Blog

Project objective

- Develop a custom blog and learn how to work with dynamic content.
- Improve your programming knowledge
- Improve your PHP knowledge
- Improve your knowledge of CSS & HTML & Javascript
- Improve your knowledge of database design and creation
- Improve your knowledge in project management and teamwork

Project requirements

The interface must have the following customer needs:

1. Log in.
2. Logout.
3. List posts.
4. To be able to comment on a post dynamically.
5. Being able to share a post (implement a third-party widget for social sharing)
6. Being able to search for posts by their content
7. Being able to assign keywords to each post
8. Being able to organize posts by category
9.

This interface should only be performed if the user is logovered to administrative party

1. Being able to leave a post in a "draft" state (will not be visible to the user)
2. Being able to indicate the publication date of a post (will not be visible if the date is earlier than the publication date)

You will also develop a control panel to be able to manage the posts that will cover the following Backend needs:

1. Login
2. Logout
3. List posts
4. Edit a post
5. Delete a post
6. Create a post
7. Delete a comment

8. Create Categories
9. Edit Categories
10. Delete Categories
11. List Categories

Project implementation

To carry out this project you will need to consider two main roles.

- The **PHP backend** system that takes care of getting the information from the database
- The **frontend** system that will take care of consuming the data provided by your own backend system.

It is important to note that there are two "Fronts":

- The public blog
- The dashboard that will be used to manage the blog

It is important to consider user roles because an **unregistered** user should not be able to perform the actions of a **succeeded** user.

Project specifications

1. You must use Git from the start of the project
2. For the PHP development part you won't have to use any framework
3. You'll need to use MySql to manage the database.
4. All the code must be properly documented
5. You should include a small user guide to understand how to interact with the tool
6. The interface must be fully **responsive**
7. All comments included in the code must be written in English
8. Use the camelCase code style
9. In the case of using HTML, never use inline styles
10. In the case of using different programming languages it always defines the implementation in separate terms
11. 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
12. A PDF version within the repository is required for project documentation
13. You should try as much as possible to make the commits and tasks planned the same
14. Delete unused files

15. You will need to create a file README.md correctly documented in the project root directory (see guidelines in **Resources**)
16. Try using a PHP Linter to make your code as concise and standard as possible.

LISTA DE TAREAS A REALIZAR

Task	Priority	Hours	Difficulty
Documentation	High	4,00	High
Organization	High	3,00	High
Pre-search for information	Normal	2,00	Normal
Repository creation	Low	0,15	Low
Documenting Project	Normal	4,00	Normal
Html structure index php	Low	0,30	Low
PILsMVC	High	12,00	High
Arquitectura MVC	High	4,00	Normal
Views Maqueta	Low	1:00	Normal
Htacces	High	1,00	High
Index.php Principal	High	2.00	High
Models	High	4.00	High
Controllers	High	4.00	High
CREATION README	Low	2,00	Low
Testing / Correction Errors	High	2,00	High

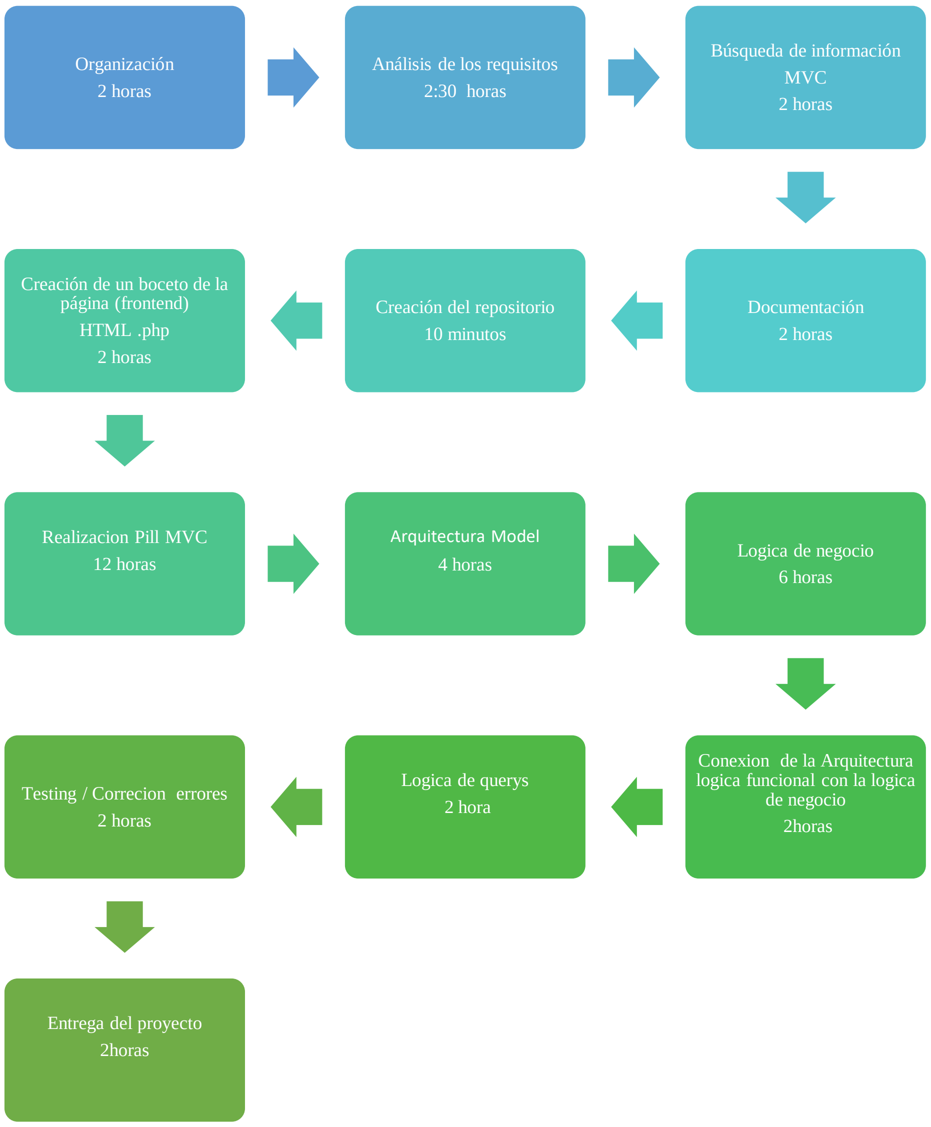
We plan to finish the project in 50 Hours

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

Incidence Record detected during the project

- Petition to database solution there are several requests how to do it
- Controller calling the model solution with a model requiere_once
- Controllador sending data to views correctly request request from views from action with controller plus method
- In route solution htacces fix
- Time

Project Calendar Tracking



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. Lift the WampServer by connecting to mysql and localhost.
2. PHP and CSS code must be validated by W3C.
3. The php code must be lint-free, as indicated by a trusted PHP INTELLIGENCE, such as php INTELLISENSE.
4. The web application must respond.
5. 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

- Project delays.
- Loss or damage of work material.

GIT WORKFLOW documentation

- Creating Git Hub https://github.com/robertfox11/BlogAMedida.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. ***WampServer,*** *comes integrated Apache Web server, openSSL for SSL support, MySQL database and PHP language*
4. ***Google Chrome Developer Tools:*** Used to debug JavaScript code and to test design settings.
5. ***Google Docs:*** Used to write project documentation.
6. ***W3C Validator***– Used to validate HTML and CSS code.
7. ***ESLint***– Used to validate JavaScript 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.

- The main relationships for the functional logic is [class]
- commits related mainly to CSS changes begin with ***[styling]***;
- those related mainly to page layout, ***[layout]***;
- Those related to the project by adding folders or documentation, plugins, library ***[us]***;
- mainly related to documentation, ***[documentation]***.
- For php view we relate it to [index]
- Esctructura mvc

File structure

Project files will be organized as follows:

```
  ProyectoBlogMedida/
      assets/
          img/       Folder containing all the images used in the interface.
          js/        Folder containing all scripts used in the user interface.
          css/       Folder containing all styles used in the user interface.
      Config/config.php
                     Main route configuration
              DataBase.php
                     Request to connect to MySql database
      Controller/
              Drivers to model and views
      Documentation/
                     Project documentation
      Helpers/Classes with Basic Functionalities
                     Models/ Class models for requesting data and functionalities
      Views/
                     Route views for the action of methods coming from controllers
                     /layout
                             Main views of header footer and php aside
      .gitignore     Folder used by  git to contain information about the repository.

      .htaccesfile that allows you to modify the apache engine to create the RewriteEngine
                     ON routing

      BaseDatoBlog

                     We'll use requests

      index.php      Web application home page.

      README.mdFile that contains instructions on how to run the project.
```

Record of lessons learned.

- Making a more detailed documentation improvement
- Class structure
- MVC concepts, to have structured code for a faster client request.
- Request to PHP database with more vision of improvement.
-