement dans les paragraphes 1.3 et 1.4.

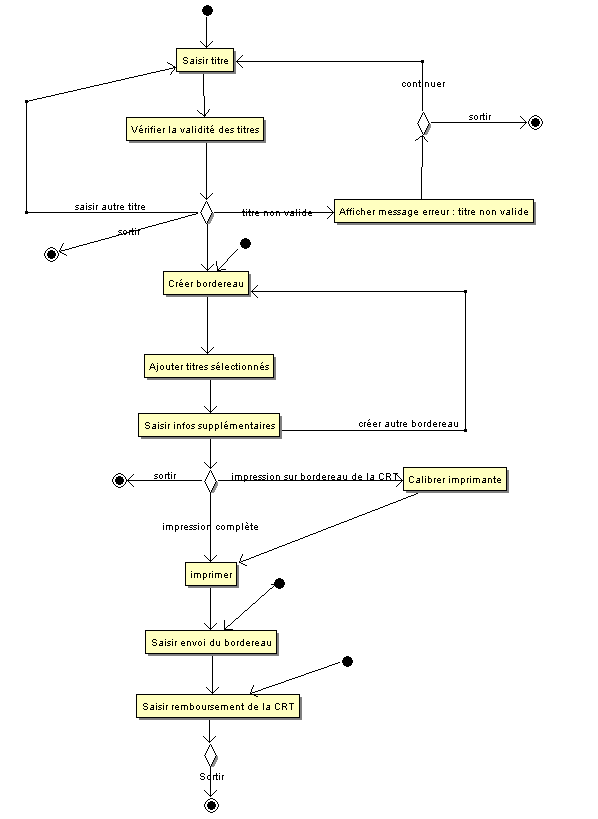


Image . : Diagramme d’activité

L’utilisateur va saisir un titre dans l’application à l’aide d’une douchette. Les informations contenues dans le code-barres vont être vérifiées (la date de validité par exemple). Si les titres ne sont pas valides, un message d’erreur sera envoyé à l’utilisateur. À ce moment l’utilisateur peut quitter l’application ou réessayer de scanner de nouveaux titres.

Dès que l’utilisateur le décide, il peut créer un bordereau de remboursement en ajoutant les titres enregistrés qui n’ont pas encore été remboursés par la CRT ou en les scannant directement. Il peut également saisir dans l’application des infos supplémentaires comme par exemple le montant à ne pas dépasser pour le bordereau en création. Ensuite l’utilisateur peut imprimer de deux façons son bordereau : soit il imprime le bordereau en entier sur une feuille vierge, soit il l’imprime sur le bordereau pré-imprimé de la CRT ce qui nécessite un calibrage de l’imprimante.

L’utilisateur peut également quand il le souhaite et quand cela est nécessaire notifier que le bordereau a été envoyé et que le remboursement a été effectué.

### Diagramme de classes

Le diagramme de classe est utilisé en UML pour représenter l’organisation des données dans un système d’information.

Après notre étude de l’existant, nous avons pu déterminer de quelles informations nous avions réellement besoin pour le bon fonctionnement de notre application.

Par exemple, pour les titres restaurant, toutes les informations nécessaires comme la valeur faciale, le type de titre (l’émetteur) se trouvent codés dans le code-barres. Les autres informations comme le nom du bénéficiaire et son adresse sont superflues.

Afin d’établir le bordereau nous n’avons besoin que du code affilié de l’établissement, les autres informations concernant l’établissement et le gérant ne sont présentes qu’à titre informatif.

Concernant le bordereau, nous avons besoin d’avoir des informations supplémentaires qui ne sont pas mentionnées sur le document comme par exemple date de remboursement, le montant remboursé et la raison de l’écart de remboursement s’il y en a un, afin d’assurer la partie statistiques de l’application.

En apportant ces modifications nous sommes arrivés au diagramme de classes suivant.

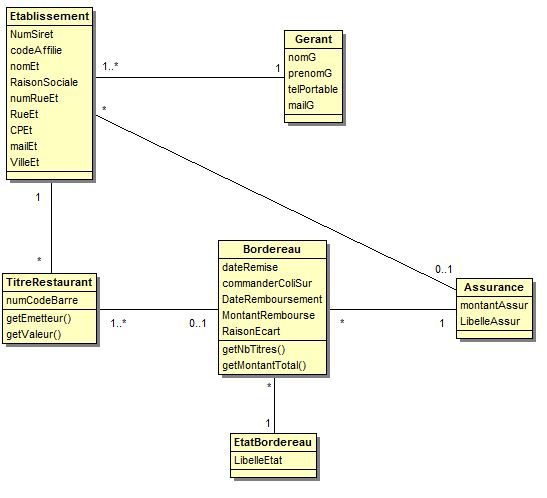


Image . : Diagramme de classes

Liens entre les informations

La classe « TitreRestaurant » n’a qu’un attribut qui est le code-barres présent sur les titres restaurants. Ce code-barres contient toutes les informations dont on a besoin concernant le titre, il est de type 2 parmi 5 entrelacé (ou 2/5 entrelacé) et il est composé de 24 caractères numériques.

Les 9 premiers chiffres représentent le numéro du titre. Les 2 suivants constituent la clé de cryptage. Les cinq chiffres suivants représentent la valeur faciale du titre en centimes. Le chiffre suivant indique l’identité de l’émetteur du titre. Les deux chiffres suivants représentent la clé de contrôle. Les 3 chiffres suivants correspondent au code famille d’utilisation. Le chiffre suivant représente le code produit. Le dernier chiffre représente le millésime

Pour rappel, les quatre émetteurs sont :

1 Chèque Déjeuner  
2 EDENRED (anciennement ACCOR Services) qui émettent les Tickets Restaurant  
3 NATIXIS Intertitres qui émettent les Chèque de Table  
4 SODEXO Solutions de Motivations qui émettent les Chèques Restaurant

La clé de contrôle est calculée comme suit :   
(

Si la clé de contrôle ne correspond à la valeur trouvée par ce calcul, le titre restaurant est invalide.

Le champ code famille est toujours dans notre cas égal à 000.

Le champ produit sera égal à 0 (titres restaurant) ou 2 (titres de service).

Le millésime est égal à 1 pour l’année 2011.

Par exemple pour un titre portant le code-barres suivant :

388663300730074011404991

388663300 représente le numéro du titre.

73 est la clé de cryptage.

Le titre vaut 7,40€.

Il s’agit d’un titre émis par Chèque Déjeuner.

Le titre est valide car (388663300+73+740+1+049+9+1)modulo97=14.

Le code famille indique que ce titre est valide pour l’alimentaire, le transport et l’énergie (pour les titres restaurant ce code sera toujours égal à 000 qui désigne les titres restaurant et dérivés).

Le titre est un CAP (pour les titres restaurant ce champ sera toujours égal à 0 ou 2).

Il s’agit d’un titre valable en 2011.

### Architecture

L’architecture du logiciel est représenté par le schéma suivant :

Vues

Classes

Contrôleur

Classes

Data

Base de données Firebird

Classes

Modèle

**Application .NET**

Image . : Schéma de l'architecture

L’architecture peut être divisée en 2 grandes parties :

* La base de données Firebird : elle enregistre toutes les données utilisées par l’application
* L’application : Utilisant le framework .NET, elle interagit avec l’utilisateur et la base de données et effectue les traitements nécessaires

L’application est composée de 4 parties, articulées autour des classes de contrôle.

Côté utilisateur, les vues permettent à l’utilisateur d’interagir avec le logiciel. Plusieurs vues sont proposées, ayant chacune un ou plusieurs contrôleurs associés. Ces contrôleurs assurent l’interface entre les classes métiers (modèle), les classes de données (data) effectuant les requêtes vers la base de données, et les vues.

## Conception détaillée du logiciel

### Diagramme de classes logiciel

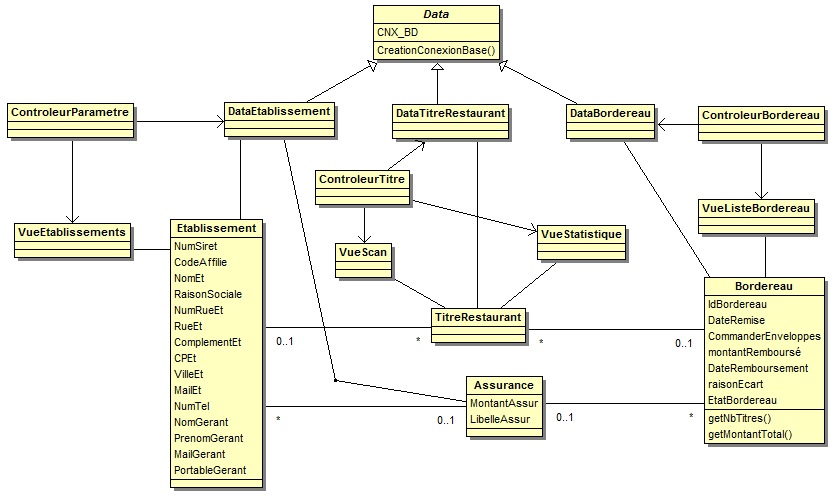


Image . : Diagramme de classes logiciel

Dans l’application que nous allons réaliser, le support physique choisi (par le demandeur) pour la persistance des données est la base de données. Étant donné qu’il n’y a qu’un type de sauvegarde choisi nous avons décidé de mettre en place pour la conception de cette application le pattern Model-View -Controller (Modèle-Vue-Contrôleur en français).

Avec cette architecture, on peut facilement répercuter les changements qui sont faits à travers la vue par l’utilisateur sur le modèle c’est-à-dire sur les données enregistrées dans la base de données et vice-versa. Cette architecture permet de garder la cohérence entre les données montrées par la vue et les données contenues dans le modèle. À chaque changement dans la vue ou dans le modèle, le contrôleur va avertir la classe nécessaire.

### Schéma de la base de données

Assurance  
(  
 MontantAssur NUMBER(5),  
 LibelleAssur VARCHAR(50)  
) ;

Etablissement  
(  
 NumSiret VARCHAR(15),  
 RaisonSociale VARCHAR(50),  
 CodeAffilie VARCHAR(12),  
 NomEt VARCHAR(50),  
 NumRueEt NUMBER(4),  
 RueET VARCHAR(100),  
 Complement VARCHAR(100),  
 CPEt VARCHAR(5),  
 VilleEt VARCHAR(50),  
 MailEt VARCHAR(120),  
 NumTel VARCHAR(20),  
 NomG VARCHAR(50),  
 PrenomG VARCHAR(50),  
 MailG VARCHAR(120) ,  
 PortableG VARCHAR(20),

#MontantAssur NUMBER(5),  
EstLocal INTEGER,  
IdDistant VARCHAR(50)

) ;

EtatBordereau

(

Libelle VARCHAR(100)

) ;

Bordereau  
(  
 IdBordereau NUMBER(6),

EtatBord VARCHAR(20),  
 DateRemise TIMESTAMP,  
 CommanderColisur NUMBER(1),  
 #MontantAssur NUMBER(5),

#NumSiret VARCHAR(15),  
 DateRemboursement TIMESTAMP,  
 MontantRembourse TIMESTAMP,  
 RaisonEcart VARCHAR(100)  
) ;

TitreRestaurant  
(  
 NumCodeBarres VARCHAR(24),  
 #IdBordereau NUMBER(6),  
 #NumSiret VARCHAR(15),  
 DateTitre TIMESTAMP  
) ;

### Cas d’utilisation

Nous avons décidé par souci de clarté de faire des diagrammes de cas d’utilisation sur deux niveaux. Le premier montre les processus dans leur globalité, sans entrer dans le détail des traitements.

Le deuxième développe tous les processus.

Ci-dessous le diagramme global des cas d’utilisation :

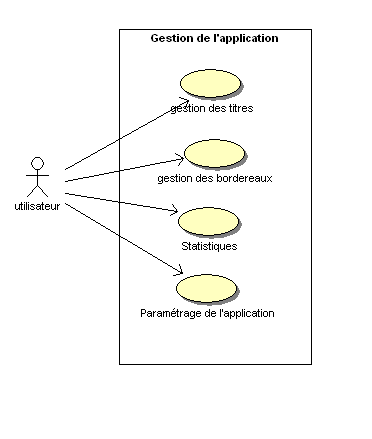


Image . : Use Case global

### Hiérarchie des fonctionnalités

Image . : Schéma hiérarchique des fonctionnalités

Nous allons par la suite expliciter chaque fonctionnalité.

### Gestion des titres restaurant

Le use-case suivant correspond à cette fonctionnalité :

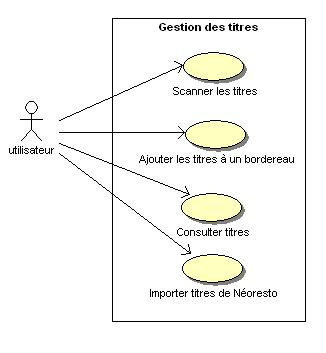


Image . : Use Case Gestion des Titres

On veut pouvoir, grâce à l’application, gérer les titres restaurant. On entend par gérer les titres, la possibilité de les scanner, de les ajouter à un bordereau, de les importer de NeoResto et de les consulter. L’utilisateur va scanner les titres afin que les informations concernant le titre qu’il enregistre soient insérées dans la base de données. Il va pouvoir aussi à partir de l’application ajouter des titres à un bordereau. L’utilisateur pourra également consulter les titres qui sont déjà enregistrés dans l’application. Enfin, pour les clients dotés du logiciel NeoResto, il sera possible d’importer les titres scannés à partir de ce logiciel, ainsi il n’aura pas à les scanner deux fois.

#### Ajout d’un titre restaurant

##### Spécifications

###### Description Sommaire

Ce use-case décrit l’action d’ajout d’un titre restaurant.

###### Déroulement des événements

1. Flux de bases

Ce use-case commence lorsqu’un utilisateur souhaite ajouter un titre restaurant.

Par défaut aucune information n’est affichée.

1. Flux de remplacement

Si l’utilisateur scanne ou tape un code-barres invalide un message d’erreur lui est affiché.

###### Exigences particulières

Aucune.

###### Pré-conditions

Le code-barres ajouté doit correspondre aux normes définies par la CRT (voir rapport de conception).

###### Post-conditions

Aucune.

##### Maquette

L’interface d’ajout de titres est telle que ci-dessous.

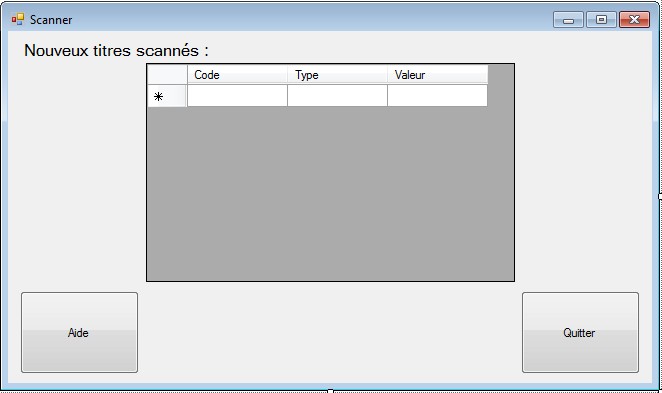


Image . : Maquette Scan des titres

Au fur et à mesure que l’utilisateur scanne les titres, ils s’affichent dans le tableau. L’utilisateur peut également accéder à la fenêtre d’aide en cliquant sur le bouton correspondant. Quand les titres sont scannés la date et l’heure sont enregistrées en même temps.

##### Diagramme de séquence

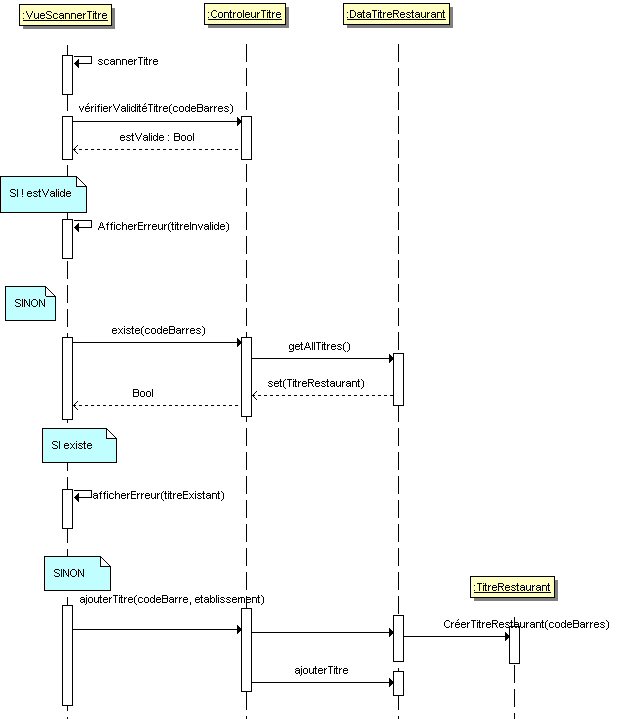


Image . : Diagramme de séquence Insertion de Titres Restaurant

#### Consultation des titres restaurant

##### Spécifications

###### Description Sommaire

Ce use-case décrit l’action de consulter les titres restaurant enregistrés dans l’application.

###### Déroulement des événements

1. Flux de base

Ce use-case commence quand l’utilisateur souhaite consulter les titres enregistrés.

* Par défaut tous les titres enregistrés dans l’application seront affichés.
* Des filtres conditionnels sont disponibles afin de modifier l’affichage.

1. Flux de remplacement

Si les filtres que l’utilisateur utilise sont incohérents, un message d’erreur lui est affiché.

###### Exigences particulières

Aucune.

###### Pré-conditions

Aucune.

###### Post-conditions

Aucune.

##### Maquette

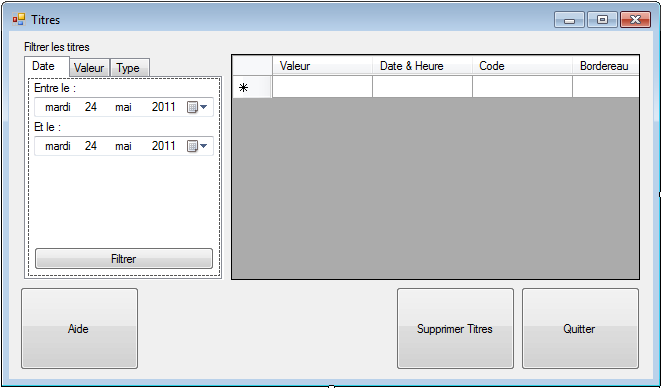
****

Image . : Maquette Consultation de Titres Restaurant

L’utilisateur peut trier les titres par date, par valeur ou par type. Pour changer de type de filtre il suffit de cliquer sur chaque onglet.

##### Diagramme de séquence

****

Image . : Diagramme de séquence Consultation des Titres Restaurant

#### Suppression d’un titre restaurant

##### Spécifications

###### Description Sommaire

Ce use-case décrit l’action de suppression d’un titre restaurant.

###### Déroulement des événements

1. Flux de base

Ce use-case commence quand l’utilisateur souhaite supprimer des titres enregistrés.

* Par défaut, tous les titres enregistrés dans l’application seront affichés.
* Des filtres conditionnels sont disponibles afin de modifier l’affichage.

1. Flux de remplacement

Si la suppression est impossible un message d’alerte est affiché au client.

###### Exigences particulières

Aucune.

###### Pré-conditions

Aucune.

###### Post-conditions

Aucune.

##### Maquettes

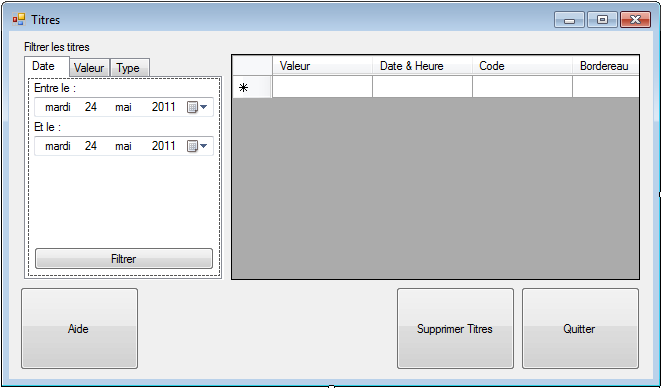


Image . : Maquette Suppression de Titres Restaurant

Pour supprimer des titres, l’utilisateur doit sélectionner les titres qu’il souhaite effacer avant de cliquer sur le bouton adéquat.

##### Diagramme de séquence

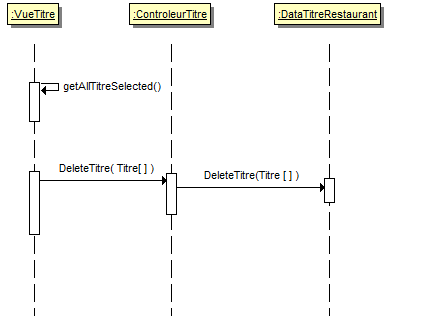


Image . : Diagramme de séquence Suppression de Titres Restaurant

### Gestion des bordereaux

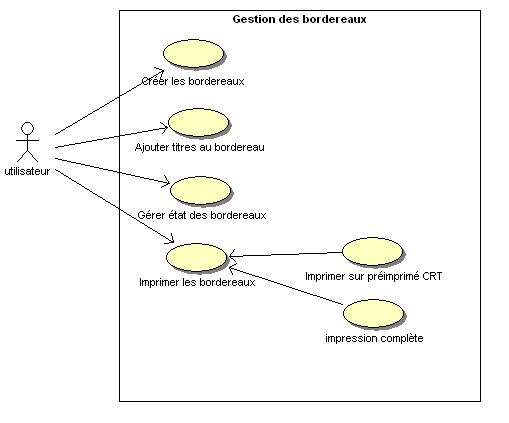


Image . : use case Gestion des bordereaux

L’application permettra de gérer les bordereaux. Il est possible de créer les bordereaux, de leur ajouter des titres qui n’ont pas encore été remboursés, de gérer l’état des bordereaux et de les imprimer. Deux modes d’impression sont mis à la disposition de l’utilisateur : impression sur le bordereau pré rempli de la CRT ou l’impression complète sur une feuille vierge.

### Diagramme de séquence global

Image . : Diagramme de Séquence Gestion des bordereaux

#### Création d’un bordereau

##### Spécifications

###### Description Sommaire

Ce use-case décrit l’action de création d’un bordereau.

###### Déroulement des événements

1. Flux de base

Ce use-case commence quand l’utilisateur souhaite créer un nouveau bordereau.

1. Flux de remplacement

Si le nom que l’utilisateur souhaite attribuer au bordereau est invalide un message d’erreur lui est affiché.

###### Exigences particulières

Aucune.

###### Pré-conditions

Aucune.

###### Post-conditions

Aucune.

##### Maquettes

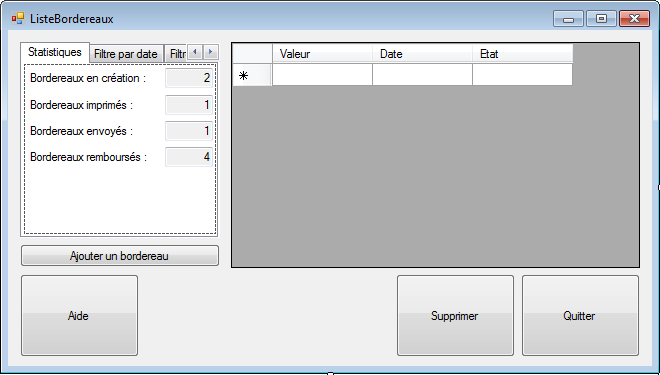


Image . : Maquette Liste des Bordereaux

Afin d’accéder à la fonctionnalité d’ajout l’utilisateur doit cliquer sur le bouton « Ajouter un bordereau ». Il accède ensuite à la fenêtre suivante.

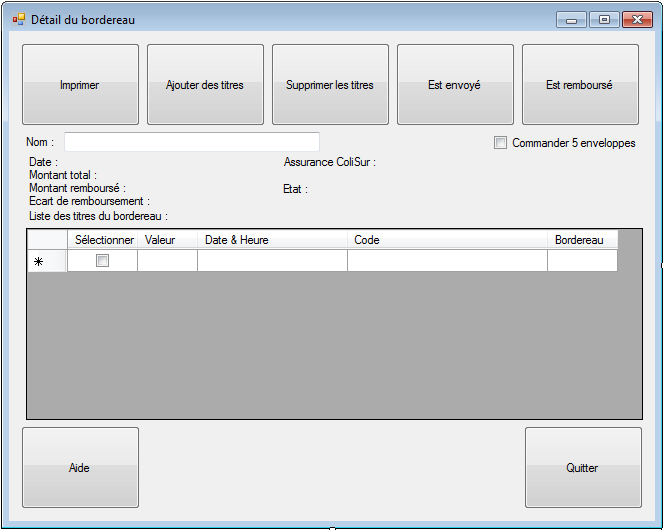


Image . : Maquette Détails d’un Bordereaux

Afin de créer le bordereau l’utilisateur rentre le nom qu’il veut attribuer au bordereau. Il peut lui ajouter des titres, en supprimer ou l’imprimé en cliquant sur le bouton adéquate. Lorsqu’il envoie le bordereau ou qu’il en reçoit le remboursement il peut changer son état.

#### Modification d’un bordereau

##### Spécifications

###### Description Sommaire

Ce use-case décrit l’action de modification d’un bordereau.

###### Déroulement des événements

1. Flux de base

Ce use-case commence quand l’utilisateur souhaite modifier un bordereau existant.

1. Flux de remplacement

Si l’utilisateur effectue des modifications invalides un message d’erreur est affiché.

###### Exigences particulières

Aucune.

###### Pré-conditions

Aucune.

###### Post-conditions

Aucune.

##### Maquettes

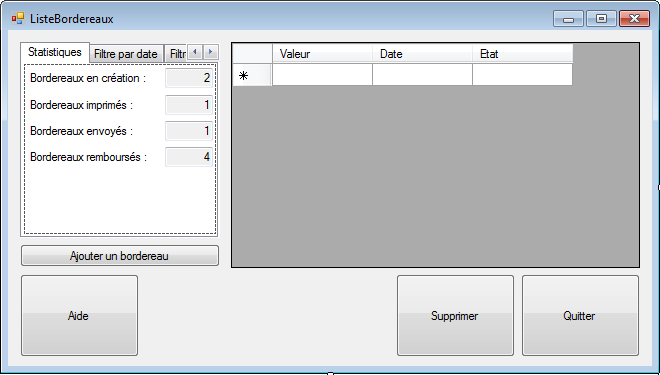


Image . : Maquette Liste des Bordereaux

Afin de modifier un bordereau l’utilisateur doit effectuer un double-clic sur le nom du bordereau qu’il désire modifier. Ainsi il peut accéder à la fenêtre détaillant le bordereau. Afin de faciliter la recherche, différents filtres sont disponibles : l’utilisateur peut donc trier la liste des bordereaux disponibles par date ou état par exemple.

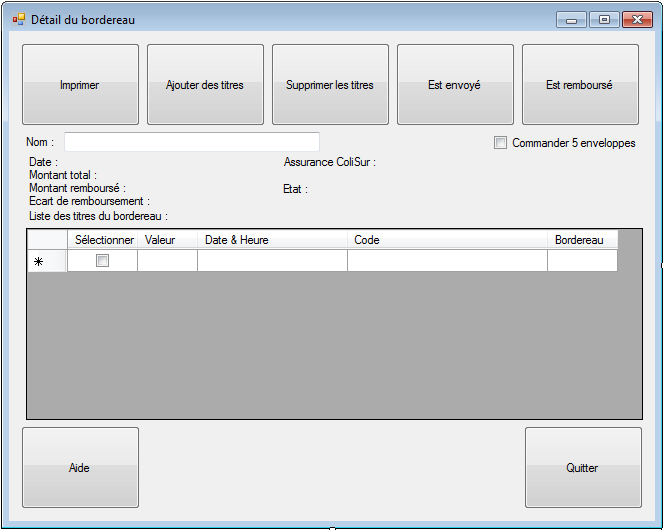


Image . : Maquette Détails d'un Bordereaux

À partir de cette fenêtre l’utilisateur possède toutes les informations relatives au bordereau : sa date de création, le montant total du bordereau (recalculé automatiquement à chaque ajout ou suppression de titres), l’état, le montant de l’assurance. Lorsque l’utilisateur envoie le bordereau à la CRT il doit cliquer sur le bouton « Est envoyé » afin que le nouvel état soit enregistré. Il peut également ajouter des titres ou en supprimer. Lorsque l’utilisateur clique sur le bouton « Est remboursé » une fenêtre de type pop-up s’ouvre et il doit renseigner le montant réellement remboursé par la CRT. Quand l’utilisateur quitte sur le bouton « Quitter » tous les changements sont enregistrés.

#### Impression d’un bordereau

##### Spécifications

###### Description Sommaire

Ce use-case décrit l’action d’impression d’un bordereau.

###### Déroulement des événements

1. Flux de base

Ce use-case commence quand l’utilisateur souhaite imprimer un bordereau.

1. Flux de remplacement

Si un problème survient lors de l’impression un message d’erreur est affiché.

###### Exigences particulières

Aucune.

###### Pré-conditions

Le bordereau doit être enregistré dans l’application.

###### Post-conditions

Aucune.

##### Maquette

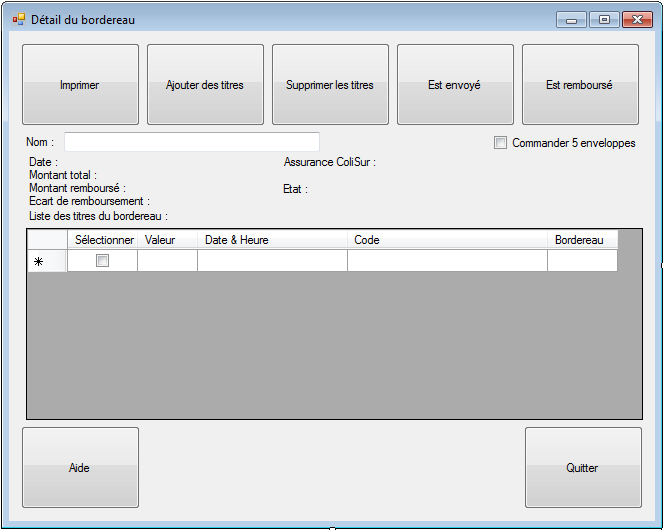


Image . : Maquette Détails d'un bordereau

Afin d’imprimer un bordereau l’utilisateur doit accéder à la fenêtre détail du bordereau. Quand il clique sur le bouton imprimer, une fenêtre de type pop-up s’ouvre où il peut choisir le type d’impression qu’il désire. Une fois le bordereau imprimé, son état passe directement à l’état imprimé.

#### Suppression d’un bordereau

##### Spécifications

###### Description Sommaire

Ce use-case décrit l’action de suppression d’un bordereau.

###### Déroulement des événements

1. Flux de base

Ce use-case commence quand l’utilisateur souhaite supprimer un bordereau de l’application.

1. Flux de remplacement

Si un problème survient lors de la suppression du bordereau un message d’erreur est affiché.

###### Exigences particulières

Aucune.

###### Pré-conditions

Aucune.

###### Post-conditions

Aucune.

##### Maquette

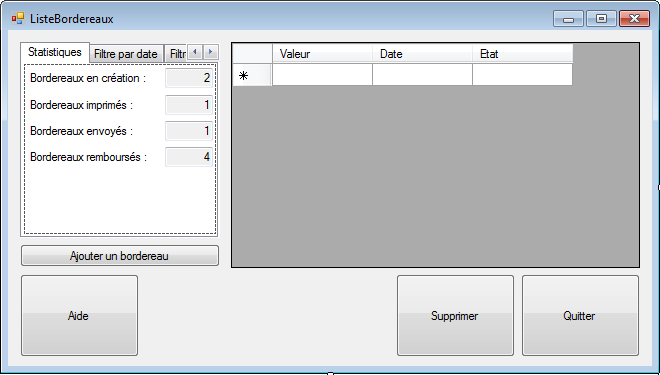
****

Image . : Maquette Liste Bordereaux

Pour supprimer un bordereau, l’utilisateur sélectionne dans la liste des bordereaux ceux qu’il souhaite effacer. Puis il doit cliquer sur le bouton « Supprimer ».

##### Diagramme de séquence

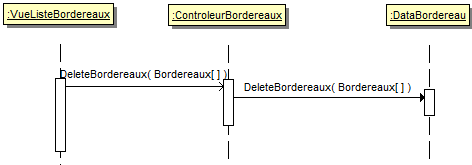


Image . : Diagramme de Séquence Suppression de Bordereaux

### Gestion des établissements

#### Consultation et modification d’un établissement

##### Spécifications

###### Description Sommaire

Ce use-case décrit la consultation et la modification d’un établissement déjà enregistré.

###### Déroulement des événements

1. Flux de base

Ce use-case commence quand l’utilisateur souhaite consulter ou modifier les informations d’un établissement.

Par défaut, les informations enregistrées sur l’établissement courant sont affichées.

1. Flux de remplacement

Si l’utilisateur efface et ne renseigne pas une information nécessaire à l’établissement des bordereaux (par exemple le numéro Siret, le code affilié ou le nom de l’établissement) un message d’erreur lui est affiché.

###### Exigences particulières

Aucune.

###### Pré-conditions

Le numéro de Siret doit être valide.

###### Post-conditions

Aucune.

##### Maquette

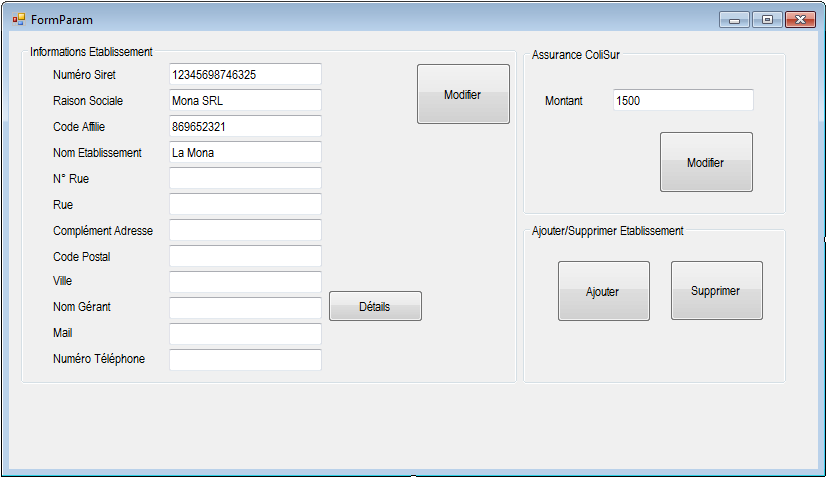


Image . : Maquette Informations Établissement

Les informations enregistrées sur l’établissement dans la base de données sont affichées. Si l’utilisateur souhaite les modifier il suffit d’accéder aux zones éditables et d’en modifier le contenu. Pour modifier les informations concernant le gérant il suffit de cliquer sur le bouton détail.

La fenêtre ci-dessous apparaît.

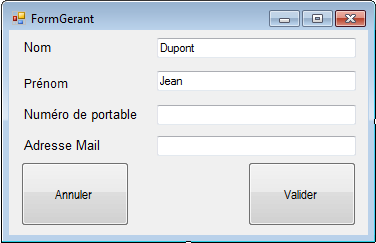


Image . : Maquette Informations Gérant

L’utilisateur peut donc entrer les informations souhaitées et ensuite valider en cliquant sur le bouton adéquate. S’il souhaite abandonner l’action il suffit de cliquer sur le bouton « Annuler ».

Afin de sauver toutes les informations qu’il a modifiées l’utilisateur doit cliquer sur le bouton « Modifier » de la première fenêtre. S’il appuie sur un autre bouton avant de valider toutes les informations rentrées seront perdues.

L’utilisateur peut également modifier l’assurance de l’établissement dont il consulte les informations. Dans l’exemple l’assurance actuelle s’élève à 1500 € pour la changer il suffit de cliquer sur le bouton « Modifier » de la section « Assurance ColiSur ».

La fenêtre suivante apparaît.

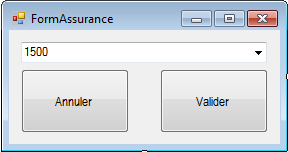


Image . : Maquette Modification Assurance

L’utilisateur peut-donc choisir le montant dans la liste contenant les montants prédéfinis. Pour enregistrer le changement il suffit de cliquer sur le bouton « Valider ».

##### Diagramme de séquence

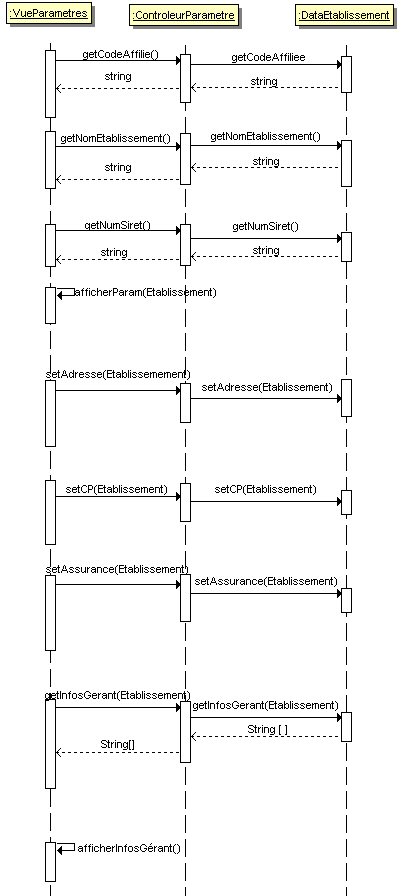


Image . : Diagramme de Séquence Consultation et modification d’un Etablissement

#### Ajout d’un établissement

##### Spécifications

###### Description Sommaire

Ce use-case décrit l’action d’ajout d’un établissement à l’application.

###### Déroulement des événements

1. Flux de base

Ce use-case commence quand l’utilisateur souhaite ajouter un établissement.

1. Flux de remplacement

Si l’utilisateur ne renseigne pas une application nécessaire à l’établissement des bordereaux (par exemple le numéro de Siret, le code affilié ou le nom de l’établissement) un message d’erreur lui est affiché.

###### Exigences particulières

Aucune.

###### Pré-conditions

Le numéro de Siret doit être valide.

###### Post-conditions

Aucune.

##### Maquettes

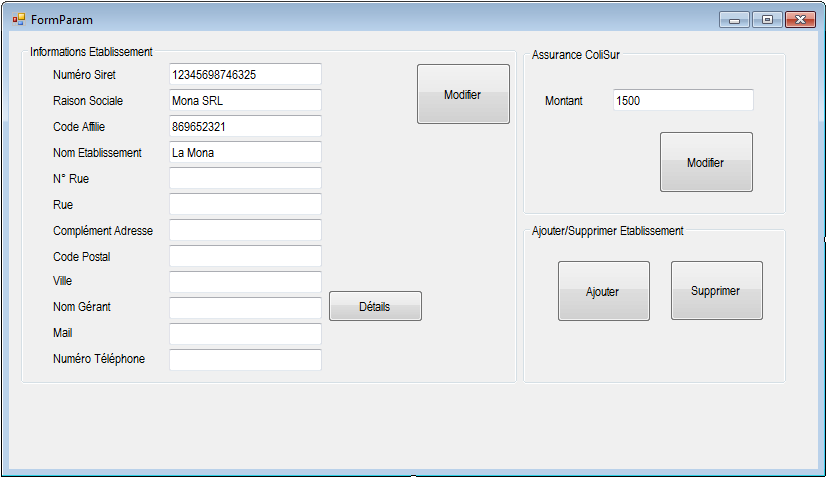


Image . : Maquette Informations Établissement

Afin d’ajouter un établissement à l’application l’utilisateur doit cliquer sur le bouton « Paramètres » puis sur le bouton « Ajouter «  se trouvant dans la section « Ajouter/Supprimer un établissement ».

La fenêtre ci-dessous apparaît.

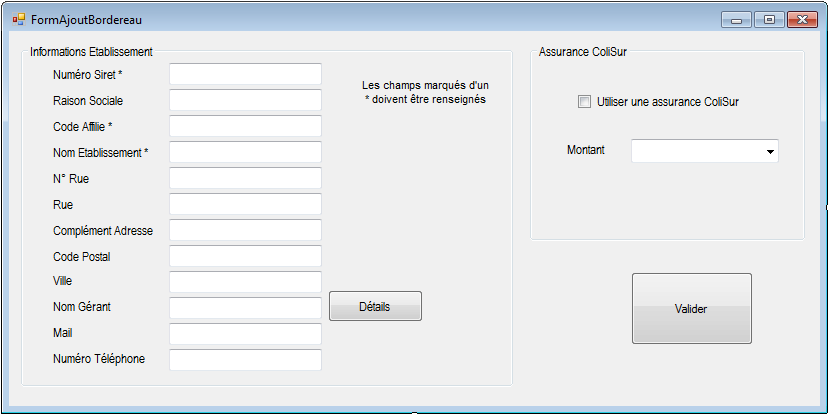


Image . : Maquette Ajout d'un Établissement

Afin d’ajouter l’établissement, l’utilisateur doit remplir les champs qu’il désire renseigner (trois champs sont obligatoires : le numéro de Siret, le code affilié et le nom de l’établissement). Il peut aussi à ce moment-là choisir l’assurance qui sera associée à l’établissement en cochant la case utiliser une assurance ColiSur puis en sélectionnant un montant dans la liste prédéfinie. Pour enregistrer les informations sur le gérant le client doit cliquer sur le bouton « Détails ». La fenêtre suivant de type pop-up apparait.

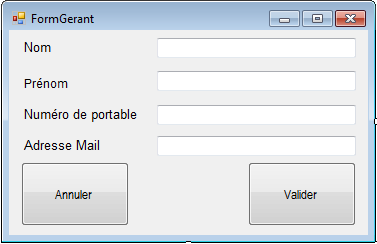


Image . : Maquette Informations Gérant

L’utilisateur peut donc entrer les informations souhaitées et ensuite valider en cliquant sur le bouton adéquate. S’il souhaite abandonner l’action il suffit de cliquer sur le bouton « Annuler ».

Afin de sauver toutes les informations qu’il a ajoutées l’utilisateur doit cliquer sur le bouton « Valider » de la première fenêtre. S’il appuie sur un autre bouton avant de valider toutes les informations rentrées seront perdues.

##### Diagramme de séquence

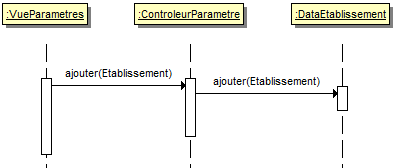


Image . : Diagramme de Séquence Ajout d'un Établissement

### Statistiques

#### Spécifications

###### Description Sommaire

Ce use-case décrit la consultation des statistiques.

###### Déroulement des événements

1. Flux de base

Ce use-case commence quand l’utilisateur souhaite consulter les statistiques.

Par défaut le chiffre d’affaires des trois derniers mois est affiché.

1. Flux de remplacement

S’il n’y a pas de titres ou d’établissements enregistrés dans la base de données, la fenêtre statistiques reste vide.

###### Exigences particulières

Aucune.

###### Pré-conditions

Il doit y avoir au moins un établissement et au moins un titre enregistrés dans la base de données pour avoir des résultats disponibles.

###### Post-conditions

Aucune.

#### Maquette

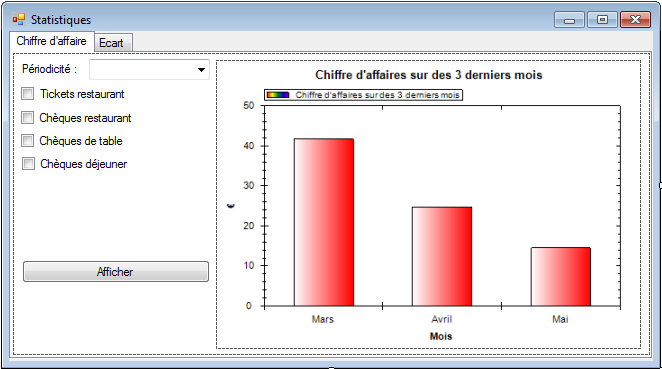


Image . : Maquette Statistiques

L’utilisateur pourra filtrer les données et avoir le chiffre d’affaires pour chaque type de titres.

En cliquant sur l’onglet écart des données concernant les écarts de remboursement lui seront affichés sous forme de camembert.

#### Diagramme de séquence

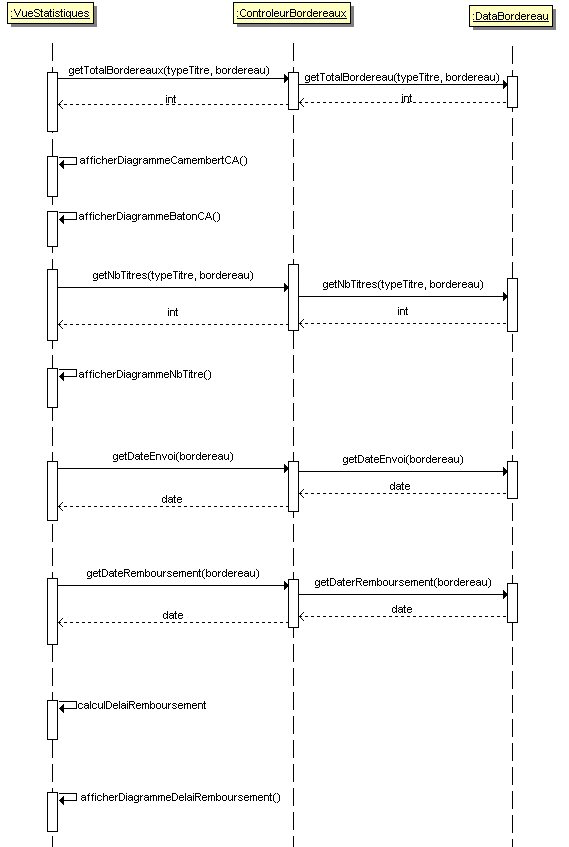


Image . : Diagramme de Séquence Statistiques

## Développement

### Fenêtre principale

L’application utilise principalement une fenêtre composée d’une bande en haut munie des boutons « Scan libre », « Bordereaux », « Statistiques », « Etablissement » et d’une liste pour changer l’établissement courant, d’un bouton « Aide » en bas à gauche et d’un bouton « Quitter » en bas à droite. Une grande place reste libre au milieu pour accueillir les différentes informations des vues.

Les 4 boutons de la bande haute affichent leur vue respective en insérant au centre un UserControl[[3]](#footnote-3).

Exemple pour le bouton « Bordereaux » :

Au clic sur le bouton (nommé bt\_bordereaux), la méthode bt\_bordereaux\_Click est appelée.

this.bt\_bordereaux.Click += new System.EventHandler(this.bt\_bordereaux\_Click);

Cette méthode commence par supprimer de la vue le UserControl actuel (nommé uc), crée le UserControlVueBordereaux, le place au bon endroit sur le fenêtre d’accueil et l’ajoute à la fenêtre.

private void bt\_bordereau\_Click(object sender, EventArgs e)

{

this.Controls.Remove(this.uc);

this.uc = new Vue.UserControlVueBordereau(this.numSIRET, this);

this.uc.Location = new System.Drawing.Point(12, 120);

this.Controls.Add(this.uc);

}

On obtient ainsi la vue telle qu’elle est présentée sur l’image 4.8.3.1.

### Schéma d’enchaînement des vues

Ce schéma récapitule comment s’agencent les fenêtres entre elles. Par exemple, à partir du contrôle utilisateur  « UserControlScan » on peut accéder à la fenêtre « FormClavier » ou à la fenêtre « FormImportNeoResto ».

Image . : Schéma d’enchaînement des vues

#### Détail « FormListeBord »

Image . Schéma de dépendance de la fenêtre « FormListBord »

#### Détails « FormChoixAjoutTitre »

Image . : Schéma de dépendance de la fenêtre « FormChoixAjoutTitre »

### Ensemble des classes

Les classes utilisées sont détaillés dans le diagramme des classes de la conception détaillée (cf. § 4.7.1 Diagramme des classes logiciel) et le schéma d’enchaînement des vues précédent. Les vues du diagramme des classes sont détaillées dans ce schéma, plusieurs Forms et UserControls composant souvent une même vue du diagramme.

### Connexion à la base de données

La connexion à la base de données est gérée dans la classe Data. Toutes les classes faisant des requêtes sur la base de données embarquée nommée « NEOTR.FDB » héritent de cette classe.

Cette classe possède un unique attribut statique de type FbConnection nommé CNX\_BD.

L’algortihme est le suivant :

public void CreationConnexionBase()

{

FbConnectionStringBuilder csb = new FbConnectionStringBuilder();

csb.Database = "neotr.fdb";

csb.UserID = "SYSDBA";

csb.Password = "masterkey";

csb.ServerType = FbServerType.Embedded;

Data.CNX\_BD = new FbConnection(csb.ToString());

try

{

Data.CNX\_BD.Open();

}

catch (Exception ee)

{

MessageBox.Show("Impossible de se connecter à la BD Firebird.\n\n" + ee.ToString());

}

}

On crée une nouvelle FbConnectionStringBuilder csb en lui indiquant tous les paramètres de connexion : le nom de la base (« neotr.fdb »), l’identifiant, le mot de passe et le type de serveur utilisé (Embedded qui signifie embarqué en français). Ensuite on initialise l’attribut CNX\_BD en utilisant csb. Enfin on ouvre la connexion. Si une erreur survient lors de l’ouverture de la connexion, un message d’erreur s’affiche.

### Initialisation de la base de données

La classe InitBD s’occupe de la création de la base de données elle hérite de la classe Data. S’il s’agit de la première utilisation de l’application le fichier « NEOTR.FDB » qui contient la base de données n’existe pas. Il faut donc créer ce fichier puis créer la base de données et remplir les tables nécessaires.

La méthode suivante permet de créer la base de données :

public void CreationBD()

{

FbConnection.CreateDatabase("Database=neotr.fdb;ServerType=1");

MessageBox.Show("Base de données créée.");

}

Les requêtes SQL permettant de construire la base se trouvent dans le fichier de ressources de l’application. Ces commandes sont utilisées dans la méthode « InitBD » afin de créer les tables.

Par exemple, l’algorithme suivant permet de créer la table assurance :

FbCommand commandAssur = new FbCommand(WindowsFormsApplication2.Properties.Resources.REQCreationTableAssurance, CNX\_BD);

int resultAssur = commandAssur.ExecuteNonQuery();

WindowsFormsApplication2.Propertie.Ressources.REQCreationTableAssurance est le chemin du fichier qui contient la requête SQL suivante :

CREATE TABLE ASSURANCE

(

MontantAssur NUMERIC(7,2) NOT NULL,

LibelleAssur VARCHAR(50),

CONSTRAINT PK\_ASSUR PRIMARY KEY(MontantAssur)

) ;

Une fois que la base de données est créée, ces algorithmes ne sont plus exécutés. La classe InitBD qui est appelée dans le programme principal ne sert qu’à établir une connexion à la base au début du lancement de l’application.

### Échanges avec la base de données

Les classes préfixée par Data sont les classes destinées à communiquer avec la base de données. Par exemple, pour récupérer tous les établissements enregistrés dans la base on commence par récupérer le nombre d’établissements enregistrés dans la base :

FbCommand commandCount = new FbCommand("SELECT COUNT(\*) FROM ETABLISSEMENT", CNX\_BD);

FbDataReader readerCount = commandCount.ExecuteReader();

readerCount.Read();

int nbEt = readerCount.GetInt32(0);

tabEt = new Etablissement[nbEt];

On utilise la classe FbCommand qui est fournit par Firebird pour transmettre la requête à la base à laquelle on est connecté. Ensuite un utilise FbDataReader pour exécuter et lire le résultat de la requête. Le résultat est stocké sous forme de tableau. Dans ce cas il s'agit d’un tableau contenant un seul élément et qui est un entier. On utilise donc la fonction GetInt32 avec le paramètre 0 (numéro de l’indice).

On peut donc par la suite récupérer la liste des établissements et construire un tableau d’établissements :

FbCommand command = new FbCommand("SELECT \* FROM ETABLISSEMENT", CNX\_BD);

FbDataReader reader = command.ExecuteReader();

int i = 0;

while (reader.Read()

{

decimal montant = reader.GetDecimal(11);

Assurance assur = new Assurance(montant);

tabEt[i] = new Etablissement(reader.GetString(0), reader.GetString(1), reader.GetString(2), reader.GetString(3), reader.GetString(4), reader.GetString(5), reader.GetString(6), reader.GetString(7), reader.GetString(8), reader.GetString(9), reader.GetString(10), assur, reader.GetString(12), reader.GetString(13), reader.GetString(14), reader.GetString(15));

i++;

}

### Lecture des codes-barres

#### Fonctionnement

L’une des premières fonctionnalités à avoir été développée est la lecture des codes-barres des titres restaurants à l’aide d’une douchette, et leur enregistrement dans le logiciel.

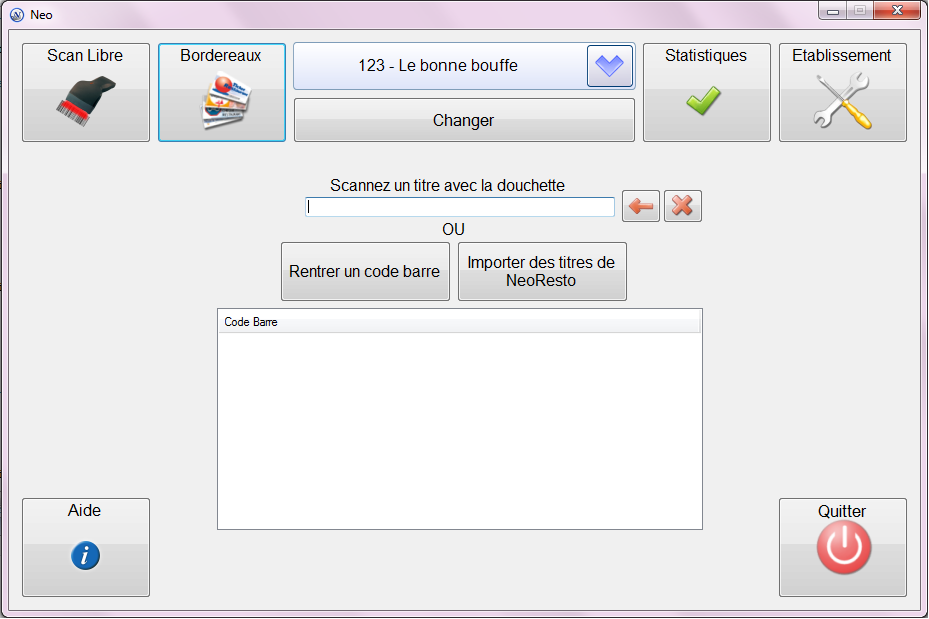


Image . : Fenêtre Scan Titres

La douchette se comporte comme une entrée standard, comparable à un clavier, branché sur un port USB. Lorsque l’on scanne un titre, la chaine de caractères est lue et envoyée à l’ordinateur sur l’entrée standard.

Il fallait également prévoir la possibilité qu’un code-barres ne puisse être lu (en cas de défectuosité de celui-ci, par effacement partiel par exemple). Pour se faire, nous avons développé un clavier virtuel muni des 10 chiffres et de boutons d’annulation, de suppression et de validation. L’utilisateur peut alors rentrer la chaine de caractère ainsi.

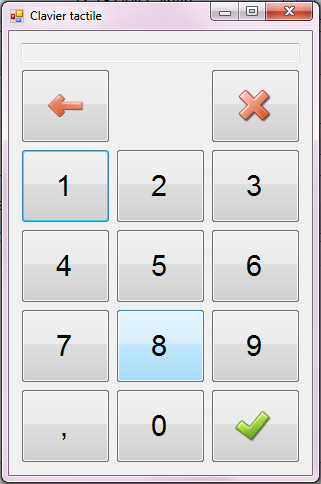


Image . : Fenêtre du clavier tactile

La lecture du code-barres est validée lorsque 26 caractères ont été saisis (24 caractères qui seront interprétés, auxquelles il faut ajouter 2 caractères de début et de fin non interprétés), que ce soit à l’aide de la douchette ou du clavier tactile.

On vérifie ensuite que le titre ne soit pas déjà enregistré dans la base de données. Si c’est le cas, on avertit l’utilisateur et le titre est rejeté, sinon, on contrôle sa validité à l’aide de la clé de contrôle contenue dans le code-barres.

Une fois tous ces contrôles effectués et validés, le titre et accepté et ajouté en base de données. L’utilisateur peut alors en saisir un autre.

L’enregistrement de titres par scan peut se produire dans deux cas, définissant deux scénarii distincts :

* Dans le cas d’un scan libre, l’utilisateur scanne des titres pour un établissement sans l’associer à un bordereau. Le titre est donc lié à un établissement mais n’est attribué à aucun bordereau : il reste libre.
* Dans le cas d’un ajout de titres à un bordereau, le titre est ajouté en étant lié à un bordereau.

#### Classes utilisées

Accueil, Titre, DataTitre, ControleurTitre, FormImportNeoResto

#### Algorithmes

Dès que le contenu de la zone de texte change, l’algorithme suivant est exécuté :

if (this.textBox\_antialiasingCB.Text.Length == 26)

{

this.ajouterTitre(this.textBox\_antialiasingCB.Text);

this.textBox\_antialiasingCB.Text = "";

}

Si la longueur du texte contenu est égale à 26, il s’agit d’un code barre complet. On l’ajoute alors dans les titres, puis on efface le contenu de la textBox.

public Boolean ajouterTitre(String pCode)

{

Boolean ajoutOK;

if (pCode.Length > 0) // Pour gérer les problèmes de buffer

{

if (this.ctrl.titreEstValide(pCode))

{

String date = DateTime.Now.ToString().Replace("/",".");

Modele.Titre monTitre = new Modele.Titre(pCode, this.SIRET, this.numBordereau, date);

ajoutOK = this.ctrl.ajouterTitre(monTitre);

if (ajoutOK)

{

this.getList().Items.Add(pCode);

}

return ajoutOK;

}

else

{

return false;

}

}

else

return false;

}

On commence par vérifier que le titre est valide avec la méthode titreEstValide de ControleurTitre.

Cet algorithme vérifie que la clé de contrôle soit valide et que le titre ne soit pas déjà présent en base de données.

Si c’est le cas, on ajoute le titre avec la méthode ajouterTitre du même contrôleur en créant un titre. La date est la date du jour, en remplaçant les slashes par des points pour respecter le format de date de Firebird. Le numéro de bordereau est également passé. Si les titres sont scannés librement, numBordereau est égal à NULL.

Si l’ajout s’est bien passé, on ajoute le titre à la liste pour que l’utilisateur sache que l’insertion s’est bien passée.

Dans le cas contraire, on retourne false et on notifie l’utilisateur.

### Gestion des établissements

#### Fonctionnement

L’utilisateur a la possibilité de gérer les titres de plusieurs établissements à la fois. Il peut donc ajouter, supprimer ou modifier des établissements.

Lorsque l’utilisateur clique sur le bouton « Etablissement », l’écran ci-dessous apparait. Par défaut, tous les établissements enregistrés dans la base sont listés.

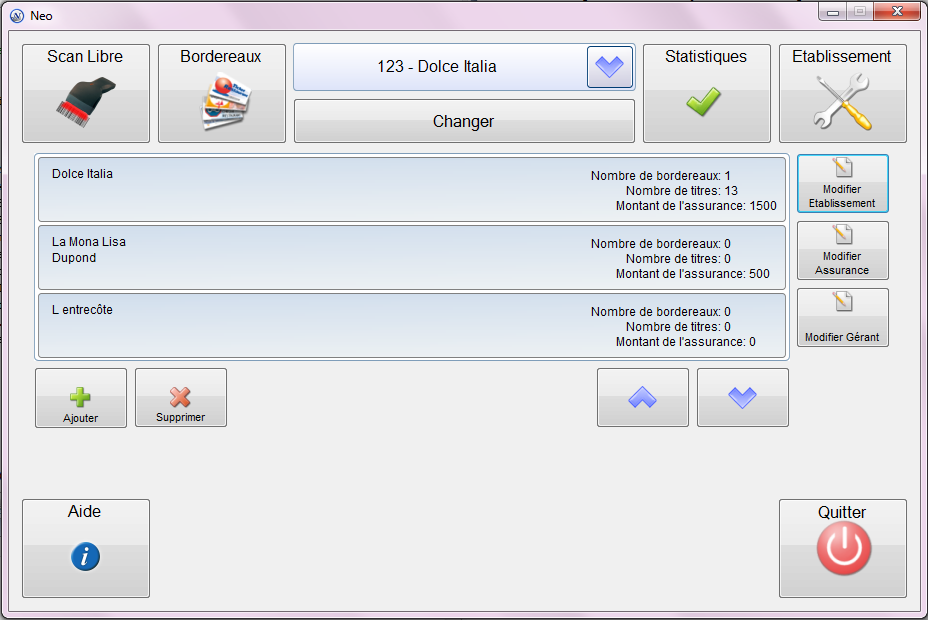


Image . : Fenêtre Informations Établissements

Si l’utilisateur souhaite ajouter un établissement, il clique sur le bouton « Ajouter » et la fenêtre suivante apparaît.

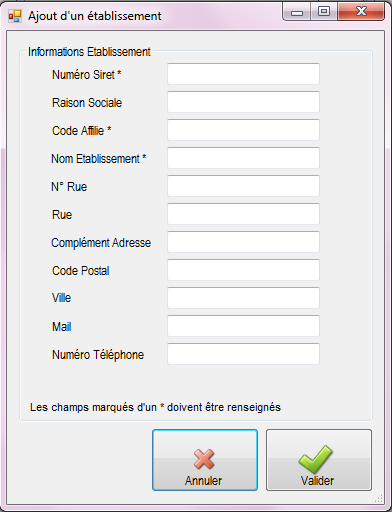


Image . : Fenêtre Ajout d'un Établissement

L’utilisateur remplit les champs qu’il souhaite renseigner. Les champs « Numéro Siret », « Code Affilié » et « Nom Etablissement » sont obligatoires. Si l’un de ces trois champs n’est pas remplit un message d’erreur de type pop-up sera affiché et l’utilisateur devra recommencer son opération.

Lorsqu’il a terminé il peut valider en cliquant sur le bouton adéquat ou en appuyant sur la touche Entrée du clavier. Il peut également annuler l’opération en cliquant sur le bouton « Annuler » ou appuyant sur la touche  Échap.

Si l’utilisateur souhait supprimer un établissement il doit sélectionner l’établissement dans la liste et ensuite cliquer sur le bouton « Supprimer ». S’il ne sélectionne pas d’établissement un message d’erreur lui est envoyé.

Pour modifier un établissement, l’utilisateur doit d’abord sélection cet établissement dans la liste et ensuite cliquer sur le bouton « Modifier Établissement ». La fenêtre suivante apparaît.

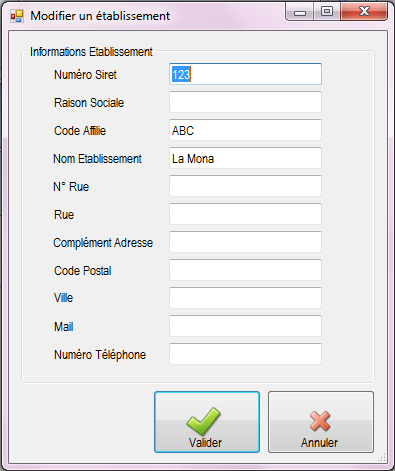


Image . : Fenêtre Modification d'un Établissement

L’utilisateur peut modifier tous les champs y compris, le numéro Siret, le code Affilié et le nom de l’établissement. Par contre s’il efface l’un de ces champs sans le remplir à nouveau, un message d’erreur sera affiché et il devra recommencer l’opération.

En sélectionnant un établissement puis cliquant sur le bouton « Modifier Assurance », l’utilisateur accède à la fenêtre suivante.

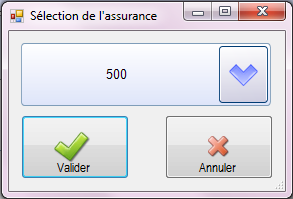


Image . : Fenêtre Modification de l'Assurance

L’utilisateur peut alors choisir dans la liste déroulante l’un des montants prédéfinis pour l’assurance. Il doit cliquer sur le bouton adéquat pour valider ou annuler l’opération.

Pour changer les informations concernant le gérant, l’utilisateur doit suivre la même démarche que pour modifier l’assurance. La fenêtre suivante apparaît.

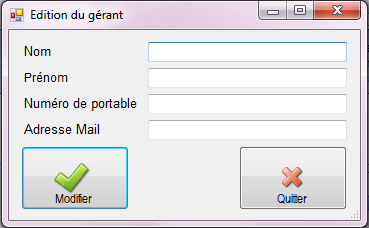


Image . : Fenêtre Modification Informations Gérant

Il remplit les champs qu’il désire puis, il valide. Aucun champ n’est obligatoire dans cette fenêtre.

#### Classes utilisées

Accueil, UserControlListeEt, UserControlInfoEt, FormAjoutEt, FormModifEt, FormAssur, FormGerantConsult

#### Algorithmes

Le contrôle utilisateur « UserControlListeEt » est utilisé pour afficher les informations concernant les établissements. Il contient un autre contrôle utilisateur (« Usercontrol\_FondEtirableEt ») qui est une liste de « UserControlInfosEt ».

Pour changer l’assurance par exemple, il faut tout d’abord récupérer l’établissement sélectionné par l’utilisateur, on utilise pour cela un énumérateur qui va parcourir la liste des UsersControlInfosEt afin de trouver celui qui est sélectionné. Ensuite, on crée une « FormAssur » en lui passant comme paramètre le numéro Siret de l’entreprise concernée. On s’assure enfin de mettre à jour les informations affichées sur le UserControlListeEt afin que les modifications effectuées par l’utilisateur soit visibles immédiatement.

System.Collections.IEnumerator enu = this.userControl\_FondEtirableEt.Controls.GetEnumerator();

int cpt = 0;

while (enu.MoveNext() && ((UserControlInfosEt)enu.Current).getSelected()==false)

{

cpt++;

}

if (((UserControlInfosEt)enu.Current).getSelected() == true)

{

FormModifEt fme = new FormModifEt(this,((UserControlInfosEt)enu.Current).getNumSiret());

fme.ShowDialog();

}

else

{

MessageBox.Show("Vous n'avez pas sélectionnez d'établissement");

}

this.afficherListeEt();

### Gestion des bordereaux

#### Attributs

La vue bordereaux dispose de plusieurs attributs importants.

private ControleurBordereau ctrl; // Utilisé pour les actions sur les bordereaux

public ControleurTitre ctrlT; // Utilisé pour les actions sur les titres restaurant

private String numSiret; // Le numéro SIRET de l'établissement courant

public Bordereau[] tabB; // Tableau des bordereaux de l'établissement

int deb = 0; // Rang du premier bordereau de tabB affiché dans la liste des bordereaux

int pas = 5; // Nombre de bordereaux affichés dans la liste

Bordereau bordActif; // Le bordereau sélectionné (dont le détail est affiché)

public Titre[] tabTitre; // Le tableau des titres du bordereau, ou tous les titres l'établissement, ou tous les titres libres de l'établissement (selon le choix de l'utilisateur)

Cette vue est également munie de méthodes pour gérer son affichage, notamment nettoyerTout qui réinitialise l’affichage des informations du bordereau, et remplirListeGauche qui met à jour le contenu de la liste des bordereaux.

#### Fonctionnement

La gestion des bordereaux est le cœur même du logiciel. En effet, le but de cette application est d’enregistrer rapidement les titres restaurant perçus et de créer les bordereaux et les imprimer sans perdre de temps à les manipuler manuellement.

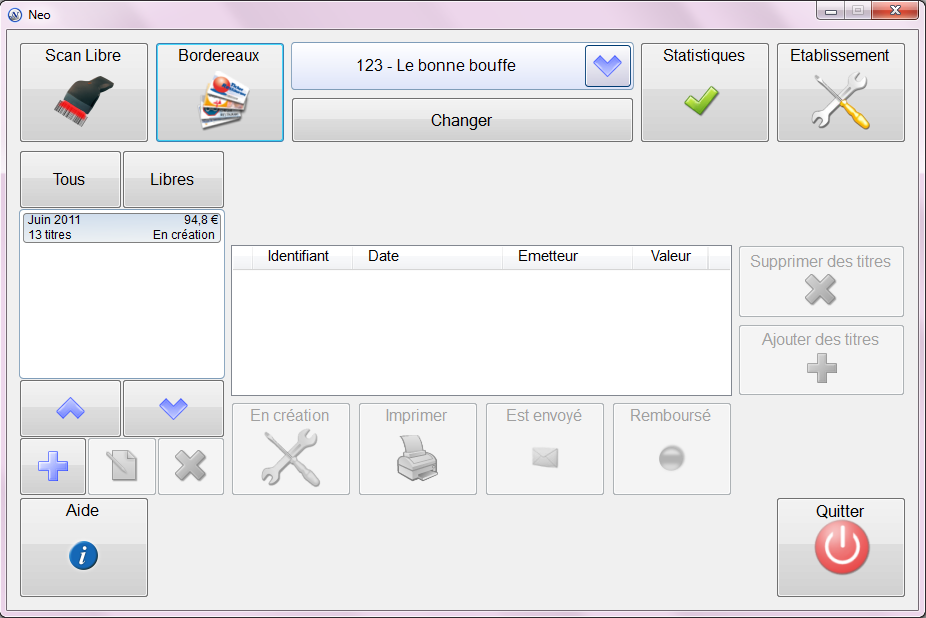


Image . : Fenêtre Bordereau

#### Création d’un bordereau

##### Fonctionnement

A partir de la vue « Bordereaux », l’utilisateur peut créer un nouveau bordereau en cliquant sur le bouton « + » sous la liste des bordereaux. La fenêtre ci-dessous s’ouvre pour en saisir le nom. Si celui-ci est inédit, le bordereau est créé dans l’état « En création ».

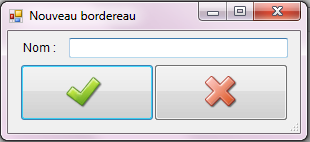


Image . : Fenêtre Ajout Bordereau

##### Classes utilisées

Accueil, UserControlVueBordereau, ControleurBordereau, Bordereau, DataBordereau, FormAjoutBordereau

##### Algorithmes

En cliquant sur le bouton « + », une fenêtre FormAjoutBordereau s’ouvre. Cette fenêtre permet de saisir le nom du bordereau. En validant la saisie, la méthode suivante est appelée.

private void bt\_valider\_Click(object sender, EventArgs e)

{

Bordereau bord = new Bordereau(this.tb\_nomBordereau.Text, this.parent.getNumSIRET());

if (ctrl.existeBordereau(bord.getIdBordereau()) == false)

{

this.ctrl.ajouterBordereau(bord);

this.Close();

this.parent.setBordereauActif(bord);

this.parent.nettoyerTout();

this.parent.remplirListeGauche();

}

else

{

MessageBox.Show("Erreur, un bordereau porte déjà ce nom !", "Nom existant", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

Lors de la validation, un bordereau est instancié. On vérifie ensuite qu’il n’existe pas dans la base de données. S’il est inédit, il est inséré en base de données avec la méthode ajouterBordereau de la classe ControleurBordereau. On définit ensuite le bordereau créé comme étant le bordereau actif de la vue bordereau que l’on met à jour avec nettoyerTout et remplirListeGauche.

Si l’ajout du borderau se passe mal, un message est affiché à l’utilisateur.

#### Affichage d’un bordereau

Lorsque l’utilisateur sélectionne un bordereau dans la liste, toutes ses informations sont affichées dans la fenêtre. Certaines informations ne sont affichées que si le bordereau est dans un certain état. Par exemple, la date de remboursement n’est affichée que si le bordereau a pour état « Remboursé ».

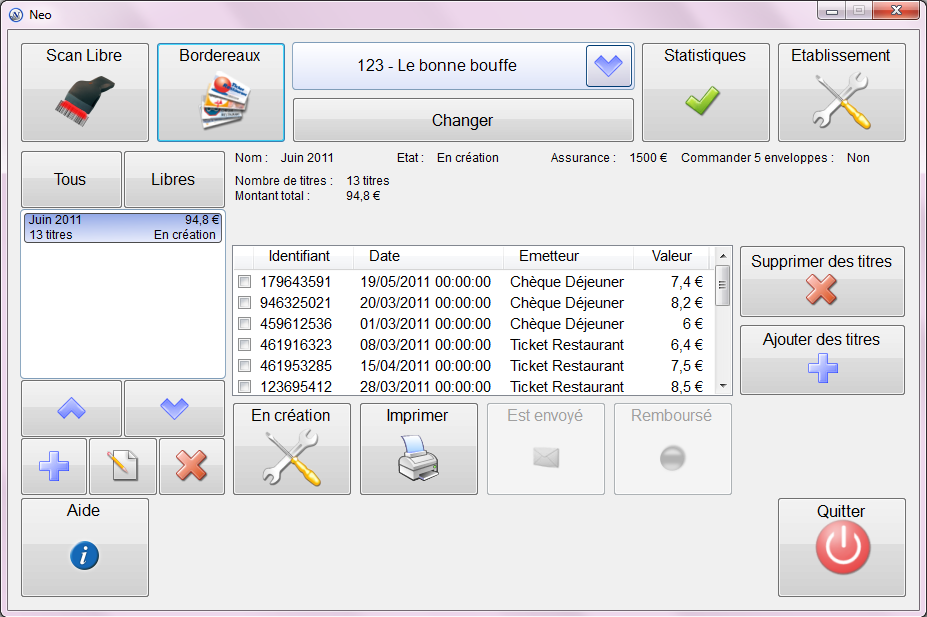


Figure . : Fenêtre Détails d'un bordereau

#### Modification d’un bordereau

##### Fonctionnement

Un bordereau n’est modifiable que s’il est en création.

En cliquant sur le bouton d’édition en dessous de la liste des bordereaux, la fenêtre ci-dessous s’ouvre permettant de modifier le nom du bordereau et de choisir ou non de commander 5 enveloppes (cette information est présente sur les bordereaux de la CRT par une case à cocher).

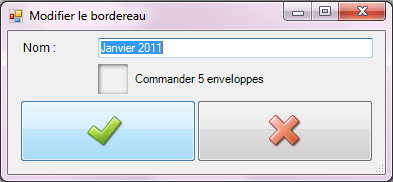


Image . : Fenêtre Modification d'un Bordereau

##### Classes utilisées

FormModifBordereau, ControleurBordereau, Bordereau, DataBordereau

##### Algorithmes

En validant la saisie, la méthode suivante est appelée.

private void bt\_valider\_Click(object sender, EventArgs e)

{

Boolean ok = true;

if (this.tb\_nom.Text != bord.getIdBordereau())

{

//MessageBox.Show("Le nom a changé !");

if (ctrlB.existeBordereau(this.tb\_nom.Text))

{

MessageBox.Show("Ce nom de bordereau existe déjà !");

ok = false;

}

else

{

ok = this.ctrlB.updateNomBordereau(bord.getIdBordereau(), this.tb\_nom.Text);

}

}

if(ok)

{

bord.setIdBordereau(this.tb\_nom.Text);

bord.setCommanderEnveloppes(this.cb\_commander.Checked);

ctrlB.updateBordereau(bord);

this.parent.nettoyerTout();

this.parent.tabB = ctrlB.getAllBordereauFromEtablissement(this.parent.getNumSIRET());

this.parent.remplirListeGauche();

this.Close();

}

}

Si le nom du bordereau a été modifié, on vérifie que le nouveau n’est pas déjà utilisé. S’il est libre, on met à jour le bordereau avec ce nouveau nom.

On met ensuite à jour le bordereau avec les méthodes setIdBordereau et setCommanderEnveloppes de la classe Bordereau, puis on appelle updateBordereau de ControleurBordereau . Enfin, on met à jour la vue bordereau.

#### Ajout de titres à un bordereau

##### Fonctionnement

Il n’est possible d’ajouter des titres à un bordereau que s’il est en création.

Pour supprimer des titres d’un bordereau, l’utilisateur les sélectionne en cliquant dans la liste des titres (les titres à supprimer doivent être cochés) puis clique sur le bouton « Supprimer des titres » à droite de cette liste. Un message pop-up demande alors s’il faut conserver les titres à l’état libre (non liés à un bordereau) ou bien les supprimer. Selon le choix de l’utilisateur, les titres sont ensuite désaffectés ou supprimés.

Pour en ajouter, il suffit de cliquer sur le bouton « Ajouter des titres ». Une fenêtre offre 3 possibilités :

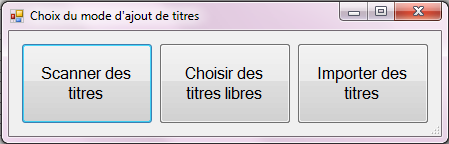
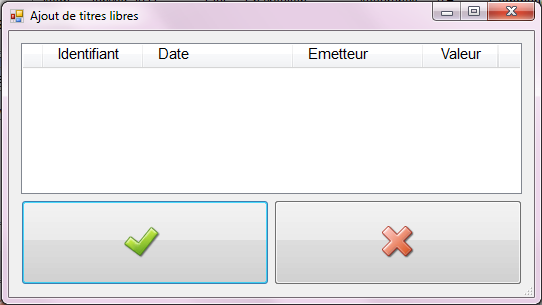


Image . : Fenêtre choix d’Ajout de Titres

* Scanner des titres – Cela ouvre la fenêtre de scan pour enregistrer de nouveaux titres qui seront automatiquement liés au bordereau.
* Choisir des titres libres – Cela ouvre une fenêtre affichant tous les titres sans bordereau. L’utilisateur sélectionne les titres à lier et valide sa sélection.



* Importer des titres – Cela ouvre une fenêtre de connexion à NeoResto permettant d’en importer des titres, enregistrés entre deux dates définies par l’utilisateur.

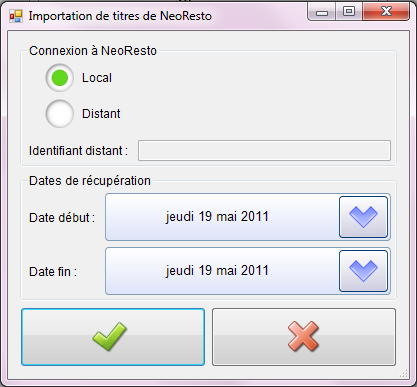


Image . : Importation des Titres de NeoResto

##### Classes utilisées

FormChoixAjoutBordereau, FormScanBordereau, FormAjoutTitresLibres, FormImportNeoResto, Bordereau, ControleurBordereau, Titre, ControleurTitre, DataTitre

##### Algorithmes

###### Scanner des titres

Ce bouton ouvre la fenêtre de scan de nouveau titre (cf.4.8.8 Lecture des codes-barres).

###### Choisir des titres libres

Cette fenêtre contient une liste affichant tous les titres sans bordereau de l’établissement courant. Lorsque l’utilisateur confirme sa sélection, la méthode suivante est appelée.

private void bt\_valider\_Click(object sender, EventArgs e)

{

System.Windows.Forms.ListView.CheckedListViewItemCollection items = this.ls\_titres.CheckedItems;

IEnumerator enu = items.GetEnumerator();

while (enu.MoveNext())

{

this.ctrlT.affecterTitre(((ListViewItem)enu.Current).Text, this.bord.getIdBordereau());

}

this.Close();

}

On commence par récupérer tous les items sélectionnés dans la liste. On parcourt ensuite cette liste avec un énumérateur (boucle tant que enu.MoveNext()) pour affecter chaque titre au bordereau à l’aide de la méthode affecterTitre de la classe ControleurTitre.

###### Importer des titres

Cette fenêtre permet de se connecter à NeoResto et de récupérer les titres scannés entre 2 dates. Ces données sont ensuite récupérées dans un DataSet avec la méthode getTR\_NeoResto.

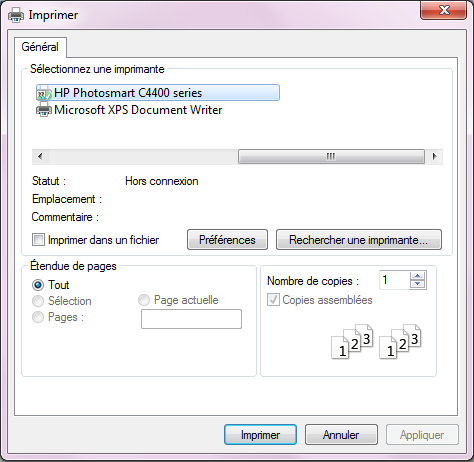
On crée ensuite les titres, qui sont insérés en base de données comme des nouveaux titres.

#### Changement d’état d’un bordereau

##### Fonctionnement

Un bordereau peut être dans 4 états différents :

* En création : le bordereau a été créé, il est encore possible de changer ses informations et de modifier les titres attachés.
* Imprimé : le titre a été imprimé, plus aucune modification n’est possible à partir de cet état ; cependant, il est possible de revenir dans l’état « En création » à tout moment. En cliquant sur le bouton « Imprimer », une fenêtre s’ouvre pour gérer l’impression du bordereau puis l’état est passé à « Imprimé ».



* Envoyé : le bordereau a été envoyé à la CRT, une date d’envoi est définie.
* Remboursé : le remboursement pour ce bordereau a été reçu de la CRT, les informations relatives à ce remboursement (le montant remboursé et la raison d’un éventuel écart) sont indiquées à l’aide de la fenêtre qui s’ouvre en cliquant sur le bouton « Remboursé ».

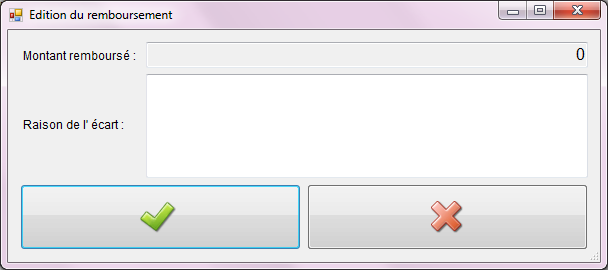


Image . : Fenêtre Edition du remboursement

Le passage d’un état à l’autre se fait séquentiellement dans cet ordre. Cependant, il est possible de revenir à un état précédent à tout moment.

##### Classes utilisées

UserControlVueBordereau, Bordereau, DataBordereau, ControleurBordereau

##### Algorithmes

Le changement d’état ne contient pas d’algorithmes compliqués. Il suffit de mettre à jour l’état du bordereau avec la méthode updateBordereau de ControleurBordereau.

#### Suppression d’un bordereau

##### Fonctionnement

À tout moment, l’utilisateur peut supprimer en bordereau en le sélectionnant dans la liste et en cliquant sur le bouton X. Un message demande alors s’il faut conserver libres les éventuels titres associés ou bien les supprimer. Selon le choix, les titres sont désaffectés ou supprimés, puis le bordereau définitivement éliminé.

##### Classes modifiées

UserControlVueBordereau, Bordereau, DataBordereau, ControleurBordereau

##### Algorithmes

L’algorithme suivant gère la suppression d’un bordereau.

if (bordActif != null)

{

DialogResult res = MessageBox.Show("Etes-vous sur de vouloir supprimer ce bordereau ?", "Confirmer suppressions", MessageBoxButtons.YesNo, MessageBoxIcon.Exclamation);

if (res == DialogResult.Yes)

{

DialogResult resTitre = MessageBox.Show(" Voulez-vous conserver les titres restaurant liés à ce bordereau ?", "Suppression des titres restaurant", MessageBoxButtons.YesNo, MessageBoxIcon.Question);

if (resTitre == DialogResult.Yes)

{

ctrlT.desaffecterTitreFromBordereau(this.bordActif);

}

else

{

ctrlT.supprimerTitreFromBordereau(this.bordActif);

}

ctrl.supprimerBordereau(this.bordActif);

this.bordActif = null;

this.tabB = this.ctrl.getAllBordereauFromEtablissement(this.numSiret);

this.nettoyerTout();

this.remplirListeGauche();

}

}

Il ne s’exécute que s’il y a un bordereau actif. Une boite de dialogue demande s’il faut conserver les titres du bordereau à l’état libre ou bien les supprimer. S’il souhaite les conserver, on appelle la méthode desaffecterTitresFromBordereau, sinon on appelle supprimerTitresFromBordereau. Ces deux méthodes sont dans la classe ControleurTitre. On supprime ensuite le bordereau (possible car il n’a plus de titres associés, pas de problème de contraintes de référence), on définit le bordereau actif à nul et on rafraichit la vue bordereau.

### Statistiques

#### Fonctionnement

L’application présente des statistiques simples calculées à partir des informations enregistrées dans la base de données.

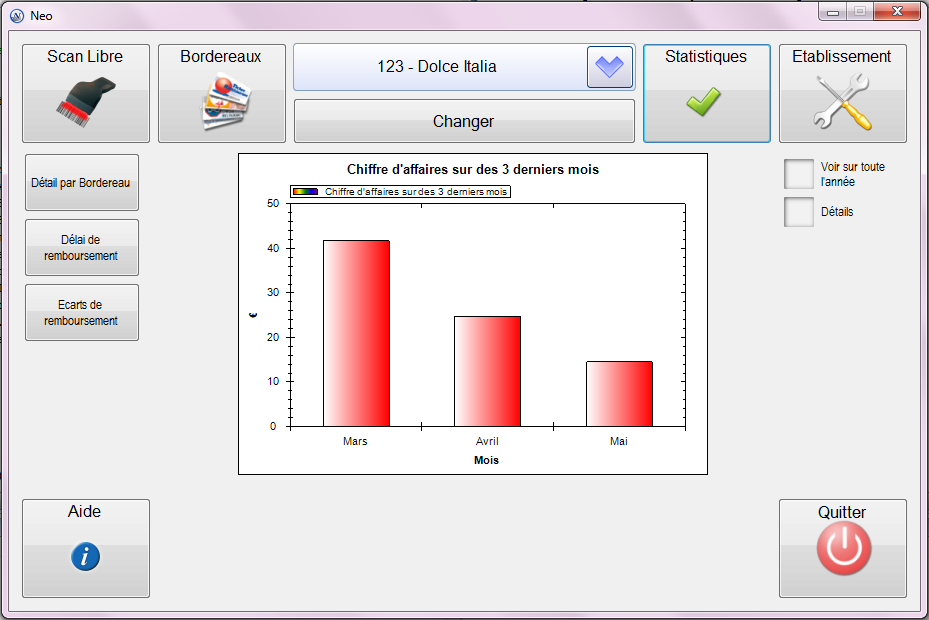


Image . : Vue Statistiques

Par défaut, le chiffre d’affaires des trois derniers mois de l’établissement courant est affiché (dans l’exemple c’est le « Dolce Italia »). Il est possible d’obtenir le chiffre d’affaires sur toute l’année et/ou le chiffre d’affaire ventilé par type de titre restaurant.

D’autres informations plus détaillées sont disponibles : la répartition du chiffre d’affaires par type de titres pour un bordereau choisi par l’utilisateur, le délai de remboursement moyen sur une période donnée et l’écart de remboursement sur chaque mois.

#### Classes utilisées

UserControlStats, UserControlStatBord, FormListeBord, UsercontrolRemb, UserControlDelai

#### Algorithmes

Afin de pouvoir modéliser les informations dans la base de données sous forme de diagrammes nous avons utilisé la bibliothèque ZedGraph. Afin de pouvoir utiliser les éléments de la bibliothèque nous avons ajouté à la boîte à outil de l’environnement VisualStudio le fichier « ZedGraph.dll » puis nous avons ajouté ce même fichier dans les références de notre projet.

Cette librairie est simple d’utilisation, par exemple pour afficher l’histogramme du chiffre d’affaires sur les 3 derniers mois on doit suivre les étapes suivantes.

On commence par donner un titre au graphique et nommer les axes.

zgc.GraphPane.Title.Text = "Chiffre d'affaires sur des 3 derniers mois";

zgc.GraphPane.XAxis.Title.Text = "Mois";

zgc.GraphPane.YAxis.Title.Text = "€";

Puis on récupère les titre sur les trois derniers mois puis pour chauqe mois on calcule le chiffre d’affaires.

Titre[] titres = this.CT.getTitresPeriode(this.Acc.getNumSiret(), DateTime.Now);

double mois1 = 0;

double mois2 = 0;

double mois3 = 0;

String[] infosDate = DateTime.Now.ToString().Split('/');

int moisCourant = Convert.ToInt32(infosDate[1]);

String lib1 = Date.getLibelleMois(((moisCourant - 2) % 12).ToString());

String lib2 = Date.getLibelleMois(((moisCourant - 1) % 12).ToString());

String lib3 = Date.getLibelleMois(moisCourant.ToString());

if (titres !=null)

{

for (int i = 0; i < titres.Length; i++)

{

String[] date = titres[i].getDate().Split('/');

if (Convert.ToInt32(date[1]) == (moisCourant - 2) % 12)

{

mois1 += Titre.getValeur(titres[i].getCodeBarre());

}

if (Convert.ToInt32(date[1]) == (moisCourant - 1) % 12)

{

mois2 += Titre.getValeur(titres[i].getCodeBarre());

}

if (Convert.ToInt32(date[1]) == moisCourant)

{

mois3 += Titre.getValeur(titres[i].getCodeBarre());

}

}

}

Ensuite on ajoute à une liste de points, le numéro du mois et la valeur du chiffre d’affaires pour ce mois.

PointPairList ppl = new PointPairList();

ppl.Add(1, mois1);

ppl.Add(2, mois2);

ppl.Add(3, mois3);

On crée une liste de labels afin d’affiché le libellé de chaque mois sous l’axe. Puis on crée les barres que l’on va ajouter à la zone de dessin (« GraphPane ») du ZedGraphControl (zgc) présent dans le contrôle utilisateur « UserControlStats ».

String[] labels = { lib1, lib2, lib3 };

BarItem myCurve = zgc.GraphPane.AddBar(zgc.GraphPane.Title.Text, ppl,Color.Blue);

Ensuite on donne la possibilité d’afficher des String sur l’axe des abscisses. Puis on indique quels labels afficher sur cet axe et on change le type de l’axe des abscisses qui par défaut est un type numérique en type Text. Enfin on paramètre l’histogramme en définissant des couleurs, le maximum et le minimum des axes. Afin d’avoir un paramétrage automatique des axes on utilise la méthode « AxisChange ».

this.zedGraphControl1.GraphPane.XAxis.MajorTic.IsBetweenLabels = true;

this.zedGraphControl1.GraphPane.XAxis.Scale.TextLabels = labels;

this.zedGraphControl1.GraphPane.XAxis.Type = AxisType.Text;

Color[] colors = { Color.Red, Color.Yellow, Color.Green, Color.Blue, Color.Purple };

myCurve.Bar.Fill = new Fill(colors);

myCurve.Bar.Fill.Type = FillType.GradientByZ;

myCurve.Bar.Fill.RangeMin = 0;

myCurve.Bar.Fill.RangeMax = 4;

zgc.AxisChange();

## Tests

Nous avons séparé le développement en fonction des différentes fonctionnalités. A chaque fois qu’une fonctionnalité était réalisée elle faisait l’objet de tests unitaires afin de vérifier si les résultats obtenus étaient corrects.

Lorsqu’une fonctionnalité était terminée nous l’intégrions au reste de l’application en s’assurant que l’intégration ne provoquait pas de régression des autres fonctionnalités déjà intégrées.

Pour réaliser nos tests, nous avons élaboré un plan de tests qui se trouve dans les annexes de ce rapport.

## Limites

L’application que nous avons réalisée ne fonctionnera que pour des titres du millésime 2011 car les contrôles et les calculs effectués sur le code barre ne sont valables que pour ce millésime et sont susceptibles de changer les années suivantes. Cependant, nous avons commenté le code et tenté de rendre l’application la plus maintenable possible. Donc il sera aisé de changer le code à modifier ; de plus, il n’y a que deux classes qui seront concernées.

## Améliorations possibles

### Impression d’un bordereau complet

Le logiciel permet actuellement de remplir un bordereau en imprimant directement les informations utiles sur un bordereau officiel de la CRT. Une amélioration possible serait d’offrir la possibilité d’imprimer un bordereau complet sur une page vierge. Il ne serait ainsi plus utile de se procurer des pré-imprimés.

### Sauvegarde de la BD

La version embarquée de Firebird enregistre toutes les données d’une base de données dans un fichier unique d’extension « .fdb ». Une suppression malencontreuse de ce fichier provoquerait la perte de toutes les données de l’application.

Il faudrait donc donner la possibilité à l’utilisateur d’effectuer des sauvegardes à la demande ou régulièrement de sa base de données afin d’éviter ce problème. Une interface pour restaurer les différentes versions de la base de données devrait également être envisagée.

### Design des fenêtres

Nous nous sommes efforcés tout au long de ce projet de rendre les visuels agréables en utilisant des images fournies par notre demandeur. Cependant, une mise en forme plus complète rendrait le logiciel plus attractif pour l’utilisateur, et assurerait une cohérence avec le logiciel NeoResto.

# Conclusion

Ce projet est le premier que nous avons eu à mener pour une entreprise. Durant ces 7 semaines entièrement consacrées à ce projet, nous avons pu mettre en œuvre une **méthodologie complète**, en planifiant nos tâches et en organisant notre temps afin de répondre aux attentes du demandeur.

Cette relation directe avec le client nous a permis de rapidement identifier les problèmes et les modifications à apporter. Nous avons également réalisé qu’il est très difficile de mettre en évidence toutes les attentes de la maîtrise d’ouvrage dans un premier temps, et que cela provoque bien souvent quelques remises en question au cours du projet.

Les échanges fréquents avec M. Baïla ont permis une grande réactivité. Il nous a également transmis quelques connaissances au niveau de l’ergonomie des IHM tactiles et du framework .NET.

Au niveau technique, nous avons dû maitriser l’environnement de travail de Microsoft et ses différents composants que sont Visual Studio et le langage objet C#. Nous avons également appréhendé les spécificités liés au développement d’une application destinée à des interfaces tactiles.

Tout au long de ce projet, nous avons pu être **force de proposition**, notamment sur le choix des statistiques à proposer et sur certaines fonctionnalités du logiciel.

Grâce à ce projet, nous avons pu nous rendre compte qu’il est important de se mettre dans la peau de l’**utilisateur**, d’adopter son **point de vue** afin d’éviter certaines erreurs d’ergonomie. En tant que concepteur, nous avons toujours l’impression que le produit est facile à utiliser, alors qu’une personne non initiée peut se sentir rapidement perdue et submergée d’informations, ou alors incapable de trouver la bonne interface pour faire ce qu’il veut. En essayant de se mettre à sa place, on évite la plupart de ces écueils, et c’est ce que ce projet nous a également apporté.

Nous pouvons dire que nous avons **atteint les objectifs définis** dans la lettre de mission. En outre, nous avons identifié quelques améliorations qu’il serait intéressant de mettre en œuvre afin de perfectionner le logiciel.

Ce travail de groupe de 7 semaines a été très enrichissant tant sur le plan humain que sur le plan professionnel.

# Annexe

# Tests unitaires

## Fenêtre principale

La fenêtre principale permet d’accéder à toutes les fonctionnalités de l’application, elle contient les boutons qui permettront de changer les User Control qui contiennent les différentes fonctionnalités. Le premier test à effectuer est de vérifier que les boutons donnent accès aux bons user controls.

Avec cette application, il est possible de gérer les titres de plusieurs établissements, il faudra donc s’assurer que les informations saisies par l’utilisateur soient reliées effectivement à un établissement. Il faudra également vérifier que les informations enregistrées sont associées au bon établissement.

## Consultation des bordereaux

Ce user control permet de lister tous les bordereaux enregistrés dans la base de données. Il faudra donc vérifier que tous les bordereaux présents dans la base de données sont bien affichés et que les filtres retournent les bonnes informations.

On devra s’assurer que le bouton « ajout bordereau » revoie sur le bon user control c’est-à-dire le user control Bordereau2.

# Jeu d’essai

|  |  |
| --- | --- |
| n° | Code barre |
| 1 | 123456789120062014304901 |
| 2 | 388663300730074011404991 |
| 3 | 567454636230102026210121 |
| 4 | 123456789120062014304901 |
| 5 | 560727392440042034408320 |
| 6 | 809271593060058783002511 |
| 7 | 513835370067511304901 |
| 8 | 926393648670062038045621 |
| 9 | 567895410550120038510101 |
| 10 | 045303603840099928902511 |

## Scanner des titres

Les 3 premiers titres doivent être enregistrés normalement.

Le 4ème est valide, mais est identique au premier. Un message d’erreur doit être affiché et le titre refusé, puisqu’il est déjà traité.

Le 5ème n’est pas du millésime 2011, mais 2010. Cependant, il respecte les normes du millésime 2011, et doit donc être rejeté.

Le 6ème à une clé de contrôle erronée. Elle a pour valeur 30 au lieu de 57. Un message d’erreur doit le signaler et le titre doit être refusé.

Le 7ème ne respecte pas la norme du millésime 2011, il n’a que 21 caractères au lieu des 24 requis. Un message d’erreur doit être affiché et le titre refusé.

## Création d’un bordereau

Créer un nouveau bordereau et le nommer « Bordereau de test scan».

Cliquer sur « Ajouter des titres » et choisir « Scanner des titres ». Scanner les titres 1 à 7, qui doivent être normalement tous refusés. Les 3 premiers doivent l’être car ils sont déjà enregistrés. Les 4 suivants pour les raisons cités ci-dessus. Scanner alors les 3 derniers titres, qui doivent être acceptés.

Le bordereau doit alors contenir 3 titres, pour une valeur totale de 28 € 19.

Créer un nouveau bordereau et le nommer « Bordereau de test affectation ».

Cliquer sur « Ajouter des titres » et choisir « Affecter des titres ». Choisir les 3 titres sans bordereaux.

Le bordereau doit alors contenir 3 titres, pour une valeur totale de 23 € 80.

# Installer Firebird et l’intégrer à Visual Studio 2008

## Prérequis

* Visual Studio 2008 doit être installé
* Dézipper le « Package Firebird Pour VS2008 »
* Faire un backup de la base de registre, on ne sait jamais

## 

## Installer la dernière version de Firebird

Installer la version la plus récente de Firebird. Suivre les instructions d’installation.

## Lancer « NETProvider-2.6.0.msi »

Normalement, il doit y avoir deux répertoires :

* C:\Program Files\Firebird
* C:\Program Files\FirebirdClient

## Copies

Copier le contenu de « NETProvider-2.6.0 » dans C:\Program Files\FirebirdClient

Normalement, cette étape est inutile.

Copier les fichiers « FirebirdSql.VisualStudio.DataTools.dll » et « FirebirdSql.VisualStudio.DataToolsUI.dll » du répertoire « FirebirdDDEXProvider-2.0.5 » dans C:\Program Files\FirebirdClient\DDEXProvider

## Configuration du GAC

Start > Programs > Microsoft Visual Studio 2008 > Visual Studio Tools > Invite de commande de Visual Studio

cd "C:\Program Files\FirebirdClient"

gacutil /i FirebirdSql.Data.FirebirdClient.dll

Normalement, une ligne indiquant le succès de l’ajout doit apparaitre, sinon se référer au message d’erreur.

gacutil /l FirebirdSql.Data.FirebirdClient

Noter les valeurs de Version, Culture et PublicKeyToken

cd DDEXProvider

gacutil /i FirebirdSql.VisualStudio.DataTools.dll

Un message doit valider l’ajout.

## Machine.config

Editer le fichier : « C:\Windows\Microsoft.NET\Framework\v2.0.50727\CONFIG\machine.config »

Y ajouter les lignes suivantes aux bons endroits en modifiant %Version% par la Version notée, %Culture% par la Culture notée et %PublicKeyToken% par la publicKeyToken notée. Ne pas changer ces valeurs pour la première ligne !

<configuration>

<configSections>

...

<section name="firebirdsql.data.firebirdclient" type="System.Data.Common.DbProviderConfigurationHandler, System.Data, Version=2.0.0.0, Culture=neutral, PublicKeyToken=3750abcc3150b00c"/>

...

</configSections>

...

<system.data>

<DbProviderFactories>

...

<add name="FirebirdClient Data Provider" invariant="FirebirdSql.Data.FirebirdClient" description=".Net Framework Data Provider for Firebird" type="FirebirdSql.Data.FirebirdClient.FirebirdClientFactory, FirebirdSql.Data.FirebirdClient, Version=**%Version%**, Culture=**%Culture%**, PublicKeyToken=**%PublicKeyToken%**"/>

...

</DbProviderFactories>

</system.data>

...

</configuration>

## Base de registre

Dans le package, aller dans FirebirdDDEXProvider-2.0.5 > reg\_files > VS2008 et éditer « FirebirdDDEXProvider32 » (ou 64 pour les architectures 64 bits).

Remplacer les occurrences de %PATH% par "C:\\Program Files\\FirebirdClient\\DDEXProvider" (sans les quotes).

Exécuter le fichier reg.

## Fin

Maintenant, Firebird doit apparaitre dans les sources de connexion !

# Résumé

L’entreprise Neorizon est une société proposant des solutions afin de gérer des établissements de type café, hôtel ou restaurant. Leur logiciel ne proposait pas d’outils pour la gestion des titres restaurant qui est la cause d’une grande perte de temps pour la profession.

Dans le cadre de notre projet industriel, nous avions pour mission de réaliser une application permettant de pallier ce problème de gestion. L’objectif principal de ce logiciel était de pouvoir automatiser le traitement des titres et l’élaboration des bordereaux de remboursement. Pour commencer, nous avons effectué une analyse de l’existant en utilisant la démarche Merise puis nous avons conçu l’application en la modélisant à l’aide des diagrammes UML. Nous avons adopté une démarche itérative pour le développement afin d’être le plus réactif possible.

Les objectifs que nous nous sommes fixés ont été atteints, toutefois il reste possible d’apporter des améliorations au niveau de la sauvegarde de la base de données, des options d’impression et de la charte graphique.

# Summary

Neorizon is a small company which sells computer equipment and offers the software NeoResto. This software allows to restaurant owners to manage their whole establishment but it doesn’t really handle restaurant ticket. The management of these restaurant tickets is an actual waste of time for this profession.

On the occasion of the industrial project, we had to make software which compensates this management problem. Our objective was to automate the sheet drafting. Firstly we carry out the environment analysis using some elements of the Merise method. Then we design the application with Unified Modeling Language. We adopt an iterative approach for the development part in order to be as responsive as possible.

We reach the goal we have fixed, however the software can be improved in particular on the database maintenance, on the printing options and on the graphical user interface design.

1. Framework (cadre d’applications en français) : kit de [composants logiciels](http://fr.wikipedia.org/wiki/Composant_logiciel) structurels utilisé pour modeler l’architecture des logiciels applicatifs. [↑](#footnote-ref-1)
2. Intégrité référentielle : Mécanisme participant à assurer la cohérence d’une base de données en vérifiant que toute information d’une table A faisant référence à une information d’une table B existe bien dans la table B. Par exemple, un livre ne pourra être associé à un auteur qui n’existe pas. [↑](#footnote-ref-2)
3. UserControl : Ensemble de contrôles autonomes pouvant être inséré dans une fenêtre comme un contrôleur de base [↑](#footnote-ref-3)