Autor(i) : Kiss Zsuzsanna Maxim Diana Mocanu Voica	Referinţă : Caiet de specificaţii
Statut : Valid	Versiune : B
Data : 09/01/2006	Modificări : Descrierea claselor

Caiet de specificatii

CUVINTE CHEIE

Istoricul acţiunilor efectuate asupra documentului			
Nume	**Data**	**Versiune**	**Acţiune(i)**
Maxim Diana	05/12/2005	A	Creare
Kiss Zsuzsanna	10/12/2005	A	Specificaţii funcţionale
Mocanu Voica	19/12/2005	A	Istorie şi proprietăţi Java
Maxim Diana	07/01/2006	B	Concepţia UML
Kiss Zsuzsanna	08/01/2006	B	Descriere module
Kiss Zsuzsanna	08/01/2005	B	Inserare descriere clase
Maxim Diana	09/01/2006	B	Recitire şi corectare

Cuprins

1. Specificaţii funcţionale (funcţionalitate)

1.1. Interfaţa aplicaţiei

Interfaţa grafică a aplicaţiei a fost creată cu Swing.
Interfaţa grafică a aplicaţiei conţine o bară de meniuri (JMenuBar). Meniurile disponibile (JMenu) sunt: File şi Edit.
Meniul File dispune de opţiunile (JMenuItem): Open, Save, Close şi Exit.
Submeniul Open este utilizat pentru deschiderea fişierului XML. Apare o fereastră (JFileChooser) în care utilizatorul poate să aleagă fişierul XML pe care doreşte să-l deschidă. Submeniul poate fi activat şi cu combinaţia de taste Ctrl+O.
Submeniul Save este utilizat pentru pentru salvarea fişierului XML după eventuale modificări (ştergerea unei cărţi, adăugarea unei cărţi). Activarea submeniului se poate face şi apăsând tastele Ctrl+S. Dacă submeniul a fost activat apare o fereastră de tip FileChooser care permite utilizatorului să aleagă directorul în care vrea să salveze fişierul.
Submeniul Close este utilizat pentru închiderea fişierului XML deschis şi poate fi activat ori alegându-l din meniu, ori apasând tastele Ctrl+W.
Alegerea submeniului Exit conduce la terminarea aplicaţiei. Poate fi activat şi apăsând tastele Ctrl+X.
Meniul Edit dispune de opţiunile: Add Book, Remove Book şi Search.
Submeniul Add Book permite adăugarea în fişierul XML a datelor referitoare la o carte nouă. Pentru introducerea datelor referitoare la cartea pe care vrem s-o adăugăm apare o fereastră de tip OptionPane în care putem să definim toate informaţiile. Pentru adăugare trebuie apăsat butonul Add, pentru anulare trebuie apăsat butonul Cancel.
Submeniul Remove Book permite ştergerea unei cărţi din evidenţă. Mai întâi se caută cartea pe care dorim să ştergem în fişierul XML. Rezultatele căutării sunt afişate într-o listă. Pentru ştergerea cărţii trebuie selectată cartea repectivă din listă şi trebuie apăsată tasta Delete.
Submeniul Search permite căutarea unei cărţi după mai multe criterii (numele autorului, titlul, editura). În fereastra care apare poate fi introdusă orice informaţie despre carte. Rezultatele căutării vor aparea într-o listă.Această listă va conţine toate cărţile la care printre informaţii apare cuvântul căutat.Utilizatorul poate să aleagă din această listă cartea pe care o caută.
Datele referitoare la o carte pot fi modificate direct în tabelul în care apar cărţile.

Sursa aplicaţiei a fost scrisă în limba engleză, toate elementele din interfaţa grafică şi din fişierul XML apar în limba engleză, în acest fel fiind internaţionalizată limba programului.

1.2. Fişiere de intrare/ieşire

Fişierele de intrare/ieşire sunt fişiere XML conţinând date referitoare la cărţile care se află în biblioteca personală a clientului.

1.3. Modele de utilizare

Avem de a face cu un singur tip de utilizator. Scenariile sunt aceleşi pentru fiecare utilizator. Oricine are posibilitatea să efectueze modificări asupra fişierului XML dacă porneşte aplicaţia. Fiecare utilizator poate să adauge cărţi, să şteargă cărţi sau să caute între cărţile aflate în evidenţă.

2. Specificaţii tehnice

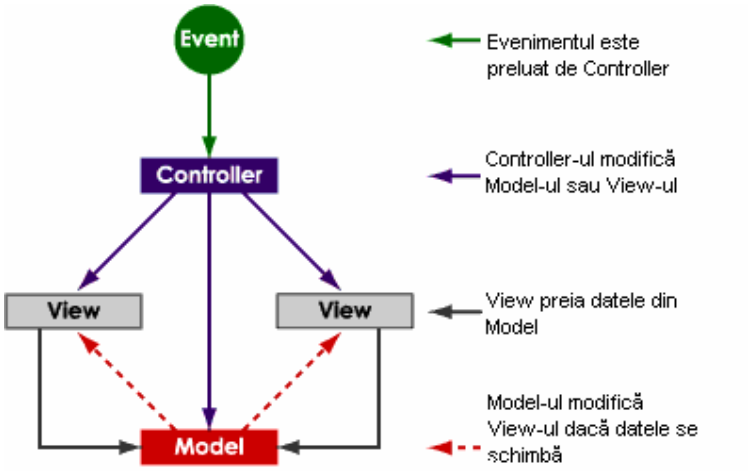
2.1. Arhitectură şi funcţionalităţi

Aplicaţia este construită pe baza modelului MVC (Model - View - Controller). Modelul este implementat folosind un fisier XML având următoarea structură:

```
<?xml version="1.0"?>

<books>
    <book>
        <author>Nume Autor</author>
        <title>Titlu Carte</title>
        <year>Anul apariţiei</year>
        <editor>Editura</editor>
        <pages>Numărul de pagini</pages>
    </book>
</books>
```

View-ul este un GUI Java, iar controllerul va fi o clasa Java care face legătura între Model şi View. Controller-ul furnizează datele din Model către View şi datele modificate din View către Model.



Aplicaţia funcţionează pe baza unei interfeţe grafice uşor de folosit. Utilizatorul poate să aleagă dintr-un meniu operaţiile pe care vrea să le efectueze.

2.1.1. Arhitectura generală a modulelor

Diagrama UML a modulelor:
Diagramele UML au fost genrate folosind plugin-ul lightUML şi programul GraphViz.

Diagrama UML a pachetului basics:

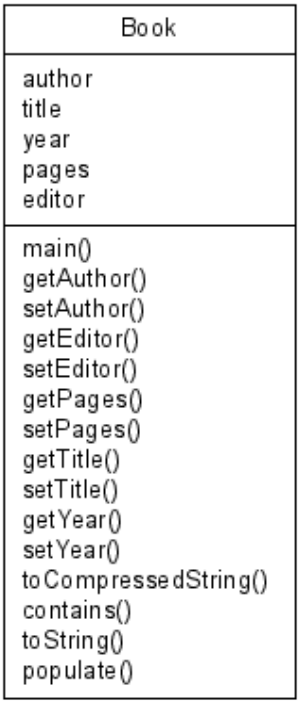


Diagrama UML a pachetului main:

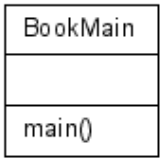


Diagrama UML a modolului Model:

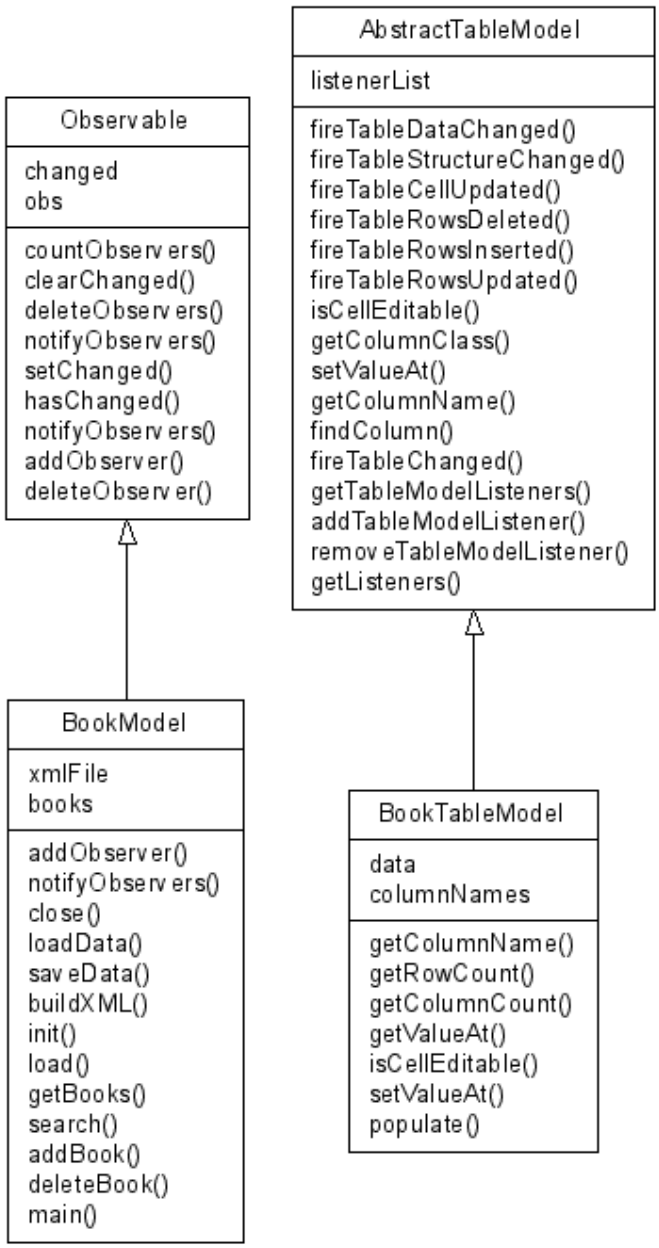


Diagrama UML a modulului View:

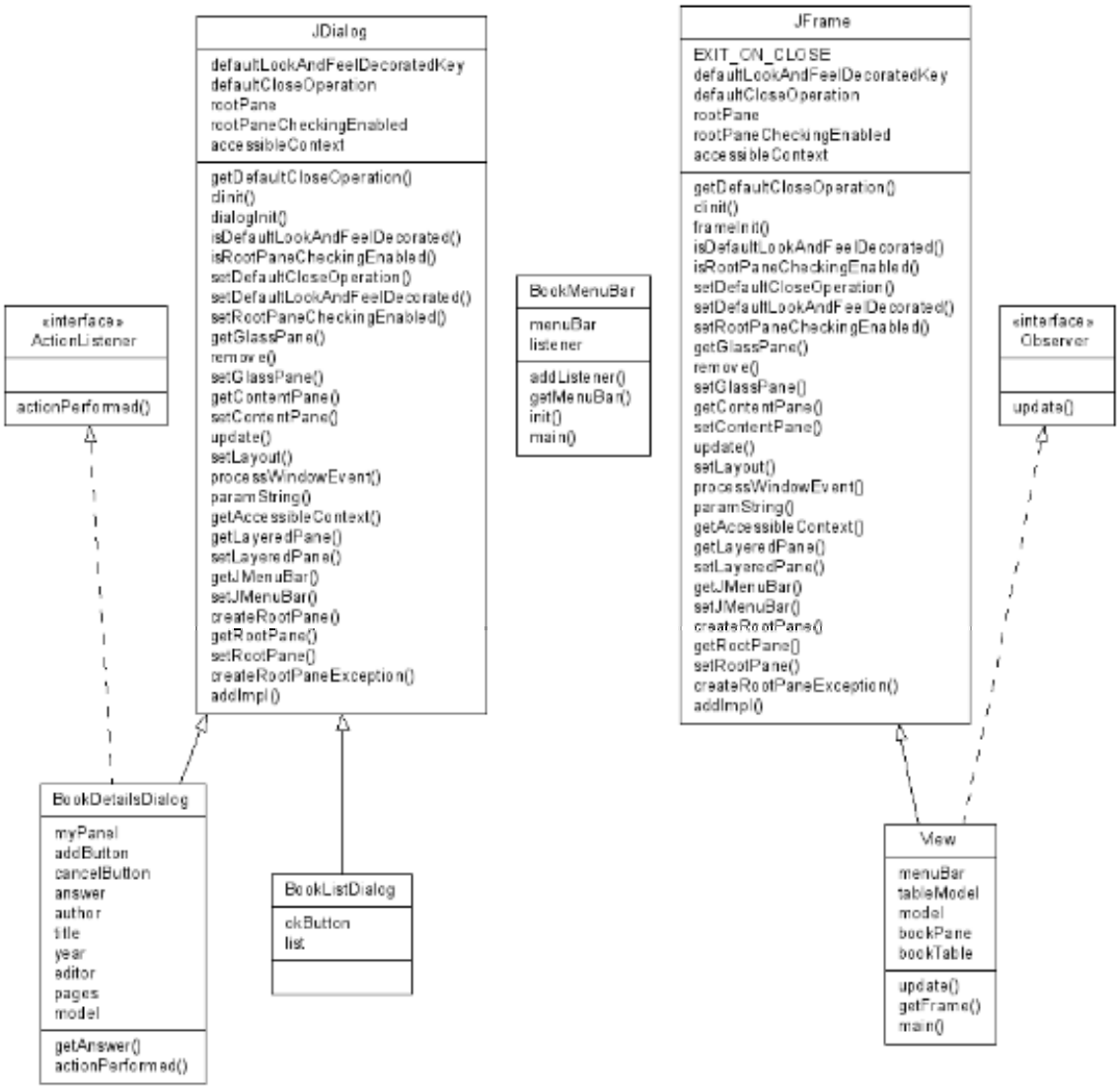
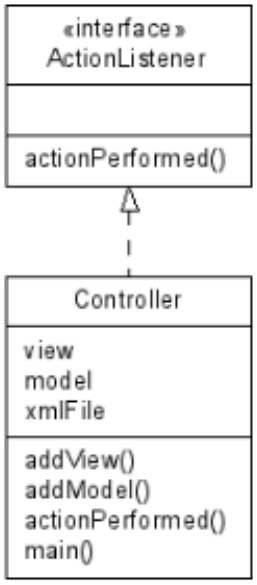


Diagrama UML a modulului Controller:



2.1.2. Concepţia UML globală a aplicaţiei

Diagrama UML globală a aplicaţiei:

Anexa1. conţine diagrama UML globală a aplicaţiei.

2.2. Performanţe

Constrângeri de performanţă ale aplicaţiei: Având foarte multe date (cărţi) stocate, viteza de procesare poate scădea foarte mult. Dacă dimensiunea fişierului XML creşte exponenţial, atunci şi timpul de execuţie a aplicaţiei va creşte exponenţial, ceea ce reprezintă un dezavantaj.

Posibile optimizări pentru ameliorarea performanţelor aplicaţiei: Dacă datele erau stocate într-o bază de date relaţională precum SQL Server sau Oracle, viteza de procesare nu depindea (teoretic) de numărul de cărţi .

2.3. Evolutivitate

Opţiuni de realizat pe viitor/extensibilitate (funcţionalităţi de adăugat):
Elaborarea unui raport referitor la cărţile împrumutate. Se vor oferi informaţii despre persoana căreia i s-a împrumutat cartea (nume, data împrumutării, data returnării cărţii, etc.).

Nu sunt cunoscute bug-uri pentru tratare în versiunile viitoare.

3. Specificaţii de realizare/implementare

3.1. Standard tehnic

3.1.1. Limbaje de programare/platforme

Limbajul de programare utilizat în realizarea aplicaţiei este limbajul Java, iar unealta IDE este Eclipse v.3.1.1.

3.1.2. Utilitatea limbajului/platformei

Avantajele limbajului de programare Java sunt:Java este un limbaj de nivel înalt, orientat obiect şi este un limbaj simplu. Limbajul de programare Java este independent de platformă (aplicaţiile scrise în Java rulează sub orice sistem de operare) şi nu în ultimul rând este gratuit.

3.1.2.1. Servicii disponibile

Servicii disponibile utilizate: documentaţia JDOM, Java tutorial de la Sun.

3.1.2.2. Istoricul limbajului

Începutul limbajul Java este în toamna anului 1991, când firma Sun Microsystems a finanţat un proiect cu numele Green condus de James Gosling. Scopul echipei Green era să plaseze firma Sun Microsystems pe piaţa produselor electronice comerciale. Inginerii şi dezvoltatorii de soft de la Sun au căutat microprocesoare care să ruleze pe o multitudine de maşini, în particular pe sisteme distribuite care lucrează în timp real. După patru ani de lucru, echipa Green finalizează specificaţiile limbajului Java. Apoi, compania Sun Microsystems vinde licenţa firmelor IBM, Microsoft, Silicon Graphics, Adobe şi Netscape.

3.1.2.3. Caracteristicile limbajului

Caracteristicile limbajului Java sunt:

- limbaj interpretat şi compilat: limbaj interpretat înseamnă că instrucţiunile unui program scris sunt procesate linie cu linie şi traduse în cod maşină; limbaj compilat înseamnă că un program scris este tradus într-un cod pe care calculatorul îl poate „înţelege” (executa) mult mai uşor.
- limbaj independent de platformă: la instalarea limbajului Java, se va crea o maşină virtuală Java care are drept scop traducerea instrucţiunilor unui byte code Java în instrucţiuni maşină pentru platforma curentă. Astfel fişierele intermediare byte code pot fi copiate şi executate pe orice platformă.
- limbaj orientat obiect: Java pune în evidenţă toate aspectele legate de programarea orientată obiect: obiecte, trimitere de parametri, încapsulare, clase, biblioteci, moştenire şi modificatori de acces.
- este un limbaj concurent: poate executa mai multe secvenţe de cod în acelaşi timp (multithreading); o secvenţă de cod Java se numeşte fir de execuţie (thread).
- limbaj simplu: posibilitatea moştenirii multiple este exclusă, şirurile sunt încapsulate într-o structură clasă, nu există pointeri, iar alocarea şi dealaocarea memoriei se face automat.
- limbaj distribuit: permite utlizarea obiectelor locale şi de la distanţă. Limbajul Java oferă posibilitatea dezvoltării de aplicaţii pentru Internet, capabile să ruleze pe platforme distribuite şi eterogene.
- limbaj performant: interpretorul Java este capabil să execute un byte code aproape la fel de repede ca un cod compilat.

- limbaj dinamic şi robust: întârzie alocarea obiectelor şi legarea dinamică a claselor pentru momentul execuţiei, astfel se vor evita erorile de alocare.
- limbaj sigur: programele Java nu pot accesa memoria heap, stack sau alte secţiuni protejate de memorie.

3.1.2.4. Puncte cheie ale limbajului

Punctele forte care au recomandat limbajul: este limbaj independent de platformă, deci aplicaţia scrisă în Java poate rula sub orice sistem de operare; este limbaj simplu, performant şi sigur.

3.1.3. Mediu de dezvoltare

Mediul de dezvoltare utilizat în crearea aplicaţiei este Eclipse-ul, versiunea 3.1.1.

3.2. Implementarea modulelor

Detalii de implementare a modulelor:
Descrierea pachetelor:
Pachetul Basics:
Conţine clasa Book care este o structură pentru menţinerea informaţiilor despre o carte: autor, titlu etc. Conţine metode de setter si getter precum şi următoarele metode mai speciale:
contains – decide dacă o carte conţine (în titlu, autor, sau orice alt câmp) textul primit ca şi parametru.
populate – populează câmpurile folosind datele din parametrul de intrare, care este de tipul Element (din jdom.jar) şi reprezintă un nod xml de tip book. Câmpurile corespunzătoare din nodul xml sunt stocate în câmpurile clasei Book.

Pachetul controller: Conţine clasa Controller. Principalul rol al acestei clase este să asigure legătura dintre View si Model. Clasa se înregistrează ca ActionListener la fiecare element activ al View-ului (respectiv fiecare menu item) şi ascultă după input-ul utilizatorului. În momentul în care se alege o opţiune din meniu, controller-ul este cel care preia această acţiune şi decide răspunsul: afişarea unui dialog, închiderea / deschiderea unui fişier xml, salvarea datelor în fişier xml, etc.
În afară de constructor şi metodele addView şi addModel prin care se spune controllerului care este view-ul respectiv modelul cu care trebuie să facă legătura, singura metodă a clasei Controller este metoda moştenită de la ActionListener, respective actionPerformed. În această metodă se verifică de unde provine acţiunea utilizatorului, şi se afişează dialogurile necesare. Controllerul trimite de asemenea mesaje către model atunci când este necesară o modificare a datelor din model (de ex. Se şterge o carte, se deschide un nou fişier xml etc)

Pachetul main:
Conţine o singură clasă, BookMain. Este punctul de intrare în aplicaţie, în care se crează modelul, view-ul şi controllerul, şi se fac legăturile dintre ele. Prin metoda addObserver a modelului (care extinde clasa java.util.Observable) se adaugă view-ul ca şi observator al modelului. Acest lucru înseamnă că, de fiecare dată când modelul îşi modifică conţinutul (adică primeşte un mesaj de la controller să facă acest lucru) el îşi va notifica observatorii, apelând metoda update() a acestora. În acest moment, observatorii (în cazul nostru, avem un singur observator şi anume view-ul) ştiu că va trebui să îşi modifice reprezentarea grafică a datelor (respectiv în cazul nostru, tabelul Jtable va trebui să facă refresh).

Pachetul model:
Conţine două clase, BookModel şi BookTableModel. BookModel este modelul general al aplicaţiei, iar BookTableModel este folosit doar fiincă Jtable din Java are nevoie de o clasă care derivă din AbstractTableModel. Deci, BookTableModel este model-ul pentru Jtable-ul folosit pentru afişarea datelor.
Metode bookModel:
AddObserver – se adaugă un nou observator către model.
NotifyObservers – prin prima instrucţiune din metodă, super.notifyobservers() se face apelul metodei update() din fiecare observator. (metoda notifyObservers din clasa abstractă Observable va face acest apel).
LoadData() – citeşte datele din fişierul XML şi creaza un arraylist cu elemente de tip Book care conţin datele despre cărţi.
SaveData() – operaţie inversă loadData(). Din arraylist se generează un Document XML, care, folosind clasa XMLOutputter din JDOM se va scrie în fişier.
AddBook, deleteBook – se adaugă respectiv se şterge o carte. Aceste operaţiuni presupun o modificare a modelului, deci de fiecare dată se aplează metodele setChanged (care anunţă faptul că s-a modificat modelul) şi notifyObservers (care trimite notificările către observator).

BookDeleteDialog – o clasă care extinde Jdialog, reprezintă fereastra în care apar cărţile pentru ştergere.

BookDetailsDialog – o clasă care extinde Jdialog, reprezintă o fereastră de dialog în care se pot introduce datele despre o carte.

BookListDialog – o clasă care extinde Jdialog, afişează o listă de cărţi într-o fereastră. Lista de cărţi se primeşte în constructor ca arraylist cu elemente de tip Book.

BookMenuBar – construieşte meniul aplicaţiei.

View – extinde Jframe, construieşte fereastra principală a aplicaţiei, crează Jtable-ul pentru afişarea datelor despre cărţi. Are metoda update(), care este apleată în cadrul notifyobservers din BookModel. În această metodă, View-ul citeşte din nou datele, după care face un revalidate() pe panoul care conţine jtable-ul pentru a se face refresh (dacă nu apelăm revalidate(), se face refresh doar când se modifică cu mouseul dimensiunile ferestrei principale).

Descrierea în detaliu claselor se va face în javadoc-ul ataşat proiectului.

Biblioteci folosite la implementare:

- org.jdom.*;
- java.io.*;
- java.util.*;
- javax.swing.*;
- java.awt.*;

3.3. Gestiunea resurselor aplicaţiei

Organizarea structurii de fişiere/directoare a aplicaţiei:
Library:
bin:

basics:
Book.class
controller:
Controller.class
lib:
jdom.jar
main:
BookMain.class
model:
BookModel.class
BookTableModel.class
view:
BookDeleteDialog.class
BookDetailsDialog.class
BookListDialog.class
BookMenuBar.class
View.class
Library.bat – fişierul executabil

doc:
conţine Javadoc-ul

lib:
jdom.jar (parserul pentru procesarea fişierului XML)

log:
Logfile.1
Snapshot.1
Version_Number

src:
basics:
Book.java
controller:
Controller.java
main:
BookMain.java
model:
BookModel.java
BookTableModel.java
view:
BookDeleteDialog.java
BookDetailsDialog.java
BookListDialog.java
BookMenuBar.java
View.java

xml:
books.xml (fişierul XML care conţine datele referitoare la cărţi)
.classpath
.project

Fişiere de tip resursă: Fişierul XML books.xml.

4. Specificaţii de gestiune a proiectului (Repartizarea sarcinilor şi a taskurilor pe membrii echipei)

4.1. Working package – Management (Gestiune)

Denominare	Realizare	Relectura	Validare
Caiet de sarcini	Ansamblul echipei de proiect	Maxim Diana	Maxim Diana & Kiss Zsuzsanna
Caiet de specificaţii	Ansamblul echipei de proiect	Maxim Diana	Kiss Zsuzsanna
Extras de reuniune externă	Mocanu Voica	Kiss Zsuzsanna	Ansamblul echipei de proiect

4.2. Working package – Realizare

Denominare	Realizare	Validare
Raport de concepţie a arhitecturii globale a aplicaţiei	Mocanu Voica	Maxim Diana
Interfata grafică GUI	Kiss Zsuzsanna	Mocanu Voica
Modulul gestionare XML	Maxim Diana	Kiss Zsuzsanna

4.3. Time Planning (planificare în timp)

Tabel cu task-urile realizate (membrii echipei) şi numărul de zile alocate pentru implementarea/executarea fiecărui task:

Kiss Zsuzsanna	View	5 zile
Maxim Diana	Model	5 zile
Mocanu Voica	Controller	4 zile

Bilanţ aproximativ de zile/ore lucrate pe proiect/membru:
Kiss Zsuzsanna: 60 de ore.
Maxim Diana: 50 de ore.
Mocanu Voica: 50 de ore.

5. Specificaţii administrative

5.1. Confidenţialitate la nivelul codului sursă / drepturi de autor

Codul sursa este distribuit sub licenta GPL (General Public License).

5.2. Datele de livrare ale aplicaţiei/Documente administrative furnizate

Documentele furnizate la client sunt: un dosar cu documentele « Caiet de sarcini», « Caiet de specificaţii», Javadoc cu descrierea detaliată a claselor.

6. Bibliografie/Referinţe/Documentaţie

1. Ştefan Tanasă, Cristian Olaru, Ştefan Andrei: Java de la 0 la expert, Polirom, Iaşi, 2003
2. Brett McLaughlin: Java and XML, O'Reilly, 2001
3. http://www.jdom.org
4. http://www.jdom.org/docs/apidocs/org/jdom/
5. http://www.oracle.com/technology/oramag/oracle/02-sep/o52jdom.html
6. http://java.sun.com/docs/books/tutorial/
7. http://java.sun.com/docs/books/tutorial/uiswing/components/table.html#editrender
8. http://lightuml.sourceforge.net/

7. Anexe

Anexa1. Diagrama UML globală a aplicaţiei.