1. Créer un dossier « La Maison »

* Localisation : C:\xampp\htdocs

1. Création d’un projet LARAVEL nommé « LaMaison »

* composer create-project --prefer-dist laravel/laravel LaMaison

1. Création de la table « products »

* php artisan make:migration create\_products\_table

1. Création de la table « categories »

* php artisan make:migration create\_categories\_table

1. Création du schéma de la table « products » (**2020\_01\_13\_085212\_create\_products\_table.php**)
2. public function up()
3. {
4. Schema::create('products', function (Blueprint $table) {
5. $table->bigIncrements('id');
6. $table->string('Title',100)->nullable();
7. $table->text('Description',100)->nullable();
8. $table->decimal('Price',5,2)->nullable();;
9. $table->enum('Size',['46','48','50','52'])->nullable();;
10. $table->string('url\_image',100)->nullable();;
11. $table->enum('Status',['publié','brouillon'])->nullable();;
12. $table->enum('Code',['solde', 'new'])->nullable();;
13. $table->string('Reference',100)->nullable();
14. $table->dateTime('published\_at')->nullable(); // DATETIME nullable non obligatoire
15. $table->foreign('category\_id') ->references('id')->on('categories');
16. $table->timestamps();
17. });
18. }

6. Création du schéma de la table « categories »

(2020\_01\_13\_085323\_create\_categories\_table.php)

    public function up()

    {

        Schema::create('categories', function (Blueprint $table) {

            $table->bigIncrements('id');

            $table->string('Title',100)->nullable();;

            $table->text('Description',100)->nullable();

            $table->dateTime('published\_at')->nullable(); // DATETIME nullable non obligatoire

            $table->timestamps();

        });

    }

7. Création du modèle « Product »

🡺 php artisan make:model Product

8. Création du modèle « Category »

🡺 php artisan make:model Category

9. Déclaration de la relation **One-to-Many** dans la classe « Product » (Product.php)

class Product extends Model

{

    public function category(){

        return $this->belongsTo(Category::class);

    }

}

10. Déclaration de la relation **One-To-Many** dans la classe « Category » (Category.php)

class Category extends Model

{

    public function products(){

        return $this->hasMany(Product::class);

    }

}

11. Création d’une database dans « PHPMyAdmin » nommée « **lamaison** »

- utf8mb4\_unicode\_ci

12. Vérification du méta fichier « .env »

**DB\_CONNECTION=mysql**

**DB\_HOST=127.0.0.1**

**DB\_PORT=3306**

**DB\_DATABASE=lamaison**

**DB\_USERNAME=root**

**DB\_PASSWORD=**

12. Réalisation de la migration de mes tables :

**C:\xampp\htdocs\LaMaison**

**λ php artisan migrate**

**Migration table created successfully.**

**Migrating: 2014\_10\_12\_000000\_create\_users\_table**

**Migrated: 2014\_10\_12\_000000\_create\_users\_table (0.07 seconds)**

**Migrating: 2014\_10\_12\_100000\_create\_password\_resets\_table**

**Migrated: 2014\_10\_12\_100000\_create\_password\_resets\_table (0.06 seconds)**

**Migrating: 2019\_08\_19\_000000\_create\_failed\_jobs\_table**

**Migrated: 2019\_08\_19\_000000\_create\_failed\_jobs\_table (0.02 seconds)**

**Migrating: 2020\_01\_13\_085212\_create\_products\_table**

**Migrated: 2020\_01\_13\_085212\_create\_products\_table (0.03 seconds)**

**Migrating: 2020\_01\_13\_085323\_create\_categories\_table**

**Migrated: 2020\_01\_13\_085323\_create\_categories\_table (0.02 seconds)**

13. Création d’une factory ProductFactory

🡺 php artisan make:factory ProductFactory --model=Product

14. Création d’une factory CategoryFactory

🡺 php artisan make:factory CategoryFactory --model=Category

15. Remplissage de ProductFactory (ProductFactory.php)

$factory->define(Product::class, function (Faker $faker)

{

            //Array values

            $sizeArrayValues = ['46','48','50','52'];

            $statusArrayValues = ['Publié','Brouillon'];

            $codeArrayValues = ['solde','new'];

        return [

            'Title' => $faker ->text($maxNbChars = 10),

            'Description'=> $faker->text($maxNbChars = 100),

            'Price'=> $faker->randomNumber(2),

            'Size'=> $sizeArrayValues[rand(0,3)],

            'url\_image'=> $faker->url,

            'Status'=> $statusArrayValues[rand(0,1)],

            'Code'=> $codeArrayValues[rand(0,1)],

            'Reference'=> $faker->text($maxNbChars = 10),

            'published\_at'=>now()

        ];

});

16. Remplissage du CategoryFactory (CategoryFactory.php)

$factory->define(Category::class, function (Faker $faker) {

    return [

        'Title' => $faker ->text($maxNbChars = 10),

        'Description'=> $faker->text($maxNbChars = 100),

    ];

});

17. Création du seeder « ProductSeeder » pour la table products

php artisan make:seeder ProductSeeder

18. Remplissage du ProductSeeder (ProductSeeder.php)

    public function run()

    {

        factory(App\Product::class, 8)->create();

    }

19. Création du seeder « CategorySeeder » pour la table categories

php artisan make:seeder CategorySeeder

20. Remplissage du CategorySeeder (CategorySeeder.php)

    public function run()

    {

        factory(App\Category::class, 3)->create();

    }

1. Injection de données dans la table products à l’aide du ProductSeeder

* λ php artisan db:seed --class=ProductSeeder

1. Injection de données dans la table categories à l’aide du CategorySeeder

* λ php artisan db:seed --class=CategorySeeder

1. Création d’un dépôt git (**A compléter !!!**)

git init

git add README.md

git commit -m "first commit"

git remote add origin <https://github.com/thcarole1/test2.git>

git push -u origin master

1. Réalisation d’un push sur github (provision : ne pas utiliser n’importe comment)

git remote add origin https://github.com/thcarole1/test2.git

git push -u origin master

1. Il n’est pas nécessaire de créer une factory ou un seeder pour la table ‘categories’ car elle ne contient que 2 catégories : ‘homme’ et ‘femme’.

On peut la créer ‘en dur’ au lancement du ProducSeeder :

    public function run()

    {

        //Création de 2 enregistrements (homme et femme) dans la table 'categories'

        App\Category::create([

            'Title' => 'homme',

            'Description' => 'Produits destinés aux hommes',

        ]);

        App\Category::create([

            'Title' => 'femme',

            'Description' => 'Produits destinés aux femmes',

        ]);

        factory(App\Product::class, 8)->create();

    }

}

1. Suppression de CategoryFactory (inutile)

(CategoryFactory.php)

1. <?php
2. /\*\* @var \Illuminate\Database\Eloquent\Factory $factory \*/
3. use App\Category;
4. use Faker\Generator as Faker;
5. $factory->define(Category::class, function (Faker $faker) {
6. $genderArrayValues = ['homme','femme'];
7. return [
8. // 'Title' => $faker ->text($maxNbChars = 10),
9. 'Title'=>$genderArrayValues[rand(0,1)],
10. 'Description'=> $faker->text($maxNbChars = 100),
11. ];
12. });

30. Suppression de CategorySeeder (inutile)

(CategorySeeder.php)

<?php

use Illuminate\Database\Seeder;

class CategorySeeder extends Seeder

{

    /\*\*

     \* Run the database seeds.

     \*

     \* @return void

     \*/

    public function run()

    {

        factory(App\Category::class, 2)->create();

    }

}

31. Ajout d’une colonne ‘Role’ dans la table ‘users’ afin de préciser le rôle des utilisateurs (client ou admin)

(2014\_10\_12\_000000\_create\_users\_table.php)

    public function up()

    {

        Schema::create('users', function (Blueprint $table) {

            $table->bigIncrements('id');

            $table->string('name');

            $table->string('email')->unique();

            $table->timestamp('email\_verified\_at')->nullable();

            $table->string('role');

            $table->string('password');

            $table->rememberToken();

            $table->timestamps();

        });

    }

32. Réalisation d’un push Github suite à

- Suppression de **CategoryFactory, CategorySeeder**

- Mise à jour de la table ‘users’ avec une colonne ‘role’ (string)

**Procédure à suivre :**

git init

git add .

git commit -m "Deletion of CategorySeeder, CategoryFactory ; users table updated with Role column"

git remote add origin <https://github.com/thcarole1/lamaison_repo.git>

git push -u origin master

33. Création du ProductController de type CRUD :

🡺 php artisan make:controller ProductController --resource

34. Lancement du serveur de développement LARAVEL

🡺 php artisan serve

35. Définition des routes dans le fichier routes/web.php

Exemple

Route::get('/', function () {

    return view('layouts.test');

});

36. Création d’un dossier views/layouts pour contenir ma page HTML générique dans laquelle je vais utiliser les directives blade « @yields »

37. Vider la database :

🡺 php artisan db :wipe

38. Il manque :

- afficher plusieurs tailles par article

**- la pagination**

**- utiliser le yield**

**- l’authentification**

**- Sur la fiche produit, modifier « boutique>soldes>homme »**

39. GITHUB

**Procédure à suivre :**

git init

git add .

git commit -m "Working Yield Directive, front views achieved"

git remote add origin <https://github.com/thcarole1/lamaison_repo.git>

git push -u origin master

40. Compléter

update.blade.php au niveau des boutons radios.

<form action="/action\_page.php" method="post">

41. GIT HUB

git init

git add .

git commit -m "Update of products function achieved, Creation of products function achieved, Deletion of products achieved, Dashboard achieved, back views achieved"

git remote add origin <https://github.com/thcarole1/lamaison_repo.git>

git push -u origin master

42. Reste à faire

- Téléchargement de ressources

- Authentification

- Mise en forme

43. GIT HUB

git init

git add .

git commit -m "Creation of product achieved, uploading function operationnal »

git remote add origin <https://github.com/thcarole1/lamaison_repo.git>

git push -u origin master

44. GIT HUB

git init

git add .

git commit -m "Uploading function operationnal – clean path generated"

git remote add origin <https://github.com/thcarole1/lamaison_repo.git>

git push -u origin master

45 Authentification LARAVEL

composer require laravel/ui –dev

php artisan ui vue –auth

npm install && npm run dev

46. VIDER LE CACHE

php artisan cache:clear