Manual Técnico - Recomendaciones para trabajar en el proyecto

Versión: 1.0

Requisitos previos:

Antes de comenzar a trabajar en el proyecto es necesario tener instalado los siguientes requisitos:

* Composer
* Git
* Un servidor web
* Opciones:

Xampp: <https://www.apachefriends.org/es/download.html>

Wampp: <https://sourceforge.net/projects/wampserver>

Apache: <https://httpd.apache.org/docs/2.4/es/install.html#download>

Si se decide instalar Apache directamente, la configuración dependerá del usuario.

* Versión de PHP >= 8.1

Si se descarga el servidor web de Xampp o Wampp, ya viene incluido PHP.

Recomendaciones para subir los cambios a Git:

* Al momento de subir los cambios, realizarlo bajo la rama "dev". Una vez recibido y aprobado el cambio, se implementaría en la rama "dev-qa". Si en esta rama el cambio no presenta problemas, se implementará en "master".
* Las variables de configuración como conexión a la base de datos, usuario, clave, etc., se colocan en el archivo ".env".

Pasos para trabajar con Git:

1. Ubicarse en la carpeta de su repositorio local. Desde la consola cmd o desde el bash de Git, ejecutar "git clone <https://github.com/tridix295/#Proyecto_Uninpahu.git>".
2. Verificar la rama sobre la que se encuentren. Por defecto, al descargar el repositorio se encontrarán bajo la rama "master". Verificar que no se está en la rama "master". Para esto, ejecutar "git branch" y en la consola le mostrará con un \* la rama sobre la que se encuentre.
3. No realizar cambios sobre "master". Para cambiar a la rama "dev", ejecutar "git branch dev".
4. Una vez agregados los cambios, agregarlos con "git add <archivo>" o también puede agregar todos los cambios con "git add". Antes de guardar los cambios, verificar los cambios realizados con "git status".
5. Confirmar los cambios con "git commit -m "Descripción breve del cambio"".
6. Subir los cambios al repositorio con "git push origin dev".
7. Descargar la última versión de la rama con "git pull origin dev".

Nomenclatura de Archivos, carpetas, variables y clases.

* + Si se requiere crear una nueva carpeta por x motivo este debe comenzar con la primera letra en Mayúscula.
  + Al crear un controlador estas deberán ubicarse en la carpeta Controllers/Http… y el nombre del archivo debe empezar en minúscula y finaliza con la palabra Controller

Ejem. userController.php

* + Al crear un modelo este deberá tener el nombre al modulo que pertenece y debe finalizar con la palabra Model

Ejem userModel.php

* + Al crear una vista este deberá estar dentro del módulo que le corresponda, y deberá finalizar con la extensión twig.

Ejem. /User/index.html.twig

* + Al crear variables en lo posible asignile un nombre sencillo y que sea auto exlicativo.

Ejem. $obtenerClave…

Pasos para trabajar con los controladores

* + Al declarar la clase del controlador este si puede tener el prefijo normal (user) y si el controlador que se está creando es el que estará manipulando algún modulo en particular es necesario que este herede al controlador base para que se pueda comunicar con su modelo y respectiva vista.

Ejem.

//Llamamos al cotrolador base

use App\Controllers\BaseController\BaseController;

// Heredamos al controlador.

class User extends BaseController

* + Para poder inicializar al controlador base debemos instanciarlo en el controlador.

Ejem.

    public function \_\_construct()

        parent::\_\_construct('Home');

Una vez realizado ya podremos acceder al modelo y comunicarnos con nuestra vista. Ejem. Caso Modelo

    public function index(){

*$this*->model::get();

Ejem. Caso Vista

public function index(){

*$this*->view->getView("MyVista","Carpeta",$datos);

Pasos para trabajar con los modelos

1. Al crear un modelo al igual que en las vistas este deberá ubicarse en la carpeta correspondiente al modulo (Models/User/userModel.php)
2. Al nombrar la clase el nombre deberá ser el modulo al que pertenece seguido de la palabra Model (userModel) y asignarle el espacio de nombre correspondiente;

Ejem.

namespace App\Models\User;

1. Para poder implementar el ORM de laravel debemos heredar la clase modelo en nuestro modelo. Ejem.

use Illuminate\Database\Eloquent\Model;

class UserModel extends Model