TP3 – Les premiers templates

Introduction, Contexte et Objectifs

Pour le moment, nous pouvons afficher en console python les pizzas, les ingrédients et les compositions. Nous avons défini les modèles, 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 plutôt :

PHP classique	**Python django**
MODEL	MODEL
VIEW	TEMPLATE
CONTROLLER	VIEW

Nous parlerons donc de template pour évoquer les fichiers interprétés par le navigateur.

Enfin, nous intégrerons à notre projet le css de bootstrap.

Affichage de la liste des pizzas

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

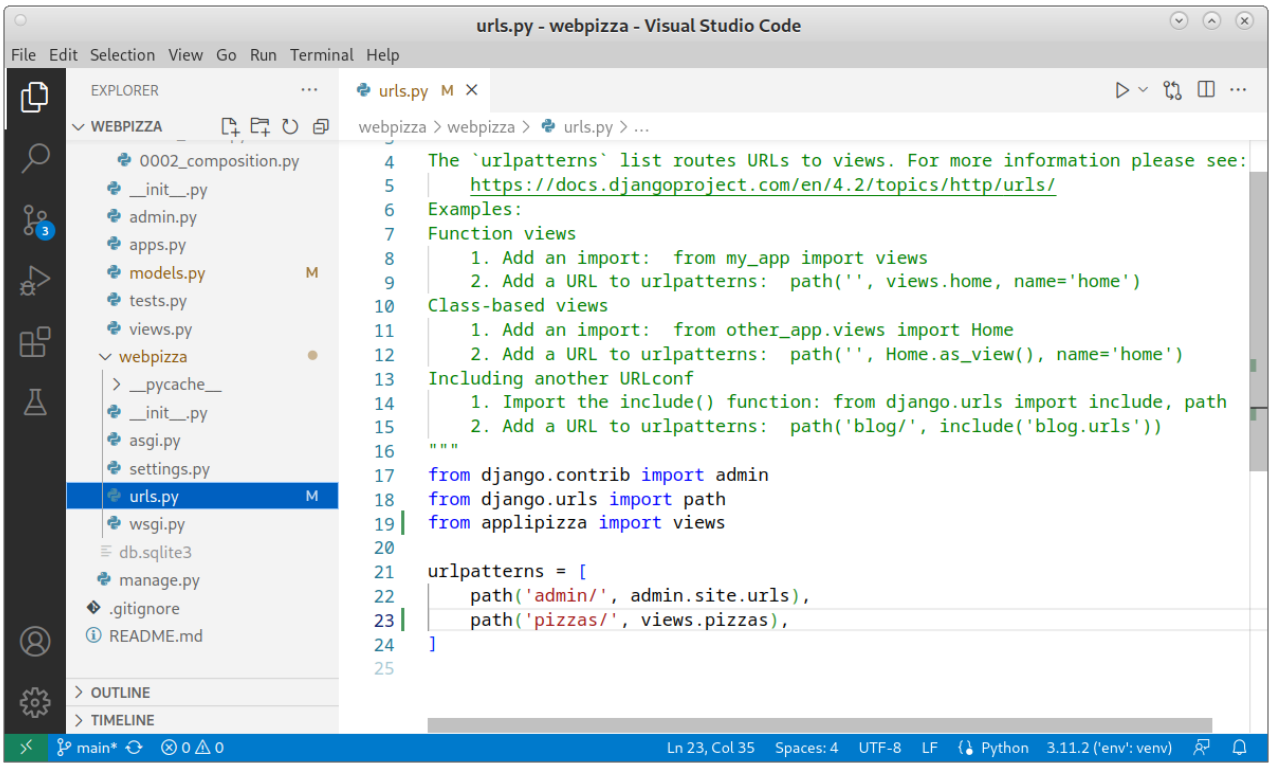
1. L'url dans le fichier urls.py
   a. dans le fichier urls.py, ajoutez la ligne d'import

      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

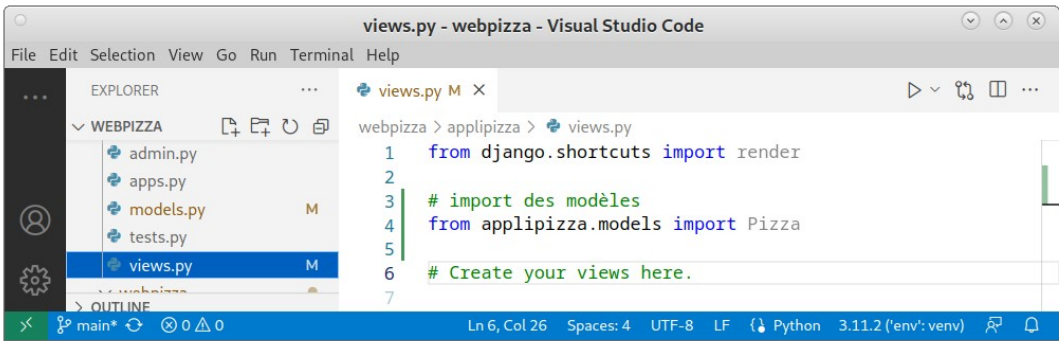
      path('pizzas/', views.pizzas),



Ceci signifie que l'url http://127.0.0.1:8000/pizzas/ dans le navigateur appellera la fonction **pizzas** du fichier **views.py**. C'est cette fonction contrôleur qui fera les calculs adaptés et appellera le template adéquat affichant les pizzas. Ce template sera précisé dans la fonction **pizzas** du fichier **views.py**.

2. La fonction pizzas dans views.py (similaire à un contrôleur)

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



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

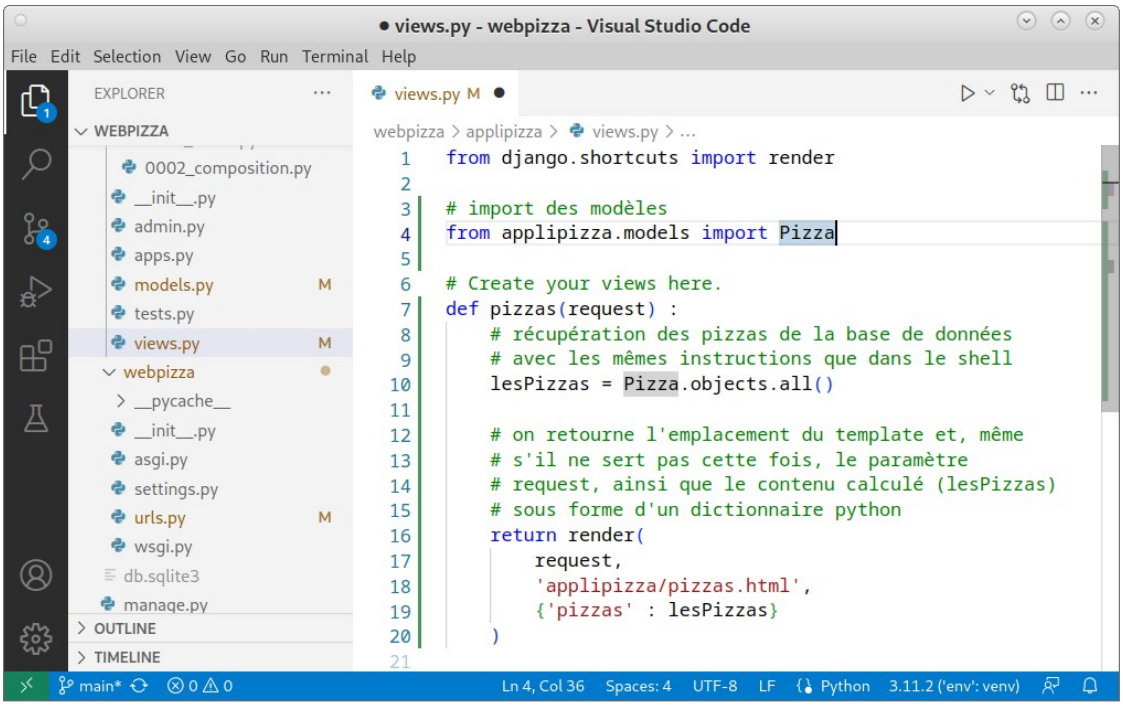
      Ce paramètre **request** sera très utile plus tard, car il embarque de nombreux renseignements, comme par exemple l'utilisateur connecté, le tableau POST (équivalent du $_POST de PHP), etc.

      Cette fonction **pizzas** récupère les données de la base sur les pizzas, et appelle le bon template (ici, un fichier applipizza/pizzas.html, pas encore créé).

      Ceci est typique d'une méthode de contrôleur à la PHP :

      - récupérer ce qui vient de l'url ou d'un formulaire (POST ou GET),
      - en fonction de la demande, récupérer les données de la base de données,
      - grâce à ces données, créer des contenus (en PHP, des objets ou des tableaux, en python, des dictionnaires par exemple),
      - appeler le bon template en lui fournissant ces contenus pour qu'il gère l'affichage attendu.

      Voici le code simple de cette fonction :



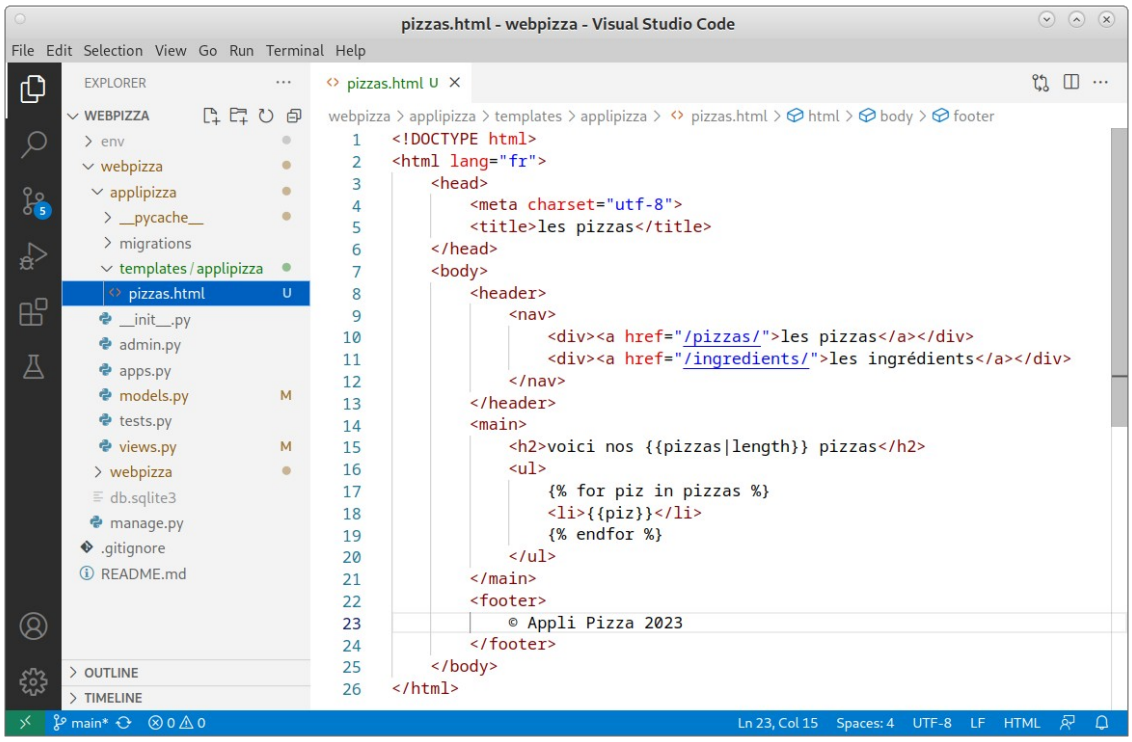
3. Le fichier pizzas.html (similaire à une vue)

   a. créez dans le répertoire **applipizza** un sous-répertoire **templates**, et dans **templates** un sous-répertoire **applipizza**.

   b. Créez, dans applipizza/templates/applipizza/, le fichier pizzas.html et donnez-lui le contenu proposé dans la capture qui suit.

      - 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.

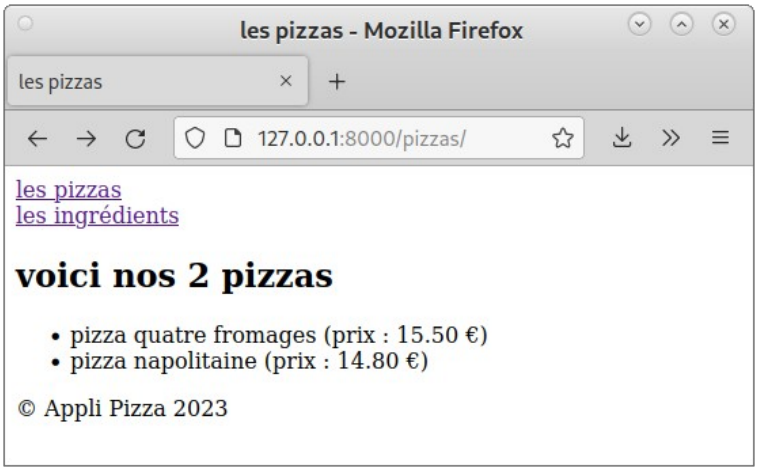
N'oubliez pas de mettre à jour le dépôt distant.



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 de la page suivante. Le menu est à moitié opérationnel, car rien n'a été fait pour la vue des ingrédients. Il fonctionnera réellement après le travail à faire sur les ingérdients. Pour le moment, le clic sur « les ingrédients » amènera un message page not found, puisque l'url des ingrédients n'a pas été écrite.



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, le détail 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 généricité des fichiers template, 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**.

Le css Bootstrap 5

Nous allons maintenant, pour terminer ce TP, intégrer Bootstrap 5 à notre projet. Cela peut se faire très simplement en indiquant, dans le head de nos templates, les liens vers les fichiers css et js de bootstrap.

Nous allons aussi changer le format html de notre menu, pour lui donner un format qui répond au css de bootstrap.

Dans le head de vos deux fichiers template, intégrez le lien vers le css et le javascript de bootstrap.

Voir le lien suivant pour intégrer dans la balise head les bons éléments :

https://getbootstrap.com/docs/5.0/getting-started/introduction/

N'oubliez pas qu'un lien css se place habituellement dans le head, alors qu'un script JS se place juste avant la balise fermante </body>. En effet, pour que le script fonctionne correctement, il faut que toutes les balises html soient chargées.

Ensuite, changez dans vos deux fichiers template le menu actuel

```
<header>
    <nav>
        <div><a href="/pizzas/">les pizzas</a></div>
        <div><a href="/ingredients/">les ingrédients</a></div>
    </nav>
</header>
```

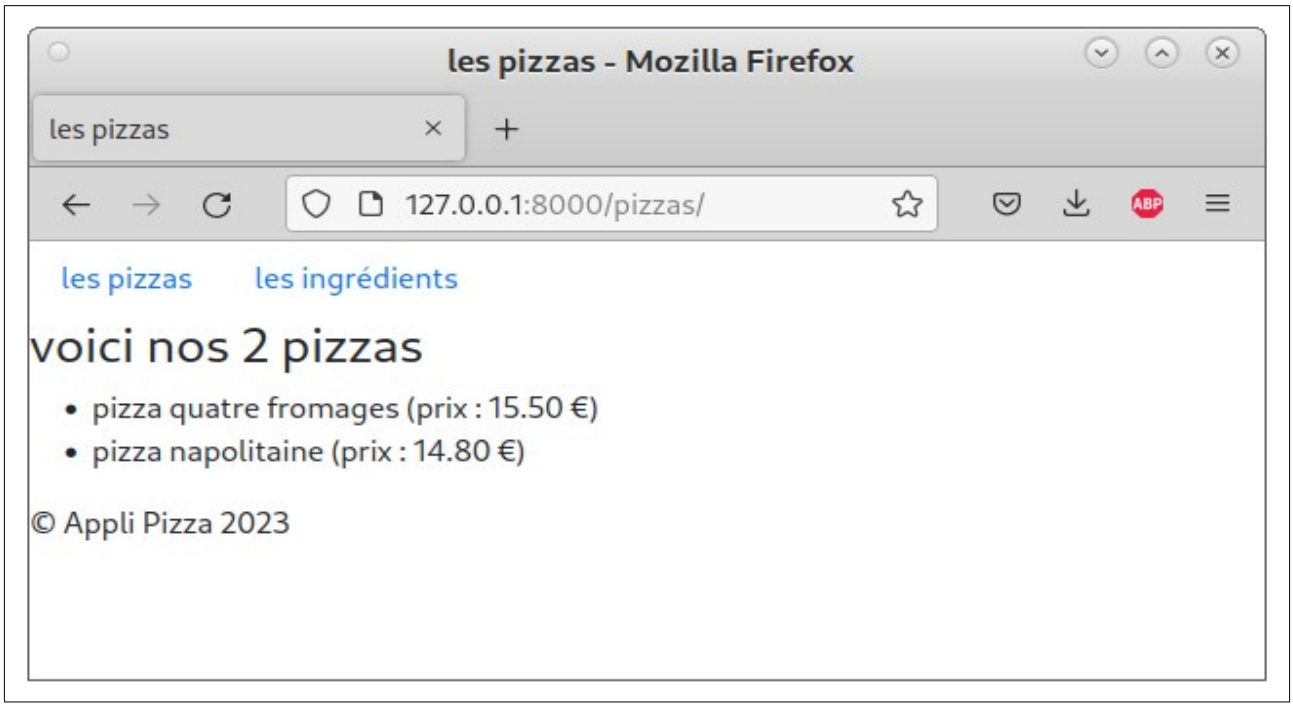
en

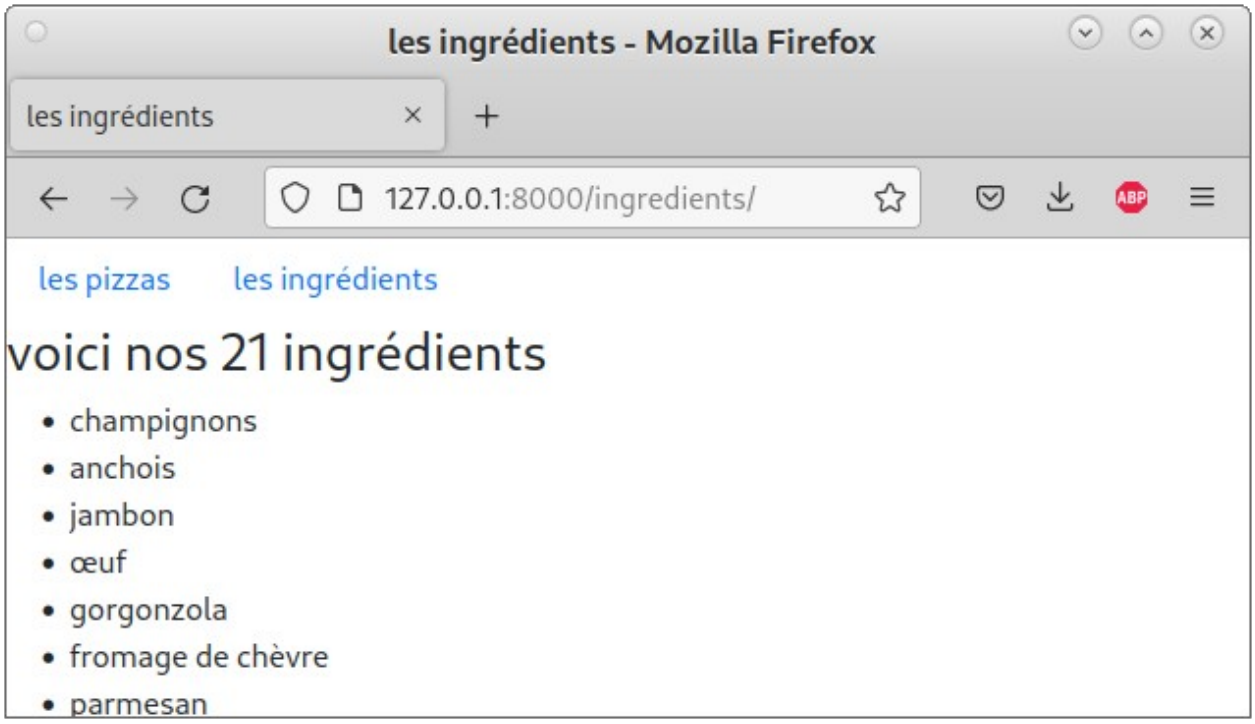
```
<header>
    <ul class="nav">
        <li class="nav-item">
            <a class="nav-link active" aria-current="page" href="/pizzas/">les pizzas</a>
        </li>
        <li class="nav-item">
            <a class="nav-link" href="/ingredients/">les ingrédients</a>
        </li>
    </ul>
</header>
```

Cette deuxième version utilise les classes css de bootstrap.

Vous allez constater le changement, y compris au niveau des polices de caractères, ainsi que des styles des titres h2 par exemple.

Par la suite, nous fouillerons les possibilités de bootstrap.


les pizzas - Mozilla Firefox
les pizzas
127.0.0.1:8000/pizzas/
les pizzas
les ingrédients
voici nos 2 pizzas
pizza quatre fromages (prix : 15.50 €)
pizza napolitaine (prix : 14.80 €)
© Appli Pizza 2023


les ingrédients - Mozilla Firefox
les ingrédients
127.0.0.1:8000/ingredients/
les pizzas
les ingrédients
voici nos 21 ingrédients
champignons
anchois
jambon
œuf
gorgonzola
fromage de chèvre
parmesan