// PROGRAMACION ORTIENTADA A OBJETOS

// vINO A SOLUCIONAR PROBLEMAS DE CODIGO MUY MUY LARGO

en donde si algo fallaba ,todo se rompia. Ademas de que es muy dificil de mantener en donde si hay un plazo de proyecto muy largo, se dificulta el proceso

Codigo espagueti: Codigo muy expandido en donde una línea puede ir a otra, y es un desorden total.

Esto era antes, ahora lo ideal es simplificar la programación, por lo cual el sistema Orientacion a objetos, surge a partir de los problemas que tenemos y necesitamos plasmar el código, son soluciones orientadas a la realidad.

POO observa los problemas en forma de objetos, lo cual facilita mas.

POO es un paradigma, es decir, es la TEORIA que suministra la base y modelo para resolver problemas, en donde nos brindan un ‘molde’ para después resolver el problema con unas bases, estas bases serian el paradigma. Pero no solo a POO sino en general en programación

PARADIGMA en POO:

Siempre

Ejm: Javascript no es por naturaleza un POO sino usa ‘prototipado’ para usar el POO

4 oObjetos:

1. Clases:
2. Propiedades:
3. Metodos:
4. Objetos:

Encapsulamiento

Abstraccion: Encapsulamiento

PHP: Es POO, pero podemos usarlo como estructurados ya que antes era asi. Dese la versión 5 esta orientada a objetos.

JAVA: POO naturalmente, es usado mas que todo en Android, no esta para hacer web, pero no significa que no se pueda hacer, sino que no esta preparado.

PHP: Es un lenguaje interpretado, esta pensado para aplicaciones web.

PYTHON: Diseñado para ser fácil de usar, multiples usos: en web, server side, análisis de datos, machine learning, etc.

JAVASCRIPT: Lenguaje interpretado, Orientada a objetos pero basado en prototipos, además de estar pensado para web.

JAVASCRIPT no tiene nada que ver con JAVA.

DIAGRAMAS DE MODELADO ORIENTADA A OBJETOS

Diagramas que nos permiten mirar mas claramente las diversas fases de un proyecto, en dnode te permiten tener un refuerzo. Tiene

OMT: Es mas antiguo,

UML: Se hace mas actualmente

1. OBJETOS: Podemos tener dos tipos:
   1. Fisicos
   2. Conceptual: Al mas abstracto dentro una solución, no algo tangente por si mismo, es como el concepto que reodea a lo físico, ejm, carro(frisico( y concepto( carretera)

Ejm: Usuario: Fisico

SEccionLOGIN: Conceptual: No podems tocarlo físicamente

PROPIEDADES también pueden llamarse atributos serán sustantivos: ejm: nombre, tamaño, peso, etc.

Comportamientos: Seran las operaciones del objeto, es decir, aquelos procesos y acciones que realizara el objeto, se hacen comúnmente con fuciones, en done le indicamos que haga un comportamiento.

Ejm: Perro:

Comportamientos: saltar, comer, correr, etc.

Proepiedad: nombre, altura, raza, color.

Ejm2: Adopcion:

Comportamiento: ser adoptado

Propiedades: Color, edad, nombre, raza, altura, id.

CLASE: Es el modelo por el cual se contruiran nuestros objetos, se llaman también instancias. Al final es un molde, en donde vamos a crear objetos con hasta solo 1 clase, los objetos provienen de las ‘clases’.

ANALIZAR objeto para crear una CLASE: ABSTRACCION: Pensar, y analizar un objeto basándome en las diferencias, características, propiedad, comportamientos para crear una clase que sea un ‘modelo’ para poder tenerlo de referencia si queremos crear mas objetos.

Ejm: Clase: Targetas para promocionar empresa

CLASE: Un molde con un modelo ‘predeterminafo’ de una targeta para conseguir hacer varias tarjetas.

OBJETOS: las tarjetas con sus características y propiedad( partes de la clase, que le dan sentido de esa clase) ya creadas que fueron hechas a parte del modelo de la tarjeta.

MODULARIDAD: Es lo mas importante DESDE la creación estructuras hasta la POO, nos salva muchas veces, se trata de la divison de pequeñas partes y estas serán nuestros modulos pudiendo funcionar independientemente.

En donde:

DISEÑO MODULAR:

Estructura: cuadro rojo

MODULAR POO: Cuadro rojo pero dentro de el tiene ya varios minicuadrados que ocupan todo su tamaño y que funcionan independientemente.

En donde podemos reutlizarlo, evitar colapsos, mantenibilidad, legibilidad(mucho mas fácil), resolución rápida de problemas.

Aquí viene la CLASE otra vez, en donde CLASE tiene modularidad, es decir. Divide el programa en diferentes partes con archivos también separados.

DIAGRAMACION UML en POO

|  |
| --- |
| NOMBRE CLASE |
| PROPieDAD |
| COMPRTAMIENTO |

Com vemos dividmos la clase.

PHP( débil tipado)

JAVA(tipado estructurado)

PYTHON(tipado interactivo)

Para crear una clase en los diversos lenguajes siempre indicaremos el class y su función.

EJM en PHP

Class person( nombre class) {

$name = ‘’;(propiedad)

Function walk() {}(Comportamiento)

}

Todo esto lo tenemos modular, es decir, lo tenemos todo dentro de un ‘contenedor’ con todo. Y si queremos hacer un objeto esta clase serviría como ‘molde’ para crear mas objetos.

Cuando compartamos clases y compartan propiedades, usar un id en ambos.

HERENCIAS: ‘don’t repeat yourself’ la idea es tratar de no repetir líneas, no significa un resumen, sino ‘optimizar’ o no tener repetir clases, propiedades, etc.

REUTILIZACION: POO Nos ayuda a reutilizar el código con las:

HERENCIAS: Creamos nuevas clases a partir de otras. Se establece una relación PADRE e HIJO en donde se intente mostrar que:

PADRE: SUPERCLASE -🡪 HIJOS: SUBCLASE. En donded subclase herede propiedades o comportamiento de la superclase.

Ejm: EquipoFutbol(PADRE)(Contiene las propiedades o comportamientos que se repiten en las sub clases y las agrupa en el SUPERCLASE, para después las clases como jugador, entrenador, masajista hereden esas propiedades y comportamientos que todos se repiten entre si, y solo seria poner en SUBCLASE(donde se ponen las porpiedades o comportamientos únicos y diferentes entre si.). Es decir, SUPERCLASE seria como el encabezado donde están las porpiedades o comportamientos que las SUBCLASES comparten.

TIP: SIEMPRE AY QUE PONER UN ‘ID’ EN LAS CLASES.

Las clase se dividen en tres secciones: nombre de la clase, porpiedades de clase y comportamientos de la clase.

FLECHA BLANCA EN LA REPRESENTACION: Significa que algo se introduce en otra cosa. En el caso de las clases ayuda a representarlo mejor.

TIP IMPORTANTE: Cuando pensemos en las propiedades y comportamientos en las clases, es muy importante que pongamos aquellas que nos servirá en un futuro para la aplicación. Aquellas con RELEVANCIA.

METODO: Es una uncion que hay dentro de una clase. METODO NO ES IGUAL A UNA FUNCION NORMAL. El METODO es una función( comportamiento convertida en método con public function) dentro una clase.

HERENCIAS(exnteds):

Usar una clase dentro de otra clase, comúnmente es padre-clase e hijo.clase, el cual el hijo-clase hereda el contenido del padre-hijo, para hacer extends el padre-hijo debe tener al menos dos hijos-clase…

El es hijo-clase el llama la información padre-clase.

Esto nos beneficia para no repetir código.

Class Hijoclase extends PadreClase.

POLIMORFISMO:

Es junto a las herencias, solo se hacen cuando se hacer HERENCIAS. Se trata de además de usar métodos del clase-padre en el clase-hijo, estos métodos se pueden modificar del padre-clase que están dentro del hijo-clase. Es decir, un padre-clase puede modicarse en varios hijos-clases para adpaptarse a las exigencias o necesidades dl hijo-clase, por eso con el POLIMORFISMO podemos modificarlo. Solo se puede hacer es un hijo-clase coja dos herencias.

TIP: Tratemos de llamar las clases igual a su archivo.

PARENT: Nos permite utilizar contenido del constructor padre-clase en el constructor de hijo-clase. Es de contrcutor-padre a constructor-hijo.

Parent::\_\_construct();

Si no existiera el ‘extends’ nos tocaria hacer cada vez una nueva clase con propiedades parecidas.

NOTA: No se pueden poner dos constyructores dentro de una clase. Ademas cuando clase-hijo hereda el constructor del padre-clase, y si creamos un nuevo constructor dentro de clase-hijo este se ‘sobrepone’ o anula el constructor delñ padre-clase. Pero aunque pase esto los datos del constructor del padre-clase no se pierden, por lo cual para usar estos datos dentro del nuevo constructor del hijo-clase usamos PARENT.

TIP: Cuadno creamos el constructor solo le estamos pidiendo unas porpiedades que siempre vamos a necesitar, por eso no se ponen todas las propiedades dentro del constructor porque puede que no necesitemos todos. Solo unas pocas propiedades que siempre vamos a necesitar.

TIP: Tambien es importante poner el required cuando hagamos las clases-hijos en sus archivos, ya que el clase-padre esta en un archivo independiente.

INTERFACE

Son Contratos donde te ‘obligan’ a utilizar un sistema, donde las clases han de cumplir. Contitnene métodos vacion que obligan a las clase a utilizarlo

Estandar de Desarrollo:

INICIAR SECCION: Si no tuviéramos secciones nos tuviéramos que desbloguear en cada pagina. Las secciones serian también como un header

Mayo 29/2019.

Como se guarda información en un archivo.

Junio 2/2019

COMPOSER. En POO

PHP-FIG: e creo el PSR que sonrecomendaciones y estándares del cpdigo de PHP. Dentro tenemos los PSR, que son reglas o ‘ideas’ de comòrtamientos de PHP en un entorno colaboracional,

Ir a pagina para leer y ver recomendaciones de reglas del código de php.

Pagina donde nos crea una directrices

COMPOSER, es básicamente un gestor de dependencias, nos ayuda a introducir esas dependencias, librerías, framewokrs de teceros que vienen a nuestro proyecto, nos ahorra tiempo. Es importante porque nos ahorrara fases, y estar alerta continuamente. ES obligatorio usarlo si querems que el proyecto se use a largo plazo. Nos permite gestionar, declarar, descargar y mantener actualizado el software del PHP. Nos ayuda para resumir las tareas de dercarga y mantenimiento de pagina web.

Nomenclatura. Cuando estamos hablando el PATH, nos dicen que tenemos ciertas versiones que nosrmalemnte estánarreglando problemas de laversiona anterior, los cambios que se harán no influirán demasiado, simplemente errores que se iran solucionando. Por lo cual PATH nos soluciona pequeños errores y problemas.

VERION MENOR: Esta muy relacionado con el PATH, aporta y suma métodos nuevos, funciones clases nuevas. Todo para implementear una versión mejorada. Por cual nos actualizan. Cuando cambiar de versionmenor, el PATH reinica a 0.

VARSION MAYOR: Aquí ya a diferencia de las anteriores, aquí nos pueden cambiar demasiado. Por lo cual hasta que n tengamos los cambios leidos no vamos a actualizar. Siempre que cambia esta versión, hay que leer los cambios antes de actualizar.

PAth: nunca nos va aafectar respecto a cambios,

VERSION MENOR: Nos da mejoras, nos están solucionadno problemas y actualizando, pero

VERSION MAYOR: Ya hay un cambio drástico, por lo cual hay que saber que cambios importantes sucedieron.

Nomenclatura de COMPOSER:

Si yo quiero pedirle a composer la versión 4.3.19. No hay problema, nos la dara, pero si

Si lo hacemos en el path, le estamos indicando que revise cual PATH usar pero que no cambie nada en la VERSION MENOR NI MAYOR,

Conservamos la 4.3 pero no me saltes a la 4.

ACCEDER ACMDER.

Dependecias: Son librería, framework, cosas de terceros, donde muchas veces si queremos usar una librería, esta esta dependida de otras por lo cual nos perjudics porque nos obliga a descargalar las otras, por lo cual COMPOSER nos lo soluciona.

COMPOSER JSN: CONfigurador de compososer, el que nos va ahacer las peticiones. Donde podemos llmar las versionar, pero va a quedar marcado en el composer.log

Php composer.phar install, nos instala el composer y nos genera una carpeta llamada vendor, donde se establecerán las librerías que traigamos con COmposer.

COMPOSER LOG: Es com un candando en donde si establezcan las versiones que el proyecto va a usar, auqneu se puede actualizar obviamente pero que nos frene de que otra persona cuando use nuestro proyecto lo actualice, lo cual perjudica el proyecto porque no es esa versión. Lo que hace es quedarse con lo que el proyecto esta instalando, si pasa a otra persona

Esta debería tener por que el log lo obliga a que tenga la misma versión que nosotros.

CREACION O INICIACION COMPOSER

Descargar composer.setup.exe en getcomposer.org.

Abrir terminal winows cmd, ir a la carpeta de PRUEBA\_COMPOSER o donde tengamos el proyecto con el composer.

Package name: lo dejamos igual, intro.

Autores. Colocar nombre y gmail

Stability,intro no colocar nada.

Tipo de Paquete: Proyecto.

En vendor tenemos la aprte del almacenamiento.

Preguntas sobre dependencias: poner ‘n’, como no porque no las vamos a usar todavía, mas adelante.

Y ponemor en el json, ‘y’ para generar el json.

Instalar VENDOR composer

Poner después composer install

Dentro de VENDOr y la caprte composer tenemos unas opciones.

AUTOLOAD: que se autocarge en otro archivos, nos ahorra trabajo, porque lo que hace es agrupar una carpeta donde tengamos muchos archivos y los necestiemos llamar a un solo archivo como hacemos con require, el Autoload nos resume el hacer require por cada archivo, ya solo llamando el autoload nos autocarga todo los archivos en un solo, en donde queramos. El autoload ya es OBLIGATORIO…

Gracias a la integración de composer, autoloads nos permite cargar archivos com un require, el vendor ya tiene autoloads, porque sin bajar ningún tipo de librería ya nos genera un autoload, EL Autoload nos permite mejor accesibiliad en erelacionar archivos externos que require.

Ahora nos vamos a la arhivo de JSON, en donde vamos a poner antes de require, “autoload”, en donde dentro de ella ponermos psr-4. Y dentro de psr-4 ponemos el namespace de models, para que este autoload lo busque y busque los archovs de esa carpeta. Es importante que los archivos de la carpeta models tenga el namespce con el nombre de la carpeta que estamos llamando en el psr-4 en json.

Psr-4: directrices que están php.

En JSON el require es para llamr a las librerías externas.

Al final lo que hacemos es que el VEDNOR es algo interno, y al usar el autoload en donde la usamos para llamar a una carpeta, esta carpeta podría ser una librería por lo cual con el autoload yla estamos metiendo dentro del vENDOR para usarla.

Estructura de autoload

"autoload": {

// tipo de autoload

"psr-4": {

// llamamos al namespace

"models\\": "models/"

}

},

En cmd de nuevo si ponemos composer require nos buscara los tipo de composer, nos mostrara un buscador, colocamos composer para buscar ‘composer’, nos preguntara cual usar, después nos preguntara que versión usar, la podemos poner con ^colocarversion, nos mostrara el contenido de ese composer y podemos seleccionar cual carpeta usar.

Cuando escogamos la versión, tengamos en cuenta la teoría, en donde las versiones mas actualizadas sond ependientes de las anteriores versiones para funcionar, por lo cual anteriormente se hacia manualmente, ahora el composer nos lo hace por el mismo.

Si le daoms intro y no especificamos la verison nos dara la ultima version

^:Significa que nos va adescargar todas las versiones anterioes para usar la nueva versión..

R: en buscador permite mostrar todas las librerías después de composer require

Composer Update: nos sirve para ctualizar librerías, sino especificamos la librería nos actualizaara todas, IMPORTANTE: sino tenemos en el JSON en require, la ultima versión colocada usamos el update, la cagaremos porque no va a coger la ultima versión.

\*: Si lo ponemos en el PATH, analizara o usara todos lo paths pero no tocara la versión menor ni mayor.

Ejm: 1.4.\*

>= o <= : Aquí le indicamos si usar esa versión una mayor o menor. Ejm:

>=1.4.3

Los json, lock y el composer.phar tienen que ir afuera con index.

En Windows el comando es solo compase (compose). Mientras que en mac es php compose.phar (comando que usaremos).

En JSON la ultima línea de código no tiene ‘,’.

RECUERDA: los archivos los cuales vamos a llamar mediante AUTOLOAd deben tener NAMESPACES con el mismo nombre que le pusimos en el archivo de JSON.

IMPORTANTE: TENEMOS QUE DARLE DOS VECES COMPOSER INSTALL, PORQUE A LA PRIMERA VEZ TE LO CREA, Y A LA SEGUNDa cuando ya tengamos las carpetas con el namespace en cada una de los archivos y con el array en JSON. LE volvemos dar este comando. Para que en el archivo del psr-4 te aparezca en el array de este, el nombre del NAMESPACE.

El composer update, nos sirve para actualizar una librería concreta..

Si en cambio usamos composer install, de nuevo es como si se reinicara todo.

Composer va a tener todos los elementos que nos van ayudar en el proyecto

Composer init: iniciar el composer con json

Composer install: nos brinda el vendor

Composer require: nos permite agregar mas carpetas

Composer delete: nos permite eliminar una libreria

GIT CON COMPOSER.

Cuando queremos hacer git con composer, es importante saber que vendor ya viene con un gitignore propio, porque lo que pasa es que el vendor no se sube en el git, sino aparece gitignore lo creamos y metemos el vendor. Esto es para evitar que cuando subamos los archivos al git, la persona que quiera descargar el proyecto y que obvaimente no va a tener el vendor porque esta en el gitignore, por lo cual la persona va a tener que descargar el composer de la pgina oficial y cuando cree su propio vendor, el json establezca la versión ideal, porque si la persona tiene una versión diferente de la establecida( que en muchos casos es la mas actualizada obviamente), entonces puede que le falle porque no es la mima versión del vendor del proyecto original que se había puesto. Por lo cual lo ideal es que ambos tengan la misma versión.

Despues lo hacemos común y corriente, git add . y git commit.

Ya cuando tengamos el primero, lo siguiente que haríamos es hacer un commit y git add para el index solamente. La declaración, osea lo que escribamos en el commit, ponemos lo que hemos hecho. y también cuando halla cambios en solo unos cuantos archivos podemos hacer un commit especifico de ellos común y corriente, es decir, hacer un commit en un solo archivo o varios cuando ha habido cambios, no siempre hacer commit de todos los archivos

GIT TIP GENERAL: Cuando tengamos una base elaborada de un archivo y vamos a hacer un commit, lo ideal también es que se hagan commit por cada archivo o grupos de archivos ‘base’ del proyecto. Esto es para que se guardan por individual los archivos base que nos pueden ayudar o para mantenerlos mas adelante, en vez de subir cada vez que hagamos un commit, subir todos los archivos. No nos beneficia.

Srcset detecta cual imagen debe ir a cual dispositivo.

Media: Analiza la acantidad de pixeles que queremos usar, 1829x769 d resolución ejm.

Como exportar una imagen o diseño echo en programa. Seleccionamos la imagen la exportamos y en las opciones leindicamos la resolución y la exportamos en las carpetas debidas a cada dispositivo.

<picture> permite poner una imagen pero es mas actual, permite poner mediante srcset y media poner la imagen ajustaa y bien según el dispositivo.

TIP PHOTOSHOP: Poner en el nombre las caracterisitcas de resolución de la imagen.

Si pones color picket en google puede escoger el color y coger el código.

IMPORTANTE: HAY QUE TENER CUIDADO SI NO TENEMOS UNA CONPAGINACION O RELACION COHERENTE CON LAS VERSIONES DEL XAMPP O EL HOUST QUE ESTEMOS USANDO Y EL PHP, PUEDE CREAR CONFFLICTOS. POR LO CUAL ASEGURARSE DE QUE LAS VERSIONES EAN AL MENOS CONPAGINADAS O LAS MAS ACTUALES…

SIstremas de Envio electrónico:

SI TENEMOS dispositivos 5 dispositivos y borramos un correo enun, se eliminan de todos los correos

SSL/TLS

FTP conexiones

SMTP:

IMAP

EMAIL LIBRERÍA:

GIT PARA VERSIONAR

git TAG: PERMITE COLOCAR UNA ETIQUETA EN EL COMMIT QUE ESTAMOS USANDO. LO QUE PERMITE COLOCAR UNA ETIQUETA PARA ESE COMMIT, ENTONCES, SI HACEMOS UN COMMIT DESPUES DE HACER ESE COMMIT. PONEMOS Y CREAMOS EL VERSIONADO:

git tag -a v0.1 -m 'my version 0.1'

git tag -a v0.1 -m 'my version 0.1'

DONDE:

-a: Permite crear la etiqueta Y ponerle el nombre de la etiqueta

-m: permite poner un comentario a la etiqueta

-l: permite buscar una etiqueta especifica y seleccionarla. Si queremos buscar por ejemplo todos los path de una versión podríamos hacer:

Ejm:

Git tag -l “v0.1.\*”. Como vemos el path esta en \*. Aquí le estamos indicando nos busque todos los path que hallan en esa versión menor.

GIT RESET HEAD: Permite sacar algún archivo del git add. Esto nos sirve mucho pir si queremosa quitar algún archivo que no queremos meter en git add ni hacerle un commit.

Ejm:

Git reset head NombreArchivo

Cuando hagamos un git log para que nos muestre los commits, si nos muestra HEAD-> master. En un commit, ese commit es donde estamos actualmente.

IMPORTANTE:

= TE IGUALA A UN VALOR QUE LE DEMOS, MIENTRAS QUE:

== TE IGUALA EL VALOR QUE LE PONGAS CON EL VALOR DE LA VARIABLE PREVIAMENTE.

EJM:

$A = $B. $A ES IGUAL $B, NO $A = $A.

EN CAMBIO:

$B = $A

$A = $B. ES DECIR AMBOS VALORES DE AMBAS VARIABLES SON IGUALES ENTRE SI, ES COMO DECIR: $A = $A.

TXT TIP: NO HAY NECESIDAD DE PONER FOPEN CUANDO USEMOS FILE(), PARA LOS ARRAYS, porque este file(), ya te lo abre. Ademas de que si vamos a introducir información con FWRITE o FPUTS necesitamos FOPEN para introducirlo mientras que si no ponemos FOPEN es porque simplemente queremos visualizar la información mas no introducir.

En cmd si ponemos php –v. Podemos ver cual es la versión de php.

DUDAS BUCLES:

DIFERENCIAS ENTRE BREAK, CONTINUE y RETURN:

BREAK: Si la condición se cumple te rompe todos los bucles que estén con el condicional. Tanto si solo hay un bucle como si hay otro bucle.

RETURN: TE retorna el valor y te rompe solo el bucle en el que estaba el condicional que se cumplio, mas no los bucles (si es que los tienes) que estén afuera.

CONTINUE: Si la condición se ccumple te aplica la condición afuera del bucle, es como si se saliera del bucle, pero solo de un bucle.

CONEXIÓN DE BASES

Dos sistemas: MYSQLI y PDO.

MYSQLI permite almacenar muy estructurado las bases de datos mientras que PDO permite cambiar de un lado a otro la base datos.

TIPO MYSQLI: El sistema para usar la base de datos no hay necesidad de cambiar la base de datos. LENGUAJE: PHP, API: MYSQLI, RDBMS: MYSQL Si optamos por PDO, tenemos que tener en cuenta que si lo vamos a hace es para mover una base de datos a otra.

MAriaDB = mysql respecto a instrucciones.

EN PHP MY ADMIN:

EN EL LADO IZQUIERDO PONER ‘NUEVA’ para crear una base de datos.

Ponemos el nombre debajo de CREAR BASE DE DATOS.

A su lado derecho tenemos los tipo de caracteres, buscamos los anglosajones, la creamos. Le ponemos: utf8\_spanish\_ci

Asignamos nombre de la tabla y el numero de columnas, le damos a continuar.

Nos mostrara una esquema para introducir los atributos y sus propiedades.

Las creamos.

Ya en VCS, tenemos que tener en la variable de conexión el mismo nomnbre que la BD que creamos.

DE NUEVO EN MYPHPADMION:

Ponemos la BD y vamos a la pestaña de insertar, nos permitirá insertar datos

TIPS PHPMYADMIN:

SI QUEREMOS CAMBIAR ALGUNA PROPIEDAD O AGREGAR NOS VAMOS A LA TABLA DONDE ESTAN LOS ATRIBUTOS Y LE PONEMOS LA OPCION AL LAD DE BORRAR. ‘CAMBIAR’, NOS PERMITIRA MODICAR LAS PROPIEDADES DEL ATRIBUTO. JUSTO HAY MISMO HAY UNA OPCION AL LADO DE GUARDAR, DONDE NOS DICE ‘PREVISUALIZAR SQL’, ESTO NOS AYUDARA A MIRAR HIZO EL CODIGO.

Junio 17/2019

Junio 21/2019

Mysql Workbenc:

Para relacionar en mysql Workbench en los diagramas, tenemos que tener en cuenta que la relación que le pongamos será importante si queremos que no nos salga error al momento de hacer una petición, ya que al hacer las relaciones en los diagramas se establece esa relación y

CONSEJO: Con el importar Base deDatos a un diagrama, podemos hacer esto con la base de datos de la empresa y asi nos mostrara el diagrama de la base de datos para guiarnos y estudiarlo mas fácilmente.+

LEFT JOIN: SE ENCARGA DE LA TABLA DE LA IZQUIERDA EN LA RELACION DE DOS TABLAS

INNER JOIN: SE ENCARGA DE LAMBAS TABLAS.

RIGHT JOIN: SE ENCARGA DE LA TABLA DE LA DERECHA.

IMPORTANTE:

LEFT JOIN; TE MUESTRA LOS ATRIBUTOS RELACIONADOS ENTRE LAS DOS TABLAS, PERO CON TUPLAS QUE TIENEN ALGUN DATOS DE LOS ATRIBUTOS RELACIONADOS VACIO, QUE ALGUNOS DE ESTOS DATOS ESTEN VACIOS.

INNER JOIN: TE MUESTRA LOS ATRIBUTOS RELACIONADOS ENTRE LAS DOS TABLAS, PERO NO TEMUESTRA AQUELLOS ATRIBUTOS RELACIONADOS QUE TENGAN EN SUS DATOS ALGUN NULL, TE MUESTRA DIRECTAMENTE AQUELLAS TUPLAS QUE TENGAN LA RELACION.

RIGHT JOIN: HACE LO MISMO QUE LEFT JOIN.

DIFERENCIAS ENTRE LEFT Y RIGHT JOIN:

EJM:

SELECT \*

FROM USUARIOS LEFT/RIGHT JOIN POST

ON USUARIOS.ID = POST.USUARIO\_ID;

SI USAMOS LEFT JOIN: LA ESTRUCTURA QUE HARA LA RELACION SE MOSTRARA EN BASE DE LA TABLA ‘USUARIOS’, ES DECIR LA 1 TABLA.

SI USAMOS RIGHT JOIN: LA ESTRUCTURA QUE HARA LA RELACION SE MOSTRARA EN BASE DE LA TABLA ‘POST’, ES DECIR LA 2 TABLA.

OUTER JOIN: SOLO LO USAREMOS EN UNAS BASES DE DATOS, NO DE MYSQL. VIENE HA HACER LO CONTRARIO DEL INNER JOIN, ES DECIR, NOS VA A MOSTRAR TODO SOBRE LAS TABLAS RELACIONADAS

UNION: PARA SUPLIR EL OUTER, PODEMOS USAR EL ‘UNION’ AL FINAL DEL CODIGO DE RELACION Y DESPUES PONER EL MISMO CODIGO DE LA RELACION PERO YA CON INNER, ESTO SERIA PARA SOLUCIONAR EL TEMA DEL OUTER SI DA ERROR. DE TODAS FORMAS YA ES MUY ANTIGUO Y POCO USADO.

SELECT \*

FROM USUARIOS OUTER JOIN POST

ON USUARIOS.ID = POST.USUARIO\_ID;

UNION

SELECT \*

FROM USUARIOS INNER JOIN POST

ON USUARIOS.ID = POST.USUARIO\_ID;

PERO SI NOS DA ERROR CON EL OUTER(YA QUE ES MUY ANTIGUO), ENTONCES PODEMOS HACER ESTO PARA QUE FUNCIONE COMO UN OUTER

SELECT \*

FROM USUARIOS LEFT/RIGHT JOIN POST

ON USUARIOS.ID = POST.USUARIO\_ID;

UNION

SELECT \*

FROM USUARIOS LEFT/RIGHT JOIN POST

ON USUARIOS.ID = POST.USUARIO\_ID;

BASICAMENTE ESTAMOS REPITIENDO DOS VECES EL MISMO CODIGO.

Tip: en WHERE, si queremos que nos muesytre según un dato vacio, usamos IS NULl, no = null.

Asimetría del JOIN: SI QUEREMOS QUE NOS MUESTRE LOS NULL DE UNA TABLA DE LAS DOS TABLAS QUE ESTAN RELACIONADAS, AL MOMENTO DE UNIR LAS TABLAS YA SEA CON LEFT/RIGHT PODREMOS USAR EN EL CONDICIONAL WHERE ‘IS NULL’, SI USAMOS LEFT JOIN: COMO SE EXPLICO ANTES, LEFT NOS MOSTRARIA TODO LOS VACIOS DE LA TABLA 1 EN BASE DE QUE EN LA CONDICION, EL ATRIBUTO SEA DE LA TABLA 2, PAR AQUE SEA MAS FACIL QUE NOS MUESTRE DIRECTAMENTE EN BASE HA ESA CONDICION Y NOS MUESTRE LAS TUPLAS QUE TIENEN ESE ATRIBUTO VACIO Y QUE ADEMAS NOS MUESTRE DIRECTAMENTE LA TUPLA DE LA TABLA 1 QUE ESTA RELACIONADA CON LA TUPLA DE LA TABLA 2 EN DONDE ESE ATRIBUTO DE LA TABLA 2 ESTA VACIO MIENTRAS QUE RIGHT SERIA HACER LO CONSTRARIO QUE LEFT.

WORKBENCH VISTA(VIEW): NOS PERMITE RECOPILAR LINEAS DE CODIGO QUE HAGAMOS YA SEAN PETICIONES U OTRAS COSAS, para poder usarlas después o para mostrarlas.

PARA LLAMAR A UNA VISTA ENB MYSQL: SELECT \* FROM nombrebasededatos.nombredelavista;

month(fecha Ejm: 2019-04-03). Lo que nos hace month() es coger de una fecha el mes.

Si queremos hacer una condición para que ese mes sea abril, es decir ‘04’:

WHERE month(fecha Ejm: 2019-04-03) = ‘04’.

El GROUP BY tiene que ir despues del WHERE

LIMIT comúnmente va aa venir acompañado del ORDER BY ASC/DESC

‘--’ Para hacer comentarios

Los WHERE no se repiten, se usa AND para hacer mas condiciones

Monthname: Permite sacar el año de una fecha.

HAVING: Nos permite sustuir el WHERE cuando el WHERE necesita ir debajo del GROUP BY porque queremos que nos agrupe primero antes que aplique la condición, por lo cual usamos HAVING, no cambia la estructura ni nada, HAVING funciona igual que un WHERE, por lo cual HAVING. Puede estar debajo/despues del GROUP BY pero no después del ORDER BY

Este caso cuando lo haríamos ¿. Lo usaríamos cuando usemos en el SELECt un count y el count necesita contar algo obviamente, por lo cual GROUP BY nos ayudara en esta labor, por lo cual la condición vendrá despues ordenarlo.

VARCHAR (255): Maximo numero de caracteres en un VARCHAR

BOLEANO: ES EL TINYINT()

JUNIO 25/2019

Google Cloud:

Ssl: certificados

Ir a la opción Usuarios:

Crear una cuenta de usuarios en google cloud, dejar la opción Permitir todos los hosts

Ahora en la opción Conexiones, activar la opción la IP publica. Buscamos nuestra ip por internet y la copiamos para pegarla en esta misma opción de IP publica, LE DAMOS NOMBRE. Y le damos a guardar.

En la parte de SSL le damos permitir conexiones seguras, no el la ‘no seguras’

Ahora le damosa crear certificados de servidor SSL, le damos un nombre y creamos, nos saldrá que nos a creado el certificado y con unos 3certificados de conexión: la llave, certificado y el archivo: key, cert, file. Estos archivos los descargarmos

Ahora nos dara una IP, esta ip nos servirá para conectarlo con WOrkbench, ya en WOrkbench creamos una nueva conexión cmom hicimos con XAmpp,

Damos a crear una conexión y en el IP colocamos la que nos dio google, ahora le damos a la opción SSL e introducimos los archivos del certificado que creamos(key, cert y file) y lo colocamos en el SSL, volvemos al inicio de la creación de datos y le damos en Username el nombre que le creamos en el google del username

Lo que hicimos fue conectar el Workbench con el google mediante un cifrado de conexión.

QUERYS DENTRO DE QUERYS:

-- TABLAS EN MEMORIA: VAMOS HA HACER UN SELECT con peticiones en donde lo que queremos que en un atributo que pedimos con este SELECt

-- se agrupen varias peticiones en ese atributo con otro SELECT, porque queremos hacer eso?. Porque queremos optimizar la cantidad de atributos que queremos mostrar en el primer SELECT.

-- y porque es mas funcional agrupar peticiones en un solo atributo. Como lo hacemos?. En este caso mediante el FROM y poniendole (), nos permitira hacer otro query(peticiones) dentro del FROM

-- con esas peticiones que queremos que se ejecuten en ese atributo. Como nombramos ese atributo?.

-- Para ello, es importante saber que las peticiones que tenemos en el 2 query las nombramos con AS, el nombre que le pongamos a esa peticion

-- se la pondremos junto al atributo en el primer SELECT. ejm: atributo.nombreAS.

-- en el atributo podemos poner cualquier peticion que hicimos dentro del FROM simplemente cambiando el nombre en el atributo.

-- Otro punto importante es que las peticiones no solo irian en el FROm puede ir en un WHERE, es dependiendo de las necesidades que tengamos. El query que esta dedntro de FROM. El primer SELECT es lo que nos va a mostrar

SELECT new\_table.date, count(\*) AS contar

-- Dentro del FROM vamoas acrear la tabla en memoria

FROM (

-- Query: hacemos otro qury para hacer la peticion, es como si fuera un query dentro de otro query.

-- min: Minimo dato de fecha, es decir la primera fecha

-- Llamamos esta peticion como date As date

SELECT DATE(min(fechaCreacion)) AS date, year(fechaCreacion) AS post\_year

FROM post

group by post\_year

) AS new\_table -- Aqui es como si se creara la tabla, para que despues pueda ejecutar el query que esta dentro del FROM

-- Agrupar para contar y lo ordenamos

group by new\_table.date

ORDER BY new\_table.date DESC

;

-- MAX: maxima fecha, es decir la ultima

EL COUNT(\*), CUENTA LO QUE PONGAMOS EN GROUP BY, OSEA VA A CONTAR EL GROUP BY DE ESE ATRIBUTO

-- GROUP\_CONCAT: nos permite concatenar datos de un atributo como si fuera una lista con cada lista

-- usamos el HAVING porque queremos que primero se haga el GROUP BY para que nos lo ordene primero.

-- Si no tenemos unos atributos concretos que vamos a escoger en el SELECT simplemente ponemos \*, y ya despues condicionaremos lo que queremoas de algunos atributos mas adelante.

-- Ejerccicio: Que etiquetas no tienen ningun POST

-- Si no tenemos unos atributos concretos que vamos a escoger en el --SELECT simplemente ponemos \*, y ya despues condicionaremos lo que -- queremos de algunos atributos mas adelante.

-- Columnas que queremos

SELECT \*

-- Tablas que queremos

FROM blog.etiqueta AS e

-- usaremos LEFT JOIN: porque queremos que me muestren todas las etiquetas aunque no tengan una relacion, se preguntara: hay una relacion? si la hay bien, sino tambien, en este caso la tabla etiquetas la pusimos en el LEFT porque queremos que esta haga ese trabajo

-- Tambien pensemos que la trabla eitquetas ahora mismo solo se relaciona con la tabla puente\_post\_etiquetas, por lo cual esta sera la unica que nos permititra saber cuales etiquetas no tiene ningun post.

LEFT JOIN blog.puentepostetiqueta AS puente

ON e.id\_Etiqueta = puente.id\_Etiqueta

-- Filtro, condiciones.

WHERE puente.id\_Etiqueta IS NULL

;

Junio 26/2019

BASES DE DATOS NO RELACIONALES

NOSQL

NO EXISTEN UN SOLO TIPO AUNQUE SE ENGLOBEN EN UNA SOLA CATEOGRIA.

SE EMPLEAN PARA CUESTIONES O NECESIDADES CONCRETAS

TIPO: CLAVE-VALOR: SON IDEALES PARA ALMACENAR Y EXTRAER DATOS DE UNA CLAVE UNICA. MANEJAN LOS DICCIONARIOS DE MANERA EXCEPCIONAL:

ALGUNAS SON DYNAMODB, CASANDRA(CREACION DE FACEBOOK),

BASADAS EN DOCUMENTOS: Son una implementación de clave-valor que varia en la forma de semiestructurada en que sew trata la información. Son basadas en documentos, es decir, estamosguardando información tipo documental y se asemeja a la clave-valor. NO ALMACENA ARCHIVOS DE DOCUMENTOS.

BASADAS EN GRAFOS: Sirver para entidades que se encuentran interconectadas por multiples relaciones, ideal para almacenar relaciones complejas. Cuando tienes una cantidad de servidores para una cantidad inmensa de información, la teoría de grafos se plica a este tipo para ser específicos en interconectar las multiples relaciones que halla en esos servidores.

EN MEMORIA: Pueden sewr de estructura variada, pero su ventaja radica en su gran velocidad, ya que al vivir en memoria de extracción de datos es casi inmediata. Memcached.

OPTIMIZADAS A LAS BUSQUEDAS: Pueden ser de diversas estructuras Su ventaja radica en que se pueden hacer querys o peticionesy búsquedas complejas de manera sencilla.

TODAS ESTAS DEPENDEN DE LAS NECESIDADES QUE TENGAMOS

FIRESTORE: BASES DE DATOS DOCUMENTAL.

Lo que ahremos primero con firestore es delegar funciones como MAtenimiento de seguridad, redes, distribución de dadtos, etc. A un operador de Cloud como puede ser Google o amazon.

Se utiliza tanto para la creación de app móviles o web.

JERARQUIA DE DATOS EN FIRESTORE: Hay una jerarquía en donde bases de datos-> Colección(dentro de cada colección tendremos documentos(JSON, XCML NO SON ARCHIVOS DE DOCUMENTOS COMO PIENSAS)-> documentos.

EN firestore no esperemos una consulta, porque FIRESTORe sirve para las aplicaciones TANTO web como móviles, donde queremos representar un momento en la aplicación para darle cierto contenido de la base de datos, es decir, cuandop el usuario pulsa un botón, el firestore permite darle la infromacion requerida y especifica para este usuario, por lo cual no sirve de nada un SELECT.

BASES DE DATOS, DISCIPLINAS EN EL TRABAJO REAL

Lo que nos encontraremos en el mundo laboral.

BIG DATA: Es un concepto que nace de la necesidad d emanejar grandes cantidades de datos. La tendencia comenzó con youtube, y e sque youtbe necesitaba guardar y consultar mucha información. Big data permite controlar esta gran cantidad de información. Consiste en el uso de direfentes tipos de bases de datos.

DATA WAREHOUSE(ALMACEN): ‘ UN GRAN ALMACEN INFINITO EN DONDE PODEMOS GUARDAR CUALQUIER COSA DE CUALQUIER TAMAÑO’. Todo es esto para almacwenar para la posteridad, se guarda como un historial de la información y datos. Nos sirve para crear estadísticas, etc.

DATA MINING: Es básicamente minería de datos, es decir, búsqueda de datos aunque estén en desuso o muertos, para convertilos en algo útil. Extraer de donde sea Identifica y extrae información, arrojar resultados, estadísticas para crear estrategias que benfician a la empresa.

ETL: ES básicamente edd un data mining, lo que hace es extracción, tranformacion y cargar. Conseguir que esos datos funciones para algo desde archivos que no están útiles o muertos, para después guardarlo también en un data warehouse.

CMR: mas que simples BD.

BUSINESS INTELLIGENCE: lo que conseguimos con esto mediante las diversas disciplinas mecionadas anteriormente para crear un plan, objetivos, métodos y medios para conseguir unos objetivos, hay diferentes especialidad en donde estas son las disciplinas.

MACHINE LEARNING:

Es una serie de tecnias que innvolucran la inteligencia artifical y la detección de patrones, tiene un grn campo de acción para business intelligence. Ayuda a encontrar patrones fortuitos encontrando correlaciion entre diversos patrones.

DATA SCIENCE: backend fullstack.

Aplicar todas las técnicas de procedimientos de datos. Se le contrata para dirigir un grupo que dominen esas disciplinas.

DATA sCIENCE: - Computeer science, Math and Statistics, Domains/Business Knowledge. Junta: machine learning, software develop, traditional research.

<https://firebase.google.com/>: nos permitirá usar el firestore de google.

No podemos dejar la colección vacia porque entonces la borrara si la creamos y la dejamos vacia.

Top level colección: Coleccion de top level. Son la mas importnate, hay mas tipos.

Creamos un colection

BOOLEAN: TYNINT(). Es un tipo de boleano

GEOPOINT: punto geográfico.

REFERENCE: no es un foreign key pero algo parecido.

La ID es un cifrado, por lo cual podemos dejar que google lo haga automáticamente aunque podemos ponerlo nosotros mismos.

Lo que nos permite al momento de añadir campos es que nos todos los ‘documentos’ que van después de las colección como vimos en la jerarnquia, tengan que tener los mismos campos igual que otro documento, es decir aquí no es obligatorio tener siempre los mismos campos en todos los documentos de una colección a diferencia de las relacionales en donde la cantidad de atributos que pongamos es para introducir información, en cambio aquí la cantidad de campos donde se va a meter la información nosotros decidimos cuantos queremos.

Lo que nos permite este tipo de NOSQL es que nos va a mostrar una información especifica que queremos, no medianteo una petición, sino como un ‘pantallazo’.

En la columan de Agregar documentos en la barra de arriba tenemos tres líneas, esto es un filtrador que nos permite hacer condiciones para ese documento.

Según el tipo de BD no relacionales, tendremos unas necesidades diferentes. En donde si tenemos una cantidad de información podemos elegir cual tipo usar según que queremos hacer con esa cantidad de información.

Subcolecciones: Son subcolecciones que nos permitirá añadir una colección a un documento que tenemos del colección principal. Por ejemplo un documento de ‘Juan’, este podría tener una colección donde se especifique el tipo de ropa que tiene. Una colección no puede exisitir de ninguna manera si no hay algún documento.

REGLA 1 FIRESTORE:

COMO SE VERA LA APLICACIÓN: preguntas: que datos queremos extraer

REGLA 2: es la excepciond e la primeras regla, son aquellas entintades que necesitrsan modificar constantemente. En el caso de los colections top level para modificar, cambiar.

En la creación de campos-> tipo-> MAP: nos permite hacer una lista con mas campos para realizar.

Geopoint: Nos permite colocar una localización, la cual podemos buscar google maps por ejemplo y copiar el código(como un ip) de la localización, que seria la latitud y altitud.

REFERENCE: nos permite relacionar una referencia(que puede ser un ID de otro documento, dependiendo de lo qu quieras) con el documento que vamos a a crear el REFERENCE. Refenciar un documento a otro documentos, por ejemplo en el BD de blog, haríamos una subcoleccion en la colección de usuarios, y los documentos de la subcoleccion ponemos un campo con REFERENCe para referencias el Id del documento de la colección POST para saber en que POST se puso ese COMENTARIO.

Las etiuetas también serian un subcoleccion de post, y este seria a diferencia de categorías(ARRAY), etiquetas serian un MAP, porque queremos que lñas etiquetas con BOOLEAN podamos activarlas o desactivarlas. Ya en php podemos usar un CHECKBOX para elegir cuales etiquetas elegir.

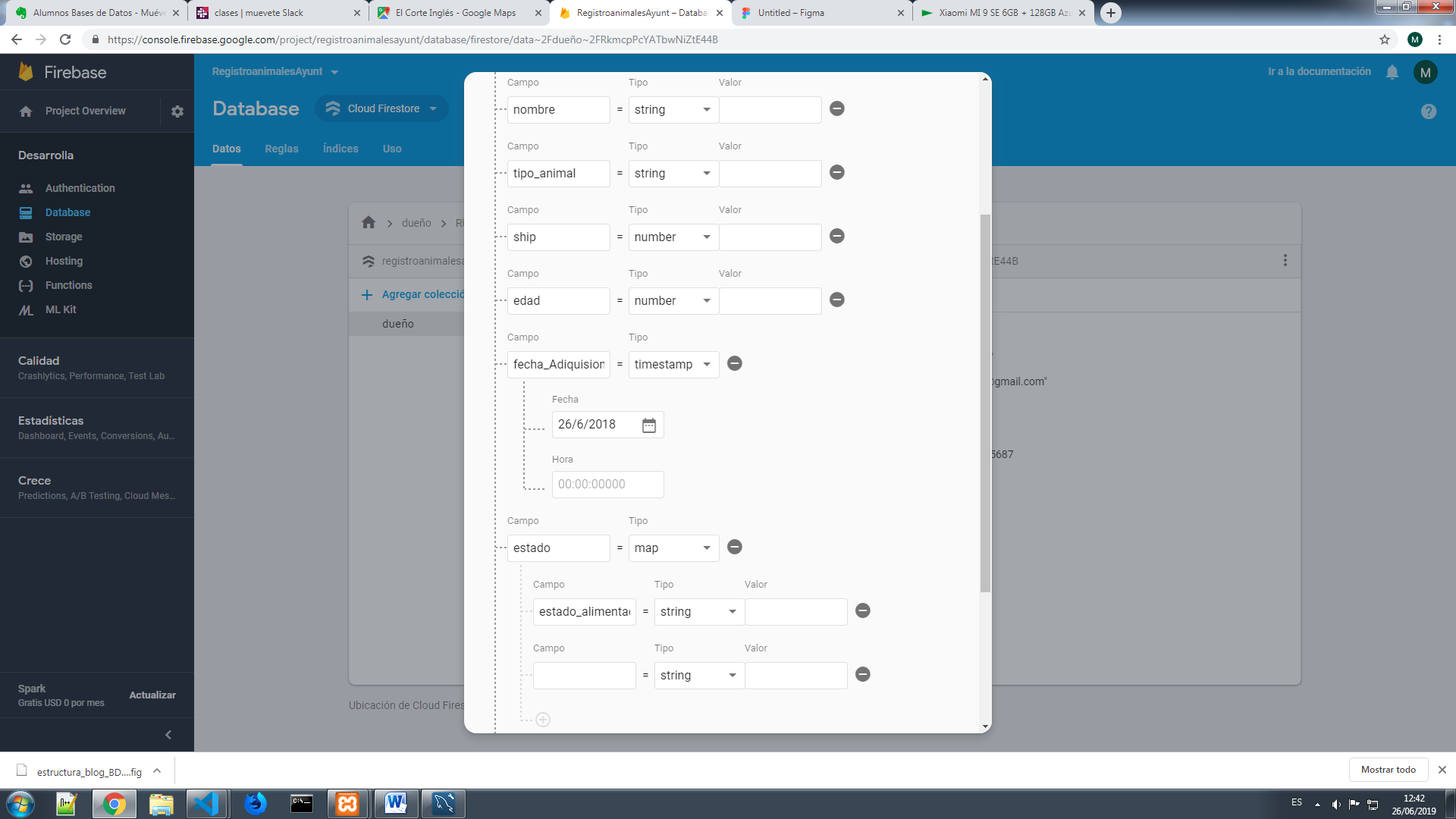
Como te imaginaras no hay FK en estas bases de datos y el PK te lo hace el propio en google.

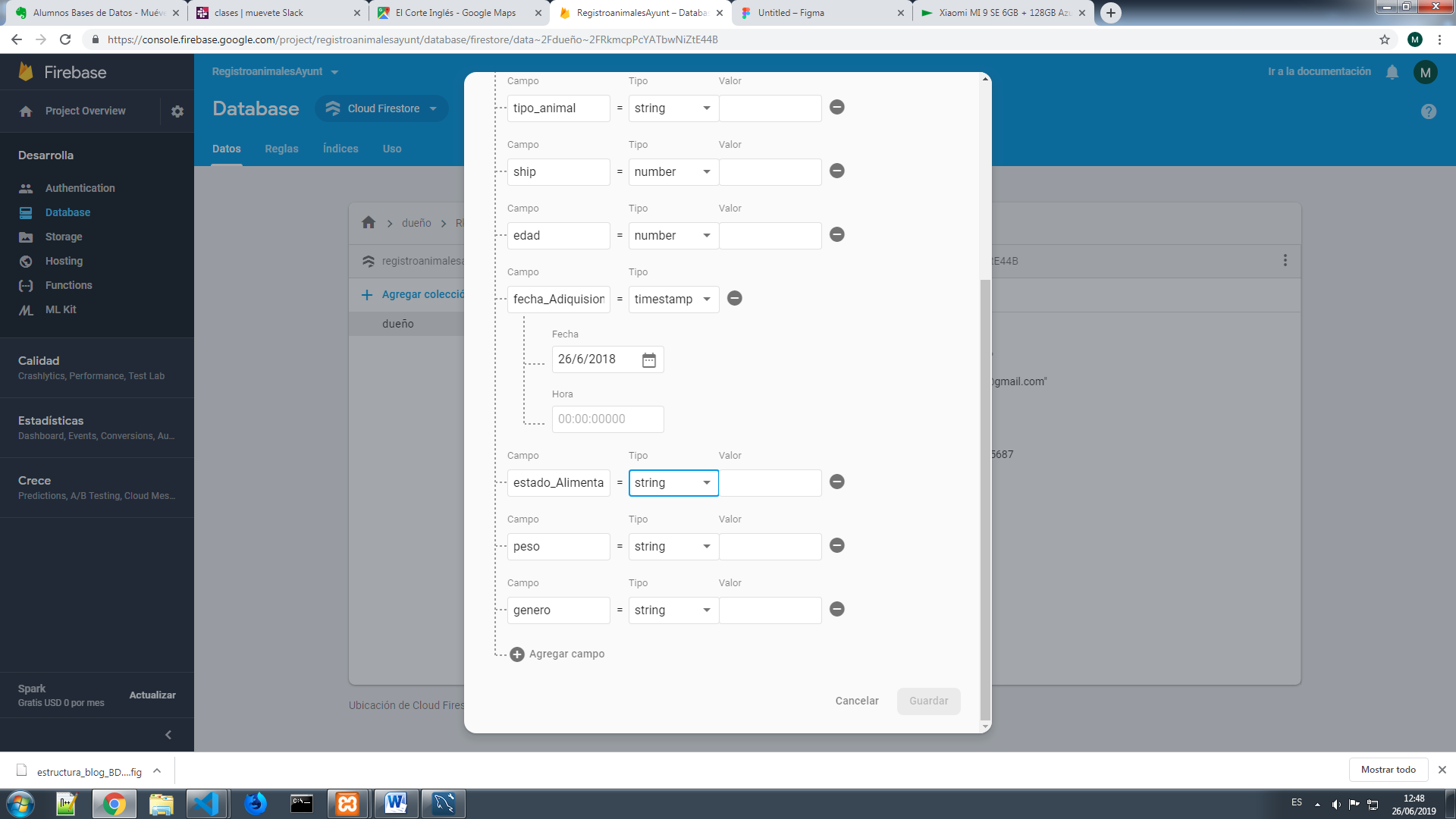
FIREBASE: noYEs es parte de javascript parte de

LO MEJOR ES HACER UN DIAGRAMA DONDE SE MUESTREN LOS ‘CAMPOS’ QUE VAMOS AHA USAR. SEL LLAMAN DIAGRAMAS PIRAMIDALES, EN DONDE SE DEJEN CLARO LA ESTRUCTURA DE LOS DOCUMENTOS Y LA JERARQUIA.

La idea en el diagrama es que vaya aumentando cada vez si hay subcolecciones o mas documentos.

AYUNTAMIENTO VALDEMOSA: QUIERE TENER CVATALOGADO A NIVEL DE ASPECTO TODAS LAS MASCOTAS DEL MUNICIPIO, ENTONCES LOS DUEÑOS TIENEN QUE IR A REGISTRARLO.



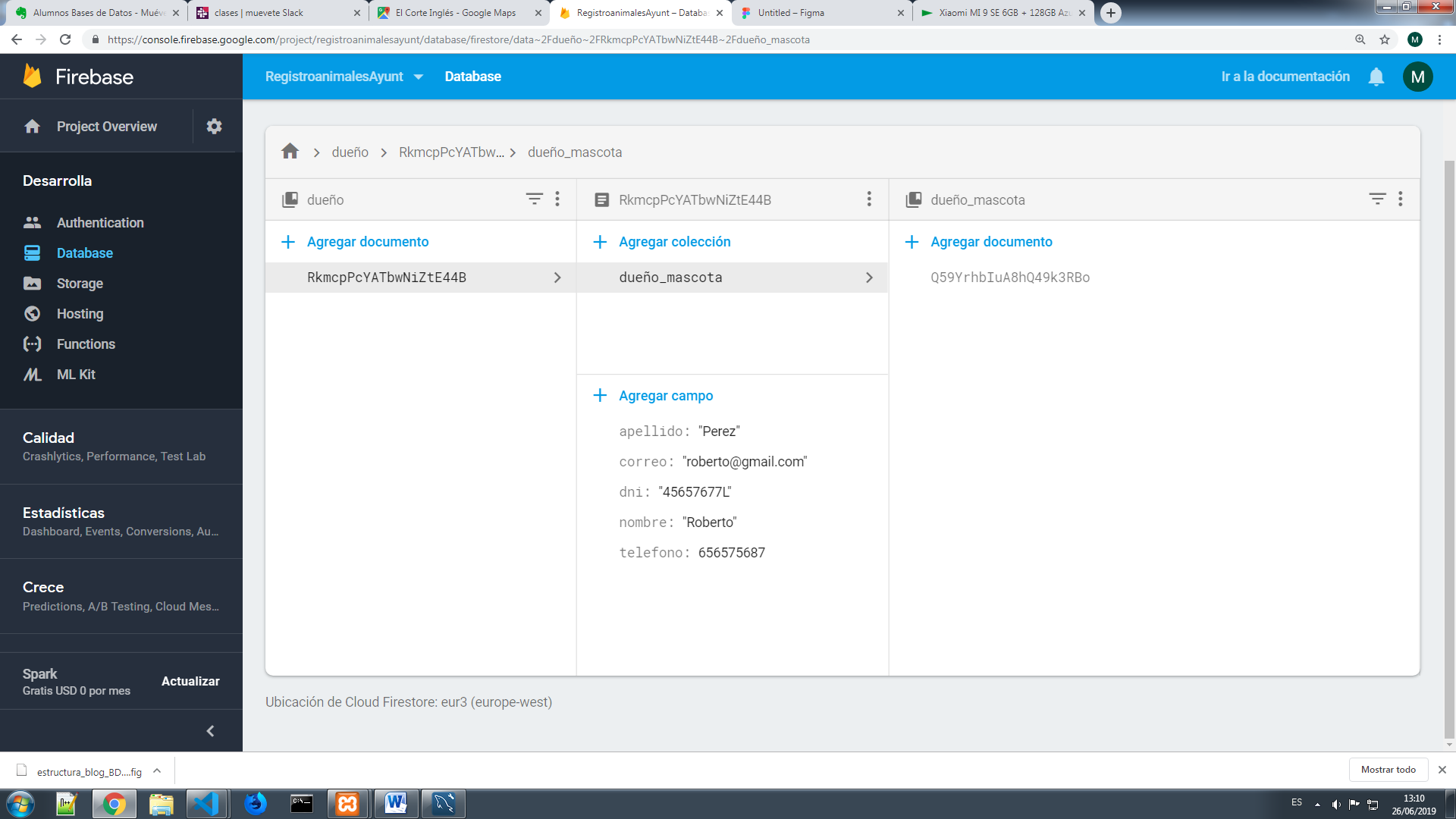


ACTITUD:

PEDIGREE:

PROPIETARIO VA AQ IR DENTRO COMO UNA SUBCOLECCION DE MASCOTAS:

Ya que MASCOTAS al final va a englobar todo



Fecha\_residente

Fecha\_nacimiento

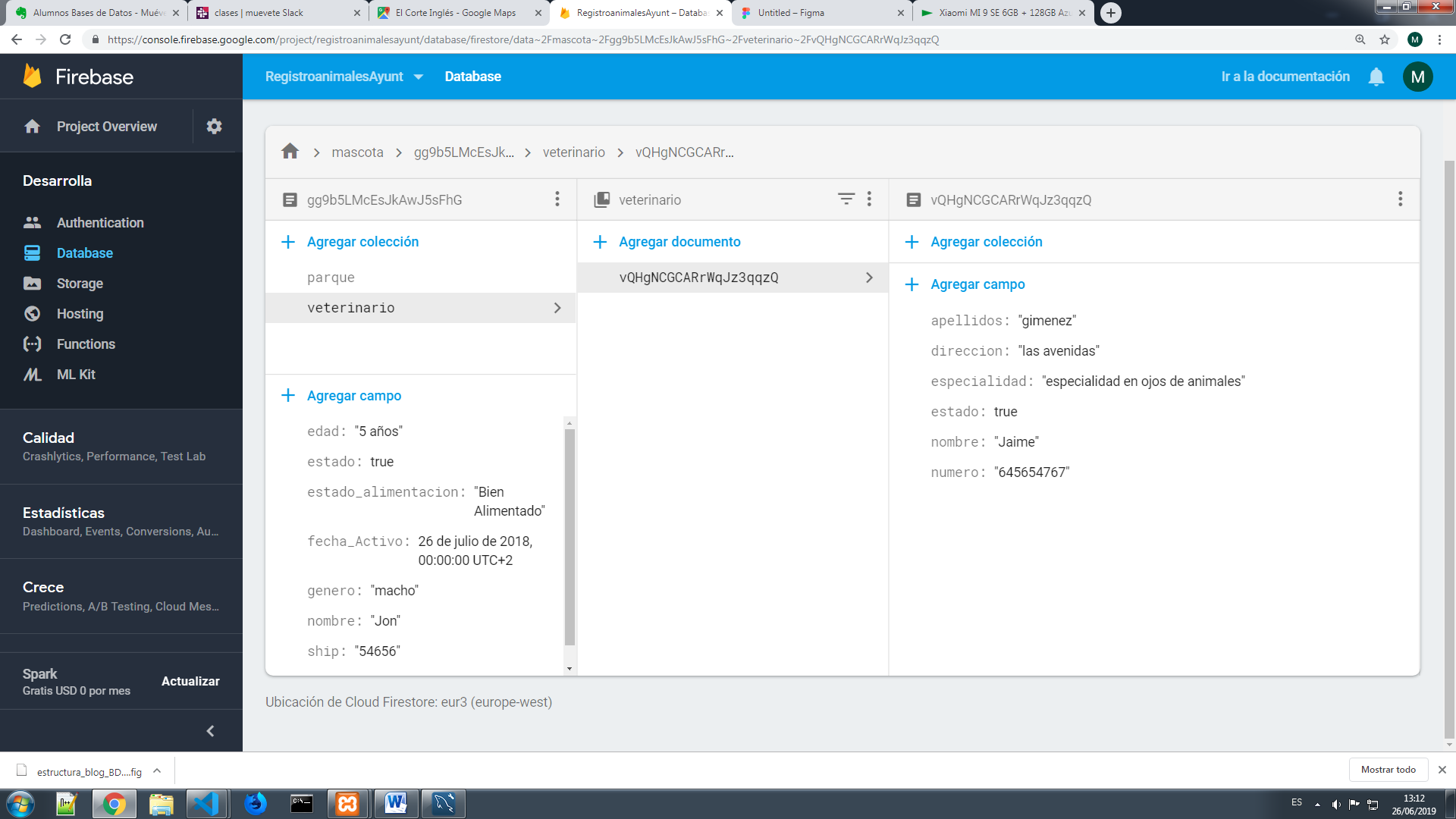
Activo\_mascota(tiene mascota o no)

Profesión

Minusvalía(map)

Genero(map)

veterinario



NUEVA SUBCOLECCION EN MASCOTAS:

SALUD: vacunas, pedigree, sexo, chi`, fecha\_ultima\_revision, especie, paseos(timestamp), parque(map)

EL DIAGRAMA PUEDE CAMBIAR A LO LARGO DE PROCESO

Los pantallazos(un esquema de como queremos que se vea la aplicación, no es el diagrama de la BD) nos definen como va a ser la base de datos

JUNIO 28/2019

YUML, PLANTTEXT: NOS PERMTE ELABORAR DIAGRAMAS

Vamos a subir los trabajos en repositorios en NUBE, ya no en LOCALES.

Github ¡= GItLAB. Github, plataforma que usa servidores para sistemas GIT.

Github tiene repositorios privados gratis.

git config --global user.name NuevoUsername

FORK en un respoitorio, nos permite copiar el codigo que se ha almacenado en FORK del proyecto de una persona, para usarlo en nuestro sitio sin tener que ir siempre al repositorio de esa persona para vferificarlos, eso no quiere decir OBVIAMENTE que ya sea nuestro, hay que dar crédito a la persona.

GIT CLONE: nos permite clonar el repositorio de un FORK de alguien mas:

EJM:

Git clone <https://github.com/matiusgg/fastapi>

Al momento de hacer GIT CLONE nos creaqra un espacio de trabajo llamado GIT LENS

GIT LEND: nos dicen las etiquetas, remove, brands,master, etc.

En remotes podemos ver

Para investigar que clase remotes tenemos que poner:

Git remote –v.: nos va a poner en que etsado esta nuestra conexión, es decir, detalles de la conexión github, para después poder hacer push(subir a repositorio los cambios que se hizo en el repositorio)(lo ideal seria descargarte los cambios que han los demás compañeros en el trabajo, ya que pueden haber problemas porque alguien no aviso de que hubo cambios).

Git remote –v: para conectar al github cualquier repositorio.

Git push –u add origin: permite subir los cambios del repositorio.

GIT PULL:

PASOS IDEALES:

1. Crear repositorio
2. Git clone en VSC del URL del repositorio
3. Git remote –v para verificar el estado de la conexión
4. Git add .
5. Git commit
6. Git push –u add origin
7. Git tag v(version). Para buscar todas las versiones hechas solo es poner git tag
8. Comprobar en GITHUB desktop app y darle al botón de arriba Fetch origin para refrescar la información, para subirlo al github como si hiciéramos un git push –a add origin.

GIT BRANCH no esobligatorio, no se suben se crean

git superlog: nos permite ver en resumidas cuentas en que git commit estamos donde nos indicara los detalles de la creación del commit.

TIP: ten en cuenta que al hacer el PUSH los demás aun todavía no tendrían

PASOS PARA TENER UNA BASE DE DATOS CONECTADA MEDIANTE CLOUD GOOGLE Y PHP:

Nos vamos

Esto NOS PERMITIRA SUBIR LAS bd HA UNA NUBE, CON ESTO ENTENDEREMOS QUE LA MAYORIA DE LAS EMPRESAS USAN ESTE METODO, O POR LO MENOS ES EL MAS ACTUAL.

Nos vamos a USUARIOS, creamos el usuario y contraseña para ponerlo en PHP:

Despues nos vamos INFORMACION GENERAL:

Activamos la opción IP publica

Vamos a la sección del Conexiones SSL: Damos click a ‘Permtitir conxiones no seguras’

Para crear certificados bajamos mas a Configurar certificados y lo creamos en ‘Crear certificados Cliente’, descargamos los 3 archivos que necesitaremos en la conexion y descargamos el certificado.

Para cambiar o modificar un certificado vamos a :Restablecer configuración de SSL.

EN INFORMACION GENERAL copiamos la IP que nos muestra y la ponemos en $servidor en php.

En CONEXIONES al lado de INFORMACION GENERAL

En BASES DE DATOs al lado de INFORMACION GENERAL, podemos ver las bases de datos que tenemos.

Cargamos la consola que es un icono que esta al lado de la foto de la cuenta. En donde es como el terminal de mysql.

PHP:

Creamos dentro del proyecto la carpeta SSL y dentro de ella los 4 archivos: key, cert, ca que descargamos cuando creamos el certificado.

'ssl/cliente-key.pem', 'ssl/cliente-cert.pem', 'ssl/cliente-ca.pem'

Y creamos un archivo com index u otro para poner el código de conexión:

CODIGO DE CONEXIÓN:

<?php

// CONEXION A LA BD CON GOOGLE CLOUD:

$servidor = ''; // IP del servidor CLOUD

$usuario = 'root'; // NUEVO Nombre de usuario, que no sea ROOT.

$password = ''; // Siempre password

$bd = ''; // nombre de la BD a conectar

$conexion = mysqli\_init(); // iniciamos la conexion

// Conexion a la BD a travez de MYSQLI

// Vamos a hacerlo hacia una conexion SSL(Certificados), si no hacemos esta parte nunca nos conectariamos:

// con esto conseguimos conextarnos al SSL

// Los ultimos dos certificados los vamos a dejar en NULL, porque en nuestro caso solo tenemos tres en google CLOUD

// Colocamos los certificados de izquierda a derecha-> key, cert, ca. En este orden

// tenemos que tener una carpeta donde tengamos ssl/cliente-key.pem para KEY

// En esta misma carpeta tambien tendremos ssl/cliente-cert.pem para CERT

// En esta misma carpeta tambien tendremos ssl/cliente-ca.pem para CA

// GENeramos son SSL\_SET(), para que se relacionen con la carpeta SSL

$conexion->ssl\_set('ssl/cliente-key.pem', 'ssl/cliente-cert.pem', 'ssl/cliente-ca.pem', NULL, NULL);

// Nos permite conectar ya el PHP y GOOGLE CLOUD como tambien REAL\_CONNECT

$conexion->real\_connect($servidor, $usuario, $password, $bd, 3306, NULL, MYSQLI\_CLIENT\_SSL\_DONT\_VERIFY\_SERVER\_CERT);

// comprobar si la conexion es correcta

if($conexion->connect\_error) {

die('Conexion Fallida');

} else {

echo 'Conexion perfecta' . '<br>';

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// para verificar aunque la conexion este bien, vamos a MOSTRAR

// DATOS DE LA TABLA

// EJM: TABLA USUARIOS

$resultados = $conexion-> query("select \* from usuarios");

foreach($resultados AS $valor) {

// ponemos lo que queremos que salga

echo $valor['id\_usuarios'];

}

// CERRAmos la $conexion porque cada vez que hacemos un query se nos abre la session

$conexion = mysqli\_close();

Si unque todo este bien nos da error, nos vamos a CONEXIONES y activamos conexiones no SEGURAS.

TIP: ADemas cada vez que poneoms un $conexion-> estamos abriendo sección, por lo cual usamos un mysqli\_close();

El vendor NUNCA SE SUBE A UN REPOSITORIO

GIT GRUPAL:

1. Git init
2. Crear carpetas desde 0, es lo mejor que se puede hacer.
3. Ls –al para comprobar que se creo el git init

ARCHIVO DE TEXTO PLANO: No son texto plano: .doc, .rtf pero on formato(no es texto plano aunque se acerque), los archivos binarios son.doc y no responden bien en git.

GGIT es una base de datos atómica, es decir, nos guarda por partes. Si escribnieramos en un texto plano para hacer commmits con git como .txt . EL git analiza que cambios hubo respecto al commit anterior y solo almacenara ese cambio mas no todo lo guarda de nuevo, solo esos cambios.

WORKING AREA:

STAGE AREA: RAM. ES la zona intemedia en donde hacemos la preparación de los archivos, se queda en la memoria ÇRAM,, solo se guarda en laBD archivar(git) cuando le hacemos un GIT COMMIT. Es un paso previo al commit, pero se guarda en memoria, osea hasta que no hacemos el commit no lo guardamos en el git.

UNTRACKED FILES: significa que no los ha llevado al STAGE AREA,

Para meterlo al STAGE AREA es con git add (nombredelarchivo). Aquí lo que consgeuimos es ‘TRAQUEAR’ o llevarlo al STAGE AREA:

Hacemos el git commit para subir este archivo al GIT.

TIP: podemos escribir en mas líneas de código en la terminal de GIT cuando acemos un commit para explicar el commit simplemente dejando abierto las commilas y dando espacio para escribir en la siguiente línea y asi sucesivamente, y lo podemos cerrar obviamente con comillas de nuevo.

HEAD: Nos indica en donde estoy, en que commit y en que BRANCH(RAMA).

Ejm: commit Hola nuevo commi 1 (HEAD -> master)

MASTER: Linea principal, no es un BRANCH.

Cuando hagamos un BRANCH, y solo estamos trabajndo en un archivo, podemos usar:

$ git commit -am 'segundo cambio en cv.txt'

Para no repetir el git add, ten en cuenta que solo estamos en un archivo por lo cual usamos este medio, pero si tuviéramos mas archivos entonces según cual queremos usamos el git add . para un archivo espefico.

Git show: nos deja observar los ambios entre un commit actual y el anterior y los compara pero solo en un archivo concreto. Si en un commit no ha ahbido cambios pasa al siguiente commit para ver si hay cambios y ese seria el que compara

Git diff: Nos permiteitra comparar las diferencias que hay entre commits. Nos saldrá algo parecid al git show pero la diferencia entre este, es que el git diff nos permite comparar los dos commits que queramos poner.

1. Git log: para saber donde estamos. Para salir le damos enter varias veces y le damos a la letra Q(quit), esto nos pasara cuando tenemos poco espacio. Usamos el git log para ver los commits.
2. Git diff (código del commit) (código del segundo commit que queramos comparar).

PARA retroceder en el tiempo a otro commit.

Usamos git reset: hay 2 tipos:

Git reset -–hard: regresa al commit que yo quiero, pero se borra los commit hechos despues del commit que escogimos, es decir se vacia el STAGE aREa(se vacia la memoria) y se borra todo los commit hechos hasta entonces después de ese commit que elegimos. HAY qUETE NER CUIDADO CON ESTE PORQUE SE PUEDE BORRAR TODO.

Git reset -–soft: se conserver los cambios que pusimos en STAGE AREA, pero de todas formas te borra los commits hechos después de ese commit que escogimos

Rjm:

Git log –stat: Nos enseña los commit con los cambios sucedidos, es decir, nos enseñara que cambios se han hecho en esos commit en los archivos. ES como una fusión entre el git log y el git show.

Git checkout: ES para volver a un commit sin borrarnos nada, a diferencia del git reset que vuelve y borra.

Ejm: git checkout 2b2682d6422e848fbbf0406c998cbf12a04fc981 index.php

Hemos ido a ese commit, es decir, hemos vuelto a ese estado de ese archivo para observar como lo hicimos en ese entonces o para recuperar cierto código para ponerlo en el commit mas actual.

Para volver simplemente es hacer lo mismo con git checkout, pero el código del commit, obviamente seria el del commit mas actual y el archivo a donde queremos estar de nuevo.

git checkout 2b2682d6422e848fbbf0406c998cbf12a04fc981 style.css

Supongamos que hacemos un commit al commit que volvimos, lo quye pasara es que ahora ese commit anterior al que volvimos se va a comvertir en el nuevo o mas actualizado commit obviamente.

TIP: ‘-‘: nos obliga a poner una letra. ‘--‘.

CREACION DE RAMAS(BRANCHS):

Git branch:

Para crearlo: git branch (nombredelbranch)

Si quisieramos poner un branch en otro

1. Para escoger el branch usamos git checkout (nombre del branch)

Y para verificar en que branch estamos solo ponemos git branch.

PARA FUSIONER UN BRANCH CON MASTER:

Git merge: Ten en cuenta que tenemos que estar en master para decirnos “Quiero recuperar este branch”. Por si usamos git merge desde un branch, estamos modificando complemtamente el MASTER a ese branch, lo cual estaríamos afectando a la rama principal, por lo cual lo hacemos desde MASTER. El git merge nos permite ‘fusiones un branch con el MASTER’.

1. Verificamos con git branch en que brnach estamos.
2. Hacemos un commit en el branch indicando que vamos a fusionar.
3. Hacemos un git log par verificar el commit que hicimos y si esta en el branch dnode hicimos el commit
4. Cambiamos con git checkout master, para ir a la rama de MASTER.
5. Comprobamos otra vez git braqnch para verificar si estamos en MASTER
6. Ponemos git merge (nombredelbranch)
7. Verificamos git log para comprobar si el el HEAD tenemos tanto el MASTER como el nombre del branch…

Si hacemos un cambio en ambos branch y hacemos sus propios commits, y hacemos un git merge, nos generara un conflicto, porque hemos hecho cambios en las mismas lienas y en mabas branch, y se genera porque le git merge dice. “Con cual te quieres quedar?”. Que se haría?, podemos borrar ambios cambios y poner solo uno, ya que como vemos estaríamos fusionando códigos en la misma línea, por lo cual dara error.

VSC nos muestre el error en ese archivo muy visualemtente en donde te mostrara y te dara las opciones de quedarte con uno o con otro, o hasta con ambos.

Hacemos un commit depenediendo de lo que escogimos para guardarlo.

Al hacer un git log te mostrara que hiciste en ese commit un MERGE.

Cuando tengamos todo, nos vamos a github creamos un repositorio sin ponerle nada solo el nombre, nos mostrara una guía de como conectar el git con el github.

Ya en el git bash para verificar si hay un enlace de conexión usamos git remote –v

Y nos dice que no hay conexión esta bien porque obviamente no hay ningún enlace. Por lo cual usamos(que nos muestra en la pagina): git remote add origin (enlace o URL del repositorio).

Esto nos lo conectara.

Y nos mostrara dos líneas(“fetch”, ”push”).

Y ahora hacemos un git push:

git push –u origin (nombre del master o del branch). Esto para subirlo al repositorio en github.

Ya en github actualizamos la pagina y nos aparecerán losa rchivos y commit.

Y si queremos trabajar en equipo y como controlar quienes suben con git push cada uno?.

Facil: se organiza que cada termine su parte y se habla para revisar y para que todos hagan el push.

IMPORTANTE: LOS GIT PUSH SE HACEN AL FINAL O CUANDO SE TENGA UNA CIERTA CANTIDAD IMPORTANTE DE COMMITS QUYE QUEREMOS SUBIR ENTONCES USAMOS EL GIT PUSH, NO UN GIT PUSH POR CADA COMMIT.

SI NECESITAMOS ARCHIVOS QUE EN UNA BRANCH NO TENEMOS

UN GIT PULL: ES LA UNION ENTRE GIT FERCH Y MERCH. ES DECIR, NOS HACE AMBAS COSAS. Como sabemos que ha pasado algún cambio?.

GIT FEHC: TRAAER INFORMACIONDE REMOTO A LOCAL, COGER ESA INFORMACION Y PONERLO EN LA RAMA QUE QUEREMOS.

Para ello vamos a github, escogemos el archivos que queremos tener y meterlo con git pull. Obviamente tenemos que ver como lo hizo para asegurarnos.

Git pull origin master

TIP: Si vamos al icono/foto de GITHUB, al lado de ella hay un “+”,

NEW GIt: se emplea para dejar codigo.

IMPORT REPOSITORY: para importar un repositorio de la nube de github.

NEW PROYECT: solo ponemos el nombre y si queremos private:

NEW ORGANIZATION: le damos un nombre, en billing email ponemos el correo que será un general para esa organización y para el tema de pasta. La creamos, después nos pedirá que invitemos a las personas.

PROCESO EN EQUIPO: UNA PERSONA SE ENCARGARA DEL MASTER, ESTE CREA LA ORGANIZACIÓN Y EL REPOSITORIO. ENVIA SOLICITUD PARA AGREGARLO COMO CONTRIBUIDOR. LAS PERSONAS QUE TIENE SU PROPIO BRANCH SE CONECTAN AL REPOSITORIO, CREAN EL BRANCH CON LOS COMMITS, Y PONEN GIT PUSH ORIGIN (NOMBREBRANCH). PARA DESPUES DE QUE LA PERSONA DEL MASTER TENGA TODO HECHO, ENTONCES EL ENCARGADO DEL MASTER USA GIT MERGE PARA UNIR LAS RAMAS.

IMPORTANTE: EL CREADOR DE LA ORGANIZACIÓN TIENE QUE AGREGAR AL EQUIPO EN CONTRIBUIDORES PARA QUE PUEDAN MODIFICAR Y AGREGAR BRANCHS.

PROCESO DE UNION DE BRANCHS CON LA LINEA MASTERS:

SASS carpeta:Subcarpetas:

NO tocar styles.css cuando en el grupo de trabajo se trabaja en SASS, tampoco styles.scss en la carpeta LIB(librerias).

Aquellos archivos cuando ponemos ‘\_’ no se recopila

Julio 08/2019

APIS(conexión a servers) Conexiones.

Si necesitamos conectar una aplicación con una BD necesitamos una cierta cantidad de archivos que nos permitan esta acción.

Funciona a tr avez de conexión http: json, xml.

API rest es mejor que el SOAB.

Necesitamos saber PHP arabel para que nos ayude en esta tarea, es un framework. Nos ayuda en el lenguaje de programación.

Hoy en día la mayoría de las empresas utilizan API REST para crear servicios. Esto se debe a que es un estándar lógico y eficiente para la creación de servicios web.

Buscando una definición sencilla, REST es cualquier interfaz entre sistemas que use HTTP para obtener datos o generar operaciones sobre esos datos en todos los formatos posibles, como XML y JSON. Es una alternativa en auge a otros protocolos estándar de intercambio de datos como SOAP (Simple Object Access Protocol), que disponen de una gran capacidad pero también mucha complejidad. A veces es preferible **una solución más sencilla de manipulación de datos como REST**.

Nos beneficia que el composer este dentro del sistema operativo. Nosinteresa porque si estamos dando larabel, la cosa va mas en serio. Ya que composer se actualiza muy seguido. Como sabemos si composer esta ene l sistema operativo?

1. Escribimos composer en el terminal de git bash. Nos aparecerá el composer con sus característica, ya que Windows impone el composer.

Apple MAC: poneoms en el terminal “composer”, hay que saber que en MAC y en Linux no hay un composer predefinido. Pero antes hay que asegurarnos que tenemos el composer, como?. Poniendo composer download y el código que nos aparece en el apagina, lo ponemos en el terminal normal en ninguna carpeta. IMPORTANTE: cada línea de código de la pagina lo copiamos y lo pegamos uno por uno. No copiar todo completo y ponerlo ene el temrinal.

1. Si ponemos composer –version veremos la versión con fechas de la verson.
2. Curl –sS <http://getcomposer.org/installer> | php. para Mac y Linux. (Curl nos permite activar funciones de instalación de unn programa) Si no nos da esto ponemos php –r “readfile(‘http://getcomposer.org/installer’);” | php
3. Sudo mv composer.phar /user/local/bin/composer (Esto es para mover el composer) para Mac y Linux

SI QUEREMOS ACTUSLIZAR COMPOSER: PONEMOS:

Composer self-update

(~: ponemos: Alt+g+r+4)

INSTALACION DEL LARABEL:

1. Vamos a la pagina de larabel, vamos a documantation, en donde nos dicen la versión del php(que es importante).
2. En VIa laravel Installer tenemos el código de composer. Pero antes ponemos en la terminal un:

composer global require laravel/installer. Nos permtira instalar LARAVEL

* Ahora lo que ahcemos es copiar y pegar de la pagina de laravel, ya que lo queremos es hace un sitio estable para coger laravel como con composer, lo siguiente es como queremos que sea la ruta: Windows: %USERPROFILE%\AppData\Roaming\Composer\vendor\bin

1. Y ponemos en la terminal touch ~/.bash\_profile. Si estas en cmder ponemos “nano ~/.bash\_profile”. NANO: es un editor que esta en todos los terminales, nos permite ir al editor para poder escribir en ese archivo en el que estemos, es decir con .bach\_profile estamos creando un archivo y con NANO podemos editarlo. Ya en NANO escribimos lo que copiamos de la pagina del laravel: export PATH= %USERPROFILE%\AppData\Roaming\Composer\vendor\bin:$PATH.

Damos intro y le damos a Control + X. Nos pregutna si queremos guardar. Le damos “Y”, enter y volvemos a la terminal. Despues poenmos laravel para comprobar si se instalo

1. Si se instalo.

En VSC instalamos como plugging: LARAVEl EXTENSION PACK, también

Apagamos el XAMPP para que no nos cree conflicto con LARAVEL.

La ventaja con MAMP PRO: Es qwue podemos hacer varios LOCALHOST, individualemten cada uno.

COMO CREAMOS UN PRYECTRO DE LARAVEL:

En mamp nos hace una carpetas en htdocs, vamos a ella en el terminal,

Invocamos al laravel escribiéndolo, para verificar.

1. Ponemos laravel new prueba\_laravel (crea un nuevo proytecto de laravel).
2. Entramos con cd al proyecto.
3. El composer.phar es el que necesitamos
4. Ponemos php artisan(nos mostrara comandos del laravel, TODOS).
5. Si ponemos php artisan server, nos dira que no hay server porque no lo hemos creado

VAMOS A CREAR EL SERVIDOR DEL LARAVEL:

Poenmos php artisan serve, nos dira que nos creara un servidor con una URL: damos click derecho a la URL y damos a abrir. Nos enviara al servidor que creamos.

No es lo mismo un servidor para PHP y uno para frameworks(laravel).

Control + C para salipor si estamos dentro de un comando, es decir que no nos deja poner comandos.

IMPORTANTE: Tenemos que tener el MAMP encendido para que nos abra el laravel.

Abrimos VSC:

Abrimos la temrinal BASH:

Abrimos los archivos del VSC, y abrimos ROUTES: que tengra adentro varios archivos .php

RUTAS EN LARAVEL:

Principal: Web.php

Sintaxis:

En la ruta le estamos diciendo que nos lleve allí y que también nos ejecute código. Podemos poner cualquier ruta que queramos e introducirle el código que pongamos

Como? EJMEPLO:

En una RUTA Podemos enviar información a una ruta como en el siguiente ejemplo, entrar en esa ruta o que esa ruta nos digira a otro archivo.

GET: Es el tipo de envio de informacion

Route::get(‘donde visitaremos la ruta Ejm: /muevete’, function () {

Return echo “Hola mundo”; (Como vemos arriba definimos la ruta, y aquí le ponemos codigo)

} )

Carpeta RESOURCE-< VIEWS: Tendremos un welcome.blade.php

Nos enconstraremos con html que es de la URL cuando activamos el servidor desde la consola.

En Web.php en ROUTES, también tenemos una ROUTe para ir a welcome.blade.php:

Route::get(‘donde visitaremos la ruta Ejm: /welcome’, function () {

Return view(‘welcome’); (como vemos podemos “llamar una VISTA”). Pero en este caso ponemos ruta arriba porque ya con VIEW, definimos la ruta

} )

BLADE: Es el motor de render del template del laravels, es una extensión, en este caso nos permite colocar directamente en el html, código de php. Pero este código de PHP están en su mayoría con @, lo cual permite colocarlo sin problemas sin tener que abrir <?php ?>.

RENDER: Convierte el lenguaje a una ejecución para visualizarPlantillas que permiten ejecutar el lenguaje.

Creamos una copia de welcome.blade.php y la llamamos “testeo”.

Nos vamos a la apgina larevel y vamos a CONTROLLS

Nos vamos de nuevo a Web.php y creamos una ruta para el archivo “testeo”.

Para poner mas código ponemos una coma “,” para separar dentro de la función VIEW.

CODIGO PHP SIN ABRIR <?PHP ?>, Y EN HTML MEDIENTE EL BLADE:

VARIABLES DE BLADE: Simbolo “{{}}”. Nos permite llamar a una variable

@if: Es como usar un if de PHP en html.

@isset(): Nos comprueba si una variable esta vacia o no. El isset en si es como un proprio condicional como sabemos

@endisset: ES como la el corchete final que cierra el @isset().

@else: Va después del condicional como lo conocemos. “SINO” hazme esto.

TODOS ESTOS CODIGOS NO NECESITAN ABRIR EL <?php ?>, simplemente en el mismismo HTML podemos usarlos, es como si HTML se mezclara con PHP.

Es decir, podemos colocarlo sin que nos ponga problemas.

Route::get(‘donde visitaremos la ruta Ejm: /testeo’, function () {

Return view(‘testeo’, [‘titulo’ => NEW Curso]);

} )

. Ya en el archive Testeo, ponemos Las variables de BLADE para llamar a la variable.

@isset($titulo)

{{$titulo}}

@else

{{‘No existe’}}

@endisset

En este caso, $titulo sale de la función VIEW() donde hicimos una variable llamada ‘titulo’, en comillas simples.

Como vemos no necesitamos requied ni nada, ya que el laravel pòr si mismo sabe donde esta.

APIS según profesor: Son