

BTS SIO 2024

☐

Administration des systèmes et des réseaux (E5 – SISR)

☒

Conception et développement d'applications (E5 – SLAM)

PAGE DE PRÉSENTATION DU DOSSIER

N° d'inscription¹ : | 0 | 2 | 3 | 4 | 2 | 4 | 3 | 9 | 4 | 4 | 4 |

NOM : BIR

PRENOM : Mohand Amokrane

date de passage¹ :/...../2024

Heure de passage¹ :h.....

CATEGORIE CANDIDAT² (UNE CASE A COCHER)

☐ Scolaire

☒ Apprenti

☐ Formation professionnelle continue

☐ Expérience professionnelle 3 ans

☐ Ex-scolaire

☐ Ex-apprenti

☐ Ex-formation professionnelle continue

¹ Informations communiquées sur votre convocation envoyée courant mars 2024 sur votre compte **Cyclades**

² Informations communiquées sur votre confirmation d'inscription.

Tampon de L'établissement

SIEC – maison des examens

7 rue Ernest Renan
94749 ARCUEIL CEDEX
Tél : 01 49 12 23 00
www.siec.education.fr



Sommaire

Introduction générale

CHAPITRE 1 : CADRE GENERAL DU PROJET ET SPECIFICATIONS DES BESOINS

1.Introduction

2. Présentation du projet

3. Choix des méthodes, conception et modélisation

4. Spécification des besoins

4.1. Besoins fonctionnels :

4.2. Besoins non fonctionnels :

CHAPITRE 1 : MODELISATION CONCEPTUELLE (UML)

1.Introduction

2.Diagramme de cas d'utilisation

3.Diagramme de classe

3.Diagramme de séquences

CHAPITRE 3 : REALISATION TECHNIQUE

1.Introduction

2.Framwork LARAVEL 10

3.Environnement de développement IDE

4. Serveur local

5.Description de développement et test d'interfaçage

Conclusion générale

Introduction générale

Le e-commerce ou commerce électronique regroupe l'ensemble des transactions commerciales s'opérant à distance par le biais d'interfaces électroniques et digitales. Il est devenu le principal canal de la vente ce qui explique le remplacement du terme de "vente par correspondance" par celui de "vente à distance". Il englobe essentiellement les transactions commerciales s'effectuant sur Internet à partir des différents types de terminaux sur des sites e-commerce ou applications mobiles marchandes.

Ce type d'applications représente un système mondial, offrant aux commerçants un pont pour accéder à toutes les informations, produits et services liés à un portail unique. Un site de vente en ligne permet aux commerçants de profiter des salons virtuels disponibles, qui sont mis à jour quotidiennement sans aucune restriction, ce qui leur permettra de ne jamais manquer de clients, donc il n'y a pas de distance de travail géographique ou de distance par rapport au calendrier de travail équitable ou à la disponibilité du transport. En revanche, ces sites offrent à l'entreprise l'opportunité de profiter de cet espace et de présenter ses produits à un groupe de commerçants plus large.

Dans ce contexte et dans le cadre de mon projet de fin d'études, je vise la mise en place d'une application web B2C que je dénoterai "Shop", une boutique en ligne de fruits et légumes organiques.

Le présent rapport permet de détailler les différentes étapes que j'ai suivies lors du développement de projet. Il se compose de trois chapitres. Dans le premier chapitre, je commence par présenter le cadre général du projet, saisir les besoins fonctionnels et non fonctionnels, ainsi que le cycle de développement adopté pour la mise en œuvre de mon projet. Le deuxième chapitre est consacré à la modélisation conceptuelle globale sur laquelle se base le projet.

Le troisième et quatrième chapitre, présentent l'étude technique tout en déterminant les technologies et outils utilisés, et démonstration de code et l'interfaçage de l'application par des captures significatives.

CHAPITRE 1 : CADRE GENERAL DU PROJET ET SPECIFICATIONS DES BESOINS

1. Introduction

Dans le premier chapitre, je m'intéresse d'abord à l'introduction du cadre global du projet et à la description des exigences. Il s'agit en fait de suivi et de la définition de la tâche, de la présentation du projet et des objectifs à atteindre.

2. Présentation du projet

Mon projet consiste alors à réaliser une application Web permettant aux marchand de présenter ces produit et ces offres commerciales aux clients avec un accès administrateur à l'application, et aussi de donner la main aux utilisateurs clients d'effectuer un achat en ligne, après avoir remplir leurs paniers.

Ce travail rentre dans le cadre de projet web de l'épreuve E5 qui vient conclure ma formation du BTS SIO (Services informatiques aux organisations), option SLAM (solutions logicielles et applications métiers).

3. Choix des méthodes, conception et modélisation

Dans mon projet, j'ai choisi la méthode de conception orientée objet, où La modélisation d'objets comprend la création de modèles informatiques qui constituent l'application, et afin de modéliser cette conception de type orientée objet, j'ai opté au formalisme UML (Unified Modelling Language) qui imposé comme un outil très performant de modélisation.

4. Spécification des besoins

4.1. Besoins fonctionnels :

La création d'une application web passe par l'élaboration de deux parties, la première partie concerne la création et la mise en place des pages accessibles par

tous les internautes (Front Office).

La seconde partie c'est la configuration et la mise en place de l'espace d'administration (Back Office).

Les fonctionnalités accessibles par l'internaute(client) peuvent être :

- Inscription.
- Visualiser produits et offres.
- Remplir son panier.
- S'authentifier.
- Passer au paiement.

Les fonctionnalités accessibles par l'internaute(client) peuvent être :

- Inscription.
- Gestion et visualisation des catégories de produits.
- Gestion et visualisation des produits.

4.2. Besoins non fonctionnels :

Les besoins non fonctionnels sont importants car ils agissent de façon indirecte sur le résultat et sur le rendement de l'utilisateur, ce qui fait qu'ils ne doivent pas être négligés, pour cela il faut répondre aux exigences suivantes :

- Fiabilité :

L'application doit fonctionner de façon cohérente sans erreurs et doit être satisfaisante.

- Les erreurs :

Les ambiguïtés doivent être signalées par des messages d'erreurs bien organisés pour bien guider l'utilisateur et le familiariser avec l'application web.

- Ergonomie des interfaces

L'application doit être adaptée à l'utilisateur sans qu'il ne fournisse aucun effort (utilisation claire et facile) de point de vue navigation entre les différentes pages, couleurs et mise en textes utilisés.

- Sécurité :

Notre solution doit respecter surtout la confidentialité des données personnelles des clients qui reste l'une des contraintes les plus importantes dans les sites web.

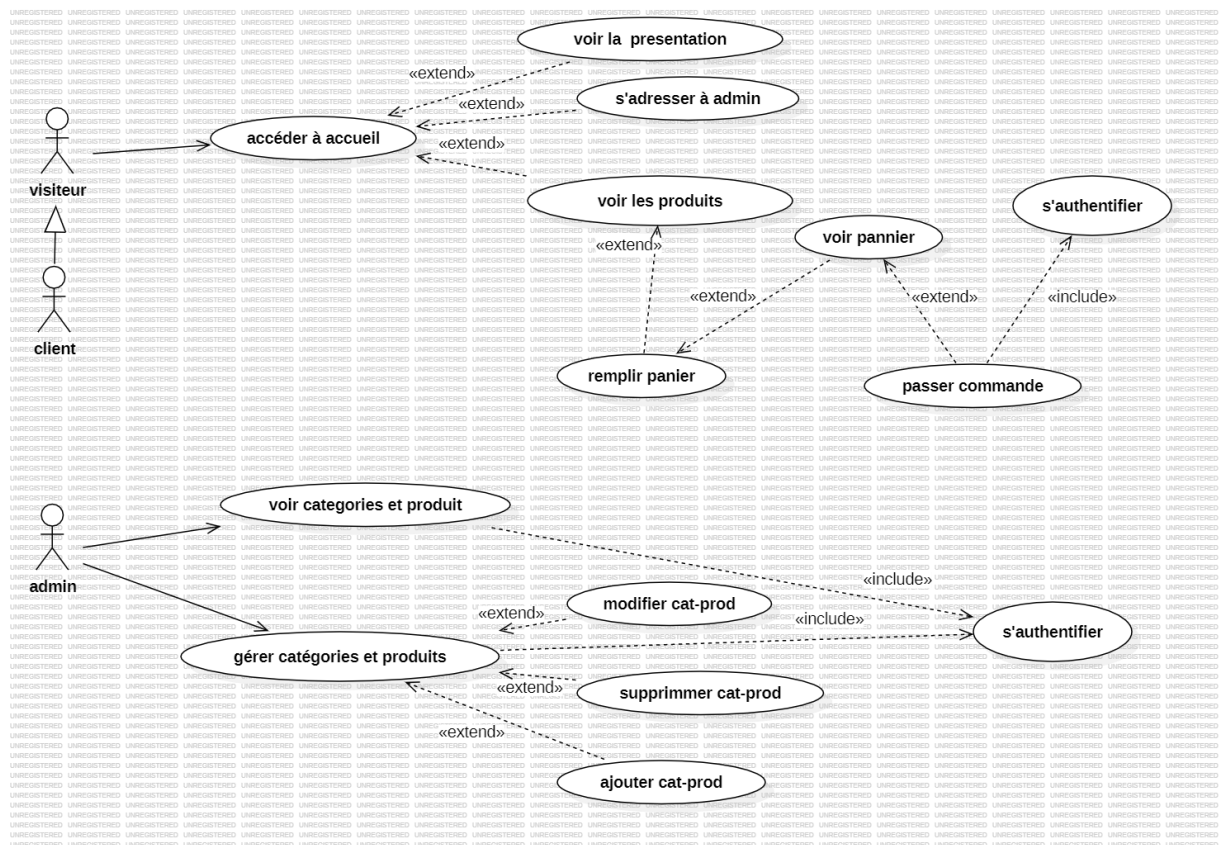
CHAPITRE 2 : MODELISATION CONCEPTUELLE (UML)

1.Introduction

Dans ce chapitre je vais présenter les modèles conceptuels des éléments principaux qui régissent mon application web, à savoir, le diagramme de cas d'utilisation, diagramme de classes et diagramme de séquence.

2.Diagramme de cas d'utilisation

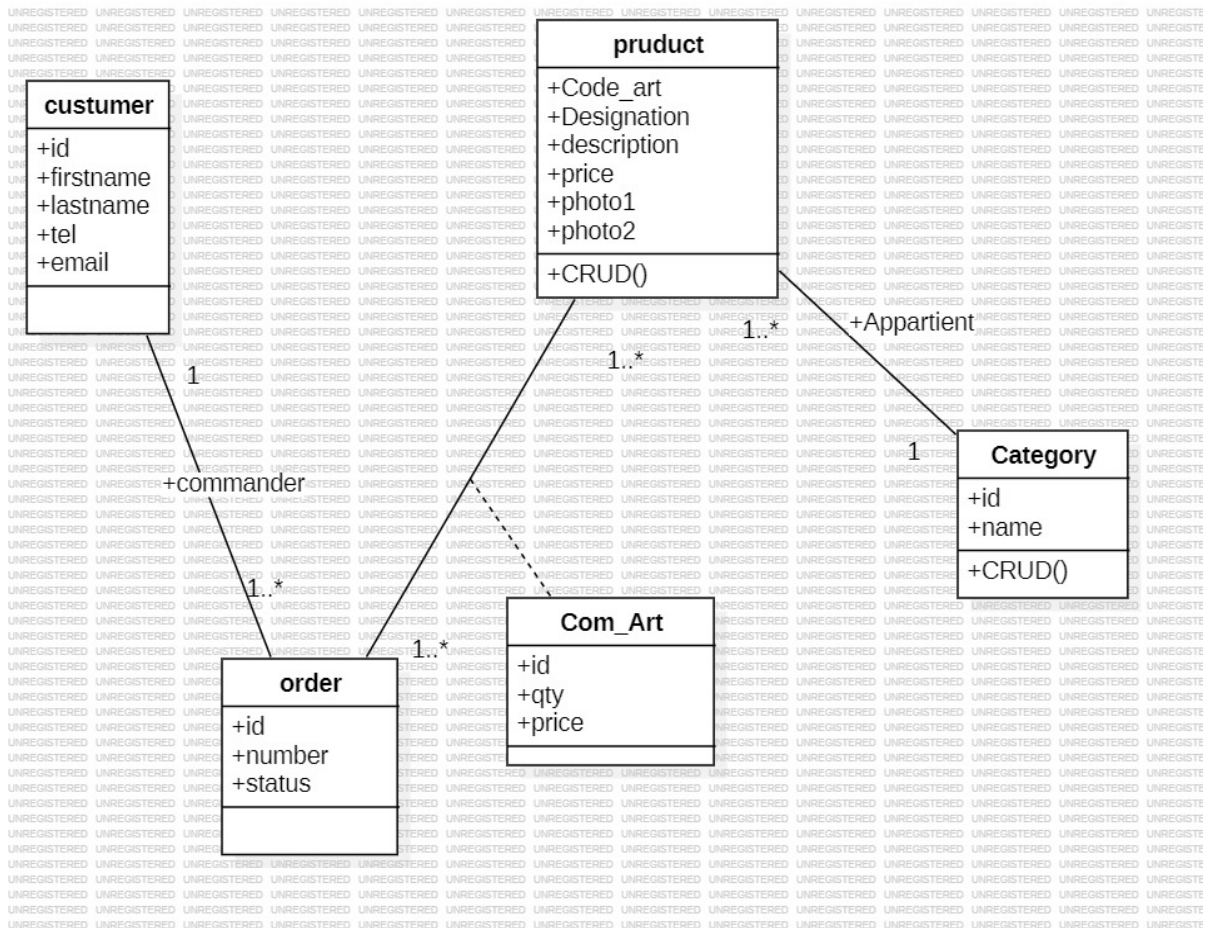
Cette section présent le diagramme de cas d'utilisation, qui fait apparaitre les fonctionnalités globales de l'application ainsi que les acteurs principaux qui utilisent cette dernière.



-Diagramme de cas d'utilisation de l'application shop.

3.Diagramme de classe

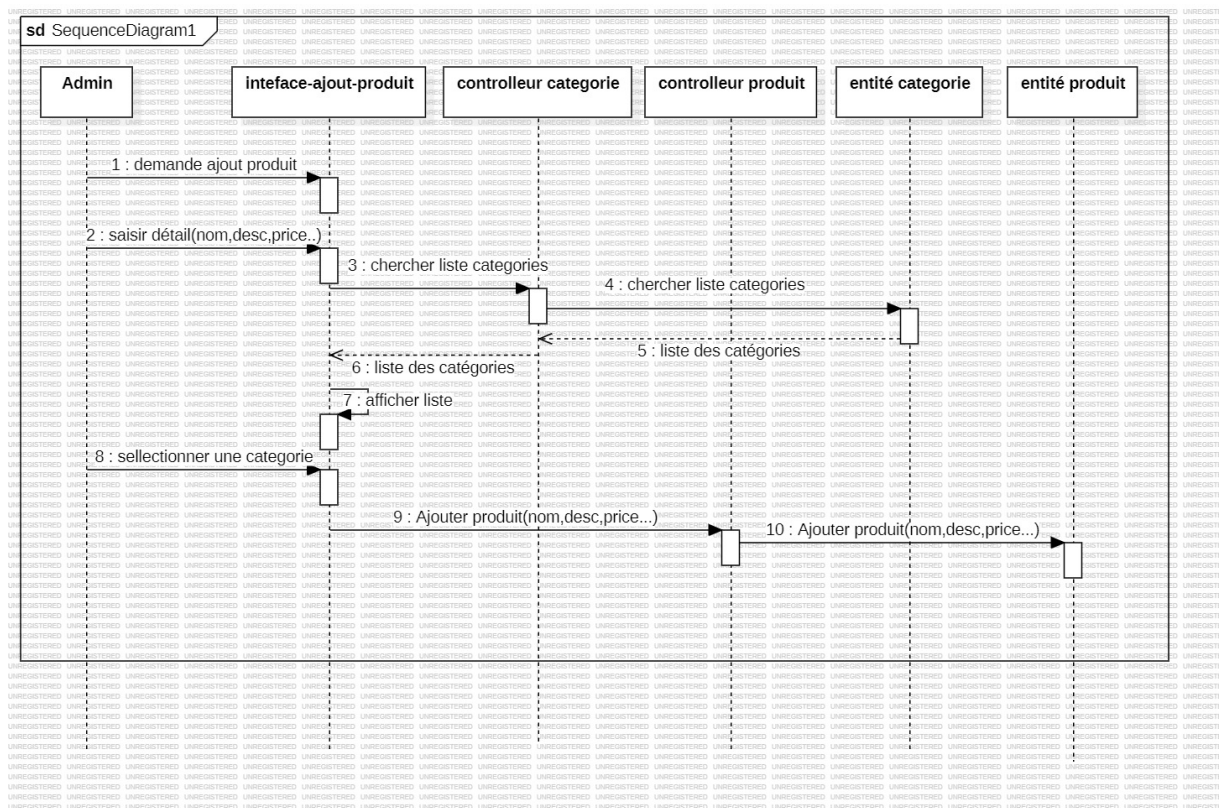
La figure ci-dessous le diagramme de classe qui régit l'application en montrant les classes qui doivent être développés ainsi leurs méthodes.



-Diagramme de classe de l'application shop

4.Diagramme de séquences

Je présente ci-dessous un diagramme de séquence d'une fonction sur le site qui permet à l'administrateur d'ajouter un produit à la vente, passant par la base de données, ainsi qu'une capture de l'interface d'ajout de produit.



-Diagramme de séquence : Ajout de produit

Ajouter un Produit

Produits

Ajouter une nouveau Produit

Nom Produit

Description Produit

Prix Produit

Photo 1 produit

Choisir un fichier
Aucun fichier choisi

Photo 2 produit

Choisir un fichier
Aucun fichier choisi

Catégorie produit

Ajouter Annuler

-Interface d'ajout de produit.

CHAPITRE 3 : REALISATION TECHNIQUE

1.Introduction

Dans cette partie de projet, je vais m'intéresser à l'aspect techniques de l'application, à savoir les technologies utilisés tel que, les Framework, les SGBD , ainsi l'environnement de développement choisi tout au long du la réalisation techniques du projet. D'une autre part, je vais montrer les étapes techniques principales du la réalisation, accompagnés par des figures montrant le code et les interfaces visuelles du l'application.

2.Framwork LARAVEL 10

Dans le cadre de travail sur un Framework, j'ai opté pour le Framework Laravel de version 10. Ce dernier est un Framework PHP open-source réputé pour sa simplicité, sa rapidité de développement et sa robustesse. Il offre une structure organisée pour construire des applications web complexes grâce à ses nombreuses fonctionnalités intégrées telles que l'ORM Eloquent, la gestion des sessions, la sécurité avancée, ainsi que son système de routage et de templating efficace. En combinant des outils modernes et des conventions de développement bien définies, Laravel facilite la création d'applications web évolutives et maintenables.

3.Environnement de développement IDE

Comme, environnement de développement, je vais utiliser Visual Studio Code au long de projet, quant a lui, est un éditeur de code source gratuit et open-source développé par Microsoft. Il offre une multitude de fonctionnalités telles que la coloration syntaxique, l'achèvement automatique, le débogage intégré, la gestion des extensions, et la prise en charge de multiples langages de programmation.

4. Serveur local

Afin de pouvoir exécuter le projet au niveau local, je vais basé sur l'outil Laragon pour plusieurs raison, premièrement il simplifier le processus de configuration et de gestion des serveurs web locaux, deuxièmement, il offre une interface conviviale permettant de démarrer rapidement des projets PHP avec des fonctionnalités telles que l'installation automatique de composer et l'intégration de bases de données

MySQL et MariaDB, ainsi que des outils pour la gestion des hôtes virtuels et des environnements de développement isolés.

5. Description de développement et test d'interfaçage

Dans cette section, je vais expliquer globalement le processus de développement et tout en démontrant des extraits de code et leurs résultats fonctionnels et visuels.

Après installer Laravel et mettre à jour Composer, j'ai ouvert l'invite de commande, dans le dossier www de Laragon, j'ai exécuté :

```
E:\laragon\www
λ cd www
E:\laragon\www\www
λ composer create-project laravel/laravel shop
Creating a "laravel/laravel" project at "./shop"
https://repo.packagist.org could not be fully loaded (curl error 6 while downloading
ckagist.org), package information was loaded from the local cache and may be out
Installing laravel/laravel (v10.3.2)
- Installing laravel/laravel (v10.3.2): Extracting archive
Created project in E:\laragon\www\shop
> @php -r "file_exists('.env') || copy('.env.example', '.env');"
Loading composer repositories with package information
https://repo.packagist.org could not be fully loaded (curl error 6 while downloading
ckagist.org), package information was loaded from the local cache and may be out
```

Un nouveau dossier shop se crée portant toute l'architecture laravel :



J'adopte un template e-commerce écrit en html-css-js pour mon projet :



Puis je crée le contrôleur webSite, et les vues blade qui constituent les pages de mon projet :

The image shows a development environment with two windows. The top window is a terminal running the command `php artisan make:controller webSiteController` in the directory `E:\laragon\www\shop`. The bottom window is a code editor showing the `WebsiteController.php` file and a file explorer on the left.

WebsiteController.php

```
1 namespace App\Http\Controllers;
2
3 use App\Models\Product;
4
5 use Illuminate\Http\Request;
6
7 class WebsiteController extends Controller
8 {
9     public function accueil(){
10         return view("website.accueil");
11     }
12
13     public function presentation(){
14         return view("website.presentation");
15     }
16
17     public function checkout(){
18         $products = session()->get('panier');
19         return view("website.checkout", compact('products'));
20     }
21
22     public function produits(){
23         $products = Product::all();
24         return view("website.produits", compact('products'));
25     }
26
27     public function contact(){
28         return view("website.contact");
29     }
30 }
```

File Explorer (Left Panel)

- SHOP
 - resources
 - views
 - website
 - accueil.blade.php
 - checkout.blade.php
 - contact.blade.php
 - navbar.blade.php
 - panier.blade.php
 - presentation.blade.php
 - produits.blade.php
 - save.blade.php

Code Editor (Right Panel)

The code editor shows the `accueil.blade.php` file. It starts with `@extends('layouts.template')` and `@section('titre')`. The main content is within `@section('contenu')`, featuring a slider and a product list.

```
1 @extends("layouts.template")
2
3 @section('titre')
4 Page d'accueil
5 @endsection
6
7
8 @section('contenu')
9 <!--Début de slider-->
10 <div class="slider-area ml-70 mr-70">
11     <div class="slider-active owl-carousel nav-style-1 owl-dot-none">
12         <div class="single-slider-2 slider-height-1 slider-height-res15 d-flex align-items-center slider-heig
13             <div class="container">
14                 <div class="row">
15                     <div class="col-xl-6 col-lg-6 col-md-7 ms-auto">
16                         <div class="slider-content-2 slider-content-fruits slider-animated-1">
17                             <h3 class="animated">Natural & Healthy</h3>
18                             <h1 class="animated">100% Organic <br>Fruits Collection</h1>
19                             <div class="slider-btn btn-hover">
20                                 <a class="animated" href="{{route('website.produits')}}>SHOP NOW</a>
21                             </div>
22                         </div>
23                     </div>
24                 </div>
25             </div>
26         </div>
27     </div>
28 </div>
```

Puis je déclare mes routes portant les URI dans le fichier web.php afin de d'afficher mes page :

EXPLORATEUR

STRUCTURE

SHOP

resources

routes

api.php

channels.php

console.php

web.php

storage

template_Site

templateAdmin

web.php

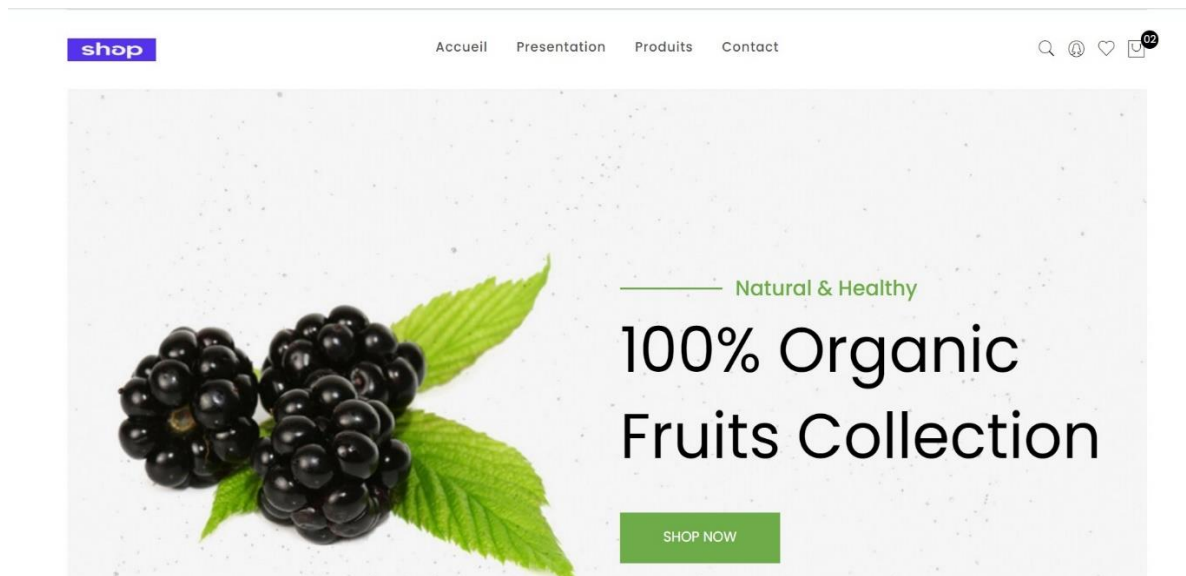
routes > web.php

```

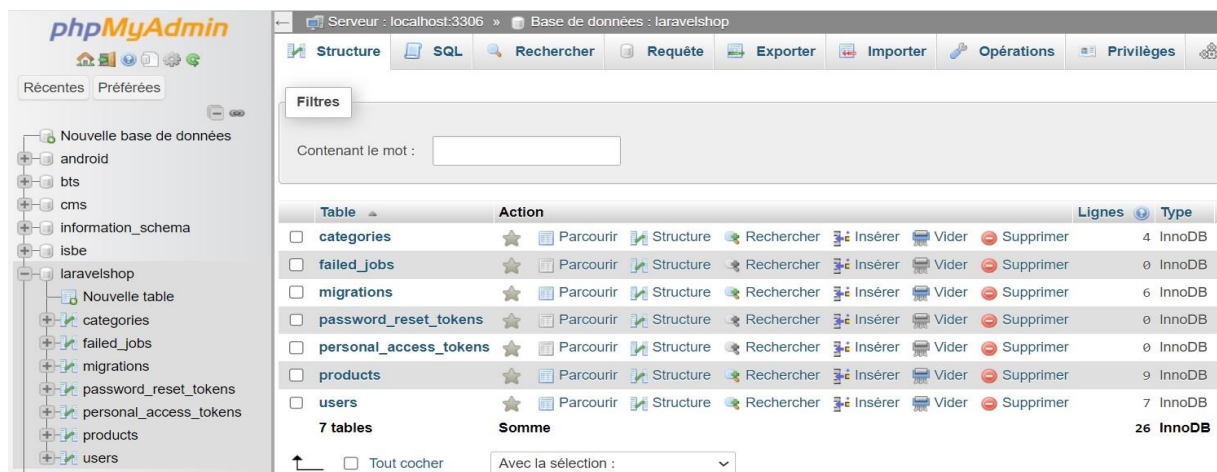
40
41 // WebSite routes :
42
43 Route::get("/",[WebsiteController::class,'accueil'])->name('website.acueil');
44 Route::get("/presentation",[WebsiteController::class,'presentation'])->name('website.presentation');
45 Route::get("/produits",[WebsiteController::class,'produits'])->name('website.produits');
46
47 Route::get("/checkout",[WebsiteController::class,'checkout'])->name('website.checkout');
48
49 Route::get("/contact",[WebsiteController::class,'contact'])->name('website.contact');
50 Route::post("/contact/save",[WebsiteController::class,'save'])->name("website.save");
51

```

J'obtiens un site statique au début avec une Template bien adapté au projet :



Et pour la base données, je crée d'abord ma base de données nommé laravelshop sur PhpMyAdmin fourni par laragon :



Et niveau de VS Code, je développe les model et les migrations afin de connecter mon projet à la base de données créée :

EXPLORATEUR

...

STRUCTURE

SHOP

app

Models

Category.php

Customer.php

Order_line.php

Order.php

Payment.php

Product.php

User.php

Providers

bootstrap

config

database

lang

public

resources

routes

storage

Product.php

X

app > Models > Product.php > ...

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

```

<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Product extends Model
{
    use HasFactory;
    protected $fillable = [
        'name',
        'description',
        'price',
        'photo1',
        'photo2',
        'category_id' ];

    public function category(){
        return $this-> belongsTo(Category::class);
    }

```

EXPLORATEUR

...

STRUCTURE

SHOP

database

migrations

2014_10_12_000000_create_users_table.php

2014_10_12_100000_create_password_reset_tokens_table...

2014_10_12_100000_create_password_resets_table.php

2019_08_19_000000_create_failed_jobs_table.php

2019_12_14_000001_create_personal_access_tokens_table...

2024_02_07_152704_create_categories_table.php

2024_02_07_155319_create_products_table.php

2024_02_12_144826_create_customers_table.php

2024_02_12_152726_create_orders_table.php

2024_02_12_155156_create_order_lines_table.php

2024_02_12_160221_create_payments_table.php

seeders

.gitignore

2024_02_07_155319_create_products_table.php

X

database > migrations > 2024_02_07_155319_create_products_table.php > class > up > Closure

7

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

```

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('products', function (Blueprint $table) {
            $table->id();
            $table->string("name",50);
            $table->longText("description")->nullable();
            $table->double("price");
            $table->string("photo1");
            $table->string("photo2");
            //declaration d'une clé étrangère
            $table->unsignedBigInteger('category_id');
            $table->foreign('category_id')->references('id')->on('categories');

            $table->timestamps();
        });
    }

```

Puis, je crée les contrôleurs correspondants à mes Models à savoir, categoriesController et productsController, et à l'aide des simples commandes j'aurai des contrôleurs préremplis des méthodes de CRUD et les routes de types ressources qui les correspond :

EXPLORATEUR

...

STRUCTURE

SHOP

app

Http

Controllers

CategoriesController.php

Controller.php

HomeController.php

OrderController.php

panierController.php

ProductsController.php

SiteController.php

TestController.php

WebsiteController.php

Middleware

Kernel.php

Models

Providers

bootstrap

config

database

lang

public

resources

routes

CategoriesController.php

X

ProductsController.php

app > Http > Controllers > CategoriesController.php > ...

8

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

```

class CategoriesController extends Controller
{
    /**
     * Display a listing of the resource.
     */
    public function index()
    {
        $categories = Category::all();
        return view('admin.categories.index', compact('categories'));
    }

    /**
     * Show the form for creating a new resource.
     */
    public function create()
    {
        return view('admin.categories.create');
    }

    /**
     * Store a newly created resource in storage.
     */
    public function store(Request $request)
    {
        //Sécurité : contrôle des champs
        $request->validate(["name" => "required|max:45|unique:categories"]);

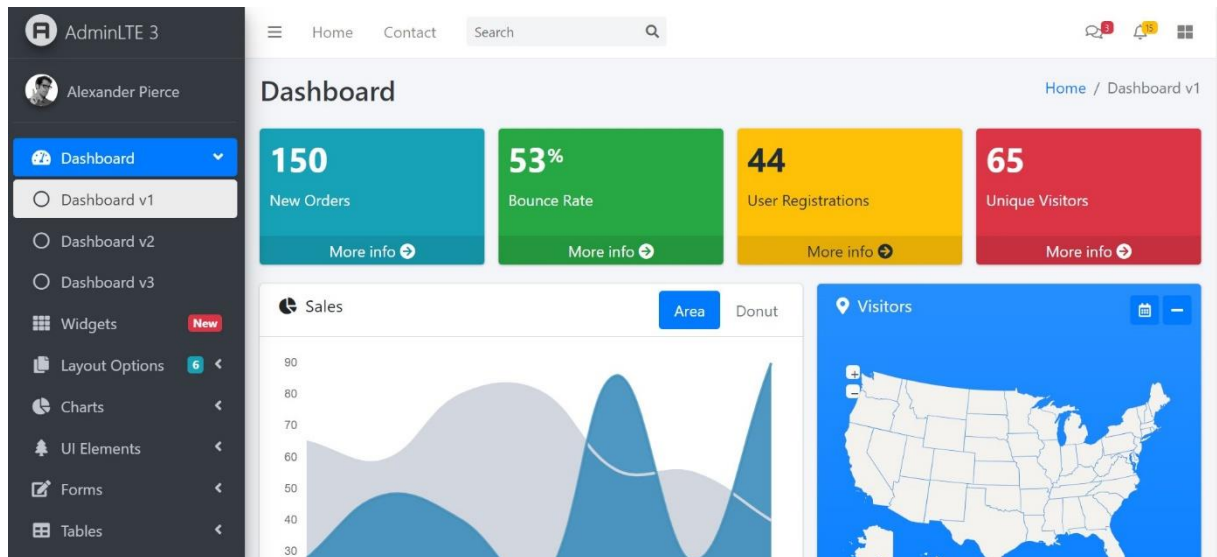
        //insertion dans la base de données
    }

```

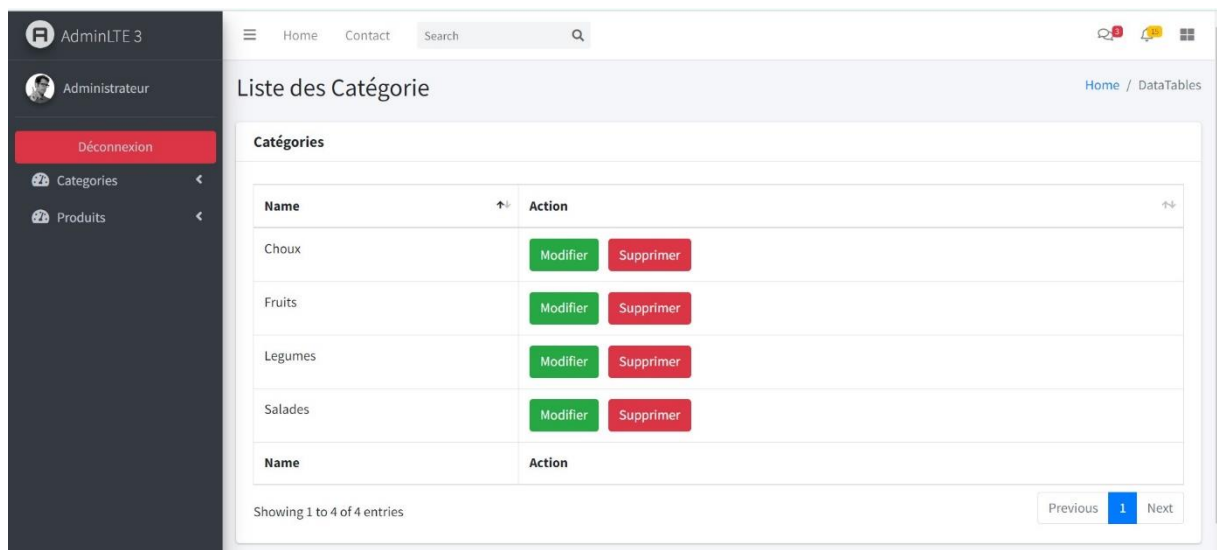
```
EXPLORATEUR
> STRUCTURE
  > SHOP
    > resources
    > routes
    > api.php

web.php
routes > web.php
57
58 Route::resource('categories', CategoriesController::class);
59 Route::resource('products', ProductsController::class);
60
```

Après avoir fait tout ça convenablement, je vais entamer la Admin , en commençant par l'import de template admin :



Puis je vais l'optimiser afin qu'il s'adapte à mon projet pour avoir le résultat ci-dessous :



Là, une grande partie de code sera consacré à appliquer les méthodes CRUD des contrôleurs sur la partie admin afin donner la main à l'admin de modifier, supprimer et d'ajouter un produits aussi sa catégorie.

Si je prends comme exemple la gestion des produits, alors je vais créer la vue index

améliorer son contrôleur qui va afficher la liste des produits et contenir des liens pour supprimer et modifier :

```

index.blade.php
resources > views > admin > products > index.blade.php > table#example2.table.table-bordered.table-hover > tbody > tr
15 <table id="example2" class="table table-bordered table-hover">
16   <thead>
17     <tr>
18       <th> Photo1</th>
19       <th> Photo2</th>
20       <th> Name</th>
21       <th> Description</th>
22       <th> Prix</th>
23       <th> Catégorie</th>
24       <th> Action</th>
25     </tr>
26   </thead>
27   <tbody>
28
29     @foreach ($products as $product)
30     <tr>
31       <td></td>
32       <td></td>
33       <td>{{ $product->name }}</td>
34       <td class="text-wrap">{{ $product->description }}</td>
35       <td>{{ $product->price }}</td>
36       <td>{{ $product->category->name }}</td>
37       <td>
38         <a href="{{route('products.edit', $product->id)}}"> <button type="button" class="btn btn-success mr-2">Modi
39         <form class="d-inline" action="{{ route('products.destroy',$product->id) }}" method="post">
40           @csrf
41           @method("DELETE")
42           <button type="submit" onclick=" return confirm('êtes-vous sûr de supprimer?')" class="btn btn-danger" >
43         </form>

```

```









public function index()
{
    $products = Product::all();
    return view('admin.products.index',compact('products'));
}

/**
 * Show the form for creating a new resource.
 */
public function create()
{
    $categories = Category::all();
    return view('admin.products.create', compact('categories'));
}

```

Déconnexion
Categories
Produits

Liste des produits

Photo1	Photo2	Name	Description	Prix	Catégorie	Action
		Iceberg	salade iceberg PCE	2.99	Salades	<div>Modifier</div> <div>Supprimer</div>
		Poivron	poivron rouge KG	3.69	Legumes	<div>Modifier</div> <div>Supprimer</div>
		Dattes	dattes d'algerie KG	7.99	Fruits	<div>Modifier</div> <div>Supprimer</div>
		Banane	Banane du Martinique KG	1.99	Fruits	<div>Modifier</div> <div>Supprimer</div>

Puis la vues modifier nommé edit.blade.php :

```
edit.blade.php x
resources > views > admin > products > edit.blade.php > ...
6
7 <!--@section('table')
8 |   Produits
9 @endsection-->
10
11 @section('contenu')
12
13     <div class="card card-primary">
14         <div class="card-header">
15             <h3 class="card-title">Modifier un Produit</h3>
16         </div>
17         <!-- /.card-header -->
18         <!-- form start -->
19         <form id="quickForm" method="POST" action="{{route('products.update', $product->id)}}" enctype="multipart/form-data">
20             @csrf
21             @method('PUT')
22             <!-- nom -->
23             <div class="card-body">
24                 <div class="form-group">
25                     <label for="name">Nom </label>
26                     <input type="text" name="name" value="{{ $product->name }}" class="form-control @error('name') border-danger">
27                     @error('name')
28                     | <div class="text-danger"> {{ $message }}</div>
29                     @enderror
30                 </div>
31             </div>
32             <!-- description -->
33             <div class="card-body">
```

Administrateur

Modifier un Produit

Home / DataTables

Produits

Modifier un Produit

Nom

Pink-lady

Description

Pomme Pink-lady du france KG

Prix

4,99

Photo 1

Choisir un fichier

Aucun fichier choisi

Et la vue ajouter nommé create.blade.php :


```





12
13 <div class="card card-primary">
14   <div class="card-header">
15     <h3 class="card-title">Ajouter une nouveau Produit</h3>
16   </div>
17   <!-- /.card-header -->
18   <!-- form start -->
19   <form id="quickForm" method="POST" action="{{route('products.store')}}" enctype="multipart/form-data">
20     @csrf
21     <!-- nom -->
22     <div class="card-body">
23       <div class="form-group">
24         <label for="name">Nom Produit </label>
25         <input type="text" name="name" class="form-control @error('name') border-danger @enderror" id="name" place
26         @error('name')
27         <div class="text-danger"> {{ $message }}</div>
28         @enderror
29       </div>
30     </div>
31     <!-- discription -->
32     <div class="card-body">
33       <div class="form-group">
34         <label for="description">Description Produit </label>
35         <textarea name="description" class="form-control @error('description') border-danger @enderror" id="des
36         @error('description')
37         <div class="text-danger"> {{ $message }}</div>
38         @enderror
39       </div>

```

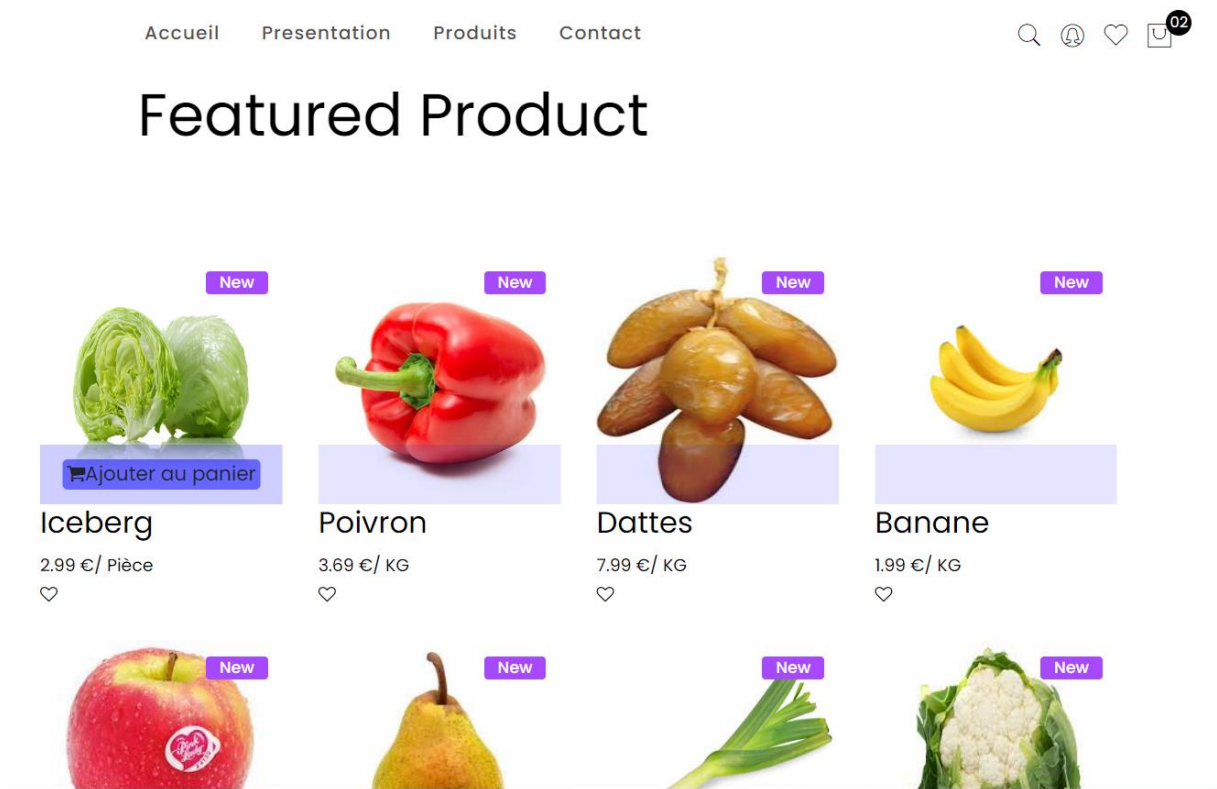
Et par exemple dans ce dernier interface, lorsque on veut ajouter un produit soit disant pomme de terre et on clique sur le bouton Ajouter produits, un nouveau produit s’ajoute dans la base de donn   en affichant un message de succ  s :

Liste des produits

un nouveau Produit est ajout   avec succ  s!

Photo1	Photo2	Name	Descreption	Prix	Categorie	Action
		Iceberg	salade iceberg PCE	2.99	Salades	Modifier Supprimer
		Poivron	poivron rouge KG	3.69	Legumes	Modifier Supprimer











```
produits.blade.php X
resources > views > website > produits.blade.php > ...
6  a pb-100">
12  ea pt-95 pb-100">
14  tainer">
17  "row flex-row-reverse">
18  ass="col-lg-9">
26
27  @foreach ($products as $product)
28
29  <div class="col-xl-3 col-md-6 col-lg-4 col-sm-6 toggle-item-active3">
30    <div class="product-wrap mb-25 scroll-zoom">
31      <div class="product-img">
32        <a href="#">
33          photo2) }}" width="117" height="200" alt="
35        </a>
36        <span class="purple">New</span>
37
38        <div class="product-action" style="width: 100%; background-color: rgba(0, 0, 255, 0.1);" >
39          <!--<div class="pro-same-action pro-wishlist">
40            <a title="Wishlist" href="#"><i class="pe-7s-l like"></i></a>
41          </div-->
42          <div class="pro-same-action pro-cart" style="width: 100%; background-color: rgba(0, 0, 255, 0.1);" >
43            <form method="post" action="{{ route('panier.addToCart') }}" title="Ajouter au panier">
44              @csrf
45              <input type="hidden" name="id" value="{{ $product->id }}">
46              <input type="hidden" name="name" value="{{ $product->name }}">
47              <input type="hidden" name="price" value="{{ $product->price }}">
```



Puis je développe le remplissage, l'affichage de panier et la vue panier :

```
panier.blade.php X
resources > views > website > panier.blade.php > ...
9  :-main-area pt-90 pb-100">
10  'container">
12  @isset('row')
13  / class="col-lg-12 col-md-12 col-sm-12 col-12">
14  <form action="#">
24  | <th>action</th>
25  </tr>
26  </thead>
27  <tbody>
28  @php
29  | $total=0;
30  @endphp
31  @foreach ($products as $key => $product)
32  @php
33  | $total+= $product['price']*$product['qty'];
34  @endphp
35  <tr>
36  <td class="product-thumbnail">
37  | <a href="#"></a>
38  </td>
39  <td class="product-name"><a href="#">{{ $product['name'] }}</a></td>
40  <td class="product-price-cart"><span class="amount"> {{ $product['price'] }} €</span></td>
41  <td class="product-quantity">
42  | <div class="cart-plus-minus">
43  | <input class="cart-plus-minus-box" type="text" name="qtybutton" value="1">
```

Votre panier

IMAGE	PRODUCT NAME	UNIT PRICE	QTY	SUBTOTAL	ACTION
	Poiré william	3.99 €	<div><div>-</div><div>1</div><div>+</div></div>	3.99 €	<div><div></div><div></div></div>
	Banane	1.99 €	<div><div>-</div><div>1</div><div>+</div></div>	1.99 €	<div><div></div><div></div></div>
	Iceberg	2.99 €	<div><div>-</div><div>1</div><div>+</div></div>	2.99 €	<div><div></div><div></div></div>

Après le remplissage je développe la partie vérification (checkout) et son bouton de redirection :

VIDER LE PANIER

I have one.

Cart Total

Total products **8.97 €**

Total shipping

- ☐ Standard \$20.00
☐ Express \$30.00

Grand Total 8.97 €

PROCEED TO CHECKOUT



shop

Accueil Presentation Produits Contact

Billing Details

Nom

votre nom

adresse email

mot de passe

confirmation

REGISTER

Your order

Product	Total
Poire william X 1	3.99 €
Banane X 1	1.99 €
Iceberg X 1	2.99 €
Shipping	Free shipping
Total	8.97 €

Cette dernière permet aux acheteurs de s'authentifier en remplissant les champs de formulaires, une fois c'est fait, ils peuvent accéder aux paiements.

Conclusion générale

J'avais essayé tout au long de mon travail de construire l'application étape par étape en suivant la méthodologie conceptuelle UML basé sur l'approche orienté objet.

J'avais commencé dans un premier lieu par comprendre le contexte général de projet et pourquoi ce projet doit-il être réalisé, ensuite j'avais fait une étude préalable afin de découvrir les exigences fonctionnelles et non fonctionnelles, et choisir la méthodologie à suivre.

J'avais passé par la suite à montrer les modèles conceptuels générale sur lesquelles l'application se base en montrant tous les besoins principaux de l'application ainsi les entités qui la correspond.

Et en fin, j'avais essayé de donner une idée globale sur la réalisation technique de projet, à partir des extraits de code, commandes et interfaces des parties cruciales de travail, en partant d'une description des technologies et outils utilisés dans cette réalisation.

Pour conclure, expérience était très important pour moi , dans la mesure où elle m'a permis d'appliquer mes connaissances acquises lors de mon cursus académique et des maîtriser davantage des nouvelles technologies de développement, concédèrent cette aventure est vraiment une occasion pour m'introduire et m'intégrer dans le milieu professionnel.