

VIRTUALISATION & CONTAINERISATION:

Projet Gestion d'étudiants



Description:

Le projet de **gestion des étudiants** est développé avec le **framework Django de Python**, offrant une interface intuitive pour implémenter les opérations **CRUD** (Create, Read, Update, Delete) avec une base de données **SQLite3**. Il permet aux utilisateurs d'**ajouter**, **supprimer**, **mettre à jour** et **afficher** les informations des étudiants. L'interface propose des boutons clairement définis pour chaque fonctionnalité, assurant une gestion efficace et conviviale des données étudiantes.

Réalisé par le groupe 4:

Afdel Desmond KOMBOU

Alglège SOUENI

Mouhamed DIOP

Sous la supervision:

Mr Madické DIOP

Plan Détailé

I. Technologies Utilisées

II. Organisation et Planification du Projet

1. Utilisation de Trello pour la Gestion des Tâches
2. Intégration de Trello avec Slack
3. Création et Gestion du Repository GitHub

III. Développement de l'Application

1. Installation des dépendances et configuration de l'IDE
2. Configuration du Backend du Projet
3. Configuration du Frontend du Projet

IV. Test et Déploiement

1. Configuration du Pipeline Jenkins
2. Connexion de Jenkins à GitHub via Ngrok
3. Construction et Déploiement de l'Image Docker
4. Vérifications après exécution du pipeline
5. Vérification de l'Application sur Docker Hub

Conclusion

- Résumé du Projet
- Remerciements

Annexes

- Notes de ChatGPT
- Chaîne YouTube "Apprendre Django"
- Documentation Django

Remerciements 😊

I.Techologies Utilisées



Trello est un outil de gestion de projet visuel qui permet de suivre les tâches et d'organiser le travail en utilisant des cartes et des tableaux. Il est utilisé pour planifier, assigner des tâches et suivre la progression du projet de gestion des étudiants.



Slack est une plateforme de collaboration et de communication en ligne qui permet aux équipes de travail de communiquer et de partager des informations de manière plus efficace.



VS Code est un éditeur de code source puissant et léger développé par Microsoft. Il est utilisé pour écrire, déboguer et tester le code Django, offrant une multitude d'extensions pour améliorer le développement et la productivité.



Django est un framework web gratuit et open-source écrit en Python. Il est conçu pour aider les développeurs à créer des applications web rapidement et efficacement. Django fournit un ensemble de composants et d'outils pour gérer les aspects de base d'une application web.



GitHub est une plateforme de gestion de code source utilisant Git. Il est utilisé pour héberger le code source du projet, permettre la collaboration entre les membres de l'équipe et gérer les versions du code.



Jenkins est un outil d'intégration continue et de livraison continue (CI/CD) open-source. Il est utilisé pour automatiser les tests et les déploiements du projet, assurant que le code est constamment testé et que les nouvelles versions sont déployées de manière fiable.



Docker est une plateforme de conteneurisation qui permet de créer, déployer et exécuter des applications dans des conteneurs isolés. Il est utilisé pour empaqueter l'application Django avec toutes ses dépendances, assurant une exécution cohérente dans différents environnements.



Ngrok est un outil qui crée des tunnels sécurisés pour exposer des services locaux à Internet. Il est utilisé pour tester l'application Django en ligne en créant une URL accessible publiquement, facilitant les démonstrations et les tests à distance.

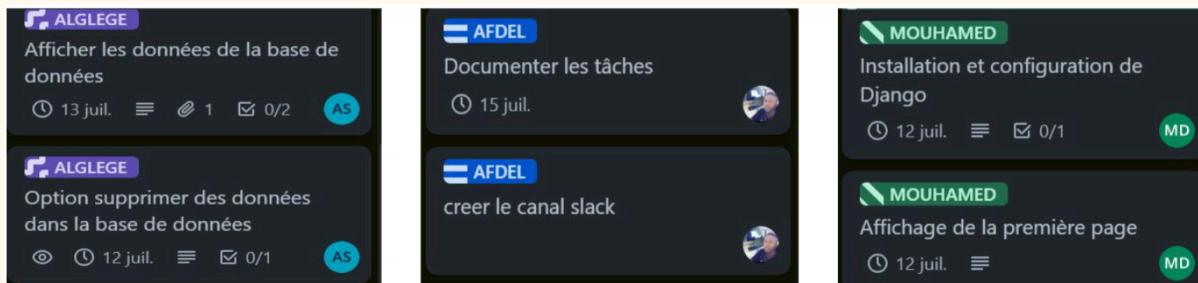
II. Organisation et Planification du Projet

Note:

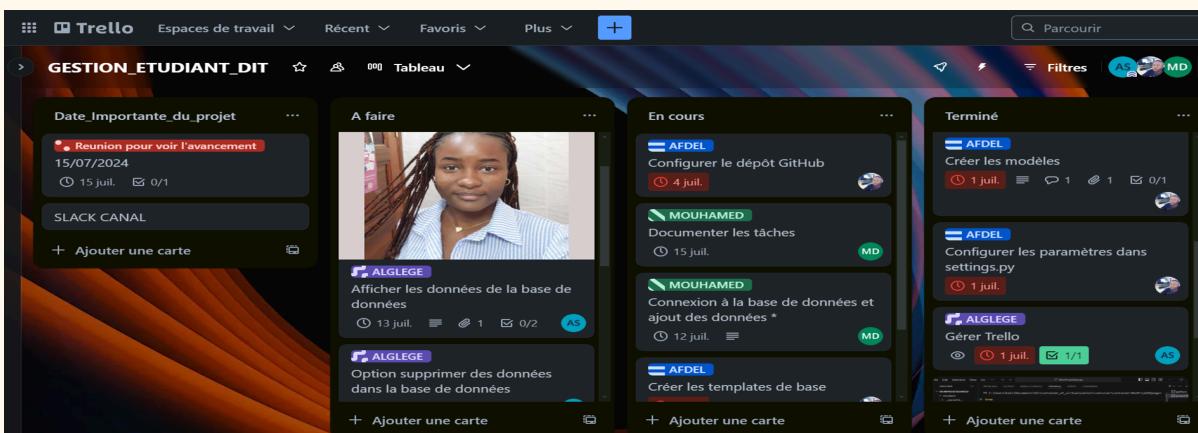
Cette section décrit comment le projet a été organisé et planifié en utilisant des outils collaboratifs comme **Trello** et **Slack**. L'équipe a utilisé Trello pour gérer les tâches, en les classant en différentes listes (**À faire**, **En cours**, **Terminé**) et en assignant des responsabilités spécifiques à chaque membre. Slack a été intégré pour recevoir des alertes et maintenir une communication efficace. Le repository **GitHub** a été créé pour la gestion du code source, permettant une collaboration fluide et une gestion de version efficace.

1. Utilisation de Trello pour la Gestion des Tâches

- **Création du tableau Trello pour le projet Ajout des membres de l'équipe et assignation des tâches**

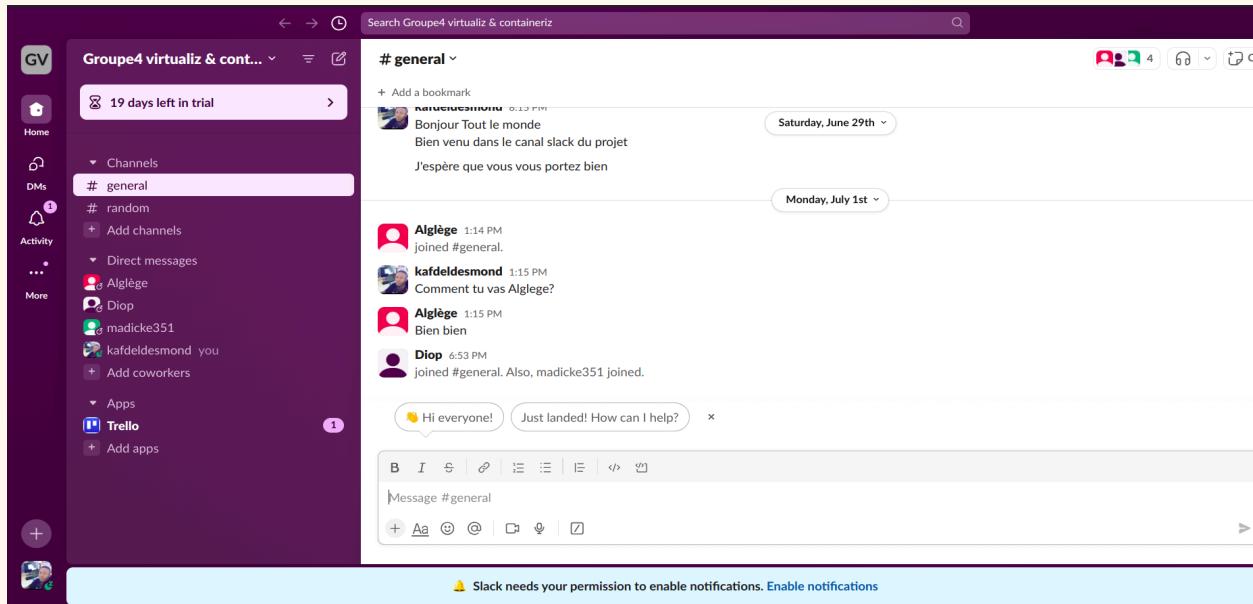


- **Classification des tâches en listes : À faire, En cours, Terminé**

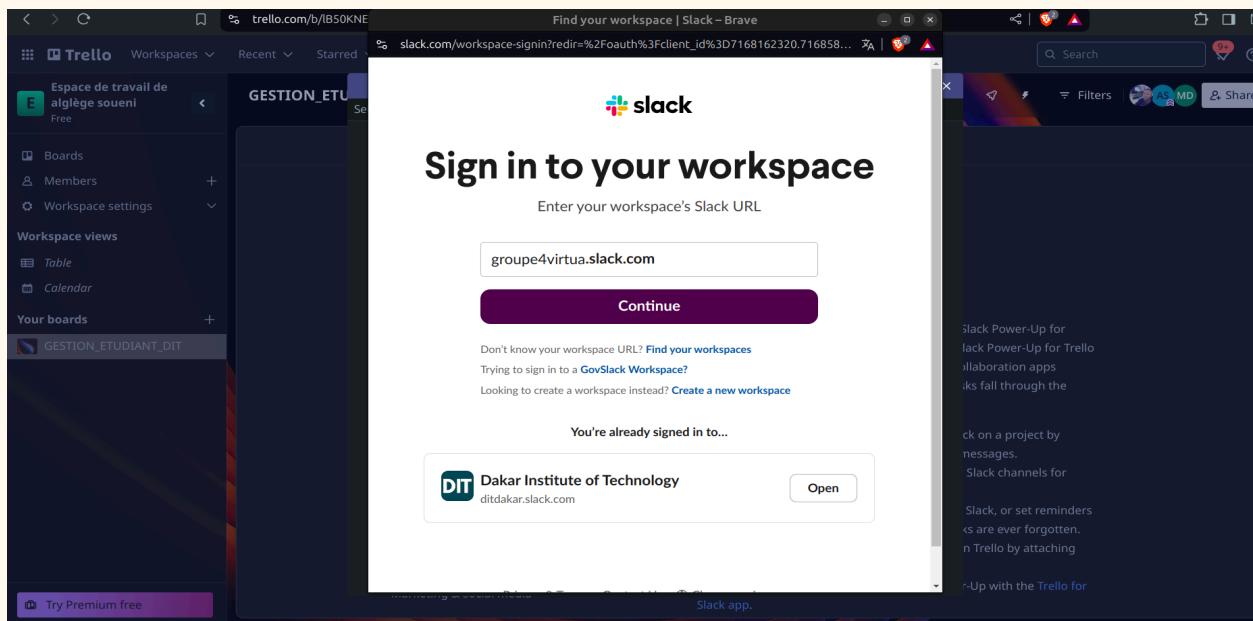


2. Intégration de Trello avec Slack

- Création d'un nouveau canal Slack pour le projet



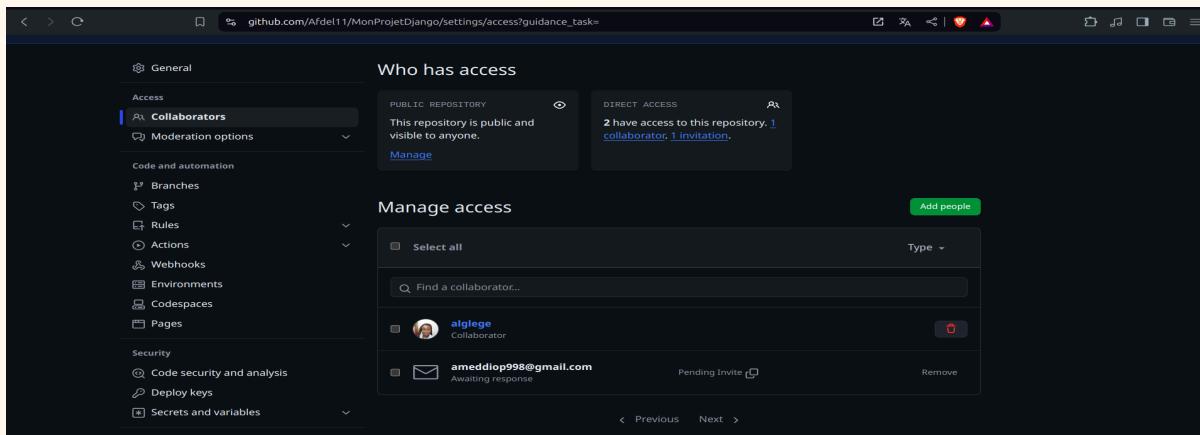
- Relier Trello avec Slack pour recevoir des alertes sur les actions des membres



3. Création et Gestion du Repository GitHub

Cette section explique comment nous avons configuré notre projet Django pour utiliser **Git**, en créant un **repository local** et en le connectant à un repository distant sur **GitHub**.

- **Création d'un repository sur GitHub avec le même nom que le projet local**



- **Initialisation du projet local en tant que repository Git**

Commande d'initialisation du projet

→ ***git init***
 → ***git add .***
 → ***git commit -m "Initial commit"***

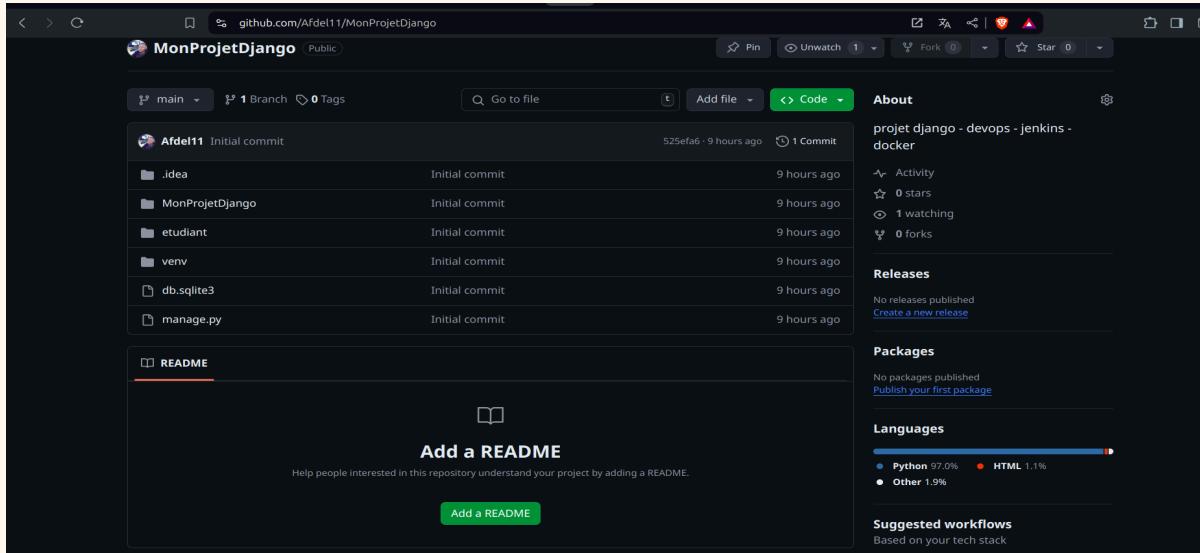
```
afdel@Laye:~/Téléchargements/container/MonProjetDjango$ git init
Dépôt Git existant réinitialisé dans /home/afdel/Téléchargements/container/MonProjetDjango/.git/
afdel@laye:~/Téléchargements/container/MonProjetDjango$ git status
Sur la branche main
Votre branche et 'origin/main' ont divergé,
et ont 1 et 6 commits différents chacune respectivement.
(use "git pull" if you want to integrate the remote branch with yours)
```



- **Envoie du code source vers le dépôt distant GitHub**

- Créez un nouveau **repository** sur GitHub depuis l'interface web. Copiez l'URL de ce nouveau repository.
- **git remote add origin https://github.com/Afde11/MonProjetDjango.git**
- **git push -u origin main**

Avec ces commandes, notre projet Django est maintenant configuré pour une gestion de version efficace via **Git** et **GitHub**, permettant une collaboration harmonieuse et un suivi des modifications apportées au **code source**.



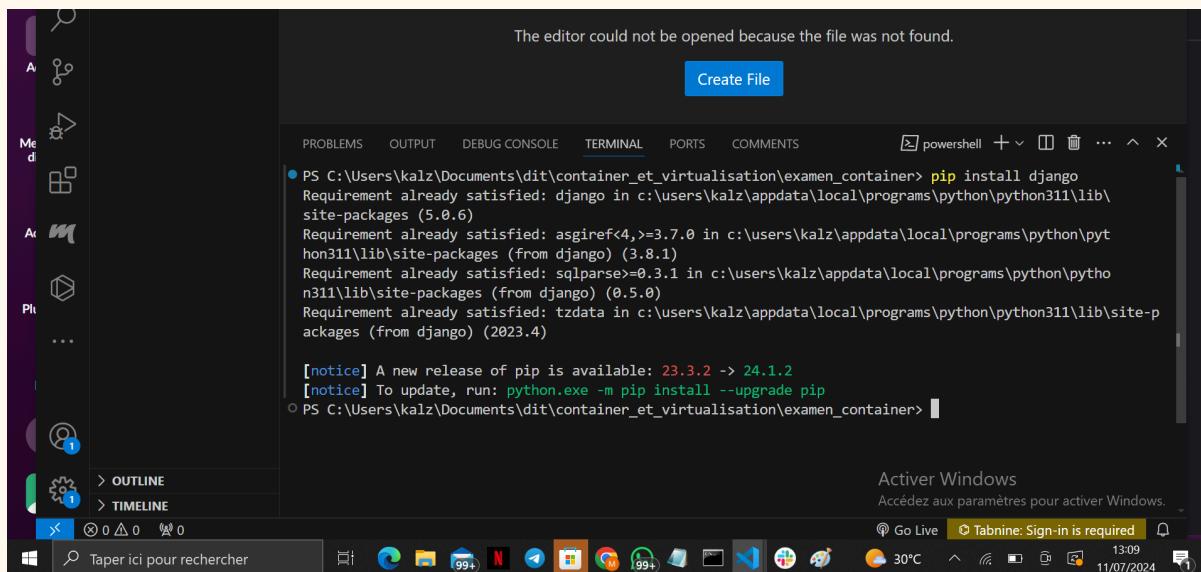
La préparation du projet étant terminée, nous pouvons maintenant passer au développement de l'application proprement dite.

III. Développement de l'Application

Note:

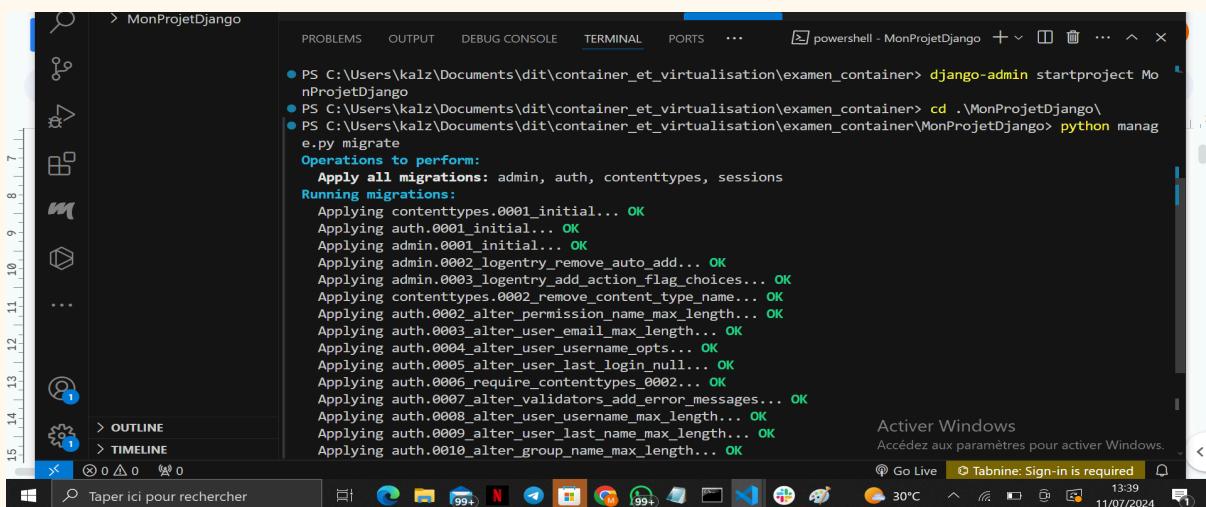
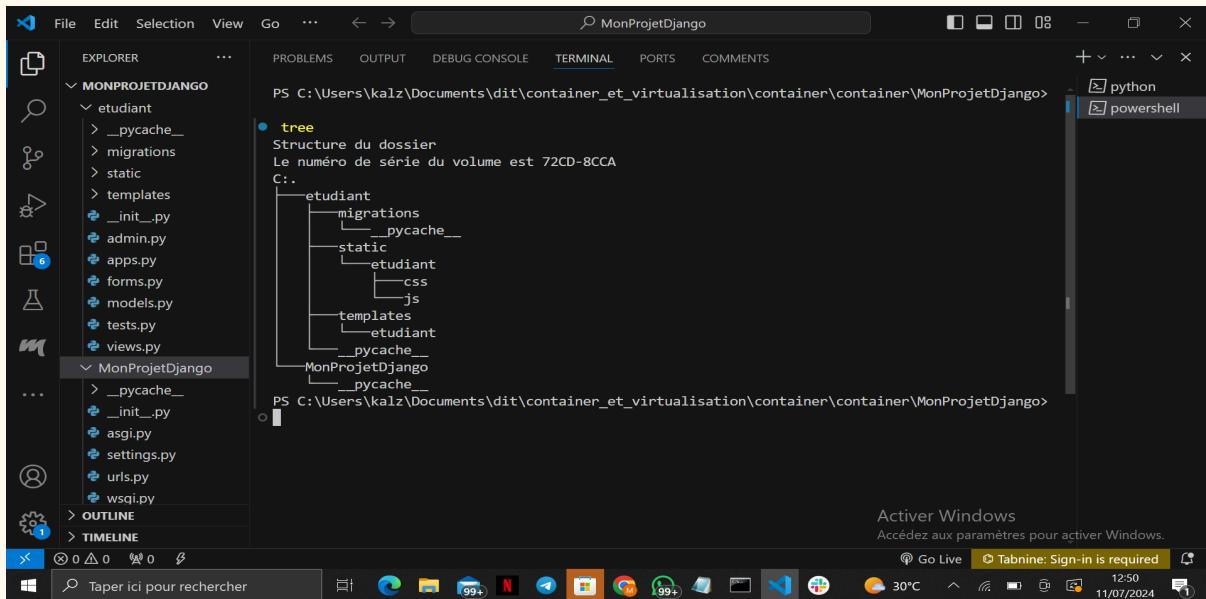
Dans cette section, nous détaillons le processus de développement de l'application **Django** pour la gestion des étudiants, en utilisant **Visual Studio Code**. Les fonctionnalités **CRUD** ont été implémentées et testées dans l'environnement de développement.

1. Installation des dependance et configuration de l'IDE



1. Configuration du Backend projet

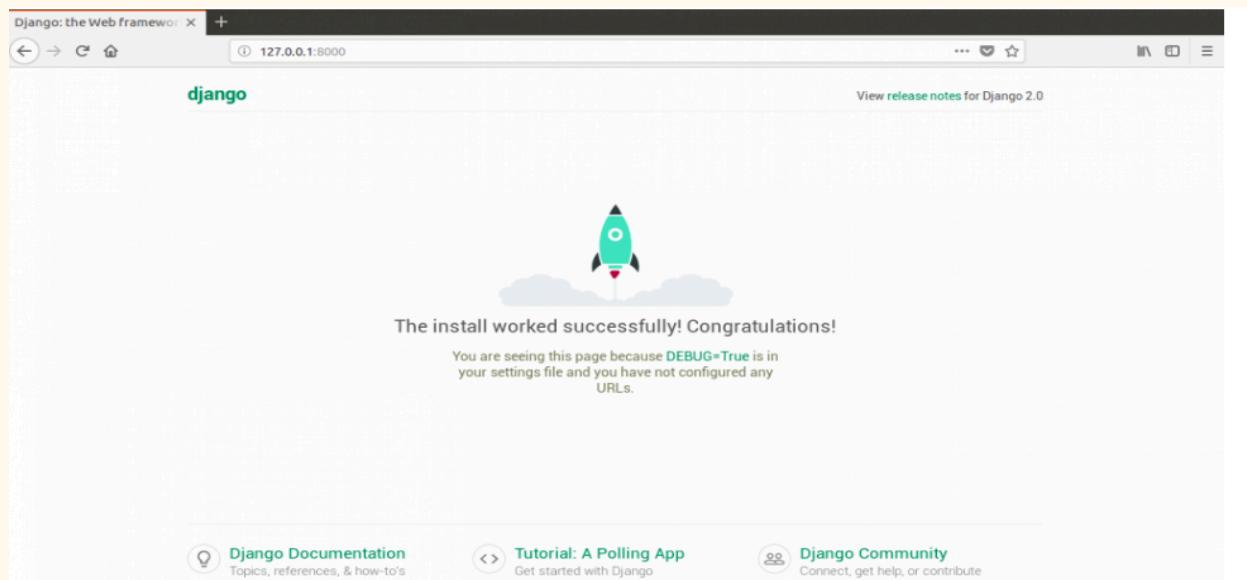
Pour configurer un projet Django, plusieurs fichiers et dossiers sont essentiels pour le bon fonctionnement du projet.



=> Démarrage du server Django

```
TERMINAL
(.venv) afdel@laye:~/Téléchargements/container/MonProjetDjango$ python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
July 15, 2024 - 01:06:04
Django version 5.0.7, using settings 'MonProjetDjango.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```



=> configuration du fichier “`settings.py`”

Contient toutes les **configurations globales** du projet Django, y compris la configuration de la **base de données**, les **applications** installées, les **middleware**, les chemins des fichiers statiques et des fichiers **médias**, les paramètres de **sécurité**, etc.

The screenshot shows the Visual Studio Code interface with the title bar "settings.py - MonProjetDjango - Visual Studio Code". The left sidebar displays the project structure:

- MonProjetDjango (selected)
- etudiant
- static
- templates/etu...
- base.html
- updatestud...
- __init__.py
- admin.py
- apps.py
- forms.py
- models.py
- tests.py
- views.py
- MonProjetDjango
- __pycache__
- __init__.py
- asgi.py
- settings.py
- urls.py
- wsgi.py
- venv
- db.sqlite3
- manage.py

The main editor area shows the content of the settings.py file:

```
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     'etudiant',
41 ]
42
43 MIDDLEWARE = [
44     'django.middleware.security.SecurityMiddleware',
45     'django.contrib.sessions.middleware.SessionMiddleware',
46     'django.middleware.common.CommonMiddleware',
47     'django.middleware.csrf.CsrfViewMiddleware',
48     'django.contrib.auth.middleware.AuthenticationMiddleware',
49     'django.contrib.messages.middleware.MessageMiddleware',
50     'django.middleware.clickjacking.XFrameOptionsMiddleware',
51 ]
52
53 ROOT_URLCONF = 'MonProjetDjango.urls'
54
55 TEMPLATES = [
56     {
57         'BACKEND': 'django.template.backends.django.DjangoTemplates',
58         'DIRS': [os.path.join(BASE_DIR, 'templates')],
```

=> configuration du fichier “Monprojet Django/urls.py ”

Contient les URL patterns pour le routage des requêtes vers les vues appropriées.

```

Fichier Edition Sélection Affichage Atteindre Exécuter Terminal Aide
EXPLORATEUR ... urls.py ...
MonProjetDjango > urls.py > ...
5     https://docs.djangoproject.com/en/5.0/topics/http/urls/
6 Examples:
7     Function views
8         1. Add an import: from my_app import views
9             2. Add a URL to urlpatterns: path('', views.home, name='home')
10    Class-based views
11        1. Add an import: from other_app.views import Home
12            2. Add a URL to urlpatterns: path('', Home.as_view(), name='home')
13 Including another URLconf
14     1. Import the include() function: from django.urls import include, path
15         2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))
16 """
17 from django.contrib import admin
18 from django.urls import path
19 from etudiant import views
20 urlpatterns = [
21     path('admin/', admin.site.urls),
22     path('', views.add_and_show, name='addandshow'),
23     path('delete/<int:id>/', views.delete_data, name='deletedata'),
24     path('<int:id>/', views.update_data, name='updatedata'),
25 ]
26

```

=> configuration du fichier “manage.py”

Situé à la racine du projet, utilisé pour exécuter des commandes liées à Django (comme `runserver`, `migrate`, `createsuperuser`, etc.)

```

EXPLORATEUR ... settings.py manage.py ...
MonProjetDjango > manage.py > ...
1 #!/usr/bin/env python
2 """Django's command-line utility for administrative tasks."""
3 import os
4 import sys
5
6 Explain Code | Generate Tests | Ask Sourcery
7 def main():
8     """Run administrative tasks."""
9     os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'MonProjetDjango.settings')
10    try:
11        from django.core.management import execute_from_command_line
12    except ImportError as exc:
13        raise ImportError(
14            "Couldn't import Django. Are you sure it's installed and "
15            "available on your PYTHONPATH environment variable? Did you "
16            "forget to activate a virtual environment?"
17        ) from exc
18    execute_from_command_line(sys.argv)
19
20
21 if __name__ == '__main__':
22     main()
23

```

```

manage.py - MonProjetDjango - Visual Studio Code
Fichier Edition Sélection Affichage Atteindre Exécuter Terminal Aide
EXPLORATEUR ...
MONPROJETDJANGO
> .idea
> .venv
> docker
> etudiant
> MonProjetDjango
> venv
> db.sqlite3
manage.py

manage.py
    Initialize Reactive Jupyter | Sync all Stale code
1  #!/usr/bin/env python
2  """Django's command-line utility for administrative tasks."""
3  import os
4  import sys
5
6
7  def main():
8      """Run administrative tasks."""
9      os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'MonProjetDjango.settings')
10     try:
11         from django.core.management import execute_from_command_line
12     except ImportError as exc:
13         raise ImportError(
14             "Couldn't import Django. Are you sure it's installed and "
15             "available on your PYTHONPATH environment variable? Did you "
16             "forget to activate a virtual environment?"
17         ) from exc
18     execute_from_command_line(sys.argv)
19
20
21 if __name__ == '__main__':
22     main()
23

```

=> configuration du fichier “urls.py”

2. Configuration de l'application “Etudiant”

L'application dans notre projet aura son propre dossier(**Étudiant**) contenant les fichiers suivants :

- **models.py** : Définit les modèles de base de données.

```

models.py - MonProjetDjango - Visual Studio Code
Fichier Edition Sélection Affichage Atteindre Exécuter Terminal Aide
EXPLORATEUR ...
MONPR... > etudiant
> static
> templates/etu...
> addandshow...
> base.html
> updatestud...
> __init__.py
> admin.py
> apps.py
> forms.py
models.py
tests.py

models.py
    Initialize Reactive Jupyter | Sync all Stale code
1  from django.db import models
2
3  # Create your models here.
4  Explain Code | Generate Tests | Generate Docstrings | Ask Sourcery
5  class User(models.Model):
6      name = models.CharField(max_length=70)
7      email = models.EmailField(max_length=100)
8      password = models.CharField(max_length=100)

```

- **views.py** : Contient les vues pour logique de traitement de l'ensemble des requêtes.

```

Fichier Edition Sélection Affichage Atteindre Exécuter Terminal Aide
EXPLORATEUR ... settings.py manage.py views.py
etudiant > Initialize Reactive Jupyter | Sync all State code
1 from django.shortcuts import render, HttpResponseRedirect
2 from .forms import StudentRegistration
3 from .models import User
4 # Create your views here.
5 #ajouter et afficher les informations d'un étudiant
6
7 Explain Code | Generate Tests | Generate Docstrings | Ask Sourcery
def add_show(request):
8     if request.method == 'POST':
9         fm = StudentRegistration(request.POST)
10        if fm.is_valid():
11            nm = fm.cleaned_data['name']
12            em = fm.cleaned_data['email']
13            pw = fm.cleaned_data['password']
14            reg = User(name=nm, email=em, password=pw)
15            reg.save()
16            fm = StudentRegistration()
17
18    else:
19        fm = StudentRegistration()
20    stud = User.objects.all()
21    return render(request,'etudiant/addandshow.html',{'form':fm, 'stu':stud})
22
23 #modifier les informations d'un étudiant
24 Explain Code | Generate Tests | Generate Docstrings | Ask Sourcery
25 def update_data(request,id):
26     if request.method == 'POST':
27         pi = User.objects.get(pk=id)
28         fm = StudentRegistration(request.POST,instance=pi)
29         fm.save()

```

- o **forms.py : Définit les formulaires de l'application.**

```

Fichier Edition Sélection Affichage Atteindre Exécuter Terminal Aide
EXPLORATEUR ... settings.py manage.py views.py forms.py
etudiant > Initialize Reactive Jupyter | Sync all State code
1 from django.core import validators
2 from django import forms
3 from .models import User
4
5 Explain Code | Generate Tests | Generate Docstrings | Ask Sourcery
6 class StudentRegistration(forms.ModelForm):
7     Explain Code | Generate Tests | Generate Docstrings | Ask Sourcery
8     class Meta:
9         model = User
10        fields = ['name', 'email', 'password']
11        widgets = {
12            'name': forms.TextInput(attrs={'class': 'form-control', 'placeholder': 'Name'}),
13            'email': forms.EmailInput(attrs={'class': 'form-control', 'placeholder': 'Email'}),
14            'password': forms.PasswordInput(render_value=True, attrs={'class': 'form-control', 'placeholder': 'Password'})

```

Fonction CRUD Basée sur la Gestion d'Etudiant

Modifier les informations d'un étudiant

Name:
qsdd

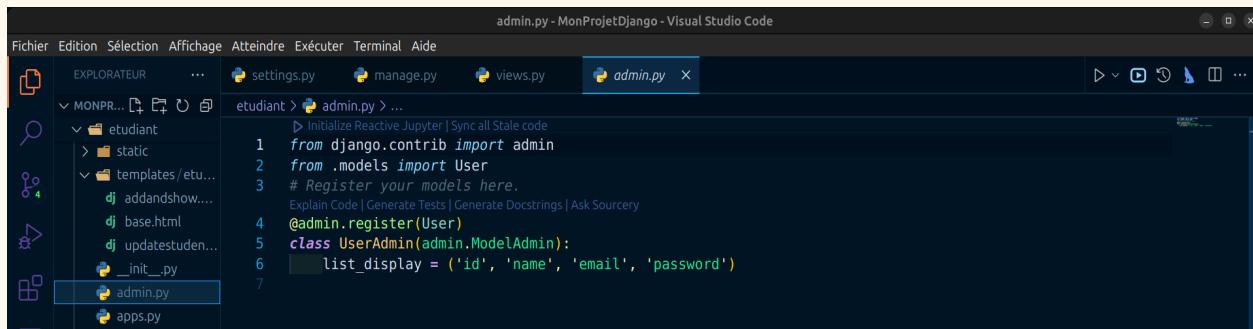
Email:
eyebayecherif@gmail.com

Password:
.....

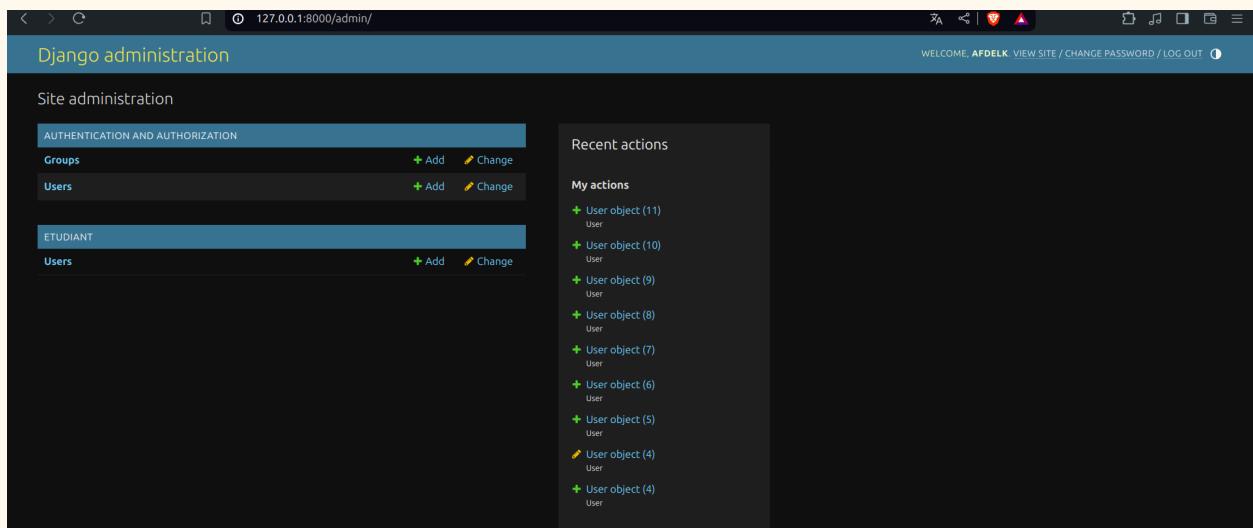
Modifier **Retour**

Activer Windows
Accédez aux paramètres pour activer Windows.

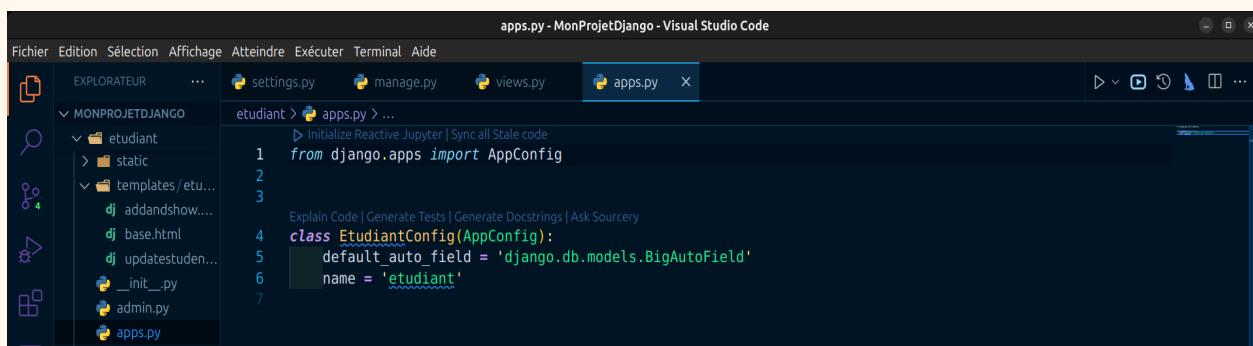
- o **admin.py** : Enregistre les modèles pour l'interface d'administration.



```
admin.py - MonProjetDjango - Visual Studio Code
Fichier Edition Sélection Affichage Atteindre Exécuter Terminal Aide
EXPLORATEUR ... settings.py manage.py views.py admin.py x
etudiant > admin.py > ...
    Initialize Reactive Jupyter | Sync all Stale code
1 from django.contrib import admin
2 from .models import User
3 # Register your models here.
4 @admin.register(User)
5 class UserAdmin(admin.ModelAdmin):
6     list_display = ('id', 'name', 'email', 'password')
7
```



- o **apps.py** : Contient la configuration de l'application.



```
apps.py - MonProjetDjango - Visual Studio Code
Fichier Edition Sélection Affichage Atteindre Exécuter Terminal Aide
EXPLORATEUR ... settings.py manage.py views.py apps.py x
etudiant > apps.py > ...
    Initialize Reactive Jupyter | Sync all Stale code
1 from django.apps import AppConfig
2
3
4 class EtudiantConfig(AppConfig):
5     default_auto_field = 'django.db.models.BigAutoField'
6     name = 'etudiant'
7
```

Wed 15:20

localhost:8000

Fonction CRUD Basée sur la Gestion d'Etudiant

Ajouter un nouveau étudiant

Name:

Email:

Password:

Ajouter

Les informations des Etudiants

pas d'enregistrement

Sat 23:33

localhost:8000

Gestion d'Etudiant

Ajouter un nouveau étudiant

Name:

Email:

Password:

Ajouter

Les informations des Etudiants

ID	Nom	Email	Mot de Passe	Action
4	algele	alsou@gmail.com	1234	<button style="background-color: #ffc107; color: black; padding: 2px 5px;">Modifier</button> <button style="background-color: #dc3545; color: white; padding: 2px 5px;">Supprimer</button>
5	desmond	desmond@gmail.com	1234	<button style="background-color: #ffc107; color: black; padding: 2px 5px;">Modifier</button> <button style="background-color: #dc3545; color: white; padding: 2px 5px;">Supprimer</button>
6	diane	diana@gmail.com	d4 blond	<button style="background-color: #ffc107; color: black; padding: 2px 5px;">Modifier</button> <button style="background-color: #dc3545; color: white; padding: 2px 5px;">Supprimer</button>
7	guillaume bertrand	guibertrang@yahoo.fr	09876kol	<button style="background-color: #ffc107; color: black; padding: 2px 5px;">Modifier</button> <button style="background-color: #dc3545; color: white; padding: 2px 5px;">Supprimer</button>
8	mouhamed diop	amediop@dit.sn	ammedy	<button style="background-color: #ffc107; color: black; padding: 2px 5px;">Modifier</button> <button style="background-color: #dc3545; color: white; padding: 2px 5px;">Supprimer</button>
9	Madické Diop	madidiop@gmail.com	mdkdp	<button style="background-color: #ffc107; color: black; padding: 2px 5px;">Modifier</button> <button style="background-color: #dc3545; color: white; padding: 2px 5px;">Supprimer</button>
10	boukar diallo	boukar_diallo@yabadou.sh	boubacar	<button style="background-color: #ffc107; color: black; padding: 2px 5px;">Modifier</button> <button style="background-color: #dc3545; color: white; padding: 2px 5px;">Supprimer</button>
11	mame diara	madiaratall@sns.pip	*****	<button style="background-color: #ffc107; color: black; padding: 2px 5px;">Modifier</button> <button style="background-color: #dc3545; color: white; padding: 2px 5px;">Supprimer</button>

127.0.0.1:8000/admin/etudiant/user/

WELCOME, AFDELK. VIEW SITE / CHANGE PASSWORD / LOG OUT

Django administration

Home > Etudiant > Users

Select user to change				
Action: -----				
	ID	NAME	EMAIL	PASSWORD
<input checked="" type="checkbox"/>	11	mame diara	madiaratall@sns.pip	*****
<input checked="" type="checkbox"/>	10	boukar diallo	boukar_diallo@yabadou.sh	boubacar
<input checked="" type="checkbox"/>	9	Madické Diop	madidiop@gmail.com	mdkdp
<input checked="" type="checkbox"/>	8	mouhamed diop	amediop@dit.sn	ammedy
<input checked="" type="checkbox"/>	7	guillaume bertrand	guibertrang@yahoo.fr	09876kol
<input checked="" type="checkbox"/>	6	diane	diana@gmail.com	d4 blond
<input checked="" type="checkbox"/>	5	desmond	desmond@gmail.com	1234
<input checked="" type="checkbox"/>	4	algele	alsou@gmail.com	1234

8 users

3. Configuration du Frontend projet

Bon à savoir: Dans un projet Django, les dossiers `templates` et `static` sont cruciaux pour gérer les fichiers HTML et les fichiers statiques (CSS, JavaScript, images, etc.).

voici comment se présente la structure du frontend de notre application



- **Le fichier base.html**

Le fichier `base.html` sert de modèle de base que les autres templates peuvent hériter. Il contient la structure HTML de base et inclut les fichiers CSS et JavaScript.

```

base.html - MonProjetDjango - Visual Studio Code
Fichier Edition Sélection Affichage Atteindre Exécuter Terminal Aide
EXPLORATEUR ... dj base.html
MONPROJETDJANGO
    > .idea
    > .venv
    < docker
        > dockerfile
        > requirements.txt
    < etudiant
        > __pycache__
        > migrations
    < static/etudiant
        > css
        > js
    < templates/etudiant
        dj addandshow...
        dj base.html
        dj updatestudent...
        __init__.py
        admin.py
        apps.py
dj base.html
etudiant > templates > etudiant > dj base.html
1  <!DOCTYPE html>
2  {% load static %}
3  <html lang="en">
4  <head>
5      <meta charset="UTF-8">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <title>Gestion Etudiant</title>
8      <link rel="stylesheet" href="{% static 'etudiant/css/bootstrap.css'%}">
9  </head>
10 <body>
11     <div class="container mt-5">
12         <h2 class="text-center alert alert-danger"> Gestion d'Etudiant</h2>
13         {%block content%} {%endblock content%}
14     </div>
15     <script src="{% static 'etudiant/js/jquery.js'%}"></script>
16     <script src="{% static 'etudiant/js/bootstrap.js'%}"></script>
17     <script src="{% static 'etudiant/js/popper.js'%}"></script>
18 </body>
19 </html>
  
```

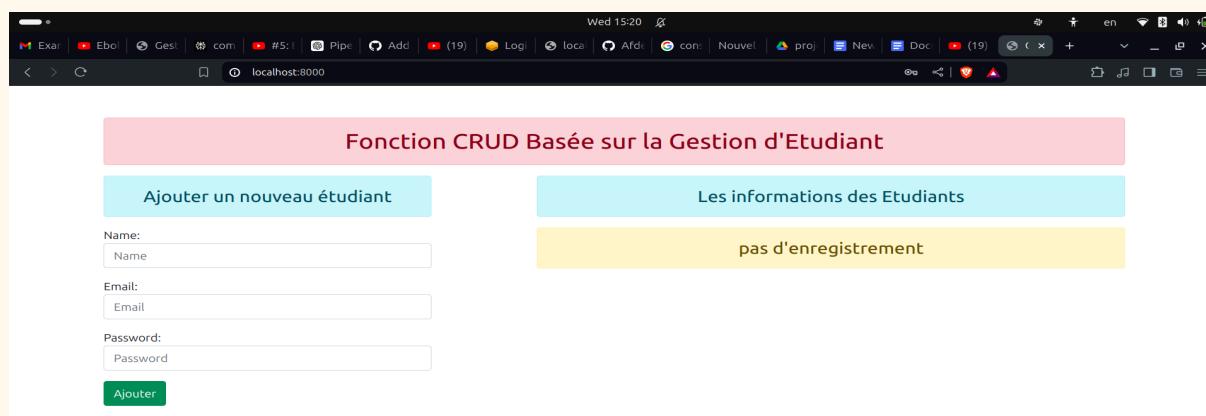
- **Le fichier addandshow.html**

Ce fichier permet d'ajouter un nouvel étudiant et de montrer la liste des étudiants ajoutés.

○ Le fichier updatestudent.html

Le fichier **updatestudent.html** permet de mettre à jour les informations d'un étudiant existant. Il est généré automatiquement à partir du modèle de l'étudiant.

```
dj updatestudent.html x
etudiant > templates > etudiant > dj updatestudent.html
1  {% extends 'etudiant/base.html' %}
2  {% block content %}
3  <div classe="row">
4      <div class="col-sm-8 offset-2">
5          <h4 class="alert alert-info">Modifier les informations d'un étudiant</h4>
6          <form action="" method="post">
7              {% csrf_token %}
8              {{ form.as_p }}
9              <input type="submit" class="btn btn-success" value="Modifier">
10             <a href="{% url 'addandshow' %}" class="btn btn-info">Retour</a>
11         </form>
12     </div>
13 </div>
14 {% endblock content %}
```



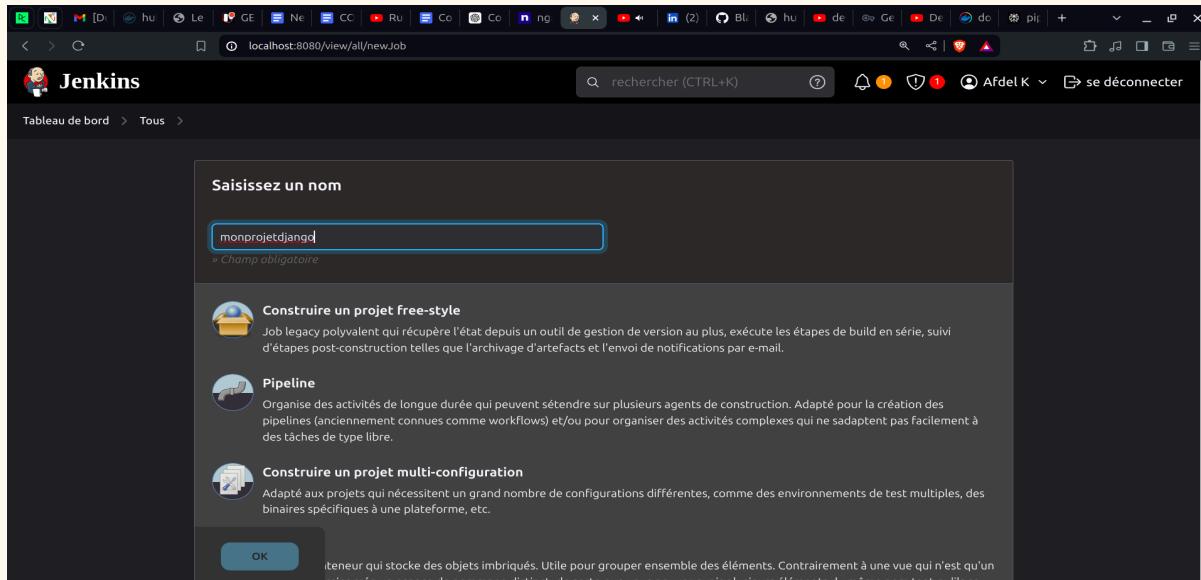
IV. Test et Déploiement

Résumé:

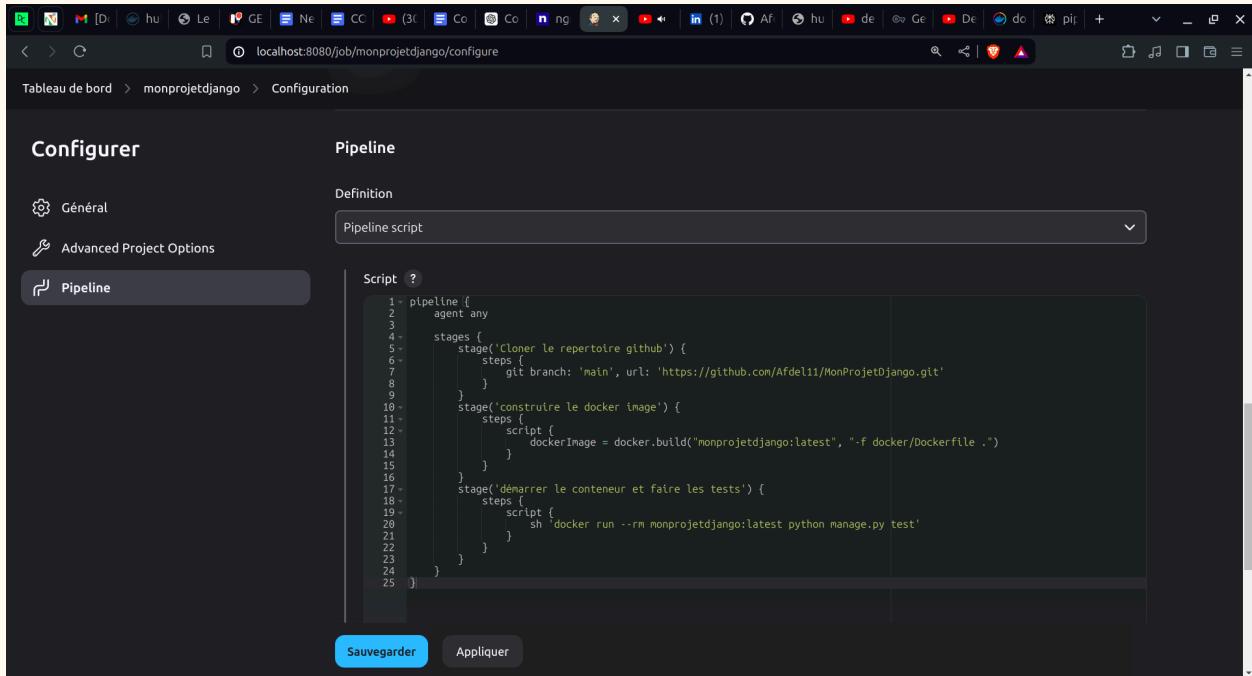
Nous avons configuré un **pipeline Jenkins** pour l'**intégration continue**, permettant de cloner le dépôt distant du projet, de **construire l'image Docker** et de faire les tests automatisés. Cela a été rendu possible grâce à **Ngrok**, qui a permis de connecter Jenkins à **GitHub** via un **webhook**, maintenant une connexion sécurisée pour les notifications de changement de code. Ce processus inclut la construction de l'**image Docker**, le déploiement manuel du conteneur et la vérification de l'application sur un **navigateur web**.

1. Configuration du Pipeline Jenkins

- Configurer un pipeline Jenkins pour l'intégration continue.
 - Création d'un nouveau job de type pipeline dans Jenkins.



- Ajouter les étapes pour cloner le dépôt distant, construire l'image Docker, et exécuter les tests automatisés.



2. Connexion de Jenkins à GitHub via Ngrok

- Utiliser Ngrok pour connecter Jenkins à GitHub.
 - Configurer Ngrok pour créer un tunnel sécurisé entre Jenkins et GitHub.

```

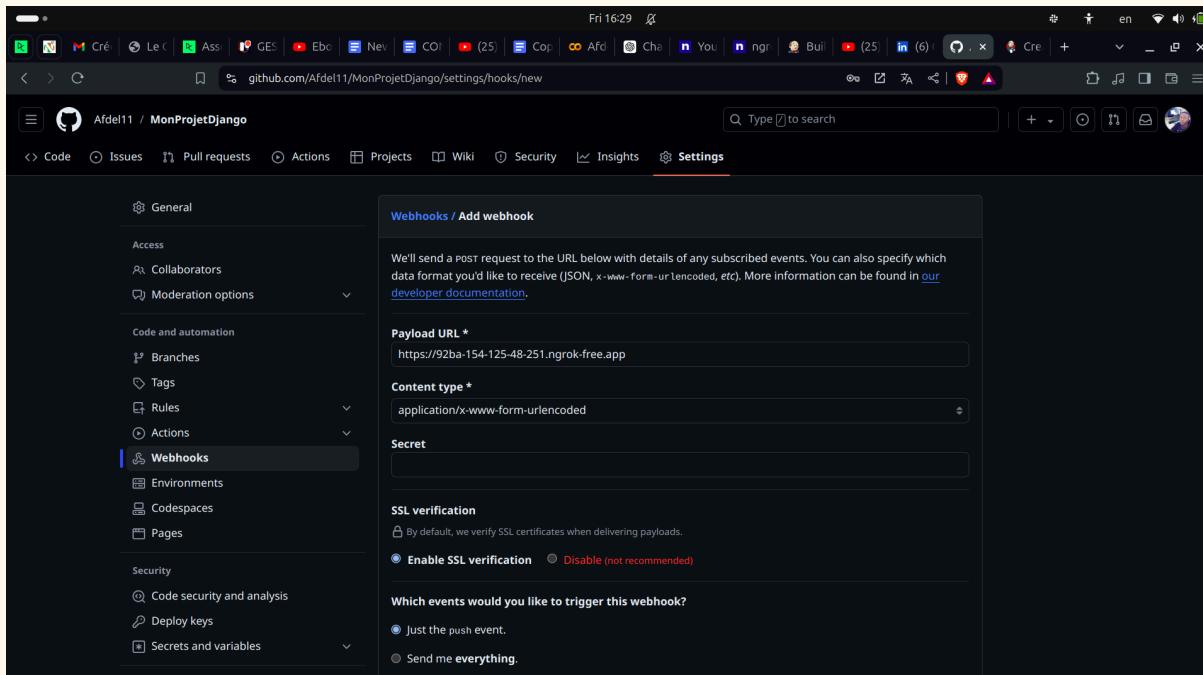
ngrok
Try our new Traffic Inspector: https://ngrok.com/r/ti

Session Status
Account          online
Afdel Desmond KOMBOU (Plan: Free)
Version          3.12.1
Region           Europe (eu)
Web Interface   http://127.0.0.1:4040
Forwarding      https://0b41-154-124-69-193.ngrok-free.app -> http://localhost:8080

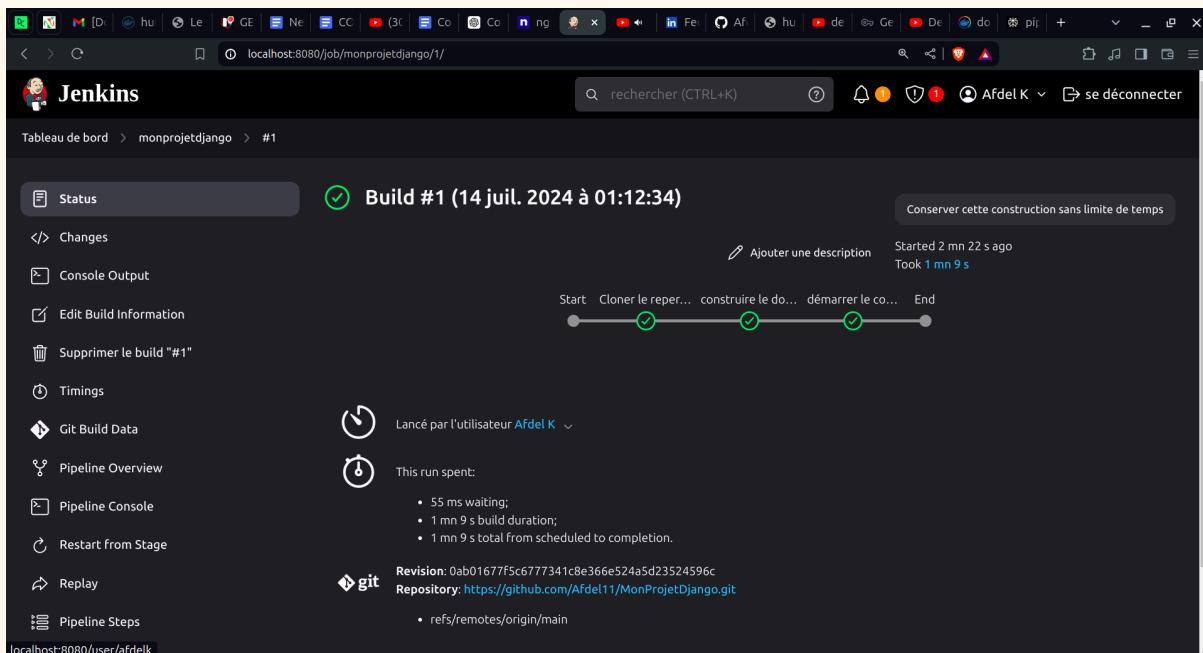
Connections
  ttl     opn      rt1      rt5      p50      p90
    0       0      0.00      0.00      0.00      0.00

```

- Configurer un webhook GitHub pour envoyer des notifications de changement de code à Jenkins via Ngrok.



○ Vérifications après exécution du pipeline.

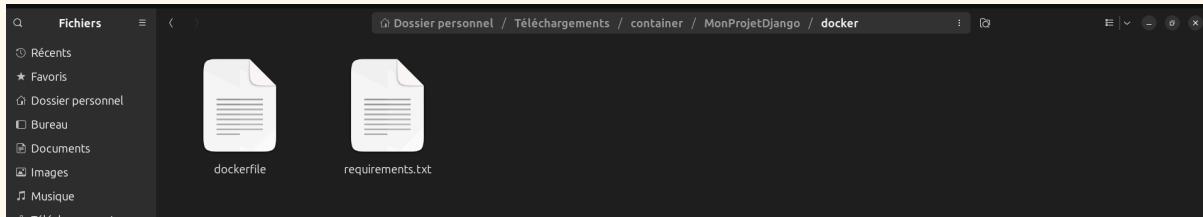


The top screenshot shows the Jenkins Pipeline Console for Build #1. It displays a successful stage named 'démarrer le conteneur et faire les tests'. The stage details show it started 3 minutes 55 seconds ago, took 7.2 seconds, and was successful. The command executed was `docker run --rm monprojetdjango:latest python manage.py test`, which completed in 6.8 seconds.

The bottom screenshot shows the Jenkins Pipeline Overview page for the 'projetdjango_Docker' pipeline. It lists the Pipeline Steps, including 'Start of Pipeline', 'node', 'stage', 'git', 'stage', 'script', 'isUnix', 'withEnv', 'stage', and 'script'. Each step is marked as successful (green status icon).

3. Construction et Déploiement de l'Image Docker

- Construire l'image Docker et déployer le conteneur.
 - Ajouter les instructions pour construire l'image Docker à partir du Dockerfile.



```

Fichiers   Dossier personnel / Téléchargements / container / MonProjetDjango / docker
Récents   dockerfile    requirements.txt
Favoris
Dossier personnel
Bureau
Documents
Images
Musique
MonProjetDjango
  +-- dockerfile
  +-- requirements.txt
  +-- etudiant
    +-- __pycache__
    +-- migrations
    +-- static
    +-- template...
      +-- dj
        +-- addandshow...
        +-- base.html
        +-- updatestudent...
      +-- __init__.py
      +-- admin.py
      +-- apps.py
      +-- forms.py
      +-- models.py
      +-- tests.py
      +-- views.py
  +-- MonProjetDjango

```

```

# Utiliser une image de base officielle de Python
FROM python:3.12-slim

# Définir le répertoire de travail dans le conteneur
WORKDIR /app

# Copier le fichier de dépendances dans le conteneur
COPY requirements.txt requirements.txt

# Installer les dépendances
RUN pip install --no-cache-dir -r requirements.txt

# Copier tout le contenu du répertoire actuel dans le répertoire de travail du conteneur
COPY . .

# Définir la variable d'environnement pour désactiver le buffering de sortie
ENV PYTHONUNBUFFERED=1

# port sur lequel l'application va écouter
EXPOSE 8000

# Commande à exécuter au démarrage du conteneur
CMD ["python", "manage.py", "runserver", "0.0.0.0:8000"]

```

○ Pousser l'image Docker sur Docker Hub.

```

Wed 15:18
Terminal
afdel@afdelk:~/Téléchargements/container/MonProjetDjango$ cd .
afdel@afdelk:~/Téléchargements/container/MonProjetDjango$ code .
afdel@afdelk:~/Téléchargements/container/MonProjetDjango$ docker build -t monprojetdjango:latest .
[+] Building 34.8s (10/10) FINISHED
  = [internal] load build definition from dockerfile
  = [internal] transfering dockerfile: 726B
  = [internal] load .dockerignore
  = [internal] transfering context: 2B
  = [internal] load metadata for docker.io/library/python:3.12-slim
  = [+] FROM docker.io/library/python:3.12-slim@sha256:d5f16749562233aa4bd26538771d76bf0dfd0a0ea7ea8771985e267451397ae4
  => resolving docker.io/library/python:3.12-slim@sha256:d5f16749562233aa4bd26538771d76bf0dfd0a0ea7ea8771985e267451397ae4
  => sha256:a64cae28bf14ee9ca961693d783fe7591aed311a7c2aad0b9898e1c310444a 3.51MB / 3.51MB
  => sha256:e9105b6a576033750931cd30420776ef569371f516702356e40d26538771d76bf0dfd0a0ea7ea8771985e267451397ae4 12.45
  => sha256:d5f16749562233aa4bd5c4bd26538771d76bf0dfd0a0ea7ea8771985e267451397ae4 1.45
  => sha256:7d0a0e0d5c4bd26538771d76bf0dfd0a0ea7ea8771985e267451397ae4 3.95
  => sha256:5eb109eb9ef435d2e4aa60bd5c388b26b391cb83112483602c4d191632850b02 1.94KB / 1.94KB
  => sha256:36d84f5948d07f0d0b1cb9ccbfb9a2609c5f25278fbe25d448611185940c03 6.67KB / 6.67KB
  => sha256:f11c1ada25e078479cc0d45312ea3b88047644191be0e698a7e07bf0d5badc 29.13MB / 29.13MB
  => sha256:808ac084160778f5927a9fc705c113dd3f2772b70c3f/bcce63ae0c38da707 230B / 230B
  => sha256:c4e9070483aaaf4f1c9afe105f5659021dd422d240b150d19c4d859a5e3594 2.88MB / 2.88MB
  => extracting sha256:f11c1ada25e078479cc0d45312ea3b8847644191be0e698a7e07bf0d5badc 3.25
  => extracting sha256:a64cae28bf14ee9ca961693d783fe7591aed311a7c2aad0b9898e1c3610444a 4.05
  => extracting sha256:0e903bae07b033750931cd30420776ef569371f516702356e40d26538771d76bf0dfd0a0ea7ea8771985e267451397ae4 0.75
  => extracting sha256:0e903bae07b033750931cd30420776ef569371f516702356e40d26538771d76bf0dfd0a0ea7ea8771985e267451397ae4 1.45
  => extracting sha256:0e903bae07b033750931cd30420776ef569371f516702356e40d26538771d76bf0dfd0a0ea7ea8771985e267451397ae4 0.05
  => extracting sha256:c4e9070483aaaf4f1c9afe105f5659021dd422d240b1550d19c4d859a5e3594 4.95
  => extracting sha256:c4e9070483aaaf4f1c9afe105f5659021dd422d240b1550d19c4d859a5e3594 4.85
  => extracting sha256:c4e9070483aaaf4f1c9afe105f5659021dd422d240b1550d19c4d859a5e3594 0.75
  => extracting sha256:c4e9070483aaaf4f1c9afe105f5659021dd422d240b1550d19c4d859a5e3594 0.15
  => extracting sha256:c4e9070483aaaf4f1c9afe105f5659021dd422d240b1550d19c4d859a5e3594 13.85
  => extracting sha256:c4e9070483aaaf4f1c9afe105f5659021dd422d240b1550d19c4d859a5e3594 4.65
  => extracting sha256:c4e9070483aaaf4f1c9afe105f5659021dd422d240b1550d19c4d859a5e3594 2.05
  => extracting sha256:c4e9070483aaaf4f1c9afe105f5659021dd422d240b1550d19c4d859a5e3594 0.05
  => extracting sha256:c4e9070483aaaf4f1c9afe105f5659021dd422d240b1550d19c4d859a5e3594 0.05
  => naming to docker.io/monprojetdjango:latest
afdel@afdelk:~/Téléchargements/container/MonProjetDjango$ 

```

○ Déployer le conteneur Docker manuellement.

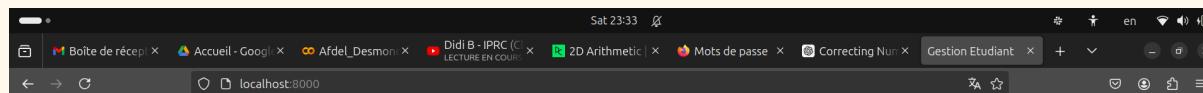
```
afdel@afdelk:~/Téléchargements/container/MonProjetDjango$ docker run -d -p 8000:8000 monprojetdjango:latest
7c3a01b284156e86371694a860473a775d0bbfb5e7aec2a051cb48e33311d
afdel@afdelk:~/Téléchargements/container/MonProjetDjango$
```

Docker Desktop interface showing the Images section. It displays 6 images with the following details:

Name	Tag	Status	Created	Size	Actions
monprojetdjango ef9a826540f1	latest	In use	5 minutes ago	222.53 MB	View Edit
somm... e1b4d2d56d95	latest	In use	1 month ago	574.36 MB	View Edit
somm... 86785a86efdd	latest	In use	1 month ago	574.36 MB	View Edit
nginx e784f4560448	latest	In use	2 months ago	187.65 MB	View Edit

```
afdel@afdelk:~/Téléchargements/container/MonProjetDjango$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS               NAMES
7c3a01b28415        monprojetdjango:latest   "python manage.py ru..."   2 minutes ago       Up 2 minutes      0.0.0.0:8000->8000/tcp, :::8000->8000/tcp   focused_shaw
afdel@afdelk:~/Téléchargements/container/MonProjetDjango$ docker logs 7c3a01b28415
Performing system checks...

Watching for file changes with StatReloader
System check identified no issues (0 silenced).
July 10, 2024 - 15:19:22
Django version 3.2.25, using settings 'MonProjetDjango.settings'
Starting development server at http://0.0.0.0:8000/
Quit the server with CONTROL-C.
[10/Jul/2024 15:19:55] "GET / HTTP/1.1" 200 1782
[10/Jul/2024 15:19:55] "GET /static/etudiant/js/popper.js HTTP/1.1" 200 20126
[10/Jul/2024 15:19:55] "GET /static/etudiant/js/jquery.js HTTP/1.1" 200 91903
[10/Jul/2024 15:19:55] "GET /static/etudiant/js/bootstrap.js HTTP/1.1" 200 59219
[10/Jul/2024 15:19:55] "GET /static/etudiant/css/bootstrap.css HTTP/1.1" 200 163873
Not Found: /favicon.ico
[10/Jul/2024 15:19:55] "GET /favicon.ico HTTP/1.1" 404 2513
afdel@afdelk:~/Téléchargements/container/MonProjetDjango$
```



The web application interface for "Gestion d'Etudiant".

Ajouter un nouveau étudiant

Name:

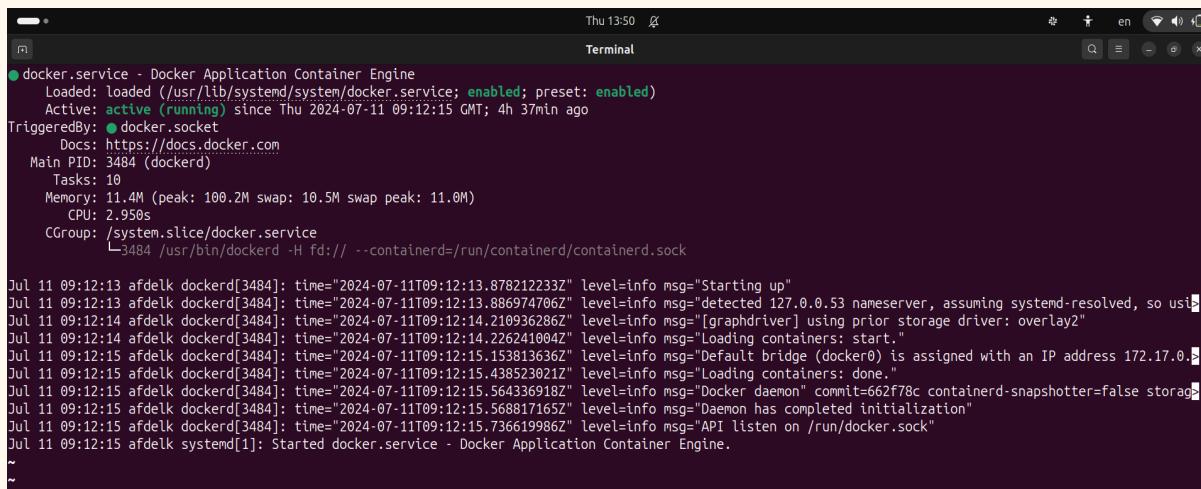
Email:

Password:

[Ajouter](#)

Les informations des Etudiants

ID	Nom	Email	Mot de Passe	Action
4	algege	alsou@gmail.com	1234	Modifier Supprimer
5	desmond	desmond@gmail.com	1234	Modifier Supprimer
6	diane	diana@gmail.com	d4 blond	Modifier Supprimer
7	guillaume bertrand	guibertrang@yahoo.fr	09876kol	Modifier Supprimer
8	mouhamed diop	amediop@dit.sn	ammedy	Modifier Supprimer
9	Madické Diop	madiop@gmail.com	mdkdp	Modifier Supprimer
10	boukar diallo	boukar_diallo@yabadou.sh	boubacar	Modifier Supprimer
11	mame diara	madiaratall@sns.pip	*****	Modifier Supprimer

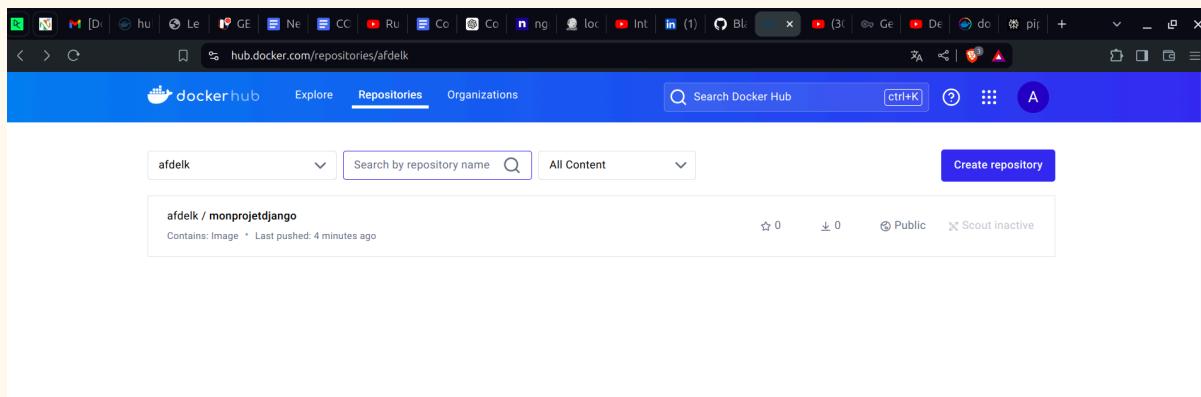


```

● docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; preset: enabled)
   Active: active (running) since Thu 2024-07-11 09:12:15 GMT; 4h 37min ago
     TriggeredBy: ● docker.socket
     Docs: https://docs.docker.com
 Main PID: 3484 (dockerd)
   Tasks: 10
      Memory: 11.4M (peak: 100.2M swap; 10.5M swap peak: 11.0M)
        CPU: 2.950s
      CGroup: /system.slice/docker.service
              └─3484 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

Jul 11 09:12:13 afdelk dockerd[3484]: time="2024-07-11T09:12:13.878212233Z" level=info msg="Starting up"
Jul 11 09:12:13 afdelk dockerd[3484]: time="2024-07-11T09:12:13.886974706Z" level=info msg="detected 127.0.0.53 nameserver, assuming systemd-resolved, so using prior storage driver: overlay2"
Jul 11 09:12:14 afdelk dockerd[3484]: time="2024-07-11T09:12:14.210936286Z" level=info msg="[graphdriver] using prior storage driver: overlay2"
Jul 11 09:12:14 afdelk dockerd[3484]: time="2024-07-11T09:12:14.226241804Z" level=info msg="Loading containers: start."
Jul 11 09:12:15 afdelk dockerd[3484]: time="2024-07-11T09:12:15.153813636Z" level=info msg="Default bridge (docker0) is assigned with an IP address 172.17.0.1"
Jul 11 09:12:15 afdelk dockerd[3484]: time="2024-07-11T09:12:15.438523621Z" level=info msg="Loading containers: done."
Jul 11 09:12:15 afdelk dockerd[3484]: time="2024-07-11T09:12:15.564336918Z" level=info msg="Docker daemon" commit=662f78c containerd-snapshotter=false storage-driver=overlay2
Jul 11 09:12:15 afdelk dockerd[3484]: time="2024-07-11T09:12:15.568817165Z" level=info msg="Daemon has completed initialization"
Jul 11 09:12:15 afdelk dockerd[3484]: time="2024-07-11T09:12:15.736619986Z" level=info msg="API listen on /run/docker.sock"
Jul 11 09:12:15 afdelk systemd[1]: Started docker.service - Docker Application Container Engine.
~
~
```

○ Vérification de l'Application sur Docker Hub



Après avoir exploré et utilisé une gamme d'outils modernes et efficaces tels que **Trello** pour la gestion de projet, **Slack** pour la communication en équipe, **VS Code** comme éditeur de code principal, **Django** comme framework de développement web, **GitHub** pour la gestion du code source et la collaboration, **Jenkins** pour l'intégration continue, **Docker** pour la conteneurisation de l'application, et **Ngrok** pour les tests en ligne, notre équipe a pu configurer un environnement **robuste** pour **concevoir, développer, tester et déployer** une application de gestion des étudiants qui implémente les fonctionnalités **CRUD**.

CONCLUSION

Le projet de gestion des étudiants a permis à notre équipe de se familiariser avec une gamme d'outils et de **technologies modernes**, de la gestion de projet à l'intégration continue et au déploiement en conteneurs. Chaque étape a contribué à garantir que l'application soit **robuste**, **testée** et prête pour la **production**. Nous remercions **M. Madické Diop** pour sa supervision et son soutien tout au long de ce projet.

Nous espérons que cette expérience sera **bénéfique pour nos futurs** projets et nous sommes impatients d'appliquer les connaissances acquises à de nouvelles initiatives.

ANNEXES:

[chatGPT](#)

[Apprendre django- youtube channel](#)

[django documentation](#)

NOS REMERCIEMENTS 😊