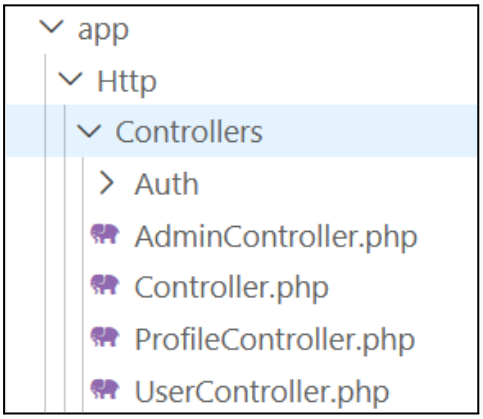
↳ **Logique interne du site: Codage des contrôleurs, middlewares, logique conditionnelle inter-vue à l'aide du pseudo langage *Blade*:**

• Une fois les vues et leurs templates importées et adaptées, la base de données configurée et étant interfacée avec notre environnement de travail, il nous faut traiter et répartir ces données – c'est pour cela qu'il nous faut programmer des contrôleurs.



• Il s'agit de la logique charnière du site, traitant et distribuant les données aux vues. Il s'agit de la majorité du code *backend* dans l'architecture MVC, y compris *Laravel*.

• Ces contrôleurs contiennent les méthodes, ces méthodes exécutent des requêtes de type "selection" et/ou "action" à la base de données grâce à un ORM (bien que des requêtes en 'raw' SQL sont possibles)

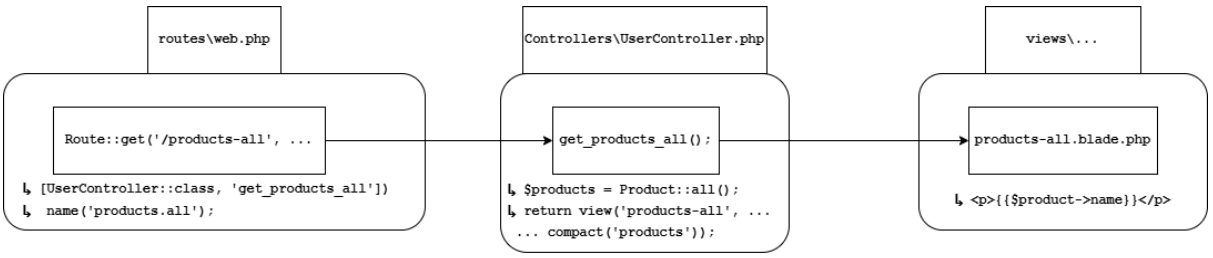


**Schéma illustrant une chaîne d'appel des méthodes aboutissant à l'affichage d'une vue:*

> *→ Dans un premier temps, une route est invoquée (en l'occurrence une route de type `get`) il est spécifié en deuxième argument de cette méthode de route la méthode que l'on souhaite appeler, et son contrôleur respectif*
>
> *→ Dans un second temps, la portée du programme est transférée à la méthode (en l'occurence `get_products_all()`). Y sont effectuées toute une variété d'opérations portant sur les tables de base de données grâce aux méthodes Eloquent. Dans notre cas, on souhaite récupérer tous les produits de la table éponyme pour les mettre dans une variable `$products`.*

→ Enfin, l'objet/variable `$products` est "compacté" pour être utilisé dans la vue vers laquelle il est redirigé. Les données compactées sont le plus souvent utilisées pour être affichées, tel est le cas pour la vue `products-all.blade.php` qui affiche le nom de chaque itération de `$products` (à partir d'une boucle `@foreach`).

→ Il arrive que la vue fasse elle-même un appel à une route à l'aide de la fonction du même nom `route()`. Si tel est le cas, il est fréquent que la méthode redirige vers cette même vue avec la fonction Eloquent `redirect()->back()`. Cela démontre que la portée du programme dans Laravel est constamment en mouvement et fait des "allers retours". Tel est par exemple cas pour la fonctionnalité d'ajout d'un produit au panier:

‣ *Vue `cart-products.blade.php`, c'est à partir de là qu'est lancée la "chaîne de redirections"*

```
@if ($product->quantity != 0)
    <a href="{{route('cart.add.product', $product->id)}}"
@endif
```

‣ *La portée du programme retourne à la "case départ", la vue qui à appelée la route.*

```
return redirect()->back()->with('cart_add_message', 'Produit ajouté au panier!');
```

• Les middlewares sont des contrôleurs spéciaux assignés à un groupement de routes dans l'objectif de définir des modalités d'accès.

```
class AdminMiddleware
{
    /**
     * Handle an incoming request.
     *
     * @param  \Closure(\Illuminate\Http\Request): (\Symfony\Component\HttpFoundation\Response)  $next
     */
    // Middleware Administratif: Définit la condition logique permettant de garantir la confidentialité des vues restreintes à un accès administratif (Voir son usage dans routes\web.php*)
    public function handle(Request $request, Closure $next)
    {
        if (Auth::check() && Auth::user()->user_type == "admin")
        {
            return $next($request);
        }
        // Une erreur de code 401 sera renvoyée
        abort(401, "Unauthorized Access");
        // route('login');
    }
}

```

**Middleware administratif servant à restreindre l'accès des utilisateurs normaux*

```
// Middleware Administratif: Seul un utilisateur dont la colonne 'user_type' a une valeur 'admin' peut accéder à ces routes (Voir AdminMiddleware.php*)
Route::middleware('admin')->group(function ()
{
    Route::get('/add-category', [AdminController::class, 'add_category'])->name('admin.add.category');
    Route::post('/add-category', [AdminController::class, 'post_add_category'])->name('admin.post.add.category');
```

[...]

```
    Route::get('/show-orders', [AdminController::class, 'show_orders'])->name('admin.show.orders');
    Route::post('/order-status/{id}', [AdminController::class, 'change_order_status'])->name('admin.order.status');
});
```

**L'usage de ce même middleware pour regrouper les routes administratives*

• Enfin le pseudo langage *Blade*, spécifique à *Laravel, permet* l'intégration de blocs logiques et l'usage de variables PHP directement au sein du code HTML des vues.

```
<td style="padding: 12px;">{{$cart_product->product->name}}</td>
<td style="padding: 12px;">{{Str::limit($cart_product->product->de
<td style="padding: 12px;">{{$cart_product->product->price}}€</td>
```

**Usage de variables PHP dans la vue d'affichage des produits du panier*

• Dans ce projet, les méthodes de contrôleurs sont nommées et implémentées en fonction de si elles traitent de requêtes de formulaires (POST), ou si elles exécutent une simple redirection vers une vue, ce qui permet un schéma de routage simple et concis.

• Le travail de requêtes est géré par la syntaxe *Eloquent*, qui elle-même est héritée de la syntaxe *QueryBuilder* – toutes ces méthodes sont syntaxiquement proches du paradigme POO, mais ont la même base logique que les requêtes SQL. Pour le reste, il s'agit de la logique conditionnelle traditionnelle de PHP. Tout comme les vues, les méthodes administratives sont réunies dans le `AdminController.php`, et les méthodes utilisatrices se trouvent dans `UserController.php`.

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use App\Models\User;
use App\Models\Product;
use App\Models\CartProduct;
use App\Models\OrderProduct;

class UserController extends Controller
{
    public function index()
    {
        $product_count = Product::count();
        $user_count = User::where('user_type', 'user')->count();
        $ordered_products_count = OrderProduct::where('status', 'pending')->count();
        $delivered_products_count = OrderProduct::where('status', 'delivered')->count();

        if (Auth::check() && Auth::user()->user_type=="user")
            return view('dashboard');
        else if (Auth::check() && Auth::user()->user_type=="admin")
            return view('admin.dashboard', compact('product_count', 'user_count', 'ordered_products_count', 'delivered_products_count'));
    }
```

**Méthode `index` du controller utilisateur, redirigeant au dashboard utilisateur ou administratif en fonction du statut de l'utilisateur actuellement connecté, vérifié à l'aide de la méthode `Auth::user()` de Breeze.*

```
<div class="row">
    @foreach ($products as $product)
        <div class="col-sm-6 col-md-4 col-lg-3">
            <div class="box hover-overlay" style="border-radius: 25px;">
                <a href="{{route('product.details', $product->id)}}" style="ba
                    <div class="img-box" style="padding-bottom: 0;">
                        <img src="{{asset('product_img/'.$product->image)}}">
                    </div>
                    <div class="detail-box">
                        {{$product->name}}<br><br>
                        Prix: {{$product->price}}€<br>
                        @if ($product->quantity != 0)
                            Quantité: x{{$product->quantity}}
                        @else
                            Quantité: En Rupture
                        @endif
                    </div>
                </a>
            </div>
        </div>
    @endforeach
```

**Extrait d'une vue faisant usage des directives logiques Blade (code dynamique) afin d'afficher le bon texte en fonction de la valeur de la colonne 'quantité' de l'itération du produit.*

• Une fois déclarés et leurs méthodes codées, ces contrôleurs sont systématiquement importés dans le/les fichier.s routes afin qu'il puissent opérer.

```
3   use App\Http\Controllers\AdminController;
4   use App\Http\Controllers\ProfileController;
5   use App\Http\Controllers\UserController;
6   use Illuminate\Support\Facades\Route;
```

**Importation des trois contrôleurs du projet dans le fichier de routage `web.php`, sans ceux-ci, aucune méthode ne pourrait être mobilisée lorsqu'une route est appelée.*