



Atelier service web

Enseignant: Nizar MAATOUG

Classe: SEM31

AU: 2023-2024

Aperçu du contenu

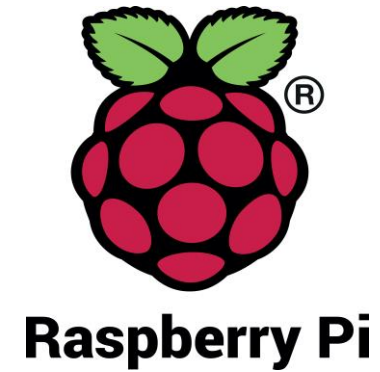
- ▶ Architecture de notre application
- ▶ Framework Django, django Rest:
 - ▶ Architecture
 - ▶ Installation
- ▶ Architecture REST: définition, implémentation avec Django
- ▶ Django channels:
 - ▶ Websockets: définition, implémentation avec Django
 - ▶ MQTT: implémentation avec Django
- ▶ Sécurité: Authentification et permissions
 - ▶ Session
 - ▶ JWT
- ▶ MongoDB
- ▶ Atelier Services web Firebase

Résultats

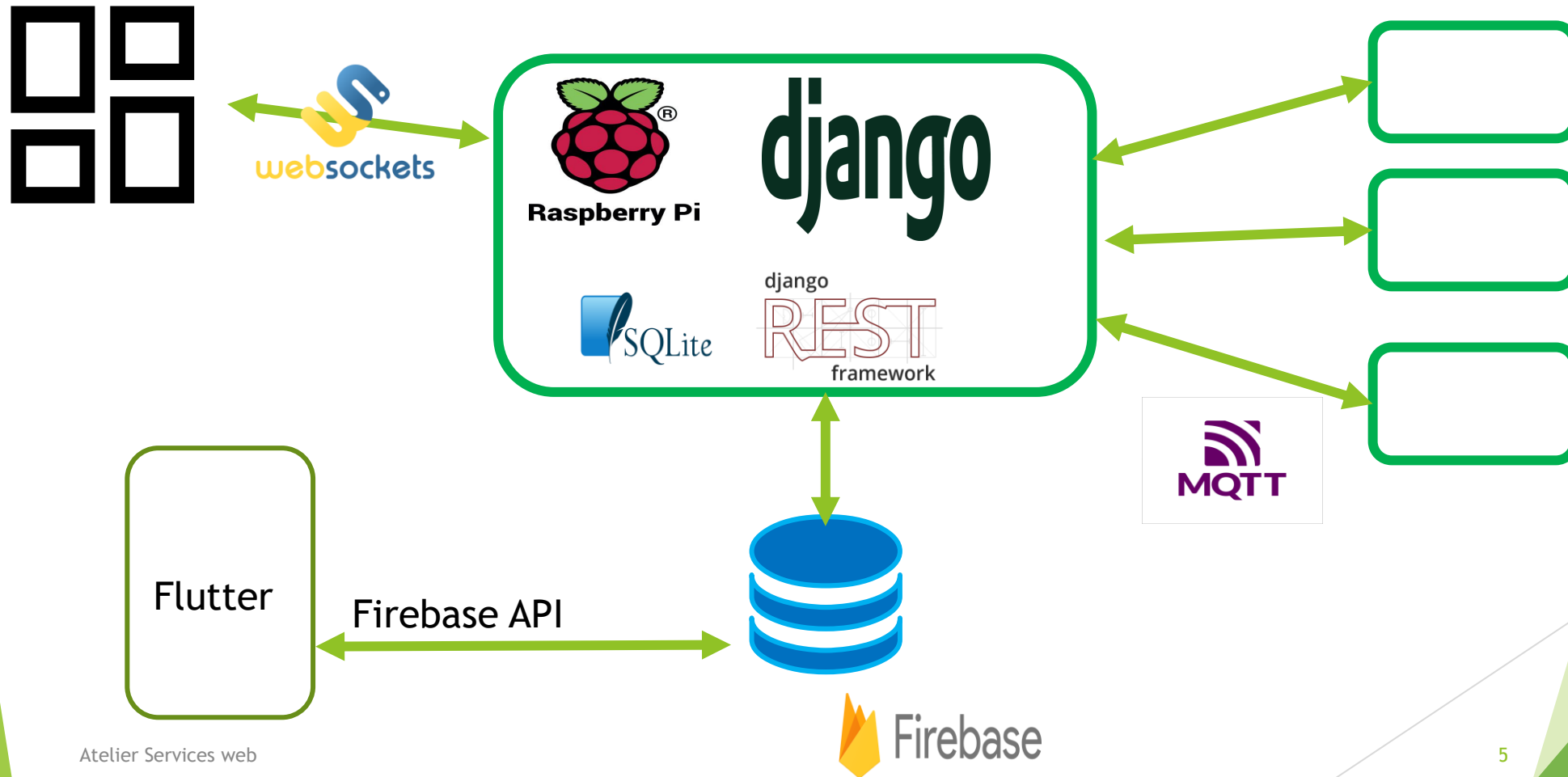
À la fin de ce module, vous serez en mesure de:

- ▶ Développer une application web avec le framework Django
 - ▶ Maîtriser l'architecture de ce framework
 - ▶ Maîtriser et développer un service web REST
 - ▶ Maîtriser et développer un webSocket
 - ▶ Communiquer avec MQTT
 - ▶ Alimenter une base de données relationnelles et de documents
- ▶ Intégrer tous ces concepts dans une architecture type.

prérequis



Architecture de notre application



django

Django

Architecture, installation

django

Framework web Python

Développer rapidement des applications web

Avec le minimum de code

Développé entre 2003 et 2005

Philosophie: Piles incluses

ORM

Templates

Forms

Admin

URL Mapping

Packages

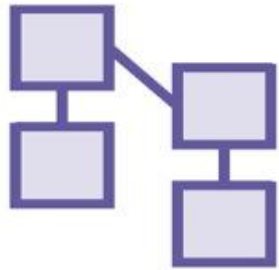
Polyvalent

- ▶ Django peut être (et a été) utilisé pour créer presque tous les genres de sites:
 - ▶ Site d'actualité
 - ▶ Gestionnaire de données
 - ▶ Wikis
 - ▶ Réseaux sociaux
- ▶ Qui utilise django ?
 - ▶ Instagram
 - ▶ Spotify
 - ▶ Youtube
 - ▶ Dropbox
 - ▶ ...

Autres

- ▶ Sécurisé
- ▶ Maintenable
- ▶ Scalable
- ▶ Portable

django: Architecture MVT



Model

Représente les données,
Assure le mapping
Objet/Relationnel

MVC: "Controller"



View

Reçoit une requête http,
effectue un traitement,
retourne une réponse http

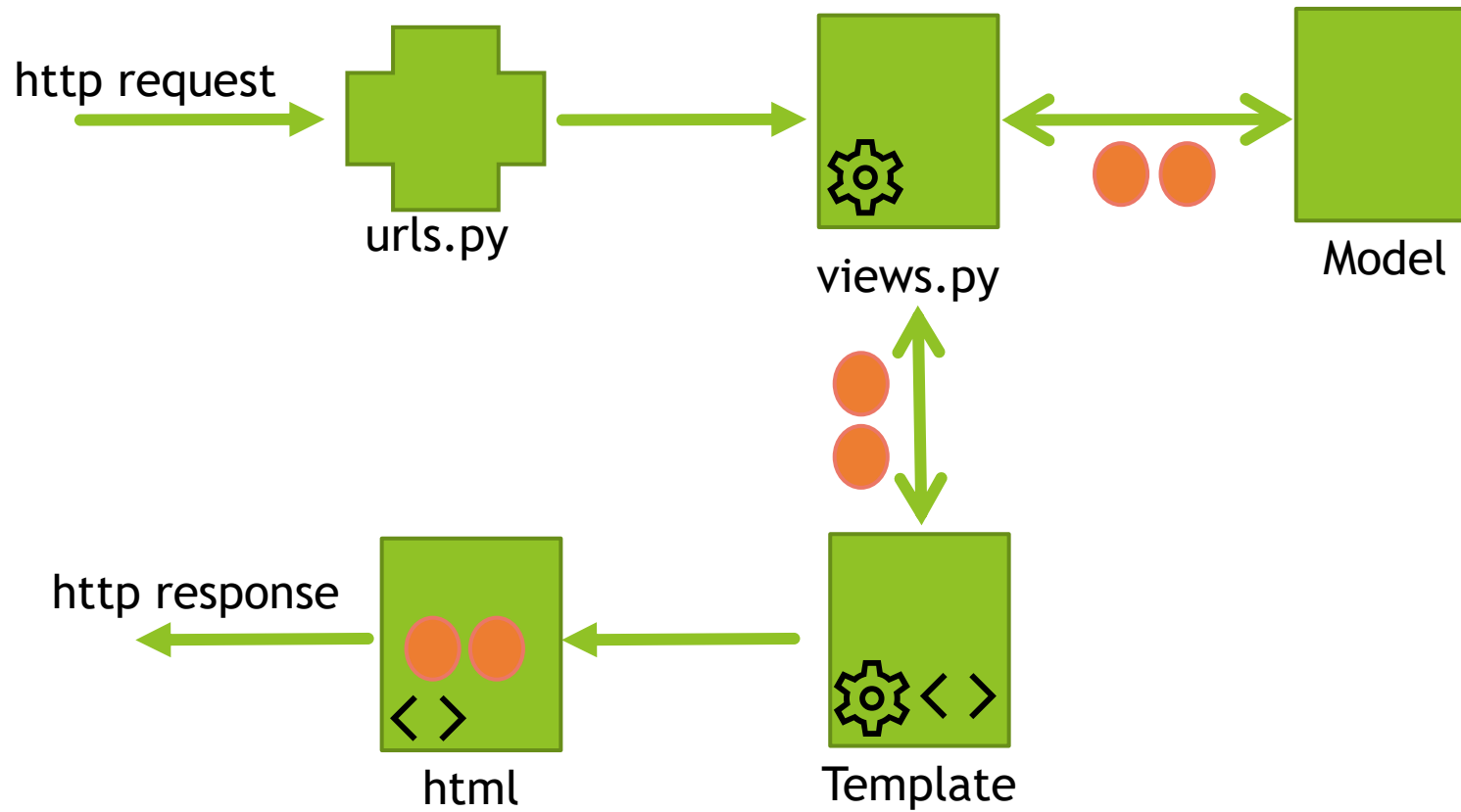
MVC: "View"



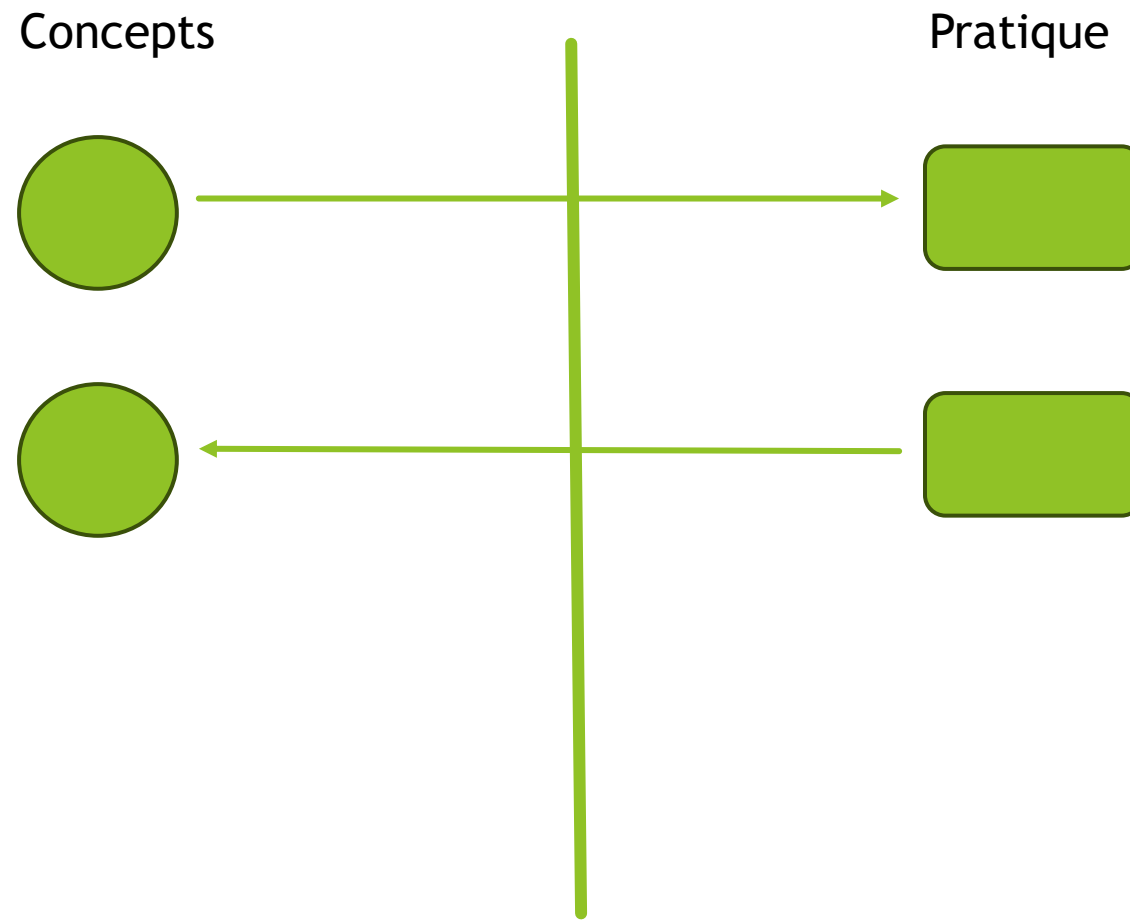
Template

Définit, génère la
présentation HTML

django: Architecture MVT



Approche





Système d'exploitation: windows, Linux, Mac OS



Python: Latest version 3.11



Éditeur de code: VS code, PyCharm

django: Première application

Étape par Étape

- ▶ Installation et configuration des outils
- ▶ Création et activation d'un environnement virtuel pour le projet
- ▶ Installation du framework django (virtual env activé)
- ▶ Sauvegarder les dépendances
- ▶ Création du projet django nommé **hello_word_project**
- ▶ Création de l'application **pages**
- ▶ Déclaration de l'application **pages**
- ▶ Première page web

Installation et configuration des outils

- ▶ Installer Python: <https://www.python.org/downloads/>
- ▶ Installer VS Code ou PyCharm

Environnement virtuel

Étape par Étape



- ▶ Ne pas installer les packages python globalement
- ▶ Toujours travailler dans un environnement virtuel
- ▶ Éviter les conflits de dépendances
- ▶ Travailler dans un contexte isolé.

Environnement virtuel

#créer un dossier pour le projet

```
$ cd framework_django\projects
```

```
$ mkdir helloworld
```

```
$ cd helloworld
```

#créer et activer l'environnement virtuel

```
$ python -m venv .venv
```

```
$ .venv\Scripts\Activate.ps1
```

```
(.venv) $ python -m pip install django
```

Étape par Étape

Sauvegarder les dépendances

#sauvegarder les dépendances

(.venv) \$ pip freeze > requirements.txt

```
requirements.txt X
requirements.txt
1  asgiref==3.7.2
2  Django==4.2.4
3  sqlparse==0.4.4
4  tzdata==2023.3
5
```

Étape par Étape

Créer le projet django_project

créer le projet django_project

(.venv) \$ django-admin startproject django_project .


lancer le projet dans VS code


(.venv) \$ code .


Étape par Étape

django_project: Anatomie

Étape par Étape

✓  django_project

>  __pycache__

 __init__.py

Marque le répertoire comme package => pouvoir importer les composants

 asgi.py

Standard Python pour application et serveur asynchrone

 settings.py


Contrôle l'application django via des configurations

 urls.py

Binding entre requête http et traitement associé

 wsgi.py

Web Server Gateway Interface

 manage.py

Ne fait pas partie du projet, mais utile pour exécuter des commandes: runserver, créer nouvelle application,...

Lancer le projet

| # démarrer le projet

| **(.venv)** \$ python manage.py runserver

| # appliquer les migrations

| **(.venv)** \$ python manage.py migrate

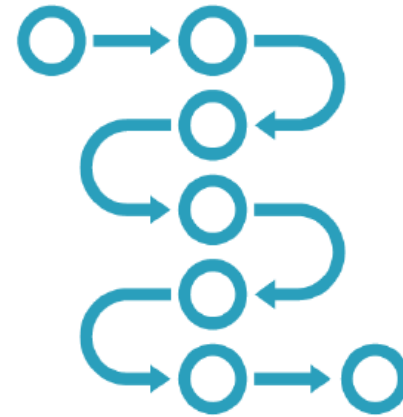
Étape par Étape

Migrations



Models

Classes Python
Associées aux tables de
la BD



Migrations

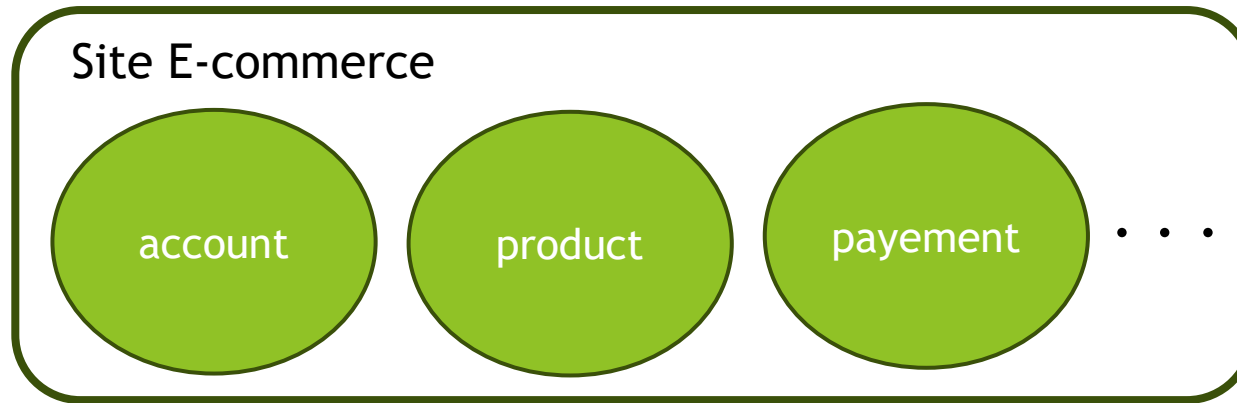
Ensemble de scripts
décrivant le schéma de
la BD

Étape par Étape

Application django: pages

Étape par Étape

- ▶ django utilise les concepts **projet** et **application** pour maintenir le code "clean"
- ▶ un projet django peut contenir plusieurs applications
- ▶ Chaque application contrôle une fonctionnalité isolée du projet.



Application django: pages

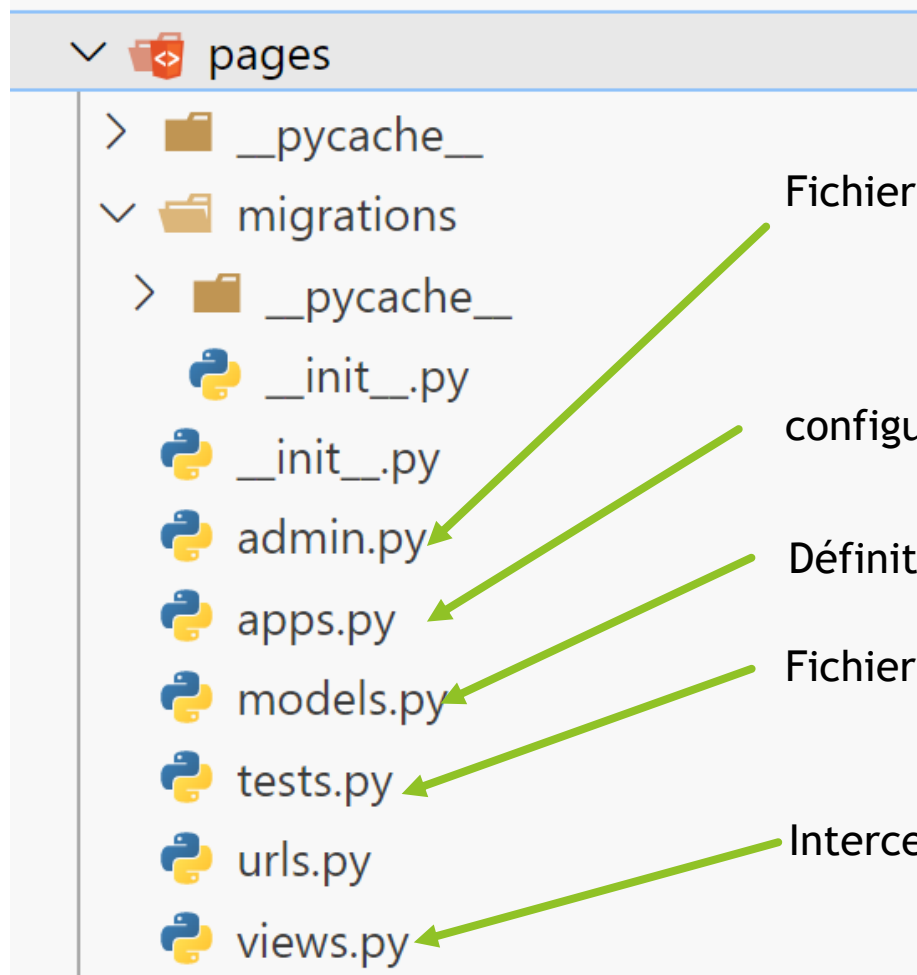
| # créer l'application pages

| (.venv) \$ python manage.py startapp pages

Étape par Étape

Application django: Anatomie

Étape par Étape



Fichier de configuration pour le module Admin-app

configuration de l'application pages

Définition du modèle

Fichier dans lequel on code les tests

Intercepter les requêtes, traitement, retourner une réponse

Déclaration de l'application pages

Étape par Étape

Application definition

```
INSTALLED_APPS = [  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
    "pages", #new  
]
```

Première page web: views.py

Étape par Étape

views.py ×

pages > views.py > ...

```
1 from django.shortcuts import render
2 from django.http import HttpResponse
3
4 # Create your views here.
5
6 def homePageView(request):
7     return HttpResponse("Hello word")
8
```

Première page web: pages.urls.py

Étape par Étape

urls.py ×

pages > urls.py > ...

```
1 from django.urls import path
2
3 from .views import homePageView
4
5
6 urlpatterns=[
7     path('',homePageView, name="home"),
8 ]
```

Première page web: urls.py

Étape par Étape

urls.py ×

django_project > urls.py > ...

```
17 from django.contrib import admin
18 from django.urls import path,include
19
20 urlpatterns = [
21     path('admin/', admin.site.urls),
22     path('',include("pages.urls")),
23 ]
24
```



- ▶ Django: framework web Python
- ▶ Développement web simple et rapide
- ▶ Piles incluses: ORM, Admin module, ...
- ▶ Architecture MVT (Model, View, Template)
- ▶ Un projet django, plusieurs applications
- ▶ Virtual environnement

Application: Meeting Planner

Résultats

- ▶ Maîtriser l'architecture MVT
- ▶ Découvrir les concepts de base de django:
 - ▶ Model
 - ▶ View
 - ▶ Template
 - ▶ URLs
 - ▶ Admin panel
 - ▶ Forms

Meeting Planner

- ▶ Permet de planifier les réunions
- ▶ Planifier une **réunion**:
 - ▶ Définir la date, la durée
 - ▶ affecter une **salle** à cette réunion

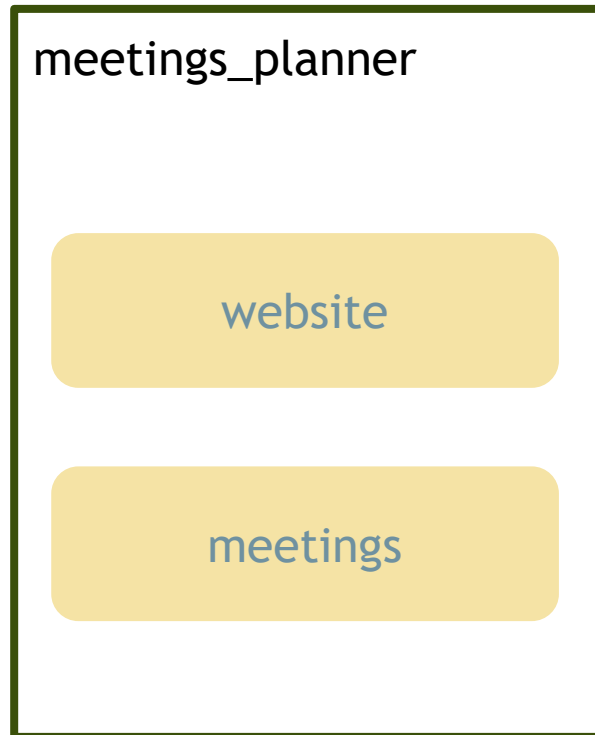
Démo de l'application

Démarche

- ▶ Créer le projet **meetings_planner**
 - ▶ Créer et activer l'environnement virtuel
 - ▶ Installer les dépendances
 - ▶ Créer le projet et les applications django
- ▶ Implémenter les pages web (accueil, about)
- ▶ Implémenter le modèle **Meeting**
- ▶ Admin Panel:
 - ▶ Créer un **superuser**
 - ▶ Gérer le modèle **Meeting**
- ▶ Gérer les **Meeting**
- ▶ implémenter le modèle Room
- ▶ Définir l'association Meeting-----Room

Étape par Étape

Architecture du projet



django apps



Python package

Contient models, views, templates, urls

Les projets django contiennent plusieurs apps

Les Apps peuvent être réutilisées

Maintenir les apps petites et simples

Création du projet

#créer un dossier pour le projet

```
$ cd framework_django\projects
```

```
$ mkdir meetings_planner
```

```
$ cd meetings_planner
```

#créer et activer l'environnement virtuel

```
$ python -m venv my_env
```

```
$ my_env\Scripts\Activate
```

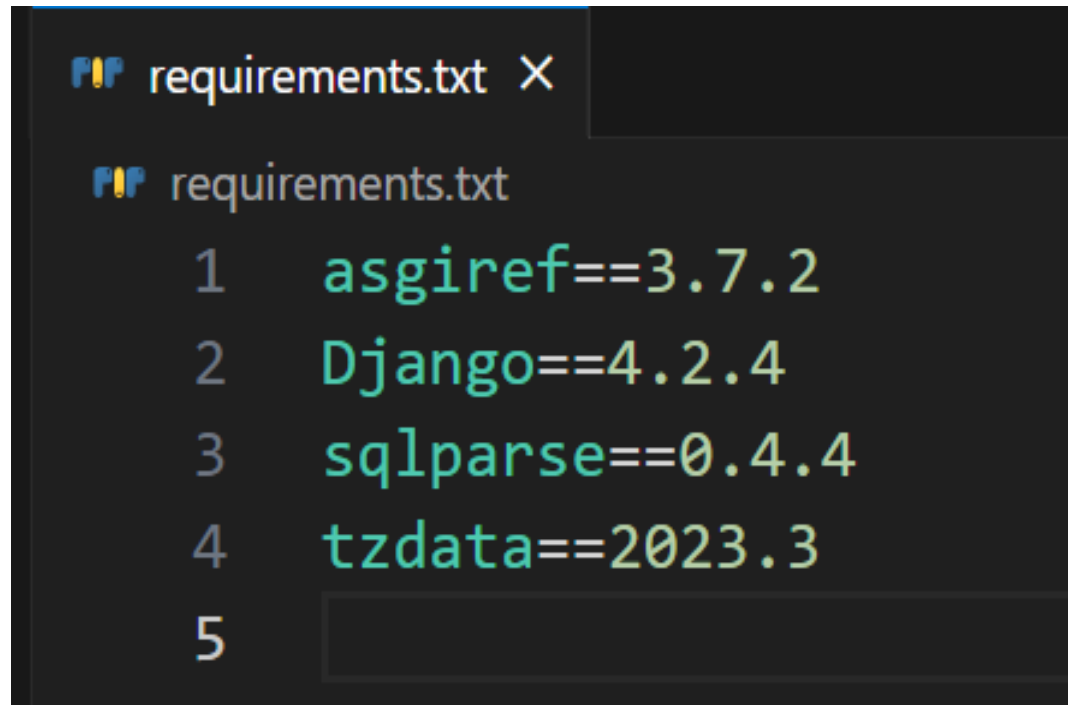
```
(my_env) $ python -m pip install django
```

Étape par Étape

Sauvegarder les dépendances

#sauvegarder les dépendances

(my_env) \$ pip freeze > requirements.txt



```
requirements.txt X
requirements.txt
1  asgiref==3.7.2
2  Django==4.2.4
3  sqlparse==0.4.4
4  tzdata==2023.3
5
```

Étape par Étape

Création des applications

créer le projet django_project

(.venv) \$ django-admin startproject meetings_planner .

lancer le projet dans VS code

(.venv) \$ code .

créer l'application website

(my_env) \$ python manage.py startapp website









créer l'application meetings

(my_env) \$ python manage.py startapp meetings

Étape par Étape

Structure du projet

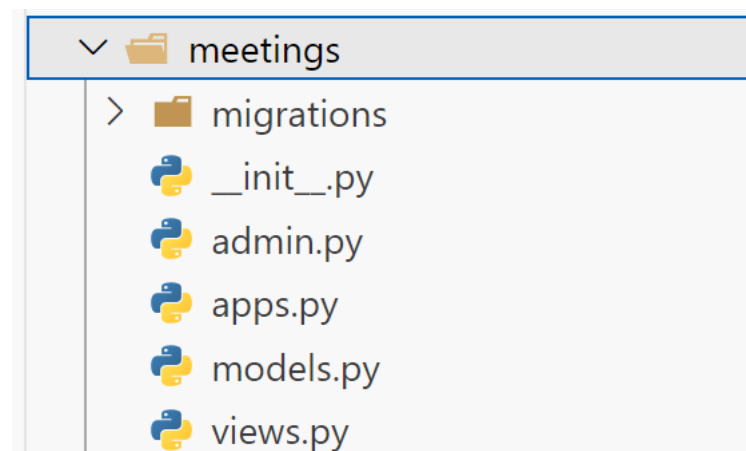
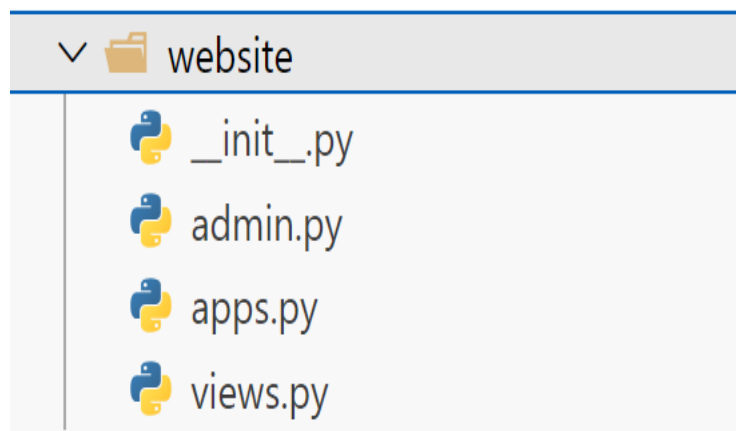
✓ MEETINGS_PLANNER

- >  meetings
- >  meetings_planner
- >  my_env
- >  website
-  .gitignore
-  db.sqlite3
-  manage.py
-  requirements.txt

Étape par Étape

Structure du projet

Étape par Étape



Settings.py: Déclaration des applications

Étape par Étape

```
INSTALLED_APPS = [  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
    'website', #new  
    'meetings', #new  
]
```

Application des migrations

| # appliquer les migrations

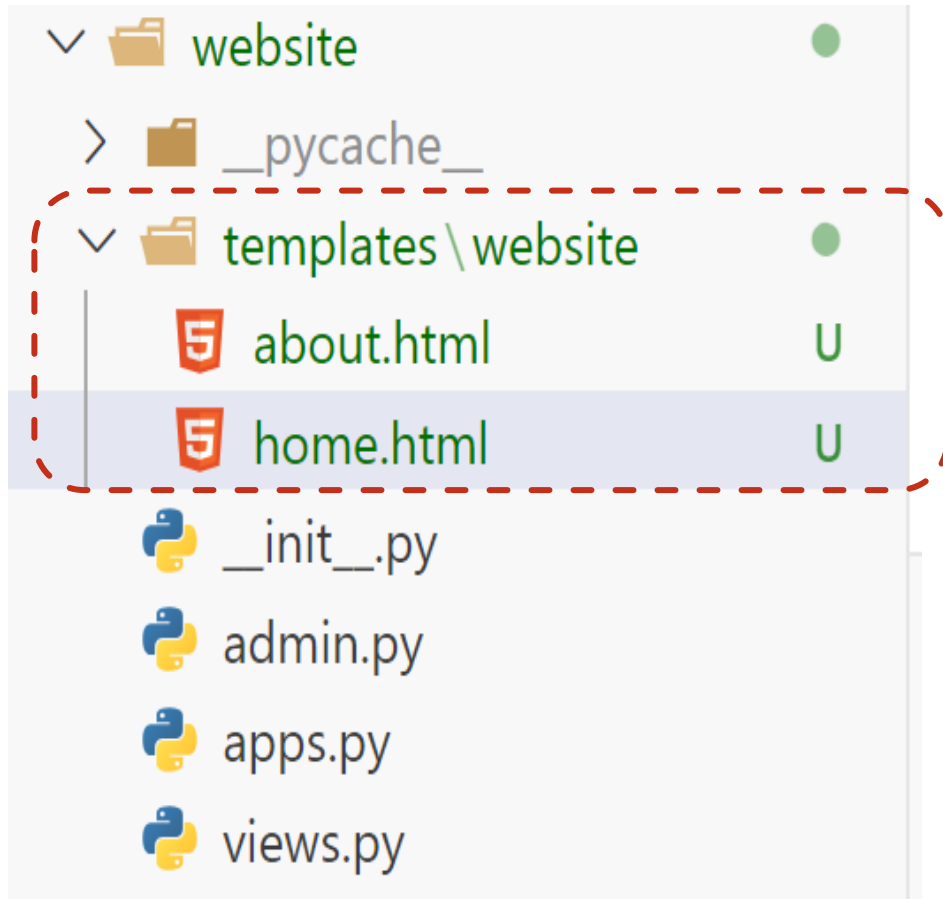
| (my_env) \$ python manage.py migrate

| # démarrer le projet

| (my_env) \$ python manage.py runserver

Étape par Étape

website: home.html & about.html



Étape par Étape

website: home.html

Étape par Étape

```
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Meeting planer: home page</title>
</head>
<body>
  <h2>Meeting planner application from Tek-up</h2>
  <h3>Lite des réunions</h3>
</body>
</html>
```

website: about.html

Étape par Étape

```
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Meeting planer: about page</title>
</head>
<body>
  <h2>Meeting planner application from Tek-up</h2>

</body>
</html>
```


website: views.py

```
from django.shortcuts import render
```

```
# Create your views here.
```

```
def home_view(request):  
    return render(request, "website/home.html")
```

```
def about_view(request):  
    return render(request, "website/about.html")
```

Étape par Étape

website: urls.py

Étape par Étape

```
from django.urls import path

from . import views

#domain.com/website/...
urlpatterns=[
    path('',views.home_view, name='home'),
    path('about',views.about_view, name='about'),
]
```

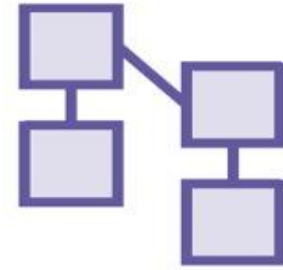
meetings_planner: urls.py

Étape par Étape

```
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('website/', include('website.urls')), #new
]
```

Créer le modèle



Django Models

Enregistrer les objets Python dans la BD

Les classes Models sont mappés à des tables

Les attributs sont mappés à des colonnes

SQL est généré

Create/Update tables (migrations)

Insert/Update/Delete lignes (admin)

Models

- Documentation: <https://docs.djangoproject.com/fr/4.2/topics/db/models/>

Démarche: Models

- ▶ Créer les classes modèles
- ▶ Créer les migrations
- ▶ appliquer les migrations
- ▶ Gérer le modèle avec Admin-interface

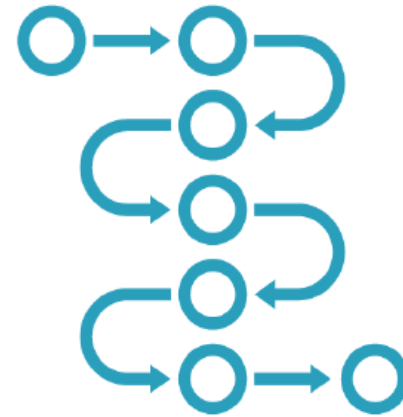
Étape par Étape

Migrations



Models

Classes Python
Associées aux tables de
la BD



Migrations

Ensemble de scripts
décrivant le schéma de
la BD

Étape par Étape

Show migrations

Étape par Étape

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
(.venv) PS C:\pc-nizar\Tek-up\AU2023-2024-S1\django\projects\meeting> python .\manage.py showmigrations
admin
[ ] 0001_initial
[ ] 0002_logentry_remove_auto_add
[ ] 0003_logentry_add_action_flag_choices
auth
[ ] 0001_initial
[ ] 0002_alter_permission_name_max_length
[ ] 0003_alter_user_email_max_length
[ ] 0004_alter_user_username_opts
[ ] 0005_alter_user_last_login_null
[ ] 0006_require_contenttypes_0002
[ ] 0007_alter_validators_add_error_messages
[ ] 0008_alter_user_username_max_length
[ ] 0009_alter_user_last_name_max_length
[ ] 0010_alter_group_name_max_length
[ ] 0011_update_proxy_permissions
[ ] 0012_alter_user_first_name_max_length
contenttypes
[ ] 0001_initial
[ ] 0002_remove_content_type_name
sessions
[ ] 0001_initial
website
```

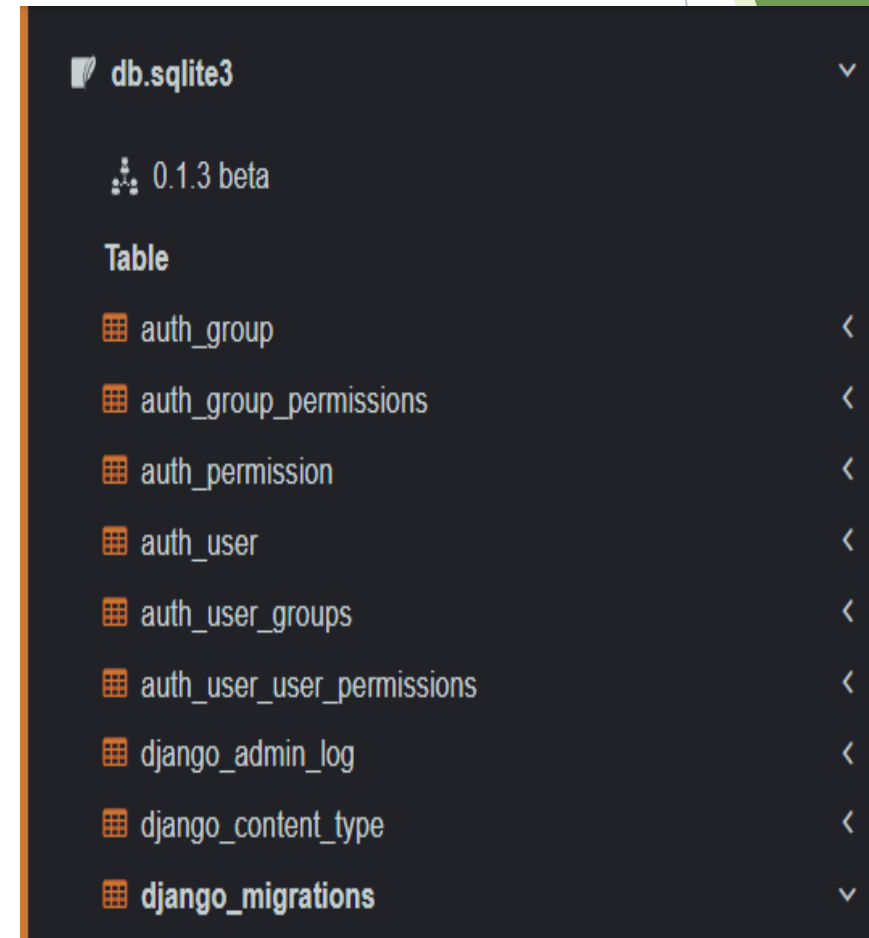
Appliquer les migrations initiales

Étape par Étape

```
# appliquer les migrations
```

```
(my_env) $ python manage.py migrate
```

Afficher le schéma de la BD: <https://sqliteonline.com/>



Implémenter la classe modèle Meeting

Étape par Étape

meetings.models.py

```
from django.db import models

# Create your models here.

class Meeting(models.Model):
    title=models.CharField(max_length=200)
    date=models.DateField()
```

Créer et appliquer la migration

Étape par Étape

créer la migration

(my_env) \$ python manage.py makemigrations

afficher SQL de la migration

(my_env) \$ python manage.py sqlmigrate meetings 0001

appliquer la migration

(my_env) \$ python manage.py migrate meetings

<https://sqliteonline.com/>

Migration Workflow

Step 1: Change Model code

Step 2: Generate migration script (check it!)

```
python manage.py makemigrations
```

Optional: Show migrations

```
python manage.py showmigrations
```

Optional: Show SQL for specific migration

```
python manage.py sqlmigrate appname migrationname
```

Step 3: Run migrations

```
python manage.py migrate
```

Django Admin

Deux parties de toute application web



End-users



Admin

django Admin

CRUD Model

Gestion Sécurité

Admin panel: éditer les modèles

Étape par Étape

- ▶ Enregistrer le modèle dans **admin.py**
- ▶ Créer un **superuser** pour pouvoir administrer l'application
- ▶ Démarrer l'application et connecter en tant que **superuser**
- ▶ Gérer le modèle

meetings.admin.py: enregistrer Meeting

Étape par Étape

```
from django.contrib import admin
```

```
from .models import Meeting
```

```
admin.site.register(Meeting)
```

Admin interface

Étape par Étape

127.0.0.1:8000/admin/login/?next=/admin/

Livre_ENI tekup Courrier - Nizar MA...

Django administration

Username:

Password:

Log in

Admin interface: créer superuser

Étape par Étape

```
# créer superuser
```

```
(my_env) $ python manage.py createsuperuser
```

Admin interface: ajouter des meetings

Étape par Étape

← → ↻ 🏠 ⓘ 127.0.0.1:8000/admin/meetings/meeting/add/

★ Bookmarks 📁 Livre_ENI 📁 tekup 📧 Courrier - Nizar MA...

Django administration

Home > Meetings > Meetings > Add meeting

Start typing to filter...

AUTHENTICATION AND AUTHORIZATION

Groups + Add

Users + Add

MEETINGS

Meetings + Add

Add meeting

Title:

Date: Today | 📅

Note: You are 2 hours ahead of server time.

«

Admin interface: ajouter des meetings

Étape par Étape

← → ↻ 🏠 ⓘ 127.0.0.1:8000/admin/meetings/meeting/

★ Bookmarks 📁 Livre_ENI 📁 tekup 📧 Courrier - Nizar MA...

Django administration

Home > Meetings > Meetings

Start typing to filter...

AUTHENTICATION AND AUTHORIZATION

Groups + Add

Users + Add

MEETINGS

Meetings + Add

Select meeting to change

Action: ----- ▾ Go 0 of 2 selected

<input type="checkbox"/>	MEETING
<input type="checkbox"/>	Meeting object (2)
<input type="checkbox"/>	Meeting object (1)

2 meetings

Exercice: Mettre à jour Meeting

Étape par Étape

```
from django.db import models
from datetime import time
```

```
# Create your models here.
```

```
class Meeting(models.Model):
    title=models.CharField(max_length=200)
    date=models.DateField()
    start_time=models.TimeField(default=time(9))
    duration=models.IntegerField(default=1)

    def __str__(self):
        return f"{self.title} at {self.start_time} on {self.date}"
```

Créer et appliquer la migration

Étape par Étape

Exercice: Ajouter le modèle Room

Étape par Étape

```
class Room(models.Model):  
    name = models.CharField(max_length=50)  
    floor = models.IntegerField()  
    room_number = models.IntegerField()  
  
    def __str__(self):  
        return f"{self.name}: room {self.room_number} on floor {self.floor}"
```

Ajouter association one to many

Étape par Étape

```
class Meeting(models.Model):
    title = models.CharField(max_length=200)
    date = models.DateField()
    start_time = models.TimeField(default=time(9))
    duration = models.IntegerField(default=1)
    room = models.ForeignKey(Room, on_delete=models.CASCADE)

    def __str__(self):
        return f"{self.title} at {self.start_time} on {self.date}"
```

Créer et appliquer la migration

Étape par Étape

meetings.views

Étape par Étape

```
def detail(request, id):  
    meeting = Meeting.objects.get(pk=id)  
    return render(request, "meetings/detail.html", {"meeting": meeting})
```

```
def detail(request, id):  
    meeting = get_object_or_404(Meeting, id)  
    return render(request, "meetings/detail.html", {"meeting": meeting})
```

website.views

Étape par Étape

```
def home_view(request):  
    context={'nbre_meeting': Meeting.objects.count()}  
    return render(request, "website/home.html", context=context)  
  
def about_view(request):  
    return render(request, "website/about.html")
```

Templates

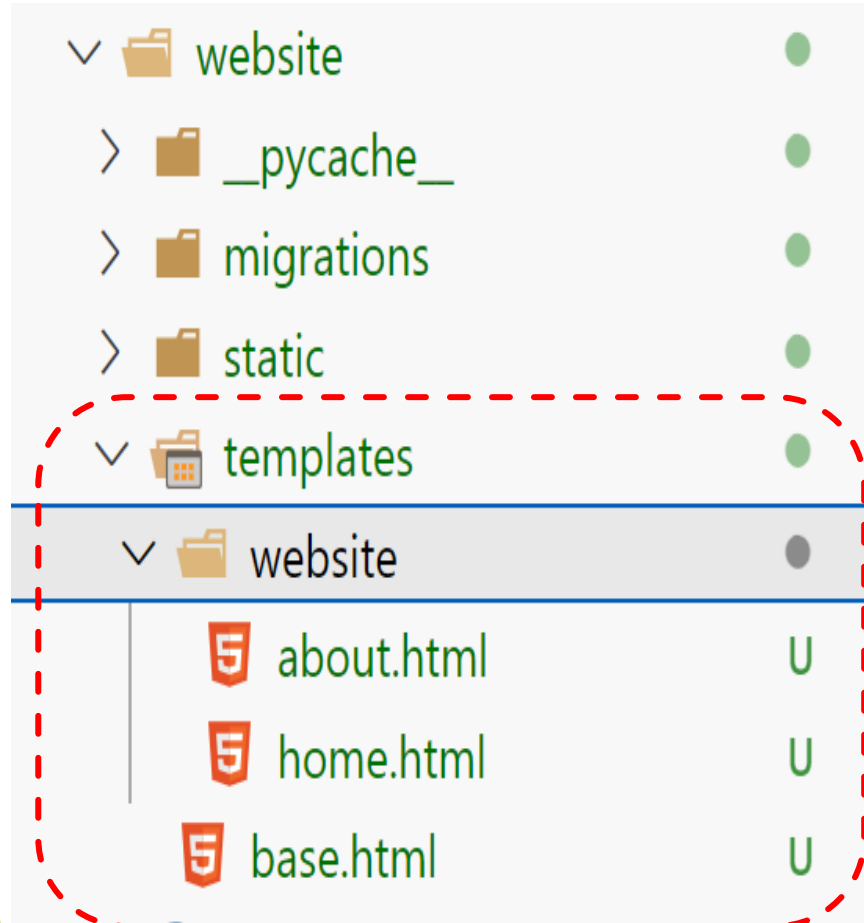
Générer les pages web

Template variables

Template tags

Template inheritance

Templates: héritage



Templates: héritage (base.html)

Étape par Étape

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">

  <title>{% block title %}{% endblock %}</title>

</head>
<body>
  {% block content %}
  {% endblock %}
</body>
</html>
```


Templates: héritage (about.html)

Étape par Étape

```
{% extends "base.html" %}

{% block title %}about{% endblock %}

{% block content %}
    <h2>Meeting planner application from Iset Bizerte</h2>

{% endblock %}
```

website: home.html v1

Étape par Étape

```
{% extends "base.html" %}
```

```
{% block title %}HOME{% endblock %}
```

```
{% block content %}
```

```
    <h2>Welcome to the meeting planner</h2>
```

```
    <p>there are currently {{nbre_meeting}} meetings</p>
```

```
{% endblock %}
```

website: home.html v2

Étape par Étape

```
<h2>Meetings</h2>
  <ul>
    {% for meeting in meetings %}
      <li>
        <a href="{% url 'detail' meeting.id %}">
          {{ meeting.title }}
        </a>
      </li>
    {% endfor %}
  </ul>
```

website: home.html v3

Étape par Étape

```
<h2>Meetings</h2>
  <ul>
    {% for meeting in meetings %}
      <li>
        <a href="{% url 'detail' meeting.id %}">
          {{ meeting.title }}
        </a>
      </li>
    {% endfor %}
  </ul>
  <a href="{% url 'rooms' %}">Rooms list</a>
```

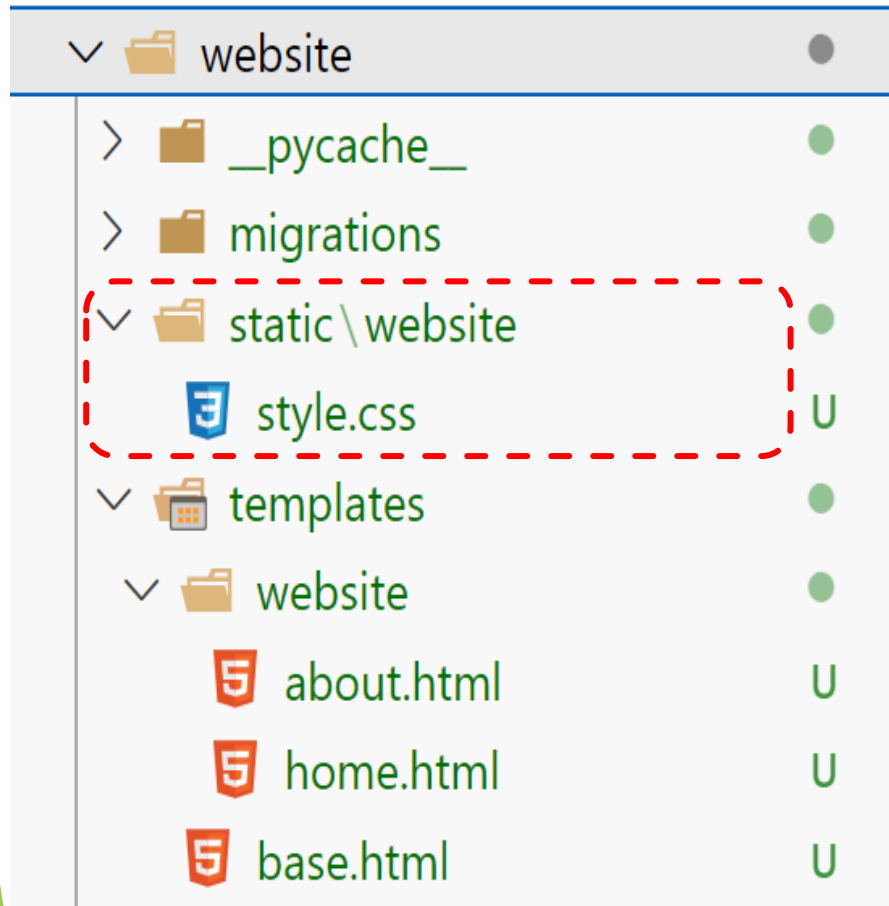
website.views v3

Étape par Étape

```
def home_view(request):  
    context={'meetings': Meeting.objects.all()}  
    return render(request, "website/home.html", context=context)
```

Templates: static files

Étape par Étape



```
body {  
    font-family: sans-serif;  
    color: cornflowerblue;  
    background-color: floralwhite;  
}
```

Templates: static files (settings.py)

Étape par Étape

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

STATIC_URL = 'static/'
```

Templates: static files (base.html)

Étape par Étape

```
{% load static %}
```

```
<!DOCTYPE html>
```

```
<html lang="en">
```

```
<head>
```

```
    <meta charset="UTF-8">
```

```
    <title>{% block title %}{% endblock %}</title>
```

```
    <link rel="stylesheet"
          href="{% static 'website/style.css' %}">
```

```
</head>
```

```
<body>
```

```
    {% block content %}
```

```
    {% endblock %}
```

```
</body>
```

```
</html>
```


Exercice

- ▶ Ajouter une page qui liste tous les rooms:
 - ▶ Views
 - ▶ template
 - ▶ url mapping

django forms

Valider les entrées utilisateurs

Automatise le code répétitif
sécurisé

Hautement personnalisé

Formulaire: Meeting

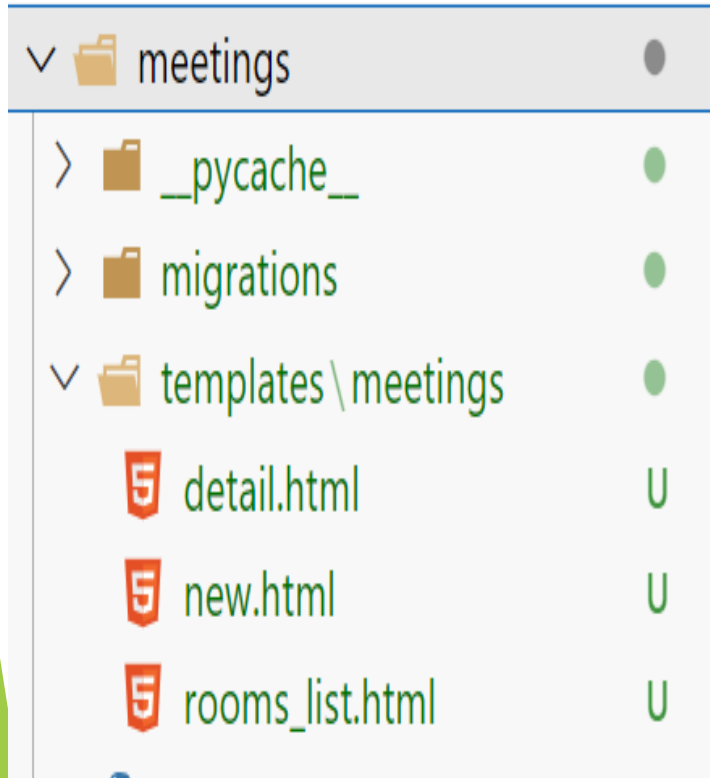
Étape par Étape

```
class MeetingForm(ModelForm):
    class Meta:
        model = Meeting
        fields = '__all__'
        widgets = {
            'date': DateInput(attrs={"type": "date"}),
            'start': TimeInput(attrs={"type": "time"}),
            'duration': TextInput(attrs={"type": "number", "min": "1", "max": "4"})
        }

    def clean_date(self):
        d = self.cleaned_data.get("date")
        if d < date.today():
            raise ValidationError("Meetings cannot be in the past")
        return d
```

Formulaire: new.html

Étape par Étape



```
{% extends "base.html" %}
{% block title %}New Meeting{% endblock %}

{% block content %}
<h1>Plan a new meeting</h1>
<form method="post">
    <table>
        {{ form }}
    </table>
    {% csrf_token %}
    <button type="submit">Create</button>
</form>
{% endblock %}
```

Exercice

- ▶ Ajouter tout le nécessaire pour permettre l'ajout d'une salle de réunion



- ▶ Architecture MVT
- ▶ Models: définition, manipulation, associations
- ▶ Admin-panel: CRUD, sécurité
- ▶ Templates: variables, tags, static files, héritage
- ▶ Forms: mapping entre templates et Models

Application: gestion grandeurs

Démarche

- Créer l'application web **gestion_grandeurs**

Étape par Étape

Création de gestion_grandeurs

Étape par Étape

Environnement virtuel

#créer un dossier pour le projet

```
$ cd framework_django\projects
```

```
$ mkdir gestion_grandeurs
```

```
$ cd gestion_grandeurs
```

#créer et activer l'environnement virtuel

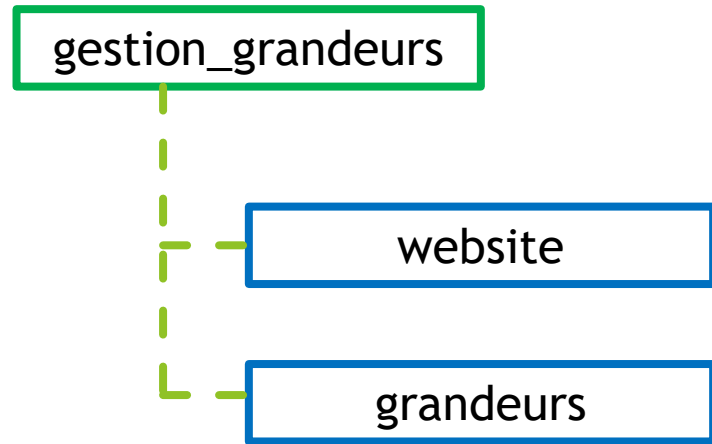
```
$ python -m venv .venv
```

```
$ .venv\Scripts\Activate.ps1
```

```
(.venv) $ python -m pip install django
```

Étape par Étape

Architecture



Créer le projet gestion_grandeurs

créer le projet gestion_grandeurs

(.venv) \$ django-admin startproject gestion_grandeurs .

lancer le projet dans VS code

(.venv) \$ code .

Étape par Étape

Création des applications

créer l'application website

(my_env) \$ python manage.py startapp website

créer l'application meetings

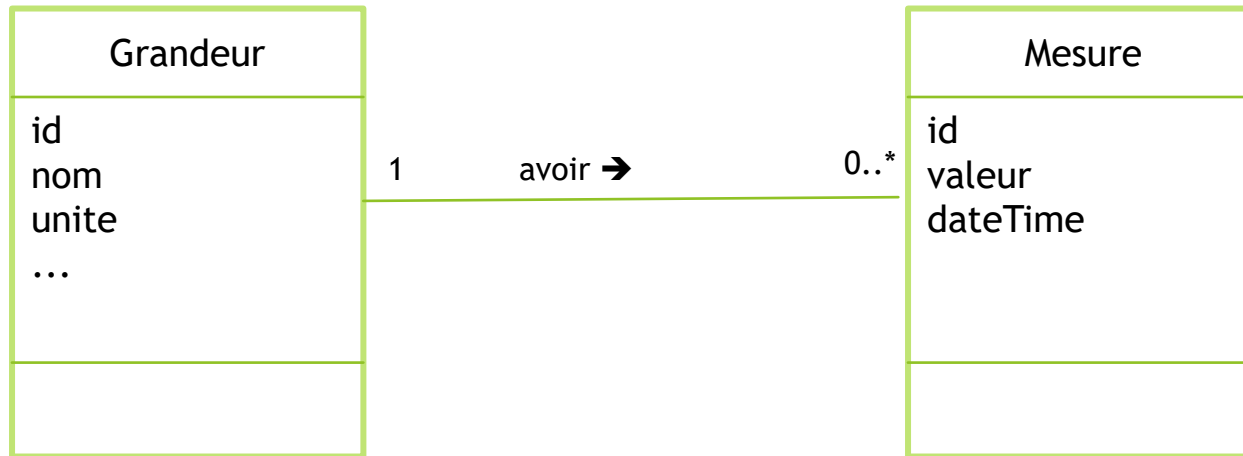
(my_env) \$ python manage.py startapp grandeurs

Étape par Étape

settings.py: installation des apps

```
INSTALLED_APPS = [  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
    'website.apps.WebsiteConfig',  
    'grandeurs.apps.GrandeursConfig'  
]
```

Modèle



Grandeur

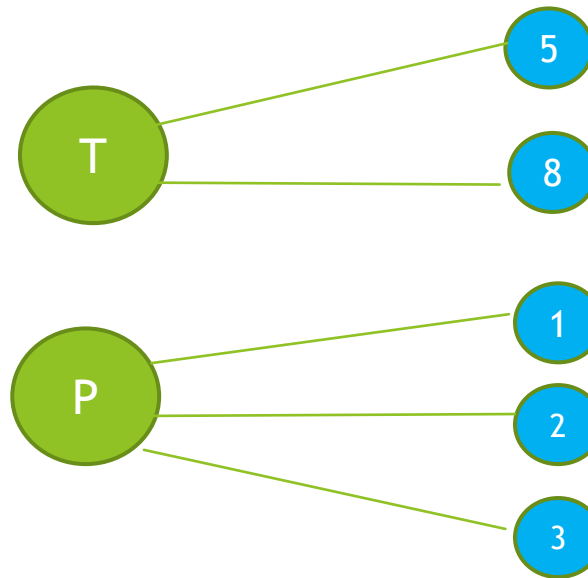
1	T	C

Mesure

		fk
5	t1	1



transparente



django Models: Fields

Éléments du contenu

- ▶ Model Fields
 - ▶ Type des Fields et options
 - ▶ Effet sur les formulaires et validation
- ▶ Relations:
 - ▶ ForeignKey
 - ▶ OneToOne
 - ▶ ManyToMany
 - ▶ Créer des relations entre les objets

```
class Person(models.Model):  
    name = models.CharField(max_length=100)  
    age = models.IntegerField()
```

Model Fields

Class attributes mapped to DB columns

Doivent être instances des **classes Field**

Exemple: CharField, IntegerField,...

Model Fields

Les Classes Fields déterminent:

- types de colonnes dans la BD (Integer, varchar,...)
- type de widget dans le formulaire

Field Options

- Validation au niveau de la BD
- Formulaire et validation
- valeurs par défaut
- . . .

documentation

- <https://docs.djangoproject.com/en/4.2/ref/models/fields/>

Storing Numbers

BooleanField

IntegerField
(et variantes)

FloatField

DecimalField

Storing Text

CharField

HTML: input text

Required
max_length

TextField

Text plus large

HTML: TextArea

EmailField

URLField

FilePathField

SlugField

Other Common Field Types

DateField

TimeField

DateTimeField

DurationField

FileField

ImageField

JSONField

BinaryField

Démo: TextFields

```
class Product(models.Model):  
    name=models.CharField(max_length=100)  
    stock_count=models.IntegerField(default=0)  
    price=models.DecimalField(max_digits=6,decimal_places=2)  
    description=models.TextField(default="")
```

Étape par Étape

migrations

rollback products migrations

(my_env) \$ python manage.py migrate products zero

créer migrations

(my_env) \$ python manage.py makemigrations

appliquer les migrations

(my_env) \$ python manage.py migrate

products.admin.py

```
from django.contrib import admin
from .models import Product

# Register your models here.

admin.site.register(Product)
```

Étape par Étape

Admin interface

Add product

Name:

Stock count:

Price:

Description:

SAVE

Save and add another

Save and continue editing

Étape par Étape

Options: Null et Blank

- ▶ Ajouter un nouveau produit sans description
- ▶ Interpréter le résultat
- ▶ Par défaut, tous les champs n'acceptent pas la valeur **null**
- ▶ Option **null**: validation **coté BD**
- ▶ Option **blank**: autorisation de valeur vide **coté formulaire**

Options: blank et null

```
class Product(models.Model):  
    name=models.CharField(max_length=100)  
    stock_count=models.IntegerField(default=0)  
    price=models.DecimalField(max_digits=6,decimal_places=2)  
    description=models.TextField(default="",blank=True)
```

Model Field Options 1

Make Field Nullable (default is non-NULL)

```
models.IntegerField(null = True)
```

Allow empty values in forms (Not db-related!)

```
models.CharField(blank = True)
```

Default value

```
models.CharField(default = 'Example')
```

Model Field Options 2

Add unique constraint

```
models.CharField(unique = True)
```

Add an index

```
models.IntegerField(db_index = True)
```

Set column name

```
models.BooleanField(db_column = "my_column_name")
```

Type-specific options

```
models.DateTimeField(auto_now = True)
```

Model Field Options 3

Set field label

```
iban = models.CharField(verbose_name = "Bank Account", ...)
```

Additional help text

```
name = models.CharField(help_text = "Enter your full name")
```

grandeurs.models.py

```
class Grandeur(models.Model):
    nom=models.CharField(max_length=60)
    unite=models.CharField(max_length=30)
    valeurMin=models.FloatField()
    valeurMax=models.FloatField()

    def __str__(self):
        return f"Grandeur: {self.nom} Unité: {self.unite} valeurs entre  
{self.valeurMin} et {self.valeurMax}"
```


grandeurs.models.py

```
class Mesure(models.Model):  
    valeur=models.FloatField()  
    datePrise=models.DateTimeField()  
    grandeur=models.ForeignKey('Grandeur', on_delete=models.CASCADE)  
  
    def __str__(self):  
        return f"Mesure: {self.valeur} at {self.datePrise}"
```

migrations

| # créer migrations

| (my_env) \$ python manage.py makemigrations

| # appliquer les migrations

| (my_env) \$ python manage.py migrate