**INGINERIA PROGRAMARII**

**- PROIECT -**

**Scopul proiectului**

Aplicatia este destinata importului unor date contabile din fisiere Excel intr-o baza de date a unei aplicatii de contabilitate numita Ciel. Clientul este o firma de contabilitate care primeste de la clienti fisiere excel continand produsele achizitionate printr-o nota de intrare-receptie (NIR). In acest fisier se specifica pentru fiecare produs: *denumirea, cantitatea, pretul unitar, pretul total, codul de bare, unitatea de masura si moneda*. Aceste date erau pana in acel moment introduse manual de angajatii firmei si se dorea automatizarea acestui proces.

**Solutia**

S-a ales implementarea unei aplicatii cu interfata grafica, din motive evidente: utilizatorii nu sunt persoane tehnice si o aplicatie de consola care sa citeasca parametrii de la linia de comanda nu ar fi corespuns asteptarilor lor.

Aplicatia se conecteaza la baza de date de unde citeste lista *notelor de intrare-receptie* existente, lasand utilizatorul sa aleaga nota pentru care va incarca date (printr-un control de tip select-box).

Incarcarea de fisiere Excel se face selectand fisierul intr-o fereastra de tip file-chooser (sau scriind de mana locatia acestuia). Dupa incarcare, continutul fisierului este afisat intr-un control de tip tabel, continut intr-un panel tabular (care imita chiar interfata programului Microsoft Excel): pentru fiecare sheet din fisier se creeaza un tab cu numele acestuia. Tabelul este editabil, pentru a permite corectarea erorilor din fisier fara a fi nevoie de editarea acestuia, insa modificarile nu vor fi salvate in fisier.

Dupa incarcarea tabelului se activeaza un buton de *import*, la apasarea caruia se va incerca importul datelor afisate. Daca acesta reuseste, butonul este dezactivat pentru sheet-ul importat (pana la terminarea aplicatiei). Aceasta deoarece exista riscul ca, prin apasarea de doua ori a butonului import pentru aceleasi date, sa fie introduse duplicate in baza de date.

De asemenea, deasupra tabelului am afisat un control de tip check-box prin care se specifica daca va fi inclusa (sau nu) prima linie in import (in multe cazuri, fisierele contin un header-row care nu trebuie importat).

Importul se realizeaza intr-o tranzactie, astfel ca in cazul aparitiei unei erori sa nu se adauge nici un produs din sheet-ul ales. Erorile sunt semnalate intr-o caseta dedicata din interfata (nu intotdeauna intr-o maniera eleganta: erorile de SQL sunt raportate asa cum le raporteaza driverul JDBC). Daca mesajul de eroare este suficient de explicit, utilizatorul are posibilitatea corectarii datelor care au dus la aparitia ei prin editarea valorilor de la linia mentionata.

**Limbaj si mediu de dezvoltare**

Programul este implementat in Java (versiunea 1.5), utilizand Swing pentru interfata grafica. Mediul de dezvoltare a fost Eclipse.

**Resurse software**

Fiind vorba de Java, nu s-a pus problema sistemului de operare; de aceea, am putut dezvolta pe o statie Linux, desi aplicatia era destinata unor masini care rulau Windows.

Pentru a avea acces la SQL Server in etapa de dezvoltare, a fost nevoie de instalarea unui Windows 2000 intr-o masina virtuala (vmware), impreuna cu SQL Server 7.

In vederea lucrului din Java cu baza de date, am folosit un driver JDBC pentru SQL Server. Initial am considerat indicat sa folosesc chiar driverul dezvoltat de Microsoft, dar acesta s-a dovedit foarte lent in stabilirea conexiunii si in executia query-urilr; prin urmare, dupa lecturarea unor forum-uri, am optat pentru o varianta de driver open-source: jTDS (versiunea 1.2).

Pentru citirea fisierelor Excel, am folosit o librarie java (de asemenea open-source) numita JExcel API.

**Resurse hardware**

Aplicatia este destinata unui sistem desktop obisnuit (memorie de cel putin 256M, procesor de minim 1GHz). Nu este necesar spatiu pe disc pentru aplicatie in sine, ci doar pentru stocarea fisierelor si pentru baza de date (care constituie resurse externe).

**Performanta**

Operatiile care pot fi consumatoare de resurse sunt citirea fisierelor excel si importul in baza de date.

Sheet-urile Excel sunt incarcate (parcurse) la fiecare accesare (chiar daca nu sunt citite de pe disc, ele fiind tinute in memorie de catre libraria jexcelapi). Aceasta abordare este excesiv de grijulie cu folosirea memoriei, insa ignora timpul necesar parcurgerii si maparii acestor date, mai ales in cazul fisierelor foarte mari, unde schimbarea sheet-urilor ar fi o operatie greoaie (nu doar la prima accesare).

Operatiile cu baza de date sunt suficient de optimizate prin folosirea de interogari SQL pre-compilate, iar driverul folosit are reputatia de “cel mai rapid”. Dar, evident, performanta acestor operatii depinde in primul rand de performanta sistemului de gestiune a bazei de date insusi, care insa este un factor extern in acest context.

**Perspective ulterioare**

Anumite functionalitati ar trebui imbunatatite intr-o eventuala noua versiune:

* Incercarea de conectare la baza de date se face doar la pornirea aplicatiei. Mult mai elegant ar fi sa se repete aceasta incercare de cate ori e nevoie in caz ca nu s-a reusit. In acest moment, chiar daca baza de date devine disponibila, aplicatia trebuie repornita pentru a constientiza acest lucru.
* O functionalitate de cache pentru sheet-urile deja accesate ar face mult mai rapida re-vizualizarea acestora. Aici trebuie facut insa un compromis, astfel incat sa nu se risipeasca nejustificat memoria in cazul fisierelor foarte mari.
* Mesajele de eroare ar putea fi “traduse” in mesaje mai lizibile pentru utilizatori, insotite de sugestii pentru rezolvarea problemei.