Analyse et conception

1. Analyse des besoins

+ Ajouter, modifier, supprimer, afficher les commandes, les clients et les produits.

+ Rechercher par mot clé produits par filtrage dynamique.

+ Imprimer une facture en format PDF.

2. Conception

A. Diagramme de classes

La figure 1 fait référence au diagramme de classes qui représente la structure conceptuelle du système et met en évidence les principales entités ainsi que leurs relations.

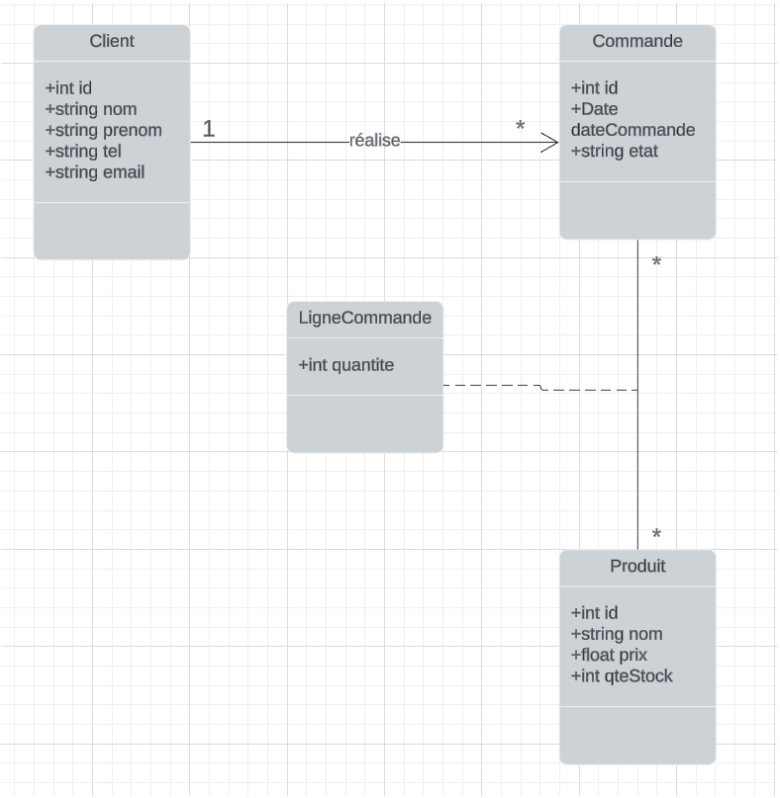


Figure 1. **Diagramme de classes**

- La classe Client représente les clients qui passent des commandes. Elle a comme attributs : id, nom, prenom, tel, email.
  Un client peut passer plusieurs commandes (relation 1 à * avec la classe Commande).

- La classe Commande représente une commande effectuée par un client. Elle a comme attributs : id, dateCommande, etat.
  Une commande est associée à un seul client mais peut contenir plusieurs produits via LigneCommande.

- La classe Produit représente un produit qui peut être commandé. Elle a comme attributs : id, nom, prix, qteStock (quantité en stock).
  Un produit peut apparaître dans plusieurs commandes via LigneCommande.

- La classe LigneCommande assure la liaison entre les produits et les commandes (relation plusieurs-à-plusieurs). Elle a comme attributs : quantite (nombre d'unités commandées pour un produit donné).
  Une commande peut contenir plusieurs produits. Un produit peut être présent dans plusieurs commandes.

B. Base de données

La figure 2 reflète la base de données (db_ecom) qui est structurée suivant le diagramme de classes précédemment réalisé. Les colonnes **created_at** et **updated_at** sont ajoutées automatiquement dans les migrations par Eloquent, elles sont mises à jour lors de l'insertion et de la modification d'un enregistrement.

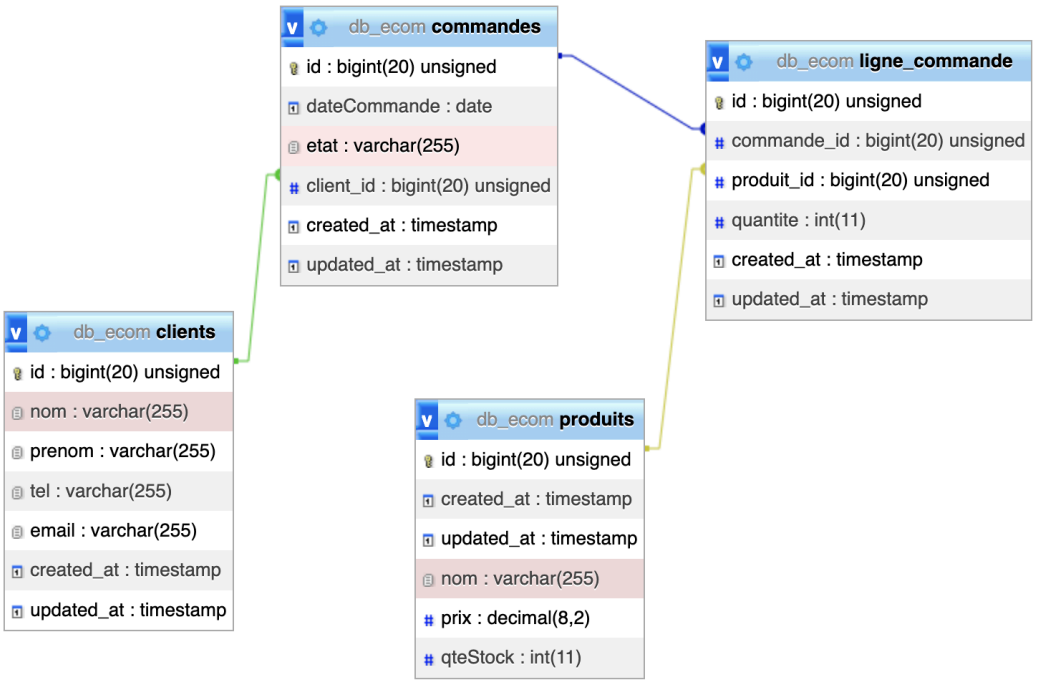


Figure 2. **Base de données (db_ecom)**

Table clients :

Contient les informations des clients avec les colonnes id, nom, prenom, tel, email,.

Table commandes :

Stocke les commandes passées par les clients avec id, dateCommande, etat, client_id (clé étrangère vers clients).

Table produits :

Contient les informations sur les produits disponibles à la vente avec id, nom, prix, qteStock.

Table ligne_commande :

Table pivot entre commandes et produits pour gérer la relation plusieurs à plusieurs.

Colonnes : id, commande_id (clé étrangère), produit_id (clé étrangère), quantite.

Réalisation

1. CRUD

Etape 1 : Créer le projet Laravel

Pour cela, ouvrir l'invite de commandes et exécuter la commande suivante pour avoir un projet prêt à l'emploi avec toutes les dépendances requises installées.:

```
composer create-project laravel/laravel ecom
```

composer : C'est un gestionnaire de dépendances pour PHP.

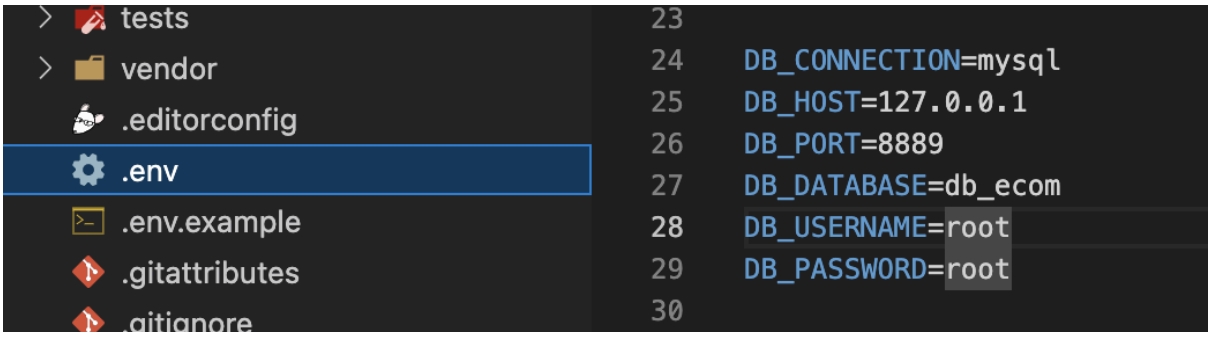
create-project : Une commande de Composer qui permet de créer un nouveau projet en téléchargeant les fichiers nécessaires depuis un dépôt.

laravel/laravel : C'est le package officiel de Laravel, qui contient l'ensemble du framework.

ecom : C'est le nom du dossier où le projet sera installé.

Etape 2 : Configuration de la base de données

Nous allons nous diriger vers le fichier .env qui se trouve au sein du projet pour spécifier les détail du serveur MySQL



On détermine le type de base de données (dans ce cas MySQL), aussi le port de connexion avec le nom de la base de données et enfin l'utilisateur et mot de passe de la base de données (dans ce cas, phpmyadmin est utilisé pour créer et gérer la base de données).

Etape 3 : Créer les modèles et migrations

Lors de cette étape, des modèles vont êtres créés pour la table clients, produits, commandes et en utilisant la commande Laravel. Exemple :

```
php artisan make:model Client -m
```

Cette commande créera un nouveau modèle Client et une migration correspondante pour la création de la table clients.

Après cette commande, il y aura deux fichiers créés, le premier dans le chemin **app/Models** et l'autre dans le chemin **database/migrations.** Les migrations définissent la structure des tables dans la base de données (colonnes, types de données, relations).
Les modèles permettent d'interagir avec ces tables à travers Eloquent, l'ORM de Laravel.

Voici la migration de la table client

```php
/**
    * Run the migrations.
    */
   public function up(): void
   {
       Schema::create('clients', function (Blueprint $table) {
           $table->id();
           $table->string('nom');
           $table->string('prenom');
           $table->string('tel');
           $table->string('email');
           $table->timestamps();
       });
   }
   /**
    * Reverse the migrations.
```

```
     */
    public function down(): void
    {
        Schema::dropIfExists('clients');
    }
```

Voici la migration de commandes

```
/**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('commandes', function (Blueprint $table) {
            $table->id();
            $table->date('dateCommande');
            $table->string('etat');
            $table->foreignId('client_id')->constrained()->onDelete('cascade');
            $table->timestamps();
        });
    }
    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('commandes');
    }
```

Voici la migration de la table Produits

```php
/**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('produits', function (Blueprint $table) {
            $table->id();
            $table->timestamps();
            $table->string('nom');
            $table->decimal('prix', 8, 2)->default(0);
            $table->integer('qteStock')->default(0);
        });
    }
    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('produits');
    }
```

Et enfin la migration de la table ligne_commande qui représente la table pivot entre commandes et produits

```php
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('ligne_commande', function (Blueprint $table) {
```

```
            $table->id();
            $table->foreignId('commande_id')->constrained()->onDelete('cascade');
            $table->foreignId('produit_id')->constrained()->onDelete('cascade');
            $table->integer('quantite');
            $table->timestamps();
        });
    }
    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('ligne_commande');
    }
```

Lorsque toutes les migrations sont prêtes, cette commande est exécutée afin de structurer la base de données:

`php artisan migrate`

```
montassiffati@MBPdeMontassif ecom % php artisan migrate
   INFO  Running migrations.
   2025_01_24_105248_create_ligne_commande ..... 142.76ms DONE
```

Voici le modèle de client

```
<?php

namespace App\Models;
use Illuminate\Database\Eloquent\Model;

class Client extends Model
```

```
{
    //
    protected $fillable = ["nom", "prenom", "tel", "email"];
    public function commandes()
    {
        return $this->hasMany(Commande::class);
    }
}
```

Voici le modèle de commande

```
<?php

namespace App\Models;
use Illuminate\Database\Eloquent\Model;
class Commande extends Model
{
    //
    protected $fillable = ["dateCommande", "etat", "client_id"];
    public function client()
    {
        return $this->belongsTo(Client::class);
    }
    public function produits()
    {
        return $this->belongsToMany(Produit::class,
'ligne_commande')->withPivot('quantite');
    }
}
```

Et voici le modèle de produit

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class Produit extends Model

{

   //

   protected $fillable = ["nom", "prix", "qteStock"];

   public function commandes()

   {

                                    return     $this->belongsToMany(Commande::class,
'ligne_commande')->withPivot('quantite');

   }

}
```

Comme le précise les modèles, la relation entre client et commande est de un à plusieurs (c'est pour cela qu'on utilise belongsTo) un client a plusieurs commandes.

La relation entre commandes et produits et de plusieurs à plusieurs (belongsToMany), les deux tables sont reliées par une table pivot nommée 'ligne_commande' et a comme attribut 'quantite' . Comme représenté dans le diagramme de classe précédemment.

Etape 4 : Ajouter les routes de ressources

Pour ajouter une route, il faut accéder au fichier "routes/web.php" et ajoutez les routes suivantes

```php
<?php

use App\Http\Controllers\ClientController;

use App\Http\Controllers\CommandeController;

use App\Http\Controllers\ProductController;

use Illuminate\Support\Facades\Route;


```

```php
Route::resource('clients', ClientController::class);

Route::resource('produits', ProductController::class);

Route::resource('commandes', CommandeController::class);

Route::get('/search/produits',[ProductController::class,
'search'])->name('produits.search');

Route::get('/facture/{id}',[CommandeController::class,
'facture'])->name('commandes.facture');
```

Les 3 premières routes créent automatiquement **toutes les routes CRUD (Create, Read, Update, Delete)** pour les ressources `clients, produits` et `commandes.` Cela permet d'éviter d'écrire manuellement toutes les routes.

La route "produits.search" concerne la recherche dynamique des produits. Et la route "commandes.facture" concerne l'affichage de la facture d'une commande donnée.

Etape 5 : Ajouter les contrôleurs

Tous les contrôleurs créés se trouvent dans le dossier avec le chemin "app/Http/Controllers/" qui contient sept méthodes qui sont listées ci-dessous

1)index() 2)create() 3)store() 4)show() 5)edit() 6)update() 7)destroy()

Voici le contrôleur du client

```php
<?php


namespace App\Http\Controllers;

use App\Models\Client;

use Illuminate\Http\Request;


class ClientController extends Controller

{

   //la fonction index permet d'afficher la liste de tous  les clients

   public function index(){
```

```
        $clients = Client::all();

        return View ("cclients.index", compact('clients'));

    }

    //la fonction show permet d'afficher les détails de chaque client suivant son id

    public function show ($id){

        $client = Client::find($id);

        return View ("cclients.show", compact('client'));

    }

    //la fonction create affiche un formulaire  pour créer un nouveau client

    public function create(){

        return View ("cclients.create");

    }

       //la fonction store ajoute le client créé à partir du formulaire dans la base
de données avec le modèle Client

    public function store(Request $request){

        Client::create($request->all());

        return redirect()->route ('clients.index');

    }

    //la fonction edit affiche un formulaire  pour modifier un client existant

    public function edit ($id){

        $client = Client::find($id);

        return View ("cclients.edit", compact('client'));

    }

        //la fonction update modifie le client existant avec les informations
du formulaire

    public function update (Request $request, $id){

        $client = Client::find($id);

        $client->update($request->all());
```

```
        return redirect()->route ('clients.index');

    }

    //la fonction destroy supprime un client choisi par l'utilisateur

    public function destroy ($id){

        $client = Client::find($id);

        $client->delete();

        return redirect()->route ('clients.index');

    }

}
```

Voici le contrôleur de produit

```
<?php


namespace App\Http\Controllers;

use App\Models\Produit;

use Illuminate\Http\Request;


class ProductController extends Controller

{

    //

    public function index(){

        $produits = Produit::all();

        return View ("produits.index", compact('produits'));

    }

    public function show ($id){

        $produit = Produit::find($id);

        return View ("produits.show", compact('produit'));

    }
```

```
    public function create(){
        return View ("produits.create");
    }

    public function store(Request $request){
        Produit::create($request->all());
        return redirect()->route ('produits.index');
    }

    public function edit ($id){
        $produit = Produit::find($id);
        return View ("produits.edit", compact('produit'));
    }

    public function update (Request $request, $id){
        $produit = Produit::find($id);
        $produit->update($request->all());
        return redirect()->route ('produits.index');
    }

    public function destroy ($id){
        $produit = Produit::find($id);
        $produit->delete();
        return redirect()->route ('produits.index');
    }
}
```

Et voici celui de commandes

```php
<?php


namespace App\Http\Controllers;


use App\Models\Client;

use App\Models\Commande;

use App\Models\Produit;

use Illuminate\Http\Request;


class CommandeController extends Controller

{

   //

   public function index(){

       $commandes = Commande::all();

       return View ("commandes.index", compact('commandes'));

   }


   public function show ($id){

       $commande = Commande::find($id);

       return View ("commandes.show", compact('commande'));

   }


   public function create(){

       $clients = Client::all();

       $produits = Produit::all();

       return View ("commandes.create", compact('clients', 'produits'));
```

```
    }

    public function store(Request $request){
        $dateCommande = date('Y-m-d', strtotime($request->dateCommande));
        $commande = Commande::create([
            'dateCommande' => $dateCommande,
            'etat' => $request->etat,
            'client_id' => $request->client_id,
        ]);

        foreach ($request->produits as $produit) {
                            if (is_array($produit) && isset($produit['id']) &&
!is_null($produit['quantite'])) {
                        $commande->produits()->attach($produit['id'], ['quantite' =>
$produit['quantite']]);
            }
        }
        return redirect()->route ('commandes.index');
    }

    public function edit ($id){
        $commande = Commande::find($id);
        $clients = Client::all();
        $produits = Produit::all();
        return View ("commandes.edit", compact('commande', 'clients', 'produits'));
    }

    public function update (Request $request, $id){
```

```
        $commande = Commande::find($id);

        $dateCommande = date('Y-m-d', strtotime($request->dateCommande));



        $commande -> update([

            'dateCommande' => $dateCommande,

            'etat' => $request->etat,

            'client_id' => $request->client_id,

        ]);



        $commande->produits()->detach();

        foreach ($request->produits as $produit) {

                            if   (is_array($produit)   &&   isset($produit['id'])   &&
!is_null($produit['quantite'])) {

                        $commande->produits()->attach($produit['id'], ['quantite' =>
$produit['quantite']]);

            }

        }

        return redirect()->route ('commandes.index');

    }



    public function destroy ($id){

        $commande = Commande::find($id);

        $commande->produits()->detach();

        $commande->delete();

        return redirect()->route ('commandes.index');

    }

}
```

Etape 6 : Créer des vues Blade

Toutes les vues se trouvent dans le dossier **resources/views.**

Tout d'abord, nous allons créer un fichier **app.blade.php** dans le dossier **layouts** qui a pour but de structurer l'application. Il y a le header avec une navbar et le footer qui resteront fixes pour toutes les pages de l'application, seuls le titre et la partie principale qui changera suivant la page actuelle. C'est pour cela **@yield** a été utilisé comme `@yield('title')`

```
<!DOCTYPE html>

<html lang="en">

<head>

   <meta charset="UTF-8">

   <meta name="viewport" content="width=device-width, initial-scale=1.0">

   <meta http-equiv="X-UA-Compatible" content="ie=edge">

                                                                                <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
rel="stylesheet"
integrity="sha384-QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwIH"
crossorigin="anonymous">

                                                <link                rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.5.0/font/bootstrap-icons.css"
/>

   <title>@yield('title')</title>

</head>

<body class="container d-flex flex-column min-vh-100">

   <header>

       <nav class="mb-4 navbar navbar-expand-lg bg-info rounded">

           <div class="container-fluid">

             <a class="navbar-brand" href="/">CRUD</a>

              <button class="navbar-toggler" type="button" data-bs-toggle="collapse"
data-bs-target="#navbarNav"      aria-controls="navbarNav"      aria-expanded="false"
aria-label="Toggle navigation">

               <span class="navbar-toggler-icon"></span>
```

```
            </button>

            <div class="collapse navbar-collapse" id="navbarNav">

              <ul class="navbar-nav">

                <li class="nav-item">

                      <a class="nav-link {{ request()->routeIs('clients.index') ?
'active' : '' }}" aria-current="page" href="{{route('clients.index')}}">Clients</a>

                </li>

                <li class="nav-item">

                     <a class="nav-link {{ request()->routeIs('produits.index') ?
'active' : '' }}" href="{{route('produits.index')}}">Produits</a>

                </li>

                <li class="nav-item">

                    <a class="nav-link {{ request()->routeIs('commandes.index') ?
'active' : '' }}" id="lien3" href="{{route('commandes.index')}}">Commandes</a>

                </li>

              </ul>

            </div>

          </div>

        </nav>

   </header>



   <main>

       @yield('content')

   </main>



   <div class="container mt-auto">

       <footer class="py-3 my-4">

         <ul class="nav justify-content-center border-bottom pb-3 mb-3">
```

```
                        <li  class="nav-item"><a  href="{{route('clients.index')}}"
class="nav-link px-2 text-muted">Clients</a></li>

                       <li  class="nav-item"><a  href="{{route('produits.index')}}"
class="nav-link px-2 text-muted">Produits</a></li>

                      <li  class="nav-item"><a  href="{{route('commandes.index')}}"
class="nav-link px-2 text-muted">Commandes</a></li>

        </ul>

        <p class="text-center text-muted">© 2025 HESTIM, Inc</p>

      </footer>

   </div>



</body>

</html>
```

Ensuite, 3 dossiers ont été créés dans views (Cclients, Commandes et Produits). Dans chacun de ces trois dossiers, il y a 4 fichiers qui sont accessibles avec la méthode **GET**:

create.blade.php : affiche le formulaire pour créer un nouveau client, un nouveau produit ou une nouvelle commande.

edit.blade.php : affiche le formulaire pour modifier les informations existantes d'un client, un produit ou une commande.

index.blade.php : affiche la liste de tous les clients, les produits ou commandes avec les boutons pour supprimer et modifier.

show.blade.php : affiche les détails d'un élément choisi, soit un client, un produit ou une commande.

Le dossier Cclients, fichier create

```
@extends('layouts.app')

@section('title', 'Ajouter client')

@section('content')

   <form method="POST" action="{{route('clients.store')}}" class="container py-4">

      @csrf
```

```
        <div class="mb-3">
            <label for="nom" class="form-label">Nom</label>
                    <input  type="text"  name="nom"  class="form-control"  id="nom"
aria-describedby="nom">
        </div>
        <div class="mb-3">
            <label for="prenom" class="form-label">Prénom</label>
                 <input  type="text"  name="prenom"  class="form-control"  id="prenom"
aria-describedby="prenom">
        </div>
        <div class="mb-3">
            <label for="tel" class="form-label">Téléphone</label>
                    <input  type="text"  name="tel"  class="form-control"  id="tel"
aria-describedby="tel">
        </div>
        <div class="mb-3">
            <label for="email" class="form-label">Email</label>
                 <input  type="email"  name="email"  class="form-control"  id="email"
aria-describedby="email">
        </div>
        <button type="submit" class="btn btn-primary">Submit</button>
    </form>
@endsection
```

Le fichier edit

```
@extends('layouts.app')
@section('title', 'Ajouter client')
@section('content')
    <form method="POST" action="/clients/{{$client['id']}}" class="container py-4">
        @csrf
```

```
        @method('PUT')

        <div class="mb-3">

            <label for="nom" class="form-label">Nom</label>

                        <input  type="text"  name="nom"  value="{{$client['nom']}}"
class="form-control" id="nom" aria-describedby="nom">

        </div>

        <div class="mb-3">

            <label for="prenom" class="form-label">Prénom</label>

                    <input  type="text"  name="prenom"  value="{{$client['prenom']}}"
class="form-control" id="prenom" aria-describedby="prenom">

        </div>

        <div class="mb-3">

            <label for="tel" class="form-label">Téléphone</label>

                        <input  type="text"  name="tel"  value="{{$client['tel']}}"
class="form-control" id="tel" aria-describedby="tel">

        </div>

        <div class="mb-3">

            <label for="email" class="form-label">Email</label>

                    <input  type="email"  name="email"  value="{{$client['email']}}"
class="form-control" id="email" aria-describedby="email">

        </div>

        <button type="submit" class="btn btn-warning">Modifier</button>

    </form>

@endsection
```

Le fichier index

```
@extends('layouts.app')

@section('title', 'Liste clients')

@section('content')

    <h3>Liste des clients</h3>
```

```
   <table class="table">

       <thead>

           <tr>

               <th scope="col">ID</th>

               <th scope="col">Nom</th>

               <th scope="col">Prénom</th>

               <th scope="col">Tel</th>

               <th scope="col">Email</th>

               <th scope="col">Actions</th>

           </tr>

       </thead>

       <tbody>

           @foreach ($clients as $client)

               <tr>

                   <th scope="row">{{$client['id']}}</th>

                   <td>{{$client['nom']}}</td>

                   <td>{{$client['prenom']}}</td>

                   <td>{{$client['tel']}}</td>

                   <td>{{$client['email']}}</td>

                   <td>

                           <a class="btn btn-success" href="{{ route('clients.show',
$client->id) }}"><i class="bi bi-search"></i></a>

                            <a class="btn btn-warning" href="{{route('clients.edit',
$client->id)}}"><i class="bi bi-pencil-square"></i></a>

                          <form action="{{ route('clients.destroy', $client->id) }}"
method="POST" style="display: inline;">

                            @csrf

                            @method('DELETE')
```

```
                                    <button class="btn btn-danger" type="submit"
onclick="return confirm('Êtes-vous sûr de vouloir supprimer ce client ?')"><i
class="bi bi-trash"></i></button>

                        </form>



                    </td>

                </tr>

            @endforeach

        </tbody>

    </table>

    <a href="{{route('clients.create')}}" class="btn btn-info">Nouveau client</a>

@endsection
```

puis le fichier show

```
@extends('layouts.app')

@section('title', 'Liste clients')

@section('content')

    <h3>Informations du client</h3>

    <div>

        <h4>Le nom prénom est {{$client['nom']}} {{$client['prenom']}}</h4>

        <h5>L'email est {{$client['email']}}</h5>

        <p>Le tel est {{$client['tel']}}</p>

    </div>

    <a href="{{route('clients.index')}}" class="text-success">Retour</a>

@endsection
```

Le dossier produits, fichier index

```
@extends('layouts.app')

@section('title', 'Liste produits')

@section('content')
```

```
    <div class="my-3 d-flex justify-content-between">
        <span class="h3">Liste des produits</span>
    </div>
    <table class="table mt-3">
        <thead>
            <tr>
                <th scope="col">ID</th>
                <th scope="col">Nom</th>
                <th scope="col">Prix</th>
                <th scope="col">Quantité Stock</th>
                <th scope="col">Actions</th>
            </tr>
        </thead>

        <tbody id="products">
            @foreach ($produits as $produit)
                <tr>
                    <th scope="row">{{$produit['id']}}</th>
                    <td>{{$produit['nom']}}</td>
                    <td>{{$produit['prix']}}</td>
                    <td>{{$produit['qteStock']}}</td>
                    <td>
                        <a class="btn btn-success" href="{{ route('produits.show', $produit->id) }}"><i class="bi bi-search"></i></a>
                        <a class="btn btn-warning" href="{{ route('produits.edit', $produit->id) }}"><i class="bi bi-pencil-square"></i></a>

                        <form action="{{ route('produits.destroy', $produit->id)}}" method="POST" style="display: inline;">
```

```
                        @csrf

                        @method('DELETE')

                                    <button class="btn btn-danger" type="submit"
onclick="return confirm('Êtes-vous sûr de vouloir supprimer cette commande ?')"><i
class="bi bi-trash"></i></button>

                    </form>


                </td>

            </tr>

        @endforeach


    </tbody>

</table>

<a href="{{route('produits.create')}}" class="btn btn-info">Nouveau produit</a>

@endsection
```

Le dossier commandes, fichier index

```
@extends('layouts.app')

@section('title', 'Liste des Commandes')



@section('content')

<h3>Liste des Commandes</h3>

<table class="table mt-3 text-center">

    <thead>

        <tr>

            <th>ID</th>

            <th>Client</th>

            <th>Date</th>

            <th>État</th>
```

```
                <th>Produits commandés</th>
                <th>Actions</th>
            </tr>
        </thead>
        <tbody>
            @foreach ($commandes as $commande)
                <tr>
                    <td>{{ $commande->id }}</td>
                       <td>{{ $commande->client->nom }} {{ $commande->client->prenom
}}</td>
                    <td>{{ $commande->dateCommande }}</td>
                    <td>{{ $commande->etat }}</td>
                    <td>
                            <table class="table">
                                <thead>
                                    <tr>
                                        <th class="text-primary">Nom</th>
                                        <th class="text-primary">Quantité</th>
                                    </tr>
                                </thead>
                                <tbody>
                                    @foreach ($commande->produits as $produit)
                                        <tr>
                                           <td>{{ $produit-> nom }}</td>
                                           <td>{{ $produit->pivot-> quantite }}</td>
                                        </tr>
                                    @endforeach
```

```
                            </tbody>

                        </table>



                </td>

                <td>

                        <a href="{{ route('commandes.facture', $commande->id) }}"
target="_blank" class="btn btn-primary"><i class="bi bi-printer"></i></a>

                          <a href="{{ route('commandes.show', $commande->id) }}"
class="btn btn-success"><i class="bi bi-search"></i></a>

                          <a href="{{ route('commandes.edit', $commande->id) }}"
class="btn btn-warning"><i class="bi bi-pencil-square"></i></a>

                        <form action="{{ route('commandes.destroy', $commande->id)
}}" method="POST" class="d-inline">

                          @csrf

                          @method('DELETE')

                                    <button class="btn btn-danger" type="submit"
onclick="return confirm('Êtes-vous sûr de vouloir supprimer ce client ?')"><i
class="bi bi-trash"></i></button>

                    </form>

                </td>

            </tr>

        @endforeach

      </tbody>

  </table>

    <a href="{{ route('commandes.create') }}" class="btn btn-info">Créer une
Commande</a>

@endsection
```

N.B : Pour ne pas prendre beaucoup de place dans le rapport, les autres fichiers des dossiers produits et commandes n'ont pas été montrés dans le rapport. Veuillez voir la vidéo pour le résultat final ou le lien Github pour voir plus de code.

2. Recherche dynamique des produits

La recherche dynamique sera insérée au niveau de la page index qui affiche tous les produits

D'abord, il faut ajouter ajouter une fonction au niveau du ProductController

```
public function search(Request $request){

       $query = $request->input('query', '');

       if ($query === '') {

           $produits = Produit::all();

       } else {

           $produits = Produit::where('nom', 'LIKE', '%' . $query . '%')->get();

       }



       return response()->json($produits);

   }
```

Ensuite, ajouter au niveau de la page index de produits, l'input pour saisir le produit voulu, dès que l'utilisateur commence à saisir, la fonction searchProducts() est déclenchée : le script permet de prendre la valeur de l'input et le mettre dans la constante query, cette valeur va être envoyé vers le lien /search/produits qui déclenche la fonction search se trouvant dans le contrôleur. Si l'input est vide, tous les produits sont affichés, sinon une requête est envoyée à l'aide du modèle pour trouver les produits correspondant à ce que cherche l'utilisateur. Ces derniers sont envoyés en format JSON pour ensuite les afficher avec le script.

```
<div class="my-3 d-flex justify-content-between">

       <span class="h3">Liste des produits</span>

                   <input   type="text"   id="search"   class="form-control   w-25"
placeholder="Rechercher un produit" oninput="searchProducts()">

        {{-- The oninput event occurs when the value of an <input> or <textarea> or
<select> element is changed. --}}

   </div>
```

```
<script>

       function searchProducts(){

           const query = document.getElementById('search').value;



           fetch(`/search/produits?query=${query}`)

               .then(response => {

                   if (!response.ok) {

                       throw new Error(`HTTP error! status: ${response.status}`);

                   }

                   return response.json();

               })

               .then(data => {

                   const tableBody = document.getElementById('products');

                   tableBody.innerHTML = '';



                   data.forEach(produit => {

                       tableBody.innerHTML += `

                           <tr>

                               <th scope="row">${produit.id}</th>

                               <td>${produit.nom}</td>

                               <td>${produit.prix}</td>

                               <td>${produit.qteStock}</td>

                               <td>

                                                   <a class="btn btn-success"
href="/produits/${produit.id}"><i class="bi bi-search"></i></a>

                                                   <a class="btn btn-warning"
href="/produits/${produit.id}/edit"><i class="bi bi-pencil-square"></i></a>

                                               <form action="/produits/${produit.id}"
method="POST" style="display: inline;">
```

```
                                    @csrf
                                    @method('DELETE')
                                     <button class="btn btn-danger" type="submit"
onclick="return confirm('Êtes-vous sûr de vouloir supprimer ce produit ?')">
                                        <i class="bi bi-trash"></i>
                                    </button>
                                </form>
                            </td>
                        </tr>
                    `;
                });
            })

            .catch(error => {
                    console.error('Erreur lors de la recherche des produits :',
error);
            });
    }
  </script>
```

3. Impression de facture PDF

```
public function facture ($id){
    $commande = Commande::find($id);
    $total = 0;
    foreach ($commande->produits as $produit) {
        $total += $produit->pivot->quantite * $produit->prix;
    }

    $pdf = Pdf::loadView('commandes.facture', compact('commande', 'total'))
```

```
            ->setPaper('a4')

                ->setOptions(['isHtml5ParserEnabled' => true, 'isRemoteEnabled' =>
true]);

        return $pdf->stream("facture_commande_{$commande->id}.pdf");

    }



<!DOCTYPE html>

<html lang="en">

<head>

    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">

    <title>Facture {{ $commande->id }}</title>

    <style>

        html,

        body {

            margin: 10px;

            padding: 10px;

            font-family: sans-serif;

        }

        h1,h2,h3,h4,h5,h6,p,span,label {

            font-family: sans-serif;

        }

        table {

            width: 100%;

            border-collapse: collapse;

            margin-bottom: 0px !important;

        }

        table thead th {

            height: 15px;
```

```
        text-align: left;
        font-size: 16px;
    }
    table, th, td {
        padding: 15px;
        font-size: 14px;
    }
    .total-heading {
        font-size: 18px;
        font-weight: 700;
        font-family: sans-serif;
    }
    .border{
        border-bottom: #5C6AC4 solid 3px;
    }
    .text-end {
        text-align: right;
    }
    .bg-blue {
        background-color: #5C6AC4;
        color: #fff;
    }
    .product th{
        color: #5C6AC4;
    }
    .main{
        padding: 50px;
        border-bottom: #EBEBED 1px solid;
```

```
        }
    </style>
</head>
<body>


    <table style="width: 100%; border-collapse: collapse;">
        <thead>
            <tr>
                <th style="width: 50%;">
                    <h2 style="font-size: 20px;">CRUD Ecommerce</h2>
                </th>
                <th style="width: 50%; text-align: right; font-size: 14px;">
                    <div>
                        <span style="display: block; font-weight: bold;">Facture Id: #{{ $commande->id }}</span>
                        <span style="color: #AFB3C1;">Date: {{ $commande->dateCommande }}</span>
                    </div>
                </th>
            </tr>
        </thead>
    </table>


<div style="background-color: #F1F7F9; padding: 10px; margin-bottom: 20px;">
      <div style="width: 100%; padding: 10px; font-family: Arial, sans-serif; border-radius: 5px;">
            <div style="display: inline-block; width: 48%; vertical-align: top; text-align: left;">
             <h6 style="margin: 0; font-size: 16px; font-weight: bold;">Informations client</h6>
```

```
            <p style="margin: 5px 0 0 0; font-size: 14px; line-height: 1.6; color:
#6c757d;">

                {{ $commande->client->nom }} {{ $commande->client->prenom }}<br>

                {{ $commande->client->tel }}<br>

                {{ $commande->client->email }}<br>



            </p>

        </div>

            <div style="display: inline-block; width: 48%; vertical-align: top;
text-align: right;">

            <h6 style="margin: 0; font-size: 16px; font-weight: bold;">Informations
commande</h6>

            <p style="margin: 5px 0 0 0; font-size: 14px; line-height: 1.6; color:
#6c757d;">

                Date: {{ $commande->dateCommande }}<br>

                État: {{ $commande->etat }}<br>

            </p>

        </div>

    </div>

</div>




    <table class="product">

        <thead>

            <tr class="border">

                <th>ID</th>

                <th>Details produit</th>

                <th>Prix</th>

                <th>Quantité</th>
```

```
                <th>Sous-total</th>

            </tr>

        </thead>

        <tbody>

            @foreach ($commande->produits as $produit)

                <tr class="main">

                    <td width="10%">{{ $produit-> id }}.</td>

                    <td>

                        {{ $produit-> nom }}

                    </td>

                    <td width="20%">{{ $produit-> prix }} DHS</td>

                    <td width="10%">{{ $produit->pivot-> quantite }}</td>

                      <td width="20%" class="fw-bold">{{ $produit->pivot-> quantite *
$produit-> prix }} DHS</td>

                </tr>

            @endforeach

        </tbody>

        <tfoot>

            <tr>

                <td colspan="3"></td>

                <td colspan="1" class="bg-blue text-end total-heading">Total :</td>

                <td colspan="1" class="bg-blue total-heading">{{ $total }} DHS</td>

            </tr>

        </tfoot>

    </table>
```

```
    <div style="width: 80%; font-family: Arial, sans-serif; margin: auto; font-size:
12px; color: #6c757d; text-align: center; position: absolute; bottom: 0; left: 50%;
transform: translateX(-50%);">

        <hr style="border: 0.5px solid #e9ecef; margin-bottom: 10px;">

        <p style="margin: 0;">

            CRUD Ecommerce | n°10 Bd zahi, 20230, Casablanca<br>

                Phone: +212 643 906 090 | Email: info@crudecommerce.com | Website:
www.crudecommerce.com

        </p>

    </div>



</body>

</html>
```

Dans la page index des commandes, chaque ligne dispose d'un bouton **Imprimer** pour visualiser la facture. La fonction **facture($id)** récupère la commande correspondant à l'ID fourni puis calcul le total en parcourant tous les produits associés et en multipliant la quantité par le prix.

Ensuite, la vue **commandes.facture** est chargée avec les données commande et total en format A4. Le fichier est généré et affiché directement dans le navigateur avec la méthode **stream**. Bien sûr, le fichier peut être téléchargé par la suite si besoin.