RAPPORT DU PROJET DU SITE WEB DU SUPER MARCHE MYN MARKET

GROUPE 3

NGATSE Aime Dauphin Achtar

DOKO Ryss l'Espoir

BABATILA Grace Saturnel

BITSINDOU Japhet Medard

BONABO MAMPASSI ELI

TCHITEMBO Jonathan

BATHA DJAKOU Josias

Formateur

Armel MOKONO

Introduction

Avec la digitalisation et l'évolution des habitudes d'achat, le commerce de détail doit s'adapter pour répondre aux attentes des consommateurs. La création d'un site web pour un supermarché représente une opportunité stratégique pour répondre à ces évolutions. Ce projet vise à développer une plateforme en ligne intuitive et fonctionnelle, facilitant les achats de nos clients, bien que le site soit actuellement en local.

Outils et Technologies Utilisés

Pour réaliser ce projet, plusieurs outils et technologies ont été utilisés :

- **HTML et CSS** : Pour structurer et styliser les pages web, offrant une interface utilisateur attrayante et facile à naviguer.
- **JavaScript** : Pour ajouter des fonctionnalités interactives et dynamiques aux pages web, améliorant ainsi l'expérience utilisateur.
- **Python et Django** : Pour le développement backend, permettant de gérer la logique serveur, les interactions avec la base de données et les fonctionnalités avancées de l'application.
- **VS Code** : Comme environnement de développement intégré (IDE), facilitant l'écriture, le débogage et la gestion du code.
- **WampServer** : Pour créer un environnement de serveur web local, indispensable pour tester et développer le site en local avant son déploiement en ligne.
- **MySQL** : Pour la gestion de la base de données, assurant un stockage fiable et efficace des informations produits et utilisateurs.
- **MySQL Connector** : Pour permettre la communication entre l'application Django et la base de données MySQL, assurant une interaction fluide et sécurisée.

Pages Développées

Dans le cadre de notre projet de création d'un site web pour notre supermarché, nous avons développé plusieurs pages HTML et CSS pour assurer une expérience utilisateur fluide et cohérente. Voici un aperçu des différentes pages de notre site web :

Pages CSS

1. **`stylee.css`** : Contient les règles générales de mise en page et de design pour l'ensemble du site, assurant une uniformité visuelle et une esthétique attrayante.
2. **`inscristyle.css`** : Spécifiquement dédiée à la page d'inscription, elle contient les styles nécessaires pour rendre le formulaire d'inscription clair et agréable à utiliser.
3. **`home.css`** : Appliquée à la page d'accueil, garantissant une présentation accueillante et engageante dès la première visite de l'utilisateur.
4. **`styl.css`** : Associée à la page "Mon Compte", elle améliore l'apparence et l'accueil de cette page.

Pages HTML

1. **`home.html`** : La page d'accueil du site. Elle présente un aperçu des produits, des offres spéciales et des nouveautés, accueillant ainsi les visiteurs avec des informations pertinentes et attractives.
2. **`accueil.html`** : Une autre version de la page d'accueil, potentiellement destinée à des utilisateurs spécifiques ou à une navigation différente selon les préférences du site.
3. **`mon_compte.html`** : Permet aux utilisateurs de gérer leurs informations personnelles, leurs commandes, et leurs préférences de compte.
4. **`connexion.html`** : Page de connexion où les utilisateurs peuvent entrer leurs identifiants pour accéder à leur compte personnel.
5. **`inscription.html`** : Page d'inscription permettant aux nouveaux utilisateurs de créer un compte sur le site.
6. **`panier.html`** : Page du panier où les utilisateurs peuvent voir les articles ajoutés, modifier les quantités, et procéder à l'achat.
7. **`catalogue.html`** : Affiche l'ensemble des produits disponibles dans le supermarché, classés par catégories, et permet aux utilisateurs de naviguer facilement entre les différentes offres.

Ces différentes pages et feuilles de style travaillent ensemble pour offrir une expérience utilisateur cohérente, intuitive et agréable. Chaque page a été conçue en tenant compte des besoins des utilisateurs, leur permettant de naviguer facilement sur le site, de trouver les informations dont ils ont besoin et de réaliser leurs achats en ligne en toute simplicité

Structure de notre site

Présentation de la partie python :

Outils utilisés :

1. **Python :**
   C’est un langage de programmation interprété, orienté objet et de haut niveau, connu pour sa syntaxe claire et lisible, qui facilite le développement rapide d'applications dans divers domaines comme le développement web, la science des données, et l'intelligence artificielle.

2. **mysql-connector-python :**
   C’est une bibliothèque officielle de Python développée par Oracle pour se connecter à des bases de données MySQL. Elle permet d'exécuter des requêtes SQL, de gérer des transactions, et d'interagir avec des bases de données MySQL directement depuis des scripts ou des applications Python.

3. **Un client SQL :** est un logiciel ou une interface qui permet aux utilisateurs de se connecter à une base de données SQL, d'exécuter des requêtes SQL, de gérer les données et de manipuler la structure de la base de données. Il offre souvent des fonctionnalités telles que l'édition de requêtes, la visualisation des résultats, la gestion des utilisateurs et des droits, ainsi que l'import/export de données. Des exemples de clients SQL incluent phpMyAdmin, MySQL Workbench, HeidiSQL, et DBeaver.

4. **Django :** est un framework web open-source pour le langage de programmation Python, conçu pour faciliter le développement rapide et pragmatique de sites web et d'applications web. Il suit le modèle architectural MVT (Modèle-Vue-Gabarit) et offre de nombreux outils intégrés pour gérer des tâches courantes, telles que l'authentification des utilisateurs, la gestion des sessions, les formulaires, et l'administration de site. Django est apprécié pour sa sécurité, sa scalabilité et sa capacité à encourager une conception de code propre et réutilisable.

Procédure de connections

Pour la connexion entre notre site web et la base de données nous avons utilisé comme backend ''python'' et mis en place un projet Django.

Note projet python st composé de deux application 'marquet/' et 'catalogue/' la structure de chaque application est constituée quasiment de la meme manière

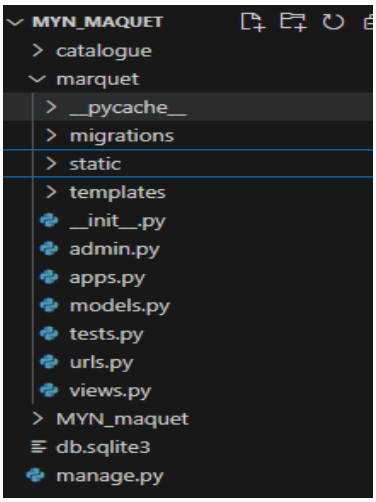
Voici un exemplaire de la structure(arbre) de notre projet web :

MYN_MARQUET/ (notre projet)

```
|
├── manage.py
├── MYN_MARQUET /
|       ├── __init__.py
|       ├── asgi.py
|       ├── settings.py
|       ├── urls.py
|       ├── wsgi.py
|
├── marquet/  (notre application)
|       ├── __init__.py
|       ├── admin.py
|       ├── apps.py
|       ├── models.py
|       ├── tests.py
|       ├── urls.py
|       ├── views.py
|       ├── migrations/
```

```
|       |       ├── __init__.py
|       |       └── ...
|       ├── static/
|       |       ├── marquet/
|       |       |       ├── inscristyle.css
|       |       |       ├── home.css
|       |       |       └── images/
|       |       └── ...
|       ├── templates/ (comtien les pages web décrites juste dans les premiers lignes)
|       |       |       └── ...
|       ├── forms.py
|       ├── signals.py
|       ├── templatetags/
|       ├── __init__.py
|       └── ...
```

Images :



A noté que la base de données ‘bd.sqlite3’ dans l’image est la base de données par défaut fournit par Django avec un SGBD SQLite, dont nous n’avons pas utiliser.

- Connexion avec la base de données MySQL :
  Il est nécessaire que Django puisse faire une migration dans bases de données où il va opérer pour implanter toutes les tables nécessaires dont il à besoins pour une meilleurs fonctionnalité :
  - Configuration du projets ‘MYN_MARQUET/settings.py’ :

```python
# Database
# https://docs.djangoproject.com/en/5.0/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'myn',
        'USER': 'root',
        'password': '',
        'HOST': 'localhost',
        'PORT': '3306',
    }
}
```

Après configuration, on a fait la migration par les commandes sur ligne de commande :

- Python manage.py makamigrations
- Python manage.py migrate

Les applications existantes ont aussi été ajouté dans la configuration pour être reconnus par python

```python
# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.
    'django.
    'django.contrib.staticfiles',
    'marquet',
    'catalogue',
]
```

(module) contrib

- **Les répertoire static aussi doivent être importé dans les configurations pour les configurations des fichiers .CSS ou des images qui vont être utiliser dans les pages web.**

```python
import os

# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/5.0/howto/static-files/

STATIC_URL = 'static/'

STATICFILES_DIRS = [
    os.path.join(BASE_DIR, "marquet/static"),
]
```

- **urls.py** : Définit les URL de niveau projet et les points d'entrée pour les applications.

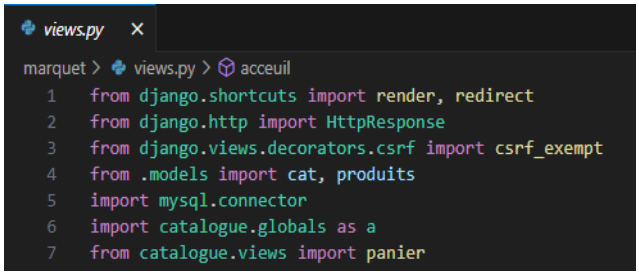
```python
from django.contrib import admin
from django.urls import path,include
from django.shortcuts import redirect

urlpatterns = [
    #path('admin/', admin.site.urls),
    path('', include("marquet.urls")),
    path('admin/', lambda request: redirect('http://localhost/phpmyadmin')),
    path('catalogue', include("catalogue.urls")),
]
```

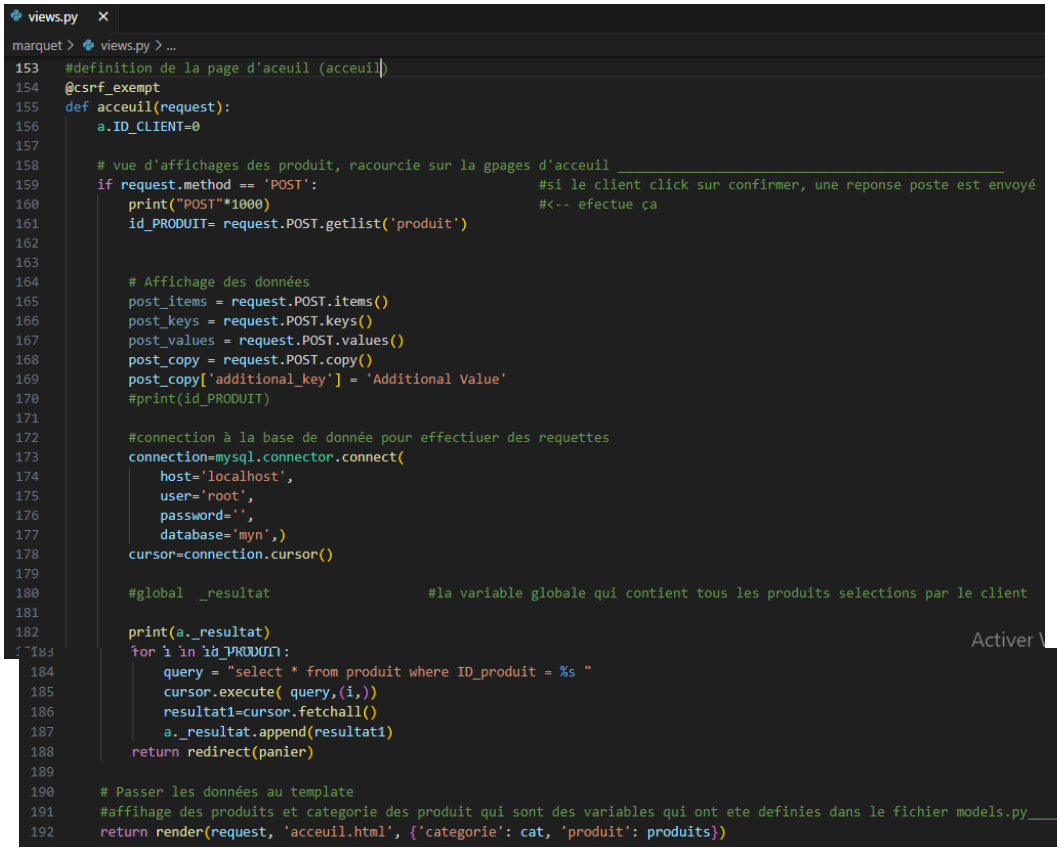
Conceptions des VUES ou MODELS et des URLS de chaque pages web :

- VUE :
  En Python, et plus spécifiquement dans le cadre d'un framework web comme Django, une **vue** est une fonction ou une classe qui reçoit une requête web (HTTP) et retourne une réponse web (HTTP). Les vues contiennent la logique d'application pour déterminer quel contenu afficher, quelles données manipuler, et comment répondre à la requête de l'utilisateur.

  Sur la tête de notre fichiers ‘marquet/views.py’ nous avons importé tout d’abord les modules nécessaires pour les fonctions définies.



1. VUE de la page d’accueil si l’utilisateur n’est pas connecté :



Extrait de la pages web, pour l’affichage des données (python) :

```
<li class="ryss"><a href="#"><img src="{% static 'marquet/images/liste.png' %}">nos categories</a>
    <!-- liste des categoriesdans longlet menu -->
    <form title="ajout_au_panier" method="post" action="#">
        <div class="menu">
            <ul class="liste2">
                {% for i in categorie %}
                    <li class="liste2_"><p>{{ i.0 }}</p>
                        <ul class="sous_liste2">
                            {% for j in produit %}
                                {% if j.6 == i.0 %}
                                    <li><p><input type="checkbox" value="{{ j.0 }}" name="produit">{{ j.2 }
                                        <div class="images">
                                            <img src="{% static j.10 %}" alt="">
                                        </div>
                                    </li>
                                {% endif %}
                            {% endfor %}
                        </ul>
                    </li>
                {% endfor %}
            </ul>
            <input class="button" type="submit"  value="CONFIRMER">
        </form>
```

```
marquet > templates > acceuil.html
1  <!DOCTYPE html>
2  <html>
3      <head>
4          {% comment %}
5              chargement des fichiers static sur notre pages
6          {% endcomment %}
7          {% load static %}
8
9          <meta sharset="utf-8">
10         <link rel="stylesheet" href="{% static 'marquet/home.css'
11     </head>
12
```

- URLS.py des pages
  Toutes les pages web, doit avoir un URL et une identité pour permettre les relations es entre eux.
  Voice une l’illustration de notre fichiers URLS.PY :
  De l’application python et de l’application ryss

views.py | connexion.html | acceuil.html | urls.py

marquet > urls.py > ...

```python
from django.urls import path
from .views import register, login, mon_compte, acceuil, home

urlpatterns = [
    # path("", views.home, name="home"),
    path("index/", register, name="registere"),
    path("connect/", login, name="login"),
    path("home/mon_compte/", mon_compte, name="mon_compte"),
    path("acceuil/", acceuil, name="acceuil"),
    path("home/", home, name="home"),
]
```

Comment Exécuter Notre Site ?

Pour permettre à quelqu'un de visualiser notre site web sur son propre ordinateur, même sans les outils de développement, suivez ces étapes :

1. **Installation de Python et de Django**

   a. **Téléchargement et Installation de Python**

1. Rendez-vous sur le site officiel de Python.
2. Téléchargez la version Python 3.12 (version utilisée) ou la version compatible avec votre système d'exploitation.
3. Lancez le fichier d'installation téléchargé. Lors de l'installation, cochez l'option "Add Python to PATH" avant de cliquer sur "Install Now".

b. Installation de Django

4. Ouvrez une invite de commande (Command Prompt sur Windows, Terminal sur macOS/Linux).
5. Tapez la commande suivante pour installer Django via pip, le gestionnaire de paquets Python :
   `pip install django`

2. **Installation de Visual Studio Code (VS Code)**
   1. Accédez au site de Visual Studio Code.
   2. Téléchargez l'installateur correspondant à votre système d'exploitation.
   3. Exécutez le fichier d'installation et suivez les instructions à l'écran pour installer Visual Studio Code.
3. **Installation de WampServer**
   1. Rendez-vous sur le site de WampServer.
   2. Téléchargez la version compatible avec votre système d'exploitation (Windows).
   3. Exécutez le fichier d'installation et suivez les instructions pour installer WampServer sur votre ordinateur.
4. **Configuration du Serveur Local**

a. Déplacement des Fichiers du Projet

1. Après avoir installé WampServer, déplacez le dossier contenant les fichiers du projet dans le répertoire `www` de WampServer, généralement situé dans `C:\wamp64\www\`.

b. Démarrage de WampServer

2. Lancez WampServer depuis le menu Démarrer.
3. Assurez-vous que l'icône WampServer dans la barre des tâches est verte, indiquant que le serveur est opérationnel.

5. **Configuration de la Base de Données (si nécessaire)**

a. Ouverture de phpMyAdmin

1. Dans votre navigateur web, accédez à http://localhost/phpmyadmin/.
2. Créez une nouvelle base de données et importez le fichier SQL du projet dans la base de données créée.

b. **Configuration de la Base de Données dans Django**

3. Ouvrez Visual Studio Code.
4. Accédez au répertoire du projet et ouvrez le fichier `settings.py` (localisé dans le répertoire du projet Django).

6. **Lancement du Serveur de Développement Django**

1. Dans Visual Studio Code, ouvrez le terminal intégré en sélectionnant `Terminal > New Terminal` ou faire la combinaison (ctrl+shift+Tilde)
2. Accédez au répertoire de votre projet Django en utilisant la ligne de commande.
3. Tapez la commande suivante pour appliquer les migrations de la base de données :

   ```
   python manage.py migrate
   ```

4. Lancez le serveur de développement Django avec la commande suivante :

   ```
   python manage.py runserver
   ```

5. Accédez au site via http://localhost:8000/accueil/ dans votre navigateur pour visualiser le site web en fonctionnement.

En suivant ces instructions, l'utilisateur pourra installer les outils nécessaires, configurer son environnement local et visualiser le site web sur son propre ordinateur.