Stage de fin d'étude réalisé dans l'entreprise

Le Loft

50, rue des couteliers
03000 MOULINS
www.leloft-online.com

sur le projet

xStoRe Corporate Management Software

Gestion intégrée d'entreprises

Déploiement de

terminaux de point de vente

sous UNIX

Du 15 avril au 21 juin 2002

I.U.T. Blaise PASCAL
Av. Aristide BRIAND
03100 MONTLUÇON

département
Génie Électrique et Informatique Industrielle
option Réseaux Locaux Industriels

2ème année
Année scolaire
2001 - 2002

Stage de fin d'étude réalisé dans l'entreprise

Le Loft

50, rue des couteliers
03000 MOULINS
www.leloft-online.com

sur le projet

xStore Corporate Management Software

Gestion intégrée d'entreprises

Déploiement de

terminaux de point de vente

sous UNIX

Du 15 avril au 21 juin 2002

I.U.T. Blaise PASCAL
Av. Aristide BRIAND
03100 MONTLUÇON

département
Génie Électrique et Informatique Industrielle
option Réseaux Locaux Industriels

2ème année
Année scolaire
2001 - 2002

Remerciements

Je tiens à adresser mes remerciements aux personnes qui m'ont permis d'effectuer ce stage sur un point aussi sensible que l'encaissement au sein de l'entreprise *Le Loft*.

Je tiens tout particulièrement à *remercier*:

- *Jean-Claude RICHARD*, concepteur et développeur de *XStore*, mon maître de stage, pour ses explications sur le concept technique de XStore, ses cours et rappels de cours en programmation.

- *Laurent BOURBON*, agent commercial spécialisé dans le déploiement de terminaux de point de vente, pour ses explications sur le concept général de *XStore* et les objectifs à atteindre.

- *Baptiste WITRAN*, développeur, pour ses aides quant au « debugging » du module d'encaissement.

- *Thierry*, *Jean-Michel* et *Nicolas*, agents commerciaux de *Cont@ctis*, entreprise spécialisée dans l'intégration de terminaux de point de vente, pour leur accueil sur Paris et leurs conseils techniques.

Résumé

Dans un premier temps, j'ai développé le pilote d'impression « DirectPrint » capable d'imprimer les pièces de caisse (tickets), les factures, les chèques, et de contrôler le viseur (l'afficheur à leds).

Puis j'ai développer le système de gestion des pièces de caisse via les périphériques d'entrées / sorties (scanner de codes barres / imprimante et viseur) et la base de données.

Le gestionnaire doit :

- gérer la mise en correspondance de l'EAN (code barres) et du produit
- récupérer le prix unitaire et lui appliquer la TVA si nécessaire
- imprimer l'entête du ticket puis la quantité, le nom et le prix des produits achetés et enfin le pied de ticket sur la pièce de caisse
- afficher la quantité, le nom et le prix du dernier produit « scanné » et le sous-total sur le viseur
- gérer le total avant remise
- gérer les différents types de remises en fin de ticket et les modes de paiement puis le total.
- imprimer le détail et le total des différents types de TVA appliqués aux produits achetés.

Summary

Initially, I have developed the printer driver "DirectPrint" able to print tickets, invoices and checks, and to control the display.

Then I have developed ticket manager via the inputs / outputs devices (bar codes scanner / printer and display) and database.

The manager have :

- to manage the mapping of the EAN (bar code) and the product
- to recover the unit price and to apply the VAT if necessary
- to print the header then the quantity, the name and the price of the bought products then the footer on the ticket
- to display the quantity, the name and the price of the last product scanned and the sub-total on display
- to manage the total before savings
- to manage the various types of savings at the end of the ticket, the payment mode and the total
- to print the detail and the total of various types VAT applied to the bought products

Sommaire

Introduction

Ce rapport est l'aboutissement de 10 semaines de stage de fin d'études effectuées au sein de l'entreprise *Le Loft*. En collaboration, mon *maître de stage Jean Claude RICHARD*, développeur, *Baptiste WITRAN*, également développeur, *Florian BARDET* (stagiaire), et moi avons créé et développé le *module d'encaissement* de *XStore*, projet auquel l'entreprise consacre maintenant tout son temps.

Je vais vous conduire, en passant par la présentation du *projet XStore*, dans le vif de mon sujet de stage, à savoir, le développement des pilotes de périphériques de terminaux de point de vente, puis la programmation de la gestion des encaissements via ces mêmes périphériques et une base de donnée.

Avant tout ceci, je vous propose la lecture d'une brève présentation de l'entreprise *Le Loft*.

Présentation de l'entreprise

Le Loft

Créé à la fin de l'année 2000 *Le Loft* était d'abord un magasin de matériel informatique et un cyberespace hors du commun. En effet, des hordes de pingouins sous toutes les formes envahissaient les locaux de la SARL ! En effet, *Le Loft* était aussi un espace de rencontre pour les passionnés du système d'exploitation au pingouin (la mascotte nommée Tux, c'est elle qui a les numéros de page du rapport tatoués sur le ventre !), à savoir *GNU/Linux*. Le cyberespace offrait à sa clientelle un espace connecté à internet « 100% *GNU/Linux* », et bien sûr, les ordinateurs vendus dans la boutique avaient le système de Tux installé par défaut.

Parallèlement à cet occupation, *Jean Claude RICHARD* développait le logiciel de gestion d'entreprise *XStore* derrière son bureau lorsque la clientelle lui laissait un moment. Bientôt celle-ci aperçois du matériel d'encaissement apparaître dans le fond des locaux. *XStore* n'a alors que quelques pas à faire pour arriver à maturité. *Jean Claude*, en contact avec des professionnels du domaine acquière leur confiance, la course vers la version 1.0 commence. L'espace ouvert au public ferme. *Jean Claude* s'associe à *Laurent BOURBON* et crée avec quelques personnes de la société *cont@ctis* (entreprise spécialisé dans l'intégration de terminaux de point de vente) une autre société appelée *Morrigan* qui est amenée à se spécialiser dans l'édition de logiciel à commencer par *XStore*. *Le Loft* n'existera donc plus.

Présentation du projet

XStore

et de son concept

L'idée dominante du système XStore tient en une seule expression : intégration totale de services.

Lors de la phase de conception, Jean Claude RICHARD est partis d'un certain nombre de constats : absence d'un système standard autorisant à la fois une gestion d'entreprise, quelle que soit sa taille et son amplitude géographique et une souplesse d'utilisation suffisante permettant de résoudre élégamment les problèmes de communication, les résidus transactionnels et autres complexités.

Un examen attentif de systèmes existants a démontré que l'intégration est rarement réalisée, ces systèmes étant constitués de couches disparates réunies par des « moulinettes », serveurs permettant une conversion de formats, les traitements, le reporting...

De ce fait, les temps de réaction et la tenue en charge des systèmes dépend davantage de la qualité des « moulinettes », de leur fréquence d'utilisation que des performances communicantes du système par lui-même.

Une

base de données

abstraite

Par interface abstraite, nous entendons une méthode d'interrogation et de mise à jour des informations unifiée, totalement indépendante du modèle physique et de la technologie de stockage.

La base de données intègre à la fois des tables classiques, des index documentaires, du

son, et de l'image.

La base de données est conçu selon le mode client-serveur, via une classe d'abstraction Metabase. Metabase est un ensemble de modules autorisant le développement d'applications selon un modèle offrant une indépendance totale face aux bases de données. Metabase fonctionne en tant que moteur de description abstrait (classe abstraite) des modèles de données utilisés. C'est le moteur qui se charge de convertir les modèles abstraits en un ensemble de requêtes natives spécifiques à chaque gestionnaire de données. Une telle abstraction autorise une portabilité totale des applications et leur déploiement sur de multiples plateformes sans avoir à retoucher le code, les requêtes SQL (requêtes à l'intension de la base de données) ou les bibliothèques système.

Le concept d'interface unifiée n'est pas nouveau. De nombreuses tentatives ont été réalisées, par exemple Database Interface (DBI) pour le langage Perl. Toutefois, aucune API n'avait encore poussé aussi loin le degré d'abstraction jusqu'à la conceptualisation de la base et la garantie de l'unification des requêtes via un meta langage SQL (langage utilisé pour effectuer les requêtes vers une base de données).

Le transactionnel réparti

Extension au concept de base de données répartie, une transaction répartie peut être validée en n'importe quel point du réseau de traitement des données. Elle permet de notifier le début d'une manipulation d'entité en temps réel (Temps t + temps de réponse) et de gérer correctement la file d'attente des réplicats. La consultation de la donnée reste possible, ce qui offre la plus grande sécurité contre les verrous cycliques et les « étreintes fatales ». Pour mieux comprendre ce concept, voyons comment il fonctionne sur le terrain.

La répartition transactionnelle est réalisée au niveau « départemental » du système. Par niveau départemental, nous entendons un sous-ensemble cohérent, soit géographique, soit logique, composé d'un sous-serveur et d'un réseau de clients.

Le sous-serveur requiert une réplique logique du schéma de la base de données lors de sa phase d'initialisation, avec les entités correspondant aux besoins locaux. Ces entités peuvent parfaitement être l'ensemble des identifiants articles d'une base de produits, la codification EPAX

de codes clients (cartes privatives, cartes fidélité,...), la gestion temporelle de promotions ou toute autre entité voulue. Ces informations sont redistribuées aux terminaux, en mode réseau local, dans le but d'autoriser un mode de fonctionnement en isolat, y compris en cas de panne physique sur le réseau interne. Donc en cas de panne du sous-serveur, les terminaux possède une réplique des tables qui leur sont utiles de façon à devenir autonomes. Les nouveaux enregistrement dans la base sont alors stockés localement. Lors de la remise en service du sous-serveur, les nouveaux enregistrements remontent au sous-serveur, et les mises à jour des données descendent vers les terminaux. Sur des terminaux de point de vente, ces mises à jour correspondent par exemple à des changements de prix, à l'ajout de nouveaux produits ou de nouvelles remises.

Un ensemble de mécanismes (authentification transactionnelle, acquittement défini...) garantissent la sécurité des transactions et la qualité du réplicat.

Des standards ouverts

L'ensemble du système utilise, pour ses communications, uniquement des standards ouverts: HTTP et HTTPS pour les services transférables avec mise en page Web, DOM pour les descriptions de tables, XML et XSLT pour les entités, RPC pour les traitements distribués. Le format de données distribuées est très proche de celui utilisé lors des requêtes DNS.

Tous ces protocoles sont standardisés, largement documentés. Ils n'utilisent aucun format de données propriétaires.

Ces choix ont été réalisés afin de permettre aux entreprises utilisatrices une indépendance du type de système d'exploitation utilisé sur les postes clients, une conformance aux standards définis.

Le choix du langage de programmation PHP, l'un des outils vedette du monde du scripting dynamique entre dans une définition identique. Les modules d'extension PHP sont réalisés en C++ standard afin d'en garantir la portabilité.

La plate-forme serveur est Linux, l'un des rares systèmes à la fois totalement ouverts et conformes à la norme POSIX.

Présentation du

sujet de stage

de fin d'études

Le projet XStore doit intégrer un module d'encaissement pour sa version 1.0. Ce module doit être capable de gérer les ventes de produits en utilisant le matériel classique d'encaissement.

Le logiciel XStore est développé en grande partie en langage PHP. Le module d'encaissement était amené à se servir de fonctions existantes programmées dans ce langage, c'est pourquoi le stage a commencé par la familiarisation de ce langage en observant les fonctions programmées en PHP dont j'allais me servir par la suite.

Ensuite, s'en suivait l'apprentissage du fonctionnement des ports séries sous GNU/Linux et leur paramétrage.

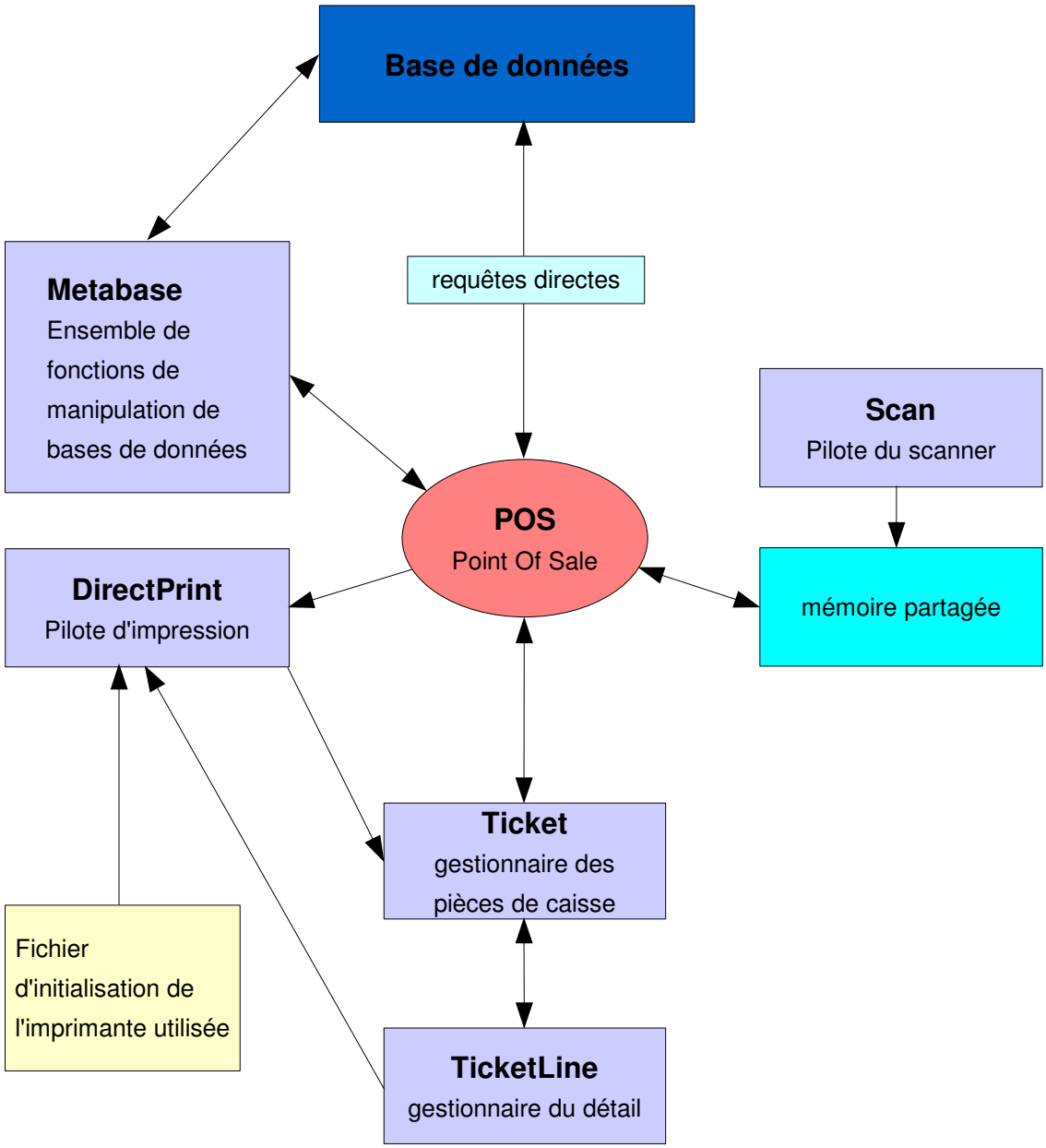
Et enfin, la mise en pratique des acquis précédents, en commençant par le développement des pilotes de périphériques (ou « drivers »).

Puis après la gestion des périphériques, leur exploitation pour la gestion des encaissements. Pendant le développement du gestionnaire d'encaissement, j'ai du apprendre à manipuler la base de données XStore sous MySQL, le système de gestion de base de données, et à utiliser les fonctions de Metabase qui permettent de manipuler cette même base.

Lançons-nous sans plus attendre dans l'organisation du module d'encaissement.

Organisation de la gestion des terminaux de

point de vente



Organisation simplifiée du module d'encaissement

Voici ci-dessus, l'organisation générale du gestionnaire du terminal de point de vente (ou module d'encaissement) :

- Le « **POS** » est le noyau du système, c'est le chef d'orchestre du module.
- Le « **DirectPrint** » s'occupe de tous les types d'impression, de l'affichage sur le viseur et de la lecture du code CMC7 des chèques.
- Le « **Scan** » écrit les codes EAN (codes barres) en provenance du scanner dans la mémoire partagée.
- Le « **Ticket** » gère les encaissements en collaboration avec le « **TicketLine** »
- Le « **Metabase** » est un ensemble de fonctions permettant de manipuler les bases de données quelque qoit le SGBD (Système de Gestion de Base de Données)
- Certaines requêtes sont envoyées directement au SGBD, celle-ci respecte un standard : ce sont des commandes SQL.

Les données enregistrées dans la base de données des terminaux remontent régulièrement au serveur central. Les mises à jours éventuelles sur les produits ou les remises par exemple descendent régulièrement jusqu'aux terminaux. Ce mécanisme assure l'autonomie des terminaux en cas de panne du serveur central.

Le module est écrit en PHP comme la quasi totalité de Xstore. Sur ce logiciel PHP offre des avantages certains :

- Il possède beaucoup de fonctions natives qui permettent notament d'utiliser plusieurs types de systèmes de gestion de bases de données.
- Le code est exécuté directement sur le serveur ce qui empêche les utilisateurs d'accéder au code source (contrairement au JavaScript par exemple)
- Il fonctionne à 80% de la vitesse du langage C++ et lui ressemble beaucoup
- Un simple navigateur internet suffi pour accéder à l'interface de XStore ce qui permet son utilisation sur un grand nombre de systèmes d'exploitation.
- Si le serveur est connecté sur internet, les différents utilisateurs peuvent accéder à leurs données à partir de n'importe quelle connection à internet. On peut par exemple changer un prix ou ajouter un produit dans la base à distance. Ceci sous-entend que la politique de sécurité doit être à la hauteur...

Développement des

pilotes de périphériques

Pilote de

l'imprimante

Le stage a commencé par le développement de ce pilote. En collaboration avec le second stagiaire, nous avons « épluché » les documentations techniques des imprimantes Epson TM-H6000II et Epson TM-U950. Le premier modèle est équipé d'un système d'impression thermique et le second d'un système matriciel. Le premier modèle, plus récent, est beaucoup plus rapide et silencieux que le second. Cependant, malgré leurs quelques différences, leurs commandes sont pratiquement semblables.

Voici les différentes fonctionnalités à gérer :

- Compatibilité avec tout type d'imprimante
- Impression « classique » des pièce de caisse (roll paper)
- Impression du journal (journal paper)
- Impression des factures (slip paper)
- Impression sur chèques (check paper)
- Affichage sur l'afficheur à leds appelé également viseur (display)
- Ouverture du tirroir caisse (cash drawer)
- Lecture du code CMC7 des chèques.

Les textes envoyé à l'imprimante sont formaté en XML. C'est un système de balises ouvrante et fermante entre lesquelles on place les données à formater. Ce système ressemble au langage HTML :

```
<nom_de_la_balise option="valeur">Données formatées par la
balise</nom_de_la_balise>
```

Pour la gestion de l'imprimante, une classe nommée DirectPrint a été créée, c'est elle qui gère toutes les fonctionnalités sitées ci-dessus. Les fonctions de la classes sont les suivantes :

Nom	***Commentaire***
DirectPrint()	Constructeur, charge le fichier d'initialisation spécifique à chaque modèle d'imprimante
parse()	Fonction d'interprétation du XML
StyleOn()	Fonction de gestion des balises XML ouvrantes
StyleOff()	Fonction de gestion des balises XML fermantes
cdata()	Fonction de gestion des données situées entre les balises XML
Open()	Ouvre le port de l'imprimante
ClosePrinter()	Ferme le port de l'imprimante
OpenCashDrawer()	Ouvre le tirroir caisse connecté à l'imprimante
Check()	Lit le code chèque et le remplit
EndTicket()	Envoie les dernières données à l'imprimante et coupe le ticket
ParseSequence()	Interprète le fichier d'initialisation
PrintIt()	Imprime les données formatées qu'on lui envoie
ChrConvert()	Gère les caractères accentués
evalsub()	Gère les caractères ASCII (méthode de codage des caractères) non-imprimable
PrinterColumns()	Retourne le nombre de caractères imprimable sur une ligne de ticket
DisplayColumns()	Retourne le nombre de caractères affichable sur une ligne de l'afficheur à leds

La compatibilité avec tous les types d'imprimante est gérée par le chargement d'un fichier d'initialisation spécifique à chaque modèle d'imprimante contenant les séquences à envoyer à l'imprimante pour chaque instruction.

Voici un extrait du fichier d'initialisation de l'Epson TM-U950 (EpsonTM-U950.ini) :

```
[Style]
center_on = (ESC)a{1}
center_off = (ESC)a{0}
right_on = (ESC)a{2}
right_off = (ESC)a{0}
bold_off = (ESC)G{0}
bold_on = (ESC)G{1}
italic_on = (ESC)E{1}
italic_off = (ESC)E{0}
upsidedown_off = (ESC){{0}
upsidedown_on = (ESC){{1}
underline_on = (ESC)-{1}
underline_off = (ESC)-{0}
fontb_off = (ESC)!{1}
fontb_on = (ESC)!{0}
emphasized_off = (ESC)!{1}
emphasized_on = (ESC)!{8}
doubleheight_off = (ESC)!{1}
doubleheight_on = (ESC)!{17}
doublewidth_off = (ESC)!{1}
doublewidth_on = (ESC)!{33}
doublesize_off = (ESC)!{1}
doublesize_on = (ESC)!{49}
```

[style] est une section du fichier d'initialisation, elle gère la mise en page du ticket. La séquence qui va être utilisée par une balise ouvrante à un nom qui se termine par **_on**. De même la séquence qui va être utilisée par une balise fermante à un nom qui se termine par **_off**. Un texte qu'on voudra centrer sera formaté en XML ainsi :

```
<center>Texte à centrer</center>
```

La fonction parse() associée aux fonctions StyleOn() et StyleOff() transformera cela par :

```
(ESC)a{1}Texte à centrer(ESC)a{0}
```

(ESC) sera transformé par les fonctions ParseSequence() et evalsub() en un caractère ASCII (méthode de codage des caractères) non-imprimable dont le code en hexadécimal est : **1B**.

{*nombre***}** sera tranformé par la fonction ParseSequence() en un caractère ASCII non-imprimable dont le code en décimal sera ***nombre***.

L'ensemble est ensuite envoyé à l'imprimante par le port série.

Un document est donc un ensemble de balises XML et de données situé entre les balises **<document>** et **</document>**.

Illustrons ceci par un exemple commenté d'une entête XML et son résultat :

<document>

Début du document

<initialize/><fr/><pc850/>

Initialisation et définition des paramètres régionaux

<priter><roll/>

Sélection de l'imprimante de ticket

** **********
**

Étoiles centrées

** <doublesize>XStore</doublesize>
**

XStore centré et en double taille

**<bold>Corporate Management Software</bold>
**

Texte mis en gras

** **********
**

Étoiles centrées

</printer>

Déselection de l'imprimante

**<display>Bonjour
<right>à vous</right></display>**

Affichage de Bonjour sur l'afficheur à leds

</document>

Fin du document

Résultat :

Ce code imprimera ceci sur l'imprimante (ticket) :

```
**********
   XStore
Corporate Management Software
**********
```

et affichera ceci sur l'afficheur :

```
Bonjour
              à vous
```

Pilote du

scanner

Le pilote du scanner fonctionne de manière assez simple. Il regarde en permanence s'il y a des données en entrée du port série sur lequel le scanner est branché. Le scanner envoie les codes barres en caractère ASCII (méthode de codage des caractère) sur le port série. Le pilote les lit et les empile dans la mémoire partagée. Le module d'encaissement dépile les codes un à un pour les exploiter. L'utilisation d'une pile en mémoire partagée évite que des produits ne soit pas facturé si le vendeur va plus vite que le module. Lorsque le vendeur demande la fin de ticket, la pile est vidée avant que la fin de ticket soit effectivement gérée. De même, à la création du ticket, la pile est vidée pour éviter que les codes scannés malencontreusement entre deux ticket soit pris en compte.

Développement de la gestion des

pièces de caisse

Gestion d'une

pièce de caisse

Pour la création d'une pièce de caisse (ou ticket mais pièce de caisse est plus correct), une classe Ticket a été créée. Les fonctions de la classes sont les suivantes :

Nom	***Commentaire***
Ticket()	Constructeur, crée un nouveau ticket
AddLine()	Ajoute une nouvelle ligne au ticket
UndoLine()	Annule une ligne
AddPayment()	Ajoute un nouveau moyen de paiement
EndTicketTotal()	Enregistre le nombre de produits, le total, la date et l'heure dans la base de données
EndTicketPayment()	Enregistre l'identifiant du paiement, la somme payée, la somme rendue et s'il le faut, le numéro du chèque, ou de la transaction pour un carte bancaire dans la base de données
Lines()	Retourne le détail du ticket
LastLine()	Compte le nombre de ligne dans le ticket
SubTotal()	Calcule le sous-total
Total()	Récupère le total enregistré la base de données
PayReturn()	Récupère le rendue monnaie enregistré la base de données
PaySum()	Récupère la somme payée enregistrée dans la base de données

Nom	*Commentaire*
Date()	Récupère la date et l'heure de création du ticket enregistrées dans la base de données au format ISO du type 2002-04-15 09:00:00
DateFormat()	Formate une date et une heure au format ISO en une date et une heure du type : lundi 15 avril 2002 09:00:00
DateStr()	Alias de la fonction Date()
SalesManID()	Récupère l'identifiant du vendeur enregistré dans la base de données
ItemCount()	Compte le nombre de ligne dans le ticket
PaymentType()	Récupère le type de paiement enregistré dans la base de données
GetSavings()	Calcule les meilleurs remises s'applicant à chaque produits.
GetHeader()	Retourne l'entête du ticket formatée pour l'imprimante
GetSubTotal()	Retourne le sous-total formaté pour l'afficheur à leds
GetSavingsLine()	Retourne les remises formatée pour l'imprimante
GetVATLine()	Retourne la TVA formatée pour l'imprimante
FinalTotal()	Calcule le total après remises
GetFooter()	Retourne le pied de ticket formaté pour l'imprimante

Liste des fonctions de la classe Ticket

On crée donc une instance de Ticket. Les variables d'instance sont ensuite enregistrée dans la base de donnée. La table correspondante s'appelle Sales_Sales, elle est composée des champs suivants :

Champ	*Type*	*Commentaire*
ID	entier	Identifiant de la pièce de caisse
SalesmanID	entier	Identifiant du vendeur
Created	date et heure	Date et heure de création
ItemCount	entier	Nombre de ligne dans le détail
Total	décimal	Total à payer par le client

Champ	*Type*	*Commentaire*
PaymentTypeID	entier	Identifiant du type de paiement
PaySum	décimal	Somme payée par le client
PayReturn	décimal	Somme rendue au client
PayAckCode	chaîne de caractères	Numéro du chèque ou de la transaction pour un carte bancaire

Structure de la table Sales_Sales

A chaque nouvelle pièce de caisse, on incrémente l'identifiant du ticket (un identifiant est unique et ne doit posséder aucun doublon dans les enregistrements de la table). Chaque pièce de caisse est liée à la table des vendeurs et à celle des types de paiement. Pour rechercher le nom du vendeur lié à une pièce de caisse par exemple, on cherchera l'unique enregistrement dans la table des vendeur ayant pour identifiant la valeur du champ SalesmanID, on aura alors accès à toutes les données du vendeur. Ceci est valable en théorie seulement car n'importe qui n'a pas accès à n'importe quelle donnée. Chaque personne qui utilise la base de donnée a des droits restreint en fonction des besoins de son rôle. Un vendeur aura uniquement accès à certains élément des tables produits, TVA, remises en lecture seulement et des tables pièces de caisse et détail en lecture et écriture sous certaines conditions. Par exemple le vendeur n'a pas accès au prix d'achat des produits de la table de produit mais à celui de vente.

Un encaissement se passe de la manière suivante :

- On crée un ticket, sont alors définis les champs ID, SalesmanID (voir tableau ci-dessus).
- On crée le détail des achats avec le scanner de codes barres ou le clavier (voir partie suivante).
- On fait le total non-définitif, en effet les remises se calcule après le choix du moyen de payment (en effet la carte de fidélité « PASS » de Carrefour par exemple est à la fois un système de remise et un moyen de paiement, à ne pas confondre avec les cartes qui accumulent des points sur un compte comme par exemple la carte « Iris » de InterMarché), de même les remises immédiates sur certains produits sont considérées comme des moyens de paiement.
- On choisi le(s) moyen(s) de paiement, on cherche dans les produits achetés ceux qui donne droit à des remises (on fouille la table des remises), et on en fait le total. On imprime ensuite toutes ces informations.

- On calcule alors le total à payer et on l'imprime.
- On fait le détail de la TVA en prenant en compte les remises qui ont été faite.
- Pour chaque moyen de paiement on calcule et imprime le moyen de paiement et la somme payée et s'il le faut ce qu'il reste à payer ou le montant du rendu monnaie.
- On complète l'enregistrement de la pièce de caisse par les dernière information obtenue (ItemCount, Total, Created, PaymentTypeID, PaySum, PayReturn, PayAckCode).
- On imprime le pied de la pièce de caisse (Date de création du ticket, identifiant du vendeur, numéro de la caisse et éventuellement les messages publicitaire et de remerciement).
- On imprime l'entête du ticket suivant et on coupe le ticket juste au dessus le d'entête qui vient d'être imprimé, cette astuce permet d'économiser le papier.

Gestion du détail

Pour la création du détail d'une pièce de caisse, une classe nommée TicketLine a été créée.

Nom	*Commentaire*
TicketLine()	Constructeur, crée une nouvelle ligne
Name()	Retourne le nom du produit
EAN()	Retourne le code barres du produit
Price()	Retourne le prix total de la ligne (prix unitaire multiplié par la quantité) avec la TVA incluse
Quantity()	Retourne la quantité de produit acheté
GetLineInfo()	Retourne les informations concernant la ligne formatée pour l'imprimante et l'afficheur à leds.

Liste des fonctions de la classe TicketLine

On crée donc une nouvelle instance de TicketLine à chaque ligne du détail de la pièce de caisse. Les variables d'instance sont ensuite enregistrée dans la base de donnée. La table correspondante s'appelle Sales_SalesLines, elle est composée des champs suivants :

Champ	*Type*	*Commentaire*
ID	entier	Identifiant de la ligne
SalesID	entier	Identifiant du ticket
ProductID	entier	Identifiant du produit
EANCode	chaîne de caractères	Code barre du produit
Quantity	entier	Quantité de produit acheté
UnitPrice	décimal	Prix unitaire du produit
VATTypeID	entier	Identifiant de la TVA
Placement	entier	Numéro de la ligne au sein du ticket

Structure de la table Sales_SalesLines

A chaque nouvelle ligne, on incrémente l'identifiant de cette ligne. Chaque ligne est liée à la table des ticket, à celle des produits (voir la partie Table des produits) et à celle de la TVA (voir la partie Table des TVA).

La création d'une ligne se passe de la manière suivante :

- On crée une ligne, sont alors directement définis les champs ID, TicketID, EANCode, Quantity et Placement (voir tableau ci-dessus).
- A l'aide du code barres (EANCode), on récupère l'identifiant du produit (ProductID) dans la table des produits, ainsi que le prix unitaire (UnitPrice) et l'identifiant du type de TVA (VATTypeID).

Table des

Produits

La table des produits dont j'ai déjà parlé précédement dans la gestion du détail, a la structure suivante :

Champ	*Type*	*Commentaire*
ID	entier	Identifiant du produit
ProductNumber	*chaîne de caractères*	*Non-utilisé dans le module d'encaissement*

Champ	***Type***	***Commentaire***
EANCode	chaîne de caractères	Code barres du produit
IntCode	*chaîne de caractères*	*Non-utilisé dans le module d'encaissement*
Name	chaîne de caractères	Nom du produit
Contents	*chaîne de caractères*	*Non-utilisé dans le module d'encaissement*
Brief	*chaîne de caractères*	*Non-utilisé dans le module d'encaissement*
Description	*chaîne de caractères*	*Non-utilisé dans le module d'encaissement*
Keywords	*chaîne de caractères*	*Non-utilisé dans le module d'encaissement*
BuyPrice	*décimal*	*Non-utilisé dans le module d'encaissement*
Ratio	*décimal*	*Non-utilisé dans le module d'encaissement*
ListPrice	*décimal*	*Non-utilisé dans le module d'encaissement*
Price	décimal	Prix unitaire du produit
ShowPrice	*booléen*	*Non-utilisé dans le module d'encaissement*
ShowProduct	*booléen*	*Non-utilisé dans le module d'encaissement*
Discontinued	*booléen*	*Non-utilisé dans le module d'encaissement*
ExternalLink	*chaîne de caractères*	*Non-utilisé dans le module d'encaissement*
IsHotDeal	booléen	Y a-t-il des remises sur le produit
ShowSavings	*booléen*	*Non-utilisé dans le module d'encaissement*
RemoteID	*chaîne de caractères*	*Non-utilisé dans le module d'encaissement*
VATTypeID	entier	Identifiant du type de TVA
ShippingGroupID	*entier*	*Non-utilisé dans le module d'encaissement*
ProductType	*entier*	*Non-utilisé dans le module d'encaissement*
VendorIDLink	*entier*	*Non-utilisé dans le module d'encaissement*
StartingTime	date + heure	Début de la validité des remises temporaires sur un produit
ExpiryTime	date + heure	Fin de la validité des remises temporaires sur un produit
Published	*booléen*	*Non-utilisé dans le module d'encaissement*

Champ	*Type*	*Commentaire*
IncludesVAT	booléen	La TVA est-elle incluse dans le prix (Price)

Structure de la table Commerce_Product

Table des

TVA

Voici la structure de la table des TVA qui est utilisé par la fonction Price() dans la gestion du détail afin de calculer le prix toute taxe comprise d'un produit :

Champ	*Type*	*Commentaire*
ID	entier	Identifiant du type de TVA
Name	chaîne de caractères	Nom du type de TVA
VATValue	décimal	Valeur de la TVA
Created	*date + heure*	*Non-utilisé dans le module d'encaissement*

Structure de la table Commerce_VATType

Table des

remises

La fonction GetSavings() dans la gestion du ticket est utilisé afin de calculer les remises valables sur les produits achetés, elle utilise la table des remises dont la structure est la suivante :

Champ	*Type*	*Commentaire*
ID	entier	Identifiant de la remise
ProductID	chaîne de caractères	Nom du type de TVA
Value	décimal	Valeur de la TVA
Type	chaîne de caractères	Type de remise (global, temporaire, carte de fidélité, etc)

Commerce_Savings

Conclusion

La gestion des terminaux de caisse n'était pas commencé au 15 avril 2002. En collaboration avec un autre stagiaire, nous devions créer les pilotes des terminaux puis développer les fonctions pour les exploiter.

J'ai du apprendre à travailler avec différents outils : le langage PHP, les bases de données et les différents ensembles de fonctions. Mon intégration au sein des équipes que j'ai rencontré n'a posé aucun problème.

Aujourd'hui le module fonctionne mais quelques améliorations devront y être apporté pour être pleinement exploitable.

L'intégration du logiciel va être fait par cont@ctis, une entreprise spécialisé dans l'intégration des terminaux de point de vente. Il fournisse à leurs clients les solutions matérielles et logicielles correspondant aux besoins. Un séjour à Paris m'a permis de rencontrer l'équipe de cette entreprise qui s'est investi dans le projet XStore. Grâce à leur expérience dans le domaine des points de vente ils ont su me conseiller sur la conception des correctifs nécessaires sur le module pour que ce dernier soit exploitable.

Actuellement le module a servi à faire les démonstrations lors des présentations du logiciel XStore. Des entreprises se sont montrés intéressé par le logiciel mais ne s'attarde pas sur le module d'encaissement pour l'instant et préfère s'intêresser à d'autres fonctionnalités que leur logiciel ne possède pas.

Les améliorations envisageables sur le module :

- Il faut optimiser la vitesse d'exécution des scripts
- Les scripts fonctionnent mais le code doit être nettoyé des lignes de « debugging »
- La gestion des factures doit être intégrée
- La gestion des remises doit être amélioré : un « bug » subsiste dans les remises temporaire (dites « flash »)

- Il faut développer la fonction de gestion des graphismes
- L'internationalisation doit être faite
- Il faudra adapté le module aux cahiers des charges fournis par les clients
- Le développement d'une interface graphique permettant la mise en page des ticket « à la souris » pourra être envisagée

Annexes

Bibliographie

Lors du stage, plusieurs documents m'ont été utiles.

Pour l'apprentissage de PHP :

- Le hors série *Programmer en PHP* du magazine *Login* aux éditions *Posse Press*
- Le site *www.php.net* dans la section *documentation*

Pour la programmation des pilotes :

- Les documentations étaient fournies par Epson, elles portent sur chacune de leurs pages la mention « Confidential »...

Pour l'apprentissage de MySQL et PHP :

- Le livre *Pratique de MySQL et PHP* au édition *O'Reilly*

GNU/Linux

Qu'est-ce que Linux?

Linux est un système d'exploitation de type UNIX, multi-tâches et multi-utilisateurs pour machines à processeurs 32 et 64 bits (en particulier les machines de type PC et PowerMac), ouvert sur les réseaux et les autres systèmes d'exploitation.

La principale singularité de Linux est d'être un logiciel libre, développé de façon collaborative et pour une grande part bénévole par des milliers de programmeurs répartis dans le monde.

Ce modèle de développement joue un grand rôle dans la qualité du résultat obtenu, qui est considéré par des analystes indépendants comme très supérieurs à des systèmes commerciaux similaires, par exemple Windows NT.

Avantages de Linux

Linux est un système :

- ***Puissant***. Il permet de faire faire beaucoup de choses à sa machine.
- ***Efficace***. Contrairement à des systèmes beaucoup plus répandus, il n'utilise pour ses besoins propres que très peu de ressources. Les logiciels que vous utilisez pour votre travail disposent donc de beaucoup plus de puissance pour fonctionner.
- ***Fiable***. Une machine sous Linux fonctionne 24h/24 si besoin sans se plaindre (si le matériel est prévu pour, en particulier au niveau thermique).
- ***Robuste***. Une erreur d'un utilisateur ou un ``plantage" éventuel d'une application n'affectent pas le reste du système. D'autre part, il est exceptionnel de devoir l'arrêter: la quasi-totalité des opérations de configuration, mise au point, etc, ne nécessitent pas l'arrêt du système.
- ***Très bon marché***. Le prix demandé par les sociétés qui vendent Linux sur CDROM ne sert qu'à couvrir leurs frais et à leur permettre de financer dans une certaine mesure la poursuite de cette activité. Linux étant développé par des passionnés pour le plaisir, personne n'a à supporter le coût de son développement.
- Enfin, Linux est conforme à la norme ***POSIX*** et aux standards du marché, en particulier de

l'Internet. Cela signifie que qu'un logiciel conçu pour un autre système de la même famille (Solaris de SUN, Digital Unix, AIX d'IBM, SCO Unix...) peut être rapidement porté sous Linux et vice-versa, ce qui assure une protection de l'investissement logiciel en cas d'obligation de changement de système.

Comme on le voit, Linux est un système exceptionnel donnant satisfaction aussi bien sur de des machines anciennes ou bas de gamme que sur des machines puissantes très sollicitées ou devant remplir des fonctions importantes.

Principales utilisations de Linux

- Comme ***serveur de fichiers et d'impression***: Linux supporte les trois principaux protocoles de partages de fichiers: NFS pour clients UNIX, SMB pour clients Windows et AppleShare pour clients MacOS, ainsi que les protocoles de partages d'imprimantes. Linux peut également servir de serveur de fax.
- Comme ***serveur Internet/Intranet***: On trouve dans les distributions standards de Linux tous les logiciels nécessaire pour réaliser un serveur Internet complet, même sur des machines de puissance modeste. Cela inclut des fonctionnalités:
  - de transfert et de distribution du courrier électronique, des *news* (Usenet).
  - de serveur Web ou FTP.
  - de serveur de noms de machines ou de domaines.
  - Comme ***serveur d'applications client/serveur***: un grand nombre de logiciels de serveurs de bases de données (SGBD), relationnels, relationnels-objets ou objets, commerciaux ou libres, sont disponibles pour Linux. Avec ces logiciels, surtout du côté commercial, ce sont des milliers d'applicatifs.
  - Comme ***station de développement***: Linux dispose, le plus souvent sous forme de logiciels libres, des outils de développement pour la plupart des langages actuels: C, C++, Fortran, Java, Cobol, LISP, Prolog, SmallTalk. On trouve aussi des outils pour le contrôle des sources, pour le travail en groupe, pour le suivi des erreurs, pour le testage.
  - Comme ***station bureautique***: grâce à des suites bureautiques intégrées commerciales comme Applix ou StarOffice et à des environnements graphiques comme KDE ou fvwm95, les stations bureautiques sous Linux offre les même fonctionnalités que leurs équivalents sous Windows ou MacOS: traitement de texte, tableur, logiciel de présentation et de dessin,

gestion de fichiers, partage de documents, intégration des technologies Internet/Intranet (courrier électronique, Web).

- Comme ***station réseau***: Grâce à un logiciel intégré comme *Netscape Communicator*, ou à des logiciels plus spécialisés, Linux permet d'accéder aux services les plus populaires de l'Internet: Web, FTP (transferts de fichiers), news (messageries thématiques), courrier électronique, IRC (discussions en temps réel).

Autres utilisations de Linux

- Comme station graphique ou PAO.
- Comme station de productivité personnelle.
- Comme console de loisirs.

PHP

Qu'est ce que PHP?

Ce qui distingue le PHP (officiellement "PHP: Hypertext Preprocessor") des langages de script comme le Javascript est que le code est exécuté sur le serveur. Si vous avez un script similaire sur votre serveur, le client ne reçoit que le résultat du script, sans aucun moyen d'avoir accès au code qui a produit ce résultat. Vous pouvez configurer votre serveur web afin qu'il analyse tous vos fichiers HTML comme des fichiers PHP. Ainsi, il n'y a aucun moyen de distinguer les pages qui sont produites dynamiquement des pages statiques.

Que peut faire PHP?

Le langage PHP possède les même fonctionnalités que les autres langages permettant d'écrire des scripts CGI, comme collecter des données, générer dynamiquement des pages web ou bien envoyer et recevoir des cookies.

La plus grande qualité et le plus important avantage du langage PHP est le support d'un grand nombre de bases de données. Réaliser une page web dynamique interfacant une base de donnés est extrêmement simple. Les bases de données suivantes sont supportées par PHP :

Adabas D	dBase	Empress	FilePro (lecture seule)
Hyperwave	InterBase	FrontBase	mSQL
Direct MS-SQL	MySQL	PostgreSQL	Sesam
Solid	Sybase	Velocis	IBM DB2
Informix	Ingres	ODBC	
Oracle (OCI7 et OCI8)	Ovrimos	Unix dbm	

Le langage PHP inclus le support des services utilisant les protocoles tels que IMAP, SNMP, NNTP, POP3 ou encore HTTP. Vous pouvez également ouvrir des connexions et interagir en utilisant d'autres protocoles.

La genèse du PHP

Le langage PHP a été conçu durant l'automne 1994 par Rasmus Lerdorf. Les premières versions (qui restèrent privées) étaient utilisées afin de savoir qui venait consulter son CV en ligne. La première version publique fut disponible au début de l'année 1995. Elle fut connue sous le nom de "Personal Sommaire Page Tools". Elle était composée d'un analyseur extrêmement simple qui ne reconnaissait que quelques macros spéciales et d'un petit nombre d'utilitaires couramment utilisés dans les pages web. Un livre d'or, un compteur, etc... L'analyseur fut réécrit durant l'été 1995 et fut appelé PHP/FI Version 2. FI etaient les initiales d'un autre package que Rasmus avait écrit qui interprétait les formulaires HTML. C'est alors qu'il combina le "Personnal Sommaire Page tools" avec le "Form Interpreter" et il y ajouta le support de mSQL: c'est comme cela que naquît PHP/FI. PHP/FI grandit de manière spectaculaire et de nombreuses personnes commencèrent à contribuer à son amélioration.

Il est relativement peu aisé de donner des statistiques, mais on estime que PHP/FI est utilisé sur 15 000 sites web dans le monde entier, fin 1996. Ce chiffre atteint 50 000 durant l'été 1997. L'été 1997 voit aussi un profond changement dans le développement du PHP: d'un projet personnel (celui de Ramsus), on passe alors à une projet d'équipe. L'analyseur fut de nouveau réécrit par Zeev Suraski et Andi Gutmans et ce nouvel analyseur forma la base de la version 3 du PHP. Une grande partie du code de PHP/FI fut complètement réécrit alors que l'autre partie fut portée pour donner le PHP Version 3. La dernière version de PHP (PHP 4) utilise le moteur d'analyse Zend pour atteindre de nouveaux niveaux de performance, et supporter un nombre encore plus grand de librairies et extensions. Il tourne de manière native sur tous les serveurs web les plus répandus.

Aujourd'hui PHP3 ou PHP4 sont distribués avec de nombreux produits commerciaux comme "C2's StrongHold web server" et "RedHat Linux" et il est admis (d'après les chiffres de NetCraft, et leurs statistiques Netcraft Web Server Survey) que le PHP est utilisé sur 5 100 000 sites web dans le monde entier. Pour comparaison, ce chiffre est légèrement supérieur au nombre de serveurs tournant sous Microsoft Information server (IIS) : 5.03 millions.

La nouvelle génération du PHP utilise les qualités de Zend pour améliorer les performances et améliore le support des serveurs web autres que Apache.