Université de Limoges

Sujet de st a ge : Élaboration d'un site internet sous python pour la société ANDMTM consulting spécialisée dans l 'informatique. Après mise en place du site, il faut s'assurer que les normes de sécurité ont été respectées pour éviter des attaques provenant des sour ces mal veillantes. Et nous procéder ons à des tests afin de vérifier la robustesse des procédures de sécurité mise en place.

Par

Mohamed Lamine NDIONGUE

Dirigée par le Professeur Philippe GABORIT

Contents

Table des matières

Chapitre 1

Développement du site

1.1 Définition de Django et Installation de ses outils

1.1.1 Présentation de l'entreprise ANDMTM Consulting

L'ANDMTM Consulting est une entreprise spécialisée dans le domaine de l'informatique, donc elle vise à organiser des conseils, des propositions de solutions et orientation dans le choix d'outils informatique aux clients. Entre autre, ANDMTM Consulting est une entreprise qui a comme domaine d'activité est : conseil en systèmes et logiciels informatiques. Elle a été mise en place en 2021 par Monsieur Abou NDIONGUE (le fondateur de l'entreprise).

L'objectif de ce rapport de stage est de créer un site internet sécurisé pour cette même entreprise dont les différentes parties de la création du site sont présentées ci-après avec le langage python.

1.1.2 Présentation de Django

Django est un framework open source (gratuit) en python conçu pour le développement

le framework Django a été developpé en 2003 dans le journal local de lawrence (Etat du Kansas, aux Etats-Un is). C'est un logiciel très pratique vu la simplicité et la facilité de faire du développement web. Sa parution a été faite en 2005 sous licence BSD (Berkeley Software Distribution License) [1].

C'est un logiciel que j'ai utilisé pour concevoir avec des fichiers html et des fichiers python.

1.1.3 Installa tion des outils de Djan go

L'installation du framework django nécessite bien-sûr une connexion à internet qui permet de télécharger tous les packages nécessaires pour pouvoir effectuer le développement web du site.

Le site principale de Django est https://www. djangoproject.com/ et son installation se distingue légèrement selon le système d'exploitation. Pour notre cas, nous utilisation la procédure d'installation sous Windo ws. Elle se fait comme suit :

- **C : Users nom_utilisateur>p ython.exe -m pip install django**
- Vérification de l'installation par la commande **C : Users nom_utilisateur>pip list** qui permet d'afiicher la liste des packages et ensuite voir si django y figure.

Une fois terminer l'installation django et la vérification, on s'intéresse à la création du projet en le démarran t par la commande **C : Users nom_utilisateur>p ython.exe -m django startpro ject nom_du_pro jet**. En effet, cette commande met en création une nouvelle structure de répertoires pour le projet django et incluant ainsi des fichiers de base. Après cette exécution de commande, nous avons le répertoire **nom_du_pro jet** créé, un sous-répertoire **nom_du_pro jet** et un fichier **manage.p y** à la racine du répertoire du projet qui est utile pour la gestion du projet. Ce dernier (le fichier manage.p y) permet d'exécuter un certain nombre de commandes pour créer des applications, faire des migrations et aussi, démarrer des serveurs pour obtenir des résultats dans un navigateur.

Le dossier nom_du_pro jet contient un certain nombre de fichiers tels que :

1. __init__.p y : ce fichier, trouvé dans un répertoire, indique que ce répertoire doit être considéré comme un package permettan t ainsi l'organisation du code en modules et en sous-modules ;
2. admin.p y : permet la configuration de l'interface d'administration et définir ce qu'on souhaite afiicher et modifier dans l'administation de l'application ;

Agence > admin.py > AdminCategorie

```python
from django.contrib import admin
from .models import Category, Product, Identifiant


# Register your models here.
class AdminCategorie(admin.ModelAdmin):
    list_display = ['name', 'date_added']

class AdminProduct(admin.ModelAdmin):
    list_display = ['title', 'price', 'category', 'date_added']

class AdminIdentifiant(admin.ModelAdmin):
    list_display = ['username', 'first_name', 'second_name', 'mail', 'number', 'password', 'date_added']


admin.site.register(Category, AdminCategorie)
admin.site.register(Product, AdminProduct)
admin.site.register(Identifiant, AdminIdentifiant)
```

3. apps.py : est un fichier contant une classe de configuration de l'application qui hérite de l'**App config** . Cette classe permet de définir des paramètres spécifiques à l'application, comme son, son label et d'autres options spécifiques. Donc, il est automatiquemen t généré lorsqu'une application est créée avec la commande **python manage.p y** ;

```
Agence > apps.py > ...
1  from django.apps import AppConfig
2
3
4  class AgenceConfig(AppConfig):
5      default_auto_field = 'django.db.models.BigAutoField'
6      name = 'Agence'
7
```

4. models.py : est un fichier définissant des modèles de l'application. Un modèle est une classe Python représentant une table dans la base de données et chaque attribut de la classe correspond à une colonne de la table ;

```
Agence > models.py > Category > __str__
1  from django.db import models
2  from django.db.models.fields.related import ForeignKey
3
4  # Create your models here.
5  class Category(models.Model):
6      name = models.CharField(max_length=200)
7      date_added = models.DateTimeField(auto_now_add=True)
8
9      class Meta:
10         ordering = ['-date_added']
11
12     def __str__(self):
13         return self.name
14
15 class Product(models.Model):
16     title = models.CharField(max_length=200)
17     price = models.FloatField()
18     description = models.TextField()
19     category = ForeignKey(Category, related_name='categorie', on_delete=models.CASCADE)
20     image = models.CharField(max_length=5000)
21     date_added = models.DateTimeField(auto_now_add=True)
22
23     class Meta:
24         ordering = ['-date_added']
25
26     def __str__(self):
27         return self.title
28
29 class Identifiant(models.Model):
30     username = models.CharField(max_length=200)
31     first_name = models.CharField(max_length=200)
32     second_name = models.CharField(max_length=200)
33     mail = models.CharField(max_length=200)
34     number = models.CharField(max_length=200)
35     password = models.CharField(max_length=200)
36     date_added = models.DateTimeField(auto_now_add=True)
37
38     class Meta:
39         ordering = ['-second_name']
40
41     def __str__(self):
42         return self.second_name
43
44 # Optional: You can add a model to track PayPal transactions if needed
45 class PayPalTransaction(models.Model):
46     product = models.ForeignKey(Product, on_delete=models.CASCADE)
47     transaction_id = models.CharField(max_length=200)
48     amount = models.FloatField()
49     currency = models.CharField(max_length=10)
50     status = models.CharField(max_length=50)
51     date_added = models.DateTimeField(auto_now_add=True)
52
53     class Meta:
54         ordering = ['-date_added']
55
56     def __str__(self):
57         return f"Transaction {self.transaction id} for {self.product.title}"
```

5. tests.p y : est utilisé pour imprimer des tests unita ires pour l'application conçue ; et ces tests vérifient la fonctionnalité des différentes parties de l'application ;

```
Agence > tests.py
1  from django.test import TestCase
2
3  # Create your tests here.
4
```

6. asgi.py : pour la gestion des applications asynchrones dont les fonctions principales sont :
   - interface ASGI (Asynchronous Server Gateway Interface) pour la gestion des requêtes asynchornes ;
   - configuration du serveur permettant la définition des paramètres nécessaires pour que le serveur ASGI parvienne à service à l'application django. Ainsi, nous avons l'inclusion des paramètres du projet et la mise en page de l'application Django pour la réception des requêtes ;
   - Support de WebSockets avec la gestion des connexions WebSocket grâce à l'interface ASGI.

```
ANDMTM_Consulting > asgi.py > ...
1  import os
2
3  from django.core.asgi import get_asgi_application
4
5  os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'ANDMTM_Consulting.settings')
6
7  application = get_asgi_application()
8
```

7. settings.py : ce fichier permet de faire une configuration globale de l'application définie et dont les différentes fonctions sont : la configuration des paramètres, la gestion des environnements, la personnalisation et la sécurité ;

```
ANDMTM_Consulting > settings.py > ...
1  from pathlib import Path
2  import os
3
4  # Build paths inside the project like this: BASE_DIR / 'subdir'.
5  BASE_DIR = Path(__file__).resolve().parent.parent
6  TEMPLATES_DIRS = os.path.join(BASE_DIR, 'templates')
7
8  # Quick-start development settings - unsuitable for production
9  # See https://docs.djangoproject.com/en/5.0/howto/deployment/checklist/
10
11 # SECURITY WARNING: keep the secret key used in production secret!
12 SECRET_KEY = 'django-insecure-w-#^l1wuz*_40208j6x4no-c$0xi&=up$&ng&)yw2d((z$73qk'
13
14 # SECURITY WARNING: don't run with debug turned on in production!
15 DEBUG = True
16
17 ALLOWED_HOSTS = ['*']
18
19 # Application definition
20
21 INSTALLED_APPS = [
22     'django.contrib.admin',
23     'django.contrib.auth',
24     'django.contrib.contenttypes',
25     'django.contrib.sessions',
26     'django.contrib.messages',
27     'django.contrib.staticfiles',
28     'Agence',
29 ]
30
31 MIDDLEWARE = [
32     'django.middleware.security.SecurityMiddleware',
33     'django.contrib.sessions.middleware.SessionMiddleware',
34     'django.middleware.common.CommonMiddleware',
35     'django.middleware.csrf.CsrfViewMiddleware',
36     'django.contrib.auth.middleware.AuthenticationMiddleware',
37     'django.contrib.messages.middleware.MessageMiddleware',
38     'django.middleware.clickjacking.XFrameOptionsMiddleware',
39 ]
40
41 ROOT_URLCONF = 'ANDMTM_Consulting.urls'
```

```
42
43  TEMPLATES = [
44      {
45          'BACKEND': 'django.template.backends.django.DjangoTemplates',
46          'DIRS': [],
47          'APP_DIRS': True,
48          'OPTIONS': {
49              'context_processors': [
50                  'django.template.context_processors.debug',
51                  'django.template.context_processors.request',
52                  'django.contrib.auth.context_processors.auth',
53                  'django.contrib.messages.context_processors.messages',
54              ],
55          },
56      },
57  ]
58
59  WSGI_APPLICATION = 'ANDMTM_Consulting.wsgi.application'
60
61  # Database
62  # https://docs.djangoproject.com/en/5.0/ref/settings/#databases
63
64  DATABASES = {
65      'default': {
66          'ENGINE': 'django.db.backends.sqlite3',
67          'NAME': BASE_DIR / 'db.sqlite3',
68      }
69  }
70
71  # Password validation
72  # https://docs.djangoproject.com/en/5.0/ref/settings/#auth-password-validators
73
74  AUTH_PASSWORD_VALIDATORS = [
75      {
76          'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
77      },
78      {
79          'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
80      },
81      {
82          'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
83      },
84      {
85          'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
86      },
87  ]
88
89  # Internationalization
90  # https://docs.djangoproject.com/en/5.0/topics/i18n/
91
92  LANGUAGE_CODE = 'en-us'
93
94  TIME_ZONE = 'UTC'
95
96  USE_I18N = True
97
98  USE_TZ = True
99
100 # Static files (CSS, JavaScript, Images)
101 # https://docs.djangoproject.com/en/5.0/howto/static-files/
102
103 STATIC_URL = 'static/'
104
105 # Default primary key field type
106 # https://docs.djangoproject.com/en/5.0/ref/settings/#default-auto-field
107
108 DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
109
110 # PayPal settings
111 PAYPAL_CLIENT_ID = 'votre_client_id'
112 PAYPAL_CLIENT_SECRET = 'votre_client_secret'
113 PAYPAL_MODE = 'sandbox'  # ou 'live' pour la production
114
```

8. urls.py : est utilisé dans un projet Django pour définir les routes URL de l'application définie. Il associe les URL aux vues correspondantes, permettant ainsi de gérer les requêtes HTTP ;

Agence > urls.py > ...

```python
from django.urls import path, include
from django import views
from .views import *


urlpatterns = [
    path('',home, name='home'),
    path('login/',login, name='login'),
    path('register/',register, name='register'),
    path('gallery/',gallery, name='gallery'),
    path('<int:myid>',detail, name='detail'),
    path('about/',about, name='about'),
    path('contact/',contact, name='contact'),
    path('gestionProjet/',gestionProjet, name='gestionProjet'),
    path('andmtm/',andmtm, name='andmtm'),
]

```

9. wsgi.py : permet le déploi de l'application créée sur un serveur web compatible avec l'interface WSGI (Web Server Gateway Interface). Il sert de point d'entrée pour le serveur web afin de servir votre application Django ;

ANDMTM_Consulting > wsgi.py > ...

```python
import os

from django.core.wsgi import get_wsgi_application

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'ANDMTM_Consulting.settings')

application = get_wsgi_application()

```

10. manage.py : est un script utilitaire dans un projet Django qui permet d'interagir avec le projet via la ligne de commande. Il fournit diverses commandes pour gérer l'application, telles que démarrer le serveur de développement, exécuter des tests, créer des migrations, etc ;

manage.py > ...

```python
#!/usr/bin/env python
"""Django's command-line utility for administrative tasks."""
import os
import sys


def main():
    """Run administrative tasks."""
    os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'ANDMTM_Consulting.settings')
    try:
        from django.core.management import execute_from_command_line
    except ImportError as exc:
        raise ImportError(
            "Couldn't import Django. Are you sure it's installed and "
            "available on your PYTHONPATH environment variable? Did you "
            "forget to activate a virtual environment?"
        ) from exc
    execute_from_command_line(sys.argv)


if __name__ == '__main__':
    main()

```

11. views.py : définit les vues de l'application définie dans le projet Django. Une vue est une fonction ou une classe qui prend une requête web et retourne une réponse web. Les vues contiennent la logique nécessaire pour traiter les requêtes et renvoyer les réponses appropriées.

```python
Agence > views.py > andmtm
from django.shortcuts import render <!-- render est une fonction qui permet de renvoyer une page html-->
<!-- cette importation depuis django.shortcuts permet de renvoyer une page html-->
from .models import Category, Product, Identifiant
from django.core.paginator import Paginator
# Create your views here.

def home(request):
    return render(request, 'home.html')

def login(request):
    return render(request, 'login.html')

def register(request):
    return render(request, 'register.html')

def gallery(request):
    product_object = Product.objects.all()
    item_name = request.GET.get('item-name')
    if item_name != '' and item_name is not None:
        product_object = Product.objects.filter(title__icontains=item_name)
    paginator = Paginator(product_object, 4)
    page = request.GET.get('page')
    product_object = paginator.get_page(page)
    return render(request, 'gallery.html', {'product_object': product_object})

def detail(request, myid):
    product_object = Product.objects.get(id=myid)
    return render(request, 'detail.html', {'product': product_object})

def about(request):
    return render(request, 'about.html')

def contact(request):
    return render(request, 'contact.html')

def gestionProjet(request):
    return render(request, 'gestionProjet.html')

def andmtm(request):
    return render(request, 'andmtm.html')
```

1.2 Implémen tation du site

Dans cette partie implémentation, nous expliquons le choix de la définition des différents fichiers établis pour la bonne conception du site internet. Tout d'abord, nous parlerons du répertoire **templates** qui est un dossier dans lequel sont hébergés les différents fichiers.

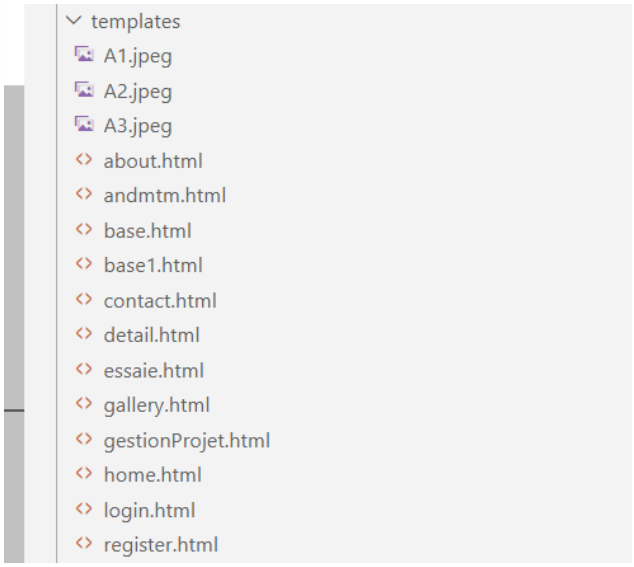

Figure 1.1 – Templates pour les fichiers HTML

Chapitre 2

Sécurité du site

Conclusion

Bibliographie

Bibliographie et webographie

Bibliographie

Webographie