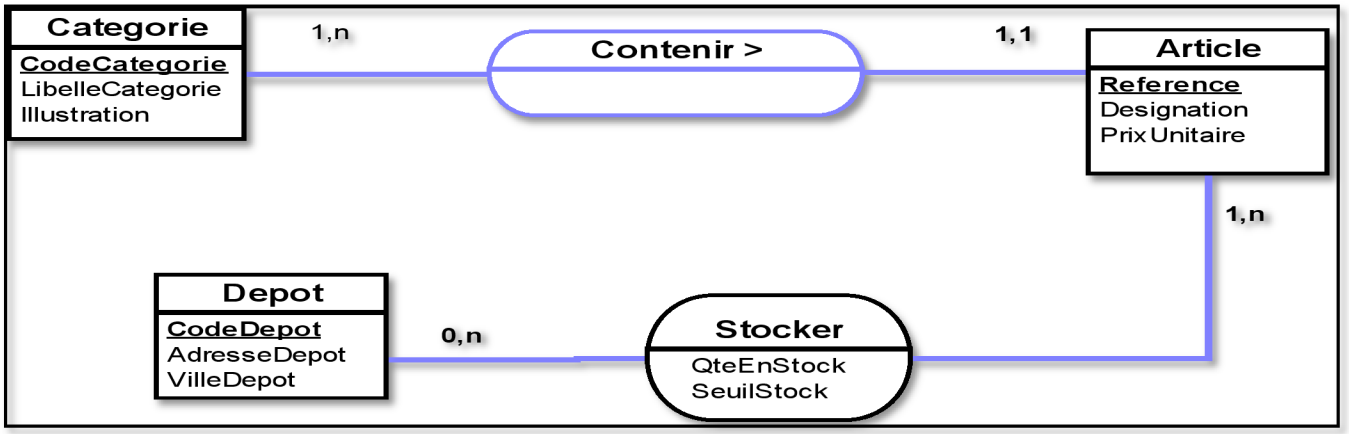
Framework Python-Django
Relations entre les modèles
(Models RelationShip)

On se propose de réaliser une application web avec le framework Python-Django pour gérer le stock des produits d'une société de distribution du matériel informatique. L'Application exploite une base de données SQLite3 dont le modèle conceptuel de données est:

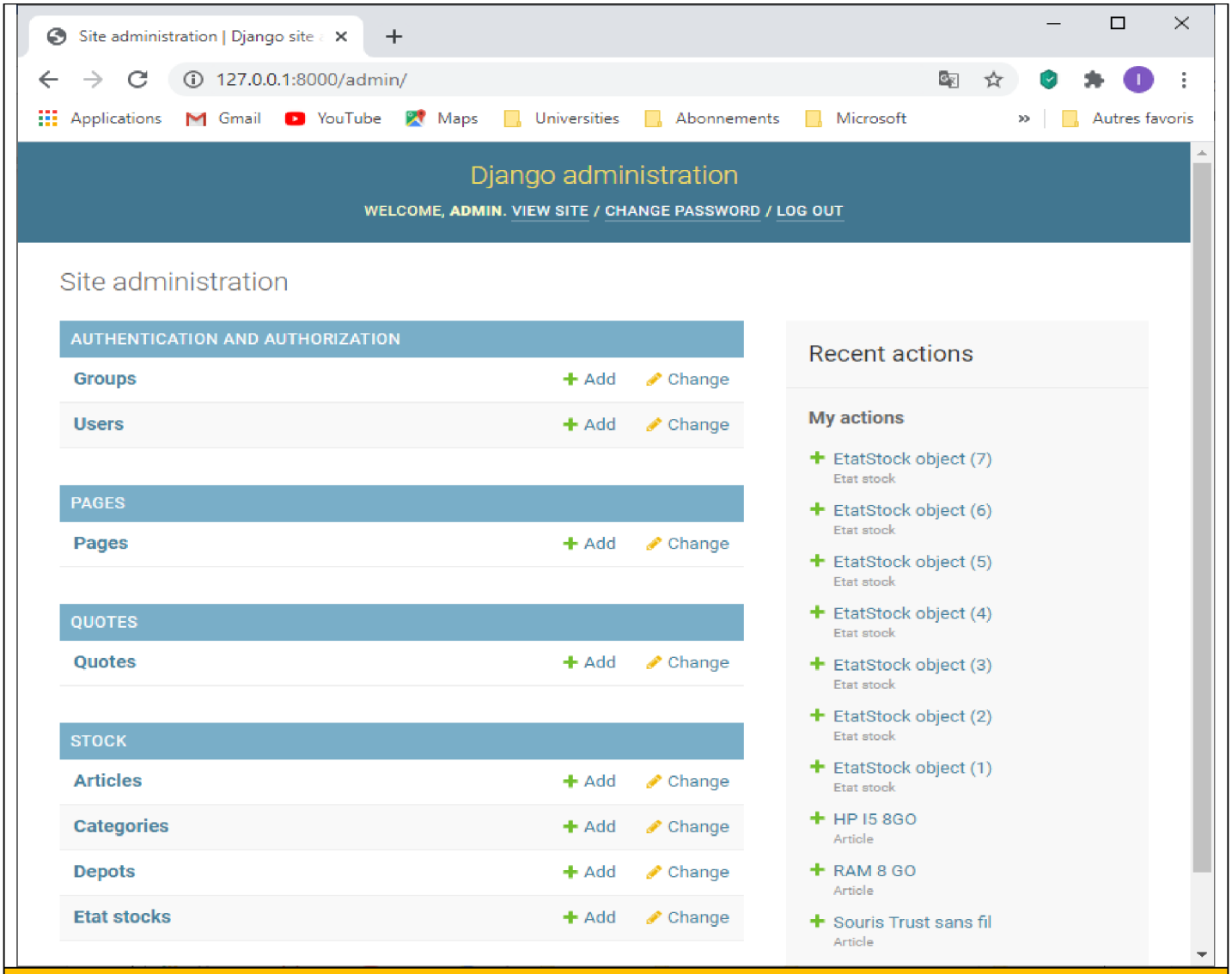


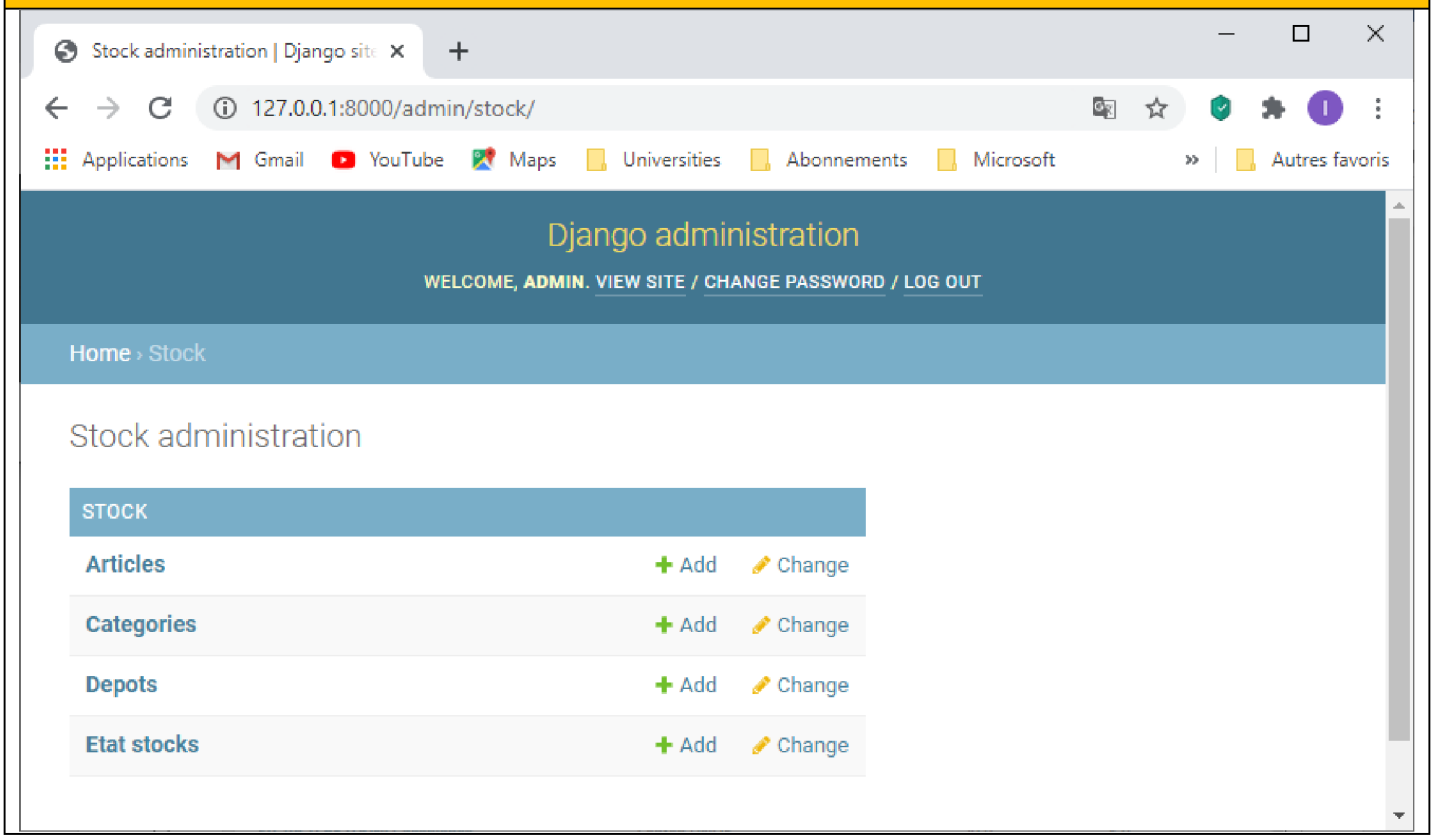
Son modèle relationnel correspondant est:

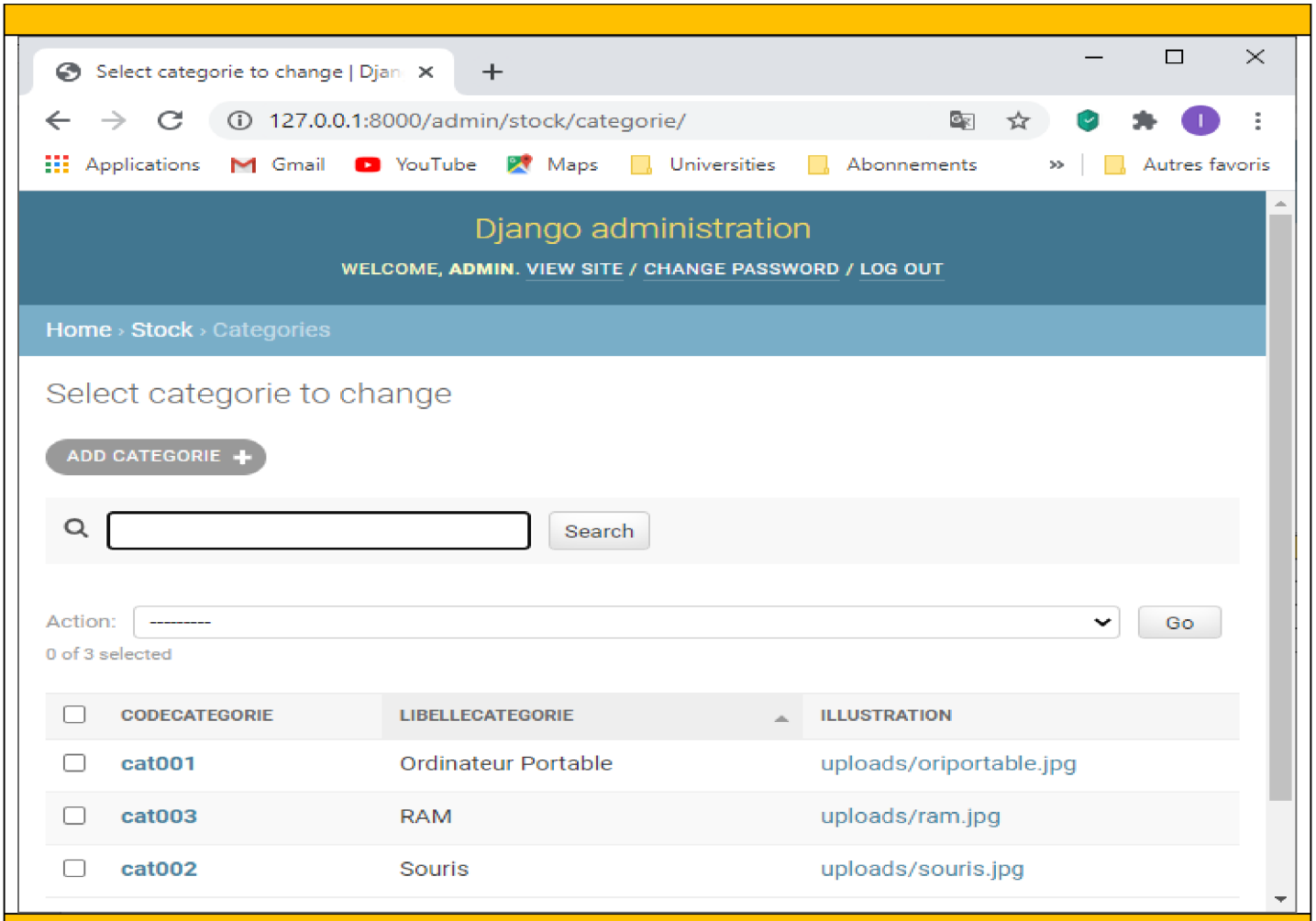
Categories(CodeCategorie, LibelleCategorie, Illustration)
Depots(CodeDepot, AdresseDepot, VilleDepot)
Articles(Reference, Designation, PrixUnitaire, #CodeCategorie)
Stock(#Reference, #CodeDepot, QuantiteEnStock, SeuilStock)

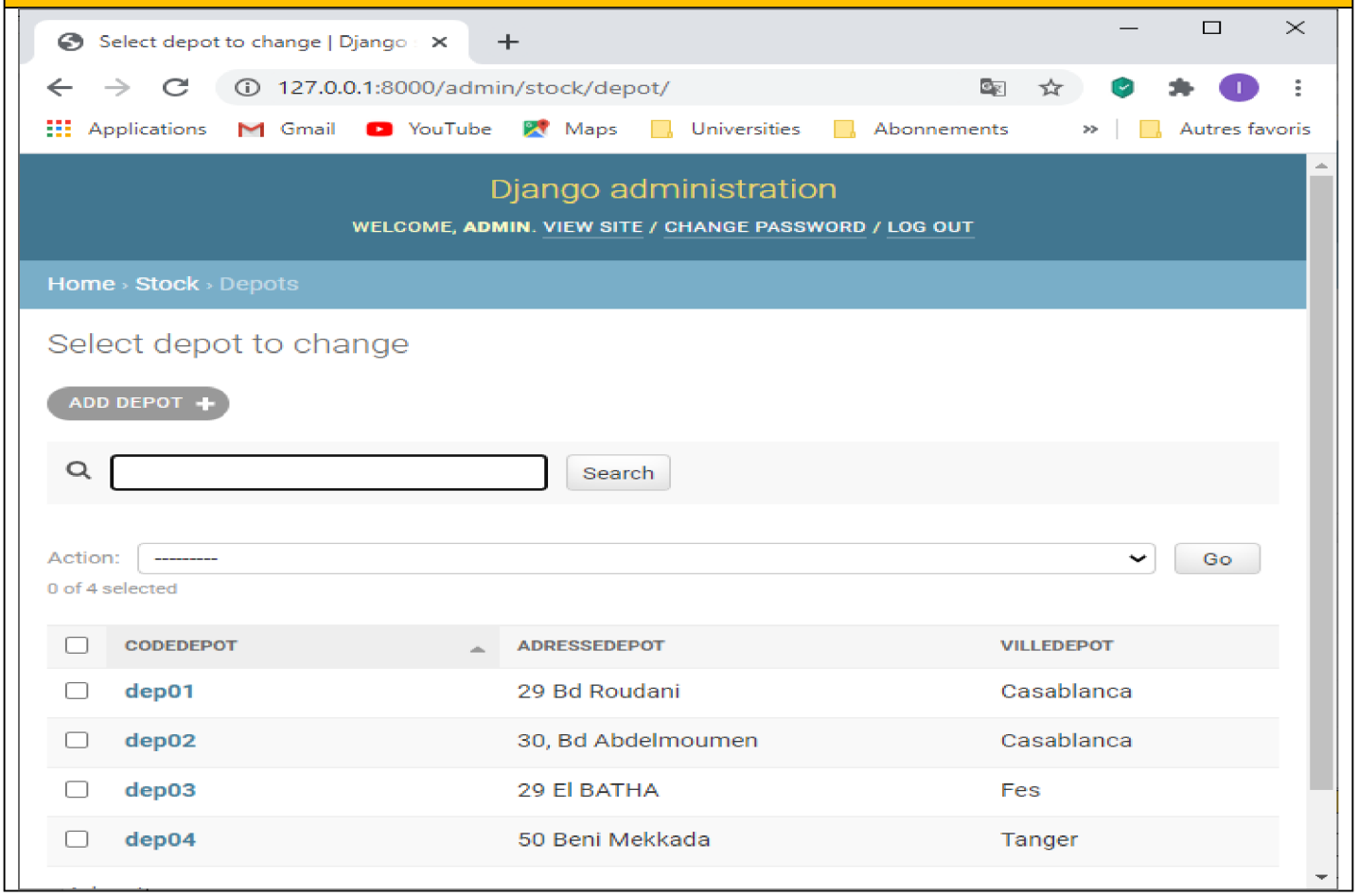
Travail à faire

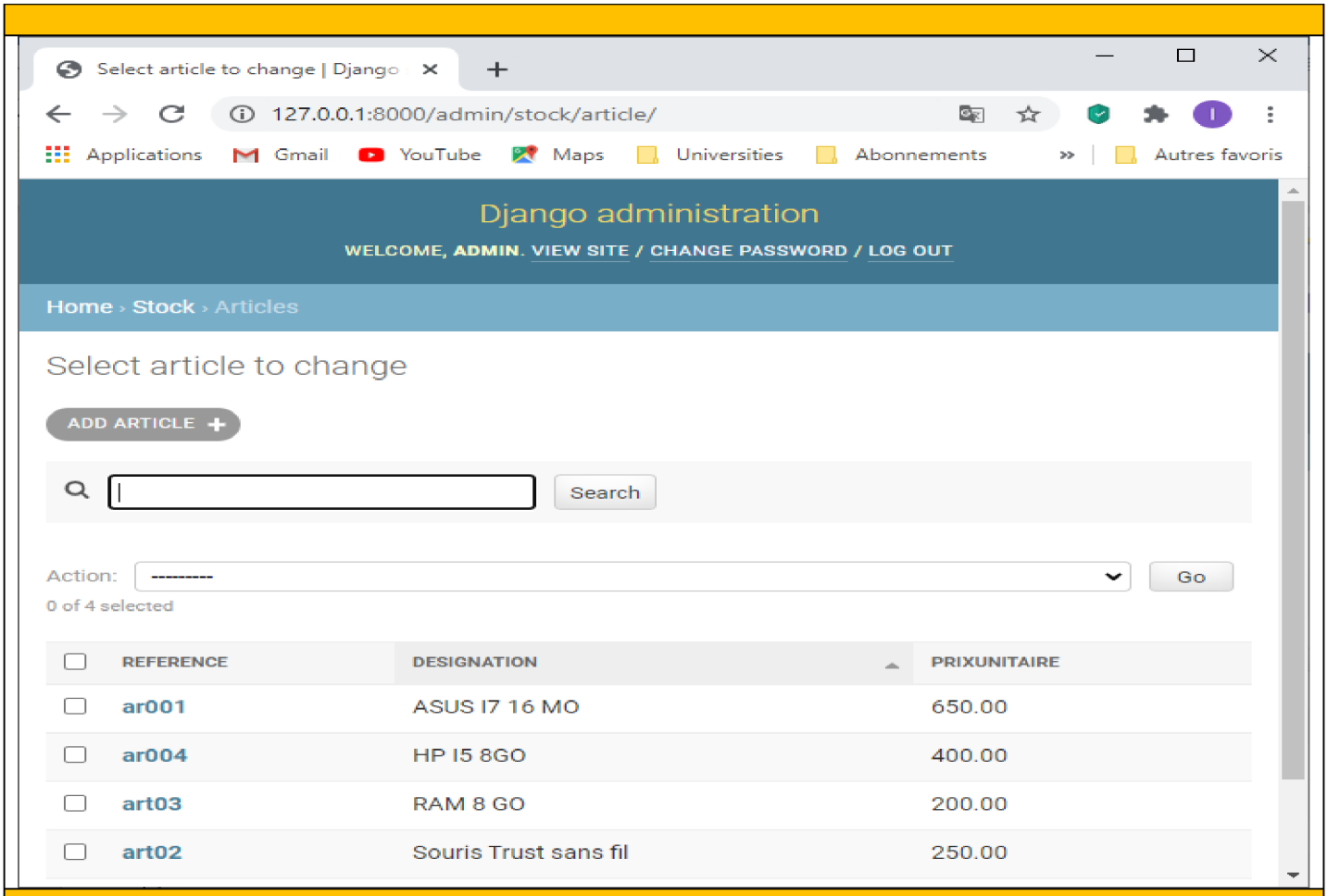
1. Créer un projet Django portant votre nom.
2. Créer une application portant votre prénom
3. Créer les modèles nécéssaires pour stocker les données
4. Faites le nécessaire pour que les pages de l'admin aient le format ci-dessous.

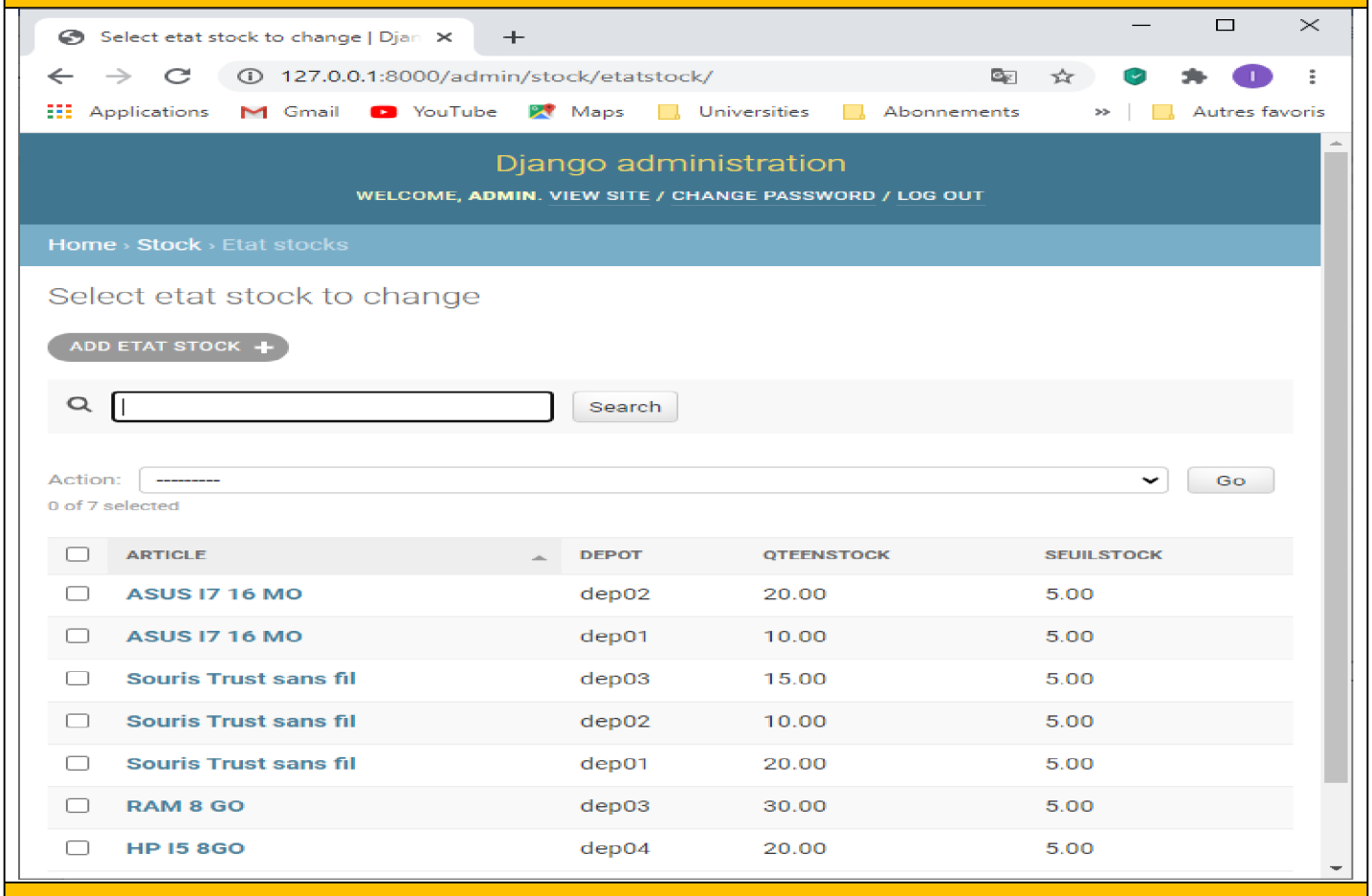

Site administration | Django site
127.0.0.1:8000/admin/
Applications
Gmail
YouTube
Maps
Universities
Abonnements
Microsoft
Autres favoris
Django administration
WELCOME, ADMIN. VIEW SITE / CHANGE PASSWORD / LOG OUT
Site administration
AUTHENTICATION AND AUTHORIZATION
Groups + Add Change
Users + Add Change
PAGES
Pages + Add Change
QUOTES
Quotes + Add Change
STOCK
Articles + Add Change
Categories + Add Change
Depots + Add Change
Etat stocks + Add Change
Recent actions
My actions
EtatStock object (7)
Etat stock
EtatStock object (6)
Etat stock
EtatStock object (5)
Etat stock
EtatStock object (4)
Etat stock
EtatStock object (3)
Etat stock
EtatStock object (2)
Etat stock
EtatStock object (1)
Etat stock
HP I5 8GO
Article
RAM 8 GO
Article
Souris Trust sans fil
Article


Stock administration | Django site
127.0.0.1:8000/admin/stock/
Applications
Gmail
YouTube
Maps
Universities
Abonnements
Microsoft
Autres favoris
Django administration
WELCOME, ADMIN. VIEW SITE / CHANGE PASSWORD / LOG OUT
Home › Stock
Stock administration
STOCK
Articles + Add Change
Categories + Add Change
Depots + Add Change
Etat stocks + Add Change


Select categorie to change | Djan
127.0.0.1:8000/admin/stock/categorie/
Applications
Gmail
YouTube
Maps
Universities
Abonnements
Autres favoris
Django administration
WELCOME, ADMIN. VIEW SITE / CHANGE PASSWORD / LOG OUT
Home › Stock › Categories
Select categorie to change
ADD CATEGORIE +
Search
Action: ---------
Go
0 of 3 selected
CODECATEGORIE
LIBELLECATEGORIE
ILLUSTRATION
cat001
Ordinateur Portable
uploads/oriportable.jpg
cat003
RAM
uploads/ram.jpg
cat002
Souris
uploads/souris.jpg


Select depot to change | Django
127.0.0.1:8000/admin/stock/depot/
Applications
Gmail
YouTube
Maps
Universities
Abonnements
Autres favoris
Django administration
WELCOME, ADMIN. VIEW SITE / CHANGE PASSWORD / LOG OUT
Home › Stock › Depots
Select depot to change
ADD DEPOT +
Search
Action: ---------
Go
0 of 4 selected
CODEDEPOT
ADRESSEDEPOT
VILLEDEPOT
dep01
29 Bd Roudani
Casablanca
dep02
30, Bd Abdelmoumen
Casablanca
dep03
29 El BATHA
Fes
dep04
50 Beni Mekkada
Tanger


Select article to change | Django
127.0.0.1:8000/admin/stock/article/
Applications
Gmail
YouTube
Maps
Universities
Abonnements
Autres favoris
Django administration
WELCOME, ADMIN. VIEW SITE / CHANGE PASSWORD / LOG OUT
Home › Stock › Articles
Select article to change
ADD ARTICLE +
Search
Action:
Go
0 of 4 selected
REFERENCE
DESIGNATION
PRIXUNITAIRE
ar001 ASUS I7 16 MO 650.00
ar004 HP I5 8GO 400.00
art03 RAM 8 GO 200.00
art02 Souris Trust sans fil 250.00


Select etat stock to change | Dja
127.0.0.1:8000/admin/stock/etatstock/
Applications
Gmail
YouTube
Maps
Universities
Abonnements
Autres favoris
Django administration
WELCOME, ADMIN. VIEW SITE / CHANGE PASSWORD / LOG OUT
Home › Stock › Etat stocks
Select etat stock to change
ADD ETAT STOCK +
Search
Action:
Go
0 of 7 selected
ARTICLE
DEPOT
QTEENSTOCK
SEUILSTOCK
ASUS I7 16 MO dep02 20.00 5.00
ASUS I7 16 MO dep01 10.00 5.00
Souris Trust sans fil dep03 15.00 5.00
Souris Trust sans fil dep02 10.00 5.00
Souris Trust sans fil dep01 20.00 5.00
RAM 8 GO dep03 30.00 5.00
HP I5 8GO dep04 20.00 5.00

5-Gestion des ventes
Réaliser le formulaire ci-dessous qui permet de saisir une vente d'un article

L'application doit saisir les ve,tes et les stocker dans une table. Elle doit ensuite afficher la situation des ventes selon le format ci-dessous:

Dépot	Référence	Date	QteVendu	Prix vente	Montant	Bénifice
xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx
xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx
xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx
xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx
				Totaux :	xxxx	xxxx

Indications:

Fichier models.py de l'application stock

```
from django.db import models

# Create your models here.
class Categorie(models.Model):
    codeCategorie = models.CharField(max_length=10, unique=True)
    libelleCategorie = models.CharField(max_length=100)
    illustration = models.FileField(upload_to='uploads/', blank=True)

    def __str__(self):
        return str(self.libelleCategorie)

class Depot(models.Model):
    codeDepot = models.CharField(max_length=10, unique=True)
    adresseDepot = models.CharField(max_length=30)
    villeDepot = models.CharField(max_length=25)

    def __str__(self):
        return str(self.codeDepot)

class Article(models.Model):
    reference = models.CharField(max_length=10, unique=True)
    designation = models.CharField(max_length=25)
    prixUnitaire = models.DecimalField(max_digits=5, decimal_places=2)
    categorie = models.ForeignKey(Categorie, on_delete=models.CASCADE)
    depot = models.ManyToManyField(Depot, through='EtatStock' )

    def __str__(self):
        return str(self.designation)


class EtatStock(models.Model):
    article = models.ForeignKey(Article, on_delete=models.CASCADE)
    depot = models.ForeignKey(Depot, on_delete=models.CASCADE)
    qteEnStock = models.DecimalField(max_digits=5, decimal_places=2)
    seuilStock = models.DecimalField(max_digits=5, decimal_places=2)
    class Meta:
        constraints = [
            models.UniqueConstraint(fields=['article', 'depot'], name='uqEtatStock'),
        ]
```

Fichier admin.py de l'application stock

```
from django.contrib import admin

# Register your models here.
from .models import Categorie, Depot, Article, EtatStock

class CategorieAdmin(admin.ModelAdmin):
    list_display = ('codeCategorie','libelleCategorie', 'illustration')
    ordering = ('libelleCategorie',)
    search_fields = ('libelleCatgorie',)

admin.site.register(Categorie, CategorieAdmin)

class DepotAdmin(admin.ModelAdmin):
    list_display = ('codeDepot','adresseDepot', 'villeDepot')
    ordering = ('codeDepot',)
    search_fields = ('villeDepot',)

admin.site.register(Depot, DepotAdmin)
```

```
class ArticleAdmin(admin.ModelAdmin):
    list_display = ('reference','designation', 'prixUnitaire')
    ordering = ('designation',)
    search_fields = ('designation',)

admin.site.register(Article, ArticleAdmin)


class EtatStockAdmin(admin.ModelAdmin):
    list_display = ('article','depot', 'qteEnStock', 'seuilStock')
    ordering = ('article',)
    search_fields = ('article',)

admin.site.register(EtatStock, EtatStockAdmin)
```

Django : Création des relations entre les modèles

Plusieurs à plusieurs

```
from django.db import models

class Publication(models.Model):
    title = models.CharField(max_length=30)

    class Meta:
        ordering = ['title']

    def __str__(self):
        return self.title

class Article(models.Model):
    headline = models.CharField(max_length=100)
    publications = models.ManyToManyField(Publication)

    class Meta:
        ordering = ['headline']

    def __str__(self):
        return self.headline
```

Plusieurs à plusieurs avec propriétés

```
from django.db import models

class Person(models.Model):
    name = models.CharField(max_length=128)

    def __str__(self):
        return self.name

class Group(models.Model):
    name = models.CharField(max_length=128)
    members = models.ManyToManyField(Person, through='Membership')

    def __str__(self):
        return self.name

class Membership(models.Model):
    person = models.ForeignKey(Person, on_delete=models.CASCADE)
    group = models.ForeignKey(Group, on_delete=models.CASCADE)
    date_joined = models.DateField()
    invite_reason = models.CharField(max_length=64)
```

Un à plusieurs

```
from django.db import models

class Reporter(models.Model):
    first_name = models.CharField(max_length=30)
    last_name = models.CharField(max_length=30)
    email = models.EmailField()

    def __str__(self):
        return "%s %s" % (self.first_name, self.last_name)
```

```python
class Article(models.Model):
    headline = models.CharField(max_length=100)
    pub_date = models.DateField()
    reporter = models.ForeignKey(Reporter, on_delete=models.CASCADE)

    def __str__(self):
        return self.headline

    class Meta:
        ordering = ['headline']
```

Un à un

```python
from django.db import models

class Place(models.Model):
    name = models.CharField(max_length=50)
    address = models.CharField(max_length=80)

    def __str__(self):
        return "%s the place" % self.name

class Restaurant(models.Model):
    place = models.OneToOneField(
        Place,
        on_delete=models.CASCADE,
        primary_key=True,
    )
    serves_hot_dogs = models.BooleanField(default=False)
    serves_pizza = models.BooleanField(default=False)

    def __str__(self):
        return "%s the restaurant" % self.place.name

class Waiter(models.Model):
    restaurant = models.ForeignKey(Restaurant, on_delete=models.CASCADE)
    name = models.CharField(max_length=50)

    def __str__(self):
        return "%s the waiter at %s" % (self.name, self.restaurant)
```