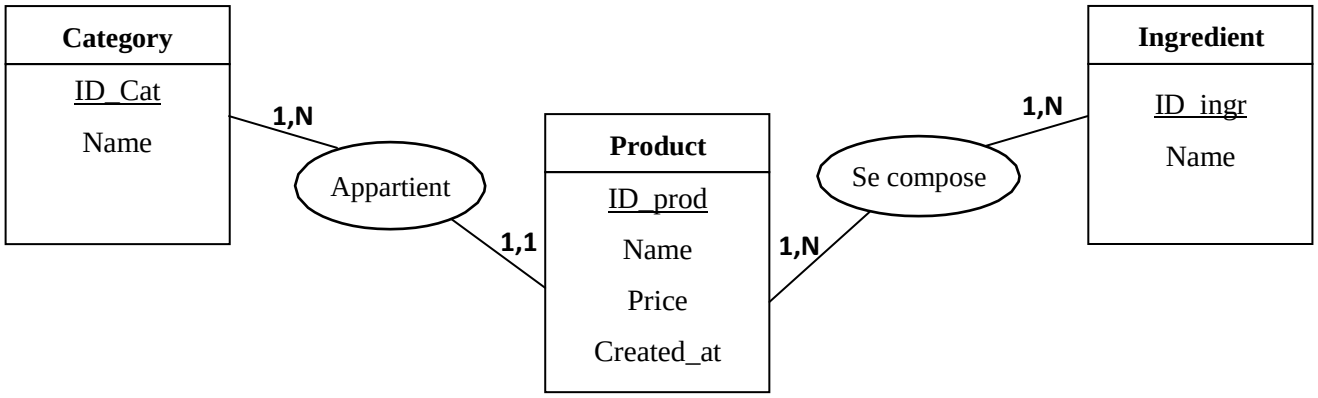
TP1.1 : Implémentation de votre premier MCD avec Django

Objectif : Le but de ce TP est d'implémenter votre premier MCD construit à partir d'un cahier des charges, l'afficher dans votre espace d'administration et faire de manipulations sur les items de vos classes (Modèles) pour tester les contraintes et les relations entre ces tables. Les deux fichiers que nous allons manipuler sont : ***models.py*** et ***admin.py***.

Exemple : Lors de la conception on a construit le mini MCD suivant :



Created_at : La date de création du produit, générée automatiquement lors de l'insertion d'un produit. Il s'agit de la date à laquelle le produit a été ajouté à votre BDD.

```
Created_at = models.DateTimeField(auto_now=True)
```

Indication : (Les relations entre les classes)

Les relations se déclarent dans la classe qui contient la clé étrangère

1 - * (*1 à plusieurs*): `name=models.ForeignKey(class2_name,on_delete=models.CASCADE)`

1 - 1 (*1 à 1*): `name=models.OneToOneField(class2_name)`

* - * (*plusieurs à plusieurs*): `name=models.ManyToManyField(class2_name, related_name="name2")`

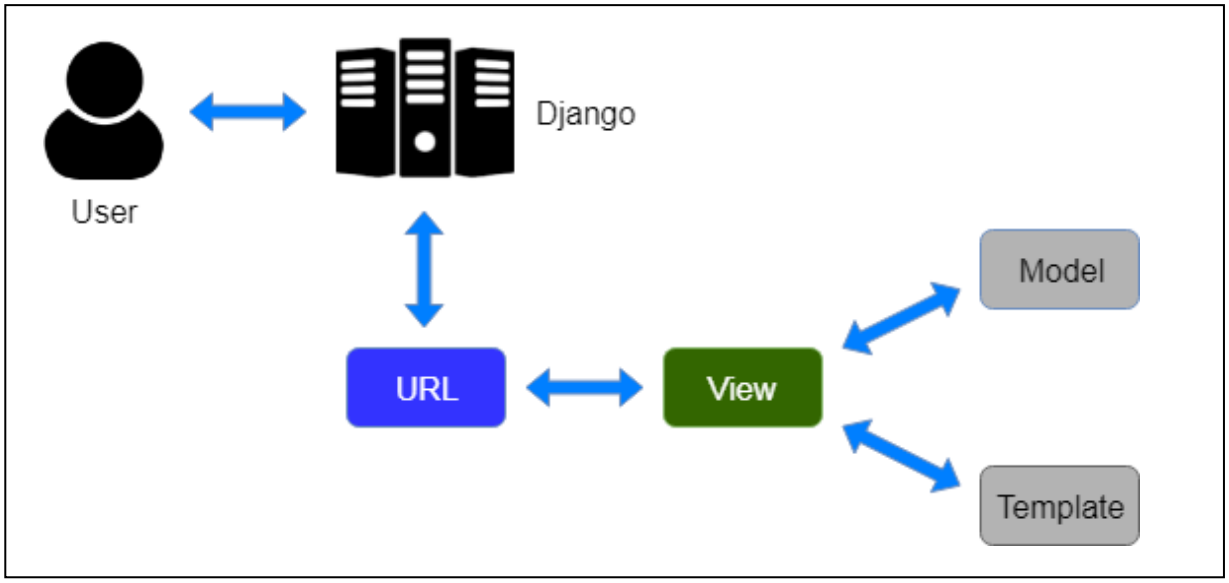
Le nom utilisé pour la relation inverse

Travail demandé :

1- Faire le passage au modèle relationnel.
2- Créer une nouvelle application Django nommée : **produit** dans votre même projet.
3- Modifier le fichier ***models.py*** pour permettre l'implémentation de la conception proposée.
4- Modifier le fichier ***admin.py*** pour permettre l'affichage de vos modèles dans l'interface d'administration Django.
5- Faire des manipulations de données sur vos tables (insertion, modification, suppression) pour tester le fonctionnement des relations et des contraintes.

TP1.2 : Les Templates (MVT pattern) dans Django

Objectif : Après avoir défini nos modèles et faire quelques manipulations des instances, il est temps d'écrire le code qui présente ces informations aux utilisateurs (clients). La première chose que nous devons faire est de déterminer quelles informations nous voulons afficher dans nos pages et de définir les URL à utiliser pour renvoyer ces ressources. Ensuite, nous allons créer un mappeur d'URL, des vues et des modèles pour afficher les pages. Les fichiers que nous allons modifier sont : ***urls.py***, ***views.py***, et templates ***/ index.html***



L'architecture de design pattern MVT.

Un ***Template*** est un fichier qui gère complètement la partie interface utilisateur (comme une page HTML), utilisé pour représenter le contenu réel.

Une ***view*** est utilisée pour exécuter le logique métier et interagir avec un modèle pour transporter des données et de les afficher dans un Template.

Travail demandé:

Dans le but d'afficher tous les produits disponibles dans une interface utilisateur (client), nous allons répondre à la requête suivante : « ***quelle sont les produits disponibles ?*** ».

1- Créer un nouveau dossier ***templates*** qui contient un fichier ***index.html*** dans le répertoire de votre application : **produit** >> ***templates*** >> ***index.html.***(pour afficher la liste des produits après les avoir récupérés de la BDD).

2- Créer un nouveau fichier ***urls.py*** dans le répertoire de votre application : **produit** >> **urls.py.**

3- Modifier le fichier ***urls.py*** (du projet) pour inclure les URL de l'application dans les URL du projet

```
from django.urls import path, include

urlpatterns = [  ...,
    path('',include('produit.urls'))
  ]
```

4- Modifier le fichier **produit >> urls.py** pour définir l'***url*** sur laquelle vous allez répondre à la requête & la ***fonction*** (vue) qui va répondre à la requête.

```
from django.urls import path
from . import views

urlpatterns = [  ...,
    path('list_produits/',views.affiche_produits))
  ]
```

Il est clair que la fonction *affiche_produits()* n'est pas encore définie, pour la définir il faut aller sur ***views.py***

5- Modifier le fichier ***views.py*** pour permettre d'interagir avec la BDD (***models.py)*** et de répondre à la requête par la suite, la réponse sera envoyé à un template (***index.html***) pour pouvoir l'afficher aux utilisateurs.

```
from django.shortcuts import render
from .models import Product

def affiche_produits(request):
    produit = Product.objects.all()
    return render(request,"index.html",{"product":produit})
```

6- Modifier le fichier ***index.html*** pour récupérer les données envoyées dans la fonction *affiche_produits()* et l'insérer dans votre page html. Apres avoir l'insérer ces données seront accessibles par tous utilisateur à partir de l'url : http://127.0.0.1:8000/list_produits/

```
<!DOCTYPE html>
<html>

<head>

</head>

<body>

<h1> la list des produits disponible : </h1>
<ul>
 {% for produit in product %}
     <li> {{produit.name}} </li>
  {% endfor %}
</ul>

</body>

</html>
```