Robert Salazar Arias 11rsahome@gmail.com

PHP UNIT

Overview

The goal is to understand TDD - Methodology Test-driven development

It's a tool that detects errors and won't let you move forward until you correct it

Table of contents

Phase I - Project Initiation 1
- Project requirements 1
- Project Specifications 1

Phase II - Project Planning 2
- Reasoning 2
- Deployments 3
- Task planning 4
- Tracking the project schedule 4
- Git 5
- Tools 5

Phase III - Project Execution 6
- Incidents 6
- Lessons 6

Phase IV - Closure Project **7**
- General Comments 7

Phase I - Project Initiation

Project requirements

- Installing PHPUnit we make use of composer
    - We have to install whether it's a development dependency or not.
    - Checking the new directory called Vendor

Project specifications

- The project must be developed in PHP.
- Create a Git repository
- Don't raise dependence
- The directory structure of the project must be well defined and organized.
- The code must be documented correctly using the English language.
- Your code must use a *camelCase style.*
- If you use HTML do not use inline styles
- The project must not contain unused files.
- The project must be developed using *git*, using explicit and concise deconfirmation messages.
- Delete files that are not necessary to evaluate the project
- The project must contain a *README* file written in *Markdown* that shows a brief description and the steps for runél.

Phase II - Project Planning

Reasoning

This pill will be divided into implementing a practical case on the PHPUnit perform the tests, explaining the presentation how it develops.

We will also carry out test implementation that will take care of testing this new method, a file called UtilTest.php will be created

Running the test, we will run PHPUnit using vendor/bin/phpunit UtilTest.php.

We will check, PHPUnit not find any tests since it is not implemented

Test Implementation

We will update the UtilTest.php file and the method test_converToSlug, import the Util.php class and make use of the converToSlug method

Input parameter for the method:
Hello I'm a new Url

Output parameter that you will compare in your test:
Hello-I'm-a-new-Url

It is important to take into account to implement this test, keep in mind that **PHPUnit** offers various methods oriented to parameter comparison and evaluation.

Within this group of methods, there are *Assertions.* With Assertions, you'll write "Affirmation that the value of X is equal to X". With this syntax your tests will be clear and concise, which will allow you to keep your project in an orderly and scalable way.

Implement new requirements

As usual in a real work environment, changes are constant and we need to verify that our code continues to work as expected. Next, you'll need to modify your test, since the expected result of the created algorithm will be lowercase:

```
hello-I'm-a-new-url
```

When modifying the expected result of your test, if you run **PHPUnit**again, you will have to show yourself an error since the test does not actually meet.

Additional information about **slugs**:

The slug is the part that identifies the url of a given page. For example, the slug in the "Job Creation" section of apple.com is"job-creation". It is important to note that they cannot contain spaces and are recommended to be lowercase. In a real environment, several checks would be required to generate a url other than removing spaces and accents, such as removing anything other than letters or digits, checking UTF8 encoding, removing duplicate hyphens, and verifying that it does not return empty. For this case, simply create a symbolic method that will give you an introduction to the world of tests.

Implement the new functionality

Modify your algorithm to convert the text string to lowercase and therefore the test will meet correctly. Once modified, verify that your test satisfactorily complies with the automated testing process.

In the case of making use of the **TDD**methodology, before implementing the solution of the **convertToSlug** method we should implement the test that would throw us error since this new functionality does not exist in our **Util.php**class. Since this project focuses on you understand how to use **PHPUnit,**you can bypass the methodology and focus on creating a test that covers the functionality of the new method.

Organize the code

In this small project we have focused on testing **PHPUnit,**so we have not taken into account the organization of our code. It is very important that you organize properly. Create the following directories:

- app (will be responsible for containing the source code of your app)
- test (will be responsible for containing the tests)
  - test/app (this directory is created to maintain the same structure as the original app to facilitate the location of the tests of each of the files)

Task planning

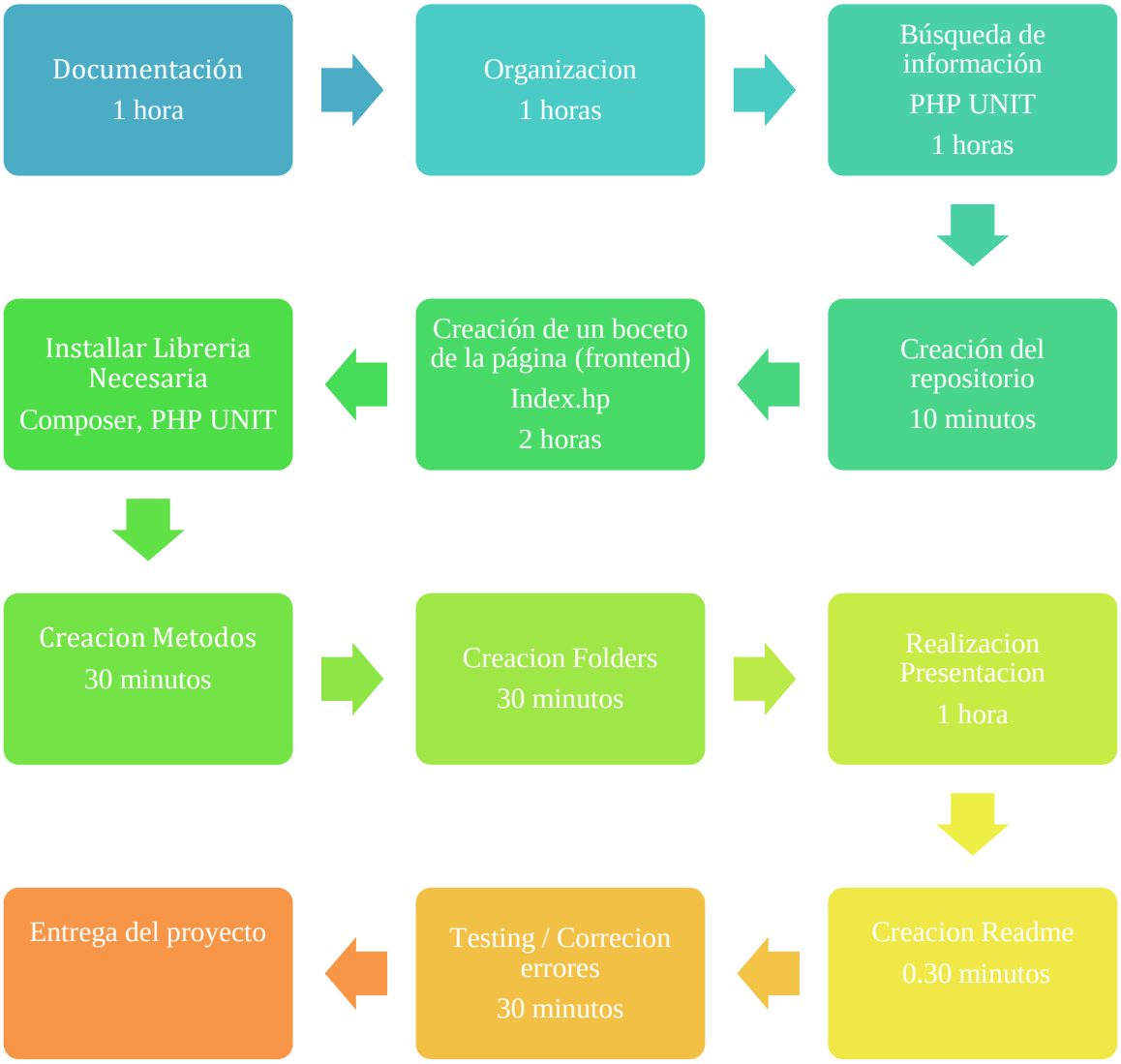
The following set of tasks, each consisting of a numerical hierarchical identifier, a short title, a priority value (from 1 to 5), a difficulty value (from 1 to 5), an estimate of the duration and its dependencies, are an effort to synthesize and streamline the Project development: By dividing the project into a task tree, where the deeper the level the greater the granularity of the task, development efforts can focus on completing one branch after the other successfully completed on time.

LISTA DE TAREAS A REALIZAR

Task	Priority	Hours	Difficulty	ID
Documentation	High	1,00	High	1

Organization	High	1,00	High	2
Pre-search for information	Normal	1,00	Normal	3
Repository creation	Low	0,15	Low	4
Index php structure	Low	0.30	Normal	5
InstallAr Library Required	Normal	0.10	Low	6
Installar PHP UNIT	High	0.10	High	7
Creacion Folder App and Test	Normal	0.30	Normal	8
Metodos Creation	Normal	0.30	Normal	9
Presentation making	High	1.00	High	11
CREATION README	Low	0,30	Low	12
Testing / Correction Errors	High	0,30	Normal	13
Project delivery	High	0.20	High	14

Project Calendar Tracking


Documentación
1 hora
Organizacion
1 horas
Búsqueda de información
PHP UNIT
1 horas
Creación del repositorio
10 minutos
Creación de un boceto de la página (frontend)
Index.hp
2 horas
Installar Libreria Necesaria
Composer, PHP UNIT
Creacion Metodos
30 minutos
Creacion Folders
30 minutos
Realizacion Presentacion
1 hora
Creacion Readme
0.30 minutos
Testing / Correcion errores
30 minutos
Entrega del proyecto

GIT WORKFLOW documentation

- Creating **Https://github.com/robertfox11/PHPUnit.git** Git Hub
- 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

Tools

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

- ***git: A powerful version control system*** that helps track changes in the work tree.
- ***Visual Studio Code: A code editor*** optimized for creating and debugging modern web applications.
- ***WampServer,*** *comes integrated Apache Web server, openSSL for SSL support, MySQL database and PHP language*
- *Composer and PHP UNIT* Library
- ***Google Chrome Developer Tools:*** Used to debug JavaScript code and to test design settings.
- ***Google Docs:*** Used to write project documentation.
- ***W3C Validator***– Used to validate HTML and CSS code.
- ***ESLint***– Used to validate JavaScript code.
- ***nano: A basic text editor that uses the*** command-line interface.
- ***curl: A*** command-line tool used to transfer data using various network protocols.
- ***Google Docs:*** Used to write project documentation.

Phase III - Project execution

Incidents

None, luckily!

Lessons

All tasks were completed without having to face any major obstacles.

Phase IV - Project closure

General comments

The pill was successfully completed in the time interval that was predicted in task *planning.*