

Institut d'Enseignement de promotion sociale de la Communauté française  
d'Uccle.

## **GESTION D'UN ECOMMERCE**

Travail de fin d'études en vue de l'obtention du bachelier informatique  
De gestion

Realisé par Ngoyi-A-Yambu Caleb



Année 2019-2020

## **Remerciements.**

L'aboutissement de ce projet n'aura certainement pas été possible sans le soutien, la force et le courage que Dieu m'a accordé le long de mon périlleux cursus scolaire. Alors je le remercie.

Je remercie également ma famille. Elle a été là lorsque j'en ai eu le plus besoin. Son étreinte s'est révélée apaisante, lorsque la tension liée à mes projets non aboutis, touchait son paroxysme. Un spécial merci à mes parents. J'espère de tout cœur, vous rendre fiers.

Je remercie mes professeurs pour l'aide, les conseils qu'ils m'ont fourni durant l'élaboration du TFE.

Je remercie toutes personnes ayant participé de près ou de loin à l'aboutissement de ce projet.

En dernier lieu, je me remercie moi-même, pour ne pas m'être laisser noyer.

## Table des matières

1. Dossier d'analyse.....	4
1.1 Présentation du sujet .....	4
1.2 Cahier de charge.....	4
1.2.1 Déploiement.....	5
1.2.2 Rôles .....	5
1.2.3 Diagramme de cas d'utilisation. ....	6
1.3 MCD .....	7
1.3.1 Dictionnaires de données du MCD.....	8
1.4 MLD .....	16
1.5 Tables et droits d'accès. ....	17
1.6 Analyse orientée objet. ....	19
1.6.1 Diagramme de package correspondant. ....	20
2. La base de données physique.....	21
3. Le dossier technique.....	30
3.1 Différents outils de programmation adoptés.....	30
3.2 Scénario de l'application. ....	31
3.2.1 Scénario pour l'administrateur et le modérateur. ....	32
3.2.2 Scénario pour le client visiteur.....	33
3.2.3 Scénario pour le client membre .....	34
3.3 Principaux écrans d'application.....	35
3.4 Conclusion .....	41
4 Sources. ....	42

# 1. Dossier d'analyse

## 1.1 Présentation du sujet

Mon projet consiste en la conception d'une application web de type e-commerce qui gère l'achat de vêtements en ligne. Pour se faire je me suis renseigné. En effet, comme il n'y a absolument rien de nouveau sous le soleil, ni rien d'introuvable sur le net, j'ai regardé un nombre incalculable de vidéos, et lu un bon paquet de documents concernant la gestion d'une application e-commerce, jusque dans les moindres détails.

L'idée d'une app de type e-commerce m'a semblé intéressante car sa conception englobe plusieurs « savoir-faire » qui me seront utiles dans mes projets futurs. (Savoir gérer différents types d'utilisateurs, gérer le compte des clients, leurs achats, leurs inscriptions etc.)

## 1.2 Cahier de charge

Le frontend de l'application se chargera :

- D'afficher les différents produits organisés en différentes catégories et genres.
- D'offrir la possibilité de rechercher les différents produits via une barre de recherche.
- De proposer l'inscription aux différents visiteurs du site.
- De permettre aux utilisateurs inscrits de visualiser leurs comptes utilisateur.
- De permettre aux utilisateurs inscrits de pouvoir commenter et liker des produits
- Permettre aux clients (inscrits et non-inscrits) d'effectuer les achats du contenu de leurs panier virtuel.

Le backend de l'application va se charger de :

- La création, la suppression, la modification des différents produits. (Changer les images, corriger les prix, les descriptions etc.)
- La gestion des commentaires. (Supprimer les commentaires non pertinents, à caractère grossier etc.)
- La gestion du stock. Dès que le statut d'une commande passe à « payé » Tous les produits qui consistent cette commande verront leurs stocks décrémentés de la quantité achetée. Et lorsque le stock d'un produit passe à zéro il ne plus être acheté. Ni même ajouté dans le panier.
- Les réductions pour des utilisateurs particuliers. Chaque utilisateur possède un compteur d'achat qui s'incrémente à chaque achat. Lorsque le compteur atteint un certain nombre l'utilisateur recevra un mail avec un code de réduction pour sa prochaine commande.

### 1.2.1 Déploiement

L'application tournera en local pour le moment. Dans un serveur de développement des applications Django aux « localhost : 8000 ».

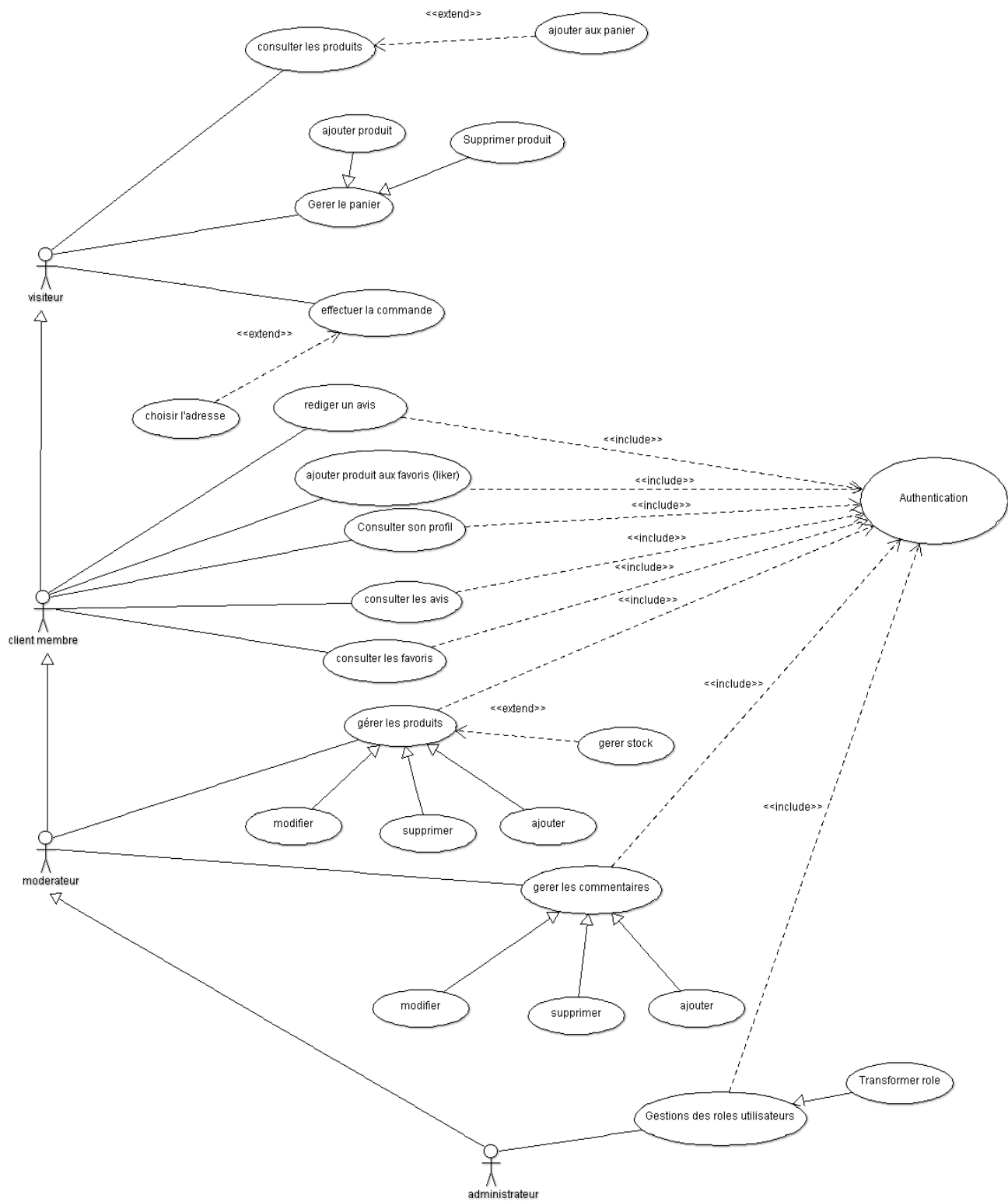
### 1.2.2 Rôles

Les différents types d'utilisateurs présents dans l'application sont :

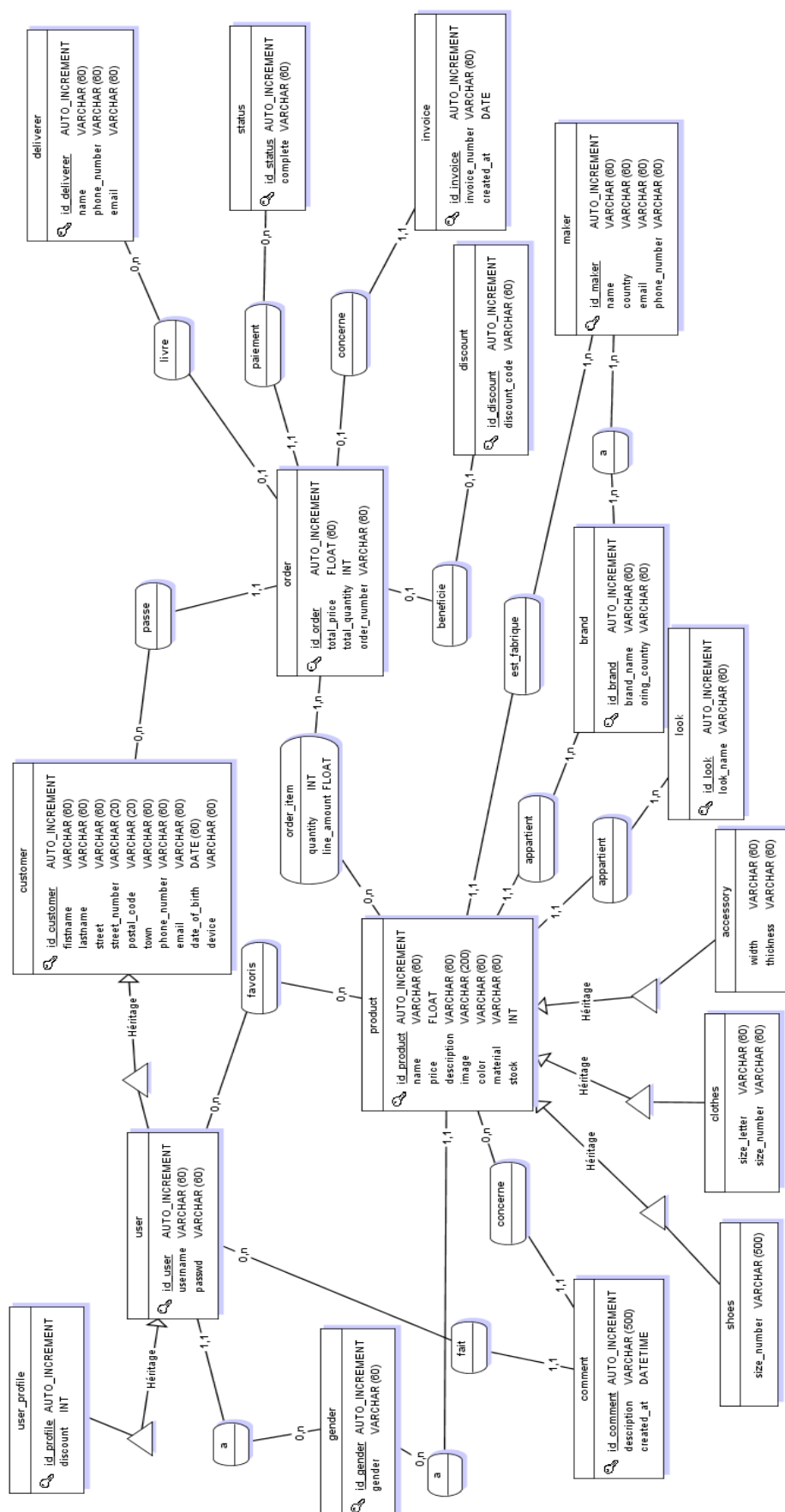
- L'administrateur
- Le modérateur
- Le client membre
- Le client visiteur

L'application se charge de faire passer automatiquement le visiteur au statut de membre lors d'une inscription. Mais c'est l'administrateur qui se réserve le droit de transformer le simple membre en modérateur, manuellement.

### 1.2.3 Diagramme de cas d'utilisation.



### 1.3 MCD



### 1.3.1 Dictionnaires de données du MCD.

#### Structure de la table gender

Colonne	Type(taille)	Null	Description
id_gender	Int	Non	Identifiant du genre
Gender	Varchar(60)	non	Nom du genre (homme, femme, enfant)

#### Structure de la table customer

Colonne	Type(taille)	Null	Description
id_customer	Int	non	Identifiant du client
firstname	Varchar(60)	oui	Le prénom du client
lastname	Varchar(60)	non	Le nom de famille du client
Street	Varchar(60)	Non	La rue où habite le client
Street_number	Varchar(20)	Non	Le numéro de rue où habite le client
postal_code	Varchar(20)	Non	Le code postal du client
phone_number	Varchar(30)	Oui	Le numéro de téléphone du client
Email	Varchar(60)	Non	L'adresse mail du client
date_of_birth	Varchar(60)	non	La date de naissance du client
device	Varchar(60)	oui	La valeur du cookie, afin d'identifier les utilisateurs non inscrits.



### Structure de la table user

Colonne	Type(taille)	Null	Description
id_user	Int	Non	Identifiant de l'utilisateur
Login	Varchar(60)	non	Nom de l'utilisateur
passwd	Varchar(60)	non	Mot de passe de l'utilisateur

### Structure de la table user\_profile

Colonne	Type(taille)	Null	Description
id_profile	Int	Non	Identifiant de l'utilisateur
discount	Int	non	C'est une valeur entière qui va s'incrémenter à chaque fois qu'un user passe une commande. A un certain nombre le user a une réduction.

### Structure de la table deliverer

Colonne	Type(taille)	Null	Description
id_deliverer	Int	Non	Identifiant du livreur
Name	Varchar(60)	non	Nom du livreur
phone_number	Varchar(60)	oui	Le numéro de téléphone du livreur
Email	Varchar(60)	non	L'adresse mail du livreur

### Structure de la table status

Colonne	Type(taille)	Null	Description
id_status	Int	Non	Identifiant du statut
complete	Bool	non	Permet de savoir si une commande a déjà été payée ou non.

### Structure de la table order

Colonne	Type(taille)	Null	Description
id_order	Int	Non	Identifiant du livreur
total_price	Int	non	Le prix total de toute la commande
order_number	Varchar(60)	non	Le numéro de commande généré automatiquement.

### Structure de la table maker

Colonne	Type(taille)	Null	Description
id_maker	Int	Non	L'identifiant du fabricant
Name	Int	non	Le nom du fabricant
country	Varchar(60)	non	Le pays du fabricant
Email	Varchar(60)	non	L'adresse mail du fabricant
phone_number	Varchar(60)	non	Le numéro de téléphone du fabricant

### Structure de la table invoice

Colonne	Type(taille)	Null	Description
id_invoice	Int	Non	L'identifiant de la facture
invoice_number	Varchar(60)	non	Le numéro de facture.
created_at	Date	non	La date de création de la facture

### Structure de la table brand

Colonne	Type(taille)	Null	Description
Id_brand	Int	Non	L'identifiant de la marque
brand_name	Varchar(60)	non	Le nom de la marque
origin_country	Varchar(60)	non	Le pays d'origine de la marque

### Structure de la table look

Colonne	Type(taille)	Null	Description
id_look	Int	Non	L'identifiant du look
look_name	Varcher(60)	non	Le nom du look

### Structure de la table product

Colonne	Type(taille)	Null	Description
id_product	Int	Non	L'identifiant du produit
Name	Varchar(60)	non	Le nom du produit
Price	Float	non	Le prix du produit
description	Varchar(60)	non	La description du produit
Image	Varchar(60)	non	L'image du produit. En type varchar car les images sont en réalité des url.
Color	Varchar(60)	non	Couleur du produit
material	Varchar(60)	non	La matière du produit (coton, daim, cuir etc.)
Stock	Int	non	Le stock du produit.

### Structure de la table shoes

Colonne	Type(taille)	Null	Description
id_product	Int	Non	L'identifiant d'un produit. Qui correspondra à une chaussure.
size_number	Varchar(30)	non	La taille d'une chaussure.

### Structure de la table comment

Colonne	Type(taille)	Null	Description
id_comment	Int	Non	L'identifiant du commentaire
description	Varchar(300)	non	La description du commentaire
created_at	Datetime	non	La date du commentaire

### Structure de la table clothes

Colonne	Type(taille)	Null	Description
id_product	Int	Non	L'identifiant d'un produit. Qui correspond ici à un vêtement
Name	Int	non	Le nom du vêtement
size_letter	Varchar(20)	non	La taille du produit en lettre (M, L, XL etc.)

### Structure de la table accessory

Colonne	Type(taille)	Null	Description
id_accessory	Int	Non	L'identifiant de l'accessoire
Name	Int	non	Le nom de l'accessoire
width	Varchar(60)	non	Largeur de l'accessoire
thickness	Varchar(60)	non	Epaisseur de l'accessoire

### Structure de la order\_item

Colonne	Type(taille)	Null	Description
id_product	Int	Non	L'identifiant du produit qui est dans le panier.
Id_order	Int	non	L'identifiant de la commande dans laquelle le produit se trouve.
quantity	Int	non	Quantité du produit commandé
line_amount	Float	non	Le montant de ligne pour un produit. (si j'ai deux chemise à 15 eu, line_amount sera 30)

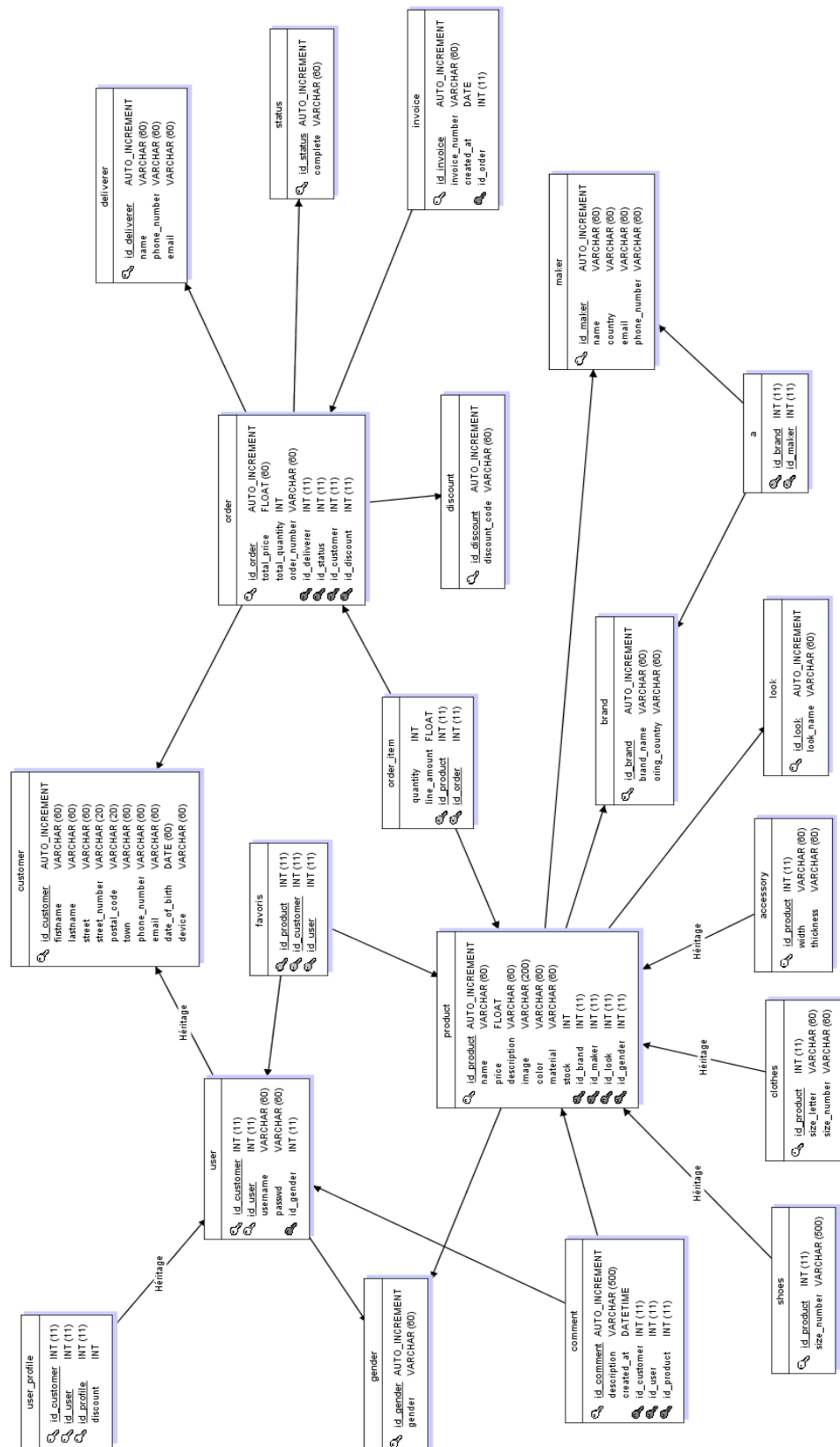
### Structure de la table favoris

Colonne	Type(taille)	Null	Description
id_product	Int	Non	L'identifiant du produit favori
Id_customer	Int	non	L'identifiant du client qui aime le produit.
Id_user	Varchar(60)	non	L'identifiant du user qui aime le produit.

### Structure de la table discount

Colonne	Type(taille)	Null	Description
id_discount	Int	Non	L'identifiant du code réduction
discount_code	Int	oui	Le code de réduction qu'on peut entrer lors de la commande pour avoir une réduction sur l'ensemble de la commande.

1.4 MLD



(\*) Au moment de la conversion de mon MCD en MLD Jmerise a gardé des relations fléchées pour les héritages. Les clés blanches sont des clés primaires et les clés sombres des clés étrangères.



## 1.5 Tables et droits d'accès.

Tables	Description	Droits Admin (1)	Droits Modérateur	Droits Membre	Droits Visiteur
gender	Les genres des produits (hommes, femmes, enfants)	Lecture, ajout, modification, suppression	Lecture	Lecture	Lecture
customer	Table qui regroupe les clients	Lecture, ajout, modification, suppression	Lecture	Lecture	Lecture
User	Table qui regroupe les utilisateurs	Lecture, ajout, modification, suppression	Lecture, ajout, Modification (2)	Lecture, ajout, Modification (2)	Lecture, ajout, Modification (2)
user_profile	Le profil de chaque utilisateurs	Lecture, ajout, modification, Suppression	Lecture, ajout, Modification (2)	Lecture, ajout, Modification (2)	Lecture, ajout, Modification (2)
deliverer	La table des livreurs	Lecture, ajout, modification, Suppression	Lecture	Lecture	Lecture
status	Le statut d'une commande (non payé, payé, livré)	Lecture, ajout, modification, Suppression	Lecture	Lecture	Lecture
maker	Tables des Fabricants	Lecture, ajout, modification, Suppression	Lecture	Lecture	Lecture
Brand	Tables des marques	Lecture, ajout, modification, Suppression	Lecture	Lecture	Lecture
Look	Tables de style (soirée, urbain etc.)	Lecture, ajout, modification, Suppression	Lecture	Lecture	Lecture
product	Table des produits	Lecture, ajout, modification, Suppression	Lecture, ajout, modification, suppression	Lecture	Lecture

Shoes	Table des chaussures	Lecture, ajout, modification, Suppression	Lecture, ajout, modification, suppression	Lecture	Lecture
comment	Table des commentaires	Lecture, ajout, modification, Suppression	Lecture, ajout, modification, suppression	Lecture	Lecture
clothes	Table des vêtements	Lecture, ajout, modification, Suppression	Lecture, ajout, modification, suppression	Lecture	Lecture
accessory	Table des accessoires	Lecture, ajout, modification, Suppression	Lecture, ajout, modification, suppression	Lecture	Lecture
discount	Table qui contient les codes de réductions	Lecture, ajout, modification, Suppression	Lecture	Lecture	Lecture
Order	Table des commandes	Lecture, ajout, modification, Suppression	Lecture	Lecture	Lecture
invoice	Table des factures	Lecture, ajout, modification, Suppression	Lecture	Lecture	Lecture
order_item	Table des produits dans la commande	Lecture, ajout, modification, suppression	Lecture	Lecture	Lecture
favoris	Tables des objets liké par l'utilisateur	Lecture, ajout, modification, suppression	Lecture	Lecture	Lecture

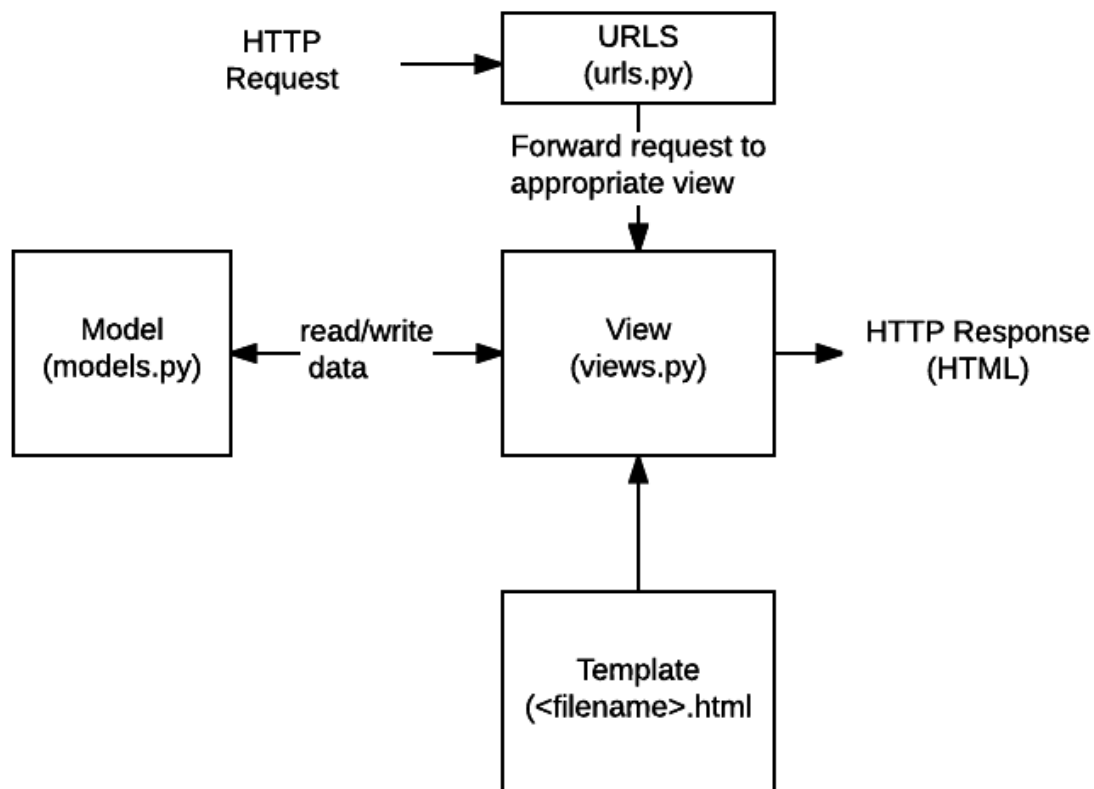
(1) L'admin possède tous les droits sur toute les tables. Car en réalité il peut via des commandes faire à peu près tout ce qu'il veut sur n'importe quelle donnée.

(2) Tous les users peuvent lire, écrire, modifier dans les tables user et user\_profile. Exemple : changer son mot de passe, modifier son genre, ou encore changer son nom d'utilisateur etc.

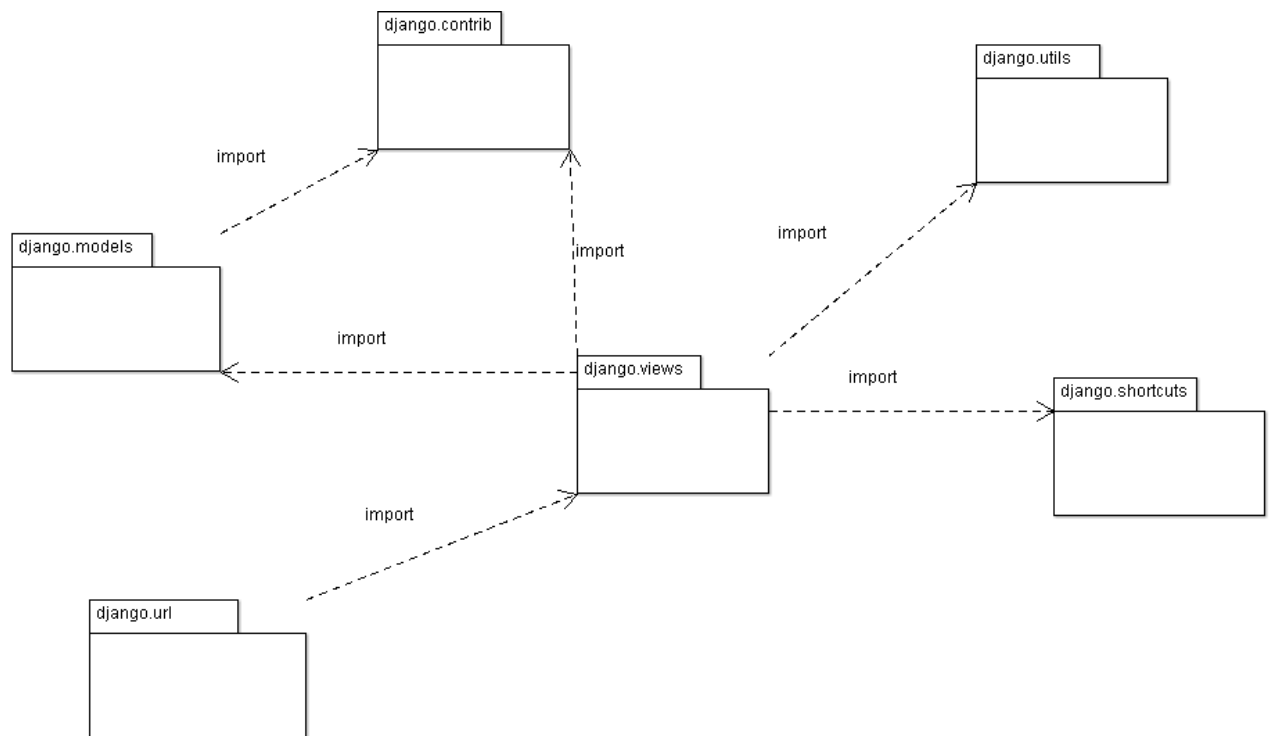
## 1.6 Analyse orientée objet.

Il me serait difficile voire impossible de représenter un diagramme de classe fidèle à celui utilisé dans ce projet car le Framework Django regorge d'une multitude de classes additionnelles qu'il utilise dans cette application. De ce fait, je montrerai plutôt un diagramme découpé en package de l'application dans son ensemble, selon le modèle MVT (Modèle View Template).

Petite explication du schéma MVT : L'utilisateur entre une requête HTTP sur le site (presse un bouton, clique sur un lien etc.) l'url lié à cette requête est identifié dans le package **urls.py** de Django. Cette url spécifique stocké dans le package **urls.py** va invoquer une vue dans le package **views.py**. Et La vue va recourir au model et au Template qui y est relié, puis va donner une réponse sur la page.



### 1.6.1 Diagramme de package correspondant.



Django.contrib contient les classes relatives à l'utilisateur. Ce qui fait que la classe utilisateur existe déjà dans Django, ce qui fait que je ne l'ai pas créé mais je l'ai importé dans mes modèles et dans mes vues.

Django.utils contient des classes utiles pour la gestion des dates.

Django.shortcuts contient des classes utiles pour le rendu des vues sur les pages.

## 2. La base de données physique.

```
#-----  
#   Script MySQL.  
#-----
```

```
#-----  
# Table: gender  
#-----
```

```
CREATE TABLE gender(  
    id_gender Int Auto_increment NOT NULL ,  
    gender Varchar (60) NOT NULL  
    ,CONSTRAINT gender_PK PRIMARY KEY (id_gender)  
)ENGINE=InnoDB;
```

```
#-----  
# Table: customer  
#-----
```

```
CREATE TABLE customer(  
    id_customer Int Auto_increment NOT NULL ,  
    firstname Varchar (60) NOT NULL ,  
    lastname Varchar (60) ,  
    street Varchar (60) NOT NULL ,  
    street_number Varchar (20) NOT NULL ,  
    postal_code Varchar (20) NOT NULL ,  
    town Varchar (60) NOT NULL ,  
    phone_number Varchar (60) ,  
    email Varchar (60) NOT NULL ,  
    date_of_birth Date NOT NULL ,  
    device Varchar (60)  
    ,CONSTRAINT customer_PK PRIMARY KEY (id_customer)  
)ENGINE=InnoDB;
```

```
#-----  
# Table: user
```

#-----

```
CREATE TABLE user(  
    id_customer Int NOT NULL ,  
    id_user    Int NOT NULL ,  
    username   Varchar (60) NOT NULL ,  
    passwd     Varchar (60) NOT NULL ,  
    id_gender  Int NOT NULL  
,CONSTRAINT user_PK PRIMARY KEY (id_customer,id_user)  
  
    ,CONSTRAINT user_customer_FK FOREIGN KEY (id_customer) REFERENCES  
customer(id_customer)  
    ,CONSTRAINT user_gender0_FK FOREIGN KEY (id_gender) REFERENCES  
gender(id_gender)  
)ENGINE=InnoDB;
```

#-----

# Table: user\_profile

#-----

```
CREATE TABLE user_profile(  
    id_customer Int NOT NULL ,  
    id_user    Int NOT NULL ,  
    id_profile  Int NOT NULL ,  
    discount   Int NOT NULL  
,CONSTRAINT user_profile_PK PRIMARY KEY (id_customer,id_user,id_profile)  
  
    ,CONSTRAINT user_profile_user_FK FOREIGN KEY (id_customer,id_user)  
REFERENCES user(id_customer,id_user)  
)ENGINE=InnoDB;
```

#-----

# Table: deliverer

#-----

```
CREATE TABLE deliverer(  
    id_deliverer Int Auto_increment NOT NULL ,  
    name        Varchar (60) NOT NULL ,
```

```

        phone_number Varchar (60) ,
        email      Varchar (60) NOT NULL
    ,CONSTRAINT deliverer_PK PRIMARY KEY (id_deliverer)
)ENGINE=InnoDB;

```

```

#-----
# Table: status
#-----

```

```

CREATE TABLE status(
    id_status Int Auto_increment NOT NULL ,
    complete Varchar (60) NOT NULL
    ,CONSTRAINT status_PK PRIMARY KEY (id_status)
)ENGINE=InnoDB;

```

```

#-----
# Table: maker
#-----

```

```

CREATE TABLE maker(
    id_maker  Int Auto_increment NOT NULL ,
    name      Varchar (60) NOT NULL ,
    country   Varchar (60) NOT NULL ,
    email     Varchar (60) NOT NULL ,
    phone_number Varchar (60) NOT NULL
    ,CONSTRAINT maker_PK PRIMARY KEY (id_maker)
)ENGINE=InnoDB;

```

```

#-----
# Table: brand
#-----

```

```

CREATE TABLE brand(
    id_brand  Int Auto_increment NOT NULL ,
    brand_name Varchar (60) NOT NULL ,
    oring_country Varchar (60) NOT NULL
    ,CONSTRAINT brand_PK PRIMARY KEY (id_brand)
)ENGINE=InnoDB;

```

```
)ENGINE=InnoDB;
```

```
#-----
```

```
# Table: look
```

```
#-----
```

```
CREATE TABLE look(  
    id_look Int Auto_increment NOT NULL ,  
    look_name Varchar (60) NOT NULL  
    ,CONSTRAINT look_PK PRIMARY KEY (id_look)  
)ENGINE=InnoDB;
```

```
#-----
```

```
# Table: product
```

```
#-----
```

```
CREATE TABLE product(  
    id_product Int Auto_increment NOT NULL ,  
    name      Varchar (60) NOT NULL ,  
    price     Float NOT NULL ,  
    description Varchar (60) NOT NULL ,  
    image     Varchar (200) NOT NULL ,  
    color     Varchar (60) NOT NULL ,  
    material  Varchar (60) NOT NULL ,  
    stock     Int NOT NULL ,  
    id_brand  Int NOT NULL ,  
    id_maker  Int NOT NULL ,  
    id_look   Int NOT NULL ,  
    id_gender Int NOT NULL  
    ,CONSTRAINT product_PK PRIMARY KEY (id_product)  
  
    ,CONSTRAINT product_brand_FK FOREIGN KEY (id_brand) REFERENCES  
brand(id_brand)  
    ,CONSTRAINT product_maker0_FK FOREIGN KEY (id_maker) REFERENCES  
maker(id_maker)  
    ,CONSTRAINT product_look1_FK FOREIGN KEY (id_look) REFERENCES look(id_look)  
    ,CONSTRAINT product_gender2_FK FOREIGN KEY (id_gender) REFERENCES  
gender(id_gender)
```



```
)ENGINE=InnoDB;
```

```
#-----
```

```
# Table: shoes
```

```
#-----
```

```
CREATE TABLE shoes(  
    id_product Int NOT NULL ,  
    size_number Varchar (500) NOT NULL  
    ,CONSTRAINT shoes_PK PRIMARY KEY (id_product)  
  
    ,CONSTRAINT shoes_product_FK FOREIGN KEY (id_product) REFERENCES  
product(id_product)  
)ENGINE=InnoDB;
```

```
#-----
```

```
# Table: comment
```

```
#-----
```

```
CREATE TABLE comment(  
    id_comment Int Auto_increment NOT NULL ,  
    description Varchar (500) NOT NULL ,  
    created_at Datetime NOT NULL ,  
    id_customer Int NOT NULL ,  
    id_user Int NOT NULL ,  
    id_product Int NOT NULL  
    ,CONSTRAINT comment_PK PRIMARY KEY (id_comment)  
  
    ,CONSTRAINT comment_user_FK FOREIGN KEY (id_customer,id_user) REFERENCES  
user(id_customer,id_user)  
    ,CONSTRAINT comment_product0_FK FOREIGN KEY (id_product) REFERENCES  
product(id_product)  
)ENGINE=InnoDB;
```

```
#-----
```

```
# Table: clothes
```

```
#-----
```

```

CREATE TABLE clothes(
    id_product Int NOT NULL ,
    size_letter Varchar (60) NOT NULL ,
    size_number Varchar (60) NOT NULL
    ,CONSTRAINT clothes_PK PRIMARY KEY (id_product)

    ,CONSTRAINT clothes_product_FK FOREIGN KEY (id_product) REFERENCES
product(id_product)
)ENGINE=InnoDB;

```

```

#-----
# Table: accessory
#-----

```

```

CREATE TABLE accessory(
    id_product Int NOT NULL ,
    width Varchar (60) NOT NULL ,
    thickness Varchar (60) NOT NULL
    ,CONSTRAINT accessory_PK PRIMARY KEY (id_product)

    ,CONSTRAINT accessory_product_FK FOREIGN KEY (id_product) REFERENCES
product(id_product)
)ENGINE=InnoDB;

```

```

#-----
# Table: discount
#-----

```

```

CREATE TABLE discount(
    id_discount Int Auto_increment NOT NULL ,
    discount_code Varchar (60)
    ,CONSTRAINT discount_PK PRIMARY KEY (id_discount)
)ENGINE=InnoDB;

```

```

#-----
# Table: order

```

#-----

```
CREATE TABLE order(  
    id_order    Int Auto_increment NOT NULL ,  
    total_price  Float NOT NULL ,  
    total_quantity Int NOT NULL ,  
    order_number Varchar (60) NOT NULL ,  
    id_deliverer Int ,  
    id_status    Int NOT NULL ,  
    id_customer  Int NOT NULL ,  
    id_discount  Int NOT NULL  
    ,CONSTRAINT order_PK PRIMARY KEY (id_order)  
  
    ,CONSTRAINT order_deliverer_FK FOREIGN KEY (id_deliverer) REFERENCES  
deliverer(id_deliverer)  
    ,CONSTRAINT order_status0_FK FOREIGN KEY (id_status) REFERENCES  
status(id_status)  
    ,CONSTRAINT order_customer1_FK FOREIGN KEY (id_customer) REFERENCES  
customer(id_customer)  
    ,CONSTRAINT order_discount2_FK FOREIGN KEY (id_discount) REFERENCES  
discount(id_discount)  
)ENGINE=InnoDB;
```

#-----

# Table: invoice

#-----

```
CREATE TABLE invoice(  
    id_invoice  Int Auto_increment NOT NULL ,  
    invoice_number Varchar (60) NOT NULL ,  
    created_at   Date NOT NULL ,  
    id_order     Int NOT NULL  
    ,CONSTRAINT invoice_PK PRIMARY KEY (id_invoice)  
  
    ,CONSTRAINT invoice_order_FK FOREIGN KEY (id_order) REFERENCES  
order(id_order)  
    ,CONSTRAINT invoice_order_AK UNIQUE (id_order)  
)ENGINE=InnoDB;
```

```
#-----  
# Table: order_item  
#-----
```

```
CREATE TABLE order_item(  
    quantity Int NOT NULL ,  
    line_amount Float NOT NULL ,  
    id_product Int NOT NULL ,  
    id_order Int NOT NULL  
    ,CONSTRAINT order_item_PK PRIMARY KEY (id_product,id_order)  
  
    ,CONSTRAINT order_item_product_FK FOREIGN KEY (id_product) REFERENCES  
product(id_product)  
    ,CONSTRAINT order_item_order0_FK FOREIGN KEY (id_order) REFERENCES  
order(id_order)  
)ENGINE=InnoDB;
```

```
#-----  
# Table: favoris  
#-----
```

```
CREATE TABLE favoris(  
    id_product Int NOT NULL ,  
    id_customer Int NOT NULL ,  
    id_user Int NOT NULL  
    ,CONSTRAINT favoris_PK PRIMARY KEY (id_product,id_customer,id_user)  
  
    ,CONSTRAINT favoris_product_FK FOREIGN KEY (id_product) REFERENCES  
product(id_product)  
    ,CONSTRAINT favoris_user0_FK FOREIGN KEY (id_customer,id_user) REFERENCES  
user(id_customer,id_user)  
)ENGINE=InnoDB;
```

```
#-----  
# Table: a  
#-----
```

```
CREATE TABLE a(  
    id_brand Int NOT NULL ,  
    id_maker Int NOT NULL  
    ,CONSTRAINT a_PK PRIMARY KEY (id_brand,id_maker)  
  
    ,CONSTRAINT a_brand_FK FOREIGN KEY (id_brand) REFERENCES brand(id_brand)  
    ,CONSTRAINT a_maker0_FK FOREIGN KEY (id_maker) REFERENCES  
maker(id_maker)  
)ENGINE=InnoDB;
```

### 3. Le dossier technique.

#### 3.1 Différents outils de programmation adoptés.



Développé en 2003 et publié en 2005 Django est un Framework Python qui permet le développement rapide de sites Web. Django prend soin d'une grande partie des tracas du développement web, de sorte à laisser le développeur se concentrer sur l'écriture de son application sans avoir besoin de réinventer la roue. Par exemple, le Framework s'occupe lui-même de la gestion de la base de données, sans avoir recours à des applications externes telles que phpMyAdmin etc. De plus Django a l'avantage de disposer d'une très grande communauté. Ce qui est plus pratique quand on cherche à se renseigner sur des forums par exemple.

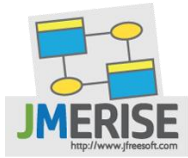
L'architecture utilisée par Django diffère légèrement de l'architecture MVC classique. En effet, la particularité de Django réside dans le fait qu'il *gère lui-même la partie contrôleur* (gestion des requêtes du client, des droits sur les actions...). Ainsi, nous parlons plutôt de Framework utilisant l'architecture **MVT** : **M**odèle-**V**ue-**T**emplate. (J'ai exposé le schéma de l'architecture plus haut. Au point 1.6)

Cette architecture reprend les définitions de modèle et de vue « classiques », et en introduit une nouvelle : le **template**. Un template en Django est un fichier HTML, aussi appelé en français « gabarit ». Il sera récupéré par la vue et envoyé au visiteur ; cependant, avant d'être envoyé, il sera analysé et exécuté par le Framework, comme s'il s'agissait d'un fichier avec du code. Django fournit un moteur de Template très utile qui permet, dans le code HTML, d'afficher des variables, d'utiliser des structures conditionnelles (if/else) ou encore des boucles (for), etc.

« Le Framework pour les perfectionnistes avec des deadlines. » C'est le slogan de Django et je dois dire que c'est pour ça que je l'ai choisi.



Bootstrap est un Framework CSS. C'est donc un ensemble de fichiers CSS et JavaScript fonctionnant ensemble, qu'on peut dès lors utiliser pour créer des design complexes, de façon plus simple. C'est principalement ce que j'ai utilisé dans mon projet. Pour embellir le rendu de mon site web.



Jmerise est un logiciel développé en java qui facilite l'élaboration de modèle selon la méthode Merise (Le j devant merise est donc pour java). Ce logiciel est facile à l'utilisation et permet à partir d'un MCD de générer automatiquement le MLD ainsi que le script SQL associé.



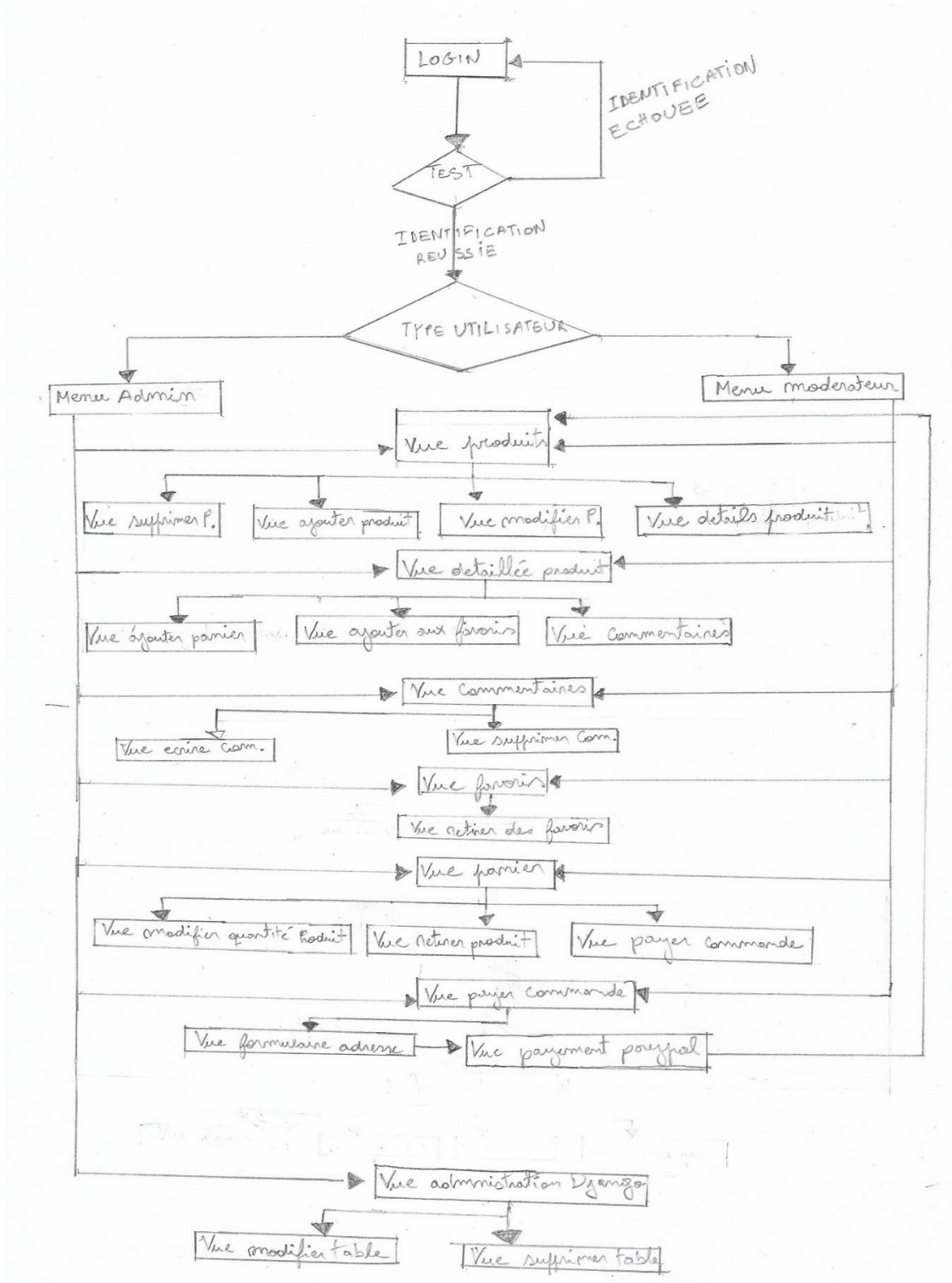
ArgoUML est un logiciel libre de création de diagramme UML. Je m'en suis servi pour dessiner mon diagramme de cas d'utilisation et celui de package.

## **ArgoUML**

### 3.2 Scénario de l'application.

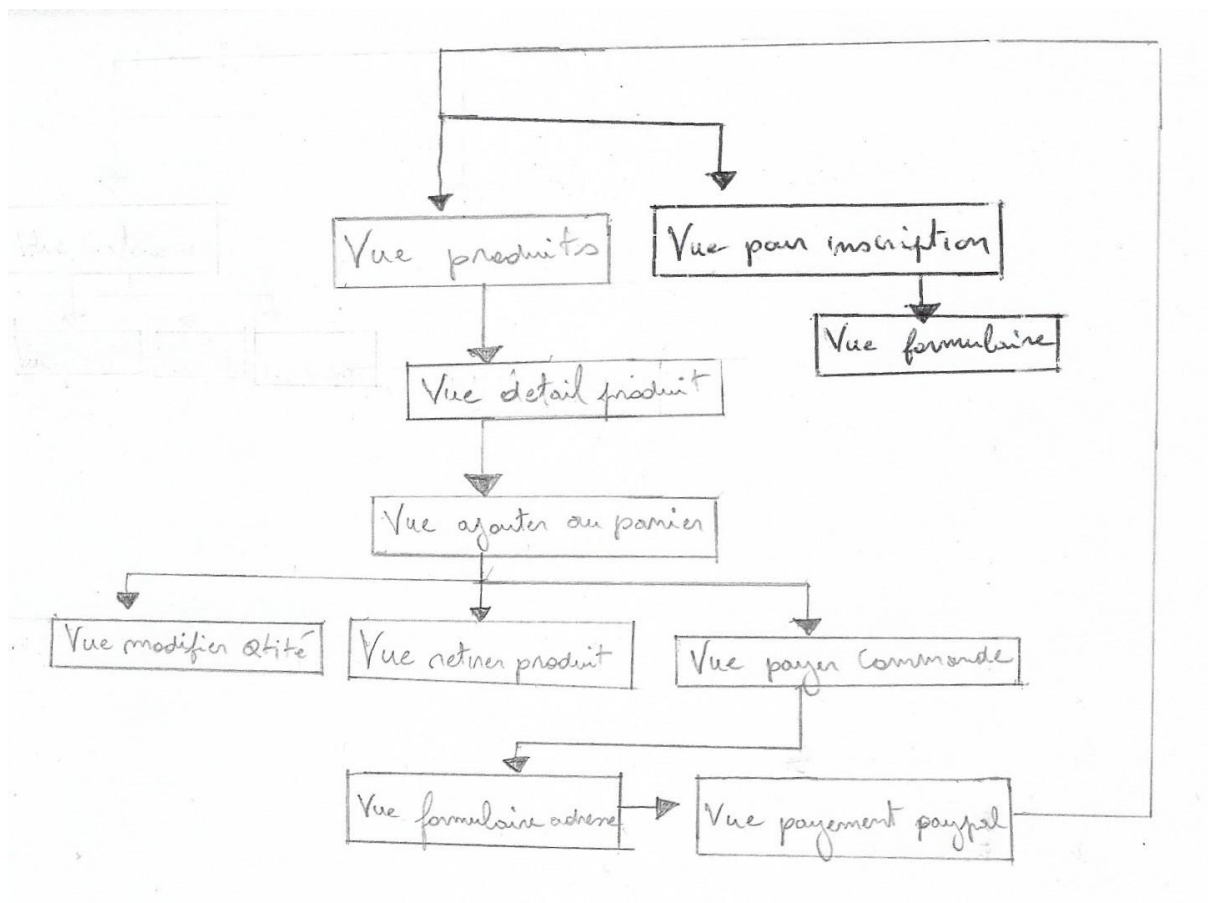
Note : Pour les schémas de navigation que j'ai dessiné dans les pages suivantes, je me suis inspiré d'un des exemples de TFE qui a été fourni dans la plateforme de l'école. N'ayant pas trouver de logiciel qui pourrait retourner le même rendu que dans l'exemple, j'ai été contraint de dessiner les scénarios de navigation à la main. Ce à mon plus grand regret. Mais j'ai fait de mon mieux pour être le plus claire possible. (Je n'ai pas trouvé d'interdiction à cela dans les consignes pour le diagramme de navigation)

### 3.2.1 Scénario pour l'administrateur et le modérateur.

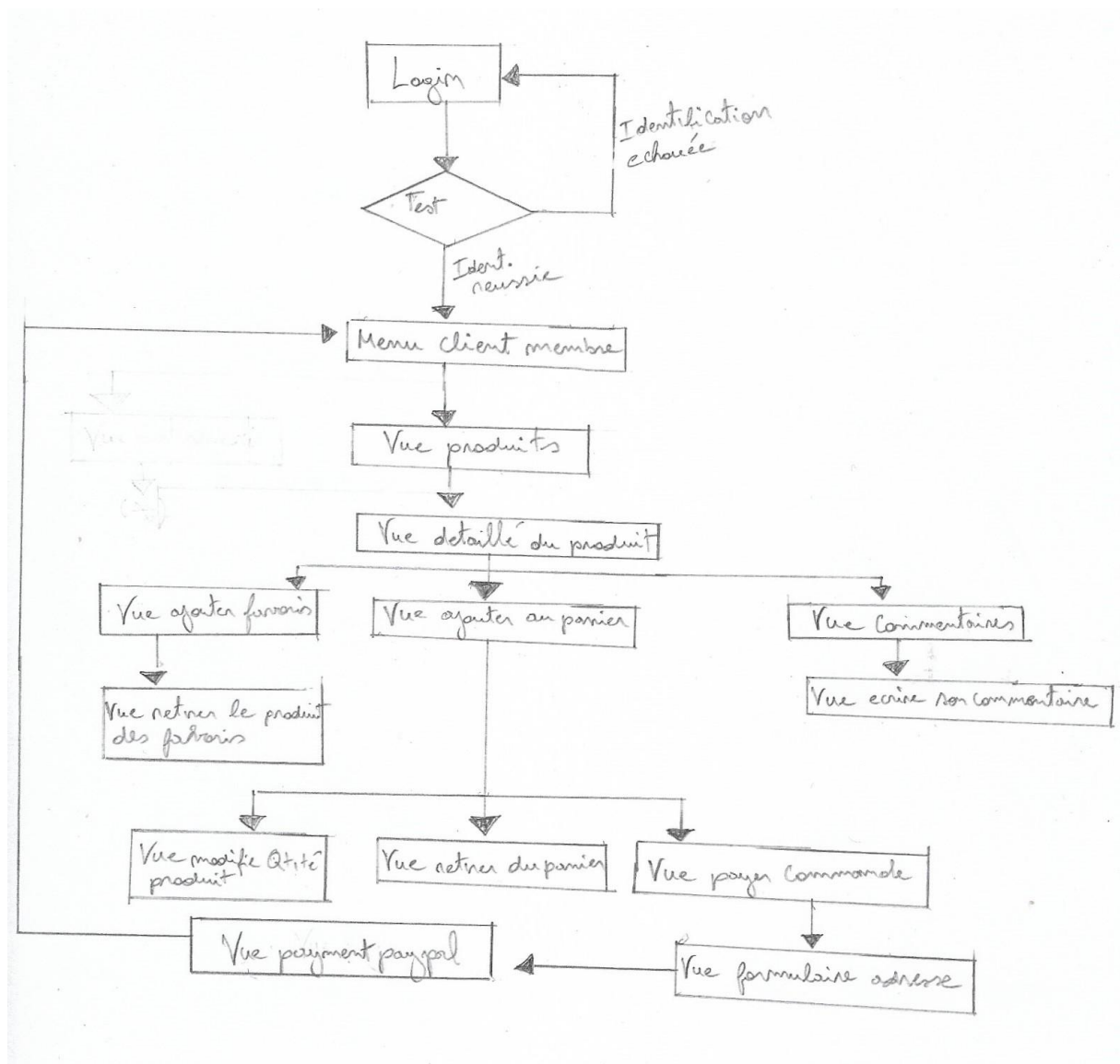




### 3.2.2 Scénario pour le client visiteur



### 3.2.3 Scénario pour le client membre



### 3.3 Principaux écrans d'application

Username\*

Required. 150 characters or fewer. Letters, digits and @/./+/-/\_ only.

First name

Last name

Email address

Password\*

- Your password can't be too similar to your other personal information.
- Your password must contain at least 8 characters.

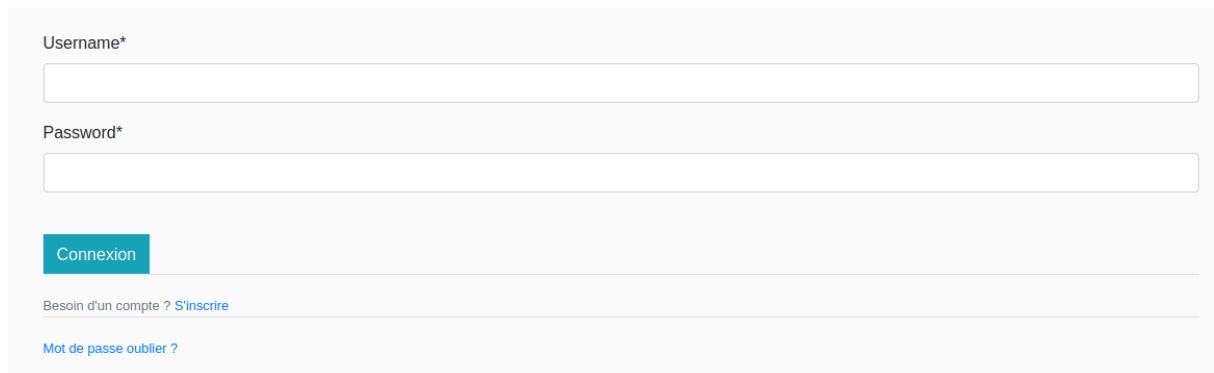
La page d'inscription. Le site adapte son affichage, en proposant à un utilisateur de s'inscrire si ce n'est pas déjà fait. Lorsqu'il cliquera sur le bouton du menu un formulaire comme ci-haut, lui sera afficher.

```
if form.is_valid():
    #username = request.POST['username']
    email = request.POST['email']
    password = request.POST['password1']
    username = form.cleaned_data.get('username')

    #ENVOI AUTOMATIQUE DU MAIL
    user=User.objects.create_user(username=username, email=email)
    user.set_password(password)
    user.is_active = False
    user.save()
    template = render_to_string('email.html',{'name':user.username})
    email = EmailMessage(
        'Sujet:Cimer cousin !',
        template,
        'caldjango@gmail.com',
        [user.email],
    )
    email.fail_silently=False
    email.send()
    form.save()
    messages.success(request, f'Compte pour {username} créé!')
    return redirect('login')
```

Fort heureusement, Django se charge lui-même de vérifier la validité des valeurs entrées par le visiteur. Vérification de l'email, le nom d'utilisateur qui doit être obligatoire, le mot de passe, qui lui doit avoir une certaine longueur, les expressions irrégulières etc. Tout ça en une commande : `if form.is_valid`. Si tout est en ordre l'utilisateur reçoit un mail automatique lui

invitant à activer son compte. (Le contenu du mail dans le code dans l'image était purement à titre d'essais.)



Username\*

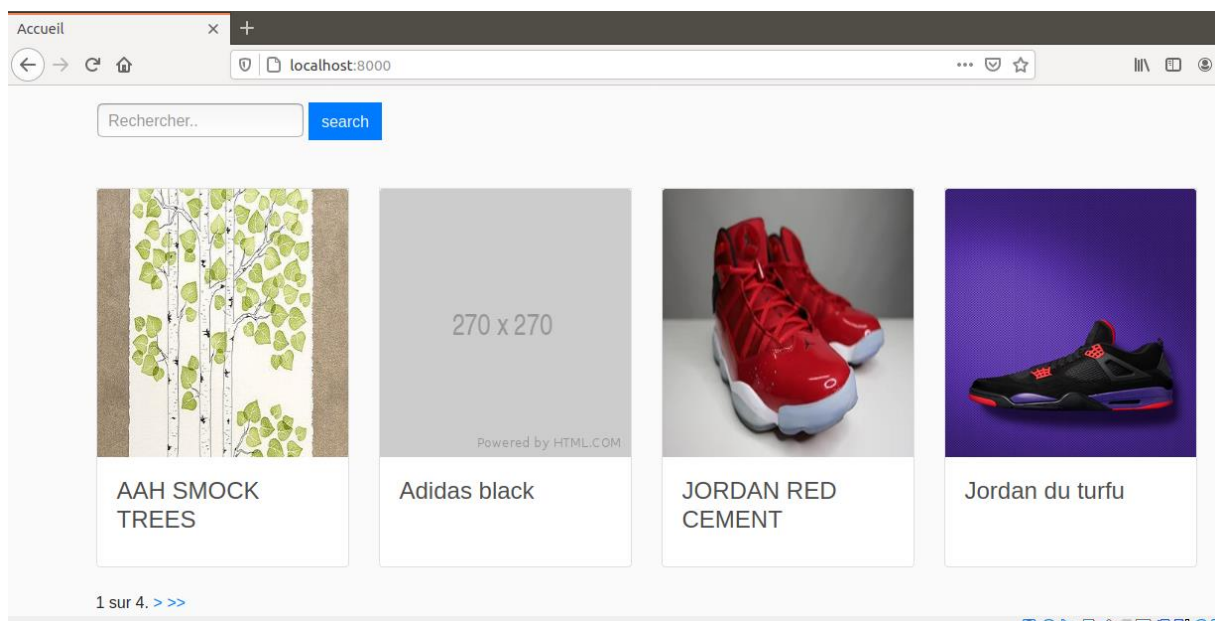
Password\*

Connexion

Besoin d'un compte ? [S'inscrire](#)

[Mot de passe oublier ?](#)

On a ici une page de connexion, ma foi assez classique, pour les membres inscrits. Le site prévoit l'envoi d'un email avec la possibilité de modifier son mot de passe au cas où un utilisateur étourdi (comme ça m'arrive tellement de l'être.) oublierait malencontreusement son mot de passe.



Un aperçu partiel de l'accueil du site. On peut voir l'adresse, « localhost:8000 » qui est le serveur de développement local Django, sur la barre url. Une barre de recherche ainsi que quelques produits, enfin, un système de pagination.

## JORDAN RED CEMENT



[Modifier produit](#) [Supprimer produit](#)

### Description

Chaussure jordan montante avec chaine du turfu.

- Prix(EUR) : 21.0
- Taille :
- Matière : cuire classique
- Couleur : Rouge

[Ajouter au panier](#)

[Avis](#)











Lorsqu'on clique sur un produit on est emmené vers une vue détaillée. Je m'étais connecté en tant qu'admin dans mon site, c'est l'unique raison pour laquelle, je peux voir le bouton modifier produit, qui me permettra d'accéder à une vue formulaire sur lequel je pourrai modifier les champs du dit produit. Le bouton supprimer qui, eh bien, supprime le produit. Ces changements affectent directement la base de données.

[← Retour aux achats](#)

Articles: **3**

Total: **58.64€**

[Commander](#)

	Item	Price	Quantity	Total	
	JORDAN RED CEMENT	21.00€	2  	42.00€	
	NIKE LUNARSTELLOS	16.64€	1  	16.64€	

Ici on a le contenu du panier, avec le nombre d'articles le prix total de la commande, le prix total de chaque élément présent. La possibilité d'augmenter et de diminuer la quantité, la possibilité de supprimer le produit. Pour ces deux dernières fonctionnalités, j'ai utilisé du javascript en associant les événements de click à une de mes vues Django qui se chargera de décrémenter l'attribut « quantity » d'un produit se trouvant dans le model « orderItem ».

Comme je l'ai déjà stipulé avant, les clients ne sont pas obligés de s'inscrire avant d'effectuer des achats sur le site. De tous ce que j'ai implémenté, ceci a été la fonctionnalité qui m'a posé le plus problème, mais après de longues heures de recherches j'ai pu trouver une solution.

L'idée consistait à créer un cookie, que j'ai appelé « device ». Ce dernier va contenir la valeur d'un très grand nombre généré aléatoirement et automatiquement, à chaque fois que le site sera lancé.

```
37
38     function uuidv4() {
39         //A function we can trigger to set the device ID
40         return 'xxxxxxxx-xxxx-4xxx-yxxx-xxxxxxxxxxxx'.replace(/[xy]/g, function(c) {
41             var r = Math.random() * 16 | 0, v = c == 'x' ? r : (r & 0x3 | 0x8);
42             return v.toString(16);
43         });
44     }
45
46
47
48
49     let device = getToken('device')
50
51     if (device == null || device == undefined){
52         device = uuidv4()
53     }
54
55     document.cookie = 'device=' + device + ";domain=;path=/"
56
57 </script>
58
```

Ici on a donc le script de la création du cookie device en Javascript.

Une fois le cookie créé (le nombre aléatoire) on peut le « get » et le placer dans un des attributs de notre table. Un attribut que j'ai aussi appelé device, qui fait partie de la table customer.

Ce qui fait que si un visiteur, forcément non connecté, veut effectuer une commande, mon application va créer un nouveau pseudo-client dans la table « customer » avec comme attribut « device » qui contiendra donc le nombre qui a été généré automatiquement au lancement du site.

```

13 <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.js"></script>
14 <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.min.js"></script>
15 <script src="https://kit.fontawesome.com/335dfa2d46.js" crossorigin="anonymous"></script>
16
17
18 <script type="text/javascript">
19 //To get the cookie
20 function getToken(name) {
21     var cookieValue = null;
22     if (document.cookie && document.cookie !== '') {
23         var cookies = document.cookie.split(';');
24         for (var i = 0; i < cookies.length; i++) {
25             var cookie = cookies[i].trim();
26             // Does this cookie string begin with the name we want?
27             if (cookie.substring(0, name.length + 1) === (name + '=')) {
28                 cookieValue = decodeURIComponent(cookie.substring(name.length + 1));
29                 break;
30             }
31         }
32     }
33     return cookieValue;
34 }
35 const csrftoken = getToken('csrftoken');
36
37


```

This file uses UTF-8 encoding

Ici le script pour récupérer le cookie au lancement du site.



Infos de livraison:

Paypal



[← Retour au panier](#)

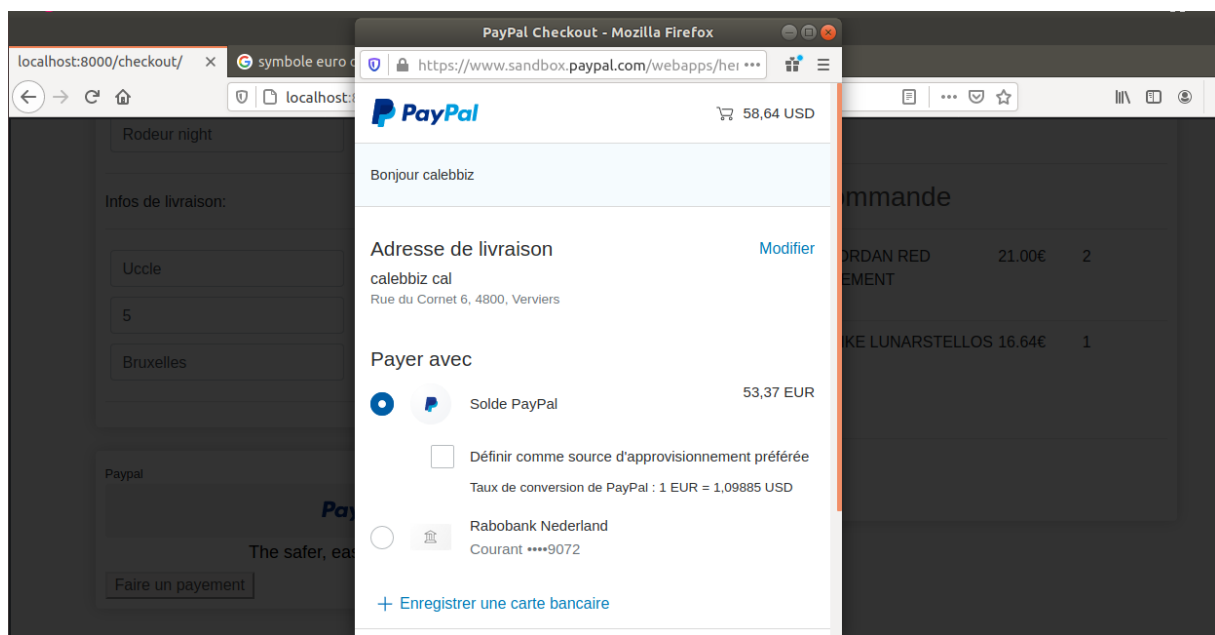
### Resumer de commande

	JORDAN RED CEMENT	21.00€	1
	LUNETTES PAILLETTE	23.04€	1

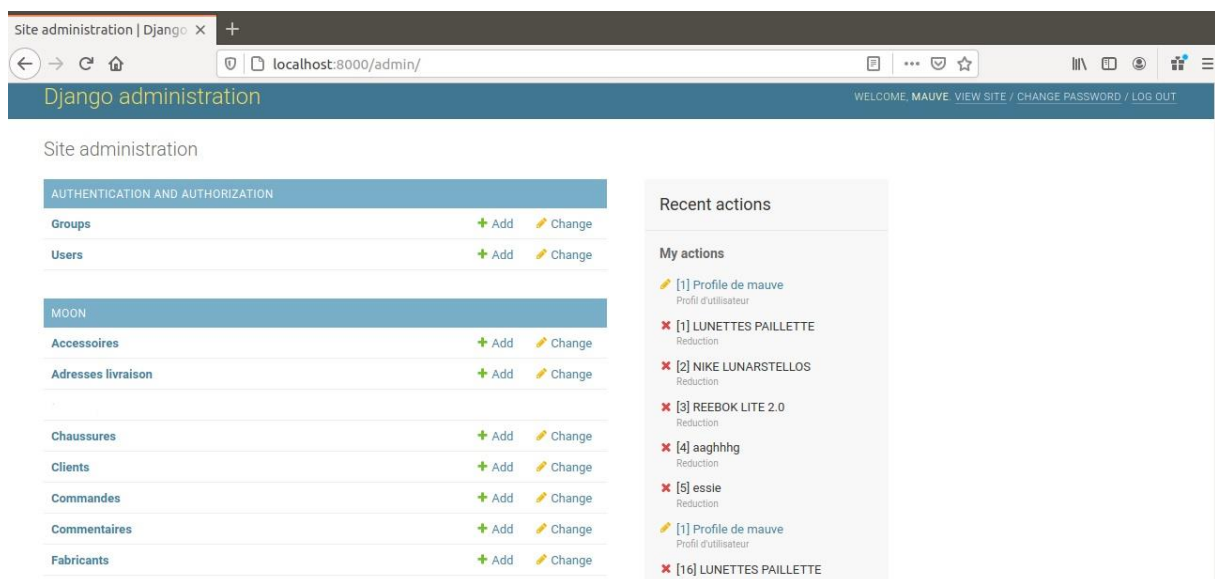
Items: 2

Total: 44.04€

Ici on a donc le résumer de la commande + le formulaire d'adresse et l'option de paiement Paypal (qui se dévoile lorsque le client appuie sur un bouton « payer »).



Enfin la vue paypal pour effectuer le paiement.



C'est ce à quoi ressemble la page d'administration Django qui gère directement la base de données. Il faut avoir une autorisation de l'admin pour y accéder. Car on peut modifier et supprimer n'importe quelle table à sa guise. On peut même voir le chemin dans la barre url « localhost:8000/admin/ »

Ceci ne représente évidemment que quelques images des pages que j'ai jugé importantes. L'application possède encore bon nombre de fonctionnalités fort sympathiques.



### 3.4 Conclusion

L'élaboration de ce projet a été l'un des travaux, les plus conséquents, qu'il me soit jamais arrivé de concevoir. Je suis content d'avoir sué, je suis content d'avoir quelques fois perdu mon calme face à des bugs de la mort qui tue... Bref, je suis content d'avoir appris. Le fonctionnement d'un Framework, la gestion approfondit des bases de données, le fonctionnement de python, javascript, Html et bien d'autres notions de programmations.

Pour améliorer cette application je pourrai éventuellement ajouter un système de messagerie et de consultation de profil entre les clients. Pour que mon application d'achat ressemble à un genre de réseau social avec possibilité d'achat. J'ai trouvé l'idée assez sympathique.

## 4 Sources.

- ° Zeste du savoir : <https://zestedesavoir.com/>
- ° Youtube : <https://www.youtube.com/>
- ° Stackoverflow : <https://stackoverflow.com/>
- ° Django : <https://docs.djangoproject.com/>
- ° Jmerise : <http://www.jfreesoft.com/JMerise/>