TP11 – connexion, déconnexion

Introduction, Contexte et Objectifs

Pour le moment, nous avons codé un CRUD comlet, sans nous préoccuper de savoir qui aura les privilèges du CRUD.

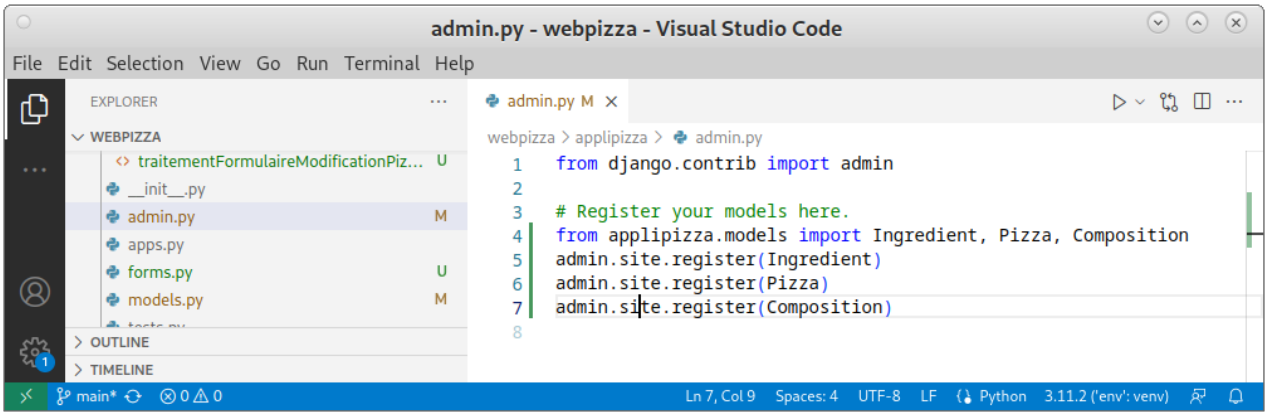
Le moment est venu de créer un administrateur du site, qui aura seul les autorisations sensibles.

Cela passe par la création, avec django, d'un « superuser ». Ce superuser aura accès à la partie administration du site, avec une interface standard toute prête. Nous verrons alors l'interface admin par défaut.

Nous aborderons aussi les fonctionnalités login et logout du système de connexion, ainsi que la différenciation des menus et des liens.

Le fichier de configuration de l'administration

Modifiez le fichier admin.py de l'application applipizza pour que son contenu soit



Par ces commandes, on prévoit l'administration des modèles Ingredient, Pizza et Composition par le superuser, via l'interface administrateur proposée par django.

Le super-utilisateur

Django a besoin d'un superuser pour garantir que quelqu'un a les super-pouvoirs d'administration. En particulier, si on crée des utilisateurs connectés, il pourra les modifier, les supprimer.

1. Créez le superuser par : python3 manage.py createsuperuser

   Renseignez l'email et le mot de passe.

2. Lancez le serveur web et connectez-vous à l'adresse

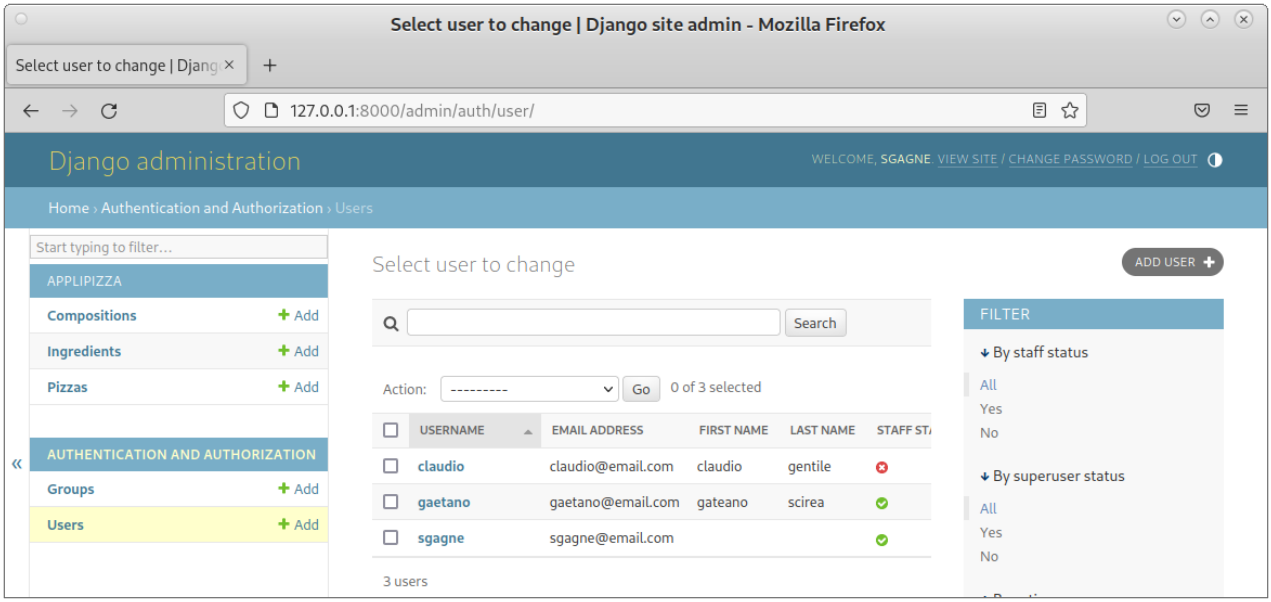
   http://127.0.0.1:8000/admin/

   en précisant les identifiant et mot de passe du superuser.

3. Vous avez acès à une interface d'administrateur permettant d'ajouter, modifier ou supprimer des pizzas, des ingrédients ou des compositions. Testez en créant une nouvelle pizza, de nouveaux ingrédients et en composant votre nouvelle pizza.

   Pour plus de détails, vous pouvez regarder le site MDN : https://developer.mozilla.org/fr/docs/Learn/Server-side/Django/Admin_site

4. En tant que superuser, créez deux autres utilisateurs. L'un ne sera pas « staff », l'autre sera « staff ». Vous verrez que ce statut hiérarchise les users et nous permettra, par la suite, de différencier les menus et les permissions.



L'application de gestion des comptes

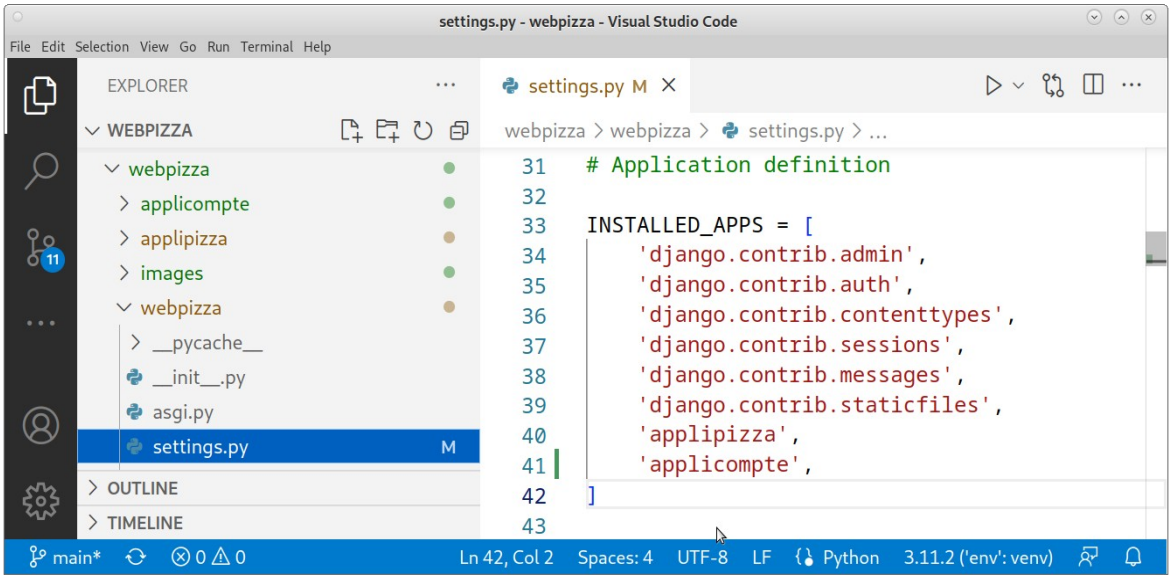
Nous allons mettre en place une nouvelle application, en parallèle de l'application applipizza. Elle gèrera tous les aspects de la connexion (login logout, création de compte, etc).

Django propose une application dédiée à la gestion des comptes, toute faite et prête à l'emploi, mais nous allons personnaliser la nôtre, et notamment ses templates, pour utiliser au mieux bootstrap et contrôler le contenu des divers formulaires. Nous utiliserons les fonctionnalités d'authentification de django (authenticate, login, logout, …).

1. Créez la nouvelle application applicompte par la commande

Comme pour applipizza, cette commande a créé de nouveaux fichiers dans le projet.

2. Enregistrez cette application dans la liste INSTALLED_APPS du fichier settings.py.



3. Nous allons maintenant nous occuper des urls. Jusqu'à maintenant, elles étaient toutes regroupées dans le fichier urls.py de webpizza, et c'est logique puisqu'il n'y avait qu'une application applipizza.

   Maintenant, nous allons gérer plusieurs applications (au moins 2) et donc nous allons pourvoir chacune d'un fichier urls.py avec ses propres url.

   Nous allons donc transférer les url propres à applipizza dans le dossier applipizza, et créer séparément des url propres à l'application applicompte.

   Enfin, le fichier urls.py de webpizza centralisera toutes les urls par des inclusions.

   Dans un premier temps, nous allons nettoyer le fichier urls.py de webpizza.

   a. Créez un fichier urls.py dans applipizza (au même niveau que models.py ou views.py)

   b. Reportez-y toutes les url propres à applipizza. Votre fichier devrait plus ou moins ressembler à :

urls.py U

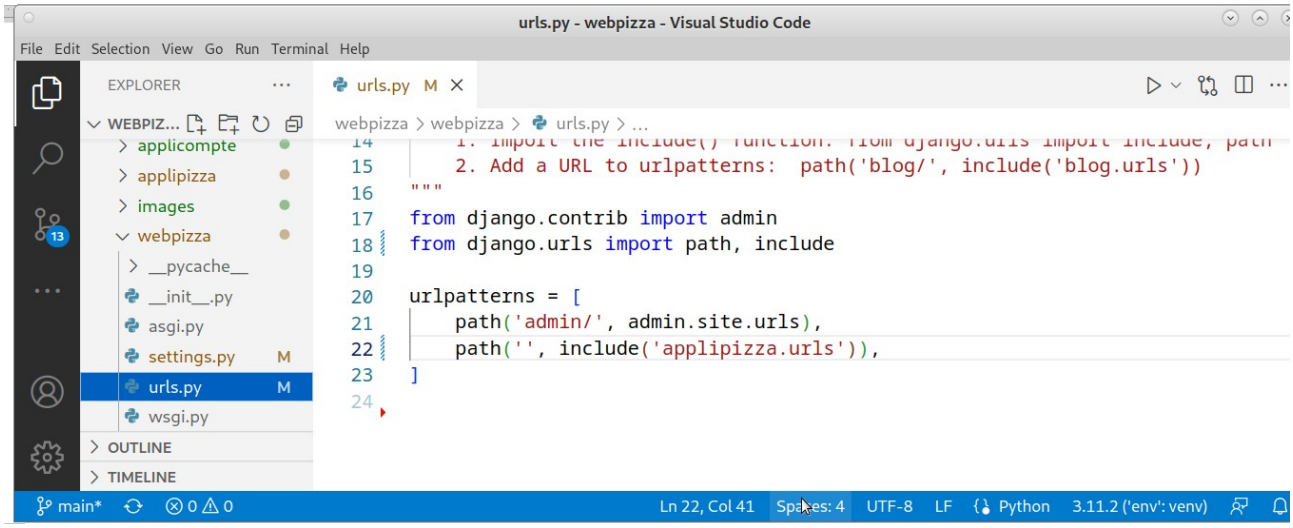
webpizza > applipizza > urls.py > ...

```python
from django.urls import path
from applipizza import views
from django.conf import settings
from django.conf.urls.static import static

urlpatterns = [
    path('pizzas/', views.pizzas),
    path('ingredients/', views.ingredients),
    path('pizzas/<int:pizza_id>/', views.pizza),
    path('ingredients/add/', views.formulaireCreationIngredient),
    path('ingredients/create/', views.creerIngredient),
    path('ingredients/<int:ingredient_id>/delete/', views.supprimerIngredient),
    path('ingredients/<int:ingredient_id>/update/', views.afficherFormulaireModificationIngredient),
    path('ingredients/<int:ingredient_id>/updated/', views.modifierIngredient),
    path('pizzas/add/', views.formulaireCreationPizza),
    path('pizzas/create/', views.creerPizza),
    path('pizzas/<int:pizza_id>/addIngredient/', views.ajouterIngredientDansPizza),
    path('pizzas/<int:pizza_id>/delete/', views.supprimerPizza),
    path('pizzas/<int:pizza_id>/update/', views.afficherFormulaireModificationPizza),
    path('pizzas/<int:pizza_id>/updated/', views.modifierPizza),
    path('pizzas/<int:pizza_id>/deleteIngredient/<int:composition_id>/', views.supprimerIngredientDansPizza),
]

urlpatterns += static(settings.MEDIA_URL, document_root = settings.MEDIA_ROOT)
```

c. Modifiez le fichier urls.py de webpizza pour qu'il insère les urls de applipizza :



d. Vérifiez que tout fonctionne. C'est plus logique de faire ainsi : à chaque application ses urls, et le fichier urls.py du projet inclut les fichiers urls.py de toutes les applications.

e. Créez maintenant un fichier urls.py propre à l'application applicompte, et n'oubliez pas de l'inclure dans le fichier urls.py du projet webpizza. Ce nouveau fichier urls.py contiendra au départ le code suivant :

webpizza > applicompte > urls.py > ...

```python
from django.urls import path

# import des views par défaut du système d'authentification
# de django, qui sera renommé auth_views
from django.contrib.auth import views as auth_views

# import des views d'applicompte, que nous allons écrire
from applicompte import views

urlpatterns = [

    # ce premier path utilise la view par défaut pour la connexion,
    # avec un template login.html que nous allons écrire (nous aurions
    # aussi pu utiliser le template par défaut de django)
    path('login/', auth_views.LoginView.as_view(template_name = 'applicompte/login.html'), name = 'login'),
]
```

Ce path, qui utilise le système d'authentification prêt à l'emploi de django (django.contrib.auth) permet d'inclure l'url de login. Cette url n'est bien sûr pas encore opérationnelle puisque nous n'avons pas encore codé le template login.html.

Comme nous allons personnaliser la connexion, nous allons redéfinir les templates nécessaires.

Nous allons aussi personnaliser les menus, car un utilisateur non connecté, un utilisateur basique connecté (un client) ou un utilisateur « staff » n'auront pas les mêmes menus.

Pour le moment, le template menu.html a été codé dans l'application applipizza, puisque c'était la seule. Cependant, il peut aussi y avoir une certaine logique à coder les menus dans applicompte, puisque les menus proposés vont dépendre du statut de l'utilisateur. C'est ce que nous allons finalement choisir comme solution : coder les menus dans applicompte.

Relocalisation du menu

Pour le moment, vous avez dans applipizza un template menu.html, qui doit, à du bootstrap près, ressembler à :

```
{% load static %}
<nav class="navbar navbar-expand-lg navbar-dark bg-primary">
    <div class="container-fluid">
        <a class="navbar-brand" href="#">
            <img src="{% static 'applipizza/img/pizza.PNG' %}" width="40" height="40" alt="">
            Appli Pizza 2023
        </a>
        <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarText"
            aria-controls="navbarText" aria-expanded="false" aria-label="Toggle navigation">
            <span class="navbar-toggler-icon"></span>
        </button>
        <div class="collapse navbar-collapse" id="navbarText">
            <ul class="navbar-nav me-auto mb-2 mb-lg-0">
                <li class="nav-item"><a class="nav-link" href="/pizzas/">les pizzas</a></li>
                <li class="nav-item"><a class="nav-link" href="/pizzas/add/">créer une pizza</a></li>
                <li class="nav-item"><a class="nav-link" href="/ingredients/">les ingrédients</a></li>
                <li class="nav-item"><a class="nav-link" href="/ingredients/add/">créer un ingrédient</a></li>
            </ul>
        </div>
    </div>
</nav>
```

Ce menu est en fait adapté à un utilisateur qui a des permissions avancées de type staff (les permissions liées à la création, la modification et la suppression d'objets).

Nous allons maintenant traduire la hiérarchie chez les utilisateurs, ce qui conduira à une personnalisation du menu.

Nous allons donc créer, sur la base de menu.html, trois fichiers menuStaff.html, menuBasicUser.html et enfin, le menu sans connexion menuNonConnecte.html.

1. Créez un répertoire templates/applicompte dans le répertoire applicompte, comme cela avait été fait pour applipizza.

2. Déplacez-y le template menu.html.

3. Dans le template base.html d'applipizza, modifiez l'include du menu pour qu'il soit récupéré dans les templates d'applicompte à l'avenir.

4. Testez que tout fonctionne. Attention, si vous obtenez une erreur de la forme « TemplateDoesNotExist », il faut probablement relancer le serveur.

Personnalisation du menu

Nous allons en fait créer trois menus différents, ce sera plus simple :

- un menu pour les internautes non connectés
- un menu pour les clients connectés
- un menu pour les administrateurs

Pour le visuel final de chacun des menus, voir la vidéo de présentation du cours.

1. Créez le template applicompte/menuNonConnecte.html dont le code permettra d'accéder à la liste des pizzas, de s'inscrire et de se connecter :

menuNonConnecte.html M

webpizza > applipizza > templates > applipizza > menuNonConnecte.html > nav.navbar.navbar-expand-lg.navbar-dark.bg-primary

```
{% load static %}
<nav class="navbar navbar-expand-lg navbar-dark bg-primary">
    <div class="container-fluid">
        <a class="navbar-brand" href="#">
            <img src="{% static 'applipizza/img/pizza2.PNG' %}" width="40" height="40" alt="">
            Appli Pizza 2023
        </a>
        <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarTex
            <span class="navbar-toggler-icon"></span>
        </button>
        <div class="collapse navbar-collapse" id="navbarText">
            <ul class="navbar-nav me-auto mb-2 mb-lg-0">
                <li class="nav-item"><a class="nav-link" href="/pizzas/">les pizzas</a></li>
            </ul>
            <ul class="navbar-nav">
                <li class="nav-item"><a class="nav-link" href="/register"> inscription </a></li>
                <li class="nav-item"><a class="nav-link" href="/login/"> connexion </a></li>
            </ul>
        </div>
    </div>
</nav>
```

2. Créez le template menuStaff.html, qui présente les 4 liens du CRUD, avec en plus un lien de déconnexion :

```
<div class="collapse navbar-collapse" id="navbarText">
    <ul class="navbar-nav me-auto mb-2 mb-lg-0">
        <li class="nav-item"><a class="nav-link" href="/pizzas/">les pizzas</a></li>
        <li class="nav-item"><a class="nav-link" href="/pizzas/add/">créer une pizza</a></li>
        <li class="nav-item"><a class="nav-link" href="/ingredients/">les ingrédients</a></li>
        <li class="nav-item"><a class="nav-link" href="/ingredients/add/">créer un ingrédient</a></li>
    </ul>
    <ul class="navbar-nav">
        <li class="nav-item"><a class="nav-link" href="/logout/"> déconnexion </a></li>
    </ul>
</div>
```

3. Créez enfin le template menuClient.html, qui propose un lien vers la liste des pizzas, et des liens vers le panier, vers l'historique de commandes et un lien de déconnexion :

```
<div class="collapse navbar-collapse" id="navbarText">
    <ul class="navbar-nav me-auto mb-2 mb-lg-0">
        <li class="nav-item"><a class="nav-link" href="/pizzas/"> les pizzas </a></li>
    </ul>
    <ul class="navbar-nav">
        <li class="nav-item"><a class="nav-link" href="/panier/"> panier </a></li>
        <li class="nav-item"><a class="nav-link" href="/commandes/"> commandes </a></li>
        <li class="nav-item"><a class="nav-link" href="/logout/"> déconnexion </a></li>
    </ul>
</div>
```

Pour le moments les liens de login, logout, inscription, panier et commandes ne sont pas opérationnels.

Remarques :

a. Nous ne pouvons pas vraiment tester ces templates puisque nous n'avons pas encore le moyen de nous connecter autrement qu'en étant staff;

b. Il va aussi falloir, plus tard, personnaliser les accès aux fonctions du CRUD accessible à partir de pizzas.html. Un utilisateur quelconque aura aussi une différentiation dans l'affichage d'une pizza (en gros, les détails et un lien d'achat). Nous réglerons cela plus tard.

4. Vous allez maintenant, dans applipizza/base.html, insérer des tests qui permettront de savoir quel menu il faut insérer :

si user est staff on insère menuStaff.html
sinon si user est authentifié on insère menuClient.html
sinon on insère menuNonConnecte.html

```
<header>
    {% if user.is_staff %}
    {% include 'applicompte/menuStaff.html' %}
    {% elif user.is_authenticated %}
    {% include 'applicompte/menuClient.html' %}
    {% else %}
    {% include 'applicompte/menuNonConnecte.html' %}
    {% endif %}
</header>
```

5. Vérifiez ce que vous pouvez vérifier, à savoir ce qui se passe dans le cas d'un user staff et dans le cas d'un user non connecté. Pour le non connecté, vous devrez vous déconnecter à partir du portail administrateur, puis relancer l'url pizzas/.

Création des templates login et logout d'applicompte

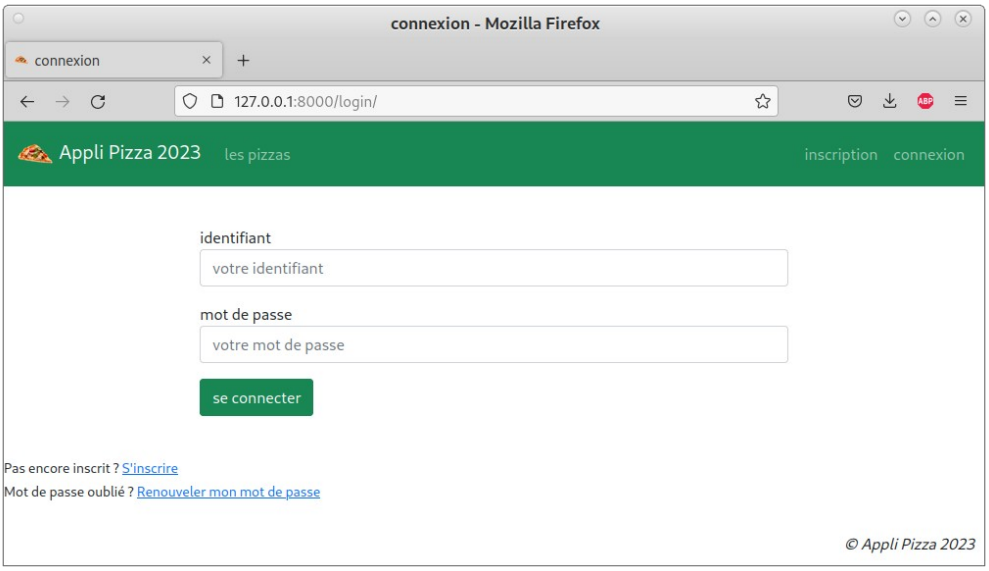
1. créez le fichier login.html avec le code suivant :

```
{% extends 'applipizza/base.html' %}
{% block title %} connexion {% endblock %}
{% block main %}
<form id="form_login" method="post" action="/connexion/">
    {% csrf_token %}
    <div class="mb-3">
        <label for="id_username"> identifiant </label>
        <input type="text" class="form-control" id="id_username" name="username" placeholder="votre identifiant">
    </div>
    <div class="mb-3">
        <label for="id_password"> mot de passe </label>
        <input type="password" class="form-control" id="id_password" name="password" placeholder="votre mot de passe">
    </div>
    <button type="submit" class="btn btn-primary">se connecter</button>
</form>
<div>
    <small>
        Pas encore inscrit ?
        <a href="/register/">
            S'inscrire
        </a>
        <br>
        Mot de passe oublié ?
        <a href="/password_reset/">
            Renouveler mon mot de passe
        </a>
    </small>
</div>
{% endblock %}
```

Dans ce code il y a un formulaire de connexion, en mode bootstrap, et hors-formulaire des liens d'inscription et de renouvellement du mot de passe.

Testez maintenant l'url http://127.0.0.1:8000/login/ et vérifiez que vous arrivez à la page de connexion.

Par contre l'envoi du formulaire provoquera une erreur puisque rien n'est prévu encore pour l'action de connexion (ni url, ni view).



2. Créez le template applicompte/logout.html sur le modèle suivant (vous pouvez personnaliser)

```
{% extends 'applipizza/base.html' %}
{% load static %}
{% block title %} au revoir {% endblock %}
{% block main %}
<div class="div-logout">
    <h2>Vous nous manquez déjà !</h2>
    <h2>Nous espérons vous revoir bientôt !</h2>
    <img class="imagePizza" src="{% static 'applipizza/img/pizza.jpg' %}" alt="imagePizza">
    <a href="/login/"> connexion </a>
</div>
{% endblock %}
```

Les url et les views

1. Complétez le fichier urls.py d'applicompte pour y ajouter l'appel de deux views : logout et connexion

webpizza > applicompte > urls.py > ...

```
from django.urls import path

# import des views par défaut du système d'authentification
# de django, qui sera renommé auth_views
from django.contrib.auth import views as auth_views

# import des views d'applicompte, que nous allons écrire
from applicompte import views

urlpatterns = [
    path('login/', auth_views.LoginView.as_view(template_name = 'applicompte/login.html'), name = 'login'),
    path('logout/', views.deconnexion, name = 'logout'),
    path('connexion/', views.connexion),
]
```

2. Dans le fichier views.py d'applicompte, créez les views suivantes :

   a. connexion, qui récupère les éléments du formulaire de connexion (username et password), authentifie la connexion (en retournant par **authenticate** un user ou bien None), et qui retourne le contexte adapté (avec notamment le user, qui servira forcément d'une manière ou d'une autre)

```
webpizza > applicompte > views.py > ...
1   from django.shortcuts import render
2   from django.contrib.auth import authenticate, login, logout
3   from applipizza.models import Pizza
4
5   # Create your views here.
6   def connexion(request) :
7       usr = request.POST['username']
8       pwd = request.POST['password']
9       user = authenticate(request, username = usr, password = pwd)
10      if user is not None:
11          login(request, user)
12          lesPizzas = Pizza.objects.all()
13          return render(
14              request,
15              'applipizza/pizzas.html',
16              {"pizzas" : lesPizzas, "user" : user}
17          )
18      else:
19          return render(
20              request,
21              'applicompte/login.html'
22          )
```

   b. deconnexion, qui déconnecte le user connecté et retourne le contexte avec le template de logout.

```
webpizza > applicompte > views.py > ...
1   from django.shortcuts import render
2   from django.contrib.auth import authenticate, login, logout
3   from applipizza.models import Pizza
4
5   # Create your views here.
6 > def connexion(request) : ···
23
24  def deconnexion(request) :
25      logout(request)
26      return render(
27          request,
28          'applicompte/logout.html'
29      )
```

3. Faites des tests de connexion : staff et client (non staff), et aussi internaute non connecté. Vous devez voir la différentiation des menus. Vous pouvez aussi, une fois connecté, aller sur la page d'administration du projet django pour voir comment vous êtes reconnu. Par contre, vous constatez que tout le monde a encore accès au CRUD ! C'est un des points sur lesquels nous revenons au TP suivant.

Différenciation des liens affichés

Pour le moment, tout le monde a les mêmes boutons d'actions dans la liste des pizzas ou dans les détails d'une pizza. Arrangez-vous, comme pour les menus, pour que :

a. Dans la liste des pizzas, un utilisateur staff ait à sa disposition les liens détails, modification et suppression,

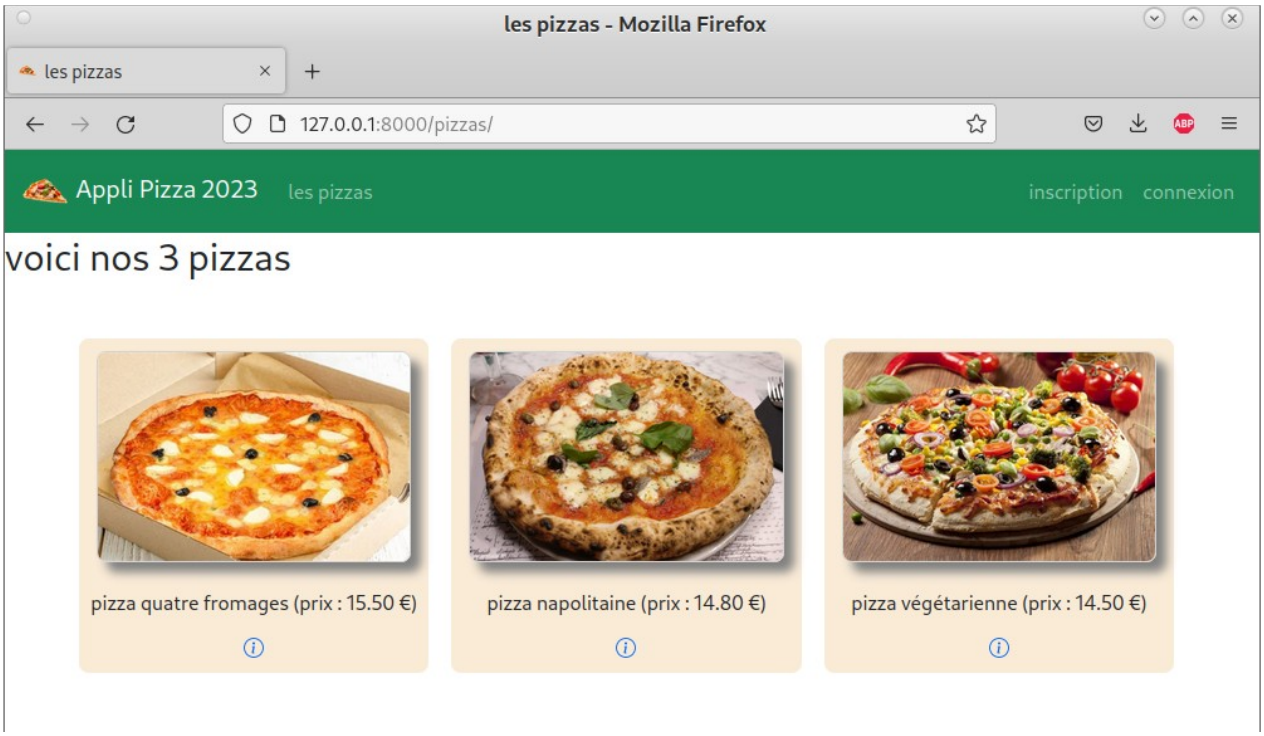
b. Dans la liste des pizzas, un client ait un lien d'achat (type panier) et le lien détails,

c. Dans la liste des pizzas, un utilisateur non connecté ait le lien détails,

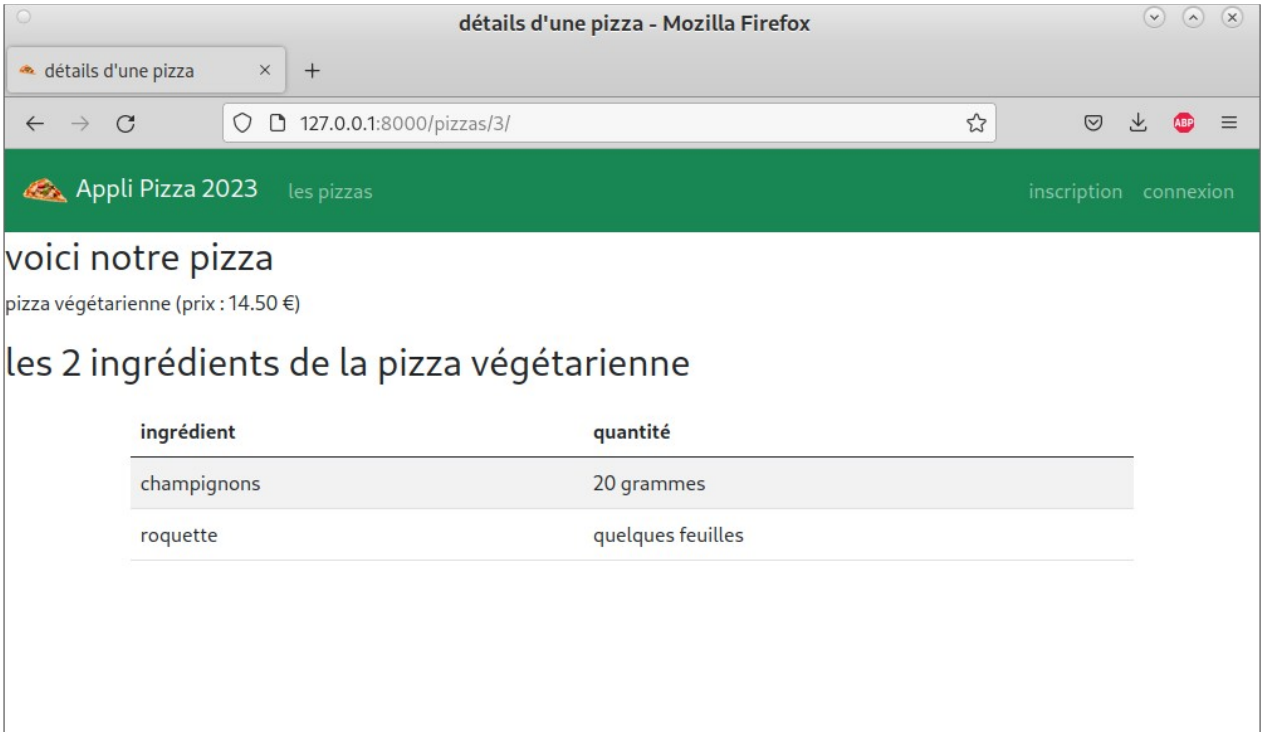
d. Dans les détails d'une pizza, les actions de suppression d'un ingrédient et d'ajout d'un ingrédient ne soient accessibles qu'au staff (et qu'un client ou un utilisateur non connecté n'aient accès qu'au tableau simplifié des ingrédients).

Vous pourrez utiliser les instructions :

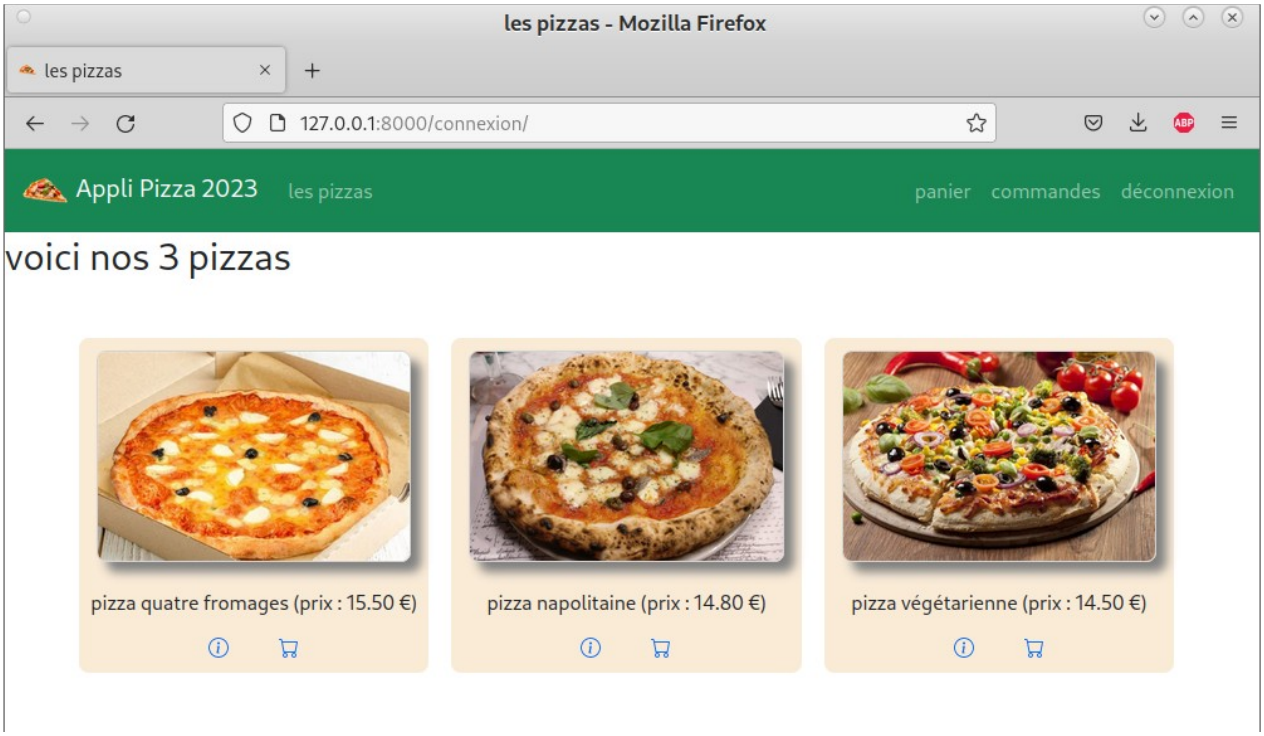
- {% if user.is_staff %} (utilisateur staff)
- {% if user.is_authenticated %} (staff ou client)



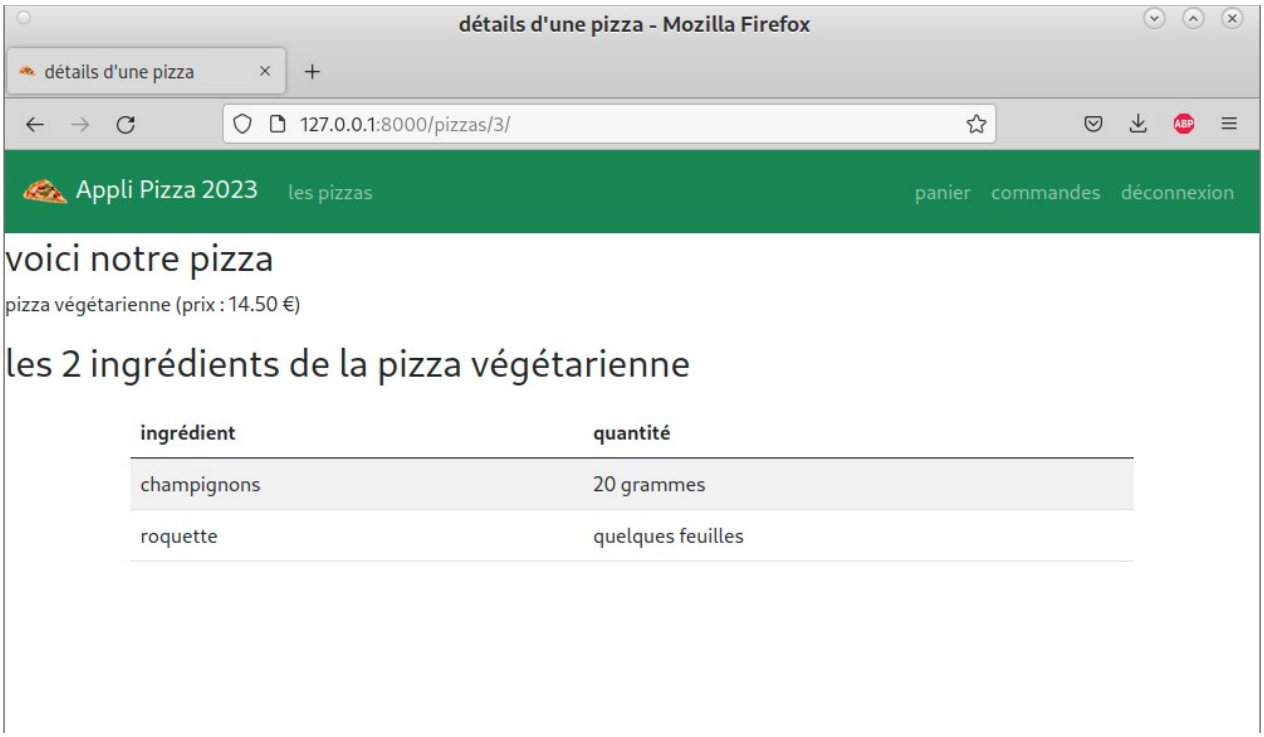
les pizzas pour un internaute non connecté



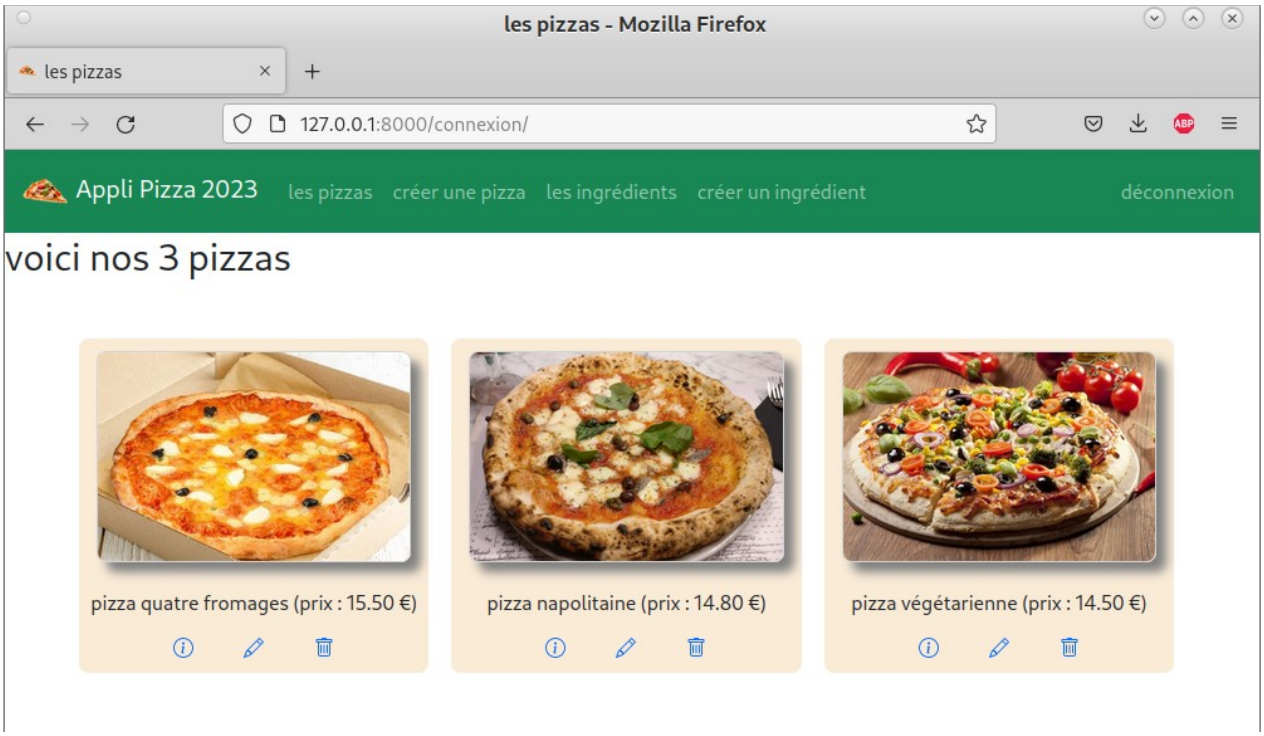
détails d'une pizza pour un internaute non connecté



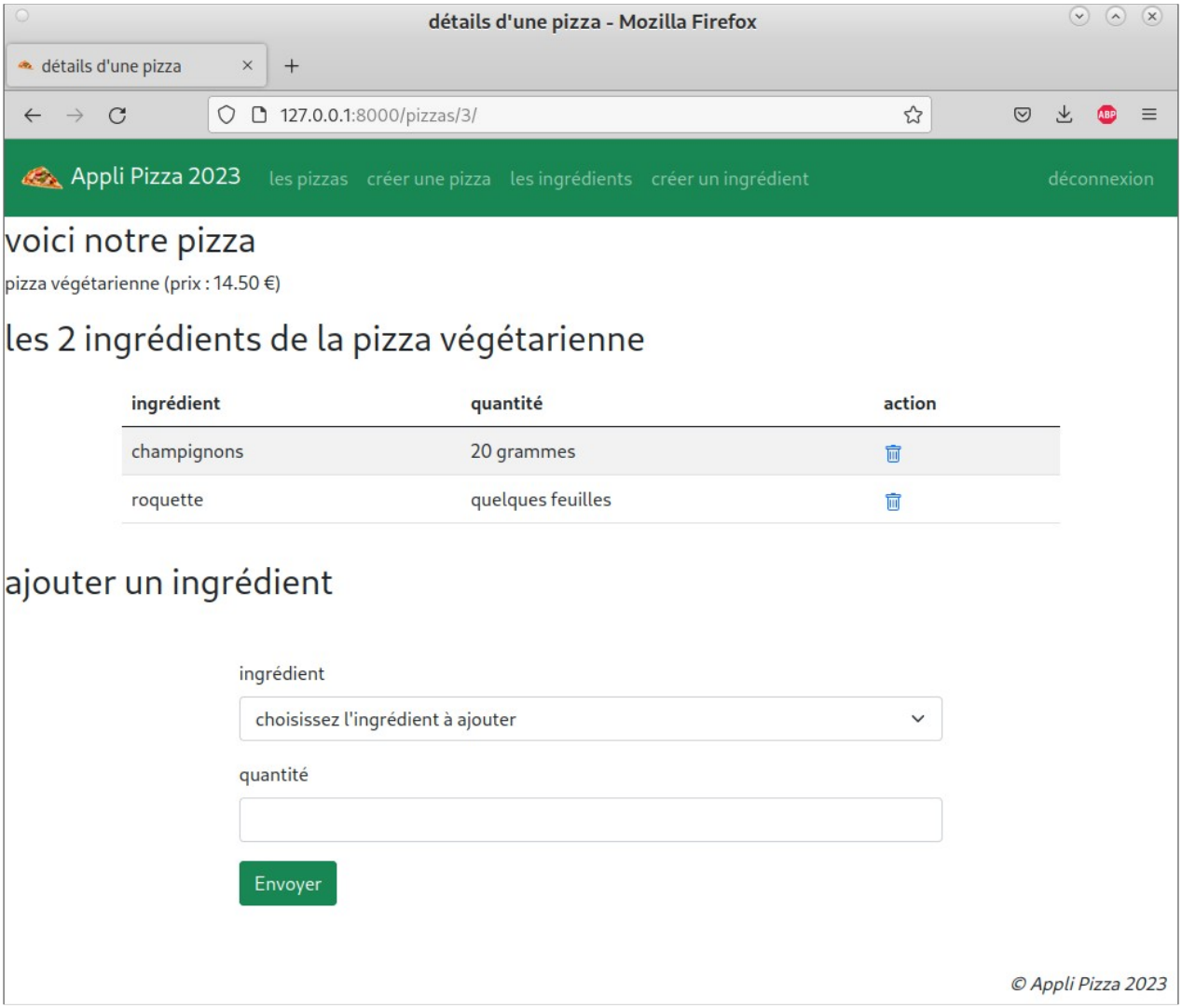
les pizzas pour un client connecté



détails d'une pizza pour un client connecté



les pizzas pour un utilisateur staff



détails d'une pizza pour un utilisateur staff

Mise en garde de sécurité !

Vous ferez attention à une chose : la sécurisation n'est pas faite sur les views (pour preuve, un utilisateur non connecté peut directement entrer une url de la forme

http://127.0.0.1:8000/pizzas/1/update/

et disposer des privilèges staff.

Il faut impérativement l'empêcher. Suite au prochain TP...