PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA
Ministry of Higher Education and Scientific Research

UNIVERSITY OF ECHAHID HAMMA LAKHDAR EL OUED FACULTY OF EXACT SCIENCES
Department of Computer Science

Final year thesis

ACADEMIC BACHELOR

Domain : **Mathematics and Computer science**
Field : **Computer science**
Specialty : **Information System**

Topic

Designing and Building Web Framework

- Case study: Master registration website

Presented by:

Chikha Youcef

Djaballah Souhaib

Kahla Mohammed Elhabib

Proposed by: Mr. *Othmani* Samir

20-05- 2018
Committee:

M. *Bali* Mouad — President

M. *Yagoub* Mohamed El-Amin — Rapporteur

Academic year: 2017-2018

Abstract

There are many Web frameworks for developers to choose. The developers should have the ability to decide whether to use an external framework such as Laravel and Django or build their own framework that suits their project. Choosing the most suitable framework to use in a project requires a good understanding of how a framework actually works under the hood. This study aimed to demonstrate and explain the concept of Web frameworks by building a complete web framework using PHP with the latest web technologies.

ملخص

هناك العديد من منصات تطوير تطبيقات الويب للاختيار بالنسبة للمطورين. يجب أن يكون لدى المطورون القدرة على اتخاذ القرار بشأن استخدام منصة تطوير خارجية مثل Laravel و Django أو بناء إطار عمل خاص بهم حسب ما يناسب مشروعهم. يتطلب اختيار أنسب منصة تطوير لمشروع ما فهم جيد لكيفية عمل منصات التطوير في الخلفية. تهدف هذه الدراسة إلى توضيح وشرح مفهوم منصات تطوير تطبيقات الويب من خلال بناء منصة تطوير كاملة باستخدام لغة الـ PHP مع أحدث تقنيات الويب.

Résumé

Il existe de nombreux Framework de Web que les développeurs peuvent choisir. Les développeurs devraient avoir la possibilité de décider d'utiliser un Framework externe tel que Laravel ou Django ou de construire leur propre framework qui convient à leurs projet. Choisir le Framework le plus approprié à utiliser dans un projet nécessite une bonne compréhension de la façon dont un Framework fonctionne réellement en arrière-plan. Cette étude visait à démontrer et expliquer le concept des frameworks Web en construisant un framework web complet en utilisant PHP avec les dernières technologies web.

Acknowledgments

We would like to express our special thanks to our supervisor ‘OTHMANI Samir’ who gave us the great opportunity to do this wonderful project, which also helped us in doing a lot of research and we came to know about so many new things.

Furthermore, we would also like to thank all the examiners who spent a lot of time to correct and review our work.

Finally, we take this opportunity to give our special gratitude to our parents, brothers, sisters, and sincere friends for the unceasing encouragement, support and attention.

El-Oued, Algeria.

May 2018

Table of Contents

List of Figures

Acronyms and Abbreviations

HTTP	Hypertext Transfer Protocol
OOP	Object Oriented Programming
SQL	Structured Query Language
ORM	Object-relational mapping
DB	Database
JS	JavaScript
MVC	Model-View-Controller
RDBMS	Relational database management system
XML	Extensible Markup Language
HTML	Hypertext Markup Language
CSS	Cascading Style Sheets
API	Application programming interface
URL	Uniform Resource Locator

1 Introduction

1.1 Overview

Framework facilitates Web programming and makes it better organized in many ways. First of all, frameworks increase programming productivity because writing a piece of code that usually takes hours and takes up hundreds of lines of code can be done in minutes with the help of framework built-in functions. Framework selection is an important step because the speed and quality of the work depends on it. Main aspects to take into consideration are usage context, license, software pattern, hosting requirements, ease of installation, core library, learning curve, DB abstraction and ORM, included JS libraries etc.

For a long time, PHP programming language was not considered as a sufficiently serious language for large Web application development. It was known that it is popular and perhaps good for small projects but most of the appreciation was reserved for the aristocratic framework elite like Spring, Ruby on Rails or Django. Only recently the situation has significantly changed. The development pace was fast and stable. Object-oriented source code that was written in PHP was elegant and maintainable. More and more new projects began to use PHP frameworks and successful completion of these projects made PHP frameworks even more popular. Popular programming languages for website development are PHP, Ruby, Python and Java. Selecting from these languages PHP was chosen for framework analysis and website development, because it is an open source scripting language with many advantages: support for different database types, many language structures were taken from C and Perl languages. Since PHP is a popular and widely used language, it has a lot of documentation on the Web. Also, as shown in Fig. 1 PHP is used by 83.4% of all websites whose server-side programming languages are known.

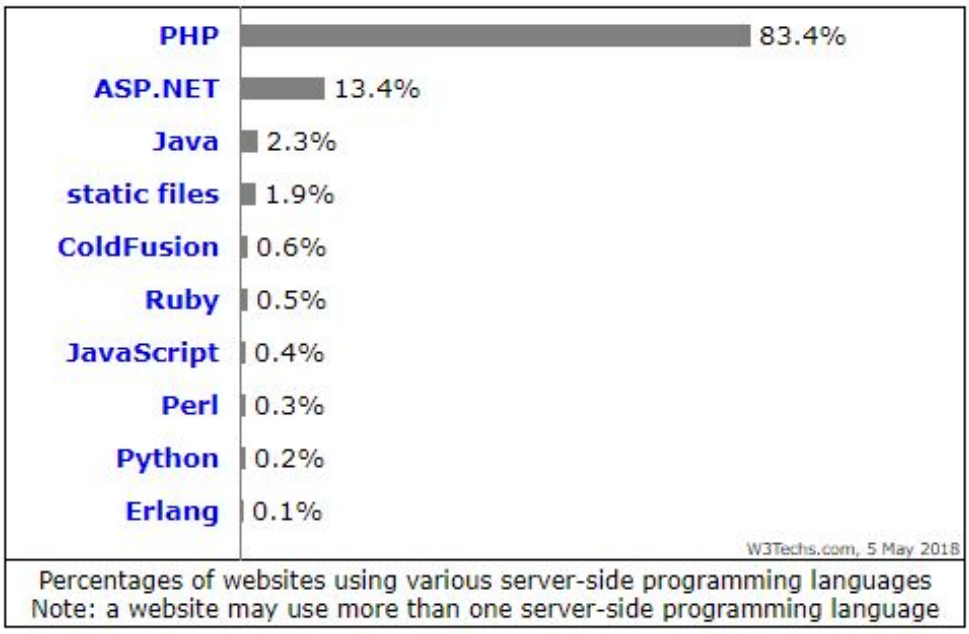

Fig. 1 Usage of server-side programming languages for websites from W3Techs.com.

1.2 Outline

All of the information mentioned above confirms that framework usage for website development is quite a hot topic. The next chapter will study some of the most popular PHP frameworks with a description of the architecture and main features of selected frameworks (routing, template engine, etc.) are offered. A brief overview of Model-view-controller (MVC) architecture style.

2 Literature Review

Most PHP frameworks are full-stack and offer the functionalities for creating web application, from front-end code writing to back-end data retrieval. Five general criteria that developers use to select a framework. They are architecture, documentation, community support, flexibility and whether they possess a list of features. While these criteria are essential, they are not task specific and lack of implementation details. They don't provide deeper help on making decision for a concrete web application. In, a method using source lines of code as a comparative metric to evaluate PHP MVC frameworks and determine the differences in effort required to implement the fundamental components of a web application was developed.

PHP frameworks can considerably improve the performance of an application. These frameworks usually are based on Model, View, and Controller design pattern. These frameworks provide, different common functionalities and classes in the form of helpers, components, and plug-in to reduce the development time. Due to these features such as robustness, scalability, maintainability and performance, these frameworks are mostly used for web development in PHP, with performance being the most important factor.

Database connection is very important for a PHP framework. When multiple parts of your application need to interact with the database the related code shouldn't need to be duplicated in each and every PHP script. The prudent thing to do would be to refactor all database code into a shared PHP file. A developer should really need to worry about opening and closing database connections. The database-related tasks should be included in the PHP framework evaluation experiments.

In software engineering, software metrics are the only tools to control the quality of software. Most of the metrics are developed covering generally programming languages such as C, C++, and Java etc. Some metrics may not be suitable for some programming or scripting languages. There is lack of metric in the literature, which measures the quality of PHP language. In this respect, specific metrics should be developed, and developing a metric for PHP, which is capable to calculate reusability quality of PHP code. A reusability metric to measure quality of PHP script language (REUphp). Since PHP is an object-oriented language, the present metric is capable to evaluate any object-oriented. Web application frameworks usually implement the Model View Controller (MVC) design pattern. The MVC pattern is a proven and effective way for the generation of organized

modular applications. The MVC pattern breaks an application into three modules: model, view and controller. The model module contains the underlying classes whose instances are to be used for manipulating the database, templating frameworks and session management, and they often promote code reuse. [1]

3 Requirement Specifications

3.1 Existing System

The exiting system is using the traditional PHP approach which also called quick and dirty approach. All files are put together using lots of include files. The application logic is mixed up with presentation. It uses the static address mapping where each address must map directly to a script file. That leads to many problems:

- Unstructured script files
- Lots of repeated code
- The code is written in a random way
- Insecure system configuration data
- Cost a lot of effort and time
- Difficulty tracking errors

3.2 Proposed System

The proposed system is a modular lightweight PHP framework using The MVC architecture with new web technologies such as dependency manager, dynamic routing and templating engine. This system has many advantages:

- Rapid development
- Secure application
- Easier Maintenance
- Strong Teamwork
- Highly structured code

3.3 Requirements

There are some requirements the proposed framework need to run properly. Certain components have dependencies to other extensions. For instance, using database connectivity will require the `php_pdo` extension. If your RDBMS is MySql/MariaDb or Aurora databases you will need the `php_mysqlnd` extension also. Similarly, using a PostgreSql database requires the `php_pgsql` extension.

- PHP >= 7.1.2
- Composer (dependency manager for PHP)

4 Design

The Web Framework was designed to make it dramatically easier to create dynamic web applications. The web framework provides an application ready environment, templating engine, a powerful Model-View-Controller framework and a library of JavaScript view controls to take the tedium out of creating web applications. Also, it drastically reduces the number of lines of code you need to write for a compelling, dynamic web application.

4.1 Design Methodology

The MVC (Model-View-Controller) pattern, originally formulated in the late 1970s, is a software architecture pattern built on the basis of keeping the presentation of data separate from the methods that interact with the data. In theory, a well-developed MVC system should allow a front-end developer and a back-end developer to work on the same system without interfering, sharing, or editing files either party is working on [2]. A visual representation of a complete MVC pattern looks like the following diagram:

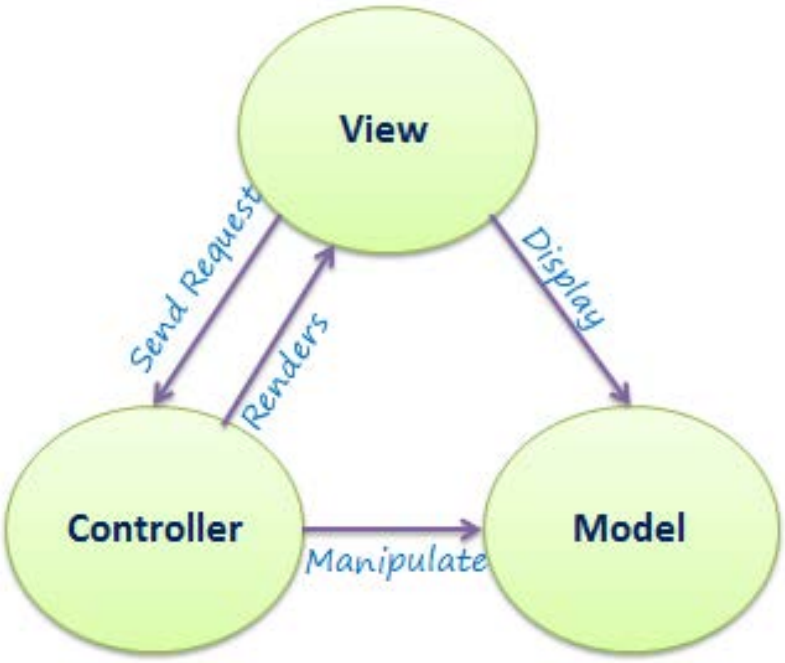

Fig. 2 MVC Workflow [3]

4.1.1 Model

The Model is the name given to the permanent storage of the data used in the overall design. It must allow access for the data to be viewed, or collected and written to, and is

the bridge between the View component and the Controller component in the overall pattern.

One important aspect of the Model is that it's technically "blind" – by this I mean the model has no connection or knowledge of what happens to the data when it is passed to the View or Controller components. It neither calls nor seeks a response from the other parts; its sole purpose is to process data into its permanent storage or seek and prepare data to be passed along to the other parts.

4.1.2 View

The View is where data, requested from the Model, is viewed and its final output is determined. Traditionally in web apps built using MVC, the View is the part of the system where the HTML is generated and displayed. The View also ignites reactions from the user, who then goes on to interact with the Controller. The basic example of this is a button generated by a View, which a user clicks and triggers an action in the Controller.

4.1.3 Controller

The final component of the triad is the Controller. Its job is to handle data that the user inputs or submits and update the Model accordingly. The Controller's life blood is the user; without user interactions, the Controller has no purpose. It is the only part of the pattern the user should be interacting with.

The Controller can be summed up simply as a collector of information, which then passes it on to the Model to be organized for storage and does not contain any logic other than that needed to collect the input. The Controller is also only connected to a single View and to a single Model, making it a one-way data flow system, with handshakes and signoffs at each point of data exchange.

4.2 Design Constraints

Although MVC, in theory, should work flawlessly in all forms of computer programming, incorporating MVC on the web with PHP can be a bit tricky. The problem is with URL routing. URL routing is an aspect that was never considered when MVC was created, and so as the Internet evolves and develops with the expansion of technology, so must the architecture patterns we use. [4]

4.3 Database Design

The database mechanism used is The ORM database or Object-relational mapping, which is a mechanism that makes it possible to address, access and manipulate objects without having to consider how those objects relate to their data sources. ORM lets programmers

maintain a consistent view of objects over time, even as the sources that deliver them, the sinks that receive them and the applications that access them change.

Based on abstraction, ORM manages the mapping details between a set of objects and underlying relational databases, XML repositories or other data sources and sinks, while simultaneously hiding the often-changing details of related interfaces from developers and the code they create.

ORM hides and encapsulates change in the data source itself, so that when data sources or their APIs change, only ORM needs to change to keep up—not the applications that use ORM to insulate themselves from this kind of effort. This capacity lets developers take advantage of new classes as they become available and also makes it easy to extend ORM-based applications. In many cases, ORM changes can incorporate new technology and capability without requiring changes to the code for related applications. [5]

4.4 Use Case Model

4.4.1 List of Use Cases (grouped by actors)

- Developer: someone who is using the framework for building a web application.
  - Create SubController
  - Create SubModel
  - Create Views
  - Use Mail Handler
  - Use Token Generator
  - Use HttpRequests Handler
  - Use Session Handler
  - Use Validation Methods
  - Use Flash Messages
- Creator: is the framework creator that maintain and update it regularly.
  - Maintain the framework
  - Release new version
  - Document the Framework

4.4.2 Use Case Description

- Create SubController
  - This use case is started by the developer. It provides the capability to create new SubControllers and use it to control the models and views.
- Create SubModel

    - This use case is started by the developer. It provides the capability to create new SubModels to store and handle specific data.
- Create Views
    - This use case is started by the developer. It provides the capability to create new Views to handle presentation part.
- Use Mail Handler
    - This use case is started by the developer. It provides the capability to generate and send Emails.
- Use Token Generator
    - This use case is started by the developer. It provides the capability to create a new random token and its hash or by using predefined token to generate its hash.
- Use HttpRequests Handler
    - This use case is started by the developer. It provides the capability to handle Http requests using an object-oriented encapsulation.
- Use Session Handler
    - This use case is started by the developer. It provides the capability to handle Session data using an object-oriented encapsulation.
- Use Validation Methods
    - This use case is started by the developer. It provides the capability to use predefined validation or methods or generate custom validations using regular-expressions.
- Use Flash Messages
    - This use case is started by the developer. It provides the capability to flash messages as global notification holder to use through the website.
- Maintain the framework
    - This use case is started by the framework creator. It provides the capability to update, debug and maintain the framework.
- Release new version
    - This use case is started by the framework creator. It provides the capability to release new framework version to the developers.
- Document the Framework
    - This use case is started by the framework creator. It provides the capability to document the framework.

4.4.3 Use Case Diagram

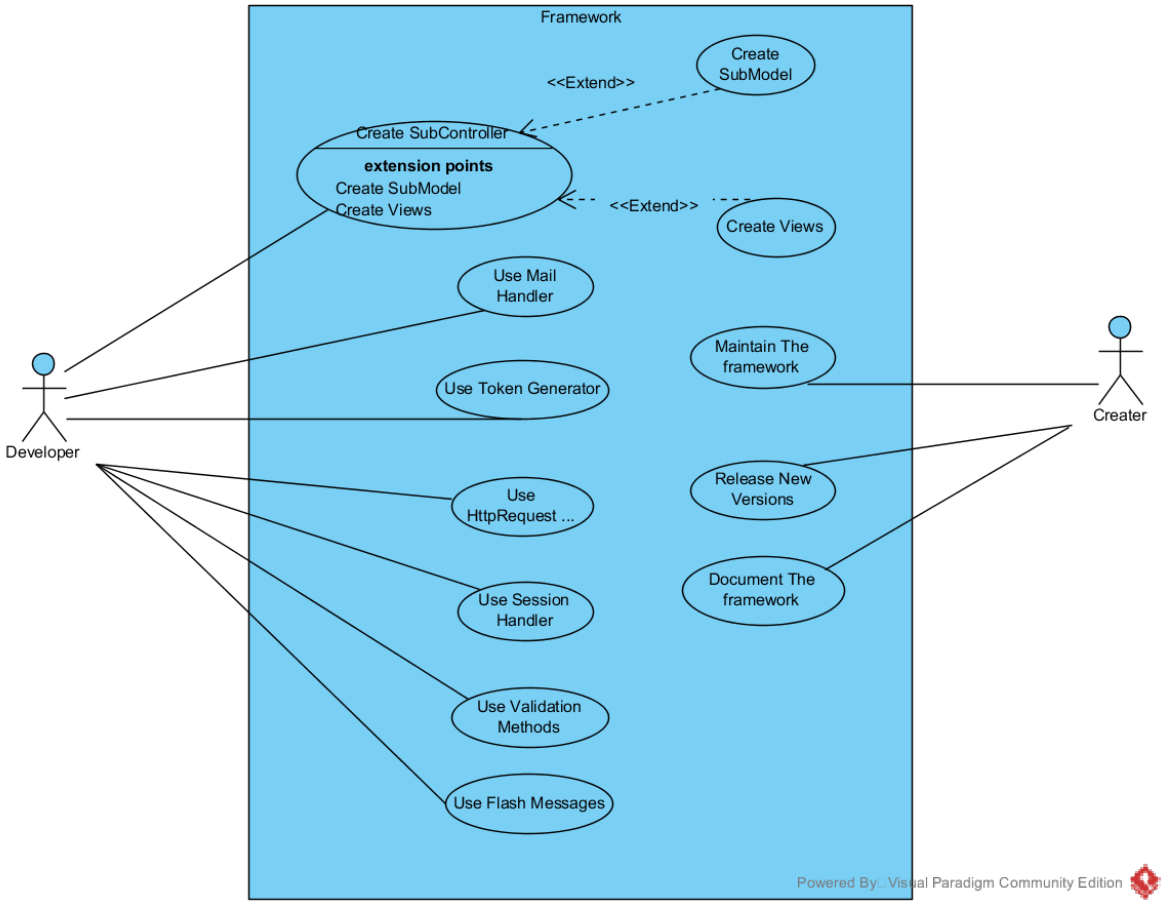

Fig. 3 Framework Use Case

4.5 Activity Diagram

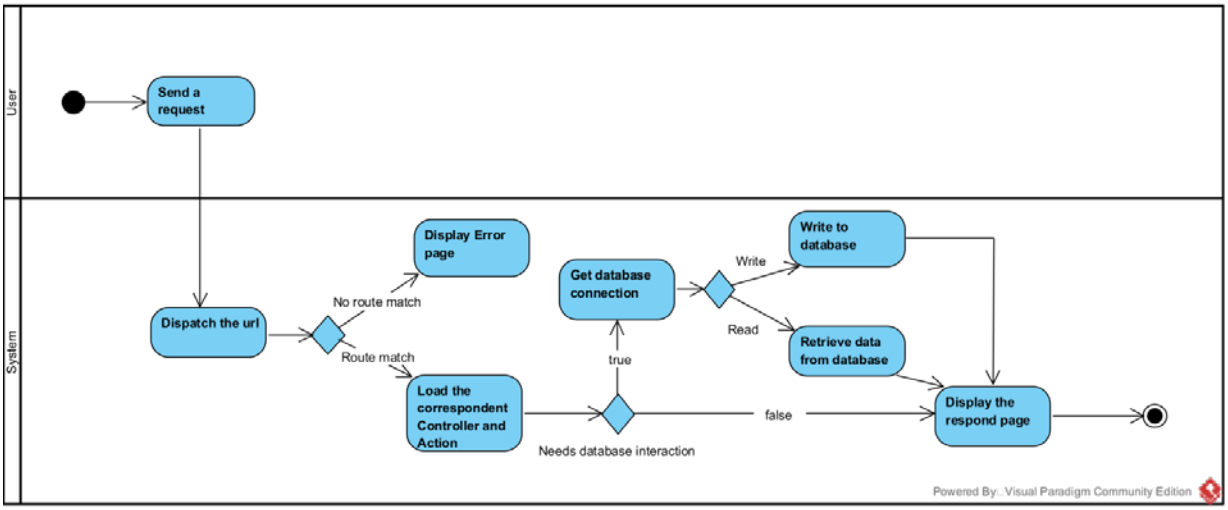

Fig. 4 Framework Activity Diagram

5 System Implementation

5.1 System Architecture

The Design chapter discussed the theoretical part of framework architecture. This chapter explains how the implementation is done using PHP with some other external tools. The framework uses a bootstrap file as an entry point i.e. front-controller. For routing the framework uses FastRoute package that handles urls decomposition. The resulted url fragments will be treated by the route filter class that will load the requested page if the url matches one of the predefined routes and load error page otherwise. The loaded controller will check whether the user authorized to access the page or not by calling the __ call magic method which is called when a non-existent or inaccessible method is called on an object of a sub-controller class. Used to execute before and after filter methods on action methods. Action methods need to be named with an "Action" suffix, e.g. indexAction, showAction etc. The Framework uses PHPdotenv package to load '.env' file configuration as environment variables where all system configuration lies in. If the request requires database interaction the controller retrieve the data from the correspondent model and pass it to the view. The view uses Twig template engine to render the page.

5.2 Class Diagram

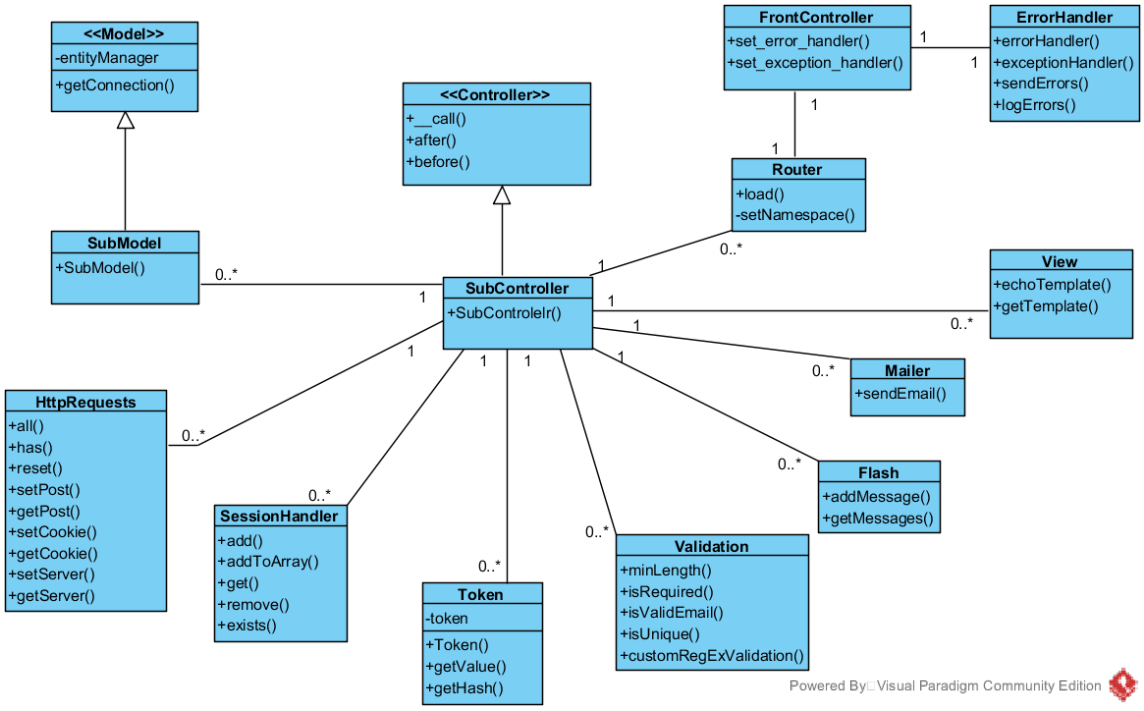

Fig. 5 Framework Implemetation Class Diagram

5.3 Files Structure

- It is a good practice to organize server-side code as in a Java application. One file per class and one directory per namespace.
- Place all classes in a separate directory, for example classes.
- Protect classes from direct HTTP access by denying access to the classes directory.
- Enable autoloading classes. This relieves us of include and require statements.

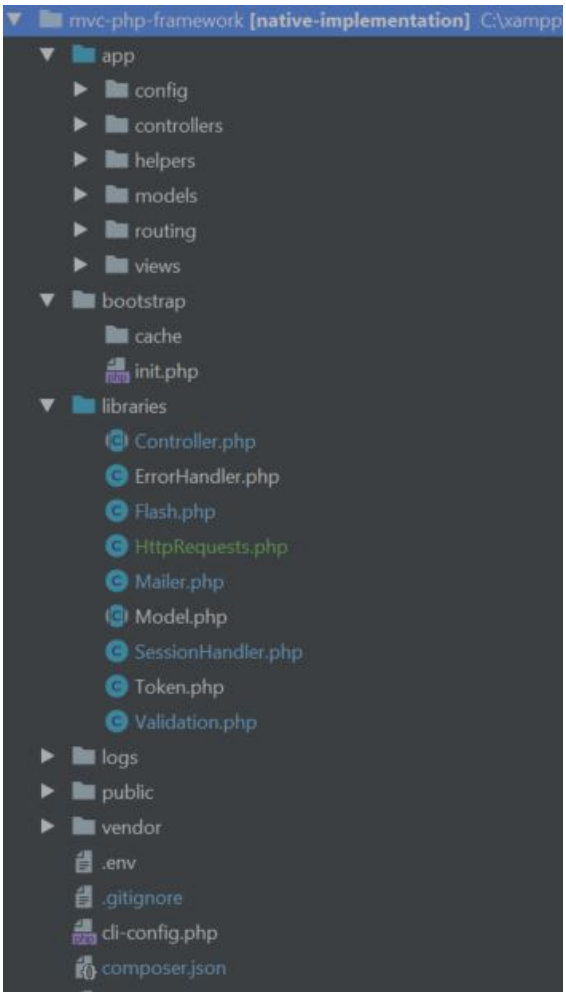

Fig. 6 Framework Files Structure

The figure shows how the files are structured. The ‘libraries’ directory is where the core of the framework lies in. The ‘app’ directory is where the web application code lies in where all controller classes created in the ‘controllers’ directory the same for model classes and for view files they repose in ‘models’ and ‘views’ directories respectively.

5.4 Database Design

For Database ORM implementation the framework uses the most powerful Doctrine 2 ORM that provides transparent persistence for PHP objects. It uses the Data Mapper pattern at the heart, aiming for a complete separation of your domain/business logic from the persistence in a relational database management system. The benefit of Doctrine for the programmer is the ability to focus on the object-oriented business logic and worry about persistence only as a secondary problem. This doesn't mean persistence is downplayed by Doctrine 2.

5.4.1 Entities

Entities are PHP Objects that can be identified over many requests by a unique identifier or primary key. These classes don't need to extend any abstract base class or interface.

An entity contains persistable properties. A persistable property is an instance variable of the entity that is saved into and retrieved from the database by Doctrine's data mapping capabilities. [6]

5.4.2 Mapping

Because Doctrine is a generic library, it only knows about your entities because you will describe their existence and structure using mapping metadata, which is configuration that tells Doctrine how your entity should be stored in the database. [7]

5.5 GUI Design

For GUI interface the implemented framework uses the popular twitter Bootstrap framework as default option. Bootstrap is a free and open-source front-end library for designing websites and web applications. It contains HTML- and CSS-based design templates for typography, forms, buttons, navigation and other interface components, as well as optional JavaScript extensions. Unlike many web frameworks, it concerns itself with front-end development only [8]. The developers can switch to any other front-end framework or simply use their own implementation.

5.6 External Tools

Apache 2.0: is a free and open-source cross-platform web server

PhpMyAdmin: is a free software tool written in PHP, intended to handle the administration of MySQL over the Web.

Composer: is a tool for dependency management in PHP

Doctrine 2: is an object-relational mapper (ORM) for PHP 5.4+ that provides transparent persistence for PHP objects

FastRoute: a fast implementation of a regular-expression based router

Twig: flexible, fast, and secure template engine for PHP

PHPmailer: email sending library for PHP

PHPdotenv: environment variables loader for PHP

Whoops: is an error handler framework for PHP

6 System Testing and Evaluation

The case study is to realize a sample university master registration website. The website let the users register an account using email address. The created user account will receive email verification to confirm the user email and the user can't login until he/she confirm their account. After confirming the account, the user can login and start filling their registration form and submit it when it's ready. This chapter will explain how all these steps are implemented using the project framework.

6.1 Installation

To start using the framework there are few steps:

1. The requirement must be met
2. Get a copy from the framework
3. Setup a virtual host custom url or uncomment the rewrite rules on ".htaccess" file
4. Change the ".env" file to match your environment configuration
5. To Switch between Developer mode and production mode set DEV_MODE constant in "env_config" file to true or false
6. Open the chosen url (virtual host) in any browser
7. You should get the welcome page!

Fig. 7 Welcome Page

6.2 Usability

To create the signup page, a Signup controller have to be create along with its related Model which in this case is the User model as shown in the figure below:

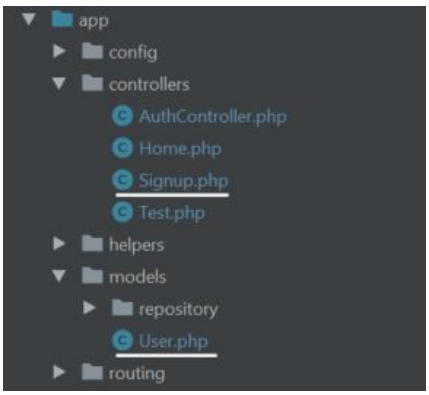

Fig. 8 Creating the Controller and its Model

The Signup controller has all the logic related to the signup process, and act as a middle man between the User model and the views related to it.

6.2.1 Signup Controller Snippet:

```php
<?php
/**
 * Signup.php
 */
namespace App\Controllers;


use App\Models\User;
use Libraries\HttpRequests;
use Libraries\View;

class Signup extends \Libraries\Controller
{

    public function indexAction()
    {
        echoTemplate('signup/signup.html');
    }

    public function createAction()
    {
        $requests = HttpRequests::all();
        $user = new User($requests->post);

        if ($user->create()){

            $user->sendActivationEmail();
            echoTemplate('signup/success.html');

        }else{
            echoTemplate('signup/signup.html', [
                'user' => $user
```

```
            ]);
        }
    }


    /**
     * Remote validation using AJAX
     *
     * @return void
     */
    public function validateEmailAction()
    {
        $is_valid = ! User::emailExists($_GET['email']);

        header('Content-Type: application/json');
        echo json_encode($is_valid);
    }

    public function activateAction($token)
    {
        if(User::activate($token)){
            redirectTo('/signup/activation-success');
        }
    }

    public function activationSuccessAction()
    {
        echoTemplate('signup/activationSuccess.html');
    }
}
```

6.2.2 User Model Snippet:

```
<?php
/**
 * User.php
 */

namespace App\Models;

use DateInterval;
use DateTime;
use Doctrine\Common\Collections\ArrayCollection;
use Exception;
use Libraries\Mailer;
use Libraries\Model;
use Libraries\Token;
use Libraries\Validation;

/**
 * @Entity @Table(name="users")
 * @Entity(repositoryClass="App\Models\Repository\UserRepository")
 */
class User
{
    /** @Id @Column(type="integer") @GeneratedValue */
    private $id;

    /** @Column(length=40) */
    private $firstName;
```

```
    /** @Column(length=40) */
    private $lastName;

    /** @Column(type="date", nullable=true) */
    private $birthday;

    /** @Column(length=40, unique=true) */
    private $email;

    /** @Column */
    private $passwordHash;

    /** @Column(length=20, nullable=true) */
    private $phoneNumber;

    /** @Column(nullable=true)*/
    private $address;

    /** @Column(type="datetime") */
    private $createdAt;

    /** @Column(type="datetime") */
    private $updatedAt;

    /** @Column(type="boolean") */
    private $emailConfirmed = false;

    /** @Column(unique=true, nullable=true) */
    private $emailActivationHash;

    /** @Column(nullable=true) */
    private $passwordResetHash;

    /** @Column(type="datetime", nullable=true) */
    private $passwordResetExpiresAt;


    /** @ManyToMany(targetEntity="Role", mappedBy="users") */
    private $roles;

    // unidirectional
    private $form;

    private $rememberedLogin;

    // for validation
    private $password;

    // for redirecting
    private $passwordResetToken;
    private $emailActivationToken;


    /**
     * Error messages
     *
     * @var array
     */
    private $errors = [];
```

```
    /**
     * Class constructor
     *
     * @return void
     */
    public function __construct($data)
    {
        $this->roles = new ArrayCollection();

        foreach ($data as $key => $value) {
            $key = convertToCamelCase($key);
            $this->$key = $value;
        };

        if (gettype($this->birthday) == 'string'){
            $this->birthday = new DateTime($this->birthday);
        }
        $this->setCreatedAt();
        $this->setUpdatedAt();
    }

    /**
     * Save the user model with the current property values
     *
     * @return boolean  True if the user was saved, false otherwise
     */
    public function create()
    {
        $this->validate();

        if (empty($this->errors)) {

            $token = new Token();
            $this->emailActivationHash = $token->getHash();
            $this->emailActivationToken = $token->getValue();

            $this->passwordHash = password_hash($this->password,
PASSWORD_DEFAULT);

            try {
                $entityManager = Model::getConnection();
                $entityManager->persist($this);
                $entityManager->flush();
            } catch (Exception $e) {
                return false;
            }

            return true;

        }

        return false;
    }
```

The User model uses doctrine data mapping to tell Doctrine how the data should be stored as in the above figure the annotation start with @ sign followed by the propriety name e.g “@id @column” given the option needed like attributes types and database proprieties.

```
/**
 * @Entity @Table(name="users")
 * @Entity(repositoryClass="App\Models\Repository\UserRepository")
 */
class User
{…}
```

By default, doctrine take the class name as the default table name in the database. It’s a good practice to explicitly name the tables so it won’t be some conflicts with the database reserved key as in the User class name. Same for variables they can be set to different names.

6.3 Graphical user interface

6.3.1 Home

Fig. 9 Website's home page

6.3.2 Signup page

Fig. 10 Website's signup page

6.3.3 Signup Success and Ask for email verification (Developer's mode)

Fig. 11 Signup success

6.3.4 New email received

Fig. 12 User recieves a confirmation email after singup

6.3.5 Confirmation Link

Fig. 13 Account confirmation link

6.3.6 Confirmation Success

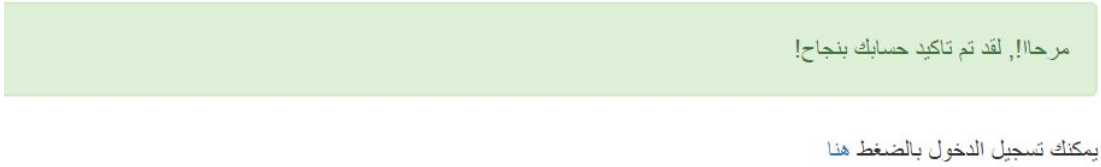

Fig. 14 Account confirmation success

6.3.7 Login page

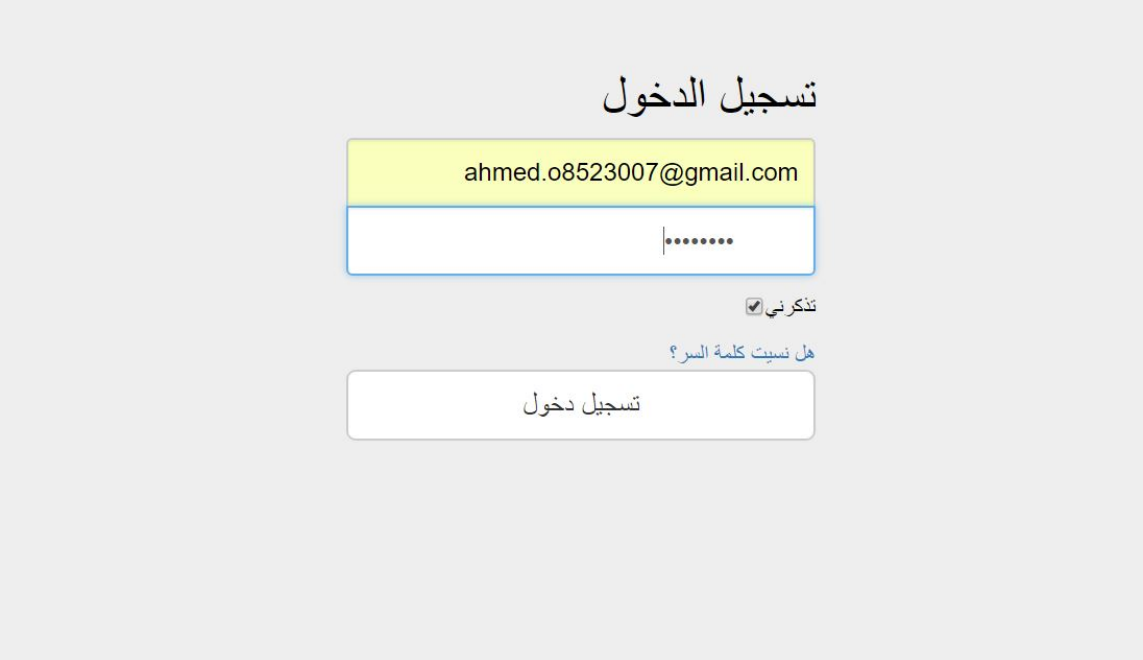

Fig. 15 Login page

6.3.8 Profile

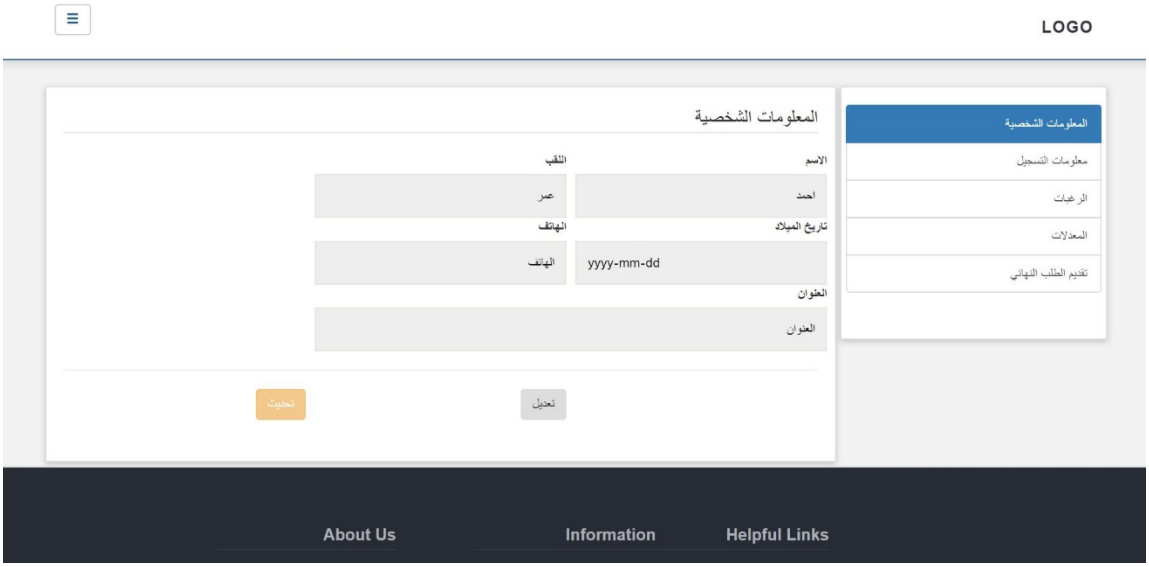

Fig. 16 Profile

6.3.9 Registration Section

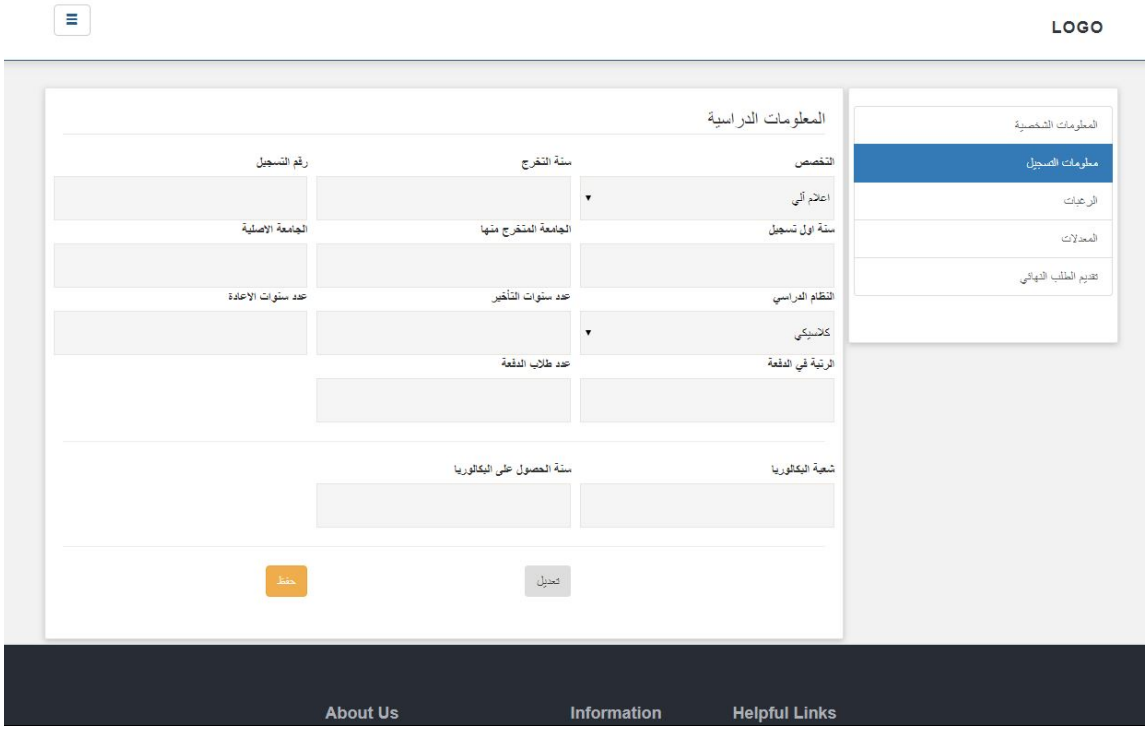

Fig. 17 Registration section

6.4 Exception handling

There are two modes for handling exception Developers mode and Production mode

6.4.1 Developer's mode (using Whoops debugging tool)

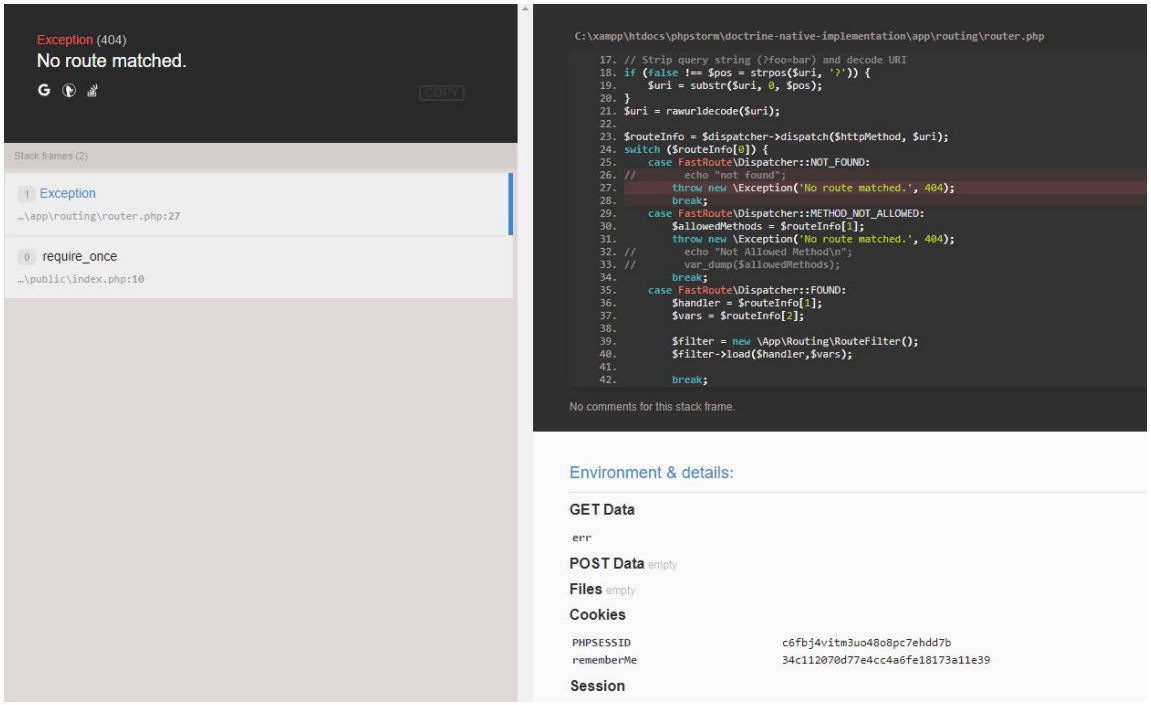

Fig. 18 Exception Handling in Developer's mode

6.4.2 Production mode

Fig. 19 Exception Handling in Production mode

7 Conclusion

A modern PHP framework is a powerful tool in a software developer's tool chest: it can save you a lot of time and effort and provide peace of mind as you develop your PHP application. Just remember that this shouldn’t replace learning core PHP first!

however, the key is selecting the right tool for the job. Every project is different, and even a framework that is perfect for one project might not be quite right for the next. Go for simplicity and ask yourself if a full-blown framework is right for the project, or if using a micro-framework could be a better solution.

8 Bibliography

[1] S. K. J. A. C. Xiaosong Li, "An Empirical Study of Three PHP Frameworks," 2017.

[2] "The MVC Pattern and PHP, Part 1," [Online]. Available: https://www.sitepoint.com/the-mvc-pattern-and-php-1/.

[3] "MVC Architecture," [Online]. Available: http://www.tutorialsteacher.com/mvc/mvc-architecture.

[4] "The MVC Pattern and PHP, Part 2," [Online]. Available: https://www.sitepoint.com/the-mvc-pattern-and-php-2/.

[5] "Object-relational mapping (ORM)," [Online]. Available: https://searchwindevelopment.techtarget.com/definition/object-relational-mapping.

[6] "Getting Started with Doctrine," [Online]. Available: https://www.doctrine-project.org/projects/doctrine-orm/en/latest/tutorials/getting-started.html.

[7] "Basic Mapping," [Online]. Available: https://www.doctrine-project.org/projects/doctrine-orm/en/2.6/reference/basic-mapping.html.

[8] "Bootstrap (front-end framework)," [Online]. Available: https://en.wikipedia.org/wiki/Bootstrap_(front-end_framework).