Introduction

In this project you will learn the **basic notions of the famous PHP language** which is so used in the world of web development. What distinguishes **PHP** from other languages such as Javascript is that the **code is executed on the server**, generating HTML and sending it to the client. The client will receive the result of running the script.

What are the main objectives in this project?

- Learn the basics to program in **PHP**
- Understand what a **server-side language** is and what it is used for

1. General analysis

1.1. Install PHP environment

It is important that before starting you have an **environment** capable of **running PHP** in a version higher than 7.2. As we are going to use **XAMPP** we will be running **version** 8 which is the newest one.

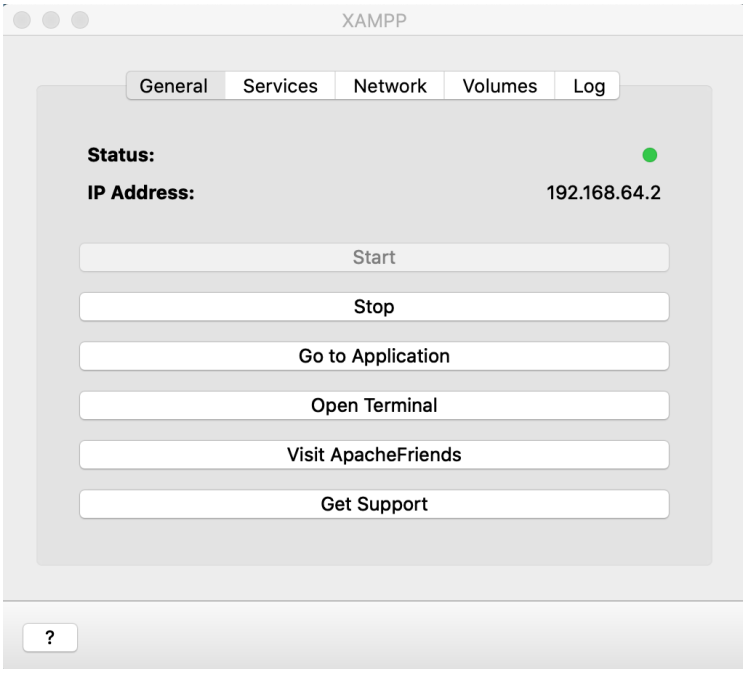
PHP is widely used to develop with a **XAMPP environment** which is basically a stack of web technologies such as web server, php interpreter and a **MariaDB database**, you can see what a XAMPP environment is here .

Since **PHP** is designed for the web it requires a **web server**. So to give an easier approach to **install all these services** at once and across **Linux**, **Mac** and **Windows** operating systems the Apache Friends created XAMPP.

Let's go to the home page and **download** the version for your operating system. There is an installer which use to be a simple installation. You also can use google to find out some tutorial of XAMPP installation on your own operating system.

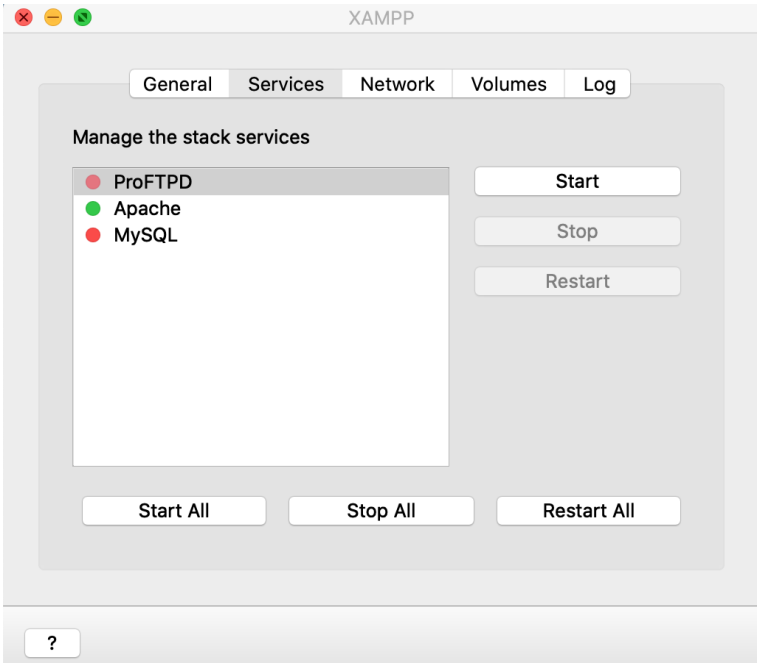
1.2. Set Up XAMPP

Once the installer finishes **open the XAMPP application** which will look like this if you are using a MacOS:



Press **start** in order to turn on the services. This will start the stack which is formed by ProFTPD which is a ftp service, **Apache** which is a **web server** without it we cannot run **php** for **web development**. And also **MySQL** which is as you know a **database engine** one of the most popular in fact. You can see the services in the

Services tab. There you can stop the services that we are not going to use for the moment, we just need the **Apache** running.



Here you have a video that shows how to install XAMPP on Windows:

- https://www.youtube.com/watch?v=h6DEDm7C37A

1.2.1. Web Server

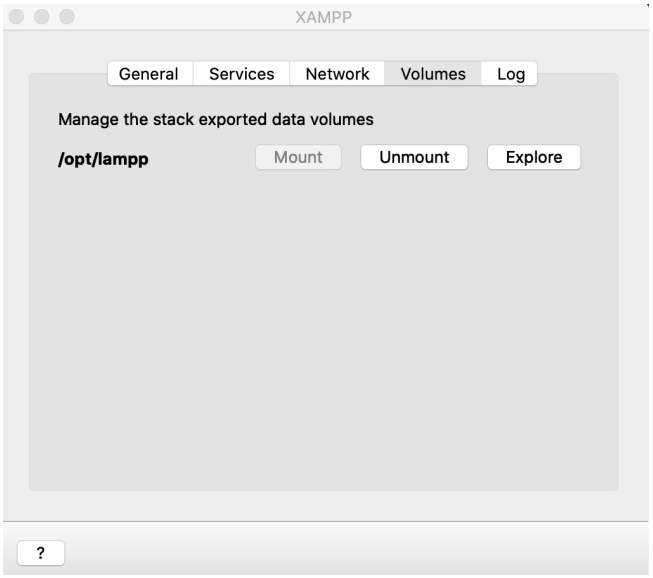
A **web server** is a **server software**, or a hardware machine dedicated to running this software, that can satisfy client requests on the **World Wide Web**. A **web server** can, in general, contain **one or more websites**. A **web server** processes incoming network requests over **HTTP** and several other related protocols. With a web serverwe can publish a web or application on the **World Wide Web**.

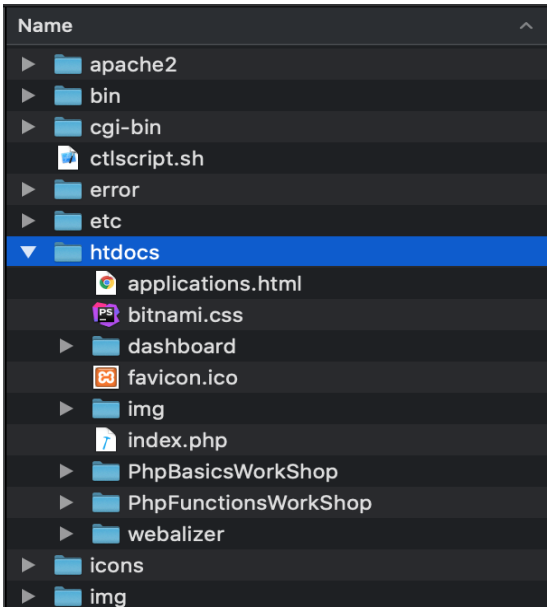
- What is a web server?
    - https://www.youtube.com/watch?v=Yt1nesKi5Ec
- Web server basics
    - https://www.youtube.com/watch?v=3VqfpVKvlxQ

The most well known web servers are **Apache** and **Nginx**.

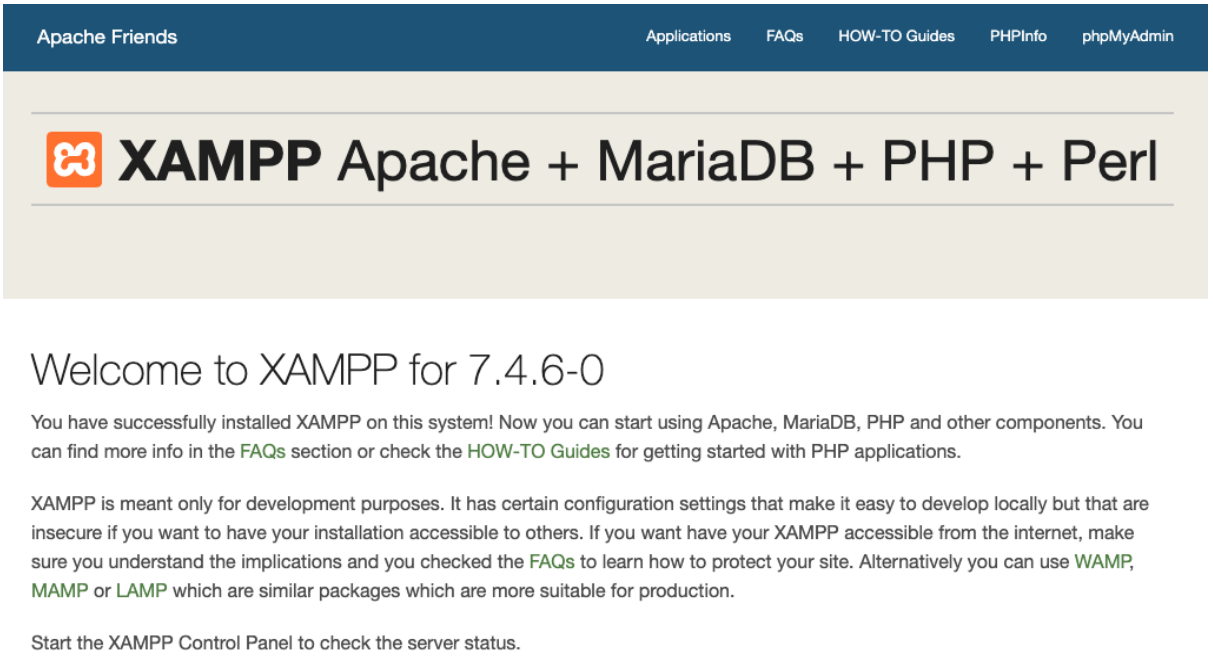
- ¿What is NGINX y APACHE?
    - https://www.youtube.com/watch?v=QiUV9b6sC_U
- Apache vs Nginx
    - https://www.youtube.com/watch?v=ZhfpYgl8BtQ

A **web server** has a **root folder** in which applications are installed. Typically this folder is **htdocs** for **Apache**. So now that we have our Apache installed under the **XAMPP** stack you need to know where is the **root folder** of the **web server**. So we can go to **XAMPP application** and press the tab Volumes and press mount button once it is mounted you can press the button explore to open the XAMPP directory.



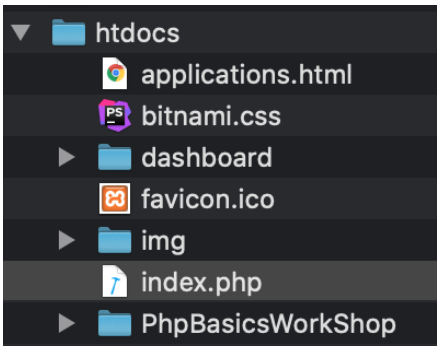


All your **PHP web applications** must reside inside this **htdocs folder**. If you go now to the **General tab** of **XAMPP** we can press "**Go to application**" which will open a web browser with the dashboard application.



There you have some **interesting information**. Take a look.

So our web server is pointing to this local address: **http://127.0.0.1:80/** or **http://localhost/** we have opened **http://127.0.0.1:80/dashboard/** which is the integrated dashboard. But we want to run our applications with PHP so for that we need to create a directory inside of **htdocs** for our project.



As you can see I've created a folder called **PhpBasicsWorkShop** which has some php files that will be interpreted as soon as I point my web browser over this directory in this way,

http://127.0.0.1:80/PhpBasicsWorkShop

To open in the web browser the directory which will list the php files since there is no an **index.php** file which would be executed directly, otherwise if **index.php** file is not, it lists the contents of directory, So to run a file over the web browser we can do this:

http://127.0.0.1:80/PhpBasicsWorkShop/arrays.php

Which will show the result of the processing of the script file in this case **arrays.php**.

If you are having trouble preparing the entire environment to execute PHP code, we recommend viewing the following video:

- How to run your first PHP program in XAMPP server
    - https://www.youtube.com/watch?v=TjFRTkw6GDQ

1.2. Php configuration file

As a server side programming language PHP has a configuration file called **php.ini**.

Which determines things like: memory of the system that PHP is going to use, the time a request or process can be alive, the size of a file that can be uploaded to the server, location of PHP log files, activation of modules and more.

You can see the file once you have the XAMPP environment installed. Each operating system has its own path to this file but typically we could find: /YourSystemPath/**xampp/volumes/root/etc/php.ini**

We can also see how it is PHP configured by using a call to the function **phpinfo()**.

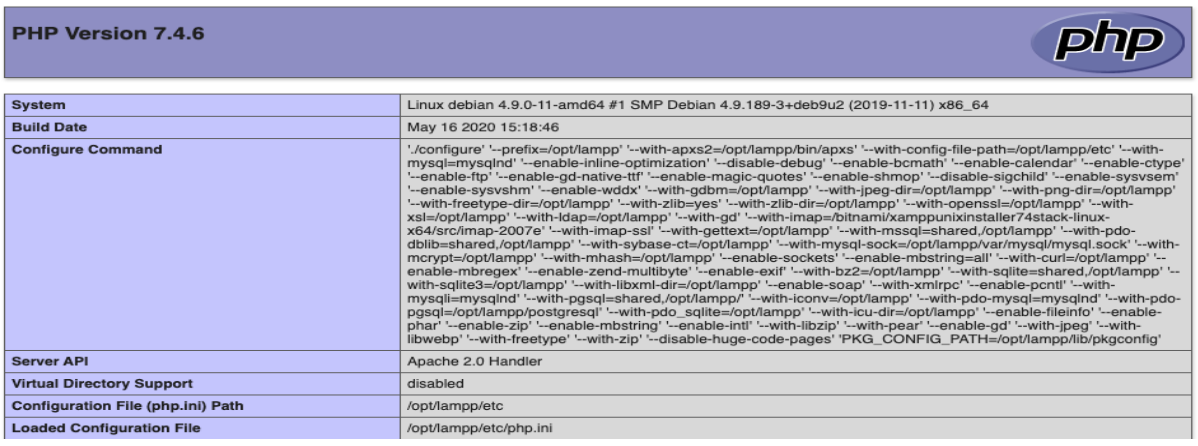
Create a folder called **phpinfo** and inside create a file called **index.php**, add this to it:

```
<?php

  phpinfo();

?>
```

Once done, point your web browser against this new directory. in this way.

http://127.0.0.1:80/phpinfo

You'll see the actual php configuration,



PHP Version 7.4.6

System	Linux debian 4.9.0-11-amd64 #1 SMP Debian 4.9.189-3+deb9u2 (2019-11-11) x86_64
Build Date	May 16 2020 15:18:46
Configure Command	'./configure' '--prefix=/opt/lampp' '--with-apxs2=/opt/lampp/bin/apxs' '--with-config-file-path=/opt/lampp/etc' '--with-mysql=mysqlnd' '--enable-inline-optimization' '--disable-debug' '--enable-bcmath' '--enable-calendar' '--enable-ctype' '--enable-ftp' '--enable-gd-native-ttf' '--enable-magic-quotes' '--enable-shmop' '--disable-sigchild' '--enable-sysvsem' '--enable-sysvshm' '--enable-wddx' '--with-gdbm=/opt/lampp' '--with-jpeg-dir=/opt/lampp' '--with-png-dir=/opt/lampp' '--with-freetype-dir=/opt/lampp' '--with-zlib=yes' '--with-zlib-dir=/opt/lampp' '--with-openssl=/opt/lampp' '--with-xsl=/opt/lampp' '--with-ldap=/opt/lampp' '--with-gd' '--with-imap=/bitnami/xamppunixinstaller74stack-linux-x64/src/imap-2007e' '--with-imap-ssl' '--with-gettext=/opt/lampp' '--with-mssql=shared,/opt/lampp' '--with-pdo-dblib=shared,/opt/lampp' '--with-sybase-ct=/opt/lampp' '--with-mysql-sock=/opt/lampp/var/mysql/mysql.sock' '--with-mcrypt=/opt/lampp' '--with-mhash=/opt/lampp' '--enable-sockets' '--enable-mbstring=all' '--with-curl=/opt/lampp' '--enable-mbregex' '--enable-zend-multibyte' '--enable-exif' '--with-bz2=/opt/lampp' '--with-sqlite=shared,/opt/lampp' '--with-sqlite3=/opt/lampp' '--with-libxml-dir=/opt/lampp' '--enable-soap' '--with-xmlrpc' '--enable-pcntl' '--with-mysqli=mysqlnd' '--with-pgsql=shared,/opt/lampp/' '--with-iconv=/opt/lampp' '--with-pdo-mysql=mysqlnd' '--with-pdo-pgsql=/opt/lampp/postgresql' '--with-pdo_sqlite=/opt/lampp' '--with-icu-dir=/opt/lampp' '--enable-fileinfo' '--enable-phar' '--enable-zip' '--enable-mbstring' '--enable-intl' '--with-libzip' '--with-pear' '--enable-gd' '--with-jpeg' '--with-libwebp' '--with-freetype' '--with-zip' '--disable-huge-code-pages' 'PKG_CONFIG_PATH=/opt/lampp/lib/pkgconfig'
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/opt/lampp/etc
Loaded Configuration File	/opt/lampp/etc/php.ini

1.2. Create a new project

The first step will be to **fork the PHP project.**

https://github.com/assembler-institute/php-basics

This is the PHP documentation tree which is really useful:

https://www.php.net/manual/en/

1.3. Generate a library of functionalities:

Next you will create a file library to be able to have your own reference to help you to locate the different features that php offers in general terms:

- Create a file called **print.php.** This file as its name indicates will be used to print on screen:
    - Generate an instruction that makes use of "echo"
    - Generate an instruction that makes use of "print"
    - Generate an instruction that makes use of "print_r", it is important that you assign a complex value to analyze its potential
- Create a file called **iterators.php.** This file as its name indicates will be used for the start-up of iterators:
    - Generate a snippet that makes use of for
    - Generate a snippet that makes use of foreach
    - Generate a snippet that uses while
    - Generate a snippet that uses do while
- Create a file called **operators.php** in this case you will need to use the **var_dump PHP function** which is to debug variables. So you can do: **var_dump(1 == 2);** This file as its name indicates will be used for working with operators:
    - Create an example of use for arithmetic operators: +, -, *, /, and%
    - Create a usage example for comparison operators: ==,! =, <,>, <=,> =
    - Create an example of use for logical operators: &&, And; ||, Or; ! (NOT); Xor
- Create a file called **dates.php.** This file as its name indicates will be used for work with dates. We will be using dates from classes in this case from **DateTime** class forgetting the use of old php functions.

        - Instance the Date Time class and then invoke the format method with the argument “Y-m-d” to show year-month-day
        - Get the current date in any format
        - Get the current day
        - Get the current month in numerical format (1-12)
        - Get the current minute with leading zeros (00 - 59)
- Create a file called **conditionals.php**
    - This file as its name indicates will be used for working with conditionals:
        - Create a simple condition that evaluates whether the current day is Monday. Only in the case that the condition is met, it shows a message of “**We are on Monday**”.
        - Create a simple condition that evaluates whether the current month is October. If the condition is met, it shows a message of the type "**We are in October**". Otherwise, if this condition is not met, show the current month in words as it come from DateTime.
        - Create a **double condition** that evaluates:
            - If the current minute is less than 10. Displays a message of type "**the current minute is less than 10**", if the current minute is greater than 15, displays a message of the type "**the current minute is more than 15**". If you do not meet any of the two conditions above: Displays a message of the type "**does not meet any conditions**”
        - Create a switch type control structure to display a different message depending on the current day of the week. You can write any type of message, because the important thing is that you understand how it works and in what cases you can use it.
- Create a file called **types.php**
    - This file as its name properly indicates will be used for working with the different basic types of variables and data:
        - Define a new variable and assign a value to each of the following types:
            - boolean
            - integer
            - float
            - string
            - array
            - object
            - NULL
- Create a file called **maths.php**

    - This file as its name properly indicates will be used for working with mathematical operators and the specific PHP functions for:
        - Define a variable whose value is the result of the function that returns an **absolute value**.
        - Define a variable whose value is the result of the function that returns a **rounded value** to the next **highest integer**.
        - Define a variable whose value is the result of the function that returns the **highest value** of a series of values that are **received by parameter**.
        - Define a variable whose value is the result of the function that returns the **lowest value** of a series of values that are **received by parameter**.
        - Define a variable whose value is the result of the function that returns a **random number**
- Create a file called **strings.php**
    - This file as its name indicates will be used to work with text strings:
        - Print a text string
        - Print a text string that interpret variables
        - Concatenate a previously declared variable in a text string
        - Execute the function that allows you to replace text in a string (case sensitive)
        - Execute the function that allows you to replace text in a string (without taking into account upper / lower case)
        - Execute the function that allows you to write a text N times
        - Execute the function that allows to obtain the length of a text string
        - Executes the function that allows to obtain the position of the first occurrence of a text within a text string
        - Execute the function that allows a text string to be capitalized
        - Execute the function that allows you to transform a text string to lowercase
        - Execute the function that allows to obtain a text substring from a given position
- Create a file called **arrays.php**
    - This file as its name indicates will be used to work with arrays:
        - Define a simple array composed of text strings
        - Define a simple array consisting of whole numbers and decimal numbers
        - Define a multidimensional array
        - Execute the function that allows to obtain the length of an array
        - Execute the function that allows to obtain the combination of two arrays

        - Execute the function that once is given an array return the last element of it
        - Execute the function that once is given an array add a new element to the array in question
- Create a file called **functions.php**
    - This file will be used to store your first tests regarding the implementation of functions:
        - Create a function that given two numbers returns the sum of both
        - Create a function that given two numbers returns the multiplication of both
        - Create a function that given two numbers returns the division of both
        - Create a function that, given two numbers and an operation (add, multiply or divide), returns the result of that operation.
    - Depending on the type of operation received by parameter, the function will execute the function responsible for performing the operation, since you have previously implemented the function for each operation separately.
- Create a file called **phpinfo.php**
    - This file as its name indicates will be used to work with the phpinfo function:
        - Call the phpinfo function and verify the result

It is important to place comments in each code snippet that indicate what it is used for. These comments should help you to remember at all times what is your operation as a library

2. Pill Organization

It’s important to document your own work in each pill or project you do. In this way you can easily after a time to read your own resume of what you did in each project.

So since we use a repository we have a **README.md** file which can be used to document how our application works. In the resources section there are some links about github guidelines to document your project.

3. Requirements

- You must use **GIT**
- You must use **PHP v8**
- Create a **clear and ordered directory structure**
- Both the **code** and the **comments** must be written in **English**
- Use the **camelCase** code style to define **variables** and **functions**
- In the case of using HTML, **never use online styles**
- In the case of using different programming languages always **define the implementation in separate terms**
- Remember that it is important to **divide the tasks** into **several sub-tasks** so that in this way you can associate each particular step of the construction with a **specific commit**
- You should try as much as possible that the **commits** and the **planned tasks** are the same
- **Delete files** that are **not used** or are **not necessary** to evaluate the project

4. Deliverables

The following deliverables will be necessary to evaluate the project:

- Forked repository with code
    - https://github.com/assembler-institute/php-basics
    - You must create a correctly documented **README** file in the root directory of the project (see guidelines in Resources)

5. Resources

- What can PHP do? [Official Website]: https://www.php.net/manual/es/intro-whatcando.php
- Sample guide for README:
    - https://gist.github.com/PurpleBooth/109311bb0361f32d87a2
    - https://gist.github.com/Villanuevand/6386899f70346d4580c723232524d35a
    - https://gitmoji.carloscuesta.me/
- XAMPP: https://www.apachefriends.org/es/index.html
- How to install XAMPP on Windows: https://www.youtube.com/watch?v=h6DEDm7C37A
- What is a web server? https://www.youtube.com/watch?v=Yt1nesKi5Ec
- Web server basics: https://www.youtube.com/watch?v=3VqfpVKvlxQ
- What is NGINX y APACHE? https://www.youtube.com/watch?v=QiUV9b6sC_U
- Apache vs Nginx: https://www.youtube.com/watch?v=ZhfpYgl8BtQ