



## PROJET D'APPLICATION WEB

# E-picier

Bootstrap

Mysql

Figma

Django

Python

ENCADRÉ PAR : MR. EL AACHAK LOTFI

RÉALISÉ PAR :

- AMLAL OUTMANE
- EL KHLIFI IMAD
- AIT SIDI LKHIR ALI





*Département des Sciences Économiques et Gestion*

---

## PROJET D'APPLICATION WEB

---

RÉALISÉ PAR :

- AMLAL OUTMANE
- EL KHLIFI IMAD
- AIT SIDI LKHIR ALI

ENCADRÉ PAR :

- MR. EL AACHAK LOTFI



---

## REMERCIEMENTS

---

*Au nom du dieu le clé-*

*louange à ALLAH le tout puissant.*

*ment et le miséricordieux*

*Nous tenons tout d'abord à remercier dieu*

*le tout puissant de nous avoir donné la santé et la force et la patience et la volonté d'entamer et de terminer ce Modeste travail. À mon prophète Moham-*

*med, que la prière d'Allah et Son soient sur lui, qui a dit : « Celui qui prend un chemin pour rechercher la science, Allah lui fait prendre par cela un chemin vers le paradis. Certes les anges étendent leurs ailes pour celui qui recherche la science. Certes tous ceux qui sont dans les cieux et sur la terre demandent le pardon pour le savant même le poisson dans l'eau. Le mérite du savant sur l'adorateur est comme le mérite de la lune par rapport aux autres astres. Certes les savants sont les héritiers des prophètes et les prophètes n'ont pas laissé comme héritage des dinars ou des dirhams mais ils ont laissé la science, celui qui la prend aura pris une part importante ». Également je remercie chaleureusement infiniment mes parents et mes frères, qui m'ont encouragé et aidé à arriver à ce stade de ma formation. Je tiens à remercier tout particulièrement mon encadrant, DR. EL AACHAK LOTFI, pour ses conseils, sa patience, sa grande disponibilité et sa générosité. La pertinence de ses questions et ses remarques ont toujours motivé et dirigé. Qu'elle trouve ici le témoignage de ma profonde gratitude, notamment en me faisant partager son esprit admirable de travail, et aussi pour les énormes efforts lors des séances de préparation de ce travail. Mes remerciements vont également aux professeurs de la Faculté des sciences économiques de Tanger pour leurs disponibilité et leurs soutien.*

*Mes remerciements vont aussi à toutes mes amies et à tous les gens aimables*

*et serviables qui m'ont soutenu durant mes études. Enfin, un grand merci à toutes les personnes qui ont contribué de près ou de loin à la réalisation de ce travail.*



*Merci à tous*

---

*Ou�rane, Imad, Ali*

# SOMMAIRE

	Page
<b>Remerciements</b>	
<b>Sommaire</b>	
<b>Table des figures</b>	
<b>Liste des abréviations</b>	
<b>Introduction générale</b>	1
<b>2   Chapitre 1 Présentation du Projet</b>	
Introduction . . . . .	2
1.1 Présentation du Projet «l'application Web E-épicier» . . . . .	2
1.2 Analyse des besoins . . . . .	3
Conclusion . . . . .	4
<b>5   Chapitre 2 Analyse conceptuelle</b>	
Introduction . . . . .	5
2.1 Méthodologie d'analyse . . . . .	5
2.2 Outils utilisés dans la conception . . . . .	6
Conclusion . . . . .	8
<b>9   Chapitre 3 la démarche de la Réalisation du projet</b>	
Introduction . . . . .	9
3.1 Outils et technologies utilisés . . . . .	9
3.2 Interfaces graphiques . . . . .	17
Conclusion . . . . .	25
<b>Conclusion générale</b>	26
<b>Bibliographie</b>	27
<b>Table des matières</b>	28

---

## TABLE DES FIGURES

---

2.1	Diagramme de classe . . . . .	7
3.1	DataBases . . . . .	11
3.2	Interface Authentification . . . . .	17
3.3	Interface de la tentative de connexion . . . . .	18
3.4	Interface du changement de mot de passe . . . . .	18
3.5	Interface de la page d'accueil . . . . .	19
3.6	Interface d'ajout de clients . . . . .	20
3.7	Interface de la modification des clients . . . . .	20
3.8	Interface d'affichage les coordonnées des clients . . . . .	20
3.9	Interface de la Suppression des clients . . . . .	21
3.10	Interface d'affichage de la liste des crédits accordés aux clients . . . . .	21
3.11	Interface de la modification crédits des clients . . . . .	22
3.12	Interface Gestion des produits . . . . .	22
3.13	Interface d'ajouts produits . . . . .	23
3.14	Interface de modification des produits . . . . .	23
3.15	D'affichage de la liste des produits . . . . .	23
3.16	Interface de la suppression des produits . . . . .	24

---

---

## LISTE DES ABRÉVIATIONS

---

- **CSS** = Cascading Style Sheets
- **HTML** = HyperText Markup Language
- **JS** = JavaScript
- **SGBDR** = Système de gestion de base de données relationnelle.
- **UML** = Unified Modeling Language

## INTRODUCTION GÉNÉRALE

D epuis la découverte de l'informatique, de nombreuses activités de la vie courante ont été simplifiés. Actuellement les individus peuvent facilement traiter des informations en se servant des logiciels et des réseaux informatiques. Compte tenu de son évolution, ce système caractérise la majorité des grandes entreprises quel que soit le secteur d'activité. Alors, l'informatique est devenue un élément clé de notre vie moderne et incontournable dans de nombreux domaines, tels que, le stockage, la gestion de données, la recherche et développement.

En effet, l'Internet est un système de communication utile qui permet la communication et l'échange facile des informations par excellence et à une grande quantité de ressources en ligne. Ce dernier permet donc, de généraliser l'utilisation des logiciels d'informatiques plus performants avec des clients légers à travers des navigateurs web complets. Cette technologie permet le développement des applications pouvant tourner sous différents navigateurs, tout en assurant la sécurité que procure une application.

A cet effet, notre travail consiste à développer une application de la gestion de crédit d'épicerie. A fin de simplifier la gestion de caisse quotidienne de l'épicerie, pour mieux gérer les produits et les transactions de crédit avec les clients, la vérification de dettes. Cette application permet de digitaliser le carnet client.

Pour réaliser notre projet on va suivre la démarche suivante :

- Chapitre 1 : Le premier chapitre est une prise de connaissance du présentation du Projet « l'application Web E-épicier », ainsi son objectif, une analyse et spécification des besoins (fonctionnels non, fonctionnels) ;
- Chapitre 2 : Analyse de conception de l'application, ce chapitre sera consacrée à la méthodologie d'analyse du Langage UML. Ainsi, le diagramme de classe et outils utilisés dans la Conception(traitements, données) ;
- Chapitre 3 : Réalisation du « l'application Web E-épicier », ce chapitre contient une description détaillé des outils technologiques utilisés (HTML, Django, PhpMyAdmin) pour développer l'application web, Ainsi, la phase d'implémentation de différentes étapes de l'application.

*“En programmation, tout ce que nous faisons est un cas particulier de quelque chose de plus général - et souvent nous nous en apercevons trop vite.”*

— Alan Jay Perlis, Epigrams on Programming (1982)

### Plan

Introduction . . . . .	2
1.1 Présentation du Projet «l'application Web E-épicier» . . . . .	2
1.2 Analyse des besoins . . . . .	3
Conclusion . . . . .	4

### Introduction

Ce premier chapitre est consacré pour présenter le rôle et l'objectif du « l'application Web E-épicier », ainsi une analyse spécifique des besoins (fonctionnels non, fonctionnels).

#### 1.1 Présentation du Projet «l'application Web E-épicier»

##### 1.1.1 Rôle et objectifs de «l'application Web E-épicier»

Le développement d'une application de gestion de crédit pour une épicerie est un aspect important de la gestion financière, être bénéfique pour les épiciers .en aidant à gérer les crédits des clients de manière plus efficace. Tels que les transactions de crédit avec les clients qui souhaitent acheter des produits sans payer immédiatement en utilisant de l'argent liquide ou par crédit.

Voici quelques éléments clés de l'application pour la gestion de crédit dans une épicerie :

- **Gestion des clients** : l'application devrait permettre de créer et de gérer une base de données de clients, avec leurs informations de contact ;

- **Gestion des crédits** : l'application devrait permettre de créer et de gérer les comptes de crédit pour les clients. Ainsi, de savoir le statut de paiement ;
- **Gestion des produits** : l'application aide l'épicier d'ajouter ou supprimer ou varier les prix des produits facilement ;
- **vérification de crédit** : Avant d'accorder du crédit à un nouveau client, l'épicier doit effectuer une vérification de crédit pour évaluer la capacité du client à rembourser son crédit à partir l'historique de crédit et des graphes statistiques.

## 1.2 Analyse des besoins

Cette partie va servir à poser les bases du recueil des besoins du système à réaliser. Pour pouvoir clarifier les besoins des utilisateurs de notre application, nous allons présenter les besoins fonctionnels ainsi que les besoins non fonctionnels.

### 1.2.1 Les besoins fonctionnels

Les besoins fonctionnels en informatique sont les exigences qui doivent être satisfaites par un système informatique pour qu'il réponde aux attentes des utilisateurs et remplisse ses objectifs. Ces besoins sont liés aux fonctionnalités du système, à ses performances, à son ergonomie, à sa fiabilité et à sa sécurité. Ils sont souvent exprimés sous forme de spécifications fonctionnelles qui décrivent les fonctionnalités attendues du système et les conditions dans lesquelles elles doivent être mises en œuvre.

Nous pouvons identifier des besoins comme suit :

- **Gestion de clients** : Permet de saisir les informations de base du client. possibilité d'ajouter, de modifier et de supprimer des clients ainsi que d'afficher leur historique de crédit ;
- **Gestion des transactions** : permettre la création, la modification et la suppression des transactions de crédit pour les clients, ainsi que la consultation de leur historique ;
- **Traitements des données** : l'application devra être calculer les sommes de crédits de chaque client, analyses graphiquement son carnet de crédit ;
- **Consultation de l'historique des clients** : l'application permet au administrateur de voir les activités réalisées dans le système ordonnées par date.

### 1.2.2 Les besoins non fonctionnels

Les besoins non fonctionnels, sont des caractéristiques d'un système informatique qui ne sont pas liées directement à ses fonctionnalités, mais qui ont une importance cruciale pour garantir sa qualité, sa performance, sa sécurité et sa convivialité.

Notre application doit nécessairement assurer ces besoins :

- **Fiabilité** : l'application doit être robuste et ne pas planter ou provoquer des erreurs ;
- **Accessibilité** : dans le cadre de ce travail, l'application devra être accessible à l'utilisateur ;
- **La performance** : l'application devra être performante c'est-à-dire que le système doit capable de traiter un grand nombre de requêtes en un temps raisonnable et de fournir des résultats rapidement ;
- **La convivialité** : l'application doit être facile à utiliser et à comprendre même par non expert ;
- **Maintenabilité** : l'application doit être facile à maintenir, à dépanner et à mettre à jour pour l'administrateur ;
- **L'extensibilité** : dans le cadre de ce travail, l'application devra être extensible, c'est-à-dire qu'il pourra y avoir une possibilité d'ajouter ou de modifier de nouvelles fonctionnalités.

## Conclusion

Ce chapitre présente l'utilité de ce projet et l'analyse de notre application. Nous avons défini les différents besoins fonctionnels et non fonctionnels de l'application « gestion de crédit dans une épicerie ». Nous entamerons dans le chapitre suivant la conception de cette application qui comporte la méthodologie d'analyse, les différents diagrammes de cas et le diagramme de classe.

# Chapitre

# 2

## ANALYSE CONCEPTUELLE

*“Certains langages de programmation arrivent à absorber le changement, mais résistent au progrès.”*

— Alan Jay Perlis, Epigrams on Programming (1982).

### Plan

Introduction . . . . .	5
2.1 Méthodologie d'analyse . . . . .	5
2.2 Outils utilisés dans la conception . . . . .	6
Conclusion . . . . .	8

### Introduction

La phase de la conception est l'étape initiale de la création et de la mise en œuvre de notre projet.

Dans ce chapitre, nous allons présenter en détails la conception du projet à travers le langage UML : le diagramme de classes.

#### 2.1 Méthodologie d'analyse

##### 2.1.1 Le Langage UML

UML (Unified Modeling Language) est un langage graphique de modélisation de systèmes informatiques. Il est utilisé pour représenter, concevoir, visualiser et documenter les différents aspects d'un système logiciel. Il permet de décrire les différents composants d'un système, leur comportement, leurs interactions ainsi que les relations entre ces composants.

Dans notre projet on a utilisé le diagramme de classe qui représente les différentes classes du

système, leurs attributs et leurs méthodes, ainsi que les relations entre les différentes classes.

## 2.2 Outils utilisés dans la conception

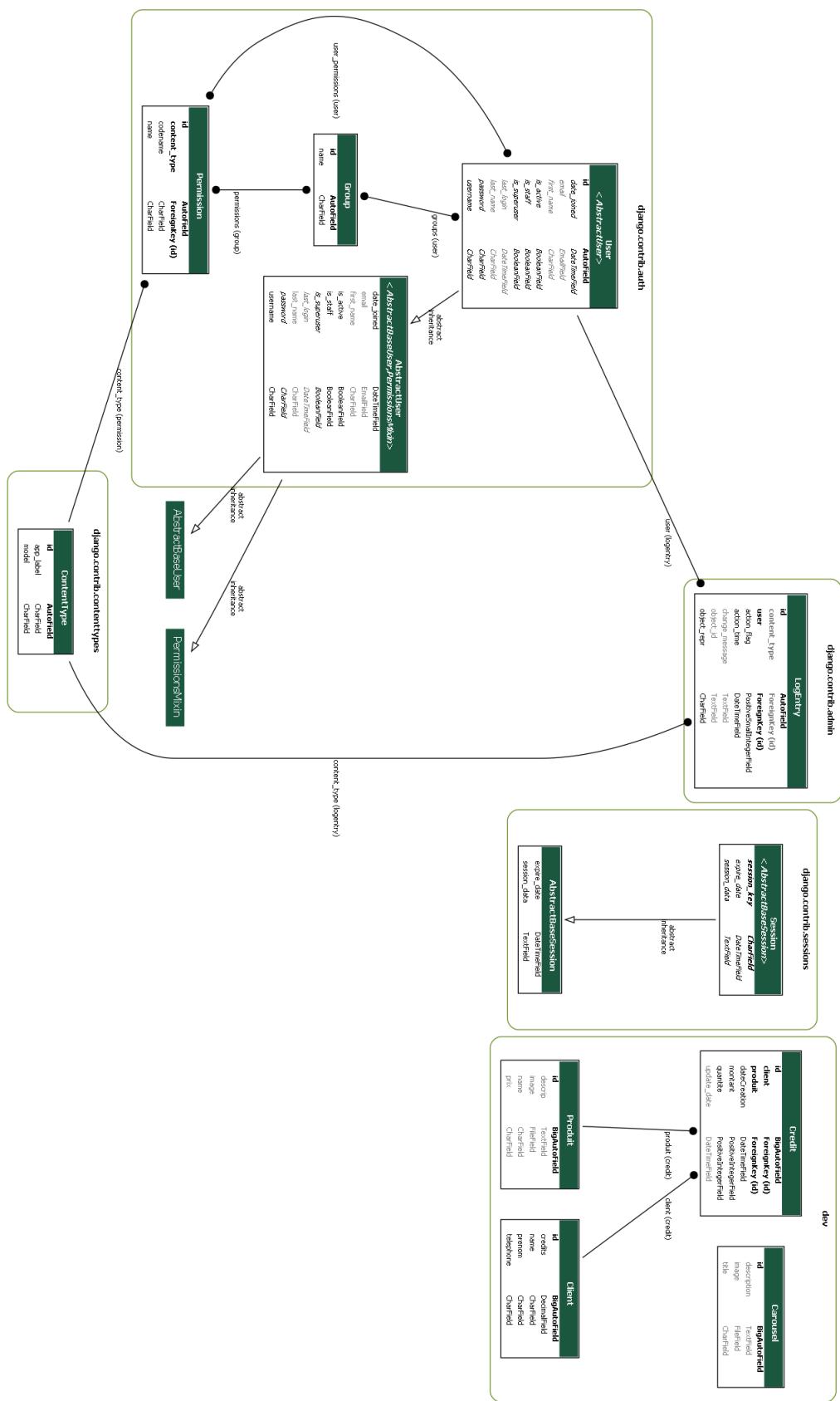
### 2.2.1 Logiciels Utilisés

- **Graphviz Django** : Graphviz est un logiciel open-source de visualisation de graphes qui permet de créer des diagrammes et des représentations visuelles de structures complexes, telles que des organigrammes, des graphes orientés et des arbres. Django est un framework web populaire écrit en Python qui permet de construire des applications web rapidement et facilement ;
- **Visual Studio Code** : est un éditeur de code source développé par Microsoft pour Windows, Linux et macOS. Il inclut des fonctionnalités telles que la coloration syntaxique, la compléction automatique de code, le débogage, la gestion de version et l'intégration de nombreux outils externes.

### 2.2.2 Conception des données

#### 2.2.2.1 Diagramme de classe

Figure 2.1 : Diagramme de classe



Source : Élaboré par nos soins à partir des sorties Django

### 2.2.2.2 Description de diagramme de classe

Ces modèles sont définis dans le langage de programmation Python en utilisant le framework Django pour créer une application web.

- Le modèle "**Carousel**" représente une diapositive de carousel avec un titre, une description et une image. Il contient également une méthode str() qui renvoie le titre de l'objet ;
- Le modèle "**Produit**" représente un produit avec un nom, une image, une description et un prix. Il contient également une méthode str() qui renvoie le nom de l'objet ;
- Le modèle "**Client**" représente un client avec un nom, un prénom, un numéro de téléphone et un crédit (décimal). Il contient également une méthode str() qui renvoie le nom de l'objet ;
- Le modèle "**Credit**" représente une transaction de crédit, enregistrant le client qui a effectué la transaction, le produit acheté, la quantité achetée, la date de création de la transaction, la date de mise à jour et le montant total de la transaction. Il contient également une méthode str() qui renvoie le nom du client et le nom du produit avec la quantité achetée.

Les modèles "Credit" et "Client" sont associés par une clé étrangère, où chaque transaction de crédit est liée à un client particulier qui a effectué la transaction. De même, les modèles "Credit" et "Produit" sont associés par une clé étrangère, où chaque transaction de crédit est liée à un produit particulier qui a été acheté.

## Conclusion

Dans ce chapitre, nous avons présenté les langages ( UML, Visual studio code, Graphviz Django) . Aussi nous avons fait la description de diagramme de classe et de contexte afin, de préparer un terrain favorable pour la prochaine étape. Maintenant, notre application est prête à être codée. Dans le chapitre suivant, nous allons intéresser à la réalisation du projet.

# Chapitre 3

## LA DÉMARCHE DE LA RÉALISATION DU PROJET

*“La plupart des gens trouvent le concept de la programmation évident, mais la réalisation impossible.”*

— Alan Jay Perlis, Epigrams on Programming (1982).

### Plan

Introduction . . . . .	9
3.1 Outils et technologies utilisés . . . . .	9
3.2 Interfaces graphiques . . . . .	17
Conclusion . . . . .	25

### Introduction

Dans le chapitre précédent nous avons présenté les étapes de conception de l'application ,ainsi que le diagramme de classe. Dans ce chapitre nous traitons les différentes étapes d'implémentation de l'application à travers un ensemble des étapes de la phase de réalisation.

Nous allons commencer par les outils technologiques utilisés en se basant sur une description de l'environnement logiciel tout en donnant par la suite un aperçu sur une présentation de l'application au cours de la période de développement.

#### 3.1 Outils et technologies utilisés

##### 3.1.1 Environnement logiciel

Nous avons présenté au cours de cette partie les différents outils utilisés tout au long de ce projet pour l'étude et la mise en place de notre application.

**Front End technologies utilisés :**

**HTML** Hypertext markup language

**CSS** Cascading Stylesheet

**JS** JavaScript

**Bootstrap** (framework of HTML, CSS, and JS)

**Back-End technologies utilisés :**

Python (for logical coding)

MySQL Database

**Back-End Python (3.11 version) :**

Django (4. x version)

Xampp

### 3.1.2 Les étapes de réalisation du projet avec Django

a. **Installer Django** : on a installé Django en utilisant pip, le gestionnaire de paquets Python.

Dans notre terminal on exécute :

```
python -m pip install django
```

b. **Créer un projet** : en exécutant la commande suivante dans le terminal :

```
django-admin startproject dev
```

c. Créer une application en exécutant la commande suivante dans le terminal :

```
django-admin startproject grofin
```

d. **La configuration des templates** : La configuration des templates et du dossier statique (ou dossier des fichiers statiques) est une étape importante dans la création d'une application Web.

e. **Configurer la base de données avec MySQL phpmyadmin** : Dans le fichier settings.py du projet, en configurer les paramètres de la base de données que nous allons utiliser.

**Figure 3.1 : DataBases**

```

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'mabase4',
        'USER': 'root',
        'PASSWORD': '',
        'HOST': '127.0.0.1',
        'PORT': '3306',
        'OPTIONS': {
            'init_command': "SET sql_mode='STRICT_TRANS_TABLES'"
        }
    }
}

```

**Source :** Élaboré par nos soins à partir des sorties Django

f. Executer la commande makemigrations pour créer modèle classe dans la base de donnée.

g. Executer la commandemigrate.

h. Executer la commande createsuperuser pour créer superuser :

- python manage.py migrate;
- python manage.py createsuperuser ;
- python manage.py runserver ;

i. Créer des models.py : Dans le fichier models.py de notre application, définissons les modèles que nous allons utiliser pour stocker les données.

```

1     classProduit(models.Model):
2         name = models.CharField(max_length=100, null=True, blank=True)
3         image = models.FileField(null=True, blank=True)
4         descrip = models.TextField(null=True, blank=True)
5         prix = models.CharField(max_length=100, null=True, blank=True)
6
7         def __str__(self):
8             return self.name
9
10
11     classClient(models.Model):
12         name = models.CharField(max_length=255)
13         prenom = models.CharField(max_length=255)
14         telephone= models.CharField(max_length=20)
15         credits = models.DecimalField(max_digits=10, decimal_places=2, default=0)
16
17         def __str__(self):
18             return self.name
19
20     classCredit(models.Model):
21         client = models.ForeignKey(Client, on_delete=models.CASCADE)
22         produit = models.ForeignKey(Produit, on_delete=models.CASCADE, null=True)
23         quantite = models.PositiveIntegerField(default=1)
24         dateCreation= models.DateTimeField(default=datetime.now)

```

```

24         update_date = models.DateTimeField(auto_now=True)
25         montant = models.PositiveIntegerField(null=True)
26
27     def __str__(self):
28         return f"{self.client.name} - {self.produit.name} ({self.quantite})"
29

```

Code Listing 3.1 – Python Models

j. **Créer des Views.py :** Dans le fichier views.py de notre application, on a défini les vues qui vont être utilisées pour traiter les requêtes. p

```

1 # Create your views here.
2
3 def home(request):
4     return render(request, 'home.html')
5
6 def mainpage(request):
7     return render(request, 'nav.html')
8
9 def main(request):
10    data = Carousel.objects.all()
11    dic = {'data':data}
12    return render(request, 'mainpage.html', dic)
13
14 # dashboard
15
16 def adminHome(request):
17     return render(request, 'adminhome.html')
18
19 def adminDashboard(request):
20     return render(request, 'admindashboard.html')
21
22 # authentication
23 def adminLogin(request):
24    msg = None
25    if request.method == "POST":
26        username = request.POST['username']
27        password = request.POST['password']
28        user = authenticate(username=username, password=password)
29        try:
30            if user.is_staff:
31                login(request, user)
32                msg = "User login successfully"
33                return redirect(creerCredit)
34            else:
35                msg = "les donnees non valides"
36        except:
37            msg = "les donnees non valides"
38    dic = {'msg': msg}
39    return render(request, 'admin_login.html', dic)
40
41 def admin_change_password(request):
42    if request.method == 'POST':
43        o = request.POST.get('currentpassword')
44        n = request.POST.get('newpassword')

```

```

44         c = request.POST.get('confirm_password')
45         user = authenticate(username=request.user.username, password=o)
46         if user:
47             if n == c:
48                 user.set_password(n)
49                 user.save()
50                 messages.success(request, "PasswordChanged")
51                 return redirect('admin_login')
52             else:
53                 messages.success(request, "Password not matching")
54                 return redirect('admin_change_password')
55         else:
56             messages.success(request, "InvalidPassword")
57             return redirect('admin_change_password')
58     return render(request, 'admin_change_password.html')

59
60 #produit

61
62 def add_produit(request):
63     if request.method == 'POST':
64         name = request.POST.get('name')
65         descrip = request.POST.get('description')
66         prix = request.POST.get('prix')
67         image = request.FILES.get('image')
68         produit = Produit(name=name, descrip=descrip, prix=prix, image=image)
69         produit.save()
70         return redirect(afficherProduit)
71     return render(request, 'add_produit.html')

72
73 def afficherProduit(request):
74     produits = Produit.objects.all()
75     return render(request, 'afficherProduit.html', locals())

76
77 def edit_produit(request, pid):
78     produit = Produit.objects.get(id=pid)
79     if request.method == 'POST':
80         form = ProduitForm(request.POST, instance=produit)
81         if form.is_valid():
82             if 'image' in request.FILES:
83                 image = request.FILES['image']
84                 produit.image = image
85                 form.save()
86                 return redirect('afficherProduit')
87         else:
88             form = ProduitForm(instance=produit)

89
90     return render(request, 'edit_produit.html', {'form': form, 'produit': produit})

91
92 def delete_produit(request, pid):
93     produit = Produit.objects.get(id=pid)
94     produit.delete()
95     messages.success(request, "produit supprime")
96     return redirect('afficherProduit')

97
98 # client

```

```
99
100 defadd_client(request):
101     if request.method == 'POST':
102         form = ClientForm(request.POST)
103         if form.is_valid():
104             form.save()
105             return redirect('client_list')
106     else:
107         form = ClientForm()
108     return render(request, 'add_client.html', {'form': form})
109
110 defclient_list(request):
111     clients = Client.objects.all()
112     return render(request, 'client_list.html', {'clients': clients})
113
114 defedit_client(request, client_id):
115     client = get_object_or_404(Client, id=client_id)
116     if request.method == 'POST':
117         form = ClientForm(request.POST, instance=client)
118         if form.is_valid():
119             form.save()
120             return redirect('client_list')
121     else:
122         form = ClientForm(instance=client)
123     context = {'form': form, 'client': client}
124     return render(request, 'edit_client.html', context)
125
126 defdelete_client(request, client_id):
127     client = get_object_or_404(Client, id=client_id)
128     if request.method == 'POST':
129         client.delete()
130         return redirect('client_list')
131     else:
132         context = {'client': client}
133         return render(request, 'client_list', context)
134
135 deftrop_risque_clients(request):
136     trop_risque_clients = Client.objects.filter(credits__gt=2000)
137     context = {'trop_risque_clients': trop_risque_clients}
138     return render(request, 'ceerCredit.html', context)
139
140 # credits
141
142 defcreerCredit(request):
143     if request.method == 'POST':
144         client_id = request.POST['client']
145         produit_id = request.POST['produit']
146         quantite = request.POST.get('quantite')
147         if quantite is not None:
148             quantite = int(quantite)
149             client = Client.objects.get(id=client_id)
150             produit = Produit.objects.get(id=produit_id)
151             montant = Decimal(produit.prix) * Decimal(quantite)
152             credit = Credit.objects.create(
153                 client=client,
```

```

154         produit=produit,
155         quantite=quantite,
156         dateCreation=timezone.now(),
157         montant=montant)
158     client.credits += montant
159     client.save()
160     messages.success(request, 'Le credit a ete cree avec succes.')
161     return redirect('creerCredit')
162 else:
163     messages.error(request, 'Veuillez entrer une quantite valide.')
164     return redirect('creerCredit')
165 else:
166     clients = Client.objects.all()
167     produits = Produit.objects.all()
168     sumCredit = Client.objects.aggregate(Sum('credits'))['credits__sum'] or 0
169     context = {
170         'clients': clients,
171         'produits': produits,
172         'sumCredit': sumCredit,}
173     return render(request, 'creerCredit.html', context)
174
175
176 def credits_list(request):
177     clients = Client.objects.all()
178     client_list = []
179     for client in clients:
180         try:
181             latest_credit = client.credit_set.latest('dateCreation')
182         except Credit.DoesNotExist:
183             logging.warning(f"No Credit objects found for Client '{client.name}{client.prenom}', (ID: {client.id})")
184             continue
185
186         client_data = {
187             'id': client.id,
188             'name': client.name,
189             'prenom': client.prenom,
190             'credits': client.credits,
191             'date_creation': latest_credit.dateCreation,
192         }
193         client_list.append(client_data)
194     context = {'client_list': client_list}
195     return render(request, 'credits_list.html', context)
196
197 def edit_credits(request, client_id):
198     client = get_object_or_404(Client, id=client_id)
199     if request.method == 'POST':
200         form = EditCreditForm(request.POST, instance=client)
201         if form.is_valid():
202             form.save()
203             return redirect('credits_list')
204         else:
205             form = EditCreditForm(instance=client)
206     return render(request, 'edit_credits.html', {'form': form, 'client': client})
207
208 #outils

```

```

209
210     def search(request):
211         query = request.GET.get('q')
212         produits = Produit.objects.filter(name__icontains=query)
213         if query else []
214         clients = Client.objects.filter(name__icontains=query)
215         if query else []
216         context = {
217             'query': query,
218             'produits': produits,
219             'clients': clients,
220         }
221         return render(request, 'search.html', context)
222

```

Code Listing 3.2 – Python Views

**k. Créer des URLs :** Dans le fichier urls.py de notre application, on a saisie les URL's qui vont être utilisées pour accéder aux vues. Dans views.py

```

1 urlpatterns = [
2     path('admin/', admin.site.urls),
3     path('', home, name="home"),
4     path('mainpage/', mainpage, name="mainpage"),
5     path('main/', main, name="main"),
6     path('adminhome/', adminHome, name="adminhome"),
7     #authentification
8     path('admin-login/', adminLogin, name="admin_login"),
9     path('admindashboard/', adminDashboard,
10         name="admindashboard"),
11     path('admin-change-password/', admin_change_password,
12         name="admin_change_password"),
13     #produit
14     path('afficherProduit/', afficherProduit, name='afficherProduit'),
15     path('edit-produit/<int:pid>/', views.edit_produit, name='edit-produit'),
16     path('add-produit/', views.add_produit, name='add_produit'),
17     path('delete-produit/<int:pid>/', delete_produit, name="delete_produit"),
18     #client
19     path('add-client/', views.add_client, name='add_client'),
20     path('client-list/', views.client_list, name='client_list'),
21     path('edit-client/<int:client_id>/edit/', views.edit_client, name='edit_client'),
22     path('delete-client/<int:client_id>/', views.delete_client, name='delete_client'),
23     path('trop_risque_clients/', views.trop_risque_clients, name='trop_risque_clients'),
24     # credits
25     path('credits_list/', credits_list, name='credits_list'),
26     path('edit_credits/<int:client_id>/', edit_credits, name='edit_credits'),
27     path('search/', search, name='search'),
28     path('creerCredit/', creerCredit, name='creerCredit'),]
29 urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)

```

Code Listing 3.3 – Python URLs

**l. Lancer le serveur :** Pour lancer le serveur, exécutez la commande python manage.py runserver dans votre terminal.

## 3.2 Interfaces graphiques

L'interface graphique est une partie Cruciale pour la réalisation d'une application Web convenable et conviviale.

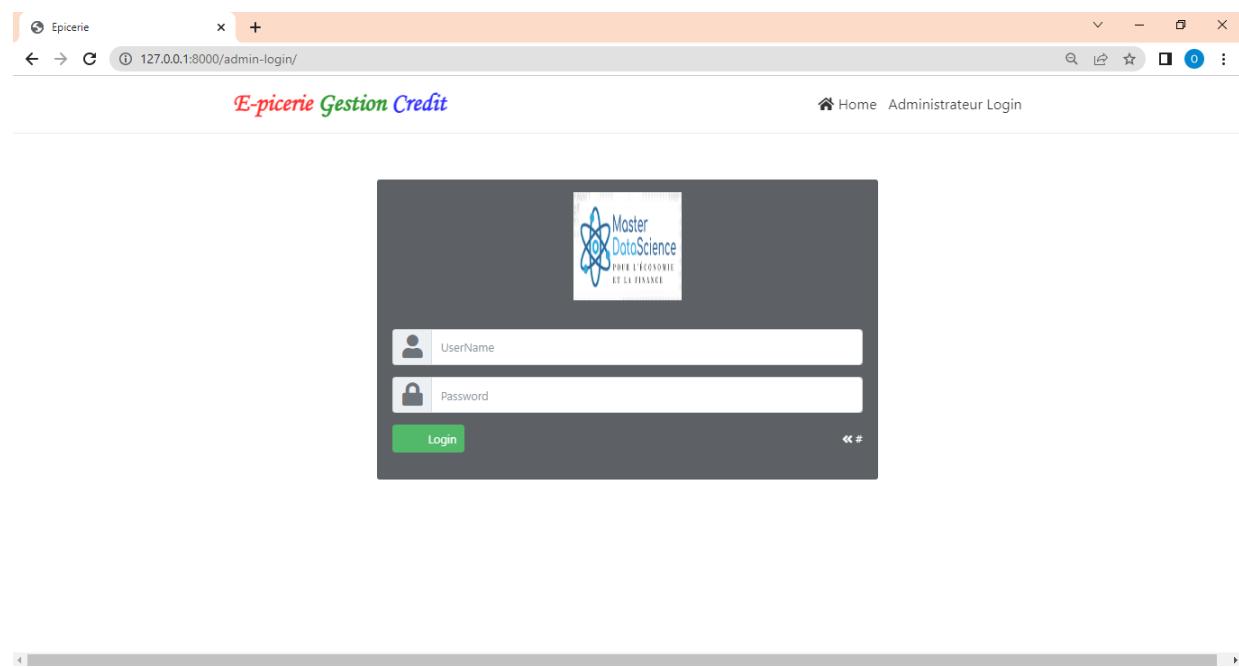
Voici maintenant un ensemble de captures d'écrans sur les principaux points d'entrées de l'application :

### 3.2.1 Interface Authentification

Lors du clic sur le bouton « Login » de l'interface Administrateur Login, une interface d'authentification est affichée. L'utilisateur doit saisir son Login et son mot de passe dans les champs correspondants pour pouvoir accéder aux différentes fonctionnalités de l'application.

Une fois que le client clique sur le bouton « Entrée », le système vérifie les données entrées .En cas d'échec, il réaffiche la page d'authentification avec un message « les données non valides ». Si le Login et le mot de passe sont valides, le système passe au menu principale. Ainsi ,l'application permet à l'utilisateur de changer son mot de passe et aussi se déconnecte sa session.

**Figure 3.2 : Interface Authentification**



**Source :** Élaboré par nos soins à partir des sorties Django

**Figure 3.3 :** Interface de la tentative de connexion

**Source :** Élaboré par nos soins à partir des sorties Django

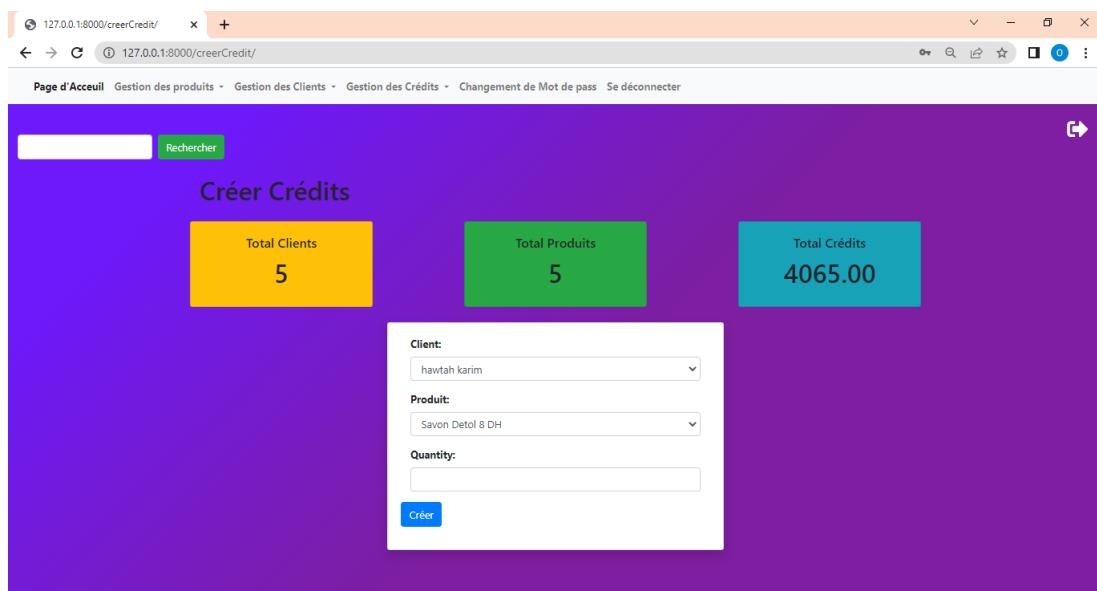
**Figure 3.4 :** Interface du changement de mot de passeA screenshot of a web application interface. The top navigation bar includes links for 'Page d'Accueil', 'Gestion des produits', 'Gestion des Clients', 'Gestion des Crédits', 'Changement de Mot de passe', and 'Se déconnecter'. Below the navigation is a purple header bar with a search input field and a 'Rechercher' button. The main content area has a purple gradient background and contains a form titled 'Changement Mot de Passe'. It includes three text input fields labeled 'Ancien Mot de Passe', 'Nouveau Mot de Passe', and 'Confirmer le Mot de Passe', each with its own validation message below it. A blue 'Submit' button is at the bottom of the form.

**Source :** Élaboré par nos soins à partir des sorties Django

### 3.2.2 Interface de la page d'accueil

La page d'accueil constitue la porte d'entrée de notre application et une fois l'utilisateur accède à son application il trouve la page d'entrée avec plusieurs fonctions qui facilite à lui les taches de gestion de credit. Tels que, la création de crédit sert que l'utilisateur pouvait créer le crédit à partir de la sélection du client concerné. Ainsi les produits et les quantités demandées par le clients. Aussi, la page d'accueil permet à l'utilisateur de savoir les totaux des produits et des crédits.

**Figure 3.5 :** Interface de la page d'accueil



**Source :** Élaboré par nos soins à partir des sorties Django

### 3.2.3 Interface Gestion des clients

Cette interface permet les ajouts, les modifications et les suppressions et des différents clients.

**Figure 3.6 :** Interface d'ajout de clients

Ajouter Clients

Nom:

Prenom:

Telephone:

Ajouter client

**Source :** Élaboré par nos soins à partir des sorties Django

**Figure 3.7 :** Interface de la modification des clients

Edit Client

Nom: hawtah

Prénom: karim

Téléphone: 06000001

Enregistrer

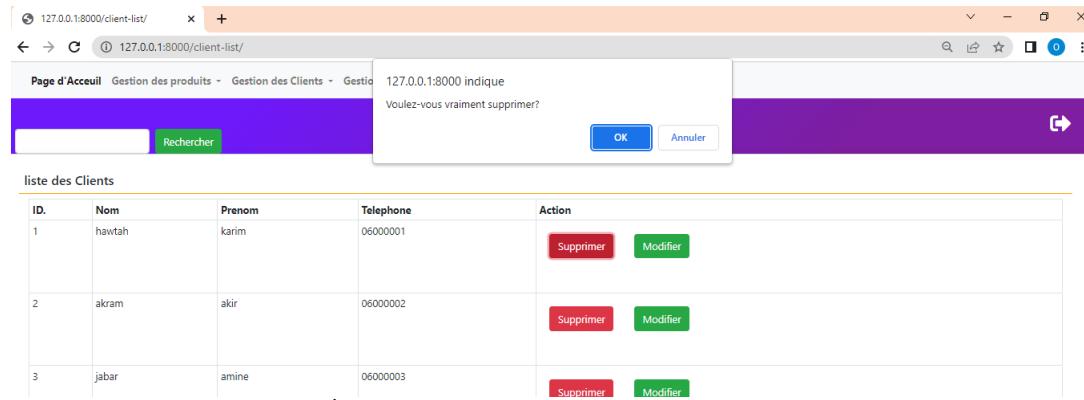
**Source :** Élaboré par nos soins à partir des sorties Django

**Figure 3.8 :** Interface d'affichage les coordonnées des clients

liste des Clients

ID.	Nom	Prenom	Telephone	Action
1	hawtah	karim	06000001	<button>Supprimer</button> <button>Modifier</button>
2	akram	akir	06000002	<button>Supprimer</button> <button>Modifier</button>
3	jabar	amine	06000003	<button>Supprimer</button> <button>Modifier</button>

**Source :** Élaboré par nos soins à partir des sorties Django

**Figure 3.9 : Interface de la Suppression des clients**

**Source :** Élaboré par nos soins à partir des sorties Django

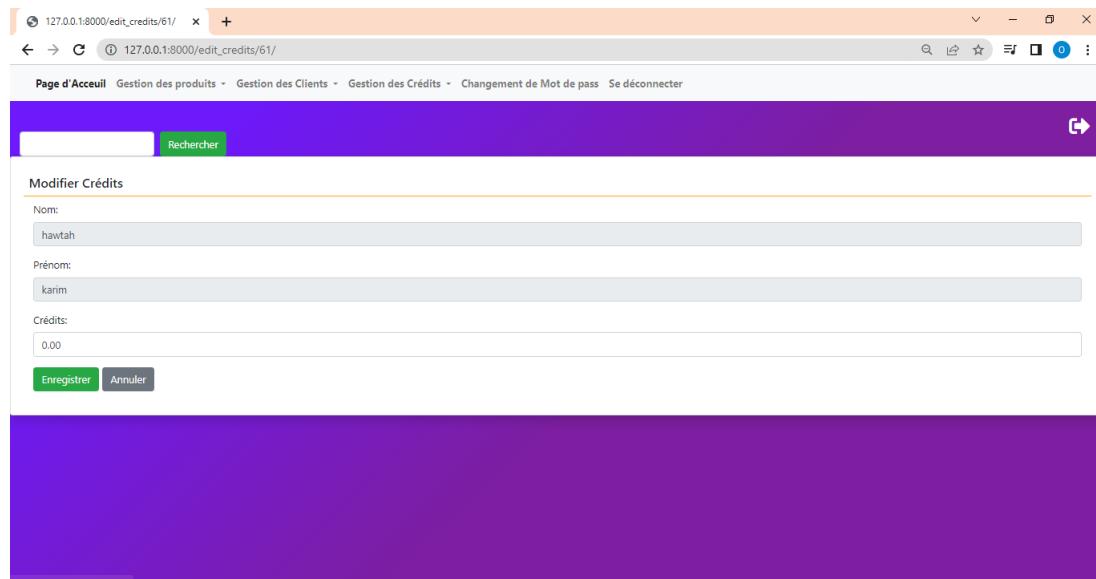
### 3.2.4 Interface Gestion des crédits

Cette interface permet à l'utilisateur de modifier les crédits associés à leurs clients, de consulter leurs soldes et de voir leurs historiques de crédits selon la date création de crédits accordés aux différents clients. Et aussi permet aux utilisateurs de savoir les statuts de leurs client en fonction de la niveau de crédit si le totale de crédit d'une client est supérieure à 2000 DH, ce client sera considérée avec une statut trop risqué si non moins risqué.

**Figure 3.10 : Interface d'affichage de la liste des crédits accordés aux clients**

ID.	Nom	Prénom	Crédits	Date	Action	Status
73	hawtah	karim	1283.00	Feb. 22, 2023, 7:58 p.m.	<button>Modifier Crédits</button>	Risqué
74	akram	akir	40.00	Feb. 22, 2023, 7:58 p.m.	<button>Modifier Crédits</button>	Moins Risqué
75	jabar	amine	2582.00	Feb. 22, 2023, 7:59 p.m.	<button>Modifier Crédits</button>	Trop Risqué
76	alaoui	said	96.00	Feb. 22, 2023, 7:58 p.m.	<button>Modifier Crédits</button>	Moins Risqué
77	bouchi	ahmed	64.00	Feb. 22, 2023, 7:58 p.m.	<button>Modifier Crédits</button>	Moins Risqué

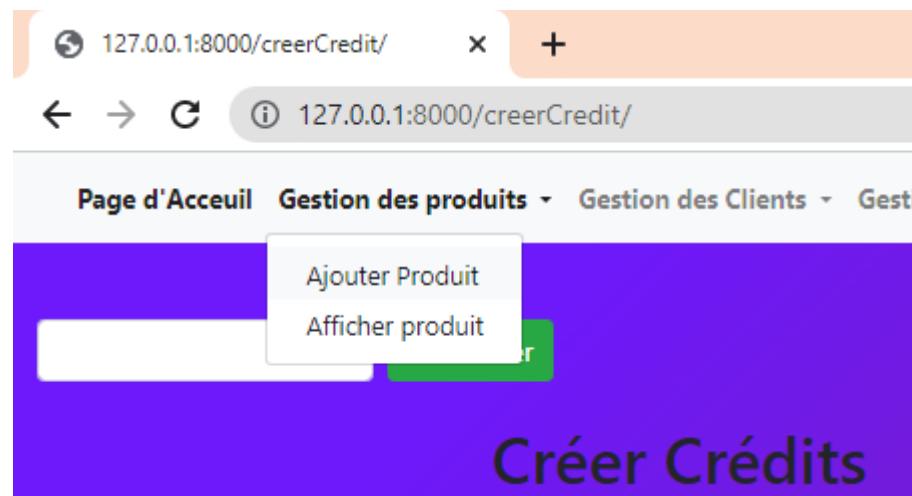
**Source :** Élaboré par nos soins à partir des sorties Django

**Figure 3.11 :** Interface de la modification crédits des clients

**Source :** Élaboré par nos soins à partir des sorties Django

### 3.2.5 Interface Gestion des produits

Cette interface permet à l'utilisateur d'ajouter ou modifier ou supprimer ses produits ,de saisir leur prix, leur description, Ainsi de voir leurs classifications.

**Figure 3.12 :** Interface Gestion des produits

**Source :** Élaboré par nos soins à partir des sorties Django

**Figure 3.13 :** Interface d'ajouts produits

**Source :** Élaboré par nos soins à partir des sorties Django

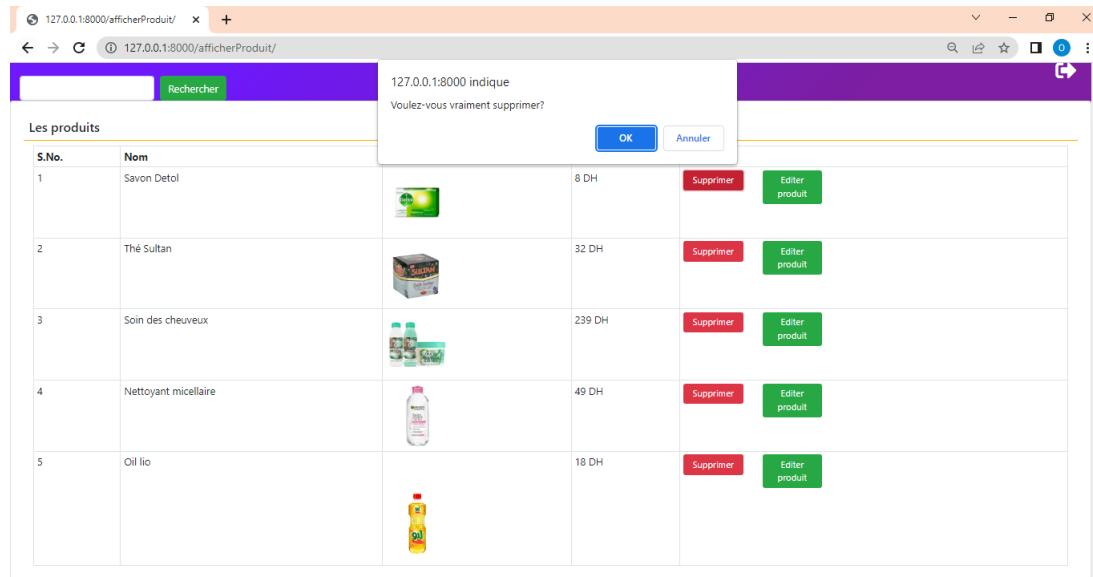
**Figure 3.14 :** Interface de modification des produits

**Source :** Élaboré par nos soins à partir des sorties Django

**Figure 3.15 :** D'affichage de la liste des produits

S.No.	Nom	Image	Prix	Action
1	Savon Detol		8 DH	<button>Supprimer</button> <button>Editer produit</button>
2	Thé Sultan		32 DH	<button>Supprimer</button> <button>Editer produit</button>
3	Soin des cheveux		239 DH	<button>Supprimer</button> <button>Editer produit</button>
4	Nettoyant micellaire		49 DH	<button>Supprimer</button> <button>Editer produit</button>
5	Oil lio		18 DH	<button>Supprimer</button> <button>Editer produit</button>

**Source :** Élaboré par nos soins à partir des sorties Django

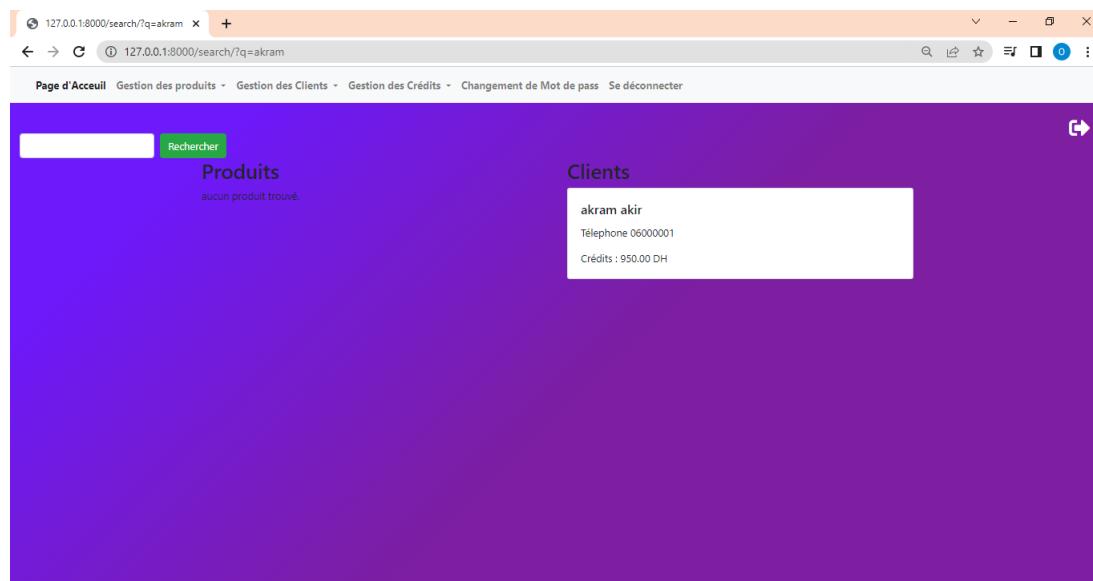
**Figure 3.16 :** Interface de la suppression des produits

**Source :** Élaboré par nos soins à partir des sorties Django

## 3.2.6 Recherche rapide

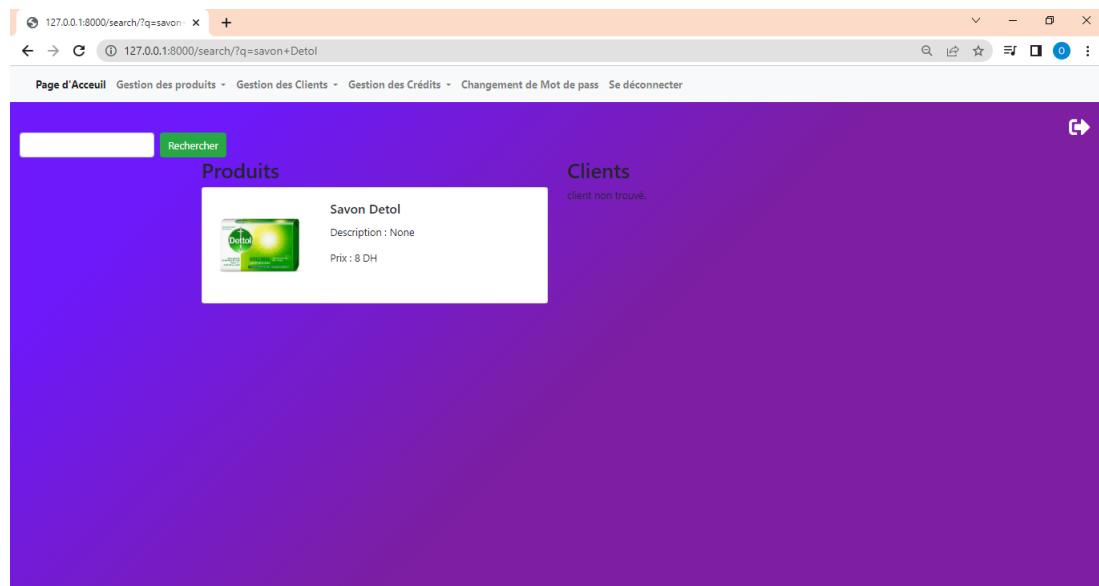
Cette méthode permet à l'utilisateur de trouver facilement le client recherché ,de consulter ses données et son total de crédit d'une manière rapide. La même chose, l'utilisateur pouvait consulter les caractéristiques de chaque produits.

### 3.2.6.1 Recherche par clients



**Source :** Élaboré par nos soins à partir des sorties Django

### 3.2.6.2 Recherche par produit



**Source :** Élaboré par nos soins à partir des sorties Django

## Conclusion

La dernière partie de ce projet était dédiée à la navigation dans notre application .Cette partie constitue le dernier volet de ce rapport. Elle a pour objet de présenter les différents outils technologiques utilisée et les étapes de réalisation durant ce projet. Elle a été clôturée par l'exposition des captures écrans décrivant tous les interfaces de notre application.

## Conclusion générale

Tout au long de ce rapport, nous avons présenté les différentes phases de réalisation de notre application « la gestion de crédit d'épicerie » à l'aide de diagramme de classe et des outils technologiques et des logiciels développés .

Ce projet nous a donné l'opportunité de s'initier à la vie professionnelle dans un milieu réel et avoir un début d'expérience significatif. Cependant, cette expérience nous a donné l'opportunité de créer une application peut faire référence à la possibilité d'élaborer une application qui répond aux besoins d'un marché ou d'une audience spécifique, ou qui offre une solution à un problème particulier. Il était une occasion pour mettre en évidence et déployer sur le plan pratique nos connaissances en informatique.

# Bibliographie

## Articles

1. BURCH, Carl (2010). "Django, a web framework using python : Tutorial presentation". In : *Journal of Computing Sciences in Colleges* 25.5, p. 154-155.

## Conférence

1. VAN ROSSUM, Guido et al. (2007). "Python Programming Language." In : *USENIX annual technical conference*. T. 41. 1. Santa Clara, CA, p. 1-36.

## Ouvrages

1. CHRISTUDAS, Binildas et CHRISTUDAS, Binildas (2019). *MySQL*. Springer.
2. DAUZON, Samuel, BENDORAITIS, Aidas et RAVINDRAN, Arun (2016). *Django : web development with Python*. Packt Publishing Ltd.
3. FORCIER, Jeff, BISSEX, Paul et CHUN, Wesley J (2008). *Python web development with Django*. Addison-Wesley Professional.
4. GREENSPAN, Jay et BULGER, Brad (2001). *MySQL/PHP database applications*. John Wiley & Sons, Inc.
5. LUTZ, Mark (2001). *Programming python*. " O'Reilly Media, Inc.".
6. — (2010). *Programming python : powerful object-oriented programming*. " O'Reilly Media, Inc.".
7. RUBIO, Daniel (2017). *Beginning Django*. Springer.

## Webographie

- |   |   |
|---|---|
| [1] <a href="https://themewagon.com">https://themewagon.com</a>       | [5] <a href="https://dev.to">https://dev.to</a>                             |
| [2] <a href="https://getbootstrap.com">https://getbootstrap.com</a>   | [6] <a href="http://localhost/phpmyadmin/">http://localhost/phpmyadmin/</a> |
| [3] <a href="https://www.youtube.com/">https://www.youtube.com/</a>   | [7] <a href="https://github.com/">https://github.com/</a>                   |
| [4] <a href="https://www.w3schools.com">https://www.w3schools.com</a> |   |

## TABLE DES MATIÈRES

	Page
<b>Remerciements</b>	
<b>Sommaire</b>	
<b>Table des figures</b>	
<b>Liste des abréviations</b>	
<b>Introduction générale</b>	<b>1</b>
Chapitre 1	
<b>2      Présentation du Projet</b>	
Introduction . . . . .	2
1.1 Présentation du Projet «l'application Web E-épicier» . . . . .	2
1.1.1 Rôle et objectifs de «l'application Web E-épicier» . . . . .	2
1.2 Analyse des besoins . . . . .	3
1.2.1 Les besoins fonctionnels . . . . .	3
1.2.2 Les besoins non fonctionnels . . . . .	4
Conclusion . . . . .	4
Chapitre 2	
<b>5      Analyse conceptuelle</b>	
Introduction . . . . .	5
2.1 Méthodologie d'analyse . . . . .	5
2.1.1 Le Langage UML . . . . .	5
2.2 Outils utilisés dans la conception . . . . .	6
2.2.1 Logiciels Utilisés . . . . .	6
2.2.2 Conception des données . . . . .	6
2.2.2.1 Diagramme de classe . . . . .	6
2.2.2.2 Description de diagramme de classe . . . . .	8

Conclusion . . . . .	8
<b>Chapitre 3</b>	
<b>9 la démarche de la Réalisation du projet</b>	
Introduction . . . . .	9
3.1 Outils et technologies utilisés . . . . .	9
3.1.1 Environnement logiciel . . . . .	9
3.1.2 Les étapes de réalisation du projet avec Django . . . . .	10
3.2 Interfaces graphiques . . . . .	17
3.2.1 Interface Authentification . . . . .	17
3.2.2 Interface de la page d'accueil . . . . .	19
3.2.3 Interface Gestion des clients . . . . .	20
3.2.4 Interface Gestion des crédits . . . . .	21
3.2.5 Interface Gestion des produits . . . . .	22
3.2.6 Recherche rapide . . . . .	24
3.2.6.1 Recherche par clients . . . . .	24
3.2.6.2 Recherche par produit . . . . .	25
Conclusion . . . . .	25
<b>Conclusion générale</b>	26
<b>Bibliographie</b>	27
<b>Table des matières</b>	28