**MODELISATION DE GALSEN :**

J’ai créé un projet nomme BourStudios. Dans ce projet, j’ai généré une application nommée galsen(en minuscule). Delà, j’ai modélisé mon projet galsen par rapport à mon document d’où dresser la liste des tables et leur relation.

Sur ce lancé, j’ai du apporte des modifications et anticiper le déplacement de certains de colonne dans certains table. Mais les relations n’ont pas changé.

Pour Django, une table utilisateur est définie par défaut. Pour y ajouter des colonnes ou y apporter des modifications, j’ai eu recours à :

from django.contrib.auth.models import AbstractUser

Ce dernier change le nom de la table User en CustomUser. Mais cela n’a pas eu d’impact important vu que le mot User peut-être toujours comme « « CLASS » »…

class CustomUser(AbstractUser):

    ROLES = [

        ('admin', 'Admin'),

        ('etudiant', 'Etudiant'),

        ('ecole', 'Ecole'),

        ('entreprise', 'Entreprise'),

    ]

    profile\_image = models.ImageField(upload\_to='profile\_images/', null=True, blank=True)

    banner\_image = models.ImageField(upload\_to='banner\_images/', null=True, blank=True)

    GENRE = [

        ('homme', 'Homme'),

        ('femme', 'Femme'),

    ]

    SITUATION\_MATRIMONIALE = [

        ('celibataire', 'Célibataire'),

        ('marie', 'Marié(e)'),

        ('fiance', 'Fiancé(e)'),

        ('divorce', 'Divorcé(e)'),

    ]

    email = models.EmailField(unique=True)

    metier = models.CharField(max\_length=150, null=True, blank=True)

    pays = models.CharField(max\_length=50, null=True, blank=True)

    ville = models.CharField(max\_length=50, null=True, blank=True)

    quartier = models.CharField(max\_length=50, null=True, blank=True)

    langue = models.CharField(max\_length=255, null=True, blank=True)

    birthday = models.DateField(null=True, blank=True)

    indicatif\_pays = models.CharField(max\_length=15, null=True, blank=True)

    number\_phone = models.CharField(max\_length=15, null=True, blank=True)

    number\_whatsapp = models.CharField(max\_length=15, null=True, blank=True)

    number\_telegram = models.CharField(max\_length=15, null=True, blank=True)

    instagram\_link = models.URLField(null=True, blank=True)

    twitter\_link = models.URLField(null=True, blank=True)

    youtube\_link = models.URLField(null=True, blank=True)

    website\_link = models.URLField(null=True, blank=True)

    rôle = models.CharField(max\_length=20, choices=ROLES, default='etudiant')

    genre = models.CharField(max\_length=20, choices=GENRE, default='homme')

    situation\_matrimoniale = models.CharField(max\_length=20, choices=SITUATION\_MATRIMONIALE, default='celibataire')

La table user sera en relation avec la table à propos. Cette table sera considérer dans la vue pour les étudiants leur CV, les écoles a propos de leur Etablissement et pour les entreprises à propos de leur entreprise :

class APropos(models.Model):

    user = models.OneToOneField(CustomUser, on\_delete=models.CASCADE)

    title = models.CharField(max\_length=255)

    description = models.TextField()

Tout utilisateur peut abonner sur un autre utilisateur : Voici comment j’ai dressé la table Abonnement :

class Abonnement(models.Model):

    user\_a = models.ForeignKey(CustomUser, related\_name='user\_a\_subscriptions', on\_delete= models.CASCADE)

    user\_b = user\_b = models.ForeignKey(CustomUser, related\_name='user\_b\_subscriptions', on\_delete= models.CASCADE)

    date\_creation = models.DateTimeField(auto\_now\_add=True)

Maintenant, nous sommes dans Post. La table USER est en relation avec table post :

class Post(models.Model):

    user = models.ForeignKey(CustomUser, on\_delete=models.CASCADE)

    video = models.FileField(upload\_to='post\_videos/')

    contenu\_post = models.TextField()

    nombre\_like = models.IntegerField(default=0)

    nombre\_commentaire = models.IntegerField(default=0)

    tag\_post = models.CharField(max\_length=255)

    date\_creation\_post = models.DateTimeField(auto\_now\_add=True)

Chaque post aura une ou plusieurs images. Du coup, pour éviter des trous vides dans la table POST, j’ai une créer un autre Table MediasPOST que j’ai mis en relation avec la table post, d’où tous les images post sont stocker :

class MediasPost(models.Model):

    post = models.ForeignKey(Post, on\_delete=models.CASCADE)

    image = models.ImageField(upload\_to='post\_images/')

Chaque poste a une ou plusieurs commentaires qui sont faites par les utilisateurs :

class Commentaire(models.Model):

    user = models.ForeignKey(CustomUser, on\_delete=models.CASCADE)

    post = models.ForeignKey(Post, on\_delete=models.CASCADE)

    contenu\_commentaire = models.TextField()

    image = models.ImageField(upload\_to='comment\_images/')

    date\_creation = models.DateTimeField(auto\_now\_add=True)

Les utilisateurs peuvent LIKER un post ou la partager :

class Like(models.Model):

    user = models.ForeignKey(CustomUser, on\_delete=models.CASCADE)

    post = models.ForeignKey(Post, on\_delete=models.CASCADE)

    date\_creation = models.DateTimeField(auto\_now\_add=True)

class Share(models.Model):

    user = models.ForeignKey(CustomUser, on\_delete=models.CASCADE)

    post = models.ForeignKey(Post, on\_delete=models.CASCADE)

    date\_creation = models.DateTimeField(auto\_now\_add=True)

Les commentaires peuvent avoir des réponses :

class Reponse(models.Model):

    commentaire = models.ForeignKey(Commentaire, on\_delete=models.CASCADE)

    user = models.ForeignKey(CustomUser, on\_delete=models.CASCADE)

    contenu\_text = models.TextField()

    image = models.ImageField(upload\_to='response\_images/')

    date\_creation = models.DateTimeField(auto\_now\_add=True)

La table JOB, est une table d’où sont stockées les offres d’emplois lancer par les utilisateurs :

class Job(models.Model):

    user = models.ForeignKey(CustomUser, on\_delete=models.CASCADE)

    title = models.CharField(max\_length=255, null=True, blank=True)

    contenu\_job = models.TextField()

    date\_creation\_post = models.DateTimeField(auto\_now\_add=True)

Qui dis dit offre d’emplois devra tout de même avoir un moyen de postuler a ces offres. Du coup dans la table Postule j’ai eu d’adopter le système Abonnement. Ainsi je pourrai récupérer l’Id de l’utilisateur que je présenterai au proprio de l’offre d’emplois qui a son tour pour vérifier le profil de ce dernier. Tout ça, par de liens.

class Postule(models.Model):

    user = models.ForeignKey(CustomUser, on\_delete=models.CASCADE)

    job = models.ForeignKey(Job, on\_delete=models.CASCADE)

    date\_creation = models.DateTimeField(auto\_now\_add=True)

Pourquoi ne pas permettre de partager l’offre d’emplois :

class ShareJob(models.Model):

    user = models.ForeignKey(CustomUser, on\_delete=models.CASCADE)

    job = models.ForeignKey(Job, on\_delete=models.CASCADE)

    date\_creation = models.DateTimeField(auto\_now\_add=True)

Tout utilisateur aura une boutique, mais dans la vue, seule les entreprises ont le droit d’en avoir :

class Boutique(models.Model):

    user = models.ForeignKey(CustomUser, on\_delete=models.CASCADE)

    nom\_boutique = models.CharField(max\_length=255)

    devise\_boutique = models.CharField(max\_length=255)

    description = models.TextField()

    photo\_profil = models.ImageField(upload\_to='boutique\_profil/')

    banner\_image = models.ImageField(upload\_to='boutique\_banner/')

    date\_creation = models.DateField(auto\_now\_add=True)

Chaque boutique devra vendre. Donc boutique sera en relation avec la table Product :

class Product(models.Model):

    boutique = models.ForeignKey(Boutique, on\_delete=models.CASCADE)

    video = models.FileField(upload\_to='product\_videos/')

    nom\_produit = models.CharField(max\_length=255)

    description = models.TextField()

    prix = models.DecimalField(max\_digits=10, decimal\_places=2)

    quantite\_stock = models.IntegerField()

    fournisseur = models.CharField(max\_length=255)

    date\_creation = models.DateTimeField(auto\_now\_add=True)

Les produits auront une ou plusieurs Images. Donc j’ai créé une table MedaisProduct associé à la table Boutique :

class MediasProduct(models.Model):

    produit = models.ForeignKey(Product, on\_delete=models.CASCADE)

    image = models.ImageField(upload\_to='product\_images/')

Les utilisateurs peuvent s’abonner à un boutique :

class Client(models.Model):

    user = models.ForeignKey(CustomUser, on\_delete=models.CASCADE)

    boutique = models.ForeignKey(Job, on\_delete=models.CASCADE)

    date\_creation = models.DateTimeField(auto\_now\_add=True)

**ORGANISATION TEMPLATE DE GALSEN :**

Dans le dossier TEMPLATE de mon application galsen, j’y ai créé des sous dossiers :

* **Admin ;**
* **Auth ;**
* **Common : (**ici, pour éviter la répétions de design sur plusieurs pages, j’y place les designs dont plusieurs page vont hériter.**)**
* **Create ;**
* **Ecole ;**
* **Entreprise ;**
* **Etudiant ;**

1. **AUTH :**

AUTH ou Authentification, se trouve mes formulaires login et register de mon application.

1. **ETUDIANT :**

Dans Etudiant, nous aurons les pages : Post, Ecole, Entreprise, Job ou offre d’emplois, boutique d’où tous les produits postés par les boutiques seront affiché. Des pages que je n’ai pas encore créé le design : Notifications, Mes amis (boutiques, Ecole, Entreprise, Etudiant), Membre d’où on peut retrouver des utilisateurs de même rôle.

Dans Etudiant, j’ai créé un dossier Profiles. Dans ce dossier vous aurez : un dossier mon profil et dossier public profil. Les noms disent tout.

1. **ECOLE :**

Dans Ecole, nous aurons les pages : Post, Etudiant, Entreprise, boutique d’où tous les produits postés par les boutiques seront affiché. Des pages que je n’ai pas encore créé le design : Notifications, Mes amis (boutiques, Etudiant, Entreprise, Ecole), Membre d’où on peut retrouver des utilisateurs de même rôle.

Dans Ecole, j’ai créé un dossier Profiles. Dans ce dossier vous aurez : un dossier mon profil et dossier public profil. Les noms disent tout.

1. **ENTREPRISE :**

Dans Entreprise, nous aurons les pages : Post, Ecole, Etudiant, boutique d’où tous les produits postés par les boutiques seront affiché. Des pages que je n’ai pas encore créé le design : Notifications, Mes amis (boutiques, Ecole, Etudiant, Entreprise), Membre d’où on peut retrouver des utilisateurs de même rôle.

Dans Entreprise, j’ai créé un dossier Profiles. Dans ce dossier vous aurez : un dossier mon profil et dossier public profil. Les noms disent tout.

1. **CREATE :**

Ici tout formulaire (post, job, update profile, update photo de profil ou banner statut entreprise, école ou étudiant…) etc… sont stockés.

**TRAVAIL DEJA ACCOMPLIS DANS GALSEN :**

* Système d’authentification : Register, Login, Redirection des utilisateurs selon les rôles.
* Récupération des utilisateurs selon les rôles et les afficher dans des pages respectives.
* Poster, modifier ou supprimer des articles.
* Poster des offres emplois.
* Update photo de profil, photo de bannière, statu des utilisateurs.