***COMMENT CRÉER UN PROJET LARAVEL***

Cmd **composer create-project laravel/laravel nom\_projet**

Cmd **laravel new nom\_projet**

Modèle MVC (modèle vue controleur) de laravel.

Partie moule (modele)

Partie vue

**Partie controleur** : permet de definir les acces

**Partie moteur de vue blade**

**DOSSIER APP :** contient la racine du projet

Un modele :

Un controleur :

**\*sous dossier http** : toute requete passe par ce dossier. Il contient 2 principaux dossiers : **contollers** (dans lequel on cree les controleurs ) et **middleware**( permet de créer les barrages specifiques pour des actions/ on l’utilise egalement pour controler les requetes qui entrent)

**Sous dossier models** (pour mettre les modeles)

**Sous dossier providers**( sont des composants externes qu’on importe )

**BOOTSTRAP** ( ne nous sert pas)

**CONFIG** : Toute classe crée ici doit être configurée.

**DOSSIER DATABASE** : Il contient 3 sous dosssiers

**\* Sous dossier migrations** : représentent le schéma (modèles : nom\_table, attributs, contraintes,...) d’une table

\* **Sous dossier seeders**: contient les elts qui vont etre ajoutes dans les BD

\* **Sous dossier factories**: permet de gerer les fausses donnees

**PUBLIC :** Dans lequel il y a le fichier index (

**RESSOURCES**

\* **Sous dossier views**: contient les vues

Blade permet de connecter la vue et le controleur

**ROUTES**

\* **Sous dossier api.php**: api est utilise a distance tandis que pour utiliser la dependance on se connecte

\* **Sous dossier web.php** (web permet de definir la route

**STORAGE** : c’est la partie privee de l’appli

**DOSSIER VENDOR** partie ou tout nos dependances sont affichees ou enregistrees

**FICHIER .env** : ne part pas en ligne

.env-example

**APP\_ENV** permet definir l’etat de la connexion

**FICHIER .gitignore** :

On liste les elts qui ne sont pas envoyes par le git

**FICHIER composer.json**

**FICHIER package.json** : contient toutes les dependances

**FICHIER server.php** :

**PROGRAMME**

***\*L’INSTALLATION DE L’ENVIRONEMENT***

***\*INITIALISATION ET UTILISATION DE GIT***

***\*NOTION DE CONCEPTION DE PROJET***

***\*HEBERGER UN PROJET***

**COMMENT EST-CE QUE GIT FONCTIONNE GIT ?**

Il permet de faciliter le travail collectif

**Respository**: « depot en ligne »

**git bash** : c’est l’invide de cmd de git

**git init** : cmd DEFINIR UN PROJET

**git add nom\_fichier** : cmd permettant de selectionner un fic

**git commit -m ’’message specifiant le travail effectuer’’** : cmd permet de sauvegarder un fichier

**git config --global user.name** nom\_utilisateur

**git config --global user.email** email\_utilisateur

**git log** : donne les details relatives a la sauvegarde effectuee

**git add .** : permet de selectionner plusieurs fichiers

pour **supprimer les fichiers**, *on les supprime manuellement dans le dossiers ou ils sont places puis on execute les cmd git add . et git commit -m « message »*

**git log --oneline**: pour faciliter l’affichage

**git reset « chaine de caracteres »** : pour effacer le fichier

**git revert « chaine de caracteres »**: pour supprimer le nœud selectionne

**ctrl c** : donne la procedure pour debloque(pour sortir ) d’une page

**git branch nom\_branche** : pour créer une branche

**git branch**: cmd pour voir les branches creees

**git checkout nom\_branch**: cmd pour se positionner sur une branche

**git merge master**: cmd pour recuperer le travail git local de master et le positionne sur git checkout nom\_branche choisi

**git branch -d nom\_branche**: pour supprimer une branche

**git checkout -b nom\_branche** : pour créer et naviguer dans la branche

**git checkout -b nom\_branche -m** : pour créer une branche et la rendre principale

site pour se connecter a git **: github.com**

*Utilisation d’un dépôt en ligne*

*Créer un repository*

Envoyer une demande d’adhésion au repository créé à un collaborateur git

**git remote add nom \_d\_ refence\_de\_la\_distance (par defaut origin) addres\_http** : pour connecter un git local au git distance)

**git remote -v**: pour verifier si le lien a été mis en depot distance (repository)

**git remote remove origin** : pour supprimer le depot local qui a été mis en ligne

**git push(partie statique du code )** **nom \_d\_ refence\_de\_la\_distance (par defaut origin) nom\_branche (par defaut main)** :pour envoyer un depot distance simple

**NB :** *Il faut être sur la branche x pour envoyer des données à une branche x*

*Il ne faut pas forcement être sur la branche y pour recevoir des données d’une branche y*

**git pull nom \_d\_ refence\_de\_la\_distance (par defaut origin) nom\_branche (par defaut main)**: pour recuperer le travail sur une branche specifique

***NOM D’UTILISATEUR GIT : laurameta6***

RESPONSIVE :

**Bootstrap** : est un framework

**Framework**: est un ensemble de bibliotheques déjà predefinies

**Template** : est un ens de bibliotheques déjà predefinies

**Copier un template c’est copier ses assets**

**\*DOSSIER MENTOR**

**Assets (outils)** : contient les fichiers de style (

**-Vendor** : dossier ou se trouve les dependances (bootstrap, jquery…)­­

**Un API** : est une dependance en ligne­­

**cmd php artisan serve** : Pour voir un projet laravel (pour cela on ouvre le dossier du projet puis on entre dans le terminal et on saisit la dite cmd)

créer un dossiers assets (contenu dans public du projet)

copier les templates et les coller dans le dossier cree

créer dans le dossier views (contenu dans le dossier RESSOURCE) un fichier app.blade.php

partir sur le navigateur faire un clic droit : « code source »

revenir a la 1ere page du navigateur faire un clic droit : « inspecter le code source »

faire un clic droit sur la 2eme ligne, cliquer sur copier ( copier l’interieur du html)

coller dans le fichier app.blade.php enregistrer

partir sur le dossier routes ( web.php) : changer la route (/app) enregistrer,

saisir la cmd php artisan serve dans le terminal, copier l’URL puis coller dans le navigateur

crtl + F : pour faire la recherche

ctrl+ H : pour selectionner et modifier

**about.blade.php**

@yield(‘………..’)

@

1. Créer un repository du nom de « training »
2. Inviter « tchamouramses@gmail.com »
3. Envoyer le projet sur le depot distant
4. Integration d’un template qui sera donne
5. Chercher et tester dans le projet que vous aviez integre les templates des commandes suivantes :

* Creation d’un model « Training »
* Creation d’une migration « CreateTrainingTable »
* Creation d’un controlleur « Training controller »
* Creation d’une vue training.blade.php

Chercher les types acceptes par des migrations

Exple : string, timestamp, increments…

Une migration permet de créer et de mettre a jour le schema de notre BD (sa structure). Il ne s’agit donc pas ici de gerer les donnees a proprement parler mais reelement de créer, modifier, supprimer… des tables, par exple.

**git pull [nom] [branche]**: permet de recuperer les donnees d’une branche

**php artisan make : model nom\_model**: permet de fabriquer un modèle

**php artisan make : migration create\_nom\_migration \_table**: permet de créer une migration

**php artisan make : controller UserController** : permet

Lorsqu’on cree une table le nom de la table doit toujours se terminer par S et leur nom de reference ne doit pas se terminer par S mais doit commencer par une lettre majuscule

Exple : users⬄ User

Students⬄ Student

Differentes fonctions pour definir les contraintes : unique(), unsigned(), default(), nullabe(),… S’ils ne contiennent pas de valeur par defaut il est possible qu’on mette.

Pour créer une migration : il faudra d’abord se positionner sur la racine du projet. Lorsqu’on cree une migration, elle apparait comme une cle primaire et pourra pas devenir ou etre definit comme cle etrangere pour qu’on puisse la definir a un moment donne comme cle etrangere. On ecrit $table->id() : =>table -> increment (‘id)

TAF :

APP d’enregistrement des produits

1. Créer une migration pour les produits
2. Créer une vue (formulaire) pour les produits
3. Créer un contrôleur qui ajoute un produit via la methode create
4. Créer le modele pour les produits
5. Faire communiquer 3 couches (MCV) pour ajouter des produits
6. Apres l’ajout dans le controleur, renvoyer la vue Welcome
7. Afficher les donnees enregistrees dans un tableau sur une nouvelle page

table-bordered : tableau avec bordure

table-dark : tableau noir

dans etudiantControllers

function store(Request

@for ($i=0; $i<count($data); $i++)

                 <tr>

                    <td> {{$data[$i]->id}} </td>

                    <td> {{$data[$i]->nom\_complet}} </td>

                    <td> {{$data[$i]->date\_naiss}} </td>

                    <td> {{$data[$i]->filiere}} </td>

                    <td> <button class="btn btn-danger"> Sup </button> </td>

                    <td> {{$data[$i]->niveau}} </td>

                 </tr>

              @endfor

<tr>

                  <td>1 </td>

                  <td>Tchamou ramses </td>

                  <td>02--2-2002 </td>

                  <td>informatique </td>

                  <td>4 </td>

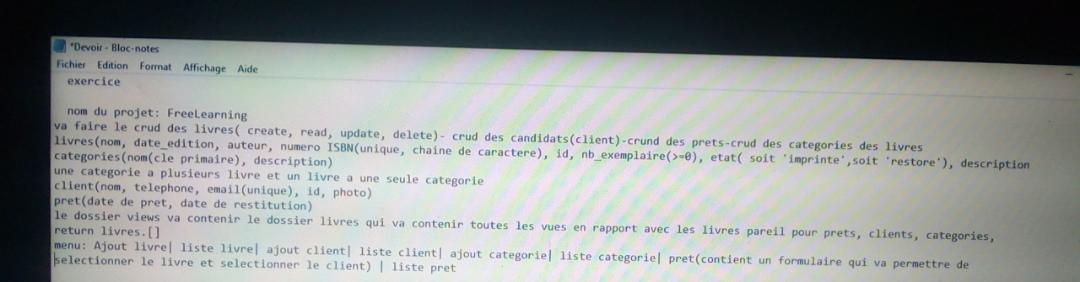
                  <td><button class="btn bnt-danger">Sup </button> </td>

              </tr>

TAF : dans le projet entrainement

1. Refaire ce qu’on a fait aujourdhui
2. Commiter le bon message
3. Chacun definit ses attributs(6
4. Créer un formulaire rechercher en haut du formulaire index( on utilisera l’attribut like)
5. Faire un ridirect dans la liste
6. Rendre le bouton Sup actif
7. Recherche sur la modification des donnees( ajouter un 3eme bouton « modifier »)
8. Documentation sur la methode create, store, finfOrfaid, find….

Recherche sur un champ : Etudiant :: where Nom\_complet (‘$nom ‘)



PROJET COLLECTIF

Nom de l’application : EasyRes

Fonctionnalités

1. MODULE 1 : Hôtels

* Gestion des hôtels (CRUD)
* Gestion des chambres d’hôtels (CRUD)

1. MODULE 2 : Réservation
   * + - Soumettre une réservation
       - Traiter une réservation
       - Lister les réservations
2. MODULE 3 : Utilisateurs
   * + - Gestion des utilisateurs (CRUD)

CONTRAINTES

On peut prolonger

L’application aura 2 parties : un site et une partie d’administration

ACTEURS(ROLE) DE BASE :

Un CLIENT (standard)

Des ADMINISTRATEURS (

Un UTILISATEUR (qui pourra jouer plusieurs rôles)

TABLE :

* + - * + Hôtels
        + Chambres
        + Utilisateurs (users)
        + Réservations
        + Categories\_chambres
        + Villes
        + Quartiers
        + Image\_chambre

Font awersone icons : pour telecharger les icones (sur le navigateur)

La fonction old() : permet de remplir le formulaire sans la soumission

LIKE : c’est un opérateur de comparaison

AJAX est connecte au JQuery

Backdrop : font d’arriere plan (lors du clic sur un bouton)