

TP1 – Commencer un projet Django

Contexte et Objectifs

Pour tout ce travail, nous allons utiliser des outils importants ou nouveaux :

- un environnement Linux
- un outil de versionnage (git, et le serveur gitlab de l'IUT)
- un framework python : Django, installé en local.

Ce framework nous permettra de concevoir petit à petit un travail classique déjà fait en PHP/MySQL pur (avec un sujet différent) pour voir comment gérer de A à Z et sans tout réinventer un projet web avec des données persistantes.

Le gitLab de l'iut

Nous pouvons utiliser le gitlab de l'IUT (ou toute autre plateforme de dépôts git) pour disposer d'un dépôt de fichiers distant. C'est une habitude essentielle en travail d'équipe, même si l'équipe en question n'est composée que de vous-même en l'occurrence.

1. Connexion au gitlab de l'iut

- a. aller à https://git.iut-orsay.fr/users/sign_in
- b. se connecter avec login court et password

2. Création d'un nouveau projet et clonage

- a. aller dans vos projets et créer un nouveau projet vierge nommé **webpizza**. Vous pouvez le créer en privé ou en public.
- b. aller dans le projet et récupérer l'url du clonage en https qui doit être du type

<https://git.iut-orsay.fr/prenom.nom/webpizza.git>

- c. Dans votre architecture locale, créer un dossier **home/Documents/projets-django/** et s'y placer
- d. lancer le clonage à cet endroit par la commande **git clone <lien du projet git>**
- e. vérifier dans votre répertoire projets-django qu'il y a maintenant un sous-répertoire mypizza, avec à l'intérieur un dossier caché .git

Remarque : il y a d'autres façons de faire, en particulier vous pouvez donner à votre IDE les autorisations de migrer votre projet local vers le gitlab. C'est possible par exemple avec VS Code.

3. Commandes essentielles git

- a. où en êtes vous ?
La commande **git status** vous indique si votre copie locale est en retard (ou en avance) sur la copie distante.
- b. Ajouter des fichiers à suivre
La commande **git add** . Ajoute tous les fichiers pas encore suivis dans la liste des fichiers à suivre. Tout changement sur ces fichiers sera indiqué.
- c. Valider les changements sur les fichiers suivis
La commande **git commit -m "message pour ce commit"** permet de valider les changements et indique un message de repérage.
- d. Pousser les changements validés sur la branche distante
La commande **git push** permet de mettre à jour la branche distante.
- e. La commande de récupération de la branche distante
La commande **git pull** est très importante. Si vous n'êtes pas seul à travailler sur le même projet, il y a de grandes chances que des push aient été effectués par d'autres et si vous vous apprêtez à faire un push sans avoir mis à jour votre version locale avant, il va y avoir des conflits de version qu'il vaut mieux éviter.
- f. Vous pourrez aussi avoir besoin des commandes git relatives à la création et à la fusion de branches. A voir au cas où.

4. Chronologie classique

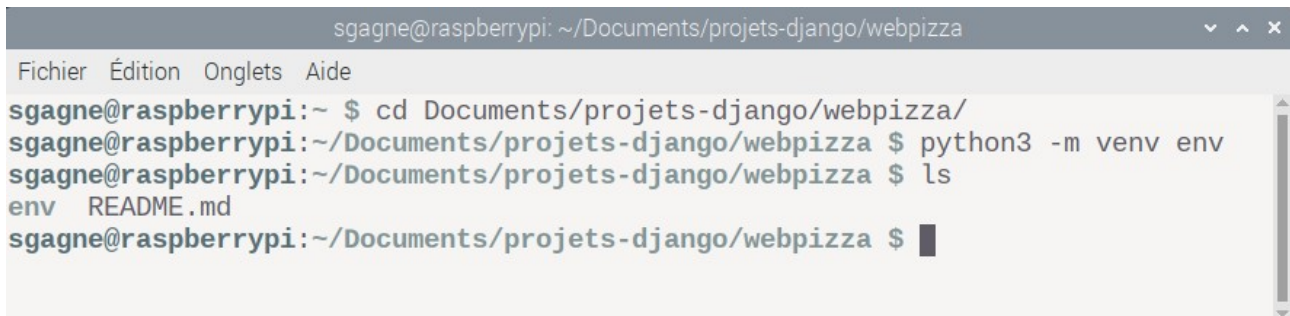
- a. `git pull`
- b. ... vos changements de code
- c. `git status`
- d. `git add .`
- e. `git commit -m "mon message personnalisé"`
- f. `git push`
- g. `git status` par acquis de conscience

Il vaut mieux faire trop de pull/push que pas assez.

L'outil Django pour le projet webpizza

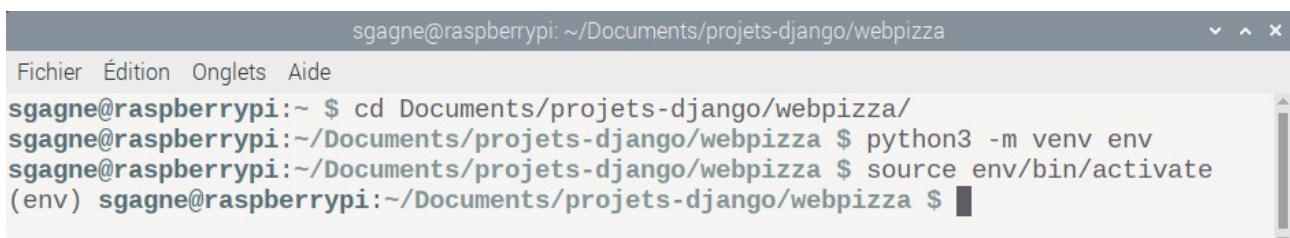
1. Installation de Django

- a. Aller dans le répertoire du projet webpizza et lancer la commande **`python3 -m venv env`**, ce qui crée un environnement virtuel nommé « env » pour Django. Vous le verrez par un `ls` dans le répertoire.



```
sgagne@raspberrypi: ~/Documents/projets-django/webpizza
Fichier Édition Onglets Aide
sgagne@raspberrypi:~ $ cd Documents/projets-django/webpizza/
sgagne@raspberrypi:~/Documents/projets-django/webpizza $ python3 -m venv env
sgagne@raspberrypi:~/Documents/projets-django/webpizza $ ls
env  README.md
sgagne@raspberrypi:~/Documents/projets-django/webpizza $
```

- b. Activer l'environnement virtuel par la commande **`source env/bin/activate`**



```
sgagne@raspberrypi: ~/Documents/projets-django/webpizza
Fichier Édition Onglets Aide
sgagne@raspberrypi:~ $ cd Documents/projets-django/webpizza/
sgagne@raspberrypi:~/Documents/projets-django/webpizza $ python3 -m venv env
sgagne@raspberrypi:~/Documents/projets-django/webpizza $ source env/bin/activate
(env) sgagne@raspberrypi:~/Documents/projets-django/webpizza $
```

L'activation se voit par `(env)`. L'environnement doit être activé faire tourner le serveur web. Si une désactivation se produit, reprendre la procédure d'activation du b.

c. Installation de Django dans l'environnement virtuel

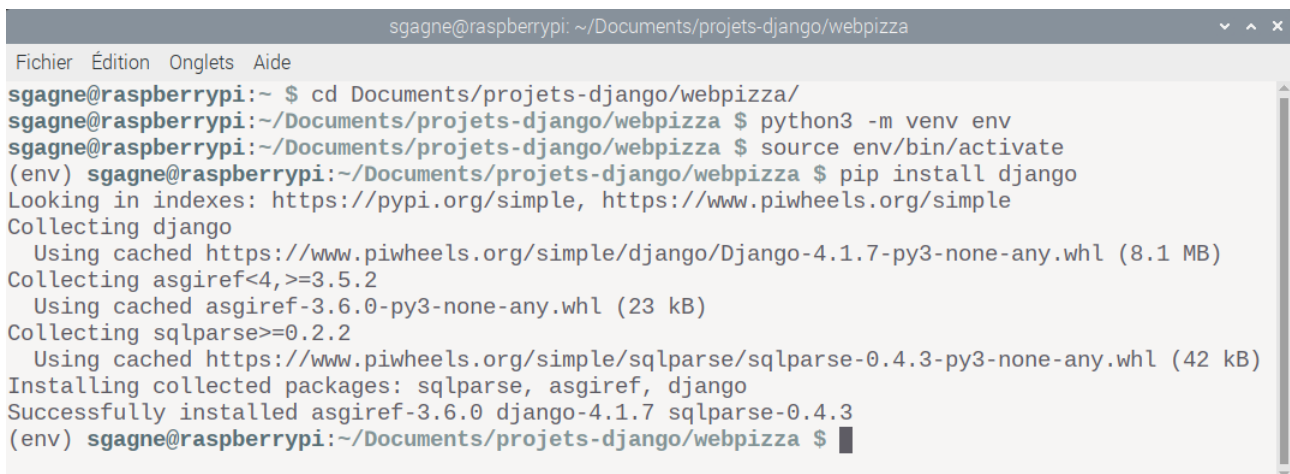
Dans webpizza, en n'oubliant pas d'activer l'environnement virtuel, nous allons installer le framework django pour notre projet. Au préalable, à l'IUT, nous avons besoin de variables pour le proxy :

D'abord, les commandes pour les deux variables :

```
HTTP_PROXY=http://proxy.iut-orsay.fr:3128
HTTPS_PROXY=http://proxy.iut-oray.fr:3128
export HTTP_PROXY
export HTTPS_PROXY
```

Puis la commande d'installation de django :

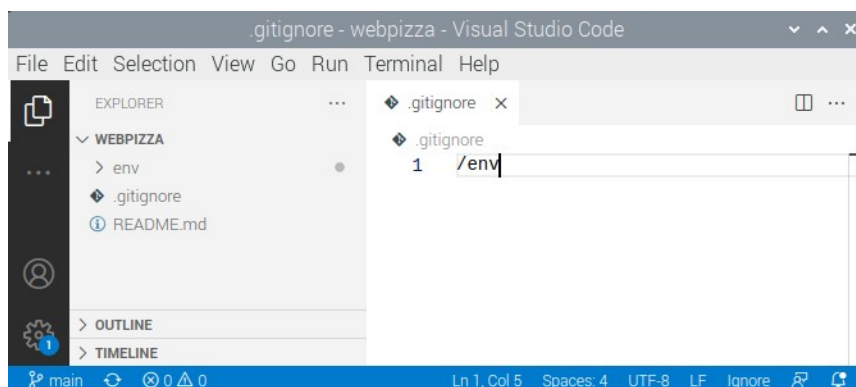
```
pip install django
```



```
sgagne@raspberrypi: ~/Documents/projets-django/webpizza
Fichier Édition Onglets Aide
sgagne@raspberrypi:~ $ cd Documents/projets-django/webpizza/
sgagne@raspberrypi:~/Documents/projets-django/webpizza $ python3 -m venv env
sgagne@raspberrypi:~/Documents/projets-django/webpizza $ source env/bin/activate
(env) sgagne@raspberrypi:~/Documents/projets-django/webpizza $ pip install django
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Collecting django
  Using cached https://www.piwheels.org/simple/django/Django-4.1.7-py3-none-any.whl (8.1 MB)
Collecting asgiref<4,>=3.5.2
  Using cached asgiref-3.6.0-py3-none-any.whl (23 kB)
Collecting sqlparse>=0.2.2
  Using cached https://www.piwheels.org/simple/sqlparse/sqlparse-0.4.3-py3-none-any.whl (42 kB)
Installing collected packages: sqlparse, asgiref, django
Successfully installed asgiref-3.6.0 django-4.1.7 sqlparse-0.4.3
(env) sgagne@raspberrypi:~/Documents/projets-django/webpizza $
```

d. Le fichier .gitignore

Ce fichier indique les répertoires ou fichiers qui doivent être ignorés lors des procédures git. Nous allons, au niveau de la racine (au même niveau que README.md et env) créer un fichier .gitignore (ce sera un fichier caché) qui contiendra la ligne suivante :



Ce fichier sera certainement mis à jour par la suite.
Mettez à jour votre dépôt distant pour ajouter ce fichier .gitignore.

```
sgagne@raspberrypi: ~/Documents/projets-django/webpizza
Fichier Édition Onglets Aide
(env) sgagne@raspberrypi:~/Documents/projets-django/webpizza $ git status
Sur la branche main
Votre branche est à jour avec 'origin/main'.

Fichiers non suivis:
  (utilisez "git add <fichier>..." pour inclure dans ce qui sera validé)
  .gitignore

aucune modification ajoutée à la validation mais des fichiers non suivis sont présents (utilisez "git add"
pour les suivre)
(env) sgagne@raspberrypi:~/Documents/projets-django/webpizza $ git add .
(env) sgagne@raspberrypi:~/Documents/projets-django/webpizza $ git status
Sur la branche main
Votre branche est à jour avec 'origin/main'.

Modifications qui seront validées :
  (utilisez "git restore --staged <fichier>..." pour désindexer)
  nouveau fichier : .gitignore

(env) sgagne@raspberrypi:~/Documents/projets-django/webpizza $ git commit -m "ajout de .gitignore"
[main d6df7f1] ajout de .gitignore
1 file changed, 1 insertion(+)
create mode 100644 .gitignore
(env) sgagne@raspberrypi:~/Documents/projets-django/webpizza $ git push
Username for 'https://git.iut-orsay.fr': sgagne
Password for 'https://sgagne@git.iut-orsay.fr':
Énumération des objets: 4, fait.
Décompte des objets: 100% (4/4), fait.
Compression par delta en utilisant jusqu'à 4 fils d'exécution
Compression des objets: 100% (2/2), fait.
Écriture des objets: 100% (3/3), 287 octets | 287.00 Kio/s, fait.
Total 3 (delta 0), réutilisés 0 (delta 0), réutilisés du pack 0
To https://git.iut-orsay.fr/sebastien.gagne/webpizza.git
0949b13..d6df7f1  main -> main
(env) sgagne@raspberrypi:~/Documents/projets-django/webpizza $
```

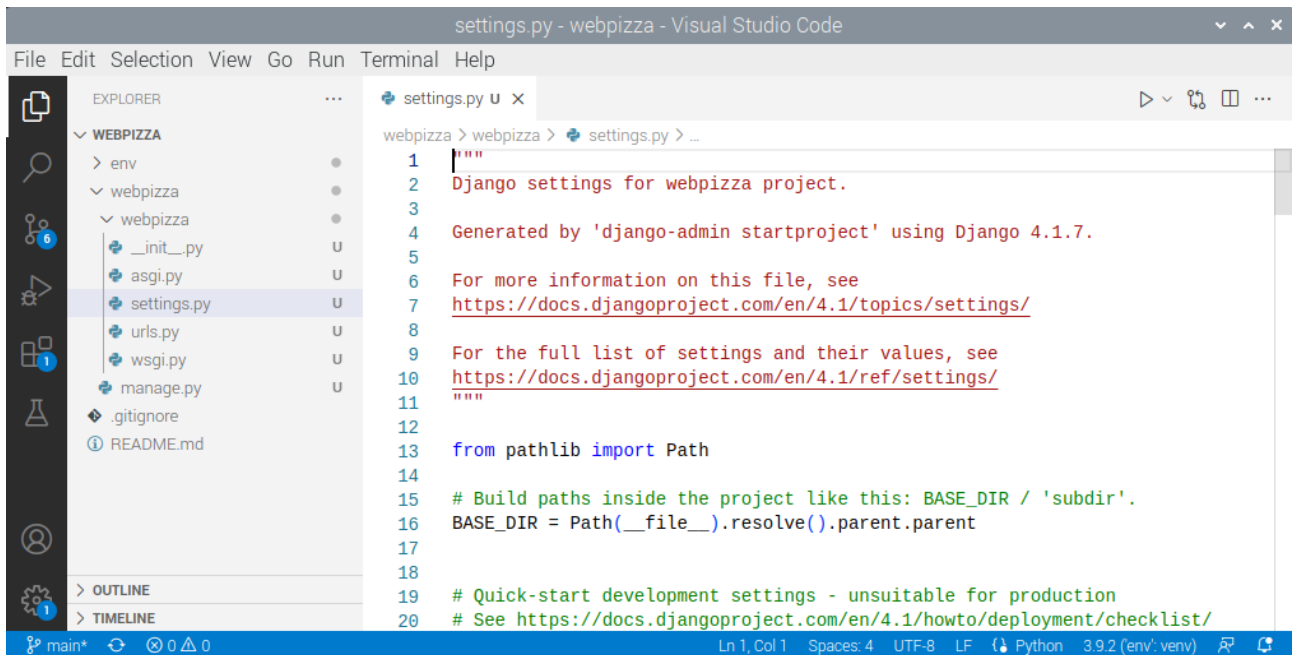
Création du projet webpizza

1. La commande **django-admin startproject webpizza**

- a. Nous allons créer un projet portant le même nom que le dossier la contenant : webpizza. Ce nom aurait pu être différent.

```
sgagne@raspberrypi: ~/Documents/projets-django/webpizza
Fichier Édition Onglets Aide
(env) sgagne@raspberrypi:~/Documents/projets-django/webpizza $ django-admin startproject webpizza
(env) sgagne@raspberrypi:~/Documents/projets-django/webpizza $
```

- b. Cette commande a en fait créé de nombreux fichiers que nous réutiliserons, et qui structurent la base de notre projet. Dans Visual Studio Code :



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

EXPLORER
webpizza
  > env
  > webpizza
    > webpizza
      __init__.py
      asgi.py
      settings.py
      urls.py
      wsgi.py
      manage.py
    .gitignore
    README.md

  > OUTLINE
  > TIMELINE

webpizza > webpizza > settings.py > ...
1  """
2  Django settings for webpizza project.
3
4  Generated by 'django-admin startproject' using Django 4.1.7.
5
6  For more information on this file, see
7  https://docs.djangoproject.com/en/4.1/topics/settings/
8
9  For the full list of settings and their values, see
10 https://docs.djangoproject.com/en/4.1/ref/settings/
11 """
12
13 from pathlib import Path
14
15 # Build paths inside the project like this: BASE_DIR / 'subdir'.
16 BASE_DIR = Path(__file__).resolve().parent.parent
17
18
19 # Quick-start development settings - unsuitable for production
20 # See https://docs.djangoproject.com/en/4.1/howto/deployment/checklist/
```

2. La base de données du projet webpizza

Une base sqlite par défaut est utilisée par Django. On peut configurer sans problème une base de données MySQL, mais à l'IUT les sécurités sont fortes pour pouvoir agir sur les bases MySQL autrement que via PHP qui a reçu les droits nécessaires.

Nous allons donc, en travaillant totalement en local, utiliser une base de données sqlite, avec comme interface graphique sqlitebrowser.

a. La commande **python3 manage.py migrate**

Nous allons utiliser la commande **python3 manage.py migrate** qui va mettre en place la base de données du projet.

Elle s'exécute **dans le répertoire webpizza** (celui du projet), **où on trouve aussi le fichier manage.py** qui permet de lancer les commandes en ligne python.

Cette commande ne sert pas qu'à la mise en place, mais de manière générale à permettre les « migrations », qui ne sont que les traductions de nos modèles vers des tables de données.

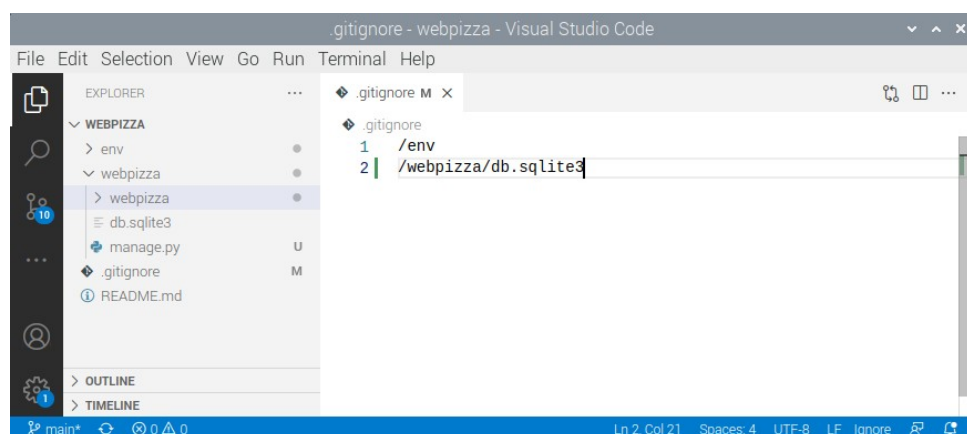
Exécutez cette commande. Le résultat en console doit être similaire à :

```
sgagne@raspberrypi: ~/Documents/projets-django/webpizza/webpizza
Fichier Édition Onglets Aide
(env) sgagne@raspberrypi:~/Documents/projets-django/webpizza $ cd webpizza/
(env) sgagne@raspberrypi:~/Documents/projets-django/webpizza/webpizza $ ls
manage.py webpizza
(env) sgagne@raspberrypi:~/Documents/projets-django/webpizza/webpizza $ python3 manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying auth.0012_alter_user_first_name_max_length... OK
  Applying sessions.0001_initial... OK
(env) sgagne@raspberrypi:~/Documents/projets-django/webpizza/webpizza $
```

Vous constatez que maintenant dans votre projet il y a un fichier **db.sqlite3**, c'est notre base de données. Pour le moment nous n'y avons pas créé de tables.

```
sgagne@raspberrypi: ~/Documents/projets-django/webpizza/webpizza
Fichier Édition Onglets Aide
(env) sgagne@raspberrypi:~/Documents/projets-django/webpizza/webpizza $ ls
db.sqlite3 manage.py webpizza
(env) sgagne@raspberrypi:~/Documents/projets-django/webpizza/webpizza $
```

Vous pouvez ajouter ce fichier db.sqlite3 à .gitignore, car il n'est pas utile de le suivre dans le dépôt distant.

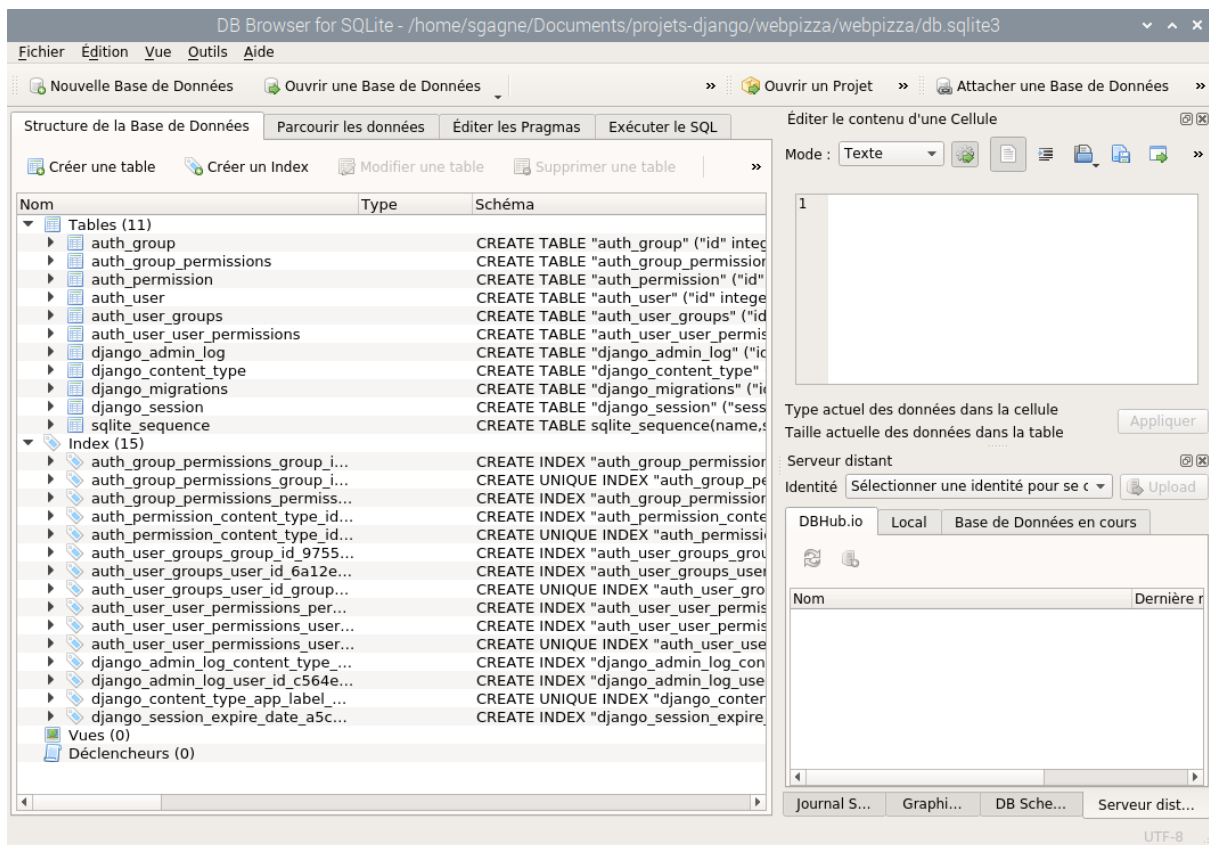


Mettez le dépôt à jour (vos commandes git doivent s'effectuer dans webpizza, pas dans webpizza/webpizza).

3. L'outil sqlitebrowser

Cet outil est une interface permettant de consulter/modifier la base de données du projet. Il nous servira probablement pour insérer des jeux de données.

Vérifiez que vous arrivez à ouvrir la base de données du projet avec sqlitebrowser. Vérifiez que dans la liste de tables, il n'y a encore rien concernant le projet, et que pour le moment il n'y a que des tables structurales, ce qui est normal car nous n'avons pas encore conçu de tables ni enregistré de données.



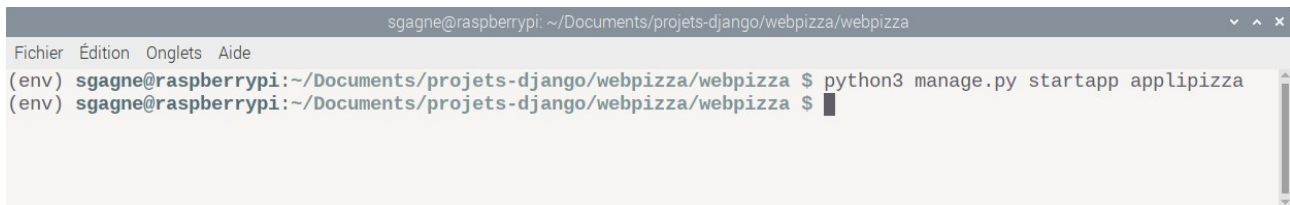
Création d'un application dans le projet

Par souci de *modularité*, un projet peut contenir plusieurs « applications ». Notre première application dans ce projet sera appelée applipizza. Le nom webpizza est interdit car c'est le nom du projet contenant l'application. Il pourrait y avoir d'autres applications dans le même projet, et une application d'un projet peut être utilisée dans d'autres projets.

Par exemple, une autre application gèrera les connexions au site.

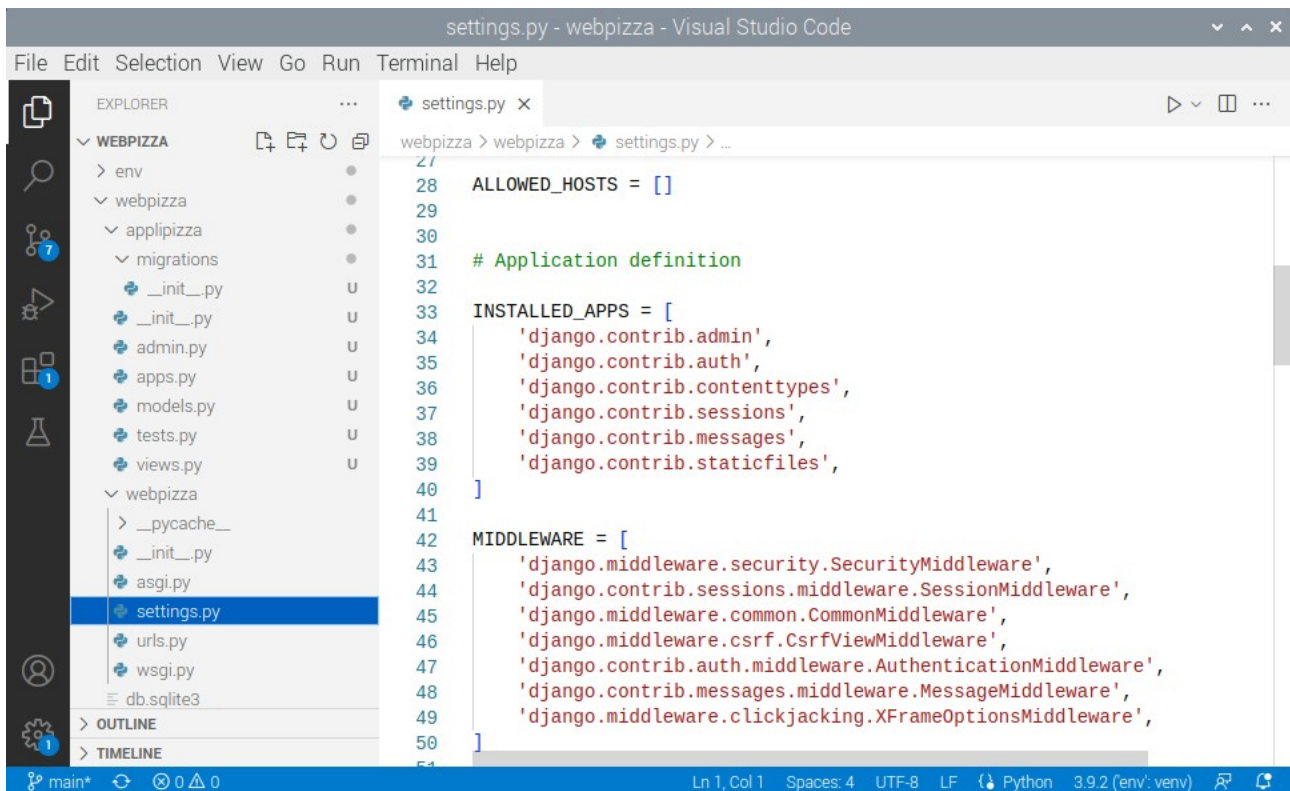
1. Création de l'application

- a. Lancez, dans l'environnement virtuel du projet webpizza, la commande **python3 manage.py startapp applipizza**



```
sgagne@raspberrypi: ~/Documents/projets-django/webpizza/webpizza
Fichier Édition Onglets Aide
(env) sgagne@raspberrypi:~/Documents/projets-django/webpizza/webpizza $ python3 manage.py startapp applipizza
(env) sgagne@raspberrypi:~/Documents/projets-django/webpizza/webpizza $
```

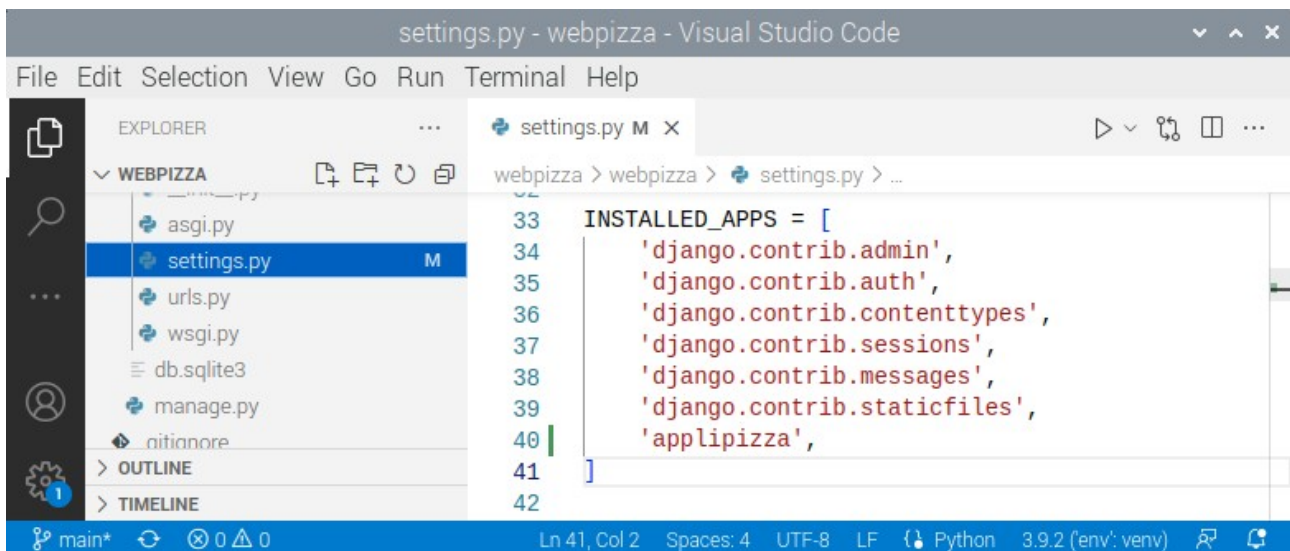
- b. On constate immédiatement l'effet produit par la création de l'application au niveau de l'architecture du projet. Dans webpizza (où on retrouve les fichiers essentiels au projet webpizza, comme settings.py), il y a maintenant un répertoire applipizza.



```
settings.py - webpizza - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
  WEBPIZZA
    env
    webpizza
      applipizza
        migrations
        __init__.py
        __init__.py
        admin.py
        apps.py
        models.py
        tests.py
        views.py
      webpizza
        __pycache__
        __init__.py
        asgi.py
        settings.py
        urls.py
        wsgi.py
        db.sqlite3
    OUTLINE
    TIMELINE
  settings.py x
webpizza > webpizza > settings.py > ...
27
28 ALLOWED_HOSTS = []
29
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 ]
41
42 MIDDLEWARE = [
43     'django.middleware.security.SecurityMiddleware',
44     'django.contrib.sessions.middleware.SessionMiddleware',
45     'django.middleware.common.CommonMiddleware',
46     'django.middleware.csrf.CsrfViewMiddleware',
47     'django.contrib.auth.middleware.AuthenticationMiddleware',
48     'django.contrib.messages.middleware.MessageMiddleware',
49     'django.middleware.clickjacking.XFrameOptionsMiddleware',
50 ]
51
Ln 1, Col 1 Spaces: 4 UTF-8 LF Python 3.9.2 (env: venv)
```

- c. déclaration de l'application dans les paramètres du projet

Dans le fichiers settings.py (voir capture précédente), il y a la liste python appelée `INSTALLED_APPS`. Il faut ajouter, en fin de la liste, l'application nouvellement créée :



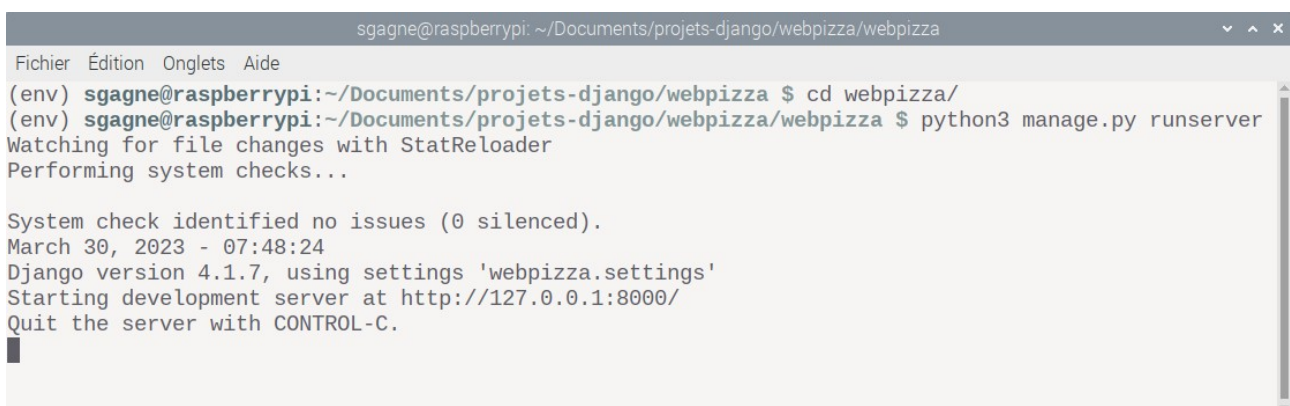
N'oubliez pas de mettre à jour votre dépôt distant, et vérifiez sur la page web du gitlab que le dépôt distant est bien à jour.

Lancement du serveur web

Nous allons maintenant observer le site créé

1. La commande **python3 manage.py runserver**

Lancez la commande `python3 manage.py runserver` qui lance le serveur web



2. Dans le navigateur

Aller à l'url <http://127.0.0.1:8000/> pour voir le site actuel



django

View [release notes](#) for Django 4.1



The install worked successfully! Congratulations!

You are seeing this page because `DEBUG=True` is in your settings file and you have not configured any URLs.



Django Documentation
Topics, references, & how-to's



Tutorial: A Polling App
Get started with Django



Django Community
Connect, get help, or contribute

Pour le moment c'est très pauvre et anonyme, mais si vous obtenez ceci, c'est bon signe pour la suite.