REFERENCE TO IMPLEMENTATION AND DESIGN FOR E-COMMERCE PORTAL SELLING FREENEST SERVICE

Jozef Paľa

Bachelor's Thesis
4 / 2013

Degree Programme in Software Engineering
School of Technology

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

Author(s)	Type of publication	Date
PAĽA, Jozef	Bachelor´s Thesis	09-05-2013
	Pages 62	Language English
		Permission for web publication (X)

Title
REFERENCE TO IMPLEMENTATION AND DESIGN FOR E-COMMERCE PORTAL SELLING FREENEST SERVICE

Degree Programme
Software Engineering

Tutor(s)
PELTOMÄKI, Juha

Assigned by
SkyNest Project

Abstract
FreeNest offers a team-oriented product development platform. The solution is provided as an open source software available for anyone to download and use within their own hardware.

The main research question was how to simplify this process and how to allow the customers to use FreeNest service without need to set up their own server environment. The use of the cloud for this purpose is the ideal solution how to satisfy the customer and how to earn money to fund the development of an open source project such as FreeNest.

The main goal was to provide for the customers an interface where they can safely order and pay for the FreeNest service in the cloud. This interface will be used as a proof of the concept for the further development of the final e-commerce portal solution.

The newest technologies for this purpose such as HTML5, CSS3 and JavaScript together with the well-established technologies such as PHP and MySQL were used. For the cloud part of the solution, OpenStack was used. The target was to create a responsive and dynamic web page, which would be very attractive to the user.

An important part was the research of payment services, which offer various local payment options. Those services can be used to provide customers with well-known and trusted payment options well suited to their location. There are numerous companies offering this type of services. Comparison of the selected ones, together with the information about their implementation, was done.

The selected services for this purpose are Adyen and Suomen Verkkomaksut. Adyen is an example of an international service providing different local payment options in various countries worldwide. Suomen Verkkomaksut focuses on the Finnish market and provides solutions useful especially for the company with the customer base mainly in the Finland.

Keywords
HTML5, CSS3, JavaScript, AJAX, jQuery, PHP, MySQL, OpenStack, E-commerce portal, Adyen, Suomen Verkkomaksut, Payment service

Miscellaneous

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

AJAX	Asynchronous JavaScript and XML
API	Application Programming Interface
AWS	Amazon Web Services
CSS	Cascading Style Sheets
HDD	Hard Disk Drive
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
JSON	JavaScript Object Notation
PHP	PHP: HyperText Preprocessor
RAM	Random Access Memory
SQL	Structured Query Language
SV	Suomen Verkkomaksut
URL	Uniform Resource Locator
XML	Extensible Markup Language

1 INTRODUCTION AND OBJECTIVES

FreeNest is a team-oriented product development platform that is fast to deploy and easy to use. FreeNest integrates together other open source software, which is commonly used in software development, and acts as a sort of glue between them. FreeNest is modular and expandable, allowing customer to create just the right environment for their needs (About FreeNest, official website).

FreeNest is an open source solution; therefore, all parts of the software code are available free of charge to everyone. It is distributed in Debian packages and can be installed on supported versions of Ubuntu server edition operating system. Due to this fact, anyone with available hardware can install and run their own instance of FreeNest development platform. On the other side, open source character of FreeNest platform brings question how to earn money needed to fund the project.

However, nowadays there is a trend to decrease costs as much as possible. Purchasing and maintaining hardware can be expensive and there might be even more costs added if a customer would like to run more stand-alone instances of FreeNest development platform, which would mean a separate server for each instance.

Moving those problems from customers to a distant cloud service is an ideal solution benefiting both sides. Instead of customers owning all necessary hardware, it is covered by a cloud service, which gives customers only an access point to the product. In case of FreeNest, this access point is a web interface where customers can access all development tools.

This solution also allows an income source for an open source project such as FreeNest. Customers pay a different amount of money for this type of service according to the package they chose while ordering it. Another possibility how to earn money is to offer advanced services such as personal support through email, instant chat or phone.

There are several questions, which need to be solved, for example how should this type of e-commerce portal look like, what type of secure payment method will it use, how can customers manage their ordered services and much more. This thesis will focus on researching and finding out answers for some of those questions.

The result of this thesis is proof of the concept of the e-commerce portal for distributing FreeNest developing platform instances inside the cloud infrastructure. The main goal is to offer customers a user-friendly page where they can order and manage their instances of FreeNest.

While full functionality is not necessary, there are few targets, which this thesis focuses on. A very important part will be the research of payment services and methods. Customers will be most likely from different countries all over the world and therefore there is need to find out an ideal solution for them through which they are able to process payment for this cloud service. The target is to look into advantages and disadvantages of few of them and find out how they can be implemented within this e-commerce portal. Some of them are practically implemented as a part of proof of the concept of this e-commerce portal.

Different companies offer versatile integrated services where customers can chose any payment method from various countries and the payment will be after sent to the merchant's account. Those services differ in functionality, list of supported countries, price and fees and therefore research to find out the ideal one is necessary.

Another important target is to look into similar portals offering cloud services. A study of their functionality can help to improve this thesis or to set ideas, which could be implemented in near future when final version of the e-commerce portal with the full functionality will be released

Finally, yet importantly, there have to be communication between the e-commerce portal and the cloud infrastructure, which will host FreeNest instances. There has to be way how the e-commerce portal will communicate with the cloud and how the instances will be created and managed there.

2 TECHNOLOGY OVERVIEW

In the following section, there will be a brief description of technologies used while creating the proof of the concept of the e-commerce portal. The main technologies chosen are HTML5, JavaScript and CSS3 to create a user-friendly, intuitive, responsive and graphically interesting web page. PHP was chosen for the server side logic and MySQL for storing all necessary data about users. OpenStack was chosen to manage the cloud services.

2.1 HTML5

HTML5 is the new standard for the HTML and it is supposed to replace the previous standard (version 4.01) from 1999. The web has changed markedly since then and therefore there is a need for a new standard. All the major browsers support most of the HTML5's new functions and APIs; however, it is not recognized as a fully defined standard yet (HTML5 Introduction, w3schools).

Even HTML5 is not official standard yet, all elements needed in this thesis are already widely supported and therefore there is no need to use older standard. This will also allow easier maintaining and updating page once HTML5 will be fully standardized. Below is an example of a simple HTML5 page.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Title of the web page</title>
  </head>

  <body>
    <p id="body_txt">Body of the web page.</p>
  </body>
</html>
```

2.2 JavaScript

JavaScript is a cross-platform, object-oriented scripting language used to create rich effects to the web applications. JavaScript runs on client's side, which allows the web

page to be fully and instantly responsive without need to communicate with server after each request from the customer. It is not used as a standalone language, but rather as an addition to extend the functionality of the web pages or other applications (JavaScript Overview, Mozilla Developer Network).

JavaScript can directly change the content of the HTML web page, which makes it very popular for creating dynamic and fast web pages. The example below shows simple JavaScript code, which would change the body text from the previous code ("Body of the web page.") into "This was changed by the JavaScript".

The fact that JavaScript allows to manipulate with HMTL document without need to call server to generate website itself allows creating very fast and user-friendly web pages. It can be used for example to generate content according to user's input (showing of additional details after a user clicks button, easier calculations and more).

```
<script>
document.getElementById("body_txt").innerHTML="This was
changed by the JavaScript";
</script>
```

2.2.1 AJAX

AJAX is not a separate or standalone programming language. It is a new way of using already existing standards such as XMLHttpRequest object, JavaScript, CSS, XML and JSON. AJAX allows web pages to update asynchronously by exchanging small amounts of data with web server, therefore it can update web page without need to reload it. It is used to create fast and dynamic webpages. Well-known websites using this technology are for example Google Maps, Gmail, YouTube and Facebook (AJAX Introduction, w3schools).

Ajax can be used for example in a chat application where messages are sent to the server and AJAX requests for new incoming ones. Once there are new messages, the text field, where they are shown, can be updated through JavaScript without need to reload whole page.

Figure 1 below shows how AJAX works. The process begins with an event on the client's side (the user clicks some button or the timer was set, which calls AJAX function in predefined time). HTTP request is formed and sent over the Internet to the web server. The web server processes it, creates a response for the client and sends it back over the Internet. The client's browser receives the response and processes it using JavaScript. JavaScript then updates the content of the web page according to the response.

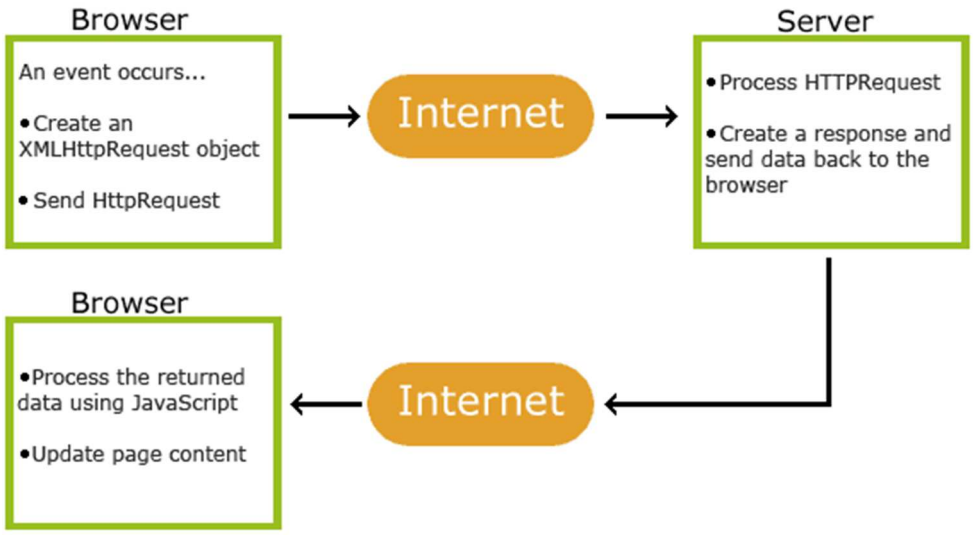


FIGURE 1: *How AJAX works (AJAX Introduction, w3schools)*

An example of the AJAX code below requests a web server, which returns plain text as a response. When this response is received, JavaScript replaces the body text ("Body of the web page.") in the code from the section about HTML5 for the text received from that server response.

```
function loadXMLDoc() {
  var xmlhttp;
  var text  = "";
  if (window.XMLHttpRequest) {
    // code for IE7+, Firefox, Chrome, Opera, Safari
    xmlhttp=new XMLHttpRequest();
  } else {
    // code for IE6, IE5
    xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
  }
  xmlhttp.onreadystatechange=function() {
    if (xmlhttp.readyState==4 && xmlhttp.status==200) {
      text = xmlhttp.responseText;
      document.getElementById("body_txt").innerHTML=text;
    }
  }
  xmlhttp.open("GET","some_page.php",true);
  xmlhttp.send();
}
```

In this thesis, AJAX will be widely used to load page content and to process user's inputs. This will ensure that a web page will load fast because its skeleton is loaded once at the beginning and specific parts requested by user are loaded just when needed. In addition, for example, when users want to update some form, they do not need to send it to another processing PHP page, as it was standard in older web pages, but the web page can just send an AJAX request and update content according to the response (Advantages of using AJAX with PHP).

2.2.2 jQuery

jQuery is fast, small and feature-rich JavaScript library. It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers (official website of jQuery).

The motto of jQuery is „write less, do more". It simplifies many common tasks, which require programmers to write many lines of JavaScript code and wraps them into simple methods. jQuery supports all browsers including Internet Explorer 6 (jQuery Introduction, w3schools).

The example below shows a code from the previous example written in the jQuery syntax.

```
$.ajax({
  url: "some_page.php",
  type: "GET"
}).done(function(data){
  $("#body_txt").html(data);
});
```

As shown in the code above, the same function was reduced by about 60-70%. AJAX is an important part of the practical part in this thesis and simplifying code through jQuery is very useful. Nevertheless, jQuery does not simplify only AJAX. Elements can be accessed and manipulated much easier and the basic library includes also several animations such as fade or slide effects, which make the user experience much better.

The “$” symbol is an alias for “jQuery”. It is used as an expression for identifying elements on a page and it combines jQuery syntax with JavaScript (Hima’s blog 2011).

2.3 CSS3

CSS define how to display HTML elements and they were added to HTML 4.0 to solve problems with style tags inside HTML code. Styling a document through those tags was very time consuming especially in extensive web sites containing many various pages. CSS solve this problem and allow developers to define several classes, which can be used universally through pages (CSS Introduction, w3schools).

CSS3 is a new standard extending functionality of CSS2 and it is fully backwards compatible. CSS3 is split up into several modules, which cover the following functionality: selectors, box model, backgrounds, borders, text effects, 2D/3D transformations, animations, multiple column layout and user interface (CSS3 Introduction, w3schools).

Some concrete examples of the new functionality are rounded borders, shadows and various colour models. A very useful feature, which will be used in this thesis as well, is the use of media queries. They allow changing of the style for example according to width and height of the screen. This allows creating web sites, which are well visible regardless of the screen size on which user browses them. HTML code stays the same as well, the only feature, which changes is the style applied. CSS3 also support custom fonts, transition effects and many others (Media Queries, Webflux).

The two examples below define a block element with green background, white text colour, rounded borders and shadow. However, currently there must be a definition of some CSS3 attributes like rounded borders or shadows with prefixes (-moz- for FireFox, -webkit- for Chrome or -o- for Opera). They are needed to support those attributes also in browsers which did not implement them fully yet or their implementation is currently in testing phase. Those prefixes are not shown in the first example below to keep it as a clear CSS3; however, they are shown in the second example for better comparison.

```
div.special_style {
  background: green;
  color: white;
  border-radius: 20px;
  box-shadow: 10px 10px 5px #888;
}

div.special_style {
  background: green;
  color: white;
  -moz-border-radius: 15px;
  border-radius: 20px;
  -moz-box-shadow: 10px 10px 5px #888;
  -webkit-box-shadow: 10px 10px 5px #888;
  box-shadow: 10px 10px 5px #888;
}
```

For the new CSS3 features, there are generators, which generate correct syntax for a specific attribute including the prefixes such as gradient generator (http://www.color-zilla.com/gradient-editor/), border radius generator (http://border-radius.com/) or box shadow generator (http://css3gen.com/box-shadow/).

2.4 PHP

PHP is a widely used, open-source scripting language. PHP scripts are executed on the server side. PHP can be used to create dynamic web pages, process server files, collect and process form data, work with cookies, access and modify database, restrict access of users to some pages and encrypt data (PHP Introduction, w3schools).

The advantages of the PHP are that it can be used across the platforms, it is compatible with almost all currently used web servers and it has support for a wide range of databases (PHP Introduction, w3schools).

PHP can be understood as a manual on how to build a web page. When a client sends a request for a web page, PHP script builds it according to his or her requirements. For example, there can be one script to build a page, which shows some article. According to which article a user wants to see, PHP script accesses a database to pick the text of that article and afterwards combines it with rest of the predefined HTML code, which is the same for each article page. The finalized HTML code is then sent to the user. The diagram of how this works is shown in Figure 2. Example below the Figure 2 is

the PHP code, which generates different text sentence depends on Boolean parameter passed through GET method.

In this thesis, PHP will be also used to process AJAX calls. AJAX will send a request including request data to some specific PHP script, which later generates a response. That response is then processed in the browser with JavaScript.

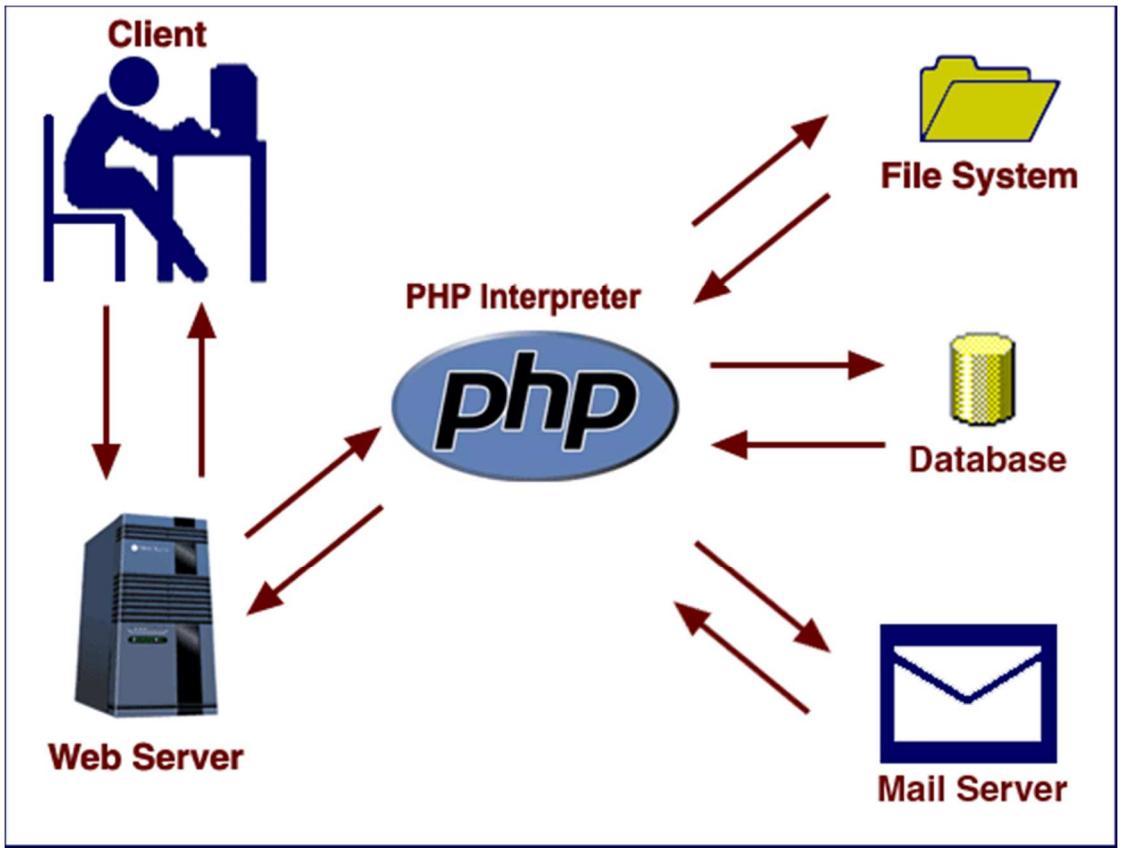


FIGURE 2: *Diagram of how the PHP works (PHP Tutorial, learnphp-tutorial)*

```
<?php
  $text = "Passed parameter was ";
  if ($_GET["some_parameter"]) {
    echo $text."true.";
  } else {
    echo $text."false.";
  }
?>
```

2.5 PHPMailer

PHPMailer is a PHP library for creating and sending e-mails. It offers many options and settings. PHPMailer is used in this thesis to send emails through external SMTP

server (Google mail) to ensure that e-mails will not be marked as spam. SMTP server of the e-commerce portal can replace this in final version.

2.6 MySQL

SQL is a standard language for accessing and manipulating databases. SQL can execute queries, retrieve data, insert, update and delete records, create databases, tables, procedures and views and set permissions on tables, procedures and views. Although SQL is a standard, there are many different versions and implementations, however, all of them implement at least major commands such as SELECT, UPDATE, DELETE, INSERT and WHERE (Introduction to SQL, w3schools).

MySQL is one of the most widely used relation database management systems and an implementation of SQL standard. Companies such as Google, Facebook, Alcatel and Adobe use it (What is MySQL, 2013).

Modern web pages are all about data. MySQL database is an ideal way to effectively store them and access them. The e-commerce portal will need to store information about users, their projects, ordered packages, chat logs and many other. MySQL database is ideal to store those data and process them with PHP to generate customized content according to user's request.

SQL language is based on queries, which define what is to be done with the data stored in the database. Queries are sentences, which must follow standardized rules of their creation. Code below shows few examples of different queries and figure 3 shows how they are processed by the database engine (for example MySQL).

```
SELECT email, address, age FROM user_data WHERE age > 25;
INSERT INTO user_data (email, age)
  VALUES (something@some.email.service.com, 28);
UPDATE user_data SET age = 56 WHERE id=4;
DELETE FROM user_data WHERE id=83;
```

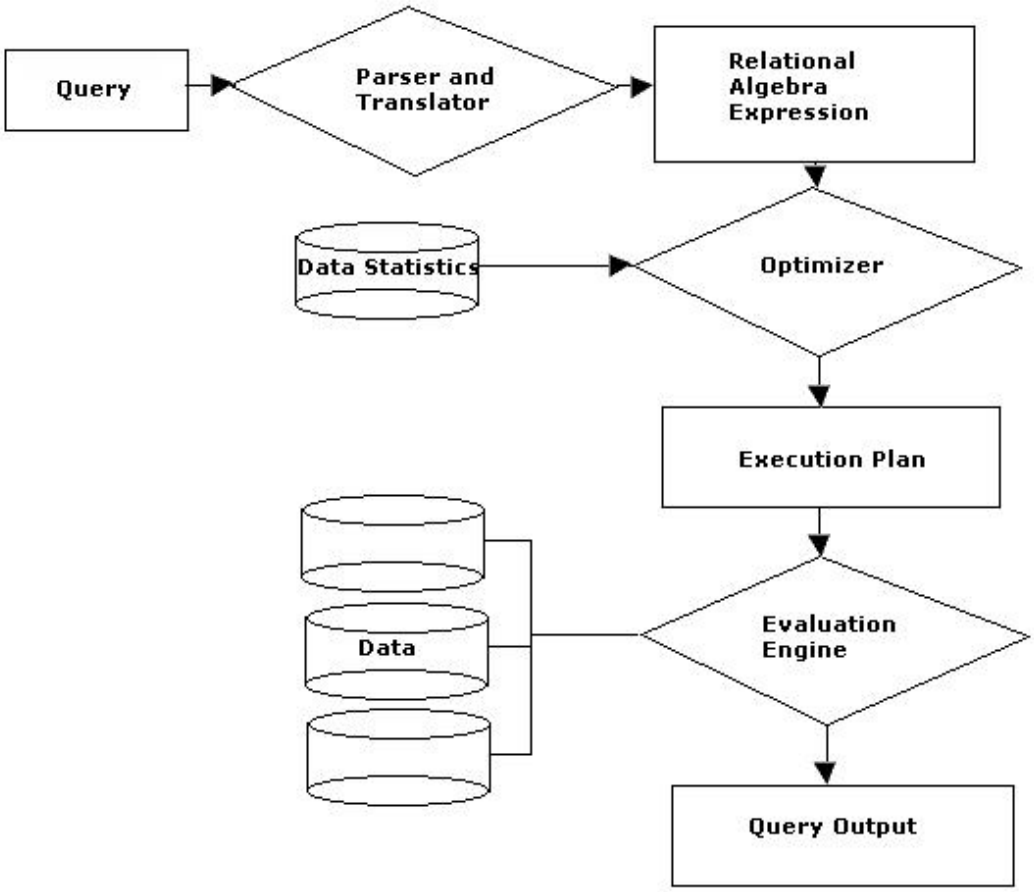


FIGURE 3: *Diagram of how the query is processed (Gopi 2010)*

2.7 OpenStack

OpenStack is a cloud operating system that controls large pools of compute, storage, and networking resources throughout a datacentre, all managed through a dashboard that gives administrators control while empowering their users to provision resources through a web interface (About OpenStack, official website).

Simplified, OpenStack acts as some type of system, which manages virtual machines. Instead of one server there can be more virtual servers running on several machines and sharing their hardware. This way there can be more servers running efficiently with less hardware.

This e-commerce portal uses OpenStack to run FreeNest instances. However, cloud infrastructure is not the main focus of this thesis and therefore this technology will be

mentioned just marginally. PHP scripts will mostly use OpenStack API which can easily just with HTTP request manage OpenStack cloud.

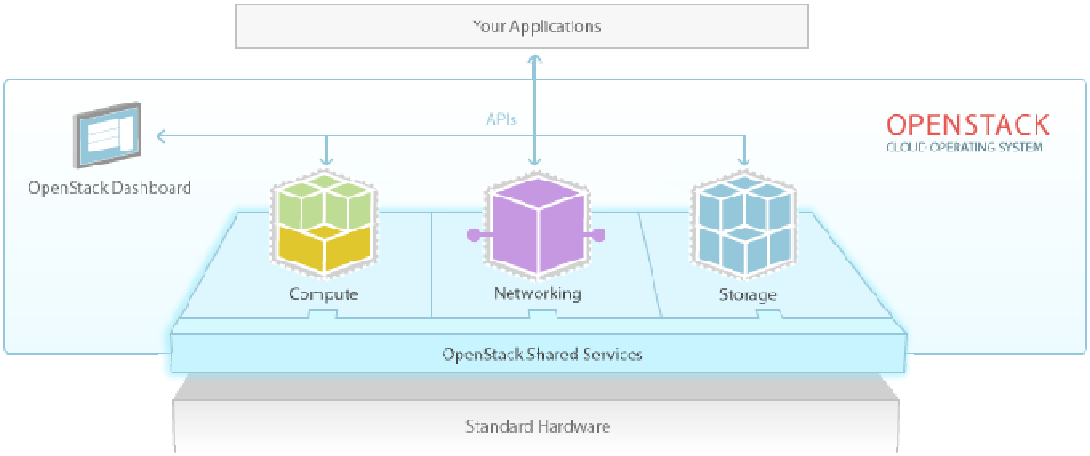


FIGURE 4: *Diagram of how OpenStack works (About OpenStack, official website)*

Figure 4 shows how OpenStack works. It consists of networking, compute and storage nodes, which are managed through OpenStack Dashboard web interface. External applications, such as FreeNest e-commerce portal, can access functions of those nodes through API.

3 E-COMMERCE PORTAL

In the following chapter, there will be analysis of the already existing similar solutions and their positive and negative sides. There is also a description of the main functionality expected from the final version of the e-commerce portal.

Customers, usually project leaders or representatives of the developer group, want to manage their work more effectively. A product development platform such as FreeNest is an ideal solution for this purpose. It allows managing the documentation through all stages of the project lifecycle with its Wiki extension. It includes tools for bug and test management, work collaboration, team board and many other useful tools.

If customers want to save time and money for creating their own FreeNest servers or if the project lifecycle is not long enough to make purchasing of the hardware efficient, they can order complete, stand-alone and instantly ready FreeNest installation inside cloud service.

The final product of this thesis will be used as a proof of concept of how this type of e-commerce portal can look like together with some part of its functionality, such as ordering FreeNest instances and their management. Therefore, an important part of this thesis will be how to offer customer safe and comfortable payment service.

3.1 Functional analysis

The idea of an e-commerce portal focuses mainly on selling packages. Packages include some number of available FreeNest instances and a specification of the virtual servers where each of them will run. A user is able to choose the package, which suits him best, and proceed to the payment.

Customers can also register on the web page without purchasing any package. Registered users can access admin panel where they can manage already purchased packages or purchase new ones.

If customers already own at least one package, they can create a FreeNest instance (project) and access it. Users can move projects between packages (in case there is more than one package purchased). They can manage automatic backups or create them when needed. They can also restore project from backup in case of error. They can close and remove old projects as well.

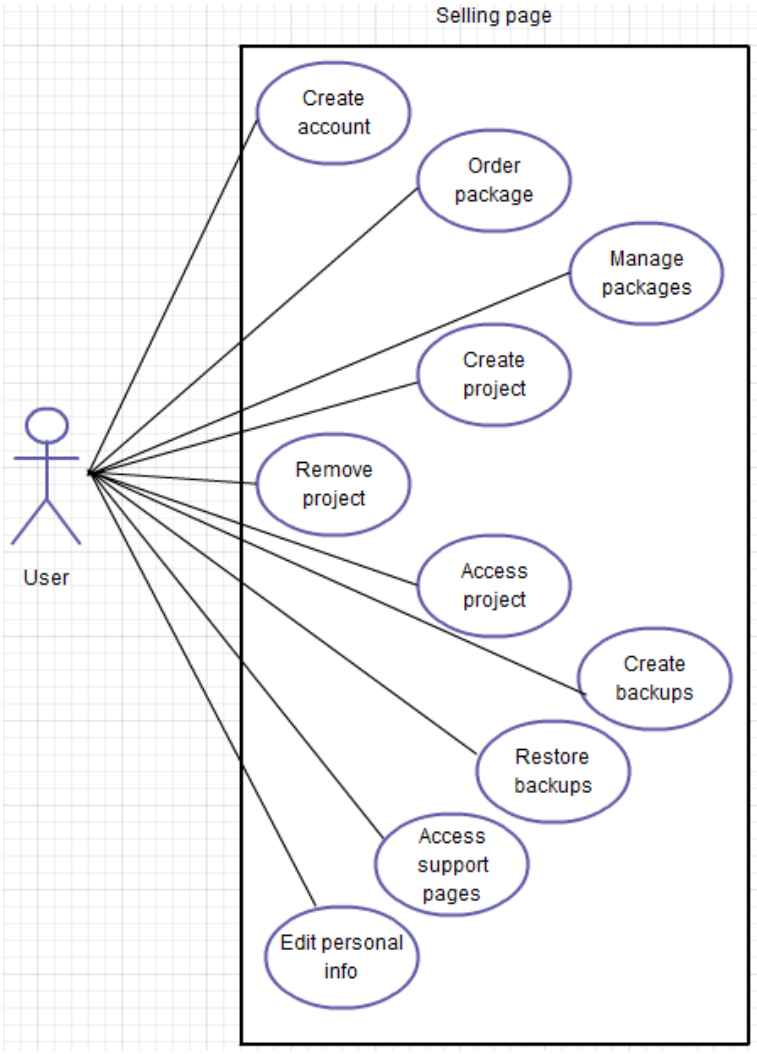


FIGURE 5: *Use case diagram*

In the case of the troubles, users can access support pages. Depends on package users have access to different types of support. Forum support is cheapest solution where community helps each other. E-mail, chat and phone are more advanced types of the support and an employee assigned to them is necessary.

Figure 5 shows the use cases of the user of e-commerce portal selling FreeNest service.

The e-commerce portal communicates with OpenStack cloud as well. When a new project is created, API is called to create a new instance of Ubuntu, inside the cloud, with FreeNest preinstalled. This can be done with a prepared snapshot.

When a user creates a backup, there will be another API call to create snapshot from currently running instance. This will save current state of the project with all data. When user restores backup, current instance will be reimaged from snapshot. Old instance will be shut down and new instance will be loaded from snapshot. This way OpenStack can handle when users move project to a different package because also specification of the virtual server can be changed at this point.

3.2 Analysis of the similar solutions

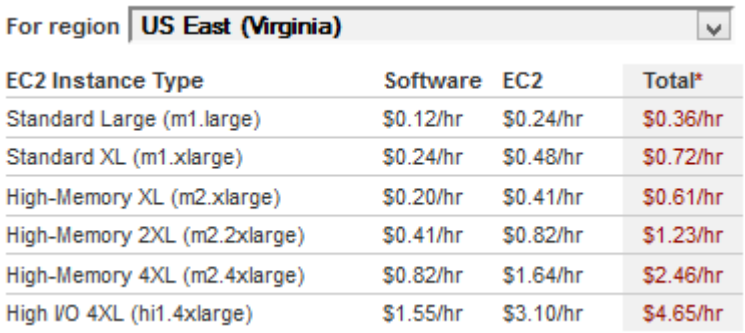
Research and analysis of the similar existing solutions is important in terms of designing this e-commerce portal. Usually those solutions have been available for a longer time already and therefore there was possibility to collect feedback of their users and reflect it into their services. Analysing them can therefore benefit this thesis to provide a user-friendly solution, which will be accepted very well by users.

3.2.1 Amazon web services

Amazon web services (AWS) is a cloud service offering customers a complete infrastructure and allowing them to run application services virtually (Official AWS website). This can include any type of server application and therefore FreeNest can also be one of the applications offered.

Users can choose from many of applications and the price for the most of them consists of two components: price for an application and price for a virtual server specification. Users pay per hour and therefore they pay just for a time during which they use the service. There are also applications based on the fixed weekly or monthly price. This is more similar to the system this thesis uses (monthly fixed payment based on the chosen package).

The solution offered by AWS is not ideal for this thesis. E-commerce portal selling FreeNest service will offer only one service. Different pricing for a different server specification will be, however, an important part of its pricing.



For region **US East (Virginia)**

EC2 Instance Type	Software	EC2	Total*
Standard Large (m1.large)	$0.12/hr	$0.24/hr	$0.36/hr
Standard XL (m1.xlarge)	$0.24/hr	$0.48/hr	$0.72/hr
High-Memory XL (m2.xlarge)	$0.20/hr	$0.41/hr	$0.61/hr
High-Memory 2XL (m2.2xlarge)	$0.41/hr	$0.82/hr	$1.23/hr
High-Memory 4XL (m2.4xlarge)	$0.82/hr	$1.64/hr	$2.46/hr
High I/O 4XL (hi1.4xlarge)	$1.55/hr	$3.10/hr	$4.65/hr

FIGURE 6: *Example of the pricing in the AWS (official AWS website)*

Figure 6 shows an example of the pricing system. FreeNest is an open source software and therefore it is available free of charge, which is the reason there is no need to count with the software component of the AWS price. Servers run non-stop and they are paid monthly. One month consists of maximum 31 days, which is 744 hours. Considering that the average solution costs $0.48 per hour (Standard XL), it makes a final price $357.12 per month. The cheapest version for $0.24 per hour (Standard Large) would cost $178.56 per month and the most expensive for $3.10 per hour (High I/O 4XL) would cost $2306.40 per month. The prices for the software part vary a lot, but EC2 component of the price is rather stable.

3.2.2 Atlassian

Atlassian is one of the leading companies offering collaboration and project management tools. They offer transparent pricing model always available on their official web site. Their tools are simple and easy to use. Their development is based on users' feedbacks (Fidelman 2012).

Atlassian offers several different project management tools on their website. Their pricing is based on the number of users, who use the solution. Prices are rather high; however, on the other side, the licence is unlimited. Customers have to pay additional costs for the maintenance period. For example, a licence for a Jira product, which is similar to a FreeNest service, costs with licence for up to 25 users $1 200. The mainte-

nance period for one year is included, however, extension for one more year costs additional $4 200. Customers can pay for plugins, which can extend basic functionality. Plugin for Git repository for 25 users costs for example $1 800. This is an interesting way to make more revenue with a service product.

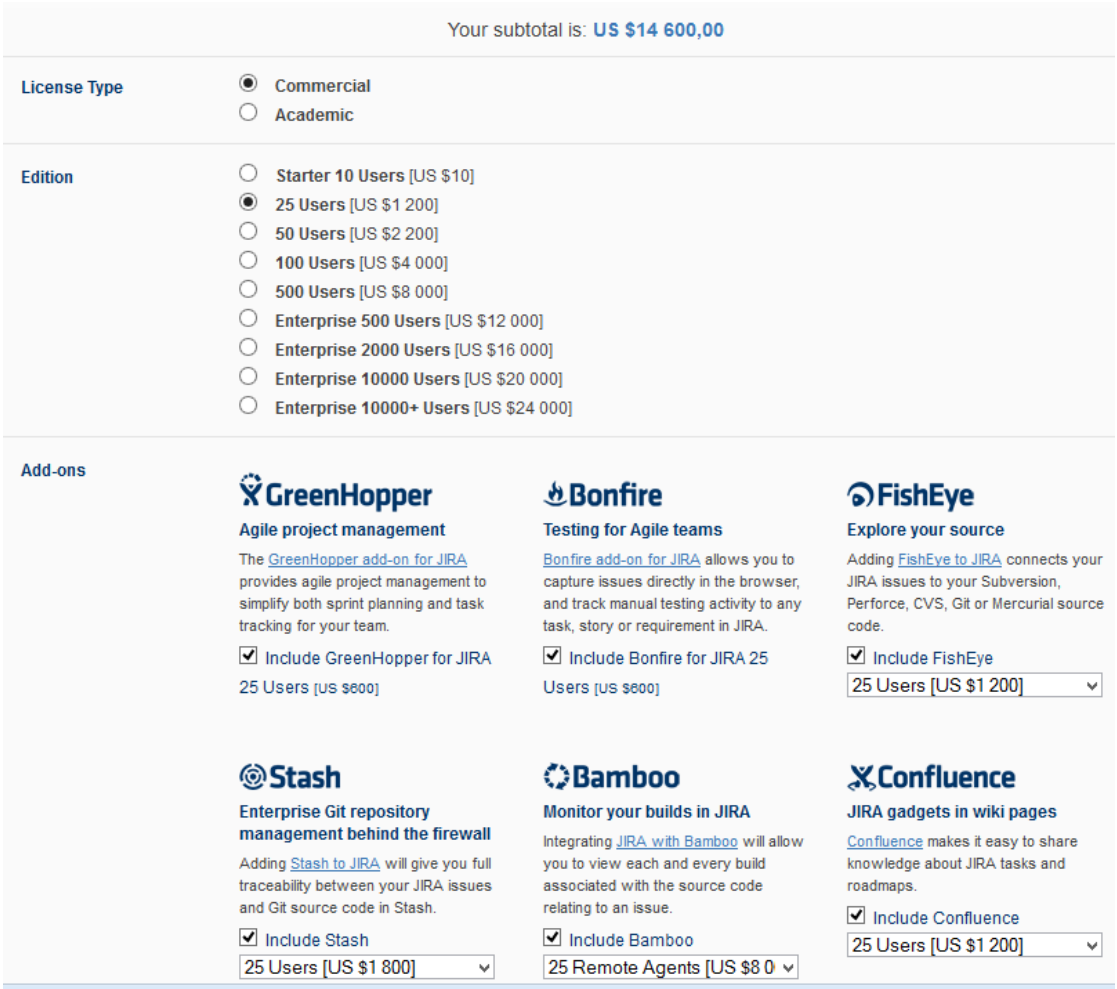


FIGURE 7: *Example of Atlassian's order form (Atlassian's official website)*

Those prices are for a downloadable version and the web site for ordering is shown in Figure 7. Atlassian also offers on-demand version, which is similar to a solution used in this thesis. The prices for this solution are available for registered users only. Because of that, they cannot be compared to the prices of the downloadable version.

3.2.3 Webhosting providers

Webhosting providers may not offer virtual server in a cloud (however, some of them already offer similar type of hosting web pages); however, they usually offer one ser-

vice (web hosting) with different specifications. This fits exactly the needs of this thesis. The e-commerce portal will offer different packages, which differ in their server specification. Design solutions of the webhosting providers may inspire the e-commerce portal selling FreeNest service.

FIGURE 8: *Example of the webhosting page (superwebhosting.sk)*

Figure 8 shows the first example of this type of page. As can be seen, the website consists of three main parts. The first is a header with the logo and name of the service. The right side of the header includes the domain checker. This can be replaced with a login form in case of the FreeNest portal. A large banner dominates the second part of the web page. This banner promotes a service, which is being sold on this page and it should show customers why to choose this provider. Finally, yet importantly, there are four columns with different packages offered. They differ in server specification (space available) and price.

Figure 9 is an example of another webhosting service. The page design is very similar to the previous example; however, it uses less interesting colours. What is the most interesting in this example is the possibility that customers can choose their own specifications. Users can write how much of the disk space they would like to have, how

many emails, FTP accounts and databases. The price is counted immediately according to the chosen specification. This can be interesting for FreeNest e-commerce portal.

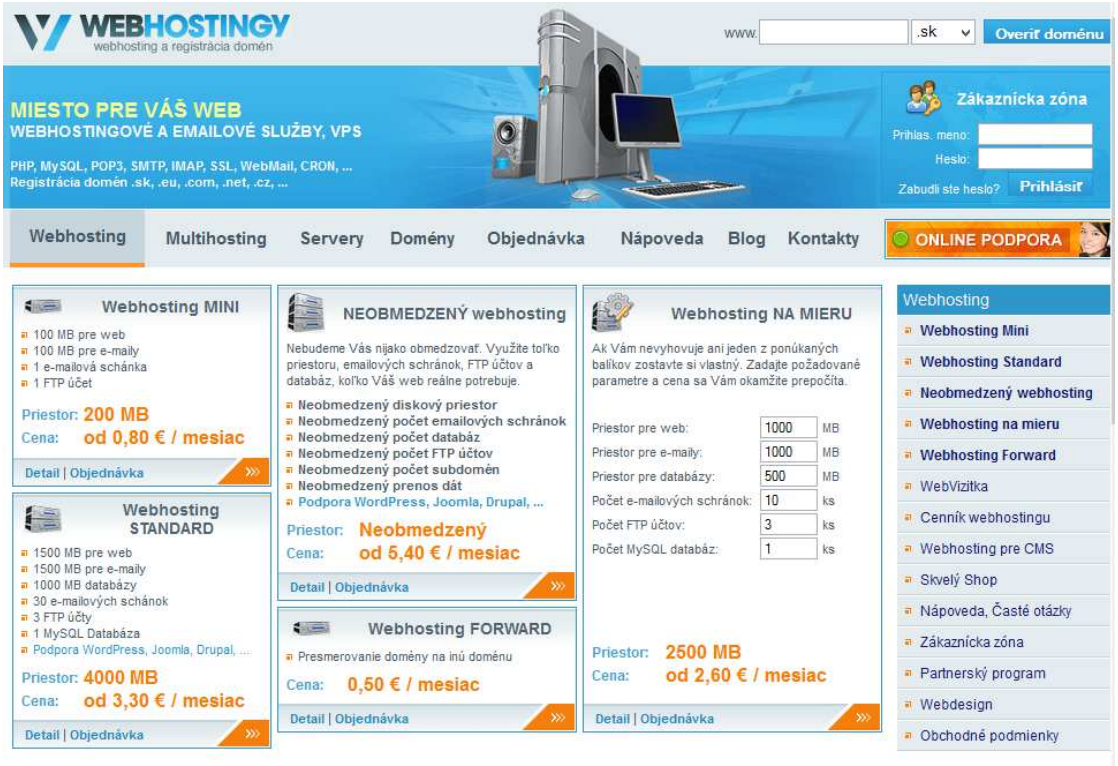


FIGURE 9: *Example of the webhosting page (webhostingy.sk, in Slovak)*

3.2.4 Summary

After a research and analysis of the current solution, the decision to make the page similar to portal offering webhosting services was made. The FreeNest e-commerce portal shall offer few basic predefined packages and one customizable. This will ensure that customers will not be overfilled by many different options to pick from, which might disappoint them and made them look for a different type of service. The pricing system will be similar to the AWS solution and will be based mainly on a server specification.

3.3 Analysis and description of the main functionality

The analysis and description of the main functionality of the FreeNest e-commerce portal is described in this section. This analysis is important for the implementation of either full functionality in the final product or in the proof of the concept, which is a result of this thesis.

3.3.1 Ordering of packages and package management

The ordering of packages and their management are the main feature of this e-commerce portal. Users will be able to order more than one package in case they like to manage more projects with different specifications.

Users can order a package from the index page. If they already have an account, they are redirected to the payment page to verify payment details and process payment. In case they do not have an account yet, they fill in a registration form and will be asked for an account verification through an e-mail. Once the account is verified, they can continue in processing the payment through a link in the admin panel.

Users can order or extend a package licence also through the admin panel. Users can display details of the current packages. Each package consists of the following attributes:

- Automatically generated name, which user can modify anytime to differentiate that package from the others.
- ID of the user to which a package belongs.
- Type of the package (small, medium, large or custom).
- Maximum number of the projects that the package can include.
- Support options that are available for the package.
- RAM and HDD space assigned to each project of the package.
- Number of backups available per project.
- Price of the package.

Package pricing

For the purpose of the proof of the concept and this thesis, following rules for pricing of the packages are to be used:

- Base price for each project in the package will be 30 € per month. This should cover expenses with maintaining service and provide income to the product owner. This should be changed in future according to the real costs.

- Support through forum will be provided free of charge because it is based on users to user help.
- E-mail support will be provided for 30 € per month.
- Chat support will be provided for 60€ per month.
- Phone support will be provided for 100€ per month.
- RAM will be charged by 0.40 € per 512 MB of RAM per month per project. This was decided by comparing the prices in the computer shop alza.sk. 4 GB RAM costs in average about 35 € there. The lowest memory offered is 512 MB, which is 1/8 of 4 GB memory. FreeNest would like the customer to pay those costs in one year. Result of this calculation is 35/8/12 ≈ 0.3646, which can be rounded to 0.4.
- HDD will be charged 0.50 € per 5 GB per month per project. The price of the HDD space of the most used cloud storage providers was checked. Dropbox asks for $9.99 per 100 GB per month and Google Drive asks for $4.99 per 100 GB per month. It was decided to sell 100 GB of the storage space for 10 € per project per month, which makes 0.5 € per 5 GB.
- Backups will be charged by 2 € per backup available per project.

There will be three types of prebuilt packages – small, medium and large. Customers can modify the fourth type of package, custom package, and its price will be counted by the rules above. A description of the packages is shown in Table 1 and pricing of prebuilt packages is shown in Table 2. After a price is counted, it is rounded by mathematical rules.

	Small	Medium	Large	Custom
Projects	1	3	7	1 - 10
Forum support	Yes	Yes	Yes	User's choice
E-mail support	Yes	Yes	Yes	User's choice
Chat support	No	Yes	Yes	User's choice
Phone support	No	No	Yes	User's choice
RAM (in MB)	512	2048	4096	512 – 4096
HDD (in GB)	5	20	50	5 – 100
Backups	1	3	5	0 – 10

TABLE 1: *Specification of the packages*

	Price counted by pricing rules	Used price	Saved money for customer
Small	63 €	59 €	4 €
Medium	209 €	199 €	10 €
Large	527 €	499 €	28 €

TABLE 2: *Pricing of the prebuilt packages*

3.3.2 Project management

Customers with a valid package can create one or more projects. Project is an instance of the FreeNest service inside the cloud environment. Users can choose packages in which a project is created and its name. A request to create a project is then sent to the OpenStack cloud.

Customers can see their projects anytime in an admin panel. They can move them to another package (if they got more) or close and delete it. They can as well manage backups.

3.3.3 Support

An important part of any online solution is support system. Customers will most probably face some problems or they will have questions about services. The way they can contact a product owner is therefore important for their user experience.

3.3.3.1 Forums

Forums are the easiest and the cheapest way of providing support. Customer can ask question and other customers who faced the problem before can answer it. To ensure that every question will be answered correctly, a regular check of the forums by employees is necessary. A major advantage of the forums is that all answers are public and people with the same problem can find them easily without need to contact support on their own. Forums can be implemented through already available external services such as phpBB.

3.3.3.2 E-mail

E-mail offers personal contact, but employee assigned to answering them is necessary. E-mails can be answered within few hours and proper answer can be formed with appropriate information.

Customers can contact e-mail support free of charge in case of questions and problems with the account itself. In case of problems with projects and packages, e-mail support must be included in package.

Customers can contact e-mail support through their own e-mail software or through the form in admin panel.

3.3.3.3 Chat

A chat requires employees to communicate with customers in real time. Problems that are more difficult might be hard to solve immediately because of that. Customers can queue for a chat support. If all operators are busy, the user is presented with the number of people in front of him. As soon as some operator is free, customers can access chat window and send messages to the support.

The support operator can see basic details about customer, previous chats and can modify some basic settings.

3.3.3.4 Phone

The most advanced method how to contact support is the phone. Customers can find contact phone numbers inside the admin panel.

3.4 Analysis of the payment services

This subsection describes and analyses several payment services. Users want to pay for a chosen package safely, fast and through trustable payment option (pay by credit/debit card or bank transfer…). They would like to have a possibility to choose from local payment services if those are available (for example, a Finnish user might prefer payment through a Finnish bank account). For an e-commerce portal alone, it

might be time consuming and financially difficult to make contracts with all payment service providers (banks, card publishers and electronic payment services).

There are plenty of companies, which have contracts with those providers already, and they provide united interface for all of them at one place. FreeNest selling portal will most likely to use one of those services and therefore their analysis and implementation are an important part of this thesis.

3.4.1 Adyen

E-wallet services such as PayPal or Google Wallet were popular and used widely across the Internet, however, a major disadvantage of those services is the need that customers have to have their account. According to Adyen, companies report 35% increase in conversion when they offer a local payment option. Adyen is one of companies, which offer this through support of 218 local payment options. They process online, mobile and point of sale payments for nearly 3000 international customers including Vodafone and fashion chain Mango (O'Carroll 2013).

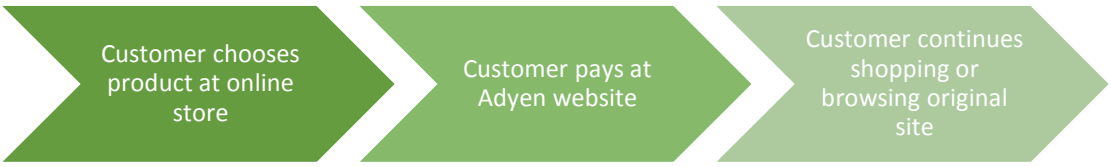


FIGURE 10: *Payment process*

Customers, as shown in Figure 10, start browsing at online store where they choose from available products. Once they choose what they wish to buy, they proceed to payment and they are redirected to Adyen service where they can complete payment through one of available payment options in their country. Customers may be redirected once again to the local payment provider (for example a bank account) if this is necessary for a chosen payment option. When a transaction is completed, customers are redirected back to the online store where they can continue shopping or, in case of our selling portal, sign in and manage their account.

When considering which payment service to choose, price is always an important factor. Because Adyen is an external company, they are charging their customers (other companies) for using their payment services.

Adyen charges a processing fee per each transaction. There are no setup fees, settlement fees, subscription fees nor any other type of hidden fees. That means that the amount of money customers pay for using this service depends only on the amount of transaction and therefore customer will never go into red numbers by using this service.

The fee for every transaction consists of two parts. The first part is a transaction fee. This fee depends on the amount of transactions and decreases with an increasing amount of them. This fee includes services such as risk management, conversion analytics or reconciliations and settlements. Transaction fees are shown in Table 3.

Number of transactions per month	0 - 1 000	1 001 - 5 000	5 001 - 25 000	25 001 - 125 000	125 000 and more
Fee per transaction	0.100 €	0.095 €	0.090 €	0.085 €	0.080 €

TABLE 3: *Adyen's transaction fees (Pricing overview, official website)*

The second part consists of commission fees. Adyen need to pay them to the external payment services, which they got a contract with, for example, bank transaction fees. Examples of commission fees for Finnish payment options are shown in Table 4.

Payment option	Type	Fee per transaction
Bank Transfer	Bank Transfer	0.20 €
Alandsbanken	Online Banking	2.0%
Aktia	Online Banking	2.0%
Handelsbanken	Online Banking	2.0%
Nordea	Online Banking	0.70 €
OP-Pohjola	Online Banking	2.0%
Open Invoice	Open Invoice	Depends on direct contract
Osuuspankki	Online Banking	0.55 €
POP Pankki	Online Banking	2.0%
Säästöpankki	Online Banking	2.0%
S-Pankki	Online Banking	2.0%
Sampo Bank	Online Banking	2.0%
Tapiola	Online Banking	2.0%

TABLE 4: *Adyen's commission fees for a Finnish local payment options (Pricing overview, official website)*

To illustrate those payments with an example, it can be assumed there is a company with the products sold for 100 € each. 120 of them are sold to customers. They pay through different payment options (10 for each, open invoice is not included), which means a gross income of 12 000 €. There are less than 1 000 transactions per month which makes the transaction fee of 0.10 € per each, together 12 €. Eight bank services request 2% fee per each transaction. Eighty transactions for 100 € each makes total fee of 160 €. Osuuspankki and Sampo Bank request 0.55 € per transaction, which means a total fee of 11 €. Nordea requests 0.70 €, which is 7 € in total and classic bank transfer costs 0.20 €, which is 2 € in total. The total sum of all fees is therefore 192 €. That is 1.6% of gross income. It is a very good price for a complete, easy and safe payment solution.

3.4.2 Suomen Verkkomaksut

Suomen Verkkomaksut (SV) was chosen as an example of a local payment option provider, which provides a solution to offer web payment options from all Finnish banks and other popular payment options (Suomen Verkkomaksut, official website). A major disadvantage in comparison with Adyen is the limitation to only Finnish payment options and therefore their market is limited to Finland only.

Shopper's journey in here is similar to Adyen. Shopper chooses a product at an online store, and then chooses a payment service at SV website from which a shopper is redirected to the payment option if necessary (for example net bank) and after transaction is completed, he or she is redirected back to online store.

The pricing of the services depends on the country where the company is registered. The price consists of three parts. The first part is a fixed monthly fee, which is 59 € for companies from Finland and Sweden and 89 € for companies from the other countries. The second and third parts are, similarly to Adyen system, transaction and commission fees. They depend on payment option and they are shown in Table 5. Klarna invoice and instalment, and PayPal services are not shown, because their monthly fee is based on a price list of those companies.

Payment option	Transaction fee (FI/SE)	Commission fee (FI/SE)	Transaction fee (other)	Commission fee (other)
OP	0.35 €	0 €	0.50 €	0 €
Danske Bank	0.35 €	0 €	0.50 €	0 €
Tapiola	0.35 €	0 €	0.50 €	0 €
S-Pankki	0.35 €	0 €	0.50 €	0 €
Säästöpankki	0.35 €	0 €	0.50 €	0 €
Nooa	0.35 €	0 €	0.50 €	0 €
POP	0.35 €	0 €	0.50 €	0 €
Aktia	0.35 €	0 €	0.50 €	0 €
Alandsbanken	0.35 €	0 €	0.50 €	0 €
Handelsbanken	0.35 €	0 €	0.50 €	0 €
Nordea	0.50 €	0 €	0.65 €	0 €
Credit/debit card	0.35 €	2%	N/A	N/A
Collector invoice and instalment	0.35 €	3%	0.35 €	3 %
Joustoraha	0.35 €	0 €	0.35 €	0 €

TABLE 5: *Pricing of the SV (Service price list, official website)*

To illustrate this better an experiment can be carried out as with Adyen pricing. There is a product for 100 € again and this time 140 pieces are sold (10 for each payment option). The gross income is 14 000 €. FreeNest is a Finnish product so we will count with fees from the second and third column. Thirteen services have a transmission fee of 0.35 € and one 0.5 €, which makes total transaction fee 50.5 €. In addition, credit cards also have a commission fee of 2% and collector invoice and instalment 3%. That is 50€ in total for commission fees. We have to pay a monthly fixed payment as well, which is 59 €. The total sum of all fees is 159.5 €, which is about 1.14% of total income. Fees would be much higher with a company from the other country than Finland or Sweden.

Additionally to those fees mentioned in previous paragraphs, SV offers external services, which cause additional expenses. For example extra IDs for a merchant's panel cost 29.50 € per month or a possibility to skip payment method page costs 149 € paid once when this service is requested and 19 € additionally per month. Furthermore, non-stop support is paid monthly and price depends on contract with SV.

3.4.3 Summary

The examples used to illustrate Adyen's and SV's payment fees are, of course, not fully reliable, because customers can prefer one payment service to another. Nevertheless, it seems that SV is a cheaper solution and with a higher amount of payments, the effect of a monthly fee will decrease and because the transmission and commission fees are much lower, the income for the company is higher.

On the other hand, Adyen offers much more payment options worldwide and it is an ideal solution for a global market. SV covers only Finnish market and therefore it is an ideal solution for a local company. Both solutions are implemented in the proof of the concept of the e-commerce portal selling FreeNest service and a detailed report and comparison of this is available in chapter 4.

4 IMPLEMENTATION

Following chapter provides information about implementation of chosen key parts of proof of the concept together with code samples.

4.1 Design of index and admin panel

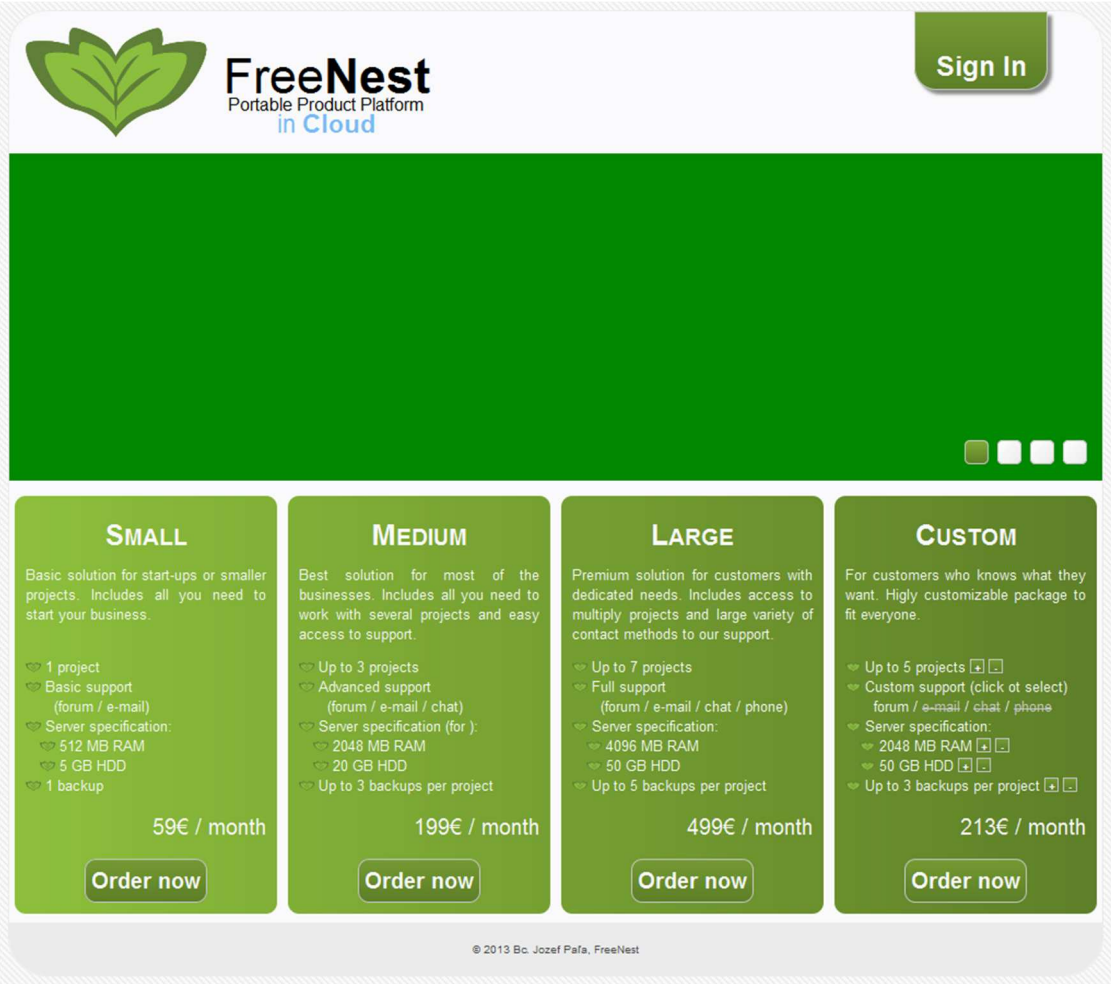


FIGURE 11: *Screenshot of the index page*

The index page is designed to follow design of the official FreeNest web page. Header includes logo and title describing the web page. Users can proceed to singing in or signing up through the green button on the right side of the header.

Middle part consists of the large banner, which is showing four different images, which circulate each seven seconds. This banner has just green colour in the Figure 11 and it should be replaced with the real banner in the final version.

Bottom part consists of four columns and each presents attributes of one package. Last column is offering custom package and data can be changed according to users' will and the price is calculated immediately with each change through JavaScript code.

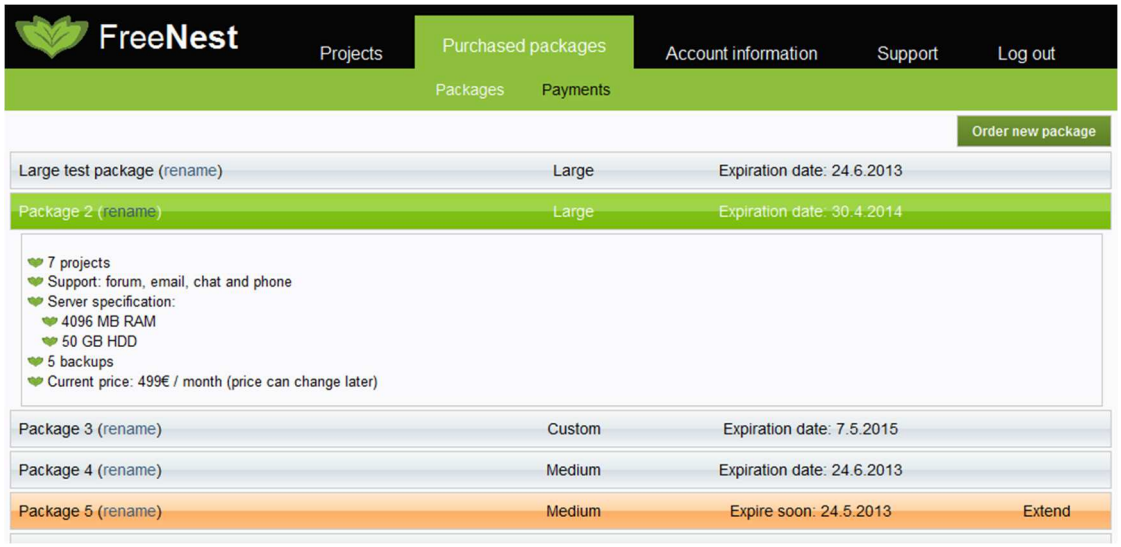


FIGURE 12: *Screenshot of the admin panel*

Admin panel is the place where customers can manage their instances, packages or access support pages. Unlike the index page, admin panel is inspired by interface of the FreeNest tool.

Header is dominated by the logo and menu. Selected tab is coloured green. If the selected menu tab consists of submenus, green bar with submenu tabs appears.

Figure 12 shows view of package list, where are information about all purchased packages, their expiration date and their details displayed. List of projects follows similar design pattern. "Payments" submenu includes table with all payments registered in the system together with their status. Account information consists of simple form where user can fill additional data such as address or phone number.

Support consists of four submenus, one for each type of support. Forum tab offers place where users can post publically their problems and other users can answer them. Large form to send e-mails dominates the e-mail tab. Chat tab consists of window, which offers to customer to queue for the chat support. Once support employee is ready to chat with customer, chat window will appear where customer can post chat messages or read response from the support. Last tab, the phone tab, consists only of

simple table showing numbers to customer centres in different countries. If there is information about additional costs available, it is presented to the customer. Images of the country flags used in this thesis are free to use for the non-commercial purpose, which this proof of the concept matches, however, they should be replaced in the final version.

4.2 Management of users

The way to manage users and their data including the way to store users' passwords securely is very important for the user's experience. If users are not satisfied by the security level, they do not trust the service.

All data of users are stored in MySQL database and passwords are hashed. Table 6 provides information about attributes and structure of the table. Names of the attributes are chosen to be self-explanatory.

Attribute	Data type	Additional information
id	integer	Primary key, auto increment
email	varchar (100)	
password	varchar (128)	
salt	varchar (128)	
name	varchar (150)	
address_line_1	varchar (150)	
address_line_2	varchar (150)	
zip_code	varchar (10)	
city	varchar (150)	
state	varchar (150)	
country	varchar (150)	
phone_number	varchar (30)	
active	integer	
reg_ip	varchar (39)	
reg_date	timestamp	Default value: current timestamp

TABLE 6: *Attributes and structure of the table "users"*

Data in database, especially data about users and their passwords, are very confidential. Safe manipulation with them and protection against intruders has to be first priority and if any intruder breaks into the system, system has to limit harm, which can be done. One of the first steps to do so is creation of MySQL user with limited access to

only necessary operations and tables. This user can select and insert data into the following tables: users, login_attempts and auth_keys. Additionally, this user can update chosen columns in table “users”. Table “users” was described in Table 7, other two tables will be described further in the following text. Code below shows MySQL commands to create this restricted database user.

```
CREATE USER 'login_reg_user'@'localhost' IDENTIFIED BY 'password';
GRANT SELECT, INSERT
      ON selling_portal.users TO 'login_reg_user'@'localhost';
GRANT SELECT, INSERT ON selling_portal.login_attempts
                     TO 'login_reg_user'@'localhost';
GRANT SELECT, INSERT ON selling_portal.auth_keys
                     TO 'login_reg_user'@'localhost';
GRANT UPDATE (email, password, salt, name, address_line_1,
              address_line_2, zip_code, city, state, country,
              phone_number, active)
      ON selling_portal.users TO 'login_reg_user'@'localhost';
GRANT UPDATE (used, used_ip)
      ON selling_portal.auth_keys TO 'login_reg_user'@'localhost';
```

Table “login_attempts” store all attempts of any user trying to log in. It tracks ID of account into which user was trying to sign in, time when that happened, IP address from which attempt was made and if attempt was successful or not. This table is checked each time user tries to sing in. If there were more than five unsuccessful attempts in the last 15 minutes, either from the same IP or into the same account, the incoming attempt is refused. This is to eliminate brute force attacks. In the future, captcha can be shown instead of temporary locking of the account.

Table auth_keys is used to store keys to validate account or reset the password through the e-mail. Each time user creates account or requests reset of the password, key is generated and sent to the user’s email in form of the hyperlink. Hashed form of the key is stored to the database together with time when key was created. When user access link in the email, key from the link is compared to the database to ensure authentication of the user and according to the request, portal activates user’s account or offers form for change of the password. Keys are valid just certain amount of time. It is 48 hours in case of the registration key and 6 hours in case of the key for reset of the password.

User’s data are processed by PHP script and information about if user is logged in is stored in the session. To create secure session, custom function, shown in code below, is used. This function, for example, forbids access to the session ID cookie file

through JavaScript and eliminates possibility of hijacking of the session, because session ID is restored each time function is called.

```
function sec_session_start() {
  $session_name = 'selling_portal_session';
  $secure = false;
  $httponly = true;
  ini_set('session.use_only_cookies', 1)
  $cookieParams = session_get_cookie_params();
  session_set_cookie_params($cookieParams["lifetime"],
        $cookieParams["path"], $cookieParams["domain"], $secure,
        $httponly);
  session_name($session_name);
  session_start();
  session_regenerate_id(true);
}
```

The PHP function to sing in user consists of several steps. User first fills in form in the index page. When form is submitted, AJAX script sends e-mail and already hashed password to the server. Password is hashed first time on the client's side. This ensures that password is not sent in plain text. The fact that password is already hashed also allows use of the any character in the password. On the server side, ID, password and salt is selected from the database where e-mail matches the one received through AJAX POST request. If user with the selected email exists, table with login attempts is checked. If there were less than five unsuccessful attempts to access this account or any other account from the same IP address, password received from the user is hashed again together with salt. This hash is compared with the password saved in the database. If they match, session is created and user is considered signed in. Server returns response code and according to it, JavaScript shows error message if something went wrong or redirects user to the admin panel if operation was successful.

During the sign in process, special string is assigned to the session. This is hashed value, which consists of user's password and browser name. This is used as a signature of the session and is checked each time when we need to find out if user is signed in.

Methods of intrudes develops with time and therefore it is necessary to keep an eye on trends in web security and update solutions used there with new technologies when possible.

4.3 Implementation of payment services

Following section consists of the explanation how to implement Adyen and Suomen Verkkomaksut payment solutions together with conclusion, comparison and comments about both of them.

Before explaining the implementation of each separately, there will be description of the common parts. Tables “packages” and “payments” are the most important in the processing of the payment. Structure of those tables is shown in Tables 7 and 8.

Attribute	Data type	Additional information
id	integer	Primary key, auto increment
name	varchar (150)	
type	integer	
user_id	integer	
projects	integer	
support_forum	integer	
support_email	integer	
support_chat	integer	
support_phone	integer	
ram	integer	
hdd	integer	
backups	integer	
price	integer	
valid_until	timestamp	Default value: current timestamp

TABLE 7: *Attributes and structure of the table “packages”*

Attribute	Data type	Additional information
id	integer	Primary key, auto increment
package_id	integer	
period	integer	
used	integer	Default value: 0
payment_status	varchar (30)	
valid_until	timestamp	Default value: current timestamp
adyen_psp_reference	bigint	Default value: NULL
adyen_payment_method	varchar (200)	
sv_reference	varchar (15)	
sv_payment_method	integer	Default value: -1
hidden	integer	Default value: 0

TABLE 8: *Attributes and structure of the table “payments”*

When new package is created, data are inserted into table “packages”. Three pre-built packages have those data fixed, but custom package can vary and it is important to keep track of the package for each user. In addition, if the content of pre-built packages changes in the future, old packages are not affected. Together with the new package, first payment is inserted into database. Total price is counted as period * price. If period is 12 (twelve months = one year), customer gets discount and price is counted just for 11 months. User can pay in two days and payment becomes invalid after this period. Hidden column is used in payment list in admin panel, where users can hide payments from the list.

FIGURE 13: *Payment summary page*

Once the payment is created, customer is redirected to the payment page. Customer can also access this page by clicking hyperlink in the admin panel. This page includes summary of the payment and logic for both payment services (Adyen and SV). Design of this page is shown in Figure 13.

In the final version, there should be only one “pay” button, however, this proof of the concept implements two different payment solutions.

4.3.1 Adyen

Adyen offers many possibilities to web developers; however, just the most important will be mentioned there. Implementation follows Integration Manual v1.73 and Skin Creation Manual v1.18, both available online at Adyen's support pages.

Adyen offers payment solution through hosted payment pages. Users are redirected to the Adyen's website, where they pay for the product. Necessary data are set in the hidden form, which is sent to the Adyen's website using POST method. All necessary form fields are written and described in Table 9. There are more optional data, which can be sent and they can be found in the official manual. HTML code of the hidden form is shown below the table.

Attribute	Description
merchantReference	Payment's reference assigned by the store. This will be also sent in all later notifications connected with this payment. Payment ID is used in this case.
paymentAmount	Amount of money, which need to be paid, in minor units (without decimal separator). For example 199 € is sent as 19900.
currencyCode	The three-letter capitalised ISO currency code. In this case it is "EUR".
shipBeforeDate	This is not important in this case, because no product is shipped. Time 7 days after page is generated is used. Format of the time is YYYY-MM-DD.
skinCode	ID number of skin used for this payment. More information about skins is available below.
merchantAccount	Merchant account through which payment should be processed.
shopperLocale	Language, which should be used in the payment pages. Default is en_GB (optional).
orderData	HTML code, which will be displayed at the payment pages, GZIP compressed and Base64 encoded. This allows, for example, showing of the payment summary including images or another interactive objects (optional).
sessionValidity	The final time by which a payment needs to be paid. In this case, it is payment validity time from the database.
merchantSig	Merchant signature to verify the request. Explanation about how to compute it is shown below the table.
shopperEmail	Email of the shopper (optional).
shopperReference	Shopper's reference. In this case, it is user's ID (optional).

TABLE 9: *POST data, which are necessary for the payment request to Adyen*

```
<form
    name="adyenForm"
    id="adyenForm"
    action="https://test.adyen.com/hpp/pay.shtml"
    method="post">
  <input
      type="hidden"
      name="merchantReference"
      value="<?php echo $merchantReference; ?>" />
  <input
      type="hidden"
      name="paymentAmount"
      value="<?php echo $paymentAmount; ?>" />
  <input
      type="hidden"
      name="currencyCode"
      value="<?php echo $currencyCode; ?>" />
  <input
      type="hidden"
      name="shipBeforeDate"
      value="<?php echo $shipBeforeDate; ?>" />
  <input
      type="hidden"
      name="skinCode"
      value="<?php echo $skinCode; ?>" />
  <input
      type="hidden"
      name="merchantAccount"
      value="<?php echo $merchantAccount; ?>" />
  <input
      type="hidden"
      name="shopperLocale"
      value="<?php echo $shopperLocale; ?>" />
  <input
      type="hidden"
      name="orderData"
      value="<?php echo $orderData; ?>" />
  <input
      type="hidden"
      name="sessionValidity"
      value="<?php echo $sessionValidity; ?>" />
  <input
      type="hidden"
      name="merchantSig"
      value="<?php echo $merchantSig; ?>" />
  <input
      type="hidden"
      name="shopperEmail"
      value="<?php echo $shopperEmail; ?>" />
  <input
      type="hidden"
      name="shopperReference"
      value="<?php echo $shopperReference; ?>" />
</form>
```

Merchant signature is mentioned in the Table 9. It is hashed authentication code counted from the string made by connecting all the data sent through the hidden form together with the shared secret, which is known only to the merchant. This ensures that data cannot be malformed by the customer. If customer, for example, lowers price, authentication code calculated on the Adyen's side will not match the code sent from the payment page. Part of the code, which is computing this signature is shown below. If any of the parameters in signature is missing, empty string is used instead.

```
$hmacData = $paymentAmount.$currencyCode.$shipBeforeDate;
$hmacData .= $merchantReference.$skinCode;
$hmacData .= $merchantAccount.$sessionValidity;
$hmacData .= $shopperEmail.$shopperReference;
$merchantSig = base64_encode(
      hash_hmac('sha1',$hmacData,$sharedSecret,true)
      );
```

When this hidden form is submitted, user is redirected to the Adyen's website where customers can choose from the available local payments in their country. This website is customizable through the skin definition. Adyen allows creating of the several skins, because one account can manage several internet stores and each of them can use different payment page design. Skins are the major advantage of the Adyen, because users can feel like they are still on the merchant's web page, even they were redirected to the Adyen's payment system.

Developers have quite large set of tools they can use to modify skins. Skin is the zipped folder, which consists of five subfolders with all the necessary resources and source codes. Subfolder "css" defines styling of the payment page. The main style sheet is screen.css. Optionally, developer can modify print.css to styling print output of the web page and screen_ie6.css to support proper styling of the web page also in Internet Explorer 6. Subfolder "img" consists of all images, which are displayed on the payment page.

The HTML code, which is included in the payment page, can be defined in the "inc" subfolder. Figure 14 shows layout of the page. Blocks cheader, pmheader, pmfooter and cfooter are loaded from the "inc" subfolder. The HMTL code has to be written in the txt file with name of the block, for example pmheader.txt. Developers can create

different HTML files for all localizations they support. For example, file for the German localization would be called phheader_de.txt.

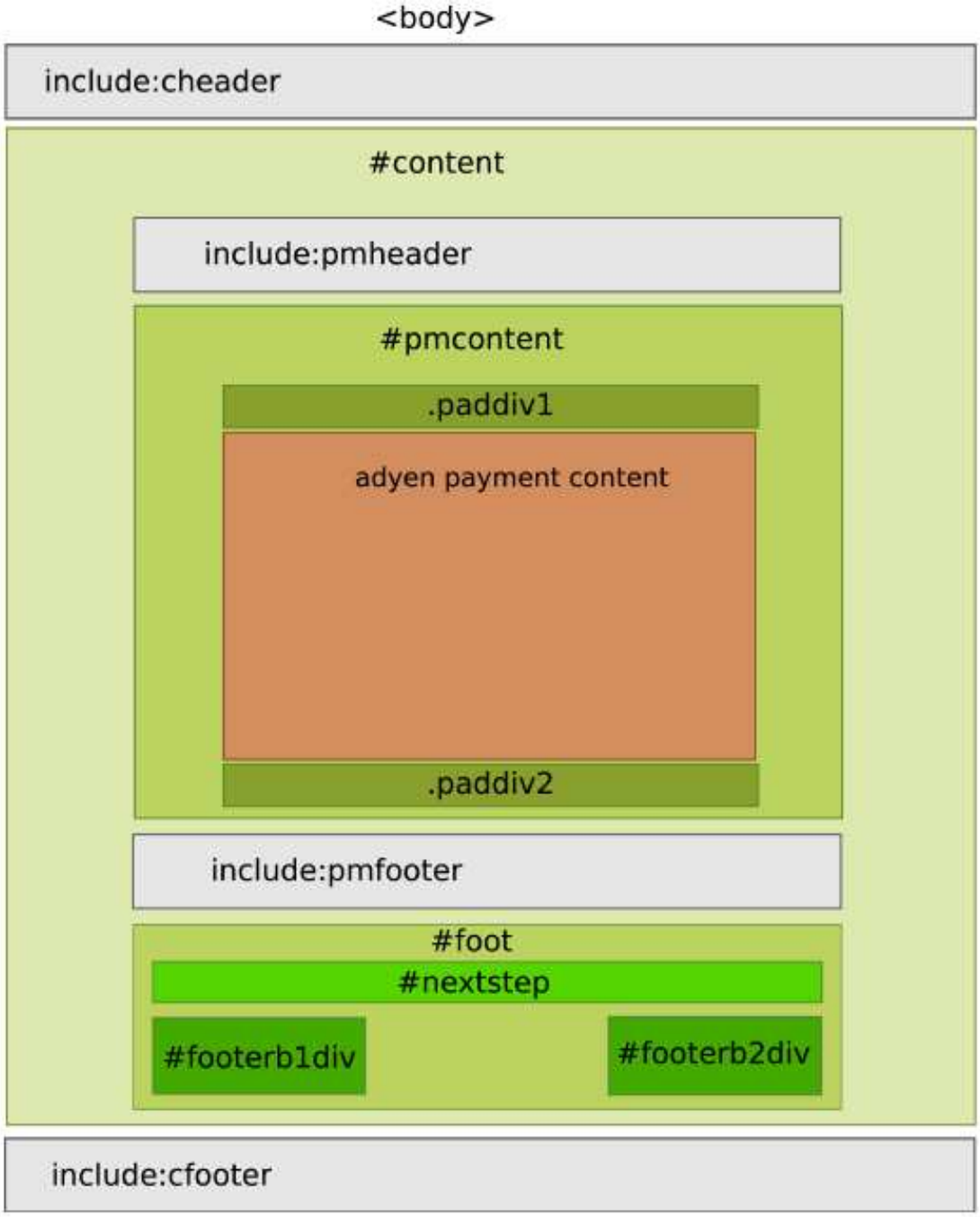


FIGURE 14: *Structure of the Adyen's payment pages*

Localization can go much further and the developer can translate all the texts displayed on the payment pages. Localization files are stored in the subfolder "resources". German translation would be stored in the file resources_de.properties.

The fifth subfolder is “js” and it stores JavaScript source codes. The main file is custom.js and it is included in all the pages. The Figure 15 shows payment page with the default skin and the Figure 16 with the customized skin.

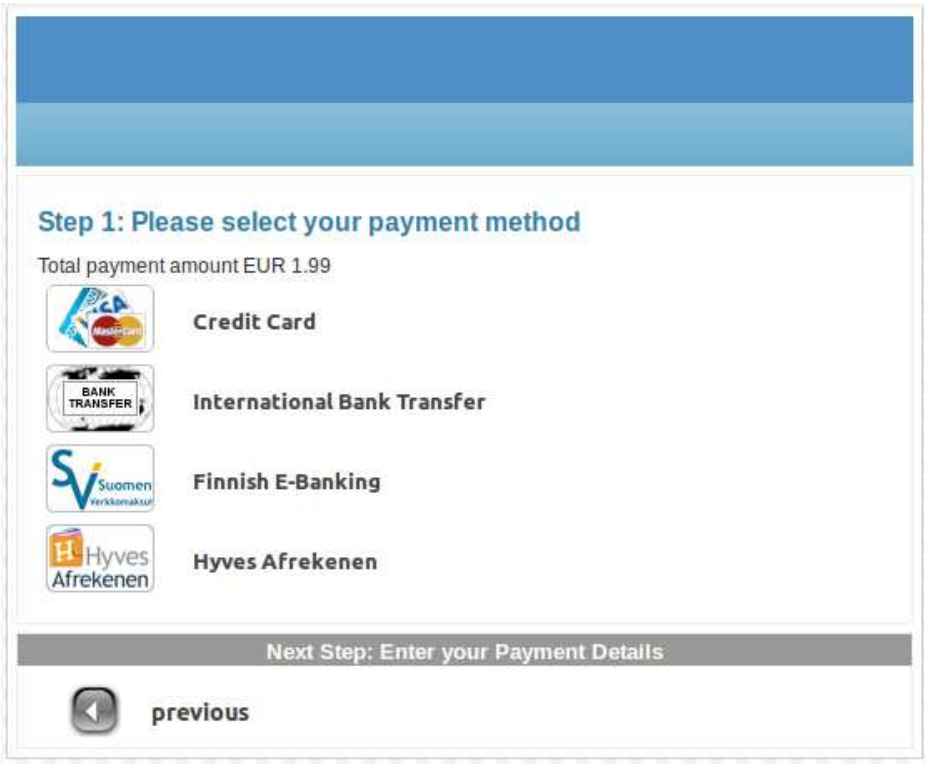


FIGURE 15: *Default design of the payment pages*

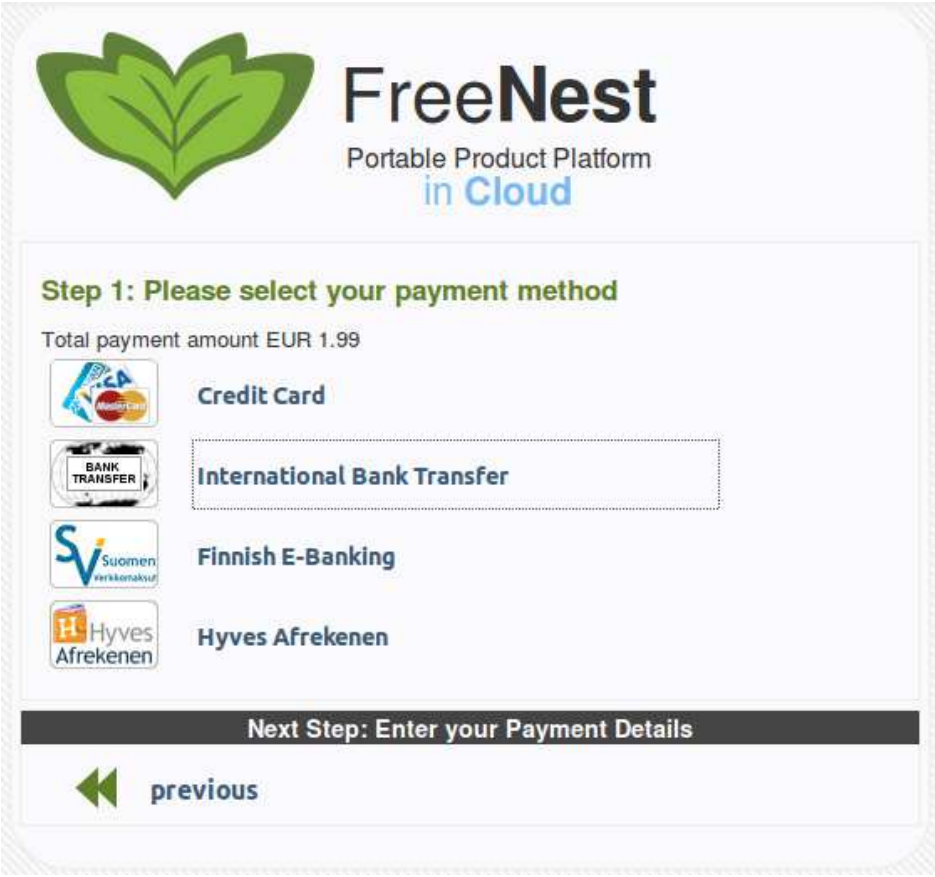


FIGURE 16: *Customized design of the payment pages*

Once the users finish their payments they are redirected back to the store's web pages. Adyen is sending details about the payment in attributes connected to the URL (GET method). They are processed by the e-commerce portal and if payment is successful, payment is accepted and package licence is extended. Table 10 explains those attributes.

Attribute	Description
authResult	The result of payment and it is one of the following: AUTHORISED, REFUSED, CANCELLED, PENDING and ERROR. Authorised does not necessary mean that payment is finished successfully. It means that payment is most likely to be successful; however, user can still refund it. Notification is sent in that case (see below). Pending occurs, for example, during bank transfer, which takes few days.
pspReference	Adyen's globally unique reference to identify the payment.
merchantReference	Merchant's reference of the payment sent with the payment request.
skinCode	Identification code of the skin used for this payment.
merchantSig	Signature to verify data returned from the Adyen. Code to compute it is shown below the table.
paymentMethod	Payment method chose by customer.
shopperLocale	Locale of the customer. It can be used to display message about payment status in the local language of the user.
merchantReturnData	Data passed with the request, returned as-is.

TABLE 10: *GET data passed back from the Adyen's payment pages*

```
$hmacData =
  $authResult.$pspReference.$merchantReference.$skinCode;
$originalMerchantSig = base64_encode(
  hash_hmac('sha1',$hmacData,$sharedSecret,true)
);
```

As mentioned in Table 10, if payment status "AUTHORISED" is returned, it does not fully guarantee that payment will be delivered. Customer still has mechanisms to cancel or refund payment. To catch those events Adyen offers notifications, which are sent to the predefined PHP script. Data are sent through the POST method and their description is written in Table 11. Notification is authorised with user name and password, which is sent through the HTTP request. Those authorisation credentials can be set in Adyen's admin panel. If notification was accepted successfully, PHP script has to return string "[accepted]".

Attribute	Description
live	Boolean value, which indicates if notification is coming from payment made through the test or live server.
eventCode	Notification's type. Those types are described in Table 12.
pspReference	PSP reference of the payment.
originalReference	Original PSP reference in case of a modification request, else blank.
merchantReference	Merchant's reference of the payment.
merchantAccountCode	The ID code of the merchant's account with which the payment or modification was processed.
eventDate	Date and time when the notification was generated.
success	If the event succeeded (Boolean).
paymentMethod	Payment method used.
operations	Modification operations supported by this payment. It is string including one or more of the following values: CAPTURE, REFUND or CANCEL.
value	Amount of the money associated with this payment in minor units (19900 for 199.00 €).

TABLE 11: *Attributes received through the notification*

Event code	Event type	Description
AUTHORISATION	Normal	Payment is in authorisation process. This event is sent even if the payment is refused (because it went through authorisation process).
CANCELLATION	Modification	Merchant requested cancellation of the payment.
REFUND	Modification	Merchant requested refund of the payment.
CANCEL_OR_REFUND	Modification	Merchant requested refund or cancellation of the payment (in case there is not clear if the payment was already captured).
CAPTURE	Modification	Merchant requested to capture payment.
REFUND_REVERSED	Modification	Merchant requested to cancel refund process.
REQUEST_FOR_INFORMATION	Dispute	User filled complaint and information from merchant are requested for this payment.
NOTIFICATION_OF_CHARGEBACK	Dispute	Chargeback is pending, but payment can be still defended by the merchant.
ADVICE_OF_DEBIT	Dispute	Not used currently.
CHARGEBACK	Dispute	Payment was charged back. This notification is not send if the request for information or the notification of chargeback were already sent.
CHARGEBACK_REVERSED	Dispute	Chargeback was cancelled.
REPORT_AVAILABLE	Other	Report of the payment is available in the Adyen's system.

TABLE 12: *Possible event codes in the notification*

In this thesis work, notification script, according to event, either extend licence of the package (if it was not done already) or reverse extension if the payment was additionally refunded or charged back.

4.3.2 Suomen Verkkomaksut

This solution is in the many cases very similar to Adyen's. Implementation follows instruction from the API Description of Payment Gateway document published by Suomen Verkkomaksut.

Similarly to Adyen, payment request is done by submitting hidden form. Attributes of this hidden form are described in Table 13 and HTML code of the hidden form is shown below the table. SV offers two types of the form formats. This thesis is using simpler one. Advanced one requests more detailed information about payment and shopper; however, it is not necessary for this kind of service. Only attributes of the simpler type are described below.

Attribute	Description
MERCHANT_ID	Merchant's ID in the SV's system.
AMOUNT	Price in euros with two 2 decimals separated by a dot (for example 199.00).
ORDER_NUMBER	Payment's ID in merchant's system.
REFERENCE_NUMBER	Automatically generated by default (optional).
ORDER_DESCRIPTION	Additional information to be sent to the payment system (optional).
CURRENCY	Currency code (only EUR is supported).
RETURN_ADDRESS	URL where user is redirected if payment was finished successfully.
CANCEL_ADDRESS	URL where user is redirected if payment was cancelled.
PENDING_ADDRESS	Not in use (optional).
NOTIFY_ADDRESS	URL to accept notification from the SV when payment is marked as paid (optional).
TYPE	Type of the form: S1 (simpler) or E1 (advanced).
CULTURE	Language of the payment page. Currently supported cultures are fi_FI, sv_SE and en_US.
PRESELECTED_METHOD	Only if the payment method is selected already at the merchant's side (optional).
MODE	Default value is 1. Value 2 is used to bypass payment selection at SV's website and requires agreement with SV (optional).
VISIBLE_METHODS	Not in use (optional).
GROUP	Not in use (optional).
AUTHCODE	Authentication code used to authorize payment request. Its calculation is described below.

TABLE 13: *POST data, which are necessary for the payment request through the SV*

```
<form
    name="svForm"
    id="svForm"
    action="https://ssl.verkkomaksut.fi/payment.svm"
    method="post">
  <input
      name="MERCHANT_ID"
      type="hidden"
      value="<?php echo $sv_merchat_id; ?>">
  <input
      name="AMOUNT"
      type="hidden"
      value="<?php echo $sv_price; ?>">
  <input
      name="ORDER_NUMBER"
      type="hidden"
      value="<?php echo $sv_order_number; ?>">
  <input
      name="REFERENCE_NUMBER"
      type="hidden"
      value="<?php echo $sv_reference_number; ?>">
  <input
      name="ORDER_DESCRIPTION"
      type="hidden"
      value="<?php echo $sv_order_description; ?>">
  <input
      name="CURRENCY"
      type="hidden"
      value="<?php echo $sv_currency; ?>">
  <input
      name="RETURN_ADDRESS"
      type="hidden"
      value="<?php echo $sv_success_address; ?>">
  <input
      name="CANCEL_ADDRESS"
      type="hidden"
      value="<?php echo $sv_cancel_address; ?>">
  <input
      name="PENDING_ADDRESS"
      type="hidden"
      value="<?php echo $sv_pending_address; ?>">
  <input
      name="NOTIFY_ADDRESS"
      type="hidden"
      value="<?php echo $sv_notify_address; ?>">
  <input
      name="TYPE"
      type="hidden"
      value="<?php echo $sv_type; ?>">
  <input
      name="CULTURE"
      type="hidden"
      value="<?php echo $sv_culture; ?>">
```

```
  <input
      name="MODE"
      type="hidden"
      value="<?php echo $sv_mode; ?>">
  <input
      name="VISIBLE_METHODS"
      type="hidden"
      value="<?php echo $sv_visible_methods; ?>">
  <input
      name="GROUP"
      type="hidden"
      value="<?php echo $sv_group; ?>">
  <input
      name="AUTHCODE"
      type="hidden"
      value="<?php echo $sv_authcode; ?>">
</form>
```

Authentication code is counted as a capitalized MD5 sum of all attributes merged into one string and separated by a vertical line "|", therefore, this symbol cannot be included in any of the attributes and have to be removed or replaced before sum is calculated and request is sent. PHP code to count authentication signature is shown below.

```
$sv_authcode = $sv_merchat_secret."|".$sv_merchat_id."|";
$sv_authcode .= $sv_price."|".$sv_order_number."|";
$sv_authcode .= $sv_reference_number."|";
$sv_authcode .= $sv_order_description."|";
$sv_authcode .= $sv_currency."|".$sv_success_address."|";
$sv_authcode .= $sv_cancel_address."|";
$sv_authcode .= $sv_pending_address."|";
$sv_authcode .= $sv_notify_address."|".$sv_type."|";
$sv_authcode .= $sv_culture."|".$sv_preselected_method;
$sv_authcode .= "|".$sv_mode."|".$sv_visible_methods."|";
$sv_authcode .= $sv_group;
$sv_authcode = strtoupper(md5($sv_authcode));
```

After hidden form is submitted, user is redirected to the SV's payment page where the payment method is selected. SV does not offer the way to customize this page, however, it can be skipped and selection can be done already on the merchant's side. Agreement with the SV is necessary for this. SV's payment page is shown in the Figure 17.

When the payment is done, user is redirected back to the merchant to the web site defined in the payment request. Attributes sent back to the merchant are described in Table 14.

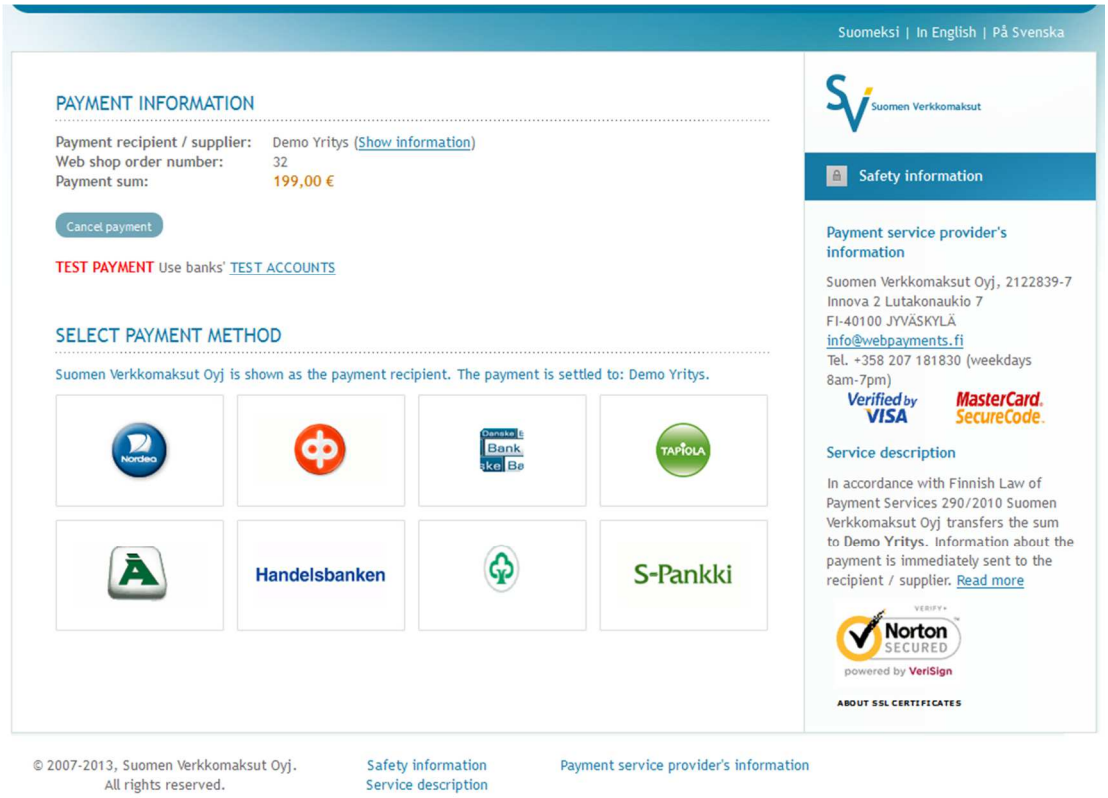


FIGURE 17: *SV's payment page*

Attribute	Description
ORDER_NUMBER	Payment ID in the merchant's system.
TIMESTAMP	The time when the message was generated.
PAID	Payment ID number generated by the gateway. If no number is generated, payment was not successful.
METHOD	Payment method chose by the customer.
RETURN_AUTHCODE	Authentication signature.

TABLE 14: *GET data, which are sent back to the merchant*

If the payment was not successful, attributes PAID and METHOD are not included. Authentication signature is counted the same way as the one sent to the gateway. All returned attributes are included.

Additionally, few minutes after payment is done, notification is sent to the URL defined in payment request. This notification includes the same attributes as those ob-

tained when user is redirected from the payment system back to the merchant's website. Notifications can be used to verify the attributes received when user was redirected back to merchant's store.

4.3.3 Summary

Both payment systems got similar API. Payment request is done by submitting the hidden form. Users then proceed to the selection of the payment option and to the payment itself. This happens on the side of the payment service provider. After payment is done, both providers redirect the user back to the merchant's website and provide payment's attributes through the URL (GET method). Additionally to this, both services offer also notifications about payments, which are sent once the payment state is confirmed or changed.

Both services are easy to use and safe; however, Adyen offers much more options for developers. It is possible to modify and customize the payment pages at the Adyen's servers. Adyen also offer detailed statuses of the payment. SV currently announce only if the payment was successful or not. On the other hand, Adyen offers much wider set of possibilities. Adyen can announce through notifications when payment is charged back or refunded. This allows developer to automate nearly all possible reactions to those states (for example reversing licence immediately when payment is refunded or charged back).

Even the SV might seem to be simpler, it offers advanced options where merchant can pass to the payment service information about VAP or discount and all those are calculated on the SV's servers. SV also offers possibility to skip their pages if the payment selection is done already on the merchant's side, however, individual contract is necessary for this.

SV's solution might be better for the company, which focus mainly on the Finnish market. The fact that the payment page is not customizable can be fixed by the individual contract mentioned before.

Adyen's solution is better for the international company or for the company with many payments so their management can be automated thanks to the various payment statuses.

4.4 Projects

Projects are very important part of the service. Customer is ordering service exactly because of them; however, this thesis is focusing mainly on the payment solutions and reference to design. Regardless of that fact, this section provides ideas about how the final version can be implemented.

Project can be in several states and this state determines the actions, which are available for the user. The states are mostly controlled by the user; however, there are states, which occur when some situation happens inside the OpenStack cloud.

When the project is created, the OpenStack API function to spawn new instance inside the OpenStack cloud is called. New instance is spawned from the universal snapshot, which includes preinstalled FreeNest instance. When instance is spawned, script to set up admin's password can be ran.

When project list is loaded in admin panel, OpenStack status of all projects is checked. There are several statuses the instance can has. The ideal status is "running", however, instance can be spawning, paused, shut down or in error.

The backup can be created from the running instance. This is done by snapshotting it. Running instance and instance in the error state can be restored from the backup. This is done by reimaging the instance from the snapshot.

Running instance can be closed. This can be done by either pausing it or shutting it down. Reopening is done by starting instance again. Closed instance can be also deleted. In this case, instance is snapshotted and after deleted. Deletion can be reversed in next 7 days and instance can be easily restored from the snapshot. After seven days, snapshot and all backups can be deleted.

Projects can be moved between packages. This can be done by rebuilding the instance with the server attributes matching the new package. OpenStack API functions, which can be used, are shown in Table 15 and their full explanation can be found in OpenStack API Reference.

API function	Method	Description
v2/{tenant_id}/servers	POST	Spawns instance.
v1.1/{tenant_id}/os-snapshots	POST	Creates snapshot.
v2/{tenant_id}/servers/{server_id}	DELETE	Deletes specific instance.
v2/{tenant_id}/servers/{server_id}/action	POST	Reboots, rebuilds or reimage specific instance.

TABLE 15: *Examples of the OpenStack API functions, which can be used*

4.5 AJAX API

The important goal is to create dynamic website. AJAX is necessary for this and it is used in many places all over the website. Server scripts, used with the AJAX calls, are located in {web_root}/server_logic/ajax_api/. Description of the implementation of few of them follows in the next paragraphs.

There are two basic types of the scripts. First type returns whole portion of the HTML code and it is used for generating of the website's parts, for example project list, list of the packages or payment list.

Second type returns integer based on the result of the operation. When zero is returned, operation was finished successfully. Other numbers specify error, which occurred. Those codes are used by the JavaScript to show correct error message.

Example of the first type is admin_project_list.php script. This script returns list of the packages formatted with HTML. In case the user is logged out, window with information about that and link back to the index page is returned instead. Code of this script is shown below.

```
<?php
include_once "../session.php";
include_once "../user.php";
include_once "../text_functions.php";
include_once "../db_connect/project.php";

sec_session_start();

if (is_logged($mysqli_login)) { // check if user is logged in
  if ($query = $mysqli_project->prepare(
        "SELECT projects.id, projects.name, projects.project_closed, packages.name,
              UNIX_TIMESTAMP(packages.valid_until), UNIX_TIMESTAMP(),
              UNIX_TIMESTAMP(projects.time_deleted)
         FROM projects, packages
         WHERE projects.package_id = packages.id AND packages.user_id = ? AND
              (projects.time_deleted IS NULL OR
              UNIX_TIMESTAMP(projects.time_deleted)+7*24*60*60>UNIX_TIMESTAMP())")) {
    $query->bind_param('i', $_SESSION['user_id']);
    $query->execute();
    $query->store_result();
```

```
    $query->bind_result($project_id, $project_name, $project_closed,
                        $package_name, $valid_until, $now, $time_deleted);
    while ($query->fetch()) { //for each project
      $table_class_string = "";
      $status_string = "";
      $button_string = "Settings";
      if ($time_deleted != NULL) { //project waits for deletion
        $table_class_string = " class=\"closed\"";
        $status_string = "To be removed";
        $button_string = "Cancel";
      } elseif($project_closed == 1) { //project is closed
        $table_class_string = " class=\"closed\"";
        $status_string = "Closed";
      } elseif($valid_until<$now) { //project is in expired package
        $table_class_string = " class=\"error\"";
        $status_string = "Expired";
        $button_string = "More info";
      } else {
        /*
         * TODO: contacting open stack and ask project status
         */
        $status_string = "Running";
      }
?>
<table<?php echo $table_class_string; ?>>
  <tr id="project_<?php echo $project_id; ?>">
    <td class="project_name">
      <?php echo html_encode($project_name); ?>
    </td>
    <td class="project_package">
      <?php echo html_encode($package_name); ?>
    </td>
    <td class="project_status" id="project_status_<?php echo $project_id; ?>">
      <?php echo $status_string; ?>
    </td>
    <td class="project_settings"
        onclick="project_details.open(<?php echo $project_id; ?>)">
      <?php echo $button_string; ?>
    </td>
  </tr>
</table>
<div class="project_details" id="details_<?php echo $project_id; ?>"></div>
<?php
    }
  } else {
    echo "Database error.";
  }
} else { // if not logged in
?>
<div class="nojs">
  <p>
    You were logged out or tried to access this page without permission.
  </p>
  <p>
    Continue back to <a href="./">index page</a>
    or contact our support at
    <a href="mailto:freenestincloud@gmail.com">freenestincloud@gmail.com</a>.
  </p>
</div>
<?php
}
?>
```

Example of the second type of the AJAX call is creating of the new project. This AJAX script is defined in the create_project.php file. It requires two parameters (name and package_id) passed through the POST method. The script returns values from 0 to 6. Code below shows source code of the create_project.php.

```
<?php
include_once "../session.php";
include_once "../user.php";
include_once "../project.php";
```

```
sec_session_start();
if (isset($_POST["name"], $_POST["package_id"])) {
  $name = $_POST["name"];
  $package_id = $_POST["package_id"];
  if (is_logged($mysqli_login)) {
    echo add_project($_SESSION["user_id"], $name, $package_id, $mysqli_project);
  } else {
    echo 6;
  }
} else {
  echo 5;
}
?>
```

Code below shows how the response from the previous script is handled by the JavaScript.

```
new_project.create = function() {
  if (input_checker.is_blank($("#project_name").val())) {
    error_message.show("new_project_error","Project name is blank.");
  } else if ($("#packages_new_project").val()=="") {
    error_message.show("new_project_error","Package is not selected.");
  } else {
    loading.go();
    $.ajax({
      type: "POST",
      url: "server_logic/ajax_api/create_project.php",
      data: { name: $("#project_name").val(),
              package_id: $("#packages_new_project").val() }
    }).done(function( code ) {
      switch (parseInt(code)) {
        case 0:
          loading.continue_once();
          new_project.close_form();
          project_list.load();
          break;
        case 1:
          error_message.show("new_project_error","Database error.");
          break;
        case 2:
          error_message.show("new_project_error",
                             "Selected package was removed
                              or you do not have permission to access it.
                              Try to reload this page please.");
          break;
        case 3:
          error_message.show("new_project_error",
                             "Package has expired and new projects cannot be added.
                              Extend its licence or choose different package.");
          break;
        case 4:
          error_message.show("new_project_error",
                             "Selected package reached limit of the projects
                              allowed.");
          break;
        case 5:
          error_message.show("new_project_error","Wrong parameters passed.");
          break;
        case 6:
          error_message.show("new_project_error",
                             "You were logged out. Please continue back to
                              <a href=\"index.php\">index page</a>
                              and sign in again.");
        break;
        default:
          error_message.show("new_project_error","Unexpected error.");
          break;
      }
      loading.stop();
    ));
  }
}
```

5 CONCLUSION

The main purpose of this thesis was to create a reference to design and implementation of the e-commerce portal selling FreeNest service with the main focus on the payment services.

This goal was successfully achieved. The current state of the e-commerce portal gives quite good reference to the future final implementation. There are already parts, which can be implemented in the final solution or implemented with a few modifications.

One of the ready to use features is the user management, which is done with focus on the security. To provide even more security, usage of the HTTPS protocol in the future is recommended.

The next nearly finished feature is the package management system connected with the payment services. The current implementation is connected to the test servers of those payment services, however, implementation of the test and live version differ only in details.

Hidden form, in the case of the Adyen service, has to be submitted to the live server instead of the test server. When it comes to Suomen Verkkomaksut, credentials of the merchant's account have to be used instead of the test account.

Project management will need the wider testing and implementation of the connection to the OpenStack cloud; however, good portion of the functionality is implemented and can be used as proof of the concept.

Support services are implemented mostly just as a design idea with dynamic interactivity added with the JavaScript. Support services were not implemented further because of the fact that they are the main goal of a different thesis work.

Especially notable is the dynamic character of the web page. The web page is not fully refreshed each time the user chooses some item in the menu, instead, AJAX calls PHP script, which returns the HTML code and JavaScript then replaces the old content by the new, received through the AJAX call.

The next example, where AJAX is helpful, is when form is submitted. For example, when user rename package, the form is not redirected to the processing page and after back to the previous page, instead the input is sent to the PHP script through the AJAX call and according to response the web content is modified by the JavaScript without any page reload.

The jQuery library allows the web page to provide user-friendly content. Instead of the web page overfilled with the various data and information, additional information is hidden and can be displayed by clicking a specific button. jQuery displays them with the nice animation or effect. Those animations are easily implemented with just a few lines, thanks to the jQuery.

Even the main goals were successfully implemented; there are parts, which could be improved in the final solution. There are numerous code duplications, caused mostly by the time pressure in which this thesis work was done.

Next improvement, which should be implemented, is connected with the AJAX API. Instead of the integer code, there could be JSON string returned by the PHP script. This string would consist of the two attributes. First would indicate whether the operation was done successfully and if not, second would include string value with the error message. This would significantly simplify the JavaScript and AJAX source code and instead of the “switch” with the various error messages, there would be just if-else clause. If the operation was successful, do appropriate action, else display received error message.

Regardless of those weak points, result of the thesis can be considered as an adequate to the former request.

References

About FreeNest, official website of the FreeNest development platform. Accessed on 22 April 2013. Http://freenest.org/about.

HTML5 Introduction, w3schools. Section: “What is HTML5?” Accessed on 22 April 2013. Http://www.w3schools.com/html/html5_intro.asp.

JavaScript Overview, Mozilla Developer Network. Section: “What is JavaScript?” Last update on 23 October 2012. Accessed on 22 April 2013. Https://developer.mozilla.org/en-US/docs/JavaScript/Guide/JavaScript_Overview.

AJAX Introduction, w3schools. Sections: “What is AJAX?”, “How AJAX Works” and “AJAX is Based on Internet Standards”. Accessed on 22 April 2013. Http://www.w3schools.com/ajax/ajax_intro.asp.

Title page, official website of jQuery. Section: “What is jQuery?” Accessed on 22 April 2013. Http://jquery.com.

jQuery Introduction, w3schools. Section: “What is jQuery?” Accessed on 22 April 2013. Http://www.w3schools.com/jquery/jquery_intro.asp.

CSS Introduction, w3schools. Sections: “What is CSS” and “Styles Solved a Big Problem”. Accessed on 22 April 2013. Http://www.w3schools.com/css/css_intro.asp.

CSS3 Introduction, w3schools. Whole article. Accessed on 22 April 2013. Http://www.w3schools.com/css3/css3_intro.asp.

Media Queries, Webflux. Accessed on 22 April 2013. Http://www.css3.info/preview/media-queries/.

PHP Introduction, w3schools. Sections: “What is a PHP file?”, “What Can PHP Do?” and “Why PHP?”. Accessed on 22 April 2013. Http://www.w3schools.com/php/php_intro.asp.

PHP Tutorial, learnphp-tutorial. Section: “How PHP Works”. Last update on 3 January 2013. Accessed on 22 April 2013- Http://www.keithjbrown.co.uk/vworks/php/php_p1.php.

Introduction to SQL, w3schools. Sections: “What can SQL do?” and “SQL is a Standard – BUT…”. Accessed on 23 April 2013. Http://www.w3schools.com/sql/sql_intro.asp.

What is MySQL, 2013. Blog about MySQL. Accessed on 23 April 2013. Http://www.howtomysql.net/what-is-mysql.html.

Gopi, A. 2010. Database System Concepts – Part 1. Section: “Query processing”. Accessed on 23 April 2013. Http://eccentricabhi.wordpress.com/2010/05/21/database-system-concepts-part-i/.

About OpenStack, official website. Accessed on 23 April 2013. Http://www.openstack.org/software/.

Official AWS website. Section: “What is AWS?”. Accessed on 23 April 2013. Https://aws.amazon.com/.

O’Carroll, L. 2013. Crowdfunder Indiegogo partners with Adyen for local card payments (electronic version of The Guardian). Posted on 4 March 2013. Accessed on 25.4.2013

Suomen Verkkomaksut, official website. Accessed on 25 April 2013. Http://www.verkkomaksut.fi/en/.

Service price list, official website of SV. Accessed on 25 April 2013. Http://www.verkkomaksut.fi/en/services/service-price-list.

Pricing overview, official website of Adyen. Accessed on 25 April 2013. Http://www.adyen.com/downloads//Pricelist/Adyen%20Pricing%20Overview.pdf.

Fidelman, M. 2012. Why Atlassian is to Software as Apple is to design. Online version of Forbes. Posted on 4 May 2012. Accessed on 26 April 2013. Http://www.forbes.com/sites/markfidelman/2012/05/04/why-atlassian-is-to-software-as-apple-is-to-design/.

Advantages of using AJAX with PHP. Blog about web technologies maintained by the company, which offers webhosting services. Accessed on 30 April 2013. Http://blog.bounceweb.com/advantages-of-using-ajax-with-php/.

What do Dollar ($) sign significance in jQuery? Hima’s blog about information technologies. Posted on 9 February 2011. Accessed on 30 April 2013. Http://beyondrelational.com/modules/2/blogs/61/posts/11224/what-do-dollar-sign-significance-in-jquery.aspx.

Integration Manual v1.73, Adyen’s official website. Accessed on 2 May 2013. Https://support.adyen.com/index.php?/Knowledgebase/Article/GetAttachment/1301/767532.

Skin Creation Manual v1.18, Adyen’s official website. Accessed on 2 May 2013 Https://support.adyen.com/index.php?/Knowledgebase/Article/GetAttachment/1311/609512.

Suomen Verkkomaksut Oy. 2012. API Description of Payment Gateway. Published on 29 October 2012. Accessed on 4 May 2013. Http://docs.verkkomaksut.fi/files/payment-api-en.pdf.

OpenStack API Reference, the official website. Accessed on 5 May 2013. Http://api.openstack.org/api-ref.html.