Secure, scalable and component-based Webshop using Struts and Hibernate

L U C A S   D Í E Z

Master of Science Thesis
Stockholm, Sweden 2006

ICT/ECS-2006-96

Secure, scalable and component-based Webshop using Struts and Hibernate

L U C A S D Í E Z

I n d u s t r y s u p e r v i s o r
V i k t o r L e w i n D u r a n
(G e n X H a n d e s b o l a g)

E x a m i n e r
A s s o c . P r o f . V l a d i m i r V l a s s o v
(E C S / I C T / K T H)

Master of Science Thesis
Stockholm, Sweden 2006

ICT/ECS-2006-96

Dedicated to

my Father, for educating me as an Engineer

Abstract

This Master Thesis project concerns the design and implementation of a Webshop following the architecture and patterns of enterprise applications. The main idea was to create a scalable, internationalized, secure, fault tolerance and component-based application.

The application was developed following the Model-View-Controller (MVC) design pattern. This was accomplished by using Struts, a framework designed for developing J2EE web applications. The Model component was developed following the Hibernate technology. It offers an easy to use framework for mapping an object-oriented domain model to a traditional relational database. The Webshop can be accessed not only by human interface via web pages, the application supports interoperable machine-machine interaction over Internet through Web Services. The Webshop prototype has been tested against use cases, evaluated and profiled.

Keywords: Webshop, Struts, Hibernate, WebServices, MVC, J2EE, JSP.

Declaration

The work in this thesis is based on research carried out at GenX Handesbolag, Stockholm and at KTH (Kungliga Tekniska högskolan - Royal Institute of Technology), Stockholm. No part of this thesis has been submitted elsewhere for any other degree or qualification and it all my own work unless referenced to the contrary in the text.

Acknowledgements

This Master's Thesis is a result of my studies as a double degree student in Sweden (Royal Institute of Technology, Stockholm) and in Spain (Universidad Politécnica de Madrid). Many people supported me in those studies, most of them friends and family.

I would also like to express my sincere gratitude to my KTH Supervisor, Associate Professor Vladimir Vlassov at the Department of Electronic, Computer and Software Systems (ECS), Royal Institute of Technology, for his support and guidance through the whole thesis process.

Words alone cannot describe how much I thank Viktor Lewin Duran, my Industry Supervisor at GenX Handelsbolag in Stockholm, for all support and for coming up with the idea that led to this Master's Thesis.

Regardless of being far away my second year in Stockholm, I would like to thank, with all my heart, Paula, who has encouraged me in the bad moments, and shared with me all the good ones.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

GenX is a small company which has some customers with the need of an electronic commerce system. Putting together all of their requirements, this company figured out that it can be done as a Master Thesis where it could be created a web shop following the architecture and patterns of enterprise applications. The goal of this project is not to create an ordinary web shop since that it is too simple, during this introduction we will motivate the different technologies we have chosen and the work amount invest on the shop.

1.1 Motivation of the project

Today there are thousands of web shops but few of them are easy scalable, the internationalization is hard and the development time is too big. To motivate this thesis we are going to work on the following properties of the shop: *scalable, internationalized, secure, fault tolerance and component based.*

Concerning scalability, we mean that the application must handle growing amounts of work in a graceful manner. The idea of this project is adding more and more features and this can be only achieved through a scalable application. The application should also be available for different languages in an easy way due to it will be destined to sell product to people from different countries at the same time. Because the sales are going to be using real money, the application must be secure with the purpose of avoiding illegal actions. It also has to trace and log the errors, and then report them to the administrator, so that it will be fault tolerance. The last main requirement of the project is the necessity of developing a component based application in order to minimal the impact to the other components if a change in one of them is made.

This implies that the application has to be separated into several layers, and the code can not have any hard coded information, otherwise the amount of work for the developer would be increased exponentially. Since we use components in the application each of them must implement some kind of interface.

One important part is the presentation layer; this component presents the data to the customer. To make it free from the code we are going to use tags that make the JSP files able to change the design without affecting the java code. This involves more work for the developing part because now we have to create tags to solve every task such as the presentation of the products where several products are presented though an iteration of tags. Nevertheless this amount of work is minimum compares to adding and updating java code in the JSP files.

The next part will be the validation of the data sent by the user. To avoid faults or make the system unstable all the processed data must be validated in the server. The reason we eliminate the validation at the client is that there are several ways of shutdown the validation such as deactivate the java script for example.

Concerning the logic of the program, we have used java classes with Hibernate capabilities to communicate with the database. The reason of choosing Hibernate instead Enterprise Java Beans are several and explained in detail in section 2.5, now the main reason is the caching possibilities of Hibernate and the transaction support. Another reason is that Hibernate does not need any special web server in contrast to J2EE and besides, Hibernate is open source that is distributed under the GNU, which decreases the price of the project and helps to offer a cheaper product.

1.2 Problem statement

The main problem for this application is to keep all of the above requirements in each component of the application, so in case the shop is sold to another company, the only thing to do is to reconfigure the components instead of rewriting the code of the application.

Another goal is if the customer wants new functionalities or components for the application, it is only necessary to add or integrate with other component without changing any of them thanks to the different interfaces for the objects.

We are also concern about the different clients to the shop, one is already mentioned which is the human interface through web pages; the other is a Web Service where the information is exchanged through messages passing. Several Web Services have been created for improve the webshop such as consulting the shipping rates. We will explain it with more details later.

Nowadays all commerce in the World Wide Web has to implement a high security. The application has several features regarding security and is prepared for adding SSL in order to assure the privacy of all transactions inside the shop.

1.2.1 Goal

There were two main goals for this Master Thesis. The first one was to design and implement the Webshop application which follows the requirements exposed in Section 1.3. Secondly, to state some conclusions and reflections about how appropriate the use of Hibernate and Struts technologies are for the development of the Webshop.

1.2.2 Expected results

The application fulfills the requirements exposed in Section 1.3, and it may be happened that the customer changes the requirements or needs more features, but the main requirements have been maintained during the whole project, so the web shop kernel is assured. The expected results of this project are also the survey of different approaches, the uses cases attached in the Appendix B, the Vision of the project which the Chapter 3 is based on, the study of different database technologies, the architecture design of the database and the application, a prototype of the application and finally an analysis process which validates the application against the use cases, measures the performance and profiles the prototype.

1.3 Shop requirements

1.3.1 Scalability

When we talk about scalability it can be many things, it is difficult to define. We can say that the scalability is the property of the application which specifies its ability to either handle increasing amounts of work in a graceful way, or to be readily enlarged [1, 2].

In terms of hardware scalability, the main idea is to build the shop scalable in both vertical and horizontal ways. To scale vertically means to add resources to a single node in a system, such as adding more or faster hardware. To scale horizontally means to connect multiple entities so that they work as a single logical unit. The vertical scalability is solved by increasing computing resources within a server in case that the shop becomes popular. When loads grow beyond the capacity of a single server, multiple servers can be added in order to increase the horizontal scalability.

In terms of software scalability, the application is based on Struts framework [3]. Struts is a robust and powerful framework which promises scalability, reusability and separation of responsibility. This framework encourages a *layered* design for applications. This design helps make applications both scalable and robust [4]. Struts uses the Servlet API, which runs multithreaded for performance due to Java supports intrinsically threads. Using threads we increase performance and scalability.

1.3.2 Security

The security is one of the main points in this application since the transactions are going to be with real money. The first security feature is authentication, which only allows buying things to persons that are registered as customers. The second one is authorization. After the login process, there are different roles and each of them has different permissions and restrictions, this is for avoiding customers from making errors or faults in the systems.

Traceability. Every transaction and action will be logged into the system both into the application container and into the internal log of the application, so in case a bug or a lack of security allows a customer or hacker to perform undesired activity, it can be traced back. With this feature the shop can be updated faster than if the developer has to search into the code for bug or errors.

1.3.3 Fault tolerance

The shop will implement different mechanisms to trace and log the errors, each error will be graded. Depending on the serious of the error the administrator or the shop manager will be informed in the case that the problem could not be solve by the application.

The fault tolerance is important to accomplish the different tasks of the shop, if a customer can not purchase an order which can be only fixed by the administrator, it will immediately report by email or SMS so he can solve the problem.

1.3.4 Component based

The shop will implement many functions; each of these functionalities will be implemented as a component so that in case of a functionality is old or no needed anymore, it can be removed or added without affecting the other application parts.

1.3.5 Internationalization

Another important characteristic supported by the Struts framework is the internationalization of applications. Using this feature the application is available in different languages depending on the country the request came from. This could be happened since the browsers send information about the language and the dialect within the request. We also concern about the format that the different languages are presented in the webpage.

1.4 Structure of the Thesis

This report is based on seven chapters. The Chapter one introduces the thesis exposing the motives of this thesis as well as the problem statement, the expected results and the shop requirements. The Chapter two gives some background with the intention of understanding the concepts shown in this report and to show what others have already done, including related work, existing approaches, related technologies, the MVC patter, Struts framework, the database technology using Hibernate, Web Services and a brief presentation of the system prototype including some screenshots. The Chapter three offers a Vision of the Project and presents the main functionalities of the system prototype. The Chapter four describes the design of the MVC (Model-View-Controller) pattern, following by an example for developing Java enterprise applications, Struts. We continue giving details about our connection to the database using Hibernate framework. Then, we introduce some knowledge about Web Services architecture and finish putting together all the technologies in order to describe the design of the whole application in Section 4.5. The Chapter five illustrates important details about the implementation. The Chapter six analyses the application with performance and profiling. In the end, the Chapter seven provides some conclusions and future work.

Chapter 2

Background

This Chapter gives some background with the intention of understanding the concepts shown in this report and to show what others have already done, including related work, existing approaches, related technologies, the MVC patter, Struts framework, the database technology using Hibernate, Web Services and a brief presentation of the system prototype including some screenshots.

2.1 Related work

Nowadays there are many different kinds of webshops using all possible technologies. We have tried to develop a web shop with all the good properties from the most used on-line shops. If we take a look at existing approaches, there are several open source webshops [5, 6, 7, 8, 9], these approaches are interesting from the develop perspective, since the business idea is to sell the service, so if the code is published anybody can copy and offer the same service.

Putting together our researches in webshops like Amazon or eBay and the customer necessities, that gave us a reference point to write the webshop requirements in Section 1.3.

We have also found searching trought Internet an open-source webshop project using Struts framework called eSarine [9]. This project is specially designed for small enterprises, which is different to our expectations; we will try to develop a more scalable application. This webshop uses like our project the MVC (Model-View-Controller) pattern which gave us at the beginning a good example to design the application. Finally, we decided not to focus on that project since the Model component is completely different because we are using Hibernate technology and managing persistent objects instead of using JDBC to carry out the connection to the database. We also concern about the security and e-Sarine uses Container-Managed security i.e. roles for Actions. We not only use those techniques, but we use Application-Managed Security, i.e. using Struts' Request Processing, Cookies and security inside the JSP files.

One of the advantages of using MVC is that it is extensible in different ways. For example, if we want to export this project to other platforms such as mobile devices, we only have to adapt the presentation layer according to the mobile standard. Related to this point there is a Master Thesis "Adaptation of a Webshop for Mobile Devices" [10] which explains how an existing webshop can be optimized for mobile devices. This point of view is very interesting because it presents a solution for our webshop to be prepared for future presentation formats, just changing the View Layer.

2.2 Existing approaches

The idea of the webshop, also called on-line shop, Internet shop or on line store, has been started since around 1994 when Pizza Hut began to offer pizza ordering on its Web page [2]. During the same year Netscape 1.0 introduced SSL encryption that reinforced the security in the transactions through Internet. In 1995 and 1996 two of the most popular webshops nowadays "Amazon.com" and "eBay" were launched to the World Wide Web. Since then, thousands of webshops have been developed and nowadays almost every company which sells something has a webshop in order to make their sales easy.

2.3 Related technologies

There are many webshops designs using different technologies, but few of them are open source. If you try to look for them in Internet you can find very quickly osCommerce [5], PHPay webshop [6], PhPepperShop [7], phpShop Project [8] and eSarine [9]. Most of these solutions have been developed using PHP technology. Using PHP carries several disadvantages for some points of views. The most important could be the logic and presentation layers are combined in the same file which does not represent a good solution for future updates. To solve this problem, we have selected the MVC architecture which is provided using Struts [3]. The whole application will be developed in Java object-oriented programming language focused on the MVC architecture.

2.4 MVC - Struts

The Apache Struts project is an open-source framework [3] sponsored by the Apache Software Foundation. It was designed for developing J2EE web applications following the Model-View-Controller (MVC) design pattern. It was originally created by Craig McClanahan and in May 2000 donated to be taken over by the open-source community.

This framework provides an easy way to separate the presentation layer and allows it to be abstracted from the data layers and the transactions. Other features are the internationalization (i18n), a great custom tag library, tiled displays and form validation among others [11].

2.4.1 MVC design pattern

Struts framework is based on the MVC pattern which is a software architecture that divides the data model, user interface, and control logic from the application into three different parts. Changes to one component can be made with smallest impact to the rest.

The Model represents the data objects of the Struts application. These objects are what are going to be manipulated and presented to the user. They can be implemented as objects representations of data stored in a relational database. At this point is when Hibernate comes into action. Hibernate is an object-relational mapping (ORM) solution which permits to manipulate a database tables as Java classes. This technology is explained with more details in Section 2.5.

The View presents to the user the information of the Model. The View components are mapped to a JSP file that contains Struts custom tags, HTML and JSP. All the JSP files are controlled by the Tile framework. Using tiles allows us to develop reusable presentation components and we can classify our presentation tier to perform greater reuse of layouts, HTML, and other visual components like images and multimedia files.

The Controller defines the way that the UI responds to the user input. This component is the one in charge of manipulate the Model.

Benefits of using MVC pattern are [11]:

Reliability. We are able to change the appearance of the application without recompiling the code of the Model or the Controller components due to they are visibly separated from the presentation and transaction layers.

High reuse and adaptability. The MVC pattern allows us to present our application in different formats, all of them connecting to the same Model and Controller parts. I.e. Web browsers through HTTP protocol or mobile devices through WAP protocol.

Very low development and lifecycle costs. We can distribute development effort, with the intention that implementation changes in one module of the application do not involve changes to another.

Rapid deployment. Deployment time can be significantly reduced, because programmers focused on the Controller module can work independently of the developers responsible for the View or Model components.

Maintainability. The clearly separation of presentation and business logic become easier to maintain and modify the application.

2.5 Database technology - Hibernate

A foremost element in the project is the database. All the information is stored in this component and must be available in an effective way. The product data, images, prices, offers, payments and the rest of the information in a webshop should be display almost immediately to the final customers in order to gain their trust. Therefore the database fulfills a vital component.

The database will be designed in accordance with the relational database model using a normalization process which eliminates duplicated data in the relational database.

The election of the Database Management System (DBMS), we decided to use MySQL [12] for many reasons. MySQL is a multithreaded, multi-user, SQL Database Management System. One of the advantages is that MySQL is available as free software under the GNU General Public License (GPL), decreasing the cost of the final product. MySQL supports transactions, procedures and SSL which provides secure communications, an essential feature in e-Commerce.

The application was designed to use a centralized database organization showed in Figure 2.1. In this design all the webshops will use the same database, a version to interconnect all the information in the different future shops. The central database stores all the information from the different shops. If a shop requires information from other shop, it has to consult it in the same database.

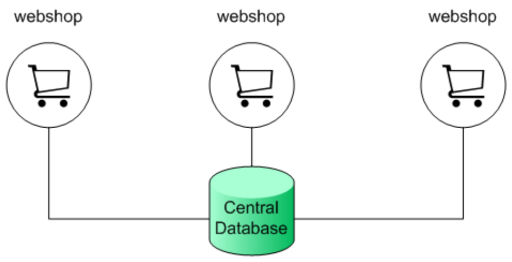


Figure 2.1: Centralized database organization

This Centralized structure is based on a centralized database that has a drawback; the database can get overloaded and become a bottleneck, as the number of client increases. To solve this problem, the project could be structured in a distributed system as the Figure 2.2 shows. Every shop has its own webshop which connects to its database looking for a data, if it is necessary to consult the other shops, the webshop will connect to the other webshop and ask for the information. This implementation will be developed in future work.

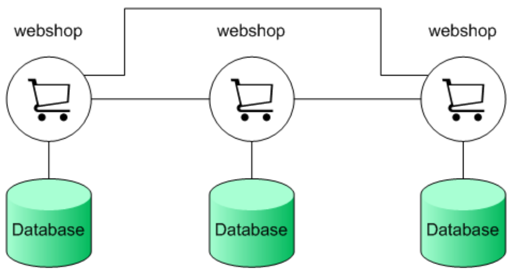


Figure 2.2: Distributed database organization

The Model layer of the application was developed by using Hibernate technology [13] instead of Java Database Connectivity (JDBC) to access to the database for many reasons exposed in the Table 2.1 - [14]. Hibernate is an Object-relational mapping (ORM) solution for Java. It is open source software that is distributed under the LGPL (Lesser General Public License). It offers an easy to use framework for mapping an object-oriented domain model to a traditional relational database. Hibernate allows us to query and persist data in a relational database through object manipulation instead of through SQL queries like JDBC.

JDBC	**Hibernate**
Most of well-known developers know it.	Easy and fast to learn.
Designed for small systems.	Good performance.
Hard coding effort.	Safe up to 95% of common data persistence related programming tasks.
Stored procedure usage.	Reduce common data persistence related programming tasks.

Table 2.1: A comparison between JDBC and Hibernate

2.6 Web Services

In order to improve the webshop we have increased the functionalities adding Web Services to the application. This means that not only the web shop can be accessed by a human interface through web pages, the application supports interoperable machine-machine interaction over Internet.

Figure 2.3 depicts the main roles and operations in a service-oriented architecture which contains three roles: a service requestor, a service provider, and a service broker/registry. The Web Services [15] can be described using a WSDL (Web Services Description Language), published to a registry of services using UDDI protocol, discovered between the Service Requestor and the Service Provider, invoked using a declared API or composed with other Web Services.

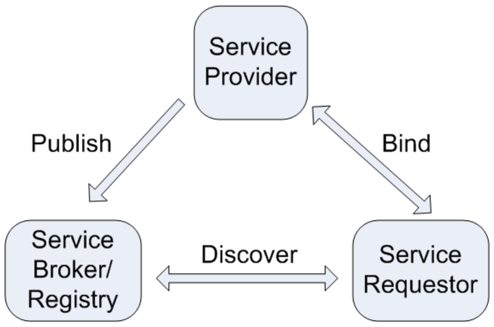


Figure 2.3: Web Services

In this web shop application we focused on the Service Provider and the Service Requestor. Service Requestor can be compared as the client side of a client-server relationship between the Service Requestor and the Service Provider. This relationship is symbolized by the Bind operation.

The web shop performs several roles. It acts as a service provider supplying several Web Services such as giving a list of all the products or products by category. It can also work as a service requestor and gather information from a shipping company which offers a Web Services with its rates depending on the distance between the origin and the destination, and the weight of the order.

Chapter 3

Method

This Chapter presents a Vision of the whole Project and presents the main functionalities of the system prototype at Section 3.5. The Vision of the Project has been written following the Rational Unified Process-specific document templates [16]. The Rational Unified Process (RUP) [17] is an iterative software development process created by the Rational Software Corporation, currently a division of IBM. RUP describes how to develop software effectively using proven techniques. The prototype is base on two main interfaces. The store-back which goal is to manage the webshop by the administrators or the employees, and the store-front which objective is to present the available products in the shop to the final user.

3.1 Positioning

3.1.1 Business Opportunity

This application will allow to the company the possibility of selling its products through Internet, which will suppose a fast and easy way to access to the data thanks to a user-friendly web interface. The application also permits the customers access directly to the company services without any middlemen.

3.1.2 Problem Statement

The problem of	Taking control of the existent stock at several stores, so that they may serve orders that receive those stores. Managing the purchase orders from the customers. Managing the accomplished orders to the suppliers.
affects	Store managers, persons in charge of freight and Invoicing department.
the impact of which is	Gather and manage all the information related to the stores and orders from the customers.
a successful solution would be	Computerizing the process using a database which is accessible from different nodes and generate a friendly and intelligent user interface.

3.1.3 Product Position Statement

For	Middle income homeowner.
Who	Feel the need for buying products in their houses saving time and money because of there are no middlemen.
"Secure, scalable and component-based Webshop using Struts and Hibernate"	Is a software product.
That	Sells products using a user-friendly web interface.
Unlike	The ordinary Webshops in the World Wide Web since they are too simple and have so many mistake in the end-user point of view, scalability, internationalization, security, fault tolerance and components based architecture.
Our product	Gives the possibility to sell products using web interface in internet, besides allows to the employee to manage the Webshop.

3.2 Stakeholder and User Descriptions

It is required to recognize and involve all of the stakeholders as part of the Requirements Modeling process in order to effectively provide products and services that meet our stakeholders' and users' real desires.

3.2.1 Market Demographics

The target market segment includes all kind of citizens who has an Internet connection and is interested in buying articles through Internet.

3.2.2 Stakeholder Summary

Hereinbellow we can find a summary list of all the identified stakeholders.

Name	**Description**	**Responsibilities**
Customer	The person who will search and buy the products.	Keep updated his information in the database.
Employee	This is a stakeholder that will work in the shop.	The person who is in charge of the shop, product stock, managing the orders and reporting possible errors to the Administrator.
Administrator	This is a stakeholder that knows the details about the application and how to solve problems.	The person who is in charge of creating and modifying the configuration for the system and maintaining the database.
Bank	This is the intermediate entity.	This entity ensures that the transaction is done without any problems.

3.2.3 User Summary

Hereinbellow we can find a summary list of all the identified users for the application.

Name	**Description**	**Stakeholder**
Customer	End user of the system	self
Employee	End user of the system	self
Administrator	End user of the system	self

3.2.4 User Environment

The environment of the target user will be a Web interface displayed in a web browser which supports HTTP protocol.

3.2.5 Stakeholder Profiles

The following lists describes each stakeholder in the system.

3.2.5.1 Customer

Representative	
Description	The person who will search and buy the products.
Type	This is an end user.
Responsibilities	Keep updated his information in the database.
Success Criteria	The success is completely defined by buying products.
Involvement	The application will store opinions from the customers in order to improve and fix the application
Comments	Nothing.

3.2.5.2 Employee

Representative	
Description	This is a stakeholder that will work in the shop.
Type	This is the end user.
Responsibilities	The person who is in charge of the shop, product stock, managing the orders and reporting possible errors to the Administrator.
Success Criteria	The success is completely defined by the ability to maintain the shop.
Involvement	There is no employee in this phase.
Comments	Nothing.

3.2.5.3 Administrator

Representative	
Description	This is a stakeholder that knows the details about the application and how to solve problems.
Type	This is an expert user which will configure and fix errors in the application.
Responsibilities	The person who is in charge of creating and modifying the configuration for the system and maintaining the database.
Success Criteria	The success is completely defined by the ability to maintain the application.
Involvement	We will have sample of administrators to improve our vision in this project.
Comments	Nothing.

3.2.5.4 Bank

Representative	
Description	This is the intermediate entity.
Type	This is a casual user, which will check the payment system.
Responsibilities	This entity ensures that the transaction is done without any problems.
Success Criteria	The success is completely defined by the transaction payment.
Involvement	We will have contact with some payment companies.
Comments	Nothing.

3.2.6 User Profiles

See previous Section 3.2.5.

3.2.7 Key Stakeholder or User Needs

The follow lists the main problems with existing solutions as perceived by the stakeholder.

Need	**Priority**	**Concerns**	**Current Solution**	**Proposed Solutions**
Secured access	High	Sales and user information	Roles and logic security in JSP files.	Encryption and SSL.
Easy to use	High	Ability to provide intuitive navigation in the web interface	User friendly and an intuitive navigation in the web interface.	-
Internationalization	Moderate	Ability to provide the user interface in the user's language.	Show the application in the request language.	-

Scalable	Moderate	None	Possibility of ample number of end users to connect to the application at the same time.	-

3.3 Product Overview

This section gives a high level view of the application capabilities.

3.3.1 Product Perspective

Using Struts we can divide the application in several layers. Struts follows the Model-View-Controller (MVC) pattern, sometimes called Model 2, MVC2 or Web MVC.

Using MVC architecture involves defining three classes of modules, the controller, the model and the view of a system. The controller manages the application flow. The model uses hibernate to access to the database and to carry out the business logic encapsulating the application state. The view consists of Java Server Pages (JSP) and custom tags in order to present the result.

3.3.2 Summary of Capabilities

Customer Benefit	**Supporting Features**
Secure sales and access to the system	Encription and authentication
Struts framework supports Internationalization	Flexible access to the system. Due to the web interface the application can be accessed by internet.

3.3.3 Assumptions and Dependencies

Correct version of J2SE 1.5 or higher must be installed and MySQL database must be reachable. Environment variables must be also correct. An application server (such as JBoss or Tomcat) must be installed and running.

3.4 Constraints

3.4.1 Security

Due to transactions are going to be done with real money, security is one of the most important points in this application. The first security feature is authentication, which only allows buying things to customers already logged in. The second one is authorization, after the login process, there are different roles and each of them has different permissions and restrictions, this avoids customers from making errors or faults in the systems.

3.4.2 Responsiveness

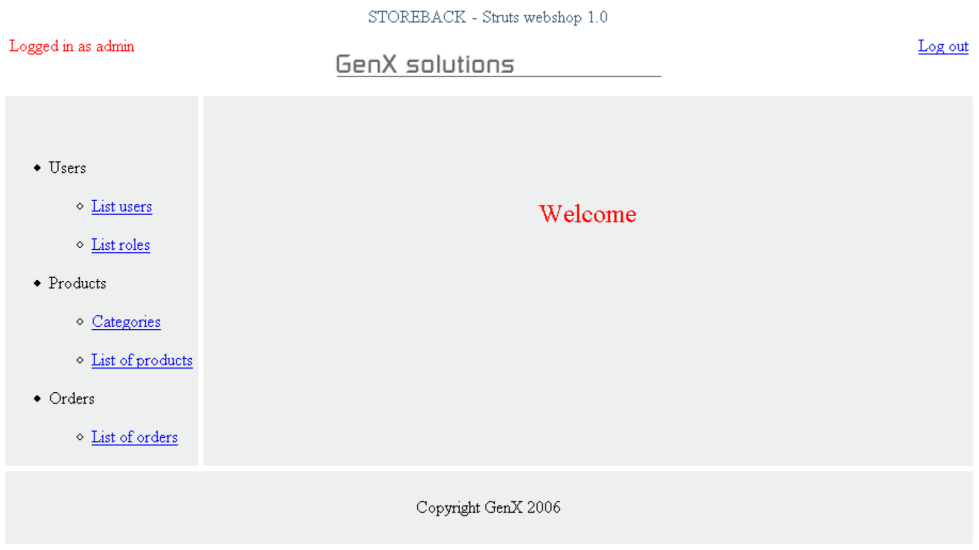
The application responds quickly to the end-user, not only to assure a short response time, the system must be also scalable, therefore this response time does not depend very much on the number of users connected at the same time.

3.5 Overview of the system prototype

The system prototype has two parts, the administrator interface where the application can be administrated, e.g., adding new products to the webshop, modifying the product description in different languages. The other interface is the real webshop published in Internet where the customers can register themselves, search and buy products among other things.

3.5.1 Administrator interface (Store back)

The store-back is the administrator interface of the application where only the administrators and employees can enter to manage the customers, employees, administrators, roles, orders, products and its categories. Below these lines we can see a screenshot of the store-back prototype.


STOREBACK - Struts webshop 1.0

Logged in as admin

GenX solutions

Log out

- Users
  - List users
  - List roles
- Products
  - Categories
  - List of products
- Orders
  - List of orders

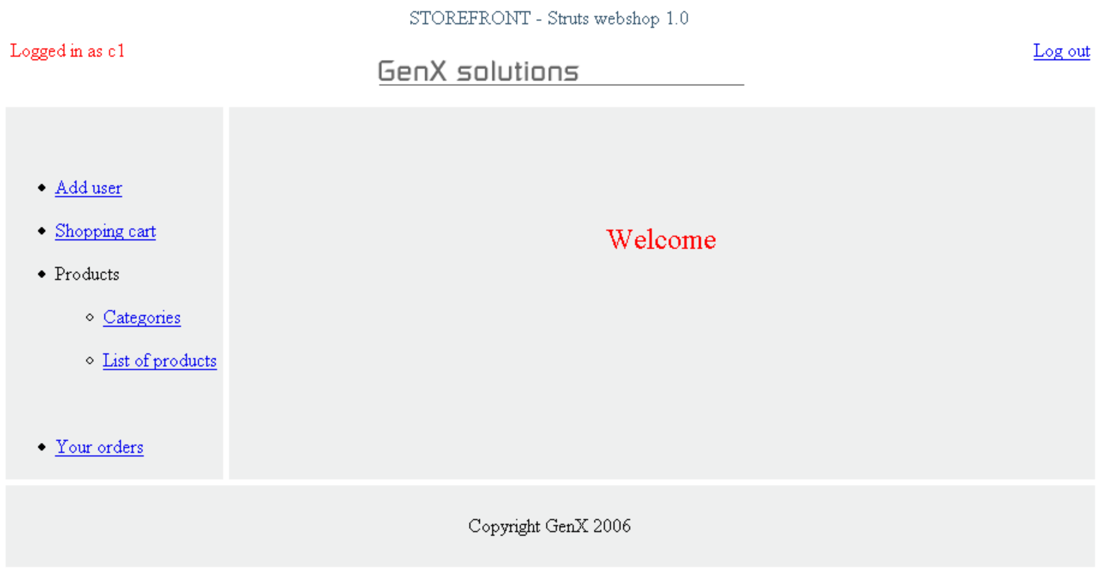
Welcome

Copyright GenX 2006

Figure 3.1: Store-back screenshot

3.5.2 Customer interface (Store front)

The store-front is the public part of the application where the customers can browse products and buy them. In this interface the customers can also manage their shopping cart, search for products by category or list all of them, get registered to themselves, always mandatory before buying, and for logged customers the interface gives the possibility to show the customers' orders. Under these lines we can find a screenshot of the store-front prototype.



STOREFRONT - Struts webshop 1.0

Logged in as c1

GenX solutions

Log out

- Add user
- Shopping cart
- Products
  - Categories
  - List of products
- Your orders

Welcome

Copyright GenX 2006

Figure 3.2: Store-front screenshot

Chapter 4

System design

It is well known the advantages of using MVC pattern, i.e. the roles of programmers and designers are well defined, and this means the designers can concentrate more on the presentation layer as well as the developers in the business logic of the applications. We chose a solid MVC-based framework for developing the application, Struts.

Struts framework [3] integrates with other technologies to provide the Model, in our case with Hibernate, and the View components (JSP files).

Hibernate maps from database tables to Java classes, and also provides data query in an easy way similar to SQL language, Hibernate Query Language (HQL). It offers retrieval facilities instead of using manual data handling in SQL and JDBC.

To be able to be compatible with old applications or simply increase the functionalities, we can put everything together and integrates it with Web Services technology.

In this chapter we are going to review every concept concerning the architecture of the application [18] and finally put together all the concepts with the purpose of presenting the architecture of our Webshop at Section 4.5.

4.1 The MVC pattern

The goal for the MVC software architecture is to minimal the impact to the other components if a change in one of them is made. With this design our project is easy to maintain and good to sell to different shops, since the big changes are only in the View module. Every module is explained in Section 2.4.1.

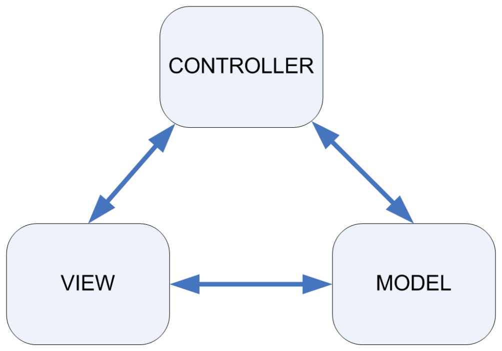


Figure 4.1: Simple Model-View-Controller (MVC) pattern.

4.2 Struts Framework

The Figure 4.2 shows in a high level the Struts Framework. As we can see Struts architecture is based on the Model-View-Controller pattern.

Struts coexists with other technologies in order to integrate the Model and View components but it has its own Controller element, the `Action` Classes, also called `ActionController` in previous versions. In our application the Model technology is Hibernate and for the View, Struts uses the JSP environment.

One of the features from Struts Framework is the internationalization, also called "i18n". An example of internationalization component is the Message Class which is administrated by the Controller module and references a resource bundle strings stored in the Resorce Properities File.

4.3 Hibernate technology

We can also described Hibernate as an Object-relational mapping (ORM) tool for Java environments. It is free, open source software that is distributed under the LGPL (GNU Lesser General Public License). It furnishes an easy to utilize framework for mapping an object-oriented domain model to a tradi-

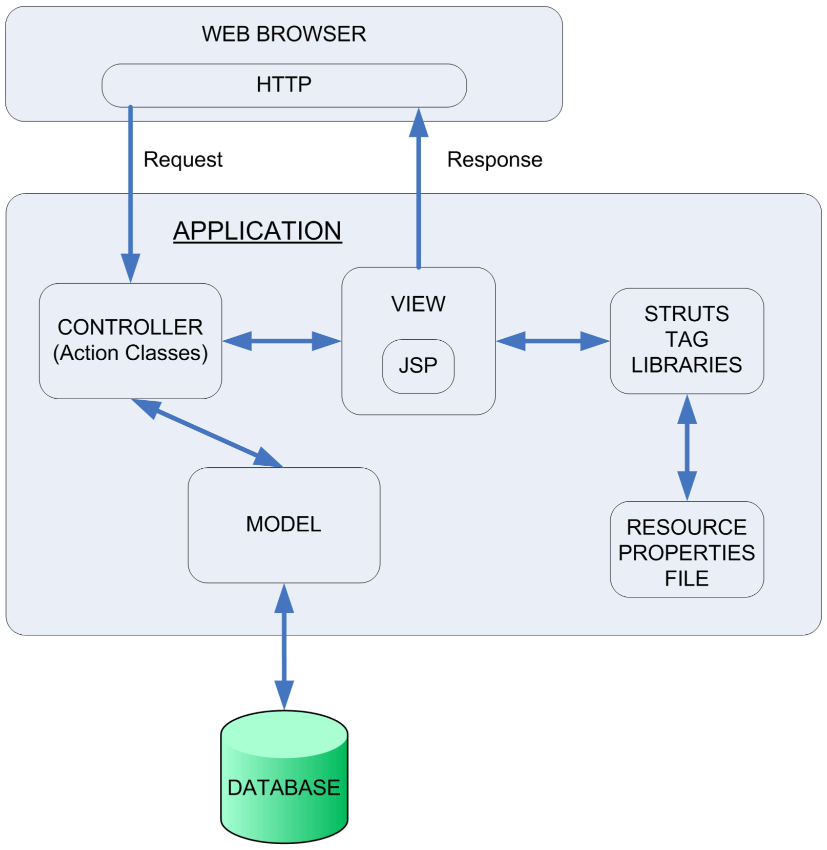


Figure 4.2: Struts Framework architecture.

tional relational database. Hibernate objective is to reduce 95% of common data persistence related programming tasks.

Hereinbellow there are some definitions of the Figure 4.3 [13]:

SessionFactory It is an immutable thread cache of compiled mappings for a specific database.

Session A short-lived single threaded object representing the relationship between the application and the persistent objects. It envelops the JDBC connection.

Persistent Objects A short-lived single threaded objects storing persistent situation and business purpose associated with a specific Session. Those objects will be released in order to use them in other layer when the Session is closed.

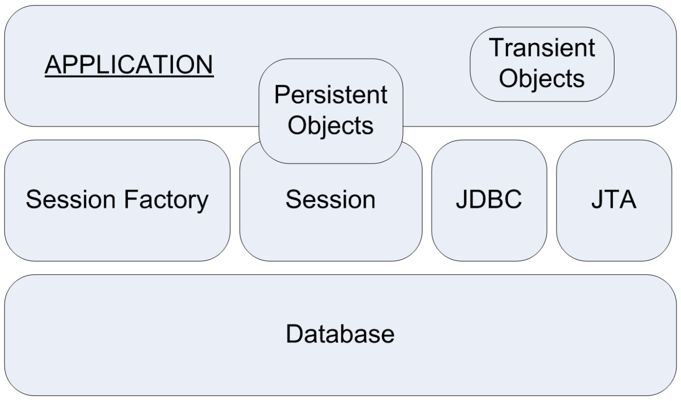


Figure 4.3: A medium level Hibernate architecture.

Transient Objects Persistent Classes which are not presently connected with a Session.

4.4 High level Web Services architecture

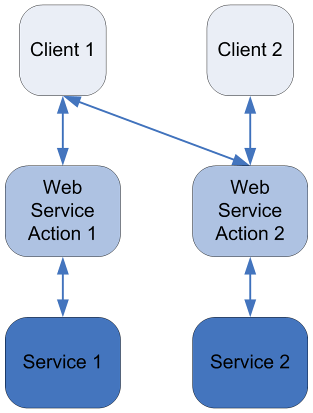


Figure 4.4: High level Web Services architecture.

As shown in Figure 4.4, associating a single function with a single service is the easiest way to develop a Web Service. These services can either retrieve data or achieve business logic. But this has several drawbacks such as there is no defined point for the user or the relation between the provider-subscriber is not well designed. The final design using MVC pattern (Struts) will improve this architecture as showed in the next section.

4.5 Struts, Hibernate and Web Services: Our Webshop

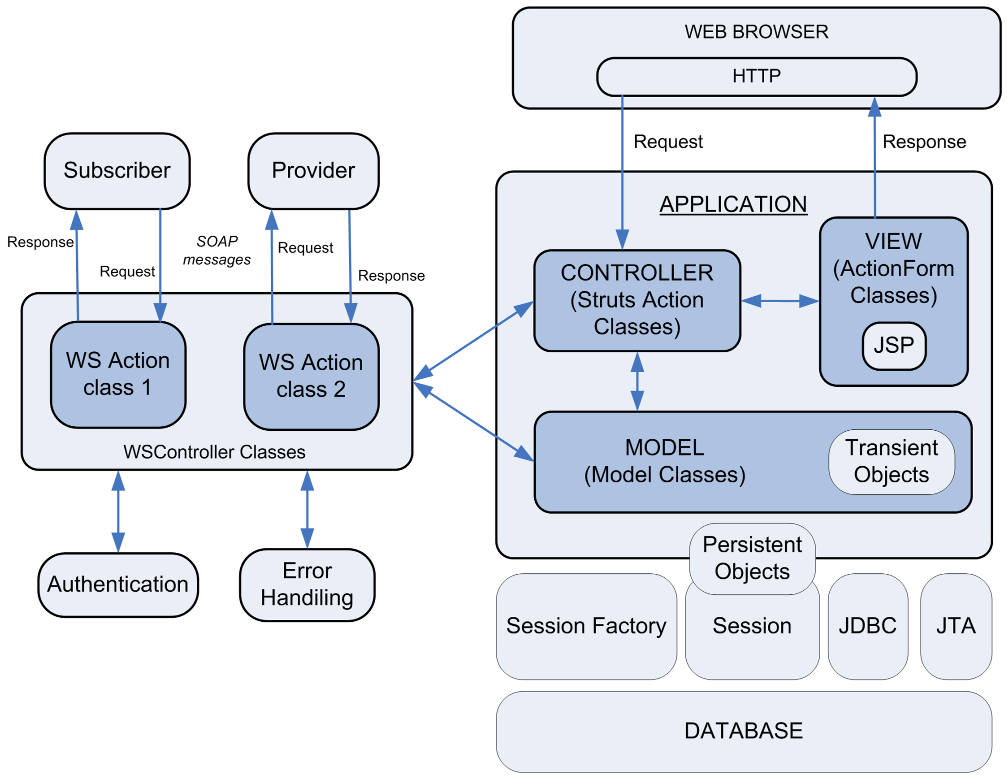


Figure 4.5: Struts, Hibernate and Web Services.

As a consequence of using the MVC pattern, Struts framework visibly differentiates the View, the Controller and the Model. Putting together Struts with Web Services allow to the Struts application either provide or subscribe to a Web service.

To be capable of understanding all the technologies working together, the next sections illustrate the most important components of the whole webshop design showed in Figure 4.5:

4.5.1 Controller

This is the backbone of the Struts applications. The primary component is the class `ActionServlet`. The Controller takes care of collecting requests from the client trough a Web browser, and decides what business logic to execute depending on the request. After that it delegates control of each request to a specific `Action` class, based on the URI of the incoming request. In the Action class, the Model of the application is gathered or modified, the returns a key to the `ActionServlet` in order to decide what View component will be presented. The WSController are called by the Struts Controller to bring into play a Web Service. The WSController obtains the required response and sends it back to the Struts Controller. Notice that the WSController can also receive a request from a Subscriber and communicate with the Struts Controller to achieve a particular function. We explain more details about the Controller implementation in Section 5.2.

4.5.2 View

Each View component in the Struts Framework is based on JSP technology which can contain any combination of static HTML and dynamic content that changes depending on the interpretation of special action tags. These tags are JSP tags or Struts custom tags. We will focus more on the Struts framework which includes an extensive custom tag library well prepared for internationalization. The JSP technology carries out two main roles in Struts application. The first one is to work as a presentation layer for the Controller component using the Struts custom tags mentioned before. The second one is to collect information from the user in order to execute a Controller Action.

4.5.3 Model

The Model components represent the data objects of the application as we explained earlier. It is based on a set of Java classes mapping the database into objects. Both the Struts Controller and WSController can call the Model components methods in order to get data in the form of Data Access Objects. These components can achieve any required business logic and then retrieve any data from the Persistent Objects. The Model components populate the Data Access Object, and pass it back to the Struts Controller or WSController classes. If an error appears, it is propagated back to the Struts Controller or WSController layer.

4.5.4 WSController

The WSController takes care of the incoming SOAP requests and passes on the business logic of the Model. Therefore the WSController handles requests and responses and acts as a contact point. It performs security services such as authentication or authorization. It can also cache the data to evade any unnecessary query.

4.5.5 Provider

Both the Provider and the Subscriber Web Services must have access to the same service description. Every WSController method is published through a WSDL file (Web Services Description Language) based on XML schemas. This schema can either be published on a public registry or in the same server if all of their clients are dedicated partners.

4.5.6 Subscriber

The application makes use of a Web Service for getting the shipping rates but it can subscribe to any other Web Service either in the public registry or in an enterprise. The WSController has methods for parse the WSDL definition file and call the procedure. These methods are called by the Struts Controller in order to execute its own functionalities. Subscribing to an external Web Services requires a WSDL document, a port number, a server name, a method name, and the necessary parameters.

4.5.7 Authentication and authorization

This is an important feature that we will improve its security level in feature work since it is necessary to implement a digital certificate for the subscriber in the WSController. We have implemented a basic user authentication encrypted only in the database.

4.5.8 Error Handling

The error handling is managed by the WSController. The WSController can throw a exception as a `SOAPFaultException` if it is working as a provider. I.e. the incoming request throws an exception if any mandatory field is missing.

In the other hand if the Web Service is working as a subscriber, the WSController gathers a SOAP exception thrown by the service provider and transform it into the format needed by the WSController.

Chapter 5

Implementation

A prototype has been built to achieve the requirements specified in Section 1.3. This Chapter is divided in several sections explaining implementation details concerning the architecture shown in Figure 4.5.

5.1 Development platform

The operating system was Windows XP SP2 and the prototype code was developed in Eclipse SDK Version 3.2 [19] using Java EE Version 5 [20], the plug-in MyEclipse Enterprise Workbench Version 5.0 [21] and the extension Eclipse Web Tools Platform (WTP) Version 1.5 [22]. The election for the web server was Apache Tomcat Version 5.5.16 [23] and for the Database Management System was MySQL Version 5.0 [12].

Notice that all above projects are free software but MyEclipse, which has a trial period of 30 days and thinking about future work its registration is very economical.

5.2 Struts

The main component of the Controller part is a Servlet (class `ActionServlet`), which is in charge of mapping URI requests to specific actions through an XML file called `Struts-config.xml`. This is the backbone of controlling the entire logical flow of the application. It is easy to read, particularly if the application is large. It holds a list of request URIs and notifies the ActionServlet how it must dispatch them.

In the View component Struts uses the JSP environment which enables to use any combination of static HTML and dynamic content that changes based on the interpretation of special action tags. The JSP Custom tags have been quickly increasing since their first introduction in the version 1.1 of JSP specification. Struts includes a set of Tag libraries that allow us too develop the View component without inserting Java code into our JSP pages. These Struts libraries are: Bean Tags, HTML Tags, Logic Tags, Nested Tags, Template Tags and Tiles Tags.

One of the main rules for using Struts is that the Model do not have to contain any View code as well as all JSP files do not have to contain any forward code since the flow of the application is controlled by the `Struts-config.xml` file.

Hereinbellow there is a snippet of the Struts-config.xml file:

```
...
<action  path="/LoginEmployee"
         type="com.genx.security.action.LoginEmployeeAction"
         name="loginForm"
         scope="request"
         input="storeback.adminlogin"
         validate="true">
        <forward name="success"
                 path="storeback.adminwelcome" />
        <forward name="error"
                 path="storeback.adminlogin" />
</action>
...
```

This entry contains data that will be stored in the `ActionMapping` which is an argument of the executed() method of the `LoginEmployeeAction`. This action should be called when the URL ends with the path `"/LoginEmployee"`. The original resource that submitted the request to this Action is a JSP page with a Tile definition of `"storeback.adminlogin"`. Depending on the returned value from `the ActionForward`, the `LoginEmployeeAction` class will forward the flow and the results of its processing to `"storeback.adminwelcome"` Tile definition or get back to `"storeback.adminlogin"` Tile definition.

Concerning the Tile definitions see Section 5.2.2 and for the ``validate'' and ''name'' field take a look the bellow Section 5.2.1.

5.2.1 Errors

Struts contains a framework used to validate form fields. In the last example the `Form` that was going to be validated was `"loginForm"` and is validated depending on whether the field “`validate`” is equal to true or not. The Validate framework has support for internationalization and is extensible. Using this framework we can validate data such as e-mails, credit cards, dates or write our own rules in Java.

5.2.2 Tiles

The Tiles framework is bundled with Struts [3] framework but not enabled by default. This framework is really useful as it allows us to create reusable presentation components, i.e. layouts, HTML, images and other visual components. Using custom tags, `<tiles:   />`, and Java scriptlets we can develop these visual elements.

The main layout in our application is divided into four regions (Header, Menu, Body and Footer) which allow us to pass parameters such as the user name logged into the header region. The complete structure of the presentation is collected in a XML file, the Tile definition. Here I enclosed a snippet of the `tiles-defs.xml` file:

```
<tiles-definitions>
<definition name="siteLayoutDef"
            path="/pages/storeback/common/siteLayout.jsp">


    <put name="title" value="GenX - Struts webshop 1.0" />
    <put name="header" value="/pages/storeback/common/header.jsp" />
    <put name="footer" value="/pages/storeback/common/footer.jsp" />
    <put name="menu" value="/pages/storeback/common/menu.jsp" />
```

```
        <put name="content" value=""/>
        <put name="logo" value="images/genx_logo.gif" />
    </definition>
    <definition name="storeback.adminlogin" extends="siteLayoutDef">
        <put name="content" value="/pages/storeback/adminlogin.jsp"/>
    </definition>
    ...
    </tiles-definitions>
```

When we install the Tiles Plug-in, we are installing a custom request processor, too. This means that in the `Struts-config.xml` file we can forward to a Tile definition:

```
    <forward name="error" path="storeback.adminlogin" />
```

Instead of calling the JSP file `‘‘/pages/storeback/adminlogin.jsp’’`.

5.2.3 Security

The problem of security is a crucial aspect in this project since we are going to sell real products. In order to introduce security in our Struts application and after studying carefully the chapter 19: "Securing struts application" in [24], we have to focus on the following topics:

Authentication and authorization. Authentication is the procedure of telling to the application that you are who you are saying. This is implemented by introducing the user name and password. Once you have introduced this information the Authorization allows you to access to different webpages. This is carried out by the Role-based access control (RBAC) and the Application-managed security.

Role-based access control. The Role-based access control (RBAC) is an alternative approach to Mandatory Access Control (MAC) and Discretionary Access Control (DAC). The permission of execute an Action can be allowed or disallowed based on the user role defined in each entry in the Struts-config.xml file.

Application-managed security. By extending RequestProcessor we can use role-based access for authorization. Once the user is authenticated, we store the user in the Session until he/she logs-out or does not call any Action through surfing in the webshop in a period of time.

In addition, we use the `<logic:   >` tag to hide or show some information relevant to security issue in the JSP files.

```
<logic:equal name="user" property="administrator" value="true">
  //security code
</logic:equal>
```

Cookies. We use Cookies to keep safe all the information relevant to the last user Session so that if the user close the web browser or the connection is cut off, next time the user logs into the application, he will be advised if he wants to keep the last session. I.e. the user can return to a precise step in his/her shopping without filling all the information again. These cookies have an expiration period for security reasons.

Secure Communications using SSL. We understand that this is a very important security issue, but since https protocol of the URL must often be hard-coded into a page, we will leave this part for future work.

5.2.4 Internationalization

Every Web browser attaches some information in the requests about the country and the language, i.e. `en_US, es_ES, sv_ SE`. It is necessary for each language a Resorce Properities File containing a list of key-value pair. Each value will be displayed in the JSP page depending on the request.

The two foremost i18n components are the `Message` Class which is administrated by the Controller module and references to a resource bundle strings stored in the Resorce Properities File as we mentioned before. And the second one is a Struts custom Tag, `<bean:message/>`, which is managed by the Controller part in order to display the actual Strings in the View module.

5.3 Database

The Figure 5.1 describes a database design developed for the prototype. This database conforms to an E-R diagram and it can be represented by a collection of tables. Entity types are related to each other using (x,1)-(x,N) mapping, also known as `one-to-many` relationship. A `many-to-many` relationship (using (x,N)-(x,M) mapping) is represented as a table with columns for the primary keys of the two participating entity, and any attributes of the relationship. In our design we can find two `many-to-many` relationships, “`is accessory`” and “`includes product`”.

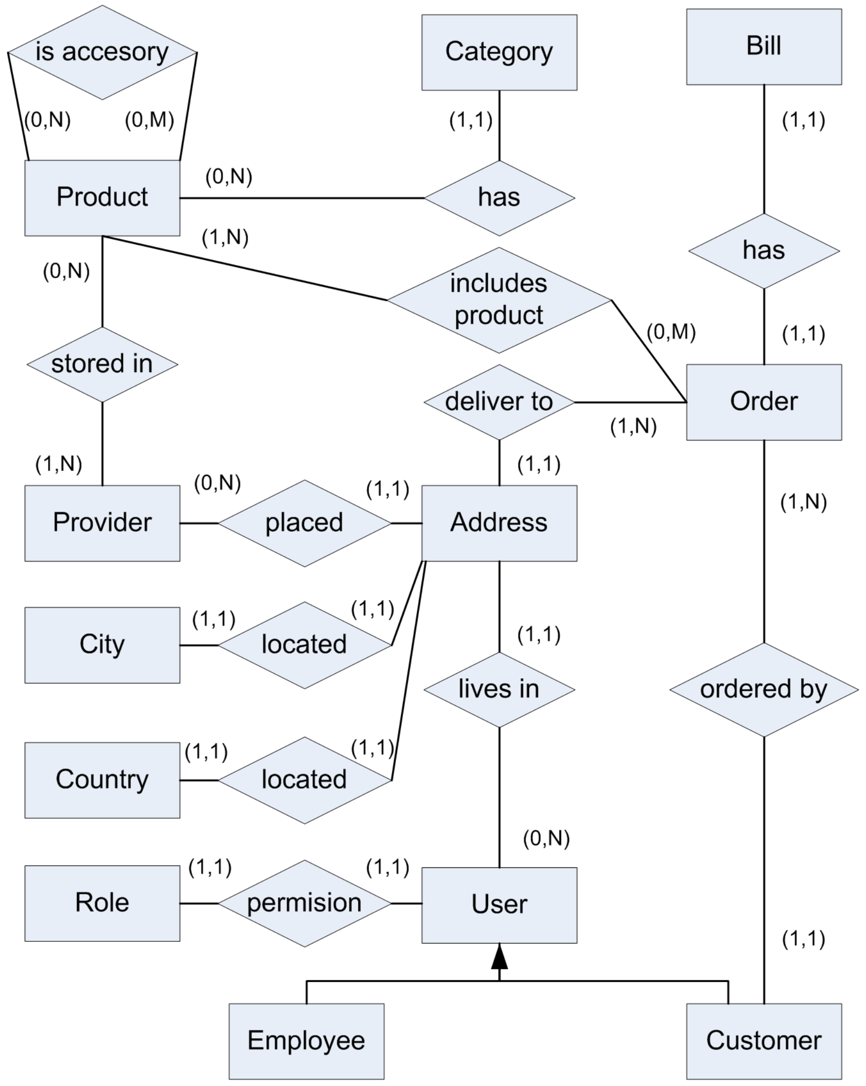


Figure 5.1: Database design

The database was implemented for MySQL. Hereinbellow we can see a snippet from the script:

```
DROP DATABASE IF EXISTS webshop; CREATE DATABASE webshop;
...
CREATE TABLE  webshop.customer (
username varchar(45) NOT NULL,
password varchar(45) NOT NULL,
role_id int(10) unsigned NOT NULL,
name varchar(45) NOT NULL,
phone varchar(45) NOT NULL,
email varchar(45) NOT NULL,
address_id bigint(20) unsigned NOT NULL,
PRIMARY KEY  (username),
KEY FK_customer_1 (address_id),
CONSTRAINT FK_customer_1 FOREIGN KEY (address_id)
REFERENCES address (id)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
...
```

This code regenerates the database deleting the old one and creating a new one. We have to create every table from the design showed in Figure 5.1. As an example it is shown how to create in SQL language the `webshop.customer` table with all its fields. The primary key is the `username` which is used to identify uniquely each record in the table. It has a foreign key, `address_id`, which matches the primary key column of another table. For security reasons, the password is stored by the result of the MD5() function in order to be encrypted, but we store a simple varchar(45).

5.4 Hibernate

Hibernate uses three basic files to provide its services[25]. The configuration file, the mapping definition file and the primary Hibernate class used to retrieve and persist java classes. Let's have a look at these files.

hibernate.cfg.xml Hibernate provides an XML formatted file for configuring the Hibernate service. First, it uses Hibernate-managed JDBC connection and then, it shows the location of the mapping definition files.

```
<hibernate-configuration>
<session-factory>
  <property name="myeclipse.connection.profile">Webshop</property>
  <property name="connection.url">  jdbc:mysql://localhost/webshop </property>
  <property name="connection.username">root</property>
  <property name="connection.password">root</property>
  <property name="connection.driver_class"> com.mysql.jdbc.Driver </property>
  <property name="dialect"> org.hibernate.dialect.MySQLDialect  </property>
...
  <mapping resource="com/genx/dbObjects/Address.hbm.xml" />
  <mapping resource="com/genx/dbObjects/Customer.hbm.xml" />
  <mapping resource="com/genx/dbObjects/Order.hbm.xml"></mapping>
...
</session-factory>
</hibernate-configuration>
```

Customer.hbm.xml This mapping document is used to provide Hibernate with information to persist objects to a relational database. The `id` element illustrates the primary key for the persistent class in addition to how the key value is generated. The `many-to-one` element defines the association to the `Address` class.

```
<class name="Customer" table="customer">
    <id name="username" column="username" type="java.lang.String">

     <generator class="native"/>
     </id>
     <property name="password" column="password"
          type="java.lang.String"  not-null="true" />
     <property name="roleId" column="role_id"
          type="java.lang.Integer"  not-null="true" />
     <property name="name" column="name"
          type="java.lang.String"  not-null="true" />
     <property name="phone" column="phone"
          type="java.lang.String"  not-null="true" />
     <property name="email" column="email"
          type="java.lang.String"  not-null="true" />
     <many-to-one name="address" column="address_id"
         class="Address"  not-null="true" />

</class>
```

AbstractCustomer.java Here we create a `Customer` class with a `many-to-one` relationship to `Address`. Note that `Customer` object has a location field, which links it to an `Address` object as well as a `Set` of orders who has ordered this `Customer`.

```
public abstract class AbstractCustomer
implements Serializable{
...
    /** The composite primary key value. */
    private java.lang.String username;
    /** The value of the address association. */
    private Address address;
    /** The value of the order2Set one-to-many association. */
    private java.util.Set order2Set;
    /** The value of the simple password property. */
    private java.lang.String password;
    /** The value of the simple roleId property. */
    private java.lang.Integer roleId;
    /** The value of the simple name property. */
    private java.lang.String name;
    /** The value of the simple phone property. */
    private java.lang.String phone;
    /** The value of the simple email property. */
    private java.lang.String email;
```

5.5 Web services

Using Eclipse Web Tools Platform (WTP) Project [22], we can create a simple Web Service from the Java class. In our case we wanted to publish a Web Service which offers the following procedures:

`getProducts()`: Provides the functionality of getting a list of all the products.

`getProductsByCategory()`: Provides the functionality of getting a list of products by a specific category.

`deliveredOrder()`: Set "delivered" in the status field. I.e. it is used by the shipping company.

`orderProducts()`: Create an order for a customer with a set of products.

We achieved this publishing a WSDL definition file created by Apache Axis [26].

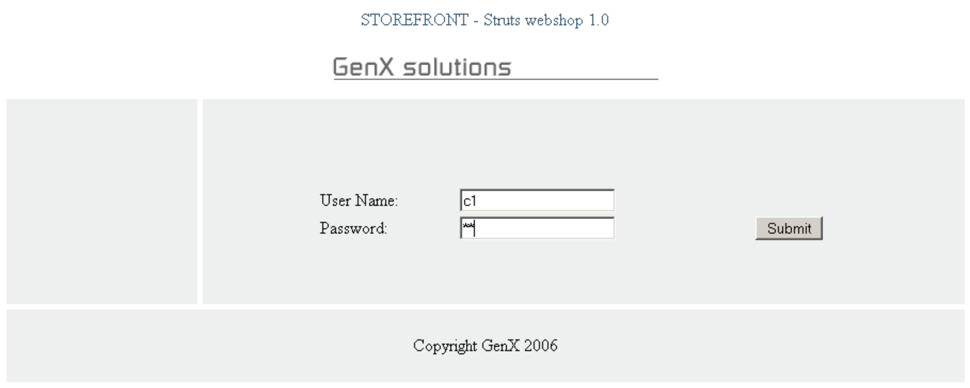
The following code shows a snippet from the `SuplyChain.wsdl`:

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="http://ws.genx.com"
xmlns:apachesoap="http://xml.apache.org/xml-soap"
xmlns:impl="http://ws.genx.com" xmlns:intf="http://ws.genx.com"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<!--WSDL created by Apache Axis version: 1.2.1
Built on Jun 14, 2005 (09:15:57 EDT)-->
<wsdl:types>
<schema elementFormDefault="qualified" targetNamespace="http://ws.genx.com"
xmlns="http://www.w3.org/2001/XMLSchema">
<element name="getProducts">
<complexType/>
</element>
<element name="getProductsResponse">
<complexType>
<sequence>
<element maxOccurs="unbounded" name="getProductsReturn" type="xsd:string"/>
</sequence>
</complexType>
</element>
...
```

This code shows that the result of the getProducts() function is a sequence of String, this means that each product was serialized to a String with all of its fields.

5.6 Flow dynamics

In this section, we will examine the use case log-in by customer detailed in Table B.12 at Section B.5 and discuss each of the steps performed by Struts and Hibernate along the way [11]. The purpose is to show with a walkthrough that ties together all of the earlier assembled components.


STOREFRONT - Struts webshop 1.0

GenX solutions

User Name: c1

Password: **

Copyright GenX 2006

Figure 5.2: Store-front, Log-in screenshot

When we start the application we should see the log-in page in the store-front interface according to the Figure 5.2. This will be changed in future work to log-in only when you are going to buy an order.

When this page loads, the following actions take place:

The `<html:form>` creates the necessary HTML used to represent a form and then checks for an instance of the `com.genx.security.action.LoginCustomerAction` in session scope.

If an instance of the `LoginCustomerAction` is found, then the value stored in the ActionForm's username and password data members are mapped to the input element values on the form and the HTML form is written to the response.

When we write into the username and password fields, and then press the Submit button, it causes the browser to call the URL named in the `<html:form/>` tag's action property, which in this situation is `LoginCustomerAction.do`. The servlet container receives the request, and it tries to find in the web.xml file a `<servlet-mapping>` with a `<url-pattern>` that ends with .do. It gets the following entry:

```
<!-- Standard Action Servlet Mapping --> <servlet-mapping>
<servlet-name>action</servlet-name>
<url-pattern>*.do</url-pattern>
</servlet-mapping>
```

It notifies the container to send the request to a servlet which has been deployed with the action `<servlet-name>`.

The next `<servlet>` entry with a `<servlet-name>` of action which points to the `ActionServlet` is located by the container. It acts as the Controller for our Struts application:

```
<servlet>
<servlet-name>action</servlet-name>
<servlet-class>
org.apache.struts.action.ActionServlet
</servlet-class>
</servlet>
```

The `ActionServlet` assume the control and retrieve the created `LoginForm`, populates the username and password data members with the values passed on the request, and finally adds the `LookupForm` to the session with a key of `loginForm`.

The `ActionServlet` tries to find an `<ActionMapping>` entry in the `struts-config.xml` file with `LoginCustomer` as a `<path>`. It discovers the next entry:

It finds the following entry:

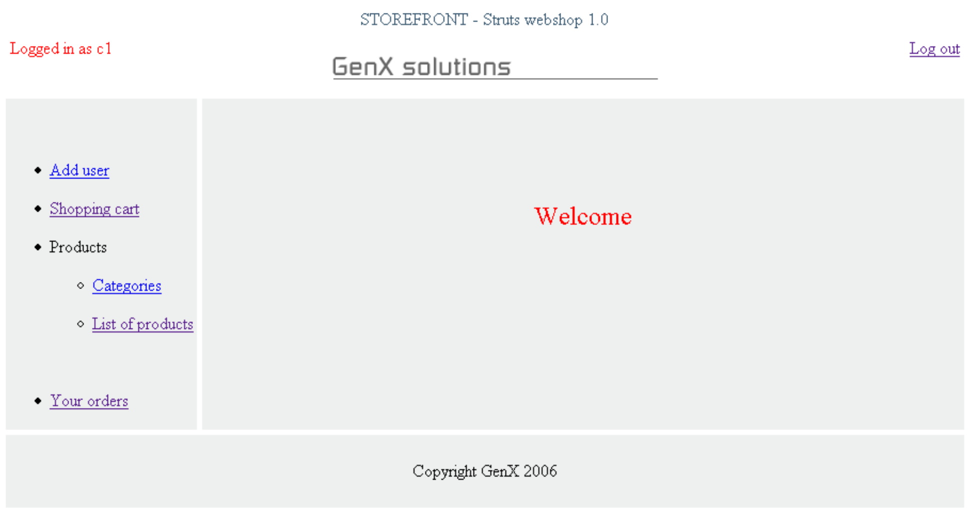
```
<action
path="/sf/LoginCustomer"
type="com.genx.security.action.LoginCustomerAction"
name="loginForm"
scope="request"
input="storefront.customerlogin"
validate="true">


     <forward name="success" path="storefront.customerwelcome" />
     <forward name="error" path="storefront.customerlogin" />

</action>
```

At this point, the `LoginCustomerAction.execute()` method retrieves the `Customer` persistent object by his username. This is carried out by Hibernate calling the Session and Transaction interfaces. It recovers the objects from the database using just the information in the mapping files and making them persistent.

After doing the logic taks using Hibernate, the `LoginCustomerAction.execute()` invokes the `ActionMapping.findForward()` method with a success or error String value.


STOREFRONT - Struts webshop 1.0

Logged in as c1

GenX solutions

Log out

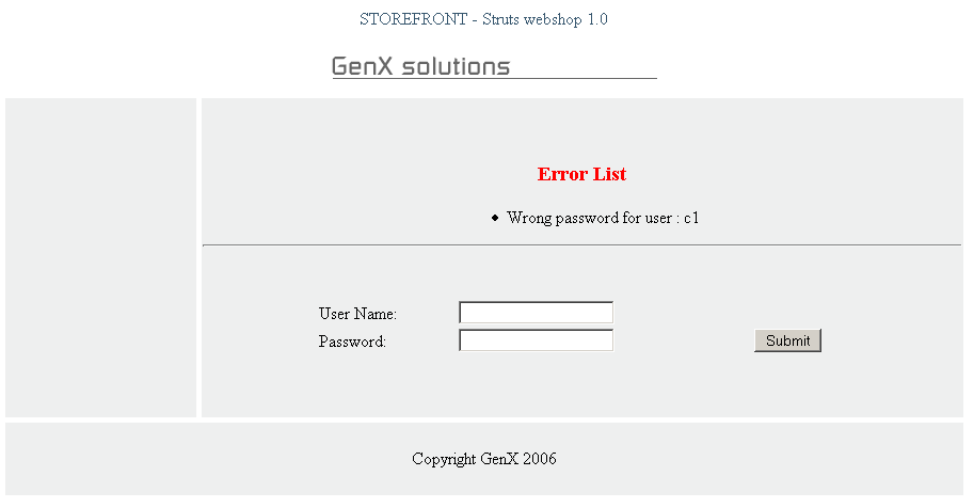
- Add user
- Shopping cart
- Products
  - Categories
  - List of products
- Your orders

Welcome

Copyright GenX 2006

Figure 5.3: Store-front, Welcome screenshot

The `ActionMapping.findForward()` method looks for a `<forward>` element corresponding to the target value. An `ActionForward` object is returned to the `ActionServlet` by the `ActionMapping.findForward()` depending on the target value. It can be either `storefront.customerwelcome` or `storefront.customerlogin` as a path attribute used for the View presentation.


STOREFRONT - Struts webshop 1.0

GenX solutions

Error List

- Wrong password for user : c1

User Name:

Password:

Copyright GenX 2006

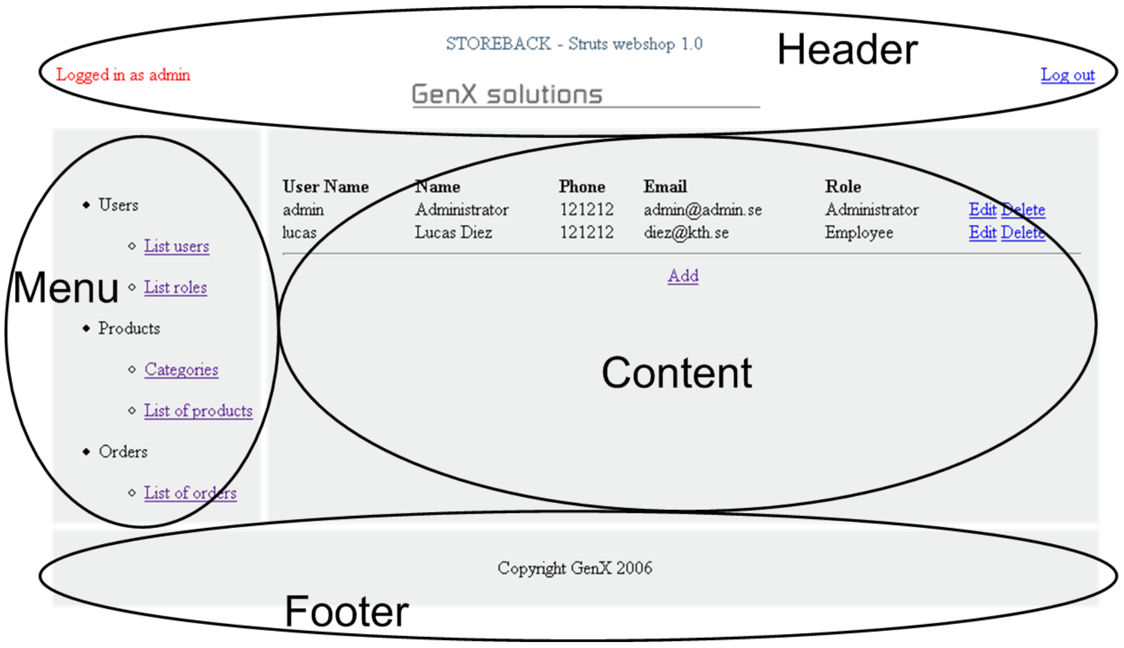
Figure 5.4: Store-front, Error Log-in screenshot

If we insert correctly the username and password, we will be forwarded to `storefront.customerwelcome` showed in Figure 5.3, but if for some reasons the introduced username and password do not match the real ones, we will be sent back to `storefront.customerlogin` illustrated in Figure 5.4, which is the error path of the `LoginCustomerAction` and we will get an error message, an validation technique provided by the Struts application.

5.7 User interface design

This Section describes the user interface designed for the application prototype. It was developed as the View component of the Struts application. This was accomplished using JSP technology that allows us to dynamically generate HTML. The JSP pages were built by assembling reusable Tiles. Essentially Tiles are other JSP pages which allow us to compartmentalize our presentation tier to achieve greater reuse of layouts, HTML, and other visual components.

By using definitions and Tile layouts, we have created reusable display components. Now, we want to show the layout explained in Section 5.2.2 that we have used for designing our user interface.



User Name	Name	Phone	Email	Role	
admin	Administrator	121212	admin@admin.se	Administrator	Edit Delete
lucas	Lucas Diez	121212	diez@kth.se	Employee	Edit Delete

Figure 5.5: Layout screenshot

As the Figure 5.5 illustrates, the layout consists of four regions: Header, Footer, Menu and Content. Observe that we can effortlessly redefine reusable sections of this application just by passing the correct parameters. I.e. al the pages from the store back might share the same header, menu and footer but a different content. It also allow us to pass parameters to the regions like the name of the user logged in the session.

5.8 Deployment strategy

If we wanted to start over the implementation phase, we would begin designing the database. After that, we could write the script in SQL to implement the database. After the creation of the database and all its tables, we could map those tables into the application using Hibernate. This was carried out by Myeclipse tool [21] automatically. At this point, we have the objects that are going to be managed and we will begin programming the Model component to manage those objects.

We can continue with the struts configuration file which describes the whole flow of the application. It may be a good idea to start with a draft design of this file. Then, we have to write the Actions in the Controller component in order to define the way that the UI responds to the user input.

The last step would be to create the View component. This was an easy task thanks to myeclipse tool which provides a friendly user interface for developing JSP files. After that, we have to introduce the struts tags in order to achieve some logic and secure tasks.

Chapter 6

Analysis

6.1 Validation

In this section we explain the number of use cases that have been supported, this means implemented and then tested.

6.1.1 Implemented use cases

The following use cases have been implemented according to the development platform exposed in Section 5.1. The Configuration management (create default, modify), Administrator management (add, edit, remove), Employee management (add, edit, remove), Customer management (add, edit, remove), Roles management (add, edit, remove), Categories management (add, edit, remove), Products management (add, edit, remove), Management orders by employee (make order, view, edit, cancel), Management orders by customer (make order, view). See Section B.5 for further details about the use cases.

The Cancel order by customer use case has been skipped for security reasons. We thought that it is better to cancel an order by the employee and not by the customer without talking first to a member of staff. We did not have time to implement the Search products by employee and Search products by customer use cases.

The Installation of the system, Start up and Shutdown the shop use cases do not require implementation, since the application is package is a `.WAR` file and the only thing to do is execute or shut down the `startup.bat` file in the

Tomcat folder.

6.1.2 Tested use cases

All the implemented use cases have been tested from different machines and through different web browsers. All of them have passed those tests without any significant programming error. This means the following list: The Configuration management (create default, modify), Administrator management (add, edit, remove), Employee management (add, edit, remove), Customer management (add, edit, remove), Roles management (add, edit, remove), Categories management (add, edit, remove), Products management (add, edit, remove), Management orders by employee (make order, view, edit, cancel), Management orders by customer (make order, view).

6.2 Performance

In this Section we test the performance of the system. It is a crucial performance factor to give a fast response to the end-users in order to gain their trust in the reliability of our application.

6.2.1 Test-bed

The computer running the tests is a Toshiba M30-801 laptop with Intel Centrino 1,5 processor, 512MB RAM with Microsoft Windows XP SP 2 operating system.

The programs used for the tests are Borland Optimizeit Enterprise Suite 2006 for analyzing the requests and Mercury LoadRunner 8.0 for generating virtual users.

6.2.2 Worst case

We are going to use the worst use case to calculate the performance of the prototype. The performance measurement is the response times of the requests and the worst use case is the one which has the maximum number of events.

This use case is *"buy product"* (or *"make an order by customer"*) and has the following events:

1. The user logs in
2. List of products
3. Select the product to buy
4. Select the quantity of the product
5. The application calculates and shows the total cost with shipping charge
6. The user confirm the order

This scenario was recordered using LoadRunner in a web browser and it took me 6 seconds to achieve all the events (I tried to do it as soon as possible). Notice that the payment management is not implemented yet.

6.2.3 Performance scenario

The first test scenario was to evaluate the *"worst case"* described above. In this scenario 50 virtual users execute the six events explained to buy a product and repeat them 5 times. The results can be found in tables 6.1, 6.2 and Figure 6.1. The test took 159777.099 ms to handle 2243 requests.

Module Name	Module Time
JDBC	2,00 %
Servlet and JSP	24,02 %
WebService	73,99 %

Table 6.1: System Module Percentage Use.

As we can see the Web Service module takes much more time than serving JSP files and communicating to the database. Observe that the database server and the WebService are tested in the same machine, far away from a real situation where we have to add ping times between the end-users and the

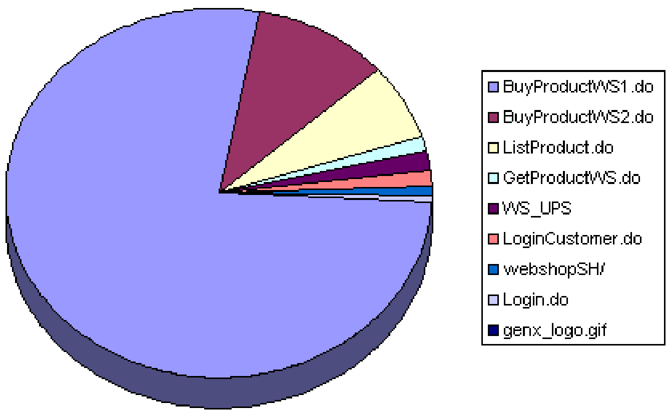


Figure 6.1: System Entrypoint Module Percentage Breakdown

servers. In this supposed scenario the WebServices and the JDBC modules would take the mayor part of the percentage use.

System Entrypoint	**JDBC**	**JSP**	**WSVC**	**Avg.time**
Servicing URI: BuyProductWS1.do (class `ActionServlet`)	0,26 %	3,33 %	**73,63 %**	**495 ms**
Servicing URI: BuyProductWS2.do (class `ActionServlet`)	0,45 %	**9,75 %**	None	66 ms
Servicing URI: ListProduct.do (class `ActionServlet`)	0,76 %	5,84 %	None	42 ms
Servicing URI: GetProductWS.do (class `ActionServlet`)	0,28 %	1,30 %	None	10 ms
Servicing URI: WS_UPS (class `AxisServlet`)	None	1,16 %	0,36 %	10 ms
Servicing URI: LoginCustomer.do (class `ActionServlet`)	0,24 %	1,13 %	None	9 ms
Servicing URI: /webshopSH/ (class `JspServlet`)	None	0,86 %	None	5 ms
Servicing URI: Login.do (class `ActionServlet`)	None	0,63 %	None	4 ms
Servicing URI: genx_logo.gif (class `DefaultServlet`)	None	0,02 %	None	<1 ms

Table 6.2: System Entrypoint Module Percentage Breakdown

The JSP module acts in all the services in opposition to the WebService module which takes action only in two of them. Even taking this into account the WebService module spends almost 75% of the whole time to handle all the request in only one Action, the one which connects to the UPS WebService and gathers the shipping rates for the order.

6.3 Profiling

Using Borland Optimizeit profiler tool, we can isolate memory and CPU performance issues in order to generate reports that help us finding out which methods consume a lot of CPU time and which methods allocate a too much memory.

6.3.1 Profiling scenario

We have chosen the same worst case scenario mentioned before but in this case the number of virtual users is 20 and they are going to buy a product only one time. The results can be found in Tables 6.3, 6.4, 6.5 and Figure 6.2.

Hotspots	**Time %**	**Time**
`org.apache.struts.action.ActionServlet.process`	32.71	721 ms
`org.apache.axis.client.Call.invoke`	29.99	661 ms
`Java core classes`	7.76	171 ms
`org.apache.jasper.servlet.JspServlet.service`	2.72	60 ms
`XML classes`	2.72	60 ms
`org.apache.axis.i18n.MessageBundle (2).getMessage`	2.27	50 ms
`org.apache.struts.util.MessageResources.messageKey`	1.36	30 ms
`org.apache.tomcat.util.log.SystemLogHandler.write`	0.91	20 ms
`org.apache.catalina.core.ApplicationContext.getRealPath`	0.91	20 ms
`org.apache.axis.encoding.DeserializationContext (2).startPrefixMapping`	0.91	20 ms
`org.apache.axis.i18n.MessageBundle (2).getMessage`	0.91	20 ms
`org.apache.axis.i18n.ProjectResourceBundle (2).handleGetObject`	0.91	20 ms
`org.apache.struts.taglib.tiles.InsertTag$InsertHandler.doEndTag`	0.50	11 ms
`org.apache.axis.message.MessageElement (2).MessageElement (2)`	0.45	10 ms
`org.apache.catalina.session.ManagerBase.createSession`	0.45	10 ms
...		
`com.genx.dbUtils.ProductDataHiber.getProductsHiber`	**0.24**	**10 ms**

Table 6.3: CPU profiler output - Sampler / Methods

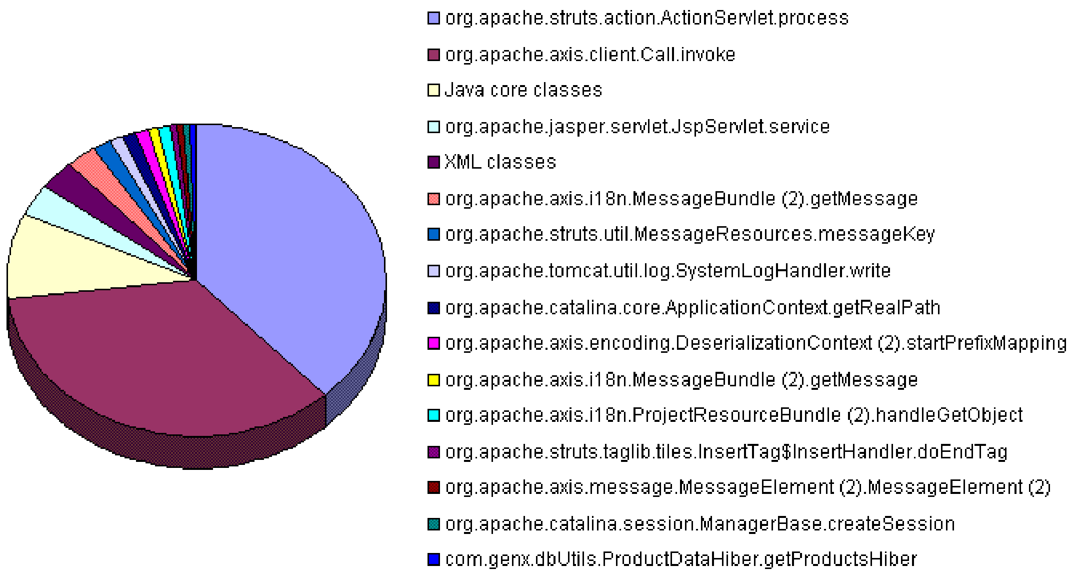


Figure 6.2: CPU profiler output - Sampler / Methods (pie chart)

The first method from the application code does not require almost nothing CPU, this means that there is no a method with important programming errors.

Class name	**Instance count**	**Difference**	**Size**	**Size difference**
`char[]`	**95589**	**+ 52624**	**13150 KB**	**+ 7368 KB**
`java.lang.String`	68842	+ 23090	1613 KB	+ 541 KB
`Object[]`	35995	+ 10652	2363 KB	+ 755 KB
`java.lang.StringBuffer`	9292	+ 9171	145 KB	+ 143 KB
`java.util.HashMap$Entry`	22931	+ 5057	537 KB	+ 118 KB

Table 6.4: Current heap of attached application

After call the Java's garbage collector, we run the scenario mentioned and we can observe that the big difference either in number of instance or size is in ‘‘`char[]`’’, therefore the table 6.5 shows the methods allocating instances of `char[]`.

Allocation locations	**Count**	**Count %**
`Java core classes`	7366	16.07%
`com.mysql.jdbc.ResultSet.getStringInternal()`	6360	13.87%
`XML classes`	4182	9.12%
`org.apache.catalina.util.StringManager.StringManager()`	2155	4.70%
`org.apache.axis.utils.JavaUtils class initialization`	1973	4.30%
`org.apache.axis.i18n.ProjectResourceBundle$Context.loadBundle()`	1789	3.90%
`org.apache.axis.i18n.ProjectResourceBundle$Context (2).loadBundle()`	1788	3.90%
`org.apache.tomcat.util.digester.Digester.endElement()`	1628	3.55%
`org.apache.jasper.compiler.Localizer class initialization`	1367	2.98%
`org.apache.naming.resources.FileDirContext.normalize()`	904	1.97%
`org.apache.catalina.loader.WebappClassLoader.findClassInternal()`	855	1.86%
`org.apache.naming.resources.FileDirContext.file()`	832	1.81%
`com.mysql.jdbc.Messages class initialization`	762	1.66%
`org.apache.tomcat.util.IntrospectionUtils.findMethods()`	582	1.27%
...		
`com.genx.dbObjects.Product$$EnhancerByCGLIB$$ 92b8d4e1$$FastClassByCGLIB$$3231e1d3.getIndex()`	**30**	**0.07%**

Table 6.5: Methods allocating instances of `char[]`

What we can be aware of is that at this workload the first method from the application code requires little memory. No errors managing memory can be found.

Chapter 7

Conclusions and future work

In this report we have present a solution for a webshop following the architecture and patterns of enterprise applications. The main idea was to create a scalable, internationalized, secure, fault tolerance and components based application.

Using Struts framework provides us an easy way to separate the presentation layer and allows it to be abstracted from the data layers and the transactions. Other features are the internationalization, a great custom tag library, tiled displays and form validation among others.

A foremost element in the project is the database. All the information is stored in this component and must be available in an effective way. In the Model layer of the application we use Hibernate. It offers an easy to use framework for mapping an object-oriented domain model to a traditional relational database. Hibernate allows us to query and persist data in a relational database through object manipulation instead of through SQL queries like JDBC.

In order to improve the webshop we have increased the functionalities adding Web Services to the application. This means that not only the web shop can be accessed by a human interface through web pages, the application supports interoperable machine-machine interaction over Internet.

When running tests, we saw that 74% of this time is taken by the Web Services, but this happens only in one Action. The rest Actions requires only serving JSP files and connecting to the database which works very fast, around 100 ms more or less for every use case except the one using Web Services which it takes 650 ms for carrying out the task.

After this master thesis I have learnt very well how the Struts framework works and quite well how we can persist objects using Hibernate technology. The development time were significantly reduced since the MVC pattern let us focus on different components in the implementation phase. An advantage of using Hibernate, having some knowledge before about SQL and database designs, is that it is very easy to learn, and also a lot of common data persistence related to programming tasks could be saved.

Through the development of this application, I have improved a wide range of knowledges, which include J2EE, MVC pattern, more precisely Struts, JSP, Hibernate, Web Services and for writing this report I have learnt LaTeX and used the LyX tool [27], very useful for the presentation and for avoiding errors in this report.

There are two different kinds of future work for this Thesis. The first one is to improve the security by adding SSL encryption and develop the payment section. The second one concerns enhancing the database design with more knowledge about invoicing.

Bibliography

[1] A. B. Bondi, “Characteristics of scalability and their impact on performance,” ser. Proceedings of the 2nd international workshop on Software and performance, Ottawa, Ontario, Canada, 2000, vol. ISBN 1-58113-195-X, pp. 195–203. 4

[2] (2006) The Wikipedia Website. [Online]. Available: http://en.wikipedia.org/ 4, 8

[3] (2006) The Apache Struts Project. [Online]. Available: http://struts.apache.org/ 4, 9, 24, 34

[4] T. N. Husted, C. Dumoulin, G. Franciscus, and D. Winterfeldt, *Struts in Action - Building web applications with the leading Java framework*. Manning Publications, 2002, vol. ISBN 1930110502. 4

[5] (2006) osCommerce, Open Source Online Shop E-Commerce Solutions. [Online]. Available: http://www.oscommerce.com/ 7, 8

[6] (2006) phPay, webshop or catalog based on SQL and PHP. [Online]. Available: http://phpay.sourceforge.net/ 7, 8

[7] (2006) PhPepperShop Webshop-System . [Online]. Available: http://www.phpeppershop.com/ 7, 8

[8] (2006) phpShop. [Online]. Available: http://www.phpshop.org/ 7, 8

[9] N. Werro, H. Stormer, D. Frauchiger, and A. Meier, “eSarine - A Struts-based Webshop for Small and Medium-sized Enterprises,” EMISA Conference - Information Systems in E-Business and E-Government, Luxembourg, October 2004. 7, 8

[10] M. Bowie, “Adaptation of a Webshop for Mobile Devices,” Master’s thesis, University Of Fribourg, Switzerland, October 2005. 8

[11] J. Goodwill and R. Hightower, *Professional Jakarta Struts*. Wrox, 2003, vol. ISBN 0764544373. 9, 10, 41

[12] (2006) MySQL AB. [Online]. Available: http://www.mysql.com/ 11, 32

[13] (2006) Hibernate Reference Documentation Version: 3.1.1. [Online]. Available: http://www.hibernate.org/ 12, 26

[14] H. Oak, “How to decide which persistence technology to use?. JDBC / CMP Entity Beans / Hibernate / JDO ,” 2004. [Online]. Available: http://www.indicthreads.com/blogs/view/101/ 12

[15] S. Graham, D. Davis, S. Simeonov, G. Daniels, P. Brittenham, Y. Nakamura, P. Fremantle, D. König, and C. Zentner, *Building Web Services with Java*, 2nd ed. Sams Publishing, 2005, vol. ISBN 0-672-32641-8. 13

[16] “Rational Unified Process-specific Requirements document template,” 2003. [Online]. Available: http://lehre.ike.uni-stuttgart.de/wn/musoft/rup-manual/webtmpl/req/ 14

[17] “IBM Rational Unified Process, RUP,” 2006. [Online]. Available: http://www-306.ibm.com/software/awdtools/rup/ 14, 62

[18] J. Josephraj, “Architect Struts applications for Web services,” *IBM.com*, April 2003. [Online]. Available: http://www-128.ibm.com/developerworks/webservices/library/ws-arcstruts/ 24

[19] (2006) Eclipse Foundation. [Online]. Available: http://www.eclipse.org/ 32

[20] (2006) Java Platform, Enterprise Edition (Java EE). [Online]. Available: http://java.sun.com/javaee/ 32

[21] (2006) MyEclipse Enterprise Workbench. [Online]. Available: http://www.myeclipseide.com/ 32, 46

[22] (2006) Eclipse Web Tools Platform (WTP) Project. [Online]. Available: http://www.eclipse.org/webtools/ 32, 40

[23] (2006) The Apache Software Foundation, Apache Tomcat. [Online]. Available: http://tomcat.apache.org/ 32

[24] J. Holmes, *Struts: The Complete Reference.* McGraw-Hill/Osborne, 2004, vol. ISBN 0072231319. 35

[25] P. Peak and N. Heudecker, *Hibernate Quickly.* Manning Publications, 2006, vol. ISBN 1932394419. 38

[26] (2006) Apache Axis. [Online]. Available: http://ws.apache.org/axis/ 41

[27] (2006) LyX document processing software, version 1.4.1. [Online]. Available: http://wiki.lyx.org/ 55

Appendix A

Abbreviations

API	Application Programming Interface
CPU	Central Processing Unit
DAC	Discretionary Access Control
DBMS	Database Management System
EJB	Enterprise JavaBeans
GUI	Graphical User Interface
GNU	GNU's Not Unix
GPL	General Public Licence
HQL	Hibernate Query Language
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transfer Protocol
i18n	Internationalization
J2EE	Java 2 Platform, Enterprise Edition
J2SE	Java 2 Platform, Standard Edition
JDBC	Java Database Connectivity

JDK	Java Development Kit
OMR	Object Relational Mapping
MAC	Mandatory Access Control
MVC	Model-View-Controller
PHP	PHP Hypertext Preprocessor
RAM	Random Access Memory
RBAC	Role-based access control
SDK	Software Development Kit
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
SSL	Secure Sockets Layer
UDDI	Universal Description Discovery and Integration
URL	Uniform Resource Locator
URI	Uniform Resource Identifier
XML	eXtensible Markup Language
WAP	Wireless Application Protocol
WSDL	Web Services Description Language
WTP	eclipse Web Tools Platform
WWW	World Wide Web

Appendix B

Use cases

B.1 Project description

The goal for this project is to create a web shop following the architecture and patterns of enterprise applications. We do not intend to create a common web shop since that it is too simple. Today there are thousands of web shops but few of them are easy scalable, the internationalization is hard and the development time is too big. To motivate this project we are going to work on the following properties of the shop: Scalable, Internationalized, Secure, Fault tolerance and Components based.

B.2 List of use cases

Using the Rational Unified Process [17] (RUP)[1] we have written here in bellow the list of use cases.

1. Installation of the system.
2. Start up.
3. Shutdown the shop
4. Configuration management (create default, modify)

[1]The Rational Unified Process (RUP) is an iterative software development process created by the Rational Software Corporation. RUP describes how to develop software effectively using proven techniques.

5. Administrator management (add, edit, remove)
6. Employee management (add, edit, remove)
7. Customer management (add, edit, remove)
8. Roles management (add, edit, remove)
9. Categories management (add, edit, remove)
10. Products management (add, edit, remove)
11. Search products by employee
12. Search products by customer
13. Management orders by employee (make order, view, edit, cancel)
14. Management orders by customer (make order, view, cancel)

B.3 Actors

Each scenario describes a sequence of events. Entities that initiate sequences are called actors and we have found three types.

Administrator The person who is in charge of creating and modifying the configuration for the system and maintaining the database.

Employee The person who is in charge of the shop, the product stock, managing the orders and reporting possible errors to the Administrator.

Customer The person who will search and buy the products.

B.4 Use case Diagrams

The following Figures show the diagrams for the use cases.

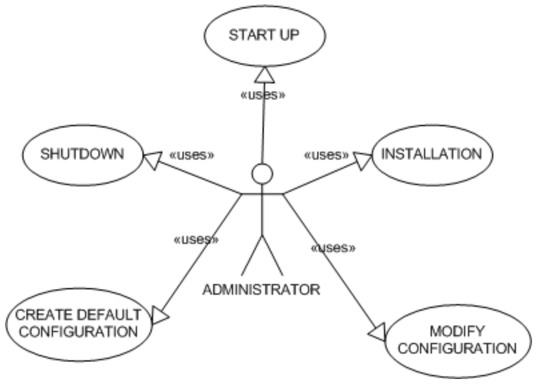


Figure B.1: Application Diagram

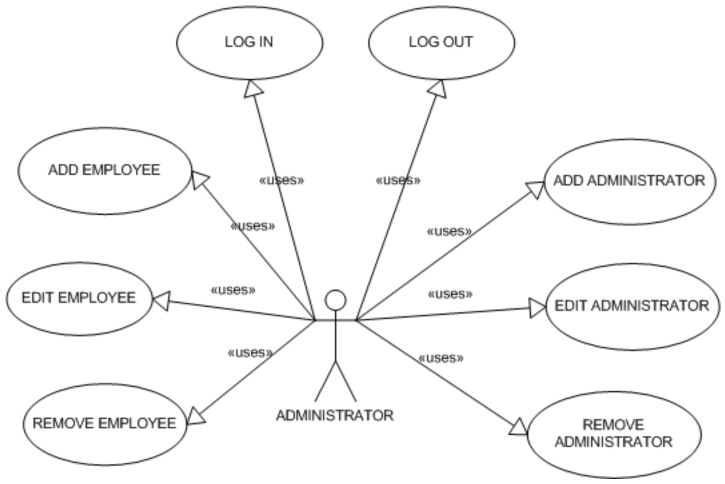


Figure B.2: Users diagram 1

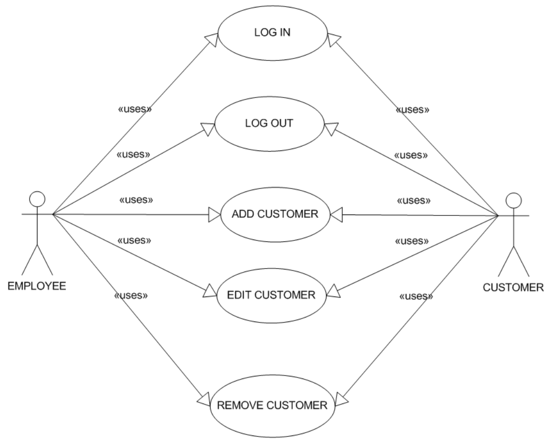


Figure B.3: Users diagram 2

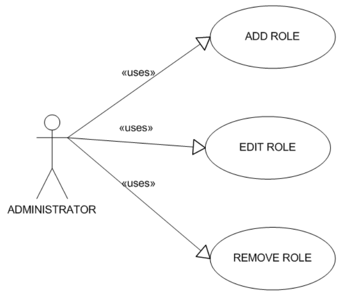


Figure B.4: Role diagram

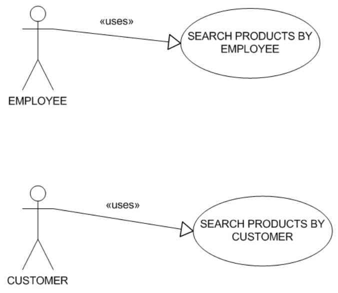


Figure B.5: Search diagram

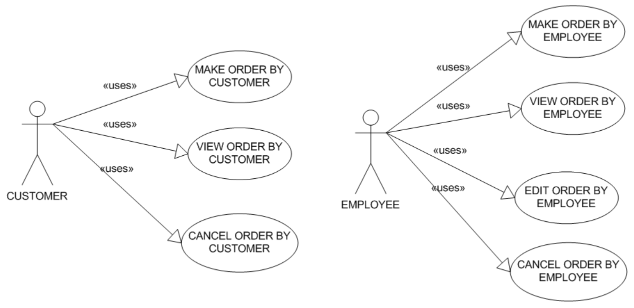


Figure B.6: Order diagram

B.5 Use cases details

The following Tables show the details for each use case.

Brief description	The application is installed and properly connected to the database
Actors	Administrator
Preconditions	Correct version of J2SE 1.5 must be installed and MySQL database must be reachable. Environment variables must be also correct. An application server (such a JBoss or Tomcat) must be installed and running.
Postconditions	The program is ready to be started. The root administrator is created.
Main Success Scenario	The program starts unpackage the directories and moving them inside the application server.The application server detects that a new application has been add.The application server deploys the EAR file.The application asks to input the details of the administrator username and password.The application creates the admin role, and fills the username and password.The Web Shop is ready to be used.
Alternative 1 Flow	There is not any applications server in the computer.The MySQL is not properly configured.
Alternative 2 Flow	There must be enough RAM for the program to execute. Technology Variations List.

Table B.2: Use case Installation of the system.

Preconditions	Correct version of J2SE 1.5 must be installed and MySQL database must be reachable. Environment variables must be also correct. An application server (such a Jboss or Tomcat) must be installed and running. The application is inside the application server and has been deployed previously.
Brief description	The application is installed and properly connected to the database
Actors	Administrator
Postconditions	Program is started, Login page shows.
Main Success Scenario	The program connects to the database.The admin starts the web browser and input the address for the application.The login web page is displayed.
Alternative 1 Flow	There is not any administrator in the database; a window shows up asking for create an administrator.
Alternative 2 Flow	The database is down; an error message shows in the console window.
Special Requirements	There must be enough RAM for the program to execute. Technology Variations List.

Table B.4: Use case Start up

Brief description	The application is shut downed properly, and all the information is going to save in the database
Actors	Administrator
Preconditions	The application is running and the database is up.
Postconditions	The program log out from the database. The application is stopped
Main Success Scenario	The administrator logs in the application. All the information is saved in the database. The program log out from the database. Stop the application
Alternative 1 Flow	A customer is still connected to the shop trying to do an operation; the application save the customer session and send him a message explaining that the application is shut downed

Table B.6: Use case Shutdown the shop

Brief description	This use case creates the first configuration during the installation
Actors	Administrator
Preconditions	The database is up.
Postconditions	A new configuration has been filled and the system changes to the new configuration
Main Success Scenario	The application is configured with the default data for Basic configuration, Welcome page configuration, Mail server configuration, Language configuration, Translation configuration.

Table B.7: Use case Create default configuration

Brief description	This use case modifies the configuration of the application
Actors	Administrator
Preconditions	The application is running and the database is up.
Postconditions	A new configuration has been filled and the system changes to the new configuration
Main Success Scenario	The administrator logs in the application.The administrator configures the application modifying any of the following data: Basic configuration, Welcome page configuration, Mail server configuration, Language configuration, Translation configuration. The administrator logs out the application.

Table B.8: Use case Modify configuration

Brief description	This use case adds to the application a new administrator
Actors	Administrator
Preconditions	The application is running and the database is up.
Postconditions	New administrator is added to the database
Main Success Scenario	The administrator introduces the new administrator name and password.
Alternative 1 Flow	The new administrator name is already added; a window shows up explaining the error and asks for a new administrator name.

Table B.9: Use case Add Administrator

Brief description	This use case edits the information about an administrator
Actors	Administrator
Preconditions	The application is running and the database is up.
Postconditions	New administrator is edited. The database is uploaded.
Main Success Scenario	The administrator introduces the new password for a specific administrator.
Alternative 1 Flow	The administrator name does not exist; a window shows up explaining the error and asks for the administrator name.

Table B.10: Use case Edit Administrator

Brief description	This use case removes to the application an administrator
Actors	Administrator
Preconditions	The application is running and the database is up.
Postconditions	New employee is added.
Main Success Scenario	The specified administrator is selected and removed.
Alternative 1 Flow	The system updates with the carried out changes
Alternative 2 Flow	The administrator name does not exist; a window shows up explaining the error and asks for the administrator name.

Table B.11: Use case Remove Administrator

Brief description	This use case logs in an user.
Actors	Administrator, Employee and Customer.
Preconditions	The application is running and the database is up.
Postconditions	The user is logged in into the session.
Main Success Scenario	The user is logged in into the session and the application gives the possibility of getting back to the last session if the user could not log out.
Alternative 1 Flow	The username and password do not match; a window shows up explaining the error and asks again.

Table B.12: Use case Log in

Brief description	This use case logs out an user.
Actors	Administrator, Employee and Customer.
Preconditions	The application is running.
Postconditions	The user is logged out from the session.
Main Success Scenario	The user is logged out from the session and save the user's information for the next session.

Table B.13: Use case Log out

Brief description	This use case adds to the application a new employee
Actors	Administrator
Preconditions	The application is running, the database is up and there is at least one Administrator
Postconditions	New employee is added to the database
Main Success Scenario	The administrator starts creating the new employee profile including the employee name and password. The profile is updated in the database.
Alternative 1 Flow	The new employee name is already added; a window shows up explaining the error and asks for a new employee name.

Table B.14: Use case Add Employee

Brief description	This use case edits to the application a new employee
Actors	Administrator
Preconditions	The application is running, the database is up and there is at least one Administrator
Postconditions	New employee is edited and the database is updated
Main Success Scenario	The administrator edits the employee profile. The profile is updated in the database.
Alternative 1 Flow	The employee name does not exist; a window shows up explaining the error and asks for the employee name.

Table B.15: Use case Edit Employee

Brief description	This use case removes to the application a new employee
Actors	Administrator
Preconditions	The application is running, the database is up and there is at least one Administrator and employee
Postconditions	New employee is added.
Main Success Scenario	The specified employee is selected and removed. The system updates with the carried out changes.
Alternative 1 Flow	The employee name does not exist; a window shows up explaining the error and asks for the employee name.

Table B.16: Use case Remove Employee

Brief description	This use case adds to the application a new customer
Actors	Employee or customer
Preconditions	The application is running and the database is up.
Postconditions	New customer is added.
Main Success Scenario	The customer or the employee starts filling the new customer profile. The profile is updated in the database.
Alternative 1 Flow	The new customer name is already added; a window shows up explaining the error and asks for a new customer name.

Table B.17: Use case Add Customer

Brief description	This use case edits to the application a new customer
Actors	Employee or customer
Preconditions	The application is running and the database is up. The Administrator is logged in, customer exits.
Postconditions	New customer is edited.
Main Success Scenario	The employee or customer edits the customer profile. The profile is updated in the database.
Alternative 1 Flow	The customer name does not exist; a window shows up explaining the error and asks for the customer name.

Table B.18: Use case Edit Customer

Brief description	This use case removes to the application a new customer
Actors	Employee or customer
Preconditions	The application is running, the database is up and there is at least one customer
Postconditions	New customer is added.
Main Success Scenario	The specified customer is selected and removed. The system updates with the carried out changes.
Alternative 1 Flow	The customer name does not exist; a window shows up explaining the error and asks for the customer name.

Table B.19: Use case Remove Customer

Brief description	This use case adds to the application a new role
Actors	Administrator
Preconditions	The application is running and the database is up. There is no other role with the same name.
Postconditions	A new role is added to the database
Main Success Scenario	The administrator adds a role to the application specifying the privileges for the role.

Table B.20: Use case Add Role

Brief description	This use case edits to the application a specific role
Actors	Administrator
Preconditions	The application is running and the database is up. There is no other role with the same name than the new name.
Postconditions	The administrator edits the role profile. The role is updated in the database.
Main Success Scenario	The administrator edits a role to the application specifying the privileges for the role.

Table B.21: Use case Edit Role

Brief description	This use case removes to the application a specific role
Actors	Administrator
Preconditions	The application is running and the database is up.
Postconditions	The role has been removed from the database.
Main Success Scenario	The administrator removes a role from the application.

Table B.22: Use case Remove Role

Brief description	This use case shows the information about products to the employee. The product can be searched by code or by keywords.
Actors	Employee
Preconditions	The application is running and the database is up.
Postconditions	The result of the search is showed.
Main Success Scenario	The employee searches by code or keywords. The result of the search is showed.
Alternative 1 Flow	There is no result; the application shows this information.

Table B.23: Use case Search products by employee

Brief description	This use case shows the information about products relevant to the customer
Actors	Customer
Preconditions	The application is running and the database is up.
Postconditions	The result of the search is showed.
Main Success Scenario	The customer searches by code or name. The result of the search is showed.
Alternative 1 Flow	There is no result; the application shows this information.

Table B.24: Use case Search products by customer

Brief description	This use case adds to the application a new order
Actors	Employee
Preconditions	The application is running and the database is up.
Postconditions	New order is added to the database.
Main Success Scenario	The employee starts filling the information for the new order. The order is updated in the database.

Table B.25: Use case Make order by employee

Brief description	This use case shows the information about a specific order
Actors	Employee
Preconditions	The application is running and the database is up.
Postconditions	The information about the order is showed
Main Success Scenario	The employee finds the order by reference number or choosing the order list from a customer. The information is showed.
Alternative 1 Flow	There is no order; the application shows this information.

Table B.26: Use case View order by employee

Brief description	This use case edits in the application a specific order
Actors	Employee
Preconditions	The application is running and the database is up.
Postconditions	An order is edited and saved in the database.
Main Success Scenario	The employee edits the order information. The order is updated in the database.
Alternative 1 Flow	The order code does not exist; a window shows up explaining the error and asks for a new one.

Table B.27: Use case Edit order by employee

Brief description	This use case removes from the application an order
Actors	Employee
Preconditions	The application is running and the database is up.
Postconditions	The order is removed from the database.
Main Success Scenario	The specified order is selected and removed. The system updates with the carried out changes.
Alternative 1 Flow	The order does not exist; a window shows up explaining the error and asks for the customer name.

Table B.28: Use case Cancel order by employee

Brief description	This use case adds to the application a new order
Actors	Customer
Preconditions	The application is running and the database is up.
Postconditions	New order is added to the database.
Main Success Scenario	The customer starts filling the information of the new order. If the client is no logged the system asks for information needed for the payment. The order is updated in the database.
Alternative 1 Flow	The payment system is not active; The system stores the order information for future sessions.

Table B.29: Use case Make order by customer

Brief description	This use case shows the information about a specific order
Actors	Customer
Preconditions	The application is running and the database is up.
Postconditions	The information about the order is showed
Main Success Scenario	The customer finds the order by reference number. The information is showed.
Alternative 1 Flow	There is no order; the application shows this information.

Table B.30: Use case View order by customer

Brief description	This use case removes from the application an order
Actors	Customer
Preconditions	The application is running and the database is up.
Postconditions	The order is removed from the database.
Main Success Scenario	The specified order is selected and removed. The system updates with the carried out changes.
Alternative 1 Flow	The order does not exist; a window shows up explaining the error and asks for the customer name.

Table B.31: Use case Cancel order by customer