Composer PHP

Project objective

- Understand that it is a dependency manager.
- Understand that it is Composer and what its fundamentals are.
- Improve your PHP knowledge in professional work environments

Analysis

The first source point of this project is to draw up a list of what dependencies your project needs. To use as a dependency a library called Guzzle.
We install Guzzle using composer and other necessary libraries.

Before installing Guzzle we need to install Composer
https://getcomposer.org/doc/00-intro.md#system-requirements

We run the imported script

Which is composer

It is not a package manager, it is packages but manages it by projects by installing a directory (vendor), does not install globally.
A dependency manager, supports global project through global command
is a tool for managing dependencies in PHP. It allows you to declare the libraries on which your project depends and manages them (both in its installation and in its update).
It's inspired by the NPM package

Assume:

- You have a project that depends on multiple libraries.
- Some of those libraries rely on other libraries.

Composer:

- Allows you to declare the libraries on which you depend.
- Find out which versions of which packages can and should be installed, and install them (meaning you download them to your project).
- You can update all your dependencies in a single command.

What form we can update the packages in Composer

```
composer update []
```

This command performs the following steps:

1. **ALWAYS** reads the composer.jsonfile.
2. Searches **Packagist** for the packages specified in that file.
3. Resolves the version to be installed for each package from the indicated versions and stability settings.
4. Resolves all dependencies for those versions.
5. For packages that have a new version available, download and install it by replacing the current version.
6. Once the packages are installed, if there is no composer.lock, create it to leave 'a still photo' of the application runtime environment. If it exists, it updates it. It also creates the application's class autoload files.

You can update all packages found in composer.json
composer update

You can update only one or more packages by separating them by spaces
composer update doctrine/dbal laravel/framework

You can update all packages from a vendor using an asterisk
composer update doctrine/*

We can eliminate dependency
The `remove` command is used to remove any dependencies that we no longer use, as follows:

```
$ php composer remove vendor/package
```

```
What the composer.json file is for
```

It is a json file that helps us to manage the dependencies installed to our project.

Project requirements

- You must perform all the steps using the command line only.

- You need to set up your repository to ignore the following files and directories
  - Directory where composer dependencies are installed
- You must be able to run the Guzzle library and make a small example using the methodology provided by composer.
- You should be clear about the use of composer and its command-line tool.
- You need to be clear about the difference between a development unit and a production unit
- Create a clear and orderly directory structure
- Both the code and the comments must be written in English
- Use the camelCase code style for defining variables and functions
- In the case of using HTML, never use inline styles
- In the case of using different programming languages it always defines the implementation in separate terms
- Remember that it is important to divide tasks into several sub-tasks so that you can associate each particular step of the construction with a specific commit
- You should try as much as possible to make the commits and tasks planned the same
- Delete files that are not used or needed to evaluate the project

Project implementation

LISTA DE TAREAS A REALIZAR

Task	Priority	Hours	Difficulty
Documentation	Normal	1,00	High
Organization	High	1,00	High
Pre-search for information	Normal	1,00	Normal
Repository creation	Low	0,15	Low
Documenting Project	Normal	1,00	Normal

Previous investigation Composer	Normal	1,00	Normal
Installar Composer	High	1,00	High
Main Index.php Structure	Normal	1,00	Normal
Import Library Needed	Normal	1:00	Normal
Primary views Index.hp	Normal	1,00	Normal
Making API requests with a few basic examples	High	1.00	High
CREATION README	Low	0.10	Low
Testing / Correction Errors	High	1,00	High

We plan to finish the project in 10 Hours

In which we give 8 more hours to anticipate incidents in the project

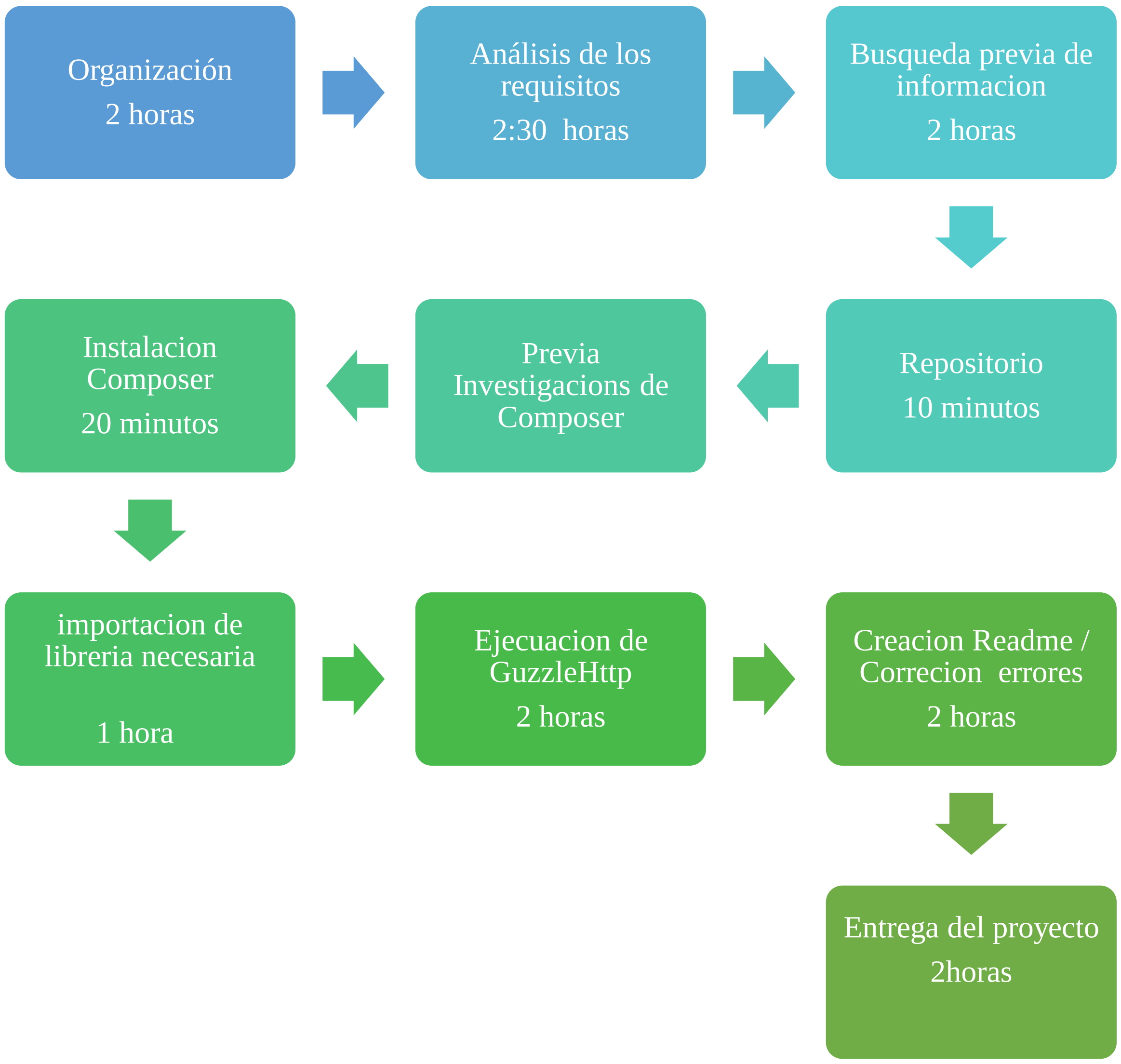
Project specifications

1. You must use Git from the start of the project
2. You must develop the project with **PHP** and **WampServer**
3. We use **Composer** to install GUZZLEHTTP
4. You'll need to use our guzzlehttp to import the necessary components
5. Components
6. All the code must be properly documented
7. All comments included in the code must be written in English
8. Use the camelCase code style
9. It is recommended to divide tasks into several subtasks so that this way you can associate each particular step of the construction with a specific commit
10. A PDF version within the repository is required for project documentation
11. You should try as much as possible to make the commits and tasks planned the same
12. Delete unused files

Incidence Record detected during the project

- None

Project Calendar Tracking


Organización
2 horas
Análisis de los requisitos
2:30 horas
Busqueda previa de informacion
2 horas
Repositorio
10 minutos
Previa Investigacions de Composer
Instalacion Composer
20 minutos
importacion de libreria necesaria
1 hora
Ejecuacion de GuzzleHttp
2 horas
Creacion Readme / Correcion errores
2 horas
Entrega del proyecto
2horas

Quality Metrics

Although the project must adhere to all project *requirements* and *specifications,* there are some conditions that, if properly met, add a sense of quality and robustness to the project itself. These conditions are:

1. Wampserver uses index.php
2. Composer installation
3. The twig code we use Visual Studio Code
4. The twig code must be lint-free.
5. The web application must respond.
6. The web application must be compatible with the main browsers on the market:
    - Internet Explorer 11 o superior.
    - Safari in one of its latest versions.

- Firefox in one of its latest versions.
- Chrome in one of its latest versions.

Risk Documentation

- Loss or damage of work material.

GIT WORKFLOW documentation

- Creating Git Hub https://github.com/robertfox11/composerPHP.git
- We make commits of the structure of the main page.
- Chance of it occurring 80%
- Project impact 60%
- Possible alternative (mitigation) Ask colleagues for help
- Chance of it occurring 30%
- Project impact 60%
- Possible alternative (mitigation) Ask colleagues for help
- Not easily finding information related to the project
- Chance of it occurring 30%
- Project impact 60%
- Alternative alternative (mitigation)
- Ask colleagues for help

From the realization of the structure, work continued only on the "master" branch, through the Workflow "Gitflow".
But information --> https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow

Project tooling

Different tools were used in the development of the project. They are as follows:

1. ***git: A powerful version control system*** that helps track changes in the work tree.

2. ***Visual Studio Code: A code editor*** optimized for creating and debugging modern web applications.
3. ***Wamps Server, to carry out the project***
4. ***Composer to make requests from libraries***
5. ***GuzzleHTTP*** installation ***and required dependencies***
6. ***Google Chrome Developer Tools:*** Used to debug JavaScript code and to test design settings.
7. ***Google Docs:*** Used to write project documentation.
8. ***W3C Validator***– Used to validate HTML and CSS code.

Git workflow

All commits will be inserted into the master branch, following a personal criterion of loading only snapshots that are functional and working correctly, not counting minor errors. There are no other branches, as it would slow down the development process.
On the other hand, confirmation messages end with their primary purpose indicated in square brackets— for example.

- We'll use a base.html.twig template that we'll install with composer
- Plugin Twing

File structure

Project files will be organized as follows:

Record of lessons learned.

- Instalaccion by Composer
- Ejecucion Guzzle HTTP basic