

# TP3 – Les premières vues

## Contexte et Objectifs

Pour le moment, nous pouvons afficher en console python les pizzas, les ingrédients et les compositions. Nous avons défini les classes, nous les avons fait migrer pour créer les tables de données.

Il reste maintenant à créer des vues pour que ces listes de pizzas et d'ingrédients puissent apparaître sur le navigateur. Pour ces premières vues, l'objectif est juste de comprendre le mécanisme qui agit pour les créer et les afficher.

Une remarque : contrairement à ce qu'on pourrait croire, le fichier `applipizza/views.py` ne correspond pas à notre idée des vues (des fichiers html classiques). Au contraire, ce fichier s'assimile plutôt à des fonctions contrôleur.

De manière générale, la correspondance de vocabulaire est environ celle-ci :

PHP classique	Python django
MODEL	MODEL
VIEW	TEMPLATE
CONTROLLER	VIEW

## La vue des pizzas

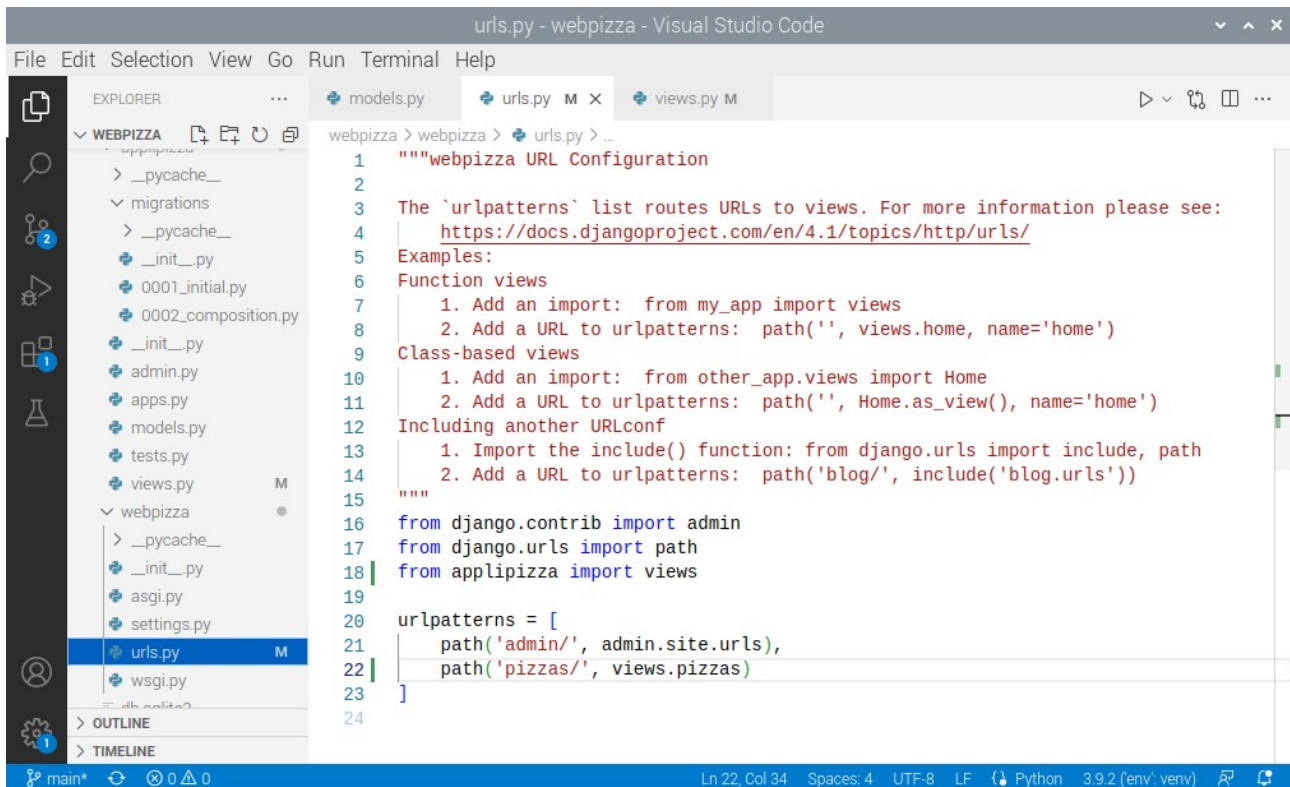
Pour arriver à l'affichage des pizzas, il y a plusieurs étapes :

1. L'url dans le fichier `urls.py`

a. dans le fichier `urls.py`, ajoutez la ligne d'import  
`from applipizza import views`

Cette ligne permet de construire des urls qui iront chercher des fonctions présentes dans `applipizza/views.py`

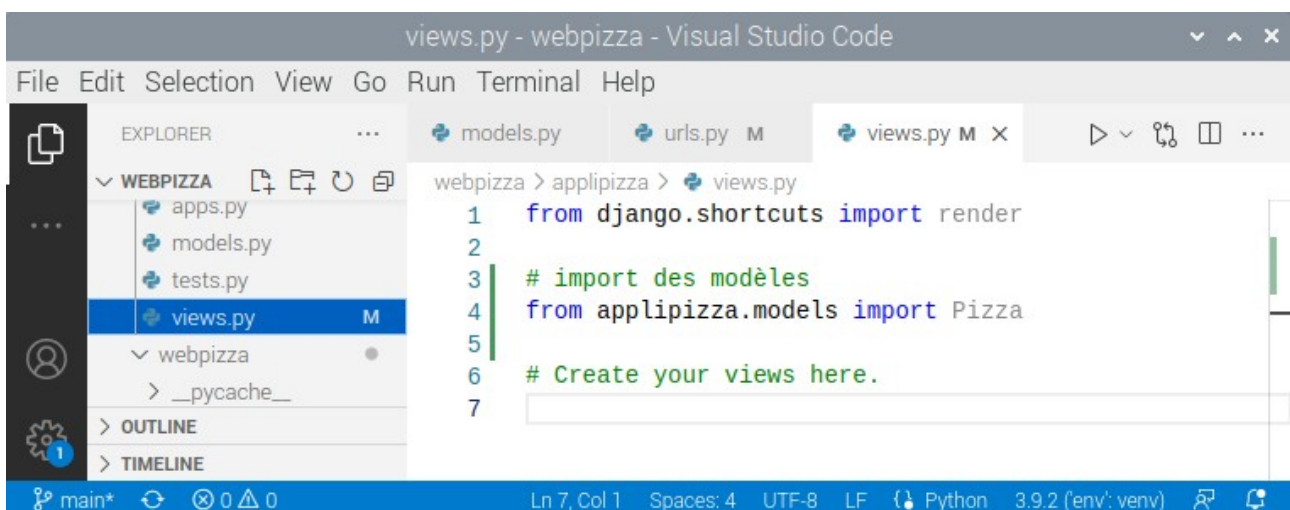
- b. dans la liste urlpatterns, ajoutez le path suivant  
`path('pizzas/', views.pizzas)`



```
1 """webpizza URL Configuration
2
3 The `urlpatterns` list routes URLs to views. For more information please see:
4     https://docs.djangoproject.com/en/4.1/topics/http/urls/
5 Examples:
6 Function views
7     1. Add an import: from my_app import views
8     2. Add a URL to urlpatterns: path('', views.home, name='home')
9 Class-based views
10    1. Add an import: from other_app.views import Home
11    2. Add a URL to urlpatterns: path('', Home.as_view(), name='home')
12 Including another URLconf
13    1. Import the include() function: from django.urls import include, path
14    2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))
15 """
16 from django.contrib import admin
17 from django.urls import path
18 from applipizza import views
19
20 urlpatterns = [
21     path('admin/', admin.site.urls),
22     path('pizzas/', views.pizzas)
23 ]
24
```

## 2. La fonction pizzas dans views.py (assimilable à contrôleur)

- a. dans le fichier applipizza/views.py, réalisez l'import du modèle Pizza :

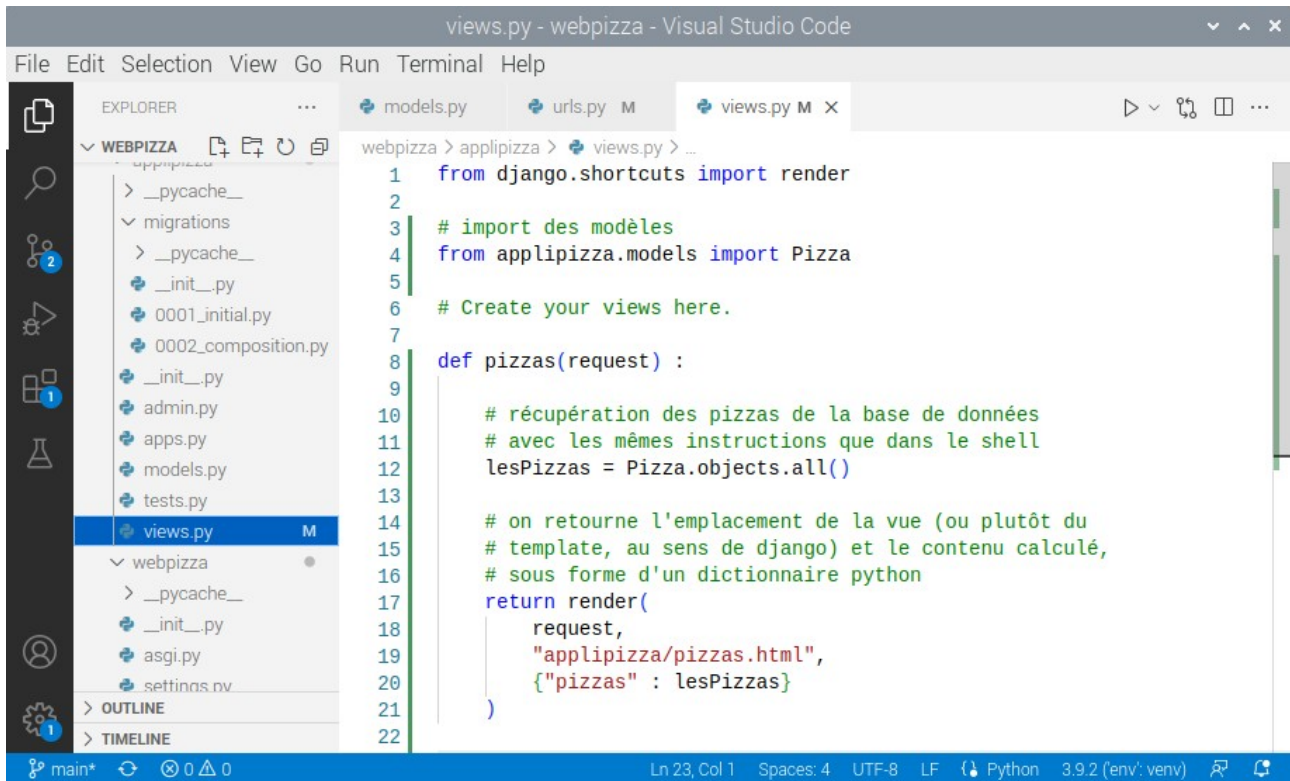


```
1 from django.shortcuts import render
2
3 # import des modèles
4 from applipizza.models import Pizza
5
6 # Create your views here.
7
```

- b. dans le même fichier, on définit la fonction **pizzas**, qui prend en charge un paramètre `request` de type `HttpRequest`, non utilisé ici.

Cette fonction récupère les données, et appelle le bon fichier vue (ici, un fichier `applipizza/pizzas.html`, pas encore créé).

Ceci est typique d'une méthode de contrôleur.



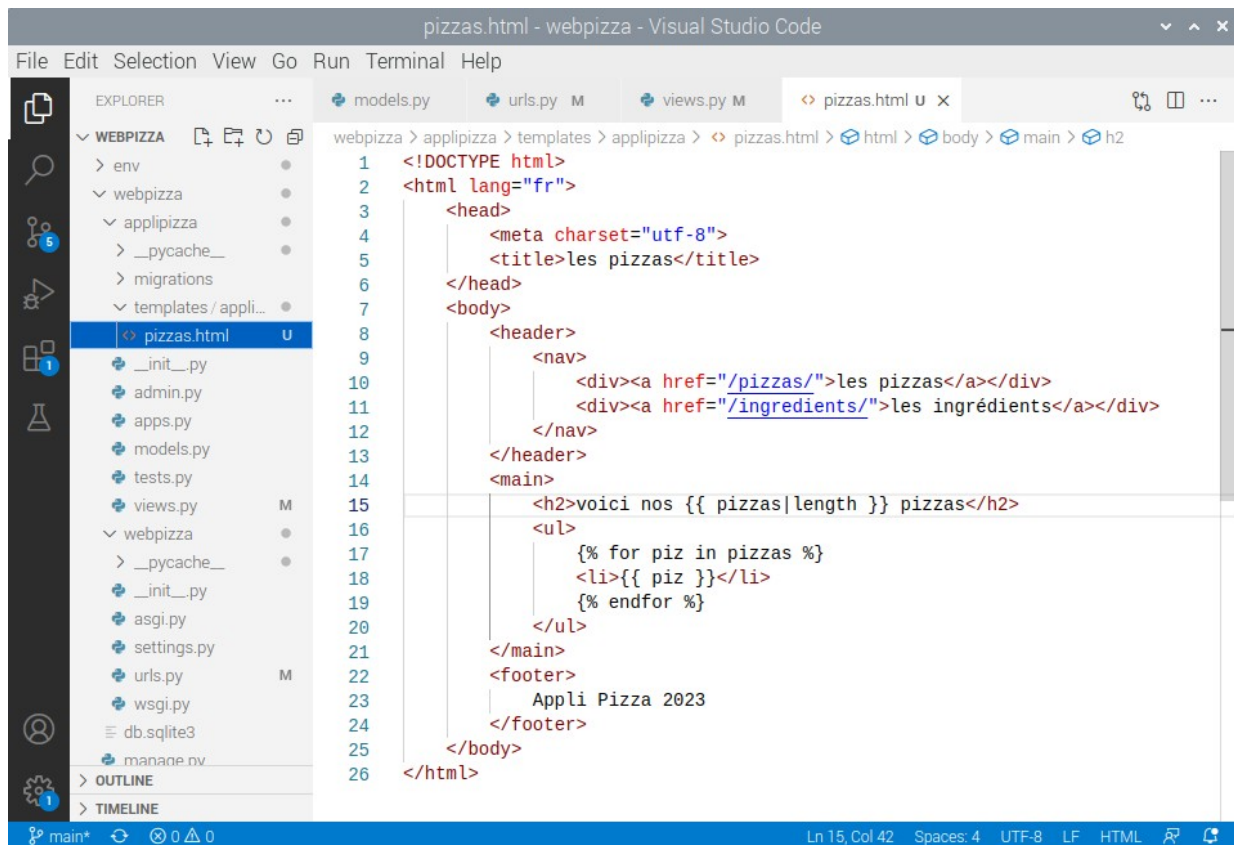
```
views.py - webpizza - Visual Studio Code
File Edit Selection View Go Run Terminal Help

EXPLORER
webpizza
  > __pycache__
  > migrations
  > __pycache__
  + __init__.py
  + 0001_initial.py
  + 0002_composition.py
  + __init__.py
  + admin.py
  + apps.py
  + models.py
  + tests.py
  + views.py M
  > webpizza
    > __pycache__
    + __init__.py
    + asgi.py
    + settings.py
  > OUTLINE
  > TIMELINE

webpizza > applipizza > views.py > ...
1 from django.shortcuts import render
2
3 # import des modèles
4 from applipizza.models import Pizza
5
6 # Create your views here.
7
8 def pizzas(request) :
9
10     # récupération des pizzas de la base de données
11     # avec les mêmes instructions que dans le shell
12     lesPizzas = Pizza.objects.all()
13
14     # on retourne l'emplacement de la vue (ou plutôt du
15     # template, au sens de django) et le contenu calculé,
16     # sous forme d'un dictionnaire python
17     return render(
18         request,
19         "applipizza/pizzas.html",
20         {"pizzas" : lesPizzas}
21     )
22
```

### 3. Le fichier `pizzas.html` (assimilable à vue)

- a. créez dans le répertoire **applipizza** un sous-répertoire **templates**, et dans **templates** un sous-répertoire **applipizza**.
- b. Créez le fichier `applipizza/templates/applipizza/pizzas.html` et donnez-lui le contenu suivant



```
1 <!DOCTYPE html>
2 <html lang="fr">
3   <head>
4     <meta charset="utf-8">
5     <title>les pizzas</title>
6   </head>
7   <body>
8     <header>
9       <nav>
10        <div><a href="/pizzas/">les pizzas</a></div>
11        <div><a href="/ingredients/">les ingrédients</a></div>
12      </nav>
13    </header>
14    <main>
15      <h2>voici nos {{ pizzas|length }} pizzas</h2>
16      <ul>
17        {% for piz in pizzas %}
18        <li>{{ piz }}</li>
19        {% endfor %}
20      </ul>
21    </main>
22    <footer>
23      Appli Pizza 2023
24    </footer>
25  </body>
26</html>
```

Remarques :

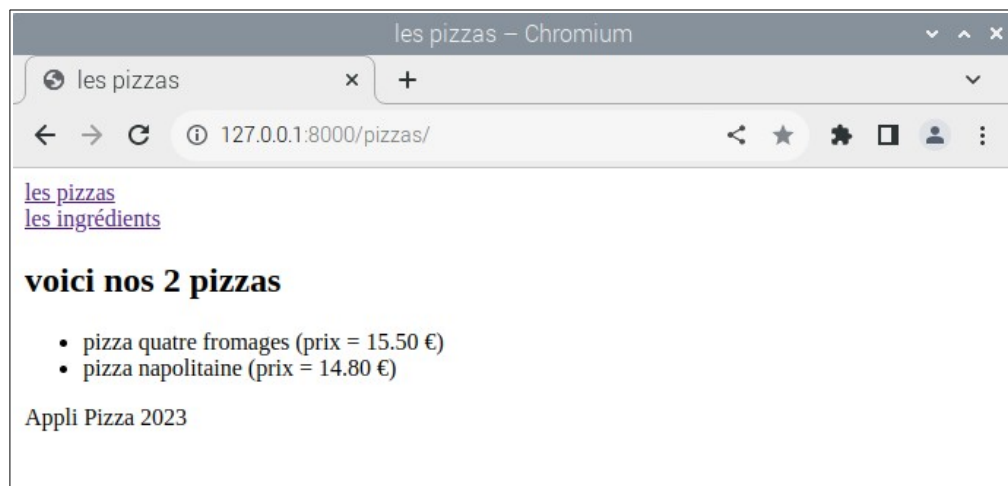
- on rencontre des « variables de gabarit », encadrées par des doubles accolades : ici, la variable pizzas qu'on utilise avec sa longueur : `{{ pizzas|length }}`.
- cette variable de gabarit est en correspondance avec la variable python `lesPizzas` du fichier `views.py`.
- on rencontre aussi des insertions de code python (un peu comme on insère du code PHP dans un fichier html par des balises `<?php ... ?>`) : ces insertions se font au moyen d'accolades et de `%`.
- ce n'est pas exactement la syntaxe python quand même : voir le « `endfor` » par exemple.

#### 4. Appel de la vue

lancez le serveur : commande **`python3 manage.py runserver`**, puis appelez l'url **`http://127.0.0.1:8000/pizzas/`**

Vous devez avoir le type de rendu de la capture suivante.

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



Le menu est à moitié opérationnel, car rien n'a été fait pour la vue des ingrédients. Il fonctionnera réellement après le travail qui suit.

## Les vues des ingrédients

Refaites le même travail que précédemment pour les ingrédients (n'oubliez pas d'importer les classes dans views.py).

A priori, la vue des compositions n'interviendra que plus tard, dans la vue de détail d'une pizza. Donc on se limite ici aux pizzas et aux ingrédients.

Le point à améliorer est clairement la genericité des fichiers vue, puisque les deux fichiers pizzas.html et ingredients.html ont la même structure.

C'est ce que nous ferons plus tard avec les **gabarits django**.

