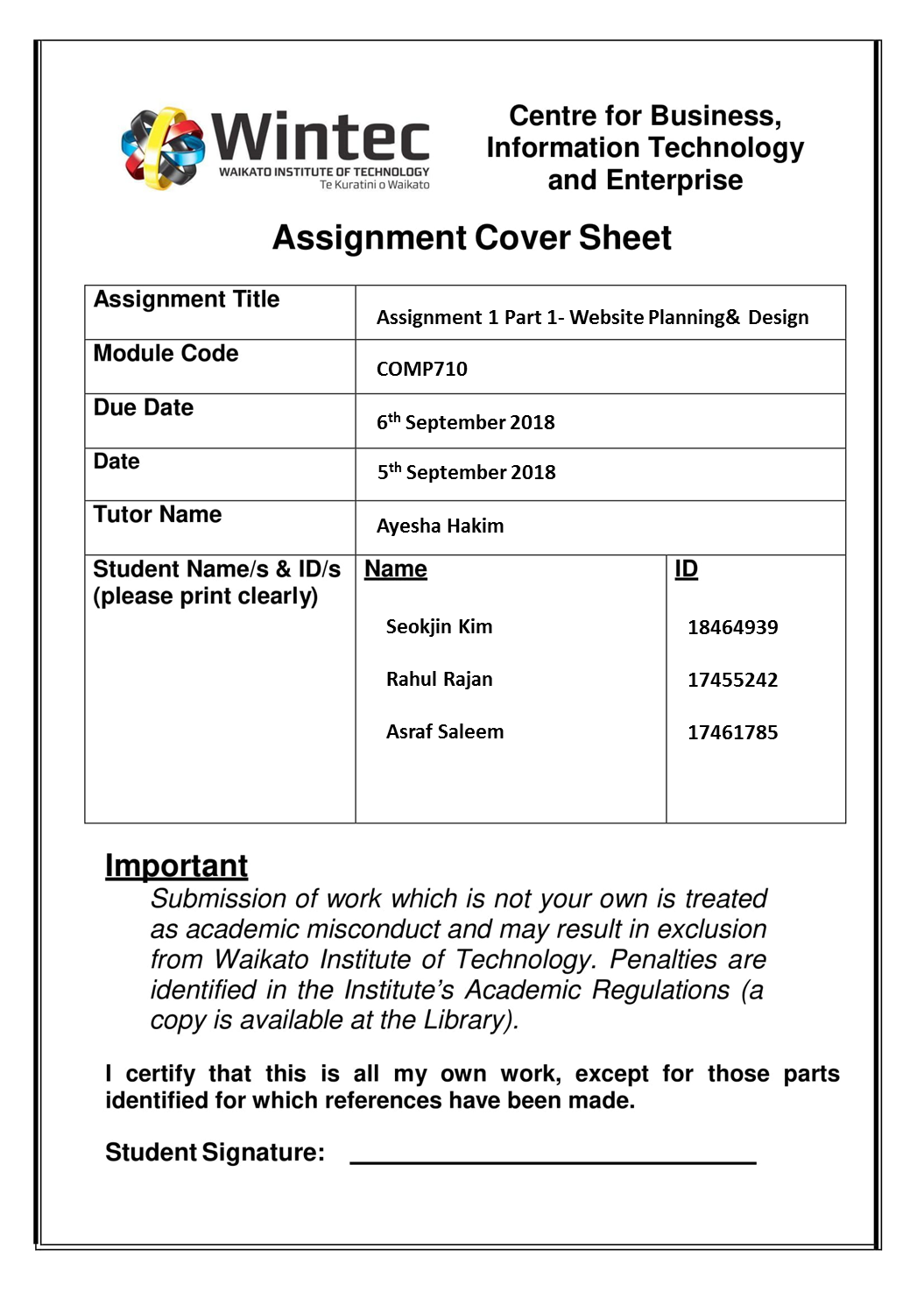
****

**1. GitHub repository:** [**https://github.com/seokjinkim91/bosscuts**](https://github.com/seokjinkim91/bosscuts)

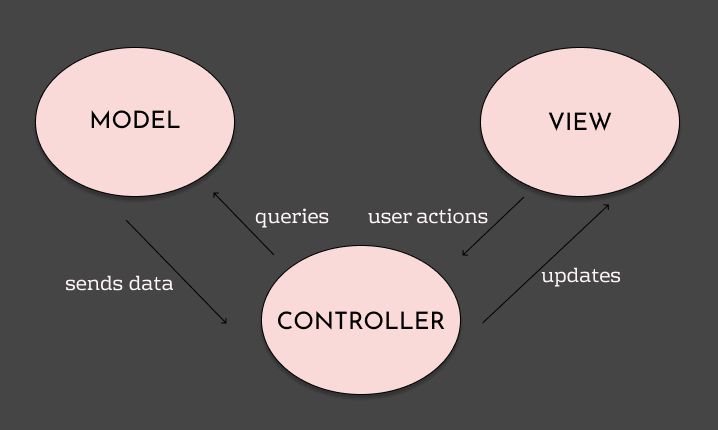
**2. Application Description**

**BossCuts** website is primarily a booking management system for the customers which would display services and their price details just like any other barbershop website. Clients could view pictures of different haircuts each stylist offer and even book an appointment with them. Features range from membership discounts for the regular customers to handling job applications of hair stylist aspirants.

* 1. **Application architecture**

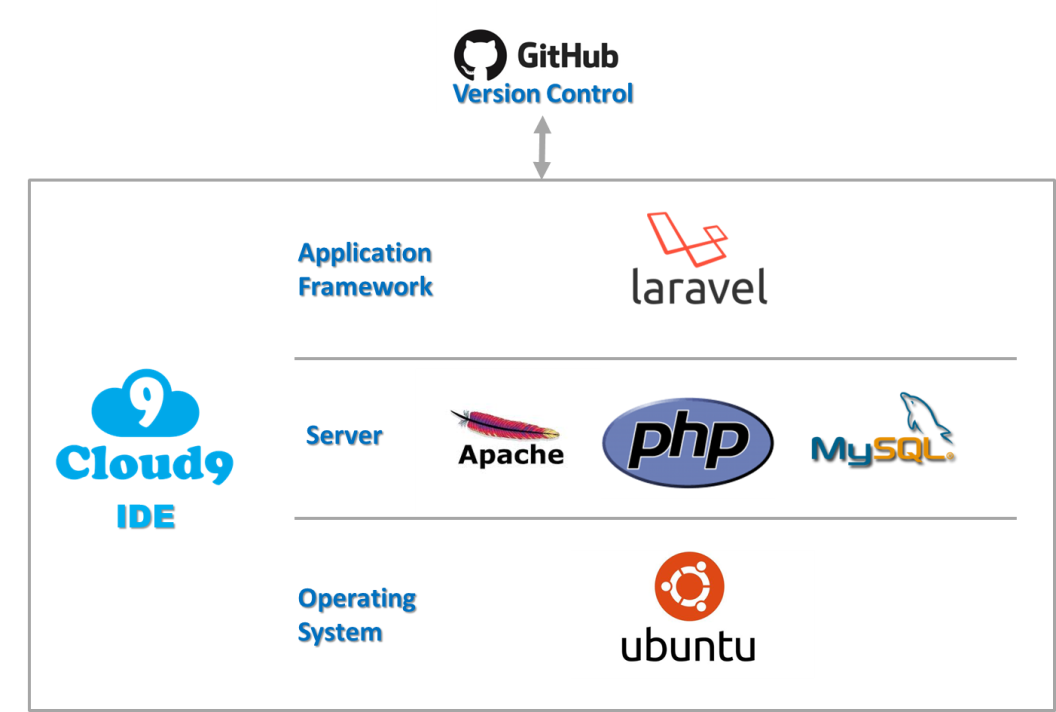
The web-application would be developed using Laravel framework and hence we would be following a Model View Controller architectural pattern in the project. This potentially segregates the business logic from the User Interface and much cleaner/maintainable code could be produced. Data of the application would be managed by the model part too.

Benefits could include easy to unit test methods written in controller, chances of spaghetti code would be less as there are reusable components(methods) in the controller.



***Figure 2.1 Diagram of MVC (Mode- View-Controller) relationship***

The application architecture which has been installed in the development environment can be illustrated as below diagram.



***Figure 2.2 Diagram of application architecture in the project***

**2.2 Application framework**

Laravel is a PHP-based web framework that is largely based on the MVC architecture and features would be constructed as a modular packaging system with a dedicated dependency manager, different ways for accessing relational databases, utilities that aid in application deployment and maintenance, and its orientation toward syntactic sugar. Loris Leiva (2017) illustrates the architecture of Laravel that consists of four simple layer and main functions and related folders of Laravel were described as below

|  |  |
| --- | --- |
| Layered Architecture overview | Layered architecture matched with Laravel folders |

*Figure 2.3 Main architecture of Laravel Framework <resource: Loris Leiva (2017)>*

**3. Application Requirements**

The below requirements are elicited following a semi-structured interview and by observing other websites in the same industry.

* 1. **Client interview**

Client needs a website application from which customers could potentially look up for the services they offer and price details. Data Gathering techniques included, client interview and a Q&A session.

|  |  |  |
| --- | --- | --- |
| **Type** | **Category/Page** | **Description** |
| Functional | Admin Page | - Manage staff information  - Monitor the booking system for the owner and staffs |
| Job applications | - Allow for storage / ease of job applications |
| Online Booking | -Owner and his staff to monitor the booking system |
| Gallery | - Have a gallery that has photos of recent haircuts and popular styles |
| Non Functional | Web Design | - Need to be broken down into several sections to advertise the many different aspects of the salon, including; the cuts, styles, products,  - List of prices and the level of service offered  - Special offers for new clients, seniors, kids, or regulars  - Must have hours, location, and contact info which is clear and easy to find |
| Digital Marketing | - Have social media links used in the website  - Would like to have an advertisement video of the shop  - Would like to have a new business card for the business and Boss Cuts team.  - Maximize the number of visitors by typical search engine. |

*Figure 3.1 List of categorized requirements for client interview*

With an hour of semi-structured interview, non-functional requirements which relate to marketing and user interface efficiency had been gathered more than functional requirements. Collected requirements could be summarized as below.

**3.2 Observation of reference sites**

To implement detailed features, observation of reference sites is one of the most effective techniques for analyzing user requirements. Three similar web sites were shortlisted for detailed examination.

|  |  |  |  |
| --- | --- | --- | --- |
| **Site Name** | **Parnell barbershop** | **Fresh barbershop** | **Brotherhood** |
| **Main Feature** |  |  |  |
| **Brief description** | Barbershop located in Auckland, flash design-based web page. | Barbershop located in Wellington, traditional style home page. | Barbershop located in Auckland, design based but traditional style |
| **Menu** | Home > Book > Online Shop > Gift Voucher> Contact > Job | About > Barbers > Book > Shop > Bench Cam | Home > About > Gallery > Services > Hours > Book > Contact |
| **Online Booking** | Online payment needed for booking | Login should be needed for any kinds of booking | Third party solution (timely)  No pay no login needed |
| **Gallery** | Instagram API has been used. | Traditional style of gallery | Flash based own pictures displayed at center of pages |
| **Job application** | Simple job application page has been existed with file uploading. | N/A | N/A |
| **Web Design** | Complicated | Traditional | Simple |
| **Digital Marketing** | Instagram based Gallery | Facebook and Instagram links on main page | No SNS links and integration |

*Figure 3.2 List of comparison with three reference sites*

Regarding the idea of gallery, using Instagram API would be the most suitable option in terms of digital marketing thus we have decided to use Instagram API for this function and job application will be referenced from the website of Parnell barbershop.

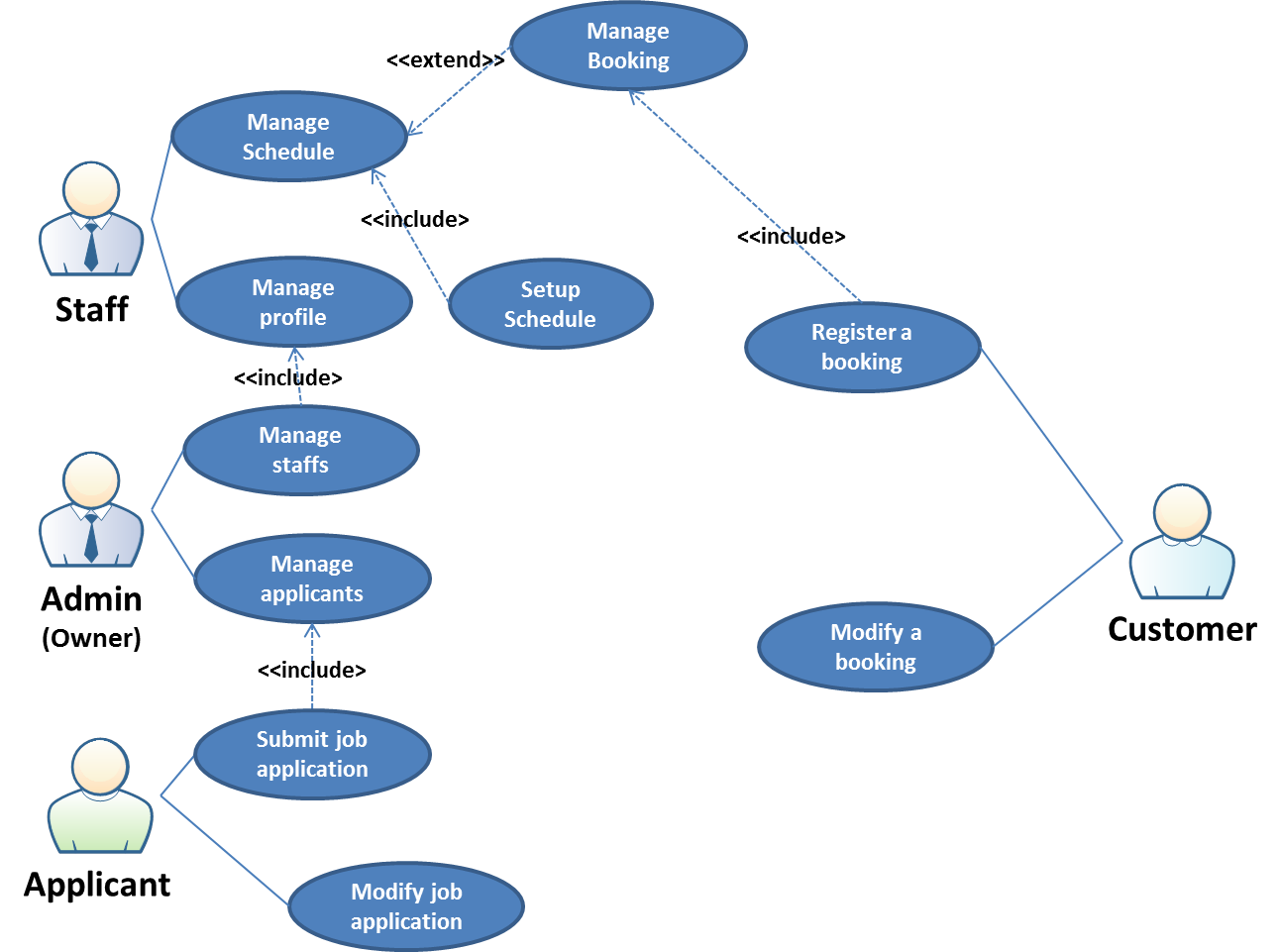
|  |  |
| --- | --- |
| **Site Name** | **URL** |
| **Parnell barbershop** | <https://www.parnellbarbershop.co.nz/> |
| **Fresh barbershop** | <https://freshbarbershop.co.nz/> |
| **Brotherhood** | <https://www.brotherhood.net.nz/> |

*Figure 3.3 List of URL in reference web sites*

**3.3 User cases diagram**

Ideally the end-users could be categorized into three; Admin/Staffs, Customers and Job-applicants.

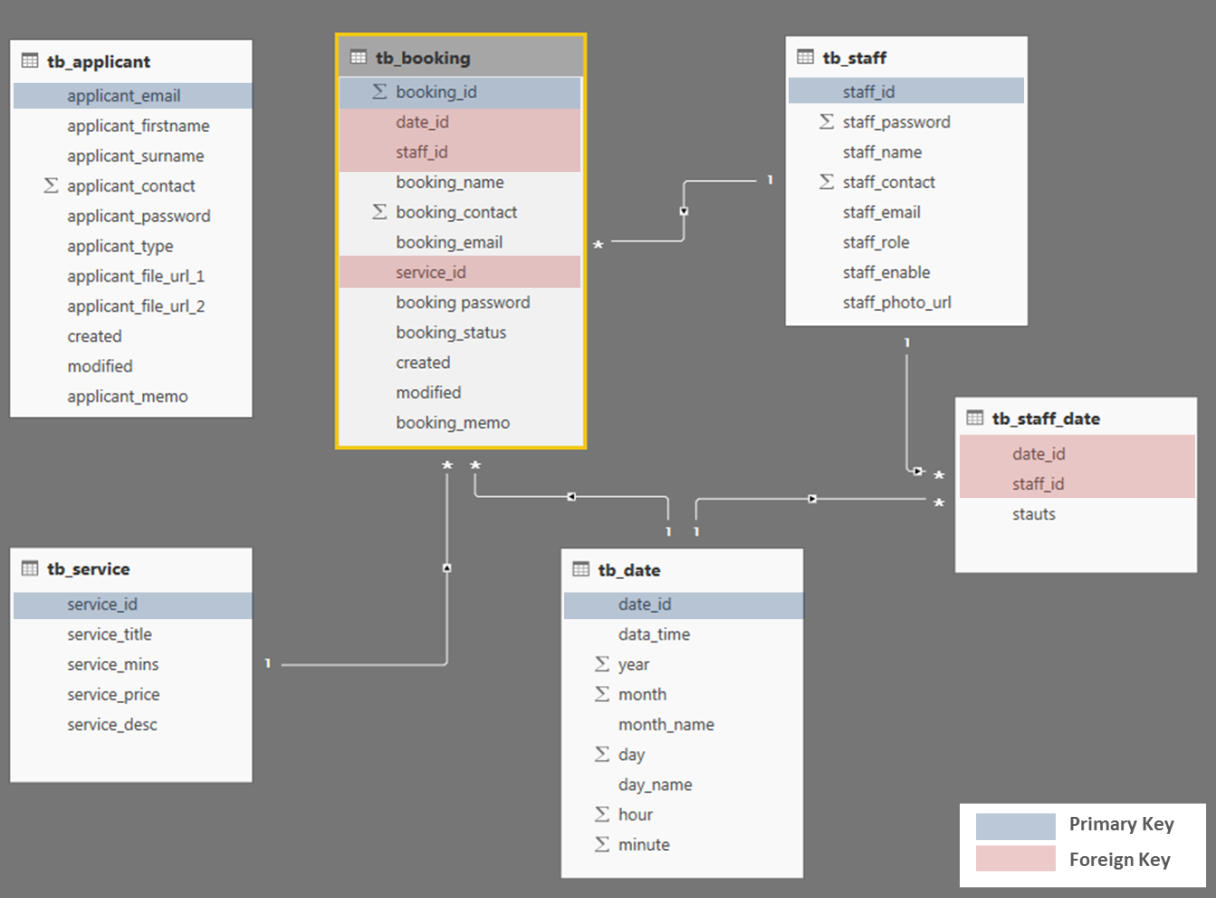
The following diagram would give a clear picture of the user roles.



*Figure 3.4 Use case diagram for barbershop web application*

**4. Entity Relationship Diagram**

Considering the user cases diagram, Entity Relationship Diagram can be illustrated as below.



*Figure 4.1 Entity Relationship Diagram for the project*

**5. Data Dictionary**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Entity** | **Field Name** | **Data Type** | **Length** | **Constrain** | **Description** |
| tb\_staff | staff\_id | varchar | 20 | Primary Key | Identification of Staff |
| staff\_password | varchar | 30 | Not null | Login password |
| staff\_name | varchar | 20 | Not null | Name of Staff |
| staff\_contact | varchar | 20 | Not null | Phone number of Staff |
| staff\_email | varchar | 30 | Not null | Email of Staff |
| staff\_role | varchar | 5 | Not null | Role identifier (admin/staff) |
| staff\_enable | char | 1 | Not null | Enable of account (Y/N) |
| staff\_photo\_url | text |  | Not null | URL of Staff photo |
| tb\_datetime | datetime\_id | int | 12 | Primary Key | Timestamp ID (YYYYMMDDhhmm) |
| data\_time | datetime |  | Not null | datetime Format conversion of ID |
| year | int | 4 | Not null | Year (YYYY) |
| month | int | 2 | Not null | Month (MM) |
| month\_name | varchar | 10 | Not null | Full name of Month |
| day | int | 2 | Not null | day (DD) |
| day\_name | varchar | 10 | Not null | Full name of Day |
| hour | int | 2 | Not null | Hour (hh) |
| minute | int | 2 | Not null | Min (mm) |
| tb\_booking | booking\_id | int | 10 | Primary Key | Sequence |
| datetime\_id | int | 12 | Foreign Key | tb\_date Primary Key |
| staff\_id | varchar | 20 | Foreign Key | tb\_staff Primary Key |
| booking\_name | varchar | 20 | Not null | Name of Customer |
| booking\_contact | varchar | 20 | Not null | Phone Number of Customer |
| booking\_email | varchar | 30 | Not null | Email of Customer |
| service\_id | int | 10 | Foreign Key | tb\_service Primary Key |
| booking password | varchar | 30 | Not null | Password for modification |
| booking\_status | varchar | 10 | Not null | Status of Booking (Accept/Deny/Pending/Cancel) |
| created | datetime |  |  | time of creation |
| modified | datetime |  |  | time of last modification |
| booking\_memo | longtext |  |  | booking text memo |
| tb\_service | service\_id | int | 10 | Primary Key | Sequence |
| service\_title | varchar | 50 | Not null | service title for display |
| service\_mins | int | 3 | Not null | Minutes of service duration |
| service\_price | int | 3 | Not null | Price of service |
| service\_desc | longtext |  |  | Service description |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Entity** | **Field Name** | **Data Type** | **Length** | **Constrain** | **Description** |
| tb\_applicant | applicant\_email | varchar | 30 | Primary Key | Email of applicant |
| applicant\_firstname | varchar | 20 | Not null | Name of applicant |
| applicant\_surname | varchar | 20 | Not null | Name of applicant |
| applicant\_contact | varchar | 20 | Not null | Phone number of applicant |
| applicant\_password | varchar | 30 | Not null | Password of applicant |
| applicant\_type | varchar | 10 | Not null | Application type (full/part/manager) |
| applicant\_file\_url\_1 | text |  |  | URL of CV 1 |
| applicant\_file\_url\_2 | text |  |  | URL of CV 2 |
| created | datetime |  |  | time of creation |
| modified | datetime |  |  | time of last modification |
| applicant\_memo | longtext |  |  | booking text memo |
| tb\_staff\_date | datetime\_id | int | 10 | Foreign Key | tb\_date Primary Key |
| staff\_id | varchar | 20 | Foreign Key | tb\_staff Primary Key |
| stauts | varchar | 10 | Not null | Status of staff time (vacancy, book, wait, cancel) |

**References**

1. Amazon Web Services. (2017). AWS Cloud9. Retrieved from https://aws.amazon.com/cloud9/?nc1=h\_ls
2. Conallen, J. (1999). Modeling Web Application Architec tures with UML By. *Rational Software White Paper*, *9*, 1–9.
3. Koch, N., & Escalona, M. J. (2004). Requirements engineering for web applications – a comparative study. *Journal of Web Engineering*, *2*(3), 193–212. doi:10.1.1.153.5974
4. Lorisleiva. (2017). Conciliating Laravel and DDD. Retrieved from http://lorisleiva.com/conciliating-laravel-and-ddd/
5. Selfa, D. M., Carrillo, M., & Del Rocio Boone, M. (2006). A database and web application based on MVC architecture. *Proceedings of the 16th IEEE International Conference on Electronics, Communications and Computers, CONIELECOMP 2006*, *2006*(November), 48. doi: 10.1109/CONIELECOMP.2006.6
6. Tehctarget. (n.d.). What is use case diagram (UML use case diagram)?. Retrieved from https://whatis.techtarget.com/definition/use-case-diagram