

TP2 – Les modèles

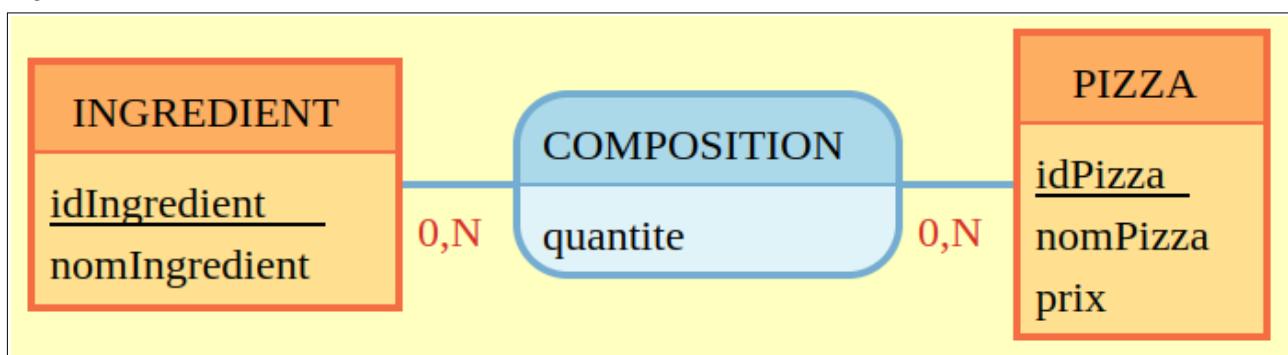
Introduction, Contexte et Objectifs

Django permet de créer à la main les modèles (équivalent des classes). Ensuite, par le mécanisme des migrations, les tables de données sont automatiquement ajoutées à la base de données.

Nous avons donc besoin du schéma relationnel de l'application applipizza. Pour le moment nous nous contenterons d'un MCD simple, avec deux entités INGREDIENT et PIZZA, et une association COMPOSITION. Un ingrédient peut entrer dans la composition de plusieurs pizzas, et une pizza est composée d'un certain nombre d'ingrédients.

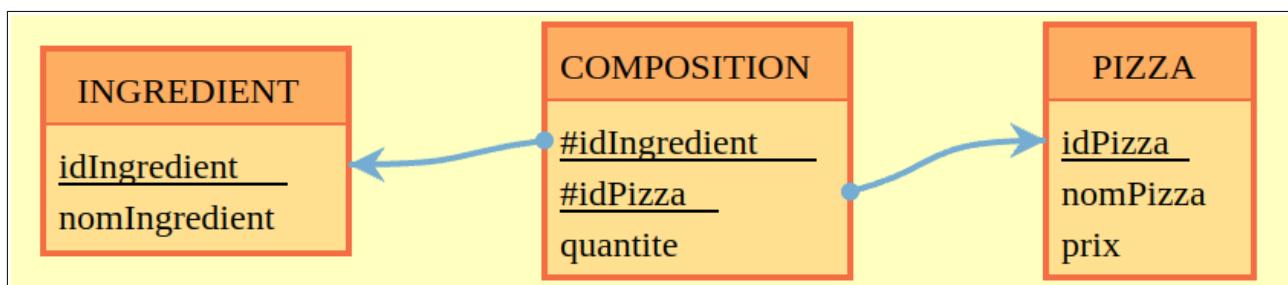
Plus tard, nous modifierons le modèle de Pizza pour lui donner un attribut supplémentaire (une image). Ce sera un complément intéressant.

MCD



Le schéma relationnel donne en théorie les trois tables suivantes, avec deux clés étrangères. La table COMPOSITION aurait pour clé primaire le couple (idIngredient, idPizza).

SR

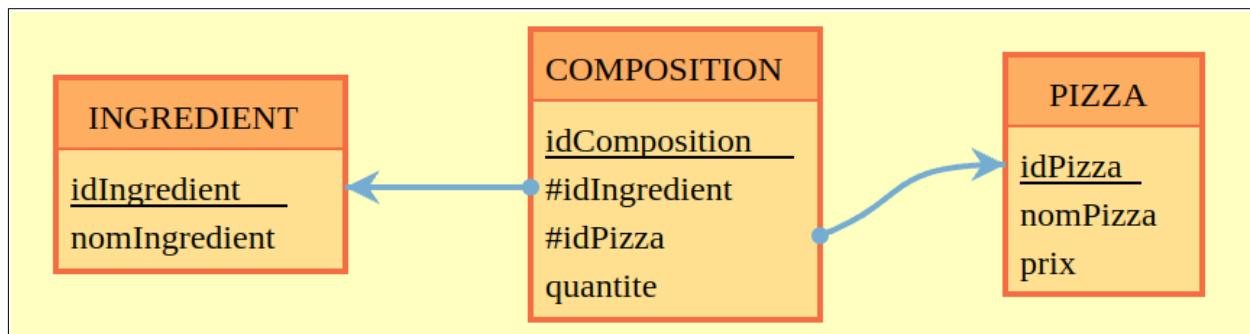


Avec la base de données sqlite locale, l'adaptation de COMPOSITION va faire apparaître :

- une clé primaire `idComposition`,
- une contrainte d'unicité pour le couple de clés étrangères (`#idIngredient`, `#idPizza`).

Ce n'est pas vraiment différent. Le schéma pour notre projet sera donc plutôt le suivant, qui est équivalent au précédent :

SR



Les modèles avec Django

Une fois que le schéma relationnel est établi, nous allons confier à django le soin de créer les tables de données, en lui indiquant seulement quelles classes nous souhaitons construire.

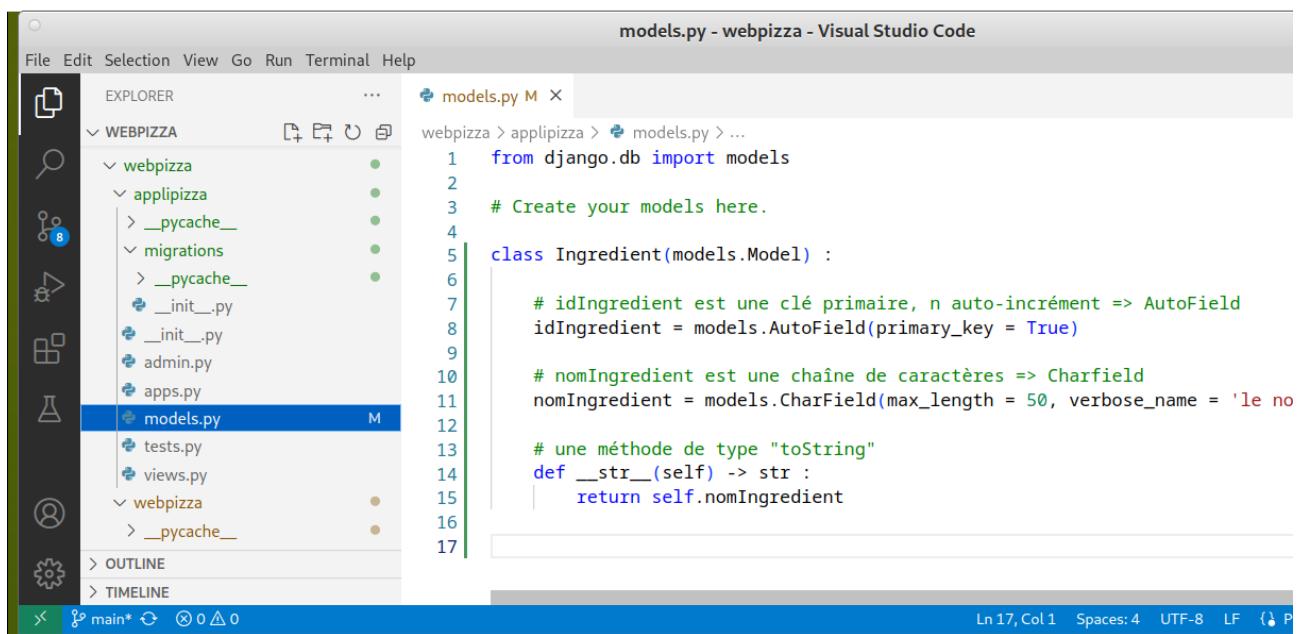
Ceci permet une cohérence forte qui pouvait être mise en défaut dans le système classique PHP / MySQL pur, puisque la création de tables n'était pas automatisée à partir des classes PHP construites.

Notre première étape est donc de construire les classes python, et ceci se fait dans le fichier `models.py` du projet.

1. Ajout des modèles dans `models.py`

- changez le fichier `models.py` du répertoire `applipizza` pour qu'il définisse la classe `Ingredient`. Cette classe héritera de la classe `Model` de la bibliothèque `models`. On indique les champs `idIngredient` et `nomIngredient`, et on définit la méthode `__str__`, équivalent de `toString()` en java.

Le code suivant est donné en modèle :



The screenshot shows the Visual Studio Code interface with the following details:

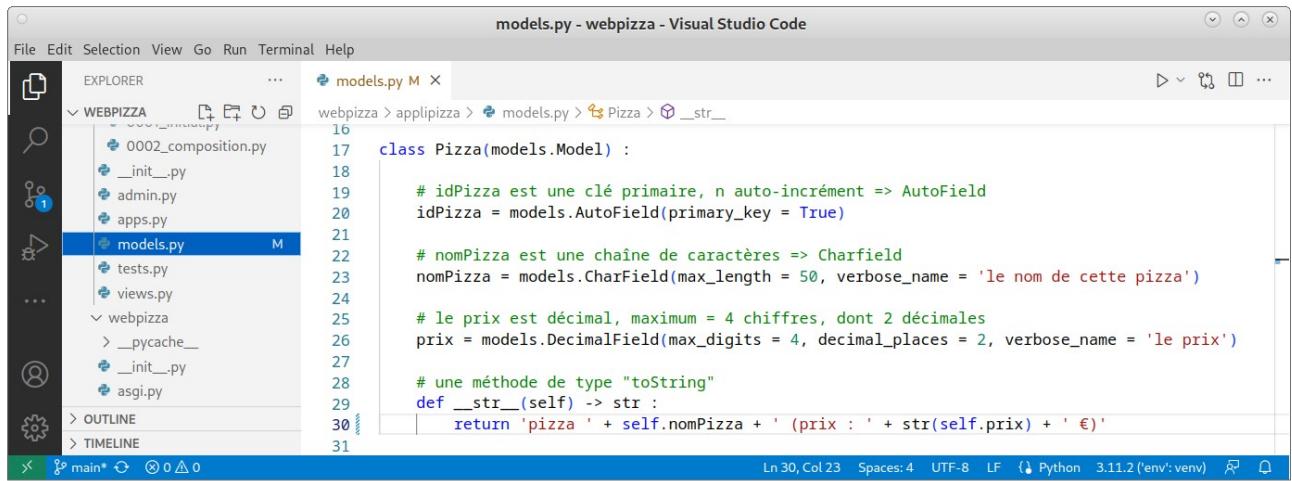
- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Explorer:** Shows the project structure under 'WEBPIZZA':
 - webpizza
 - applipizza
 - migrations
 - __pycache__
 - __init__.py
 - admin.py
 - apps.py
 - models.py (selected)
 - tests.py
 - views.py
 - webpizza
 - __pycache__
 - OUTLINE
 - TIMELINE
- Editor:** The 'models.py' file content is displayed:


```

1  from django.db import models
2
3  # Create your models here.
4
5  class Ingredient(models.Model) :
6
7      # idIngredient est une clé primaire, n auto-incrément => AutoField
8      idIngredient = models.AutoField(primary_key = True)
9
10     # nomIngredient est une chaîne de caractères => Charfield
11     nomIngredient = models.CharField(max_length = 50, verbose_name = 'le nom')
12
13     # une méthode de type "toString"
14     def __str__(self) -> str :
15         return self.nomIngredient
16
17
      
```
- Bottom Status Bar:** Ln 17, Col 1, Spaces: 4, UTF-8, LF, {P}

voyez cette page pour plus de documentation :
<https://docs.djangoproject.com/fr/4.1/topics/db/models/>

- b. Créez de même la classe Pizza (dans le même fichier, à la suite de la classe Ingredient). Vous pourrez utiliser le type DecimalField pour le prix.



The screenshot shows the Visual Studio Code interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Explorer:** Shows the project structure under 'WEBPIZZA':
 - 0002_composition.py
 - __init__.py
 - admin.py
 - apps.py
 - models.py (selected)
 - tests.py
 - views.py
 - webpizza
 - __pycache__
 - __init__.py
 - asgi.py
- Editor:** The 'models.py' file content is displayed:


```

16
17     class Pizza(models.Model) :
18
19         # idPizza est une clé primaire, n auto-incrément => AutoField
20         idPizza = models.AutoField(primary_key = True)
21
22         # nomPizza est une chaîne de caractères => Charfield
23         nomPizza = models.CharField(max_length = 50, verbose_name = 'le nom de cette pizza')
24
25         # le prix est décimal, maximum = 4 chiffres, dont 2 décimales
26         prix = models.DecimalField(max_digits = 4, decimal_places = 2, verbose_name = 'le prix')
27
28         # une méthode de type "toString"
29         def __str__(self) -> str :
30             return 'pizza ' + self.nomPizza + ' (prix : ' + str(self.prix) + ' €)'
31
      
```
- Bottom Status Bar:** Ln 30, Col 23, Spaces: 4, UTF-8, LF, {Python 3.11.2 ('env': venv)}

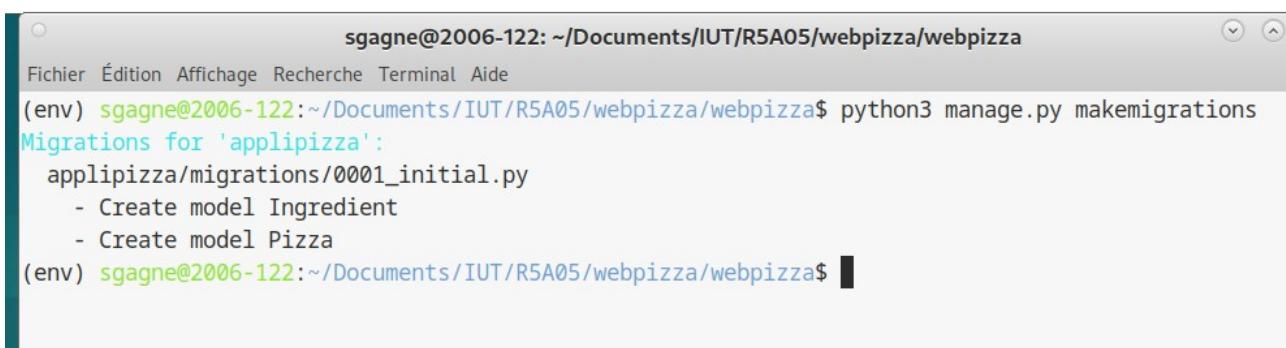
2. Migrez les modèles vers la base de données

a. Dans le répertoire webpizza (là où il y a le fichier qui gère les commandes, manage.py), lancez la commande

python3 manage.py makemigrations

qui prépare l'adaptation de la base de données en fonction des nouveaux modèles.

Cela produira un affichage comme :



```
sgagne@2006-122: ~/Documents/IUT/R5A05/webpizza/webpizza
Fichier Édition Affichage Recherche Terminal Aide
(env) sgagne@2006-122:~/Documents/IUT/R5A05/webpizza/webpizza$ python3 manage.py makemigrations
Migrations for 'applipizza':
    applipizza/migrations/0001_initial.py
        - Create model Ingredient
        - Create model Pizza
(env) sgagne@2006-122:~/Documents/IUT/R5A05/webpizza/webpizza$
```

Cet affichage montre les deux modèles créés. Un fichier de migration nommé 0001_initial.py a aussi été écrit, dans lequel vous trouverez les instructions qui permettront de créer la structure de données dans la base sqlite.

b. Ensuite, validez la migration par la commande

python3 manage.py migrate



```
sgagne@2006-122: ~/Documents/IUT/R5A05/webpizza/webpizza
Fichier Édition Affichage Recherche Terminal Aide
(env) sgagne@2006-122:~/Documents/IUT/R5A05/webpizza/webpizza$ python3 manage.py makemigrations
Migrations for 'applipizza':
    applipizza/migrations/0001_initial.py
        - Create model Ingredient
        - Create model Pizza
(env) sgagne@2006-122:~/Documents/IUT/R5A05/webpizza/webpizza$ python3 manage.py migrate
Operations to perform:
  Apply all migrations: admin, applipizza, auth, contenttypes, sessions
Running migrations:
  Applying applipizza.0001_initial... OK
(env) sgagne@2006-122:~/Documents/IUT/R5A05/webpizza/webpizza$
```

- c. Constatez l'effet sur la base de données en ouvrant sqlitebrowser. Pour le moment vous ne voyez que la structure des tables Ingredient et Pizza.

Nom	
Tables (13)	
▶	applipizza_ingredient
▶	applipizza_pizza
▶	auth_group
▶	auth_group_permissions
▶	auth_permission
▶	auth_user
▶	auth_user_groups
▶	auth_user_user_permissions
▶	django_admin_log
▶	django_content_type
▶	django_migrations
▶	django_session
▶	sqlite_sequence

Pour le moment, il n'y a pas encore de données, mais vous pouvez accéder à la vue des données dans l'onglet parcourir les données :

Table :	applipizza_pizza	Actions			
	<table border="1"> <thead> <tr> <th>idPizza</th> <th>nomPizza</th> <th>prix</th> </tr> </thead> </table>	idPizza	nomPizza	prix	
idPizza	nomPizza	prix			

Table :	applipizza_ingredient	Actions		
	<table border="1"> <thead> <tr> <th>idIngredient</th> <th>nomIngredient</th> </tr> </thead> </table>	idIngredient	nomIngredient	
idIngredient	nomIngredient			

3. Enregistrer, consulter des données

- a. Vous pouvez enregistrer ou consulter des données par l'interface, c'est le plus simple, grâce au bouton d'insertion d'un nouvel enregistrement.

N'oubliez pas d'enregistrer les modifications de la table (si vous ne le faites pas, le navigateur vous demandera de sauvegarder en quittant).

Enregistrez 3 nouveaux ingrédients : dans la capture suivante, champignons, anchois et jambon.

Table : applipizza_ingredient

1	1	champignons
2	2	anchois
3	3	jambon

b. Vous pouvez aussi ajouter des enregistrements en ligne de commande grâce au shell python :

- **python3 manage.py shell** pour entrer dans le shell python (**CTRL + D** pour quitter)
- ```
from applipizza.models import Ingredient
oeuf = Ingredient()
oeuf.nomIngredient = 'œuf'
oeuf.save()
```

```
sgagne@2006-122: ~/Documents/IUT/R5A05/webpizza/webpizza
Fichier Édition Affichage Recherche Terminal Aide
(env) sgagne@2006-122:~/Documents/IUT/R5A05/webpizza/webpizza$ python3 manage.py shell
Python 3.11.2 (main, Mar 13 2023, 12:18:29) [GCC 12.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>> from applipizza.models import Ingredient
>>> oeuf = Ingredient()
>>> oeuf.nomIngredient = 'œuf'
>>> oeuf.save()
>>>
```

Ouvrez de nouveau sqlitebrowser pour constater qu'il y a bien une nouvelle entrée, puis quittez sqlitebrowser :

Table : applipizza\_ingredient

| 1 | 1 | champignons |
|---|---|-------------|
| 2 | 2 | anchois     |
| 3 | 3 | jambon      |
| 4 | 4 | œuf         |
|   |   |             |

c. Vous pouvez consulter les données dans la console python (accessible par la commande **python3 manage.py shell**).

Ci-dessous, on a récupéré tous les ingrédients, puis parcouru la liste correspondante et appelé la méthode print sur chaque ingrédient (ceci a implicitement appelé la méthode **\_\_str\_\_** redéfinie précédemment).



```
sgagne@2006-122: ~/Documents/IUT/R5A05/webpizza/webpizza
Fichier Édition Affichage Recherche Terminal Aide
(env) sgagne@2006-122:~/Documents/IUT/R5A05/webpizza/webpizza$ python3 manage.py shell
Python 3.11.2 (main, Mar 13 2023, 12:18:29) [GCC 12.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>> from applipizza.models import Ingredient
>>> oeuf = Ingredient()
>>> oeuf.nomIngredient = 'œuf'
>>> oeuf.save()
>>> lesIngredients = Ingredient.objects.all()
>>> len(lesIngredients)
4
>>> for ingredient in lesIngredients :
... print(ingredient)
...
champignons
anchois
jambon
œuf
>>>
```

Vous pouvez donc agir sur les données ou les consulter par le shell python ou par l'interface.

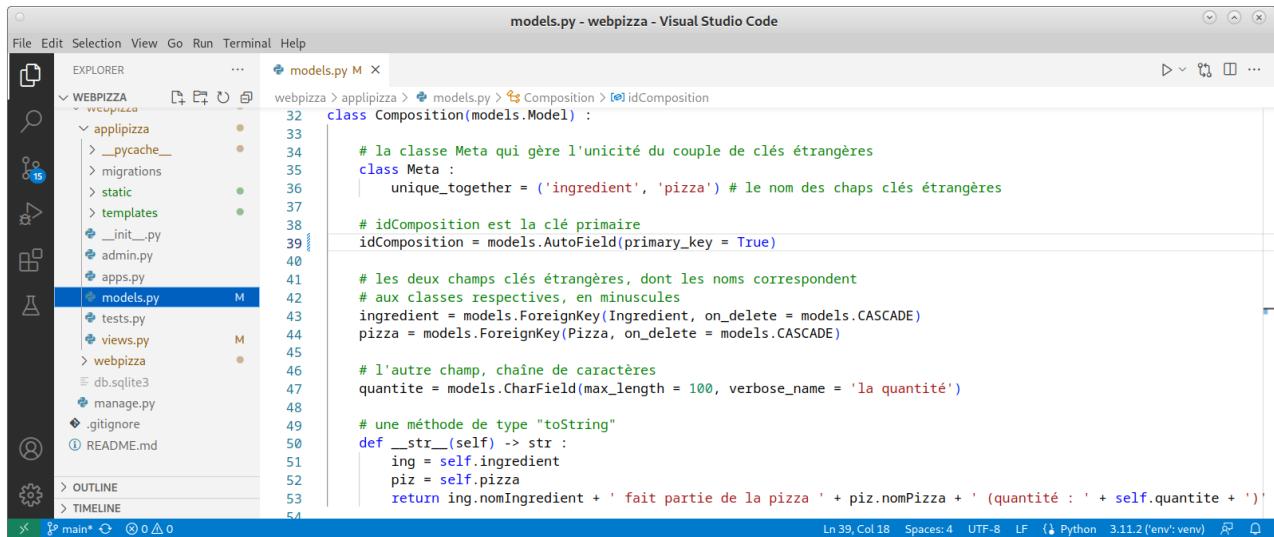
Remarques : en console vous devez comme dans l'éditeur de texte inclure des tabulations (boucle for) sinon l'instruction python n'est pas comprise. En python, les blocs sont délimités par la hiérarchie des indentations. La fonction python **len** renvoie la longueur de l'ensemble passé en paramètre : ici, 4 ingrédients en tout.

N'oubliez pas de mettre votre dépôt distant à jour.

#### 4. Un modèle avec clés étrangères

La classe `Composition` aura donc une clé primaire `idComposition`, et une contrainte d'unicité sur le couple (`idIngredient`, `idPizza`).

Voici le code correspondant. Vous noterez que la contrainte d'unicité est gérée dans une classe interne « `Meta` » qui gère tout ce qui est hors déclaration de champs.



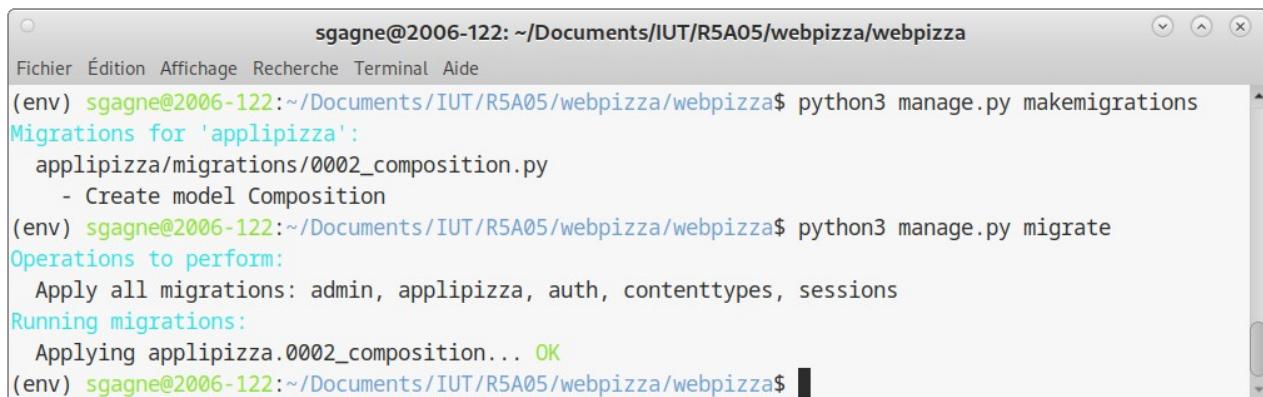
```

models.py - webpizza - Visual Studio Code
File Edit Selection View Go Run Terminal Help
models.py M ...
WEBPIZZA > webpizza > models.py > Composition > idComposition
32 class Composition(models.Model) :
33
34 # la classe Meta qui gère l'unicité du couple de clés étrangères
35 class Meta :
36 unique_together = ('ingredient', 'pizza') # le nom des champs clés étrangères
37
38 # idComposition est la clé primaire
39 idComposition = models.AutoField(primary_key = True)
40
41 # les deux champs clés étrangères, dont les noms correspondent
42 # aux classes respectives, en minuscules
43 ingredient = models.ForeignKey(Ingredient, on_delete = models.CASCADE)
44 pizza = models.ForeignKey(Pizza, on_delete = models.CASCADE)
45
46 # l'autre champ, chaîne de caractères
47 quantite = models.CharField(max_length = 100, verbose_name = 'la quantité')
48
49 # une méthode de type "toString"
50 def __str__(self) -> str :
51 ing = self.ingredient
52 piz = self.pizza
53 return ing.nomIngredient + ' fait partie de la pizza ' + piz.nomPizza + ' (quantité : ' + self.quantite + ')'
54
Ln 39, Col 18 Spaces:4 UTF-8 LF { Python 3.11.2 ('env': venv) ⌂ ⌂

```

a. créez la classe dans `models.py`

b. mettez en place la migration.



```

sgagne@2006-122: ~/Documents/IUT/R5A05/webpizza/webpizza
Fichier Édition Affichage Recherche Terminal Aide
(env) sgagne@2006-122:~/Documents/IUT/R5A05/webpizza/webpizza$ python3 manage.py makemigrations
Migrations for 'applipizza':
 applipizza/migrations/0002_composition.py
 - Create model Composition
(env) sgagne@2006-122:~/Documents/IUT/R5A05/webpizza/webpizza$ python3 manage.py migrate
Operations to perform:
 Apply all migrations: admin, applipizza, auth, contenttypes, sessions
Running migrations:
 Applying applipizza.0002_composition... OK
(env) sgagne@2006-122:~/Documents/IUT/R5A05/webpizza/webpizza$

```

c. enregistrez des données via sqlitebrowser, pour avoir un jeu de données intéressant (vous devrez actualiser la base pour voir la nouvelle table composition) :

| Table : applipizza_ingredient |                   |
|-------------------------------|-------------------|
| idIngredient                  | nomIngredient     |
| 1                             | champignons       |
| 2                             | anchois           |
| 3                             | jambon            |
| 4                             | œuf               |
| 5                             | gorgonzola        |
| 6                             | fromage de chèvre |
| 7                             | parmesan          |
| 8                             | mozzarella        |
| 9                             | crème fraîche     |
| 10                            | oignon            |
| 11                            | basilic           |
| 12                            | origan            |
| 13                            | coulis de tomate  |
| 14                            | tomate            |
| 15                            | pomme de terre    |
| 16                            | miel              |
| 17                            | roquette          |
| 18                            | saumon            |
| 19                            | thon              |
| 20                            | lardons           |
| 21                            | viande hachée     |

| Table : applipizza_pizza |                 |      |
|--------------------------|-----------------|------|
| idPizza                  | nomPizza        | prix |
| 1                        | quatre fromages | 15.5 |
| 2                        | napolitaine     | 14.8 |

| Table : applipizza_composition |                      |               |          |
|--------------------------------|----------------------|---------------|----------|
| idComposition                  | quantite             | ingredient_id | pizza_id |
| 1                              | 75 grammes           | 5             | 1        |
| 2                              | 75 grammes           | 6             | 1        |
| 3                              | 30 grammes           | 7             | 1        |
| 4                              | 1 boule              | 8             | 1        |
| 5                              | 1 cuillérée à soupe  | 9             | 1        |
| 6                              | 1 oignon haché       | 10            | 1        |
| 7                              | 4 cuillérées à soupe | 13            | 1        |
| 8                              | 200 grammes          | 14            | 2        |
| 9                              | 200 grammes          | 8             | 2        |
| 10                             | à volonté            | 11            | 2        |
| 11                             | 200 grammes          | 21            | 2        |
| 12                             | 2 cuillérées à soupe | 9             | 2        |

N'oubliez pas d'enregistrer les modifications de la base, et n'oubliez pas de mettre à jour le dépôt distant.

d. Entraînez-vous aussi à consulter les tables dans le shell python pour vous habituer à la manœuvre.

```
sgagne@2006-122: ~/Documents/IUT/R5A05/webpizza/webpizza
Fichier Édition Affichage Recherche Terminal Aide
(env) sgagne@2006-122:~/Documents/IUT/R5A05/webpizza/webpizza$ python manage.py shell
Python 3.11.2 (main, Mar 13 2023, 12:18:29) [GCC 12.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>> from applipizza.models import Ingredient, Pizza, Composition
>>> for pizza in Pizza.objects.all() :
... print(pizza)
...
pizza quatre fromages (prix : 15.50 €)
pizza napolitaine (prix : 14.80 €)
>>> for ligne in Composition.objects.all() :
... print(ligne)
...
gorgonzola fait partie de la pizza quatre fromages (quantité : 75 grammes)
fromage de chèvre fait partie de la pizza quatre fromages (quantité : 75 grammes)
parmesan fait partie de la pizza quatre fromages (quantité : 30 grammes)
mozzarella fait partie de la pizza quatre fromages (quantité : 1 boule)
crème fraîche fait partie de la pizza quatre fromages (quantité : 1 cuillérée à soupe)
oignon fait partie de la pizza quatre fromages (quantité : 1 oignon haché)
coulis de tomate fait partie de la pizza quatre fromages (quantité : 4 cuillérées à soupe)
tomate fait partie de la pizza napolitaine (quantité : 200 grammes)
mozzarella fait partie de la pizza napolitaine (quantité : 200 grammes)
basilic fait partie de la pizza napolitaine (quantité : à volonté)
viande hâlée fait partie de la pizza napolitaine (quantité : 200 grammes)
crème fraîche fait partie de la pizza napolitaine (quantité : 2 cuillérées à soupe)
>>> █
```