# Integration of the POI API with Java for Information Processing from Heterogeneous Sources

Pedro Galicia Galicia

Computing Research Center. National Polytechnic Institute.
México City, México
pgalicia@cic.ipn.mx

**Abstract.** Nowadays, there is a growing need for people to search for specific information. Frequently, the relevant information is stored in different formats such as Excel, Access, Word, etc. An additional complication is that diverse data can be located in a single file, and pieces of the same information can be contained in different files. Often people that accomplish data processing tasks have to extract, transform, and load information from Excel format files to a database environment. The Apache POI Project offers APIs to access files in the Microsoft OLE 2 Compound Document format, including .XLS format files. To solve the problem of interoperability between Excel and databases, we propose to integrate this API with Java technology, obtaining a technological combination powerful and easy enough to be combined with existing applications. We consider the case of information management and dynamic retrieval in an enterprise, in a setting where several departments work together within an integrated information environment.

**Keywords:** Information process, ETL process, Java, EXCEL, Apache POI API, MySQL, XAMPP, TOMCAT, JDBC.

## 1 Introduction

Since many years ago enterprises have had an urgent need for data integration related with areas such as finance, management, production planning, production and sales. Currently there is a great demand from the users to obtain accurate and timely information for decision-making. Often people that accomplish data processing tasks have to extract, transform, and upload information from Excel format files to a database environment. All of this is because the data processing environments tend to be more complexes, and decentralized, and have to incorporate flexibilities and the modularity, using new technologies. To focus these requirements is needed to innovate and generate new forms of data processing considering proper handling of the information, and Excel-Java-Database combination is another option. The extraction of data residing in Excel format files, as a result from business activities, to process it and upload it into a corporative database is the main goal.

adfa, p. 1, 2011.



Advances in Computer Science and Engineering
Research in Computing Science 58, 2012 pp. 183–196
Paper Recived 01-10-2012 and Acepted 07-11-2012

## 2 The information analysis

Enterprises base their performance on the information management that different areas produces and for this reason have to process a lot of data contained in several files such as Excel format files, and as result of this, users have to manage many data in different ways. The problems arise when the information contained in Excel format files is required for several areas at the same time and the data into spreadsheets turns in very huge files to be interchanged.

Hence information that resides in those Excel format files is copied into different computers with the risk to be processed or updated in a wrong way. This can create a disagreement conflict between the enterprise areas about the decision making that can be solved with a corporative database; however there is a situation, the interoperability among Excel format files and the Databases.

To solve this problem we propose to integrate the POI API with Java technology and a Database manager as MySQL to obtain a technological powerful and easy enough combination jointly with existing environments as Apache Web server and TOMCAT application server for instance. Therefore to let's briefly describe the Excel files and features of POI API.

### 2.1 Excel Files

**Excel Files.** These are files based upon Microsoft format to files Compound Document format OLE 2 (OLE2CDF) [1], and OLE2CDF file contains a complete filesystem, laid out using nested Entries Directory, that contain Entries. We are interested in Entry elements of the Entries Document type and an Entry Document contains data structures of an application-specific (e.g. Excel) [2].

This kind of files are known as workbook type, and each workbook can include multiple spreadsheets either empty or with information. These document types are known as Excel format files, and have an extension type ".XLS" or "XLSX".

Excel files supports two basic types of spreadsheets to contain text or numerical data, and can be used to develop calculations and to contain formulas or graphs, also are used to display information by using multiple schemes charts (bar charts, pie, 3D, etc.).

**Spreadsheet Documents.** This type of document basically consists of rows and columns, which contain several data types, for instance the case of a normal spreadsheet that can contain values and formulas, graphs, images or macros.

These documents also contains a structure and its records are bounded mainly by the type BOF (Begin-Of-File) and EOF (End-Of-File), also includes information about dimensions, view, fonts list that contains, list of names and external references, and the format of the cells, the width of the rows and the height of the columns.

## 2.2 POI API

**POI API.** The Apache Software Foundation [3] includes into its Project Listing, the Apache POI's Project, the mission of this last one is to create and maintain JAVA APIs for manipulating various file formats based upon the Office Open XML standards (OOXML) and Microsoft's OLE 2.

The Apache POI's Project, contains the POI (Poor Obfuscation Implementation) API [4] this is based in a pure Java API and allows manipulate various file formats. This API facilitates the reading and writing Excel (HSSF), Word (HWPF) and PowerPoint (HSLF) files.

For purposes of this paper we will focus on the part of HSSF and its interaction with POIFS component, needed to create or read documents .XLS or .XLSX.

The module HSSF (Horrible Spread Sheet Format) provides several packages, as shown in Table 1, and allows interacting with Excel type documents and also creates other new ones.

**Table 1.** HSSF Packages

| Package | Function |
|---|---|
| Org.apache.poi.hssf.eventmodel | Handles different events generated in the processes of reading and writing Excel documents |
| Org.apache.poi.hssf.eventusermodel | Provides classes for the process of reading Excel documents |
| Org.apache.poi.hssf.record.formula | Contains classes to handle FORMULA used in Excel documents |
| Org.apache.poi.hssf.usermodel | Contains classes for creating Excel documents |
| Org.apache.poi.hssf.util | Common environment to handle different kinds of documents attributes Excel |

Figure 1 shows an excerpt from the HSSF module and its Usermodel class diagram.

Another use of the Apache POI API is for text extraction applications such as web spiders, index builders, and content management systems.

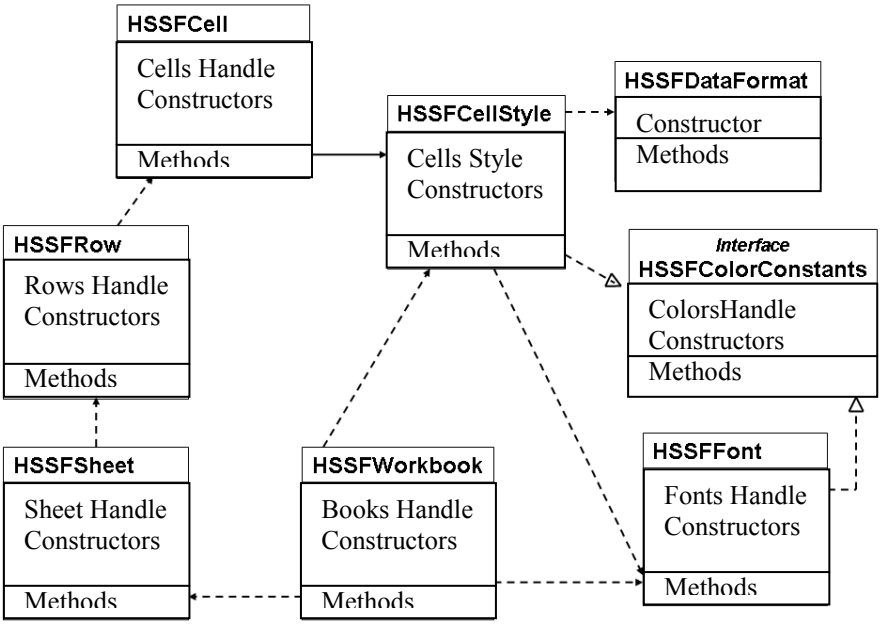


**Fig. 1.** Excerpt from the Usermodel Class Diagram HSSF[1].

Important aspects:

- The workbook is represented by the HSSFWorkbook class.
- Each worksheet in the document is represented by the class and creates HSSFSheet from HSSFWorkbook class through: object_HSSFWorkbook.createSheet ("Sheet name").
- Before making a cell reference must reference the row.
- The reference to a row is represented by the class HSSFRow.
- The row is obtained from the sheet (HSSFSheet).
- Cell HSSFCell is obtained from the row (HSSFRow).

Create a book involves several actions as create sheets, and each sheet creates rows, and cells, and settings to contain data, for instance such as the formula's type for a Total or a region for data capture, finally the created document must be saved.

[1] Usermodel Class Diagram by Matthew Young. poi.apache.org/spreadsheet/diagram1.html

Although there are other applications [5] to read and write Excel format files, for the problem posed in this article we have chosen the option to develop an application based in the combination of POI API with Java, jointly with MySQL and supported in a Web environment.

## 3 Design of the application

So far we have presented an overview of the features of Excel format files and POI API, and we have mentioned others development environments based on the management of these format types. Now, let's set the initial specification of the application, how to get the input data, user profiles and technological support required as well as the objectives design and to identify the actors involved.

### 3.1 Initial specifications

The main goals of the application are: to retrieve data from Excel format files, to upload their references into a MySQL database, to process the retrieved data and update the relevant information into the test database, to perform information queries, to manage access according to the user profile, as well as the users registration and the Excel resource registration too.

Therefore, it is necessary to define and configure the test database to incorporate information from Excel files and from the users; as well as the design and develop the graphical interfaces for communication client/server, the reports, queries and the data validation. At this step the application development [6] considers: the requirements analysis, design, development, testing and install, additionally could be included the maintenance and operating manuals.

### 3.2 Graphical User Interfaces (GUI)

GUI should have:

- Security, users access the application via a user name and password respectively registered in the database, this will give them access according to the user profile.
- Users management, users have to register data such as user name, user profile identifier (login) and password (password).
- Interaction with the user must have Web interfaces in each modules of the application.
- Show the results of the updating, modifying, or deleting records process, respectively.
- Show options for the start and stop of a user session.
- Queries information.

### 3.3 Technological support

An application development like this considers the following elements: have a connection to a public or private network communication (Internet), a GUI scheme of access via a Web Browser, a test database MySQL, usage the POI API, the Java language programming [11] and its specifications JSP and JDBC, an a based pattern design architecture as the Model-View-Controller, as well as a MySQL database manager, a XAMPP Web Server, and a TOMCAT Application Server, and for this work the web and the application server will be installed at the same computer.

### 3.4 Application Architecture

When developing applications with Beans and JSPs, is required to identify and separate the presentation logic and business logic, for which we need to use a standard software architecture that separates the application data, the user interface, and the logic control into three distinct components, the architecture pattern is called Model-View-Controller (MVC) [7].

Generally, while the application is operating via GUI via the Web server [8], the application server [9] considers environment operating requirements, and business logic is provided by the application components. In fact, these last ones components perform user validation and where appropriate, the needed tasks to establish the connection to the MySQL [12] database manager to execute the respective operations in the database environment. Figure 2 depicts the operation of the architecture considered for this application.

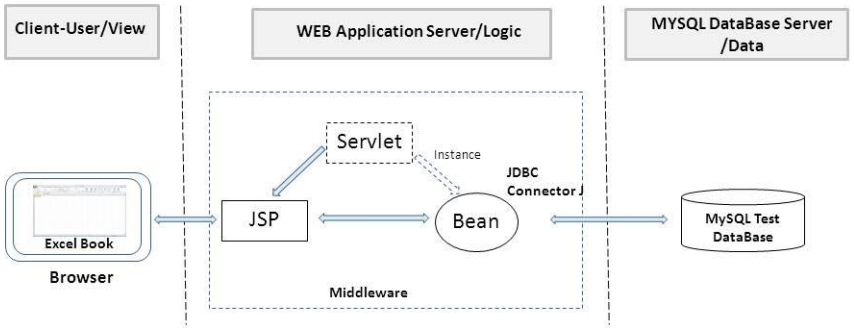


**Fig. 2.** Application Architecture

Modeling the Test Database [10] is considered at the corporate level as a study case and focused to defining appropriate information resources.

### 3.5 Application Packages

Several packages are part of this application for the management and dynamic information retrieval from heterogeneous sources (MDIR). These packages are composed of different classes that provide the necessary functionality to the application. The corresponding diagram is shown in Figure 3.

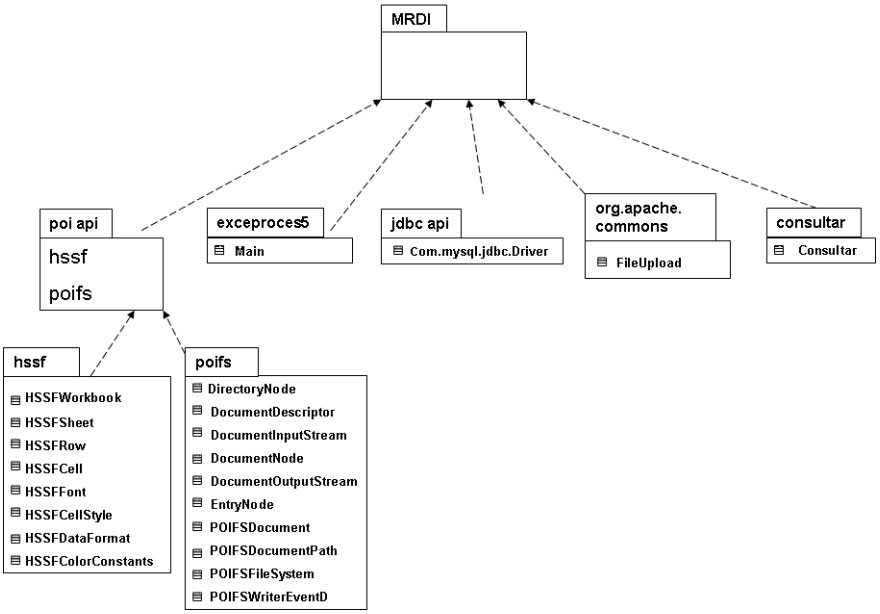


**Fig. 3.** Diagram packages that integrate the application

## 4 Application Development

The purpose is to solve the problem mentioned at Initials Specifications paragraph. For this application has defined two types of actors [13]. The first one is the basic user to use the Excel module resources, and the second one is the management user that can use the other modules that make up the application.

The following context diagram shows the use cases for, see Figure 4.

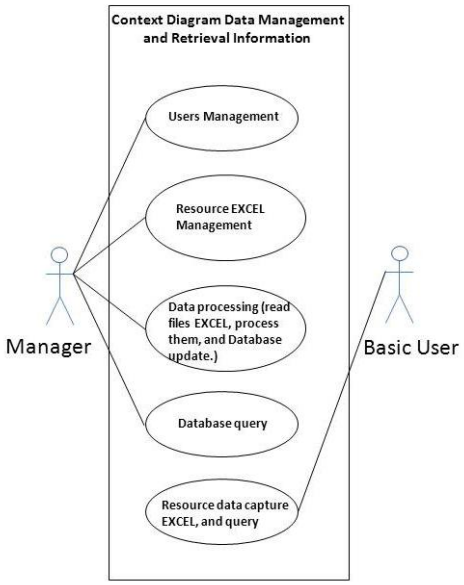


**Fig. 4.** Context Diagram Data Management and Retrieval Information.

### 4.1 Outline of the operation

The flow pattern information processing includes: the data collection process and data capture in Excel format files (events held outside the application), the access to the application via the Web browser, to perform user authentication via the login and password processes. When the user is a basic user only can access the Excel module resources to data capture and upload the information that resides in Excel files to the database and consult. When the user is a manager user can use the management, processing and queries modules, The Management module allows the users management and the Excel resource management, the Processing module allows the user to choose the Excel files that will be read and to execute the process for retrieving data and updating the relevant information into the test database, and the Queries module allows to generate the queries for each user.

The process operating diagram shows the dynamic part performed into a web environment in which to start a session, the user must be validated according to its level of access, see Figure 5.

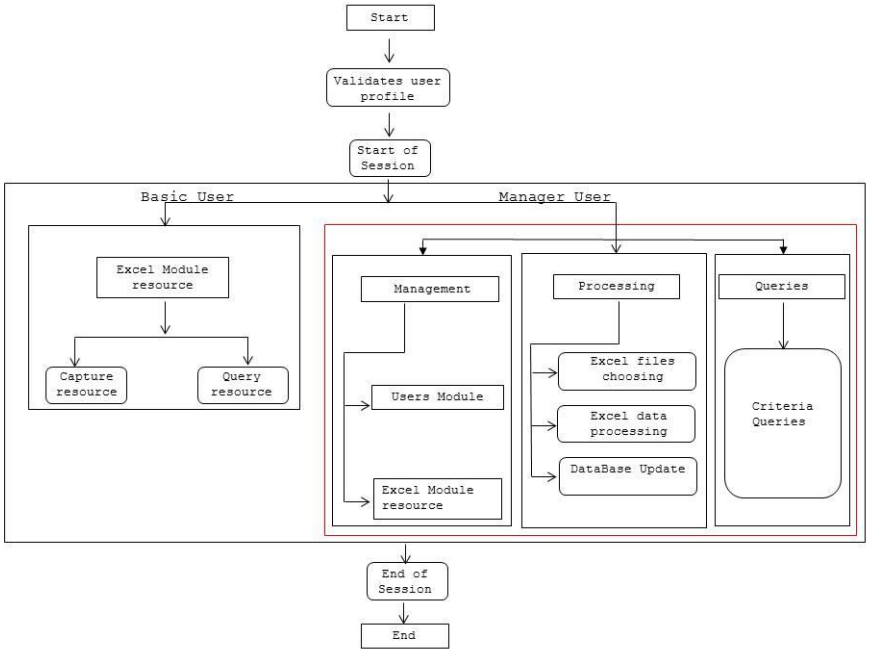


**Fig. 5.** The process operating diagram that shows dynamic part of the process

## 5 Test and Results

This section presents the graphical interface for access the application as well as some the modules that make up. Now let's see the operation tests and results assessment. Here we have a corporate testing database, which in fact has already been modeled, and which we have added new tables related to user management, Excel resource, and other components like budgets. Figure 6 shows the graphical interface to access the application, which validates the user name and password against the database.

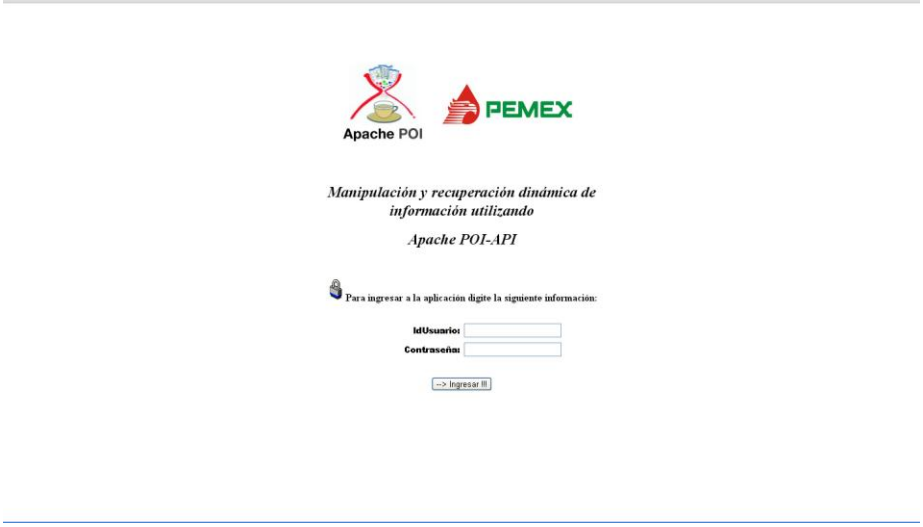


**Fig. 6.** Graphical Interface Access

The upload of Excel files to database is important because this task has to be checked it up to verify the load time of the Excel file to the server. The upload time can be checked from both a local and remote environments that emulates the uploading time. For this test the respective code was implemented to the loading process to measure the size of the latter. Table 2 shows uploading results.

**Table 2.** Uploading time to the server of resource files Excel.

| No. | File Name | Size (KB) | Local uploading time (seg.) | LAN uploading time (seg.) |
|---|---|---|---|---|
| 1 | Capturador_prueba1 | 1,851 | 0.015 | 0.016 |
| 2 | Capturador_prueba2 | 1,979 | 0.016 | 0.016 |
| 3 | Capturador_prueba3 | 1,995 | 0.016 | 0.018 |
| 4 | Capturador_prueba4 | 2,110 | 0.032 | 0.035 |
| 5 | Capturador_prueba5 | 1,853 | 0.015 | 0.016 |

Figure 7 shows the graphical interface for import Excel data, where the user choose files to use and the application displays the path the file as well the information retrieved, and the amounts of the global costs, which are shown in red color, to be upload into the database.

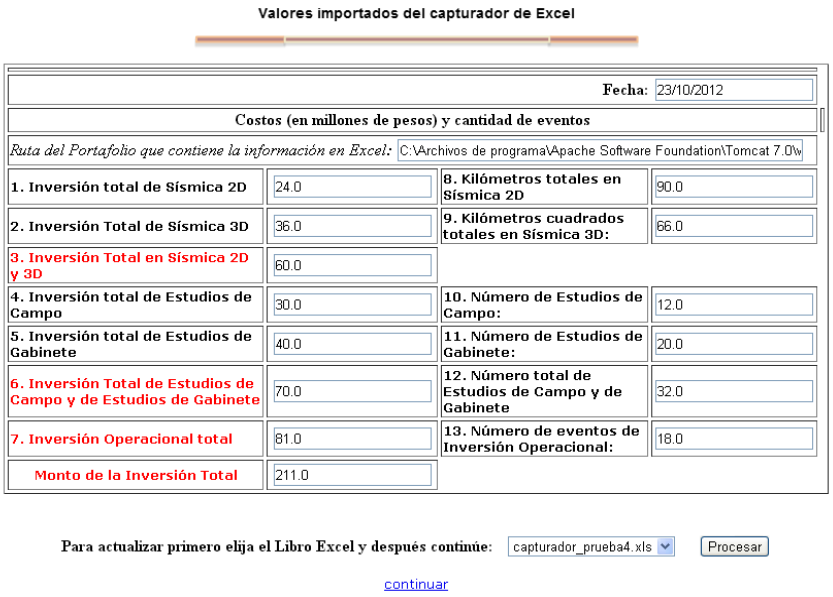


**Fig. 7.** Graphical interface for the import process of Excel files

The graphical interface to see the results is shown in Figure 8.

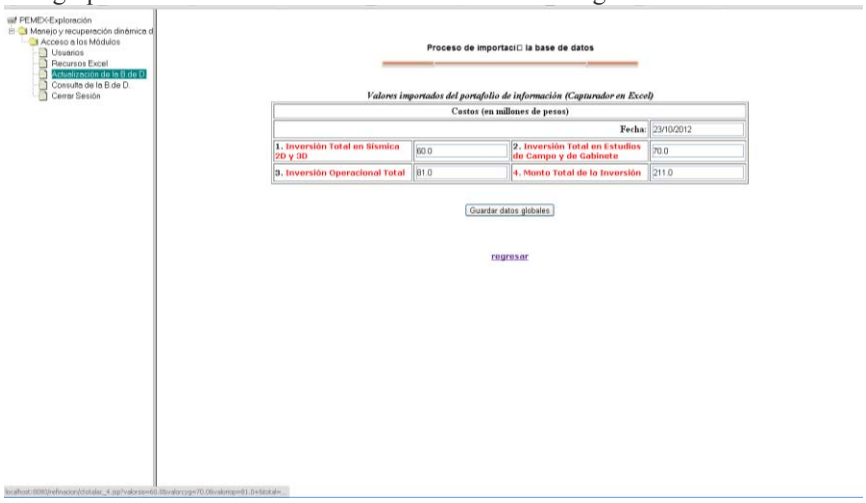


**Fig. 8.** Graphical interface to see the results of the import process

The graphical interface to see the results in the database is shown in Figure 9 in case of different update processes; the application can to accumulate each amount respectively.

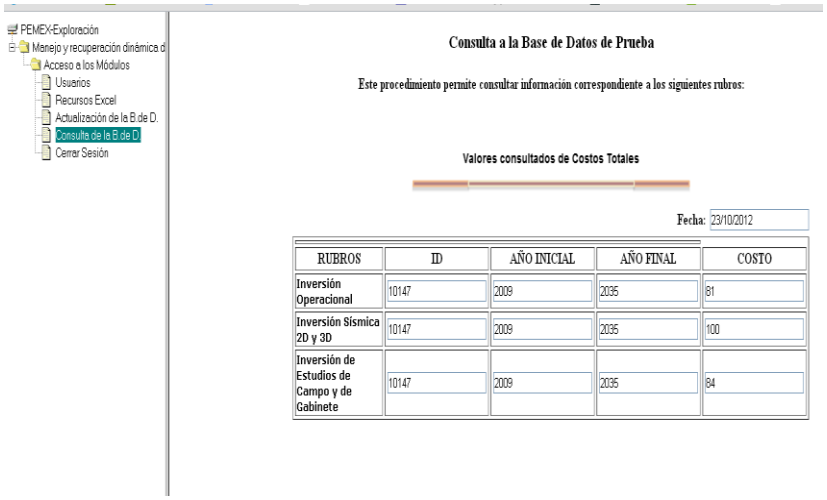


**Fig. 9.** Graphical interface to query the database to see the results of the update

## 6 Conclusions

Enterprises are integrated of different areas, which perform several operations such as the consolidating operations; this involves its consolidation, through registration and dynamic management of data, by using appropriated tools.

This is why the data processing environments tend to be complex, decentralized and changing, and so are required to incorporate flexibility and modularity, using new technologies.

The development of this work is aimed to solve the problem before described, to establish an application environment that allows combining the use of tools to facilitate the Excel-Java-Database combination, and the POI API allows working this type of scheme at a lower cost supported in the use of free technologies.

The application presents an option to processing simulation at low cost, readily available resources and consistent with rapid and appropriate integration with existing processes, supported in an architecture based on widely recognized standards, such as the pattern called Model-View-Controller (MVC), allowing a greater versatility and facilitates interoperability with other applications.

The operation of the application is not at a disadvantage with the processes that could exist, since it is based on a test environment and easy scalability, supported by the integration of Java with other technologies such as Excel and MySQL, which together allow for greater versatility.

## Acknowledges

We appreciate the support of the Computing Research Center and of the National Polytechnic Institute, México, to carry out this work.

## References

1. Rentz, Daniel [et al], OpenOffice.org's Documentation of the Microsoft Excel File Format. http://sc.openoffice.org/excelfileformat.pdf.
2. Sengupta, Avik. Oliver Andrew. Opening Microsoft file formats to Java. http://onjava.com/pub/a/onjava/2003/01/22/poi.html
3. Apache Software Foundation. http://www.apache.org.
4. API POI (Poor Obfuscation Implementation**)**. http://poi.apache.org.
5. Java Libraries to read/write Excel (.XLS) document files. http:// Schmit.devlib.org/java/libraries-excel.html.
6. Metzger, Philip. Administración de un proyecto de programación. Trillas 1978.
7. Design Patterns Model-View-Controller. http://java.sun.com/blueprints/patterns/MVC.html.
8. XAMPP for WINDOWS. http://www.apachefriends.org/en/xampp-windows.html
9. Apache TOMCAT. http://tomcat.apache.org
10. James, Martin-1933-Computer data-base organization / James Martin -- 2a.Ed. – Englewood Cliffs, N.J.: Prentice-Hall 1977.
11. Deitel, Harvey M. Java: how to program / H.M. Deitel, P. J. Deitel -- 2a.Ed. -- Upper Saddle River, NJ : Prentice Hall 1998.
12. MySQL. http://www.webestilo.com/mysql.
13. Schmuller, Joshep. Aprendiendo UML en 24 horas. PEARSON EDUCACIÓN. México, 2000. 1ª. Edición.