**Projet Web - M2 AMI2B** **2023/2024**
**Rosanne Phébé, Hugo Pointier, Nathan Maligeay, Rémi Poulard**

# Documentation Technique

**Framework** : Django 5.0
Nous utilisons Django comme cadre de développement pour notre application web. Notre code est formaté selon les normes de style PEP 8 grâce à l'utilisation de l'outil black. De plus, nous avons configuré flake8 pour détecter les messages liés au formatage, aux variables non utilisées et au respect des conventions de style. Nous avons également intégré django-stubs pour bénéficier de l'autocomplétion et de la validation statique des modules spécifiques à Django.
L'environnement de développement Python est isolé dans un environnement virtuel. La liste complète des packages utilisés est disponible dans le fichier README.
**Langages utilisés :** Python, Htmldjango, CSS (Bootstrap), JavaScript.
Pour les détails relatifs à l'installation et lancement du website, se référer au README disponible sur le gitlab.

**Ajout d'une commande django launch pour démarrer la base de donné du site**

Notre site web est organisé au sein d'une seule application Django, nommée "GenomeTag". Cette application comprend les modèles de la base de données relatifs aux fonctionnalités du site, telles que les annotations (pouvant être liées à plusieurs positions) et le génome. De plus, elle gère toute la partie utilisateur, y compris les utilisateurs personnalisés étendus à partir du modèle de base de Django.

**Recherche :** Nous avons mis en place un formulaire de recherche permettant aux utilisateurs de choisir sur quoi effectuer la recherche. En fonction de l'entité sélectionnée, nous gérons dynamiquement l'ajout ou la suppression d'opérateurs logiques à l'aide de JavaScript.

**Visualisation :** Pour afficher les chromosomes et leurs annotations, nous avons intégré la librairie JavaScript IGV.

**Utilisateurs / Droits :** Nous étendons le modèle utilisateur de base de Django pour ajouter des fonctionnalités supplémentaires, permettant notamment aux utilisateurs de fournir plus d'informations lors de leur inscription. Ils peuvent se connecter à l'aide de leur nom d'utilisateur ou de leur adresse e-mail. Nous gérons les rôles des utilisateurs à l'aide du système de groupes de Django, que nous avons étendu via un décorateur post_migrate. Les groupes sont assignés aux utilisateurs via un décorateur on_save du modèle en fonction de leur rôle.

**Blast :** Nous utilisons la librairie BioPython pour interroger l'API BLAST du NCBI. Les résultats sont renvoyés au format XML et affichés en HTML.

**Bootstrap :** Nous avons utilisé Bootstrap pour la mise en forme de l'interface utilisateur de notre site web. La structure de base de chaque page est définie dans le fichier base.html,

qui contient notre barre de navigation. Nous utilisons également crispy-forms pour personnaliser les formulaires HTML à l'aide de Python.

En terme de développement annexe a django, nous disposons de scripts permettant de parser des fasta de génomes, des cds du génome et de peptide. Ce script est utilisé dans le script d'initialisation qui va charger la base de données avec les données locales, créer des utilisateurs et administrateur et ajouter comme annotations les CDS. Les personnes disposant de droits administrateur peuvent rajouter des génomes dans la base de données.

**Domaine Protéique:** Nous interrogeons la banque de données PFAM à l'aide du package python prody avant d'envoyer les résultats au template pour les visualiser à l'aide du plug-in pviz.

**Description de code standard django :**

Modèle:

Le modèle Annotation représente une annotation dans notre application.

- Champs:
    - accession: Un champ CharField qui stocke un identifiant unique pour l'annotation, avec une longueur maximale de 15 caractères. Il est défini comme clé primaire (primary_key=True).
    - author: Une clé étrangère (ForeignKey) vers le modèle CustomUser, indiquant l'auteur de l'annotation. En cas de suppression de l'utilisateur associé, la valeur de cet attribut est définie sur null.
    - status: Un champ CharField avec des choix prédéfinis pour le statut de l'annotation. Les choix disponibles sont "unreviewed" (non revu), "rejected" (rejeté) et "validated" (validé). Par défaut, le statut est défini sur "unreviewed".
    - position: Un champ ManyToManyField qui relie l'annotation à plusieurs positions. Chaque annotation peut être associée à plusieurs positions, et chaque position peut être liée à plusieurs annotations.
    - tags: Un champ ManyToManyField qui relie l'annotation à plusieurs tags. De la même manière que pour position, chaque annotation peut être associée à plusieurs tags, et chaque tag peut être lié à plusieurs annotations.
    - commentary: Un champ TextField qui stocke un commentaire ou une explication supplémentaire pour l'annotation. Par défaut, il est vide.
    - reviewer: Une clé étrangère (ForeignKey) vers le modèle CustomUser, représentant l'utilisateur qui a examiné ou revu l'annotation. En cas de suppression de l'utilisateur associé, la valeur de cet attribut est définie sur null.

Vue create_peptide :

La vue create_peptide est responsable de la création d'une nouvelle instance de Peptide. Avant de permettre la création, elle vérifie si l'utilisateur actuel possède la permission nécessaire ("GenomeTag.review"). Si l'utilisateur n'a pas cette permission, il est redirigé vers une autre vue (reverse("GenomeTag:userPermission")).

La vue prend en charge les méthodes GET et POST :

- **GET** :
    - Si la méthode est GET, cela signifie que l'utilisateur accède simplement à la page pour créer un nouveau peptide. Dans ce cas, un formulaire vide est envoyé à la page pour permettre à l'utilisateur de saisir les informations du peptide.
- **POST** :
    - Si la méthode est POST, cela signifie que l'utilisateur a soumis le formulaire pour créer un nouveau peptide. Dans ce cas, la vue valide le formulaire. Si le formulaire est valide, un nouvel objet Peptide est créé à partir des données saisies dans le formulaire.
    - Les tags associés au peptide sont récupérés à partir des données POST et associés au peptide.
    - Enfin, le peptide est enregistré dans la base de données et l'utilisateur est redirigé vers une autre page ("/GenomeTag:create_peptide").

Formulaire :

Classe AttributionForm

- **Champs**:
    - Creator: Un champ CharField qui affiche le créateur de l'annotation. Ce champ est en lecture seule (readonly) et sa valeur est fournie lors de l'initialisation du formulaire.
    - Annotator: Un champ EmailField qui permet de saisir l'adresse e-mail de l'annotateur auquel l'annotation doit être attribuée.
    - Chromosome: Un champ ChoiceField qui permet de sélectionner le chromosome auquel l'annotation sera associée. Les choix disponibles sont dynamiquement générés à partir des instances de Genome et Chromosome dans la base de données. Ce champ est requis.
    - Strand: Un champ ChoiceField qui permet de sélectionner le brin (strand) auquel l'annotation sera associée. Les choix disponibles sont "+" et "-".
    - Start: Un champ IntegerField qui permet de saisir la position de départ de l'annotation sur le chromosome.
    - End: Un champ IntegerField qui permet de saisir la position de fin de l'annotation sur le chromosome.
- **Méthode __init__**:
    - Cette méthode personnalisée est utilisée pour initialiser le formulaire. Elle configure le champ Chromosome en modifiant son label pour le rendre plus explicite et en le rendant requis. De plus, elle génère dynamiquement les choix pour le champ Chromosome en parcourant les instances de Genome et Chromosome dans la base de données et en construisant une liste de tuples contenant l'ID du génome et le numéro d'accès du chromosome.