UNIVERSAL ERP-API FOR MAGENTO

TIM-DANIEL M JACOBI

Evaluation of an optimal data-interchange format

Department Informatik
Fakultät für Mathematik, Informatik und Naturwissenschaften
Universität Hamburg

July - September 2012 – Version 4.0

SUPERVISORS:
Dr. Guido Gryczan
Dr. Axel Schmolitzky

LOCATION:
Hamburg

TIME FRAME:
July - September 2012

As soon as questions of will or decision or reason or choice of action arise, human science is at a loss.

— Noam Chomsky

Dedicated to the loving memory of Walter Jacobi.

1919 – 2002

ABSTRACT

The market of electronic commerce has experienced a significant growth since businesses started to sell products online in the late nineties[1]. More products are being sold online every year and the forecast for 2012 again promises an increase in sales. The success of the existing members of the market encourages additional businesses to start selling their products through electronic commerce which strengthens the development even more.

Since electronic commerce is heavily dependent on the software that provides platforms to manage and sell a company's products and and represent its appearance on the internet a reliable and fully functional software foundation is one of the most essential parts when participating in this business. In many cases the software needed for electronic commerce has to be integrated in an already existing software environment the company uses for their enterprise resource planning and selling processes. This integration process can vary from company to company and is often a too less considered problem when starting to act in electronic commerce.

This paper aims for providing and documenting a solution for this problem. It is the result of my occupation as a student trainee at the Lemundo GmbH, Germany that will use the developed software and gained knowledge to serve their clients.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LISTINGS

ACRONYMS

Part I

FUNDAMENTALS

1 INTRODUCTION

1.1 PROBLEM DEFINITION

When a company which plans to enter the market of electronic commerce reaches the phase where a new software component - the electronic commerce platform - needs to be integrated in the current system - most likely an enterprise resource planning platform - problems arise frequently. The aforementioned entities are self-contained software platforms which are designed to support the companies in different areas of their business. But when a company starts to expand its business to the internet, all the product, client and order related data needs to be in sync on both platforms furthermore the data needs to be processed to push them to various electronic commerce platforms like Ebay or Amazon. Hence the platforms need to be able to communicate with each other in order to keep their data in sync. As of today there is no standardised way for those software entities to communicate with each other. Some of them offer an API but are uninformed about the specification of other APIs.

1.2 STRUCTURE OF THIS PAPER

This paper consists of three main parts. The first part covers the subjects it builds up on. It introduces the reader to the problem covered in this paper, the theoretical and architectural requirements needed to solve the problem and the aspired outcome. Furthermore it introduces the reader to fundamental topics that are important to know in order to understand the purpose and mode of planning and the implementation covered in this paper. This is followed by a market analysis to show the reader what is currently available and where the market still lacks of products. The fundamental part closes with a study on an eligible data-interchange format which is an integral part of the resulting software system.

The second part of this paper documents the engineering process of the resulting software system including its environmental and functional requirements as well as the design process. Furthermore the

description of the development process is included in this part which gains the reader an insight of the development environment, the software architecture and the technologies used to implement the resulting software system.

The third part than covers the final matters of the project which include a customer evaluation and a conclusion on the whole project.

1.3 REQUIREMENTS

To solve the problem we need to make sure the enterprise resource planning and electronic commerce software systems are able to communicate with each other. Since we only have limited ability to modify the software itself in most cases and we want to be able to use another product without much effort we need to create an abstraction layer that acts as a connecting entity and is able to communicate with both the enterprise resource planning software and the electronic commerce software.

Furthermore the connector exactly needs to know which data in one entity represents what data in the other in order to be able to maintain steady synchronicity. The connector requires the entities to provide an interface that it can use. Fortunately this is given by the majority of enterprise resource planning and electronic commerce software.

Another important requirement is the ability to aggregate the data from both sources and save them in an appropriate format to push the data to other sales platforms at any given point of time.

1.4 ARCHITECTURAL PREREQUISITES

In order to make the connector software able to communicate with a large variety of enterprise resource planning and electronic commerce software systems it is important that the connector software is easy to extend so that it is able to adapt to the way of communication used by a specific software system it communicates with. Furthermore the software has to be scalable to be able to adjust to growth in business a client might experience while he is running the system.

Since the electronic commerce platform is most likely to run on the Internet it is intended to have the connector software run in the same environment to offer the client an interface to remote control the software's processes, hence it has to be built on a secure platform

in order to keep the clients data private. Moreover the environments performance has to be fast enough and its availability high to guarantee that the software is responsive and almost always available to the client. This is necessary because electronic commerce software currently available operates day and night and thus has to be able to communicate with the enterprise resource planning software at any given point of time.

1.5 OBJECTIVES

The aim of this project is to reflect the current state of the market for ERP solutions and online shops and compare their features relevant for this project. Furthermore a set of data-interchange formats will be examined to evaluate their capabilities regarding the processes between the aforementioned entities.

Based on these findings the objective is to subsequently develop an extensible middleware that acts as a universal connector between ERP solutions and online shops and synchronises all relevant business data between the two entities. Moreover the middleware should be able to aggregate the data from both entities to offer a backup solution and spread product data out to multiple e-commerce channels.

2

FUNDAMENTAL DEFINITIONS

2.1 ENTERPRISE RESOURCE PLANNING SOFTWARE

An Enterprise Resource Planning (ERP) system is an integrated set of programs that provides support for core organisational activities such as manufacturing and logistics, finance and accounting, sales and marketing, and human resources. [2] The purpose of ERP is to facilitate the flow of information between all business functions inside the boundaries of the organisation and manage the connections to outside stakeholders. [3] Nevertheless the functionality varies depending on the vendor and edition of the software. ERP systems can run on a range of computer hardware and networks, typically using a database as a memory for the data that has to be managed. [4] A large number of ERP systems are divided into a set of modules which allows the client to for selection of necessary modules only. Typical modules are manufacturing, human resources, accounting and finance, materials management, customer relationship management, supply chain management or resource management. The most important module in the regard of this paper is the materials management since it handles the product related processes in the ERP system.

2.2 ELECTRONIC COMMERCE SOFTWARE

2.2.1 *Mode of operation*

Electronic commerce software or in the regard of this paper an online shop is a kind of mail order where the communication between business and client is handled mainly over the Internet. The online shop presents the product and handles the process of buying and ordering. Because of its form of presentation - the computer - an online shop serves product information through descriptions, images and/or some other kind of multimedia presentation like video or 3D model. Especially the latter sees more usage lately as Internet lines become faster. The majority of online shops also offer the client to write reviews in order to help other clients selecting the right prod-

uct. Furthermore an online shop might save products the client liked and/or bought and compare this data to the that of other client which is a form of machine learning that enables the online shop to make recommendations to the customer.

2.2.2 *Software foundation*

Almost every website that offers products can be called an online shop. That varies from simple, static HTML pages to highly complex web applications using some kind of application framework and a database to store client, product and order data. Typically the web applications are split into different modules like database, presentation engine, recommendation engine or payment gateway.

2.3 APPLICATION PROGRAMMING INTERFACE

An application programming interface is a part of a software system that offers the system's implemented services to other software systems. Typically the consuming software system sends a request to the API of the serving software system which responds to the request with some kind of data. All APIs examined or used in this paper are protocol oriented which means that they are hardware and software independent. Having the protocol implemented and knowing the API is the only requirement for the consuming software system. Often a protocol API is wrapped into a functional API to ease its usage. Typically functional APIs require to open a handle through which the consuming software system can execute functions on the serving software system.

2.4 DATA-INTERCHANGE

Data interchange is the - mainly standardised - interchange of data between two software systems or entities usually through a data-interchange format. Data interchange between two software systems happens e.g. when a consuming software system sends a request to the serving software systems API. Another use case is the interchange of data between different companies. They use data interchange formats to store and share specifications or protocols between their subsidiaries and branches.

3

MARKET ANALYSIS

3.1 ENTERPRISE RESOURCE PLANNING SOFTWARE

Enterprise Resource Planning (ERP) software is a software product which is used to represent and manage operations and processes in a company [5]. Most ERPs are split into different modules differentiated by scope of responsibility. The most commonly used modules are Accounting, Human Resources, Manufacturing, Materials management, Supply chain management, Project management, Customer relationship management. Not all ERP software systems provide the functionality of all these modules. This paper focuses on solutions that provide at least the functionality of the materials management and accounting modules. The ERP software market is very diverse and fragmented. Top tier vendors include SAP, Oracle and Microsoft followed by tier 2 vendors like Epicor, Sage, Infor etc. and a large variety of tier 3 vendors.[**?**] The tier 1 vendors products share between 70 and 80% of the market but are also the most expensive solutions available furthermore they focus mainly on providing solutions for larger companies which are most likely not to be part of the client base of Lemundo.[6]

Together with the company Lemundo a set of ERP solutions was selected that might come into consideration with their clients. These solutions were subject to examination and evaluation of their APIs and capabilities to communicate with other software like the resulting application this paper is about. Obtaining detailed information was often difficult as many ERP software vendors reveal details on their software only reluctantly. Unfortunately trail versions for academic purposes were not provided at all. Furthermore it is almost impossible to create a reference setup with some of the ERP solutions examined since they are build to reflect certain business processes and therefore need to be customised which is not the scope of this paper.

3.1.1 *Sage ERP X3*

Sage ERP X3 provides a web-based ERP solution for medium-sized companies [7]. It contains modules for manufacturing, finance, sales, customer relationship management, purchasing and inventory. [8] The main focus of X3 is its international orientation. It comes with support for nine languages and legislations which is intended to be supportive for both users and clients. [9]

The software architecture consists of four modules. [10]

- Sage ERP X3 Application Server
  - available for Windows, Unix and Linux
- Web-server (Apache)
- Data-Server
  - based on MySQL or Oracle
- Web application

3.1.2 *Microsoft Dynamics NAV*

Microsoft Dynamics NAV provides an ERP solution for small and medium-sized companies. It contains modules for analytics, customer-relationship management, electronic commerce, finance, manufacturing and supply chain management.[11][12] Dynamics NAV is customisable through its internal programming language *C/AL (Client/Server Application Language)* used by the *C/SIDE (Client/Server Development Environment)*.

Dynamics NAV has a built-in SOAP module but Microsoft distributors describe it as buggy and reported that it is not able to cover business logic in an appropriate way. The recommendation given was to develop a new Dynamics NAV module which offers the SOAP functionality. Other options would be to create a module which provides any other kind of connecting possibility like XML-RPC or CSV import/export.

3.1.3 *OpenERP*

OpenERP provides an ERP solution for small and medium-sized companies. It contains modules for Accounting, Asset management, Customer relationship management, Human resource management, Man-

ufacturing, Point of sale, Project management, Purchasing and Sales management. OpenERP architecture consists of a client and server which communicate through XML-RPC. It supports a variety of APIs using EDI, XML-RPC, SOAP, CSV and is extensible through modules that are written in Python and XML[13].

3.1.4 *JTL*

JTL is a free ERP solution for administrative tasks in small businesses. It provides modules for orders, bills, reminders, letters, products and accounts.[14] JTL offers import and export of product and client data via CSV and order data via XML. [15] It is not possible to extend JTL with self-written modules.

3.1.5 *Faktura-XP*

Faktura-XP is an ERP solution for administrative tasks in small businesses. It provides modules for material management, order management and point of sale. The software is not extensible but provides a variety of data interchange APIs including XML-RPC, XML and WSDL via DOM, SOAP and SAX [16].

3.1.6 *Lexware Warenwirtschaft Pro*

Lexware Warenwirtschaft Pro is an ERP solution for small businesses. It provides module for the management of clients, supply chain, orders, bills and warehouse. Unfortunately Lexware Warenwirtschaft Pro does not provide any APIs for data interchange and hence will not be working with the connector system. [17]

3.1.7 *Siller Intersys*

Siller Intersys is a ERP solution which is optimised for INTERSPORT partners. It provides modules for materials management, product management and analysis, limit planing and control as well as a point of sale module. [18] Siller Intersys offers an SFTP API but unfortunately does not reveal what its capable of.

3.1.8 *Overview*

The following table sums up the important facts of the ERP solutions examined.

3.2 ELECTRONIC COMMERCE SOFTWARE

Electronic commerce software is a software product which is used to present the companies products to the client and handle the process of ordering and buying. The Electronic commerce software market is very diverse and fragmented. Top tier solutions include Magento (24.8%), Zencart (16.8%), VirtueMart (15.2%) and osCommerce (10.1%). The remaining third of the market is shared by about 10 smaller solutions and modifications of the top tier shops.[19]

The e-commerce solutions offered by Lemundo are all based on Magento but yet a set of Shop Systems was selected to ensure comparability. These solutions were subject to examination and evaluation of their APIs and capabilities to communicate with other software like the resulting application this paper is about. Obtaining detailed information was only difficult with solutions you have to purchase as many Shop software vendors reveal details on their software only reluctantly and their website often do only offer a superficial insight on the abilities of the software offered. Nevertheless it was easy to examine the open source solutions since they are available to download for free.

3.2.1 *Gambio GX2*

Gambio GX2 is an electronic commerce solution distributed by the German software company Gambio. Assumingly it aims at small to medium-sized companies. Among others it includes modules for bill and delivery note generation, special offers, newsletter, security, voucher management, warehouse management and advertisement. [20] Gambios approach is to provide some of the functionality of ERP software which leads to the assumption that it is not meant to support large-sized companies. Unfortunately this approach includes that Gambio does not provide any kind of API and is therefore not able to communicate with the connector system.

3.2.2 *Magento*

Magento is an open source electronic commerce solution distributed by the US based company Magento Inc. Magento is the current market leader for electronic commerce software. [19] Its scalability allows for its usage in small to large-sized companies. Magento comes in three editions include Enterprise Edition, Professional Edition and Community Edition. The latter is for free and therefore the mainly used edition. Magento provides an almost unlimited number of modules for any imaginable purpose through its module marketplace MagentoConnect. To communicate with other software systems Magento provides SOAP and XML-RPC APIs [21] moreover it enables one to write a module for this task which accesses the Magento Core API. Therefore Magento will be able to communicate with the connector system in multiple ways enabling the user to select the API that works best.

3.2.3 *osCommerce*

osCommerce is an open source electronic commerce solution. osCommerce provides an almost unlimited number of modules for any imaginable purpose through its module repository. Since the website does not reveal any information about APIs provided osCommerce was installed during the research. Unfortunately osCommerce does not offer any APIs by default and will therefore without modification not be able to communicate with the connector system.

3.2.4 *Oxid eShop*

Oxid eShop is a proprietary electronic commerce solution which is available in three different editions including Enterprise, Professional and Community Edition. Therefore it provides an appropriate solution for small to large-sized companies. At the time of writing this paper Oxid is on the threshold of version 4.7 and 5.0 announcing profound changes like caching or master/slave architecture planned for version 5.0. [22] Oxid provides standard features like SEO and role management et cetera by default and is extensible through modules. [23] A third party module which provides a SOAP API is available [24] which enables Oxid to communicate with the connector system.

3.2.5 *PrestaShop*

PrestaShop is an open-source electronic commerce solution developed by the french company PrestaShop SAS. It assumingly provides a solution for small to medium-sized companies. By default PrestaShop provides standard functionality but is extensible through modules [25] including a SOAP API module. [26] Therefore it is able to communicate with the connector system.

3.2.6 *VirtueMart*

VirtueMart is an open-source electronic commerce module for the content management system Joomla! It provides standard features by default and is extensible through modules including a module for SOAP support [**?**] which enables VirtueMart to communicate with the connector system.

3.2.7 *xt:Commerce*

xt:Commerce is a proprietary electronic commerce solution which comes in three editions including Professional, Professional+ and Multishop. It provides a standard feature-set by default [27] and is extensible through modules including a module for SOAP support [28] which enables xt:Commerce to communicate with the connector system.

3.3 CONNECTORS

A connector is a software extension which enables the software to communicate with another specific software system like another Enterprise Resource Planning Software or Online Shop. Most connectors only work for one specific Enterprise Resource Planning Software enabling it to communicate with one or more Electronic commerce software systems. This section lists a subset of the connectors currently available to prove that fact.

3.3.1 *Open ERP Connector*

This connector enables Magento to communicate with Open ERP. [29]

3.3.2 *Epages for ERP*

This connector enables a set of ERP Systems to communicate with the Epages Electronic Commerce software.[**?**]

3.3.3 *JTL Connector*

JTL offers a set of connectors to enable the JTL ERP to communicate with various Electronic Commerce software systems including osCommerce, xt:Commerce, Zen Cart, VirtueMart. [30]

3.3.4 *Siller Webshop-Schnittstelle*

Siller includes the ability to communicate with Electronic commerce systems in their Intersys ERP. Unfortunately they do not reveal which products they support. [18]

3.3.5 *OpenERP Prestashop sync*

This project aims to provide connectivity between OpenERP and Prestashop for synchronisation of Products, Categories, Customers, Sale Orders, Taxes, Payments and much more. [31]

3.4 SYSTEMS OVERVIEW

3.5 CONCLUSION

The extensive research on the ERP and electronic commerce software markets confirms the introductory assumption that the market is diverse and fragmented. While we find a significant differentiation between different tiers in the ERP software market the electronic commerce software market is even more fragmented below the top tiers Magento, zencart and VirtueMart. Most electronic commerce software systems are extensible through modules which enables them to support APIs like SOAP or XML-RPC. The ERP software systems on the other hand tend to be less extensible or harder to extend as some of them require you to use their internal programming language. Nevertheless the market offers some extensions for various ERP software systems to enable them to exchange their data with a specific set of

SOFTWARE	APIS
OpenERP	EDI, XML-RPC, SOAP, CSV
Sage ERP X3	XML, UDDI, SOAP
Microsoft Dynamics NAV	none by default
JTL	CSV (products, clients), XML (orders)
Faktura XP	XML-RPC, SOAP, SAX
Lexware	none
Siller Intersys	proprietary

Table 1: ERP systems overview

SOFTWARE	APIS
Gambio	none
Magento	SOAP, XML-RPC, internal
osCommerce	none
Oxid	SOAP with extension
PrestaShop	SOAP with extension
VirtueMart	SOAP with extension
xt:Commerce	SOAP with extension

Table 2: Electronic commerce systems overview

online shops yet the sets are restricted and might not cover an individual combination of ERP and electronic commerce system.

Even though there are three top tiers on the ERP market they only offer solutions for larger companies that are not part of the client base of Lemundo. Therefore it is needed to concentrate on the fragmented part of the market which underlines the need for an universal connector since we have to deal with even more different software systems.

Part II

IMPLEMENTATION

While theory is important, practice is the key, but whats the point on every graph, one really wants to be?

4

REQUIREMENTS

4.1 SYSTEM REQUIREMENTS

In order to keep the cost for running the connector system low it is required that the system runs along one of the clients existing software solutions. Since all electronic commerce software runs on a web-server regardless of which solution it might be the connector system shall run on a web-server as well and hence shall be a web application.

4.2 LOOK AND FEEL REQUIREMENTS

The user interface shall adhere to the requirement for the connector system to be a web application. Thus it shall be an interface which runs in the browser but still shall offer the look and feel of a responsive user interface.

4.3 FUNCTIONAL REQUIREMENTS

4.3.1 *Plugins*

In order to be able to communicate with third party electronic commerce or enterprise resource planning systems the connector system shall be extensible through plugins. A Plugin shall be capable of obtaining data from the third party systems and converting it into the internal data representation used by the connector system. Furthermore it shall be capable of using data occurring in the connector system, converting it into the format of the third party systems format and providing it to the them. Plugins shall be placed in a separate folder and shall be recognised by the connector system and made available in the user interface.

4.3.2 *Attribute mapping*

Any kind of entity residing in the third party system is assumed to be saved in a format that includes the following specifications.

- The entities given name
- The information when the entity got updated at last
- A list of attributes the entity possesses

The attribute mapping shall provide a way to synchronise the values of the attributes between a plugin and the connector system. Therefore an attribute mapping shall consist of the following parts.

- An attribute name from one of the plugins
- An attribute name specified by the user to be used by the connector system
- A mapping direction

Once the mapping is set the selected attribute shall be synchronised with the connector system depending on the chosen mapping direction. To enable the user to control the flow of data there shall be three different mapping directions which shall cause the following effects.

- Update third party system only
  - Changes in the connector system attribute shall be forwarded to the system connected through the plugin
  - Changes in the system connected through the plugin shall *not* be forwarded to connector system
- Update connector system only
  - Changes in the system connected through the plugin shall be forwarded to connector system
  - Changes in the connector system attribute shall *not* be forwarded to the system connected through the plugin
- Update both
  - Changes in the system connected through the plugin shall be forwarded to connector system
  - Changes in the connector system attribute shall be forwarded to the system connected through the plugin

4.3.3 *Data aggregation*

As described in the previous section a set of attribute values is supposed to be synchronised with the connector system. The use of attribute mappings shall allow for synchronisation of all attribute values with the connector system even of those that are not meant to be forwarded to other third party systems. The connector system shall store all attribute values from all plugins for one entity in the same record. This is where the attribute values shall be aggregated.

4.3.4 *Backup*

The Backup functionality in the connector system shall provide a way to save all aggregated data currently residing in the system. The backups shall be save in a specific folder. Furthermore it shall be possible to load a saved backup into the system.

5

ELIGIBLE DATA-INTERCHANGE FORMATS

In order to backup the data occurring in the system it is needed to identify an appropriate data format. Since the environment the connector system runs on uses PHP as programming language one of the requirements for the data interchange format is that at least one implementation for PHP exists. Moreover the reading and writing speeds of these implementation are important since we are likely to deal with a lot of data. Another important detail to take into account is the storage footprint each file-format causes and how the reading and writing speeds differ when the amount of data to be saved increases.

Four common file formats that were assessed to perform well with regard to the aforementioned matters were examined and analysed. JSON, SQLite, XML and YAML.

5.1 JSON

JSON stands for JavaScript Object Notation. This data-interchange format was specified and introduced by Douglas Crockford and is as the name suggests based on the JavaScript syntax which it seems to conform to but does not.[32] Fortunately JSON has a pretty small footprint and is supported by a large variety of programming languages. [33] There are four implementations for using JSON in PHP which are provided by PEAR, ZEND etc. Yet benchmarks have proven that the native PHP implementation is the best choice. [34]

5.2 SQLITE

SQLite is an embedded SQL database engine. Unlike most other SQL databases, SQLite does not have a separate server process. SQLite reads and writes directly to ordinary disk files.[35] Thus it is portable and can be considered a data-interchange format. SQLite is implemented in C and runs as a PHP extension [36]

5.3 XML

The eXtensible Markup Language is the longest standing data-interchange format among the ones tested in this paper. It was derived from the industry standard SGML and aims at simplification of and concurrently compatibility with SGML as well as usability over the internet.[37] Not only is XML a data-interchange format but also a meta language to specify markup languages. Many well-known standards like RSS, MathML, XHTML or WSDL were designed on the XML foundation. There are XML parsers for a large variety of languages including PHP which implements it as an extension as well. The PHP extension is split into XMLReader and XMLWriter. Even though the XMLReader itself already provides a way to parse XML documents there are other options that have to be evaluated since they might come in more convenient for the discussed purpose.

5.3.1 *XMLReader only*

The XMLReader acts as a cursor going forward on the document stream and stopping at each node on the way. [38] To its advantage XMLReader runs fast and does not consume much memory. Nevertheless writing and debugging is complicated since a lot of code is needed for a sparse result. The more code you use the more code you need to maintain and the higher is the risk to produce an error.

5.3.2 *XMLReader + SimpleXML*

The SimpleXML extension provides a toolset to convert XML to an object that can be processed with normal property selectors and array iterators. [39] SimpleXML provides an API which is easier to use than XMLReader furthermore it does not use much memory either since it only needs the memory to create one node. The downfall is that you need to create one SimpleXML object per node. Since the documents used with the connector system are likely to consist of thousands of nodes this could be an issue.

5.3.3 *XMLReader + DOM*

The Document Object Model is an API to interact with XML an HTML documents. It represents the analysed document as a tree of objects

that one can iterate over. DOM uses about as much memory as SimpleXML, its syntax requires more understanding and effort compared to SimpleXML but yet it is not as complicated to use as XMLReader. It's simplicity lies halfway between XMLReader and SimpleXML.

5.3.4 *YAML*

YAML stands for Yet Another Markup Language, its structure is derived from Perl. YAMLs fundamental structures include scalars, maps and lists. It uses whitespace characters to denote the relation between the data it contains.[40] There are not many native implementations for YAML there is especially none for PHP but one can use the Spyc Yaml Class.[41]

5.4 BENCHMARKS

5.4.1 *Environment*

To test these file formats the EntityCollection, EntityObject and EntityAttribute containers described earlier were used.

A script for each file format was created to measure their reading and writing speeds. To produce a result that is as accurate as possible in order to gain a better comparability the scripts run each specific task 10000 times. Additionally the scripts were run 10 times each and the gross average was calculated. To make sure my test system is not biased by any circumstances two other systems were employed where exactly the same tasks were run on. That way the results were related to each other to discover possible biases.

The first test system was a MacBook with an Intel®Core™2 Duo 2.1 Ghz processor, 4 GB memory and running Mac OS X Lion. The second test system was a Root Server hosted at Host Europe. It had four vCores instantiated from AMD CPUs, 16 GB memory and running Ubuntu 12.04 Server. The third system was an Amazon EC2 virtual server t1.micro type running with one virtual core, 615 MB memory and Ubuntu 12.04 Server as operating system. The Amazon t1.micro servers specifically have bad I/O performance but are still employable for the comparison.[42]

5.4.2 *Result analysis*

The results show that JSON is the fastest data-interchange format for both reading and writing. While XML placed second in writing it placed only third in reading following SQLite whose process time is outside the measurable range when it comes to writing. YAML had the worst measurable result over all benchmarks which makes it an inconsiderable data-interchange format.

5.4.3 *Conclusion*

The aforementioned results leave us only with JSON as data-interchange format to recommend in combination with the system the formats were tested on. Alternatively one could use XML if JSON is unwanted. As the test environment is almost similar to the environment the connector system will run on, JSON we be used as its standard format for data-interchange and backup purposes.

6

DESIGN

6.1 SOFTWARE DESIGN

After the requirements were gathered the system design process was initiated. The process went through multiple stages. Starting with basic sketches to gain a first overview on how the system could be structured the design was then lead over to a more graphical representation that delimited the modules in the design more precisely which was followed by a formal design using UML diagrams and Entity Relationship models which formed the foundation for the development process. During the whole process the design was iteratively evaluated and revised to ensure satisfaction of the requirements.

6.1.1 *Sketches*

Since creativity and change are important while creating the first outline of a software system, sketches were chosen as a first design step because they are easy to overview, change and reassemble. Using software in this phase of the design process might limit ones freedom of thought and creativity caused by limitations the software might entail. This phase of the design process lasted about four days and included constant consultation with the client to ensure the requirements gathered were correctly understood.

6.1.2 *Graphical representation*

The following phase of the design process included the creation of a graphical representation of the design achieved in the first phase. This intermediate step was chosen to come up with a representation of the software design that is presentable and shows all important components of the software and how they are planned to be layered. Yet it does not contain to much information on the technical aspects of the design in order not to confuse the client.

6.1.3 *UML Class Diagram*

In order to push the design process more towards a formal and standards-compliant specification a UML Class Diagram was created, reviewed and revised carefully since the UML diagram formed the foundation for the development process. The creation of the UML diagram lead to a stricter specification of the software system since it exactly depicts how the different parts of the system are supposed to work together in the final product. This helped to recognise parts of the system that were not cohesive enough or coupled to tightly and resulted in another revision of the software design which then was used to initiate the development process.

6.2 DATABASE DESIGN

To be able to save the data occurring in the system a database was designed that offers the necessary infrastructure the system has to run on. The data-structure is flat and occupies one table per occurring entity. Prior to the creation of the database a Entity Relationship model was created to depict and formalise the layout and connections of the database. The creation of the Entity Relationship Model was very straightforward since the implementation does not require a complicated database layout.

7

DEVELOPMENT

7.1 DEVELOPMENT ENVIRONMENT

The connector system was developed with the following hardware/-software setup.

- Mac Book (early 2008), 2.1 GHz Intel Core 2 Duo, 4GB 667 MHz DDR2 SDRAM
- NetBeans PHP 7.1.2
- PHPUnit
- MAMP 2.0 Stack including
  - PHP 5.3
  - Apache 2.2.22
  - PHPmyAdmin 3.5.2.2
  - MySQL 5.5.25
- Google Chrome 22
- ArgoUML 0.34
- PHPUnit 3.6.11
- MySQL Workbench 5.2.33
- Adobe InDesign CS4
- Adobe Illustrator CS4

7.2 SOFTWARE DEVELOPMENT PROCESS

Due to the lack of time and the fact that academically developed software usually is more of a prototypical kind the software was developed in the manner of the rapid application development methodology (RAD). This allows for a faster creation of a prototype that might not deliver the optimal performance but is fully functional which was intended.

After the requirements were gathered and the software was designed the development process was initiated immediately. Requirements and design were not revised but verified during the development process.

The process was characterised by the object-oriented programming paradigm with concurrent use of the test-driven-development approach. Prior to the implementation a set of tests for the classes to design was specified using the PHPUnit testing framework.

The classes then were implemented iteratively among constant reruns of the specified tests. After all specified tests for all classes were passed the modules were expected to run properly.

Yet some classes were not unit-tested during development as they included database related code and therefore were not testable within an appropriate amount of time. Though the testing of those classes was not canceled but rather postponed until the system tests.

7.2.1 *System tests*

To ensure that the system does not only pass the unit-tests but also is able to work in productive operation system tests were defined through the specification of test-cases with parameters and results to be expected.

In order to be able to test the system two dummy plugins were introduced which simulate the behaviour of an ERP and Electronic commerce system respectively. For an easier understanding of the test cases the plugins will be called *Plugin A* and *Plugin B* in the following paragraph.

The following list displays a subset of the system test-cases that were run. Some test-cases are not displayed since they do not represent a considerable variation of the test-cases shown in the list.

- Test-case 1

  - set mapping between Plugin A and connector system to "update both"
  - set mapping between Plugin B and connector system to "update both"
  - change product with ID x in Plugin A
  - expect product with ID x in Plugin B to be updated
  - run synchronisation
  - check if product with ID x was *successfully updated* in Plugin B

- Test-case 2
  - set mapping between Plugin A and connector system to "update both"
  - set mapping between Plugin B and connector system to "update connector system only"
  - change product with ID x in Plugin A
  - expect product with ID x in Plugin B not to be updated
  - run synchronisation
  - check if product with ID x was *not updated* in Plugin B

- Test-case 3
  - set mapping between Plugin A and connector system to "update third party system only"
  - change product with ID x in Plugin A
  - run synchronisation
  - expect product with ID x in connector system not to be updated
  - check if product with ID x was *not updated* in connector system
  - change product with ID x in Plugin B
  - expect product with ID x in Plugin A not to be updated
  - run synchronisation
  - check if product with ID x was *not updated* in Plugin A

- Test-case 4
  - set mapping between Plugin A and connector system to "update both"

- set mapping between Plugin B and connector system to "update both"
- change product with ID x in Plugin A
- change product with ID x in Plugin B
- expect product with ID x in Plugin A to contain the same data like the corresponding product in Plugin B
- run synchronisation
- check if product with ID x was *successfully updated* in Plugin A

7.3 ARCHITECTURE

The architecture follows the three-tier organisation consisting of backend, front-end and database which is typical for modern web applications.

7.3.1 *Backend*

The systems backend contains the business logic and is responsible for services such as database interaction, file handling, processing user input and registering plugins. All services related to the management of the system like plugin activation and deactivation, backup loading and saving as well as the CRUD [1] functions for the attribute mapping management are exposed to its users through a RESTful API [2]. The responses the API generates are encoded in JSON to make them processable for a large variety of client languages that might communicate with the backend. JSON and not XML was chosen for the communication because "JSON overtook XML as the primary data format in Web APIs"[43] the reason for this "might be the inherent impedance mismatch between XML and object oriented programming constructs".[43]

7.3.2 *Frontend*

The front-end is provided as web application and therefore runs exclusively in a browser. It provides all user interface elements needed

1 Create, read, update, delete
2 Representational State Transfer is a kind of architecture for distributed systems

to control the services the backend offers. Furthermore it consumes the backends RESTful API to communicate with it.

7.3.3 *Interfaces*

- Plugin
  - This interface contains a set of methods a plugin must implement in order to be able to integrate neatly in the system
  - The methods are primarily responsible for the interaction with the API of the third party system
  - They are supposed to provide algorithms for the conversion of the third party systems data format to MetaConnects data format and vice versa

7.3.4 *Entities*

The following entity classes were introduced to process the emerging data internally

- EntityObject
  - Objects of this Class represent any kind of Entity that may reside in the systems connected through the plugins
  - It contains the Entities name and id, a list of attributes and the time of the last update
- EntityAttribute
  - Contains the attributes name and its value
- EntityCollection
  - Provides the functionality of a container that contains EntityObjects
- MappingTable
  - Contains attribute mappings
- PluginCollection
  - Provides the functionality of a container for plugins in order to pass them to a Service

7.3.5 *Services*

The following classes have been introduced to provide the systems services described earlier.

- Aggregator
  - Stores and loads aggregated data in and from the database
- BackupManager
  - Stores and loads aggregated data to and from a file
- MappingManager
  - Stores and loads attribute mappings to and from the database
- PluginManager
  - Checks for new plugins and registers them
  - Activates and deactivates them
- Synchroniser
  - Obtains the data from the Aggregator and the plugins and forwards it to the corresponding module.

The systems front-end is responsible for presenting the current state of the system and dealing with user input that changes this state. It offers UI elements that allow the user to use the services the backend offers.

7.4 TECHNOLOGIES USED

In the past years modern web-development has seen a lot of frameworks coming up for all kinds of purposes. Nowadays frameworks are widely used among all popular websites. [44] Since frameworks help to build more secure, modular, extensible, reliable and agile software heavy use of frameworks was made throughout the whole project.

7.4.1 *Laravel PHP framework*

The Laravel framework was chosen because it gained momentum in a quite a short amount of time. Active users confirmed the well though out architecture of the framework and showed a lot of enthusiasm claiming that the build in functionality sped up their development

limitless. Moreover Laravel seemed to be well dimensioned for the project size while the well-known frameworks CodeIgniter and Zend seemed to be an overkill for the project.

Laravel offers and is therefore partially responsible for the following purposes.

- MVC design pattern
  - not applicable for the system described since we only provide a RESTful interface through Laravel
- Eloquent object relationship mapper
- Database migrations
- Routing
- Error handling and logging
- Security precaution mechanisms
- Unit-testing (see following section)
- Class autoloading

Laravel provides controllers that allow for the simple creation of the RESTful interface described above since they offer actions for all HTTP methods. [45]

7.4.2 *PHPUnit testing framework*

The PHPUnit testing framework was chosen because it is - while being also available independently - a part of Laravel and therefore works perfectly together with it. It enables its users to write unit-tests in PHP like in any other unit-test framework like JUnit et al.

7.4.3 *CoffeScript scripting language*

"CoffeeScript is a little language that compiles down to JavaScript. The syntax is inspired by Ruby and Python, and implements many features from those two languages." [46] Usually CoffeeScript code uses about 2/3 of the code compared to JavaScript. [47] It was chosen in order to save time while getting the hands on a relatively new invention. Furthermore the support of classes made its use an advantage in conjunction with the chosen JavaScript framework.

7.4.4 *backbone.js JavaScript framework*

Backbone.js is a simple and lightweight but flexible framework for building JavaScript dependent web applications. Providing model, controller and view classes Backbone.js was build with the MVC design pattern in mind. It supports synchronisation between the web application and a server through a RESTful interface. [48] [49] The market of JavaScript frameworks is very diverse and fragmented offering a wide range of frameworks that use a pattern similar to MVC. Yet Backbone.js was chosen due to its vast developer-base. Furthermore it seems to be the most sophisticated JavaScript MVC framework on the market.

7.4.5 *Twitter Bootstrap CSS framework*

Twitter Bootstrap is a CSS framework that provides design-templates for standard HTML Elements such as fonts, forms, buttons, navigation etc. Furthermore it supports responsive design and uses HTML5 and CSS3 in supported Browsers. Bootstraps acts as the design basis of the project to provide a robust and functional user interface and experience. It was chosen for the simple reason of being the most famous CSS framework on the market according to GitHub. [**?**]

Part III

FINAL MATTERS

8

POST PROJECT

8.1 CUSTOMER EVALUATION

"Within our company we faced the problem that there exist several different shop systems and several different ERP systems which often need to be synchronized. This creates a lot of work, since similar work has to be done again and again. Therefore, the goal of Tim's work was to create a system which can potentially synchronize shop systems on the one hand and ERP systems (and potentially other systems like eBay, Amazon or Google) on the other hand.

He took the approach to build the system in PHP using the Laravel framework and Backbone.js as a frontend framework. He did not have any prior experience in any of these frameworks. Especially the usage of the frontend framework was a challenge since he had not used any comparable system before.

The result of his work is a prototype which is able to connect any shop and any ERP given a so-called plugin is written for the two systems. Consequently, there is still a bit of work needed for each new project, but this was already apparent from the beginning. His design allows a very nice separation of concerns, because the two plugins can be developed independently, i.e. for example by a shop specialist and by an ERP specialist.

Given the time of the project and the resulting product, we are very happy with Tim's work. He did an excellent job in working independently while communicating and discussing the essential parts of the system. Tim had a hands-on approach, divided his time to pick the most relevant work items first and managed to quickly master two new frameworks. He was always enthusiastic, willing to learn new things and very open to accept criticism or suggestions. It was a pleasure to have him in our team and we will certainly miss him when he is in London. We wish him all the best for his future and hope that we will stay in contact."

8.2 CONCLUSION

The conduction of this project has shown how diverse software can be on one hand but still connectable on the other. It has proven that

software is as diverse as the humans are in this world who all come up with different ideas and ideologies of how things should go their way but still share a common foundation. While the inequality and inconsistency in software this fact causes can be quite disillusioning it formed the challenge of this project.

The creation of software was what drove me during me whole undergraduate studies. It is not without good reason that I have chosen a software project as the topic of the thesis that will be the key to my advancement in the field of computer science. Whilst only being able to get my hands on smaller scaled problems during the time being an undergraduate student this final software project including all its facets was my most comprehensive challenge yet. My aim was to prove myself that I was able to assemble all parts that belong to a project like this and come up with something that is not only useful to others but also presentable and something that I can be proud of. After three month full of intensive research, design, development, testing and writing I am very sure that I have achieved what I was aiming for - technically and also personally.

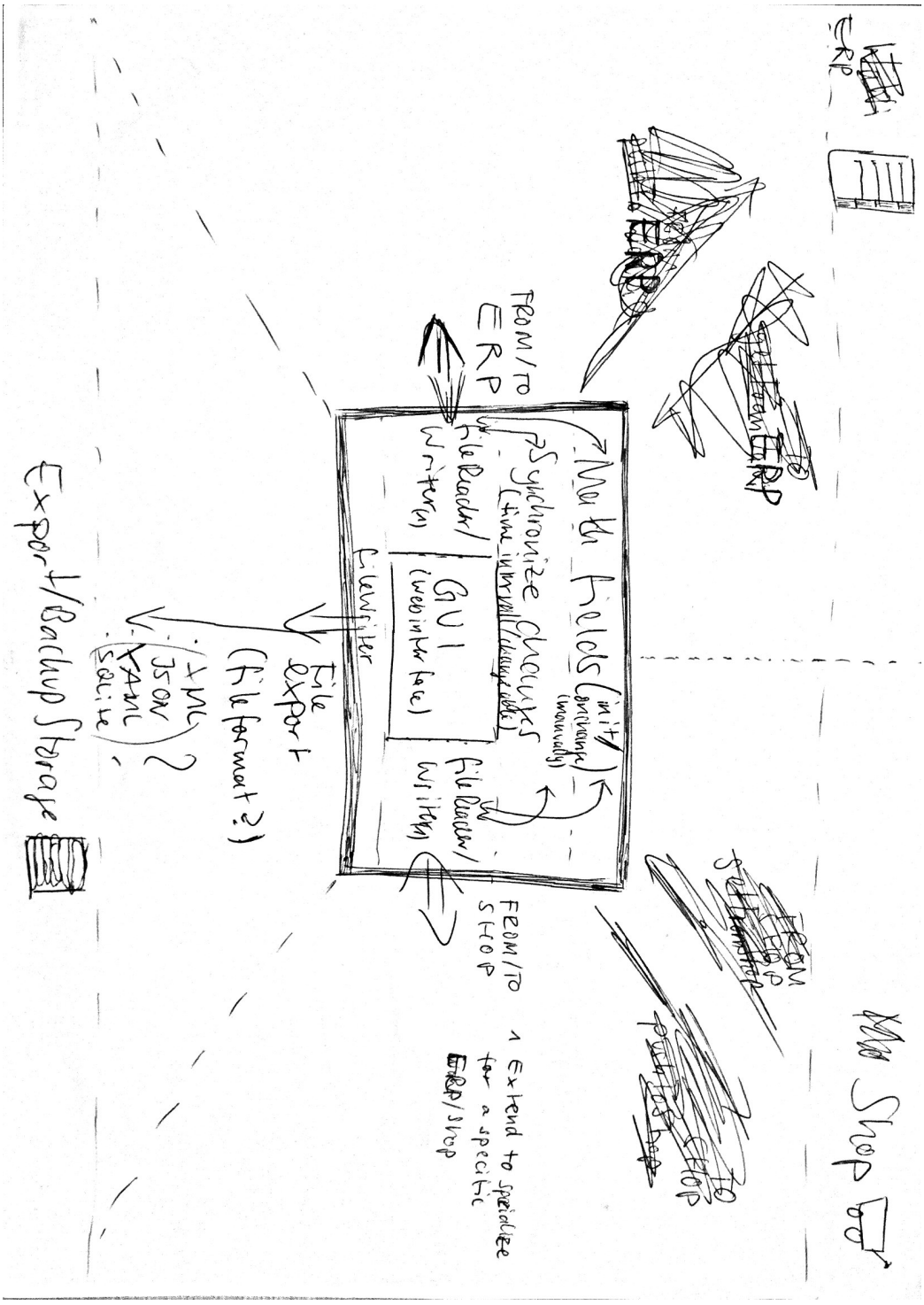
Part IV

APPENDIX

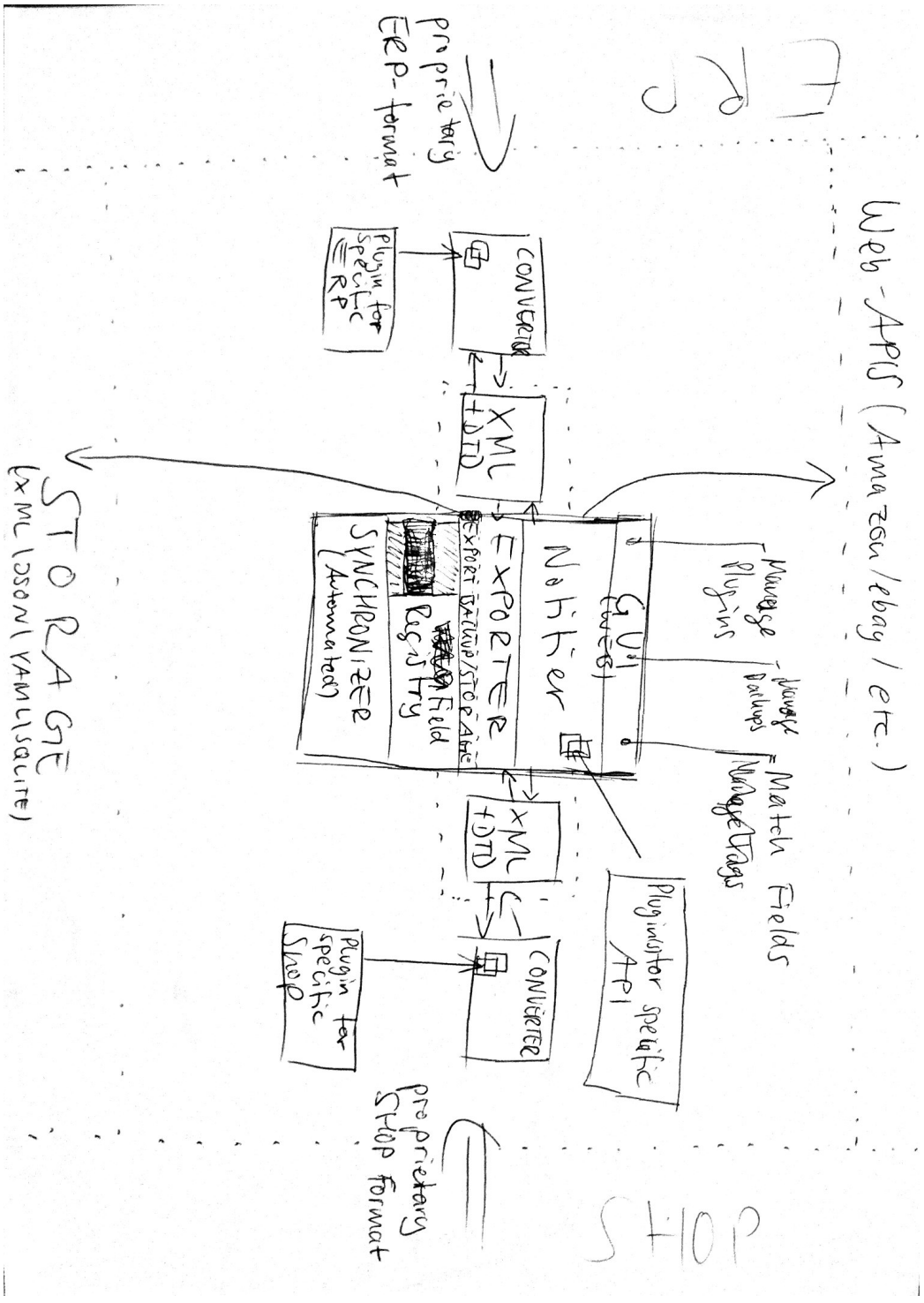
A

SYSTEMS DESIGN LAYOUTS

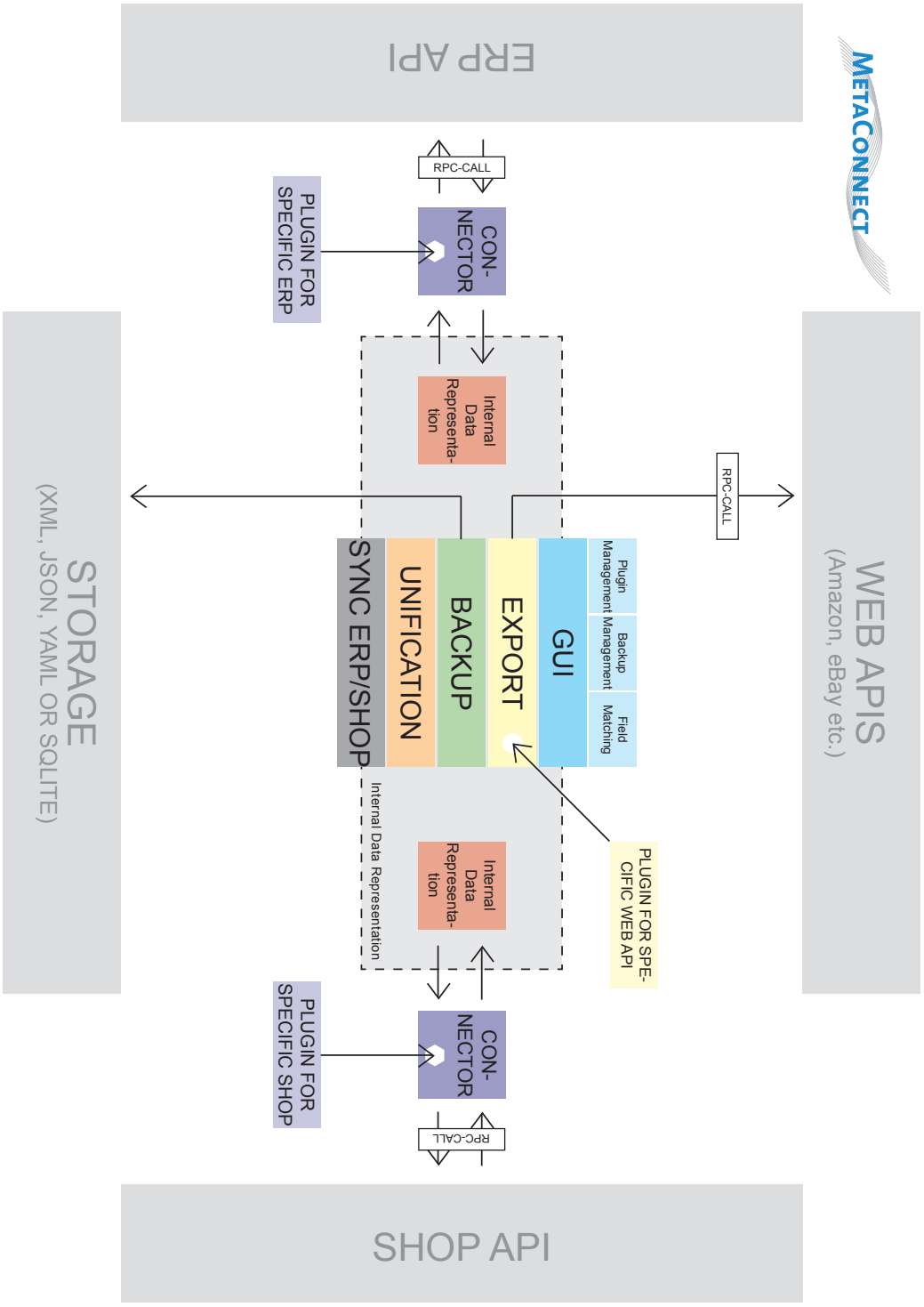
A.1 FIRST SKETCH



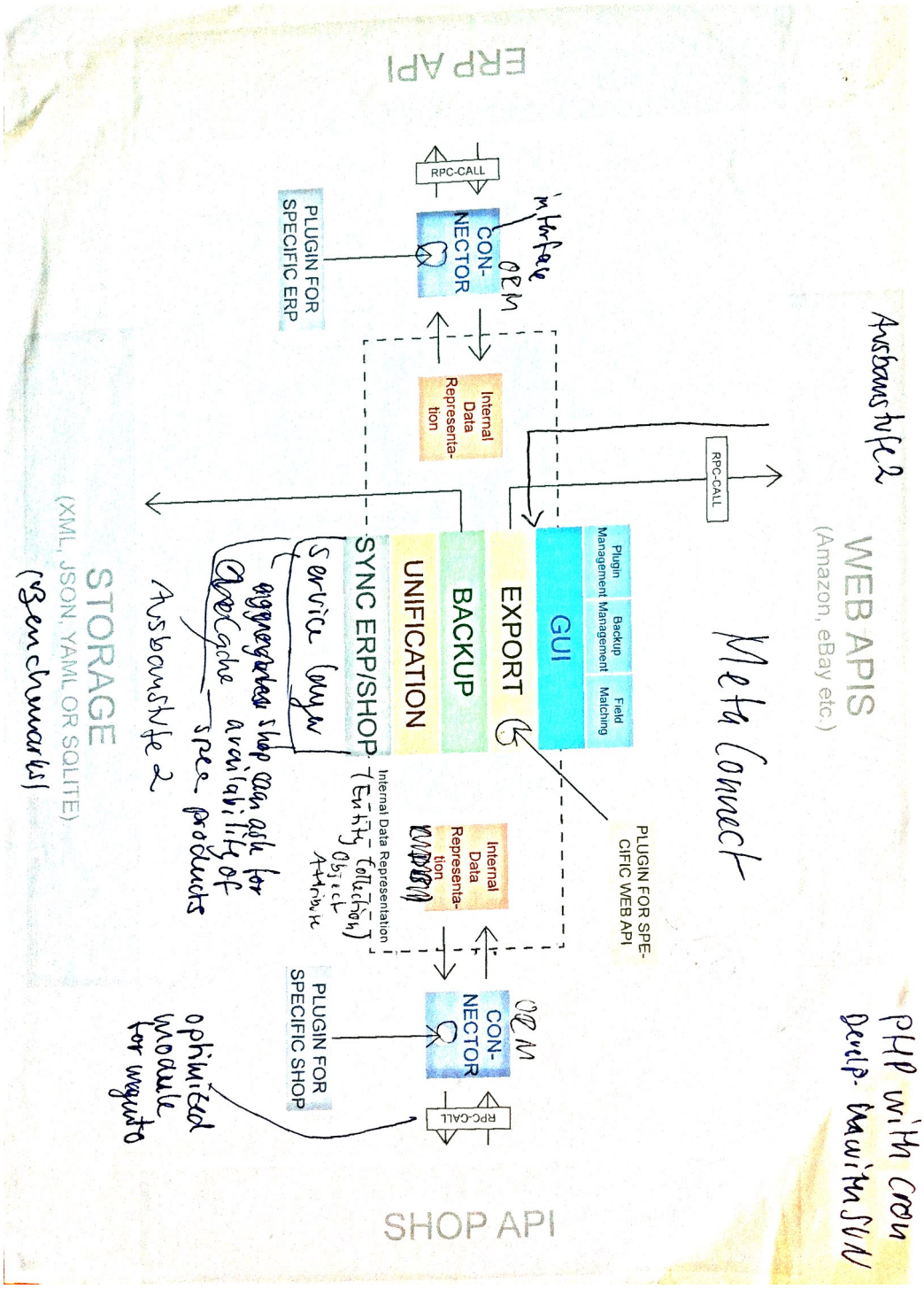
A.2 REFINED SKETCH



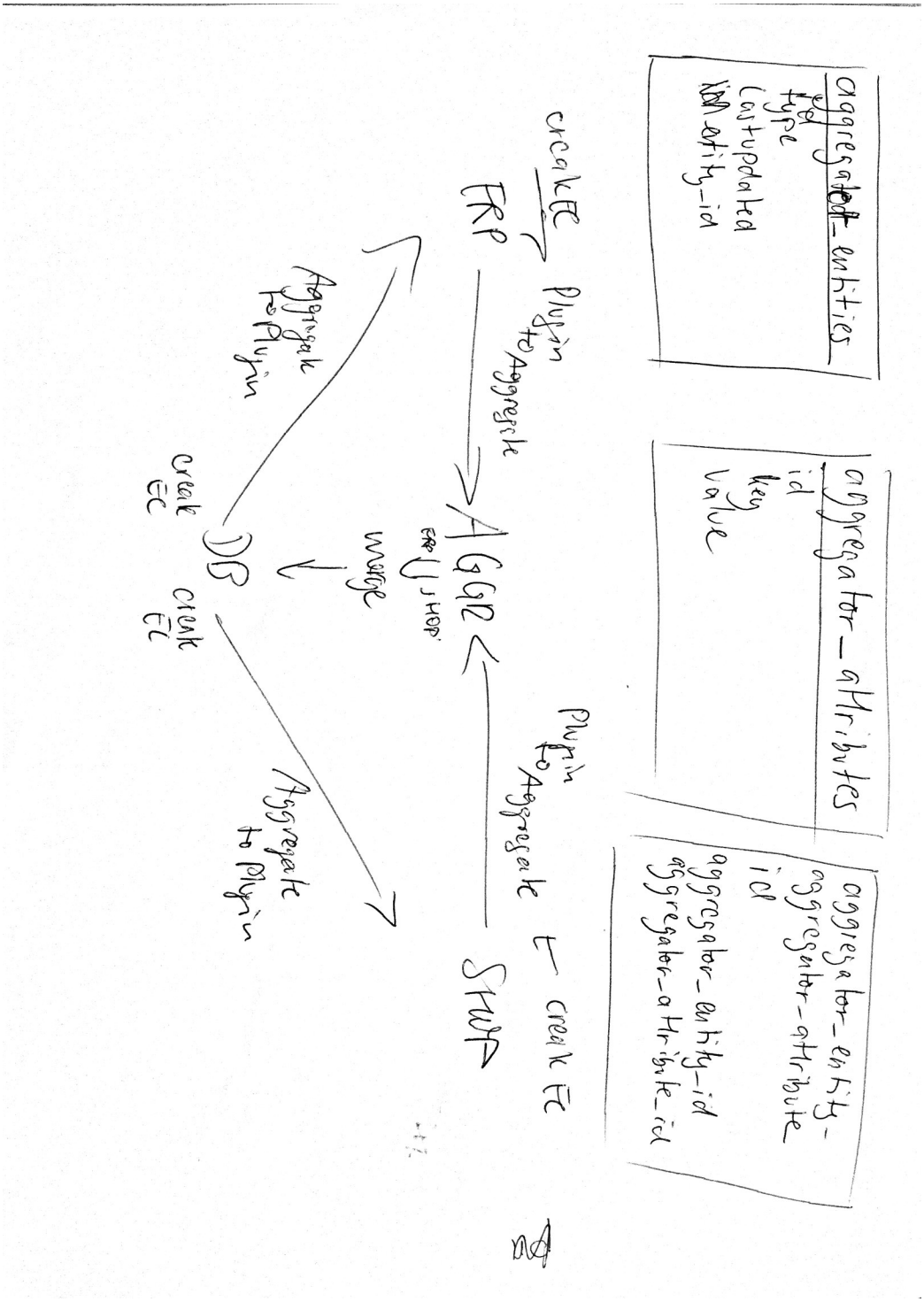
A.3 GRAPHICAL INTERPRETATION



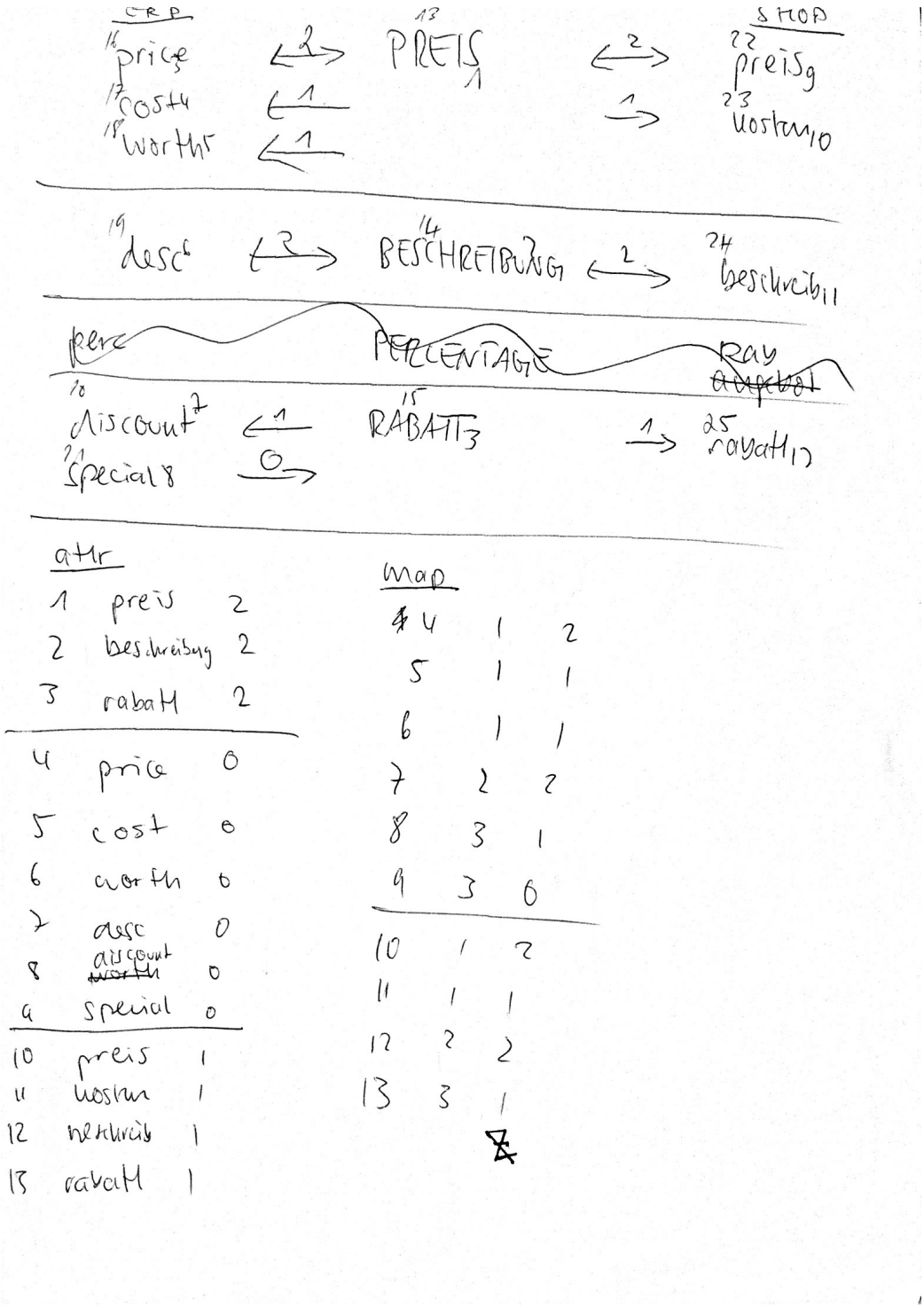
A.4 CORRECTION



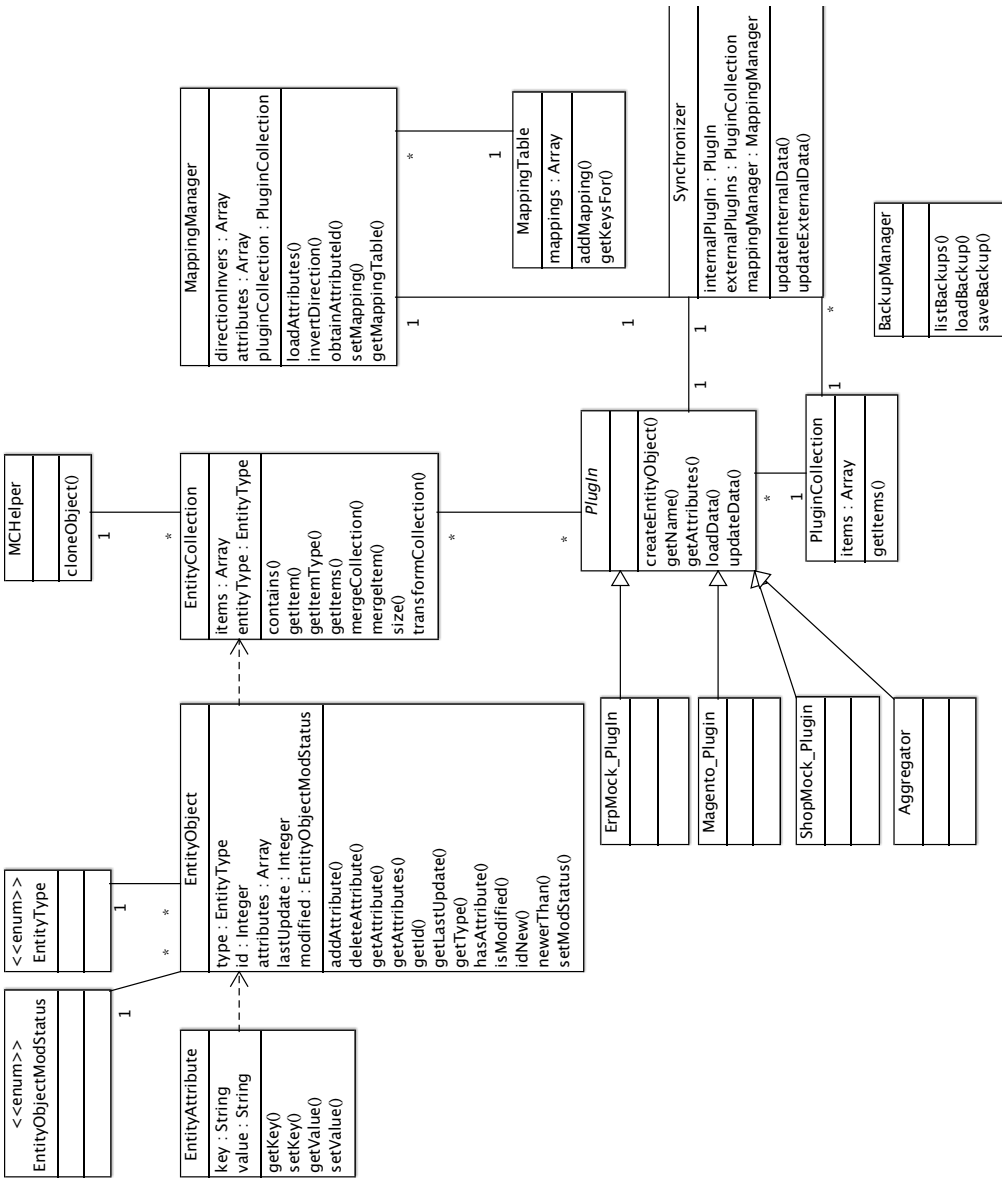
A.5 STRUCTURAL SKETCH



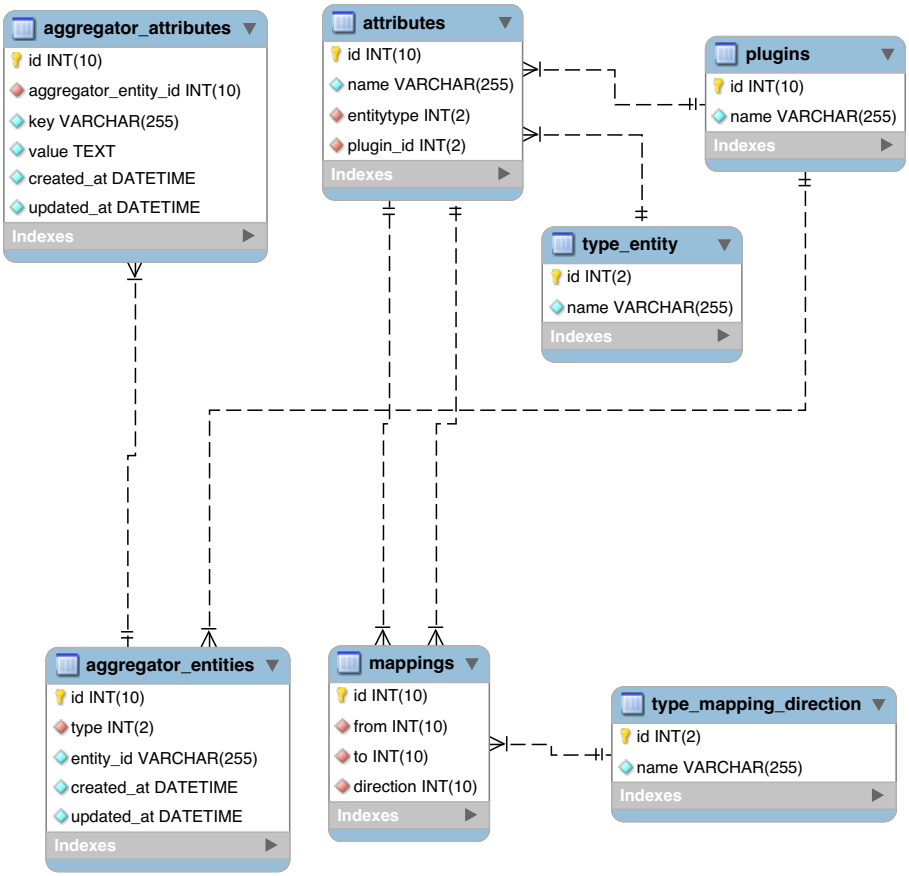
A.6 EXAMPLARY MAPPING SKETCH



A.7 UML CLASS DIAGRAM



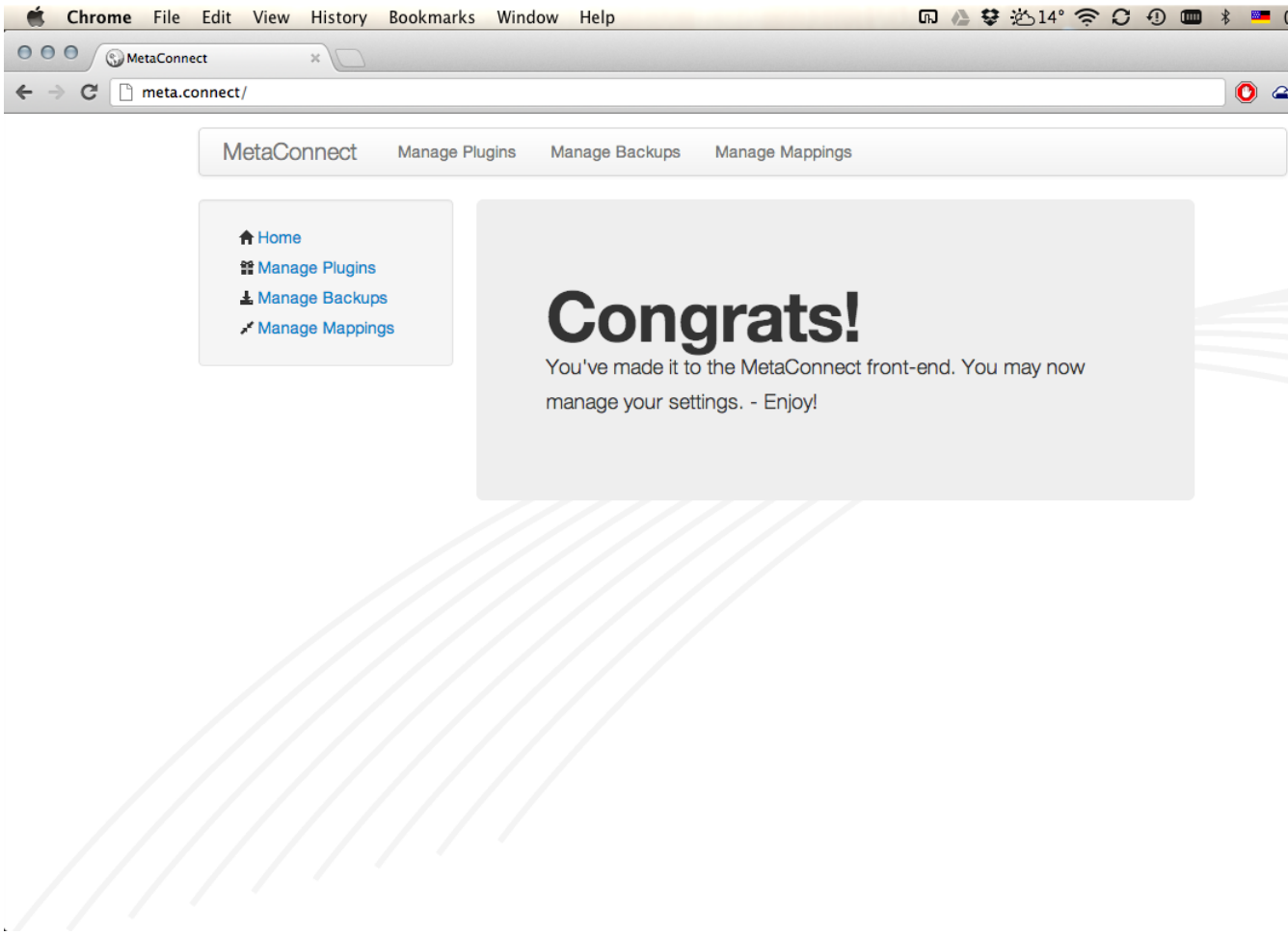
A.8 DATABASE MODEL



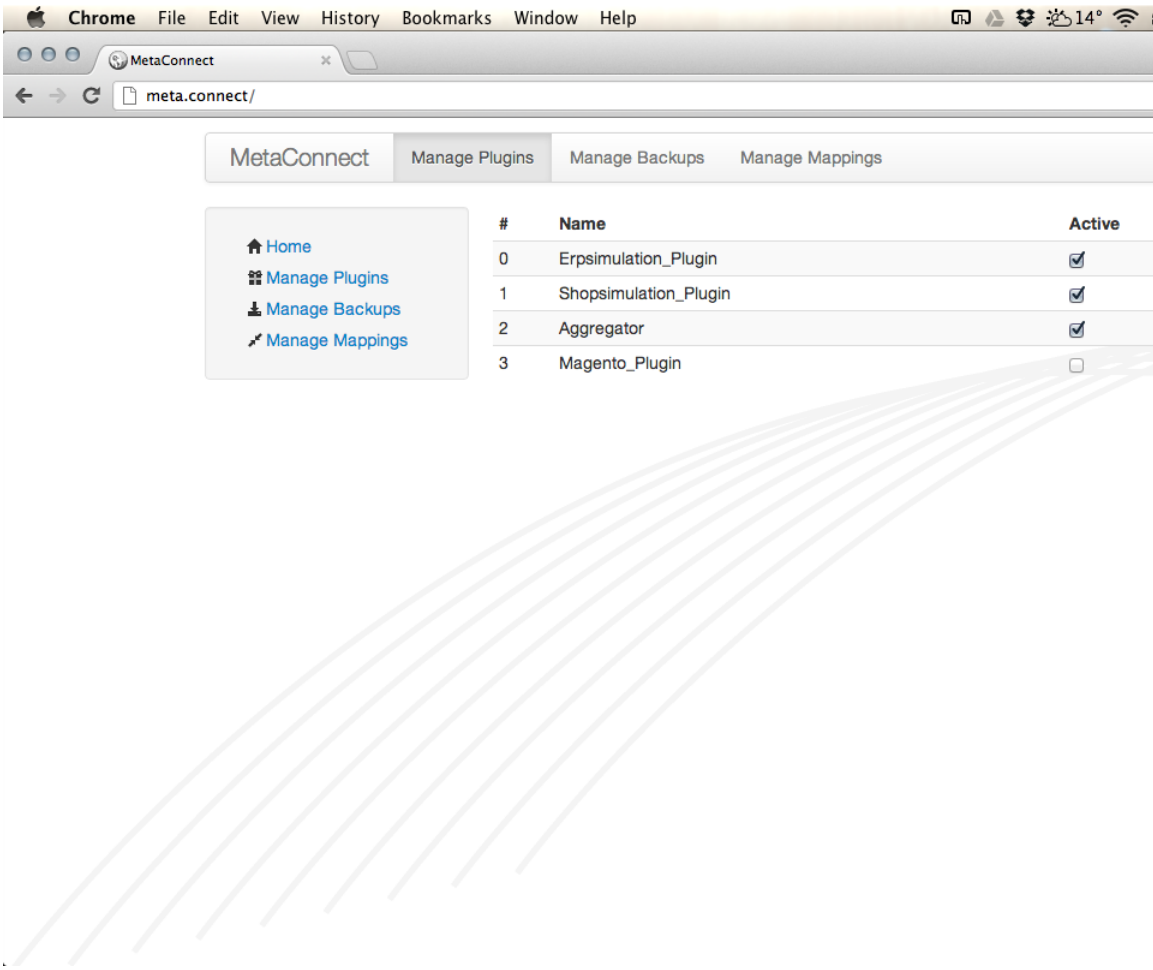
B

USER INTERFACE SCREENS

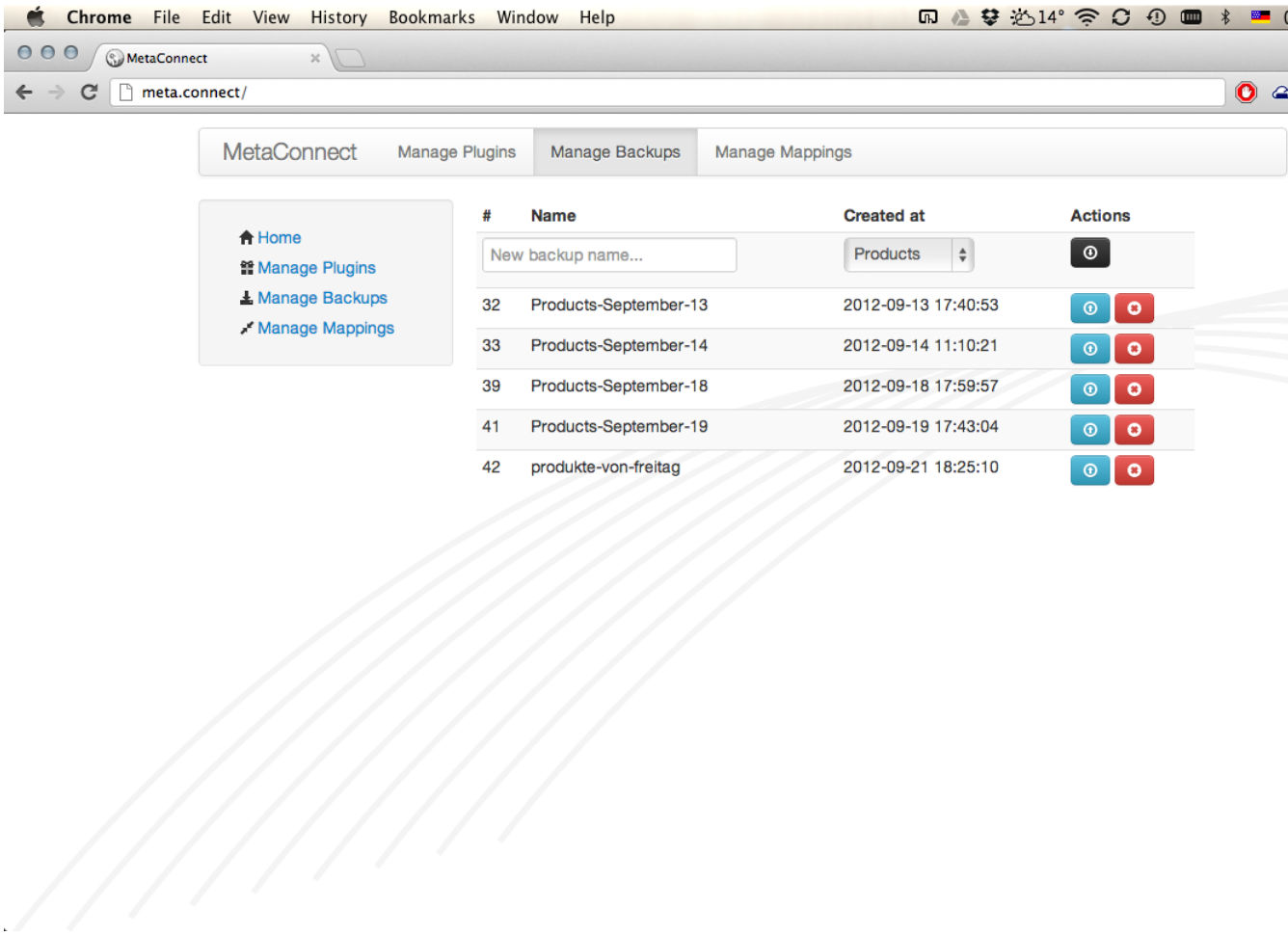
B.1 WELCOME SCREEN



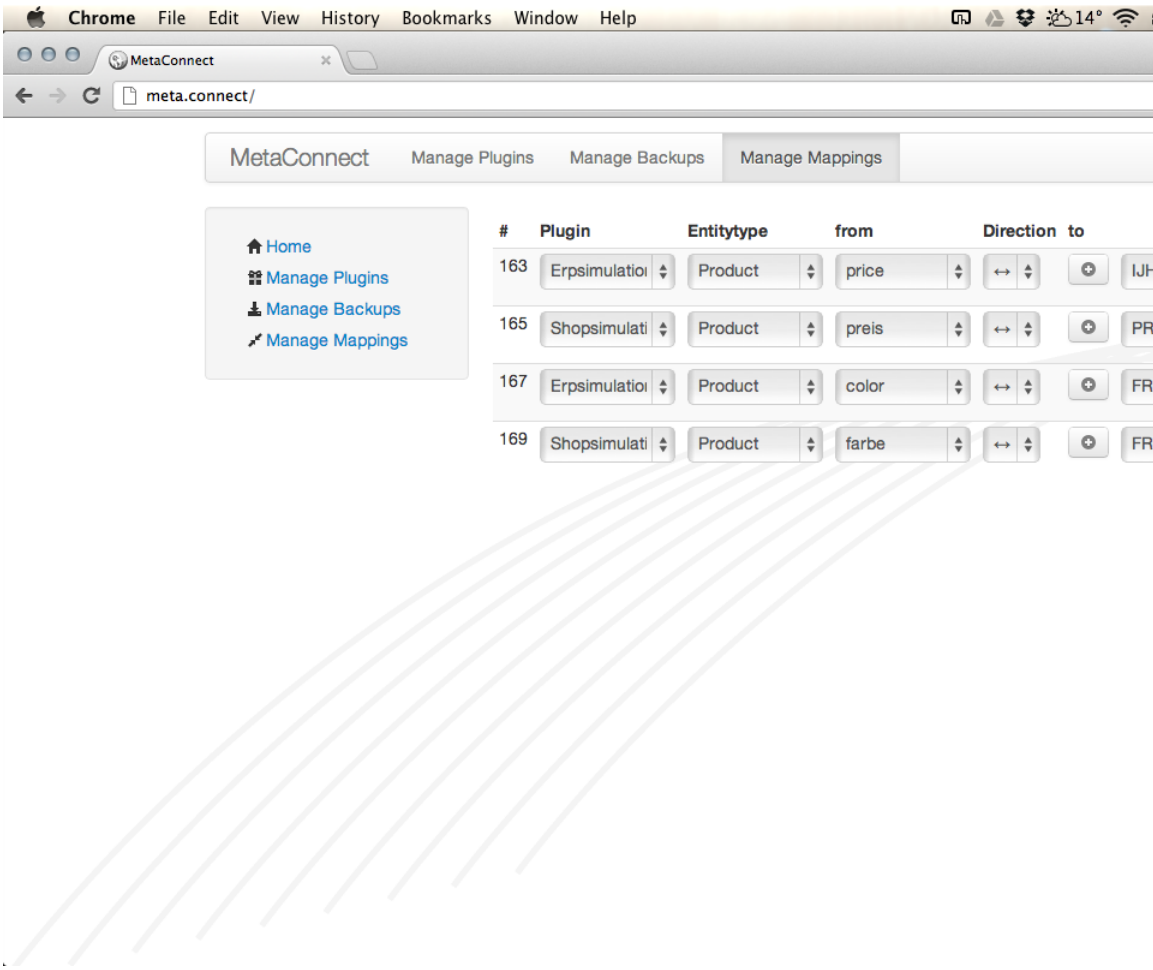
B.2 PLUGIN MANAGEMENT



B.3 BACKUP MANAGEMENT



B.4 MAPPING MANAGEMENT



C

BENCHMARK RESULTS

Environment 1 MacBook

reading 8 records
10.000 times

measurement	JSON	SQLite	XML	YAML
1	3.344764948	12.35234213	15.64031911	48.21261907
2	3.343389034	11.5903151	15.72129917	48.56260085
3	3.342483997	11.59613109	15.80639696	48.54866481
4	3.341795921	11.64996099	15.71654701	48.83457112
5	3.350692034	11.56203318	15.81153202	48.12112808
6	3.355707169	11.70489287	15.93528008	48.5264039
7	3.344056845	11.68908596	15.49009395	48.83996606
8	3.275630951	11.59707713	16.15463686	49.48261285
9	3.349509954	11.73990202	15.61178088	48.72427082
10	3.464950085	11.61571097	15.76563501	48.23481202
average	3.351298094	11.70974514	15.76535211	48.60876496
best	3.275630951	11.56203318	15.49009395	48.12112808
worst	3.464950085	12.35234213	16.15463686	49.48261285

writing 8 records
10.000 times

measurement	JSON	SQLite	XML	YAML
1	3.121242046	stopped after 120s	5.809509039	26.10591507
2	3.11026597	stopped after 120s	5.956521988	25.27612495
3	2.796883106	stopped after 120s	5.42859602	25.42989016
4	3.04662919	stopped after 120s	5.683878183	25.33390617
5	3.080001831	stopped after 120s	5.550884008	25.75835896
6	4.101543903	stopped after 120s	5.557695866	25.0835309
7	4.018939972	stopped after 120s	5.503942013	25.13497591
8	4.77639699	stopped after 120s	5.657356977	25.10111499
9	3.063992023	stopped after 120s	5.636440992	25.35941696
10	3.773044109	stopped after 120s	5.66640687	25.32925582
average	3.488893914	-	5.645123196	25.39124899
best	2.796883106	-	5.42859602	25.0835309
worst	4.77639699	-	5.956521988	26.10591507

Environment 2 Host Europe

reading 8 records
10.000 times

measurement	JSON	SQLite	XML	YAML
1	2.224435091	9.277062893	9.527079821	40.51149297
2	2.339328051	8.872797966	9.925572872	36.72159195
3	2.128660917	9.13043499	9.929268122	39.64556193
4	2.224161148	8.523307085	10.54573703	39.39475393
5	2.107556105	9.432006121	9.821069002	36.18680406
6	2.346214056	8.644522905	9.882927895	37.23000288
7	2.287798882	8.675698996	10.17907095	37.30954289
8	2.408726931	8.75156498	10.0653851	37.7401011
9	2.110493898	8.94362998	10.56099796	36.97162414
10	2.097667933	8.981865883	10.520998	37.41181684
average	2.227504301	8.92328918	10.09581068	37.91232927
best	2.097667933	8.523307085	9.527079821	36.18680406
worst	2.408726931	9.432006121	10.56099796	40.51149297

writing 8 records
10.000 times

measurement	JSON	SQLite	XML	YAML
1	13.677356	stopped after 120s	14.50166893	26.41431999
2	14.21045208	stopped after 120s	10.57201099	25.98474789
3	15.0155499	stopped after 120s	12.60026288	28.29997993
4	12.12916899	stopped after 120s	12.78036094	26.09623098
5	14.62764812	stopped after 120s	10.41261601	26.0448029
6	16.55112386	stopped after 120s	11.56860089	26.13607597
7	10.77413702	stopped after 120s	11.35292912	25.59532309
8	11.50811601	stopped after 120s	12.02803206	24.808851
9	10.50431895	stopped after 120s	12.44596887	23.96159506
10	12.46828389	stopped after 120s	10.18657804	24.83457994
average	13.14661548	-	11.84490287	25.81765068
best	10.50431895	-	10.18657804	23.96159506
worst	16.55112386	-	14.50166893	28.29997993

Environment 3 Amazon EC2

reading 8 records
10.000 times

measurement	JSON	SQLite	XML	YAML
1	2.140808105	not supported	58.62773299	205.8866398
2	2.291484118		46.37091899	186.497628
3	2.362462997		45.90025115	177.4760079
4	2.152868032		53.31388807	190.077003
5	2.166412115		35.18192005	228.4907298
6	11.96863604		51.92143703	187.6231689
7	10.18331289		42.26877117	201.1431241
8	2.188052177		63.09498096	217.0930951
9	2.148638964		54.59226489	192.2938981
10	18.80944991		48.50078201	187.2824252
average	5.641212535	-	49.97729473	197.386372
best	2.140808105	-	35.18192005	177.4760079
worst	18.80944991	-	63.09498096	228.4907298

10.000 times

measurement	JSON	SQLite	XML	YAML
1	21.74479508	not supported	41.41932011	85.34918308
2	21.69668221		40.34828496	123.210891
3	22.40220594		19.96676993	83.28403401
4	21.72817588		17.90592003	61.27904296
5	21.52501702		14.24857497	113.1236761
6	15.73689795		15.96622992	118.4832141
7	11.34773684		16.58091021	76.21556401
8	11.49295807		16.53627396	63.11704206
9	11.61165285		14.32251692	121.387917
10	11.44437003		36.91662192	112.729224
average	17.07304919	-	23.42114229	95.81797884
best	11.34773684	-	14.24857497	61.27904296
worst	22.40220594	-	41.41932011	123.210891

BIBLIOGRAPHY

[1] E-Commerce als Chance für den Einzelhandel? » shopbetreiber-blog.de. [Online; accessed 08/08/2012].

[2] Jim Odhiambo Otieno. Enterprise Resource Planning (ERP) Systems Implementation Challenges: A Kenyan Case Study, 2008.

[3] Nick Wailes Christopher Wright David Grant Richard Hall. The false promise of technological determinism: the case of enterprise resource planning systems. *New Technology, Work and Employment*, 21(1):2–15, March 2006.

[4] Mehdi Khosrow-Puor. Emerging Trends and Challenges in Information Technology Management. (865), 2006.

[5] Carl Dahlén and Johan Elfsson. An analysis of the current and future ERP market -with focus on Sweden. (February), 1999.

[6] ERP Fight Club : SAP vs Oracle vs Microsoft Dynamics « SAP Ignite. [Online; accessed 14/08/2012].

[7] Sage ERP X3, Product Highlights. [Online; accessed 19/09/2012].

[8] Sage ERP X3, Features & Functions. [Online; accessed 19/09/2012].

[9] Sage ERP X3, Easy set-up for multiple countries. [Online; accessed 19/09/2012].

[10] Sage ERP X3, Technology. [Online; accessed 19/09/2012].

[11] Paul M. Diffenderfer and Samir El-Assal. *Microsoft Dynamics NAV Jump Start to Optimization*. Vieweg+Teubner, 2008.

[12] Microsoft. Microsoft Dynamics NAV Capabilities. 2008.

[13] Fabien Pinckaers, Geoff Gardiner, and Els Van Vossel. Open ERP, a modern approach to integrated business management. 2011.

[14] Kategorie:JTL-Wawi:Einleitung – JTLWiki. [Online; accessed 14/08/2012].

[15] Kategorie:JTL-Wawi:Im-Export Schnittstellen – JTLWiki. [Online; accessed 14/08/2012].

[16] Faktura-XP 2012 - Warenwirtschaft / ERP - Schnittstellen aktuell. [Online; accessed 14/08/2012].

[17] Lexware warenwirtschaft pro - Vom Einkauf bis zum Verkauf besser wirtschaften - shop.lexware.de. [Online; accessed 14/08/2012].

[18] Siller Retail Solutions. Intersys 5.0 - High Performance für Warenwirtschaft und Kasse.

[19] Magento Covers the Quarter of the e-Commerce Market. [Online; accessed 19/09/2012].

[20] Shopsoftware - Die wichtigsten Funktionen und Module - Gambio GX2 - Gambio GmbH. [Online; accessed 19/09/2012].

[21] Adam McCombs and Robert Banh. The Definitive Guide to Magento. In *The Definitive Guide to Magento*, pages 179–190.

[22] OXID eSales | Preview | Facts | Produkte. [Online; accessed 19/09/2012].

[23] OXID eSales | Produktinformationen | OXID eShop Enterprise Edition | Facts | Produkte.

[24] Digidesk - media solutions | OXID eShop Lösungen | PHP Shopsystem | Shopsysteme die verkaufen! - Schnittstelle ERP SOAP/CSV Connector für OXID eShop. [Online; accessed 19/09/2012].

[25] PrestaShop Module und PrestaShop Themen für Ihren Online-Shop - PrestaShop. [Online; accessed 19/09/2012].

[26] prestasoap - Simple Prestashop SOAP - Google Project Hosting. [Online; accessed 19/09/2012].

[27] Shopsoftware-Funktionen - Funktionen Shopsystem - Funktionen Onlineshop. [Online; accessed 19/09/2012].

[28] VEYTON SOAP. [Online; accessed 08/10/2012].

[29] Open ERP Connector by Openlabs - Magento Connect. [Online; accessed 08/10/2012].

[30] JTL-Wawi Connectoren: JTL Software. [Online; accessed 08/10/2012].

[31] techreceptives / openerp_prestashop_sync / overview — Bitbucket. [Online; accessed 08/10/2012].

[32] JSON: The JavaScript subset that isn't — Timeless. [Online; accessed 14/08/2012].

[33] Douglas Crockford. JSON. [Online; accessed 14/08/2012].

[34] A comparison of php json libraries. [Online; accessed 14/08/2012].

[35] About SQLite. [Online; accessed 14/08/2012].

[36] PHP: Introduction - Manual. [Online; accessed 14/08/2012].

[37] Extensible Markup Language (XML) 1.0 (Fifth Edition). [Online; accessed 14/08/2012].

[38] PHP: Introduction - Manual. [Online; accessed 14/08/2012].

[39] PHP: Introduction - Manual. [Online; accessed 14/08/2012].

[40] Yet Another Markup Language (YAML) 1.0. [Online; accessed 14/08/2012].

[41] spyc - A simple YAML loader/dumper class for PHP - Google Project Hosting. [Online; accessed 14/08/2012].

[42] Instance Families and Types - Amazon Elastic Compute Cloud. [Online; accessed 14/08/2012].

[43] Markus Lanthaler and Christian Gütl. On using json-ld to create evolvable restful services. pages 25–32, 2012.

[44] 45% of the popular websites use a javascript framework. [Online; accessed 16/08/2012].

[45] Fielding Dissertation: CHAPTER 5: Representational State Transfer (REST). [Online; accessed 08/10/2012].

[46] Alex MacCaw. The little book on coffeescript. 2011.

[47] Interview: Jeremy Ashkenas Talks About CoffeeScript. [Online; accessed 08/10/2012].

[48] Backbone.js. [Online; accessed 08/10/2012].

[49] Alex MacCaw. *JavaScript Web Applications (Otx)*. O'Reilly Media, Inc., 2011.

DECLARATION

Ich versichere, dass ich die Arbeit selbstständig verfasst und keine anderen, als die angegebenen Hilfsmittel - insbesondere keine im Quellenverzeichnis nicht benannten Internetquellen benutzt habe, die Arbeit vorher nicht in einem anderen Prüfungsverfahren eingereicht habe und die eingereichte schriftliche Fassung der auf dem elektronischen Speichermedium entspricht.

Ich bin mit der Einstellung der Bachelor-Arbeit in den Bestand der Bibliothek des Departments Informatik einverstanden.

London, 20.10.2012

Tim-Daniel M Jacobi