RAPPORT DE PROJET 2021/2022

PPRO605

Site de gestion de conventions

LEMASSON CHRISTOPHER

Table des matières

I) Préparation à la modélisation

1) Objectif du projet

Dans le cadre de PPRO605, un projet m'a été décerné, celui de la création d'un site web permettant le suivi de conventions de stages directement en ligne. Le site devait donc, en réponse à cette thématique, contenir différentes choses :

- Moyen de s'inscrire
- Système permettant de différencier les utilisateurs et leurs relations avec la convention (Secrétaire de faculté, responsables de formations, tuteurs d'entreprise)
- Zone de dépôt d'une convention / Retrait d'une convention

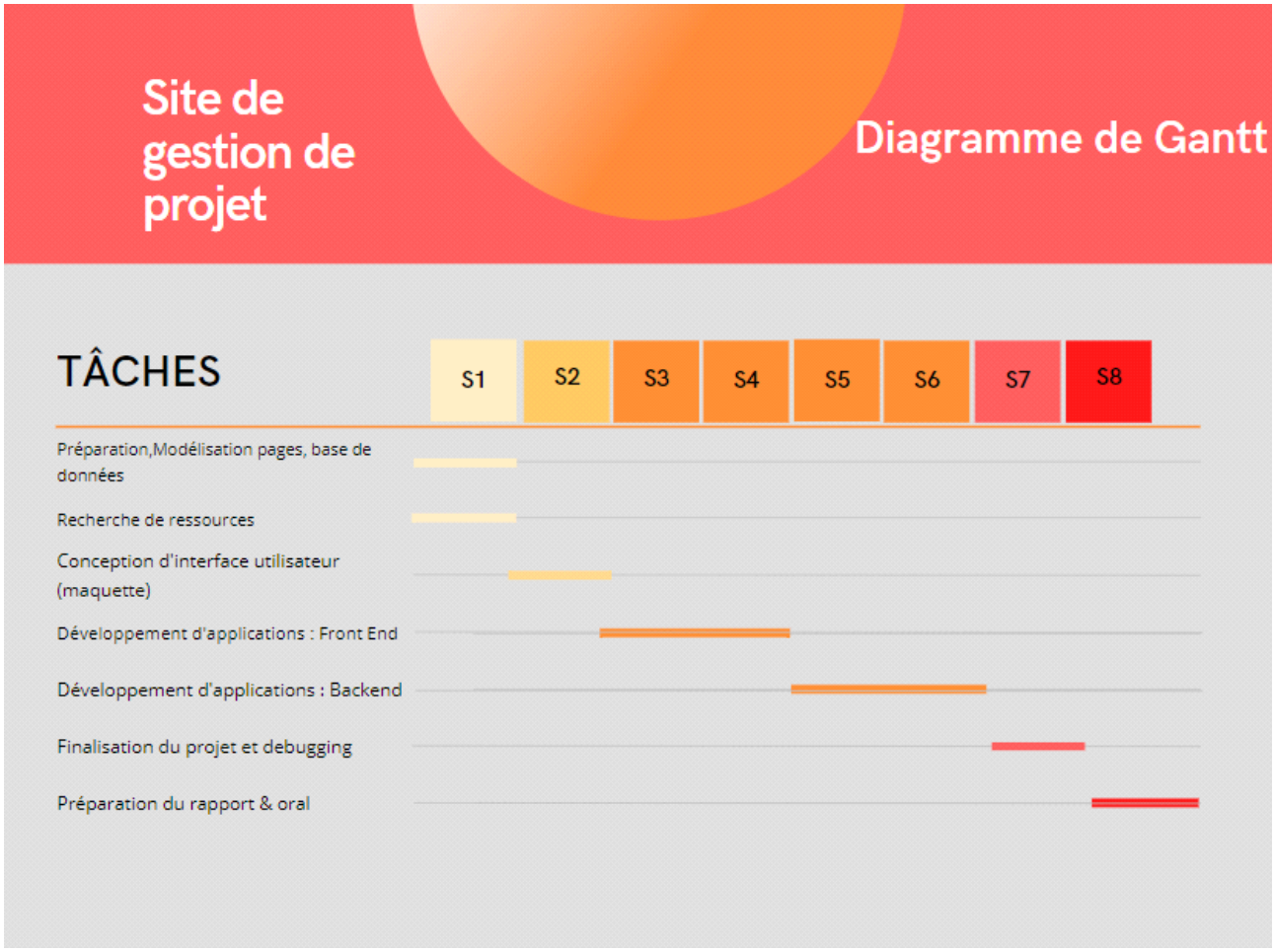
En plus de ces différentes choses, nous avons fixer quelques fonctionnalités utiles pour le secrétariat, tel que l'import de compte élèves afin de pouvoir inscrire directement les étudiants d'une formation à partir d'un format CSV, ainsi qu'un moyen pour une faculté de construire une procédure grâce à différentes étapes clefs, et ainsi, pouvoir apporter un choix de suivi de convention par une procédure en fonction de la formation, ou de si le stage s'effectuera dans le territoire français ou non.

2) Support utilisés

Afin de rendre interactif le site , il a donc fallu au projet une base de donnée, pour pouvoir la gérer, phpmyadmin a été utilisé, afin de modéliser celle-ci, j'ai utilisé Looping, qui permet maintenant un transfort de MCD à MLD afin de se rapprocher de laravel et de son système de gestion de base de données .

3) Répartition du travail au fil du temps.

Dans un objectif de répartir la totalité du travail sur la durée du projet, j'ai construit un



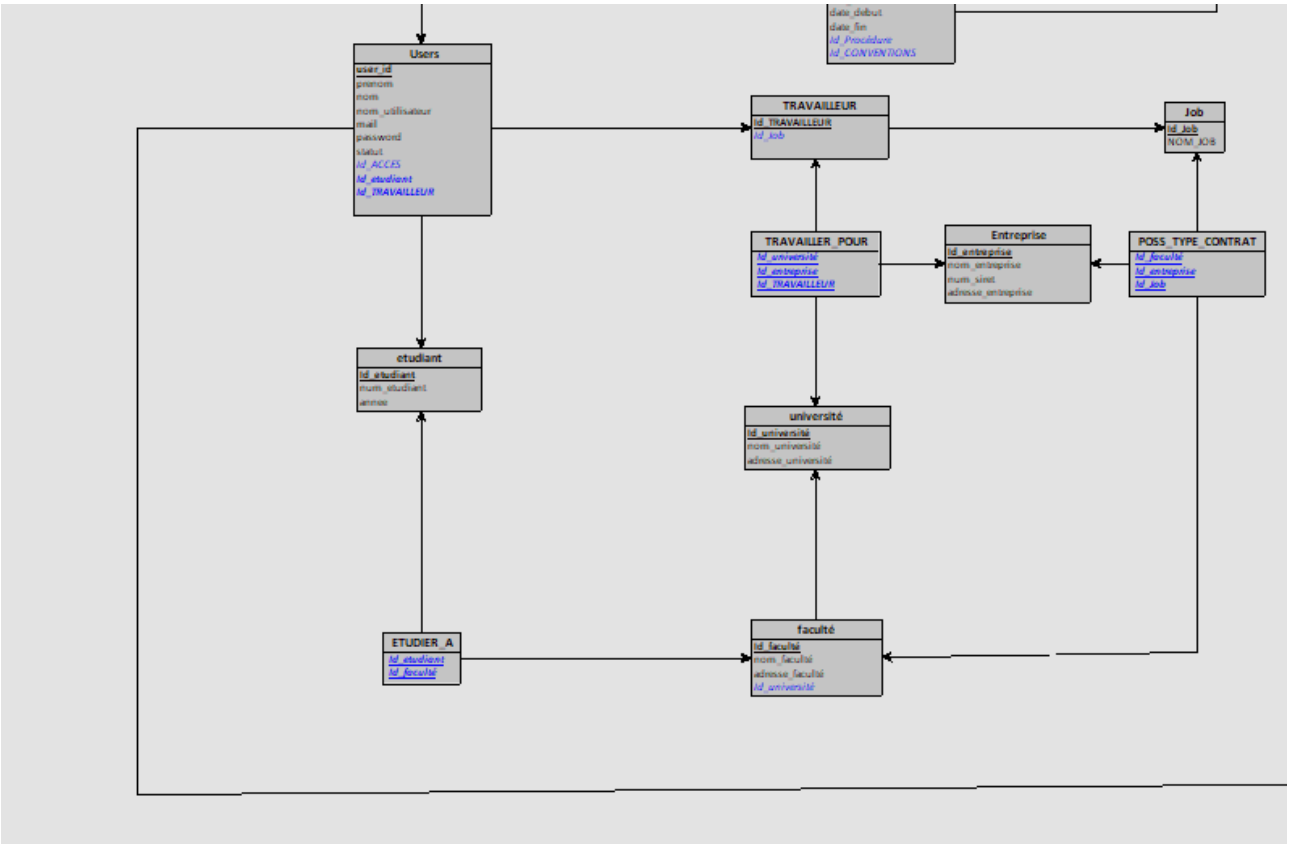
premier diagramme de Gantt, permettant de me fixer des objectifs par semaine,

Finalement, lorsque nous regardons les différents rapports quotidiens, on remarque que le temps consacré au développement front-end a été plus cours, et le développement du back-end plus long, ça vient d'une grande raison : L'imprévu, effectivement, certaines pages ont été introduites et n'avaient pas été pensé à l'origine, j'ai donc du faire certains front-end en même temps que le back-end. Sinon, globalement, en dehors de la base de données qui a pris une forme finale vers la cinquième semaine, j'ai réussi à maintenir les grandes lignes directrices de ce projet.

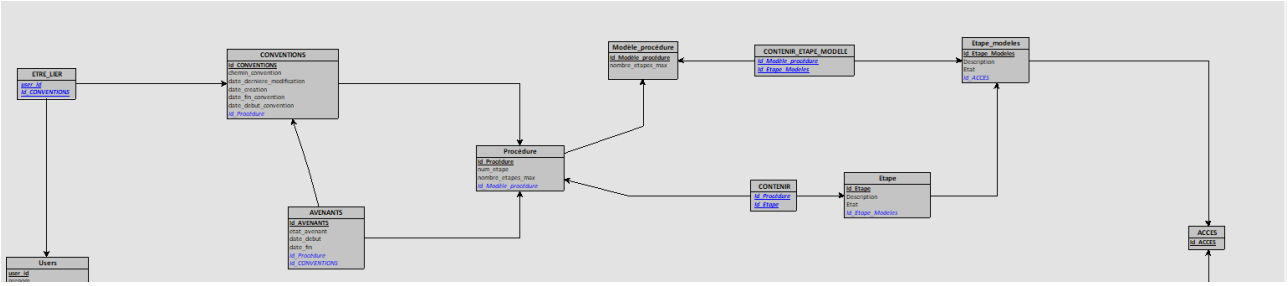
II)Modélisation

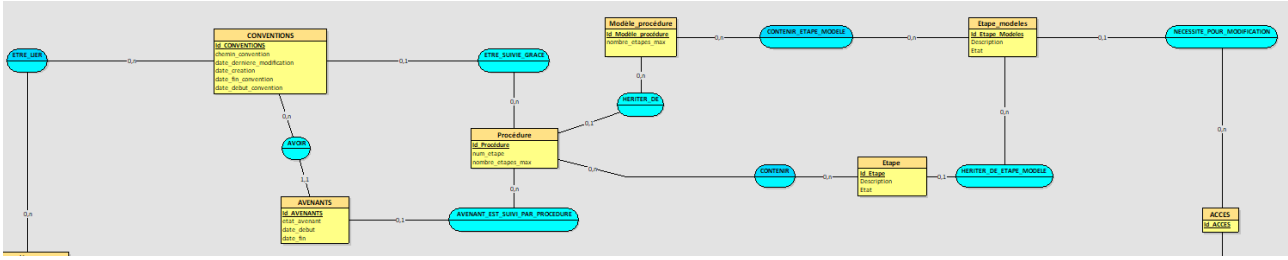
Concernant la modélisation, étant donné qu'il s'agit d'un projet web, elle est surtout basée sur le back-end, notamment le base de données, qui a été dans un premier temps imaginé à l'aide d'un MCD, puis d'un MLD .

Le MCD peut être divisé en deux grandes parties : Les relations d'utilisateurs,et les relations de conventions



Partie utilisateurs



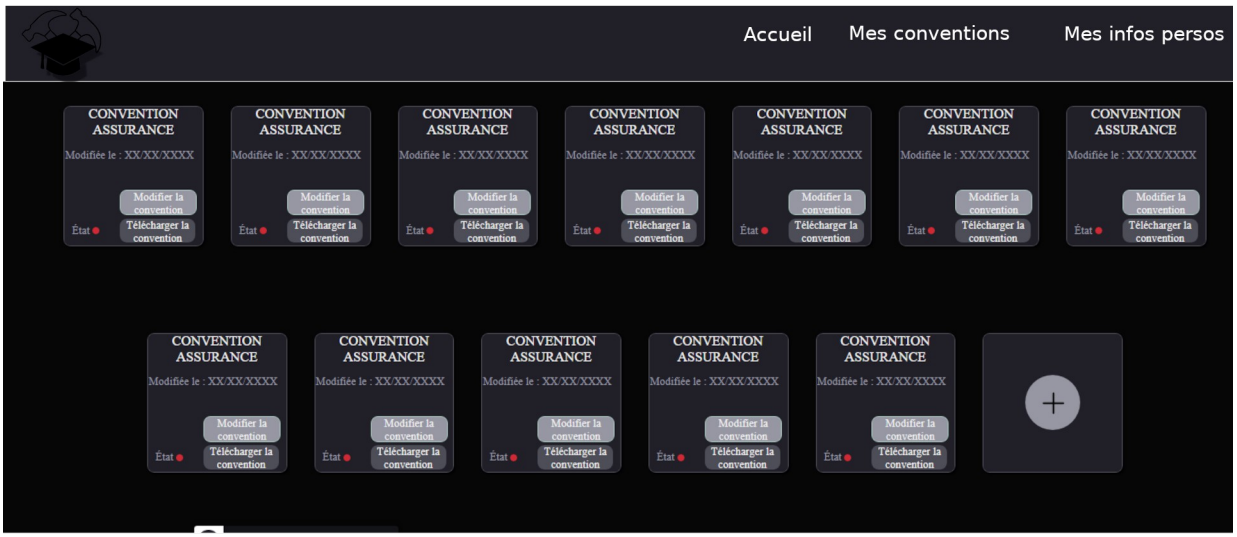


Partie conventions

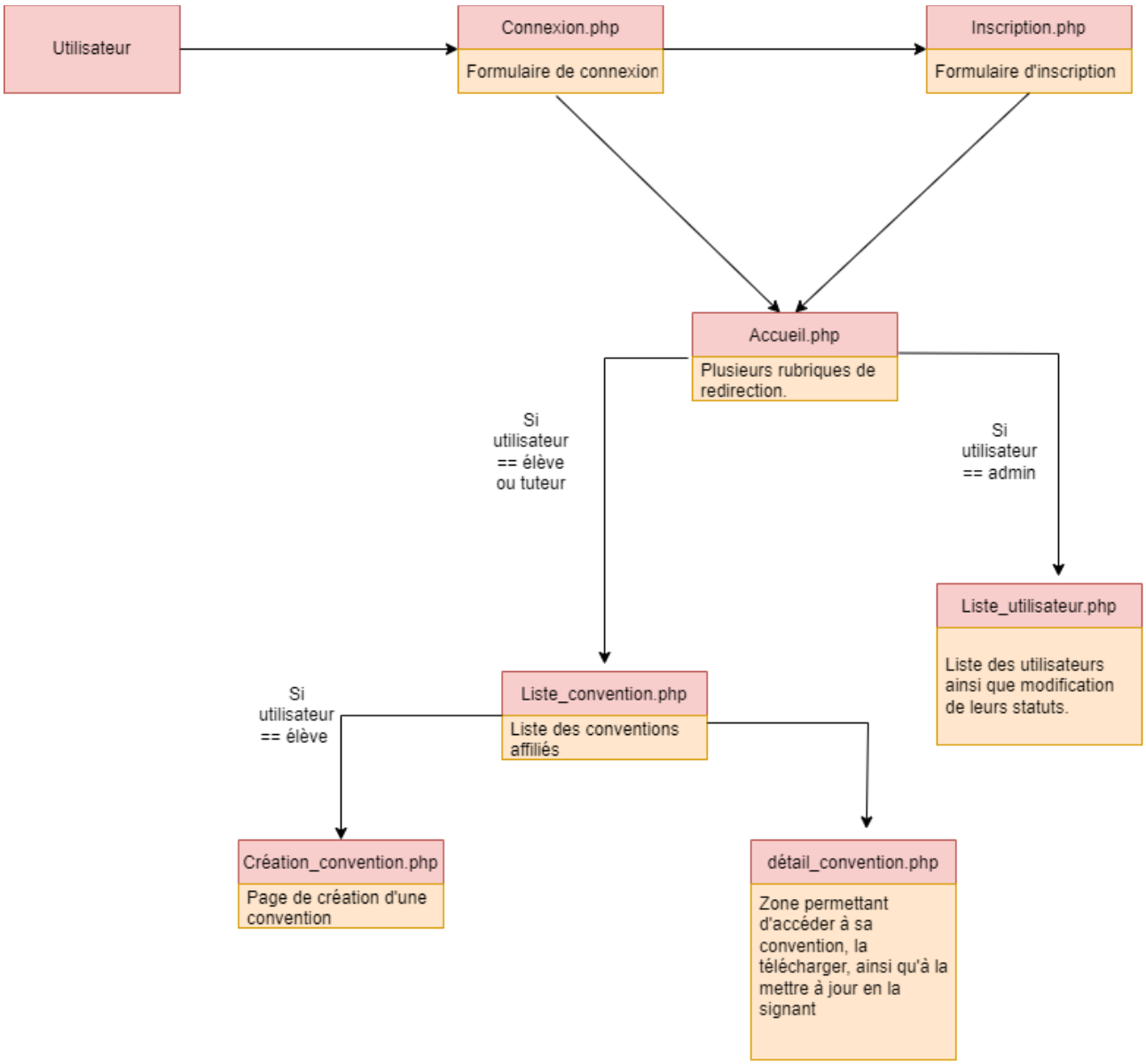
La modélisation est visible dans : PPRO605 → Modélisation → Fichier_modelisation-> base de données

Concernant la modélisation du front-end, elle a en grosse partie était imaginer sur GIMP, puis exporter sur codepen.io, pour finalement être reproduite différement sur le site, mais en gardant globalement les mêmes idées d'affichages, certains layouts ont changés.

Voici quelques maquettes, de ce à quoi ça devait originellement ressembler.



Concernant le parcours du site, crucial pour l'accès utilisateur, étant donné qu'il fallait quelque chose de simple et intuitif, j'ai crée un premier diagramme de navigation



Au final, le diagramme actuel du site est plutôt proche de celui-ci :

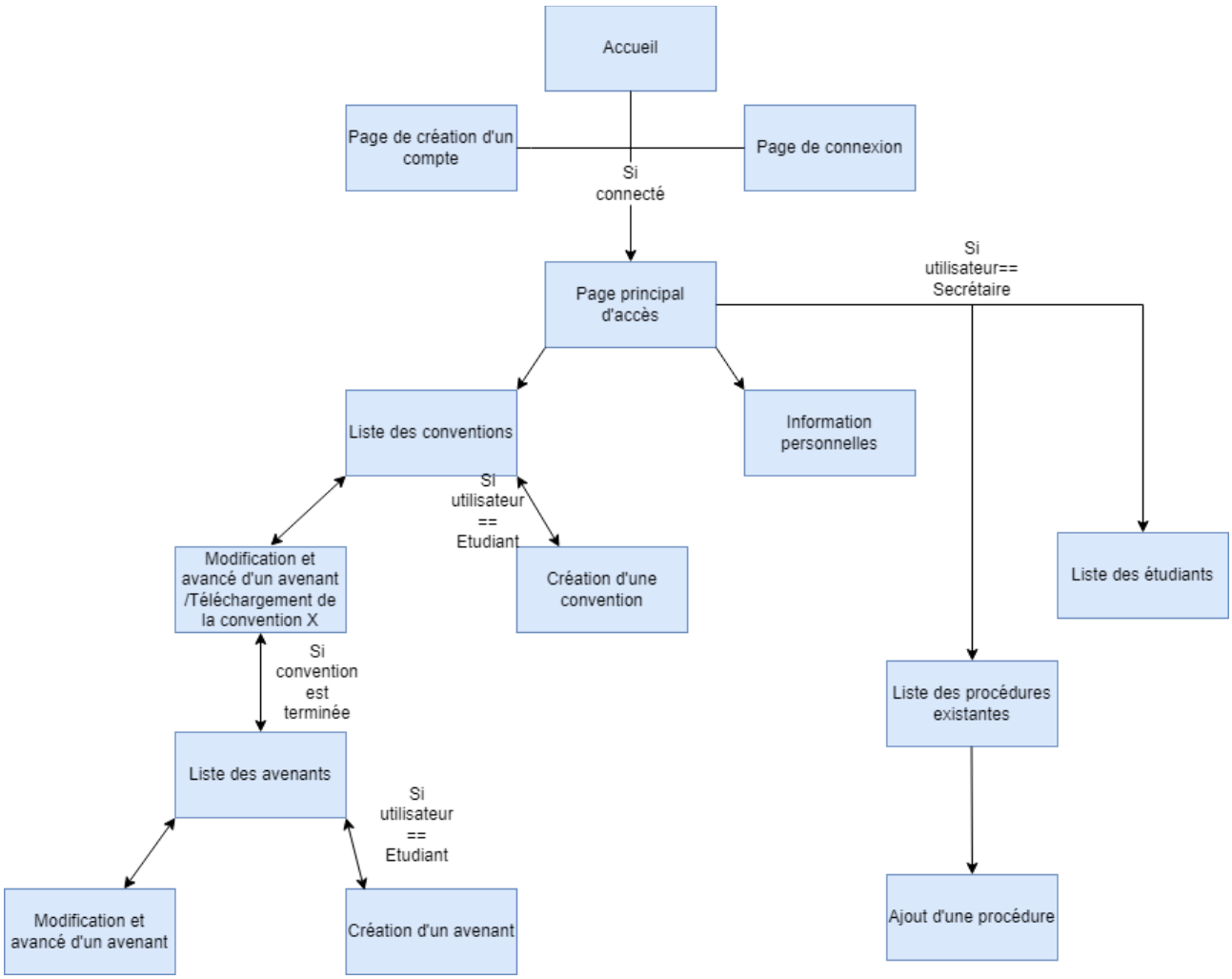


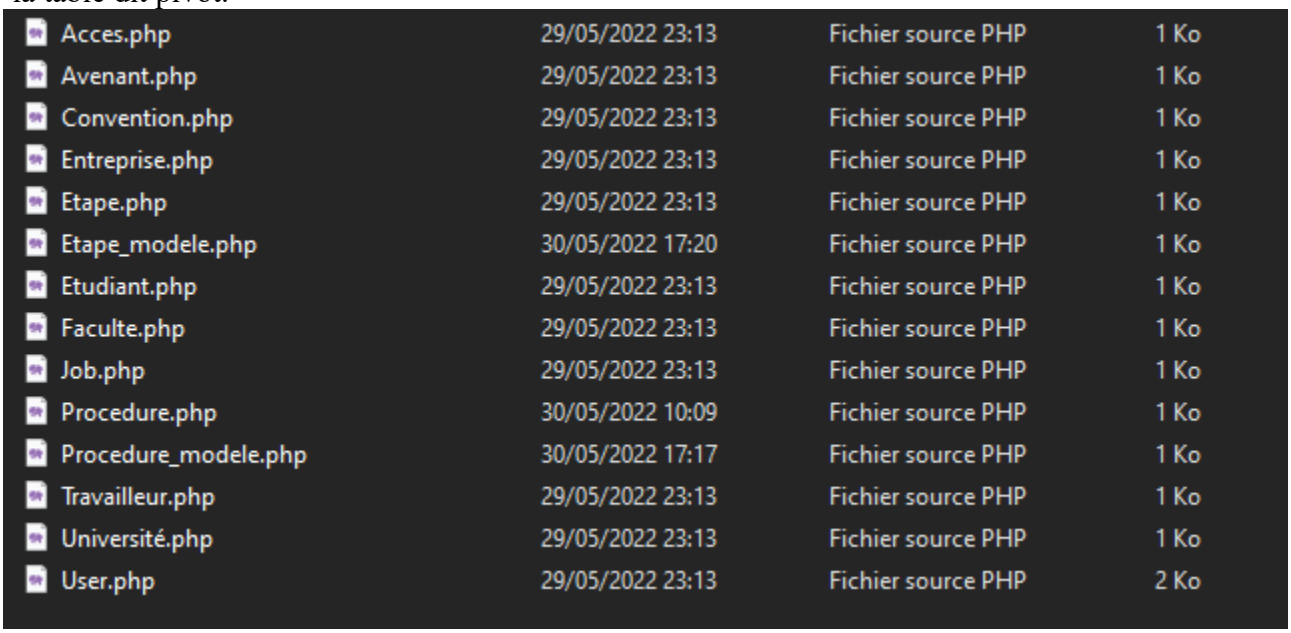
Diagramme de navigation final

III) Explication de l'implémentation

1)Modèles

Ce site fonctionnant grâce à une base de données, il faudra lier une base de donnée à Laravel, pour ceci, nous utiliserons la partie Modèle de l'architecture MVC. La liste des modèles étant dans PPRO605 → app → Models , Ils y renseignent les différentes relations. Il faut donc un modèle par table dans notre base de données, pour les tables dit «Pivot», dans le cas des relations 0,n – 0,n, il ne faudra pas forcément de modèle. A l'intérieur de ces différents fichiers, nous y retrouvons nos

différentes relations, par exemple, un utilisateur possédera des accès, pour Laravel, nous pourrons y accéder alors de cette manière : User → accès, pour cela, il faudra créer des fonctions permettant de renvoyer l'accès lié à l'utilisateur grâce à l'ID de l'accès présent dans la table utilisateur, Laravel va faire la jointure dessus automatiquement, grâce à la fonction HasOne. Il en est de même pour les autres , s'il s'agit d'une relation HasMany (0,n à 0,n) le modèle qui contient aura une fonction retournant le contenu grâce à HasMany, en lui spécifiant la table de jointure (ça peut être automatiser, suivant la convention utiliser, Laravel peut le faire seul), dans la table contenu, il faudra la relation inverse, en utilisant BelongsTo / BelongsToMany, et en respecifiant si existante , la table dit pivot.

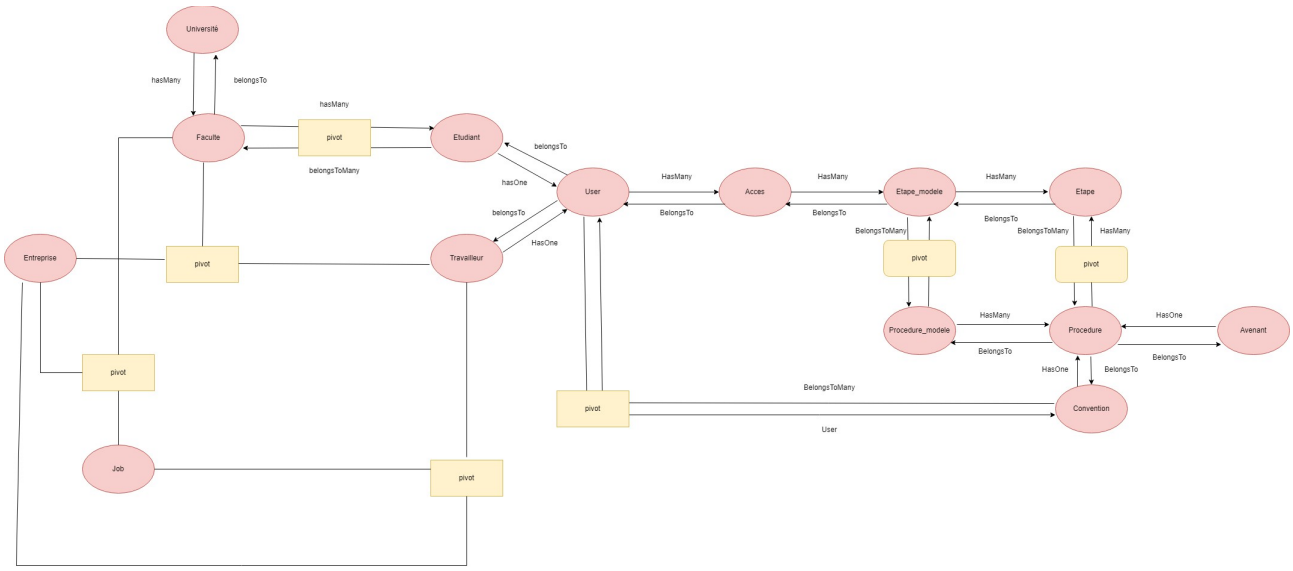


Acces.php	29/05/2022 23:13	Fichier source PHP	1 Ko
Avenant.php	29/05/2022 23:13	Fichier source PHP	1 Ko
Convention.php	29/05/2022 23:13	Fichier source PHP	1 Ko
Entreprise.php	29/05/2022 23:13	Fichier source PHP	1 Ko
Etape.php	29/05/2022 23:13	Fichier source PHP	1 Ko
Etape_modele.php	30/05/2022 17:20	Fichier source PHP	1 Ko
Etudiant.php	29/05/2022 23:13	Fichier source PHP	1 Ko
Faculte.php	29/05/2022 23:13	Fichier source PHP	1 Ko
Job.php	29/05/2022 23:13	Fichier source PHP	1 Ko
Procedure.php	30/05/2022 10:09	Fichier source PHP	1 Ko
Procedure_modele.php	30/05/2022 17:17	Fichier source PHP	1 Ko
Travailleur.php	29/05/2022 23:13	Fichier source PHP	1 Ko
Université.php	29/05/2022 23:13	Fichier source PHP	1 Ko
User.php	29/05/2022 23:13	Fichier source PHP	2 Ko

A l'intérieur de ces différents modèles, il y aura les contraintes d'accès aux données, par exemple, si dans un contrôleur, nous enregistrons un utilisateur, il faudra dans une variable de type protected nommé $fillable, spécifié l'ensemble des champs remplissable, sinon, dans la base de donnée, de base, il n'y aura pas les accès et les champs seront remplis avec NULL.

Ces différentes relations, pourront nous permettre, lorsque nous avons une collection Eloquent en retour (Une sorte de tableau Eloquent), de passer d'un modèle à l'autre, que ce soit dans une vue ou dans un controller, permettant alors l'accès facile à un tas de donnée. (Pour obtenir les étapes d'une procédure, par exemple)

Voici un schema montrant les différentes relations de Eloquant mis en place pour le projet :



(PPRO605->modélisation->fichier_modelisation->base de données

2)Vues

Pour intégrer le visuel à notre site, et générer le HTML grâce au PHP, nous allons utiliser le système de vue Laravel , ici, notre architecture de fichier est plutôt séparé, notamment avec l'utilisation de Breeze qui importe ses propres vues. (PPRO605 → ressources → views). Les vues crée par moi même, et non par Breeze, sont placé librement directement dans le fichier. Certaines utilises des boucles Foreach, permettant de parcourir un tableau à la manière classique de PHP, couplé à Eloquent afin d'accéder aux différentes tables en relations avec celle parcouru . Dans le fichier Auth/Components/layouts/vendor, il s'agit des sources Breeze. Concernant le fichier layouts, il s'agit en fait de la barre de navigation ainsi que le pied de page, intégré à d'autre grâce à Laravel et include(''), elles sont placés séparement. Il y a aussi le fichier svg contenant certaines images sous format svg, et intégrer à différentes pages.

Les vues, ici, réceptionnent les données ou les envois aux travers de formulaire PHP aux contrôleurs qui les séparent, les utilises, ou les renvois à l'utilisateur .

3) Routes

Dans le système Laravel, les routes vont permettre, en accédant à une URL , que ce soit en Get, en Post, en Put, de philtrer et de renvoyer vers les contrôleurs qui vont permettre d'exécuter du code. Ici, nous avons tout un tas de routes, dans PPRO605 → routes → web.php, permettant par exemple, à l'aide d'Ajax, d'aller chercher des données en fonction d'autres dans la base de données, par exemple, dans notre vue register, nous avons un <select> html classique d'université,qui , lorsqu'il y a une modification de choix dessus, va envoyer grâce à Ajax l'ID de l'université choisit sur une route en get, exécutant alors une fonction détaillée ci-dessous dans une fonction du contrôleur .

Avec les routes, nous pouvons passer des données facilement, et ainsi, les intégrer à la base de donnée grâce au contrôleur, ici, c'est ce que nous avons fait pour tout les systèmes d'enregistrement, que ce soit des enregistrements de comptes, de conventions, de procédures, d'élèves, mais nous pouvons aussi aller chercher grâce aux données envoyés au travers de la route

3) Contrôleur

Les fonctions du contrôleur vont avoir plusieurs rôles ici, certaines vont prendre des requêtes HTTP en paramètres, afin de traîter les données qui y circule, par exemple, pour le formulaire de création d'une convention ou d'un avenant, il va réceptionner le modèle de procédures choisit, créer une procédure unique copiée de celle-ci, créer une convention possédant cette procédure, puis lier cette convention à l'utilisateur. Certaines routes servent uniquement à renvoyer une vue générant l'html, comme pour le contrôleur gérant la page d'affiche des conventions affiliées, et l'envoyant au client.

Le contrôleur va aussi s'occuper de l'aspect sécurité de certaines routes, par exemple, imaginons la route en post servant à mettre à jour d'un avenant, elle est accessible de cette manière :

```
Route::post('/maj_avenant/{id}',[GlobalController::class,'maj_avenant'])-
>name('maj_avenant');
```

Nous voyons qu'il faut l'ID de l'avenant à modifier, imaginons maintenant qu'un utilisateur ne soit pas lié à l'avenant numéro 5, il pourrait malgré tout, tenter d'injecter une requête en post sur l'ID numéro 5 grâce au formulaire renvoyant justement sur cette route en post.

Le contrôleur, ici, va vérifier, que parmi tous les avenants de l'utilisateur connectée, celle dont le numéro est ID en fait bien parti, et ainsi bien effectuer la mise à jour, sinon, il renverra une erreur 404.

Ce procédé de sécurité est aussi présent pour d'autres formulaires, comme celui de modification d'une convention.

Le code a été commenté afin d'expliquer les routes et le contrôleur, ainsi que leurs intéractions.

IV) Conclusion :

Le cahier des charges imposaient donc un site permettant d'accéder et de suivre des conventions et leurs procédures pouvant être crée manuellement par un secrétaire, l'édition, la mise à jour des conventions en temps réel afin de simplifier, un rajout avait été fait dans le cahier des charges, celui de pouvoir inscrire à l'aide d'un fichier CSV des élèves par un secrétaire.

Au final, le projet a été développé de sorte à ce que le cahier des charges soit remplis, certaines fonctionnalités facultatives ont été implémentées et le code est fonctionnel.

Sources :

Playlist apprendre Laravel 8 NordCoders:
https://www.youtube.com/watch?v=EaSgftRyvAM&list=PLeeuvNW2FHVj4vHJRj9UDeDsXshHlnHJk&ab_channel=NordCoders

https://codepen.io/pen/

https://laravel.com/docs/9.x/eloquent-relationships#one-to-many

https://riptutorial.com/Download/laravel-fr.pdf

https://hackthestuff.com/article/laravel-8-country-state-city-dependent-dropdown-with-ajax

https://www.akilischool.com/cours/laravel-importer-exporter-les-donnees-en-excel-ou-csv-avec-spatiesimple-excel