

pOSTMAN

PILLS



23 de enero de 2020

ASSEMBLER

Passeig de Joan de Borbó, 101

POSTMAN

Postman is a tool that is used, above all, for API REST testing, although it also supports other features that come out of what is included in the testing of this type of systems.

Thanks to this tool, in addition to testing, consuming and debugging REST APIs, we can monitor them, write automated tests for them, document them, mock them, simulate them, etc.

DIEGO ARTURO SILVA ROJAS

BARCELONA, SPAIN

17 JANUARY 2020

Copyright © 2020, Diego Arturo Silva Rojas. All rights reserved

POSTMAN

Content table

[1. General Analysis. 3](#_Toc30673310)

[1.1 Items pill. 3](#_Toc30673311)

[1.2 Own algorithm. 4](#_Toc30673312)

[1.3 Answer for presentation. 5](#_Toc30673313)

[2. Pill requirements. 7](#_Toc30673314)

[3. Pill Planning. 8](#_Toc30673315)

[3.1 Pill organization. 8](#_Toc30673316)

[3.2 Priority of each task. 8](#_Toc30673317)

[3.3 List Task. 9](#_Toc30673318)

[3.4 Total approximate project estimate. 10](#_Toc30673319)

[3.5 GIT Workflow documentation. 10](#_Toc30673320)

[3.6 Software requirements. 10](#_Toc30673321)

[3.7 Tools used in the project. 10](#_Toc30673322)

[4. Incident management. 11](#_Toc30673323)

[5. Irrigation Management. 11](#_Toc30673324)

[6. Development. 12](#_Toc30673325)

[7. Monitoring Control. 13](#_Toc30673326)

POSTMAN

# General Analysis.

## Install Postman

*The first step is to install Postman to start using it. To do this, follow the steps indicated in the official guide:*

*https://www.getpostman.com/downloads/*

## Analyze the interface

*Next, review the menus provided by the interface and add in the documentation a section explaining what it is for and how each of the following elements is used:*

* *Request.*
* *Collection.*
* *Environment.*
* *API Documentation.*
* *Mock server.*
* *Monitor.*
* *API.*

*It is possible that to have access to any of the above elements you must create a Postman account*

## Put it into practice

*Finally, you must implement the following Postman functionalities using an example online API to make the requests:*

* *Create a new POSTMAN Collection (research previously about it).*
* *Create a request for each of the following methods as shown in the image above:*
* *GET*
* *POST*
* *PUT*
* *DELETE*

*The main idea is that you know how to use each type of request and thus be able to test your own or third-party applications. For example, for the request that uses the GET method, you would call “/ employee” for the POST “/ create”,…*

* *Create a test for each previous request where you will verify that the response meets the following conditions:*
* *The state code is as expected*
* *The name of the state code is the corresponding one (OK, Created, ...)*
* *The answer has body*
* *The answer is in JSON*
* *The header has “Content-Type”*
* *The response time is less than 200ms*
* *Create a monitor that runs the previously created collection every 5 min*
* *Create an API document from the previous collection with the Postman application*
* *Export the created collection and import it back into Postman (you must incorporate the generated file in the project repository.*

*You must document all the actions performed previously and take screenshots of all the processes.*

## Search Information.

* Workflow without PHP UNIT.
* Add new class with functionality.
* Call class.

# Pill requirements.

* *“Create a clear and orderly 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.”*

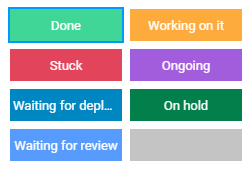
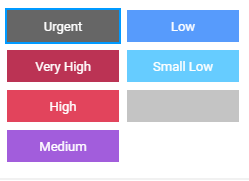
# Pill Planning.

## Pill organization.

* *“Next you will have to create a document where you can explain in detail how the current pill is organized. It is important that it be updated throughout the life of the pill. The document must include at least:*
* *Requirements documentation.*
* *List of tasks to be performed.*
  + *Priority of each task*
  + *Title and description of each of them*
  + *Difficulty level*
  + *Estimated time for each task.*
* *Record of incidents that were detected during the execution of the pill.*
* *Documentation about the GIT WORKFLOW you are going to use*
* *Documentation about the tools used in the project*
* *Record of lessons learned.”*

## Priority of each task.

A total of 5 SPRINTS of the project were defined with each of its tasks, assigning the priority of execution and the state of the task.

8 scenarios with their respective colors were proposed for the status of each task:

* *Done* (Light Green), for task have been finished
* *Working on it* (Orange), tasks are underway and almost finished.
* *Stuck* (Red), tasks are stagnant for some reason and could not be finished.
* *Waiting for review* (Blue), tasks are waiting for a review and approval.
* *Waiting for deployment* (Dark Blue), tasks are pending for status update and repository update.
* *Ongoing* (Purple), tasks are on development.
* *On hold* (Dark Green), tasks are waiting for something resource or search information or depend on another tasks to start.
* (Gray), tasks without assign.

6 scenarios with their respective colors were proposed for assign the tasks priority. The priority upper level is *High* (Red) and *Small Low* (Light Blue) is the priority lower level. For tasks with null scenario (it doesn’t have priority) it will be assigned with *Nothing* (Gris). The *Urgent (Dark Gray)* scenario was established for tasks with extreme and excessive importance.

## List Task.



## Total approximate project estimate.

Using a tool for project management, called monday.com. Tasks are reflected within sprints or stages. Adding the hours each task for each sprint we have that the project will be executed in 21.1 hours in 4 days, it will be 5.5 hours average per day for the project deadline.



The first sprint called General Analysis were divided in two main sub-sprints: Structure analysis and Project Planning.

## GIT Workflow documentation.

The GIT repository is located at: https://github.com/diegosilva91/PHP-Unit.git and was cloned into the computer to create the corresponding static portal files. The master BRANCH was used to carry out all the loads with their respective commit.

## Software requirements.

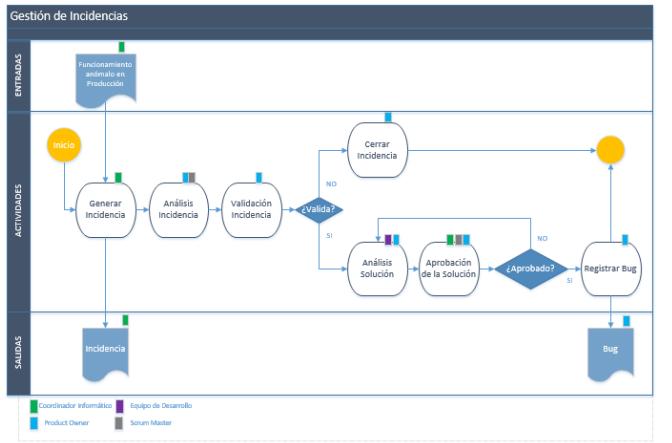
It must be installed for develop the pill:

* PHP.
* Composer.

## Tools used in the project.

Project management was carried out with Monday.com, to plan tasks, sprints, calendar and schedule.

# Incident management.

The procedure to manage the incidents presented in the project is proposed.

The incidents will be recorded in the updates of each task of the support tool for project management used in its development. This section will record the incident detected, if validated, the solution found for this solution and if it was correctly solved. The incidents will be reviewed again in the partial and final meetings of each sprint, to review the pending incidents and generate actions so that they are not recurring.

# Irrigation Management.

A procedure is proposed to manage the possible risks that may arise in the project. Based on the SCRUM methodology, we have to:

*“Project Risk management includes the processes to carry out risk management planning, as well as identification, analysis, response planning. To increase the probability and impact of negative project events. It is important periodically to manage the possible risks that affect the project, both directly and indirectly. Therefore, the risk management process must be carried out prior to scope management, taking advantage of the scope control and risk control reports obtained at the end of each iteration.*

*Once identified, they perform an enumerated analysis of each of them, in order to be able to propose actions in this regard.”*

“Prepare a series of proposals for reaction to risks, which depending on the severity of the risks, or the probability of their occurrence, may be of the following type:

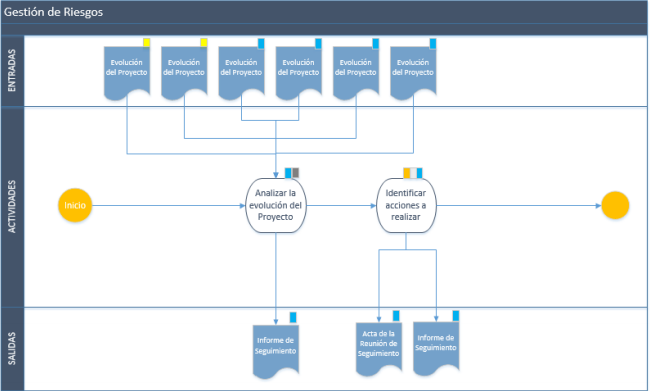
a.) Evasive: To be able to perform some action that allows us to definitely avoid the risk

b.) Mitigating: If you cannot avoid the risk, try to minimize the impact

c.) Transferred: Obtain through any agreement or action that the risk is assumed by third parties

d.) Acceptance: Sometimes it is not possible to perform any of the above actions and the risk must be assumed by absorbing the impact in the best possible way

e.) In the risk response planning, the need to initiate the response plans is evaluated, so that they can be included in the scope management.”

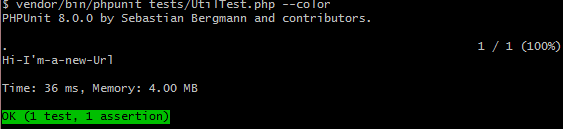


In case of detecting any risk in the execution of any task, they will be recorded in the updates of the same in the project management support tool. The management procedure will be carried out as a high priority. In case of not detecting any risk, the possible risks that may exist in the partial and final reviews of each sprint will be analyzed.

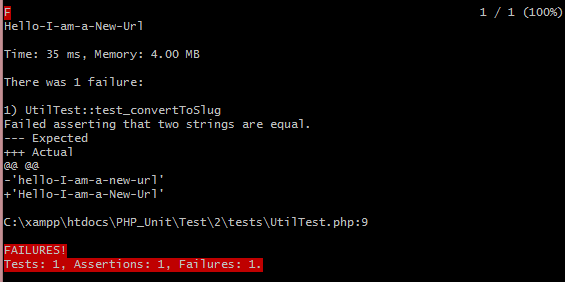
# Development.

The output expected parameter was set up before the run. It was change the order of parameter into the function assertEquals, because it should be written in this order:

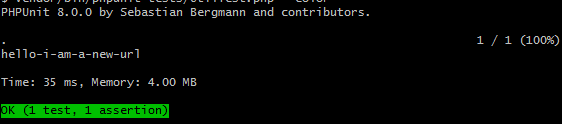
Expected parameter, Actual parameter



The new implement the test, shows an error.



Then the solution was implemented for show the text string to lowercase. The retrospectives of the development lessons learned was observed during the sprint execution, using this methodology based on test.



# Monitoring Control.

The requirements were join into a list for last checking and avoid delivery issues. The table was created for compare each requirement with the development.

