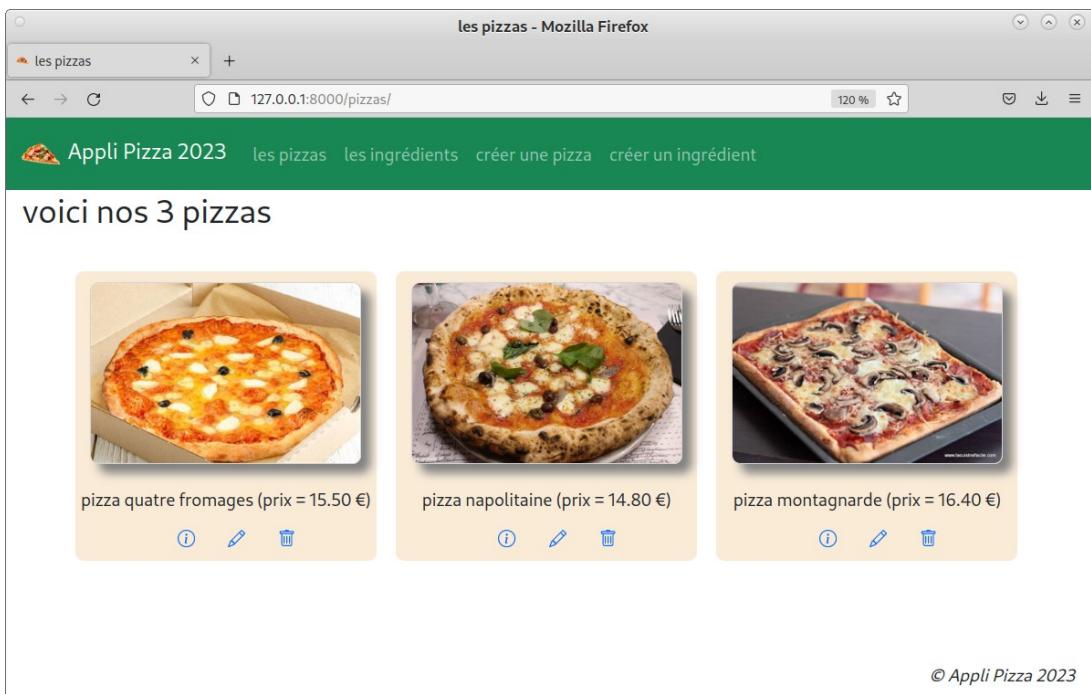


TP10 – Modification du modèle Pizza - Upload d'image

Introduction, Contexte et Objectifs

Le moment est venu d'apporter un élément visuel à notre projet. Le CRUD est opérationnel, mais la liste des pizzas n'est pas vraiment appétissante. Rien ne vaut une belle image de pizza pour nous mettre en appétit :



Dans ce TP nous allons, étape par étape, arriver à intégrer une image au modèle de Pizza.

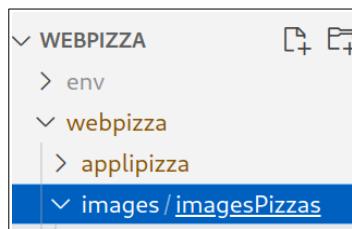
Cette image sera dynamique : au moment de la création d'une pizza ou de sa modification, l'image sera à renseigner ou à modifier dans le formulaire dédié.

Nous allons donc faire évoluer le modèle Pizza pour ajouter un attribut de type fichier (ImageField).

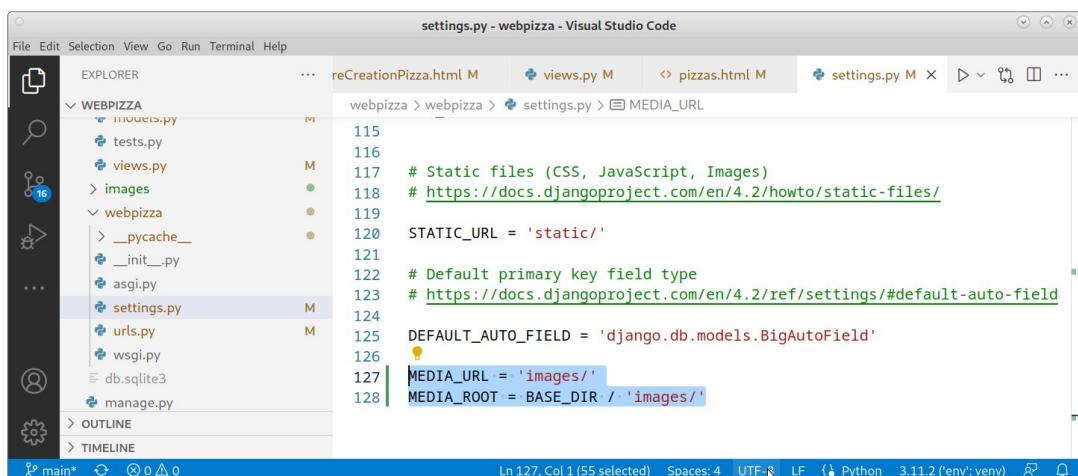
Nous allons aussi donner un aspect plus pratique à la liste des pizzas. L'image de la pizza sera cliquable pour accéder aux détails, et nous intégrerons les icônes bootstrap habituels.

Le répertoire de stockage des images

- Créez, au niveau de la racine webpizza du projet, un répertoire « images » puis un sous-répertoire imagesPizzas qui contiendra les images qui seront téléchargées (« uploadées ») par l'utilisateur plus tard :



- Dans le fichier settings.py, ajoutez les lignes suivantes, qui indiquent le répertoire des fichiers téléchargés :



```

# Static files (CSS, JavaScript, Images)
MEDIA_URL = 'https://docs.djangoproject.com/en/4.2/howto/static-files/'

STATIC_URL = 'static/'

# Default primary key field type
# https://docs.djangoproject.com/en/4.2/ref/settings/#default-auto-field
DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'

MEDIA_URL = 'images/'
MEDIA_ROOT = BASE_DIR / 'images/'
  
```

- Dans le fichier settings.py, ajoutez à la variable TEMPLATES la dernière ligne de la capture suivante. Elle permet aux templates d'accéder aux fichiers images (notamment) :

```

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
                'django.template.context_processors.media'
            ],
        },
    },
]
  
```

- Dans le fichier urls.py, ajoutez les deux import suivants :

```
from django.conf import settings
from django.conf.urls.static import static
```

Ces deux import permettent l'ajout qui suit :

- Ajoutez à la fin du fichier urls.py la ligne suivante :

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

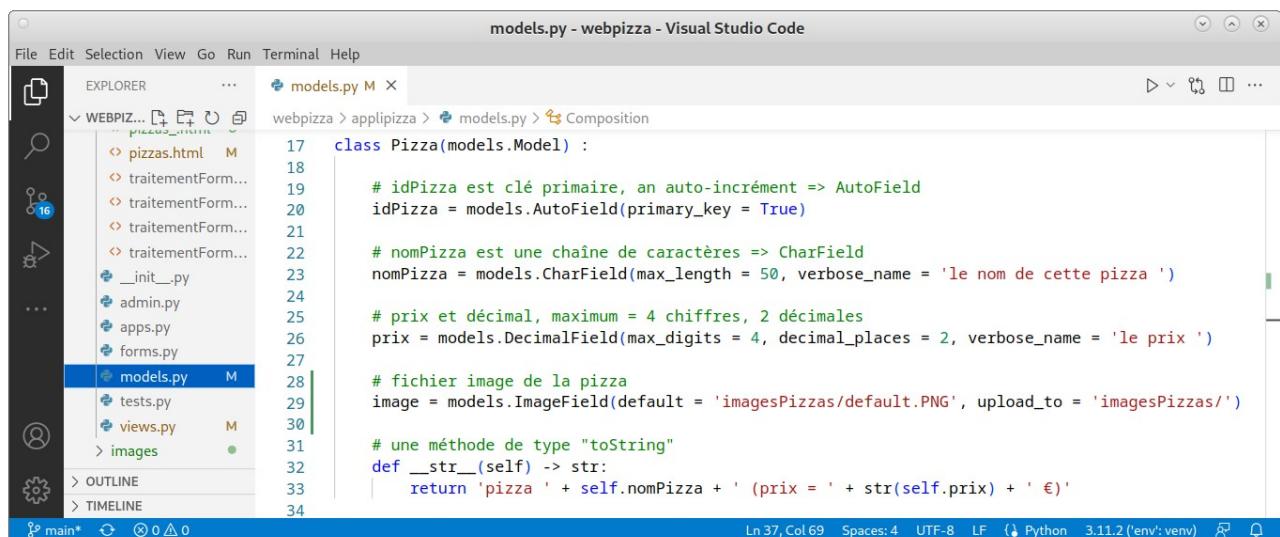
Les modifications précédentes vont permettre d'afficher les images des pizzas.

- Ajoutez, dans un répertoire quelconque (par exemple sur le Bureau) les images de pizzas à votre disposition sur Moodle. Ne les mettez pas d'avance dans images/imagesPizzas, pour que l'on constate bien qu'elles seront uploadées. Tout est alors prêt pour aller chercher les images au moment de la création ou de la modification.

Modification du modèle Pizza dans models.py

Dans cette partie, nous incorporons au modèle Pizza un nouvel attribut image qui indiquera le nom du fichier à afficher.

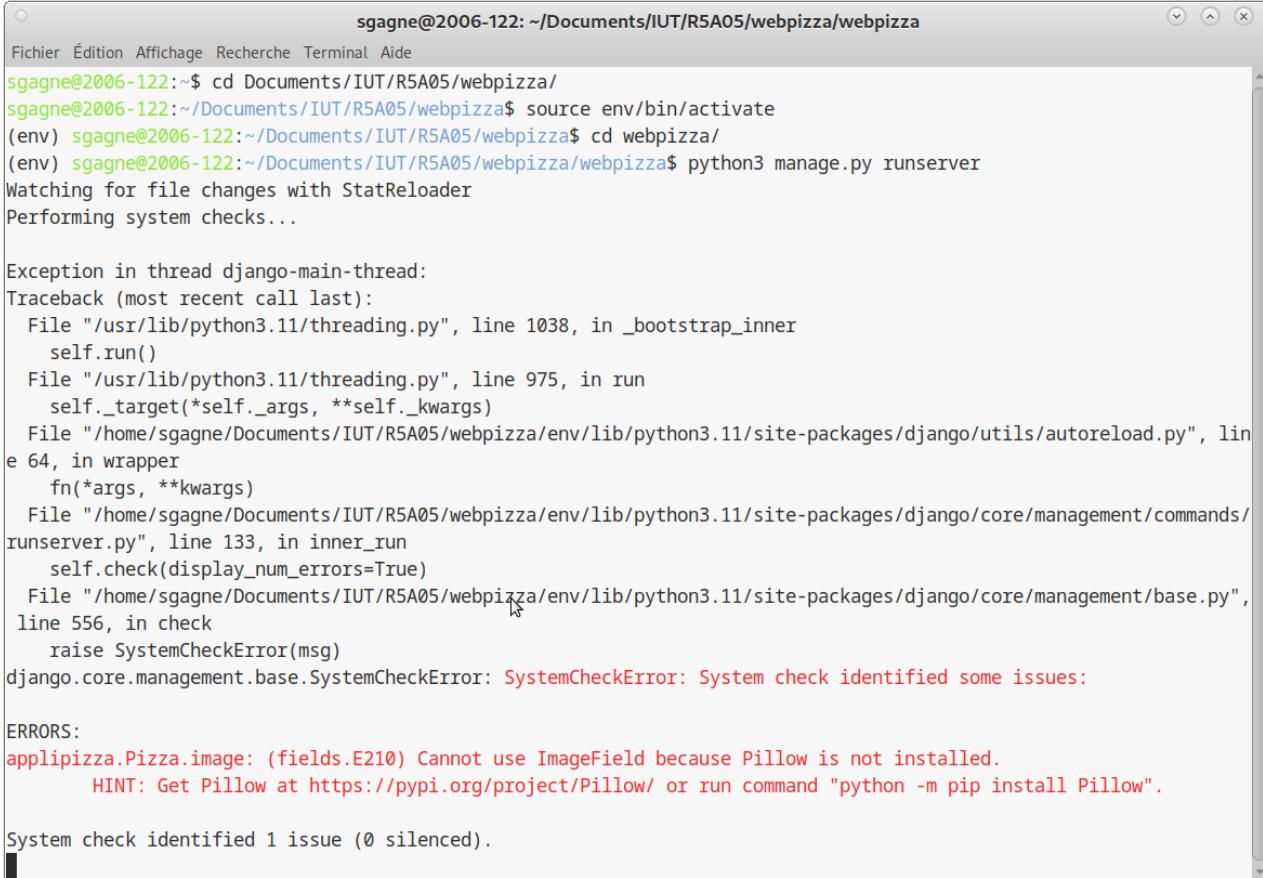
- Créez, dans le modèle Pizza, un attribut de type ImageField :



```
models.py - webpizza - Visual Studio Code
File Edit Selection View Go Run Terminal Help
models.py M
webpizza > applipizza > models.py > Composition
17 class Pizza(models.Model):
18     # idPizza est clé primaire, an auto-incrément => AutoField
19     idPizza = models.AutoField(primary_key = True)
20
21     # nomPizza est une chaîne de caractères => CharField
22     nomPizza = models.CharField(max_length = 50, verbose_name = 'le nom de cette pizza ')
23
24     # prix et décimal, maximum = 4 chiffres, 2 décimales
25     prix = models.DecimalField(max_digits = 4, decimal_places = 2, verbose_name = 'le prix ')
26
27     # fichier image de la pizza
28     image = models.ImageField(default = 'imagesPizzas/default.PNG', upload_to = 'imagesPizzas/')
29
30     # une méthode de type "toString"
31     def __str__(self) -> str:
32         return 'pizza ' + self.nomPizza + ' (prix = ' + str(self.prix) + ' €)'
```

Ln 37, Col 69 Spaces:4 UTF-8 LF Python 3.11.2 ('env:venv')

2. Tentez de lancer le serveur web. Vous devriez avoir une erreur :



```

sgagne@2006-122: ~/Documents/IUT/R5A05/webpizza/webpizza
Fichier Édition Affichage Recherche Terminal Aide
sgagne@2006-122:~$ cd Documents/IUT/R5A05/webpizza/
sgagne@2006-122:~/Documents/IUT/R5A05/webpizza$ source env/bin/activate
(env) sgagne@2006-122:~/Documents/IUT/R5A05/webpizza$ cd webpizza/
(env) sgagne@2006-122:~/Documents/IUT/R5A05/webpizza/webpizza$ python3 manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

Exception in thread django-main-thread:
Traceback (most recent call last):
  File "/usr/lib/python3.11/threading.py", line 1038, in _bootstrap_inner
    self.run()
  File "/usr/lib/python3.11/threading.py", line 975, in run
    self._target(*self._args, **self._kwargs)
  File "/home/sgagne/Documents/IUT/R5A05/webpizza/env/lib/python3.11/site-packages/django/utils/autoreload.py", line 64, in wrapper
    fn(*args, **kwargs)
  File "/home/sgagne/Documents/IUT/R5A05/webpizza/env/lib/python3.11/site-packages/django/core/management/commands/runserver.py", line 133, in inner_run
    self.check(display_num_errors=True)
  File "/home/sgagne/Documents/IUT/R5A05/webpizza/env/lib/python3.11/site-packages/django/core/management/base.py", line 556, in check
    raise SystemCheckError(msg)
django.core.management.base.SystemCheckError: System check identified some issues:

ERRORS:
applipizza.Pizza.image: (fields.E210) Cannot use ImageField because Pillow is not installed.
    HINT: Get Pillow at https://pypi.org/project/Pillow/ or run command "python -m pip install Pillow".

System check identified 1 issue (0 silenced).

```

Cette erreur vient du fait que les données de type ImageField nécessitent l'installation de la bibliothèque Pillow. Nous allons donc l'installer :



```

sgagne@2006-122: ~/Documents/IUT/R5A05/webpizza
Fichier Édition Affichage Recherche Terminal Aide
(env) sgagne@2006-122:~/Documents/IUT/R5A05/webpizza$ pip install Pillow
Collecting Pillow
  Downloading Pillow-9.5.0-cp311-cp311-manylinux_2_28_x86_64.whl (3.4 MB)
    3.4/3.4 MB 11.8 MB/s eta 0:00:00

Installing collected packages: Pillow
Successfully installed Pillow-9.5.0
(env) sgagne@2006-122:~/Documents/IUT/R5A05/webpizza$ 

```

3. Relancez maintenant le serveur web. Si tout se passe bien, c'est que la bibliothèque est opérationnelle et que nous allons pouvoir gérer les fichiers image.

Migration de Pizza

- Procédez maintenant à la migration qui permettra de tenir compte de la modification du modèle Pizza :

```
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/0003_pizza_image.py
        - Add field image to 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.0003_pizza_image... OK
(env) sgagne@2006-122:~/Documents/IUT/R5A05/webpizza/webpizza$
```

- Vérifiez, dans sqlitebrowser, que tout a été pris en compte :

idPizza	nomPizza	prix	image	
			Filtre	Filtre
1	6 napolitaine	14.8	imagesPizzas/default.PNG	
2	7 quatre fromages	15.5	imagesPizzas/default.PNG	

- Déplacez à la main, dans images/imagesPizzas, les images qui vous intéressent pour ces pizzas uniquement (en quelque sorte, nous faisons un upload à la main).
- Changez ensuite à la main, dans sqlitebrowser, la valeur du champ image de chaque pizza en lui donnant le nom du fichier correspondant.

Table : applipizza_pizza

idPizza	nomPizza	prix	image	
			Filtre	Filtre
1	9 etna	16.1	imagesPizzas/etna.PNG	
2	10 quatre fromages	14.3	imagesPizzas/quatre_fromages.PNG	
3	11 napolitaine	15.5	imagesPizzas/napolitaine.PNG	

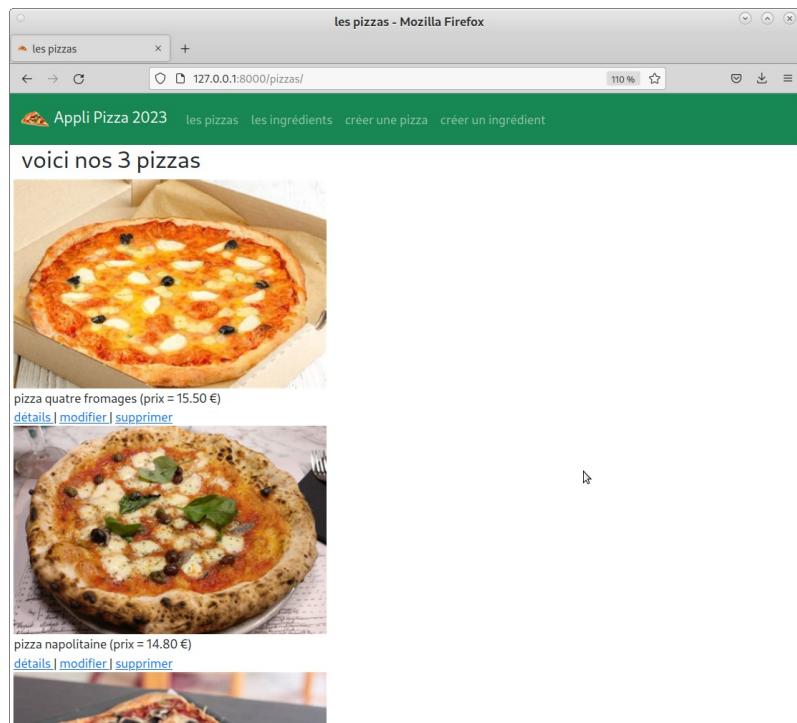
- Pensez à enregistrer les modifications dans sqlitebrowser.

Modification du template pizzas.html

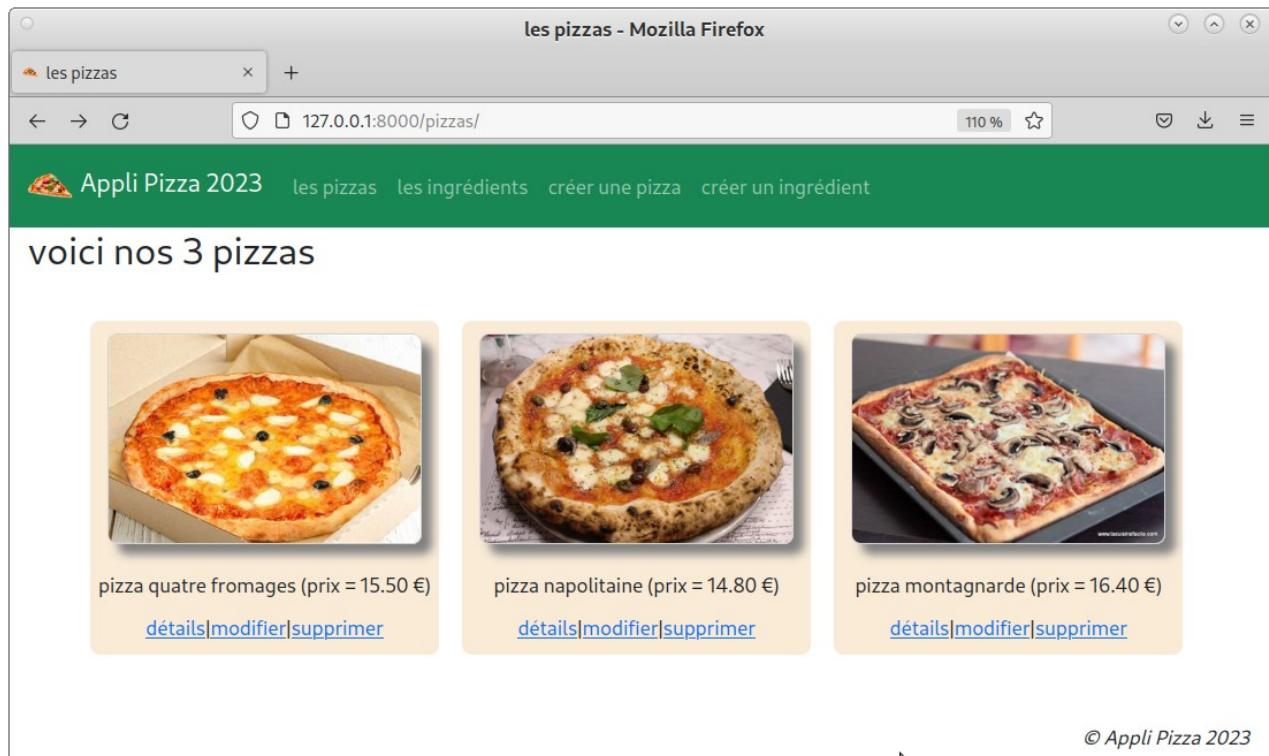
- Dans le template pizzas.html, faites en sorte que l'affichage d'une pizza incorpore l'image en ajoutant une balise image :


```
pizzas.html M X
webpizza > applipizza > templates > applipizza > pizzas.html > div > div > div
1   {% extends 'applipizza/base.html' %}
2
3   {% block title %}
4     les pizzas
5   {% endblock %}
6
7   {% block main %}
8     <h2>voici nos {{pizzas|length}} pizzas</h2>
9     <div>
10    {% for piz in pizzas %}
11      <div>
12        <div>
13          <a href="/pizzas/{{piz.idPizza}}/">
14            
15          </a>
16        </div>
17        <div>{{piz}}</div>
18        <div>
19          <a href="/pizzas/{{piz.idPizza}}/"> détails </a> |
20          <a href="/pizzas/{{piz.idPizza}}/update/"> modifier </a> |
21          <a href="/pizzas/{{piz.idPizza}}/delete/"> supprimer </a>
22        </div>
23      {% endfor %}
24    </div>
25  {% endblock %}
```

Si tout s'est bien passé, vous devriez avoir un rendu à peu près similaire à :



- Dotez les différentes balises div de classes css dont vous écrirez vous-même le code pour qu'avec un peu de technologie flex, vous arriviez au résultat suivant :



- Enfin, en utilisant les icônes bootstrap (vous pouvez les récupérer sur la page <https://icons.getbootstrap.com/>), remplacez les liens textuels par des liens plus intuitifs comme le suggère la toute première capture du TP.

Mise à jour du formulaire de création d'une pizza

Dans le formulaire du template pizza.html, il va y avoir plusieurs changements importants que nous allons détailler :

```
webpizza > applipizza > templates > applipizza > formulaireCreationPizza.html > ...
1  {% extends 'applipizza/base.html' %}
2
3  {% block title %}
4  création d'une pizza
5  {% endblock %}
6
7  {% block main %}
8  <h2>création d'une pizza</h2>
9  <form enctype="multipart/form-data" action="/pizzas/create/" method="post">
10    {% csrf_token %}
11    <div class="mb-3">
12      <label for="id_nomPizza" class="form-label">Le nom de cette pizza </label>
13      <input type="text" name="nomPizza" class="form-control" maxlength="50" required id="id_nomPizza">
14    </div>
15    <div class="mb-3">
16      <label for="id_prix" class="form-label">Le prix de cette pizza </label>
17      <input type="number" name="prix" value="0" min="0" step=".01" class="form-control" required id="id_prix">
18    </div>
19    <div class="mb-3">
20      <label for="id_image" class="form-label">Fichier image</label>
21      <input class="form-control" name="image" type="file" accept="image/*" id="id_image">
22    </div>
23    <button type="submit" class="btn btn-primary">Envoyer</button>
24  </form>
25  {% endblock %}
```

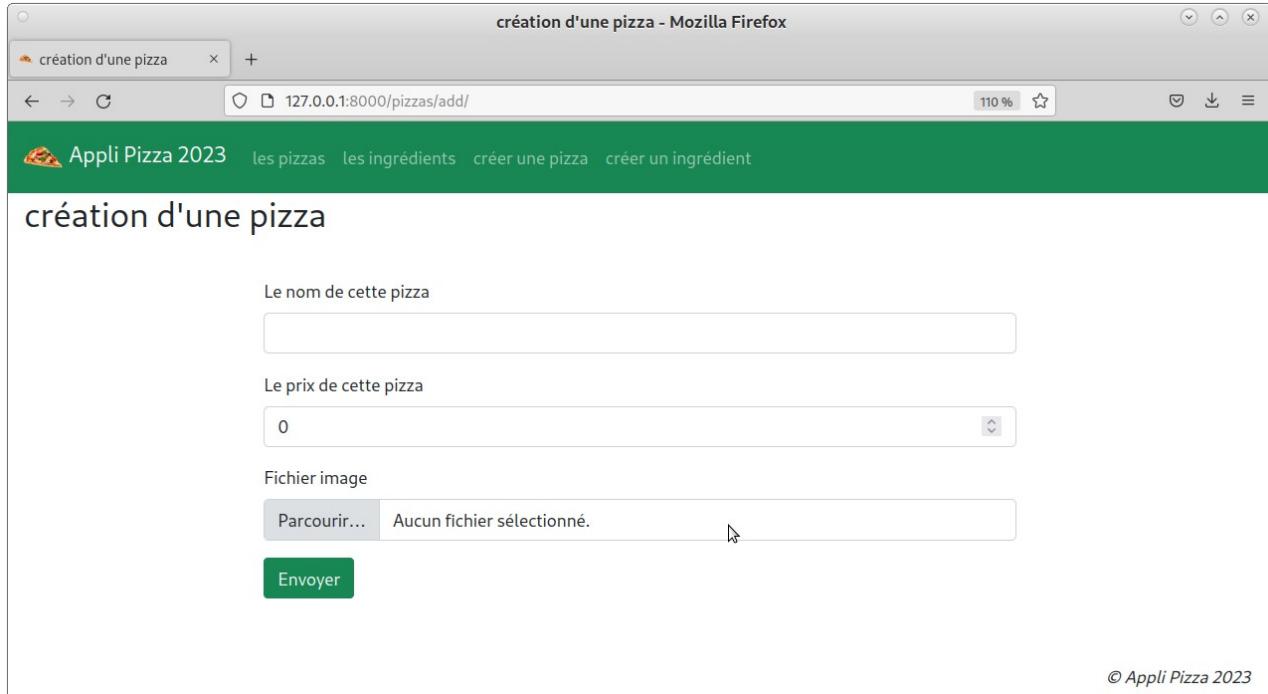
1. insérez les nouvelles balises correspondant à la div de sélection de l'image, encadrée en rouge. Notez que le type est « file » et que le name de l'input est « image ».

Ce name permettra plus tard, dans le traitement du formulaire, de récupérer le fichier sous la forme request.FILES['image'].

2. Modifiez la balise form pour lui permettre de gérer l'envoi de fichiers (voir encadré vert).

Cette modification interviendra également dans le formulaire de modification (voir plus tard).

3. Testez votre nouveau formulaire. Vous devriez avoir un rendu similaire à :



The screenshot shows a Mozilla Firefox window with the title "création d'une pizza - Mozilla Firefox". The address bar shows "127.0.0.1:8000/pizzas/add/". The page header includes the logo "Appli Pizza 2023" and links for "les pizzas", "les ingrédients", "créer une pizza", and "créer un ingrédient". The main content area is titled "création d'une pizza" and contains three input fields: "Le nom de cette pizza" (empty), "Le prix de cette pizza" (value "0"), and "Fichier image" (button "Parcourir...", message "Aucun fichier sélectionné.", and a file input field). A green "Envoyer" button is at the bottom. The footer of the page says "© Appli Pizza 2023".

Mise à jour de la view creerPizza dans views.py

- Modifiez la view creerPizza pour qu'elle tienne compte du nouvel élément posté :

```
def creerPizza(request) :
    form = PizzaForm(request.POST, request.FILES)

    if form.is_valid() :
        piz = Pizza()

        piz.nomPizza = form.cleaned_data['nomPizza']
        piz.prix = form.cleaned_data['prix']
        piz.image = request.FILES['image']

        piz.save()

    return render(
        request,
        "applipizza/traitementFormulaireCreationPizza.html",
        {"nom" : piz.nomPizza}
    )
```

- Testez que tout fonctionne en créant une nouvelle pizza. Vérifiez, dans votre architecture de fichiers, que la fichier

l'image a bien été uploadé et que son nom a bien été renseigné dans la base de données.

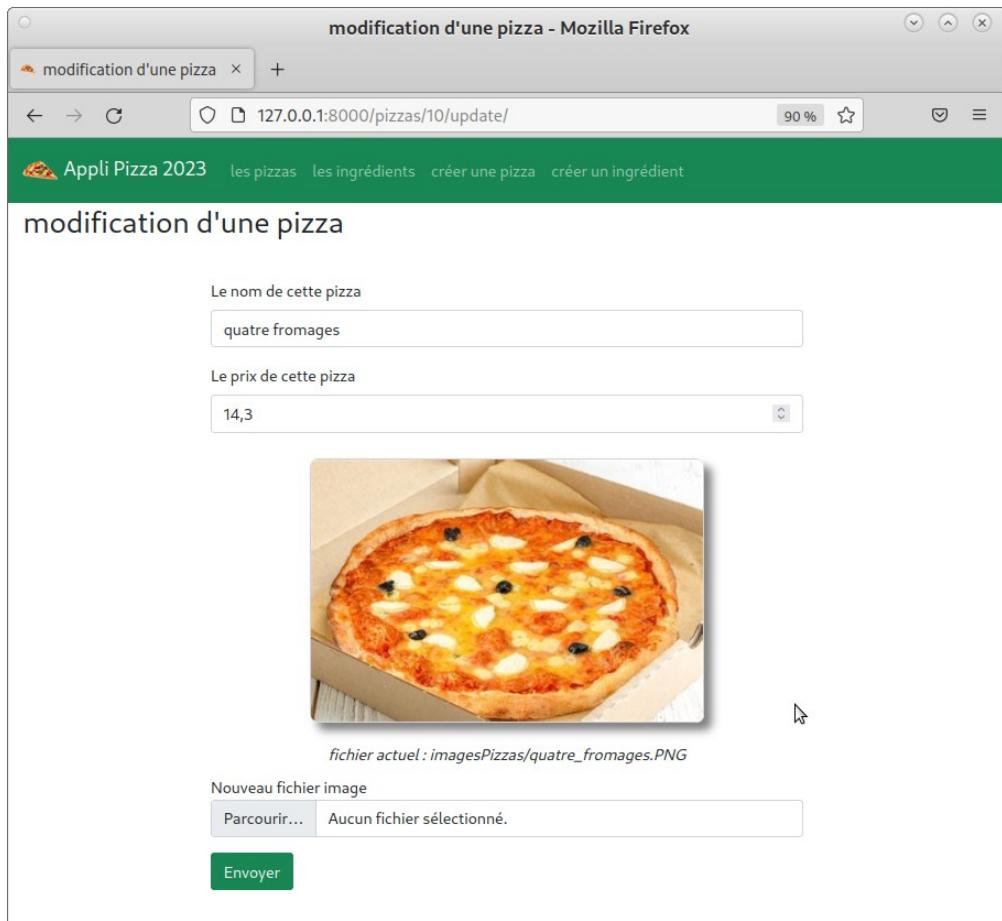
Mise à jour du formulaire de modification d'une pizza

Les modifications apportées au formulaire de création vont être également apportées à celui de la modification.

- Modifiez la balise form pour permettre l'envoi de fichiers.
- Ajoutez un champ « input » permettant de sélectionner un fichier, comme pour la création.

```
<div class="mb-3">
  <div class="divImgPizza">
    
  </div>
  <div class="divNomPizza"><i>fichier actuel : {{pizza.image}}</i></div>
  Nouveau fichier image
  <input class="form-control" name="image" type="file" accept="image/*" id="id_image">
</div>
```

- rendu possible :



Mise à jour de la view modifierPizza dans views.py

Dans la view modifierPizza, ajoutez les modifications de la capture suivante :

- insérez le paramètre request.FILES dans la récupération du formulaire,
- insérez la ligne qui met à jour l'attribut image de la pizza,
- vérifiez que tout fonctionne.

```
def modifierPizza(request, pizza_id) :  
    lapizza = Pizza.objects.get(idPizza = pizza_id)  
    form = PizzaForm(request.POST, request.FILES, instance = lapizza)  
  
    if form.is_valid() :  
        lapizza.image = request.FILES['image']  
        lapizza.save()  
        lapizza = Pizza.objects.get(idPizza = pizza_id)  
  
    return render(  
        request,  
        "applipizza/traitementFormulaireModificationPizza.html",  
        {"pizza" : lapizza}  
    )
```

Complément CSS

Si ce n'est pas déjà fait, stylisez un peu la liste des ingrédients pour avoir quelque chose de plus « bootstrap » :

les ingrédients - Mozilla Firefox

les ingrédients +

127.0.0.1:8000/ingredients/ Importer les marque-pages... Débuter avec Firefox IUT web rigaudie.fr DoctoPrism C'est le moment de vo...

Appli Pizza 2023 les pizzas créer une pizza les ingrédients créer un ingrédient

voici nos 23 ingrédients

ingrédient	action
champignons	 
anchois	 
jambon	 
œuf	 
gorgonzola	 
fromage de chèvre	 
parmesan	 
mozzarella	 
crème fraîche	 
oignon	 
basilic	 