PARA Yaël
GONZALEZ Jules
GEITNER Teva

Rapport Technique

Ce document comporte les documents annexes qui n'ont pas pu être incorporés dans le rapport final.

Backlog :

A	B	C
essential elements for the final product		
element / target	sub divided target	priority (/10)
Application Flutter	Pages d'aceuilles de l'application	3
	Menu de choix d'un objet témoins et explication du procédé a l'utilisateur	5
	Prise de photo sur l'application (Habit + objet témoin)	9
Calcul de l'échelle	Définition des bordures de l'objet témoin par l'utilisateur sur la photo	8
	Indication de la taille réelle de l'objet par l'utilisateur	8
	Stockage de l'échelle	3
Traitement de l'image (vectorisation ...)	Génération de l'image vectorisée	4
	Stockage du modèle vectorisé en base de donnée	3
Recherche des points clé pour calculer les distance	Indication d'un exemple pour placer les points sur l'habit	5
	Interface utilisateur pour placer les points sur la photo	7
	Calcul et stockage des distances nécessaires pour l'habit sélectionné	7
Comparaison avec les modèles stockés	Comparaisons des distances de l'habit du client avec les distances l'habit souhaité dans des tailles différentes	6
Base de données des modèles vectorisés	Mise en place d'une base de donnée cloud	4

Liste des endpoints du backend :

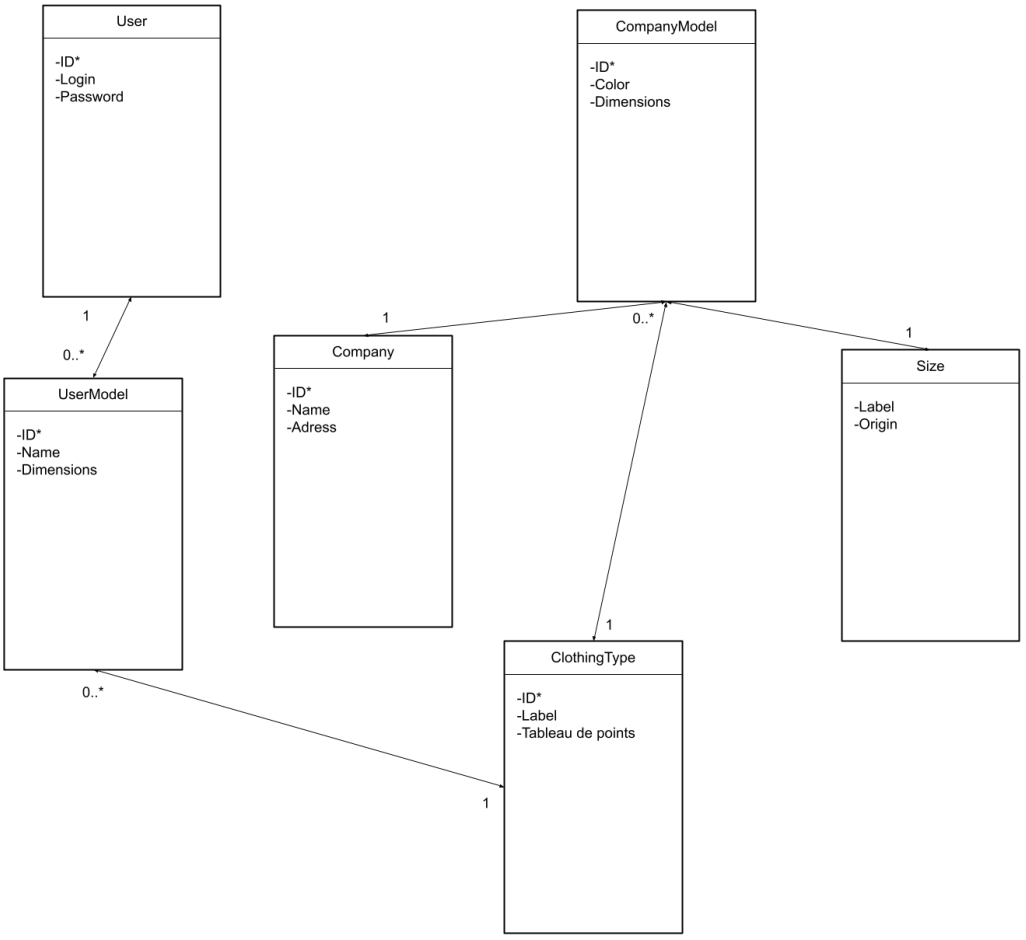
{api} = http://127.0.0.1:8000/

User	GET	{api}/polls/user/	Obtenir tous les users
	GET	{api}/polls/user/{id}/	Obtenir l'user d'identifiant {id}
	POST	{api}/polls/user/	Créer un user
	DELETE	{api}/polls/user/{id}/	Supprimer l'user d'identifiant {id}

UserModel	GET	{api}/polls/usermodel/	Obtenir tous les usermodels
	GET	{api}/polls/usermodel/{id}/	Obtenir l'usermodel d'identifiant {id}
	POST	{api}/polls/usermodel/savedimensions/	Créer un usermodel avec support pour déterminer les dimensions du vêtement par rapport à l'échelle
	POST	{api}/polls/usermodel/	Créer un usermodel
	DELETE	{api}/polls/usermodel/{id}/	Supprimer l'usermodel d'identifiant {id}
Company	GET	{api}/polls/company/	Obtenir toutes les compagnies
	GET	{api}/polls/company/{id}/	Obtenir la compagnie d'identifiant {id}
	POST	{api}/polls/company/	Créer une compagnie
	DELETE	{api}/polls/company/{id}/	Supprimer la compagnie d'identifiant {id}
CompanyModel	GET	{api}/polls/companymodel/	Obtenir tous les companymodels
	GET	{api}/polls/companymodel/{id}/	Obtenir le companymodel d'identifiant {id}
	POST	{api}/polls/companymodel/	Créer une compagnie
	DELETE	{api}/polls/companymodel/{id}/	Supprimer le companymodel d'identifiant {id}
Size	GET	{api}/polls/size/	Obtenir toutes les sizes
	GET	{api}/polls/size/{id}/	Obtenir la size d'identifiant {id}
	POST	{api}/polls/size/	Créer une size
	DELETE	{api}/polls/size/{id}/	Supprimer la size d'identifiant {id}

ClothingType	GET	{api}/polls/clothingtype/	Obtenir touts les clothingtype
	GET	{api}/polls/clothingtype/{id}/	Obtenir le clothingtype d'identifiant {id}
	POST	{api}/polls/clothingtype/	Créer un clothingtype
	DELETE	{api}/polls/clothingtype/{id}/	Supprimer le clothingtype d'identifiant {id}

Diagramme UML représentatif des Models de la base de données :



Guide d'utilisation de la partie Backend du système FitSize

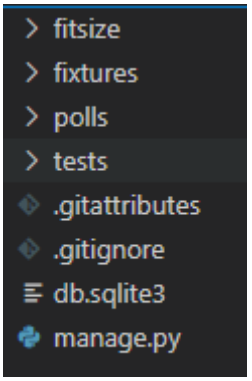
Installation des outils nécessaires :

La procédure d'installation est disponible sur le lien ci après. Il n'est pas nécessaire d'installer Apache et mod_wsgi afin d'utiliser notre projet :
https://docs.djangoproject.com/fr/2.2/topics/install/

Pour notre projet nous avons utilisé Django 4.0.3 et Python 3.10.2

Note : Lors du lancement du serveur, le virtual environment doit être activé. Le script de lancement du virtual environment n'est pas disponible sur le répertoire Git, il est fourni lors de l'installation de Django et toutes les indications sont dans la documentation de Django.

Architecture du projet :



Restauration de la base de données par défaut (utilisation de fake data):

Au clonage du projet, la base de données ne sera pas présente, il faudra l'initialiser avec les données par défaut si on désire avoir un exemple de stockage des modèles, et pour expérimenter avec Postman.
Afin de remettre la base de données par défaut et la peupler avec des données par défaut, on utilise les fixtures contenues dans le fichier fixtures/defaultData.json (possibilité de modifier, ajouter d'autres sets de fake data en créant un fichier json suivant cette syntaxe)
Si on souhaite obtenir une base de donnée vierge (avec uniquement les fake data) :

- Supprimer la base de données à la racine du projet (db.sqlite3)
- Se placer à la base de la racine et recréer les tables dans la base de données avec la commande : **python manage.py migrate**
- Envoi des fake data dans la base de données avec : **python manage.py loaddata fixtures/defaultData.json**

Se placer à la racine du projet afin d'effectuer les commandes suivantes :

Lancement du serveur : python manage.py runserver

```
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
March 17, 2022 - 16:35:18
Django version 4.0.3, using settings 'fitsize.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

Lancement tous les tests : python manage.py test

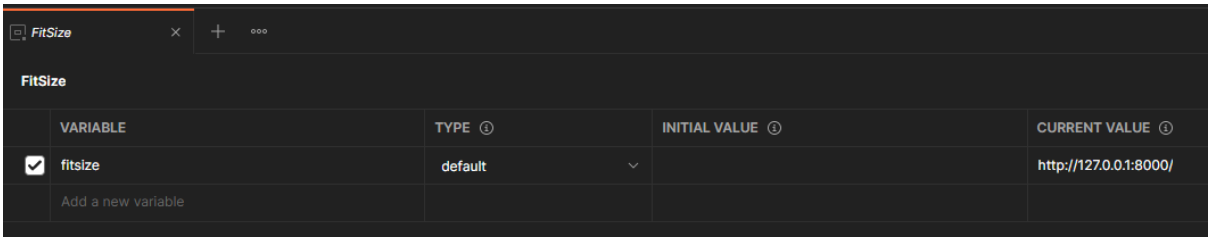
```
Found 31 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
...............................
----------------------------------------------------------------------
Ran 31 tests in 0.368s

OK
Destroying test database for alias 'default'...
```

Lancement d'un fichier de test spécifique : python manage.py test tests.testSerializers (pour lancer le fichier de test tests/testSerializers)

Utilisation des endpoints via Postman :

Les fichiers permettant d'importer le projet Postman utilisé afin d'utiliser les endpoints sont disponibles dans le dépôt git sous le répertoire 'postman'. Ne pas oublier d'importer le fichier d'environnement, une variable d'environnement {{fitsize}} est définie (égale au lien localhost où est lancé le serveur).



Le format des requêtes sous format JSON est disponible dans la section "Body" :