DOCUMENTATION TECHNIQUE

INTRODUCTION

L'orientation est une étape essentielle de la vie d'un l'étudiant. Après les récents débats et manifestations concernant les réformes de l'accession aux études supérieures, nous nous sommes demandé comment il serait possible d'apporter une aide aux futurs étudiants concernés. Ayant nous-même vécu l'épreuve de l'orientation, nous savons que la meilleure façon de récolter des informations est d'interagir avec des étudiants ou des professeurs du supérieur. Malheureusement, ces rencontres sont rares et se limitent souvent aux seules portes ouvertes des écoles.

Afin de pallier à ce problème, nous avons décidé de créer un outil permettant de mettre en relation des élèves en recherche de formation avec les étudiants d'écoles à travers la France. Cela a conduit à la réalisation de Brorientation, un site Web où chacun peut trouver, auprès d'une communauté d'étudiants, les informations qui lui sont nécessaires.

CONTENU TECHNIQUE

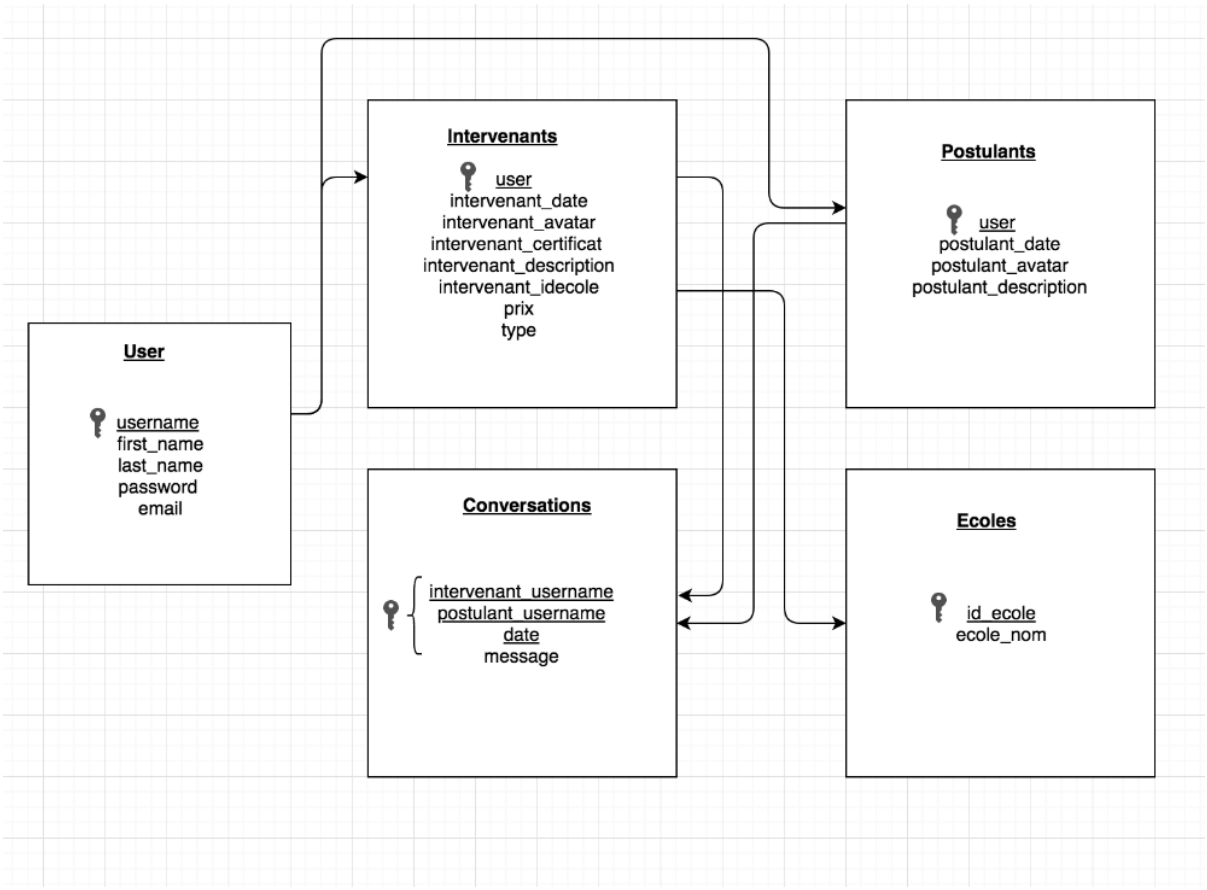
Le site se compose en 3 grandes parties : la recherche d'école, la partie connexion/inscription et l'espace personnel.

Quand il arrive sur la page d'accueil, l'utilisateur peut effectuer une recherche. Pour cela, il doit taper le nom de l'école dans la barre de recherche puis cliquer sur rechercher. Il accède alors à une page de résultats de recherche, qui est tout simplement une liste d'intervenants disponibles pour l'école en question. Seulement, s'il est en mode non connecté, il ne pourra pas contacter les intervenants. Il faut pour cela être connecté, ce qui nécessite d'être inscrit.

Dès la page d'accueil, mais également à partir de n'importe quelle page grâce à la barre de navigation, il est possible pour l'utilisateur de se connecter (s'il possède déjà un compte) ou de s'inscrire le cas échéant. Grâce à la barre de navigation, on peut également cliquer sur « how to » pour lire cette documentation technique, ou sur « à propos » pour nous contacter (on peut aussi descendre en bas de la page en scrollant).

Une fois connecté, la barre de navigation change et il est alors possible : de faire une recherche d'école (en cliquant sur « Brorientation »), d'accéder à son espace personnel (pour le modifier, rajouter une photo de profil ou une description…) et de se déconnecter.

1. Base de données



La base de données comporte 5 tables : User, Intervenants, Postulants, Conversations, Écoles. La table User est particulière : nous avons choisi de la pré-implémenter avec Django. Ainsi, lors de la création d'Intervenants ou de Postulants, un User est aussi créé : la table User est étendue par Intervenants et Postulants. L'intérêt d'avoir choisi cette implémentation est de pouvoir utiliser toutes les fonctionnalités fournies par Django pour permettre une connexion, une authentification et une gestion de mots de passe. La mise en place de ces fonctionnalités est ainsi plus sûre et plus simple.

Ce que stocke chaque table est précisé ci-dessous :

User	Intervenant/Postulant, password stocké haché
Intervenants	Données personnelles
Postulants	Données personnelles
Ecoles	Nom
Conversations	Message envoyé pour un intervenant, un postulant et une date

2. Fonctionnalités

A. PAGE D'ACCUEIL

La page d'accueil est le cœur de notre site. C'est sur cette page que nous retrouvons les principales fonctionnalités (comme la possibilité d'accéder aux interfaces de connexion et inscription). Nous avons décidé de conserver un design épuré afin que notre site soit simple d'utilisation et que sa prise en main soit la plus simple possible. Sur cette page, hormis la barre de navigation, nous ne retrouvons que la barre de recherche permettant aux étudiants de trouver une école.

Lors de la connexion, un message de bienvenue s'affiche avec un message défini en fonction de la date de la dernière connexion de l'utilisateur. La simplicité de Django s'illustre encore à travers cet exemple car ce genre d'information sur l'utilisateur peut être récupéré grâce à une méthode pré-implémentée. L'aspect graphique du message est ensuite géré par un Script JQuery.

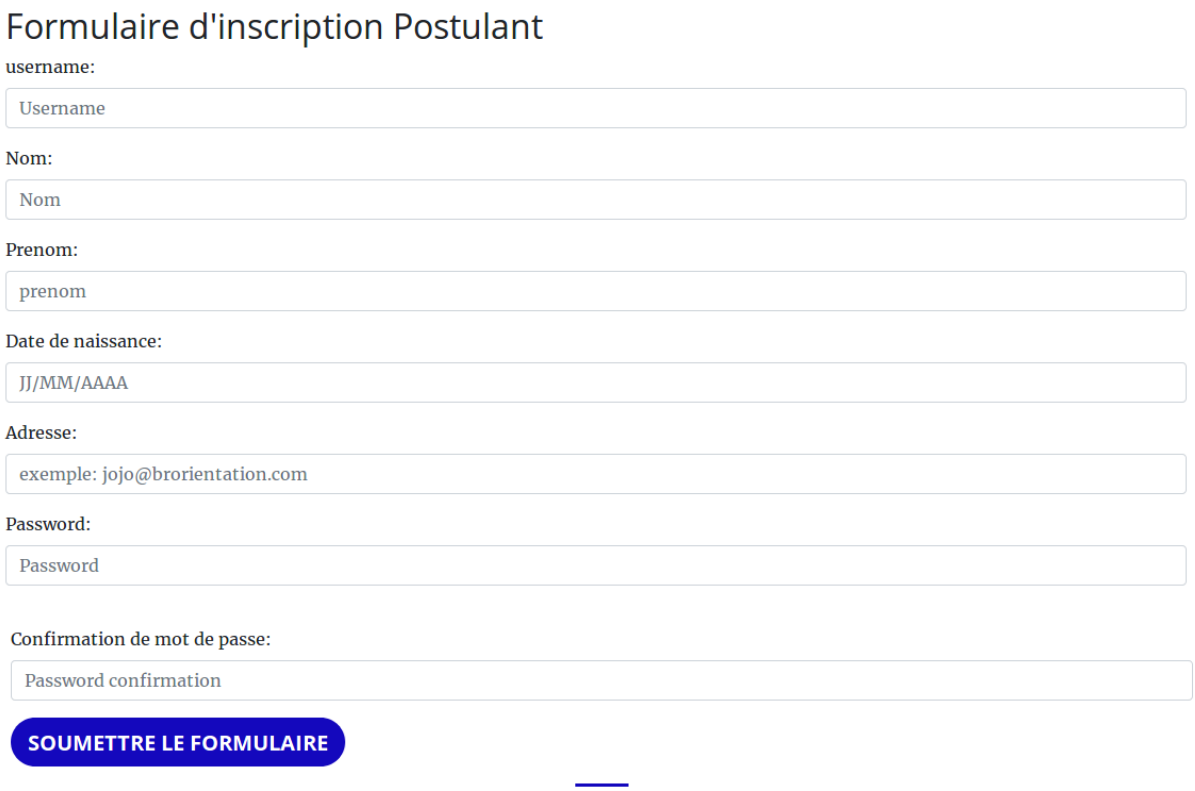
Page d'accueil du site Brorientation

Nous pouvons également remarquer l'onglet « à propos » qui va délocaliser la page vers le bas afin de montrer les coordonnées à appeler en cas de problème ou demande d'information. La mention « visiteur » en haut à droite sera remplacée par le pseudo de l'utilisateur une fois qu'il sera connecté.

B. CREATION D'UN PROFIL

Tout d'abord, notre site met en place la création d'un profil. Nos utilisateurs sont répartis en deux catégories, postulant ou intervenant, les deux ne présentant pas les mêmes caractéristiques. Les deux profils peuvent être créés sous l'onglet Inscription : nous avons mis en place deux formulaires différents qui vont couvrir la plupart des champs des deux possibilités. Nous avons également mis en place une vérification des données entrées par l'utilisateur, afin qu'il ne rentre pas n'importe quoi

(pseudonyme unique, obligation de mot de passe supérieur à 6 caractères...). Les informations sont ensuite stockées dans la base de données. Les deux instances Intervenant et Postulant héritent de la classe User de Django. Nous avons également mis en place un peu de dynamisme lors du choix de la catégorie Postulant ou Intervenant, les deux boutons vont s'animer au passage de la souris.



Formulaire d'inscription pour un Postulant

C. CONNEXION

A l'aide du kit fourni par Django, notre site web prend en charge la connexion des utilisateurs et sait déterminer s'il s'agit d'un intervenant ou d'un postulant. Le formulaire vérifie ainsi le pseudo et le mot de passe dans la base de données. Sur cette page, nous avons voulu que l'utilisateur ne se sente pas perdu : ainsi, s'il a cliqué sur « Connexion » par mégarde alors qu'il n'a pas de compte, il peut facilement cliquer sur le lien redirigeant vers la page d'inscription.

Page de connexion

D. ESPACE UTILISATEUR

Nos deux différents types d'utilisateurs peuvent donc consulter leur profil sous l'onglet « Espace personnel » de la barre de navigation. Les champs correspondants s'affichent et une photo de profil temporaire vient remplir leur espace avatar. Sous chacun des deux profils se trouve un bouton « Modifier », qui, comme son nom l'indique, va permettre à l'utilisateur de modifier certaines de ses données, notamment sa description et sa photo de profil. Les données sont ensuite sauvegardées dans la base de données. Dans le profil Intervenant se trouve également le champ « Conversation » que nous développerons plus tard.

Espace personnel d'un Postulant

E. BARRE DE RECHERCHE

La barre de recherche accède à la classe Ecole et permet de chercher une école existante. Si l'école recherchée n'existe pas, elle renvoie un message d'erreur. Après la recherche, tous les intervenants disponibles pour cette école seront affichés avec quelques-unes de leurs caractéristiques personnelles dans un grand menu déroulant. Il sera ensuite possible de leur envoyer un message (possible uniquement en mode connecté – si l'utilisateur est encore en mode anonyme, il verra un message en rouge lui indiquant de se connecter pour contacter l'intervenant).

Résultat de la recherche « INSA »

F. CONVERSATIONS

Lorsque nous cliquons sur "Contacter" sur la page des résultats de recherche, une nouvelle page s'affiche, où il est désormais possible d'envoyer un message à la personne choisie. La personne concernée va ensuite recevoir le message et le résultat sera affiché dans son espace personnel. Elle va ensuite recevoir un mail sur l'adresse qu'elle aura renseignée au moment de son inscription contenant toutes les informations dont elle a besoin afin de pouvoir ensuite recontacter le postulant.

Ecriture du message à envoyer

Affichage des conversations dans l'userSpace Intervenant

3. Structure du code

L'architecture et l'organisation des fichiers sont primordiales dans la réalisation d'un site avec Django. Il faut donc prévoir quelles applications seront à réaliser, et découper le projet en étapes-clés pour avancer. Néanmoins, cette hiérarchisation des fichiers permet de mieux se répartir le travail et offre aux développeurs un cadre de travail strict et normé. Il est ainsi beaucoup plus facile de s'y retrouver dans le code.

A. RACINE

L'un des principaux avantages de Django est la réutilisation du code. Afin de ne pas avoir de redondance dans le code, il est possible de définir des templates de référence à la racine pour pouvoir les utiliser plus tard. Nous avons d'ailleurs utilisé un fichier index.html qui nous a servi de squelette tout au long du projet. Cela nous a permis de tout de suite avoir une certaine cohérence dans le projet, nous avons pu tous débuter notre partie du travail sans se soucier outre-mesure de l'aspect général qui avait déjà été défini.

Django génère lui-même un certain nombre de codes à la création du projet, comme des fichiers de settings qui permettent de centraliser les différents paramètres du projet (comme les options de debug, les chemins pour les fichiers d'images, etc…).

Le fichier urls.py centralise toutes les url et permet de rediriger vers les mêmes fichiers urls.py dans les applications correspondantes.

```
urlpatterns = [
    path('admin/', admin.site.urls),
    path('accueil/',include('accueil.urls')),
    path('inscription/',include('inscription.urls')),
    path('ecole/',include('ecole.urls')),
    path('userSpace/',include('userSpace.urls')),
    path('', include('django.contrib.auth.urls')),
]
```

A la racine nous trouvons aussi le fichier de la base de données qui est généré par Django en fonction des différents fichiers models.py de chacune de nos applications. Celle-ci est créée et modifiée en fonction de nos modifications à l'appel des commandes "makemigrations" et "migrate".

B. APPLICATIONS

Afin de permettre à chacun de travailler de façon indépendante, sans entrer en conflit avec les autres membres du groupe, et surtout pour respecter l'utilisation de Django, nous avons découpé notre projet en plusieurs applications. Celles-ci sont au nombre de quatre :

Accueil	Page principale : recherche d'écoles, connexion/inscription.
Inscription	Contient les deux formulaires pour s'inscrire en tant qu'Intervenant ou Postulant.
UserSpace	Accès à l'espace utilisateur, modification des données personnelles
Ecole	Permet d'afficher les écoles de la base de données et les intervenants correspondants.

Dès le début du projet, nous avons pu définir cette architecture qui n'a pas beaucoup évolué au cours de celui-ci. En effet, avec le cahier des charges que nous avions réalisé, nous avions une idée suffisamment précise de ce que nous prévoyions de réaliser pour ne pas avoir à redéfinir ces applications. Dans chacune des applications se trouvent plusieurs fichiers remarquables :

- **models.py** permet de définir les classes de la base de données qui seront utilisées dans l'application.

Dans notre cas, nous avons tout centralisé dans le models.py d'Inscription que nous avons importé dans tous les autres afin d'accéder rapidement à notre modèle de base de données.

```
class Postulants(models.Model):
    user = models.OneToOneField(User, on_delete = models.CASCADE)
    postulant_date = models.CharField(max_length=10,default='')
    postulant_avatar = models.ImageField(upload_to='photos/',default='photos/profil/profile.jpg')
    postulant_description = models.TextField(default='')
```

Comme nous pouvons le remarquer, nous définissons aussi le type(CharField, ImageField…) et les dépendances entre les classes (OneToOneField).

- **forms.py** est un fichier présent dans les applications qui nécessitent l'utilisation d'un formulaire.

Il va permettre de définir toutes les entrées nécessitant la méthode POST et les informations demandées. Par exemple, pour les champs « date de naissance » ou « adresse mail », il va s'assurer qu'il s'agit bien d'une entrée sous le bon format. D'autre part, il permet à l'aide de widget de définir l'id du champ qui sera utilisé dans le code html relié ou un texte qui sera affiché et qui permettra à l'utilisateur de savoir quelles informations il doit renseigner.

```
class postulantForm(forms.Form):

    username = forms.CharField(
        required = True,
        widget = forms.TextInput(attrs={
            'class' : 'form-control' , 'placeholder' : 'Username' , 'id':'username'
        })
    )
```

- **views.py** permet de définir la fonction à effectuer en correspondance avec l'url en cours.

Les fonctions ainsi codées peuvent répondre à GET, par exemple dans le cas de l'userSpace où les données vont simplement être affichées, mais aussi à la méthode POST pour le formulaire où la fonction va récupérer les entrées postées dans les champs définis par forms.py pour pouvoir mettre à jour la base de données. C'est ici que seront gérés les templates à afficher à travers la fonction render de Django.

```python
def afficher_profil(request):
    current_user = request.user

    try:
        q = Intervenants.objects.get(user=current_user)
        conversation = Conversations.objects.filter(intervenant_username=current_user)
        return render(request, 'profil.html',{'data': conversation})
    except:
        q = Postulants.objects.get(user=current_user)
        return render(request, 'profilPost.html')
```

Cette fonction permet de d'afficher le profil d'un utilisateur en fonction de son type, soit intervenant, soit postulant. Elle va donc charger le html correspondant.

- **urls.py** permet de définir toutes les url qui complètent le fichier urls.py disponible à la racine.

Le fichier met en relation les path url et les fonctions du fichier views.py. Les fonctions seront donc appelées lorsque l'url correspondante sera demandée.

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

urlpatterns = [
    path('',views.home),
    path('formulaireIntervenant',views.get_intervenant_infos, name = 'get_intervenant_infos'),
    path('formulairePostulant',views.create_postulant, name = 'create_postulant')
]
```

C. TEMPLATES

Nous avons placé notre template principal dans le dossier /templates/ sous le nom de index.html. Ce dernier est étendu sur tous les autres utilisés dans nos applications.

```
{% extends "index.html" %}
{% block title %}Choix du type d'inscription{%endblock%}
{% load static %}

{% block link %}
<link rel="stylesheet" href="{% static "inscription/css/inscriptionChoice.css" %}"/>
{% endblock %}
```

Début de chacun de nos templates applications, avec le css propre à chaque fois

A la racine se trouve aussi le template automatiquement reconnu par Django pour la page de connexion /templates/registration/login.html. A partir de cela, dans chaque application se trouve également un dossier templates où se trouveront les templates propres à chaque application.

D. STATIC

Les fichiers static permettent de stocker simplement les différents codes CSS, JavaScript ou encore les images présentes sur le site. Une fois le path défini dans le fichier Settings présent à la racine, il est très facile d'accéder à ces fichiers dans chacune de nos applications (avec le mot clé {% static path/to/folder %}). Encore une fois, l'organisation du code défini par Django nous a été très utile car chaque application suit la même structure et permet donc aux différents codeurs de l'équipe d'avoir des repères lorsqu'il s'agit d'évoluer dans une application qu'il n'a pas codée.

E. MEDIAS

La gestion et le stockage des photos de profil ou des certificats de scolarité des intervenants se gère grâce à la librairie Pillow que nous avons téléchargée. Ces fichiers ne sont pas directement stockés dans la base de données (il n'y figure qu'une référence permettant d'y accéder ensuite).

Lorsqu'un utilisateur soumet de nouveaux fichiers, ces derniers sont automatiquement importés dans le dossier médias (présent dans la racine). Les path sont ensuite automatiquement générés par Django et disponibles pour chaque profil (après initialisation dans le fichier settings présent à la racine). L'utilisation de ce dossier média n'est néanmoins possible que lors de la partie développement du site. En effet, stocker les fichiers dans un dossier statique expose notre site web à de fortes failles de sécurité. Pour passer le site en production (publication sur le Web), il faut stocker ces différentes images sur un serveur apache externe.

ANNEXE

Pour lancer le site :

A la racine du projet, ouvrir un terminal et taper en ligne de commande :
python3 manage.py makemigrations
python3 manage.py migrate
python3 manage.py runserver
Puis, dans le navigateur :
localhost:8000/accueil/accueilVisiteur