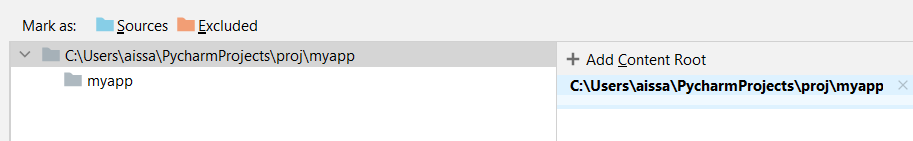
Formation Django

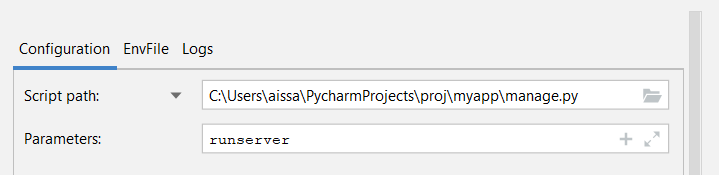
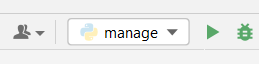
TP2 : Initiation au Framework Django

# **Génération d'un Nouveau Projet Django**

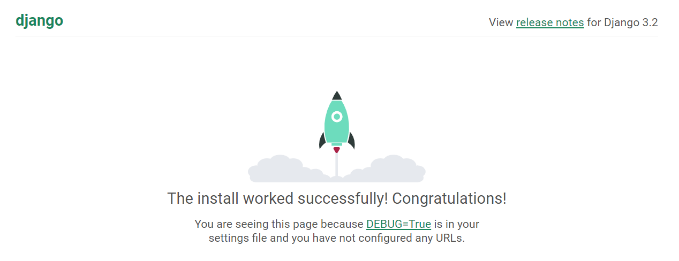
1. Lancer PyCharm CE et créer un nouveau projet appelé django-app avec un nouvel environnement virtuel nommé django-app.
2. Dans le Terminal de PyCharm, Installer le package django en utilisant la commande : **pip install django.**
3. Dans le même Terminal, exécuter la commande : **django-admin startproject myapp**

Ceci va créer un nouveau dossier nommé **myapp**.

1. Dans PyCharm, Sélectionner ***File->Settings***, puis sélectionner **Porject: django-app** puis **Project Structure.**
2. Supprimer la racine existante (django-app) de la liste puis Sélectionner le dossier myapp en tant que nouvelle racine du Projet en utilisant "**add Content Root"**.
3. Fermer tous les terminaux ouverts de PyCharm puis ouvrir un nouveau Terminal.
4. Aller Dans le menu **Run** puis choisir "**Edit configurations**"
5. Editer la configuration "**main**" en appliquant les changements suivants :

* Pointer le chemin du script vers le fichier **manage.py**
* Dans la case "**parameters**" mettre : **runserver**

1. Valider les changements puis lancer l'application en utilisant soit le bouton start de la barre d'outils soit la combinaison **MAJ+F10** (**MAJ+F9** pour le débogage).
2. Lancer un navigateur de votre choix et puis visiter l'URL : <http://127.0.0.1:8000/>

Que constatez-vous ?

**from django.contrib import admin**

**from django.urls import path**

**from . import views**

**urlpatterns = [**

**path('admin/', admin.site.urls),**

**path('', views.hello),**

**]**

1. Ajouter un fichier nommé **views.py** et y insérer le code suivant :

**from django.http.response import HttpResponse**

**def hello(request):**

**return HttpResponse("Hello World !!")**

1. Modifier le code dans le fichier **urls.py** pour inclure la fonction **hello()**
2. Enregistrer les modifications puis revisiter l'URL : <http://127.0.0.1>:8000

Que constatez-vous ?

1. Générer une nouvelle application dans le projet myapp en utilisant la commande :

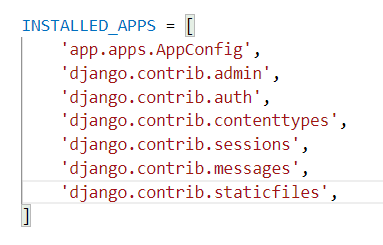
**python manage.py startapp app** dans le Terminal de PyCharm. Que se passe-t-il ?

# **Le Système ORM**

## **Les Modèles et Relations**

1. Définir un modèle nommé **Product** dans l'application **app** contenant les attributs suivants :

* **name**: string (CharField, max\_length=200)
* **code**: string (CharField, max\_length=10)
* **description**: texte (TextField)
* **price**: entier (IntegerField)

1. Avant de pouvoir utiliser l'application app dans le projet, il faut l'ajouter dans la liste des applications installés (**INSTALLED\_APPS**) dans le fichier **settings.py** (voir image).
2. Lancer la préparation des migrations en utilisant la commande :

**python manage.py makemigrations**

1. Appliquer les migrations sur la base de Données en utilisant la commande :

**python manage.py migrate**

1. Ouvrir une console Django en utilisant la commande : **python manage.py shell**
2. Effectuer les opérations suivantes :
3. Importer le modèle **Product** depuis le module **models.**
4. Insérer les 5 produits suivants :

|  |  |  |  |
| --- | --- | --- | --- |
| Nom | Code | Description | Prix |
| HP printer | HP-101 | Imprimante Laser | 3200 |
| Dell XPS M1630 | D-M1630-S2 | Dell Laptop i7 | 12500 |
| Logitech Pad | LD-230 | Trackpad Logitech | 2000 |
| ADATA 16GB | UKEY-40 | Clé USB 16GB | 100 |
| HP Tracer | T-ZE-230 | Traceur HP | 25000 |

1. Rechercher les produits dont le prix est inférieur à 3000
2. Afficher le Produit avec l’ID : 3
3. Modifier le produit avec l'ID 5 en mettant à jour le prix à 45000
4. Rechercher les produits contenant "HP" dans leurs noms
5. Rechercher les produits dont le code contient "30" et dont le prix est supérieur à 1000.
6. Insérer un produit avec des données de votre choix sans donner un prix. Que se passe-t-il ?
7. Pour remédier à ce problème, ajouter les attributs suivants dans la définition du champs prix :

**(blank=True, null=True)**. Refaire l'insertion et constatez le changement.

1. En utilisant ***un seul QuerySet***, effectuer les opérations suivantes :

* Rechercher les produits contenant "e" dans leurs noms
* Eliminer les produits dont le prix est supérieur à 5000
* Garder les produits contenant "HP" dans leurs descriptions.

1. Définir un deuxième modèle nommé **Characteristic** contenant les attributs suivants :

* **nom**: string (CharField, max\_length=200)
* **valeur**: string (CharField, max\_length=200)

1. Modifier le code du modèle Product pour créer une relation de type One-to-Many entre **Product** et **Characteristic** en utilisant une relation **ForeignKey** (Un produit possède zéro ou plusieurs caractéristiques), avec une suppression en cascade.
2. Ouvrir une Console Django, puis effectuer les opérations suivantes :
3. Importer les modèles **Product** et **Characteristic** depuis le module **models**
4. Insérer les caractéristiques suivantes :

|  |  |  |
| --- | --- | --- |
| Product ID | Nom Caractéristique | Valeur Caractéristique |
| 3 | Poids | 200g |
| Voltage | 5v |
| 4 | Capacite | 16gb |
| Plug and Play | Compatible |
| 5 | Modèle | M1 |
| Poids | 50kg |

1. Rechercher la caractéristique avec ID : 4. Afficher ses détails et le produit correspondant
2. Rechercher le produit N° 5 et afficher ses caractéristiques.
3. Rechercher les caractéristiques dont le nom de produit contient "16".
4. Afficher les produits dont une de ses caractéristiques contient la valeur "M1"
5. Créer un nouveau modèle nommé Tag possédant les attributs suivants :

* **code**: string (CharField, max\_length=20)
* **value**: string (CharField, max\_length=20)

1. Ajouter une relation de type **Many-To-Many** entre **Tag** et **Product** en ajoutant un champ **tags** de type **ManyToManyField** au model **Product**
2. Relacer la création et application des migrations pour que les changements prennent effets.
3. Dans un Shell Django, importer les modèles Product et **Tag** puis :
4. Créer 4 Tags avec des valeurs de votre choix.
5. Récupérer le Produit N° 4 et lui assigner les Tags N° 2, 4
6. Récupérer le Produit N° 3 et lui assigner le Tag N° 2
7. Sauvegarder les changements sur les produits en appelant la méthode save() sur chaque instance.
8. Récupérer le produit N° 4 et afficher les Tags associés
9. Récupérer le Tag N° 2 et afficher les produits associés

## **L'agrégation**

1. Récupérer le nombre de produits dans la base de données en utilisant la méthode **count()**
2. Récupérer la moyenne des prix, le prix maximal et le prix minimal.
3. Récupérer La différence entre le prix maximal et le prix moyen
4. Récupérer tous les produits en ajoutant un champ "**tags\_count**" qui contient le nombre de tags pour chaque produit. Afficher les valeurs de "**tags\_count**" en utilisant une boucle **for**.
5. Afficher tous les produits en les triant par ordre décroissant des prix.

## **L'héritage entre les modèles**

1. On souhaite ajouter un modèle pour la gestion des produits électroniques (PDF, Music) en se basant sur le modèle Product.
2. Créer un deuxième modèle **OnlineProduct** qui hérite de Product
3. Ajouter dans **OnlineProduct** les attributs suivants :

* **product\_url**: URL (URLField)
* **is\_available**: Boolean (BooleanField)

1. Relancer la création des migrations et Appliquer-les.
2. Tester le nouveau Modèle en effectuant des opérations d'insertion, de recherche et modification.
3. On souhaite intégrer des marqueurs temporels dans les modèles Product et Characteristic pour enregistrer la date de création (**created\_at**) et date de dernière mise à jour (**updated\_at**).
4. Créer un Modèle nommé **TimeStampedModel** contenant les attributs :

* **created\_at**: datetime (DateTimeField, avec **auto\_now\_add=True** et **null=True**)
* **updated\_at**: datetime (DateTimeField, avec **auto\_now=True** et **null=True**)

1. Rendre la classe **TimeStampedModel** abstraite en spécifiant **abstract=True** dans la définition de **class Meta**
2. Modifier les modèles Product et Characteristic pour hériter de la classe **TimeStampedModel**
3. Relancer la création des migrations et Appliquer-les (si le Framework demande une valeur par defaut, donnez **datetime.now()** ou **None**).
4. Créer un nouveau Produit p avec des valeurs de votre choix. Afficher sa date de création et de dernière mise à jour. Que constatez-vous ?
5. Récupérer le Produit p à nouveau depuis la base de données puis modifier son prix à 1500. Sauvegarder la modification en appelant la méthode save().

Afficher la date de dernière mise à jour et apprécier le changement.

# **Exercice applicatif**

On souhaite développer une site web sous forme d'un blog dynamique qui présente la structure suivante :

1. Création des modèles

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | Attributs | Type | Obligatoire | Propriétés |
| Post | **title** | **string** | **yes** | **max\_length=200** |
|  | **slug** | **slug** | **no** |  |
|  | **teaser** | **text** | **no** |  |
|  | **body** | **text** | **yes** |  |
|  | **thumbnail** | **image field** | **no** |  |
|  | **main image** | **image field** | **no** |  |
|  | **author** | **user** | **yes** | **on\_delete=cascade** |
|  | **created\_by** | **user** | **yes** | **on\_delete=set\_null** |
|  | **created\_at** | **datetime** | **no** | **auto\_now\_add** |
|  | **updated\_at** | **datetime** | **no** | **auto\_now** |
|  | **updated\_by** | **user** | **no** | **on\_delete=set\_null** |
|  | **category** | **Category** | **yes** | **On\_delete=cascade** |
| Comment | **author** | **user** | **yes** | **on\_delete=cascade** |
|  | **title** | **string** | **yes** |  |
|  | **content** | **text** | **yes** |  |
| Tag | **code** | **string** | **no** | **max\_length=50** |
|  | **value** | **string** | **yes** | **max\_length=50** |
| Category | **title** | **string** | **yes** | **max\_length=200** |
|  | **description** | **text** | **no** |  |
|  | **parent** | **category** | **no** | **on\_delete=cascade** |

* Relations ManyToMany :
* **Post** a une relation nommée **related\_posts** avec le model **Post**
* **Post** a une relation nommée tags avec le modèle **Tag.**

1. Générer les migrations et appliquer-les.
2. Tester l'insertion de quelques modèles et