Simulation on the Web

IMA5 2018/2019 P05

Rodolphe Toin

Encadrants :
Damien Marchal
Jérémie Dequidt

Sommaire

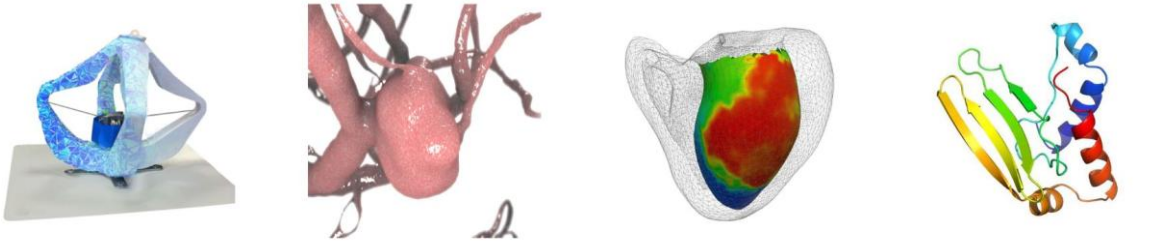
Introduction

Dans le cadre du développement de leur logiciel SOFA, l'équipe Defrost de l'Inria de Lille souhaite offrir aux potentiels utilisateurs un moyen d'interagir avec le logiciel directement depuis le Web. L'objectif est de permettre à chacun d'aborder rapidement et facilement le logiciel sans avoir à compiler lui-même tout le code sur sa machine personnelle.

Mon travail consistera à implémenter l'interface Web en question en m'appuyant au besoin sur des travaux déjà effectués auparavant par des stagiaires. Une première étape consistera à créer un front end Web faisant tourner une base de donnée et communicant avec des serveurs de l'Inria faisant tourner une version compilée de SOFA. Une seconde étape consistera à évaluer la viabilité de Qt et WebGL pour nous permettre d'embarquer une version simplifiée de SOFA directement dans un navigateur web afin de s'affranchir de l'impératif de faire tourner des instances de SOFA sur des serveurs de l'Inria.

Contexte

SOFA est un logiciel open-source de simulation physique premièrement ébauché en 2004. SOFA est utilisé pour la simulation médicale et la robotique : le logiciel offre des modélisations de mécanique des solides, de dynamique des fluides ou encore de thermodynamique, le tout sur des modèles interactifs en temps réel.



Soft robot control | Endovascular simulation | Cardiac electrophysiology | Protein structure prediction

Aujourd'hui SOFA est constitué de plus de 500 millions de lignes de code ainsi que de quelques millions de lignes supplémentaires pour ses plugins. La prise en main du logiciel étant difficile pour une personne non initiée au C++ ou aux règles de physique utilisées, le logiciel propose un binding Python permettant aux utilisateurs de fournir des scripts ou « scènes » qui seront ensuite modélisés.

Compte-tenu du volume actuel du logiciel la compilation se révèle très longue, tournant autour de 4-5h sur une machine moyenne. Ajouté aux contraintes d'environnement pour la compilation, de téléchargement de code, etc, la découverte de SOFA peut s'avérer laborieuse pour de nouveaux utilisateurs.

L'une des volontés actuelles de l'équipe Defrost travaillant sur SOFA est de favoriser la démocratisation du logiciel en le rendant accessible depuis un navigateur internet. Grâce à une interface front-end web, l'objectif est de permettre à un utilisateur d'éditer du code, de sauvegarder ses simulations en ligne et d'obtenir un retour visuel temps-réel des simulations.

Cahier des charges

Après de premiers entretiens avec monsieur J. Dequidt et monsieur D. Marchal voici le cahier des charges initial du projet concernant l'interface web à réaliser:

- L'utilisateur doit pouvoir éditer le code de sa scène (script python par exemple) ;
- L'utilisateur doit pouvoir paramétrer son projet comme "privé" ou "publique" et avoir accès aux projets qui sont publiques ;
- L'utilisateur doit pouvoir se connecter à un compte afin de sauvegarder ses projets et y accéder ;
- Dans un premier temps (en rouge sur le schéma ci-dessous), l'objectif est de fournir un retour en format vidéo (sachant qu'il existe déjà un moyen de générer une vidéo avec SOFA) ;
- Dans un second temps (en bleu sur le schéma ci-dessous), l'objectif est d'avoir une version embarquée de SOFA dans le navigateur permettant un rendu 3D en temps réel avec lequel on puisse interagir ;

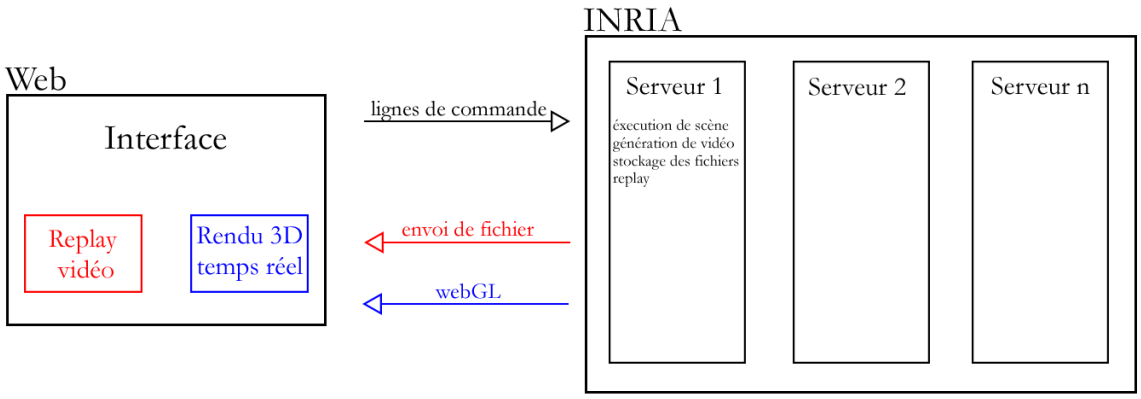


Figure 1: Schématisation de l'interface souhaitée

Afin de réaliser la dite interface nous utiliserons :

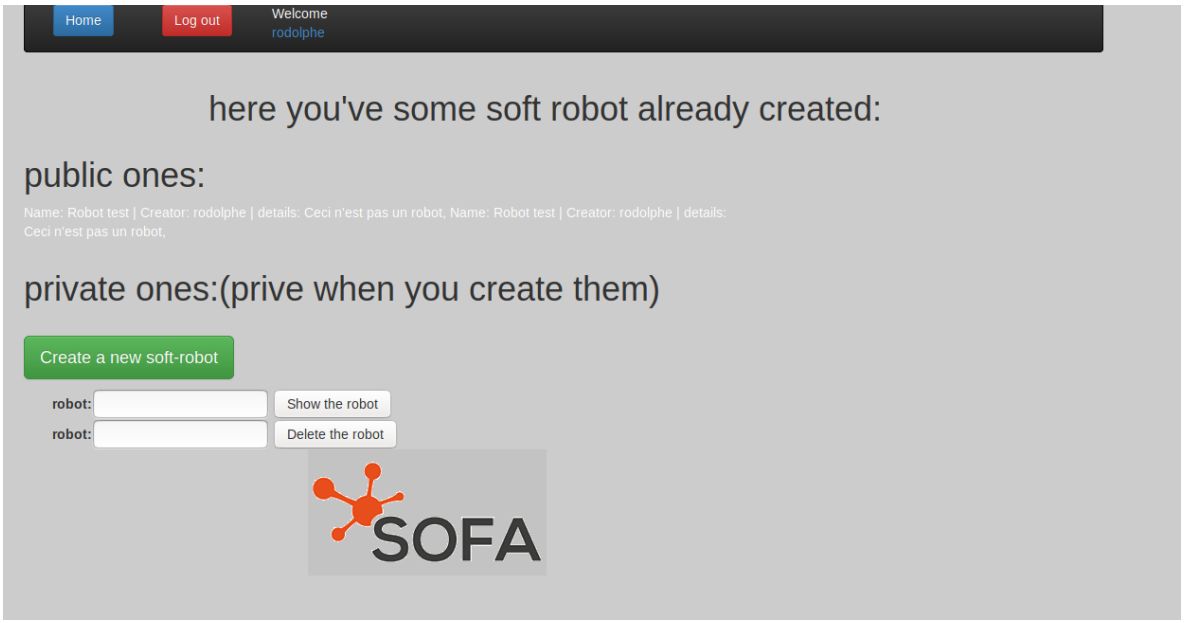
- Python et HTML pour coder l'interface
- Sqlite pour le format de la base de données
- Django pour la gestion de la base de données
- WebGL pour permettre un rendu en temps réel

Travail effectué

Rapport sur les projets existants

La première étape de mon travail a été de fournir un rapport sur les projets déjà développés auparavant par deux stagiaires distincts. Les codes sont hébergés en privé sur le GitLab de l'Inria.

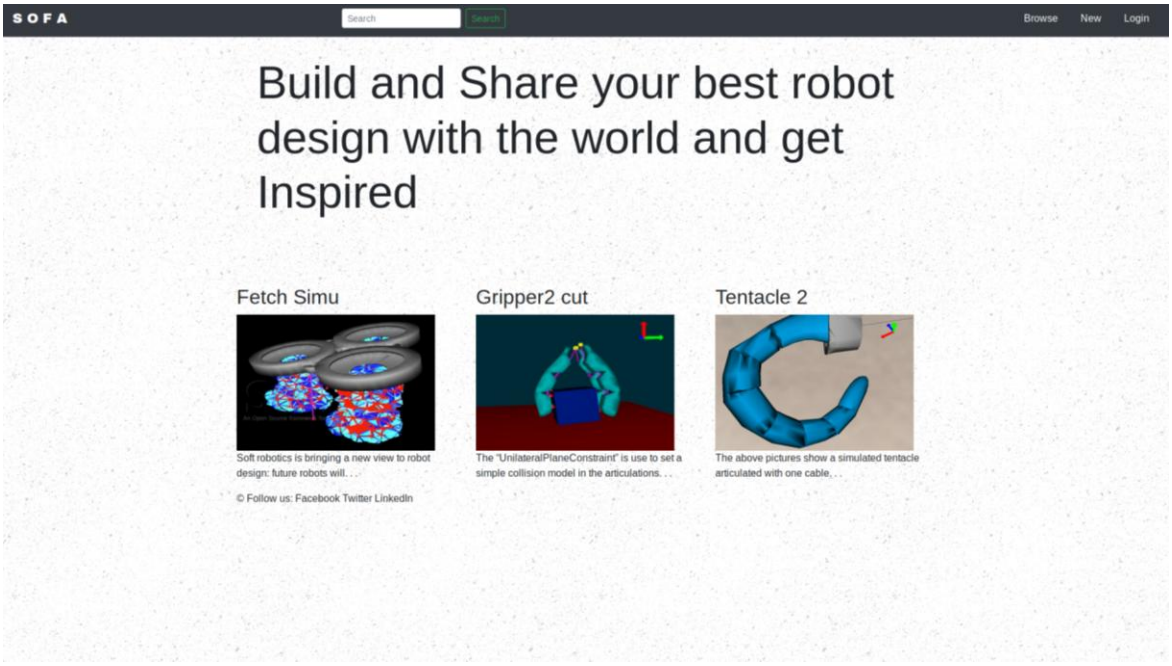
Le premier projet effectué se présente ainsi :



Ce projet a été jugé inutilisable pour les raisons suivantes :

- La saisie de scène n'est pas le modèle qui nous intéresse (saisie de points sur une grille pour créer un maillage) et s'appuie sur un travail en développement d'un chercheur de l'équipe DEFROST
- L'arborescence du projet n'est pas satisfaisante : toutes les fonctions sont séparées en petites applications (login, logout, gestion des scènes, gestion des utilisateurs, etc) ce qui serait bien pour créer un modèle très réutilisable or nous souhaitons avoir un modèle simple, pouvant être rapidement déployé et facilement maintenu.
- La base de données n'est pas structurée et ne présente aucun modèle
- L'interface graphique n'est pas satisfaisante : elle est fonctionnelle dans le peu qu'elle fait mais n'est pas agréable à utiliser
- La rédaction du code existant n'est pas satisfaisante : les noms de variables sont inconsistants, le français côtoie l'anglais avec des fautes d'orthographes, etc. De manière générale, la rédaction du code effectuée entrave une potentielle facilité de maintenance.

Le second projet effectué a l'allure suivante :



Ce projet nous intéresse davantage pour les raisons suivantes :

- Le modèle de la base de donnée est similaire a celui que nous souhaitons réaliser ;
- L'interface graphique est satisfaisante, propre et intuitive en plus d'être fonctionnelle dans ce qu'elle fait (barre de navigation notamment);
- Les pages web disponibles sont proches des pages principales recherchées ;
- La rédaction du code est de bien meilleure que pour le projet précédent

Néanmoins, le projet date un peu et utilise quelques méthodes obsolètes de Django, le modèle de la base de donnée n'est pas celui que nous souhaitons vraiment, la saisie de scène est toujours impossible, la navigation parmi les projets est basique (simple affichage d'une image par projet) et l'application contient pas mal de code optimisable. Pour toute ces raisons, il a été décidé de repartir à zéro afin d'avoir des bases solides. Néanmoins, ce projet avait pris beaucoup de bonnes directions et restera une inspiration lors de la rédaction de mon propre code.

Définition du modèle de la base de donnée

Afin de se fixer sur ce que nous souhaitions obtenir, j'ai dans un premier temps défini le modèle de la base de données que nous souhaitons utiliser (cf schéma UML ci-dessous). Nous souhaitons uniquement gérer des utilisateurs et des projets SOFA. Les champs présentés sont les champs estimés comme indispensables, d'autres champs pourraient être ajoutés si nécessaire.

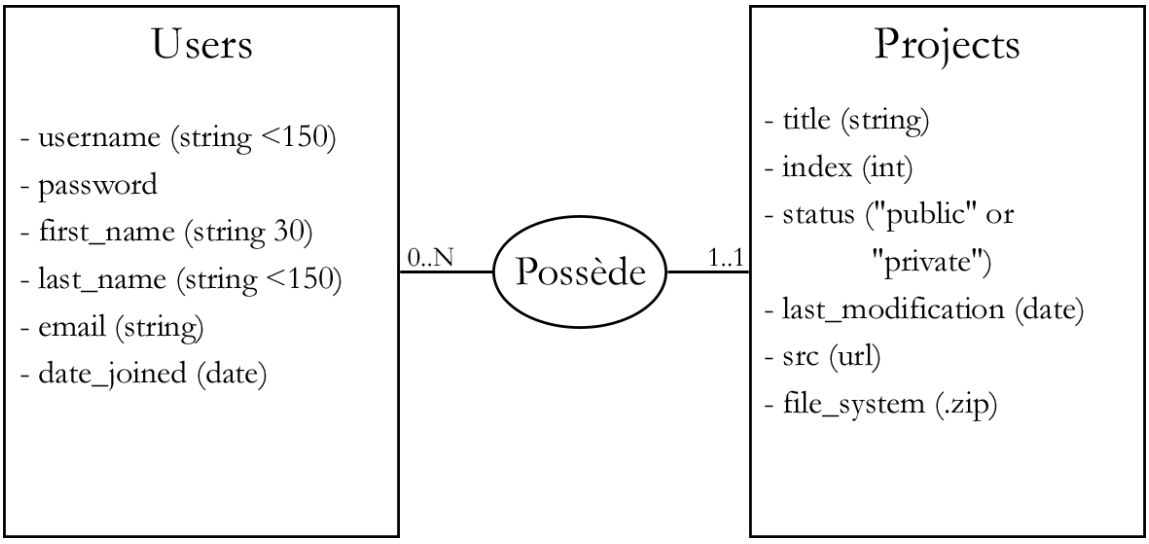


Figure 2: Schéma UML du modèle de la BDD

Le seul champ non explicite est le champ "src" de "Projects" : il s'agit d'un url renvoyant au projet initial dans le cas où le "project" est une copie d'un projet disponible dans notre base de données.

Le fichier ZIP "file_system" regroupera les fichiers suivants :

- index.html
- index.png
- index.mp4
- main.html
- main.py (or .pyscn .psl .scn)
- data/
- scripts/

Il s'agit là du contenu type qu'un utilisateur doit fournir s'il souhaite importer un projet sur le site. Il s'agira aussi de l'architecture téléchargeable par un utilisateur souhaitant travaille sur un projet hébergé dans notre base de données.

Dès lors, cette mise à plat du modèle de notre BDD nous a fait ajouter des éléments au cahier des charges :

- l'utilisateur devra pouvoir uploader un projet déjà existant sur sa machine et devra pouvoir télécharger les projets publiques
- l'utilisateur doit pouvoir créer un nouveau projet à partir d'un projet déjà existant

Définition des pages web à concevoir

Afin d'être cohérent dans les fonctionnalités à implémenter à notre application j'ai défini les pages web indispensables :

- www.sofasite.com/home: page d'accueil avec message d'accueil et aperçu des projets récents ;
- www.sofasite.com/login : page de login et de création de compte ;
- www.sofasite.com/logout : page de logout ;
- www.sofasite.com/browse : page de parcours des projets publique existants. Proposera une fonction de tri alphabétique et chronologique ;
- www.sofasite.com/search/xxxxxx : page de recherche (via recherche par mots de l'utilisateur) ;
- www.sofasite.com/new : page de création d'un nouveau projet. Permet seulement de définir le nom du projet et son statut (publique ou privé) ;
- www.sofasite.com/import : page permettant d'importer directement un projet déjà existant sur la machine de l'utilisateur ;
- www.sofasite.com/project/xxx : page d'un projet permettant de voir toutes les données de celui-ci. L'URL sera idéalement haché pour ne pas pouvoir deviner les adresses d'autres projets ;
- www.sofasite.com/project/xxx/edit :page de modification d'un projet (édition de code) et proposant un retour direct d'exécution (retour vidéo ou 3D + résumé d'exécution retourné par SOFA) ;
- www.sofasite.com/username/projects : page permettant de voir tous les projets pour un utilisateur connecté ;
- www.sofasite.com/username/profile : page permettant à un utilisateur connecté de consulter et modifier son profile ;

Une schématisation de l'interface de chaque page a été réalisée. À titre d'exemple, j'inclus ci-dessous le modèle défini pour la page www.sofasite.com/project/xxx/edit :

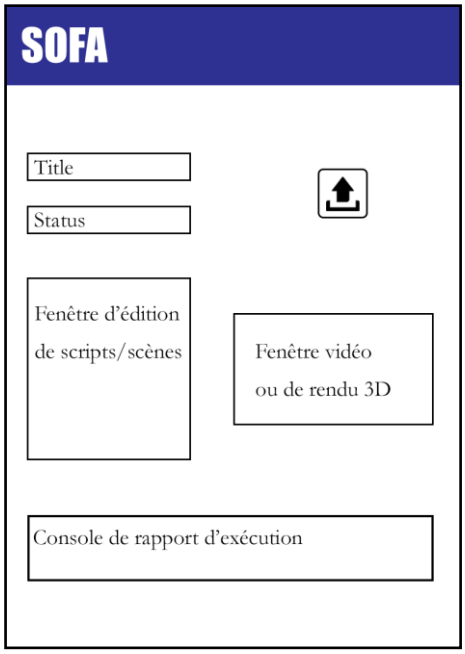


Figure 3: Modèle de la page d'édition d'un projet

Par ailleurs, le site proposera une barre de navigation présente sur toute les pages et que j'ai défini de la sorte :

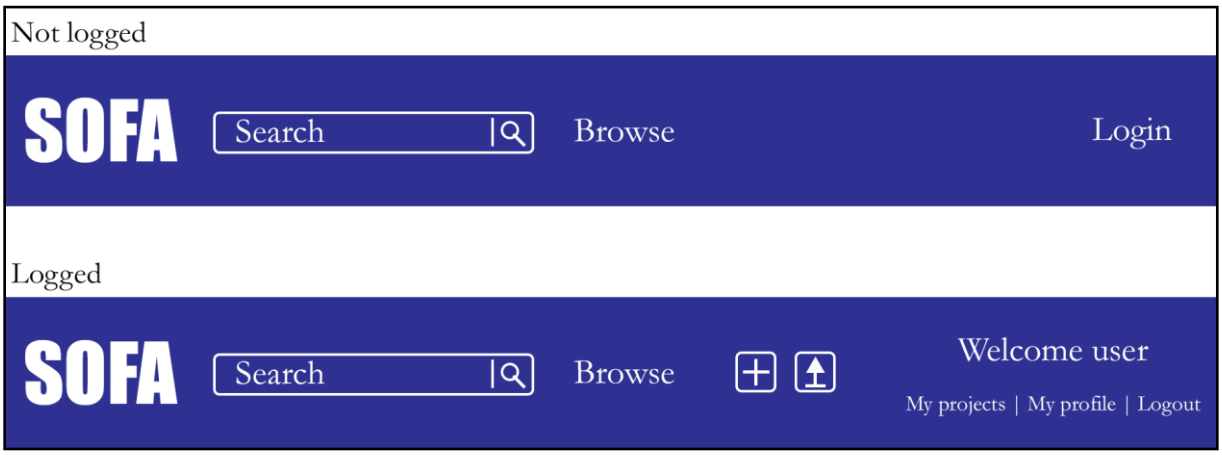


Figure 4: Modèle de la barre de navigation

Codage

Avant tout codage concernant directement le projet j'ai pris le temps de réaliser les tutoriels de la documentation officielle Django. Après être familiarisé avec le fonctionnement de Django je me suis penché sur le second projet déjà existant pour tenter de l'enrichir. C'est après cela que nous avons décidé de repartir sur un projet neuf et donc j'ai commencé par mettre en place le modèle de la base de données.

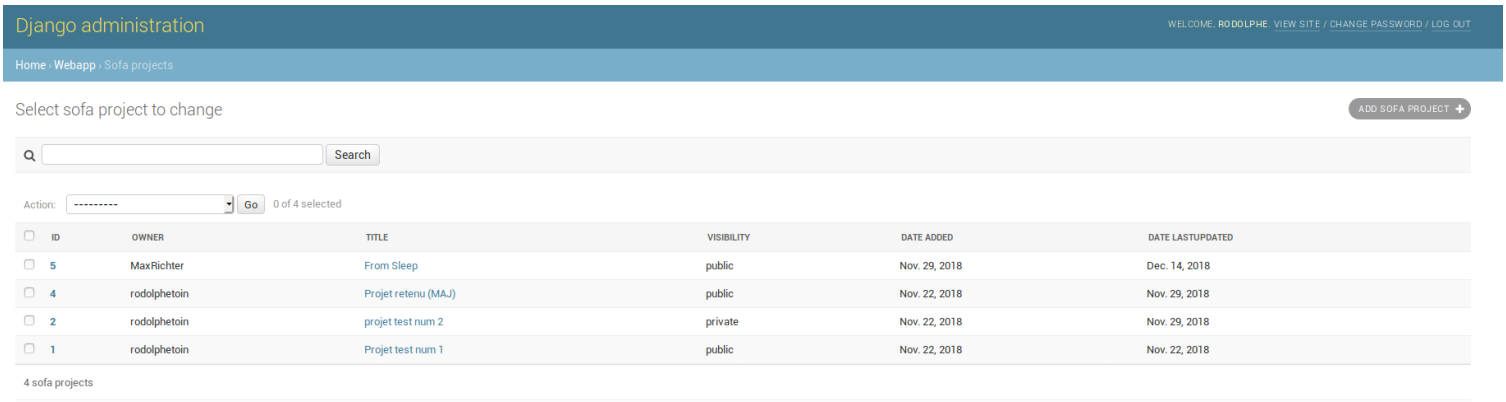
```python
from django.db import models
from django.contrib.auth.models import User

#Reminder of what is useful for us in the default User class of Django
#class User():
#    username        = models.CharField(max_length=150)
#    password
#    first_name      = models.CharField(max_length=30, blank=True)
#    last_name       = models.CharField(max_length=150, blank=True)
#    email           = models.EmailField(blank=True)
#    date_joined     = models.DateField(auto_now_add=True)

class SofaProject(models.Model):
#   id              = incremental integer added by default by django
    owner           = models.ForeignKey(User, on_delete=models.CASCADE , null=True, blank=True)
    title            = models.CharField(max_length=50,default='')
    VISIBILITY = (
        (True,'public'),
        (False,'private'),
    )
    visibility      = models.BooleanField (choices=VISIBILITY)  #if 1 project is public, if 0 project is private
    date_added      = models.DateField(auto_now_add=True)
    date_lastupdated = models.DateField(auto_now=True)
    src             = models.URLField(default='', blank=True) #contains the URL of the original project if the project is a copy
    zip_filesystem  = models.FileField(blank=True) #zip file containing the source files of the project

    def __str__(self):
        return self.title
```

J'ai pu vérifier la conformité du modèle en peuplant ma base de données via l'interface Django. Ci-dessous la vue du mode administrateur :



La barre de navigation a aussi été réalisée en html :

```html
<link href="https://fonts.googleapis.com/css?family=Archivo+Black|Prompt" rel="stylesheet">

<nav class="navbar navbar-expand-lg navbar-dark fixed-top bg-dark ">
    <a class="navbar-brand" href="/" style="font-family: 'Archivo Black', sans-serif; font-size: 22px; color: #FFFFFF">S O F A </a>
    <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSofaSite" aria-controls="navbarSofaSite" aria-expanded="false" aria-label="Toggle navigati
        <span class="navbar-toggler-icon"></span>
    </button>

<!-- Items -->

    <div class="collapse navbar-collapse" id="navbarSofaSite">
<!-- Searching bar -->
        <form class="form-inline my-2 my-lg-0">
            <input class="form-control-sm mr-sm-2" type="search" placeholder="Search" aria-label="Search">
            <button class="btn-sm btn-outline-success my-2 my-sm-0" type="submit">Search</button>
        </form>

<!-- Links at the end -->
        <ul class="navbar-nav mr-auto mt-2 mt-lg-0">
            {% if request.user.is_authenticated %}
                <li class="nav-item">
                    <a class = "nav-link" href = "{% url 'profile' %}" style="font-family: 'Prompt', sans-serif; color: #FFFFFF">
                    Welcome {{ request.user.username }} |</a>
                </li>
            {% endif %}
            <li class = "nav-item">
                <a class = "nav-link" href = "{% url 'browse' %}" style="color: #FFFFFF">Browse</a>
            </li>
            <li class = "nav-item">
                <a class = "nav-link" href = "{% url 'new' %}" style="color: #FFFFFF"> New</a>
            </li>
            {% if request.user.is_authenticated %}
            <li class = "nav-item">
                <a class = "nav-link" href = "{% url 'logout'%}" style="color: #FFFFFF"> Logout</a>
            </li>
            {% else %}
            <li class = "nav-item">
                <a class = "nav-link" href = "{% url 'login' %}" style="color: #FFFFFF"> Login</a>
            </li>
            {% endif %}
        </ul>
    </div>
```

Travail restant

Jusqu'à maintenant la plupart du projet a consisté en de la décortication de l'existant, de la conception et un début de codage. Il m'aura fallut un moment avant de commencer à coder ma propre branche du projet mais maintenant que je suis familiarisé avec Django les progrès sont rapides.

Je prévois de rapidement pouvoir terminer la navigation dans le site et l'entrée de scène dans la base de donnée (2 semaines) puis je pense me donner une semaine supplémentaire pour valider le tout et valider la communication entre l'application et un SOFA tournant sur un serveur de l'Inria. Le reste du temps sera consacré à une partie plus expérimentale et plus proche d'un sujet de recherche à savoir l'utilisation de webGL pour aboutir à une version miniature de SOFA qui puisse tourner sur un navigateur. Une période d'évauation des technologies sera nécessaire avant de tenter de modifier SOFA car l'équipe ne sait pas ce qui est réalisable pour le moment.

Figure 5: Planning prévisionnel

Conclusion

Maintenant que je suis bien lancé sur le projet j'attends avec impatience la période d'après vacances pour pouvoir m'y consacrer à plein temps et aboutir rapidement à un résultat. Je suis confiant quand à la validation d'un modèle permettant un retour vidéo des simulations.

Concernant la partie webGL à venir je suis un peu plus perplexe. Comme indiqué par monsieur Marchal il s'agit d'une partie plus expérimentale où nous ne sommes pas certains de pouvoir créer quelque chose de compatible entre SOFA et nos besoins. Néanmoins, je suis enthousiaste à l'idée de pouvoir me pencher sur des technologies qui me sont inconnues afin de développer une fonctionnalité de A à Z greffée à un projet de plus grande envergure.