



FSJES Tanger

Master Spécialisé :

DATA SCIENCE EN ECONOMIE ET FINANCE

Rapport sous thème :

Création d'application web

E-épicier

Réalisé par :

ACHARGUI AFKIR AYMANE

TRIBAK MOHAMMED

YAHIAOUI ANASS

Encadré par :

MR. EL AACHAK LOTFI

Année scolaire 2022/2023

Table des matières

Table de matiers	2
Introduction.....	3
I. Les outils pour créer l'App Web	4
1. Figma	4
2. Django.....	4
3. Bootstrap	5
4. MySQL.....	6
II. Les étapes de l'installation des Frameworks	7
1. Installation de Django	7
1.1. Virtual environnement	7
1.2. Installation de Virtualenv	8
1.3. Création d'un environnement virtuel.....	8
1.4. Activation de l'environnement virtuel.....	8
1.5. Installation de Django	8
2. Vérification de l'installation de Django	8
3. Création d'un projet en utilisant le Framework Django	8
4. Liaison de Django et MySQL.....	9
5. Remplissage de la base de données	10
6. Création d'une application en utilisant le Framework Django et création de base de données de notre projet	11
III. Les étapes de création de la base de données de notre projet	13
1. Étapes à suivre pour créer l'authentification et le login et mot de passe	13
2. Étapes à suivre pour créer une application qui permet aux épiciers à la gestion des clients, des produits et articles, et la gestion des crédits par clients	15
Conclusion	21
Bibliographie.....	22

INTRODUCTION

Dans le cadre de ce projet, nous avons développé une application web full stack basée sur le Framework Django. L'objectif de l'application web est de permettre aux épiciers de gérer automatiquement les crédits de leurs clients. Pour ce faire, nous avons défini un cahier des charges qui spécifie les fonctionnalités attendues de l'application web. Nous avons également sélectionné les outils nécessaires pour la réalisation de ce projet, notamment Django, MySQL, Bootstrap et Figma.

I. Les outils pour créer l'App Web

Pour créer une application web e-épicer, on va utiliser quelques outils et Frameworks pour succéder notre projet, parmi les outils on cite :

1. Figma

Figma est un outil de conception d'interface utilisateur (UI) qui permet aux concepteurs de créer des maquettes de conception pour des applications web et mobiles. Il permet aux équipes de travail de collaborer en temps réel sur des designs et des prototypes, ce qui facilite la communication et la coordination entre les membres de l'équipe.

Le rôle de Figma dans la création d'une application web est de permettre aux designers de concevoir l'apparence visuelle de l'application en utilisant une interface graphique conviviale. Les designers peuvent créer des maquettes de conception de haute qualité qui incluent des éléments tels que des boutons, des menus, des formulaires, des icônes et des images.

En utilisant Figma, les designers peuvent également créer des prototypes interactifs qui simulent le fonctionnement de l'application. Les prototypes peuvent être partagés avec les clients et les membres de l'équipe pour obtenir des commentaires et des suggestions avant de passer à la phase de développement.

En somme, Figma facilite la conception et la communication des idées de conception pour les applications web, ce qui permet aux équipes de travail de collaborer efficacement et de créer des designs de qualité supérieure

2. Django

Django est un Framework web open-source écrit en Python. Il permet aux développeurs de créer rapidement des applications web robustes et évolutives en fournissant un ensemble de fonctionnalités prêtes à l'emploi, telles que la gestion des bases de données, la gestion des URL, l'authentification des utilisateurs, la génération de formulaires, et bien plus encore.

Le rôle de Django dans la création d'une application web est de faciliter le processus de développement en fournissant une structure et une organisation claires pour les applications web. Django suit le modèle MVC (Modèle-Vue-Contrôleur), qui sépare les préoccupations de

l'application en trois parties distinctes : le modèle qui gère les données, la vue qui affiche les données à l'utilisateur, et le contrôleur qui gère les interactions entre l'utilisateur et l'application.

En utilisant Django, les développeurs peuvent créer des applications web efficacement en utilisant des fonctionnalités de base telles que l'ORM (Object-Relational Mapping) pour interagir avec les bases de données, l'API de routage pour gérer les URL, et les systèmes d'authentification pour sécuriser l'application.

En somme, Django est un Framework web populaire qui facilite la création d'applications web robustes, évolutives et sécurisées. Il permet aux développeurs de se concentrer sur le logique métier de leur application plutôt que sur les détails techniques du développement web.

3. Bootstrap

Bootstrap est un Framework front-end open-source développé par Twitter. Il fournit un ensemble de composants prêts à l'emploi pour la création d'interfaces utilisateur (UI) pour les applications web, ainsi que des outils pour la mise en page, la typographie, les formulaires, les boutons, les icônes, les modales, les menus déroulants, les onglets et bien plus encore.

Le rôle de Bootstrap dans la création d'une application web est de simplifier la conception et la mise en page de l'interface utilisateur. Il fournit des styles CSS préconçus et des composants HTML pour les développeurs, ce qui permet de gagner du temps et de réduire les efforts de développement. Les développeurs peuvent utiliser les classes prédéfinies de Bootstrap pour appliquer des styles à leur HTML sans avoir à écrire des styles CSS personnalisés.

Bootstrap est également connu pour être compatible avec les différents navigateurs, les appareils mobiles et les résolutions d'écran, ce qui permet de concevoir des interfaces utilisateur adaptatives et réactives pour différents types de dispositifs.

En somme, Bootstrap est un framework front-end populaire pour la conception d'interfaces utilisateur pour les applications web. Il permet aux développeurs de créer des interfaces utilisateur modernes, adaptatives et réactives plus rapidement et plus facilement, en utilisant des composants et des styles prêts à l'emploi.

4. MySQL

MySQL est un système de gestion de base de données relationnelles open-source qui permet aux développeurs de stocker, organiser et récupérer des données pour les applications web. Il utilise le langage SQL (Structured Query Language) pour interagir avec la base de données.

Le rôle de MySQL dans la création d'une application web est de fournir un système de gestion de base de données qui permet de stocker et d'organiser efficacement les données de l'application. Il permet aux développeurs de créer des tables, des vues et des index pour stocker les données, ainsi que des requêtes pour récupérer et manipuler les données.

MySQL est un système de gestion de base de données populaire pour les applications web en raison de sa capacité à gérer efficacement les bases de données de grande taille et de sa fiabilité. Il prend en charge des fonctionnalités avancées telles que la réplication des bases de données, les transactions ACID (Atomicité, Cohérence, Isolation, Durabilité) et la gestion des verrous pour assurer l'intégrité des données.

En somme, MySQL joue un rôle crucial dans la création d'applications web en fournissant un système de gestion de base de données robuste et performant. Il permet aux développeurs de stocker et d'organiser efficacement les données de l'application, tout en garantissant l'intégrité des données et la cohérence de l'application.

- **Nb :** Dans notre cas on a suffi d'utiliser Bootstrap dans le côté design de l'application web.

II. Les étapes de l'installation des Frameworks

1. Installation de Django

Django est un Framework web populaire et open source écrit en Python. Pour l'installer, il est recommandé d'utiliser un environnement virtuel. Pour ce faire, nous avons utilisé l'outil "virtualenv". Voici les étapes que nous avons suivies pour installer Django, mais avant il faut définir l'environnement où on doit installer, lancer et créer notre projet sur Django qui le Virtual environnement:

1.1. Virtual environnement

Un environnement virtuel (ou Virtual environment en anglais) est un outil de développement logiciel qui permet d'isoler les dépendances et les bibliothèques d'un projet Python spécifique de l'installation globale de Python sur votre ordinateur.

Lorsque vous installez des bibliothèques Python sur votre système global, celles-ci peuvent entrer en conflit avec d'autres bibliothèques ou versions de Python installées sur votre ordinateur. Cela peut causer des erreurs de compatibilité et rendre le développement plus difficile.

Les environnements virtuels permettent de contourner ce problème en créant une copie isolée de Python et de ses bibliothèques. Vous pouvez alors installer les bibliothèques nécessaires à votre projet dans cet environnement virtuel sans affecter les autres projets ou applications Python sur votre ordinateur.

Cela offre plusieurs avantages, tels que :

Une meilleure gestion des dépendances et des versions des bibliothèques

La possibilité de travailler sur plusieurs projets Python en même temps, chacun avec son propre environnement virtuel

La portabilité du projet, car vous pouvez facilement partager l'environnement virtuel avec d'autres développeurs ou machines sans avoir à installer manuellement toutes les bibliothèques requises.

En somme, l'utilisation d'environnements virtuels facilite la gestion des dépendances et contribue à une meilleure organisation du développement Python.

1.2. Installation de Virtualenv

```
pip install virtualenv
```

1.3. Création d'un environnement virtuel

```
Virtualenv nom_de_votre_environnement
```

Ou bien

```
Python -m venv nom_de_votre_environnement
```

1.4. Activation de l'environnement virtuel

```
nom_de_votre_environnement/Scripts/activate
```

1.5. Installation de Django

```
pip install django
```

2. Vérification de l'installation de Django

Une fois que Django est installé, vous pouvez vérifier si l'installation a réussi en exécutant la commande suivante dans un terminal :

```
Django-admin --version
```

Ou bien:

```
pip freeze
```

3. Création d'un projet en utilisant le Framework Django

Une fois l'installation de Django est vérifiée, on peut maintenant créer un projet en suivant les commandes suivantes:

```
django-admin startproject "nom du projet"
```


4. Liaison de Django et MySQL

Pour lier Django et MySQL il faut accéder au **settings.py** du projet déterminé et on va modifier **sqlite3** par **MySQL**; et on va changer 'NAME': BASE_DIR / 'db.sqlite3' par 'Django';

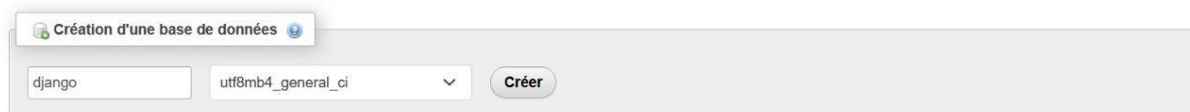
On ajoute 'USER' défini par défaut par 'root', 'PASSWORD' est facultatif ça dépend l'utilisateur, et le HOST est toujours défini par 'localhost', si on trouvera un problème de nom de localhost on peut le modifier par l'adresse 127.0.0.1

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}
```

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'django',
        'USER': 'root',
        'PASSWORD': '',
        'HOST': 'localhost',
        'PORT': '',
    }
}
```

On doit créer une base de données nommée django pour ne pas tomber dans l'erreur :

Bases de données



Création d'une base de données

django utf8mb4_general_ci Créer

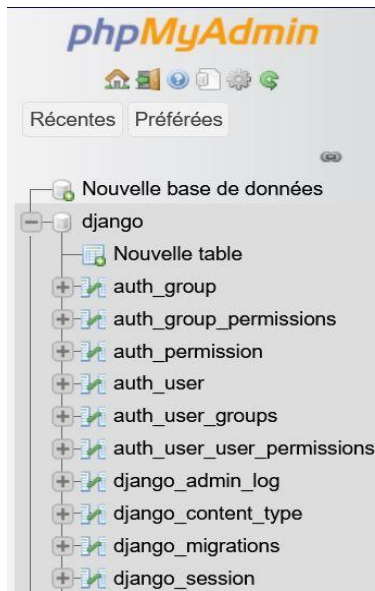
Après cette modification, on doit utiliser ce syntaxe pour appliquer la migration c'est à dire la création des tables dans la base de données

```
pip install mysqlclient
```

Maintenant, on peut ajouter des tables en utilisant la commande ci-dessous:

```
python manage.py migrate
```

Et on trouve ensuite les tables seront affichées dans la base de données :



The screenshot shows the phpMyAdmin interface. On the left, a sidebar lists the database 'django' and its tables: auth_group, auth_group_permissions, auth_permission, auth_user, auth_user_groups, auth_user_user_permissions, django_admin_log, django_content_type, django_migrations, and django_session. On the right, a table structure view displays the following data:

Table	Action	Lignes	Type	Interclassement	Taille	Perte
auth_group	Parcourir Structure Rechercher Insérer Vider Supprimer	0	InnoDB	utf8mb4_general_ci	32,0 kio	-
auth_group_permissions	Parcourir Structure Rechercher Insérer Vider Supprimer	0	InnoDB	utf8mb4_general_ci	48,0 kio	-
auth_permission	Parcourir Structure Rechercher Insérer Vider Supprimer	36	InnoDB	utf8mb4_general_ci	32,0 kio	-
auth_user	Parcourir Structure Rechercher Insérer Vider Supprimer	4	InnoDB	utf8mb4_general_ci	32,0 kio	-
auth_user_groups	Parcourir Structure Rechercher Insérer Vider Supprimer	0	InnoDB	utf8mb4_general_ci	48,0 kio	-
auth_user_user_permissions	Parcourir Structure Rechercher Insérer Vider Supprimer	4	InnoDB	utf8mb4_general_ci	48,0 kio	-
django_admin_log	Parcourir Structure Rechercher Insérer Vider Supprimer	41	InnoDB	utf8mb4_general_ci	48,0 kio	-
django_content_type	Parcourir Structure Rechercher Insérer Vider Supprimer	9	InnoDB	utf8mb4_general_ci	48,0 kio	-
django_migrations	Parcourir Structure Rechercher Insérer Vider Supprimer	24	InnoDB	utf8mb4_general_ci	16,0 kio	-
django_session	Parcourir Structure Rechercher Insérer Vider Supprimer	2	InnoDB	utf8mb4_general_ci	32,0 kio	-

5. Remplissage de la base de données

Pour notre application web, nous avons utilisé MySQL comme système de gestion de base de données. Pour remplir la base de données, nous avons créé un script Python qui lit un fichier CSV contenant des données de test et les insère dans la base de données. Voici les étapes que nous avons suivies :

1- Installation de la bibliothèque MySQL pour Python :

```
pip install mysql-connector-python
```

2- Création d'une base de données dans MySQL.

3- Création des tables de la base de données en utilisant les modèles Django.

4- Écriture d'un script Python pour lire le fichier CSV et insérer les données dans la base de données.

6. Création d'une application en utilisant le Framework Django et création de base de données de notre application :

Pour créer une application sur Django, il faut suivre les étapes suivantes:

- 1- Naviguons vers le répertoire de notre projet en exécutant la commande suivante:

```
Cd nom_du_projet
```

- 2- Vérifions que le projet fonctionne correctement en exécutant la commande suivante:

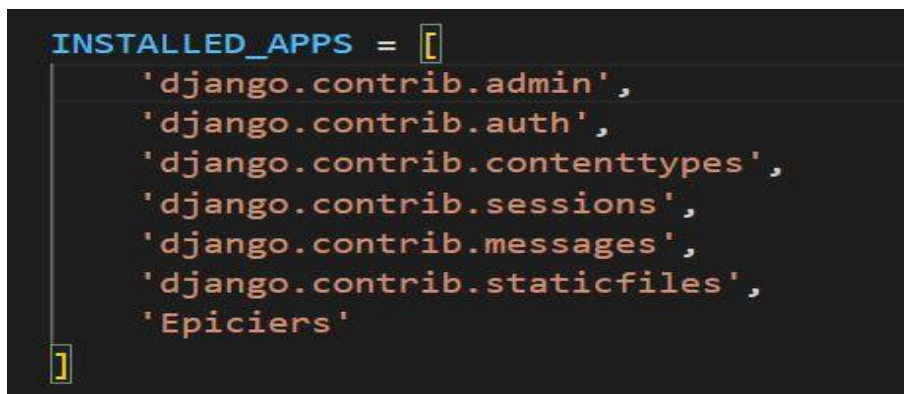
```
Python manage.py runserver
```

Cela démarrera le serveur de développement Django et on pourra accéder à l'application en ouvrant le navigateur à l'adresse <http://127.0.0.1:8000>.

- 3- Maintenant on va créer notre application en exécutant la commande suivante:

```
django-admin startapp myapp
```

Remplaçons myapp(Epiciers) par le nom de l'application. Lorsqu'on crée l'application sur django on doit l'insérer dans un fichier nommé settings.py:



```
INSTALLED_APPS = [  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
    'Epiciers'  
]
```

- 4- On va créer un fichier urls.py dans notre application Epiciers.
- 5- Si vous n'avez pas encore créé de modèle de base de données pour votre application, vous pouvez le faire en modifiant le fichier models.py de votre application. Vous pouvez y ajouter des classes de modèles qui correspondent à des tables de votre base de données.

- 6- On a déjà créé des modèles de base de données pour notre application, on peut effectuer les migrations en exécutant la commande suivante:

```
python manage.py makemigrations
```

ensuite :

```
Python manage.py migrate
```

Cela créera les tables de base de données correspondantes pour nos modèles dans notre base de données MySQL.

- 7- On peut configurer les URLs pour nos vues dans le fichier **urls.py** de notre application.
8- On peut ensuite créer nos vues en modifiant le fichier **views.py** de notre application.

Voilà, on a créé un projet Django en utilisant Django avec MySQL et les migrations ont été effectuées.



III. Les étapes de création de la base de données de notre projet :

La création de l'application web E-épiciier nécessite plusieurs étapes, voici une proposition d'étapes pour la réalisation de ce projet :

1. Étapes à suivre pour créer l'authentification et le login et mot de passe

Tout d'abord, on peut créer une nouvelle application Django en utilisant la commande

```
Python manage.py startapp nom_de_l_application.
```

Dans le fichier models.py de notre application, on va créer un modèle pour stocker les informations d'authentification des utilisateurs. Par exemple :

```
from django.db import models
from django.contrib.auth.models import AbstractUser

class CustomUser(AbstractUser):
```

Ici, nous avons utilisé la classe AbstractUser de Django pour définir un modèle personnalisé pour les utilisateurs.

Ensuite, dans le fichier settings.py de notre projet, on ajoutera notre nouvelle application à la liste des applications installées :

```
INSTALLED_APPS = [ ... 'nom_de_l_application', ]
```

Dans le fichier form.py de notre application, on va créer un formulaire pour l'inscription d'un nouvel utilisateur :

```
from django import forms
from django.contrib.auth.forms import UserCreationForm
from .models import CustomUser

class CustomUserCreationForm(UserCreationForm):

    class Meta:
        model = CustomUser
        fields = ('username', 'email', 'password1', 'password2')
```

Ici, nous avons utilisé la classe `UserCreationForm` de Django pour créer un formulaire de création d'utilisateur personnalisé qui inclut des champs pour le nom d'utilisateur, l'e-mail et le mot de passe.

Dans le fichier `views.py` de notre application, on va créer une vue pour le formulaire d'inscription :

```
from django.shortcuts import render, redirect
from .forms import CustomUserCreationForm

def register(request):
    if request.method == 'POST':
        form = CustomUserCreationForm(request.POST)
        if form.is_valid():
            form.save()
            return redirect('login')
        else:
            form = CustomUserCreationForm()
            return render(request, 'registration/register.html', {'form': form})
```

Ici, nous avons créé une vue `register` qui utilise le formulaire `CustomUserCreationForm` pour enregistrer un nouvel utilisateur. Si le formulaire est valide, l'utilisateur est créé et redirigé vers la page de connexion.

Créer un template HTML pour le formulaire d'inscription :

```
{% extends 'base.html' %}

{% block content %}
    <h2>Inscription</h2>
    <form method="post">
        {% csrf_token %}
        {{ form.as_p }}
        <button type="submit">S'inscrire</button>
    </form>
{% endblock %}
```

Dans le fichier `urls.py` de notre application, on ajoute des URL pour les vues de connexion et d'inscription :

```

from django.urls import path
from django.contrib.auth import views as auth_views
from .views import register

urlpatterns = [
    path('login/',
auth_views.LoginView.as_view(template_name='registration/login.html'),
name='login'),
    path('logout/', auth_views.LogoutView.as_view(), name='logout'),
    path('register/', register, name='register'),
]

```

Ici, nous avons utilisé les vues intégrées de Django LoginView et LogoutView pour gérer la connexion et la déconnexion des utilisateurs, ainsi que notre vue register pour l'inscription.

Créer des templates HTML pour les vues de connexion et de déconnexion :

login.html :

```

{% extends 'base.html' %}
{% block content %}
.
.
{% endblock content %}

```

L'objectif de cette syntaxe est d'extraire tout le code d'une base html vers un nouveau fichier html.

On peut ajouter entre les points un formulaire pour créer un compte et le login et le password à partir du Framework Bootstrap.

On utilise Bootstrap pour styliser les pages d'inscription, de connexion et de profil, en utilisant les classes CSS de Bootstrap pour créer un design cohérent et attrayant.

2. Étapes à suivre pour créer une application qui permet aux épiciers à la gestion des clients, des produits et articles, et la gestion des crédits par clients

Voici les étapes détaillées pour créer une application qui permet aux épiciers à la gestion des clients, gestion des produits et articles et gestion des crédits par clients en utilisant Django, MySQL et Bootstrap :

Définir les modèles de données : Créer des modèles de base de données pour les clients, les produits et articles, et les crédits des clients dans le fichier **models.py**.

```
E_epicie > Epiciers > models.py > ...
1  from django.db import models
2
3  # Create your models here.
4
5
6  class Client(models.Model):
7      nom= models.CharField(max_length=200)
8      adresse=models.CharField(max_length=200)
9      telephone=models.CharField(max_length=10)
10     email=models.EmailField(max_length=254)
11
12     # image=models.ImageField(upload_to='photos', height_field="45", width_field="45")
13
14     montant_de_credit=models.FloatField(null=True)
15     montant_à_payer=models.FloatField(null=True)
16
17     def __str__(self):
18         return self.nom
19
20 class Produit(models.Model):
21     nom=models.CharField(max_length=200)
22     description=models.TextField()
23     catégorie=models.CharField(max_length=200)
24     prix=models.FloatField(null=True)
25     quantité_en_stock=models.FloatField(null=True)
26
27     def __str__(self):
28         return self.nom
```

Définir les vues Django : Créer des vues pour les pages de gestion des clients, produits et articles, et crédits des clients dans le fichier **views.py**. Ces vues doivent interagir avec la base de données MySQL en utilisant les modèles de données.

```
def client(request):
    clients = Client.objects.all()
    context={"clients":clients}
    return render(request,'client.html',context)

def ajouter_client(request):
    form=ClientForm()
    if request.method=='POST':
        form=ClientForm(request.POST)
        if form.is_valid():
            form.save()
            return redirect('client')
    context={'form':form}
    return render(request,'ajouter_client.html',context)

def modifier_client(request,pk):
    client = Client.objects.get(id=pk)
    form=ClientForm(instance=client)
    if request.method=='POST':
        form=ClientForm(request.POST,instance=client)
        if form.is_valid():
            form.save()
            return redirect('client')
    context={'form':form}
    return render(request,'ajouter_client.html',context)

def supprimer_client(request,pk):
    client = Client.objects.get(id=pk)
    if request.method=='POST':
        client.delete()
        return redirect('client')
    context={'item':client}

    return render(request,'supprimer_client.html',context)
```



```

def produit(request):
    produits = Produit.objects.all()
    context={"produits":produits}
    return render(request,'produit.html',context)

def ajouter_produit(request):
    form=ProduitForm()
    if request.method=='POST':
        form=ProduitForm(request.POST)
        if form.is_valid():
            form.save()
            return redirect('produit')
    context={'form':form}
    return render(request,'ajouter_produit.html',context)

def modifier_produit(request,pk):
    produit = Produit.objects.get(id=pk)
    form=ProduitForm(instance=produit)
    if request.method=='POST':
        form=ProduitForm(request.POST,instance=produit)
        if form.is_valid():
            form.save()
            return redirect('produit')
    context={'form':form}
    return render(request,'ajouter_produit.html',context)

def supprimer_produit(request,pk):
    produit = Produit.objects.get(id=pk)
    if request.method=='POST':
        produit.delete()
        return redirect('produit')
    context={'item':produit}

    return render(request,'supprimer_produit.html',context)

```

Créer les URLS : Définir les URLs pour les pages de gestion des clients, produits et articles, et crédits des clients dans le fichier **urls.py**.

```

from django import views
from django.urls import path
from Epiciers import views
from django.contrib.auth import views as auth_view

urlpatterns = [
    path('',views.home,name='home'),
    path('login',views.logIn,name='login'),
    path('register',views.register,name='register'),
    path('contacte/',views.contacte,name='contacte'),
    path('client/',views.client,name='client'),
    path('logout/', auth_view.LogoutView.as_view(next_page='login'),name='logout'),
    path('profile/',views.profile,name='profile'),
    path('ac/',views.ajouter_client,name='ac'),
    path('mc/<str:pk>',views.modifier_client,name='mc'),
    path('sc/<str:pk>',views.supprimer_client,name='sc'),
    path('produit',views.produit,name='produit'),
    path('ap/',views.ajouter_produit,name='ap'),
    path('mp/<str:pk>',views.modifier_produit,name='mp'),
    path('sp/<str:pk>',views.supprimer_produit,name='sp'),
    path('a_propos',views.a_propos,name='a_propos')
]

```

Créer les Template HTML : Créer des Template HTML pour les pages de gestion des clients, produits et articles, et crédits des clients en utilisant Bootstrap.

```
▼ templates
  <> a_propos.html
  <> ajouter_client.html
  <> ajouter_produit.h...
  <> base.html
  <> client.html
  <> compte.html
  <> contacte.html
  <> home.html
  <> login.html
  <> produit.html
  <> register.html
  <> supprimer_client....
  <> supprimer_produi...
```

Définir les formulaires : Créer des formulaires pour la saisie et la modification des informations clients, produits et articles, et crédits des clients en utilisant Django Forms dans le fichier **forms.py**.

```
from django import forms
from django.contrib.auth.forms import UserCreationForm
from django.contrib.auth.models import User

class UserForm(UserCreationForm):
    class Meta:
        model = User
        fields = [
            'username',
            'email',
            'first_name',
            'last_name',
            'password1',
            'password2',
        ]
```

```
from django.forms import ModelForm
from .models import Client
from .models import Produit

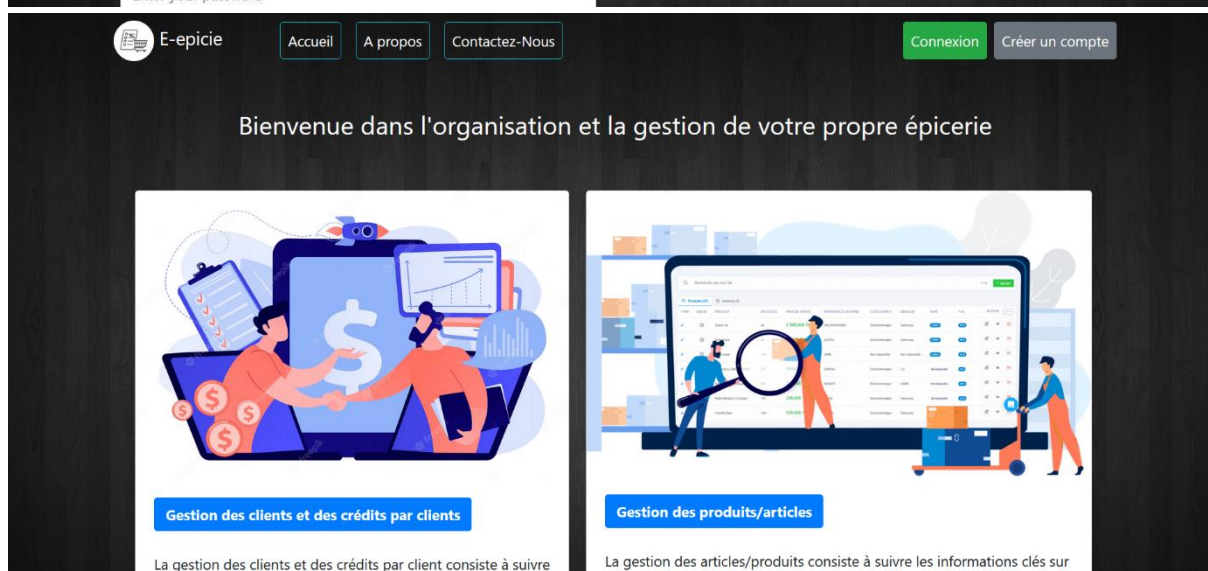
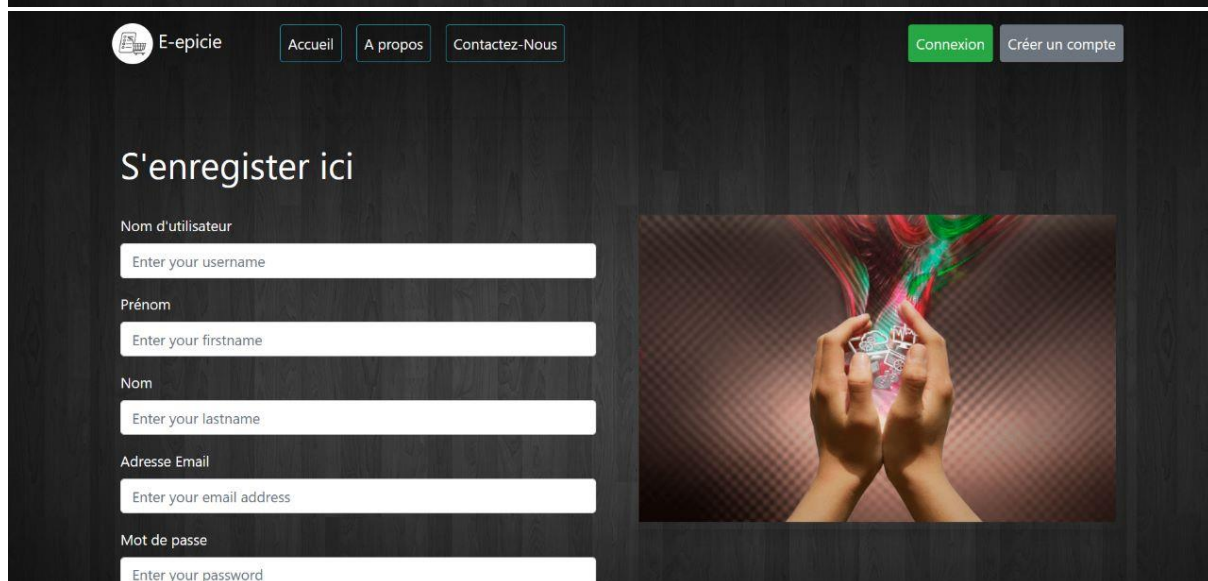
class ClientForm(ModelForm):
    class Meta:
        model=Client
        fields='__all__'


class ProduitForm(ModelForm):
    class Meta:
        model=Produit
        fields='__all__'
```

Mettre en place la logique de recherche : Définir la logique de recherche pour permettre aux utilisateurs de rechercher des clients, produits et articles, et crédits des clients par nom, référence, etc.


Tester et déployer : Tester les fonctionnalités et déployer l'application sur un serveur web pour permettre aux utilisateurs d'accéder à l'application.

- Quelques captures d'écran pour le projet réalisé :





E-epicie


[Accueil](#)
[A propos](#)
[Contactez-Nous](#)


Med


CRÉER UNE FICHE CLIENT

[AJOUTER UN CLIENT](#)



#ID	Nom	Adresse	Telephone	Email	Montant de Credit	Montant à Payer	Mise à jour	Supprimer
15	Mohammed	@@@@@@@@@	0659936078	mohamed.tribak01@gmail.com	0.0	2000.0	MODIFIER	SUPPRIMER
16	Anass	@@@@@@@@@	0630941218	anasslmot@gmail.com	0.0	2000.0	MODIFIER	SUPPRIMER
17	Ayman	@@@@@@@@@	0615559228	achergui1810@gmail.com	0.0	2000.0	MODIFIER	SUPPRIMER


E-epicie


[Accueil](#)
[A propos](#)
[Contactez-Nous](#)


Med

CRÉER UNE FICHE PRODUIT

[AJOUTER UN PRODUIT](#)


#ID	Nom	Description	Catégorie	Prix	Quantité en stock	Quantité vendues	Mise à jour	Supprimer
5	X	X	X	50.0	199.0	199.0	MODIFIER	SUPPRIMER


E-epicie

[Accueil](#)
[A propos](#)
[Contactez-Nous](#)

[Connexion](#)
[Créer un compte](#)

Epicie est un site web e-épicerie qui a pour objectif de simplifier la gestion de votre épicerie en ligne. Notre plateforme vous permet de gérer efficacement vos clients, vos produits et vos articles, ainsi que les crédits accordés à vos clients. En tant que gestionnaire de votre épicerie, vous avez accès à une interface conviviale et facile à utiliser qui vous permet de suivre, vos clients, vos stocks et vos ventes en temps réel. Vous pouvez également ajouter de nouveaux produits, mettre à jour les prix et les descriptions. Grâce à notre système de gestion des clients, vous pouvez facilement suivre s, vérifier l'historique d'achat de vos clients, gérer leurs crédits et ajuster leurs informations de profil. En somme, Epicie est la solution idéale pour les épiciers qui cherchent à gérer efficacement leur activité en ligne. Notre plateforme est facile à utiliser, flexible et évolutive pour s'adapter aux besoins de votre entreprise. Essayez Epicie dès aujourd'hui et commencez à gérer votre épicerie de manière plus efficace et plus rentable.

CONCLUSION

L'application web E-épiciier développée avec Django permet aux épiceries de gérer efficacement les informations relatives à leurs clients et à leurs articles. En utilisant cette application, les épiceries peuvent facilement suivre les ventes, les stocks et les préférences des clients.

Grâce à la facilité d'utilisation de Django, les épiceries peuvent créer des interfaces utilisateur conviviales pour leurs clients, ainsi que des tableaux de bord pour gérer les ventes et les stocks. Les fonctionnalités avancées, telles que la gestion des paiements, la personnalisation des rapports et l'intégration de diverses méthodes de paiement, rendent l'application encore plus pratique et utile pour les épiceries.

En somme, l'application web E-épiciier permet aux épiceries de gérer efficacement leur entreprise, de mieux comprendre leurs clients et de fournir un service de qualité supérieure. Elle peut être une solution rentable pour les épiceries qui souhaitent moderniser leur système de gestion et offrir une expérience de magasinage en ligne à leurs clients.

BIBLIOGRAPHIE

- <https://www.youtube.com/watch?v=eDwQxJ9zvco&list=PLh-rUZWaw76Grm5uRdntEhupKs-8pC4My&index=1>
- <https://www.w3schools.com/django/>
- <https://dev.to/sm0ke/how-to-use-mysql-with-django-for-beginners-2ni0>