

TP10 – Administration du site – connexion (1/2)

Contexte et Objectifs

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

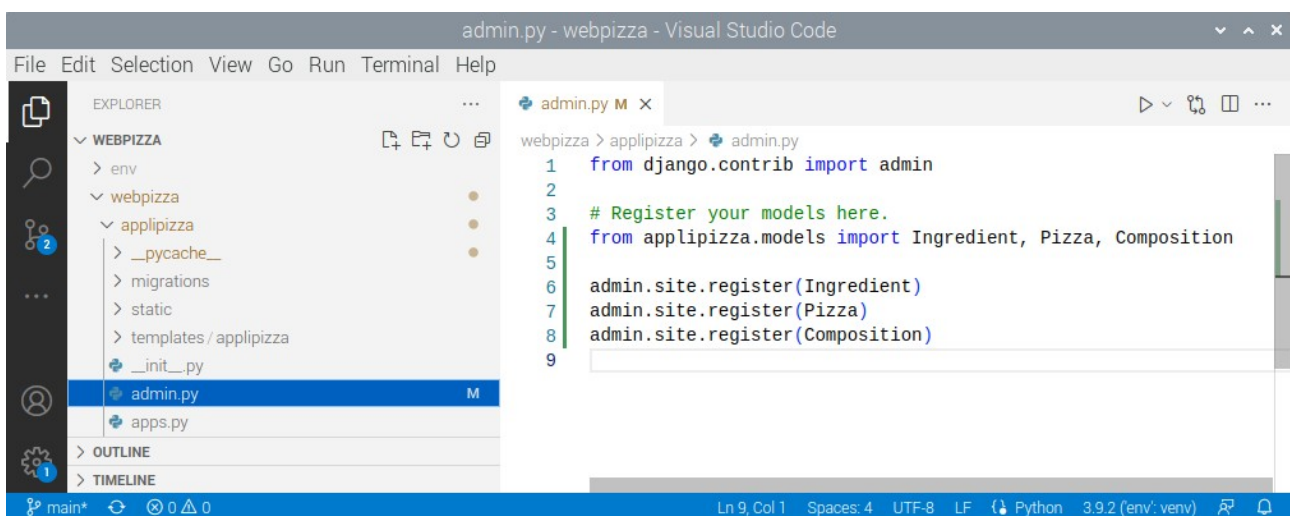
Le moment est venu de créer un administrateur du site, qui aura seul les autorisations sensibles.

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.

Le fichier de configuration de l'administration

Modifiez le fichier `admin.py` pour que son contenu soit



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

EXPLORER
  WEBPIZZA
    > env
    > webpizza
      > applipizza
        > __pycache__
        > migrations
        > static
        > templates / applipizza
        > __init__.py
        > admin.py
        > apps.py
    > OUTLINE
    > TIMELINE

admin.py M x
webpizza > applipizza > admin.py
1 from django.contrib import admin
2
3 # Register your models here.
4 from applipizza.models import Ingredient, Pizza, Composition
5
6 admin.site.register(Ingredient)
7 admin.site.register(Pizza)
8 admin.site.register(Composition)
9
```

Par ces commandes, on prévoit l'administration des modèles Ingredient, Pizza et Composition par 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 la commande

```
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 super-utilisateur.

3. Vous avez accès 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.

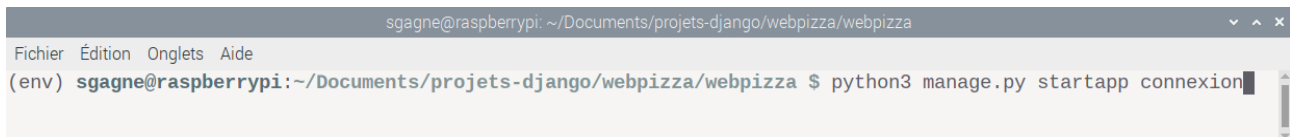
Pour plus de détails, vous pouvez regarder le site MDN :
https://developer.mozilla.org/fr/docs/Learn/Server-side/Django/Admin_site

4. En tant que super-utilisateur, créez au moins un autre utilisateur. Renseignez son email qui servira pour les changements de mot de passe.

L'application de connexion

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).

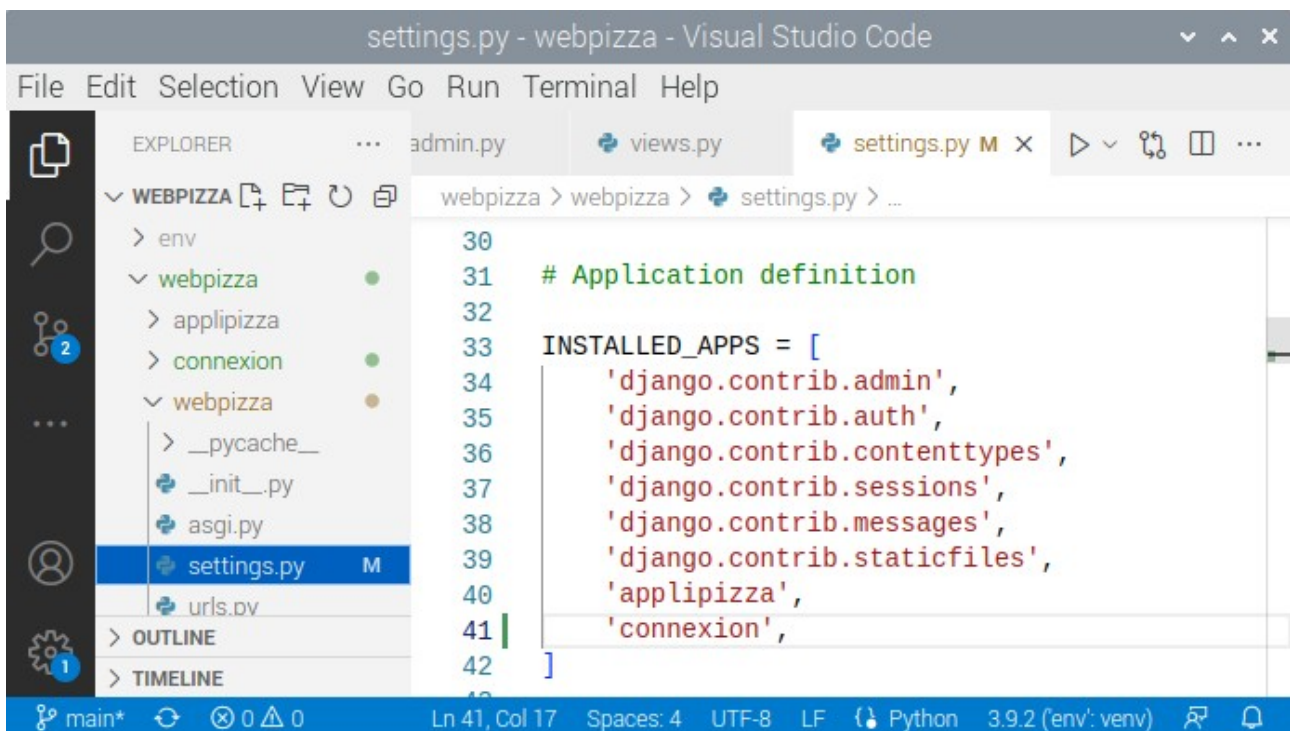
1. Créez la nouvelle application connexion par la commande



```
sgagne@raspberrypi: ~/Documents/projets-django/webpizza/webpizza
(env) sgagne@raspberrypi:~/Documents/projets-django/webpizza/webpizza $ python3 manage.py startapp connexion
```

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`.

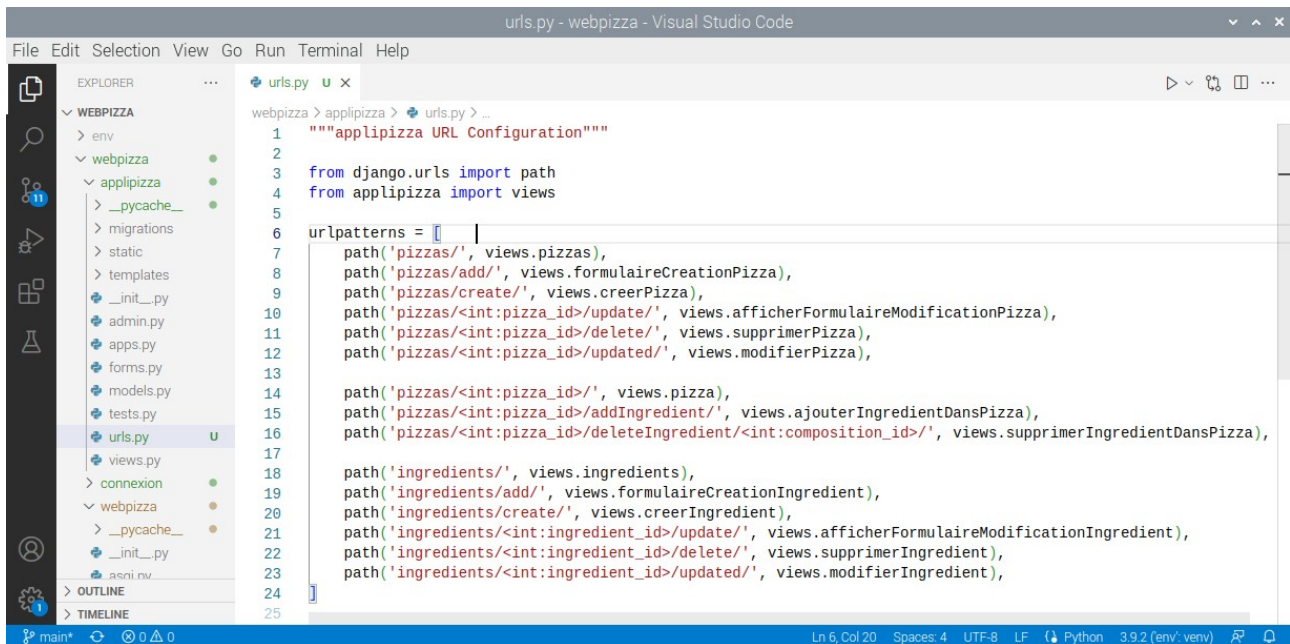


```
settings.py - webpizza - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
  WEBPIZZA
    > env
    > webpizza
      > applipizza
      > connexion
      > webpizza
        > __pycache__
        > __init__.py
        > asgi.py
        > settings.py M
        > urls.py
    > OUTLINE
    > TIMELINE
  admin.py
  views.py
  settings.py M x
webpizza > webpizza > settings.py > ...
30
31 # Application definition
32
33 INSTALLED_APPS = [
34     'django.contrib.admin',
35     'django.contrib.auth',
36     'django.contrib.contenttypes',
37     'django.contrib.sessions',
38     'django.contrib.messages',
39     'django.contrib.staticfiles',
40     'applipizza',
41     'connexion',
42 ]
```

3. Nous allons maintenant nous occuper des urls. Mais 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 :

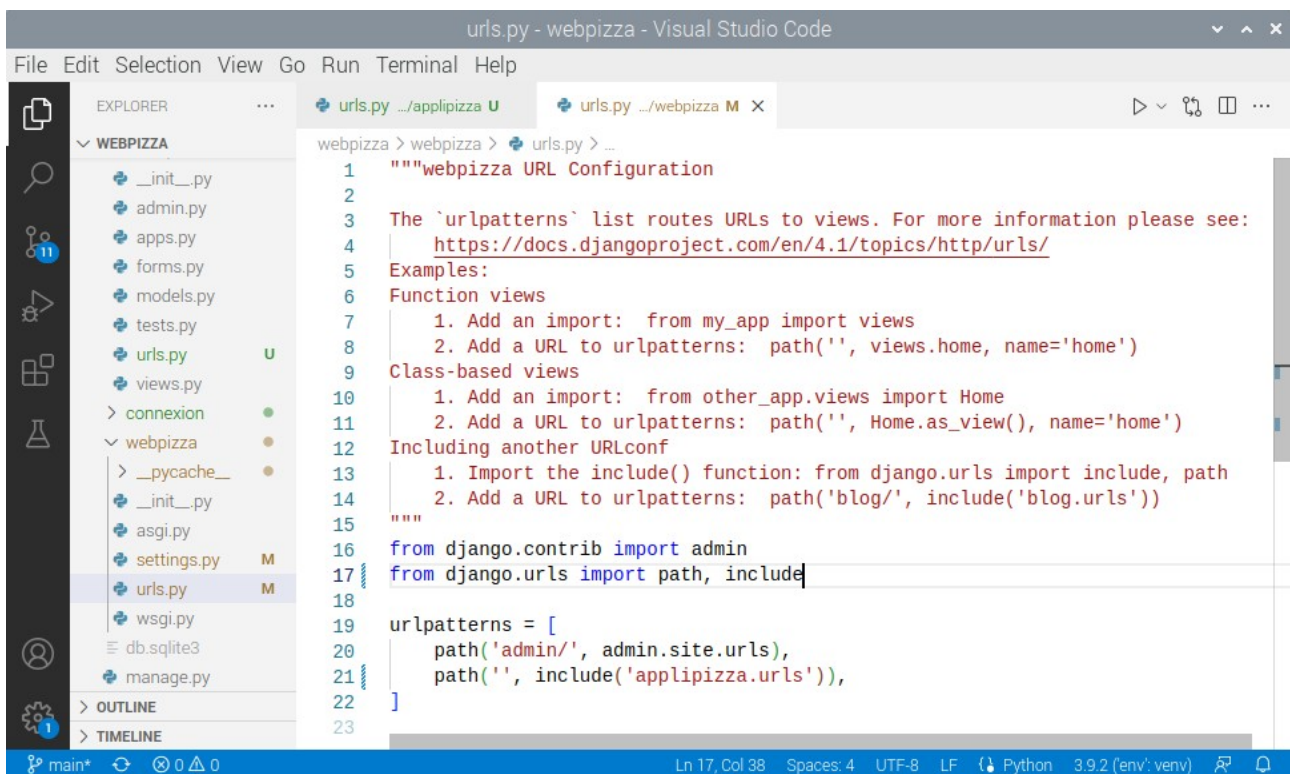


```
urls.py - webpizza - Visual Studio Code
File Edit Selection View Go Run Terminal Help

EXPLORER
  WEBPIZZA
    env
    webpizza
      applipizza
      __pycache__
      migrations
      static
      templates
      __init__.py
      admin.py
      apps.py
      forms.py
      models.py
      tests.py
      urls.py
      views.py
    connexion
    webpizza
      __pycache__
      __init__.py
      asgi.py
    OUTLINE
    TIMELINE

webpizza > applipizza > urls.py > ...
1  """applipizza URL Configuration"""
2
3  from django.urls import path
4  from applipizza import views
5
6  urlpatterns = [
7      path('pizzas/', views.pizzas),
8      path('pizzas/add/', views.formulaireCreationPizza),
9      path('pizzas/create/', views.creerPizza),
10     path('pizzas/<int:pizza_id>/update/', views.afficherFormulaireModificationPizza),
11     path('pizzas/<int:pizza_id>/delete/', views.supprimerPizza),
12     path('pizzas/<int:pizza_id>/updated/', views.modifierPizza),
13
14     path('pizzas/<int:pizza_id>', views.pizza),
15     path('pizzas/<int:pizza_id>/addIngredient/', views.ajouterIngredientDansPizza),
16     path('pizzas/<int:pizza_id>/deleteIngredient/<int:composition_id>', views.supprimerIngredientDansPizza),
17
18     path('ingredients/', views.ingredients),
19     path('ingredients/add/', views.formulaireCreationIngredient),
20     path('ingredients/create/', views.creerIngredient),
21     path('ingredients/<int:ingredient_id>/update/', views.afficherFormulaireModificationIngredient),
22     path('ingredients/<int:ingredient_id>/delete/', views.supprimerIngredient),
23     path('ingredients/<int:ingredient_id>/updated/', views.modifierIngredient),
24 ]
25
```

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



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

EXPLORER
  WEBPIZZA
    __init__.py
    admin.py
    apps.py
    forms.py
    models.py
    tests.py
    urls.py
    views.py
  connexion
  webpizza
    __pycache__
    __init__.py
    asgi.py
    settings.py
    urls.py
    wsgi.py
    db.sqlite3
    manage.py
  OUTLINE
  TIMELINE

webpizza > webpizza > urls.py > ...
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, include
18
19 urlpatterns = [
20     path('admin/', admin.site.urls),
21     path('', include('applipizza.urls')),
22 ]
23
```

- 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 connexion, 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 :



```
2 from django.urls import path, include
3 from django.contrib.auth import views as auth_views
4 from connexion import views
5
6 urlpatterns = [
7     path(
8         'login/', auth_views.LoginView.as_view(template_name='connexion/login.html'), name = 'login'),
9 ]
10
11
```

Ce path 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 de connexion.

Pour le moment, les url menant à la connexion admin sont celles contenues dans le path

```
path('admin/', admin.site.urls),
```

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

Modification du template base.html

1. Nous allons inclure un test de connexion dans base.html. Ce teste vérifiera si un utilisateur est identifié.
 - a. Si oui, alors le menu classique est affiché, ainsi qu'un lien de déconnexion,
 - b. Sinon, un lien de connexion est affiché.

Complétez votre code de base.html ainsi :

```
<header>
  <nav>
    {% if user.is_authenticated %}
    <div id="menu">
      <div><a href="/pizzas/">les pizzas</a></div>
      <div><a href="/ingredients/">les ingrédients</a></div>
      <div><a href="/pizzas/add/">créer une pizza</a></div>
      <div><a href="/ingredients/add/">créer un ingrédient</a></div>
    </div>
    <div id="logout">
      bienvenue, {{user}}
      <a href="{% url 'logout' %}">déconnexion</a>
    </div>
    {% else %}
    <div id="login">
      <a href="{% url 'login' %}">connexion</a>
    </div>
    {% endif %}
  </nav>
</header>
```

2. Testez maintenant la connexion :

- connectez-vous, par exemple en tant que super user, grâce à la page par défaut <http://127.0.0.1:8000/admin>
- une fois connecté, allez à <http://127.0.0.1:8000/pizzas/> et vous devriez constater l'évolution de l'interface :

les pizzas – Chromium

les pizzas x +

127.0.0.1:8000/pizzas/

les pizzas les ingrédients créer une pizza créer un ingrédient bienvenue, sgagne déconnexion

voici nos 8 pizzas

- pizza quatre fromages (prix = 15.55 €)	détails	modifier	supprimer
- pizza napolitaine (prix = 14.80 €)	détails	modifier	supprimer
- pizza océane (prix = 15.60 €)	détails	modifier	supprimer
- pizza végétarienne (prix = 14.75 €)	détails	modifier	supprimer
- pizza montagnarde (prix = 14.35 €)	détails	modifier	supprimer
- pizza reine (prix = 14.25 €)	détails	modifier	supprimer
- pizza vésuve (prix = 14.15 €)	détails	modifier	supprimer
- pizza périgourdine (prix = 18.95 €)	détails	modifier	supprimer

Appli Pizza 2023

c. Cliquez sur déconnexion, vous verrez que cela fonctionne (vous n'êtes plus connecté, ce que vous pouvez vérifier en retournant sur l'url admin). Par contre, le template logout n'est pas encore codé, d'où l'erreur qui apparaît pour le moment.

Création des templates login et logout de l'appli connexion

1. Les templates liés à la connexion vont tous hériter d'un template de base propre à l'application connexion, exactement comme pour l'application applipizza.

Créez dans connexion un dossier static/templates, et créez-y un fichier base.html dont le code peut ressembler à :

```
webpizza > connexion > templates > connexion > <> base.html > html
1  {% load static %}
2  <!DOCTYPE html>
3  <html lang="fr">
4      <head>
5          <meta charset="utf-8">
6          <link rel="stylesheet" href="{% static 'applipizza/css/styles.css' %}" />
7          <link rel="icon" href="{% static 'applipizza/img/pizza.PNG' %}" />
8          <title>{% block titre %}{% endblock %}</title>
9      </head>
10     <body>
11         <header>
12
13         </header>
14         <main>
15             {% block contenu %}{% endblock %}
16             <div id="div_image_accueil" >
17                 
18             </div>
19         </main>
20         <footer>
21             Appli Pizza 2023
22         </footer>
23     </body>
24 </html>
```

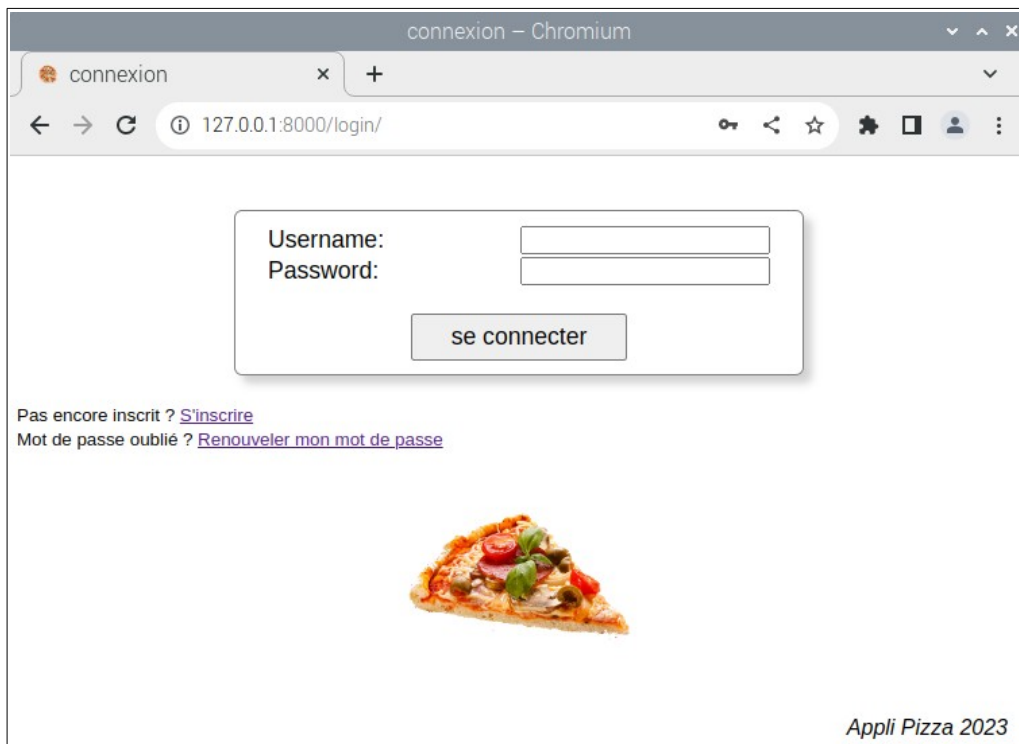
2. créez le fichier login.html avec le code suivant. Ce fichier contient un formulaire pré-conçu dont les éléments sont standard pour un formulaire de connexion : login et mot de passe

webpizza > connexion > templates > connexion > <> login.html > ...

```
1  {% extends 'connexion/base.html' %}
2  {% block titre %}connexion{% endblock %}
3  {% block contenu %}
4    <div>
5      <form id="form_login" method="post">
6        {% csrf_token %}
7        {% if form.non_field_errors %}
8          {% for error in form.non_field_errors %}
9            <p class="alert alert-danger">{{ error }}</p>
10          {% endfor %}
11        {% endif %}
12        <table>
13          {% for field in form %}
14            <tr>
15              <td>{{ field.label_tag }}</td>
16              <td>{{ field }}</td>
17            </tr>
18          {% endfor %}
19        </table>
20        <button type="submit">se connecter</button>
21      </form>
22    </div>
23
24    <div>
25      <small>
26        Pas encore inscrit ?
27        <a href="">
28          S'inscrire
29        </a>
30      <br>
31      Mot de passe oublié ?
32      <a href="">
33        Renouveler mon mot de passe
34      </a>
35    </small>
36  </div>
37  {% endblock %}
```

Pour le moment, les liens pour l'inscription et pour le renouvellement du mot de passe ne sont pas renseignés. Nous verrons cela en temps utile.

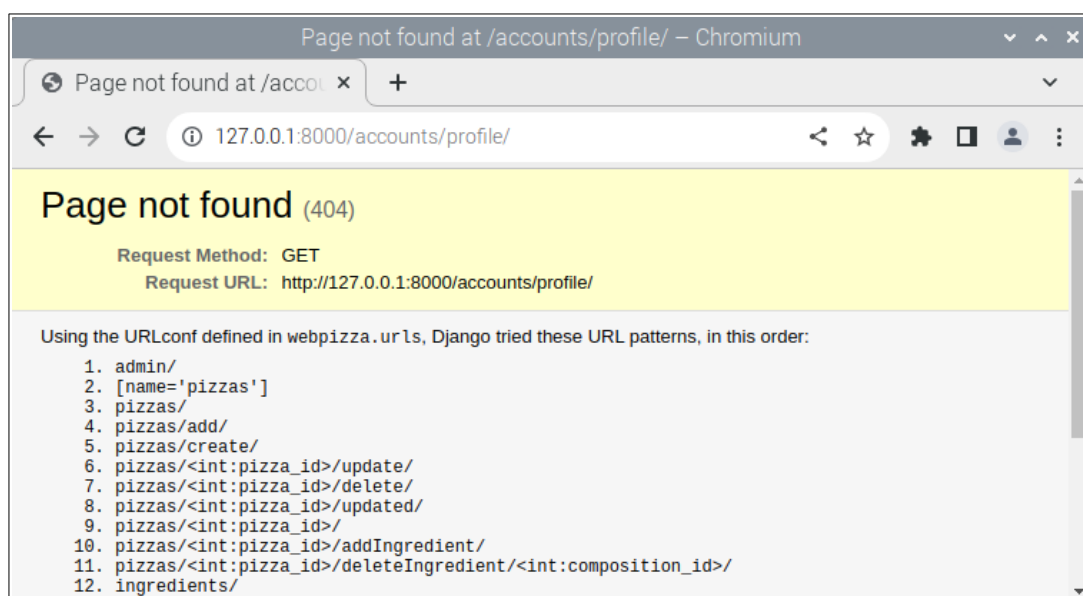
Testez maintenant l'url <http://127.0.0.1:8000/login/> et vérifiez que vous arrivez à la page personnalisée de connexion.



Dans le fichier settings.py, repérez la ligne

`LANGUAGE_CODE = 'en-us'`

3. Si vous changez 'en-us' en 'fr-fr', alors votre page sera personnalisée en français. Si la ligne LANGUAGE_CODE n'existe pas, insérez-la et vérifiez le changement.
4. Maintenant, si vous tentez la connexion, vous aurez une erreur, car par défaut, django vous redirige après connexion vers une page accounts/profile :



5. Pour corriger cela, nous allons forcer la redirection vers la page de présentation des pizzas en deux étapes :

a. dans le fichier settings.py, on indique la nouvelle redirection :

```
LOGIN_REDIRECT_URL = 'pizzas'
```

b. dans le fichier urls.py de applipizza, on ajoute la ligne correspondante (sur la capture, celle dont le path est une chaîne vide) :

```
"""applipizza URL Configuration"""

from django.urls import path
from applipizza import views

urlpatterns = [
    path('', views.pizzas, name = 'pizzas'),

    path('pizzas/', views.pizzas),
    path('pizzas/add/', views.formulaireCreationPizza).
```

Vérifiez que ça fonctionne. En principe, après connexion, vous devez vous retrouver sur la page qui liste les pizzas.

6. Nous allons maintenant ajouter le template logout.html. Très simple, en deux étapes :

a. on ajoute le path dans le fichier urls.py de connexion :

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

b. on crée pour connexion un template logout.html dont le code peut être :

```
{% extends 'connexion/base.html' %}
{% block titre %}au revoir{% endblock %}
{% block contenu %}
<h2>Nous espérons vous revoir bientôt !</h2>
<a href="{% url 'login' %}">Se reconnecter</a>
{% endblock %}
```

Après ces 2 étapes, vérifiez que tout fonctionne.