**Git**

**Observación:** Aprender el funcionamiento y características de Git.

Es como un diario en el cual se registra toda la vida de un proyecto para esto se usa un SISTEMA DE CONTROL DE VERSIONES (VCS) distribuido, nos permite trabajar en equipo para que cada desarrollador pueda aportar al proyecto cada nuevo desarrollo o cambio en el.

El creador es el mismo que creo el Kernel de Linux Linux Torball.

Es el sistema más usado en el mundo, es muy rápido ya que cada desarrollador tiene en su máquina un clon del proyecto, tiene integridad en datos.

**Terminos:**

**Git:** Es un sistema de control de versiones, es decir un software que permite registrar todo el historial de cambios de un proyecto.

**Repositorio:** Es todo proyecto que está siendo seguido por Git, el cual ya tiene un historial y se están registrando sus cambios.

**Commit:** Es cada uno de los cambios registrados en el historial de Git, son versiones.

**Ramas o Brash:** Son ramificaciones, nuevos caminos que toma el proyecto, es como una copia separada del proyecto, pero aislada esto para no dañar el proyecto en sí.

**Clon:** Copia exacta del repositorio, es lo primero que se debe hacer por cada desarrollador, clonar el repositorio en cada computadora para trabajar de manera local.

**Fork:** Es un proyecto completamente diferente que se crea a partir de otro proyecto ya existente, a diferencia de la rama, se basa en otro proyecto.

**GitHub**

Es un control de versiones en la nube

Usuario: CMTZ10 uiscloud2021 base datos

email: [jcmtzesp@gmail.com](mailto:jcmtzesp@gmail.com) sistemas@uis.com.mx

Contraseña: SystemsFC10 UISChihuahua2021

AMAZON: [julia.miranda@uis.com.mx](mailto:julia.miranda@uis.com.mx) UnidadCHH2021

**Paquetes necesarios para instalación y funcionamiento**

* Xampp
* Visual Studio Core
* Composer
* Laravel
* Nodejs
* GitHub

**Pasos para instalar un AdminLTE en Laravel**

Una vez instalados todos los programas necesarios seguir los siguientes pasos:

1. Abrir la consola de comandos (CMD) ir a la ruta de la carpeta Xampp – Htdocs
2. Una vez dentro de la carpeta en CMD crear un nuevo proyecto y seguir las siguientes líneas:

laravel new “Nombre del proyecto” --jet (línea para crear un proyecto con login jetstream)

php artisan migrate (línea para migrar las tables de sesión una vez creada la base de datos manual)

php artisan serve (línea para ejecutar el servidor localhost)

composer require jeroennoten/laravel-adminlte (línea para instalar componentes laravel adminlte)

php artisan adminlte:install (línea para instalar adminlte)

instalar componentes npm que indica jetstream para el login

1. Las siguientes líneas son para cambiar en el php de dashboard creado por default por jetstream

@extends('adminlte::page')

@section('title', 'Dashboard')

@section('content\_header')

<h1>Dashboard</h1>

@stop

@section('content')

<p>Welcome to this beautiful admin panel.</p>

@stop

@section('css')

<link rel="stylesheet" href="/css/admin\_custom.css">

@stop

@section('js')

<script> console.log('Hi!'); </script>

@stop

Utilizar para variables pascal case; ver laravel route resource

App- modelos, representación grafica de las tablas, en tablas plural y minúsculas modelos singular y la primera mayúscula

Protected $table ‘parts’ dentro del modelo hace referencia a jalar los datos de esa tabla dentro del modelo en singular Part

Class Part extends model{ protected $table ‘parts’; }

http – controllers ; enlazar los controladores a la vista se divide en 4 partes (index, créate, store, show, edit y update destroy)

index: hace un elocuent consulta a la table del modelo

créate : es para crear uno nuevo y esta relacionado con store

edit: es para editar desde la lista y esta enlazado con update

destroy: es para eliminar el id pero borrado lógico no físico cambia la actividad de 1 a 0 es decir no visto pero si en base de datos

Resources – view aquí se encuentran todas las vistas de donde se llaman los modelos y controladores, nombre de las carpetas con minúsculas y los nombres de las vistas con minúsculas

Para utilizar php hay que hacer 2 llaves que abren y 2 que cierran {{}} para un condicional foreach o if utilizar @

@yield @section para enviar y recibir de un base a otro php

Routes: archivo web.php siempre es ese archivo enlazar las vistas con los controladores: Route::resource(‘nombrevista’, ‘nombrecontrolador’); enviarla desde botón etc {{route(nombrevista:créate o edit etc)}}//nombre del controldaor lo que se va hacer

**Codigos para crear**

**Modelo:** php artisan make:model Nombremodelosingular –m //-m es para crear la migración la tabla de ese modelo en terminal.

**Controlador:** php artisan make:controller NombrecontroladorsingularController –r //Es para tener ya todos los métodos formados para el crud como créate destroy edit etc en terminal.

**Rutas:** Route::resource('Nombremodeloplural', 'App\Http\Controllers\Nombrecontrolador');//ruta para el controlador esto va en carpeta routes – web.php

**Migraciones:** php artisan migrate //Aquí se ponen los campos de la tabla creada desde el modelo para migrarla después con el código en terminal

**Vistas:** Estas se crean en Resouce – views creamos una carpeta con el nombre en singular, dentro de esta irán todas las vistas es decir el index, créate etc. Todas con el nombre.blade.php

**Layouts:** Aquí creamos una plantilla base de bootstrap para ello nos vamos a la pagina bootsrap y getstarted y copiamos la plantilla base y la pegamos en el nuevo layouts que hemos creado llamad plantillabase.blade.php

**Mostrar el contenido en ventana**

En plantilla base agregar un <div class="container"> @yield('contenido') </div> //esto para mostrar el contenido de nuestro index

En index.blade.php de views agregar

@extends('layouts.plantillabase')

@section('contenido')

<h1>Vista Index</h1>

@endsection

En el controlador agregar la ruta en index return view('file.index');

También en el controlador agregar la ruta del modelo que se usara con ese controlador use App\Models\Nombredelmodelo

Para agregar todo nuestras ventanas dentro del dashboard de adminlte es necesario que en cada ventana view agregar este código: <https://github.com/jeroennoten/Laravel-AdminLTE/wiki/usage> para que quede dentro del tema adminlte

Para crear la carpeta vendors en vistas codigo: php artisan adminlte:install --only=main\_views es para crear todas las plantillas de adminlte y poder personalizarlas

Para poder modificar adminlte mas fácil es directamente desde config – adminlte aquí se configura todo lo del admin desde inicio de sesión hasta navbar sidebar, colores etc.

Para cambiar el login de jetstream por el de adminlte es necesario ir a resources – views – auth – login.blade.php y cambiar todo el código por este: @extends('adminlte::auth.login').

Asi es lo mismo con registro etc. Visitar pagina de códigos: <https://github.com/jeroennoten/Laravel-AdminLTE/wiki/Artisan-Console-Commands#authentication-views>

Para proteger las vistas para solo logeados es necesario crear un constructor dentro del controlador de cada vista dentro de la clase controller poner el constructor:

public function \_\_construct(){

$this->middleware('auth');

}

**Crear cuenta y enlace con S3 AWS**

Para crear la cuenta nos dirigimos a amazon aws creamos la cuenta mediante un registro y una tarjeta de crédito o débito se descontará 1 dólar por el registro.

Después vemos el siguiente video para enlazar la cuenta con nuestro sistema de laravel: <https://www.youtube.com/watch?v=gTZt8igdpnA>.

De aquí en delante se utilizaron los videos de Coders Free para realizar las demás actividades de subir archivos, formularios con laravel collective etc.

Despues se instalo el laravel permission para realizar los permisos de los usuarios, en el siguiente link se podrá ver la documentación de la librería: <https://spatie.be/docs/laravel-permission/v4/installation-laravel>

**Codigos consola git cmd para subir cambios a github**

1) Al modificar un/unos archivo/s:

-git add . //Esto añade todos los archivos al repositorio local (no al web)

o sino tambien

-git add nombreDelArchivo //Esto añade solamente el archivo indicado al repositorio local.

2) Luego... se genera un commit para confirmar todos los archivos modificados agregando un mensaje que brinde una idea de lo que se modificó en el/los dichos archivo/s

-git commit -m "aqui el mensaje"

3) Para enviar finalmente los archivos al repositorio online se debe tipear:

-git push origin master

Apendice A) Si se quiere ir verificando el estado de la transaccion luego de cada instruccion, puede tipear el siguiente comando:

-git status

Video de ejemplo para subir a github:

<https://www.youtube.com/watch?v=H03E8_HImAU>

configurar aws

<https://www.youtube.com/watch?v=4SBFz4jG7KU>

base de datos UISAdmin – contraseña: uiscloud2021

Para cambiar los archivos en gitbash primero descartamos cambios locales si los hay dentro del directorio uiscloud

git reset –hard

después damos

git pull origin master