OUTILS DE DÉVELOPPEMENT – FRAMEWORK PYTHON

Les structures de contrôles, les collections et les classes python dans les Framework python, l'accès aux données, la gestion des vues, la création des Template et des formulaires.

1) Elément du langage python
2) L'accès aux données et les ORM
3) Gestion des vues et les Template
4) Gestion des formulaires

PR. ABOUABID HAMZA

AGENDA

- Introduction: Rappel sur les notions de base
- **Partie 1: Le Framework Django**
- Partie 2: Les URLs et les vues
- Partie 3: L'accès aux données et les ORM
- Partie 4: Gestion des vues
- Partie 5: Template
- Partie 6: Les formulaires

PARTIE 1: LE FRAMEWORK DJANGO

SÉANCE 3

1) **Présentation du Framework Django**
2) Composants du Framework Django
3) Préparation et environnement du serveur
4) Application Web avec le Framework Django

1) PRÉSENTATION DU FRAMEWORK DJANGO

- **Django** est Framework libre et open source pour le création et le développement des applications Web avec Python.

1.1. Alternatives:

En Python les alternatives sont notamment :

- **Flask** (microframework avec Jinja2 pour les templates et SQLAlchemy pour ORM)
- **Pyramid**
- **Tornado** (orienté programmation asynchrone et Web Sockets)
- **FastAPI** pour construire rapidement une API en Python
- Falcon
- ..

1.2. Application créer avec Django :

✓ Youtube, Spotify, Instagram, DropBox, ...

1) PRÉSENTATION DU FRAMEWORK DJANGO

1.3. Caractéristiques et fonctionnalités:

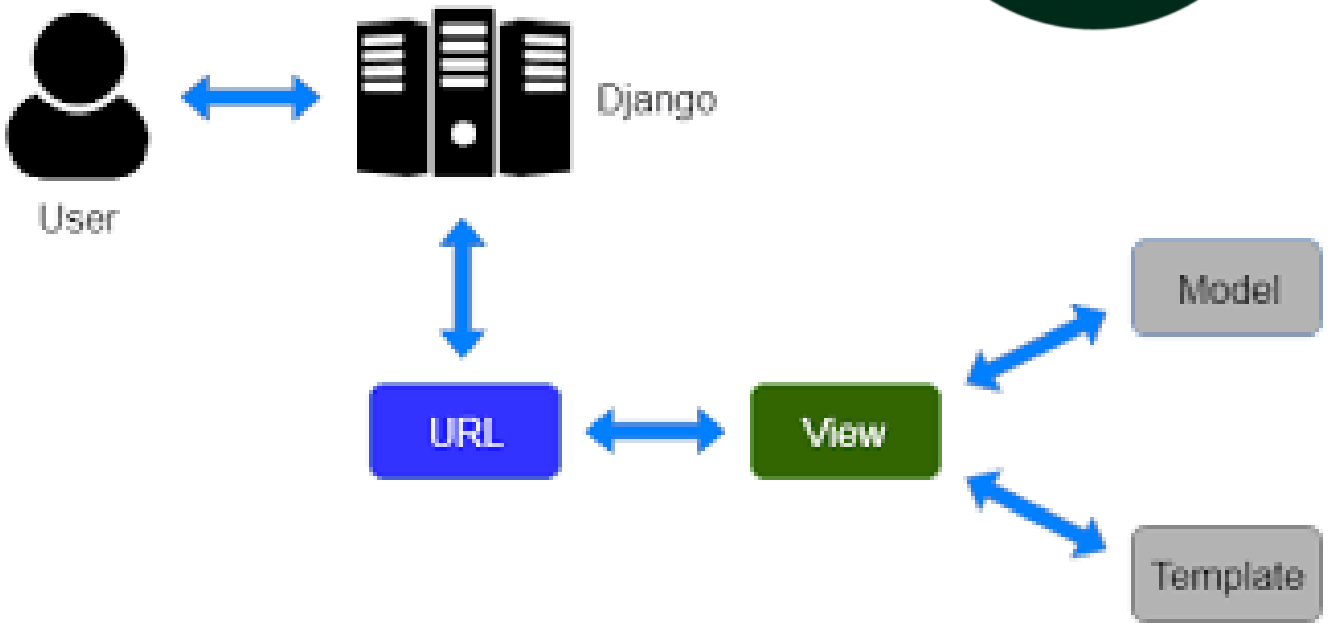
- Interface Admin
- Framework Web de référence
- ORM (Object Relational Mapper)
- Les Templates
- Migrations (intégrées tardivement, l'un des meilleurs système de migrations existant)
- Formulaires, Vues génériques
- Authentification
- Gestion complète des exceptions
- Bonne documentation

1) PRÉSENTATION DU FRAMEWORK DJANGO

1.4. Framework Web généraliste :

Django offre :

- **MVT : Modèle Vue Template**
- **Système de Templates**
- **ORM:** Object Relational Mapper (comme SQLAlchemy ou Doctrine)
- Serveur Web intégré
- **Interface d'Admin** complète, souple et extensible



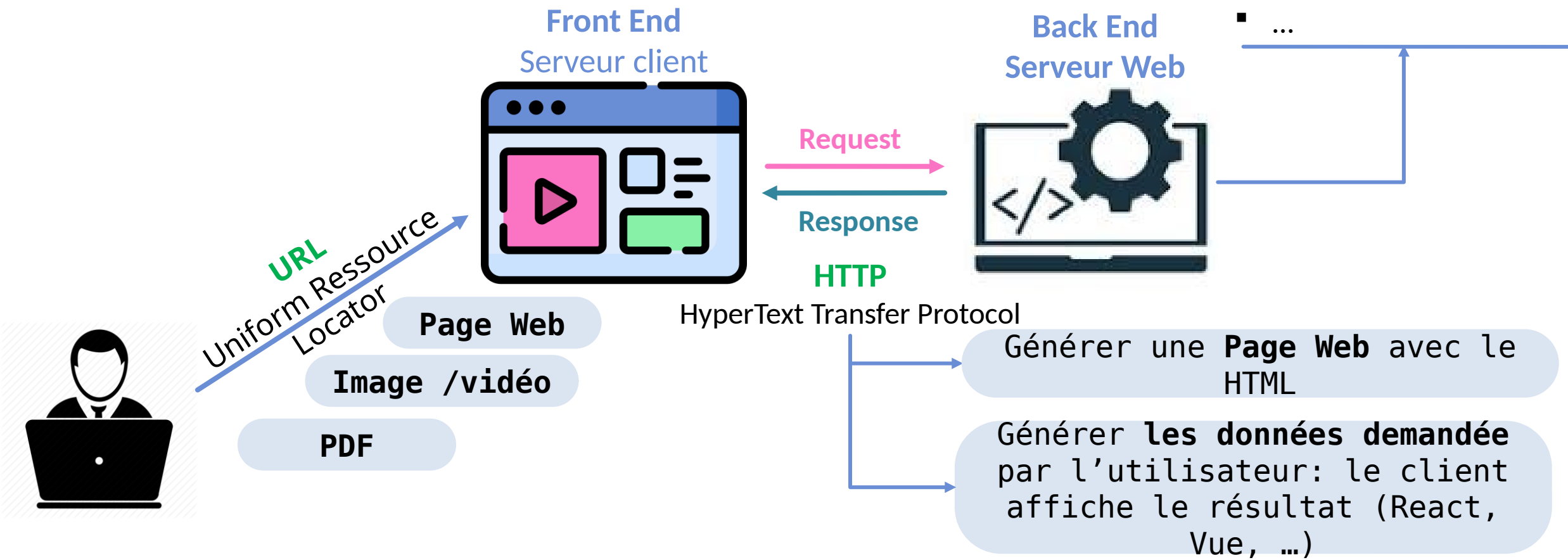
PARTIE 1: LE FRAMEWORK DJANGO

SÉANCE 4

1) Présentation du Framework Django
2) **Composants du Framework Django**
3) Préparation et environnement du serveur
4) Application Web avec le Framework Django

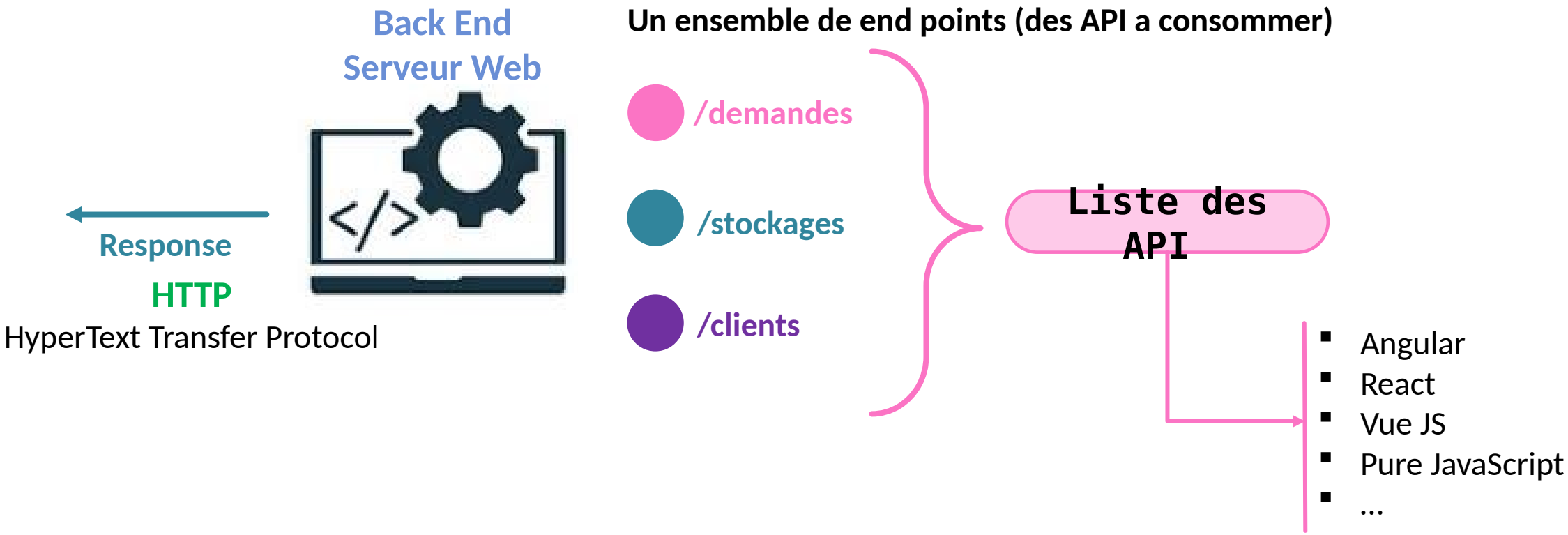
2) COMPOSANTS DU FRAMEWORK DJANGO

- **Structure d'une application Web standard :**



2) COMPOSANTS DU FRAMEWORK DJANGO

- **Structure d'une application Web standard :**



PARTIE 1: LE FRAMEWORK DJANGO

SÉANCE 3

1) Présentation du Framework Django
2) Composants du Framework Django
3) **Préparation et environnement du serveur**
4) Versionnage GIT
5) Application Web avec le Framework Django

3) PRÉPARATION ET ENVIRONNEMENT DU SERVEUR

✓ L'environnement de développement Python:

1 **Interpréteur Python**

https://www.python.org/downloads/

2 **Editeur (IDE):** VS Code, PyCharm, Vim, Brackets, Sublime, ...

Microsoft Visual Code: https://visualstudio.microsoft.com/

✓ Les dépendances et les système de gestion de version :

3 **L'outil pip:** Système de gestion des paquets

4 **Système décentralisés:** GitHub

https://github.com/

PARTIE 1: LE FRAMEWORK DJANGO

SÉANCE 3

1) Présentation du Framework Django
2) Composants du Framework Django
3) Préparation et environnement du serveur
4) **Application Web avec le Framework Django**

4) APPLICATION WEB AVEC LE FRAMEWORK DJANGO

4.1. Architecture d'une application Web Django: (1/2)

1 Demande de l'utilisateur

l'utilisateur fait une demande en cliquant sur l'URL de son **navigateur**

2 Résolution d'URL

L'URL est résolue et correspond à une VUE.

URL

3 La logique Vue

logique personnalisée et connexion aux modèles et Template.

VIEW

4 Model

Données obtenues à partir d'une base de données

Model

ORM

5 Template

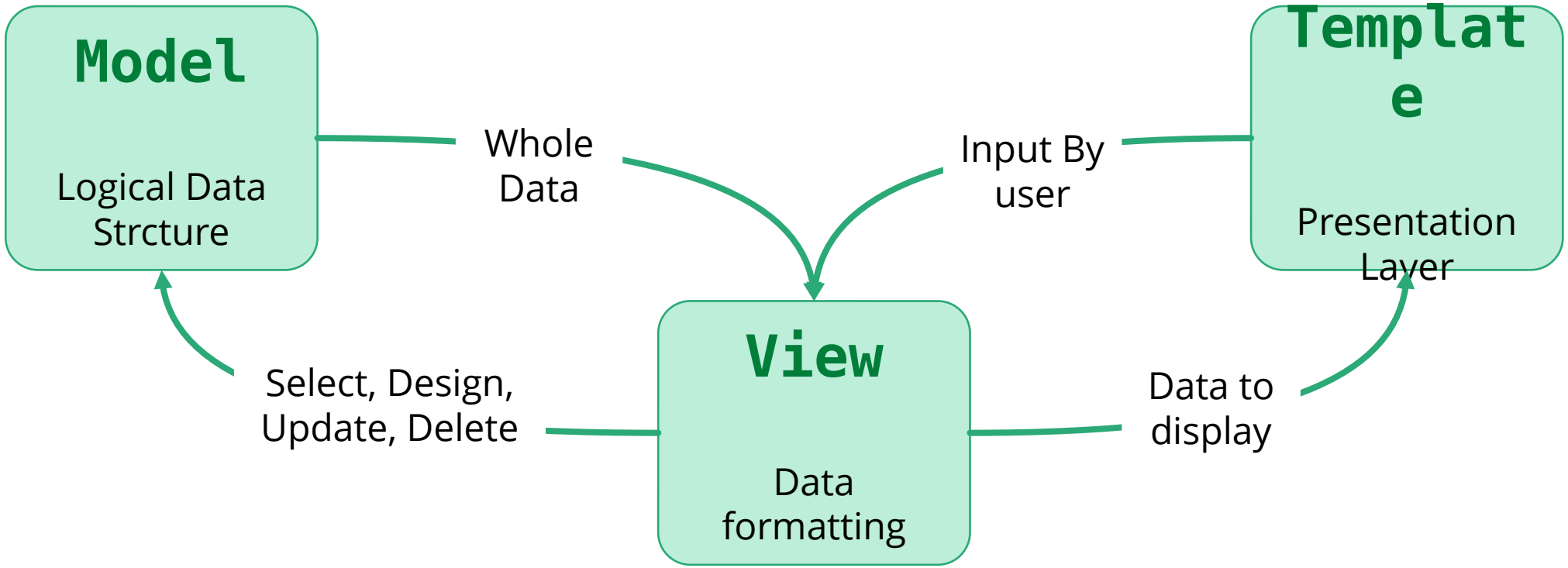
Données de contexte passées de la vue au html

Template

4) APPLICATION WEB AVEC LE FRAMEWORK DJANGO

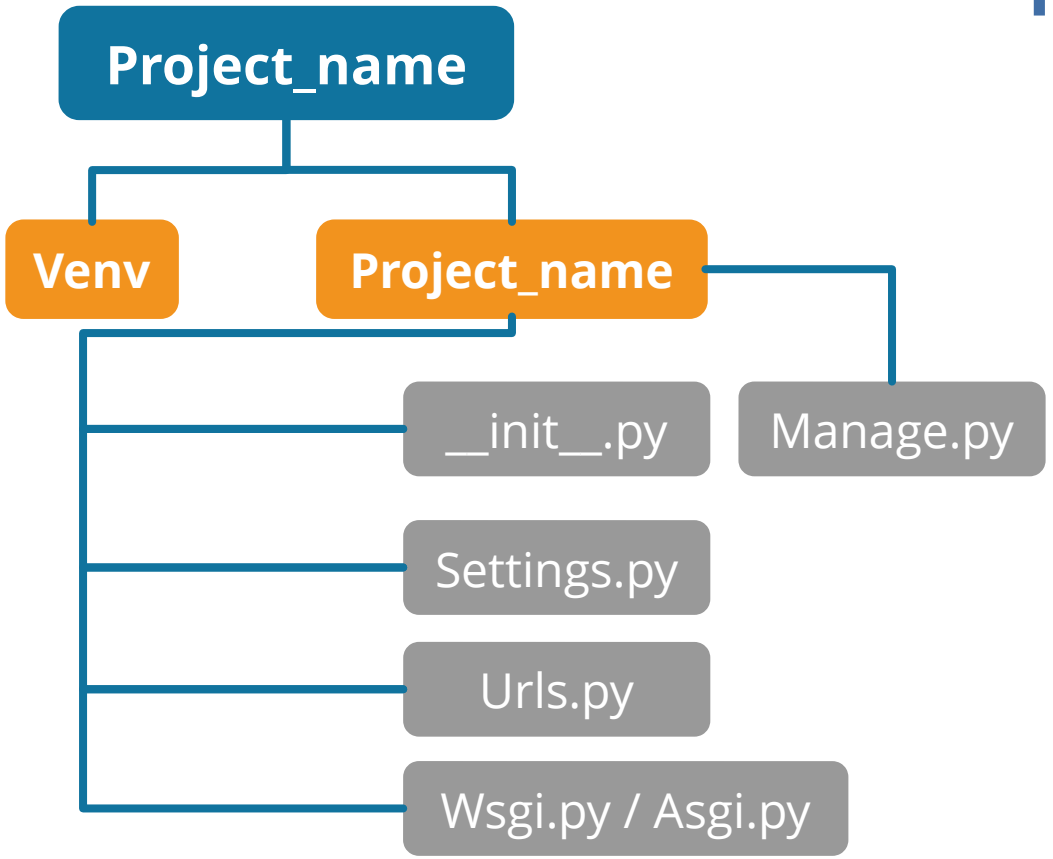
4.1. Architecture d'une application Web Django: (2/2)

✓ **Simplification de l'architecture :**



4) APPLICATION WEB AVEC LE FRAMEWORK DJANGO

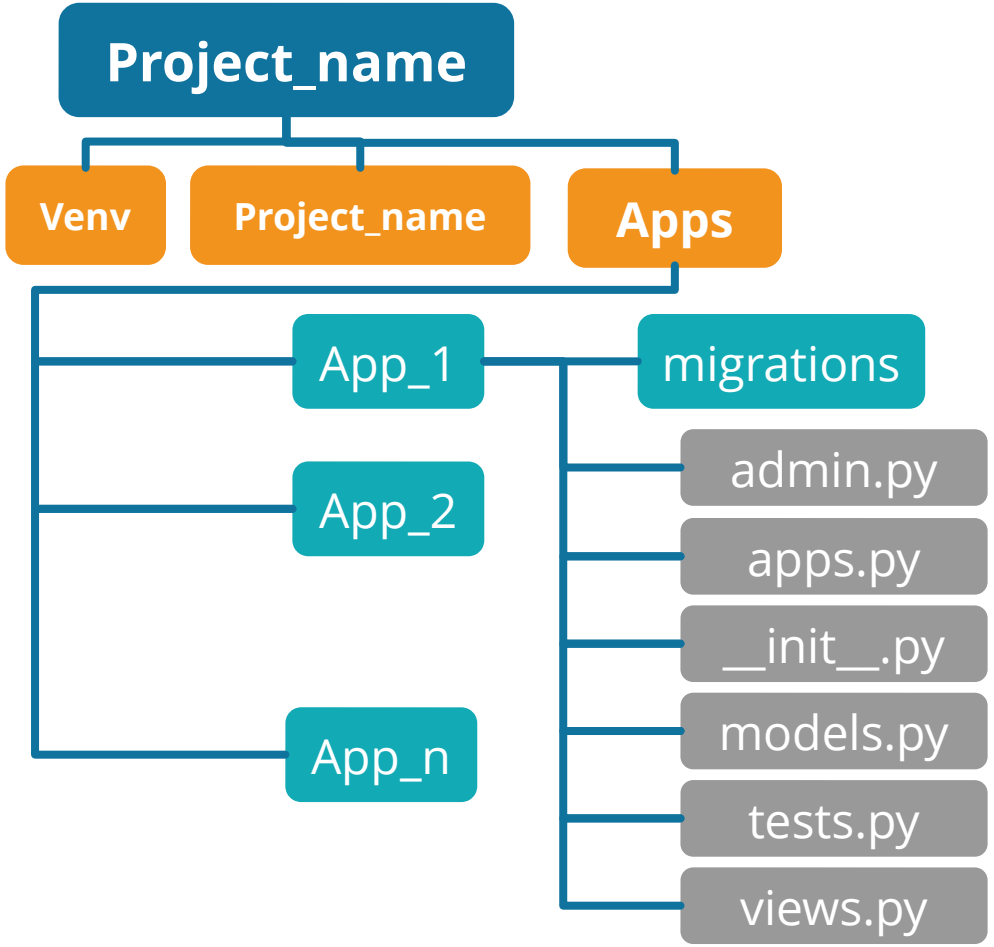
4.3. Structure fichier d'un projet Django : (1/3)



- **`__init__.py`** définit que ce dossier un package complet:
  - Vide par défauts
  - L'implication de ce fichier en fait un projet python
- **`Settings.py`** détermine les propriétés de l'application:
  - Utilisé pour ajouter toutes les applications et applications middleware au projet.
- **`Urls.py`** définit les URLs utiliser par l'application:
  - Il contient tous les points de terminaison que nous devrions avoir pour notre site Web.
- **`Asgi.py`** et **`wsgi.py`** utiliser pour le déploiement:
  - Asynchronous Server Gateway Interface.
  - Web Server Gateway Interface
- **`Manage.py`** l'équivalent de **`django-admin`**

4) APPLICATION WEB AVEC LE FRAMEWORK DJANGO

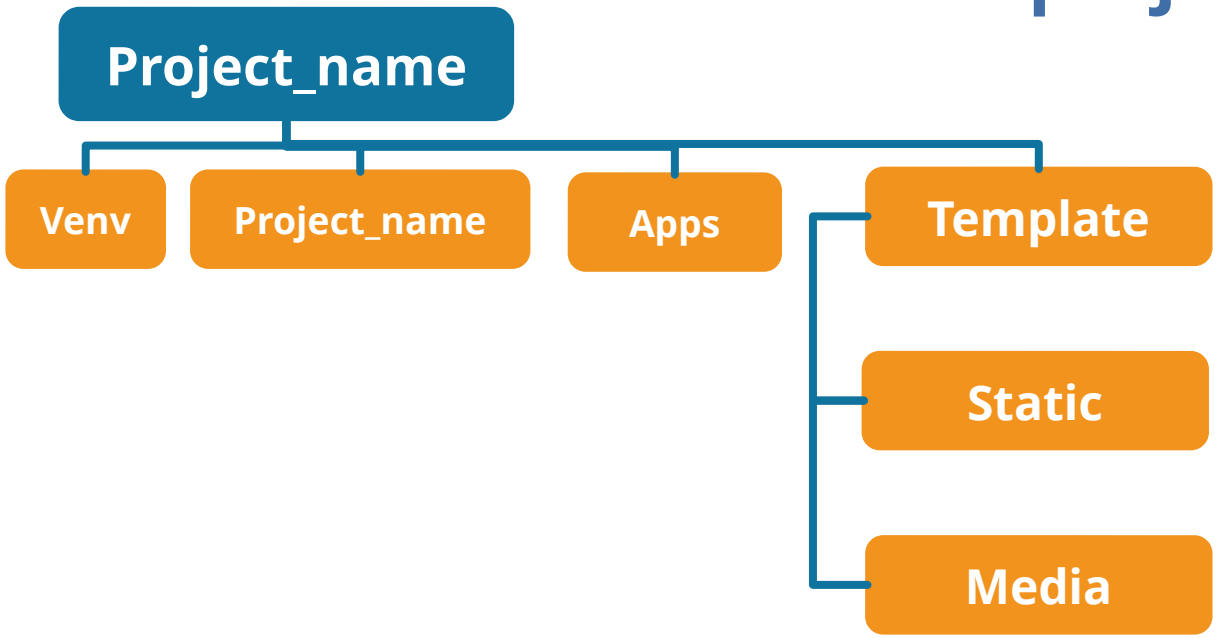
4.3. Structure fichier d'un projet Django : (2/3)



- **`Migrations`** les migrations utiliser pour générer les tables de la base de données.
- **`admin.py`** utilisé pour enregistrer les modèles Django dans l'administration Django.
- **`apps.py`** utilisé pour aider l'utilisateur à inclure la configuration de l'application pour son application.
- **`__init.py`** représente que le répertoire de l'application est un package complet
- **`models.py`** les modèles d'applications web sous forme de classes et méthodes.
- **`Test.py`** code de test pour l'applications Web
- **`views.py`** fournir une interface à travers laquelle un utilisateur interagit avec une application web Django.
  - Toutes les vues sont sous forme de classes et méthodes.

5) APPLICATION WEB AVEC LE FRAMEWORK DJANGO

5.3. Structure fichier d'un projet Django : (3/3)



- **`Template`** Données de contexte passées de la vue au html.
- **`Static`** un dossier contenant les fichiers statique utiliser par les Templates
- **`Media`** les médias afficher dans le contexte de le Template

5) APPLICATION WEB AVEC LE FRAMEWORK DJANGO

5.2. Création d'un projet et application Django : (1/8)

1) Installation et configuration du `pip` :

```
curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py
py get-pip.py
Pip help || pip -- version
```

2) La dépendance de l'environnement virtuel `pipenv` :

```
Pip install pipenv
```

3) Installation du Django à l'aide de `pipenv` :

```
Pipenv install Django
```

Si vous préférer travailler avec **venv**:

```
python -m venv myproject
myproject\Scripts\activate.bat
py -m pip install Django
```

4) Vérification de l'installation :

```
django-admin --version
```

5) APPLICATION WEB AVEC LE FRAMEWORK DJANGO

5.2. Création d'un projet et application Django : (2/8)

5) Activer l'environnement virtuel :

- `Pipenv shell`

6) Créer un projet Django :

- `Django-admin` `#verifier les commandes`
- `Django-admin startproject projectName`

7) Exécuter le projet sur un serveur :

- `Py manage.py runserver`

```
|_projectName
|__|__projectName/
|    |  __|-
♡   __init__.py
|    |  __|♡  asgi.py
|    |  __|-
♡   settings.py
|    |  __|♡  urls.py
|    |  __|♡  wsgi.py
|
|_♡ manage.py
```

5) APPLICATION WEB AVEC LE FRAMEWORK DJANGO

5.2. Création d'un projet et application Django : (3/8)

8) Créer une application du projet :

- `Py manage.py startapp applicationName`

- Ajouter le **nom de l'application** dans le fichier **`settings.py`** (paramètres du projet) dans le dossier **`projectName`**

`projectName/settings.py`

```
…
INSTALLED_APPS = [
    ….
    'applicationName'
]
…
```

```
projectName
    projectName/
    applicationName/
        migrations/
            __init__.py
        __init__.py
        admin.py
        apps.py
        models.py
        tests.py
        views.py
    manage.py
```

5) APPLICATION WEB AVEC LE FRAMEWORK DJANGO

5.2. Création d'un projet et application Django : (4/8)

9) Créer la première VIEW:

- Dans le fichier **applicationName/views.py** on ajoute les modèles à utiliser:
  - Obtenir du data de la base de données
  - Traitement de données
  - …

Exemple:

```
from django.http import HttpResponse

def hello(request):
    return HttpResponse('hello world')
```

```
|_projectName
|__ projectName/
|
|__ applicationName/
|     |__ migrations/
|     |     |__ __init__.py
|     |__ __init__.py
|     |__ admin.py
|     |__ apps.py
|     |__ models.py
|     |__ tests.py
|     |__ views.py
|
|
|__ manage.py
```

5) APPLICATION WEB AVEC LE FRAMEWORK DJANGO

5.2. Création d'un projet et application Django : (5/8)

10) Associer VIEW à une URL: (1/2)

- Créer le fichier **`applicationName/url.py`**
- Ajouter le chemin de la méthode aux ensembles d'URL de l'application

Exemple:

```
from django.urls import path
from . import views

urlpatterns=[
    path('hello/',views.hello)
]
```

```
projectName
 projectName/
 applicationName/
     migrations/
         __init__.py
     __init__.py
     admin.py
     apps.py
     models.py
     tests.py
     views.py
     urls.py
 manage.py
```

5) APPLICATION WEB AVEC LE FRAMEWORK DJANGO

5.2. Création d'un projet et application Django : (6/8)

10) Associer VIEW à une URL: (2/2)

- Créer le fichier **`projrctName/url.py`**
- Ajouter le chemin de la méthode aux ensembles d'URL de l'application.
- Ajouter les chemins des URL de l'app aux ensemble d'URL du projet.

Exemple:

```
from django.urls import path,include

urlpatterns = [
    …
     path('django_demo1_app1/’,
include('django_demo1_app1.urls'))
]
```

```
|_projectName
|__ projectName/
|
|__ applicationName/
|     |__ migrations/
|     |     |__ __init__.py
|     |__ __init__.py
|     |__ admin.py
|     |__ apps.py
|     |__ models.py
|     |__ tests.py
|     |__ views.py
|     |__ urls.py
|
|__ manage.py
```

5) APPLICATION WEB AVEC LE FRAMEWORK DJANGO

5.2. Création d'un projet et application Django : (7/8)

11) Création d'une Template :

- Créer un répertoires: **applicationName/template**
- Créer un fichier html: **template/index.html**

Exemple:

```
….
<body>
    <h1>premier projet Django</h1>
</body>
…
```

```
|_projectName
|__ projectName/
|
|__ applicationName/
|     |__ migrations/
|     |__ templates/
|     |     |__ index.html
|__ manage.py
```

5) APPLICATION WEB AVEC LE FRAMEWORK DJANGO

5.2. Création d'un projet et application Django : (8/8)

11) Création d'une Template :

- Créer un répertoires: **applicationName/template**
- Créer un fichier html: **template/index.html**
- Ouvrir la vue **applicationName/views.py** et modifier le retour de la méthode

Exemple:

```
from django.template import loader

def index(request):
  template =
loader.get_template('index.html')
  return HttpResponse(template.render())
```

Ou:

```
from django.shortcuts import render
def index(request):
  return render(request,'index.html')
```

```
|_projectName
|__ projectName/
|
|__ applicationName/
|     |__ migrations/
|     |__ templates/
|     |     |__ index.html
|__ manage.py
```

PARTIE 1: LE FRAMEWORK DJANGO

SÉANCE 3

1) Présentation du Framework Django
2) Composants du Framework Django
3) Préparation et environnement du serveur
4) Versionnage GIT
5) Application Web avec le Framework Django
   - ↳ **Démonstration 1:**
     - Se familiariser avec un Projet Django
     - Le Versionnage des dépendances

PARTIE 1: LE FRAMEWORK DJANGO

SÉANCE 4: DÉMONSTRATION

DÉMO 1: Première Application Web avec ASP.NET Core

- **Tache 1 :** Préparation de l'environnement de développement
- **Tache 2 :** Installation Django et environnement virtuel
- **Tache 3 :** Se familiariser avec les différents composants du Framework
- **Tache 4 :** Versionnage des dépendance « **requirements** »
- **Tache 5 :** Activer une **VENV** et Création d'un projet Django
- **Tache 6,7 :** l'IDE et Création de la première application du projet
- **Tache 8 :** La première vue et URL de l'application

PARTIE 1: LE FRAMEWORK DJANGO

SÉANCE 3: TRAVAUX PRATIQUE

Atelier 2: Premier projet et application Django