

NOTE DE SYNTHÈSE



HAFAIDH MOHAMED

Deuxième Année **BTS SIO-2**

Stage en entreprise

06 Janvier - 14 Février 2025

Sommaire

1. Présentation de l'entreprise-----	3
1.1 Organigramme de l'entreprise-----	3
1.2 Contexte du projet-----	4
1.3 Pourquoi ce projet ?-----	4
1.4 Contraintes et défis-----	5
1.5 Pour qui ce projet est-il destiné ?-----	5
2. Présentation de mon Poste de Travail-----	6
2.1 Présentation du projet-----	6
3. Installation et Configuration d'un Environnement de Développement Django sous Linux-----	7
3.1 Création d'un nouveau projet Django-----	7
3.2 Migration de la base de données-----	9
3.3 Création d'un utilisateur administrateur-----	9
3.4 Démarrage du serveur de développement-----	22
3.5 React-Native-----	
4. Formulaire Hummov -----	11
5. Remerciements-----	28
6. Conclusion-----	29

Présentation de l'entreprise

L'entreprise Humoov, fondée le 1er mars 2018, est une entreprise innovante spécialisée dans la collecte de données personnelles, offrant aux clients une compensation financière en échange de leurs données.

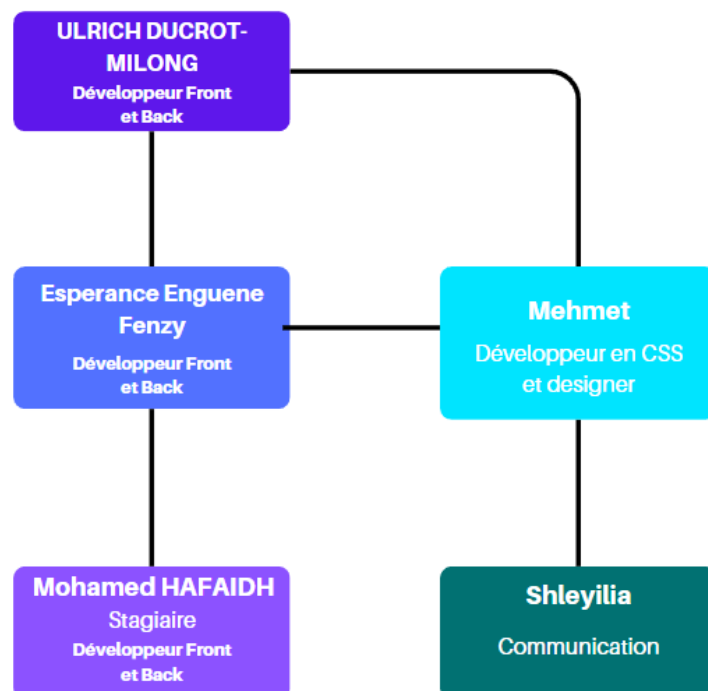
Pour chaque utilisateur qui partage ses données personnelles avec Humoov, Humoov offre 10 euros par mois, soit 120 euros par an. Ce programme reflète l'engagement à valoriser les contributions des utilisateurs et à leur offrir une reconnaissance financière pour leur participation.

Organigramme de l'entreprise



HUMOOV

**COMPANY
ORGANIZATION**



Contexte du projet

L'objectif principal est de faciliter la collecte de données des utilisateurs de manière efficace et sécurisée via l'application développée par Humoov. Les formulaires doivent être intuitifs, clairs et adaptés aux attentes des utilisateurs pour maximiser leur engagement tout en respectant les normes en matière de protection des données personnelles (RGPD).

Pourquoi ce projet ?

La création de formulaires dédiés à la collecte des données répond à plusieurs facteurs :

1. **Optimiser l'expérience utilisateur** : Des formulaires bien conçus augmentent les chances que les utilisateurs remplissent les informations demandées sans abandonner le processus.
2. **Respecter les contraintes légales** : La collecte et le traitement des données doivent se conformer aux réglementations en vigueur, notamment en termes de **transparence** et de **sécurité**.
3. **Assurer la pertinence des données collectées** : Les formulaires doivent permettre de recueillir des données précises et exploitables pour répondre aux besoins de l'entreprise.

Contraintes et défis

Techniques :

- Maîtriser les outils et technologies nécessaires au développement, notamment Python et Django.
- Garantir la sécurité des données collectées.

Légales :

- Respecter la réglementation RGPD (ex. : gestion des consentements, droit à l'oubli, transparence sur l'usage des données).

Humaines :

- Proposer une interface qui inspire confiance aux utilisateurs et limite les abandons.

Pour qui ce projet est-il destiné ?

Le projet s'adresse principalement aux utilisateurs de l'application Humoov, qui sont prêts à partager leurs données en échange d'une rémunération.

Ces utilisateurs doivent se sentir en confiance et être assurés que leurs données seront utilisées de manière responsable.

Présentation de mon Poste de Travail

En tant que stagiaire en développement web, je vais vous présenter aujourd'hui mon environnement de travail ainsi que les projets qui me sont confiés. Lors de ma première semaine de stage, j'ai appris à travailler avec Python et le framework Django, ce qui m'a permis d'acquérir des bases solides pour les projets à venir.

Actuellement, je suis en charge du développement de formulaires destinés à recueillir des données, ce qui me permet de mettre en pratique mes compétences tout en évoluant dans mon rôle.

Présentation du projet

Pour mon deuxième stage chez Humoov, j'ai eu la chance de participer à des réunions où j'ai pu donner des avis ainsi que des améliorations pour l'application. Mon projet était de créer des formulaires pour recueillir des données.

Pour cela j'ai dû installer le projet que mon tuteur m'avait envoyé sur GitHub.



Une fois installé j'ai dû installer python sur mon poste de travail.

```
sudo apt update
sudo apt install software-properties-common
sudo add-apt-repository ppa:deadsnakes/ppa
sudo apt install python3.10
```

Python installé vous devez regarder la version en faisant :

```
user500@pop-os:~$ python3 --version
Python 3.10.12
user500@pop-os:~$
```

Installation et Configuration d'un Environnement de Développement Django sous Linux

Création de l'environnement virtuel :

Il faut entrer la commande `python3 -m venv humoov-main`

```
user500@pop-os:~$ python3 -m venv humoov-main
```

Accès au dossier de l'environnement :

Se déplacer dans le répertoire de l'environnement : `cd Bureau/humoov-main`

```
user500@pop-os:~/Bureau/humoov-main$
```

Activation de l'environnement :

Activer l'environnement virtuel pour utiliser les packages spécifiques : `source bin/activate`

```
user500@pop-os:~/Bureau/humoov-main$ source bin/activate  
(humoov) user500@pop-os:~/Bureau/humoov-main$
```

Installation de Django

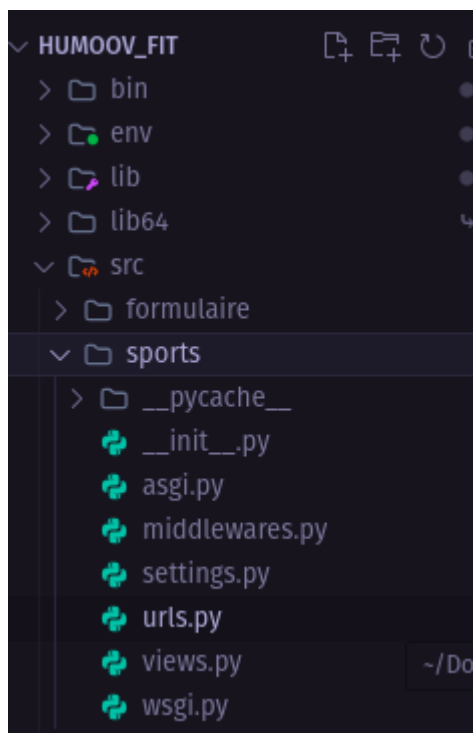
```
(humoov) user500@pop-os:~/Bureau/humoov-main$ pip install django
```

Création d'un nouveau projet Django

Pour démarrer un projet, Django propose une commande permettant de générer automatiquement la structure de base :

`django-admin startproject sports`

Cela fait nous avons des fichiers qui se créent automatiquement



Chaque fichier joue un rôle spécifique dans le fonctionnement et la configuration de l'application Django.

__pycache__/: Ce dossier est automatiquement généré par Python pour stocker les fichiers compilés des modules Python. Il n'est pas nécessaire de s'en préoccuper directement.

__init__.py: Ce fichier indique à Python que ce dossier doit être traité comme un module. Il est souvent vide, mais il peut être utilisé pour exécuter du code d'initialisation pour le module.

asgi.py: Ce fichier est utilisé pour déployer l'application Django avec ASGI (Asynchronous Server Gateway Interface), qui permet de gérer les requêtes asynchrones. C'est utile pour les applications en temps réel ou nécessitant des communications WebSocket.

middlewares.py: Ce fichier contient les middlewares de votre application. Les middlewares sont des composants logiciels qui s'exécutent entre le serveur web et votre application Django. Ils peuvent traiter les requêtes et réponses avant qu'elles ne soient gérées par les vues.

settings.py: Ce fichier contient toutes les configurations et paramètres de votre projet Django, comme les configurations de la base de données, les applications installées, les paramètres de sécurité, etc.

urls.py: Ce fichier définit les URL que votre application va gérer et les vues correspondantes. Il fait le lien entre les URL saisies par l'utilisateur et les fonctions de vue qui génèrent les réponses.

views.py: Ce fichier contient les fonctions de vue, qui gèrent la logique de traitement des requêtes et génèrent les réponses pour les utilisateurs. Les vues peuvent rendre des pages HTML, rediriger vers d'autres pages, etc.

wsgi.py: Ce fichier est utilisé pour déployer l'application Django avec WSGI (Web Server Gateway Interface), qui est le standard pour les communications entre les serveurs web et les applications web Python.

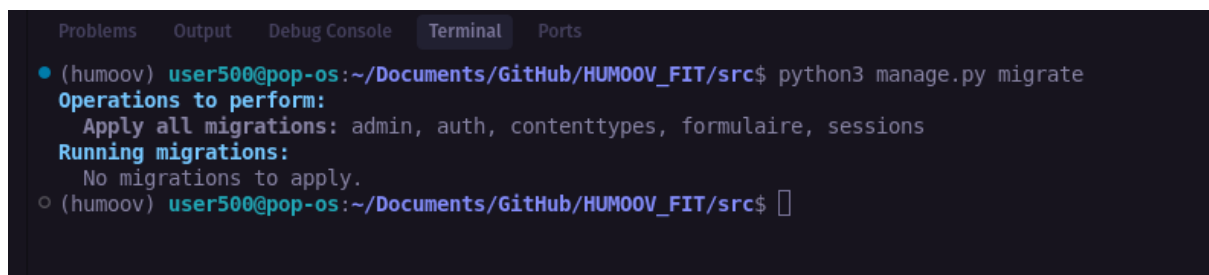
Migration de la base de données

Django utilise un ORM (Object Relational Mapping) pour interagir avec des bases de données.

Avant de commencer, il faut appliquer les migrations initiales qui créent les tables nécessaires (comme pour l'authentification).

Commande :

`python3 manage.py migrate`



```
Problems Output Debug Console Terminal Ports
• (humov) user500@pop-os:~/Documents/GitHub/HUM00V_FIT/src$ python3 manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, formulaire, sessions
Running migrations:
  No migrations to apply.
○ (humov) user500@pop-os:~/Documents/GitHub/HUM00V_FIT/src$
```

Cette étape configure la base de données par défaut (SQLite, mais cela peut être changé ultérieurement).

Création d'un utilisateur administrateur

Django propose une interface d'administration pour gérer les utilisateurs, les données, etc. Pour y accéder, il faut créer un utilisateur administrateur avec :

```
python3 manage.py createsuperuser
```

```
Username (leave blank to use 'user500'): mo
Email address:
Password:
Password (again):
This password is too short. It must contain at least 8 characters.
This password is too common.
This password is entirely numeric.
Bypass password validation and create user anyway? [y/N]: y
Superuser created successfully.
(humoov) user500@pop-os:~/Documents/GitHub/HUM00V_FIT/src$
```

Le système demande :

- Un nom d'utilisateur.
- Une adresse e-mail (facultative).
- Un mot de passe sécurisé.

Démarrage du serveur de développement

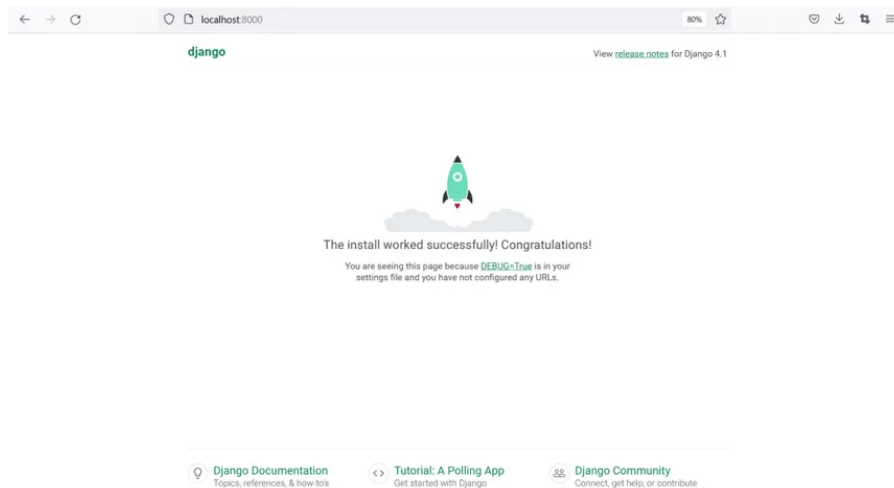
Enfin, pour tester l'environnement de travail, il faut démarrer le serveur web intégré :

```
python3 manage.py runserver
```

```
(humoov) user500@pop-os:~/Documents/GitHub/HUM00V_FIT/src$ python3 manage.py runserver 8080
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
January 14, 2025 - 17:32:47
Django version 5.1.4, using settings 'sports.settings'
Starting development server at http://127.0.0.1:8080/
Quit the server with CONTROL-C.
```

Voici la page par défaut quand vous lancez votre projet :



Formulaire Humoov

Comme je l'ai dit précédemment, j'ai eu l'accès au projet grâce au repository, donc un dépôt sur GitHub. Dedans il y avait les dossiers ainsi que les fichiers du projet que je devais toucher pour que je puisse développer.

Pour mettre le projet en marche, je devais refaire les manipulations précédentes, donc créer un environnement virtuel Django, puis activer le projet grâce au source bin/active.

```
Problems  Output  Debug Console  Terminal  Ports
● user500@pop-os:~/Documents/GitHub/HUMOOV_FIT$ source bin/activate
○ (humoov) user500@pop-os:~/Documents/GitHub/HUMOOV_FIT$
```

Cela fait nous allons accéder au répertoire src donc le dossier racine du projet.

```
Problems  Output  Debug Console  Terminal  Ports
● (humoov) user500@pop-os:~/Documents/GitHub/HUMOOV_FIT$ cd src/
○ (humoov) user500@pop-os:~/Documents/GitHub/HUMOOV_FIT/src$
```

Puis Makemigrations, cette commande **crée des fichiers de migration** qui décrivent les modifications apportées aux modèles Django (ajout, suppression ou modification de champs, de tables, etc.).

Puis Migrate c'est la commande qui **applique les migrations** à la base de données, elle synchronise l'état de la base de données avec les modèles définis dans le code.

```
(humooov) user500@pop-os:~/Documents/GitHub/HUM00V_FIT/src$ python3 manage.py makemigrations
No changes detected
(humooov) user500@pop-os:~/Documents/GitHub/HUM00V_FIT/src$ python3 manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, formulaire, sessions
Running migrations:
  No migrations to apply.
(humooov) user500@pop-os:~/Documents/GitHub/HUM00V_FIT/src$
```

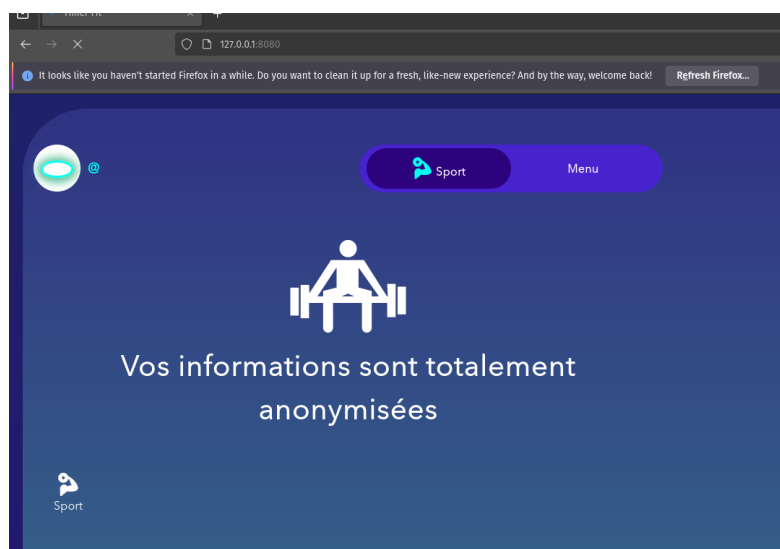
Une fois cela terminé, vous devez lancer le serveur. La commande est **python3 manage.py runserver 8080**

```
(humooov) user500@pop-os:~/Documents/GitHub/HUM00V_FIT/src$ python3 manage.py runserver 8080
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
January 24, 2025 - 14:36:55
Django version 5.1.4, using settings 'sports.settings'
Starting development server at http://127.0.0.1:8080/
Quit the server with CONTROL-C.


```

Maintenant voici la page du formulaire dont je suis en charge pour développer les questions pour la collecte de données des utilisateurs :



Le thème du formulaire c'est le sport, le premier jour de stage j'étais chargé de trouver des questions que je devais écrire sur un GoogleDocs, j'ai dû écrire des questions logiques dont on peut récolter des informations pertinentes sur l'utilisateur.

Cela terminé, je devais envoyer les questions à mon tuteur.



SUJET : Formulaire sport

Date : 16-12-2024

Auteur : Mohamed HAFAIHD / Espérance ENGUENE-FENZY /Ulrich DUCROT-MILONG /

I - CORRECTION

Titre du questionnaire : Sport
Question sur le sport :

Faites-vous du sport

- ☐ Oui
- ☒ Non

Pour quelle raison ?

- ☐ J'ai pas le temps
- ☐ J'ai pas la motivation
- ☐ J'aime pas
- ☐ Problème de santé
- ☐ Problème de surpoids
- ☐ Autre

Il m'a renvoyé un corrigé, maintenant je peux passer au côté code du formulaire.

Tout d'abord il faut créer la question dans le fichier HTML :

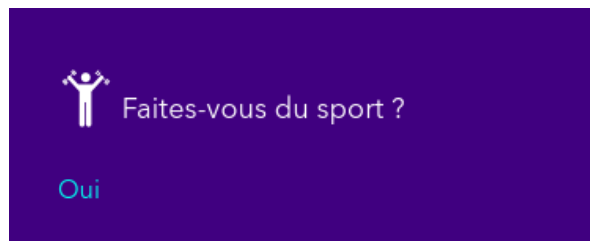
```
</> timer_sports.html
```

Maintenant il faudrait écrire en html pour qu'on puisse affiché la question :

Code source :

```
<!-- SPORTFIT STATUS INPUT -->
<div class="col-md-6">
  <label class="label-text hu-regular hu-white">
    <i class="icofont-gym icofont-2x" id="icon-container"></i> Faites-vous du sport ?
  </label>
  <select class="form_select hu-regular hu-timer hu-cust hu-id-input" id="hu_fit">
    {% if my_sports.sport_fit == False %}
      <option value="NO">Non</option>
      <option value="YES">Oui</option>
    {% else %}
      <option value="YES">Oui</option>
      <option value="NO">Non</option>
    {% endif %}
  </select>
</div>
<!-- SPORTFIT STATUS INPUT ENDS -->
```

Version WEB :



Cependant j'ai créé un fichier en JS pour intégrer la partie logique, si oui cela m'affiche les autres questions, sinon cela m'affiche Pour quelle raison ?

```
JS timer_sports.js
```

```
$(document).ready(function() {
  var selectChoiceInput = document.getElementById('hu_fit');
  var iconContainer = document.getElementById('hu_input_div');
  var reasonDiv = document.getElementById('reason_div');

  console.log('selectChoiceInput:', selectChoiceInput);
  console.log('iconContainer:', iconContainer);
  console.log('reasonDiv:', reasonDiv);

  function updateIcons() {
    if (selectChoiceInput.value === 'YES') {
      iconContainer.classList.remove('not-visible');
      reasonDiv.classList.add('not-visible');
    } else if (selectChoiceInput.value === 'NO') {
      iconContainer.classList.add('not-visible');
      reasonDiv.classList.remove('not-visible');
    }
  }

  // Appeler updateIcons lors d'un changement dans selectChoiceInput
  selectChoiceInput.addEventListener('change', updateIcons);

  // Appeler updateIcons au chargement de la page
  updateIcons();
});
```

Ce code JavaScript utilise **jQuery** pour manipuler l'affichage de certains éléments HTML en fonction de la sélection d'un utilisateur dans une liste déroulante.

Chargement du script à l'initialisation de la page

```
$(document).ready(function() {
```

Cette ligne signifie que le script s'exécute uniquement lorsque le document HTML est complètement chargé.

Déclaration des variables :

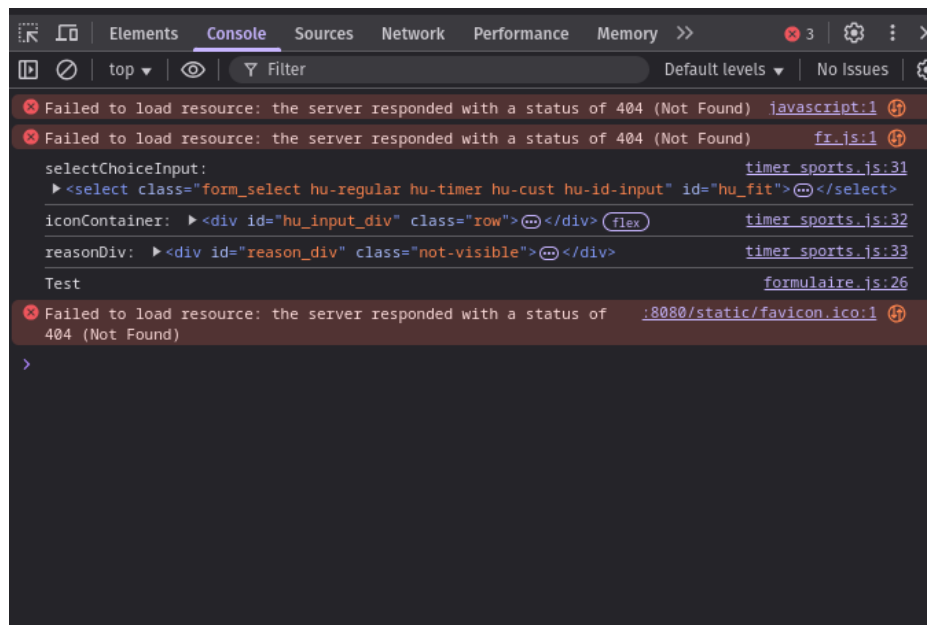
```
var selectChoiceInput = document.getElementById('hu_fit');  
var iconContainer = document.getElementById('hu_input_div');  
var reasonDiv = document.getElementById('reason_div');
```

Ces variables stockent des références aux éléments HTML ayant les ID **hu_fit**, **hu_input_div**, et **reason_div**

Affichage des éléments dans la console :

```
console.log('selectChoiceInput:', selectChoiceInput);  
console.log('iconContainer:', iconContainer);  
console.log('reasonDiv:', reasonDiv);
```

Cela permet de vérifier si les éléments **HTML** ont bien été sélectionnés.



```
function updateIcons() {
    if (selectChoiceInput.value === 'YES') {
        iconContainer.classList.remove('not-visible');
        reasonDiv.classList.add('not-visible');
    } else if (selectChoiceInput.value === 'NO') {
        iconContainer.classList.add('not-visible');
        reasonDiv.classList.remove('not-visible');
    }
}
```

Cette fonction met à jour la visibilité des éléments :

- Si l'utilisateur sélectionne "YES", l'élément `iconContainer` devient visible et `reasonDiv` est caché.
- Si l'utilisateur sélectionne "NO", `iconContainer` est caché et `reasonDiv` devient visible.

Elle utilise la classe CSS "`not-visible`" pour gérer la visibilité des éléments.

Ajout d'un écouteur d'événements :

```
// Appeler updateIcons lors d'un changement dans selectChoiceInput
selectChoiceInput.addEventListener('change', updateIcons);
```

Cette ligne détecte un changement de sélection dans `selectChoiceInput` et appelle `updateIcons()`.

Exécution de `updateIcons()` au chargement de la page :

```
// Appeler updateIcons au chargement de la page
updateIcons();
```

Exemple de la version web :

Répondez le plus précisément possible

 Faites-vous du sport ?

 Pour quelle raison ?

Non Sélectionner

Répondez le plus précisément possible

 Faites-vous du sport ?

Oui

Maintenant j'ai du écrire toute les questions pour le **Oui** cela ce passe dans le fichier HTML :

```
<!-- FREQUENCY INPUT -->
<div class="col-md-6">
  <label class="label-text hu-regular hu-white">
    <i class="icofont-gym-alt-2 icofont-2x"></i> à quelle fréquence pratiquez-vous des activités sportives ?
  </label>
  <select class="form select hu-regular hu-timer hu-cust hu-id-input" id="hu_fit_frequency">
    <option value="{{my_sports.frequency}}">{{my_sports.get_frequency_display}}</option>
    <option value="TLJP">Tous les jours ou presque</option>
    <option value="3A4">3 à 4 fois par semaine</option>
    <option value="1A2">1 à 2 fois par semaine</option>
    <option value="TL15">Tous les 15 jours</option>
    <option value="TLM0">Tous les mois</option>
    <option value="TL23">Tous les 2 ou 3 mois</option>
    <option value="P0">Plus occasionnellement</option>
    <option value="AUTRE">Autre</option>
  </select>
</div>
<!-- FREQUENCY INPUT ENDS -->
```

 à quelle fréquence pratiquez-vous des activités sportives ?

Sélectionner

Cela fait j'ai du codé le reste des questions que mon tuteur m'a fournie :

 Faites-vous du sport ?	
Oui	
 à quelle fréquence pratiquez-vous des activités sportives ?	 Quel(s) type(s) de sport pratiquez-vous régulièrement ?
Sélectionner	Wing chun
 Depuis combien de temps pratiquez-vous votre sport ou activité physique préféré ?	 À quelle fréquence pratiquez-vous un sport ou une activité physique chaque semaine ?
Sélectionner	Sélectionner
 Quelles sont vos principales motivations pour faire du sport ?	 Avez-vous un(e) coach personnel ?

 Quelles sont vos principales motivations pour faire du sport ?	 Avez-vous un(e) coach personnel ?
Sélectionner	Sélectionner
 Préférez-vous pratiquer des sports ?	 Où pratiquez-vous généralement vos activités sportives ?
Sélectionner	Sélectionner
 À hauteur de combien estimez-vous vos dépenses mensuelles pour le sport ?	
Sélectionner	
 Avez-vous des objectifs spécifiques liés à votre pratique	 Avez-vous déjà participé à des compétitions ?

Tout cela terminé coté HTML, maintenant on doit relier ses données html a notre base de données en SQL Lite

```
timer = models.OneToOneField(User, on_delete=models.CASCADE, related_name='timer_sports', null=True)
sport_fit = models.BooleanField(default=False, null=True, blank=True)
reason = models.CharField(max_length=20, choices=REASON_CHOICES, null=True, default='S', blank=True)
frequency = models.CharField(max_length=30, choices=FREQUENCY_CHOICES_FIT, null=True, default='S', blank=True)
practice = models.CharField(max_length=100, choices=PRACTICE_CHOICES_FIT, null=True, default='S', blank=True)
time = models.CharField(max_length=30, choices=TIME_CHOICES_FIT, null=True, default='S', blank=True)
week = models.CharField(max_length=20, choices=TIME_CHOICES_WEEKLY_HOURS, null=True, default='S', blank=True)
motivation = models.CharField(max_length=20, choices=CHOICES_MOTIVATION_FIT, null=True, default='S', blank=True)
coach = models.CharField(max_length=2, choices=CHOICES_COACH_FIT, null=True, default="S", blank=True)
depense = models.CharField(max_length=20, choices=CHOICES_DEPENSE_FIT, null=True, default='S', blank=True)
prefer = models.CharField(max_length=10, choices=CHOICES_PREFER_FIT, null=True, default='S', blank=True)
sporting = models.CharField(max_length=10, choices=CHOICES_SPORTING_FIT, null=True, default='S', blank=True)
monthly = models.CharField(max_length=30, choices=CHOICES_MONTHLY_FIT, null=True, default='S', blank=True)
objectives = models.CharField(max_length=30, choices=CHOICES_OBJECTIVES_FIT, null=True, default='S', blank=True)
participation = models.CharField(max_length=2, choices=CHOICES_PARTICIPATION_FIT, null=True, default='S', blank=True)
remporte = models.CharField(max_length=2, choices=CHOICES_REMPORTE_FIT, null=True, default='S', blank=True)
profit = models.CharField(max_length=30, choices=CHOICES_PROFIT_FIT, null=True, default='S', blank=True)
obstacles = models.CharField(max_length=30, choices=CHOICES_OBSTACLES_FIT, null=True, default='S', blank=True)
application = models.CharField(max_length=2, choices=CHOICES_APPLICATION_FIT, null=True, default='S', blank=True)
use = models.CharField(max_length=30, choices=CHOICES_RAISON_USE_FIT, null=True, default='S', blank=True)
connect = models.CharField(max_length=30, choices=CHOICES_FIT_CONNECT, null=True, default='S', blank=True)
appareils = models.CharField(max_length=30, choices=CHOICES_FIT_APPAREILS, null=True, default='S', blank=True)
new = models.CharField(max_length=30, choices=CHOICES_FIT_NEWS, null=True, default='S', blank=True)
```

Ce modèle est une **table de base de données** Django, qui stocke des réponses utilisateur concernant leurs habitudes sportives.

Il est lié à un utilisateur via **OneToOneField(User)**, assurant qu'un seul ensemble de réponses est associé à chaque utilisateur.

Les autres champs sont des **choix prédéfinis** (avec **choices**) ou des booléens, permettant d'enregistrer des préférences, motivations, fréquence d'entraînement, et autres données liées à leur activité physique.

Cela crée nous devons créer la page admin pour qu'on puisse collecter les données des utilisateur grâce aux questionnaires :

Pour créer la page admin on doit créer un fichier admin en .py



```

> formulaire > admin.py > ...
1  from django.contrib import admin
2  from .models import FitPreferences
3
4  class FitPreferencesAdmin(admin.ModelAdmin):
5      # Liste des champs à afficher dans la liste d'administration
6      list_display = [
7          'sport_fit', 'reason', 'frequency', 'practice', 'time', 'week',
8          'motivation', 'coach', 'depense', 'prefer', 'sporting', 'monthly',
9          'objectives', 'participation', 'remporte', 'profit', 'obstacles',
10         'application', 'use', 'connect', 'appareils', 'new',
11
12
13
14         'created', 'updated', 'timer',
15         'completion', 'completed',
16
17     ]
18
19     # Ajouter une recherche sur certains champs pour faciliter la gestion
20     search_fields = ['sport_fit', 'practice', 'reason', 'frequency']
21
22     # Ajouter des filtres pour les champs
23     list_filter = ['frequency', 'reason', 'practice']
24

```

Le code que j'ai mis en place permet d'afficher et de gérer les préférences sportives des utilisateurs dans l'interface d'administration de Django.

127.0.0.1:8000/admin/formulaire/fitpreferences/

Avec L'URL 127.0.0.1:8000 on ajoute le /admin pour accéder à notre page administrateur.

Django administration

Username:

Password:

Log in

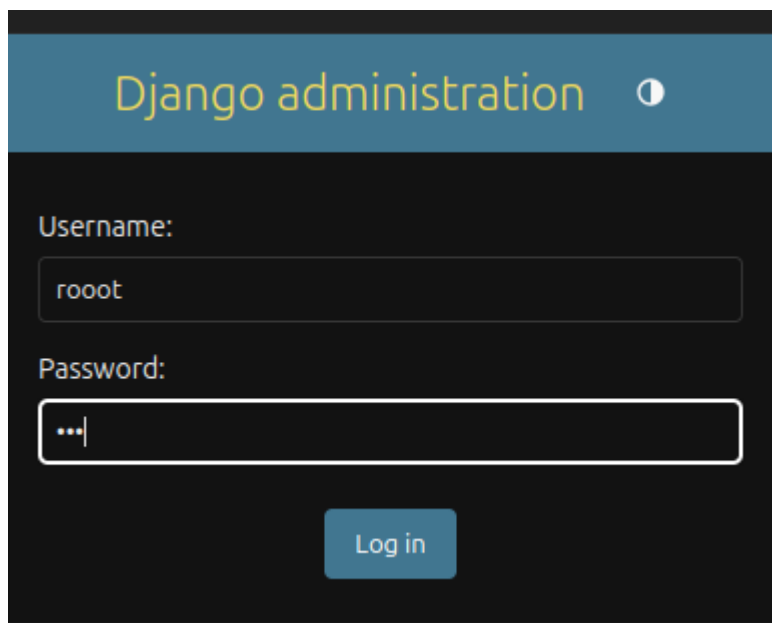
Nous arrivons sur une page de connexion avec un username ainsi que le Password.

Pour créer votre utilisateur nous devons taper sur notre terminale de notre Projet, `python3 manage.py createsuperuser` :

```
(humooov) yooceyy@yooceyy-Latitude-E7270:~/Bureau/HUM00V_FIT/src$ python3 manage.py createsuperuser
Username (leave blank to use 'yooceyy'): 
```

```
(humooov) yooceyy@yooceyy-Latitude-E7270:~/Bureau/HUM00V_FIT/src$ python3 manage.py createsuperuser
Username (leave blank to use 'yooceyy'): root
Error: That username is already taken.
Username (leave blank to use 'yooceyy'): rooot
Email address:
Password:
Password (again):
Error: Blank passwords aren't allowed.
Password:
Password (again):
Error: Blank passwords aren't allowed.
Password:
Password (again):
This password is too short. It must contain at least 8 characters.
This password is too common.
This password is entirely numeric.
Bypass password validation and create user anyway? [y/N]: y
Superuser created successfully.
```

Maintenant on se connecté à notre admin avec notre `rooot` ainsi que votre mot de passe.



FORMULAIRE		Action: ----- Go 0 of 1 selected											
Fit preferences + Add		☐	SPORT FIT	REASON	FREQUENCY	PRACTICE	TIME	WEEK	MOTIVATION	COACH	DEPENSE	PREFER	SPORTING
		☒	Sélectionner	Sélectionner	Wing chun	Sélectionner	Sélectionner	Sélectionner	Sélectionner	Sélectionner	Sélectionner	Sélectionner	Sélectionner

React Native

Durant ma cinquième semaine de stage on a fait une réunion entre développeurs ainsi que mon tuteur pour faire une migration de l'application web sous React Native.

Chaque personne a donné son idée d'amélioration pour la migration donc avantage et inconvénient.

Pour cela je devais faire un tableau pour montrer les avantages de React Native ainsi que ses inconvénients.

Une migration de **Django** (backend) vers **React Native** (frontend mobile) implique un changement majeur dans l'architecture de l'application.

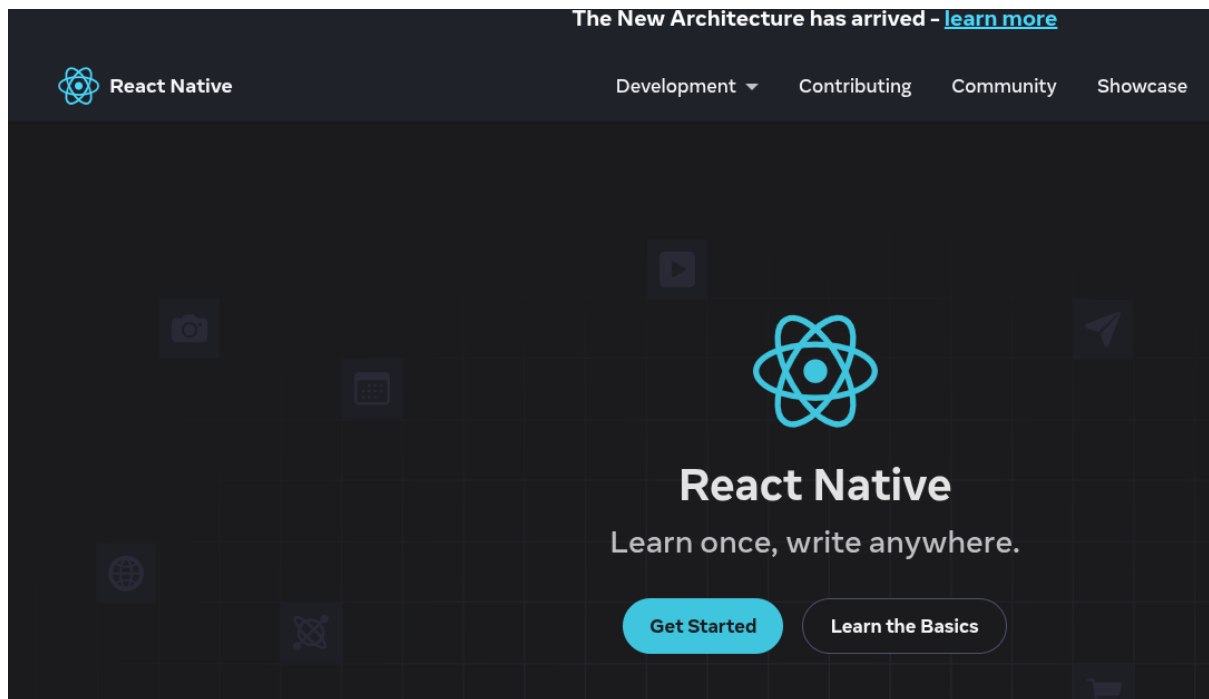
Avantages 	Inconvénients 
- Meilleure expérience mobile - Application plus fluide sur mobile (code natif optimisé) - Un seul code pour android et IOS - Large communauté et nombreuses bibliothèques React - Django peut toujours être utilisé comme API avec Django REST Framework (DRF) - Plus rapide grâce aux composants réutilisables - Moins de frais si l'équipe connaît déjà JavaScript/React	- Nécessite une refonte complète de l'interface - Peut être plus lent que du pur natif (Swift/Kotlin) - Certaines dépendances peuvent être instables - Nécessite d'implémenter une API REST ou GraphQL - Temps d'apprentissage pour les devs Django

Grâce à mon projet GSB que j'ai réalisé pendant ma deuxième année de BTS j'avais déjà quelques connaissances sur le Framework ainsi que la mise en place.

Donc tout d'abord je me suis renseigné sur le Framework React Native car l'avantage c'est aussi de déployer sur mobile et c'était le but de Humoov car avant cela Humoov passait par un système de rétroprojection de leur site web sur mobile.

Pour le début de l'application j'ai consulté la documentation de React Native :

<https://reactnative.dev/>



Puis j'ai procédé à l'installation :

Node JS :

Télécharger et installer nvm :

```
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.40.1/install.sh | bash#
```

Télécharger et installer Node.js :

```
nvm install 22# Vérifier la version de Node.js :
```

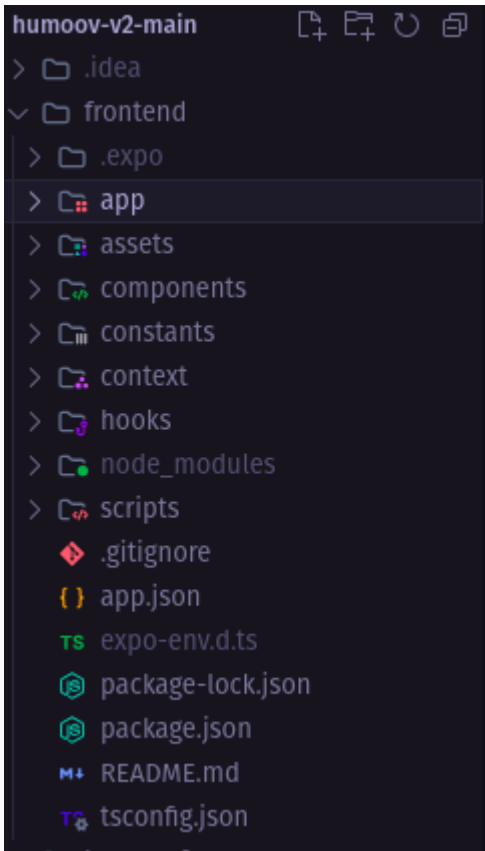
```
node -v # Doit afficher "v22.14.0".nvm current # Doit afficher "v22.14.0".
```

Vérifier la version de npm :

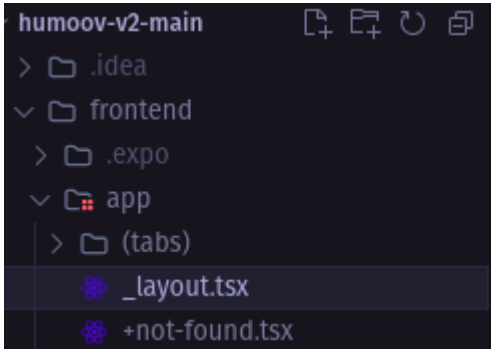
```
npm -v # Doit afficher "10.9.2".
```

React Native :

npx create-expo-app@latest



Voici le projet sous React-Native :
humoov-V2-main



+not-found.tsx	_layout.tsx
- Ce fichier gère les erreurs 404 (page non trouvée). - Si un utilisateur accède à une route qui n'existe pas, ce composant est affiché.	- Ce fichier sert de layout principal pour les pages de l'application. - Il est utilisé par Expo Router pour définir une navigation globale. - Typiquement, il contient une barre de navigation (tabs, stack, drawer, etc.).

```
import { Link, Stack } from 'expo-router';
import { StyleSheet } from 'react-native';

import { ThemedText } from '@components/ThemedText';
import { ThemedView } from '@components/ThemedView';

export default function NotFoundScreen() {
  return (
    <>
      <Stack.Screen options={{ title: 'Oops!' }} />
      <ThemedView style={styles.container}>
        <ThemedText type="title">This screen doesn't exist.</ThemedText>
        <Link href="/" style={styles.link}>
          <ThemedText type="link">Go to home screen!</ThemedText>
        </Link>
      </ThemedView>
    </>
  );
}
```

(**tabs**) / : Indique une **navigation avec onglets** (Tabs).

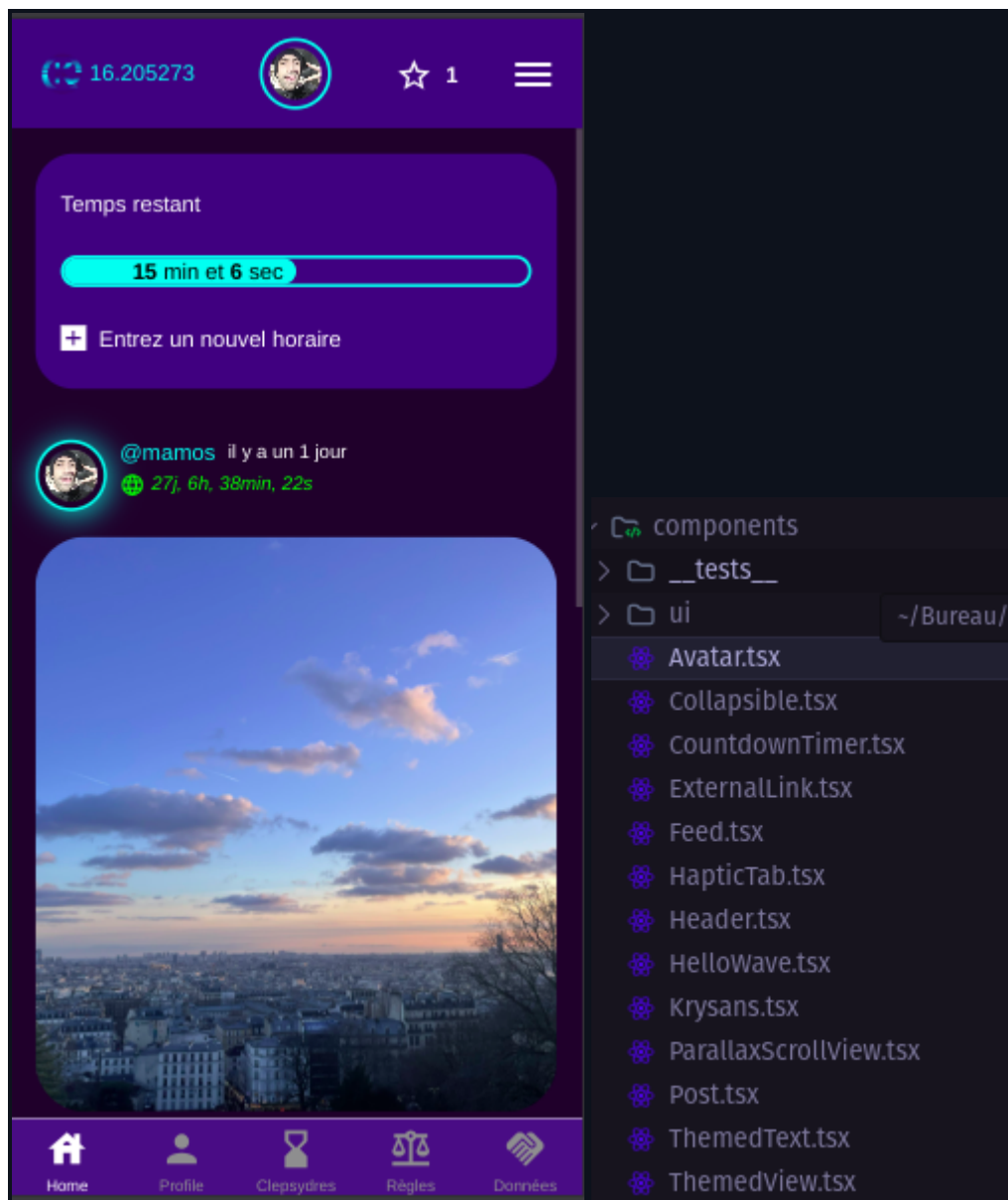
app/ : Contient toutes les pages et la structure des routes selon Expo Router.

Création du fichier profile en .jsx :  profile.tsx

```
export default function TabTwoScreen() {
  const [showDetails, setShowDetails] = useState(false);
  const [showEditProfile, setShowEditProfile] = useState(false);
  const [showMenu, setShowMenu] = useState(false);

  const menuItems = [
    { id: 'profile', icon: <Ionicons name="person" size={30} color="white" />, label: 'Profil' },
    { id: 'photos', icon: <Ionicons name="images" size={30} color="white" />, label: 'Photos' },
    { id: 'videos', icon: <Ionicons name="videocam" size={30} color="white" />, label: 'Vidéos' },
    { id: 'infinity', icon: <Ionicons name="infinite" size={30} color="white" />, label: 'Infini' },
    { id: 'dice', icon: <MaterialIcons name="casino" size={30} color="white" />, label: 'Dés' },
    { id: 'settings', icon: <Ionicons name="settings" size={30} color="white" />, label: 'Réglages' },
  ];
}
```





```
import { View, Image, ViewProps, ImageSourcePropType } from 'react-native';
import tw from 'twrnc';

type Props = ViewProps & {
  blur?: boolean;
  avatar?: ImageSourcePropType;
};

const Avatar = ({ blur = false, avatar }: Props) => {
  return (
    <View
      style={[
        tw`rounded-full border-2 border-[#01ffef] p-1`,
        blur && {boxShadow: '1px 2px 20px 0px rgba(1,255,239,0.9)'},
      ]}
    >
      <Image
        source={avatar}
        style={[
          tw`rounded-full`,
          { width: 33, height: 33 },
        ]}
      />
    </View>
  );
};

export default Avatar;
```

Malheureusement, je n'ai pas pu terminer le projet car mon stage s'est terminé avant que je puisse le finaliser. Je trouve cela dommage, car j'aurais aimé aller plus loin et voir le projet aboutir.

J'ai beaucoup appris en travaillant sur ce projet, et j'aurais aimé avoir plus de temps pour le finaliser. Cependant, l'expérience acquise me sera précieuse pour la suite

Remerciements

Je tiens à exprimer ma profonde gratitude à toute l'équipe de Humoov pour m'avoir accueilli et encadré tout au long de mon stage.

Cette expérience a été extrêmement enrichissante et m'a permis de renforcer mes compétences en développement informatique, tout en découvrant le fonctionnement d'un environnement professionnel.

Un grand merci à mon tuteur, Ulrich Ducrot, pour sa patience, ses conseils avisés et son accompagnement tout au long de mon apprentissage. Grâce à son soutien, j'ai pu progresser et mieux appréhender les bonnes pratiques.

Je remercie également mes collègues pour leur bienveillance et leur esprit d'équipe, qui ont contribué à rendre ce stage très agréable. Les échanges et les projets sur lesquels j'ai travaillé m'ont permis de développer mes compétences techniques et de mieux comprendre les enjeux du domaine.

Conclusion

Mon stage chez **Humoov** a été une expérience enrichissante, tant sur le plan technique que personnel. J'ai eu l'opportunité de travailler sur des projets concrets, d'améliorer mes compétences en développement et de découvrir le fonctionnement d'un environnement professionnel. Grâce à l'encadrement de l'équipe, j'ai pu apprendre de nouvelles méthodes de travail et approfondir ma compréhension des bonnes pratiques du développement.

Cependant, tout n'a pas été parfait. J'ai parfois eu du mal à faire entendre mes idées et à être pleinement intégré aux discussions techniques. Certaines de mes propositions n'ont pas toujours été prises en compte, ce qui a parfois rendu mon apprentissage plus difficile. Cette situation m'a néanmoins appris à mieux m'affirmer et à persévérer dans un cadre professionnel.

Malgré ces quelques défis, ce stage m'a permis de gagner en autonomie, d'acquérir de nouvelles compétences et de mieux comprendre les exigences du monde du travail. Je repars avec une expérience précieuse qui me servira dans la suite de mon parcours.

Je remercie toute l'équipe de **Humoov** pour cette opportunité et pour le savoir qu'ils m'ont transmis. Cette expérience, avec ses aspects positifs comme ses difficultés, restera une étape importante dans ma formation.