



HAZEM ALABIAD

SOPHOMORE "2'rd Grade"

21403802

BACK-END DEVELEPOER

INTERNSHIP DURATION: 30 DAYS

STARTED:10/07/2017 | ENDED: 18.8.2017



1. INTRODUCTION:

1.1. Internship Subject:

Web Development “Server-side Development”.

1.2. Internship Goals:

- 1.2.1. Server-side programming.
- 1.2.2. Dealing with the database.
- 1.2.3. Working on projects with VCS such as GitHub.

1.3. Internship Work and its result:

- 1.1.1. Learnt to develop as a team using VCS.
- 1.1.2. Learnt how to use NPM “server-lite” in the developing process.
- 1.1.3. Learnt MVC concept in the web developing processing.
- 1.1.4. Learnt the basics of “Laravel” framework primarily.
- 1.1.5. Had a brief look at “CodeIgniter” framework.
- 1.1.6. Learnt basics of “Bootstrap”.

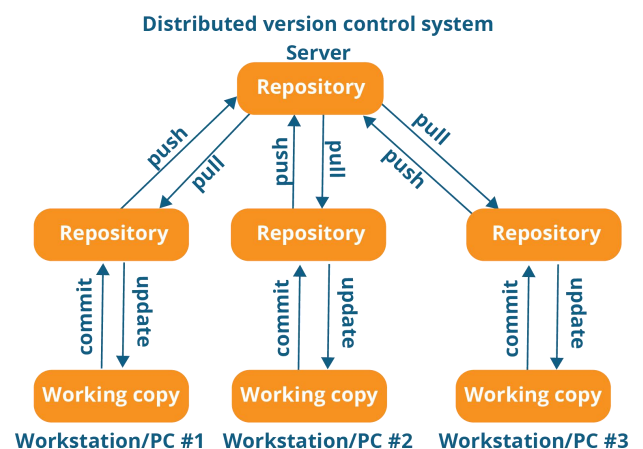
2. COMPANY INFORMATION:

ASE Bilişim: A Gaziantep Technopolis based company. It works mostly on Web projects for instance: Web Design, web programming, SEO, web application, E-commerce systems, mobile applications, MRP and CRM.

3. INTERNSHIP DURATION:

3.1. VCS:

An important technology that is used in almost every single company to develop applications and software as a team of developers. It gives the opportunity to track the developing



process and see who added, modified the code with notes and commits of each change a developer made to the code. It highlights the added, removed, modified lines also, makes us able to go back to previous versions of the code.

Additionally, it provides us with an important feature which is uploading the code to a cloud platform in order to access the code synchronized and updated anywhere.

First, this technology was invented by Linux Torvalds to help him in developing the Linux Kernel then, got public and started to be used in all companies.

Today we have Github, and BitBucket as the most popular web-based Git version control repository. During my internship, I used SourceTree “a Git GUI” that connects to Github and BitBucket and performs all actions using a desktop application available for all OS's.

There are two ways to perform the actions

- a) Using “cmd” on Windows, or “terminal” on Mac and Linux.
- b) Using the buttons provided by SourceTree.

- In order to start a project we go to the target directory and open terminal and type: *git init*

- To start the project on Github or BitBucket we create a repository and type in:

git remote add origin <repository link>

- To stage a file “track it”: *git add <filename>* , to stage all files in the directory: *git add .*

- To commit changes after staging a file's: *git commit -m “My Message”*

- To push changes online: *git push origin master -u*

- To pull code from cloud: *git pull -all*

- To get the status of the code at a specific moment: *git status*

- To get the brief log: *git log -online*

- To clone a project from an online repository:

git clone <repository link> <folder-name>

- To clone a repository from cloud: `git clone <repository link>`

- To revert to a previous version of the code:

git checkout <commit #> <filename>

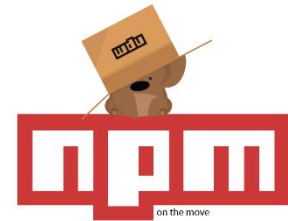
git reset HEAD

git checkout --<filename>

- Finally, merge command which is used when a team is working on and modifying the same file; we use merge to have it updated on all computers.

3.2. NPM:

NPM: Node Package Manager. A very useful tool that helps us have the web page we are working on updated and synchronized once we have saved the changed without any need to refresh the page manually on the browser.



To activate this feature we first need to install Node.js first from its website then, we open the terminal or cmd and type in this command: `npm init`

We get package.json file; Then we need to install lite-server by this command:

npm install lite-server --save-dev

Afterward, we edit package.json file as the following:

"scripts": {

"start": "npm run lite",

"test": "echo \"Error: no test specified\" && exit 1",

"lite": "lite-server"

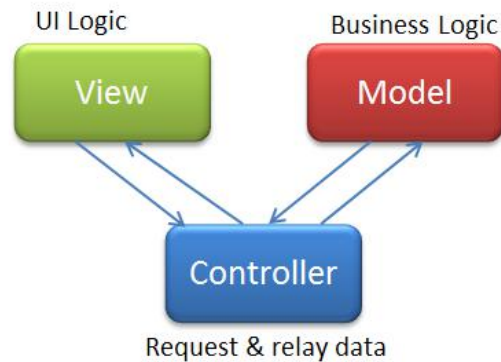
} ,

Then, on terminal: *npm start*

Finally, a tab opens on the browser and our server is ready to use!

3.3.MVC:

The Model-View-Controller (MVC) is an architectural pattern that separates an application into three main logical components: the model, the view, and the controller. Used to create extensible, and more secure and productive projects.



- Model: Responsible for database related business.
- View: Responsible the UI logic of the application, and rendering and displaying the pages that the final user interacts with.
- Controller: Responsible for interacting with the user, and updating the database. It acts as an interface between Model and View.
- This architectural design speeds up the developing process and enables us to use ready-made powerful and secure classes and functions that we always need; applying the programming rule: DRY. So, we can focus on the logical part of the project having better results.

3.4.Laravel:

- The most popular PHP framework that deals with server and database. It uses “Composer” to manage its dependencies so first, we need to install it from its website then, we open the terminal at the target directory and type:

composer create-project --prefer-dist laravel/laravel <project_name> “5.4”

Then, we go to the created project and type: *php artisan serve* to create a local virtual host.

- Root directory: consists of the following folders:

- a. app: contains the core code of our Laravel application.
- b. bootstrap: contains scripts that bootstraps the application.
- c. config: has the configuration files of the application.
- d. database: contains the database migration, and seeds.
- e. public: the document's root. It starts the application. It also contains assets like JavaScript, CSS, etc.
- f. resources: contains assets like LESS & Sass, language files, and templates that render the content as HTML pages to the user.
- g. storage: contains app storage, cache, and file uploads, etc.
- h. test: contains many test cases.
- i. vendor: contains composer dependencies.

- App directory: contains the following folders:

- a. console: stores the artisan commands.
- b. events: stores the events created by the app.
- c. exception: contains the exception handler.
- d. HTTP: contains controllers, filters, and requests.
- e. jobs: contains the queueable jobs for the app.
- f. listeners: contains the handlers classes for the event.
- g. policies.
- h. providers.

* Brief logical functionality of the framework:

- All requests are directed to public/index.php which is a start point for the application. It loads the autoloader and bootstrap/app.php.

- All the next incoming requests are sent to either HTTP/kernel or console/kernel.

* Simple Project: The idea behind this simple project is to practice the basic concepts in Laravel, for instance: initializing new project, creating a database and connecting it with the application, creating migrations, and creating view pages based on blade. It's a simple website build by Bootstrap 4.0 and Laravel which receive entries from the user and save them into the database and retrieve them again from the model and display them. Here are screenshots of the project:



This screenshot shows a teal 'Add New' button at the top center. Below it is the title 'ToDo List:' in a large, bold, black font.

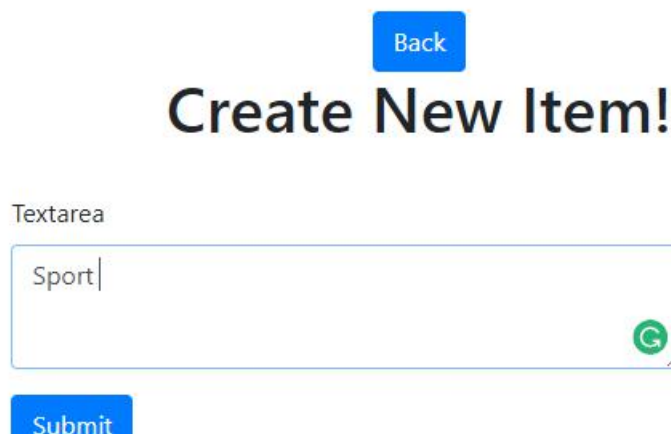
Figure 1

Figure 2



This screenshot shows a single text input field with a light gray border. The text 'BBM203 2'nd' is entered into the field.

Figure 3



This screenshot shows a form titled 'Create New Item!' in a large, bold, black font. Above the title is a blue 'Back' button. Below the title is a large text area with a light blue border. The text 'Sport' is entered into the text area, and a green circular icon with a white 'G' is visible in the bottom right corner of the text area. Below the text area is a blue 'Submit' button.

Figure 4

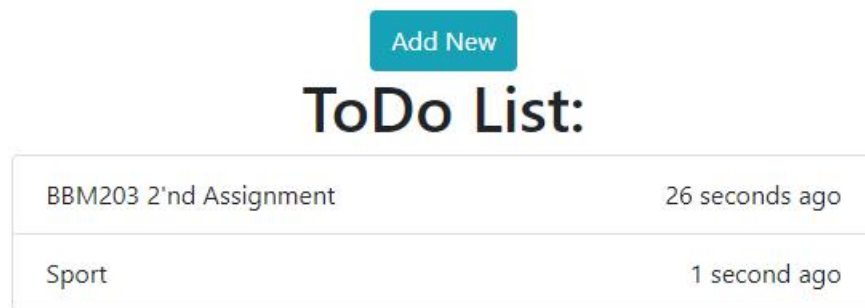


Figure 5

- And here are the notes I wrote down while learning Laravel:

Laravel 5.4:

1. To transfer an array of data from a controller to a view page:

`$ourArray = [...]`

1) `return view('pageName')->withName($ourArray);`

*Name: to be used in view page. If Name starts with N or n it's the same.

2) `return view('pageName')->with(['nameToBeUsedInView' => $ourArray]);`

3) `return view('pageName', ['nameToBeUsedInView' => $ourArray]);`

4) `return view('pageName', compact($ourArray));`

2. @unless(condition)

`## do something ## if condition was not matched`

`@endunless`

3. To call a specific controller's method:

`Route::get('/', 'controllerName@actionName');`

4. Relationship:

- a) One-One: `hasOne(Class) / belongsTo(Class)`
- b) One-Many: `hasMany(Class) / belongsTo(Class)`
- c) Many-Many: `belongsToMany(Class)` in the related models.
- d) Many-Through: `hasManyThrough(Class1, Class2)`
- e) Polymorphic: In the shared model:
`myFunction(){morphTo()}`
In the related models `morphMany(sharedModelName::class, myFunction)`
- f) Many-Many Polymorphic: in the related models: `morphToMany(Class, relation)`
, in the shared model: create function for each model contains:
`morphByMany(Model::class, relation)`

5. Accessors:

Inside the model: `getDataNameAttribute()`

3.5. Bootstrap:

A front-end framework built by Twitter for responsive, mobile-first web pages. It uses HTML, CSS, JavaScript. It eases the work and enables us to write less code and makes the page organized by its grid system. It's has been the most popular web-end UI framework.

* Mini Project: The project idea is to create a responsive web page using Bootstrap 4.0. This project is to develop using Node Lite Server to display the code live on a browser. Also, the created project will be uploaded on BitBucket Git platform.

First and after downloading NPM, we go to the directory at which we want to save our project in then, we type:

```
git npm init
```

- A file called package.json appears:

```
git npm install lite-server --save-dev
```

- A folder named node_modules will be installed. We then open package.json file and modify script body as the following:

```
"scripts": {  
  
  "start": "npm run lite",  
  
  "test": "echo \"Error: no test specified\" && exit 1",  
  
  "lite": "lite-server"  
  
},
```

- For uploading the project into BitBucket we type the following commands:

```
git init
```

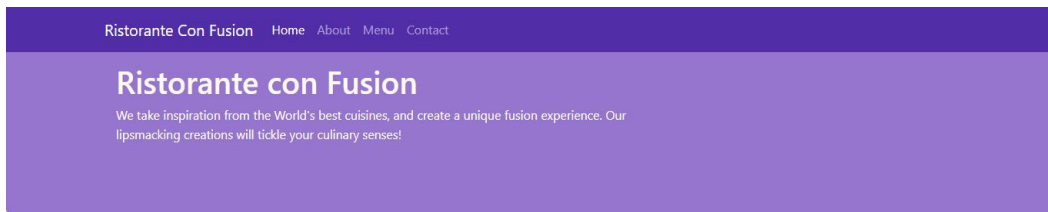
```
git remote add origin <repository_link>
```

- Then we create “.gitignore” file which includes the files names which will not be staged and uploaded. The next is to download bootstrap by typing:

```
npm install bootstrap@4.0.0-beta
```

- To start the NPM server lite: *npm start*

- Finally, our project is ready to start developing.



Uthappizza

A unique combination of Indian Uthappam (pancake) and Italian pizza, topped with Cerignola olives, ripe vine cherry tomatoes, Vidalia onion, Guntur chillies and Buffalo Paneer.

Our Lipsmacking Culinary Creations

This Month's Promotions

Weekend Grand Buffet

Featuring mouthwatering combinations with a choice of five different salads, six enticing appetizers, six main entrees and five choicest desserts. Free flowing bubbly and soft drinks. All for just \$19.99 per person

Alberto Somayya

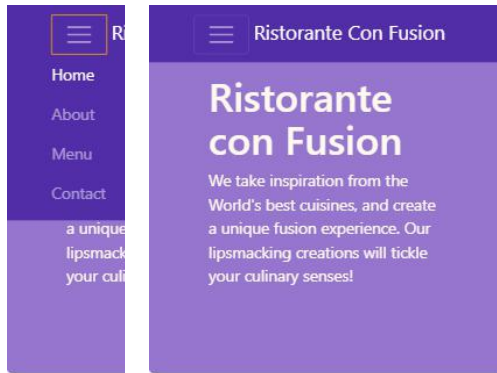
Executive Chef

Award winning three-star Michelin chef with wide International experience having worked closely with whos-who in the culinary world, he specializes in creating mouthwatering Indo-Italian fusion experiences.

Meet our Culinary Specialists



Here are screenshots of the projects in mobile view and laptop view:



Uthap

A unique cor
Uthappam (p
topped with
cherry tomat
burnt and n

Our Lip Culinar

Uthappizza

A unique combination of Indian
Uthappam (pancake) and Italian pizza,
topped with Cerignola olives, ripe vine
cherry tomatoes, Vidalia onion, Guntur
burnt and n

Our Lipsmacking Culinary Creations

This Mo Promo Week Buffet

Featuring me
with a choice
enticing app
five choicest
and soft drin
person

This Month's Promotions Weekend Grand Buffet

Featuring mouthwatering combinations
with a choice of five different salads, six
enticing appetizers, six main entrees and
five choicest desserts. Free flowing bubbly
and soft drinks. All for just \$19.99 per
person

Albert Executiv

Award winni
with wide Int
worked close
culinary worl
mouthwateri
experiences.

Meet o Special

Alberto Somayya Executive Chef

Award winning three-star Michelin chef
with wide International experience having
worked closely with whos-who in the
culinary world. he specializes in creating
mouthwatering Indo-Italian fusion
experiences.

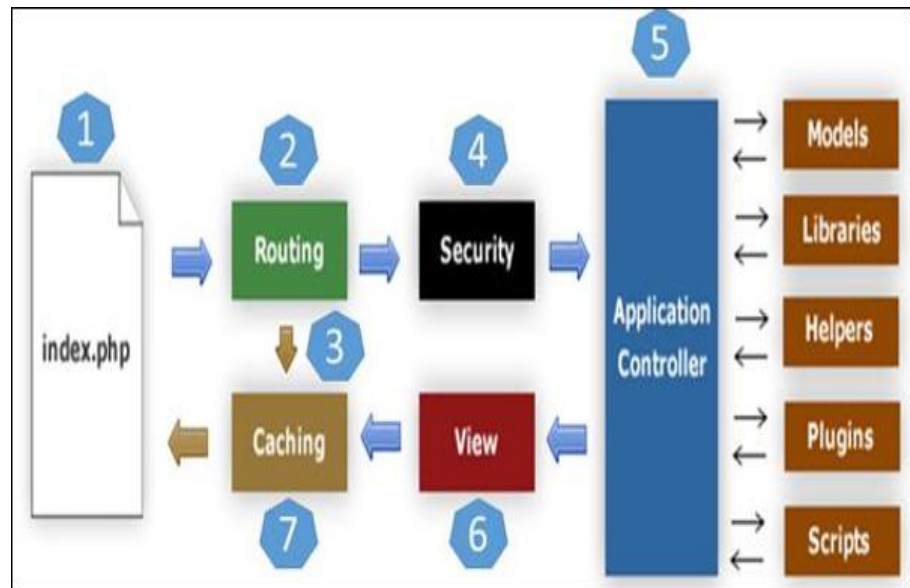
Meet our Culinary Specialists



CodeIgniter: One of the most famous PHP framework; usually used in small and simple projects.

Created by Rick Ellis in 2006.

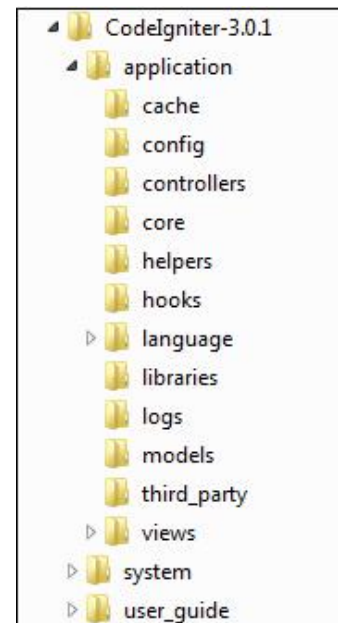
CodeIgniter was born from ExpressionEngine, essentially a collection of refactored classes originally written for EllisLab's



flagship CMS. - To start a project, we need to download the CI folder from its website then unzip it. - How does it work? All requests go to index.php page. Routing will decide whether the requested page exists in caching if so, the requested page will go back to the user if not, the request is sent to Security checks then, the controller loads the needed Plugins, Libraries, Helpers, Models, etc and sends the request to View. View renders the page and passes it on to Caching.

* Root Directory:

1. Application: The code we write.
2. System: The core code.
 - Inside application folder, we have:
 - a. cache: the cache for the application.
 - b. config: configuration for the app through config.php and database.php
 - c. controllers: stores the app controllers.
 - d. core: contains the base class of the app.
 - e. helpers: contains ready made classes and functions that ease the process.



- f. hooks.
- g. language.
- h. logs.
- i. models: contains our database.
- j. third_party .
- k. views: contains HTML files.

Note: In Codelgniter, there were small exercises but no mini projects as it was not my primary goal.

4. RESULT:

- This internship introduced me to Web Development (both “front-end” & “back-end”) and VCS which I can use even in non-web development projects. Now, I have the basic knowledge in Web Development and can go further and learn more about this interesting field which is being in a very rapid advancement to meet the real needs by creating new technologies, and patterns to make the procedure easier and more productive.
- In addition, understood what is MVC architecture that is used in almost all web-projects so now, I can learn any MVC framework as all have the same architecture with some differences. With a bit of working and experience, I could create a real web application by working on UI and both front and back ends.
- Gave me experience in working with development teams and integrate my code with others code to have the full and complete project done.

5. REFERENCES:

- ☪ <https://www.atlassian.com/git/tutorials>
- ☪ <https://www.youtube.com/watch?v=mYjZtU1-u9Y&list=PL1F56EA413018EEE1>
- ☪ https://www.youtube.com/watch?v=Y9XZQO1n_7c

- ☞ https://www.youtube.com/watch?v=Ytux4lOAR_s&list=PLAwxTw4SYaPk8_-6lGxJtD3i2QAU5_s_p
- ☞ <https://git-scm.com/>
- ☞ <https://www.tutorialspoint.com/git/>
- ☞ <https://www.coursera.org/learn/bootstrap-4>
- ☞ <https://www.youtube.com/playlist?list=PL442FA2C127377F07>
- ☞ <https://www.w3schools.com/php>
- ☞ <http://php.net/manual/en/intro-what-is.php>
- ☞ <https://www.lynda.com/PHP-tutorials/Welcoming/587674/623942-4.html>
- ☞ <https://laravel.com/docs/5.4>
- ☞ https://www.youtube.com/playlist?list=PLe30vg_FG4OQz1yZq0z19ZuWD_C3MZbA4&disable_polymer=true
- ☞ <https://laracasts.com/series/laravel-from-scratch-2017>
- ☞ <https://www.tutorialspoint.com/laravel/>
- ☞ <https://getbootstrap.com/docs/4.0/getting-started/introduction/>
- ☞ <https://bootswatch.com>
- ☞ <http://builtwithbootstrap.com/>
- ☞ <https://www.lynda.com/CodeIgniter-tutorials/Learn-CodeIgniter-3-Basics/505770-2.html>
- ☞ <https://www.youtube.com/playlist?list=PLh89M5IS1CIATULcdS4UHx9pj3GGW8nNM>
- ☞ <https://www.youtube.com/playlist?list=PLUpnKy5Si8zDouvZiUMHwSSyVrowJmH22>