

TP6 – Premiers formulaires

Introduction, Contexte et Objectifs

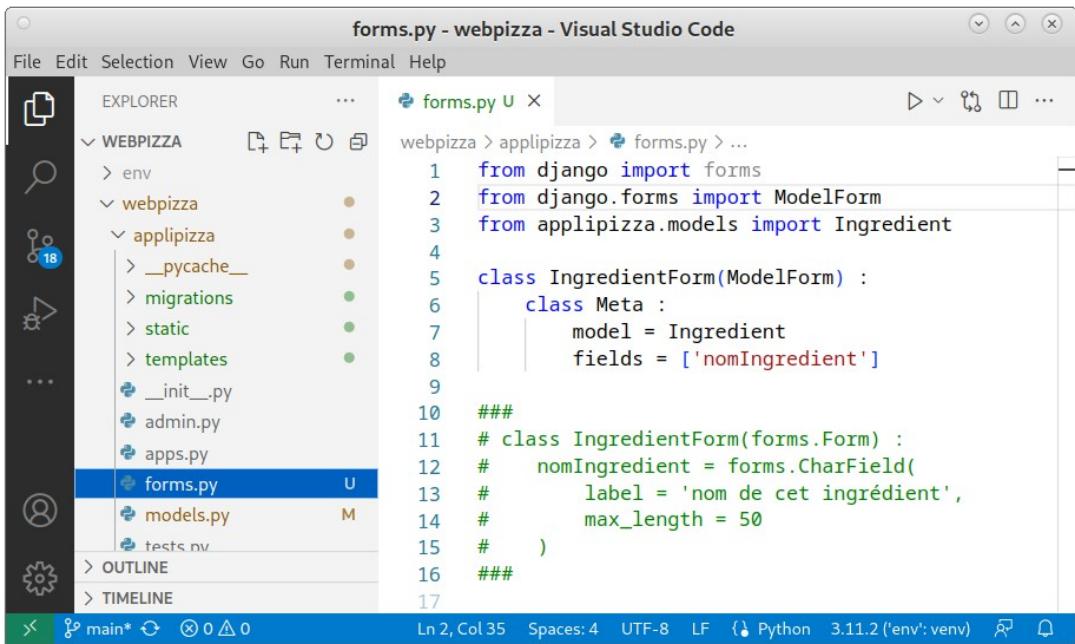
Pour le moment nous accédons à la base de données, et nous consultons les données, pour affichage. C'est de la « lecture seule ». L'idée est ici de :

- construire un formulaire pour récupérer les données de futurs ingrédients,
- traiter l'envoi du formulaire et ainsi avoir une action d'écriture sur la base de données. Cette action se fera sans être connecté au site. Autrement dit, tout le monde est « administrateur ». Nous verrons la connexion plus tard.

Il y a une méthode complètement automatique pour créer un formulaire adapté aux attributs d'une classe. Nous allons l'utiliser pour les formulaires de création d'une pizza et d'un ingrédient.

Création du fichier forms.py

1. Au même niveau que views.py, créez un fichier forms.py. Nous allons y définir notre formulaire.
2. Dans ce fichier, importez la bibliothèque forms de django, puis de forms importez ModelForm, qui permet les formulaires automatiques. Créez la classe IngredientForm, qui hérite de ModelForm, et qui représentera le formulaire correspondant à un nouvel ingrédient. Le code, donné juste après, est vraiment des plus simples. La classe Meta fait le travail. En commentaires, la version non-automatique, où on déclare les champs équivalents à ceux de la classe.



```

forms.py - webpizza - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER ... forms.py U X
WEBPIZZA ...
    > env
    > webpizza
        > __pycache__
        > migrations
        > static
        > templates
        > __init__.py
        > admin.py
        > apps.py
        > forms.py U
        > models.py M
        > tests.py
    > OUTLINE
    > TIMELINE
Ln 2, Col 35  Spaces: 4  UTF-8  LF  { Python  3.11.2 ('env': venv)  ⚙  ⚡

```

```

from django import forms
from django.forms import ModelForm
from applipizza.models import Ingredient

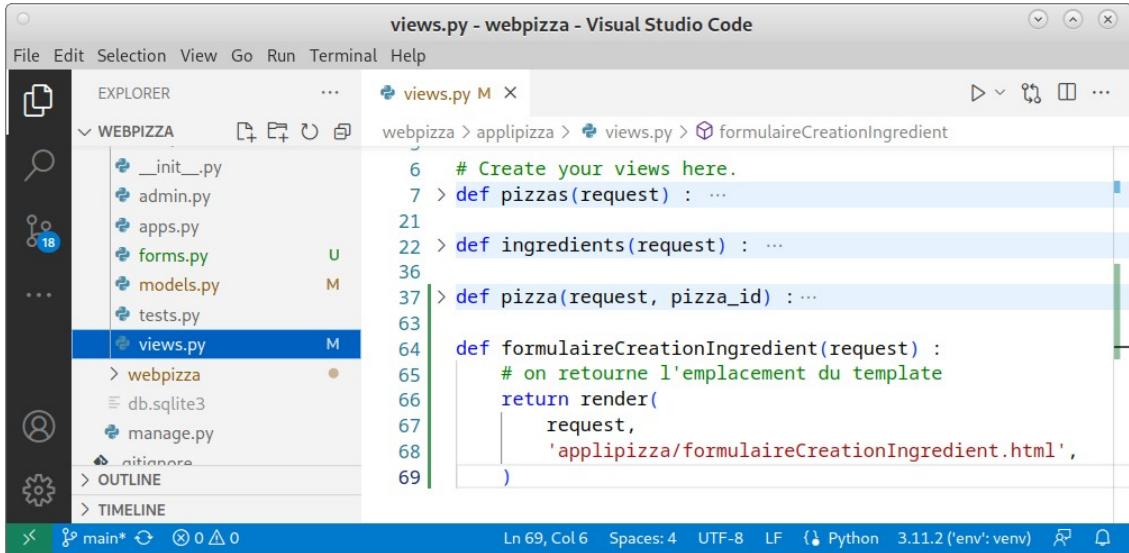
class IngredientForm(ModelForm):
    class Meta :
        model = Ingredient
        fields = ['nomIngredient']

    #####
    # class IngredientForm(forms.Form) :
    #     nomIngredient = forms.CharField(
    #         label = 'nom de cet ingrédient',
    #         max_length = 50
    #     )
    #####

```

Création d'une view dans le fichier views.py

- Créez dans views.py la view formulaireCreationIngredient qui retourne un render qui indique un template (pas encore créé) **applipizza/formulaireCreationIngredient.html** :



```

views.py - webpizza - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER ... views.py M X
WEBPIZZA ...
    > __init__.py
    > admin.py
    > apps.py
    > forms.py U
    > models.py M
    > tests.py
    > views.py M
    > webpizza
        > db.sqlite3
        > manage.py
        > __init__.py
    > OUTLINE
    > TIMELINE
Ln 69, Col 6  Spaces: 4  UTF-8  LF  { Python  3.11.2 ('env': venv)  ⚙  ⚡

```

```

# Create your views here.
> def pizzas(request) : ...
21
> def ingredients(request) : ...
36
> def pizza(request, pizza_id) :...
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69

```

2. Dans le fichier urls.py, définissez un path qui permet d'accéder à ce formulaire. Ce path pourra être de la forme ingredients/add et appellera la « vue » précédente.

The screenshot shows the Visual Studio Code interface with the following details:

- Title Bar:** urls.py - webpizza - Visual Studio Code
- File Menu:** File Edit Selection View Go Run Terminal Help
- Explorer Bar (Left):** Shows the project structure under WEBPIZZA:
 - WEBPIZZA
 - webpizza
 - __pycache__
 - __init__.py
 - asgi.py
 - settings.py
 - urls.py (highlighted)
 - wsgi.py
 - db.sqlite3
- Editor Area (Right):** The code for urls.py is displayed:

```
from django.contrib import admin
from django.urls import path
from applipizza import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('pizzas/', views.pizzas),
    path('ingredients/', views.ingredients),
    path('pizzas/<int:pizza_id>/', views.pizza),
    path('ingredients/add/', views.formulaireCreationIngredient)
]
```
- Bottom Status Bar:** Ln 26, Col 64 | Spaces: 4 | UTF-8 | LF | Python 3.11.2 ('env': venv) | ⚙️

3. Dans le répertoire template/applipizza, créez un fichier formulaireCreationIngredient.html qui aura la même base que les autres :

formulaireCreationIngredient.html - webpizza - Visual Studio Code

File Edit Selection View Go Run Terminal Help

EXPLORER

WEBPIZZA

- # styles.css
- > img
- templates / applipizza
 - base.html
 - footer.html
 - formulaireCreationIngredient.html
 - ingredients.html
 - menu.html
 - pizza.html
 - pizzas.html
- __init__.py
- admin.py
- apps.py
- forms.py
- models.py
- tests.py
- views.py
- webpizza
 - __pycache__
 - __init__.py
 - asgi.py
 - settings.py
 - urls.py
 - wsgi.py
- db.sqlite3
- manage.py
- .gitignore
- README.md

formulaireCreationIngredient.html U

```
webpizza > applipizza > templates > applipizza > formulaireCreationIngredient.html > form
1  {% extends 'applipizza/base.html' %}

2
3  {% block title %}
4      création d'un ingrédient
5  {% endblock %}

6
7  {% block main %}
8      <h2>création d'un ingrédient</h2>
9      <form action="#" method="post">
10         {% csrf_token %}
11         <div class="mb-3">
12             <label for="id_nomIngredient" class="form-label">Le nom de cet ingrédient </label>
13             <input type="text" name="nomIngredient" class="form-control" maxlength="50" required id="id_nomIngredient">
14         </div>
15         <button type="submit" class="btn btn-primary">Envoyer</button>
16     </form>
17  {% endblock %}
```

OUTLINE

TIMELINE

NO NEW NOTIFICATIONS

Ln 9, Col 16 Spaces: 4 UTF-8 LF HTML

On y retrouve le code html d'un formulaire avec des classes bootstrap.

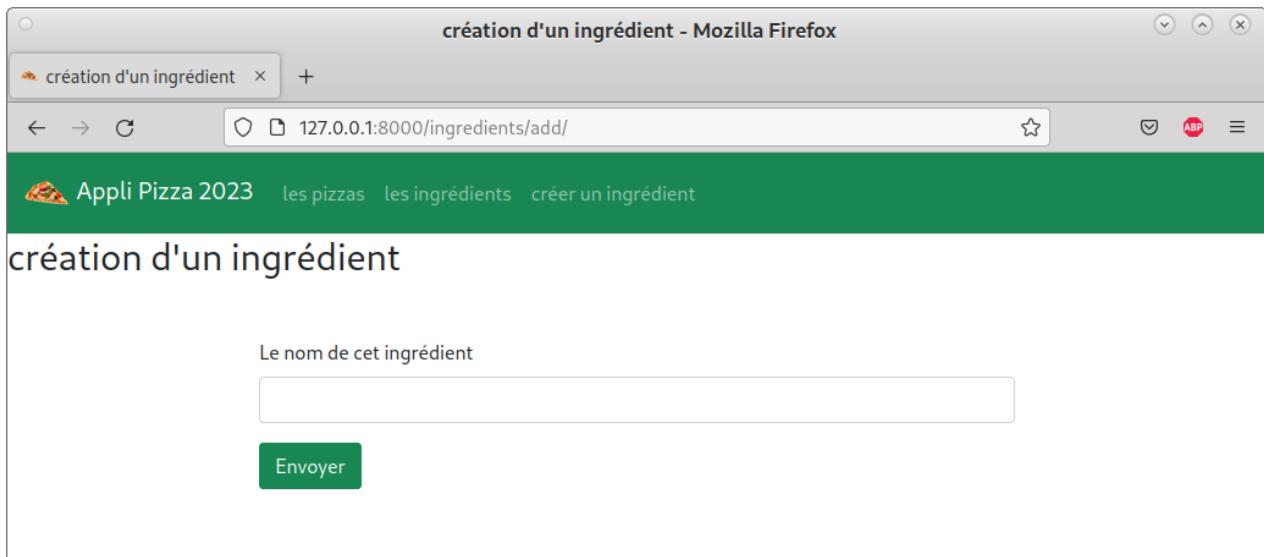
Le « csrf_token » est un « jeton » qui sert à lutter contre les failles CSRF. Nous pouvons laisser cela de côté, ce n'est pas notre thème.

<https://www.leblogduhacker.fr/faille-csrf-explications-contre-mesures/>

Pour le moment, on ne renseigne pas l'action du formulaire. Plus tard, l'action sera la création du nouvel ingrédient.

Si tout est bien fait, l'url 127.0.0.1:8000/ingredients/add/ doit amener au formulaire, qu'on peut remplir, mais dont l'envoi ne donnera rien, puisque le traitement du formulaire n'a pas encore été programmé.

Pensez à ajouter un lien dans le menu pour amener au formulaire :



L'action lancée suite à l'envoi du formulaire

1. Créez dans views.py la view creerIngredient qui va gérer le traitement des informations reçues via le formulaire :

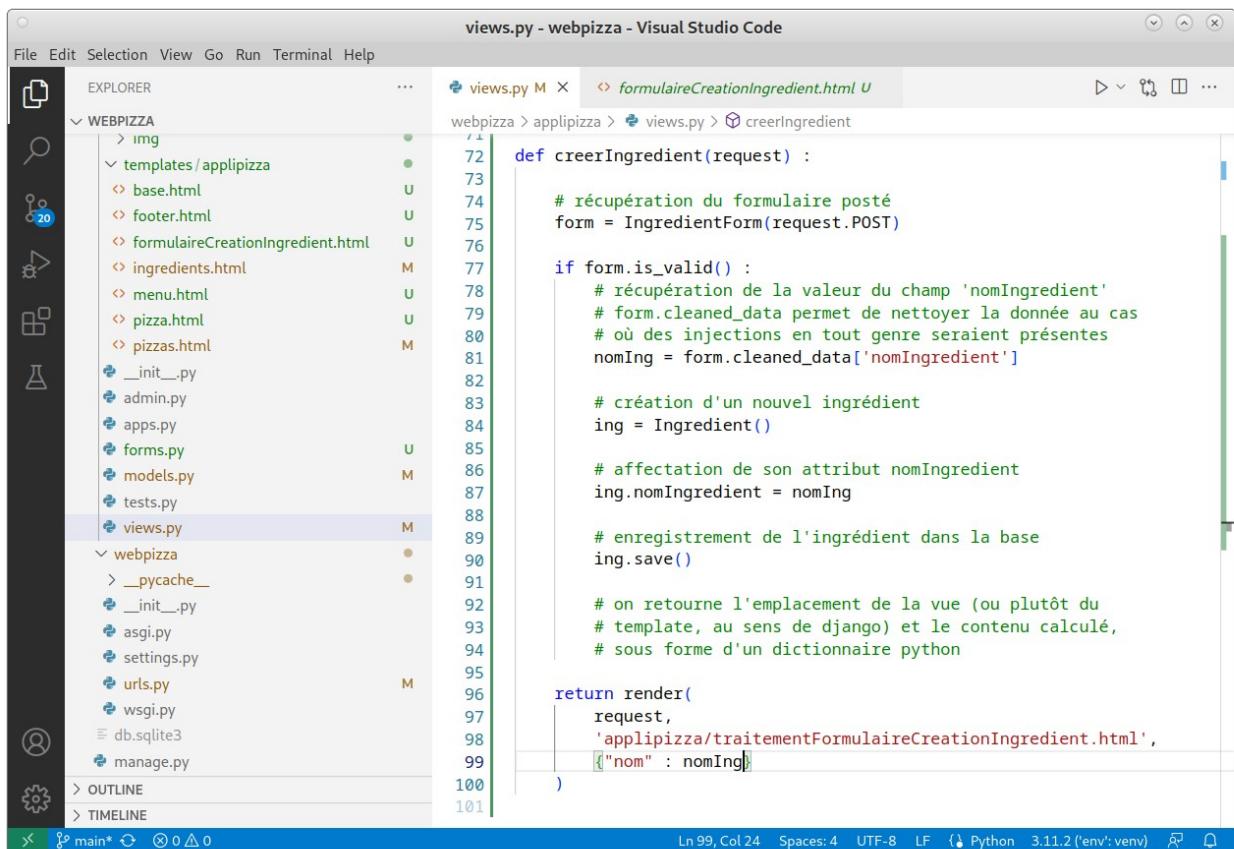
a. on récupère le formulaire :

```
form = IngredientForm(request.POST)
```

- b. si le formulaire est « valide » (vérification automatique) alors on récupère le nom de l'ingrédient par :

```
nomIng = form.cleaned_data['nomIngredient']
```

- c. puis on crée une instance de la classe Ingredient, on lui donne comme nomIngredient la valeur récupérée, et on sauve cet Ingredient dans la base.
- d. Enfin, on retourne un render vers le template nommé applipizza/traitementFormulaireCreationIngredient.html, avec comme contenu le nom récupéré (template pas encore créé).



```
views.py - webpizza - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
WEBPIZZA
    > img
    < templates / applipizza
        > base.html
        > footer.html
        > formulaireCreationIngredient.html
        > ingredients.html
        > menu.html
        > pizza.html
        > pizzas.html
    < __init__.py
    < admin.py
    < apps.py
    < forms.py
    < models.py
    < tests.py
    < views.py
    < webpizza
        > __pycache__
            < __init__.py
            < asgi.py
            < settings.py
            < urls.py
            < wsgi.py
        < db.sqlite3
        < manage.py
    > OUTLINE
    > TIMELINE
views.py M X formulaireCreationIngredient.html U
def creerIngredient(request):
    # récupération du formulaire posté
    form = IngredientForm(request.POST)

    if form.is_valid():
        # récupération de la valeur du champ 'nomIngredient'
        # form.cleaned_data permet de nettoyer la donnée au cas
        # où des injections en tout genre seraient présentes
        nomIng = form.cleaned_data['nomIngredient']

        # création d'un nouvel ingrédient
        ing = Ingredient()

        # affectation de son attribut nomIngredient
        ing.nomIngredient = nomIng

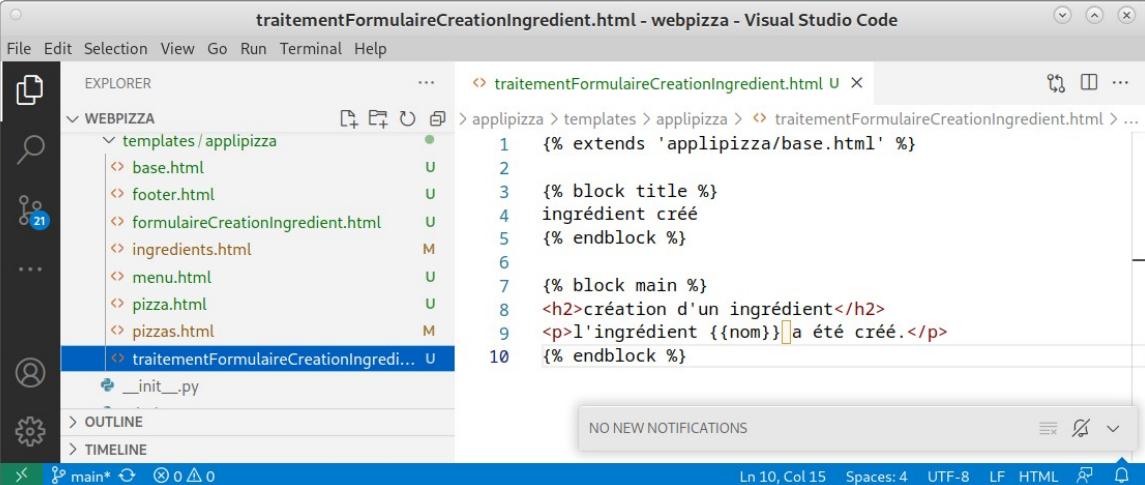
        # enregistrement de l'ingrédient dans la base
        ing.save()

        # on retourne l'emplacement de la vue (ou plutôt du
        # template, au sens de django) et le contenu calculé,
        # sous forme d'un dictionnaire python

    return render(
        request,
        'applipizza/traitementFormulaireCreationIngredient.html',
        {"nom": nomIng}
    )
Ln 99, Col 24  Spaces: 4  UTF-8  LF  (Python 3.11.2 ('env':venv))  ⌂ ⌂ ⌂
```

2. Dans le fichier urls.py, définissez un path qui permet d'accéder à la page annonçant que le traitement a bien été effectué. Ce path sera de la forme ingredients/create/ et appellera la view précédente.
3. Dans le template formulaireCreationIngredient.html, indiquez maintenant la bonne action : /ingredients/create/

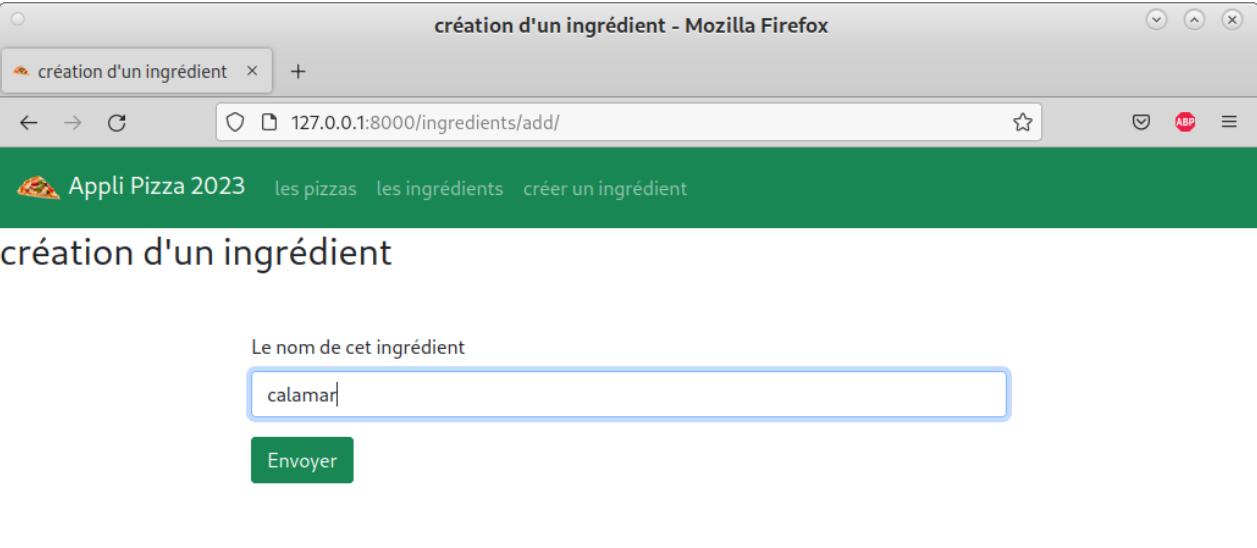
4. Créez traitementFormulaireCreationIngredient.html, template qui hérite de base.html, comme tous les autres, et prévient juste que la création a été effectuée, et utilisant la variable de gabarit {{nom}} qui permet de rappeler le nom du nouvel ingrédient :



```
traitementFormulaireCreationIngredient.html - webpizza - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER WEBPIZZA templates/applipizza ...

```

5. Testez que tout fonctionne. Vous pourrez aussi contrôler, dans sqlitebrowser, l'enrichissement progressif de la table des ingrédients.



création d'un ingrédient - Mozilla Firefox

creation d'un ingrédient x +

127.0.0.1:8000/ingredients/add/

Appli Pizza 2023 les pizzas les ingrédients créer un ingrédient

création d'un ingrédient

Le nom de cet ingrédient

calamar

Envoyer

The screenshot shows a Mozilla Firefox window titled "ingrédient créé - Mozilla Firefox". The address bar displays "127.0.0.1:8000/ingredients/create/". The page content is from "Appli Pizza 2023" and shows the heading "création d'un ingrédient". Below it, a message says "l'ingrédient calamari a été créé." At the bottom right, there is a copyright notice "© Appli Pizza 2023".

Avez-vous bien tout assimilé ?

Recommencez tout, cette fois pour la création d'une nouvelle pizza (nom, et prix).

Voici un exemple de rendu :

The screenshot shows a Chromium browser window titled "création d'une pizza – Chromium". The address bar displays "127.0.0.1:8000/pizzas/add/". The page content is from "Appli Pizza 2023" and shows the heading "création d'une pizza". It includes fields for "Le nom de cette pizza" (filled with "végétarienne") and "Le prix de cette pizza" (filled with "14.75"). A green "Envoyer" button is at the bottom. At the bottom right, there is a copyright notice "© Appli Pizza 2023".

création d'une pizza – Chromium

creation d'une pizza x + 127.0.0.1:8000/pizzas/create/

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

création d'une pizza

La pizza végétarienne a bien été créée !

© Appli Pizza 2023

les pizzas – Chromium

les pizzas x + 127.0.0.1:8000/pizzas/

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

voici nos 4 pizzas

- pizza quatre fromages (prix = 15.50 €) | [détails](#)
- pizza napolitaine (prix = 14.80 €) | [détails](#)
- pizza montagnarde (prix = 1.00 €) | [détails](#)
- pizza végétarienne (prix = 14.75 €) | [détails](#)

© Appli Pizza 2023

détails d'une pizza – Chromium

détails d'une pizza x + 127.0.0.1:8000/pizzas/4/

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

voici notre pizza

pizza végétarienne (prix = 14.75 €)

la pizza végétarienne n'a pas encore d'ingrédient !

© Appli Pizza 2023

Remarque : pour la dernière capture, il y a eu un choix : s'il n'y a pas encore d'ingrédient, alors ...