Dossier Professionnel

Concepteur Développeur d’Application

Session : 2022-2023

Candidat : Saïd Chamchi

Tuteur Pédagogique : Mathieu Capon

Tuteur Technique : Jean-François Beachler



**Web Shop Application**

During my training as a web developer, I had the opportunity to do a work placement with InfoJef. This Mulhouse-based company specialises in IT hardware and servicing.

Already very well established locally and keen to offer its services to a wider audience.

I suggested that they create an a web shop.

The main purpose of the application will be to present the company and the various products and services it offers.

The functionalities were to have a product catalogue visible on the application . But also to be able to create, modify and add new products via an administration panel.

Knowing the objectives of my internship, and with the agreement of my tutor Mr Beachler. We decided to use the Django .

This tools are based on the Python language.

The main advantages of this Framework are its ORM, its user management and its template engine.

The application will consist of two main parts: the front end, which will be based on HTML/CSS and Javascritp, and the JinJa template engine.

The second part will be the back end, which will be developed with Django (controllers, routing, urls, user management and authentication).

The database will be SQLlite.

# Introduction

Depuis le début, des années 2000 avec la démocratisation d’Internet a la naissance du Web2.0 avec les applications complexe et les différents outils (Framework Front-end, Framework Back-end ).

Internet est aujourd’hui indispensable et très commun à beaucoup d’usage. Aussi bien pour un usage lambda que pour les Professionnels.

Le Web est devenu un vecteur de communication indispensable pour beaucoup de personnes. Aussi des administrations publiques ou les Professionnels.

Mais aujourd’hui internet est bien plus qu’un simple outil de communication. Il est devenu un outil indispensable pour différentes entités économiques.

En effet internet est bien plus qu’un simple outil de communication, il permet aussi de commercer avec des millions de personnes a travers le monde.

Cet outil formidable a pu engendrer des Acteurs bien connue et surtout indispensable a notre quotidien.

Au travers de mon stage, j’ai pu étudier cet usage principalement.

Effectivement, durant mon stage que j’ai effectué dans l’entreprise « InfoJef ». Je tiens à remercier ici le responsable et le gérant Monsieur Jean-Frederic Baechler pour son soutien et sa bienveillance.

Durant cette période, mon tuteur de stage m'a confié la mission de développer une application de commerce en ligne « e-commerce ».

L’objectif de cette application était simple mais aussi passionnant.  
Permettre de mettre en ligne le catalogue des différents produits que l’entreprise propose.  
Ainsi que pouvoir créer, modifier, mettre a jour les différents articles.  
(Un CRUD, Create, Read, Update, Delete).  
  
Mais aussi de pouvoir réalisé des transactions électroniques « paiement électronique » via des api tel que PayPal, mais aussi l’API de la banque de l’entreprise.  
  
Suite à nos différents échanges avec Mr Baechler, nous avons pu développer un cahier des charges de ce projet ainsi qu’une charte graphique.  
  
Avec ces différents éléments, j’ai pu réaliser dans un premier temps réalisé une maque de l’application avec Figma .  
  
Après la validation de mon tuteur.  
J’ai commencé à faire les différents diagrammes UML pour structure notre projet. (Use Cases, Diagramme de Classes)  
  
Vous pouvez trouver tous ces éléments en Annexe. (Maquette et les différentes diagrammes)

# Création du Projet

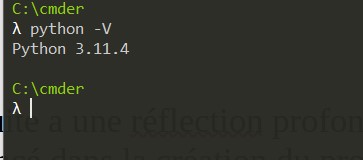
Suite à une réflexion, et la réalisation des digrammes, j’ai pu me lancer dans la création du projet.

Avec les différents diagrammes et suite à une longue réflexion, j’ai choisi d’utiliser le Framework Python Django / Django REST Framework.

Mon choix, c’est porté sur ce Framework pour multiples raisons.

Django et un formidable outil, il dispose d’un excellent ORM (Mapping objet-relationnel), une suite impressionnante de modules disponible grâce à l’outil PIP (pip est le gestionnaire de paquets). Qui me permettra de développé facilement ce projet.

Avant de pouvoir installer Django ou même lancé la création de notre application, je vérifie que Python est bien installer sur mon poste avec la commande « Python –V ».



Une fois que j’ai vérifié que Python est bien installer sur mon poste, je vais pouvoir créer mon environnement virtuel.

C’est quoi un environnement virtuel est quel est son utilité ?

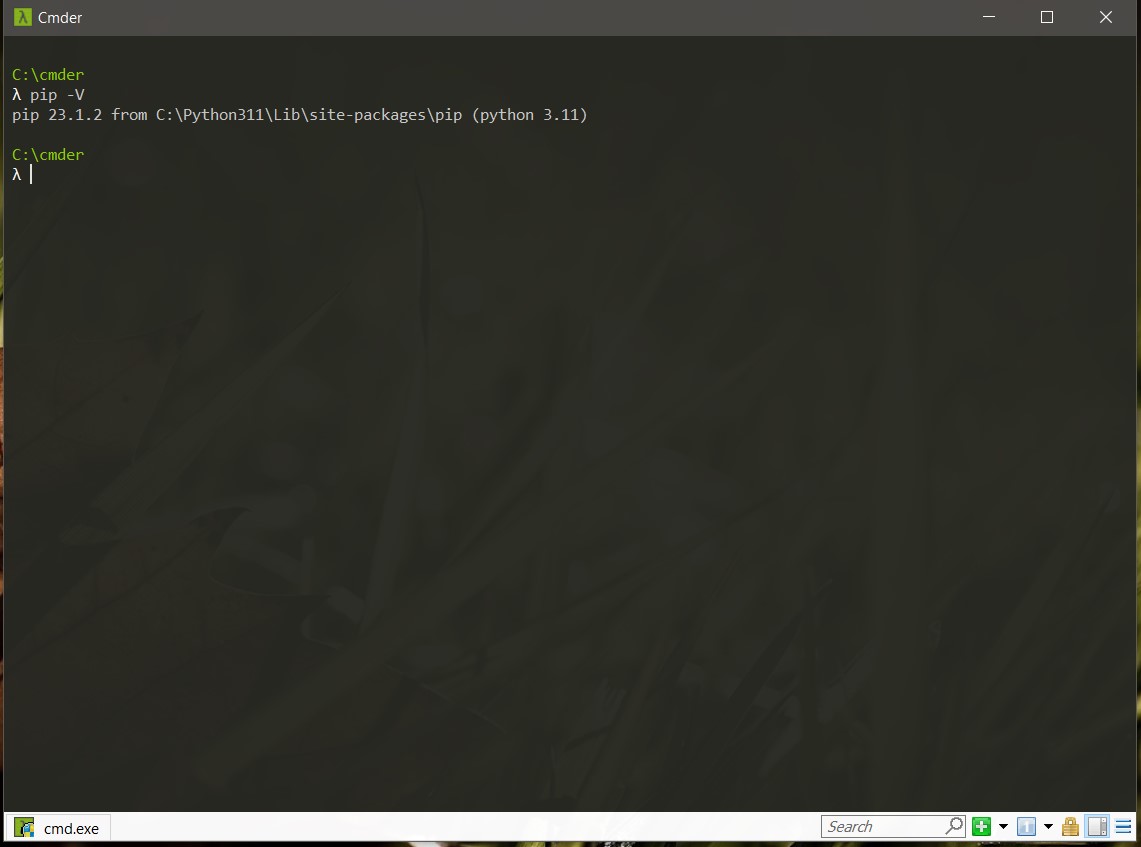
1 - Un environnement virtuel permet de créer un environnement complètement isolé du système

Ça permet aussi d’isoler toutes les dépendances du projet dans l’environnement virtuel propre à l’application

Cela évite de polluer son système d’exploitation. Ainsi qu’éviter les différents soucis que l’on peut avoir avec des versions de modules non-compatibles entre eux.

Après ces différentes vérifications, je décide de me lancer.

Dans un premier temps, je vérifie que la commande pip fonctionne bien.



Comme vous en doutez certainement PIP est le gestionnaire de paquets de Python, comme npm pour JavaScript ou composer pour PHP.

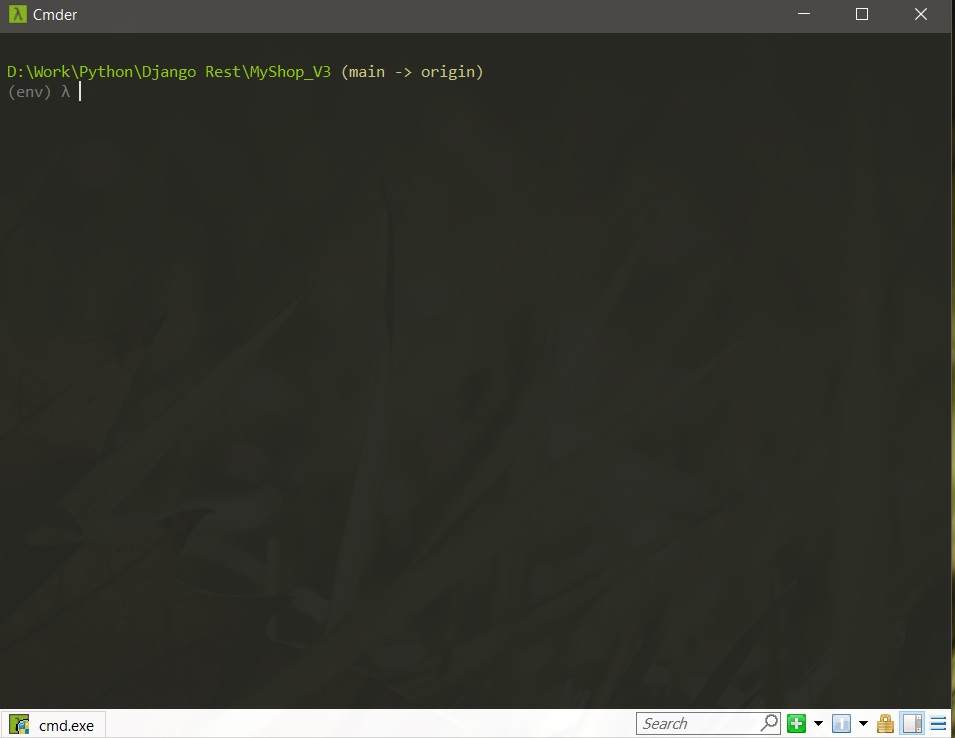
Après ces différentes versifications, je créais mon environnement virtuel avec la commande « python -m venv env ».

Cette commande va me créer un dossier « env », ou sera isolé toutes les dépendances du projet.

Maintenant, que l’environnement est installé, je vais l’activer et installer les différents paquets indispensables au Projet.

- Pour activer l’environnement virtuel, j’utilise la commande.

« env/script/activate ».



- Comme on peut le constater dans l’image, suite à la commande « env/script/activate », l’environnement est bien actif. « (env) »

-Maintenant il est possible d’installer les différentes dépendances nécessaires au projet.

Dans un premier temps, je commence par installer « Pillow ».

- « Pillow » est une dépendance indispensable pour gère les images dans l’application. Ce module est nécessaire dans la majorité des applications Django.

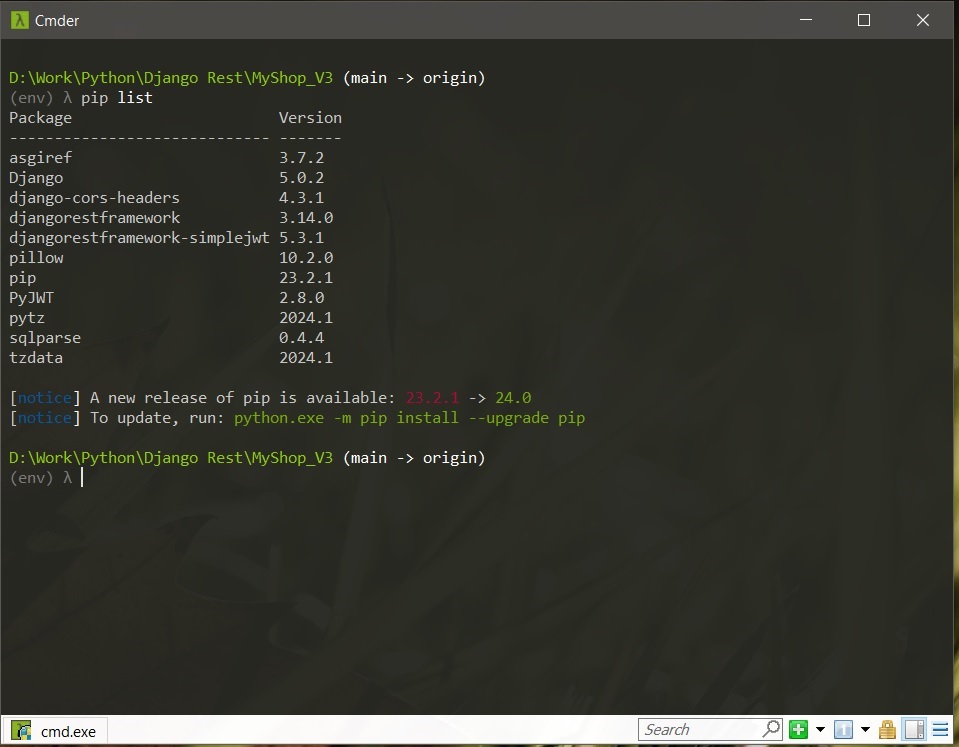
- Apres l’installation de « Pillow » j’installe Django et Django Rest Framework. Il est important de remarque que Django et Django Rest Framework sont de dépendances différentes. Pour ce faire je vais utiliser la commande «pip install Django » et « pip install djangorestframework ». Dans un second temps j’installe aussi simple JWT, avec la commande « pip install djangorestframework-simplejwt ». Une dépendance qui nous permettra de génère facilement un Token JWT (Json Web Token) pour sécuriser notre application.

- Afin de gère les requetés externes, je dois installer django-cors-headers.

Pour vérifier que ces dernières dépendances sont bien installées, je vais utiliser la commande « pip list » afin de m’assurer que tous les modules, c'est bien installer.

Comme on peut le voir dans l’image ci-dessous. On voit que tous les modules sont bien installés.

Après avoir vérifié que les dépendances sont installées, je vais pouvoir faire un pip freeze > requiement.txt.



Cette commande me permet d’inscrire toutes les dépendances du projet dans un fichier texte « requiement.txt » automatiquement. Mais aussi me permet d’installer toutes les dépendances avec une seul commande « pip install requiement ». Un peu comme un npm install.

Maintenant, que toutes les vérifications sont faites.

Je peux lancer l’installation du projet en utilisant la commande « django-admin startproject MyShop .».

Une fois la commande lancée cela m’a généré la structure du projet.

Dans le jargon des développeurs Django « un projet » englobe l’application et contient toutes les dépendances au bon fonctionne de notre future application.

A noter que Django peut gérer une multitude d’applications dans « un Seul Projet »

Une fois que le projet et créer nous pouvons nous atteler à la création de l’application, en utilisant la commande « python manage.py startapp "Nom de l'application" ».

Dans notre cas d'application ce nommera « shop ».

Structure du Projet

Comme on peut le constater dans la capture d’écran ci-dessous, on peut voir la structure du projet.

Ici, le nom du projet est MyShop « L’ensemble de l’application » et l’application s’ appelle « shop ».

Le projet MyShop est composé de plusieurs fichiers. Les deux fichiers les plus importants ici sont « setting », « views » et « url ».

« Setting » et le fichier de configuration du projet. Il contient toutes les informations relatives à l’application, comme le listing de l’ensemble des applications du « projet » mais aussi les « Middlewars», la configuration de la base de données et aussi le « Password Hasher », le Language code ainsi que la gestion des origines des requêtes autorisées et etc.

Le fichier « views.py » est un des fichiers les plus important, car il contient tous les contrôleurs de l’application.

Le fichier « url.py » contient comme son nom l’indique l’ensemble des urls du projet.

Maintenant, regardons de plus près notre application « shop ». Comme dit précédemment ce répertoire est le cœur de notre projet.

C’est bien ici que nous allons coder notre application.

Le répertoire est lui-même composer de plusieurs répertoires et de fichiers.

Il compte deux répertoires, qui sont « migrations » qui contient toutes les migrations des entités créer dans la base de données. La migration est un peu comme le plan de notre structure de notre base de données.

Nous trouvons un répertoire « tests » qui contient les tests de l’application. (Nous développerons plus tard cette partie).

Au sein de notre répertoire shop, nous trouvons aussi des fichiers de type Python :

- Le premier est « \_init\_.py », ce fichier sers principalement a indiqué la présence de package ou de fichier python dans le répertoire.

- « admin.py » :

Ce fichier est un peu particulier, il est utilisé pour importer les différentes entités du projet dans la page d’administration du Projet.

En fait Django contient en sont seins un panneau d’administration.

Que vous pouvez configurer comme vous le souhaitez via ce fichier.

- « Model » :

Ce fichier est vraiment très important, il contient toutes les entités de notre application. (Produits, Catégories et etc).

- « urls.py » :

Ce document, contient toutes les urls de l’application. Ce fichier est très utile aussi et on va souvent l’utiliser.

- « view.py » :

Le fichier view.py contient tous les contrôleurs et la logique de l’application.

- « manage.py » :

Le fichier manage.py est le point d’entrée de l’application, il contient la méthode main(). Il est indispensable pour lancer l’application !!

Avec la commande « python manage.py runserver ».

- « app.py » :

Le fichier app.py est un fichier de configuration pour une application Django. En résumé, ce fichier de configuration définit les paramètres de l'application shop pour Django, notamment le type de champ utilisé pour les ID automatiques et le nom de l'application.

- « serializers.py » :

Le fichier serializers.py est un fichier de code Python qui est utilisé pour définir les serialisateurs dans une application Django.

Un serialisateur est un objet qui permet de convertir des objets Python en données JSON ou XML, et vice versa. Dans Django, les serialisateurs sont utilisés pour transmettre des données entre l'application et les clients (Dans notre cas, notre application React JS et l’application Mobile React Native).

- « permission.py » :

Dans une application Django REST, le fichier permissions.py est utilisé pour définir les permissions qui sont nécessaires pour accéder aux ressources protégées par des autorisations.

Après avoir vu les Principaux Fichiers de notre application, nous allons commencer à construire notre Futur Application.

# Conception du Model

Après une certaine réflexion sur note future application de Commerce Électronique.

J’ai réalisé un diagramme de Cas d’utilisation « Uses Cases ». Afin de définir les principales fonctionnalités du projet.

Comme le diagramme le montre les principales fonctionnalités de l’application sont :

Pour l’utilisateur :

* L’authentification
* Listing des produits
* La vue détaillée du Produit
* La fonction ajoutée au panier
* La vue détaillée du Panier

Pour L’administrateur :

- L’authentification

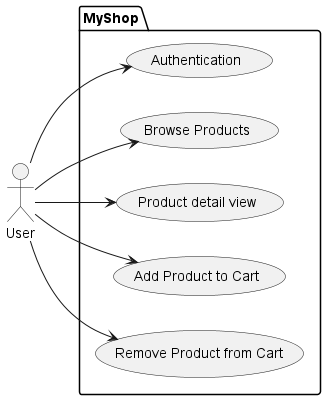
- Connexion

- Interface de gestion

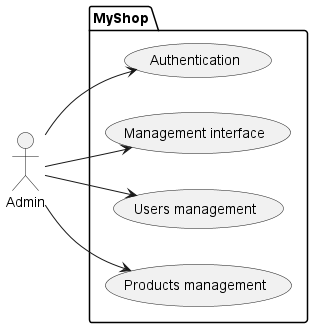
- Gestion des Utilisateurs

- Gestion des Produits

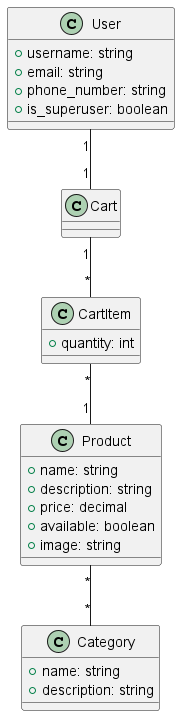
Diagramme « Use Case Utilisateur » :



**Diagramme Use Cases « Administrateur » :**



Une fois que nous avons identifiées les principales fonctionnalités de l’application. J’ai pu créer le diagramme de Classe.



Comme on peut le constater dans le diagramme. Nos Entités principales sont la Classe Produits, Catégorie, Panier, Produit du panier.

* User :

Cette classe représente un utilisateur dans le système. Elle a les attributs suivants :

- username : une chaîne de caractères représentant le nom d'utilisateur.

-email : une chaîne de caractères représentant l'adresse e-mail de l'utilisateur.

- phone\_number : une chaîne de caractères représentant le numéro de téléphone de l'utilisateur.

- is\_superuser : un booléen indiquant si l'utilisateur est un superutilisateur ou non.

Product :

Cette classe représente un produit dans le système. Elle a les attributs suivants :

- name : une chaîne de caractères représentant le nom du produit.

- description : une chaîne de caractères décrivant le produit.

- price : un nombre décimal représentant le prix du produit.

- available : un booléen indiquant si le produit est disponible ou non.

- image : une chaîne de caractères représentant l'URL de l'image du produit.

On peut remarquer que les entités sont relier par des relations.