

Master Thesis

OSS for booking purposes: Fork of Open-Eshop

Slobodan Josifovic

Barcelona 2014

UPC Supervisor: Xavier Franch

ABSTRACT

This master thesis aims to describe the process involved in the development of a web-based content management system. This work is an extension of a prior project on Open-Source (GPL v3), called Open-Eshop.

This project is motivated by two research questions:

1. How to modify and improve an existing web-based content management system?
2. How to use open source tools, programming languages, and frameworks to reach a final and reliable result?

This project attempts to address the following areas; methodology followed, requirements set, analysis, development and testing stage. It will also discuss the decisions made to define the database model, domain model and some other implementation choices.

The result obtained from this master project is a web content management system, used for booking and renting purposes, but with many other functionalities as well, such as: translations, automatic installation and updates, mails, newsletters, blogs, reviews, custom pages etc.

The aim of this project is a scalable, modern, and extendable system. With the most important contribution for future clients, who are going to use it for free.

ACKNOWLEDGMENTS

Special thanks, I dedicate to the staff of Open-Classifieds in particular José María Chema Garrido, for being my mentor, friend and amazing boss for the past 2 years. To whom I owe so much, for a chance to learn and grow as a professional and his unconditional patience.

I would also like to thank my academic advisor and great teacher Xavier Franch. This thesis would not have been possible without his support and dedication.

And I do not want to forget some Professors, and courses that made me love what I am doing today. ES1 Pablo Casado, DSBW Pere Botella, ES2 Xavier Franch and Claudia P. Ayala Martínez.

Finally, I would like to express my deepest gratitude to my parents Mladen and Slavica, girlfriend Natasha and friend Andrey. Their support and unwavering confidence in my ability, helped me achieve my academic dreams.

1 TABLE OF CONTENTS

Abstract.....	2
Acknowledgments.....	3
List of Tables and Figures.....	10
1 Introduction.....	13
1.1 Background	13
1.2 Kuul-dev booking system	14
1.2.1 What is kuul-dev?	14
1.2.2 Idea behind Kuul-dev bookings.....	14
1.2.3 Vision.....	15
1.2.4 Project Purpose	15
1.3 About Open Eshop (OE)	16
1.4 Motivation	17
1.5 Objective	17
1.6 Thesis Structure.....	18
2 Research and analysis	19
2.1 Guidelines.....	19
2.2 Methodology followed.....	22
2.3 Scrum Cost Estimation.....	23
2.3.1 Initial estimation	24
2.4 Stakeholders	27
2.5 Requirements.....	28
2.5.1 Interface requirements	29
2.5.2 Regulatory/Compliance Requirements	30

2.5.3	Business Requirements	31
2.6	Non-functional requirements.....	32
2.7	System requirements	33
2.7.1	User Space	33
	Home.....	33
	Listing	34
	Advanced Search	34
	Contact	35
	RSS	35
	Login.....	35
	Sign in.....	36
	Single (product view).....	36
	Forums	36
	Topics.....	37
	New topic	37
	Single (product view).....	37
	Blog.....	38
	Single (blog view).....	38
	Custom pages	38
2.7.2	Admin Space.....	39
	Panel Home.....	40
	Market.....	40
	Support.....	40
	Stats.....	40
	Widgets	40

Cache.....	41
Products.....	41
Category and Location.....	42
Bookings.....	43
Blog, Page, FAQ, Forums, Topics.....	43
Newsletter.....	43
Translations.....	43
Themes.....	44
Theme Options.....	44
Menu.....	44
Social Auth.....	44
Settings: General, Payment, Email, Product, Affiliates.....	45
Users.....	45
Roles.....	45
Updates.....	45
Sitemap.....	45
Optimize.....	46
Logs.....	46
PHP info.....	46
Edit Profile.....	46
2.8 Technologies.....	47
2.8.1 Localhost Server.....	47
2.8.2 Backend - Server Side.....	50
2.8.3 Php 5.4.....	51
2.8.4 MySQL.....	53

2.8.5	Front end – Client side.....	55
2.8.6	HTML5	58
2.8.7	Javascript / JQuery.....	60
2.8.8	CSS	61
2.9	Framework – Kohana.....	63
2.9.1	Core Services	64
2.9.2	About Kohana	64
2.10	ORM.....	67
2.10.1	Basic Usage.....	68
2.10.2	Active Record pattern	68
2.11	MVC.....	69
2.12	RESTful.....	73
3	Design and Implementation	74
3.1	General Architecture	74
3.2	Application layer, REST API.....	75
3.2.1	System routes	75
3.2.2	REST api.....	76
3.3	SQL coding standard.....	77
3.3.1	Common.....	77
3.3.2	Table names	77
3.3.3	Fields.....	78
3.3.4	Primary key “PK”	78
3.3.5	Indexes “IK”	78
3.3.6	FK index name	78
3.3.7	Unique index name	78

3.4	Database schema	79
3.4.1	Table “ob_locations”	80
3.4.2	Table “ob_users”	82
3.4.3	Table ob_bookings.....	84
3.4.4	Table ob_product	87
3.5	Installation process	90
3.5.1	User requirements	90
3.5.2	Installation use case	92
3.5.3	Implementation	94
3.6	Update System.....	99
3.6.1	Use case	99
3.6.2	Implementation	100
3.7	Domain layer and API.....	103
3.7.1	Model Product.....	104
	Use case	104
	API	106
3.7.2	Model Category	109
	Use case	109
	Manage categories	110
	Categories widget.....	110
	API	111
3.7.3	Model Booking	116
	Use case	116
	API	117
3.8	Automatic email system.....	120

3.8.1	Available templates.....	120
3.8.2	Newsletters.....	121
3.9	Translations System	122
3.9.1	How it works.....	122
3.9.2	Implementation	123
3.10	Widgets.....	124
3.10.1	Use case Widget.....	124
3.10.2	Use case: Create new widget	125
3.10.3	Implementation	125
3.10.4	Abstract Widget class	125
3.10.5	Helper class.....	127
3.11	Testing	128
3.11.1	Application design testing.....	129
3.11.2	Production environment testing	129
4	Project planning and final cost.....	130
4.1	Initial Time estimation	130
4.2	Final time estimation.....	133
4.3	Cost Estimation	134
5	Future work & Conclusions.....	135
5.1	Future work	135
5.2	Summary	137
5.3	Conclusion.....	137
	Bibliography	138
	Glossary	139

LIST OF TABLES AND FIGURES

Table 1 - Cost estimation	25
Figure 1 - Cost estimation.....	26
Figure 2 - Stakeholders	27
Figure 3 - Sitemap User space UI.....	33
Figure 4 - Sitemap Admin space UI	39
Figure 5 - Flowchart symbols	39
Table 2 - Typical settings of XAMPP configuration	48
Figure 6 - Xampp Control panel	49
Figure 7 - LocalHost Success window.....	49
Figure 8 - PhpMyAdmin Control Panel.....	50
Figure 9 - Php logo	51
Figure 10 - MySQL logo.....	53
Figure 11 - 2011 statistics of screen resolutions.....	55
Figure 12 - Latest market share of most popular web browsers	56
Figure 13 - Responsive design illustration.....	57
Figure 14 - HTML 5 logo	58
Figure 15 - HTML 5 Structure.....	59
Figure 16 - jQuery logo.....	60
Figure 17 - JavaScript logo.....	60
Figure 18 - CSS logo.....	61
Figure 19 - CSS rule set constants	62
Figure 20 - Kohana Architecture	63
Figure 21 - Kohana Cascading File-system	66
Figure 22 - ORM Flow	67
Figure 23 - Active Record Pattern	68
Figure 24 - MVC pattern class structure and cycle.....	72
Figure 25 - RESTful routing chart	73

Figure 26 - General system architecture	74
Table 3 – Api routes.....	76
Figure 27 - Complete Database Schema.....	79
Table 4 - Locations table schema.....	81
Table 5 - Location table Indexes.....	81
Table 6 - Users table schema	83
Table 7 - User table indexes.....	84
Table 8 - Booking table schema.....	86
Table 9 - Booking table indexes	87
Table 10 - Product table schema.....	89
Table 11 - Product table indexes.....	90
Figure 28 - Installation form.....	92
Table 12 - Default Installation variables.....	97
Table 13 - Class Install API.....	97
Table 14 - Class core API.....	98
Figure 29 - Domain layer Model class diagram.....	103
Figure 30 - Product create view	104
Table 15 - Model product API.....	107
Table 16 - Product public action list.....	108
Table 17 - Product private action list.....	108
Figure 31 – Category creation view.....	109
Table 18 - Model Category API.....	112
Table 19 - Model Category public action list	113
Table 20 - Model Booking API	118
Table 21 - Booking Controller actions.....	119
Table 22 - Email templates	120
Figure 32 - Translation edit view.....	122
Table 23 - Translation Controller action list.....	124
Table 24 – Default variable list of Abstract Widget Class	125
Table 25 - Abstract Widget Class API.....	126
Table 26 – Helper Widget Class default variables.....	127

Table 27 - Helper Widget class API	127
Table 28 - Initial plan estimation	130
Figure 33 - Final time planning execution	133
Table 29 - Final cost estimation.....	134

1 INTRODUCTION

In this thesis, I shall describe the process of developing a web-based Content Management System (CMS). To describe this process, we begin by taking a closer look at the technologies used in the development of this system, review the factors of the development stage, and describe the different stages of implementation.

Our purpose is to modify and improve functionality of the existing open source platform made for online commerce. Our modified system aims to help small and medium size companies, not able to spend a lot of money for development. The resulting system created, was a booking content management system called “Kuul-dev Bookings”.

Our focus is on allowing new clients to start or extend their business by renting e.g. rooms, vehicles, flats, restaurant tables etc.

1.1 BACKGROUND

Since 2012, I have been working and contributing to an OSS (open source software) project called Open-Classifieds 2. This project is CMS for creating modern and good looking classified marketing web portals.

I was hired to do a migration, from the old 90s technology to a modern MVC (model view controller) implementation. This was an interesting experience for me, and I learned much since then.

But I could say, I was fortunate enough to work in a very small, successful company,

When the migration was finished, I was part of not just development, but also had many other responsibilities. Such as, Software Architect, Designer, Tester, Customer support, Client interaction, Business advisor etc.

Being able to be part of all these roles, gave me a great perspective to business as a form, in an IT based company.

My goal since then was to learn and grow as a professional who could be able one day to start something on his own. This project is just that, one step towards my life goals.

1.2 KUUL-DEV BOOKING SYSTEM

1.2.1 What is kuul-dev?

What is branding? And why is so important for businesses? Branding goes way beyond just a logo or graphic element. When we think about our brand, we really want to think about our entire customer experience. Everything from logo, application, and social media experiences to the way our customers experiences our product. In short, “brand” is the way the customer perceives us as a whole.

The Kuul-dev, is my idea of using a synonym or branding to future projects as well as towards a company name. Later on, the plan would be to do advertising, marketing, and design around that brand name. For now, as a first step, the decision has been made to name this Master project “Kuul-dev Bookings”.

1.2.2 Idea behind Kuul-dev bookings

The knowledge obtained from developing Open-Classified 2 and Open-Eshop, will help to deliver better results. The main objective of this project is to develop and implement a new OSS web platform for booking, through the forking of an existing project. The most significant reason is to avoid re-development of some basic, but yet time-consuming to implement functionalities.

Some functionalities that we want to avoid are user Authentication and Validation, User space and Admin panel design, external plugin and web service usage etc. The most efficient use of the time, available was to focus on design enhancements, database and domain model modifications, while improving the overall result.

Since I am quite familiar with the “Open-Eshop” platform. With the time saved, the plan is to conduct more research, do a detailed analysis, while defining and documenting mandatory steps in the software development process etc.

1.2.3 Vision

Kuul-dev Bookings is a web based CMS platform for companies or individuals to provide services for booking products. This applies to restaurants, bars, clubs, hostels, rent-a-car, housing, flight booking, ticket selling companies etc.

Nowadays, many small and medium size companies exist that are interested in expanding their business. In fact, they are facing a great cost of development or requested to pay sizable fees / percentages to other companies providing this kind of service. They have very low control of the data, and their progress is limited. Or they have to pay a high cost for licenses that are usually limited to a time duration.

Kuul-dev Bookings offers a solution. The plan is to implement web-based application under total control of the owner. The application will have functionalities of a typical web CMS platform, with the potential to be used as an eCommerce, blogging site or forum.

1.2.4 Project Purpose

The user is able to configure and setup environments using the Admin panel, with many options available to them. They can create products for bookings, which are then publicly available on the user space. Providing the details of particular product, a user is able to read and interact with the product by posting comments, ratings and most important booking products.

Bookings are easy to do, providing just basic information, such as date (from / to). And clicking on the button “Booking”, action is confirmed.

For this version, only existing (registered) users are able to do bookings. Where we insuring the authenticity of customer, adding a small portion of security to our system.

Example: Hostel in center of Barcelona offers rooms. Each room is one product, with xx amount of beds. And client can book this room on xx amount of time. Price applied.

All configurations related to bookings, product, user, emails, translations etc. (As well as general website configurations) can be accessed in the Admin user space. This is a panel that allows the user full flexibility when it comes to the customization of the website.

There are some extra features which will make the site even more complex and functional. Such as Blog page, Forum page, Rating, Comments, RSS, Article / Content insertion, Newsletters, Subscriptions etc.

1.3 ABOUT OPEN ESHOP (OE)

Open Eshop is an online e-Shop platform. Project developed by Chema Garrido CEO, Founder of Open-Classifieds and myself. Started January 2014, and still in progress.

After the huge success of the Open classifieds 2 project and the many digital goods offered as an extension to OC (open classifieds), there was lacking in sales optimization. Before, many different web applications were used for each work aspect. Necessity was one unified place to do all. After many hours spent in researching and investigating software that can help with this problem, many were found, but none of them gave a 100% solution. From that problem, an idea arrived towards an “Open-Eshop” custom implementation to fit the needs of current demand.

After the first beta version, “Open-Eshop” was made public. And now, the complete application helps us run a fully functional digital goods store. The purpose of the software is to give an easy all in one tool to use. Available for anyone who wants to sell designs, music, eBooks, software packages or any digital good they can think of. Also, “Open-Eshop” can be modified to be used for other purposes, with many customization possibilities.

The main issue is, not being able to do any kind of physical goods sales, due to logistics and fraud handling.

The team behind OE is constantly improving the software, working on new releases, adding new features, to make customer experience rich and fulfilling.

1.4 MOTIVATION

Motivation arises from a desire to deploy modern and stable open source software, for self-fulfilment, enjoyment and bring to a closure my studies at the UPC master program. On the other hand, motivation is used to perform and succeed for the sake of accomplishing a specific result or outcome and in the future I may start my own business.

Due to the success of online shopping, this phenomenon is increasingly getting more popular. Private companies are trying to evolve and adapt to follow modern standards. While most of them do not even have a website or a mobile application to satisfy the demand.

Research that was conducted on this topic, have revealed, that there are many booking oriented web-based platforms. (Although, not so many are available as open source solutions.) Open market is in need of small and free tools that can be easily installed, maintained and extended.

1.5 OBJECTIVE

This thesis discusses the following matters.

Research, where we try to briefly explain more about the specific tools, programming languages, back-end and front-end frameworks this software implements. It also defines the requirements analysis, methodology and general structure.

This study aims to uncover long-term goals, (long-term goals are very important and can reveal hidden information). Make sure we understand the scope of the project in terms of technical details (number of pages/screens to be designed, platform/device support, accessibility requirements, third-party integration etc.)

Development, in this section, the discussion will point to the design choices and development process of this project. While keeping pre-set requirements in mind, and following best-practise guidelines of tools and technologies used.

Business model research. This chapter will summarize the ground covered, by discussing the resulting product while making plans for the future work and conclusions

1.6 THESIS STRUCTURE

This brief introduction gives a basic understanding on the topic of this project. The continuation will be presented in the next 4 chapters

CHAPTER 2 - RESEARACH AND PLANNING

This section discusses the hardware and software configuration and requirements, as well as technologies used for this project.

CHAPTER 3 – DESIGN AND IMPLEMENTATION

This section discusses technical solutions. The main objective is to explain the most important sections of the platform and discuss the Database schema, Domain model API and explain some of the core features extending this system.

CHAPTER 4 – PROJECT PLANNING AND FINAL COST

This section describes the initial and final estimation. As well as, the approximate cost of development.

CHAPTER 5 – FUTURE WORK & CONCLUSIONS

This section discusses the conclusions made, and what can be done to enhance it in the future.

2 RESEARCH AND ANALYSIS

2.1 GUIDELINES

1. Getting familiar with the architecture

Software architecture is the most fundamental and most crucial part in the early stage of the development process. Architectural decisions are crucial to the success of a software-intensive project.

Therefore, we need a patient and rigorous decision-making to minimize risks and maximize profits in the late stage. Mistake in this stage, often means losing valuable time and money invested later on.

The best way to avoid early mistakes is to look at the system architecture and learn more about it. In fact learn as much possible, and question decision making. This is the right time to do this. Because, our ultimate goal is to make something new and better.

2. Understanding the code

As mentioned in the previous section, the goal was to learn concepts and understand architecture. However, what about project size? Certainly, any system nowadays, can have thousands of lines of code and hundreds of classes. So, how do we handle this exhausting work? This joke question, explains it all.

“How do you eat an elephant? One bite at a time!”

To conclude, this is an important matter to attend to. Starting development without general knowledge such as where to find something, or when to call some method and or not, understanding relations between two classes. Is a really big problem. This can stretch development time and waste precious resources.

3. Adopting which methodology to follow

Definition: A software development methodology or system development methodology in software engineering is a framework that is used to structure, plan, and control the process of developing an information system.

Although many are available, as can be seen from following list:

- Agile Software Development
- Dynamic Systems Development Model (DSDM)
- Extreme Programming (XP)
- Rapid Application Development (RAD)
- Rational Unified Process (RUP)
- Scrum etc.

It is crucial to pick one and follow it. Companies tend to follow them very strictly. In this project few modifications are needed, due to a lack of an actual team.

4. Requirements

The Software Requirements Specification (SRS) is a comprehensive description of the intended purpose and environment for software under development. The SRS fully describes what the software will do and how it will be expected to perform. An SRS minimizes the time and effort required by developers to achieve desired goals and also minimizes the development cost. A good SRS defines how an application will interact with system hardware, other programs and human users in a wide variety of real-world situations. Parameters such as operating speed, response time, availability, portability, maintainability, footprint, security and speed of recovery from adverse events are evaluated.

5. Database schema

This stage is vital, the aim is to identify the right tables needed and their relationships. The booking process, in theory, can be a very specific or a very broad subject. Depending on what path we pick, it changes drastically.

The main objective is to find the most generalized DB schema, which would be easy to extend in the future. In this thesis, The DB schema was kept very simple, but powerful enough to hold basic data.

6. Testing

The process of software development consists of certain stages. Each stage takes time and is exhausting. Thus, how do we know, that the work done was good? In other words, knowing if a program for specified inputs gives correct and expected results.

Today, various methods and techniques exist.

Test generation involves the analysis of the specifications and determination of which functionalities will be tested, determining how they can be tested, while developing and specifying test scripts. Test execution involves the development of a test environment in which the test scripts can be executed, the actual execution of the test scripts are then analysed and a verdict is assigned about how well the script perform under various tests. In a more detailed view of the testing process, more activities can be distinguished: test organization, test analysis, test specifications, test implementations, test application, test result analysis and maintenance.

2.2 METHODOLOGY FOLLOWED

We use formal methodologies for structuring the delivery of projects. Without a methodology, we would be neither competitive nor successful. There are significant risks and challenges associated with systems integration, particularly on a fixed time or price basis. We assume those risks and challenges only because we have a process in place which, when followed, guarantees success.

On this project development followed simplified Agile Programming method, combined with some aspects of Scrum methodology. Deliverables are submitted in stages and recorded with special tools. Taking into consideration, that this application was developed with a limited amount of people (only 1). The decision has been made to adapt and modify certain working standards.

In the scrum methodology, a sprint is the basic unit of development. Each sprint is preceded by a planning meeting, where the tasks for the sprint are identified. Additionally, estimated commitment for the sprint goal is made, and followed by a review or retrospective meeting. Where the progress is reviewed and lessons for the next sprint are identified. During each sprint, the team creates finished portions of a product.

In the Agile software development method, each iteration involves a team working through a full software development cycle, including planning, requirements analysis, design, coding, unit testing, and acceptance testing when a working product is demonstrated to stakeholders.

The simplified Agile path was more native to follow since the method suggests small portions of the deliverables in each delivery cycle called “iteration”.

This mix of standards may have a serious impact on keeping requirements intact. The impact of this problem may lead to frequent code and database schema modifications. It is important to be careful because not following correct path can affect the development process, and lead to the huge drain in time.

2.3 SCRUM COST ESTIMATION

Estimating the costs of implementation and software development could be a punishing experience.

Why? Requirements change during development, which changes the level of effort and therefore the cost. The operational environment is not, and cannot be completely understood before work begins.

There are different techniques we could use, and on this project, the decision was made to use Scrum cost estimation. Since, there are some advantages we can count on.

Note: Development and implementation were done by one man team. Roles suggested by this methodology were done by me, playing the certain role in the certain time.

In the perfect Scrum proposed methodology, there are procedural patterns every team should follow, and execute proposed strategy as precise possible.

All Scrum projects begin with a story. What does the product owner want, to be able to do, when the project is complete! From that vision, will follow the initial requirements. Scrum cost estimating methods use the wisdom of the team to generate estimated effort, acknowledging from the beginning that final cost and duration will vary as uncertainty, and changing requirements are reflected in the actual development.

When the Scrum team meets for its initial planning, the vision is laid out by the product owner. The team then considers how to create the desired functionality, which are called story points. Each story point is rated for the relative difficulty.

Steps followed in assessing the cost:

1. The product owner describes a user story/feature the team is estimating. The product owner may provide clarification on the feature in response to questions.
2. Each team member chooses a development effort card from a deck of values to represent their relative estimate of level of effort. Cards are placed upside down on the table so that they do not influence others.
3. After all estimates are on the table, the cards are flipped over.
4. If there is a significant range among estimates, the team members who submitted the high and low values provide a rationale for their estimate.
5. Once the discussion on the range has been conducted, steps 2 through 4 are repeated until a consensus is reached.

One of the advantages of this method is that we do not have to estimate the entire project in excruciating detail or even know the complete set of features at the beginning of the project. This is more real world than the traditional “Waterfall” type of project planning and estimating.

2.3.1 Initial estimation

It was important to have an approximate workflow, and overall estimation to get started with a development process. Originally, initial planning was created and stories were defined. Table 1 indicates list of initial stories planned disregarding iterations. To each story was given the approximate number of points. Where points represent relative difficulty to complete the work.

Table 1 - Cost estimation

Story	Estimation Points
Locations integration	2
Sample locations	1
Bookings	40
Different user rights	13
External usage of the bookings	40
Refactor product creation	8
Improved booking creation	13
Improve translation system	8
Refactor update system	3
Refactor install system	8
Email templates	8
Known error fixing	13
Front end Theme refactor	20
Admin panel Theme refactor	40
Payment system	20
PayPal integration	8
Paymil integration	40
Bitcoin integration	40
New widget location	2
Automatic module installation system	40
UnitTests	100

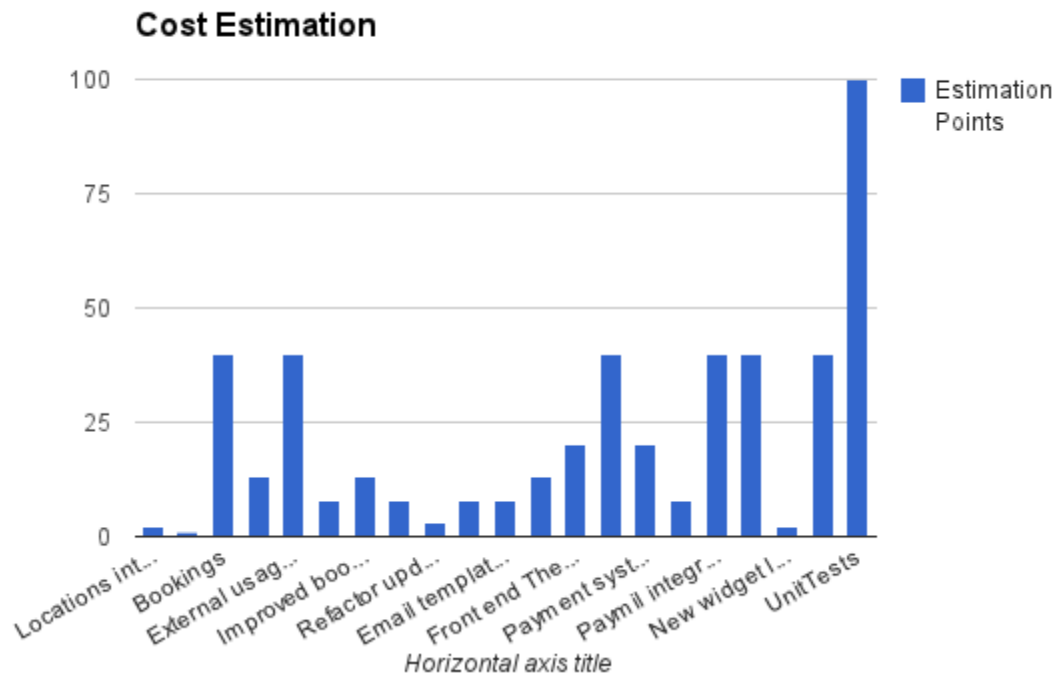


Figure 1 - Cost estimation

After the initial estimation, decision has been made to remove following list of stories, from the initial estimation.

- Unit Testing,
- Automatic module installation system,
- Payment system with all following integrations (PayPal, Bitcoin, Paymil), and
- Admin Panel Theme refactor

The reason behind was a great cost and really high value in the overall scope of the system.

2.4 STAKEHOLDERS

Due to, the nature of the system, as content management system offering bookings. In the future development continues, and it will be extended. We cannot say with 100% of certainty which stakeholders are going to interact with the system. At the moment, we can name following stakeholders:

Day to day Users: They are going to use services provided by this software, depending on its context. This stakeholder has access only to read and use public features.

Owners of the software: This stakeholder can be individual users or company. They are allowed to access private and configure data for its own purpose. Offering their own services, and using their own resources to do business. Owner has a full access to the system.

Moderators of the system: They interact with the system with limited access, only to add additional content (text, images, categories, locations etc.). This is the partial user access right, nothing that can alter settings or private data.

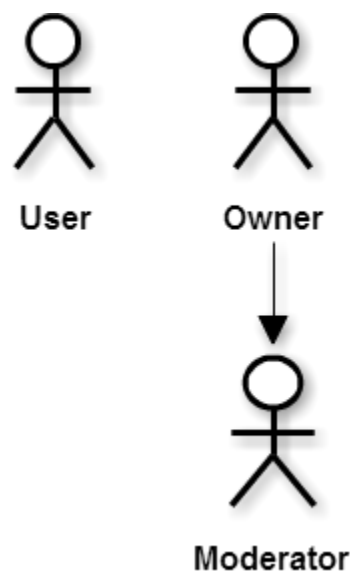


Figure 2 - Stakeholders

2.5 REQUIREMENTS

“Kuul-dev Bookings” needs to allow owners to change, configure and create content. As well as, be able to be “booked” for a certain product in given time period. Booking is then processed and Owner is notified of the latest actions. This software is web-based platform, and for each installation, can be associated with one domain name and database. Users are able to access the software from any portable device as well. Because the theme is going to be responsive and resizes its width and height with known resolution.

Each owner, can without any limits, use their own branding and advertising.

Minimal technical requirements can be seen in the following list:

1. Internet connection
2. Server or localhost machine
3. Apache 2+
4. PHP 5.3+
5. Short Tags
6. GD support
7. mod_rewrite
8. mcrypt
9. Gettext
10. Curl
11. MySQL 5+

2.5.1 Interface requirements

General overview and default views:

1. System must provide a list of booking products, clearly showing name, short description, and descriptive image
2. Home screen need to:
 - Contain appropriate menu on the top with the list of items activated by the software owner. Otherwise default one is used.
 - Contain most recent products or featured product, as well as, list of categories of the system
3. Sidebar, if active, can show widgets with data depending on the functionality
4. Search form is supposed to have following fields: Name (text), Category (select), Price range (numeric)
5. Contact form view:
 - Contains following fields: Name (text), Email (text), Subject (text), Message (text), Captcha (auto generated image + text field)
 - User can send a message to the owner of the software
6. Users are given an easy access to login and log out
 - Register form contains following fields: Name (text) required, Email (text) required
 - Login form contains following fields: Name (text) required, Email (text) required
 - Login, additionally, provides option for users that forgot password and remember me (keeps email in cache to auto fill next time)
 - Social login, by default have uses OpenId and AOL services, since they do not use any extra configuration

Single product view:

- Title or name of the product is clear and visible
- This view must hold all descriptive information in full.
- Contains gallery of images applied to this product, and users are able to see all images on the big screen
- Coupon field is present, less important to notice. So its desired to be close to the footer
- Some additional details are necessary, such as product score, visits etc.
- Booking form is visible and clear
 - Contains only date start and end
 - Selectable dates are only working days, and not past dates
 - Selection of the dates is clear and simple, it is desired to be in the special format similar to actual calendar overview
 - Button that registers the bookings, is clear, big and visible
 - If correct dates are not selected, display the appropriate message
 - If date is not selected, display appropriate error message

2.5.2 Regulatory/Compliance Requirements

- Database
 - Must contain all basic configurations set by user after installation process
 - Must contain sample categories data if selected by user
 - Must contain sample locations data if selected by user
 - Must contain extra configuration data to assure basic and minimal requirements for system to work properly
 - Basic installation will assure only the most common tables for basic usage of the system
- System
 - Will limit access rights to authorized users
 - Owners have full privileges and are allowed to alter system functionalities
 - Moderator, have only rights to add, and edit the content, FAQ, blog, templates, widgets, translations. Without access to users, configs and products.

- Basic user has only permission to view and interact with web-site and do booking if logged
- Will cache most common data to assure increased speed.
 - Sitemap
 - Locations
 - Categories
 - Emails
 - Typical queries
- Users could be registered with social media as well as standard in system registration mechanism
- Emails are automatic, and sent on appropriate system language
- Only user that booked the product, can make a review and give a mark to that product

2.5.3 Business Requirements

- License
 - Software is covered and uses GPL v3 license standard
 - All content even premium is allowed to be extended, and altered by third party companies or users
- Support
 - Free support is given to all users
 - Premium support is given to the users who purchased at least one premium content
 - All the system features will be explained and covered with typical “How To” and FAQ format pages to help users with basic interacting
- Bookings
 - Date must be entered before it can be approved
 - If price applied, booking is in “Pending” state before it is approved active
 - Only registered and logged users can request booking

2.6 NON-FUNCTIONAL REQUIREMENTS

“Kuul-dev Bookings” is web-based and multi-platform oriented. Above all, we are obligation to provide following non-functional requirements:

- Reliability

Reliability is important, because of the direct correlation with our clients and their trust in our application. Also, in the future, it could potentially lower the cost of the support we offer to unsatisfied clients.

- Extensibility

One of the main objectives of this platform, is to evolve, and be able to serve different work-flows.

- Usability

Nowadays, unavoidable NFR. Since, this platform does not require economic means. It could be used by anyone. We can conclude that easy to use and pleasant to work with is of major importance.

- Maintainability

We need to assure the ease with which our platform can be maintained. To allow future maintenance easier, maximize efficiency, reliability, and meet new requirements.

- Documentation

This document will be used to fulfil documentation NFR. And, in future keep documentation updated, when new information's and modifications are added.

2.7 SYSTEM REQUIREMENTS

In this chapter are documented use cases and its definitions. System could be separated in two parts. First is User or Public space, and second is Admin or Private space. Figure 3 shows the sitemap and connections between views.

2.7.1 User Space

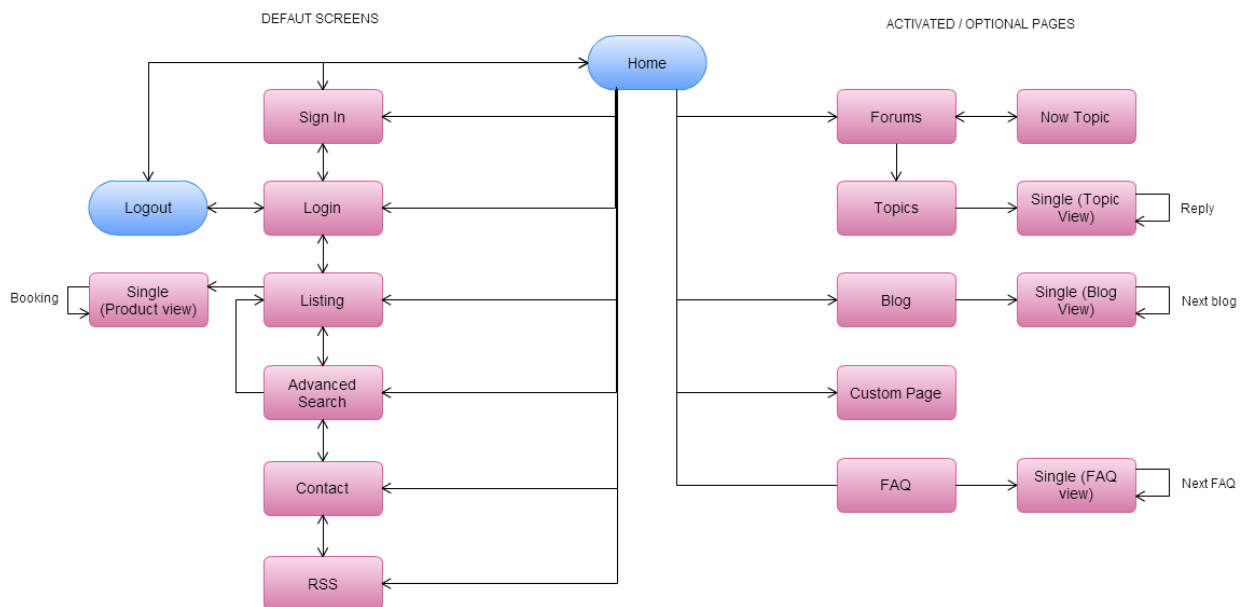


Figure 3 - Sitemap User space UI

Home

This is start screen, it is always the first screen user see when connecting to the application. Contains the list of categories of all system in menu form and as an element. By clicking on one of the categories, users are redirected to “Listing screen” with all products in that category. Also, this screen contains carousel slider with latest products by default.

Note: Each screen has a top menu with links to all existing screens in the system. As well as login button and search bar. And widgets if enabled taking left side of the screen by default.

Listing

This screen contains the list of all products. This list should be available two different formats.

- As the list: each product is wide to occupy space vertically. And highlight text.
- As the grid: more rectangular shape to save space and give more importance to image, rather than text.

This screen could be sorted with few different options:

- Name (A-Z)
- Name (Z-A)
- Price (Low)
- Price (High)
- Featured
- Newest
- Oldest

Each represented product holds short information about the product (by default):

- Image
- Title
- Description

Advanced Search

This screen contains fields for detailed search on product. By default we have:

- Name
- Category
- Price from
- Price to

After the search, list of product will be shown below the advanced search field. Allowing us to refactor search more.

Contact

This is the typical form for sending messages to the Owner of the system. It is used for users who would like to ask questions or send complaints etc. It consist of fields:

- Name (sender)
- Email (sender)
- Subject
- Message
- Captcha (by default)

RSS

This screen prints RSS XML contents, for each product (as <item>), for users to copy and follow the feeds.

Login

User can use this screen to login and gain certain access to the system. There are two main field that the user need to fill in (with the correct input):

- Email
- Password

Users can as well do action “Forgot Password” to change the password if lost. “Remember me” to enable an option that fill in an email every time they access login screen. Also, they can press the button “Register” to swich to Sign in screen.

There is optional login system called “Social login”, which Owner needs to enable and configure. Giving Users way to access the software using some of the social accounts they hold. After selecting one of many social login options, user is then redirected to the corresponding Social network for the further validation. And returned back, after that process is done.

Sign in

User can use this screen to register and create new account gaining certain access to the system.

There are two main field that user need to fill in (with correct input):

- Name
- Email

There is optional login system called “Social login”, which Owner needs to enable and configure. Giving Users way to access the software using some of the social accounts they hold. After selecting one of many social login options, user is then redirected to the corresponding Social network for the further validation. And returned back, after that process is done.

Single (product view)

This screen holds all necessary information of the product Owner offers and is enabled for booking. By default will have images, full description, title, number of hits, rating (if any) and booking form.

Booking form consists of start date, end date and button to confirm booking. After all fields are selected properly, system sends email confirmation with data selected. And on screen success alert message.

If enabled, this screen can contain “Disqus” plugin. This plugin is comment 3rd party service. And allows users to leave comments for a particular product they are looking. Also, if enabled QR code can be generated and printed for each product.

Forums

It is a list of forums, that the owner creates for users of the system to have some topics of discussion and open new topics under particular forum. By clicking on a title of any forum, they are redirected to “Topics” screen.

Each forum have information on:

- Forum name
- Last message (date)
- Topics counter (integer, number of active topics)

Topics

It is a list of all topics created by any type of user of the system. By clicking on the title it opens “Single (topic view)”.

Each topic contains information on:

- Topic name
- Created (date)
- Last Message (date)
- Replies (integer, count)
- Edit button (if Owner is logged in)

New topic

This screen is uses the form for creating new topic. Topics can be created by any user. And by doing so, they are become owners of the particular topic under the forum.

Field required are:

- Forum (select drop down, of existing forums)
- Title (string)
- Description (string)
- Captcha (by default)

After creation user is than redirected to “Single (topic view)”.

Single (product view)

This screen is a list of all discussions over the topic created by some user. Owner of the topic is highlighted and his message visible all the time. If owner is accessing his topic he has an option to edit his topic.

Users are able to reply to him writing down some text. In the case of logged in user, his account will be related to this message. Other case, users need to leave their name.

This is open system of communication between all users.

Blog

This screen is similar to the “Listing” screen. It has all the same options except it is intended for Blogs (articles), written by Owner or Moderator. These articles could be of any type and context. Anything Owner or Moderator considers relevant, no restrictions except they are the only one that can create, edit and delete.

Single (blog view)

This screen is the verbatim copy of the article created by Owner or Moderator. It can contain different text formatting, images, code, links etc. No restrictions.

Moreover, anyone with public access can read them.

Custom pages

Custom pages are an extension of the default and optional views enabled by the system in general. Owner or Moderator can create a new page that will be shown in the footer of the website.

This feature is intended for creating “Terms of use”, “About” etc. pages. In a way personal or not standard blog pages.

2.7.2 Admin Space

Following sitemap (figure 4) defines all screens used in admin space and connections between them. Admin space is the section was Owner and Modifier can change settings and add new content to improve User space. As well as, moderate certain relevant data of the system.

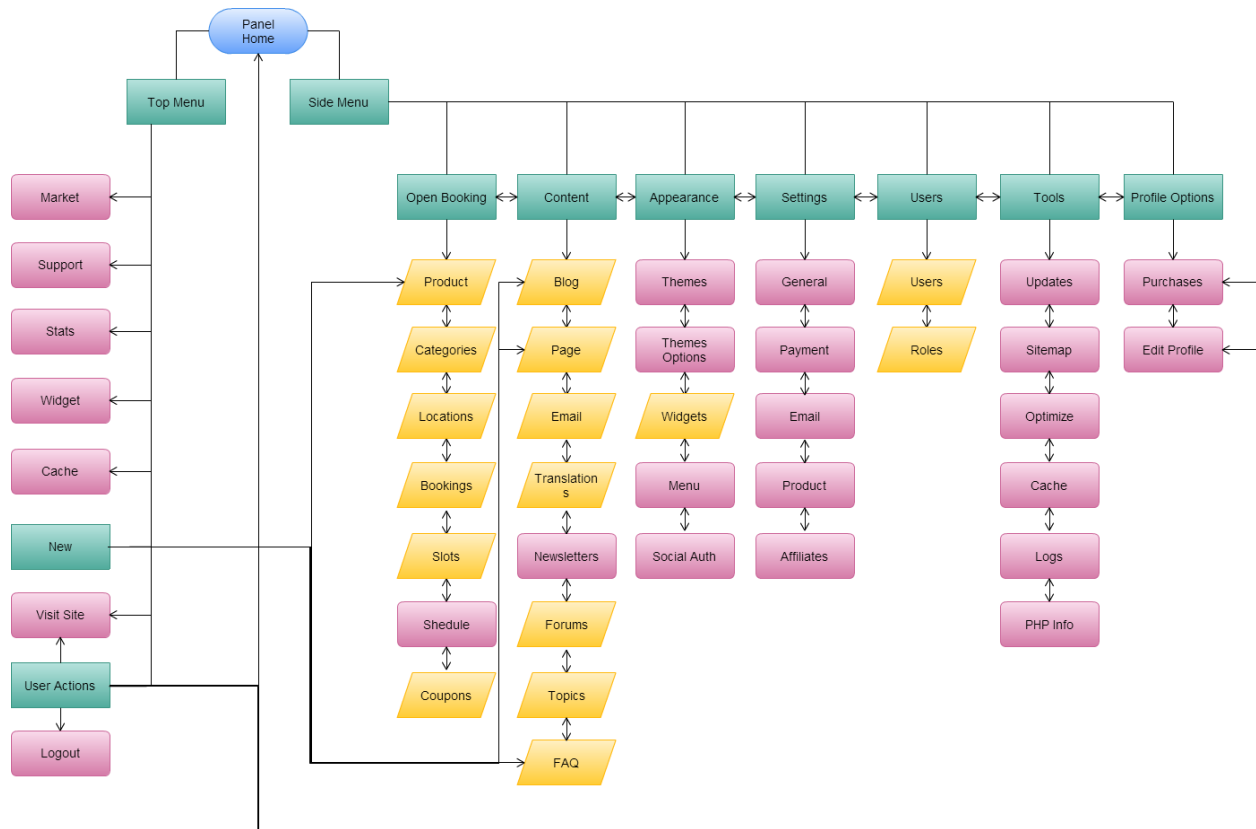


Figure 4 - Sitemap Admin space UI

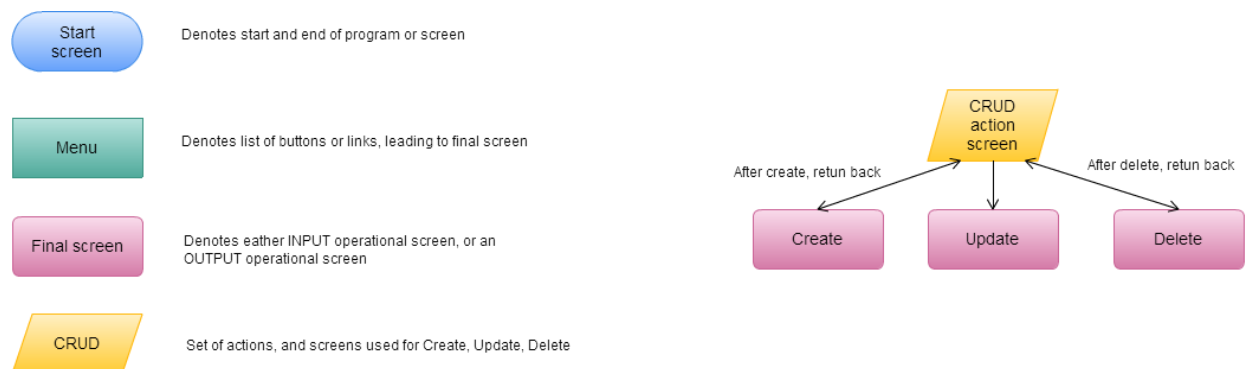


Figure 5 - Flowchart symbols

Panel Home

This is a welcome screen for owners and modifiers. It is the first screen we see in the admin panel. For the first version, only welcome message and some news about “Kuul-dev Bookings” are printed.

Market

This is the list of products printed on the screen, that “Kuul-dev Bookings” offers for sale. Each product redirects user to the portal where that product could be both.

Support

For this version of the application, we do not have this option available. Although, it should redirect users to external web portal, where “Kuul-dev Bookings” are currently giving support to its customers.

Stats

This screen has some charts with actual data happening on the Owners website. Such as an amount of hits, bookings etc. User is able to set some filter conditions in order to see some dates more clearly.

Widgets

This is CRUD action. Containing all widgets offered at the moment. Owner is able to create multiple of the same type and position them in few different placeholder spaces.

- Sidebar
- Header
- Footer
- Inactive (To deactivate for a time period)

Each widget has its own relevant configuration data. But not mandatory. Each widget could be created with default data.

Since its CROD action, each widget could be Created, Updated and Deleted.

Cache

This screen prints information about cached data.

- Config file – root path of the config file is stored
- Driver – file type
- cache_dir – root directory where cache files are stored
- Default_expire – time when they are assumed irrelevant to the system
- Ignore_on_delete – array of cache files ignored on delete

It also contains two buttons

- Delete expired – remove old cache
- Delete all – remove all cache files

These actions are useful when some data has been cached, and user doesn't see his changes. By cleaning cache, system is forced to regenerate them. As well as, when some corrupted cache files are generated.

Products

This screen is a CRUD action. Owner can create remove and delete Products.

The index screen should contain a list off all products with, for each, some basic information.

- Title – name of the product
- Status – is it active?
- URL – link to the product in User space

When creating and editing. Owner will have to fill some information before.

General and required:

- Category
- Currency
- Title
- Description

Details:

- Booking price – standard fee for each booking
- Product price
- Price offer – Reduced price
- Offer valid – date of until offer is valid
- Purchase notes – Sent via email

Files:

- Images – For product view
- Digital File – sent when product is booked

If the owner wants to publish this product he need to confirm it by checking the “Active?” check-box.

After, at least, required information is added user can click save to publish the product.

Category and Location

This screens are CRUD action. Owner can create remove and delete Categories or Locations.

When creating and editing. Owner will have to fill some information before.

- Name
- Order
- Id category parent – Set to belong to other category
- Seoname – or a slug, that would be used for generating the URL
- Description

Bookings

This screen is a CRUD action. Owner can create remove and delete Bookings. As well as change their status. There is few different statuses

- Created
- Paid
- Refused
- Fraud
- Refund

The index screen should contain a list off all bookings with, for each, some basic information:

- User – user that made booking
- Product – booked product, as a ling to actual product
- Coupon – if coupon was used
- Date – date of booking

Blog, Page, FAQ, Forums, Topics

These screens are CRUD actions. Owner can create remove and delete them to fill in some content to the public space. If used then need to activate, since these are special content. They can be activated in the section Settings->General.

Newsletter

Owner is able to fill the form and select type of users he wants. Clicking the send button will trigger the email sending to all users with that type. Description field of the newsletter should allow Html as well as BBcode text formatting.

Translations

In this screen user is able to change between languages of the system. By selecting which one is active. When language is activated it country locale, number and money formatting in the settings is changed as well.

User is able to enter in edit mode and change source and translated language string.

Themes

This screen holds a list of all themes owned by Owner. As well as, all themes currently on the market for “Kuul-dev Bookings” software.

By activating one of the themes owned. Public or User space will change the overall look and feel. If no theme is purchase, site will use a default theme.

Theme Options

Every theme even default, have certain options that could modify the theme. This option vary from theme to theme. By changing this options, User space will be changed.

Menu

Here is a form that can rearrange top menu of User space. But creating new menu links, the default one will be substituted for new list of links.

Form inputs are:

- Title – new link title name
- URL – full url
- Target – html link target option e.g. _blank.
- Icon – Bootstrap 3 framework class for enabling icons (optional)

Social Auth

This filed contains configuration input fields, for each social network authentication API “Kuul-dev Bookings” is supporting. If correct information is provided, button appears in Login and Register screen. With corresponding name and logo of the social network.

Settings: General, Payment, Email, Product, Affiliates

This screens should contain input fields for each configuration data. It is divided into corresponding categories. This configurations directly alter certain functionalities of the system. Enabling and disabling, changing default values etc.

Users

This screen is a CRUD action. Owner can create remove and delete Users. As well as change their status. There is few different statuses

- Created
- Paid
- Refused
- Fraud
- Refund

The index screen should contain a list off all bookings with, for each, some basic information:

- Inactive
- Active
- Spam – tagged as spammer

Roles

Here we can add additional permissions and create new roles of the system.

This is CRUD action.

Updates

This screen prints all relevant data of current system version. The Owner is able to check for new updates and apply them by one press on the button. Also Owner is able to see details of all previous versions and where to download them etc.

Sitemap

By clicking on the “Generate” button, Owner can force regeneration of the Sitemap zip file.

Optimize

This screen prints some relevant data regarding database. Following list describes them:

- Rows
- Size KB
- Save Size KB

User is able to optimize database by clicking on the Optimize button, script will automatically update and optimize the system database.

Logs

In this screen should be printed information's on bugs recorded by the system. Owner is able to filter the printed records using date. To see record for specific date in past.

PHP info

Print out of the server or local host version of php and all information related.

Edit Profile

Every user can change basic information about their account.

Following list describes information available for modification:

- Name
- Password
- Newsletter acceptance
- Email
- Avatar image

2.8 TECHNOLOGIES

In this chapter, the discussion will point to the environment used to implement application. There is sufficient research on current demands in the field of design for web-based applications.

It proceeds with describing back-end framework, general architecture and high-level architectural patterns. As well as other relevant issues regarding this topic.

2.8.1 Localhost Server

For the development of this project simple laptop, or desktop PC, of any hardware configuration is sufficient enough.

Typical environment configuration requires Apache server, MySQL, and PHP to be installed on the development machine. This is very common combination for web development, nowadays. In fact, so usual, many tools are available that can install and configure all at once.

Depending on the operating system, we have the choice in Lamp, Wamp, Xampp and Mamp. Lamp is for Linux OS, Wamp is for windows OS, and Mamp is for Mac OS X. Additionally, Xampp have integrated additional features including support for Perl, FileZilla, mercury mail etc. And is available for all major operating systems.

For this project, decision has been made to use Xampp. It has been set up to be incredibly easy to install and to use. It stands for Cross-Platform (X), Apache (A), MySQL (M), PHP (P) and Perl (P). People know from their own experience that it's not easy to install an Apache web server and it gets harder if we want to add MySQL, PHP and Perl. The goal of Xampp is to build an easy to install a distribution for developers to get into the world of Apache. To make it convenient for developers, Xampp is configured with all features turned on.

- Xampp Installation
 - Download installer for specific OS
 - Run the installation, and accept the default settings

Table 2 - Typical settings of XAMPP configuration

Components	Description
Apache	Default web server application
MySQL	Database management system
PHP	Server-side, general purpose programming language
phpMyAdmin	Admin tool for working with MySQL
OpenSSL	An open-source implementation of two popular security protocols – SSL and TLS
XAMPP Control Panel	A simple control panel for working with different XAMPP components
Webalizer	An analytics tool that generates user logs and usage metrics.
Mercury Mail Transport System	A simple, open-source mail server
FileZilla FTP Server	A FTP (File Transfer Protocol) server to make file transfers smoother
Tomcat	A freeware Java servlet for serving Java applications
Strawberry Perl	A popular distribution of Perl for Windows

- Start Xampp control panel, and activate Apache server and MySQL

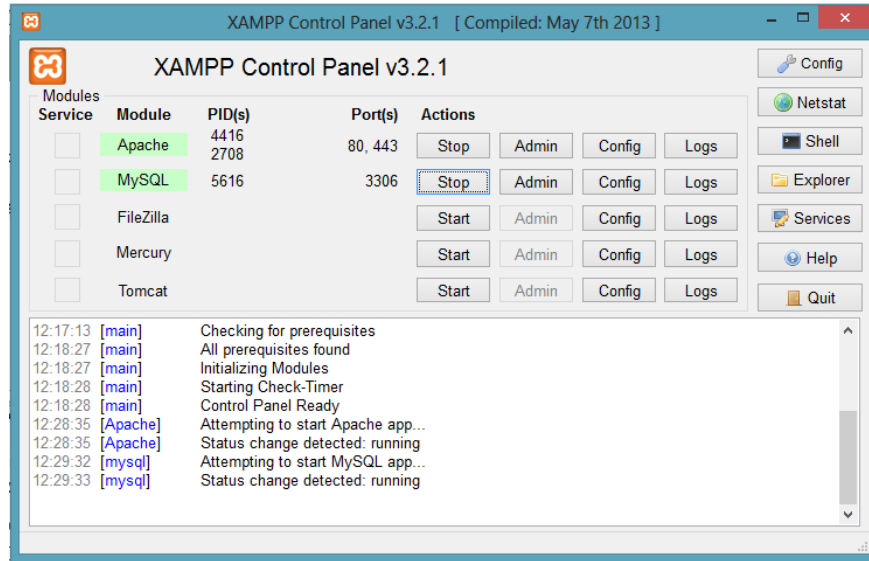


Figure 6 - Xampp Control panel

- Check if everything is running correctly
 - Apache -> type <http://localhost>



Figure 7 - LocalHost Success window

- Mysql -> type <http://localhost/phpmyadmin>

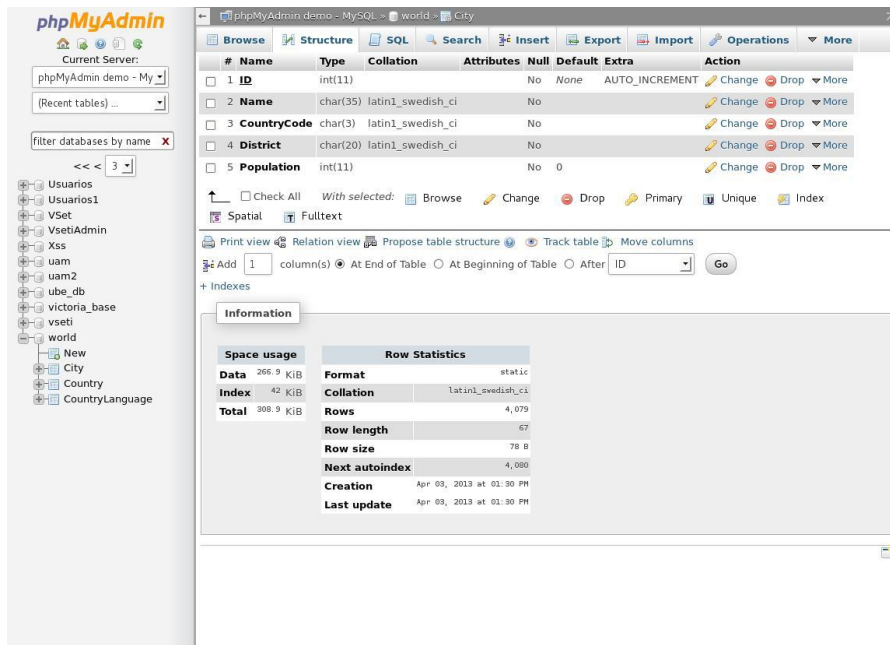


Figure 8 - PhpMyAdmin Control Panel

2.8.2 Backend - Server Side

Typically, a server is a computer program, such as a web server, that runs on a remote server, reachable from a user's local computer or workstation. Operations may be performed server-side because they require access to information or functionality that is not available on the client, or require typical behavior that is unreliable when it is done client-side. Server-side operations also include processing and storage of data from a client to a server, which can be viewed by a group of clients.

2.8.3 Php 5.4

PHP is a popular general-purpose scripting language that is especially suited to web development. Fast, flexible and pragmatic. In an HTML document, PHP script (similar syntax to that of Perl or C) is enclosed within special PHP tags. Because PHP is embedded within tags, the author can jump between HTML and PHP (similar to ASP and Cold Fusion) instead of having to rely on heavy amounts of code to output HTML. And, because PHP is executed on the server, the client cannot view the PHP code. PHP can perform any task that any CGI program can do, but its strength lies in its compatibility with many types of databases. Also, PHP can talk across networks using IMAP, SNMP, NNTP, POP3, or HTTP.



Figure 9 - Php logo

- Advantages

Easy to learn: PHP has a short learning curve and programmers can quickly become productive. PHP was designed to appeal to Web designers and HTML coders, and they appreciate the ability to freely mix HTML and PHP. PHP allows them to easily and gradually add dynamic page generation features to their websites.

Open Source: PHP is distributed under an Apache-style license that allows for both commercial and non-commercial use and development. This means that we can use it freely, without paying any licenses fees for machine, CPU, and so on. Also, there is a worldwide network of talented developers continuously improving and enhancing PHP. We can fix bugs or customize the software to our own specific needs (or pay someone to do so) because the source code is available. This is not possible with commercial, off-the-shelf products.

Community: PHP has a large base of users and developers. It is easy to find programmers fluent in the language. Many online resources are dedicated to PHP (websites, mailing lists, and so on) that provide valuable information and support.

Database support: PHP provides extensive database support. It supports ODBC, open source databases such as MySQL and PostgreSQL, as well as commercial ones such as Microsoft SQL Server, Oracle, and Sybase.

Multiplatform support: PHP runs on a variety of platforms and Web servers. PHP runs in most flavors of UNIX and Windows as well as other OS such as Mac OS, OS X, or OS/2. PHP supports a wide variety of Web servers, ranging from the popular Apache, Microsoft IIS, and Netscape servers to less-known ones such as thttpd or AOLserver. This allows us to standardize on a common development language across a heterogeneous environment of systems and servers. We can build a solution with PHP on a specific platform/server/database combination and then migrate to a different combination gradually, replacing one component at a time. We can develop our code on a Windows workstation running IIS and deploy it on a UNIX server running Apache with little or no changes.

Session support: Most Web applications require us to keep and manage state between requests. PHP offers native session management and an extension API so users can provide their own backend storage mechanisms.

Rapid development: PHP gets compiled to a special byte code format before getting executed. That step is completely transparent to programmers and users. Developers can make changes to a PHP page and see the results immediately in their browsers. By comparison, Java servlet development requires compile cycles and careful configuration of things such as class loaders, and so on.

- Disadvantages

Security: Since it is open sourced, so all people can see the source code, if there are bugs in the source code, it can be used by people to explore the weakness of PHP

Not suitable for large applications: Hard to maintain since it is not very modular.

2.8.4 MySQL



Figure 10 - MySQL logo

MySQL is the world's most popular open source database software, with over 100 million copies of its software downloaded or distributed throughout its history. With its superior speed, reliability, and ease of use, MySQL has become the preferred choice for Web, Web 2.0, SaaS, ISV, Telecom companies and forward-thinking corporate IT Managers because it eliminates the major problems associated with downtime, maintenance and administration for modern, online applications. Many of the world's largest and fastest-growing organizations use MySQL to save time and money powering their high-volume websites, critical business systems, and packaged software — including industry leaders such as Yahoo, Alcatel-Lucent, Google, Nokia, YouTube, Wikipedia, and Booking.com. The flagship MySQL offering is MySQL Enterprise, a comprehensive set of production-tested software, proactive monitoring tools, and premium support services available in an affordable annual subscription. MySQL is a key part of LAMP (Linux, Apache, MySQL, PHP / Perl / Python), the fast-growing open source enterprise software stack. More and more companies are using LAMP as an alternative to expensive proprietary software stacks because of its lower cost and freedom from platform lock-in.

Many of the world's largest and fastest-growing organizations including Facebook, Google, Adobe, Alcatel Lucent and Zappos rely on MySQL to save time and money powering their high-volume Websites, business-critical systems and packaged software.

Many web applications use MySQL as the database component of a LAMP software stack. Its popularity for use with web applications is closely tied to the popularity of PHP, which is often combined with MySQL. Several high-traffic websites (including Flickr, Facebook, Wikipedia, Google (though not for searches), Nokia and YouTube) use MySQL for data storage and logging of user data.

- Advantages

It's secure: MySQL includes solid data security layers that protect sensitive data from intruders. Rights can be set to allow some or all privileges to individuals. Passwords are encrypted.

It's fast: In the interest of speed, MySQL designers made the decision to offer fewer features than other major database competitors, such as Sybase* and Oracle*. However, despite having fewer features than the other commercial database products, MySQL still offers all of the features required by most database developers.

Cross Platform Operability One of the biggest factors which makes MySQL the most opted form of database is Cross Platform Operability. It has proved itself in getting installed in all the major platforms such as Linux, Windows, Solaris and so on and at the same time performance has not been affected. Apart from that, the presence of APIs makes its integration with C, C++, Perl, Java and Python etc pretty easy.

It manages memory very well: MySQL server has been thoroughly tested to prevent memory leaks.

2.8.5 Front end – Client side

Software presentation, it's just as important as a functional part of the back office.

In the past, software design focused on specific platforms. The design elements were static on the screen, since internet was accessible only from personal computers and terminal. In other words, big and wide screens.

Today, this is not the case! Portable devices are present and the future of user interphase (UI) design. Thus, the constant evolution of the portable devices, web development has become more complex.

Various problems

When designing an application, we are forced to think beyond just the typical personal computer (PC). Not only, do we have multiple OS's to deal with, but also various browsers with different versions and array of portable devices.

The graph in Figure 11 shows usage of screen resolutions recorded 2011.

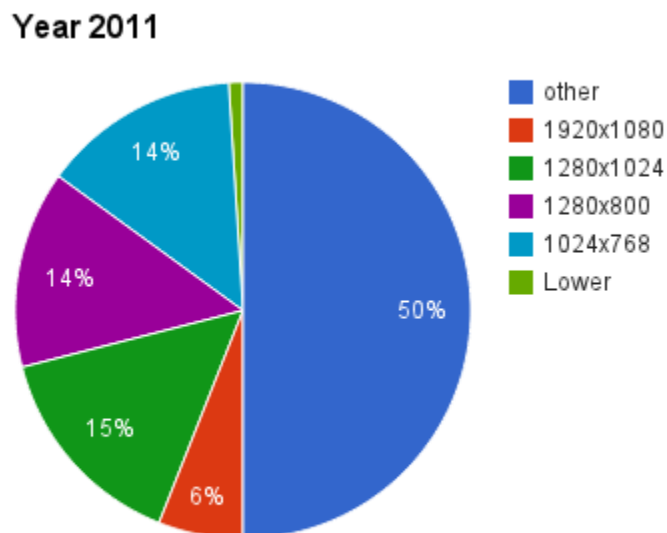


Figure 11 - 2011 statistics of screen resolutions.

The most significant reason why screen resolution is important and not screen size. Is because we can have a small screen but big resolution. Which is very common this days.

For example, “Samsung Galaxy 4”, have 5 inch Display Size, and 1.920 x 1.080 resolution. Or new iPad with Retina display.

In addition, some laptops have much bigger screen size but lower resolution. Take for instance, my laptop:

“ASUS X501A”, have 15.6 inch, and 1.366 x 768 resolution.

Moreover, today on our disposal are more than 5 different web browsers. And even more versions of each one. Take the following list for instance, and note the corresponding versions marked in [].

- Chrome – [Latest version 35]
- IE - Internet Explorer [7, 8, 9, 10, 11]
- Mozilla Firefox – [Latest 30]
- Safari – [Latest 7]
- Opera – [Latest 22]

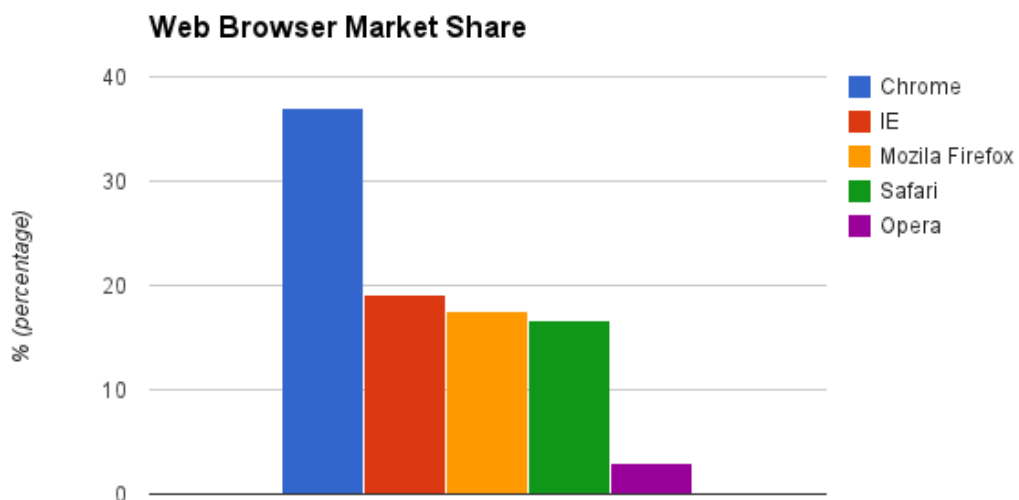


Figure 12 - Latest market share of most popular web browsers

Different browsers interpret HTML, CSS and JavaScript with some differences. Sometimes sufficient enough to brake overall cohesiveness of the application design. Since it is almost impossible to now visitor's personal choice of the browse. It becomes the most vital requirement for a web designer, is to make application that are fully compatible across different browsers.

- Where we headed in terms of innovations?

Google Glass, for instance, could drastically change our interaction with the application. As a conclusion, we can say that it's very important to approach this subject carefully. Our intention is to design our application as an easy to use, functional and beautiful.

- Design bullet points

Frameworks – Developing an application from the scratch is already a pretty complex and comprehensive work. So in our best interest is to reduce this load. Picking the right front-end framework could come handy and help us bootstrap our application.

Responsive design - This notation and idea was introduced with the need to adjust working websites and web apps for the speedy growing market of portable devices. Above all, we need that our app can be reached easily from any device type.

Thinking In Multiple Views – This is where we build our application based on template themes. This can help maximize personal user experience, but as well allow for expansion into different layouts for future.



Figure 13 - Responsive design illustration

2.8.6 HTML5

1. HTML stands for Hyper Text Mark-up Language
2. HTML is a mark-up language
3. A mark-up language is a set of mark-up tags
4. The tags describe document content
5. HTML documents contain HTML tags and plain text
6. HTML documents are also called web pages



Figure 14 - HTML 5 logo

HTML5 is a core technology mark-up language of the Internet used for structuring and presenting content for the World Wide Web.

It is the latest version of Hypertext Mark-up Language, the code that describes web pages. It's actually three kinds of code: HTML, which provides the structure; Cascading Style Sheets (CSS), which take care of presentation; and JavaScript, which makes things happen. It has been designed to deliver almost everything we need, to do online without requiring additional software such as browser plugins. It does everything from animation to apps, music to movies, and can also be used to build incredibly complicated applications that run in our browser. There's more. HTML5 isn't proprietary, so we do not need to pay royalties to use it. It's also cross-platform, which means it doesn't care whether we are using a tablet or a smartphone, a netbook, notebook or Ultrabook or a Smart TV: if our browser supports HTML5, it should work flawlessly. Inevitably, it's a bit more complicated than that. More about that in a moment.

We've come a long way since HTML could barely handle a simple page layout. HTML5 can be used to write web applications that still work when we are not connected to the net, to tell websites where we are physically located, to handle high definition video, and to deliver extraordinary graphics.

- HTML Structure

`<tagname>content</tagname>`

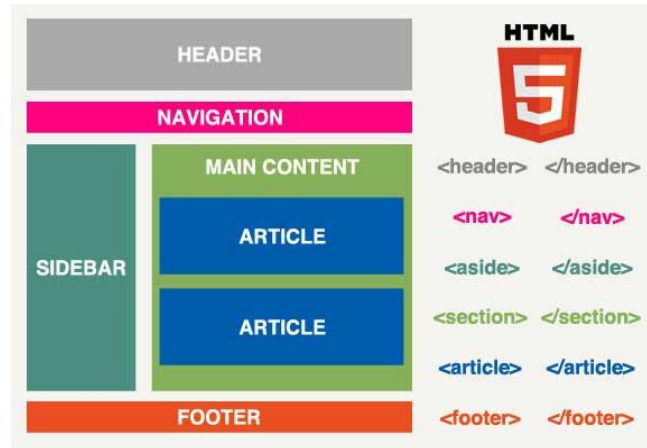


Figure 15 - HTML 5 Structure

1. HTML tags are keywords (tag names) surrounded by angle brackets like `<html>`
2. HTML tags normally come in pairs like `<p>` and `</p>`
3. The first tag in a pair is the start tag, the second tag is the end tag
4. The end tag is written like the start tag, with a slash before the tag name
5. Start and end tags are also called opening tags and closing tags

- Example

```

<!DOCTYPE html>
<html>
  <body>
    <h1>My First Heading</h1>
    <p>My first paragraph.</p>
  </body>
</html>

```

- Output

My First Heading

My first paragraph.

2.8.7 Javascript / JQuery

- Change HTML elements
- Delete HTML elements
- Create new HTML elements
- Copy and clone HTML elements
- And much more ...



Figure 16 - jQuery logo

JavaScript is an object-oriented computer programming language commonly used to create interactive effects within web browsers.



Figure 17 - JavaScript logo

It is the most popular programming language in the world. It is the language for HTML, for the Web, for computers, servers, laptops, tablets, smart phones, and more.

With just few simple instructions JavaScript can do great things.

In the browser, JavaScript adds interactivity and behaviour to HTML content. Without JavaScript, web pages would be static and boring. JavaScript helps bring a web page to life.

JavaScript was originally designed to add interactivity to web pages, not to be a general programming language, which makes it a scripting language. Scripting languages are regarded to be more productive than general languages because they are optimized for their specific domain (in this case, the web browser). However, recent advancements have brought JavaScript to the server-side (Node.js) so it can now be used in place of languages like PHP, Ruby, or ASP.

2.8.8 CSS

1. CSS stands for Cascading Style Sheets
2. Styles define how to display HTML elements
3. Styles were added to HTML 4.0 to solve a problem
4. External Style Sheets can save a lot of work
5. External Style Sheets are stored in CSS files



Figure 18 - CSS logo

Cascading Style Sheets (CSS) is a style sheet language used for describing the look and formatting of a document written in a mark-up language. CSS is designed primarily to enable the separation of document content from document presentation, including elements such as the layout, colours, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple pages to share formatting, and reduce complexity and repetition in the structural content (such as by allowing for table less web design). It obviates those portions of mark-up that would specify presentation by instead providing that information in a separate file. For each relevant HTML element (identified by tags), it provides a list of formatting instructions.

For example, it might say (in CSS syntax), "All heading 1 elements should be bold." Therefore, no formatting mark-up such as bold tags (``) is needed within the content; what is needed is simply semantic mark-up saying, "this text is a level 1 heading." CSS can also allow the same mark-up page to be presented in different styles for different rendering methods, such as on-screen, in print, by voice (when read out by a speech-based browser or screen reader) and on Braille-based, tactile devices. It can also be used to allow the web page to display differently depending on the screen size or device on which it is being viewed. While the author of a document typically links that document to a CSS file, readers can use a different style sheet, perhaps one on their own computer, to override the one the author has specified. However if the

author or the reader did not link the document to a specific style sheet the default style of the browser will be applied.

- CSS Syntax

A CSS rule set consists of a selector and a declaration block:

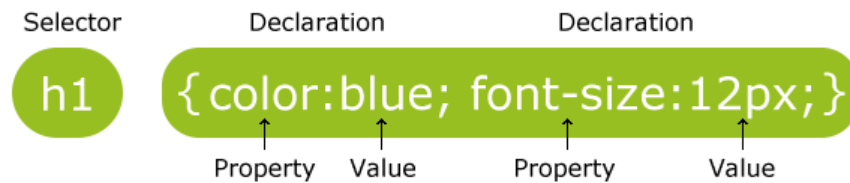


Figure 19 - CSS rule set constants

The selector points to the HTML element we want to style. The declaration block contains one or more declarations separated by semicolons. Each declaration includes a property name and a value, separated by a colon.

- Example

```
<!DOCTYPE html>
<html>
<head>
  <style>
    p {
      color: red;
      text-align: center;
    }
  </style>
</head>
<body>
  <p>Hello World!</p>
  <p>This paragraph is styled with CSS.</p>
</body>
</html>
```

Result:

Hello World!

This paragraph is styled with CSS.

2.9 FRAMEWORK – KOHANA

Essentially, framework is a layered structure intended to serve or guide the process of development and implementation. Allows us to connect too many different API's as well as determine the structure of our own application. For this project, decision has been made to use Kohana Framework. It's not the easiest to learn, but certainly has everything we need for a great application.

Modern PHP frameworks use Model-View-Controller architectural pattern. Other frameworks include CodeIgniter and Symfony. Some like CodeIgniter for its smaller footprint. It's all a matter of preference. Whichever framework picked it's important to use the documentation on the sites as it is essential to understanding how best to use it. Also, very helpful is to peak at the code every once in a while just to get a better understanding of how things work.

Figure 20 shows general MVC architecture of Kohana framework.

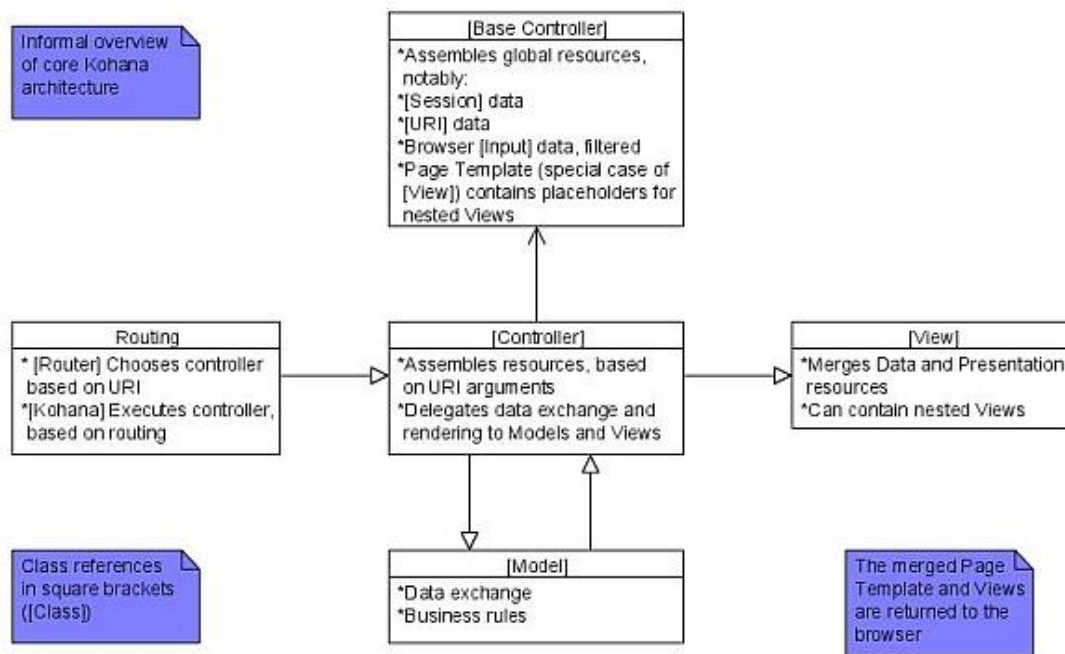


Figure 20 - Kohana Architecture

2.9.1 Core Services

In addition to the core components in the diagram above, the following list describes standardized services.

- Configuration
- Benchmarking
- Profiler
- Events
- utf-8
- Internationalization, including locale and time zone
- Error/Exception Handling
- Logging
- Input filtering

Some clarifications:

- Input: filters global data -> XSS, gpc_magic_quotes, etc
- Router: chooses controller/method based on URI
- Kohana: executes controller, based on routing
- Controller: dynamically collects data for presentation, based on URI arguments

2.9.2 About Kohana

Kohana is an open source, object-oriented MVC web framework built using PHP5 programming language. Developed by team of volunteers that aims to be swift, secure, and small. It is licensed under a BSD license, so we can legally use it for any kind of open source, commercial, or personal project.

Anything can be extended using the unique file system design, little or no configuration is necessary, error handling helps locate the source of errors quickly, and debugging and profiling provide insight into the application.

To help secure applications, tools for XSS removal, input validation, signed cookies, form and HTML generators are all included. The database layer provides protection against SQL injection. Of course, all official code is carefully written and reviewed for security.

Kohana implements set of functional “Helper” classes that could be used through the application. Moreover, Kohana uses very nice **Cascading File-system**, which makes organization of the files simpler and more native.

It separates the executing files in 3 sections:

- **Application Path**
Defined as APPPATH in index.php. The default value is application.
- **Module Paths**
This is set as an associative array using Kohana::modules in APPPATH/bootstrap.php. Each of the values of the array will be searched in the order that the modules are added.
- **System Path** Defined as SYSPATH in index.php. The default value is system. All of the main or "core" files and classes are defined here.

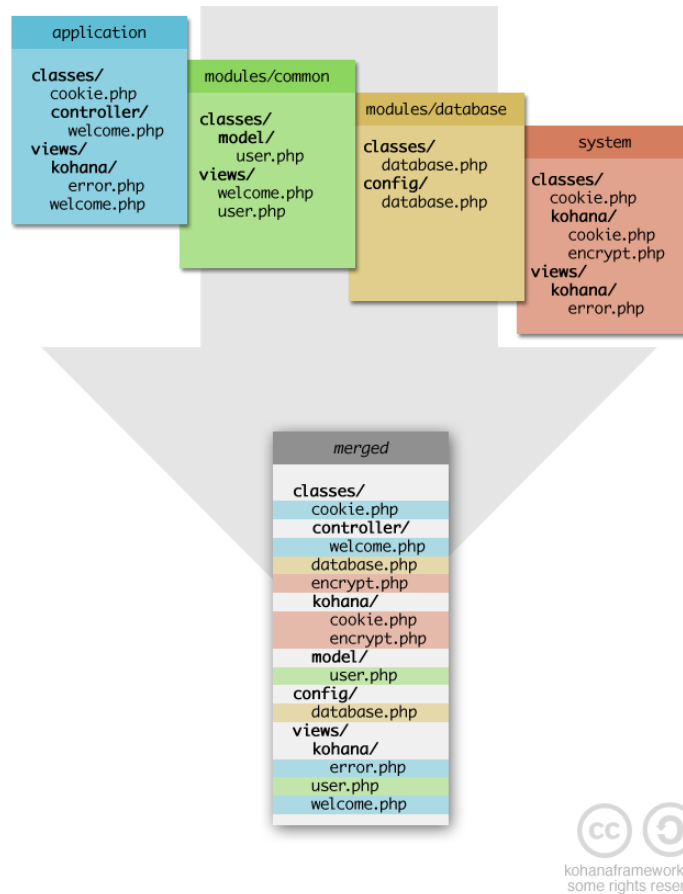


Figure 21 - Kohana Cascading File-system

One more thing worth of mentioning are Modules that are simply the addition to Cascading File-system. They can implement any kind of file (controllers, config, HTML view etc.). It can separate the system functionality and make it more transportable and sharable among the different applications.

2.10 ORM

Object-relational mapping in computer science is a programming technique for converting data between incompatible type systems in object-oriented programming languages.

ORM is a method of abstracting database access to standard PHP calls. All table rows are represented as model objects, with object properties representing row data. ORM in Kohana generally follows the Active Record pattern.

Kohana 3.X includes a powerful Object Relational Mapping (ORM) module that uses the active record pattern and database introspection to determine a model's column information. ORM is integrated tightly with the Validate library. The ORM allows for manipulation and control of data within a database as though it was a PHP object.

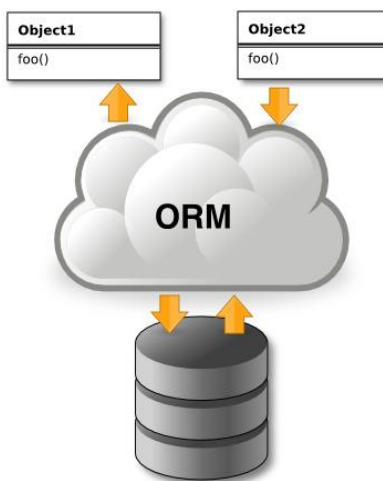


Figure 22 - ORM Flow

Once defined the relationships ORM allows us to pull data from our database, manipulate the data in any way we like and then save the result back to the database without the use of SQL. By creating relationships between models that follow convention over configuration, much of the repetition of writing queries to create, read, update and delete information from the database can be reduced or entirely removed. All of the relationships can be handled automatically by the ORM library and we can access related data as standard object properties.

ORM is included with the Kohana 3.X install but needs to be enabled before we can use it.

2.10.1 Basic Usage

- To create new Model instance:

```
$user = ORM::factory('user');  
// Or  
$user = new Model_User();  
  
Basic Crud  
$user = ORM::factory('user');  
  
$user->first_name = "Slobodan";  
$user->last_name = "Josifovic";  
$user->city = "Barcelona";  
  
$user->save();  
  
$user = ORM::factory('user', 20);  
$user->delete();
```

2.10.2 Active Record pattern

The active record pattern is a design pattern frequently found in software that stores its data in relational databases. Active record is an approach to accessing data in a database.

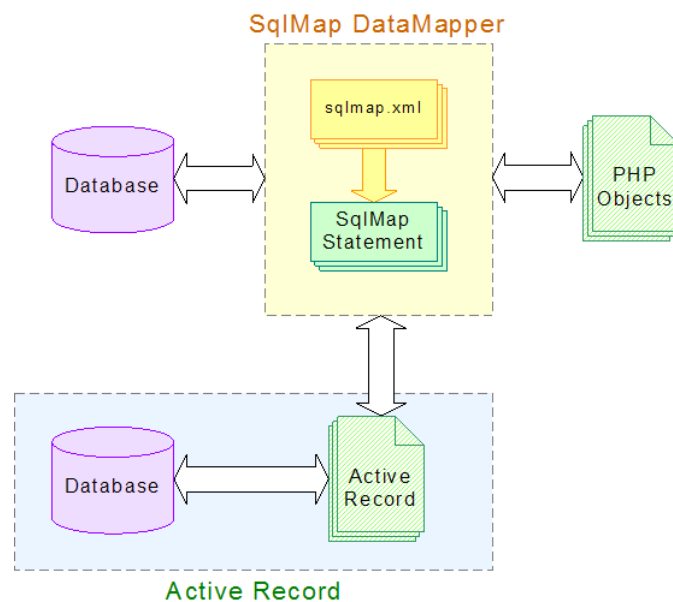


Figure 23 - Active Record Pattern

2.11 MVC

The Model-View-Controller pattern separates the modeling of the domain, the presentation, and the actions based on user input into three separate classes.

- MODEL

This is the domain that the software is built around. It is very similar to Java class, since it usually represents a real-world item. Such as Person, Employee, Account etc. It holds structure and data that defines that item. And it's usually a bond between the database table and the domain of the system. In the model, we define relationships between other models and configuring together relevant information's.

Nowadays, almost every PHP framework integrates MVC pattern. Since it's the most popular web-based architectural pattern.

Following examples is code representation of the Model:

```
class Model_User extends ORM {} // Declares the class, extends ORM that we
discussed above
protected $_table_name = 'users'; // Makes a connection with corresponding DB
table
protected $_primary_key = 'id_user'; // Defines primary key
// Relations, with other Models (Role, Location)
protected $_belongs_to = array(
    'role' => array(
        'model' => 'role',
        'foreign_key' => 'id_role',
    ),
    'location' => array(
        'model' => 'location',
        'foreign_key' => 'id_location',
    ),
);
```

This is the way to define Model class. Also, each framework implements its own system functionalities. Such as Form Validation, Error Handling, Routing, Response Request Flow etc.

Validation nowadays is present in almost every application, from Authentication of the user to Creating new users and in our case, for example, booking of the product.

- Front-end validation

In fact, frameworks do not support this feature. Since, there are many JavaScript plugins and Front-End frameworks that we are going to use in this project.

- Back-end validation

In the case of back-end validation, Kohana implements an easy way to deal with this problem.

First we need to define rules in the model. These rules are bound to table names, and magically do checking of type and other restrictions applied.

```
public function rules()
{
return array(
'id_user'=> array(array('numeric')),
'name'=> array(array('max_length', array(':value', 145))),
'email' => array(array('not_empty'), array('max_length', array(':value',
145)), ),
'password'=> array(array('not_empty'), array('max_length',
array(':value', 64)), ),
'status'=> array(array('numeric')),
'id_role'=> array(array('numeric')),
);
}
```

Next we need to process upcoming \$_POST information and we do this in Controller by calling Validation class function check().

- VIEW

This is a visual representation of data coming from the model. It's usually mock-up for of the page we want to send as in HTTP response. View interacts directly with Controller and Browser, and it's usually a bad practice to call Model directly from this document.

It consists of HTML code as Mock-up, CSS as style cosmetics, and JavaScript or similar library or framework that adds motion to view.

- CONTROLLER

Controller is considered to play the role between Model and View! Controller is responsible for processing input, acting upon the model, and deciding on what action should be performed, such as rendering a view or redirecting to another page.

Controllers should be kept very light and dynamic. There are no harm in overscheduling controllers, but with the idea of keeping the project scalable and modular. The best practice is to keep it simple and light.

Kohana controllers are called by `Request::execute()` function based on the route URL.

Creating controllers must follow a certain rule set, described in the following list:

- Reside in `classes/controller` (or a sub-directory)
- Filename must be lowercase, e.g. `articles.php`
- The class name must map to the filename (with `/` replaced with) and each word is capitalized
- Must have the `Controller` class as a (grand)parent

Following example is the code representation of Controller class:

```
class Controller_Maintenance extends Kohana_Controller {

public function action_index() {
    //in case there's no maintenance go back to the home.
    if (core::config('general.maintenance') !=1)
        Request::current()->redirect(Route::url('default'));

    $this->response->status(503);
    $this->response->body(View::factory('pages/error/503')->render());
}
}
```

This is a food example of clean and light controller class. First, we are declaring that controller Maintenance extends Kohana_Controller and its constructor functions. Second, we are naming the action index, with “action_name” prefix, to inform Kohana that this is not a standard function. And finally we write our logic, which in this case checks config table and redirects us to the appropriate view. If maintenance mode is ON (true) we block users from accessing site, and if OFF (false) we redirect them to Home (default) page.

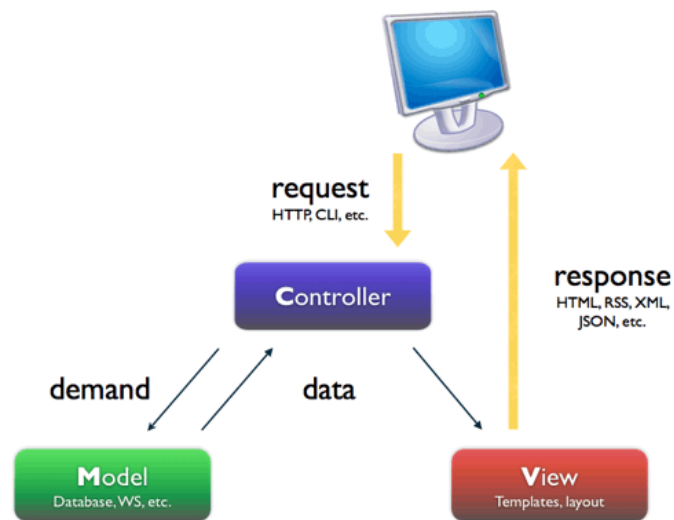


Figure 24 - MVC pattern class structure and cycle

2.12 RESTFUL

Representational state transfer (REST) is an abstraction of the architecture of the World Wide Web. More precisely, REST is an architectural style consisting of a coordinated set of architectural constraints applied to components, connectors, and data elements, within a distributed hypermedia system. REST ignores the details of component implementation and protocol syntax in order to focus on the roles of components, the constraints upon their interaction with other components, and their interpretation of significant data elements.

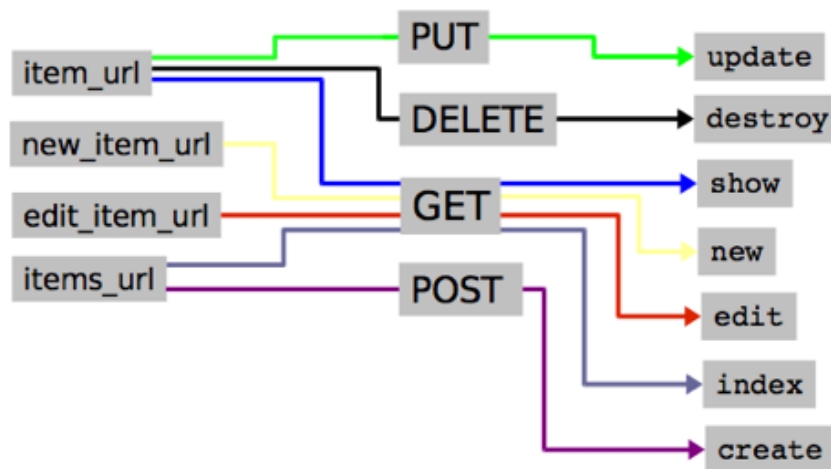


Figure 25 - RESTful routing chart

Kohana implements Abstract Controller class for RESTful controller mapping.

Default uses of HTTP methods will be mapped to these actions.

- GET
Mapped to the "index" action, lists all objects
- POST
Mapped to the "create" action, creates a new object
- PUT
Mapped to the "update" action, update an existing object
- DELETE
Mapped to the "delete" action, delete an existing object

3 DESIGN AND IMPLEMENTATION

In this chapter will be described design and implementation choices made. We are going to use our knowledge from a previous chapter, to discuss general architecture domain model implementation, database schema. As well as, some additional functionalities such as Installation and Update system, automatic emailing, translation solution, widgets etc.

3.1 GENERAL ARCHITECTURE

“Kuul-dev Bookings” was developed using Business and Application architecture logic with Kohana Framework, and for Presentation layer was used Bootstrap 3 framework. In the Figure 26 we can see the architecture described more descriptive.

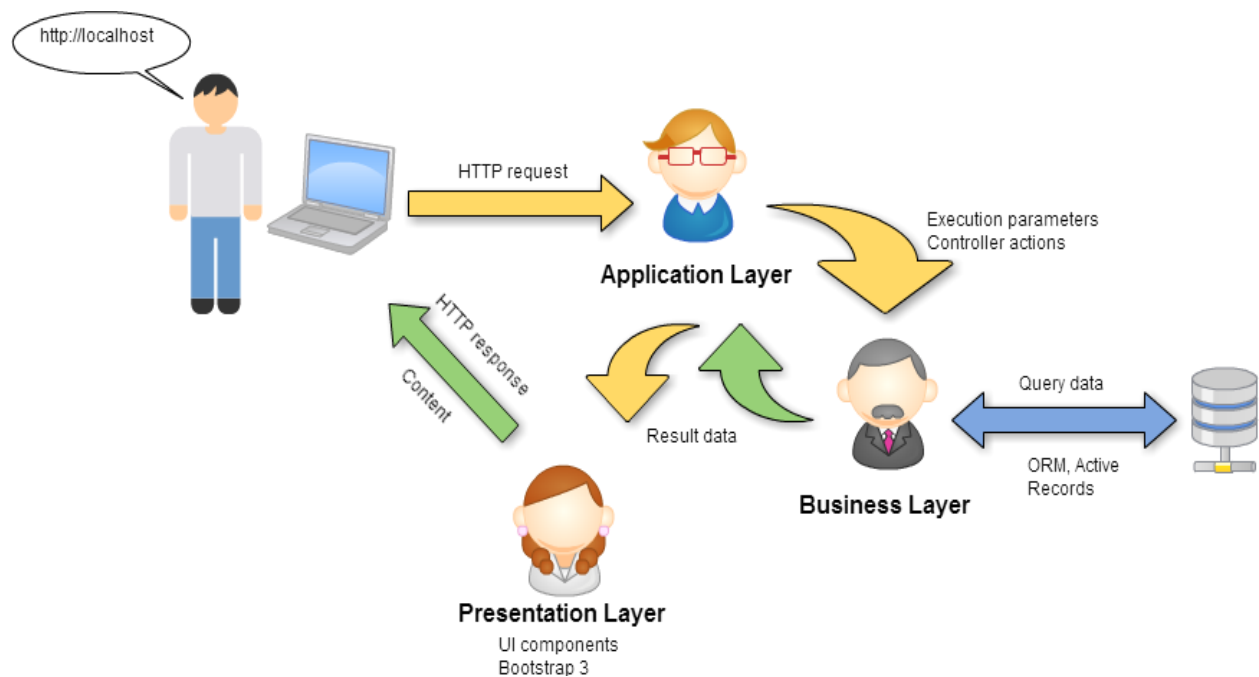


Figure 26 - General system architecture

3.2 APPLICATION LAYER, REST API

Kohana provides a very powerful routing system. In essence, routes provide an interface between the URLs and our controllers and actions. With the correct routes, we could make almost any URL scheme correspond to almost any arrangement of controllers, and we could change one without impacting the other.

It is important to understand that routes are matched in the order they are added, and as soon as a URL matches a route, routing is essentially "stopped" and the remaining routes are never tried. Because the default route matches almost anything, including an empty URL, new routes must be placed before it.

3.2.1 System routes

Each route must have a minimum of a name, a URI and a set of defaults for the URI.

Let's take a closer look at each of the parameters of `Route::set`, which are `name`, `uri`, and an optional array `regex`.

- Name

The name of the route must be a unique string. If it is not it will overwrite the older route with the same name. The name is used for creating URLs by reverse routing, or checking which route was matched.

- URI

The `uri` is a string that represents the format of URLs that should be matched. The tokens surrounded with `<>` are keys and anything surrounded with `()` are optional parts of the `uri`. In Kohana routes, any character is allowed and treated literally aside from `()<>`. The `/` has no meaning besides being a character that must match in the `uri`. Usually the `/` is used as a static separator but as long as the `regex` makes sense, there are no restrictions to how we can format our routes.

Reserved pages for "Kuul-dev Bookings" usage. They use the `i18n` translation. We will use them with extension `.html` to avoid repetitions with others.

- Regex

The Kohana route system uses perl compatible regular expressions in its matching process. By default each key (surrounded by <>) will match `[^/.,;?\n]+` (or in English: anything that is not a slash, period, comma, semicolon, question mark, or newline). We can define our own patterns for each key by passing an associative array of keys and patterns as an additional third argument to `Route::set`.

- Default values

If a key in a route is optional (or not present in the route), we can provide a default value for that key by passing an associated array of keys and default values to `Route::defaults`, chained after `Route::set`. This can be useful to provide a default controller or action for API, among other things.

3.2.2 REST api

Table 3 – Api routes

Name	URI	Controller	Action
search	/search.html	product	search
contact	/contact.html	contact	index
maintenance	/maintenance.html	maintenance	index
page	/ <code><seotitle></code> .html	page	view
rss-blog	/rss/blog.xml	feed	blog
rss	/rss/ <code><category></code> / <code><location></code>).xml	feed	index
sitejson	/info.json	feed	info
blog	/blog/ <code><seotitle></code> .html)	blog	index
forum-new	/forum/new_topic.html	forum	new
forum-topic	/forum/ <code><forum></code> / <code><seotitle></code> .html	forum	topic
faq	/faq/ <code><seotitle></code> .html	faq	index
oc-panel	/oc-panel(<code><controller></code> / <code><action></code> / <code><id></code> / <code><current_url></code>))	home	index

forum-list	/forum/<forum>	forum	list
forum-home	/forum	forum	index
product-minimal	/<<category>/embed/<seotitle>.html	product	view
product-goal	/<<category>/thanks/<order>/<seotitle>.html	product	goal
product-demo	/<<category>/demo/<seotitle>.html	product	demo
product-paypal	/<<category>/paypal/<seotitle>.html	paypal	pay
product-review	/<<category>/reviews/<seotitle>.html	product	reviews
product	/<<category>/<seotitle>.html	product	view
list	/<<category>	product	listing
error	/oc-error/<action>/<origuri>/<message>	error	index
default	/(<controller>/<action>/<id>))	home	index

3.3 SQL CODING STANDARD

3.3.1 Common

- No usage of camel case, *everything lower case*.
- To separate words usage of underscore symbol *_*.
- Default charset and collation 'utf8'.

3.3.2 Table names

- Always plural (when need)
- Prefix {{{prefix_table_name}}}

Examples:

- Simple table: ob_users
- 1-N relation table: ob_user_categories
- N-M relation table: ob_users_locations

3.3.3 Fields

- Singular (except when they're plural... i.e. glasses)
- Descriptive to the content
- Default NULL, if is numeric 0
- If type DATE default CURRENT_TIMESTAMP
- Varchar for any variable data < 255
- Text for huge amount (remember can't create index at InnoDB, try to avoid text type)
- If needed, comment (about structure)

3.3.4 Primary key "PK"

- Singular
- Autoincrement
- id_[table_name_singular]
- SELECT id_user FROM users

Example: id_user → id_user_payment

3.3.5 Indexes "IK"

- [table_name]_IK_[field_name] [more than one AND]

Example: ob_posts_IK_id_user_AND_id_post

3.3.6 FK index name

- [current_table_name]_FK_[foreign_primary_key_field]_AT_[foreign_table_name]

Example: users_FK_id_site_AT_sites

3.3.7 Unique index name

- [table_name]_UK_[field_name] [more than one AND]

Example: ob_posts_UK_id_user_AND_id_post

3.4 DATABASE SCHEMA

For convenience and size of the project, only the most important tables are going to be discussed. Following Figure 27, is the complete overview of the designed database schema.

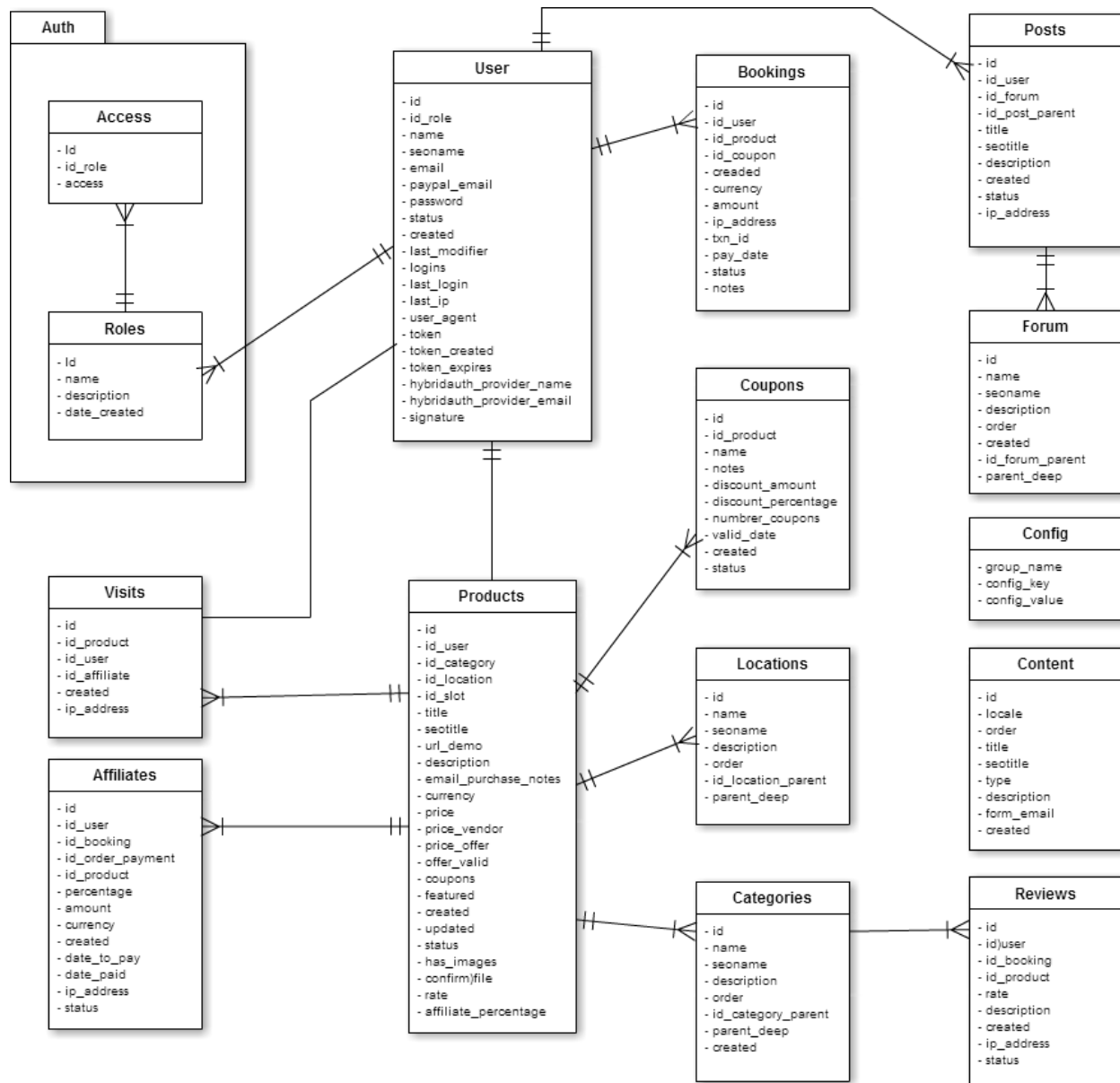


Figure 27 - Complete Database Schema

3.4.1 Table “ob_locations”

In this project location is essentially the list of geographical locations used to classify products. This list is manually introduced to the system, by the owner or moderator role.

Note: Locations and Categories are essentially equivalent. Because of small difference between them, decision has been made to present only one - “Locations”.

- Fields definition

id_location: Unique field used as identifier.

name: Location name.

order: Assigned display order.

created: Date and time, set when created with current time() function.

id_location_parent: If location is a child, record Id of the parent location, 0 if otherwise.

parent_deep: Sublevel number of the location from location parent.

seoname: SEO friendly name of the location. Must be unique.

description: Short description.

- Usage

Locations are developed with the CRUD functionalities, they can be created, edited or deleted at `/oc-panel/locations`.

- Relations

Each product has a location assigned.

```
protected $_belongs_to = array(  
    'parent' => array('model' => 'Location',  
                     'foreign_key' => 'id_location_parent'),  
);
```

- Schema

Tables 4 and 5 show detailed “ob_location” table structure, its data type, restrictions and special options.

Table 4 - Locations table schema

Field	Type	Null	Key	Default	Extra
id_location	int(10) (unsigned)		PRI		auto_increment
name	varchar(145)				
order	int(2) (unsigned)			0	
created	timestamp			current_timestamp	on update current_timestamp
id_location_parent	int(10) (unsigned)			0	
parent_deep	int(2) (unsigned)			0	
seoname	varchar(145)		UNI		
description	varchar(255)	YES		NULL	

- Indexes

Table 5 - Location table Indexes

Keyname	Type	Cardinality	Field
PRIMARY	PRIMARY	1	id_location
ob_locations_IK_seo_name	UNIQUE	1	seoname

3.4.2 Table "ob_users"

Table "ob_users" is directly responsible for storing all user related data.

- Fields

id_user: Unique field used as identifier.

name: Name of the user.

seoname: SEO friendly name of the user. Must be unique.

email: Email of the user. Must be unique.

paypal_email: PayPal email of the user for payments.

password: Hashed password of the user. Password is encrypted in the DB on save of the model.

status: Status of the user. 0 user inactive. 1 user active. 5 user tagged as spam.

id_role: The role ID assigned to the user. Roles are defined in ob_roles table.

created: Date and time of user creation.

last_modified: Date last time the user was modified.

logins: Number of times user has logged.

last_login: Date and time of the last time the user logged.

last_ip: IP address of the last time the user logged. Stored as Long.

user_agent: Sha1 hashed user agent of the user.

token: Unique token for user autologin.

token_created: Date and time of token creation.

token_expires: Date and time of token expiration.

hybridauth_provider_name: Name of the provider if user logged using "Social Login".

hybridauth_provider_uid: User provider identifier using "Social Login".

signature: Email signature of the user.

- Usage

Each time user is registered, this information is saved in table “ob_users”. Moreover, for this iteration, only registered users are able to interact with booking use case.

Users can be added, edited or deleted at /ob-panel/user.

- Relations

Each user has a role assigned.

```

/**
 * @var array ORM Dependency/hirerachy
 */
protected $_belongs_to = array(
    'role' => array(
        'model'      => 'role',
        'foreign_key' => 'id_role',
    ),
);

```

- Schema

Tables 6 and 7 show more detailed “ob_user” table structure, its data type, restrictions and special options.

Table 6 - Users table schema

Field	Type	Null	Key	Default	Extra
id_user	int(10) unsigned		PRI		auto_increment
name	varchar(145)	YES		NULL	
seoname	varchar(145)	YES	UNI	NULL	
email	varchar(145)		UNI		
paypal_email	varchar(145)	YES		NULL	
password	varchar(64)				
status	int(1)			0	
id_role	int(10) unsigned	YES		1	
created	timestamp			current_timestamp	
last_modified	datetime	YES		NULL	

logins	int(10) unsigned			0
last_login	datetime	YES		NULL
last_ip	float	YES		NULL
user_agent	varchar(40)	YES		NULL
token	varchar(40)	YES	UNI	NULL
token_created	datetime	YES		NULL
token_expires	datetime	YES		NULL
hybridauth_provider_name	varchar(40)	YES	UNI PT1	NULL
hybridauth_provider_uid	varchar(245)	YES	UNI PT2	NULL
signature	varchar(245)	YES		NULL

- Indexes

Table 7 - User table indexes

Keyname	Type	Cardinality	Field
PRIMARY	PRIMARY	1	id_user
ob_users_UK_email	UNIQUE	1	email
ob_users_UK_token	UNIQUE	1	token
ob_users_UK_seoname	UNIQUE	1	seoname
ob_users_UK_provider_AND_uid	UNIQUE	01/01/2014	hybridauth_provider_name hybridauth_provider_uid

3.4.3 Table ob_bookings

Information about all bookings generated.

- Fields

id_booking: Unique field used as identifier.

id_user: User ID assigned to the booking. Users are defined in ob_users table.

id_product: Product ID assigned to the order. Products are defined in ob_prodcuts table.

id_coupon: Coupon ID assigned to the order (if any). Coupons are defined in ob_coupons table.

paymethod: Payment method.

created: Date and time of order creation.

currency: Currency code used for payment.

amount: Order amount.

ip_address: Client IP address. Stored as Long.

txn_id: Paypal transaction ID

pay_date: Date and time of order payment.

status: Status of the order.

0 – Default creation status.

1 – Order paid.

5 – Tried to paid but not succeed.

99 – Refunded order.

notes: Booking notes

- Usage

Table “ob_booking” is used to record booking data. For any booking request, this table is updated. It is implemented to record information about payment as well.

- Relations

Each booking has a user and product assigned.

```
/**
 * @var array ORM Dependency/hierarchy
 */
protected $_belongs_to = array(
    'product' => array(
        'model' => 'product',
        'foreign_key' => 'id_product',
    ),
    'user' => array(
        'model' => 'user',
        'foreign_key' => 'id_user',
    ),
);
```

- Schema

Tables 8 and 9 show more detailed “ob_booking” table structure, its data types, restrictions and special options

Table 8 - Booking table schema

Field	Type	Null	Key	Default	Extra
id_booking	int(10) unsigned		PRI		auto_increment
id_user	int(10) unsigned		IND		
id_product	int(10) unsigned	YES		NULL	
id_coupon	int(10) unsigned	YES		NULL	
paymethod	varchar(20)	YES		NULL	
created	timestamp			current_timestamp	
currency	char(3)				
amount	decimal(14,3)			0	
ip_address	float	YES		NULL	
txn_id	varchar(255)	YES		NULL	
pay_date	datetime	YES		NULL	
status	tinyint(1)		IND	0	

notes	varchar(245)	YES	NULL	
-------	--------------	-----	------	--

- Indexes

Table 9 - Booking table indexes

Keyname	Type	Cardinality	Field
PRIMARY	PRIMARY	0	id_order
ob_orders_IK_id_user	INDEX	none	id_user
ob_orders_IK_status	INDEX	none	status

3.4.4 Table ob_product

Information about all products created. This information need to be manually introduced by owner.

- Fields

id_product: Unique field used as identifier.

id_user: User ID who created the product.

id_category: Category ID assigned to the product.

id_location: Location ID assigned to the product.

title: Title of the product.

seotitle: SEO friendly title of the product. Must be unique.

url_demo: URL of product demo.

description: Full description of the product.

email_purchase_notes: Notes that user will receive by email when complete purchase.

currency: Currency used for price.

price: Bookin fee.

price_vendor: Product price. Used for selling products/

price_offer: Price used as an offer.

offer_valid: Offer expiration date.

featured: Date until the product will be featured.

created: Date and time when product was added.

updated: Date and time when product was updated.

status: Product status.

0 is inactive (not displayed).

1 is active.

has_images: If the product has images.

0 if no images, 1 if has at least one image.

rate: Product rate based on reviews.

affiliate_percentage: Commission percentage that affiliate will have for the product.

- Usage

Products can be added, edited, removed or change status at `oc-panel/product`.

- Relations

Each product have assigned a user, category, and location.

```
protected $_belongs_to = array(  
    'user' => array(  
        'model' => 'user',  
        'foreign_key' => 'id_user',  
    ),  
    'category' => array(  
        'model' => 'category',  
        'foreign_key' => 'id_category',  
    ),  
    'location' => array(  
        'model' => 'location',  
        'foreign_key' => 'id_location',  
    ),  
);
```

Tables 10 and 11 show more detailed “ob_booking” table structure, its data types, restrictions and special options

Table 10 - Product table schema

Field	Type	Null	Key	Default	Extra
id_product	int(10) unsigned		PRI		auto_increment
id_user	int(10) unsigned	YES		NULL	
id_category	int(10) unsigned	YES		NULL	
id_location	int(10) unsigned	YES		NULL	
title	varchar(145)				
seotitle	varchar(145)				
url_demo	varchar(145)				
description	text				
email_purchase_notes	text				
currency	char(3)				
price	decimal(10,2)			0	
price_vendor	decimal(10,2)			0	
price_offer	decimal(10,2)			0	
offer_valid	datetime	YES		NULL	
featured	datetime	YES		NULL	
created	timestamp			current_timestamp	
updated	datetime	YES		NULL	
status	tinyint(1)			0	
has_images	tinyint(1)			0	
rate	float(4,2)	YES		NULL	
affiliate_percentage	decimal(14,3)			0	

- Indexes

Table 11 - Product table indexes

Keyname	Type	Cardinality	Field
PRIMARY	PRIMARY	0	id_product
oe_products_IK_id_user	INDEX	none	id_user
oe_products_IK_id_category	INDEX	none	id_category
oe_products_IK_id_location	INDEX	none	id_location

3.5 INSTALLATION PROCESS

Kuul-dev Bookings comes with a very easy installation process. There are few things required from the user in order to make installation successful. Above all, most important is Hosting plan or Local host server configured correctly. We will discuss about the process of installing as well as optimal requirements for “Kuul-dev bookings”.

3.5.1 User requirements

- Domain name

Domain name, commonly, is included in most of the hosting plans, user can pick any. Once the software is installed, routing API is bound to this domain name. Except in a case of domain migration, extra steps are required to insure correct application functioning.

Moreover, in case of localhost, instead of Domain name we can configure pseudo-links in `httpd-vhosts.conf` file following this steps from many online tutorials found on the web, of course dependant on operating system.

Or simply using apache localhost server ports `http://localhost:8080/name_of_installation_folder`.

- Compatible Hosting or localhost server

Requirements are the same in both cases and are necessary for proper functioning of the software. Following list describes environment minimal requirements:

1. Apache 2+
2. PHP 5.3+
3. Short Tags
4. GD support
5. mod_rewrite
6. mcrypt
7. Gettext
8. Curl
9. MySQL 5+

- Uploading files

This step can be carried in few ways. In case of hosting, we can use FileZilla FTP transport Client, or if hosting have its own File Manager this is even easier way of uploading files to the server.

In case of FTP clients (e.g. FileZilla), we need to connect to the server using server connection information:

1. Hostname
2. Username
3. Password
4. Port (optional)

This information is provided by hosting in the section FTP info!

Moreover, in case of localhost, we just need to pick the place where to install the software, copy files and link our project to that path.

- Create Database

Each hosting allows multiple databases, we need to create one (name is optional). And later on, in the installation we link to it.

Moreover, in case of localhost, XAMPP or any other environment tool, we should use phpMyAdmin (usually <http://localhost/phpmyadmin>) control panel and MySQL database. So, we just need to do the same there, as previously described.

3.5.2 Installation use case

After the successful file upload is done. We have to the link (domain) where our software is hosted. The window will open installation form (Figure 28).

The screenshot displays a web-based installation form with a green header bar. The header contains navigation links: 'Install', 'Support', 'Requirements', and 'About'. On the right side of the header, there is a button that says 'We install it for you. Buy now!'. Below the header, there is a light green banner with the text 'Get Hosting! Less than \$5 Month'. The main content area is divided into two columns. The left column is titled '1. Site Configuration' and contains several input fields: 'Site Language' (a dropdown menu with 'en_US' selected), 'Site Name' (a text input field with 'Site Name' as a placeholder), 'Time Zone' (a dropdown menu with 'Los_Angeles [-08:00]' selected), 'Administrator email' (a text input field with 'your@email.com' as a placeholder), and 'Admin Password' (a text input field). The right column is titled '2. Database Configuration' and contains a heading 'How to create a MySQL database?'. Below this heading are several input fields: 'Host name' (a text input field with 'localhost' as a placeholder), 'User name' (a text input field with 'root' as a placeholder and a tooltip that says 'User name'), 'Password' (a text input field), 'Database name' (a text input field with 'openclassifieds' as a placeholder), and a checkbox labeled 'Sample data' which is checked. At the bottom of the form, there is a dark blue button labeled 'Install'.

Figure 28 - Installation form

- Site language
Pick one of the languages from the list. Kuul-dev Bookings, for now, supports only English and Spanish.
- Site Name
Represents a name that the site will use as a meta title. And SEO linking, that google uses to find us.
- Time Zone
Time zone will configure the software to use appropriate time format. In the select are listed most used nowadays.
- Administrator email
This field will register first user and give him all access (Admin rights).
- Admin Password
It sets the password for Admin User
- Host name
Hosting company provides us with this info, it's the unique name of our server.
- User name
Registered user of the Hosting using, usually hosting provider adds one by default. But it can be added more.
- Password
This is database password we have set when creating new db.
- Database name
Database name, we spoke of this step above. And this name needs to exist and consists with password.
- Simple Data
If checked, this field will populate some dummy data while installing. It's good to have it, if first time using this software.

If all data is filled correctly and requirements are met, installation should be successful. And reloading the page will open fresh and ready to use web platform for booking.

3.5.3 Implementation

Installation scripts and views are located in a root/install folder. This system works as a separate module, which is removed once installation is done. And no more relevant to functioning of the system.

Inside of the install folder we can distinguish few different sections.

- Samples

Sample files are base or dummy files that are used to replace original configuration files.

1. auth.php
2. database.php
3. example.htaccess
4. install.sql.php
5. robots.txt

From this previous list, 2 most important files are database.php:

```
return array
(
    'default' => array(
        'type' => 'mysqli',
        'connection' => array(
            'hostname' => '[DB_HOST]',
            'username' => '[DB_USER]',
            'password' => '[DB_PASS]',
            'persistent' => FALSE,
            'database' => '[DB_NAME]',
        ),
        'table_prefix' => '[TABLE_PREFIX]',
        'charset' => '[DB_CHARSET]',
        'profiling' => (Kohana::$environment===Kohana::DEVELOPMENT)?
TRUE:FALSE,
    ),
);
```

Sets the configuration of the database connection.

And install.sql.php, used to create tables and populate data to newly created database.

Following code is an example of the template used to create table, in this case table “products”:

```
mysqli_query($link,"CREATE TABLE IF NOT EXISTS
`.core::request(TABLE_PREFIX)."products` (
  `id_product` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `id_user` int(10) unsigned DEFAULT NULL,
  `id_category` int(10) unsigned DEFAULT NULL,
  `id_location` int(10) unsigned DEFAULT NULL,
  `id_slot` int(10) unsigned DEFAULT NULL,
  `title` varchar(145) NOT NULL,
  `seotitle` varchar(145) NOT NULL,
  `url_demo` varchar(145) NOT NULL,
  `description` text NOT NULL,
  `email_purchase_notes` text NOT NULL,
  `currency` char(3) NOT NULL,
  `price` decimal(10,2) NOT NULL DEFAULT '0',
  `price_vendor` decimal(10,2) NOT NULL DEFAULT '0',
  `price_offer` decimal(10,2) NOT NULL DEFAULT '0',
  `offer_valid` DATETIME NULL,
  `coupons` tinyint(1) NOT NULL DEFAULT '0',
  `featured` DATETIME NULL,
  `created` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
  `updated` DATETIME NULL,
  `status` tinyint(1) NOT NULL DEFAULT '0',
  `has_images` tinyint(1) NOT NULL DEFAULT '0',
  `confirm_file` varchar(40) DEFAULT NULL,
  `rate` FLOAT( 4, 2 ) NULL DEFAULT NULL,
  `affiliate_percentage` decimal(14,3) NOT NULL DEFAULT '0',
  PRIMARY KEY (`id_product`),
  KEY `".core::request(TABLE_PREFIX)."products_IK_id_user` (`id_user`),
  KEY `".core::request(TABLE_PREFIX)."products_IK_id_category`
(`id_category`)
) ENGINE=MyISAM DEFAULT CHARSET=".core::request(DB_CHARSET).";";
```

Our next example shows insert of “pre-set” data, of the user access role / permissions table:

```
mysqli_query($link,"INSERT INTO `".core::request(TABLE_PREFIX).`roles` (`id_role`,`name`,`description`) VALUES (1, 'user', 'Normal user'), (5, 'translator', 'User + Translations'), (10, 'admin', 'Full access');");
```

```
mysqli_query($link,"INSERT INTO `".core::request(TABLE_PREFIX).`access` (`id_access`,`id_role`,`access`) VALUES (1, 10, '*.*'), (2, 1, 'profile.*'), (3, 1, 'stats.user'), (8, 1, 'support.*'), (4, 5, 'translations.*'), (5, 5, 'profile.*'), (6, 5, 'stats.user'), (7, 5, 'content.*');");
```

As well as, we insert “pre-set” **email** templates used, **config** table with lots of pre-set data. And if Sample data checkbox from installation is checked we add some “dummy” data to locations and categories table.

- Views

This is where we add all the pages needed for installation, in our case we have 4 pages.

1. form.php
2. hosting.php
3. requirements.php
4. success.php

Note: hosting.php is under development, but intention is to use it for purposes of advertising affiliate Hosting plans. This is a nice way of earning money.

Other files are self-explanatory. And all contain standard HTML mock-up structure with some CSS.

- class.install.php

This file is very important, in short contains small helper scripts and default configuration variables necessary for installation. As well as, logic for the installation process. Following Tables (12, 13, 14), define API of this file.

Table 12 - Default Installation variables

Name	Definition	Description
const VERSION	1.0.1	Software verison
public static \$locale	en_US'	default locale/language of the install
public static \$url	NULL	suggested URL with folder were to install
public static \$folder	NULL	suggested folder were to install
public static \$msg	string	message to notify
public static \$error_msg	string	installation error messages here
public static \$hash_key	string	used to hash the password and set in the

Table 13 - Class Install API

Name	Return value	Parameters	Description
public static function initialize()	void		initializes the install class and process
public static function requirements()	array		checks that our hosting has everything that needs to have
public static function is_compatible()	bool		checks from requirements if its compatible or not. Also fills the msg variable
public static function view(\$file)			includes a view file
public static function phpinfo()	string		get phpinfo clean in a string
private static function gettext_init(\$locale,\$domain = 'messages',\$charset = 'utf8')	string	string, string, string	loads gettexts or droppin
public static function get_select_timezones(\$select_name='TIMEZONE', \$selected=NULL)	string	string, string	return HTML select for the timezones
public static function replace_file(\$orig_file,\$search, \$replace,\$to_file = NULL)	bool	srting, array, array	replaces in a file
public static function execute()	bool		installs the software

Table 14 - Class core API

Name	Return value	Parameters	Description
public static function generate_password(\$length = 16)	string	int	generate_password
public static function get_browser_favorite_language(\$lang = 'en_US')	string	string	Parse Accept-Language HTTP header to detect user's language(s) and get the most favorite one
public static function format_offset(\$offset)	string	string	gets the offset of a date
public static function get_timezones()	array		returns timezones ins a more friendly array way, ex Madrid [+1:00]
public static function slug(\$s)	string	string	cleans an string of spaces etc
public static function write_file(\$filename,\$content)	bool	string, string	write to file
public static function delete(\$file)	void	\$_FILE	deletes file or directory recursevely
public static function rss(\$url)	array	string	rss reader
public static function get(\$key,\$default=NULL)	string	string, string	shortcut for the query method \$_GET
public static function post(\$key,\$default=NULL)	string	string, string	shortcut for \$_POST
public static function request(\$key,\$default=NULL)	string	string, string	shortcut to get or post

- Index.php

This is default HTML container, with all CSS and JS code that is rendered at the installation process. It contains mock-up structure, and at the beginning makes calls to all necessary scripts mentioned before.

Note: When doing fresh install, in the install folder there is one extra file install.lock. This file is empty. It is used only to detect new installations. After the installation, this file is deleted and all connections to install module are broken.

3.6 UPDATE SYSTEM

Kuul-dev bookings comes with a very easy update system. Everything is automatized and summarized in one script. And it's could be accomplished with one button click.

3.6.1 Use case

To update, proceed to Panel->Tools->Updates->check for updates. Then a button that says "Update" appears. Clicking on the indicated button tigers the process automatically.

If errors occur, and prevent usual update flow from executing. If that happens manual update should be followed, and certain steps need to be followed.

1. Download new software version manually
2. Extract and open files (NOT in root folder)
3. Upload / copy-paste only this folders
 - a. oc/classes/
 - b. oc/modules/
 - c. themes/
 - d. languages/ (optional, in case new languages are added, and/or we want new languages)

Done! This files will be overwritten and new added.

3.6.2 Implementation

Update class is composed from set of actions that will be executed recursively.

Path: oc/classes/controller/panel/update.pho

Essentially, update.php is a controller implemented slightly different than other controllers.

- action_index

This is where the application renders the view of our update system. Following code, is a verbatim copy of this action.

```
public function action_index()
{
    //force update check reload
    if (Core::get('reload')==1 )
        Core::get_updates(TRUE);

    $versions = core::config('versions');
    if (Core::get('json')==1)
    {
        $this->auto_render = FALSE;
        $this->template = View::factory('js');
        $this->template->content = json_encode($versions);
    }
    else
    {
        Breadcrumbs::add(Breadcrumb::factory()-
>set_title(__('Updates')));
        $this->template->title = __('Updates');

        //check if we have latest version of OC
        if (key($versions)!=core::VERSION)
            Alert::set(Alert::ALERT, __('You are not using latest
version of OC, please update.')).
                <br/><br/><a class="btn btn-primary
update_btn" href="'.Route::url('oc-
panel',array('controller'=>'update','action'=>'latest')).'">
                .__('Update').</a>;

        //pass to view from local versions.php
    }
}
```

```

        $this->template->content = View::factory('oc-
panel/pages/tools/versions', array('versions' =>$versions,
'latest_version' =>key($versions)));
    }
}

```

- version actions [action_11]

We don't discuss one action here, moreover, it is set of actions. And each one implements one software version. Containing files to replace, configs to update and add extracts.

```

public function action_11()
{
    // build array with new (missing) configs
    $configs = array(array('config_key' =>'thanks_page',
        'group_name' =>'payment',
        'config_value' =>''),
        array('config_key' =>'blog',
        'group_name' =>'general',
        'config_value' =>'0'),
        array('config_key' =>'blog_disqus',
        'group_name' =>'general',
        'config_value' =>''));

    // returns TRUE if some config is saved
    $return_conf = Model_Config::config_array($configs);
}

```

This function defines the content to be added to implement new functionalities introduced. We can add inside more than just simple config table update. Basically, all that is changed, and it doesn't involves actual physical files, but DB.

- action_latest

This action is a “trigger action”. It actually contains flow of execution and logic that updates the software.

Flow of execution:

1. activate maintenance mode
2. get latest version
3. set update directory
4. set uploading file name
5. check if file name exists, delete if it does
6. check if directory exists, create if it doesn't
7. verify we could get the zip file, and write the file if we can
8. open and extract zip
9. set files to be replaced / move specific files
10. start pasting to appropriate locations
11. delete downloaded files when all is done

This is the list of mandatory files and folders to be updated:

```
//files to be replaced / move specific files
$copy_list = array(
    'oc/config/routes.php',
    'oc/classes/',
    'oc/modules/',
    'oc/vendor/',
    'oc/bootstrap.php',
    'oc/ko323/',
    'themes/',
    'languages/',
    'index.php',
    'embed.js',
    'README.md',
);
```

Note: Of course this list could be extended, if system extends.

3.7 DOMAIN LAYER AND API

Following Figure 29 is representing class diagram of all Models and relations between them.

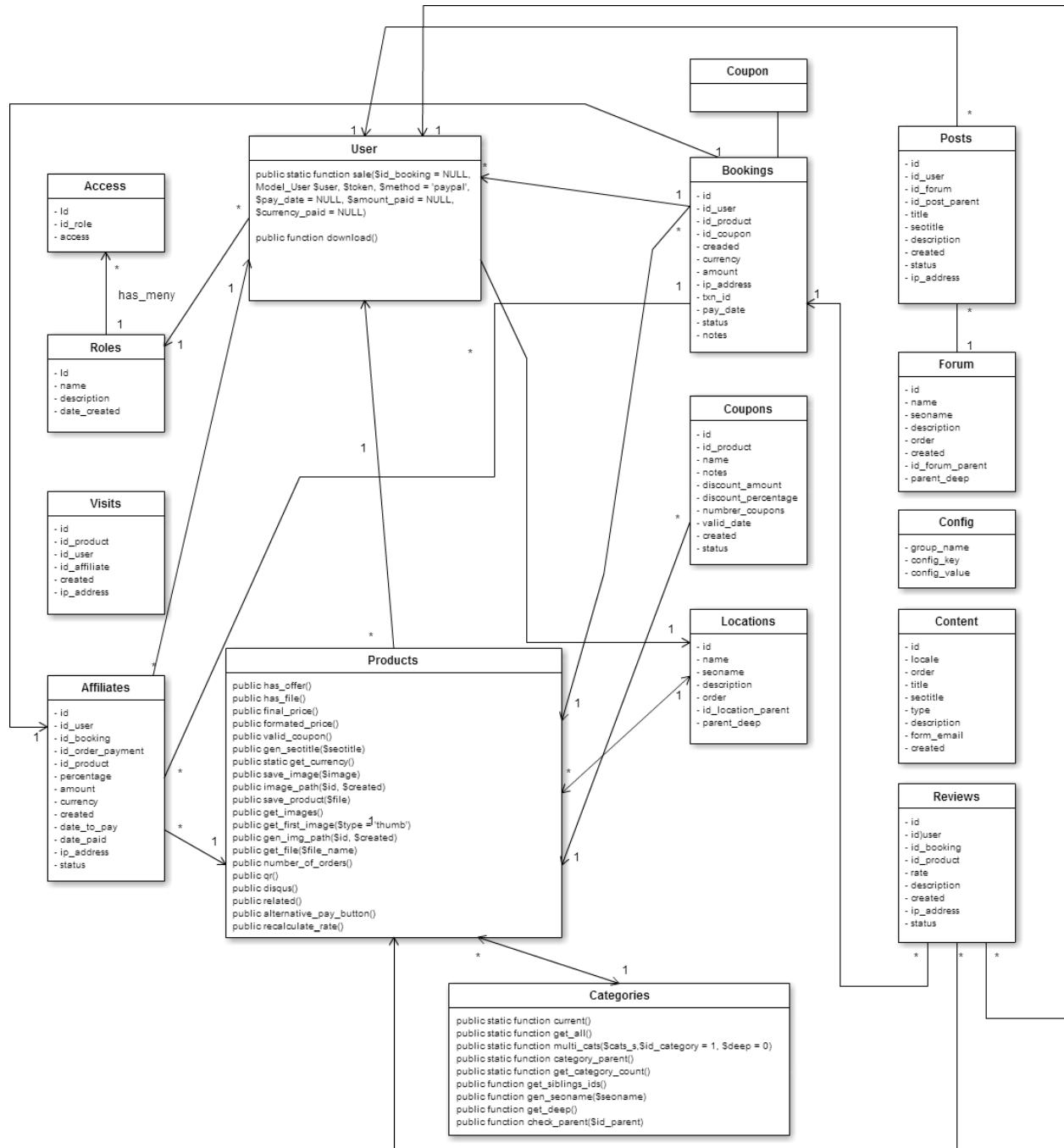


Figure 29 - Domain layer Model class diagram

3.7.1 Model Product

Product is a core feature of “Kuul-dev bookings”. It consists on information that will be presented for the user, as information about what they are booking. There are two different type of data.

Use case

Here are the steps to add a product:

- Log in to admin panel
- Go to products and press “New”

Once done we will get to a form that needs to be filled with the product information (not all fields are mandatory), I will explain each point in that form and what it does exactly (there are some very useful boxes here)

The screenshot shows the 'New Product' form in the eShop admin panel. The form is divided into three main sections: General information, Details, and Additional information. The General information section includes fields for Category, Currency, Title, and Description. The Details section includes Price, Price Offer, Offer Valid, License Days, and Support Days. The Additional information section includes URI demo, Version, Skins, Feature product, and Purchase notes. Below these sections is an 'Upload files' section with five image selection boxes and a digital file upload area. A 'Save' button is located at the bottom right of the form.

Figure 30 - Product create view

General or mandatory, such as:

1. **Title:** Booking product name.
2. **Category.** Relevant category from the options that we already created in the categories section. Serves to organize products by certain category.
3. **Currency.** Relevant currency for the product.
4. **Description.** Text for customers to give basic information about the product. And understand better what they are paying for

Extra or Optional details:

1. **Booking Price.** Is used for charging user's standard fee after booking is done. If left empty, booking is free
2. **Product Price.** It is a price set if product have a buy-out price. This is extra feature that allows users to do both business variations. Renting and selling.
3. **Price Offer.** When we want to make an offer on that product (it would override the original price for a specific period of time).
4. **Offer Valid.** The end date of the price offer (after that date the original price will be active).
5. **Url demo.** If there is some demo version, or external website that we want to redirect user to.
6. **Feature product.** Makes product pops on the front page, giving extra importance to it.
7. **Purchase notes, sent via email.** This section allows users to send some information after booking is done to let them know of some extra information and maybe give send some material important for booking.

Note: this is different from confirmation email.

Finally, if all information was filled, clicking save button will publish the product. To show and hide a product is simple, by checking and un-checking the "Active" check-box at the bottom.

API

- Model

Path: *oc/classes/model/product.php*

Following code describes table configuration in model Product:

```
/**
 * status constants
 */
const STATUS_NOACTIVE = 0; //not displayed
const STATUS_ACTIVE   = 1;

/**
 * @var string Table name
 */
protected $_table_name = 'products';

/**
 * @var string PrimaryKey field name
 */
protected $_primary_key = 'id_product';
```

Following code describes relations between model Product and other models:

```
protected $_belongs_to = array(
    'user' => array(
        'model'      => 'user',
        'foreign_key' => 'id_user',
    ),
    'category' => array(
        'model'      => 'category',
        'foreign_key' => 'id_category',
    ),
    'location' => array(
        'model'      => 'location',
        'foreign_key' => 'id_location',
    ),
);
```

Table 15 defines complete model product API:

Table 15 - Model product API

Name	Return value	Parameters	Description
public has_offer()	bool		returns if the product is in offer
public has_file()	bool		returns if the product has a file to download
public final_price()	float		returns the price of the product checking if there's an offer or coupon
public formated_price()	float		returns the price of the product formated using the product currency
public valid_coupon()	bool		validates if a coupon its added and valid for that product
public gen_seotitle(\$seotitle)	string	string	return the title formatted for the URL
public static get_currency()	array		returns allowed Paypal currencies
public save_image(\$image)	bool	array of \$_FILE	save_image upload images with given path
public image_path(\$id, \$created)	string	int, date	make unique dir path with a given date and id
public save_product(\$file)	bool	array of \$_FILE	upload images with given path
public get_images()	array		Gets all images
public get_first_image(\$type = 'thumb')	string	string	Gets the first image, and checks type of \$type
public gen_img_path(\$id, \$created)	string	int, date	Generate image path with a given parameters \$seotitle and date of advertisement creation
public get_file(\$file_name)	string	string	Gets the file by name
public number_of_orders()	integer		Number of product purchased
public qr()	sting		prints the QR code script from the view
public disqus()	string		prints the disqus script from the view
public related()	view		Query for related products
public alternative_pay_button()	string		renders a modal with alternative paymethod instructions
public recalculate_rate()	float		saves the rates recalculating it

- View

User view: *themes/default/views/pages/product/single.php*

Admin panel responsible views:

themes/default/views/oc-panel/products/index.php

themes/default/views/oc-panel/products/create.php

themes/default/views/oc-panel/products/update.php

- Controller

Client: *oc/classes/controller/product.php*

Table 16 represents public REST api. And table 17 private REST api.

Table 16 - Product public action list

Name	Description
action_view	Show minified and standard product view
action_review	Review page
action_demo	Show demo, if exists
action_listing	Show listing of all products
action_search	Show listing of all products, with search results

Admin: *oc/classes/controller/panel/product.php*

Table 17 - Product private action list

Name	Description
action_index	Listi of created products
action_create	Create product
action_update	Update / edit product
action_upload	Upload specific file

action_delete_file	Delete specific file
--------------------	----------------------

3.7.2 Model Category

Categories are useful when many products are used. They are made to sort products, label, categorized, and simplify user interaction.

Use case

The screenshot shows the 'New Category' form in an admin panel. The sidebar on the left contains various menu items, with 'Categories' highlighted and a red arrow pointing to it. The main content area displays the form with the following fields:

- Name:** A text input field.
- Order:** A dropdown menu with '1' selected.
- Id Category Parent:** A dropdown menu with 'Select an Option' selected.
- Parent Deep:** A dropdown menu with '0' selected.
- Seoname:** A text input field.
- Description:** A rich text editor with a toolbar containing bold, italic, underline, link, unlink, list, and image icons.

A blue 'Submit' button is located at the bottom left of the form area.

Figure 31 – Category creation view

Adding categories is essential, allowing classification of the product as a set. To add a new category proceed to admin panel, selecting top menu link “Bookings” dropdown list appears. Afterwards, selecting link Categories, and then button “New”. This set of actions, will open screen with form to fill in:

- **Name:** Choose a name of product category e.g. Themes, Services. Basically, this field is the most important, the rest are optional.
- **Order:** We can choose the order subcategories will be displayed within a parent category. It’s not obligatory – later we can use drag & drop to change the order.

- **ID category parent:** Choose under which one of existing main categories subcategory will be displayed. Choose Home Category while creating main category. Later categories can be easily moved to other parent.
- **Parent deep:** Shows how many levels deep newly created category is under the main one. This doesn't really have to be set. It will be adjusted automatically.
- **Seoname:** Seoname will be auto generated based on the name of category, but we can also type it if we want it to be different.
- **Description:** We can add few words about what is available in this category.

Press button “submit”. After submitting we should see the information: “Success. Item created. Please to see the changes delete the cache”. Continue creating new categories if necessary, delete cache after finishing to see the changes. To delete cache go to: Tools > Cache on the left sidebar and press ‘Delete all’. We can now visit the site to see the changes made. While adding categories we should remember that only 2 levels of categories will be displayed in the theme and be accessible to view from the main page.

Manage categories

Go to Panel – choose Categories on the left sidebar. Managing categories is super easy. If we want to move categories and change their order we need to drag and drop selected category to the chosen place. To change something e.g. name or description of the category we can click Edit button. To delete category press red button with trash bin. Note that when we delete parent category, subcategories inside of it will be moved level up – to the parent of the deleted category.

Categories widget

Additional options to deal with categories are given by special widget. To activate it go to Panel and choose Widgets on the left sidebar. Choose ‘Categories’ widget from the list and click create. Name the widget’s title and select if we want to display it in a sidebar or footer. Alternatively, we can also keep it inactive. Thanks to this widget navigation between categories is easier. List of categories will be displayed all the time at the side or in the bottom of the page.

API

- Model

Root: *oc/classes/model/category.php*

Following code describes table configuration in model Category:

```
protected $_table_name = 'categories';  
  
protected $_primary_key = 'id_category';
```

Following code describes relations between model Category and other models:

```
/**  
 * @var array ORM Dependency/hierarchy  
 */  
protected $_has_many = array(  
    'products' => array(  
        'model' => 'product',  
        'foreign_key' => 'id_category',  
    ),  
);  
  
protected $_belongs_to = array(  
    'parent' => array('model' => 'Category',  
        'foreign_key' => 'id_category_parent'),  
);
```

Table 18 defines complete model product API:

Table 18 - Model Category API

Name	Return value	Parameters	Description
public static function current()	model		returns the current category
public static function get_all()	array		we get the categories in an array and a multidimensional array to know the deep
public static function multi_cats(\$cats_s,\$id_category = 1, \$deep = 0)	array	array, int, int	gets a multidimensional array wit the categories
public static function category_parent()	array		returns a list of of top categories
public static function get_category_count()	array		counts the categorie products
public function get_siblings_ids()	array		returns all the siblings ids+ the id_category, used to filter the products
public function gen_seoname(\$seoname)	string	string	return the title formatted for the URL
public function get_deep()	int		returns the deep of parents of this category
public function check_parent(\$id_parent)	int	int	rule to verify that we selected a parent if not put the root location

- View

Admin path: themes/default/views/oc-panel/pages/categories.php

- Controller

Admin path: *oc/classes/controller/panel/category.php*

Table 19 represents public REST api.

Table 19 - Model Category public action list

Name	Description
action_index	Listi of created categories
action_saveorder	Changes the order of categories
action_update	Update / edit category
action_multy_categories	Creates multiple categories just with name
action_delete	Delete category

Note: Action create is missing, this is since Kohana has system for creating forms. Called Kohana Form Builder. In the model it's important to set few parameters. Since Kohana assumes all database fields to be part of the form. This configuration is done in Model.

Set rules for Validation:

```
/**
 * Rule definitions for validation
 *
 * @return array
 */
public function rules()
{
    return array(
        'id_category' => array(array('numeric')),
        'name' => array(array('not_empty'), array('max_length',
array(':value', 145))),
        'order' => array(),
        'id_category_parent' => array(),
        'parent_deep' => array(),
        'seoname' => array(array('not_empty'),
array('max_length', array(':value', 145))),
        'description' => array(array('max_length', array(':value',
255))),
    );
}
```

Following code will set Label names and assign them to input:

```
/**
 * Label definitions for validation
 *
 * @return array
 */
public function labels()
{
    return array(
        'id_category' => __('Id'),
        'name' => __('Name'),
        'order' => __('Order'),
        'created' => __('Created'),
        'id_category_parent' => __('Parent'),
        'parent_deep' => __('Parent deep'),
        'seoname' => __('Seoname'),
        'description' => __('Description'),
    );
}
```

Following code will add filtering, or special conditions:

```
/**
 * Filters to run when data is set in this model. The password filter
 * automatically hashes the password when it's set in the model.
 *
 * @return array Filters
 */
public function filters()
{
    return array(
        'seoname' => array(array(array($this, 'gen_seoname'))),
        'id_category_parent' => array(array(array($this, 'check_parent'))),
    );
}
```

[Optional] Exclude certain fields:

```
public function exclude_fields()
{
    return array('created');
}
```

Finally, adding specific types of form inputs:

```
/**
 * formmanager definitions
 */
public function form_setup($form)
{
    $form->fields['description']['display_as'] = 'textarea';
    $form->fields['id_category_parent']['display_as'] = 'select';
    $form->fields['id_category_parent']['caption'] = 'name';
    $form->fields['order']['display_as'] = 'select';
    $form->fields['order']['options'] = range(1, 100);
}
```

3.7.3 Model Booking

This is where the magic gets done. This table contains much more information than the user sees. And it is getting filled during the whole process of booking one product. Moreover, the admin can organize and maintain this bookings, and correspond appropriately using a nice interface set in his admin space.

Use case

Any new purchase/booking that is done on the website hosted is added as a booking at the admin panel, which is useful. Especially to access information about new sales, and possibly manage them.

But sometimes in case when purchase is not complete, or there have been some errors in the payment flow. Or we might want to register a cash payment and add it to sales statistics. In those cases here are the steps to do so:

- Log in to admin panel
- go to **tab** – > **Bookings** and press on “**New**”

Here we need to fill in the following fields:

Name: the preferred name of the booking or client.

Email: The e-mail the product will be sent to and registered with, this field is very important.

Pay method: How we would like to register the payment method for the order.

Product: Will be selected from a drop-down menu containing all of your products

Currency: Self-explanatory. Ex USD.

Amount: The payment we want to register

Pay Date: Pick the day we want to register the sale at.

Notes: We have 245 characters to put whatever text we wish to add for future references.

API

- Model

Path: *oc/classes/model/booking.php*

Following code represents table configuration of model Booking:

```
protected $_table_name = 'bookings';

protected $_primary_key = 'id_booking';
```

Following code represents relations between model Booking and other models:

```
/**
 * @var array ORM Dependency/hierarchy
 */
protected $_belongs_to = array(
    'product' => array(
        'model' => 'product',
        'foreign_key' => 'id_product',
    ),
    'user' => array(
        'model' => 'user',
        'foreign_key' => 'id_user',
    ),
    'coupon' => array(
        'model' => 'coupon',
        'foreign_key' => 'id_coupon',
    ),
);

protected $_has_one = array(
    'affiliate' => array(
        'model' => 'affiliate',
        'foreign_key' => 'id_booking',
    ),
);
```

Following code sets the global payment statuses (constants) to be used in booking authentication:

```

const STATUS_CREATED           = 0;    // just created
const STATUS_PAID              = 1;    // paid!
const STATUS_REFUSED           = 5;    //tried to paid but not succeed
const STATUS_FRAUD             = 66;   //fraud!!!!
const STATUS_REFUND            = 99;   //we refunded the money

```

Table 20, defines model booking API.

Table 20 - Model Booking API

Name	Return value	Parameters	Description
public static function sale(\$id_booking = NULL, Model_User \$user, Model_Product \$product, \$token, \$method = 'paypal', \$pay_date = NULL, \$amount_paid = NULL, \$currency_paid = NULL)	void	int, User, Product, string, string, date, int, string	creates new bookings for a product
public function download()	file		downloads product attached to this booking if there's one

- View

User view: *themes/default/views/pages/product/single.php*

Admin views:

themes/default/views/oc-panel/booking/index.php

themes/default/views/oc-panel/booking/create.php

themes/default/views/oc-panel/booking/update.php

themes/default/views/oc-panel/booking/import.php

- Controller

Path: *oc/classes/controller/panel/booking.php*

Table 21, defines public REST api for booking.

Table 21 - Booking Controller actions

Name	Description
action_index	Listi of created categories
action_create	overwrites the default crud index
action_update	Update / edit category
action_import	Adds file for product, that the user downloads after booking
action_book	Saves booking information

3.8 AUTOMATIC EMAIL SYSTEM

Emails are core functionality of almost every application nowadays. We all interact with them on a daily basis. And “Kuul-dev Bookings” is no different. This feature was developed for Open-Classified 2 and then in this project extended and improved.

When user of “Kuul-dev” does certain actions, e.g. register, book product or request to change the password, he receives an automatic email for confirmation.

To see and manage those emails, we can go Admin Panel and choose Content > Email from the left sidebar. Email templates are already created and translated to supported languages. We can edit them according to our needs by clicking blue button next to the template.

3.8.1 Available templates

Table 22 - Email templates

Title	Name	Description
Change Password	auth.remember	welcoming email sent to user after completing registration, reminding data for logging it
Welcome to [SITE.NAME]!	auth.register	mail informing that user have been contacted regarding his advertisement, quoting the message
Hello [USER.NAME]!	user.contact	sent to user to notify when another user is contacting them directly via their profile
Hello [USER.NAME]!(userprofile.contact	message sent during registration of the new user providing information about how to log in for the first time with a temporary autogenerated password
Hello [USER.NAME]!	user.new	message sent to admin when a visitor uses a contact form
[EMAIL.SENDER] wants to contact you!	contact.admin	message notifying that the message was created and informing how to edit it and that it still have to be validated by administrator

This is just most relevant and basic templates, and more will come Ex: Booking confirmation etc. Because of lack of time, and low relevance at this stage I decided not to write more.

3.8.2 Newsletters

Kuul-dev Bookings offers useful functionality that enables us to send the newsletter to all registered users. We can interact with newsletters in Panel and choose Content > Newsletters from the left sidebar.

This screen contains form to be filled, before sending a newsletter. Following list discusses more about each input field used to successfully send newsletter

- **To:** different groups of users to who we want to send

This feature is implemented with set of queries that returns array of user emails

Example:

```
$query = DB::select('email')->select('name')
->from(array('users','u'))
->join(array('orders','o'))
->using('id_user')
->where('o.status','=',Model_Booking::STATUS_PAID)
->group_by('u.id_user')
->execute();
```

- **From:** Our email display name
- **From Email:** display email and in case of reply this email will be used
- **Subject:** self-explanatory
- **Message:** actual text, and we can use text formatting that will be translated in HTML code. In case of newsletter template, we can copy-paste it here!

3.9 TRANSLATIONS SYSTEM

The majority of this system was developed for purposes of “Open Classifieds 2”, but in this project has been some improvements. Its worth of mentioning it, since it adds a big flexibility to the system in general. Allowing us to translate very easy, and or add more locales and language translations.

Adding new locale to the system is now fully automatized and synchronizes with time zone, money format and language of the user.

3.9.1 How it works

- Open “Kuul-dev bookings” folder/translations/
- Create a copy of one of the existing translation files
- Rename the new file to the language we want to translate to e.g. nl_NL
- Proceed to admin panel > Content > Translations
- Pick the language that is created (as shown in the picture below) and press on “edit”
- Fill the translation boxes with the new text

#	Original Translation	Translation en_US	Save
0	# items to display	# items to display	Save
1	Accept Terms Alert		Save
2	Access not allowed	Access not allowed	Save
3	Actions	Actions	Save

Figure 32 - Translation edit view

- Save
- Send us the file after everything is done (would help us a lot)

There are some issues that can arise with translating. Such as:

- Hosting is not supporting locales
- Permission files .po and .mo should have 755 (read/write)
- Different charset collection, depending on locale

Adding and changing text is supported by Kohana framework with I18n (**I**nternationalization: *i*, then 18 letters, then *n*) system.

As well Kohana has a `__()` function to do translations for us. This function is only meant for small sections of text, not entire paragraphs or pages of translated text.

For example:

```
<?php echo __ ('Hello, world!');?>
```

This will echo “Hello, world!” unless we changed the defined language, which is explained above.

Kohana deals with translations by returning array with translated strings. Since this is too static we made user interface and synchronized it with rest of related functionalities.

3.9.2 Implementation

Translation system does not contain Model, since this feature is covered by Kohana framework. It was decided to added script's to deal with CRUD (create update delete) actions.

- View

Paths:

themes/default/oc-panel/pages/translations/edit.php

themes/default/oc-panel/pages/translations/index.php

- Controller

Path: oc/classes/controller/panel/translations.php

Table 23 - Translation Controller action list

Name	Description
action_index	List of locale languages in the system
action_edit	Edit language
__construct	extends parent functionality coming from Kohana

3.10 WIDGETS

This section discusses design, implementation, and functioning of “Widgets” service. This service was implemented to allow users to create and display simple auto-generated information on the screen. Desire behind this service is to make user interaction more accessible.

3.10.1 Use case Widget

Widgets can be found in the admin panel under "widgets" link. There are many useful pre-set widgets:

- **Categories** Shows categories we have, and links that will filter ads by selected category.
- **Locations** Shows locations we have, and links that will filter ads by selected location.
- **Disqus** Is plugin that allow users to post small comments under posted advertisement. And this widget will display last few comment posted.
- **Map** Shows all pinpoints where ads have their address.
- **Pages** Display CMS pages
- **RSS** Displays RSS links
- **Share** Is widget to allow sharing in most popular social networks
- **Stats** Shows site stats in simple text form. Such as number of ads, views in total and number of registered users.
- **Text** Is the HTML textarea. Here we can copy/paste some custom html.

“Placeholders” represents section where widgets are printed on the screen. We have following three:

- **Sidebar** Widget will appear on the right side of the browser.
- **Footer** Widget will appear in footer of the page.
- **Header** Widget will appear in the header below the top menu

Note: all widgets can be edited, deleted, or placed in inactive field. Inactive field will preserve widgets and their configuration. But they would not be shown.

3.10.2 Use case: Create new widget

To create a new widget: navigate to the uri `oc-panel/widget/` and click create button of desired widget. Each widget has custom configuration. In this screen, we can select between two placeholders, give a title to widget add additional info etc. After data insertion is finished click save button and widget is created and shown in page.

Note: If widget does not appear, go to `oc-panel/tools/cache` and delete cache.

3.10.3 Implementation

This section discusses API and implementation. “Widgets” are implemented as the module. Moreover, each individual widget has its definition class extending main class `Widget`. Modules could be found in `oc/modules` directory.

3.10.4 Abstract Widget class

The following Table 24, discusses list of variables of Abstract widget class. This class is extended by all other widget classes.

List of default variables:

Table 24 – Default variable list of Abstract Widget Class

Name	Definition	Description
<code>public \$fields</code>	array	array fields the widget have, defined in the construct
<code>public \$data</code>	array	data stored for each field
<code>public \$banned_placeholder</code>	array	limit placeholders for this widget
<code>public \$title</code>	Widget Title'	widget title

public \$description	Widget description'	description what the widget does
public \$widget_name	NULL	unique name of the widget in the configs, used to retrieve it later
public \$placeholder	NULL	placeholder where the widget is stored
public \$created	NULL	when was created/saved
public \$loaded	FALSE	easy way to know if the widget is loaded

The following Table 25, discusses Abstract widget class API. Its definitions and gives a brief description for each function.

List of functions:

Table 25 - Abstract Widget Class API

Name	Return value	Parameters	Description
public static function factory(\$widget_name, \$load = TRUE)	widget	string, bool	generates an instance of the correct widget
public function load(\$widget_name, array \$widget_data = NULL)	bool	string, string	gets the fields value form the DB config
public function save(\$old_placeholder = NULL)	bool	string	saves current widget data into the DB config
public function delete()	bool		delete current widget data from the DB config
public function unload()	void		unload the widget
public function render()	html string		renders the widget view with the data
public function form()	html string		renders the form view to fill the data and then saves it
public function gen_name()	string		generates a name for this widget
public function id_name()	string		returns the name of the widget class
public function title(\$title = NULL)	string		get the title for the widget
public function before()	void		Automatically executed before the widget action. Can be used to set class properties, do authorization checks, and execute other custom code.
public function after()	void		Automatically executed after the widget action. Can be used to apply transformation to the request response, add extra output, and execute other custom code.
public function __set(\$name,	void	string, json	Magic methods to set

\$value)			
public function __get(\$name)	json	string	Magic methods to get

3.10.5 Helper class

Helper class is implemented to handle printing of the widgets. In fact, the logic of the entire class is declared static, to be able to print them from anywhere. Following Tables 26 and 27 are discussing API of the helper class.

List of default static variables:

Table 26 – Helper Widget Class default variables

Name	Definition	Description
public static \$theme_placeholders	array	rray of widget placeholders in theme, @ /themes/THEMENAME/init.php
public static \$default_placeholders	array	array of default placeholders, @ /modules/widgets/init.php
public static \$theme_widgets	array	array of widget specific to theme, @ /themes/THEMENAME/init.php
public static \$default_widgets	array	array of default widgets, @ /modules/widgets/init.php

List of helper functions:

Table 27 - Helper Widget class API

Name	Return value	Parameters	Description
public static function get(\$name_placeholder, \$only_names = FALSE)	array	string, bool	Gets from conf DB json object of active widgets
public static function get_widgets(\$only_names = FALSE)	array	bool	returns all the widgets
public static function get_installed_widgets()	array		get the widgets that he finds in the folder
public static function get_placeholders(\$only_names = FALSE)	array	bool	returns placeholders names + widgets

In addition, we have defined one class for each widget, that implements widget specific configuration. As well as, one View for each, that prints data on the screen.

3.11 TESTING

Testing is performed to verify that the completed project functions are according to the expectations defined by the requirements/specifications. The overall objective was not to find every software bug that exists, but to uncover situations that could negatively impact the customer, usability and/or maintainability.

As mentioned before, in this project, decision has been made that automated testing is not to be implemented in this iteration. Due to, great cost of development and complexity overall. Never the less, testing is important and unavoidable.

This project was tested accordingly, using 2 different methods.

- Fault Tolerant Testing

Verifies that individual software unit does not perform in the expected manner without illegal or out-of-range input parameters. At the application level, testing verifies that the entire application functions together in a graceful manner according to the requirements when presented with unexpected and/or out-of-range values.

Decision was made and followed to conduct fault tolerant testing after each major change is done. For instance: Passing null values, wrong value type, and correct type but exceeding size etc.

- Regression testing

Regression testing is retesting sub-systems/modules/units to insure that modifications to one sub-system/module/unit does not cause unexpected results in another sub-system/module/unit. This is also known as “Ripple effect testing”.

During a development of this project, decision was made to conduct regression tests once a month. This was important, because many times modifications in one part of the code cause unexpected problems in a "totally unrelated" area of the code.

3.11.1 Application design testing

Application design was tested to uncover bugs and UI element displacement. It was important, to find out how application design behaves on different screen sizes. Due to the huge amount of portable and desk screen sizes, it was impossible to obtain them all. Thus, the different approach was taken, using external applications and browser plug-ins.

For example:

Web Developer: Browser extension with various tools to test web application behaviour.

Mozilla Firebug: Browser extension for testing HTML, CSS, JavaScript code. As well as, to analyse network usage and performance.

Screenfly: Web application for screen size testing.

3.11.2 Production environment testing

During the development stage, application was tested in the localhost environment. Results of these tests are not always 100% accurate, due to the uniformity of the environment. Problems could occur when application is deployed on the hosted server. Difference in server configurations, used by hosting providers, could possibly crash individual parts of the system. As well, exists possibility that application environment is incompatible and it will not run at all.

4 PROJECT PLANNING AND FINAL COST

Planning is an important step in software development. A plan is necessary to ensure a steady workflow and organization. On this project, The Scrum cost estimation methodology was used, where points were assigned to each story. Knowing the story costs in combination with software requirements, planning was easy to do. The priority approach selected was “top – up”, or from more to less important. Although the initial time estimation was not clear, (in hours) the story points were considered a 1-1 relation, or in other words one point is equivalent to one day.

4.1 INITIAL TIME ESTIMATION

The following table represents all the different stories planned for development. The initial estimated time to get a final result was approximately 71 working days.

Table 28 - Initial plan estimation

Story	Task	Effort in days
Environment setup		1
	Clone Open-Eshop	
	Database Clean-up	
	Domain structure Clean-up	
Database refactor		2
	Bookings table	
	User refactor	
	Product refactor	
Domain model refactor		2
	Create MVC bookings structure	
	MVC product refactor	
	MVC categories	
Locations integration		2
	Create table structure	
	MVC integration structure	
	Visual representation	
	Widget location	

	Functionalities	
	Sample locations	
Bookings		5
	Bookings structure improvement	
	REST API	
	Adding functionalities	
	Visual improvements	
Refactor product creation		2
	REST API	
	Refactor interaction with bookings	
	Refactor model	
	Refactor controller	
	Visual improvements	
Improve translation system		1
	Add Spanish language	
	Fixing bugs	
Emails		1
	Booking confirmation (en/es)	
	Booking last check (en/es)	
	Booking cancel (en/es)	
Payment system		4
	General logic and integration	
	REST API	
	Bug fixing	
	Testing	
PayPal integration		2
	General logic and integration	
	Testing	
Paymil integration		5
	Request user account	
	General logic and integration	
	Testing	
Bitcoin integration		5

	Request user account	
	General logic and integration	
	Testing	
Different user rights		3
	Controller refactor	
	Modifier role	
	Views refactor	
	Configs refactor	
	Installation system, adding configs	
	Update system adding configs	
External usage of the bookings		4
	JQuery script	
	REST API	
	Visual representation	
Automatic module installation system		5
	New table Modules	
	Vendor controller refactor	
	New MVC structure	
	REST API	
	Functionality Improvements	
	Visual representation	
	New sample module	
	Testing	
UnitTests		20
	Learning PHP UnitTest framework	
	Writing Controller tests	
	Writing Model tests	
	Writing View Tests	
Testing Stage		7
	Running PHP UnitTests	
	Rewriting code to comply with UnitTests	
	Bug fixing	
TOTAL		71

4.2 FINAL TIME ESTIMATION

During the project development, it has been decided that certain stories are not to be developed. The reason is, some stories took more time to develop. And some requirements were changed during the development stage.

Refactoring code more frequent than anticipated had made a list of stories shorter for the similar amount of time. Following is detailed planning represented with Gantt chart.

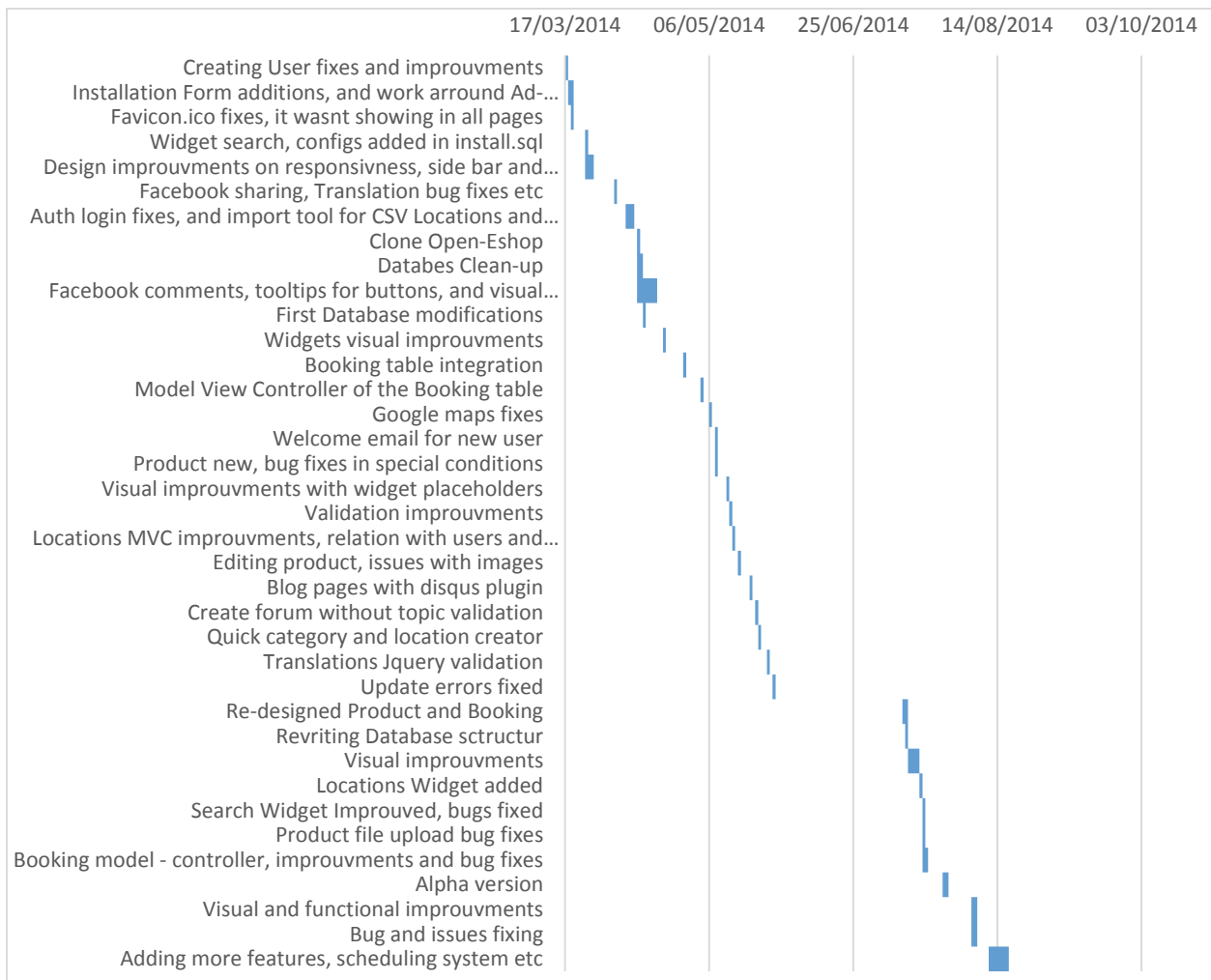


Figure 33 - Final time planning execution

4.3 COST ESTIMATION

In this section, we discuss the economic means for this project. The actual cost is approximate, and scales with the time invested in development.

- Cost of Hardware

The complete project was developed using my home environment. An ASUS x501a laptop. Sufficient for running all my requirements, a second Screen Acer x243w 27", a stable Wi-Fi connection. And, OS Windows 8. Similarly, hosting plan used to test in production was free. This includes: free subdomain name, up to 10 database installations, and server with 100 Mb disk space.

- Cost of Software and tools

Due to the nature of this project, all resources regarding software environment were Open Source. And no money was spent purchasing.

- Human resources

We can apply developer cost multiplying days 63 * average of 6 hours. Six hours is derived from mostly working after job hours approximately of 4-5 during the week, and from 7-8 hours weekends. With "salary", average for intern working half time (as well my last rate per hour) of 8 EUR.

- Extra expenses

No extra expenses were taken into account, such as electricity or home supplies consumption. Or any repro material. This spending's are not significant and hard to calculate.

Table 29 - Final cost estimation

Source	Cost per hour (EUR)	Hours invested	Total (with taxes)
Human resource	8	378	3024

5 FUTURE WORK & CONCLUSIONS

5.1 FUTURE WORK

As previously mentioned, in the planning section, the project was not developed entirely from the beginning. We can conclude that the main stories are mostly done. And it was imperative to have the structure to support the modular extension of the system. The stories finished up until now can serve as a starting point for the easy management of information and further improvements on both the Backend and the Frontend. Some future enhancements could include the following:

- Automated tests with the Php UnitTest framework. This is considered to be a major development. This system improvement was scheduled in the initial planning, but the scale and size decision has been made to move implementation for the next iteration.
- Installation system for modules and new functionalities. One of the most important feature of this application. Importance comes from the desire to facilitate the extensions and take advantage of software branching.

The plan is to implement an installation routine, which can then install the modules with different user case scenarios.

Note: this feature is not the same as the “Installation System”, that is currently up and running.

- Platform could be used as a Web Service. We could offer to users more possibilities, and add more flexibility overall.
- We could integrate different Payment methods. At the moment, PayPal is integrated but disabled since it needs further improvements. Apart from PayPal, other payment gateways working with credit cards and Bitcoins are needed and can be applied.
- Admin panel re-design. This functionality is not crucial, but the one we currently have is plane and basic.
- Push the first version in production: The making of the first demo of the version can be a great example of what the software can do.
- Design several templates for a front end application design, to extend the premium offer.
- Start advertising “Kuul-dev Bookings”. The plan, at the beginning, is to use social networks and the “word-of-mouth” method.

5.2 SUMMARY

The project was conceived with the motivation of getting a product that aids small companies and individuals to start new or improve existing businesses, by allowing them to do booking for their own products.

The second motive of this project was to develop a methodology for reusing parts or the entire existing project in order to alter the behaviour to serve other purposes. We passed through software development processes, modifying only specific sections.

5.3 CONCLUSION

In conclusion, we may say that the overall result of the final project is satisfactory. Although, development was not entirely finished according to the initial planning. Basic structure and architecture were implemented successfully.

The current state of the application fulfils the initial requirements and can be considered as the first “Alpha” version. Alongside, the main use case “Booking”, many other services are implemented or improved.

Further development should be conducted to remove remaining bugs, improve existing features, and continue implementation of all scheduled stories from the initial planning.

My role in this project was substantial, since I created it. The whole involvement was very educational and helpful, as well as huge experience in problem solving software analysis and planning. I learned how to organize myself and how to push more to reach certain goals and deadlines. This is a good start, and place to continue with further development and implementation in the future.

Clearly, some of development solutions offered are not entirely unique. Our study serves as a window to an understanding of the processes for reusing existing Open Source project. To get the desired result and save time, while following software development standards and processes such as initial analysis, gathering and setting requirements, estimation and planning, implementation and testing etc.

.

BIBLIOGRAPHY

Advanced PHP Programming [Book] / auth. Schlossnagle George. - [s.l.] : SAMS Publishing, 2004.

Apache Administrator's Handbook [Book Section] / auth. Bowen Rich, Ridruejo Daniel López and Liska Allan // Apache Administrator's Handbook. - [s.l.] : Sams, 2002.

apache.org [Online] / auth. 2.0 Apache. - 2012. - <http://www.apache.org/>.

Benefits of MySQL [Online] / auth. Novell. - Novell, 2014. - http://www.novell.com/documentation/nw65/web_mysql_nw/data/aj5bj52.html.

kohanaframework.org [Online] / auth. framework Kohana. - 2013. - <http://kohanaframework.org/>.

open-classifieds.com [Online] / auth. Open-classified. - 2014. - <http://open-classifieds.com/>.

open-eshop.com [Online] / auth. Open-eshop. - 2014. - <http://open-eshop.com/blog>.

php.net [Online] / auth. php.net. - 2014. - <http://www.php.net/>.

phpunit.de [Online] / auth. PHPUnit. - 2014. - <http://phpunit.de/manual/current/en/index.html>.

W3Counter [Online] / auth. consorcium W3. - W3 consorcium, 2014. - <http://www.w3counter.com/globalstats.php>.

w3schools.com [Online] / auth. consorcium W3 // general review of Web technologies. - W3 consorcium, 2014. - <http://www.w3schools.com/>.

wikipedia.org [Online] / auth. Wikipedia. - 2014. - <http://www.wikipedia.org/>.

GLOSSARY

PHP

PHP is a server-side scripting language designed for web development but also used as a general-purpose programming language.

APACHE

The Apache HTTP Server, colloquially called Apache, is a Web server application notable for playing a key role in the initial growth of the World Wide Web.

HTML

Hypertext Mark-up Language (HTML) is the standard mark-up language used to create Web pages.

CSS

Cascading Style Sheets is a style sheet language used for describing the look and formatting of a document written in a mark-up language.

JAVASCRIPT

JavaScript is a dynamic computer programming language. It is most commonly used as part of web browsers, whose implementations allow client-side scripts to interact with the user, control the browser, communicate asynchronously, and alter the document content that is displayed. It is also being used in server-side network programming (with Node.js), game development and the creation of desktop and mobile applications.

JQUERY

JQuery is a cross-platform JavaScript library designed to simplify the client-side scripting of HTML. It is used by over 60% of the 10,000 most visited websites, jQuery is the most popular JavaScript library in use today.

API

Application programming interface specifies a software component in terms of its operations, their inputs and outputs and underlining types.

RESTFUL

Representational state transfer is an abstraction of the architecture of the www i.e, World Wide Web.

ORM

Object-relational mapping in computer science is a programming technique for converting data between incompatible type systems in object-oriented programming languages.

MVC

Model–view–controller is a software architectural pattern for implementing user interfaces. It divides a given software application into three interconnected parts, so as to separate internal representations of information from the ways that information is presented to or accepted from the user.

OC

Open Classifieds is an Open Source (GPL v3) project that lets us easily create our own fully customizable classifieds site. OC can be used to create car/auto sales, job search board, buying & selling real estate and almost anything we can think of. Thousands of web developers trust Open Classifieds to run their big classifieds websites.

DBMS

A database is an organized collection of data. The data are typically organized to model aspects of reality in a way that supports processes requiring information. For example, modelling the availability of rooms in hotels in a way that supports finding a hotel with vacancies.

PK

The primary key of a relational table uniquely identifies each record in the table. It can either be a normal attribute that is guaranteed to be unique (such as Social Security Number in a table with no more than one record per person) or it can be generated by the DBMS.

IK

An index can be created in a table to find data more quickly and efficiently. The users cannot see the indexes, they are just used to speed up searches/queries.

WEB HOSTING

A web hosting service is a type of Internet hosting service that allows individuals and organizations to make their website accessible via the World Wide Web. Web hosts are companies that provide space on a server owned or leased for use by clients, as well as providing Internet connectivity, typically in a data centre.

LOCALHOST

In computer networking, localhost means this computer. It is a hostname that the computer's software and users may employ to access the computer's own network services via its loopback network interface. Using the loopback interface bypasses local network interface hardware.

WINDOWS

Microsoft Windows is a series of graphical interface operating systems developed, marketed, and sold by Microsoft.

OSS

Open-source software is computer software with its source code made available with a license in which the copyright holder provides the rights to study, change and distribute the software to anyone and for any purpose.

UI

The user interface, in the industrial design field of human–machine interaction, is the space where interactions between humans and machines occur. The goal of this interaction is effective operation and control of the machine on the user's end, and feedback from the machine, which aids the operator in making operational decisions.

ZIP

.ZIP is an archive file format that supports lossless data compression. A .ZIP file may contain one or more files or folders that may have been compressed. The .ZIP file format permits a number of compression algorithms.