

[2022-
2023]

HELB MANAGER

COURS DE WEB II, Q4
BERKAY FINDIK

ILYA PRIGOGINE HELB

Table des matières

Introduction.....	2
Ressources du projet	2
Fonctionnalités de l'application	2
Connexion/Inscription.....	2
Home.....	3
Création de projets	4
Update project.....	4
Création d'une tâche	5
Le drag and drop, TaskFlow	5
Project history.....	6
Comment est construite l'historique ?	6
Progression des tâches	6
Comment est construite l'interface de progression ?.....	7
Système de notifications	7
Difficultés rencontrées.....	7
Assignation des rôles lors de la création, assignation des tâches aux user	7
Affichage des tâches sur la page détail du projet.....	8
Assignation de plusieurs personnes à un seul projet.....	9
Ajax, drag & drop update*	10
Filtrage de l'affichage des projets ou tâches.....	11
Limitations de mon application ?.....	11
Analyse	12
Conclusion	12
Bibliographie	13

Introduction

Dans le cadre du cours de WEB 2 de ce Q4, nous devons développer une application Web de gestion de projets avec Django, un framework de la bibliothèque Python.

L'application doit donner la possibilité aux utilisateurs de créer des projets, d'y ajouter des collaborateurs, créer des tâches et pouvoir gérer les statuts du projet et des tâches en tant réel avec également une interface d'historique.

Je vais d'abord vous présenter les ressources utilisées pour ce projet. Par la suite, je vous présenterai les fonctionnalités que l'application propose, les difficultés rencontrées lors du développement. Je terminerai avec la conclusion

Ressources du projet

Les technologies utilisées durant le développement de l'HELB Manager sont :

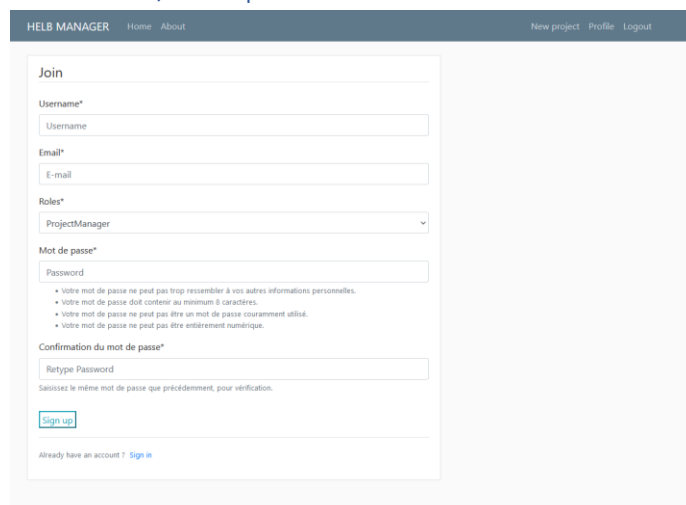
- Framework Django
- Bootstrap pour le CSS
- Ajax
- Javascript

Nous sommes partis d'une base fonctionnelle des tutoriels de Correy Schafer (un blog avec des posts), le but a été de s'éloigner petit à petit de cette base et adapter l'application à nos besoins.

Fonctionnalités de l'application

- Interface de connexion/inscription/Log In/Log out
- Interface Home
- Interface de création projets
 - o Interface de création de tâches
 - Interface de création de sous-tâches
 - o Gestion des statuts des tâches via une interface Drag & Drop
- Interface d'historiques/suivi
- Interface de progression des tâches
- Système de notifications

Connexion/Inscription



The screenshot shows the 'Join' form in the HELB Manager application. The form is located on the left side of a light gray page. At the top, there is a dark blue navigation bar with the text 'HELB MANAGER' and links 'Home' and 'About' on the left, and 'New project', 'Profile', and 'Logout' on the right. The form itself has a white background and a thin gray border. It contains the following fields and elements: a 'Join' title, a 'Username*' label with a text input field, an 'Email*' label with a text input field, a 'Roles*' label with a dropdown menu showing 'ProjectManager', a 'Mot de passe*' label with a password input field, and a 'Confirmation du mot de passe*' label with a 'Retype Password' input field. Below the password field, there are four small bullet points providing password requirements. At the bottom of the form is a 'Sign up' button and a link 'Already have an account ? Sign in'.

HELB MANAGER Home About New project Profile Logout

Join

Username*

Username

Email*

E-mail

Roles*

ProjectManager

Mot de passe*

Password

- Votre mot de passe ne peut pas trop ressembler à vos autres informations personnelles.
- Votre mot de passe doit contenir au minimum 8 caractères.
- Votre mot de passe ne peut pas être un mot de passe couramment utilisé.
- Votre mot de passe ne peut pas être entièrement numérique.

Confirmation du mot de passe*

Retype Password

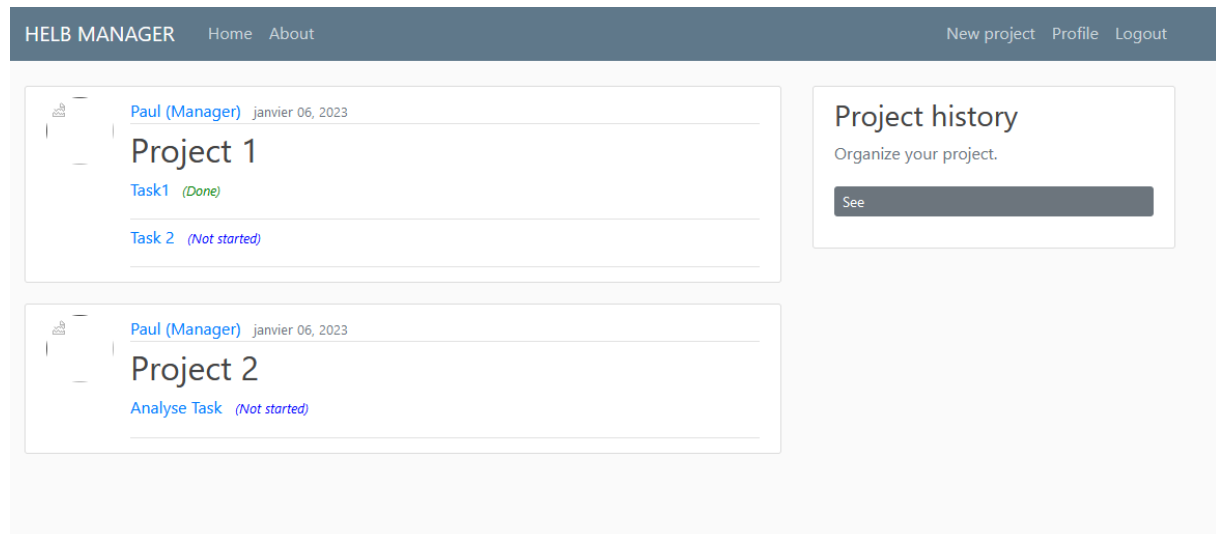
Saisissez le même mot de passe que précédemment, pour vérification.

Sign up

Already have an account ? [Sign in](#)

L'application propose une page de connexion/inscription basique en se basant sur le *Crispy Form* de Django. Une fois que vous aurez rempli le formulaire, la page vérifie si tout est bien rempli et cohérent, dans le cas contraire elle vous affichera un warning.

Home



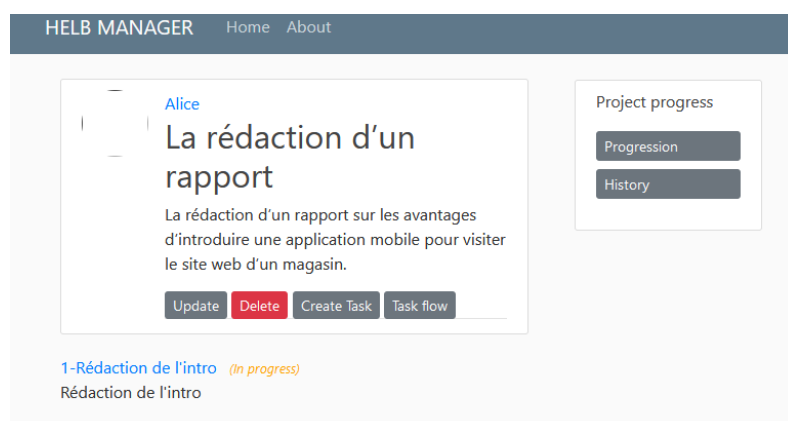
La page Home a deux affichages différents selon votre rôle :

- Project Manager : le home vous affichera tous vos projets et toutes les tâches du projet (interface divisée par projet, on voit les tâches du projet en dessous de celui-ci), vous serez le seul à pouvoir modifier les détails du projet et voir toutes les tâches et sous tâches du projet.
- Collaborater : le home vous affichera le projet auquel vous êtes assignés et seulement les tâches que vous devez faire, vous ne pouvez pas update le projet ou créer des tâches.

En cliquant sur le nom du projet, vous pourrez accéder aux détails du projet.

Sur la page Project detail vous aurez :

- Project Manager : le projet peut être modifié, delete. Il pourra également accéder au taskflow des tâches et créer des tâches.



- Collaborater : vous pourrez voir le projet et les tâches auquel vous êtes assigné, mais vous n'avez pas accès à tous les boutons que le rôle Manager a. En tant que collaborateur du projet, vous pouvez tout de même accéder au task flow pour modifier le statut de votre tâche

Création de projets

Une interface simple de création de projets. Vous pouvez lui donner un nom, un contenu et l'assigner à un ou plusieurs utilisateurs.

The screenshot shows a web application titled 'HELB MANAGER' with a navigation bar containing 'Home' and 'About'. The main content area is titled 'Project' and contains a form with the following fields: 'Title*' (a text input), 'Content*' (a large text area), and 'Resource*' (a dropdown menu). The dropdown menu is open, showing two options: 'Paul (Manager)' and 'Bruce'. At the bottom left of the form is a 'Done' button.

Update project

The screenshot shows the 'Update project' form in the 'HELB MANAGER' application. The form is titled 'Project' and contains the following fields: 'Title*' (a text input with the value 'Project of Josh Updated'), 'Content*' (a large text area with the value 'Project of Josh content'), 'Status' (a dropdown menu with a value of '-----'), 'Start date' (a text input), and 'End date' (a text input). At the bottom left of the form is a 'Done' button.

Vous avez en cliquant sur le bouton (en tant que ProjectManager) la possibilité de pouvoir mettre à jour le projet (le titre, le contenu).

Création d'une tâche

HELB MANAGER

Home About

Task

Name*

Nom

Content*

Contenu

Status*

Not started

Assign*

Create

Vous pouvez créer une tâche, lui donner un nom, un contenu, un statut de base et pouvoir l'assigner qu'à une seule personne. La tâche, se trouvant dans les détails d'un projet, est automatiquement assigné aux projets dans lequel vous le créer. Vous pourrez également mettre à jour une tâche, exactement comme un projet.

Les sous-tâches c'est exactement la même chose. Cependant, elles seront assignées à une tâche, donc créer à partir du détail d'une tâche.

J'ai construit ces interfaces selon une certaine structure logique :

Project > Task > Subtask

Le drag and drop, TaskFlow

Vous avez la possibilité de mettre à jour les statuts de vos tâches et sous tâche grâce à une interface drag and drop 100% fonctionnel accessible pour les collaborateur et PM, cependant le collaborateur voit seulement ses tâches :

HELB MANAGER

Home About

Drag and Drop

To do	In progress	Done	In revision
			Task1 (assigned to Bruce)
Task 2 (assigned to Bruce)			
		Subtask1 (Task1, Bruce)	

Task

Subtask

Project history

HEL B MANAGER Home About New project					
Project History					
Element Name	Element Type	Old value	New value	When	User
task 1 updated updated	TASK	To do	In progress	8 janvier 2023	Josh
task 1 updated updated	TASK	In progress	Done	8 janvier 2023	Josh

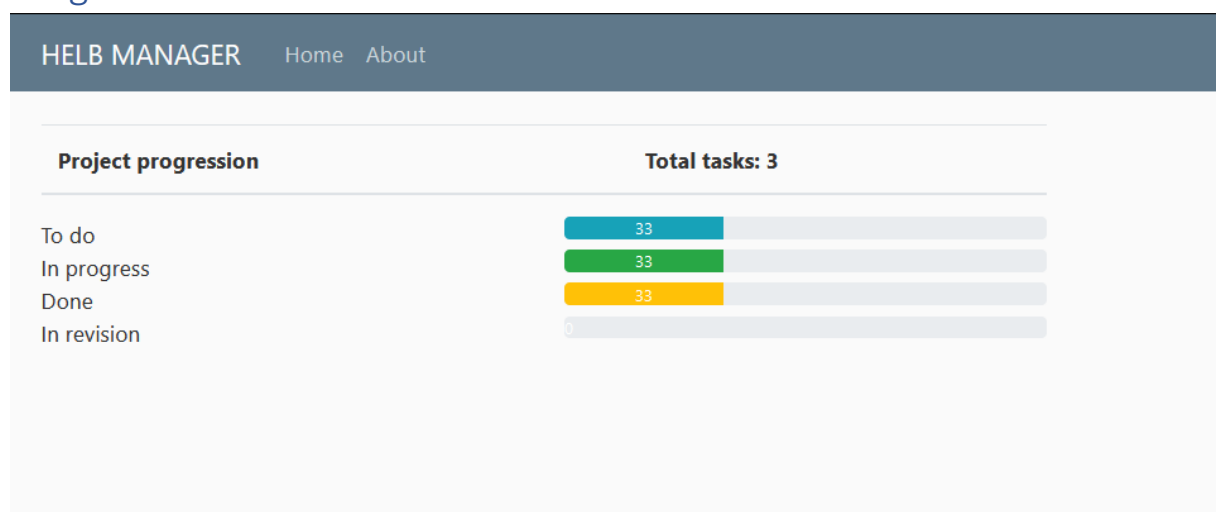
L'application propose une interface historique se basant sur le drag and drop des projets. A chaque fois qu'une personne modifie le statut d'une tâche (ou sous-tâche) via le TaskFlow, l'historique enregistre l'action et le met dans un tableau sur cette page.

Comment est construite l'historique ?

J'ai créé un nouveau modèle (projectHistory) comprenant un nom, un type, un temps (when), un user. Ce modèle sera rempli à chaque fois que mes fonctions, dans views.py, subtaskFlow/taskFlow (fonctions qui envoient les nouveaux statuts à la db, en les récupérant depuis l'url) seront appelé, en même temps un nouveau modèle projectHistory sera créer et rempli avec les nouvelles informations que l'url lui passera.

J'affiche tous les modèles créés sur une page. Seulement le Project manager pourra accéder à cette interface afin d'avoir, en tant que gérant, une vue globale sur son projet.

Progression des tâches



Cette interface affiche le pourcentage des tâches du projet en fonction de leur statut.

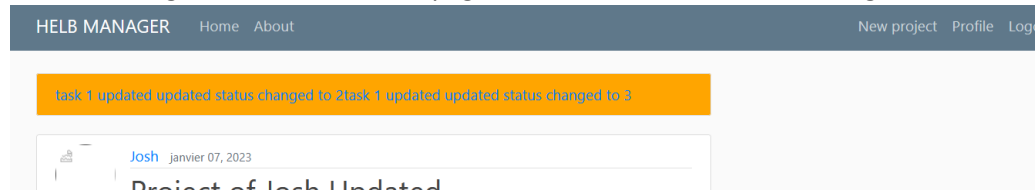
Comment est construite l'interface de progression ?

J'ai une fonction `projectProgress` qui est appelé depuis l'url où je calcule simplement par rapport aux nombres de tâches et statuts les pourcentages à afficher. Je place les résultats dans un contexte avec des variables et je retourne tout ça.

Seulement le Project manager pourra accéder à cette interface afin d'avoir, en tant que gérant, une vue globale sur son projet.

Système de notifications

L'HELB Manager vous affiche sur la page Home une notification de changement de statuts.



Quand un membre assigné à un projet change le statut de l'un de ses projets, une notification apparaît sur la page Home des autres membres (sauf lui-même du coup).

Pour cette fonctionnalité, j'ai créé un nouveau modèle `ProjectResource` qui va faire récupérer l'id d'un projet et l'id d'un User pour faire le lien entre les deux.

Ensuite, j'ai également un modèle `ProjectNotification` qui va à chaque drop (au même endroit ou j'enregistre des données dans le modèle historique) dans le `TaskFlow` enregistrer les anciennes/nouvelles actions, la date et la personne qui a fait la notification et la personne chez qui la notification doit apparaître.

Voici ce que j'ai comme tables dans ma DB :

SQL ▼

1 / 1

1 - 8

id	projectId	descriptoin	when	read	author_id	notify_id
1	2	task 1 updated updated status changed to 2	2023-01-08	0	4	3
2	2	task 1 updated updated status changed to 2	2023-01-08	0	4	4

Difficultés rencontrées

- Assignment des rôles lors de la création
- Assignment des tâches aux user
- Affichage des tâches sur la page détail du projet.
- Assignment de plusieurs personnes à un seul projet
- Afficher les tâches appartenant à un projet dans projet détail
- Ajax, drag & drop update
- Filtrage des affichages des projets ou tâches.

Assignment des rôles lors de la création, assignment des tâches aux user

L'application vous permet d'assigner le rôle à la création du compte, d'assigner les tâches (et sous-tâches) aux user. Ce sont des points de difficulté que j'ai déjà abordé dans mes itérations précédentes, donc je ne m'y attarderai pas trop.


```
# Create your models here.
class CustomUser(AbstractUser):
    ROLES = (
        (1, 'ProjectManager'),
        (2, 'collaborater'),
        (3, 'Any'),
        (4, 'Any2')
    )
    username = models.CharField(max_length = 50, blank = True, null = True, unique = True)
    email = models.EmailField(('email address'), unique = True)
    image = models.ImageField(default='default.jpg', upload_to='profile_pics')
    role = models.PositiveSmallIntegerField(choices=ROLES, blank=True, null=True)

    def __str__(self):
        return "{}".format(self.username)
```

J'ai re crée un User moi-même avec comme propriété importante, le rôle qui peut soit prendre ProjectManager ou collaborater.

```
class Task(models.Model):
    STATUS = (
        (1, 'Not started'),
        (2, 'In progress'),
        (3, 'Done'),
    )
    project = models.ForeignKey(Project, on_delete=models.CASCADE)
    assign = models.ManyToManyField(settings.AUTH_USER_MODEL)
    name = models.CharField(max_length=50)
    content = models.TextField()
    status = models.PositiveSmallIntegerField(choices=STATUS, blank=True, null=True)

    def __str__(self):
        return self.name

    def get_absolute_url(self):
        return reverse('project-detail', kwargs={'pk':self.pk})
```

Ayant modifié mon User de base, j'ai dû (pour l'assignation des tâches), appelé mes settings :

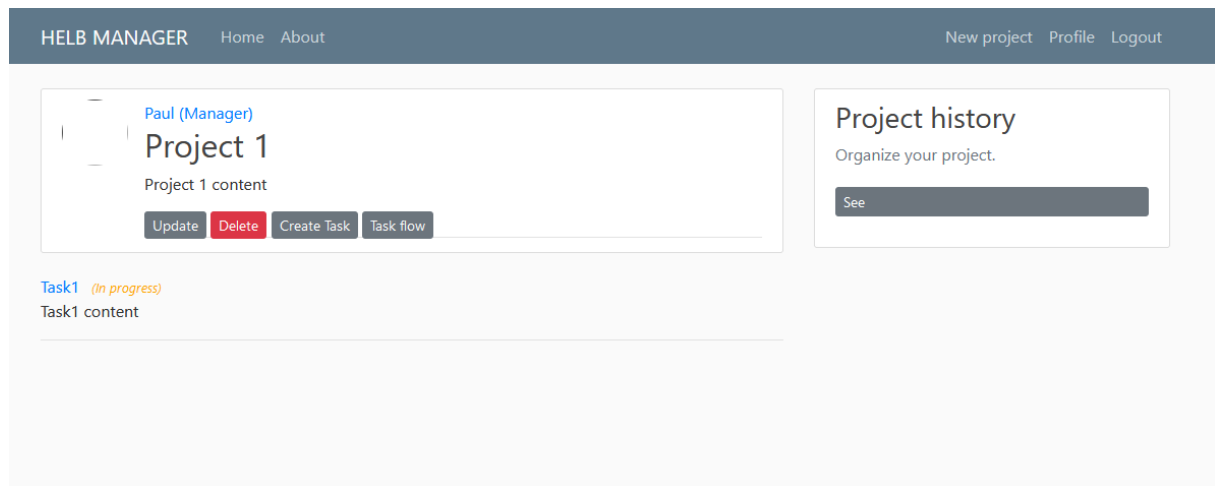
```
AUTH_USER_MODEL = "users.CustomUser"
```

Affichage des tâches sur la page détail du projet

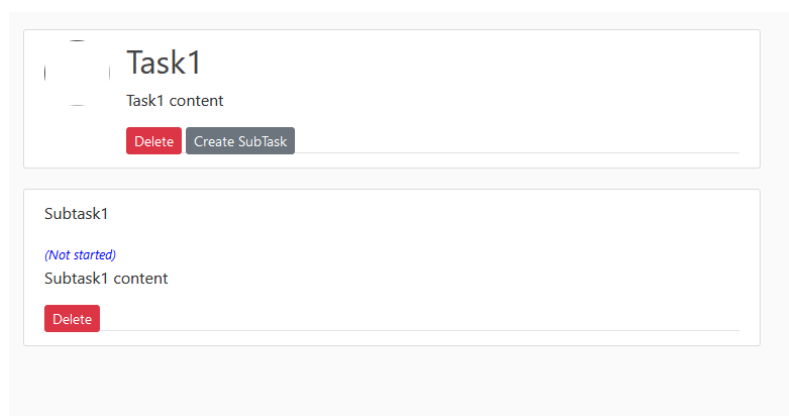
J'ai trouvé plus intéressant le fait d'afficher les tâches du projet dans les détails du projet et non sur la page Home pour une question d'ergonomie. Suite à la base fonctionnelle de Correy MS, le views qui permet d'afficher les détails d'un projet n'a pas de context (Context ? Cela me permet de créer un context dans lequel je mets des tâches ou projets pour ensuite les retourner, les afficher). J'ai trouvé sur Google la possibilité d'ajouter des « extra context ».

```
class ProjectDetailView(DetailView):
    model = Project
    def get_context_data(self, **kwargs):
        context = super(ProjectDetailView, self).get_context_data(**kwargs)
        context['tasks'] = Task.objects.filter(project_id=self.kwargs.get('pk'))
        return context
```

En fait, même si ici on ne le voit pas, Django créer automatiquement des contexts même quand on utilise des éléments déjà existants de la librairie Django (comme ici, DetailView). Il suffit dans ce cas d'ajouter un nouveau context et filter dans toutes les tâches en fonction de l'Id du projet : afficher seulement les tâches du projet.



Il s'agit exactement du même principe pour le détail d'une tâche :



Assignment de plusieurs personnes à un seul projet

C'est un des seuls points de mon projet qui ne fonctionne pas à 100%. L'application doit pouvoir assigner (ajouter) plusieurs personnes à un seul projet.

```
# the project in the db with title, ...
class Project(models.Model) :
    STATUS = (
        (1, 'Not started'),
        (2, 'In progress'),
        (3, 'Done'),
    )
    title = models.CharField(max_length = 25)
    content = models.TextField()
    resource = models.ManyToManyField(settings.AUTH_USER_MODEL)
    date_posted = models.DateTimeField(default=timezone.now) #For the time now
    start_date = models.DateField(default=None, blank=True, null=True)
    end_date = models.DateField(default=None, blank=True, null=True)
    status = models.PositiveSmallIntegerField(choices=STATUS, blank=True, null=True)
    author = models.ForeignKey('users.CustomUser', on_delete=models.CASCADE, related_name = "user_name")
    #assignCollaborator = ManyToManyField(settings.AUTH_USER_MODEL, related_name = 'user_name')
```

J'ai ajouté dans le modèle Project, une ligne « resource » qui contient l'Id des User assigner au projet.

HELB MANAGER

Home About

Project

Title*

Content*

Resource*

Paul (Manager)

Bruce

Done

Tout cela fonctionne, on peut assigner plusieurs personnes à un seul projet et on le voit dans la DB :

SQLite

×

SQL ▼

id	project_id	customuser_id
1	1	1
2	2	1
3	2	2

La fonctionnalité marche bien mais le problème est le suivant : J'aimerais, pour une question de logique, afficher lors de la création d'une tâche seulement les User qui font partie du projet dans lequel on crée la tâche. C'est pour cela, cette fonctionnalité ne fonctionne pas à 100% et je n'ai malheureusement pas trouvé de solution.

Ajax, drag & drop update*

Le drag & drop n'était pas facile à faire. Le challenge ici, c'était de pouvoir envoyer à la data base le changement du statut d'une tâche lorsque qu'on fait un drop. J'ai dû faire quelques recherches sur Google. J'ai trouvé un moyen de faire des appels à la data base depuis l'HTML grâce à Ajax.

Avant de vous expliquer comment mon implémentation Ajax fonctionne, je vous montre rapidement mes fonctions drag et drop.

```

function drag(ev) {
  console.log(ev.target.id)
  ev.dataTransfer.setData("taskId", ev.target.id);
}

function drag2(ev) {
  console.log(ev.target.id)
  ev.dataTransfer.setData("subtaskId", ev.target.id);
}

function drop(ev) {
  ev.preventDefault();
  var taskId = ev.dataTransfer.getData("taskId");
  var subtaskId = ev.dataTransfer.getData("subtaskId");
  var taskStatus = ev.target.attributes["name"].value;
  var url;
  if(taskId)
  {
    ev.target.appendChild(document.getElementById(taskId));
    url = "http://localhost:8000/update-task-status/" + taskId + "/status/";
  }
  if(subtaskId)
  {
    ev.target.appendChild(document.getElementById(subtaskId));
    url = "http://localhost:8000/update-subtask-status/" + subtaskId + "/status/";
  }
}

```

Par rapport à l'élément qu'on glisse, le drag utilisé change :

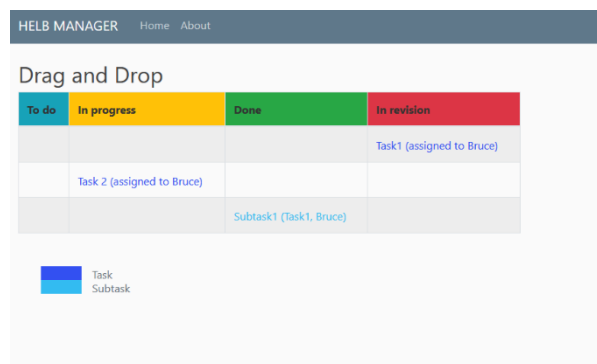
- `drag(ev) =>` Une tâche, on lui passe l'Id de la tâche qu'on glisse.
- `drag2(ev) =>` Une sous tâche, on lui passe l'Id de la sous tâche qu'on glisse.

Tout se passe dans la fonction Drop, où par rapport aux types de l'élément (task ou subtask) nous allons mettre une url adaptée (la façon d'écrire l'url de cette façon vient des sites que j'ai mis en Astérix sur la page précédente). Il nous reste ensuite, à insérer l'Id du statut de la tâche ou de la sous tâche.

```
if(taskStatus == 'inprogress'){
  console.log('inprogress');
  $.ajax({
    url: url + 2,
    //data : { request_data: request_data},
    success : function(json) {
      $("#request-access").hide();
      console.log("requested access complete");
    }
  })
}
if(taskStatus == 'done'){
  console.log('done');
  $.ajax({
    url: url + 3,
    //data : { request_data: request_data},
    success : function(json) {
      $("#request-access").hide();
      console.log("requested access complete");
    }
  })
}
if(taskStatus == 'inrevision'){
  console.log('inrevision');
  $.ajax({
    url: url + 4,
    //data : { request_data: request_data},
    success : function(json) {
      $("#request-access").hide();
      console.log("requested access complete");
    }
  })
}
```

Il s'agit sur ce screen, du code Ajax, on vérifie avec le taskStatus (défini en haut, en lui passant le nom (« Name ») du statut) quel statut l'élément possède par rapport à ça, nous passons l'Id (car c'est un nombre) du statut concerné à l'url afin qu'il puisse envoyer le nouveau statusId à la data base.

Le développement de ce Drag & Drop n'a pas été assez simple, ça m'a pris beaucoup de temps.



Filtrage de l'affichage des projets ou tâches

J'ai dû pour plusieurs pages de l'application filtrer l'affichage de mes objets :

- Home, filtrer les tâches et les afficher en dessous du projet auquel elles sont assignées :

```
'tasks' : Task.objects.filter(project_id__in=(list(Project.objects.filter(author=request.user))))],
```

- TaskFlow, afficher seulement les sous-tâches des tâches appartenant aux projets :

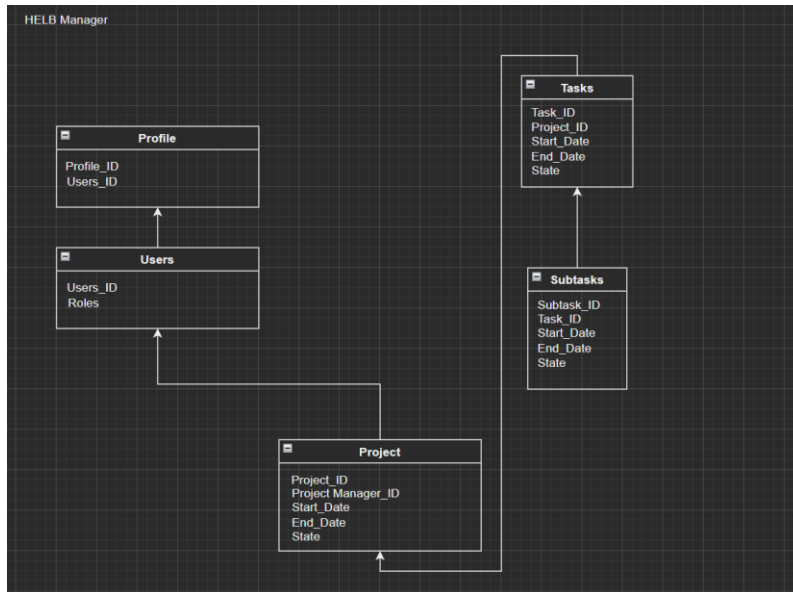
```
'subtasks' : SubTask.objects.filter(task_id__in=(list(Task.objects.filter(project_id = projectId))))
```

Il peut s'agir de lignes de code assez simple mais ça m'a tout de même pris pas mal de temps à chercher et trouver sur Google des exemples fonctionnels.

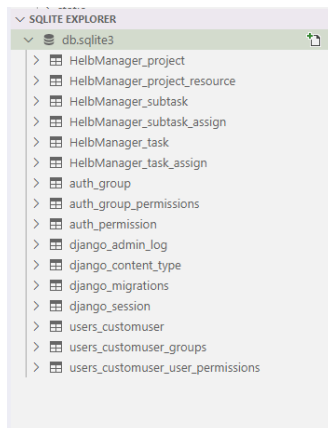
Limitations de mon application ?

- Les collaborateurs ne peuvent pas de créer de projet : Mon code est fait sur une base structure spécifique : mon home par exemple est affiché par rapport aux rôles. Le souci est que j'assigne les rôles des User à l'inscription donc automatique un collaborateur ne pourra pas créer de projet. Le mieux ça aurait été de pouvoir assigner durant la création du projet, qui est Manager et qui Collaborateur.
- Afficher dans création de tâche ou sous tâche seulement les personnes assignées aux projets, j'ai voulu n'afficher durant la création d'une tâche ou sous tâche seulement les personnes inscrites sur le projet mais je n'ai pas réussi à le faire.

Analyse



Voici le schéma d'analyse de base. Il n'a pas totalement respecté, par exemple les start_date et end_date n'ont pas utilisés. J'ai eu d'autres idées au fur et à mesure du développement de l'application, c'est pourquoi le schéma n'a pas été totalement respecté.



Voici comment est construite ma DB. Cette section SQLITE EXPLORER de VS Code m'a beaucoup aidé. Notamment pour pouvoir voir que les tâches étaient assignées aux bons projets, etc.

Conclusion

Pour conclure le rapport, j'ai acquis beaucoup de connaissances (et skills) durant le développement de cette application, loin d'être simple.

Django est un grand jeu de lego. Les structures de construction se ressemblent : passer les id via l'url, appelé une fonction, dire ce qu'elle doit retourner, etc.

C'était un développement très laborieux mais enrichissant non seulement sur un niveau connaissance du langage python mais également sur un niveau de logique, les fonctionnalités faites demandent une certaine logique à avoir.

Mon projet respecte parfaitement le scénario demandé dans l'énoncé, il reste cependant des petites subtilités que j'expliquerai durant la présentation.

Je n'ai pas voulu mettre trop de screen de code pour éviter de trop charger le rapport.

Bibliographie

- <https://docs.djangoproject.com/en/4.1/topics/db/queries/>
- https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Generic_views
- <https://stackoverflow.com/questions/21758731/how-can-i-get-pk-or-id-in-get-context-data-from-cbv>
- <https://stackoverflow.com/questions/31878960/calling-django-view-from-ajax>
- <https://stackoverflow.com/questions/46345388/ajax-is-not-a-function-in-django>
- <https://sharepoint.stackexchange.com/questions/285269/ajax-post-call-to-more-than-one-sharepoint-list>
- <https://www.javascripttutorial.net/web-apis/javascript-drag-and-drop/#:~:text=Introduction%20to%20JavaScript%20Drag%20and%20Drop%20API&text=To%20drag%20an%20image%2C%20you,you%20would%20drag%20an%20image.>
- <https://getbootstrap.com/docs/4.0/content/tables/>
- <https://stackoverflow.com/questions/31878960/calling-django-view-from-ajax>
- <https://stackoverflow.com/questions/46345388/ajax-is-not-a-function-in-django>