

Experius and TU Delft

Experius Magento Gateway

Report



Moncif Ghezza
Ron Boortman
3-7-2012

Preface

At the end of the bachelor study at the Delft University of Technology, the students have to do their bachelor thesis. The aim of the thesis is to familiarize the students with a real business environment and to apply the skills taught during their studies.

This report is about our bachelor thesis and the web application we created. The application was commissioned by Experius and was designed, created and implemented at both the Begijnkade in Utrecht where Experius is located and the Delft University of Technology from February to July 2012.

We want to express our gratitude to the university, in particular to our coordinators Peter van Nieuwenhuizen and Gerd Gross, and our supervisors Peter van der Reijden and Daniel Cronie at Experius. They supported us greatly during the project with useful input, valuable ideas and helpful assistance. Finally, we want to thank our friends and family for their endless and abundant love and support.

Moncif Ghezza & Ron Boortman.

Summary

The main goal of this project is to create a standalone application that can interact with the database of the e-commerce platform Magento. The user can use this program to collect data from his web shop database or put data back into his database. In order to run the project as smooth and fast as possible, it was divided into five stages.

The Development Stages

The project started with the Research Stage, where a few possible programming languages, frameworks and implementation methods were reviewed. This stage was meant to prepare the team for what was about to come and set them off in the right direction. Additionally, a few tools were researched with similar functionalities as the program described here. This was a good way of picking up useful ideas for the application. In this stage, the team decided to use a Test Driven Development and the Scrum development method to implement the application.

The Research Stage was followed by the Design Stage in which the design was created. During this stage, a global design was created as well as a list of requirements. Especially this list of requirements was very important since the team decided to implement the application with Scrum. This list of requirements was the Product Backlog in Scrum. Furthermore, a few Use Cases were added to help with user tests later on, as well as some interaction diagrams to create an overview of the communication between classes.

When the Design Stage was over, the team started with the Implementation Stage. This stage consisted out of four short Scrum sprints in which the application was implemented. At the end of each sprint, a part of the program was finished and shown to Experius, the Product Owner. The fourth and final sprint was not used to create a whole new functionality, but to improve some of the newly implemented functionalities.

After the main functions of the application were implemented, the team started with the Test Stage. Since the team used Test Driven Development to implement the application, it had been already extensively unit tested.

The last stage was the Documentation Stage. During this stage, this report was composed to provide information about what was implemented and to reflect the project.

Project results

In the final version of the prototype, the user can upload and download a lot of data of the Product category at once in a matter of seconds. Furthermore, it is possible to upload and download files in different formats, in particular formats used by Microsoft Office. This way, the user can use already known software to make changes in the products.

Additionally, it has been taken into account that Experius would like to expand the application with support for the other two categories, Sales and Customers, or extra features. For instance, room has been left on the webpage to add an online view of the data so the user can change information on the fly.

Table of Contents

Preface	i
Summary	ii
I. Preliminaries	2
Introduction	3
Experius	5
Magento	6
The Problem	8
II. The Project	10
Planning	11
Design	14
Implementation	18
Unit Testing	21
III. Result and Future Work	22
Conclusion	23
Recommended Improvements	25
IV. Bibliography	27
Bibliography	28
V. Appendices	30
Appendix A: Use cases of Magento	I
Appendix B: Plan of Action	VI
Appendix C: Research Document	XII
Appendix D: Design Document	XXXI
Appendix E: Sprint Reports	XLIV
Appendix F: Class Diagrams	LX
Appendix G: Magento Database	LXIII
Appendix H: SIG's reply	LXVI
Appendix I: SIG's second reply	LXVII

I. Preliminaries

- Introduction
- Experiens
- Magento
- The Problem

Introduction

According to their own description, Experius is a “full service e-commerce company in Utrecht,” who realizes the following three goals for their customers:

- Developing a successful web shop
- Raising current sales
- Gaining more profit from current customers.

Experius does this with the help of the open-source e-commerce platform called “Magento,” which enables merchants to run world-class online stores.

Although Magento is a very useful tool for creating a web shop, it is also a platform, which is yet far from ready to use by non-skilled users. Merchants who want to expand their business to the Internet can benefit from Magento. However, to set up a web shop requires a great deal of experience in programming and web development. They are skills that an average merchant does not possess. That is where Experius comes in. They set up the web shop for the merchants and even help maintaining it.

Additionally, Experius supports the marketing of the merchant’s products by, for instance, doing search engine optimization.

When a Magento web shop is being set up, the most time consuming process is filling the database. This database virtually contains all information about the store. Most data are divided into three main categories, which can turn out to be very large:

- Sales, with placed orders
- Products, containing the sellable products
- Customers

Especially in this part the normal user interface of Magento does not contains a clear and effective way to insert entities into the database or to change large quantities of data. Therefore, to save a lot of time, Experius needed an application, which can ease this process of data manipulation. The core of the project was to make such a tool

During this project, such an application is built.

In order to do this, it was important to get a good understanding of the Magento system, in particular its database. Furthermore, Experius preferred the application to be far bigger than what is possible within the scope of this project. Therefore, it is imperative that the application is expandable so that Experius can easily add features as they need.

The first part of this report, the Preliminaries, sketches a good view of the situation before the project started. The demands of Experius will be elucidated, together with some background information on the system that is currently used. Also a description of Magento will be given.

The second part, the Project, describes the process of designing, creating, implementing and testing the application. In this part, Scrum, the chosen development method, will play a big role.

The third part, Results and Future Work, focuses mainly on evaluating the application as is, as well as giving recommendations about possible future improvements. In course of time, this helps Experius to decide what features they need to see in the application.



Experius is a small company with eleven employees and two offices: one in Utrecht, the Netherlands and the other one in Macedonia. Peter van der Reijden and Daniel Cronie founded the company with their vision to change the online marketing world. They noticed that the Internet was, and still is, playing a big role in the marketing and success of ones company and products.

At the same time they noticed that most merchants use the Internet barely or poorly to their advantage. The main reason why merchants do not use the Internet is because it is relatively difficult to set up a web shop and most people do not have the knowledge to do this themselves. This is how Experius can be of service.

Experius is a 'full service e-commerce' company, which offers three main functions:

1. The development of web stores based on the framework 'Magento'. Magento is, at this moment, the most widely used open source e-commerce framework and will be highlighted further in the report. Experius is herein an official partner of Magento.
2. The development of software extensions for the Magento framework, which improves the system.
3. Online marketing services like 'Search Engine Optimization' (SEO) and 'Search Engine Advertising' (SEA).

This basically means that Experius takes on the responsibility of the customer to fully create and design a web shop and put it online. Furthermore, the customer can choose to let Experius manage and maintain their web shop, so the customer does not have to worry about anything concerning the website. Experius also has a graphical designer employed, who can create the perfect 'look-and-feel' for the customer.

For more information on Experius, please visit their website (Experius) or contact them on telephone number: 030-8200238.

Magento

Magento is a company, which offers several products and services for merchants (Magento). Their main goal is creating a simple to use e-commerce platform for their customers. First, a short summary is given about the company itself together with its products. Next, a few experiences with the Magento Platform will be given.

Magento's Products

The most commonly used product is 'Magento Community.' This is an open source e-commerce platform used to create a web shop. The code of Magento Community is freely available and everyone can download, modify and publish it under the open source OSL 3.0 license (Open Source Initiative). This is the unsupported version of the Magento Platform. Developers have to match this to their own desires and add their own functionalities when they miss something. Magento still develops these versions and releases new versions of the platform, but older versions cannot be updated easily. Merchants have to port the newer version to their own platform, if they want to upgrade.

Besides the 'Magento Community,' this company also offers a 'Magento Enterprise' package. This focuses on the same Magento Platform version as the free Community package. However, when chosen for Magento Enterprise, Magento takes over the creation and maintenance of a web shop from the merchants. For a yearly fee, Magento takes care of the whole implementation process and offers 24/7 support for their customers. Merchants can even request custom made functionalities for their website to be developed by Magento. In short, the Magento Platform is free of charge, but the Magento Company makes profit by supporting their own software and customizing it to their customers.

As a solution for small businesses, there is also 'Magento Go.' The Magento Company designs and sets up the web shop for their customer and for a monthly fee also hosts the website.

In addition to the platform and the service Magento offers, there is also a very live and active community on the website of Magento. There is a well-maintained wiki page, as well as a forum to ask questions to both Magento certified developers, as well as other people with

experience. Magento even created some sort of easy to use search engine to search through several topics or questions. If a question has not been asked yet, it is easily posted and generally answered quickly.

Real-life Experience

Besides the knowledge about all the products and services of Magento, it is also important to have a good understanding about the open source platform. A few use cases about Magento are added in Appendix A: Use cases of Magento.

These use cases describe some frequently used functionalities of the Magento Platform. Although everything looks quite fancy, it generally takes a long time and a lot of mouse clicks to change or add any information into the system. What is even worse, the import function is very prone to faulty input, which can cause a system crash at unexpected times. On top of that, the system barely gives any feedback about a failed import. Therefore it frequently happens that normal users do not understand the system or feel like the system is doing something else than requested.

For instance, it is fairly easy to overlook a random space behind a word as a user, but very hard for the system to see the resemblance between a word and the same word with an added space.

Overall the Magento Platform is a technically good platform. It looks fancy, is very extensive and mature as an e-commerce system.

If nothing goes wrong in the system, and the user does not want to divert a lot from the standard options, someone with basic computer knowledge is able to operate it. However, once something needs to change, or something goes wrong, an average computer user will not be able to fix it.

The Problem

For customers of Experius it is a problem that there is little to no possibilities for viewing and reviewing their database, which holds all sorts of information about their sales and customers. Web shop owners want to be able to see their quarterly reports and adjust the prices of their own products. At the moment this cannot be done easily in Magento. Especially large chunks of data are hard to manipulate in Magento. Therefore, Experius needs a separate tool to be developed to change this situation.

The goal of the project is to create a system that web shop owners can login to via their Magento page and with the use of 'drag and drop' conventions, can quickly create a clear overview of either all or only part of the data in their database. This data needs to be downloadable, so the user can manipulate, add or change any information in a familiar environment, like Microsoft Office Excel, and upload it again to the database.

This tool needs to be intuitive in use, easy to start with and still powerful and fast enough to do what the user wants. It is important to bare in mind the domain knowledge of the user. For instance, a lot of web shop owners would not know how to create a file with a '.sql' extension.

The data that needs to be viewed, manipulated, created and uploaded can be divided into the following three categories:

- Customer data
- Sales data
- Product data

Another issue for this project is the communication of the database with independent comparison websites like 'Independer.nl.' These websites need special 'feeds' of the products in the database for its users to compare them to products of other web shops. These feeds needs to be generated into a special XML file. Experius wants to give its customers the opportunity to create those feeds on their own.

Technical Challenge

Magento is a technically good product for implementing a reliable web shop. However, Magento lacks a good user interface for client to analyze and modify data in the database. This makes it almost impossible for general users to maintain their web shop. Moreover, even for a “silver partner” like Experius it is hard to do this.

In the Research Stage it was concluded that Magento is too complex to extend it with such a user-friendly interface. At that point it was decided to make a completely new, stand-alone application, which can interact with Magento.

The main technical challenge was the communication between this application and Magento’s database. The realizing this communication turned out to be a complex and time-consuming process.

II. The Project

- Planning
- Design
- Implementation
- Unit Testing

Planning

For a project like this, it is important to have a sound and solid planning. A planning describes the balance between time, money and resources on one hand and a good product on the other. By making a good planning, all available time, money and resources for building the application can be spent most efficiently. Of course, a planning cannot be perfect and sometimes it happens that deadlines are not made. To show how the team dealt with these deadlines, an initial planning as well as an overview of the total time spend is added. For more information about the initial planning, please look at Appendix B: Plan of Action.

Initial planning

The project is divided into the following five stages:

- Research Stage
- Design Stage
- Implementation stage
- Testing Stage
- Documentation Stage

Those stages will be roughly described here. For a more detailed description, please consult Appendix B: Plan of Action, Appendix C: Research Document, Appendix D: Design Document or Appendix E: Sprint Reports.

Research Stage

Before the project started, some research needs to be done. The first relevant thing was to dig deeper into the Magento Platform and see what makes it tick.

Besides looking at Magento, the team also needed to make some implementation choices. For instance, which programming language suited the work best and which framework fitted the best with that language. Furthermore, there are some already existing tools that preform similar tasks to what is wished by Experius. A more detailed explanation about the choices made in the Research Stage, can be found in Appendix C: Research Document.

Design Stage

Scrum prescribes a global design and list of requirements. After all relevant information was gathered a global design was made. The list of requirements has been set up together with Experius, playing the role of Project Owner, to come up with the maximum of possibilities. The final list as well as the global design can be found in Appendix D: Design Document.

Implementation Stage

Due to the nature of “the Sprint way” of Scrum it was important to cut the project into several smaller problems for the sprints. Additionally, a more detailed design had to be created in the beginning of each sprint based on both the global design and the list of requirements. This detailed design was not only important for the team to create a good overview of what was about to develop, but also this design formed a base for the test cases. After all, the test cases had to be created before any real implementation was done, because of the use of Test Driven Development. All sprint reports with a more detailed description of the design, as well as recommended improvements can be found in the Appendices.

Testing Stage

During the Implementation Stage, most parts of the prototype were already unit tested due to the use of Test Driven Development. Unit testing however does not give any guarantees about the correctness of the program as a whole. Therefore, after the Implementation Stage it was planned to do extensive user testing. This turned out to be too ambitious and so extensive user testing has been dropped by the team due to time constraints. Of course, at the end of each sprint, Experius reviewed the prototype. This offered some insight in the user’s experience of the prototype. Unit testing and Test Driven Development will be covered later in this part in chapter Unit Testing.

Documentation Stage

Since the documentation is often underestimated the team tried to start with documenting at an early stage. The initial planning was to work on the final report during the other stages, so it would not take up so much time in the end. Unfortunately, this turned out to be very difficult and most documentation was written at the end.

The initial planning can be found in Figure 1. Some elaboration on this chart can be found in Appendix B: Plan of Action



Figure 1: The initial planning

Actual time spent

In early stage it was clear that the desires of Experius were a lot bigger than was possible to realize within the scope of this project. Therefore, right from the start choices were made and priorities needed to be set. It resulted in a fairly good separation of what were the realistic targets for the project and what was just desired.

During the Research Stage it was noted that the list of available tools was a lot bigger than the team had anticipated. This resulted in a longer stretched Research Stage, as well as some additional work in the Design Stage. In the end, all targets were met in the time allocated for this project

An overview of the resulting planning can be seen in Figure 2.



Figure 2: The resulting time spent

Design

As aforementioned in this document, there were two separate design processes. The first process was during the Design stage, where a global design was created. The second process was during the Sprints, where a more detailed design was drawn. In this chapter, the requirements will be elaborated on, as well as other parts of the design. At the end, a brief summary of the design of each sprint will be given.

Design Stage

During the Design Stage, four different types of designs are created:

1. List of Requirements
2. Global Class Diagram
3. Some important Use Cases
4. Interaction Diagram

These designs will be briefly described here, but for more information, please refer to Appendix D: Design Document.

List of Requirements

The List of Requirements can basically be divided into two groups: Functional Requirements and Quality Requirements. As the names suggest, Functional Requirements cover the requirements about the functionality of the application. They tell about what the application should be capable of performing. Quality Requirements however, are about the quality of the application and source code.

The requirements are put together with the team and Experius. In one big brainstorming session a lot of wishes from Experius surfaced, which resulted in the list of requirements.

Functional Requirements

The Functional Requirements are again split into parts: the Front End and the Back End. The Front End is the part visible to most users. This is where the real power of the application lies. Therefore, this part holds the most requirements.

The Back End of the application is not visible to the regular user and is meant to manage the application and its users. In here, a 'super user' is able to add or remove users from the system.

The Front End describes several functions, regarding: Login, Import, Export and creating Reports. These functions form the heart of the application and will cover the main functionality.

Quality Requirements

Like it was stated before, Quality Requirements are about the quality of the software and source code.

Important here is that the application is scalable, maintainable and expandable. One side note is required however; because Experius is stationed in two different countries, it is important to have to whole system at least in English, with Dutch as option.

Global Class Diagram

Due to the chosen framework, the application had to follow the rules of a Model-View-Controller (MVC) model. The basic idea of a MVC model, is that classes with different functions are separated into three classes:

- *Model*, where all the data manipulation and calculation is done.
- *View*, this is the communicator between the user and the system.
- *Controller*, which directs all tasks to the correct classes.

This MVC model is clearly seen in the Global Design Diagrams. It has to be noted that all class designs created during the Design Stage are not representative anymore for the resulting prototype. For a complete and up-to-date overview of the prototype's class diagram, refer Appendix F: Class Diagrams

The Controller

In Figure 3 the classes representing the controllers are shown. The controller consists out of three main classes: ControllerLogin, ControllerFrontend and

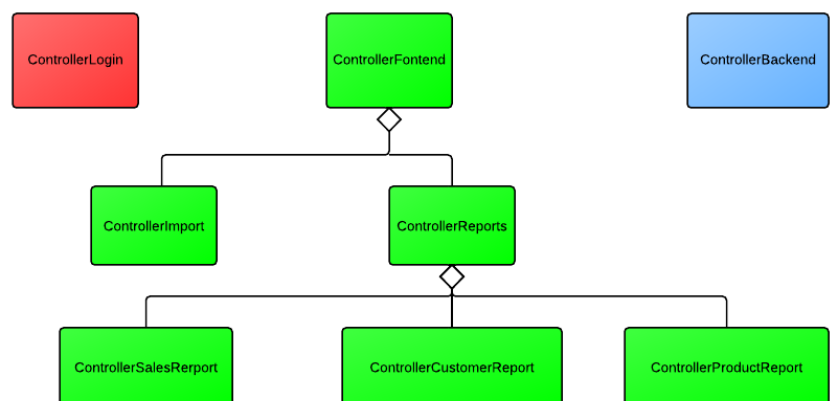


Figure 3: Global design - the controller

ControllerBackend. These controllers are used respectively for the login function, the front end functionality and the back end functionality. Once more, the classes representing the front end are the biggest.

The Model

In Figure 4 the classes representing the models are shown. Already very noticeable at first sight, is that this diagram is a lot bigger and more sophisticated than the one for the controller. This is because most of the actual work will be done in one of the Model's classes. In Figure 4,

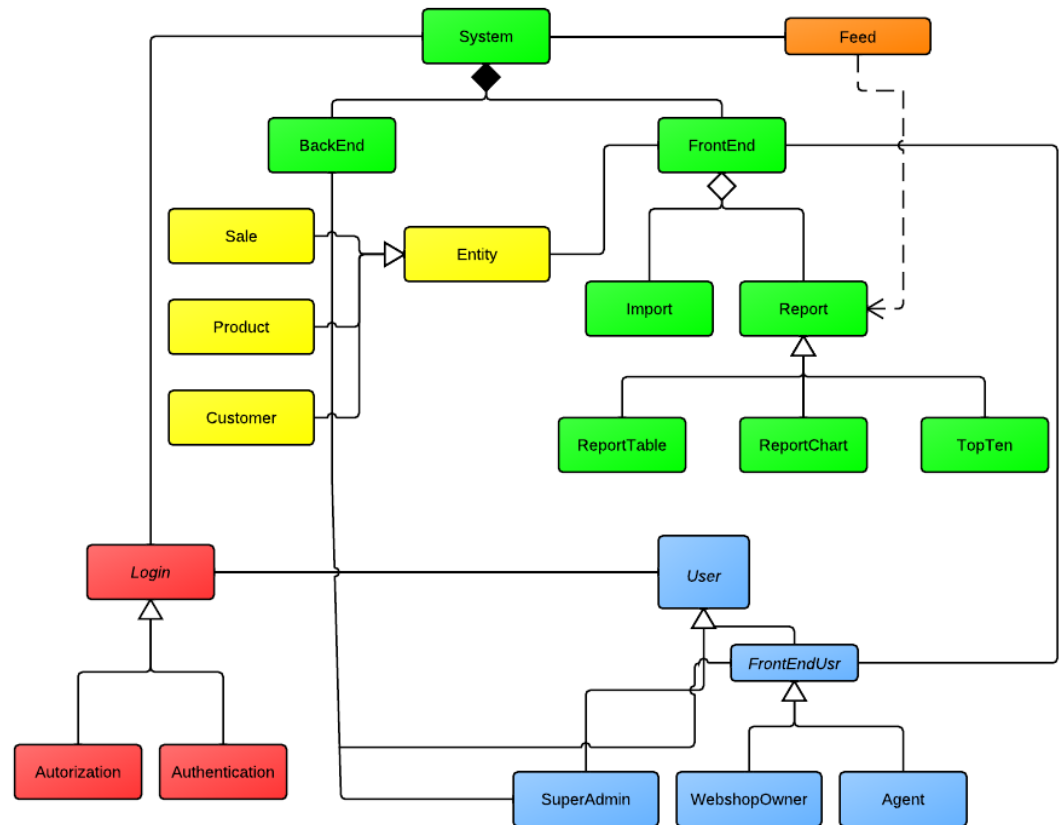


Figure 4: Global design - the model

the colours are coherent with those of the Controller. The two extra colours here, yellow and orange, represent respectively the different main categories and the feed functionality.

View

Of the view, no class diagram can be made, since the View does not consist out of classes. Instead, every controller class will have its own view.

Use Cases

The Use Cases itself can be found in Appendix D: Design Document. These Use Cases describe some of the main functionalities of the system and will form a perfect base for the user testing. Here again, the Use Cases are divided into four functionalities: Login, Import, Export and creating Reports. The reason that not all possible Use Cases were created, but rather the ones that cover the main functions, is because the team regarded the Use Cases as

less important as other aspects of the design. However, due to the Use Cases' importance later on during the Testing Stage, they were still created and added to the design.

Interaction Diagram

The Interaction Diagrams, also to be found in Appendix D: Design Document, are added to give a good insight in the 'flow' of the application. Since not all flows are evenly complex and time was limited, only two Interaction Diagrams are made of the most complex communication between classes. The login function was chosen, because it interacts with the most classes and the import function was chosen, because it has to interact with the Magento database. In both Interaction Diagrams the underlying MVC model is visible again; an assignment goes from the user through the controller to the model, and back again.

Implementation

This chapter illustrates the process during the Implementation Stage. Due to the nature of Scrum, this part can easily be divided into the different sprints. In total there were four sprints. Refer to Appendix E: Sprint Reports for more information about the specific sprints. It has to be noted that because time was limited, the first priority was on the Products category. However, it has been taken into account that Experius wanted to be able to extend this prototype to other categories. With that view, the code has been made very scalable.

Sprint 1

The first sprint turned out to be more difficult than anticipated. The login functionality was planned for the first sprint, which is not as big and difficult as the other functionalities in store. However, to get the project up and running took a lot of effort. The team tried to make a connection between a PHP server (PHP Cloud), a GIT server and Zend-Studio. This way, there would have been an online active version of the program, saved with the version control of GIT that could be edited in a platform familiar with the Zend-Framework: Zend-Studio. Alas, it kept giving difficulties for the team, so they decided to replace the GIT server with an SVN server for the TU Delft, which turned out to work a lot better.

Besides setting up the project, the team also struggled to get things going with PHPUnit. PHPUnit is specially created for the PHP language to do unit testing. The team never worked with this extension before, which meant the team had to put some extra time into learning the methods.

Due to the obstacles encountered during the set up, the team was already threatened to fall behind. However, since the login function was not too difficult, the team still managed to finish a working version at the end of the first sprint.

Despite this success, the team could not finish all unit test cases, which meant more work later on.

Sprint 2

In the second sprint the Product Owner, Experius, wanted to see the most important and difficult function of the whole application: the import function. The team found out that the API delivered by Magento dramatically slowed down the import of a big file when connecting to the database. Therefore, it was Experius' desire to connect to Magento's database directly without the use of that API. In order to implement this, the team first needed to get a better understanding of the underlying Magento database. This understanding has been put to words in Appendix G: Magento Database.

After the team had a better understanding of the Magento Database, they still had a monumental task ahead of them. For instance, all data from the database and from the uploaded file needed to be mapped, linked and saved. Additionally, a difference had to be made between a new product, which needed to be added, and already existing products, which only needed some change of information.

Despite all the difficulties, at the end of the sprint a working version could be presented to Experius!

Sprint 3

After the second sprint ended, the team, the Product Owner and the coordinator from the TU Delft came together to discuss the progress. During this meeting it became clear what the purpose of the third sprint would be: the export function. Experius explained it as a circle, in which the user could first export data from the database, then modify the data in a familiar setting and finally import the modified data again to the database, to be exported again for changes later on. In order to do this safely, it is also important to create some kind of backup of the database, before any file will be uploaded.

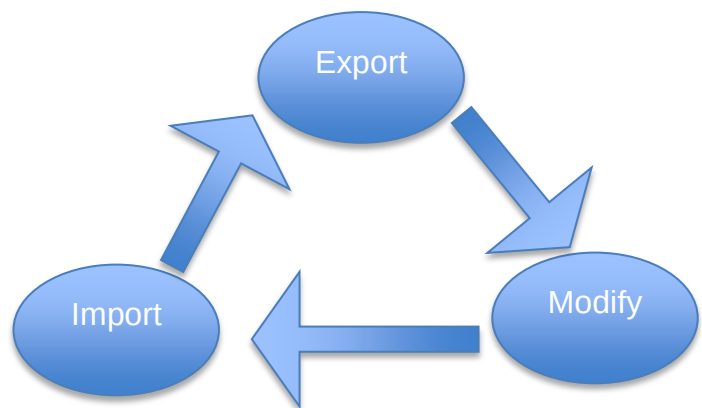


Figure 5: The application circle

During the third sprint the team found out that some changes needed to be made in the original requirements regarding reports. The new requirements have been written down in the Sprint Report and can be found in Appendix E: Sprint Reports.

At this time, the team already had a better understanding of Magento's database, which made creating an export function a lot easier. Moreover, an external library called "PHPExcel" was used to create '.xlsx' and '.csv' extended files. At the end of the sprint, a working version could be presented to Experius.

Sprint 4

During the forth and last sprint, no new features were added. During this sprint some of the existing features had to be improved.

Initially there was time allocated in the fourth sprint to make changes to the code as a result of the review from the Software Improvement Group. However, as can be read in Appendix H: SIG's reply, they were also very pleased with the work of the team. There was one point of improvement suggested by the Software Improvement Group, which has been solved during this sprint.

At the end of the third sprint, when the prototype was shown to Experius, a few bugs came up. One bug that was solved during this sprint was a bug that occurred when a new product was uploaded. The team solved this issue by checking for wrongly updated attributes. Besides the mentioned bug, the team also made some efficiency changes.

Now the prototype works a lot faster and more efficient than ever before.

Unit Testing

Since the team was working with a Test Driven Development method, Unit Tests were created for most classes before the actual class was implemented. The idea of Test Driven Development is to create a test case first, before implementing any functional code. This way, more professional ideas go into the actual implementation of the class itself.

In the PHP language, a set of classes are created for the exact purpose of Unit Testing called 'PHPUnit.' This package comes with great methods and assertions, which can help the programmer to test his code. This project uses PHPUnit for Unit Testing.

That Test Driven Development greatly improved the quality and expandability of the source code can be seen in the feedback from a company called: 'the Software Improvement Group.' This company tests the quality of source code based on a few criteria.

The team is proud to announce that the source code created during this project got a rating of five stars out of five! The fully elaborated reply of the Software Improvement Group can be found in Appendix H: SIG's reply

III. Result and Future Work

- Conclusion
- Recommended Improvements

Conclusion

The main goal of the project was to create a prototype, which can interact with a Magento database, reading and writing its data. This main goal has been achieved and a successful application has been built.

The problem was more complicated than initially anticipated. A more extensive project description involves among others: creating reports, importing and exporting files, expandable coding, error handling and feeds. Unfortunately, time was limited and therefore the team did not to implement everything as perfectly as the team would have liked.

The team did create a prototype to the best of their efforts and managed to implement most requested functions. The most important thing is that the “application’s circle” (Figure 5) has been closed. This is a circle that visualizes the main and basic function in a basal way. The application can be used to retrieve information from Magento’s database, let the user modify this data in a familiar environment and put the modified data back into the database. With a few bugs aside, this circle has successfully been completed and implemented.

During the course of this project, a lot of obstacles have been overcome. One of the biggest obstacles was to establish a connection between the application and Magento’s database. It was difficult to implement, but it finally resulted in a faster and more efficient way than possible with Magento’s API.

Another big problem that was overcome was changing data once the connection was established. One of the difficulties was that the prototype had no knowledge in advance whether an uploaded product already exists, and thus needs to change, or is non-existent, which means it has to be added.

As aforementioned, the requirements can be divided into two lists: functional and quality requirements. The full list of requirements can be found in Appendix D: Design Document. Table 1 represents the functional requirements implemented by the team. In this table, green represents implemented requirements and red represents not yet completed functions. A similar overview of Quality requirements is represented in Table 2.

Table 1: Functional requirements

1			2			3			4			5								6									7	8
1	1	2	1	2	3	4	1	2	3	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8	9	1	1		

Table 2: Quality requirements

1	2	3	4	5	6	7
---	---	---	---	---	---	---

All together, the project was a success. A good and usable prototype was created and the team got a good look at what it means to work in a team and in a business environment. In its current state, Experius can use the prototype.

The code of this prototype is of high quality, which was concluded by the Software Improvement Group by rating it a five star rating out of five. In Appendix H: SIG's reply an e-mail can be found with the reply from the Software Improvement Group on the source code.

With a bit more time, the user interface can be improved and the application will be useful for the customers of Experius too.

Recommended Improvements

During this project, the team encountered some points that need to be improved. A detailed overview of recommended improvements is given at Appendix E: Sprint Reports. Below a summary of these improvements is given.

Login

Login improvements are recommended regarding safety and privacy. User passwords are saved with some encryption (MD5), but there are other and more sophisticated methods for adding encryption. Furthermore, the login via the Magento back end is not through a secure channel.

Import

One of the most urgent points of improvement regards the system's feedback. During the import of a file there is little to no feedback to the user in case something goes wrong. At the moment the system just halts, or gives a PHP error message, which is not very useful. This can be improved by catching several errors and use AJAX to generate output in a fancy matter.

Back up

A back up function has been added to the system in order to make a back up of the Magento database before uploading a file. This back up is saved in a unsecure way, which should be improved to retain the user's privacy.

Database Handling

A different library exists concerning database translation and communication than the one used in the application, which could prove to be more efficient. This other library is called 'Doctrine2' and is specialized in database relations and mapping data to objects.

Overall Improvements

One of the overall improvements concerns the user interface. The application has been designed, created and implemented by prospective computer engineers and performs as required. However, it could be made more fancy by a graphical designer.

Furthermore, it is recommended, and probably also wanted by Experius, to implement other categories of Magento like Sales and Customers. While implementing the program, it has been taken into account that Experius wanted this and made sure that those two categories can be added any time in the same manner as the Product category is implemented now.

IV. Bibliography

Bibliography

AheadWorks. (2012). *Advanced Reports Magento*. Retrieved 03 07, 2012, from AheadWorks: <http://ecommerce.aheadworks.com/magento-extensions/advanced-reports.html>

de Sousa, S. (2012). *The Advantages and Disadvantages of Agile SCRUM Software Development*. Retrieved April 21, 2012, from My PM Expert: <http://www.my-project-management-expert.com/the-advantages-and-disadvantages-of-agile-scrum-software-development.html>

Experius. (n.d.). *Home page*. Retrieved June 20, 2012, from Experius: <http://experius.nl/>

GoMage. (n.d.). *GoMage Feed Pro*. Retrieved 03 07, 2012, from <http://www.gomage.com/extensions/gomage-feed-pro.html>

Jakobsg, e. a. (2009, 11 19). *List of web service frameworks*. Retrieved 03 07, 2012, from Wikipedia: http://en.wikipedia.org/wiki/List_of_web_service_frameworks

Laganière, T. C. (2005). *Object-Oriented Software Engineering*. Mc Graw Hill.

Magento. (n.d.). *Home*. Retrieved June 20, 2012, from Magento eCommerce Platform for Growth: <http://www.magentocommerce.com/>

Magento. (n.d.). *Magento - API*. Retrieved 03 07, 2012, from Magento: <http://www.magentocommerce.com/wiki/doc/webservices-api/api>

Magento. (2012). *Magento Features eCommerce*. Retrieved 03 07, 2012, from Magento: <http://www.magentocommerce.com/product/features/analytics-and-reporting/view>

Magento-knowledge base. (n.d.). Retrieved 03 07, 2012, from Magento: <http://www.magentocommerce.com/knowledge-base/entry/tutorial-using-magentos-import-export-profiles>

MagneticOne. (n.d.). *Store Manager for Magento*. Retrieved 03 07, 2012, from Store Manager for Magento: <http://www.mag-manager.com/>

Magneticone. (n.d.). *Store Manager guide*. Retrieved Maart 22, 2012, from <https://support.magneticone.com/index.php?/Knowledgebase/List/Index/4/magneticone-store>

Open Source Initiative. (n.d.). *Open Source Initiative OSI - The Open Software License 3.0 (OSL-3.0:Licensing | Open Source Initiative*. Retrieved June 20, 2012, from Open Source Initiative: <http://www.opensource.org/licenses/osl-3.0.php>

Schwaber, K., & Sutherland, J. (2011, October). *The Scrum Guide*. Retrieved April 21, 2012, from http://www.scrum.org/storage/scrumguides/Scrum_Guide.pdf

Scrum Alliance. (2012). *Scrum's Three Roles*. Retrieved April 21, 2012, from Scrum Alliance: http://www.scrumalliance.org/pages/scrum_roles

Waterfall Model. (2012). *All about the Waterfall Model*. Retrieved April 21, 2012, from Waterfall Model: <http://www.waterfall-model.com/>

Wickens, C., & Liu, Y. (2004). *An Introduction To Human Factors Engineering*. Pearson Prentice Hall.

Wikipedia. (2012). *V-Model (software development)*. Retrieved April 21, 2012, from Wikipedia: [http://en.wikipedia.org/wiki/V-Model_\(software_development\)](http://en.wikipedia.org/wiki/V-Model_(software_development))

V. Appendices

Appendix A: Use cases of Magento

In this chapter, a few important use cases are described about the current situation in Magento. Actions like inputting new orders, creating products and importing data.

ID	1
Name	Creating new product.
Description	The web shop owner wants to add a new product to his available list of products.
Primary actor	Web shop owner.
Precondition	User is logged in.
Use case	• User goes to the <i>Product Management</i> window, by hovering over the menu item <i>Catalog</i> with its mouse and clicks on <i>Manage Products</i>. • Now the User clicks on the <i>Add Product</i> button. • In the next page, the User can select the attribute set and type of product to add, before clicking on the <i>Continue</i> button. • In the appearing page, all fields with a red asterisk have to be filled in. • After all necessary fields are filled in, the User has to click on <i>Prices</i> in the navigation bar on the left side of the screen. • On this page, also all fields with a red asterisk have to be filled in. • If the User desires to add more information to its product, it can do so in one of the menus listed in the navigation bar. • Now the User can click on the <i>Save</i> button, in the top right corner to save its product in the list of available products.
Post condition	A new product is created and added to the available products.

Pros

The page for adding new products looks quite fancy. The eye of the User is attracted to the right places. Also all the fields are clearly named.

Cons

It takes a lot of effort to add one product. Let alone adding several products. Also the menu structure is overwhelming and not very clear. For instance, it is not clearly defined that the User has to go to *Prices* after filling in all the fields for the product details.

ID	2
Name	Creating new customer.
Description	The web shop owner wants to add a new customer to the list of all (active) customers.
Primary actor	Web shop owner.
Precondition	User is logged in.
Use case	• User goes to the <i>Customer Management</i> window, by hovering over the menu item <i>Customers</i> with its mouse and clicking on <i>Manage Customers</i>. • Now the User clicks on the <i>Add New Customer</i> button. • In the appearing page, all fields with a red asterisk has to be filled in. • If the User desires to add more information to the customer, it can do so in one of the menus listed in the navigation bar. • Now the User can click on the <i>Save</i> button, in the top right corner to save the customers in the list of all customers.
Post condition	A new customer is created and added to the list of all (active) customers.

Pros

The pages here are looking fancy as well. When scrolling down, all important navigation buttons are moved with the screen. Also checking is done for fields like *Email*.

Cons

Especially adding an address for a customer is not really intuitive. It is a bit tucked away in hordes of menus.

ID	3
Name	Creating new order.
Description	The web shop owner has a customer <i>A</i> , who wants to order a product <i>P</i> . Now the web shop owner has to create a new order.
Primary actor	Web shop owner.
Precondition	User is logged in. Both customer <i>A</i> and product <i>P</i> are in the database already. Customer <i>A</i> already has an address added.
Use case	• User goes to the <i>Order Management</i> window, by hovering over the menu item <i>Sales</i> with its mouse and clicking on <i>Orders</i>. • Now the User clicks on the <i>Create New Order</i> button. • In the appearing page, customer <i>A</i> can be chosen from a list of all customers. • When the customer is chosen, a product can be added by clicking on the <i>Add Product</i> button. • Now, a list of all available products is displayed. The User has to select product <i>P</i> from this list, together with the amount that was ordered by <i>A</i>. • After selecting the right product(s), with the right quantities, the User clicks on the <i>Add Selected Product(s) to Order</i> button. • Now customer <i>A</i> and product <i>P</i> have been chosen, the User has to choose (or input if not available) the correct billing and shipping address for <i>A</i>. A side note has to be made here: although the field <i>State/Province</i> is not marked by a red asterisk, it is still required to fill this in. • When the correct addresses have been chosen, the User has to choose the right <i>Payment Method</i> and <i>Shipping Method</i>. Another side note: also these fields do not contain any red asterisk. • Now the User can click on the <i>Submit Order</i> button, in the top right or the lower right corner to place the order.
Post condition	There exist a new order for customer <i>A</i> of product <i>P</i> .

Pros

All relevant information comes at the right time. It is quite easy to select and add a product or a customer from a list.

Cons

The error messages about invalid or incomplete data come only in the end. When this message appears, some fields are even reset, which is pretty annoying. Furthermore, all fields and buttons are far apart from each other. All in all, it takes a lot of time to create a new order.

ID	4
Name	Exporting products
Description	The web shop owner wants to export a list of all products, so he can make changes to it.
Primary actor	Web shop owner.
Precondition	User is logged in.
Use case	• User goes to the <i>Export</i> window, by hovering over the menu item <i>System</i> with its mouse, then hover over the list item <i>Import/Export</i> and then click on <i>Export</i>. • Now the User can choose <i>Products</i> in the <i>Entity Type</i> field and the format of the file in <i>Export File Format</i>. • Here, the User can apply certain filters to the file it wants to download. • If the User applied all filters it wants to apply, it can click on the <i>Continue</i> button, in the lower right corner to download a file in the requested format.
Post condition	An external file is downloaded with information about all products.

Pros

Other than the existence of this function, it is quite hard to find any good points about it.

Cons

When the User clicks on export, only two initial options are shown, which looks promising. Though after selecting something, a whole list of (to the User) meaningless information is shown. There are a lot of attributes which are only used by Magento and trivial

to the User. Furthermore, the options given work counter-intuitive and only slow the whole process down. For instance, it is not very clear what the option *Skip* does.

ID	5
Name	Importing products
Description	The web shop owner has an external file with changed data about product <i>P</i> and wants to import this in the database.
Primary actor	Web shop owner.
Precondition	User is logged in and has an external file with the changes made to some products in the database.
Use case	• User goes to the <i>Import</i> window, by hovering over the menu item <i>System</i> with its mouse, then hover over the list item <i>Import/Export</i> and then click on <i>Import</i>. • Now the User can choose <i>Products</i>. The manner of importing can be chosen in the <i>Entity Type</i> (appending, replacing or deleting). • Then the User clicks on the <i>Choose File</i> button, navigates to the file it wants to upload and clicks <i>Open</i> to select a file. • After the right file is selected, the User clicks on the <i>Check Data</i> button, in the top right corner. • If the file passes the test, the User has to click <i>Import</i> on the bottom of the page to import the data from the file
Post condition	The changes to the products have been made.

Pros

Also here, other than the existence of this function it is hard to find any good points about this.

Cons

The import function is very prone to faulty input. When one or two columns are forgotten, Magento already gives a few error messages. Also the option *Replacing Existing Complex Data* is buggy. When this was tested, already existing data was appended, rather than changed to the correct format.

Appendix B: Plan of Action

Introduction

This document describes the Plan of Action of the Bachelor Project: *"Insert app name here."* Here will be described how the project is set up. Things like a project description, planning and project management tools will be highlighted.

Motivation

Experius is a company specialized in the creation and design of e-commerce websites for its clients on a Magento platform. Experius has experienced that its customers are having troubles with reading and customizing the data saved in the Magento database. For instance, it is very hard for a web shop owner to adjust many product prizes in a good and efficient way, without consulting Experius. To get a better customer satisfaction, Experius wants to change this by developing an application.

Project description

Experius has requested a web-based application for better, faster and more intuitive connection to the Magento database. This way, Experius is in a better position of satisfying its customers.

Environment

The application will be developed in an environment known and familiar to Experius. The application needs to run on one of the servers of Experius and communicate with different Magento users. Also a database will be kept to give or deny certain users access.

Demands and limitations

The application should not take up more space than 1000 MB. This includes not only the database, but also the written code as well as the artwork for the application. Because Magento is developed in PHP (using the Zend Framework) and there is a demand of Experius for maintainability, the application will be developed in the Zend Framework as well. The

application will also have to be intuitive and easy to use. This way it has real added value for the web shop owners.

About the project itself, Experius wants to be closely involved and have an overview over the project. Therefore Experius has requested that the application will be designed and developed at Experius at least once a week.

Also every step needs to be good documented. It is important for Experius that they can understand every part of the process, and recreate all steps in the project. This way it will be easy to extend and expand the application. It is indicated by Experius that they desire this.

Approach

Before the application can be developed, it first needs to be designed. Designing in greater detail will result in a more efficient development phase, because everybody knows exactly what to do.

Research

Before a design can be made, some research needs to be done by the team. The research stage can include finding similar tools to what will be developed and test that, but also it can include finding the right programming language to implement. The following things will be created in the research stage:

- *Domain analysis:*

The domain analysis will describe the domain of the current situation. This includes a glossary of all terms used throughout the report and an analysis of all the stakeholders involved.

- *Current situation:*

This part will describe the current situation. It describes some flows, actions and tools available now, which are relevant to the application. This includes some of the report techniques already existent in Magento as well as how import is done now.

- *Available tools:*

This part describes all already available tools. Either tools which can be similar to the application or tools, which can be used to develop the application.

Design

The application will follow the '*Model-View-Controller*' (MVC) model. This means there is a distinctive difference between the models, the views and the controllers of the software to be created. Evidently, this will be visible in the design of the application.

In the design stage, the following documents will be created:

- *Requirements:*

The requirements will be a document, which describes what the application should and should not be able to do. The requirements will be divided into a few categories, in accordance to their importance.

- *Class diagram:*

A general class diagram will be created. This diagram describes the rough outlines of the software. Especially the interaction between classes and the communication between the different MVC components will be highlighted. This diagram shows directly how big the application is going to be and everything that needs to be implemented.

- *Use cases:*

A few use cases will be made for the application. These use cases will be about the main functionality and important scenarios of the application. The use cases will form a good base for testing.

- *Interaction diagrams:*

For the important use cases, an interaction diagram will be created. These diagrams will make it apparent how the application handles and processes requests and can be used as a base for integration testing.

Implementation

Two very important principles will be applied to the implementation stage: '*Test Driven Development*' (TDD) and '*Scrum*.' Both principles will not be followed completely, but rather adapted to the situation.

- *TDD:*

The application will be developed with a TDD point of view. This will save a lot of time in the test stage later on. Though PHP does not support as many testing techniques as Java, it does have a lot of classes for unit testing. Furthermore, the Zend Framework provides an extensive set of functions for this type of testing. This will come in handy when doing TDD.

- *Scrum:*

For the development, a new and agile development method will be used: Scrum. Though the development team is too small to implement a full Scrum method, certain base principles will be applied. For instance the sprints in Scrum will be used. In general, sprints of six days, spread out over two part-time weeks, will be used. After those two weeks, a (small) part of the application will have been implemented and can be shown to Experius and users.

Testing

Because TDD will be applied to this project, the testing stage will be relatively small. This stage will mainly consist of interaction testing and user testing. Bugs found in this stage will be added to the bug report so that they can be fixed. At the end of the project, if this bug report is not empty yet, an explanation will be given.

Planning

Of every project, planning is a very important aspect. It is used to describe a good balance between time, money, resources and good products. In this planning, every stage will be described with its deadline.

- **Research Stage** Deadline: March 25th, 2012
 - Looking for tools already available.
 - Creating the research document.
 - Creating the plan of action document.
 - **Deliverables:** Research Document, Plan of Action.

- **Global Design Stage** Deadline: April 23rd, 2012
 - Setting up the requirements of the system.
 - Creating all the diagrams, which will help with the implementation and testing.
 - Designing a rough graphical user interface for the application.
 - **Deliverables:** List of requirements, Class diagram, Use Cases and Interact diagrams.

- **Implementation Design Stage** Deadline: After start of each sprint
 - Defining what to implement each sprint.
 - **Deliverables:** Implementation plan.

- **Implementation Stage** Deadline: June 10th, 2012
 - Implementing all diagrams described in the design stage.
 - Implementing the graphical user interface.
 - Every two weeks a new sprint starts.
 - **Deliverables:** Working software, Source code.

- **Testing Stage** Deadline: June 25th, 2012
 - Test the application for bugs and faults.
 - Test the application whether it corresponds to everything in the design stage.
 - Removing all known bugs.
 - **Deliverables:** Working software, Source code.

- **Documentation** Deadline: June 25th, 2012
 - Writing documentation about the application.
 - Writing a report about all the processes and progresses.
 - **Deliverables:** Bachelor Thesis report.

- **Presentation** Deadline: n.a.
 - Presenting the created application.
 - **Deliverables:** Presentation, slides.

Time schedule

A schedule has been created to visualize the time scheduled for every part. The schedule is divided into nine time slots, each consisting of two weeks. During the first eight weeks (first four slots), two days a week will be allocated by the team to work on the project. In the last ten weeks (time slot five to nine), three days a week will be allocated by the team to work on the project. This has also been visualized in the table below.

Furthermore, there will not be worked on the project in the eighth week.

	Time slot								
Stage	1	2	3	4	5	6	7	8	9
Research									
Global Design									
Implementation Design									
Implementation									
Testing									
Documentation									

The time slots are divided in the following way:

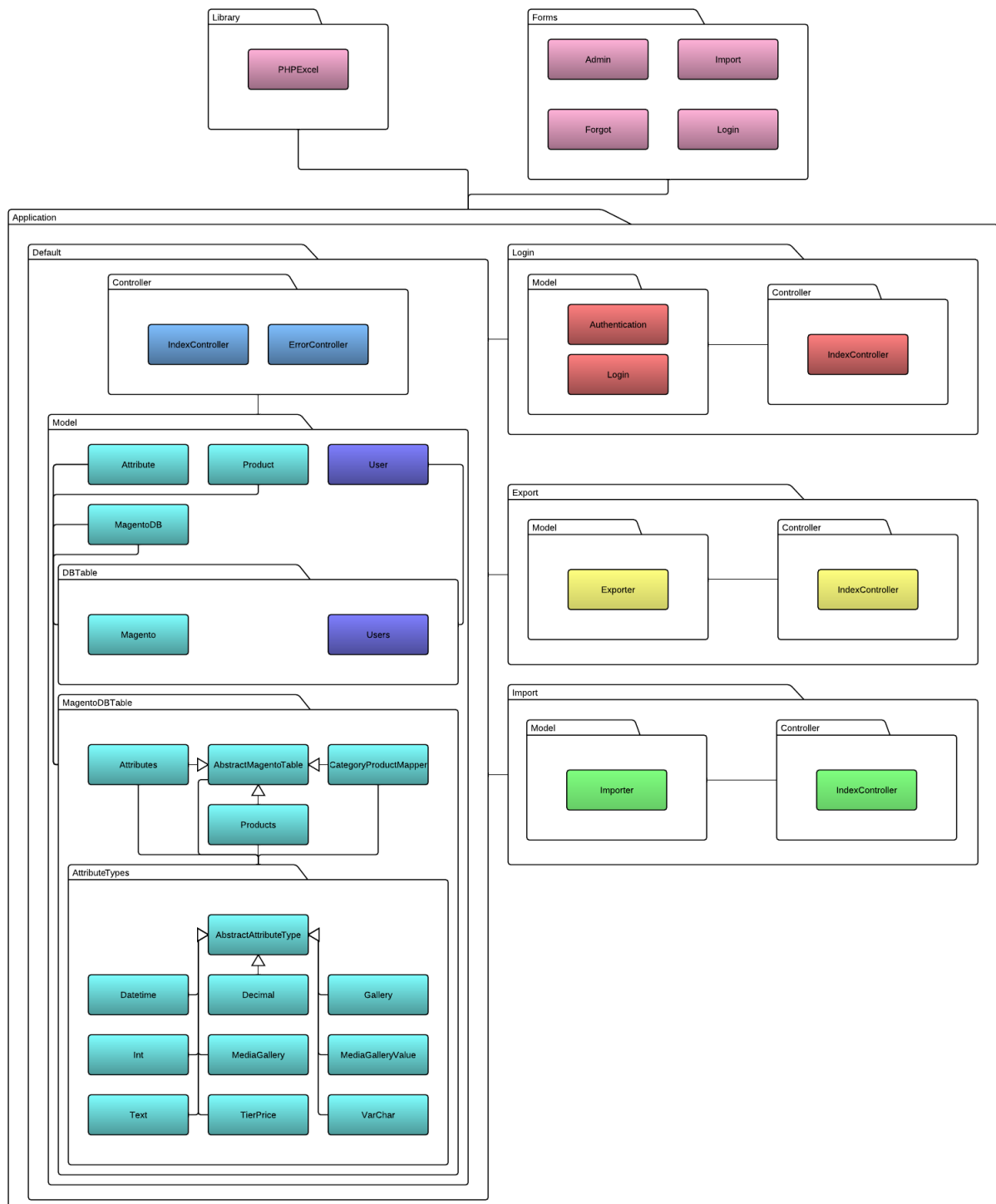
- 1) From: February 20th, To: March 11th
- 2) From: March 12th, To: March 25th
- 3) From: March 26th, To: April 8th
- 4) From: April 9th, To: April 22nd
- 5) From: April 23rd, To: May 6th
- 6) From: May 7th, To: May 20th
- 7) From: May 21st, To: June 3rd
- 8) From: June 4th, To: June 17th
- 9) From: June 17th, To: July 1st

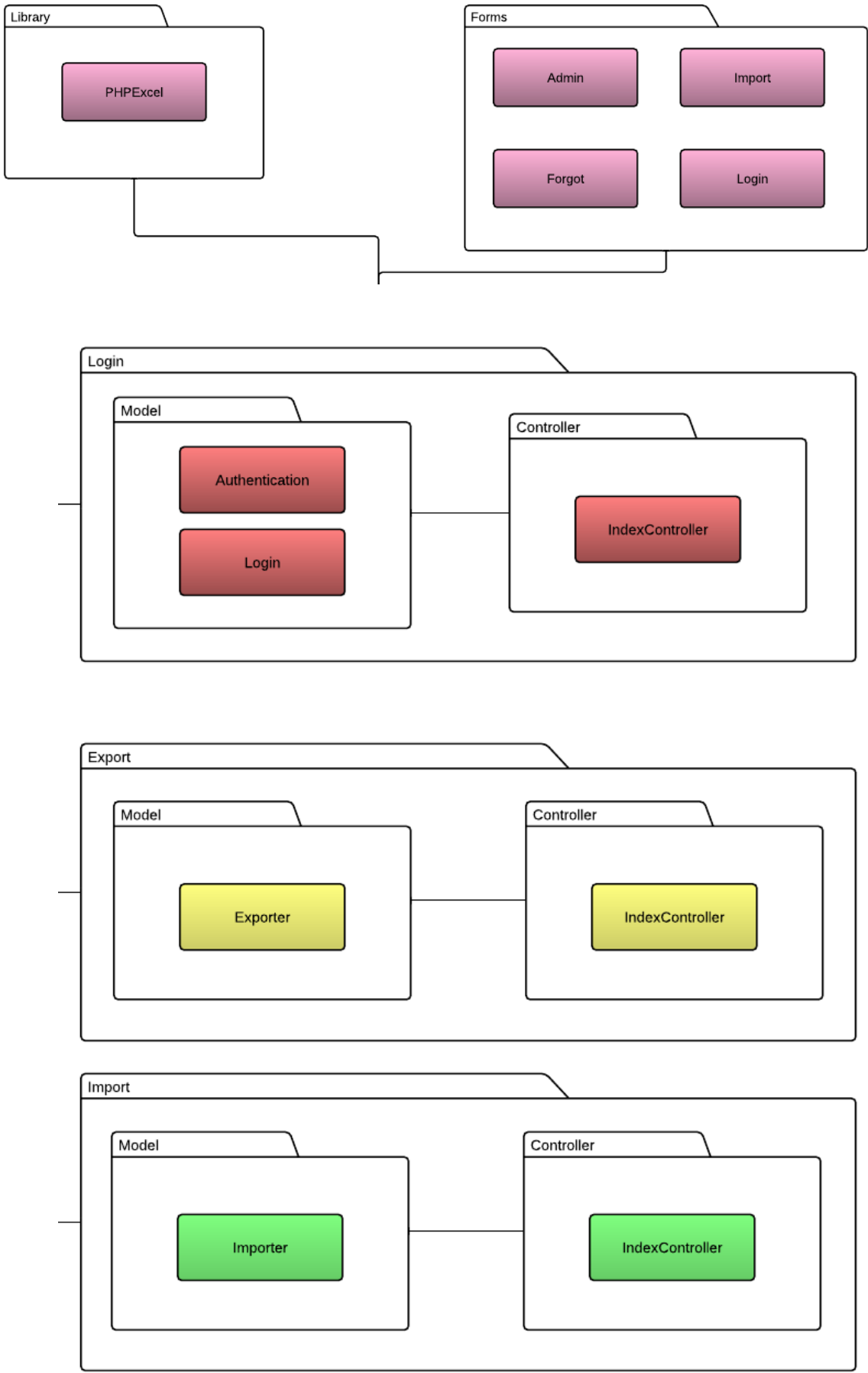
Appendix C: Research Document

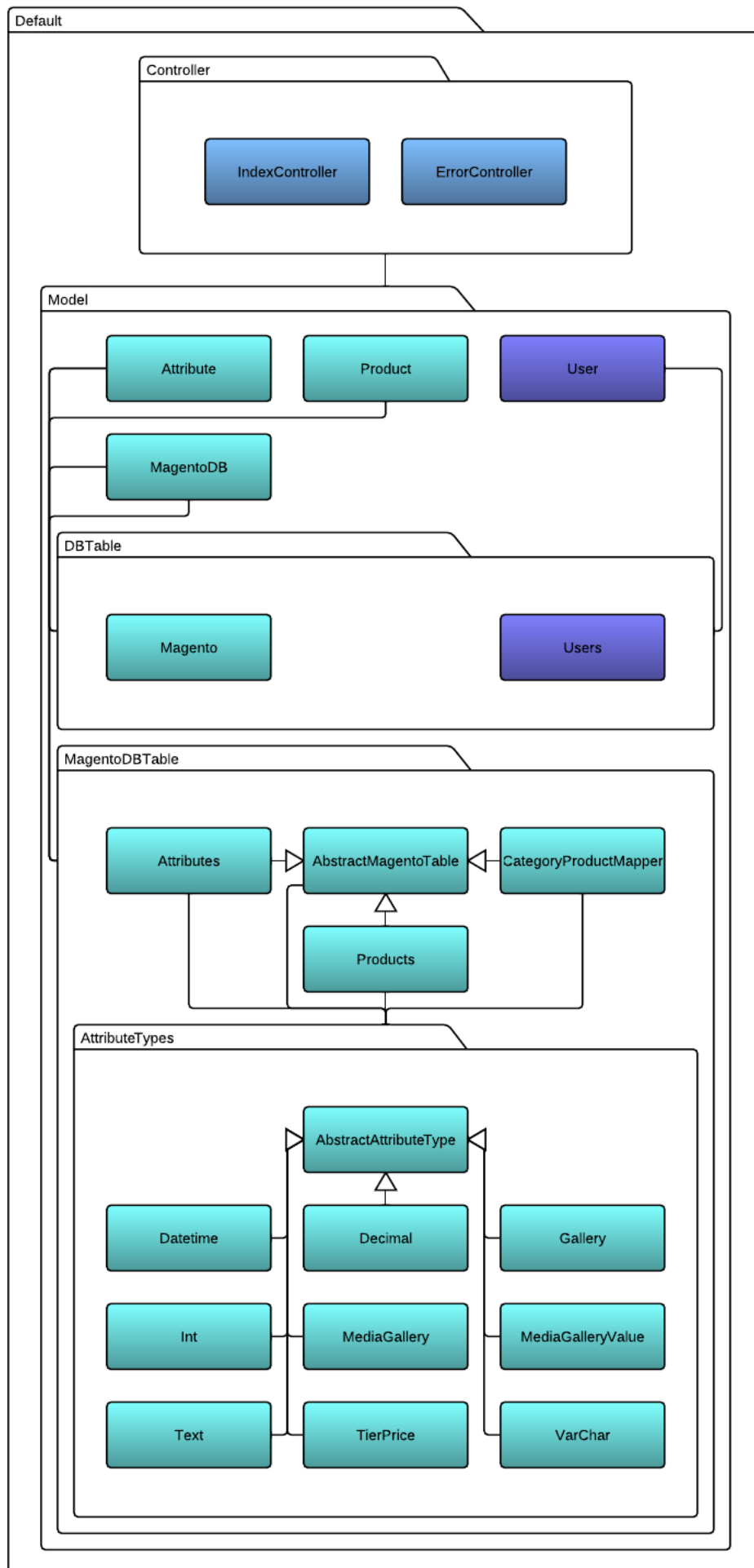
Appendix D: Design Document

Appendix E: Sprint Reports

Appendix F: Class Diagrams







Appendix G: Magento Database

For this project, a short analysis has been done about Magento's Database. This analysis give a more detailed insight in the way data is saved within Magento and is very important once direct communication with the database is desired.

`catalog_category_anc_products_index_idx :`

⇒ This table has nothing interesting.

``catalog_category_product`:`

⇒ This table links the category with the product and gives the position of the product.

``catalog_category_product_index`:`

- ⇒ In this table one can find the store id of the linked category and product. They can be linked to a particular store.
- ⇒ It is also possible to set the position and visibility.

`catalog_category_product_index_enbl_idx,`

``catalog_category_product_index_enbl_tmp`, `catalog_category_product_index_idx, `catalog_category_product_index_tmp``

⇒ These are empty tables and might be used for next versions or prevention.

``catalog_eav_attribute` :`

⇒ In this table you can set the options of an attribute, but this is not interesting for this project.

``catalog_product_bundle_*``

⇒ These tables start with the index for bundled products: products that are composites of other products.

``catalog_product_enabled_index``

⇒ Here, the columns of the product and linked store can be found, as well as the visibility, which can be set.

``catalog_product_entity``

- ⇒ In this table the products are defined. The `entity_id` is the product id. The `entity_type_id` is the type of the product.
- ⇒ Additionally, it can be defined which set of attributes is used for this product.

``catalog_product_entity_datetime`:`

- ⇒ Defines the different attributes, which are date time types. For instance, product is new from date x, or customized design from December until January for Christmas.
- ⇒ Check `eav_attribute` for all attributes and their labels.

``catalog_product_entity_*`:`

- ⇒ Those tables name end with the type of the attribute, just like above.
- ⇒ The int tables are also used for Boolean attributes type.
- ⇒ For some reason the value of visibility is always set to 4.

``catalog_product_entity_tier_price``

- ⇒ This table is for the tiered price: price per group.

``catalog_product_flat_1``

- ⇒ The attributes are flattened here.
- ⇒ This table is probably used for caching.
- ⇒ Check <http://stackoverflow.com/questions/5193150/magento-problem-with-catalog-product-flat> for more information.

`catalog_product_index_eav`

- ⇒ This table probably links the eav value to the store.

`catalog_product_index_*:`

- ⇒ All those tables are tables to link the attributes to the store or website.
- ⇒ Like `catalog_product_index_price` is for linking the price to the customer group and website.

`catalog_product_index_website:`

- ⇒ When it was indexed.

``catalog_product_link_attribute``

- ⇒ The link between attribute and the type: From here a table can be linked.
- ⇒ The table `catalog_product_link` is an empty table but it should be the one that will create the link.

``catalog_product_website``

- ⇒ Products linked to the website.

`eav_attribute`

- ⇒ This table is the most important table for eav. EAV is an architect of Magento that is heavily used.
- ⇒ Read this for example: <http://fishpig.co.uk/blog/magento-database-structure-eav.html>.

LXV

- ⇒ From this table the attribute code, type, isRequired and attribute id can be retrieved.
Entity_type_id = 4 is for product.

eav_attribute_group

- ⇒ Here, the attributes are divided into groups as shown in the backend.

Appendix H: SIG's reply

Beste Moncif Ghezza,

Hierbij ontvang je mijn evaluatie van de door jou opgestuurde code. De evaluatie bevat een aantal aanbevelingen die meegenomen kunnen worden in de laatste fase van het project.

Deze evaluatie heeft als doel om studenten bewuster te maken van de onderhoudbaarheid van hun code en dient niet gebruikt te worden voor andere doeleinden.

Mochten er nog vragen of opmerkingen zijn dan hoor ik dat graag.

Met vriendelijke groet,
Eric Bouwers

[Aanbevelingen]

De code van het systeem scoort net vijf sterren op ons onderhoudbaarheidsmodel, wat betekent dat de code volgens onze score zeer goed onderhoudbaar is. Het enige punt waarop een lagere score wordt behaald is Duplicatie.

Voor Duplicatie wordt er gekeken naar het percentage van de code welke redundant is, oftewel de code die meerdere keren in het systeem voorkomt en in principe verwijderd zou kunnen worden. Vanuit het oogpunt van onderhoudbaarheid is het wenselijk om een laag percentage redundantie te hebben omdat aanpassingen aan deze stukken code doorgaans op meerdere plaatsen moet gebeuren. De duplicatie binnen dit systeem zit hem in de verschillende 'getInstance'-methoden binnen de verschillende sub-classes van 'Default_Model_MagentoDbTable_AttributeTypes_AbstractAttributeType'. Het lijkt er echter op dat dit een limitatie van de gebruikte bibliotheek is, wat het oplossen van deze duplicatie lastig zal maken. Over het algemeen is het echter aan te raden de hoeveelheid duplicatie in de gaten te houden om eventuele ontwerp-problemen te detecteren.

De code van dit systeem scoort dus zeer goed op onze score voor onderhoudbaarheid, hopelijk lukt het om dit niveau te behouden tijdens de rest van de ontwikkelfase. De aanwezigheid van test-code is in ieder geval veelbelovend, hopelijk zal het volume van de test code ook groeien op het moment dat er nieuwe functionaliteit toegevoegd wordt.

Appendix I: SIG's second reply

Beste Moncif Ghezza,

Hierbij ontvang je de resultaten van de hermeting van de door jou opgestuurde code. In de hermeting heb ik met name gekeken naar of/hoe de aanbevelingen van de vorige evaluatie zijn geïmplementeerd. Ook deze hermeting heeft het doel om studenten bewuster te maken van de onderhoudbaarheid van hun code en dient niet gebruikt te worden voor andere doeleinden.

Mochten er nog vragen of opmerkingen zijn dan hoor ik het graag.

Met vriendelijke groet,
Eric Bouwers

[Hermeting]

In de tweede upload zien we dat de omvang van het systeem met 30% is gestegen en dat daarbij de score voor onderhoudbaarheid vrijwel gelijk is gebleven. We zien dat de gerapporteerde duplicatie is verwijderd en dat er geen andere duplicaten zijn toegevoegd. Wat verder opvalt is dat ondanks de groei in code-volume er geen aanpassingen lijken te zijn geweest aan de test-code.

Uit deze observaties kunnen we concluderen dat de aanbevelingen van de vorige evaluatie zijn meegenomen in het ontwikkeltraject. Het is goed om te zien dat, ondanks het niet aanpassen van de test-code, het eerdere niveau van onderhoudbaarheid is behouden tijdens de rest van de ontwikkelfase.