



Le génie pour l'industrie

RAPPORT D'ACHÈVEMENT PRÉSENTÉ À L'ÉCOLE DE TECHNOLOGIE
SUPÉRIEURE DE MONTRÉAL DANS LE CADRE DU COURS
LOG795/GTI795 - PROJET DE FIN D'ÉTUDES

PROTOGEST

PFE 001 - Logiciel de coordination des intervenants pour la cour

présenté à
Professeur-superviseur
Alain April

DÉPARTEMENT DE GÉNIE LOGICIEL ET DES TI

réalisé par

Yanis Sofiane Moussaoui MOUS10109203

Arnaud Girardin GIRA18039403

Jean-Michel Lafrance LAFJ01089004

MONTRÉAL, 14 AOÛT 2019
ÉTÉ 2019



Lafrance, Girardin, Moussaoui 2019



Attribution - Partage dans les Mêmes Conditions

CC BY-SA

Cette licence permet aux autres de remixer, arranger, et adapter votre œuvre, même à des fins commerciales, tant qu'on vous accorde le mérite en citant votre nom et qu'on diffuse les nouvelles créations selon des conditions identiques. Cette licence est souvent comparée aux licences de logiciels libres, "open source" ou "copyleft". Toutes les nouvelles œuvres basées sur les vôtres auront la même licence, et toute œuvre dérivée pourra être utilisée même à des fins commerciales. C'est la licence utilisée par Wikipédia ; elle est recommandée pour des œuvres qui pourraient bénéficier de l'incorporation de contenu depuis Wikipédia et d'autres projets sous licence similaire.

Tables de matières

| | |
|--|-----------|
| Tables de matières | 3 |
| REMERCIEMENTS | 7 |
| Liste des abréviations, sigles et acronymes | 8 |
| INTRODUCTION | 9 |
| Mandat | 10 |
| Équipe | 11 |
| Livrables | 12 |
| Cas d'utilisation | 13 |
| Méthodologie de travail | 15 |
| Gestion de projet | 15 |
| Outils | 15 |
| Slack | 15 |
| Trello | 15 |
| Google Drive | 15 |
| Github | 15 |
| Discord | 15 |
| Intellij | 16 |
| AWS CLI | 16 |
| Technologies utilisées | 17 |
| Angular | 17 |
| Spring Boot | 17 |
| Cognito | 17 |
| DynamoDB | 17 |
| FullCalendar | 18 |
| Architecture | 19 |
| Frontend | 19 |
| Connexion | 19 |
| Enregistrement | 20 |
| Création d'un protocole | 21 |
| Liste de protocole | 24 |
| Outil de résumé | 27 |
| Affichage d'un protocole | 28 |

| | |
|--|-----------|
| Backend | 30 |
| Architecture | 30 |
| Base de données | 30 |
| Vérification d'authentification: | 30 |
| Code mal écrit et ne suit les bonnes lignes directrices | 31 |
| Solutions et Architecture actuelle | 31 |
| Monolith | 31 |
| API Rest | 31 |
| Déplacement de Cognito vers le backend | 32 |
| Migrer vers DynamoDB | 32 |
| Vérification de l'authentification | 32 |
| Gestion des protocoles | 32 |
| Stocker les schémas de protocoles d'instance dans DynamoDB | 34 |
| Base de données | 34 |
| Cognito | 36 |
| Transfert cognito vers le backend | 36 |
| Solution implémentée | 36 |
| Validation des tokens | 37 |
| Solution implémentée | 37 |
| Oubli de mot de passe et Profil | 38 |
| Ajout des informations utilisateur | 38 |
| Discussions | 39 |
| Angular vs ReactJS | 39 |
| MySQL vs DynamoDB | 39 |
| Problèmes rencontrés | 40 |
| Modifications futures | 41 |
| Enjeux Environnementaux | 42 |
| CONCLUSION | 43 |
| RÉFÉRENCES | 44 |
| Annexe 1 - | 45 |
| Cas d'utilisations | 45 |

LISTE DES TABLEAUX

| | | |
|-----------|-----------------------------|----|
| Tableau 1 | Tableau des définition | 8 |
| Tableau 2 | Tableau des acronymes | 8 |
| Tableau 3 | Membre et rôle de l'équipe | 11 |
| Tableau 4 | Documents à livre | 12 |
| Tableau 5 | Cas d'utilisation | 13 |
| Tableau 6 | cycle de vie d'un protocole | 26 |

LISTE DES FIGURES

| | | |
|-----------|--|----|
| Figure 1 | Page de connection original | 19 |
| Figure 2 | Page de connection v2 | 20 |
| Figure 3 | Page d'enregistrement | 20 |
| Figure 4 | Création de protocole - Nom et type | 22 |
| Figure 5 | Création de protocole - Sélection des sections | 22 |
| Figure 6 | Création de protocole - Remplissage du protocole | 23 |
| Figure 7 | Création de protocole - Sauvegarde et envoi | 24 |
| Figure 8 | liste des protocoles original | 24 |
| Figure 9 | Liste des protocoles modifier | 25 |
| Figure 10 | section sent de la liste de protocol | 27 |
| Figure 11 | outil de récapitulation de la liste des protocoles | 28 |
| Figure 12 | Affichage d'un protocole | 29 |
| Figure 13 | Diagramme simplifié de l'architecture du backend | 31 |
| Figure 14 | Diagramme simplifié de l'architecture du backend | 32 |
| Figure 15 | Diagramme simplifié de l'architecture du backend | 34 |
| Figure 16 | Diagramme simplifié de l'architecture du backend | 35 |
| Figure 17 | Diagramme simplifié de l'architecture du backend | 37 |

REMERCIEMENTS

Nous aimerions commencer ce document en remerciant toutes les personnes qui ont su participer de près ou de loin à la réussite du projet. Nous aimerions tout d'abord remercier notre professeur superviseur Alain April pour son implication et ses bons conseils tout au long du projet.

Par la suite, nous voulons souligner l'implication de Mathieu Dupuis, étudiant diplômé de l'ÉTS et expert en technologie Amazon Web Service (AWS) pour ses disponibilités et connaissances qui ont su nous aider à faire avancer le projet dans la bonne direction.

De plus, nous souhaitons particulièrement éprouver notre reconnaissance envers, Frank Calandriello, le promoteur du projet, d'avoir libéré du temps de son horaire chargé pour venir voir notre présentation à l'école et nous donner son point de vue sur l'avancement du projet.

Finalement, pour tous ceux qui ont joué un rôle dans le projet, plus particulièrement Xavier Reid membre du groupe qui a dû quitter, mais qui a fait un travail exemplaire et qui a facilité la prise en main du projet lors de son départ pour les membres restants. Il ne faut pas oublier Guillaume Larente, membre d'une année précédente, qui a su libérer de son temps pour répondre à nos questions.

Liste des abréviations, sigles et acronymes

| Terme | Définition |
|-----------------------------|---|
| Angular | Framework Open source pour la construction d'interface web. |
| API | Interface de programmation qui sert de façade pour les logiciels qui l'utilisent |
| JSON | Structure de données textuelles, permettant la représentation d'information selon un format précis. |
| NoSQL | Famille de base de données non relationnelles pouvant être sous plusieurs formes, notamment sous forme de paires clés-valeurs ou même de graphes. |
| Protocole d'instance | Formulaire rempli par les avocats et les intervenants pour s'entendre sur les dates de déroulement d'une cause. |
| React | Bibliothèque JavaScript libre pour la création des applications Web. |
| Service REST | Style d'architecture définissant un ensemble de contraintes et de propriétés basées sur le protocole HTTP. |
| Spring Boot | Framework Open Source, basé sur Java, permettant de créer des services web. |

Tableau 1 - Tableau des définitions

| Terme | Définition |
|----------------|--|
| AWS | Amazon Web Services |
| AWS CLI | Interface de ligne de commande du service Web Amazon |
| ÉTS | École des technologies supérieur |
| PFE | Projet de fin d'études. |
| SRS | Spécification des exigences d'un logiciels |
| IDE | Environnement de développement |

Tableau 2 - Tableau des acronymes

INTRODUCTION

Le projet a débuté grâce à Maître Frank Calandriello et à plusieurs étudiants en droit à l'Université McGill, qui se plaignaient que le processus de demande d'instance était inefficace et long. Dans le contexte actuel, un avocat qui désire faire une demande en justice doit remplir un document nommé protocole d'instance. Pour se faire, il doit tout d'abord proposer des dates pour chaque présence en cours pour prouver qu'il peut mener à bien la cause. Par la suite, il doit contacter tous les intervenants concernés afin de trouver des dates qui conviendront à chacun des participants, et ce, pour chacune des sections du protocole d'instance. Présentement tout est rempli à la main par l'avocat lui-même ou, dans certains cas, sa secrétaire. Ce processus doit être recommencé à chaque fois qu'il y a un conflit d'horaire avec un des intervenants, ce qui cause une perte de temps pour plusieurs avocats qui ont déjà peu de temps. Avec l'aide d'Alain April, un projet de fin d'études a été soumis au département logiciel de l'ÉTS. Suite à plusieurs sessions et de plusieurs étudiants, le projet Protogest a vu le jour afin de faciliter le processus de demande d'instance pour les avocats.

Cette plateforme est un outil de gestion et de communication entre les avocats afin de faciliter la création d'un protocole d'instance. La plateforme permet aux avocats de communiquer facilement si les dates proposées par le protocole d'instance leur conviennent ou non.

À ce jour, trois équipes ont déjà participé au projet. Au fil des sessions, chaque équipe a su améliorer la qualité de l'application ainsi qu'ajouter de nouvelles fonctionnalités rendant l'application de plus en plus complète. Au départ, la première équipe a dressé les besoins du client et a développé une première solution en React qui aura permis d'exposer les fonctionnalités de base de l'application. Par la suite, la seconde équipe aura recommencé le projet à neuf en changeant de cadre technique pour se tourner vers Angular. De plus, elle aura aussi apporté des modifications au backend afin de le convertir en architecture micro service ainsi qu'un changement de base de données se tournant ainsi vers MySQL au lieu de H2. La troisième équipe aura continué sur les pas de l'équipe précédente en utilisant Angular et en modifiant minimalement le backend. Par contre ceux-ci ont commencé à intégrer plusieurs services Amazon dans le projet.

Ce rapport présente l'évolution de cette plateforme lors du projet de fin d'études d'été 2019. Les changements fonctionnels de celle-ci ont été basés sur les demandes des clients et les fonctionnalités non complétées par les équipes des sessions passées.

Mandat

Le mandat du projet est l'amélioration et le développement d'une plateforme web qui servira à gérer et faciliter la création de protocoles d'instance. Malgré le fait que le projet a beaucoup avancé, le passage de plusieurs équipes aura fait place à une grande accumulation de dette technique. Le mandat sera principalement de réusiner le code existant afin de le réorienter dans la bonne direction afin que les équipes suivantes aient une base solide pour continuer le projet.

Il est important de noter qu'en milieu de mandat les besoins du client ont dévié drastiquement par rapport à ce qui était initialement demandé. Le projet développement du projet a donc commencé par une solution axée sur la rapidité et l'automatisation des fonctions, pour ensuite mettre de l'emphase sur la simplicité d'utilisation et l'amélioration des fonctions de base. Donc, le prototype devra permettre à un utilisateur de créer et remplir un protocole d'instance plus facilement et permettre à l'utilisateur et aux autres intervenants de voir les protocoles reçus de façon simple. Ultimement, l'application saura mieux satisfaire les besoins du client par sa facilité d'utilisation, mais sera une base solide pour les développements à venir.

Équipe

L'équipe a été formée par l'entremise d'un sondage Moodle. Malgré le fait qu'aucun des membres de l'équipe ne se connaissait, l'équipe est quand même parvenue à bien s'entendre pour toute la durée du projet. Les membres de l'équipe ont fait bien attention de diviser les tâches de manière à faire ressortir les forces de chacun de ses membres. En cas de problème, il n'y a pas eu d'hésitation pour demander de l'aide que ce soit à un autre membre de l'équipe ou même à au représentant du client, Mathieu, pour les problèmes concernant AWS.

| Nom | Description | Responsabilités |
|-------------------------|------------------------|---|
| Frank Calandriello | Client | <ul style="list-style-type: none">- Émettre les besoins,- valider les fonctionnalités des prototypes- donner son avis sur les résultats |
| Alain April | Superviseur de PFE | <ul style="list-style-type: none">- Supporter l'équipe de PFE- Superviser les activités de l'équipe |
| Mathieu Dupuis | Représentant du client | <ul style="list-style-type: none">- Aider l'équipe de PFE au niveau technique- Offrir ces connaissances en service Amazon Web Services |
| Jean-Michel Lafrance | Développeur | <ul style="list-style-type: none">- Concevoir et programmer le frontend- Programmer le backend |
| Arnaud Girardin | Développeur | <ul style="list-style-type: none">- Concevoir et programmer le frontend- Programmation backend |
| Yanis Sofiane Moussaoui | Développeur | <ul style="list-style-type: none">- Concevoir et programmer le backend- Programmer le frontend- Intégrer les services AWS à Protogest |
| Xavier Reid | Développeur | <ul style="list-style-type: none">- Programmer le frontend |

Tableau 3 - Membre et rôle de l'équipe

Livrables

Durant la session, plusieurs livrables, en plus du code source des différentes sections de la plateforme, étaient nécessaires au bon déroulement du projet. Pour débiter, une nouvelle version du document de vision a été créée pour documenter les nouvelles demandes du client. De plus, un rapport final a été produit pour répondre aux exigences académiques du cours LOG792. Celui-ci récapitule tout le travail effectué durant la session, incluant les décisions prises, problèmes rencontrés et autres discussions.

| Nom du livrable | Description |
|------------------------|--|
| Document de vision 2.0 | Document de vision contenant les nouvelles demandes du client. |
| Code source frontend | Code de l'application frontend. |
| Code source backend | Code de l'application backend. |
| Rapport final | Document expliquant le travail accompli durant la session. |

Tableau 4 - Documents à livrer

Cas d'utilisation

Lors du projet, des cas d'utilisation ont été mis en place afin de cerner les nouveaux besoins du clients et mettre en place de ligne de conduite pour le logiciel. Orientant ainsi la production et le développement de l'application. Les cas suivant représente toutes les fonctionnalités qui seront mis à la disposition du client lorsque l'application sera complétée.

| Cas d'utilisation | Description |
|--|--|
| CU01 - Inscription de l'utilisateur | L'utilisateur peut créer un nouveau compte dans le système. |
| CU02 - Connexion de l'utilisateur | L'utilisateur peut se connecter en utilisant ses informations d'identification |
| CU03 - Récupération d'un mot de passe oublié par l'utilisateur | L'utilisateur peut récupérer son mot de passe si celui-ci a oublié l'original |
| CU04 - Vérification de l'authentification et redirection | L'utilisateur doit être authentifié afin de visiter certaines pages |
| CU05 - Modifier ces informations de profil | L'utilisateur peut changer ces informations personnelles après s'être connecté au site |
| CU06 - Modifier le mot de passe | L'utilisateur peut changer son mot de passe après s'être connecté à la plateforme |
| CU07 - Lister les protocoles d'un utilisateur | L'utilisateur peut voir la liste des protocoles d'instance reliés à son compte, autant ceux reçus qu'envoyés |
| CU08 - Créer un protocole d'instance | L'utilisateur peut créer un protocole d'instance avec les dates qu'il a choisies |
| CU09 - Afficher les conflits de dates | L'utilisateur peut récupérer les dates occupées de son calendrier personnel |
| CU10 - Créer un Gabarit personnalisé | L'utilisateur a la possibilité de créer un gabarit de protocole d'instance personnalisé et de le sauvegarder pour une future utilisation |
| CU11 - Éditer un Gabarit personnalisé | L'utilisateur à la possibilité d'éditer un gabarit de protocole d'instance personnalisé |
| CU12 - Inviter des utilisateurs à un protocole d'instance | L'utilisateur a la possibilité d'inviter plusieurs utilisateurs sur un protocole d'instance |

| | |
|---|---|
| CU13 - Accepter les instances d'un protocole | L'utilisateur doit pouvoir accepter une ou plusieurs dates d'un protocole d'instance reçu |
| CU13 - Refuser les instances d'un protocole | L'utilisateur doit pouvoir refuser une ou plusieurs dates d'un protocole d'instance reçu |
| CU14 - Éditer un protocole | L'utilisateur peut éditer un protocole d'instance à n'importe quelle étape |
| CU15 - Supprimer un protocole | L'utilisateur peut supprimer un protocole d'instance à n'importe quelle étape, excepté archivé |
| CU16 - Effectuer l'envoi final d'un protocole | L'utilisateur peut effectuer un envoi final du protocole lorsque celui-ci est accepté par tous les intervenants |
| CU17 - Afficher le protocole dans le calendrier | L'utilisateur peut afficher un calendrier avec toutes les dates d'instances du protocole |

Tableau 5 - Cas d'utilisation

Voir [l'annexe 1](#) pour plus de détails

Méthodologie de travail

Gestion de projet

Pour assurer l'avancement du projet, les membres de l'équipe ont opté pour une approche Scrum. C'est ainsi que les membres de l'équipe ont décidé de faire des Sprints d'une semaine avec une rencontre chaque vendredi après-midi. Cette rencontre permettait aux membres de l'équipe de partager les tâches accomplies afin que le superviseur de projet constate l'avancement du projet ainsi que de planifier les tâches à effectuer au cours de la semaine à suivre. De plus, cette réunion permet de détecter les tâches qui sont bloquées soit par d'autres tâches ou par un défi technique, ce qui permet à l'équipe de s'entraider et réduire les retards dans la planification.

Outils

Slack

Slack est une application de messagerie permet de faciliter les communications entre les membres de l'équipe entre les réunions. De plus, elle permet le partage de liens et de fichiers utiles pour réaliser le projet.

Trello

Trello est un outil permettant la gestion de tâches sous la forme de listes. Cet outil a été utilisé pour assigner les tâches aux membres de l'équipe et aussi de faire le suivi de celle-ci en les déplaçant selon leur statut (À faire, En cours, Complété). Cet outil est la base pour planifier les Sprints lors de nos réunions.

Google Drive

Google drive est un outil de partage et de collaboration utilisé pour travailler sur les différents documents à produire au cours de la session. Tous les documents qui ne sont pas du code sont centralisés sur cette application.

Github

Github est utilisé pour la centralisation et la gestion du code source des différents modules du projet. Ce service est utilisé puisque le projet se trouvait déjà sur cette plateforme et qu'elle est facile à utiliser.

Discord

Discord est une application de communication vocale qui a été utilisée durant la fin du projet afin de les membres du projet puissent discuter entre eux.

IntelliJ

IntelliJ est un IDE utilisé pour travailler autant sur le code Java du backend. Celui-ci permet une bonne intégration avec le système de gestion de dépendances Maven. Finalement, IntelliJ offre plusieurs outils de rétro-ingénierie qui permettent d'avoir une meilleure compréhension de l'architecture du projet initial.

AWS CLI

AWS CLI permet la communication et la gestion de leurs services à partir d'une interface en ligne de commande. Celle-ci a été utilisée pour communiquer et diagnostiquer les différents services utilisés dans AWS, ex: Cognito.

Technologies utilisées

Angular

Comme frontend l'équipe a décidé d'opter pour Angular. Ce cadriciel a été choisi puisqu'il est l'un des plus utilisés dans le domaine, mais aussi puisque la dernière équipe ayant travaillé sur le projet semblait déjà avoir implémenté une bonne partie des fonctionnalités de l'application avec celui-ci. De plus, le fait que certains membres avaient déjà de l'expérience avec Angular aura été une forte motivation vers ce choix de technologie.

Spring Boot

Pour le backend, le cadriciel d'infrastructure Spring Boot en Java a été sélectionné puisque c'était le cadriciel de prédilection de l'équipe précédente. Une autre motivation pour ce choix est le fait qu'un des membres de l'équipe travail régulièrement avec Spring Boot dans son emploi. Il est aussi important de mentionner que Maven a été utilisé pour faire la gestion des dépendances du projet. De cette façon, il a été assez facile d'inclure plusieurs modules nécessaires pour l'application web ainsi que les modules facilitant l'intégration d'AWS. Ces modules auront permis de grandement réduire le code écrit grâce aux annotations fournies par le module permettant l'intégration avec DynamoDB. De plus, il est assez facile d'implémenter de nouvelles routes avec les annotations fournies par Spring Boot.

Cognito

Ce service est utilisé pour faire l'authentification des utilisateurs dans l'application Protoquest ainsi que de garder certaines informations relatives au profil de l'utilisateur. Ce service était déjà en place dans le frontend lors de la prise en main du projet. Cependant, ce service aura été déplacé dans le backend afin de ne pas exposer certaines clés propres au service par exemple le PoolId. La principale motivation de garder le service est le fait de ne pas avoir à maintenir les informations des utilisateurs dans la base de données permettant ainsi de focaliser sur d'autres aspects de l'application.

DynamoDB

DynamoDB est une base de données NoSQL hébergée directement sur les serveurs Amazon. Le service était déjà partiellement utilisé par l'application quand le projet a débuté. En effet, l'application était dans un état où elle utilisait deux bases de données différentes en même temps, soit une base de données DynamoDB ainsi qu'une base de données MySQL. L'équipe a fait le choix d'éliminer la base de données MySQL puisque DynamoDB offre la possibilité de mettre directement un document JSON dans une colonne ce qui a permis de réduire la

complexité de la sauvegarde et la récupération d'information d'un protocole d'instance qui aurait nécessité plusieurs jointures en MySQL.

FullCalendar

FullCalendar est un module permettant l'affichage d'un calendrier comportant des événements pour certaines dates. Le module était déjà utilisé par l'équipe précédente, cependant des modifications ont été apportées pour l'utiliser à son plein potentiel et permettre ainsi d'afficher les événements (dates) du protocole d'instance correctement dans le calendrier.

Architecture

Frontend

Lorsque le projet a débuté, l'équipe a hérité d'un frontend déjà entamé utilisant le cadriciel Angular. Les membres de l'équipe avaient donc le choix de continuer en utilisant le code déjà existant en Angular ou de réutiliser le code source de la première session qui était fait en React. Après avoir fait l'analyse des deux bases de code source, le code source utilisant Angular a été choisi, car il était beaucoup plus avancé que celui utilisant React.

Malgré que le code source était le plus avancé des deux, certaines pages de l'application fonctionnaient partiellement ou pas du tout. Donc la première étape était d'analyser chaque page qui a été créée par les équipes antérieures, soit les pages:

- Connexion
- Enregistrement
- Création d'un protocole
- Liste de protocole
- Affichage d'un protocole

Les problèmes rencontrés, changements et amélioration de chaque page seront discutés en plus grand détail dans les sections suivantes.

Connexion

Pour la page de connexion, celle-ci permet à l'utilisateur de se connecter ou de se déplacer entre les différentes fonctionnalités pour l'utilisateur. Après une analyse au niveau du design, aucune modification n'était nécessaire (Figure 1). Toutefois un bouton a été ajouté afin de diriger vers la nouvelle page mot de passe oublié (Figure 2).

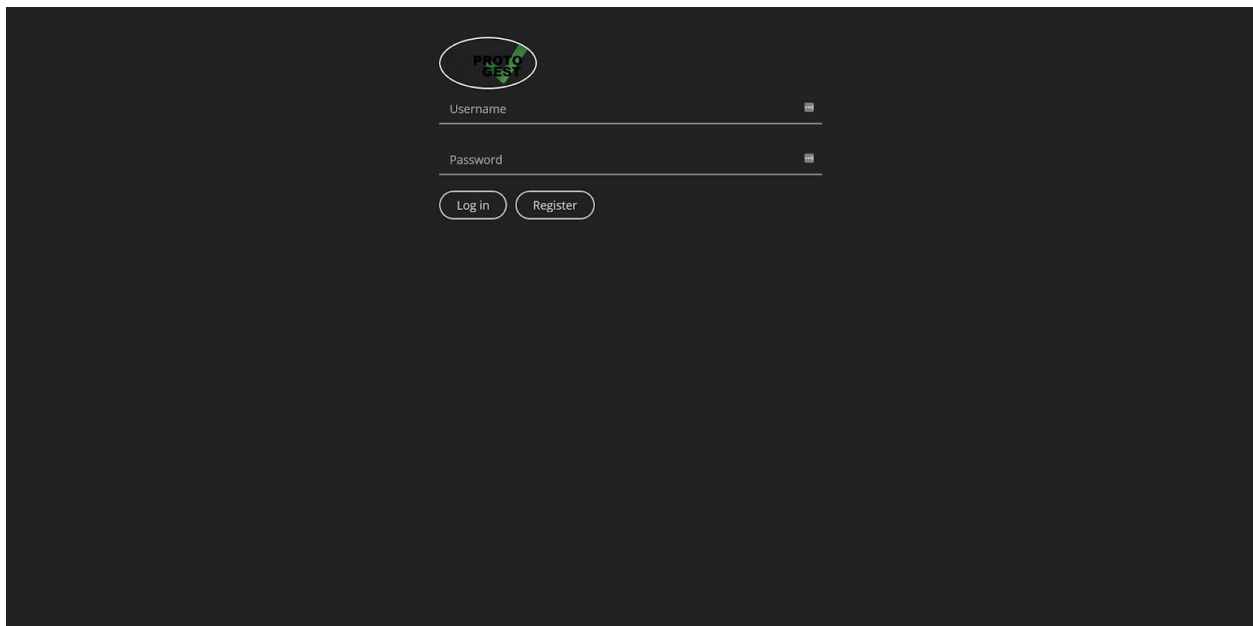


Figure 1 - Page de connexion original

Cependant la page représentait des risques de sécurité au niveau de l'utilisation de Cognito. Il y avait donc un choix à faire entre ignorer les risques de sécurité et garder en gardant le code dans le frontend ou le déplacer vers dans le backend. Nous avons décidé de déplacer celui-ci dans le backend. Pour plus d'information sur le sujet se référer à la section appropriée ([Cognito](#)).

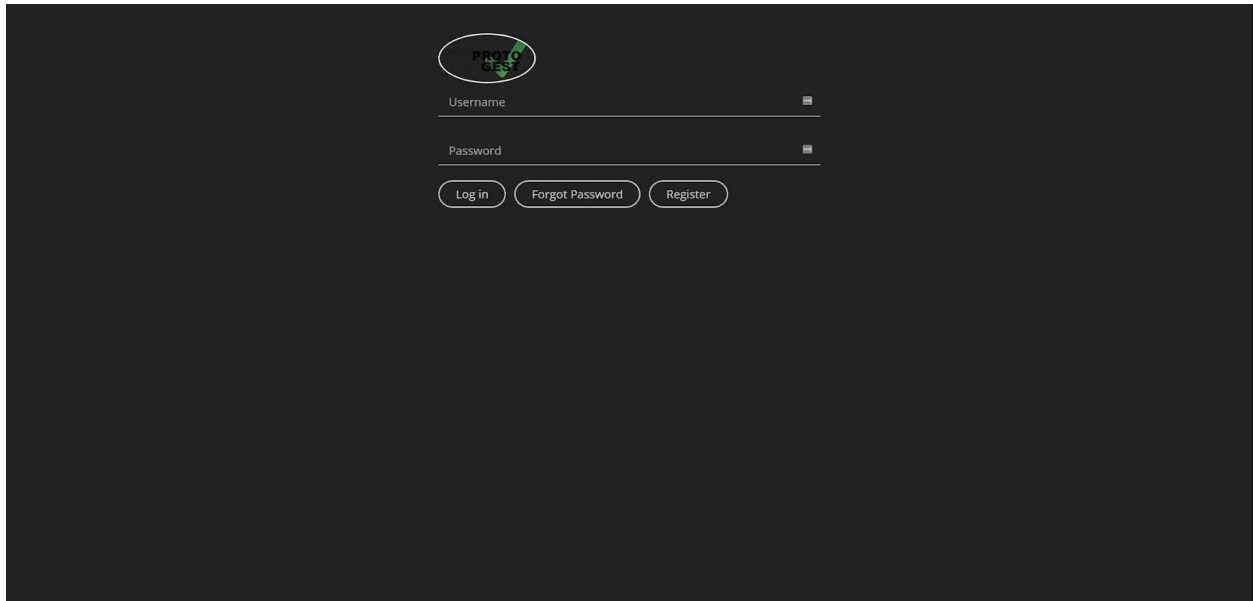


Figure 2 - Page de connexion v2

Enregistrement

La page Enregistrement permet à l'utilisateur de rentrer ces informations personnelles afin de créer un compte sur la plateforme. Le design des équipes des sessions antérieures était déjà très simple et facile à comprendre pour le client. Donc, aucun changement n'était pas nécessaire (Figure 3).

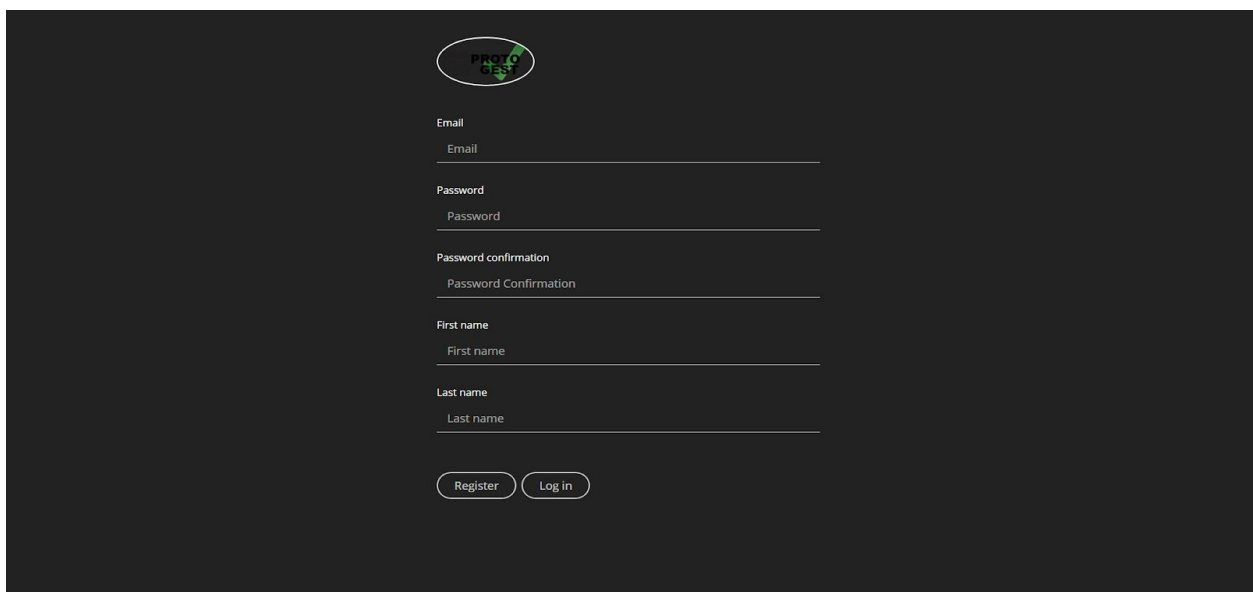


Figure 3 - Page d'enregistrement

Par contre, malgré que tous les champs apparaissent à l'inscription, seulement courriel et mot de passe étaient utilisés. Deux options existaient pour résoudre le problème, soit retirer les champs non utilisés ou créer de nouvelles fonctionnalités dans le backend pour enregistrer les champs manquants comme il se doit. Originellement les informations n'étaient pas utilisées nulle part donc l'option de les retirer aurait été la bonne, mais pour personnaliser le frontend celles-ci ont été rajoutées dans l'en-tête et la liste de protocole. Ces champs sont maintenant obligatoires, le backend a donc été modifié pour les ajouter dans les fonctionnalités de création. Pour plus d'information, se référer à la section appropriée ([Cognito](#)).

Création d'un protocole

La création de protocoles représente le cas d'utilisation le plus important de l'application. En effet, sans cette section, toutes les autres fonctionnalités de l'application ne peuvent pas fonctionner. Il était donc important de réviser l'état actuel de la section avant de passer à autre chose. Au départ, la section était fonctionnelle ce qui était une bonne chose puisque ce n'était pas le cas pour certaines autres fonctionnalités. Cependant, la section n'était pas nécessairement claire pour un utilisateur n'ayant jamais touché à l'application ce qui allait un peu à l'encontre du but premier de l'application qui est de rendre celle-ci plus simple à utiliser. La section était déjà divisée en étapes, cependant les étapes n'étaient pas nécessairement claires et certaines d'entre elles étaient même redondantes. Plusieurs décisions ont été prises afin de rendre l'application plus simple à utiliser, mais aussi de rendre l'affichage plus facile à lire.

Tout d'abord, il n'y avait pratiquement rien à faire au niveau du frontend en ce qui concerne l'intégration des dates d'un calendrier Outlook. La seule chose qui a été ajoutée à ce niveau, est la désactivation de la sélection des dates dans le datepicker en fonction des dates d'indisponibilité provenant d'Outlook.

Par la suite, un immense réusinage au niveau des étapes nécessaires à la création d'un protocole a été fait. Le processus de création a donc été divisé de façon plus concise pour en arriver aux sections suivantes :

- Nom et type
- Sélection des sections
- Remplissage du protocole
- Sauvegarde et envoi

La première section est assez similaire à celle qui existait auparavant. En effet, elle comporte toujours la sélection du type de protocole. Le grand changement apporté à cette section est la possibilité d'ajouter un nom au protocole. Cette fonctionnalité est un gros plus, puisqu'elle permet maintenant d'identifier les protocoles d'une façon plus précise.

The screenshot shows the 'Create a Protocol' form in the Protogest application. The form is divided into four steps: 1. Protocol name & type, 2. Sections, 3. Fill in the details, and 4. Collaborate. The first step is active, showing a text input for 'Protocol Name' with the value 'Mon Protocole'. Below the input, there are two radio buttons: 'Quebec protocol' (selected) and 'Canadian protocol'. A 'Next' button is located at the bottom of the form.

Figure 4 - Création de protocole - Nom et type

La deuxième section est complètement nouvelle. Sa naissance est principalement due au fait que la section de remplissage contenait trop de contenu qui n'est parfois pas nécessaire pour tous les avocats. Cette section permet donc d'ajouter explicitement les sections qu'on veut retrouver dans son protocole afin d'éviter d'avoir une liste déroulante de sections qui demande beaucoup trop de défilement.

The screenshot shows the 'Create a Protocol' form in the Protogest application, specifically the 'Sections' step. The form is divided into four steps: 1. Protocol name & type, 2. Sections, 3. Fill in the details, and 4. Collaborate. The second step is active, showing a list of sections to be added to the protocol. The sections are: 'Moyens Préliminaires', 'Autres Procédures', 'Défense', 'Expertises', 'Interrogatoires', and 'Pièces'. Each section has a checkbox next to it. The 'Moyens Préliminaires' section is selected. Below the list, there are 'Back' and 'Next' buttons.

Figure 5 - Création de protocole - Sélection des sections

La troisième section est celle qui aura le plus changé. Au départ, cette section ne contenait qu'un gigantesque composant accordéon qui contenait toutes les sections, utiles ou non. Le fait d'avoir ajouté la section précédente a permis de réduire le problème puisqu'elle permet

d'afficher moins de sections. Cependant, le problème de défilement reste le même à cause du composant accordéon. Ce composant a donc été modifié pour qu'il utilise un stepper afin de rendre le processus de remplissage plus linéaire, mais aussi d'éviter d'avoir une énorme liste demandant beaucoup de défilement de la page. De plus, le formulaire a été modifié afin qu'il soit mieux divisé en général à l'intérieur des sections pour rendre le tout plus clair pour l'utilisateur.

Protogest

Create a Protocol

1 Protocol name & type 2 Sections 3 Fill in the details 4 Collaborate

1 Moyens Preliminaires 2 Autres Procedures

Moyens Preliminaires

Moyens declinatoires Yes No

Renvoi au tribunal competent ou rejet (art. 177 C.p.c.) 2019-08-10

Autre (avec reference a l'article C.p.c.) 2019-08-06

Soumis par (inscrire le nom de la partie)

Moyens d'irrecevabilite Yes No

En rejet (art. 168 C.p.c.) 2019-08-17

Soumis par (inscrire le nom de la partie)

Back Next

Figure 6 - Création de protocole - Remplissage du protocole

Pour terminer, il y a la section de sauvegarde et envoi. Cette section est plutôt similaire à la section qui existait auparavant, cependant plusieurs changements ont été apportés. Cette section comporte une combinaison de la section d'invitation, qui était une section à part entière dans la version précédente, ainsi que la section de sauvegarde d'un protocole. Il est maintenant possible d'ajouter un ou plusieurs courriels afin d'inviter un ou plusieurs intervenants au protocole afin qu'ils valident les dates proposées. Pour terminer, un deuxième bouton permettant de sauvegarder sans envoyé d'invitation aura été ajouté afin de permettre une utilisation ultérieure du protocole en question.

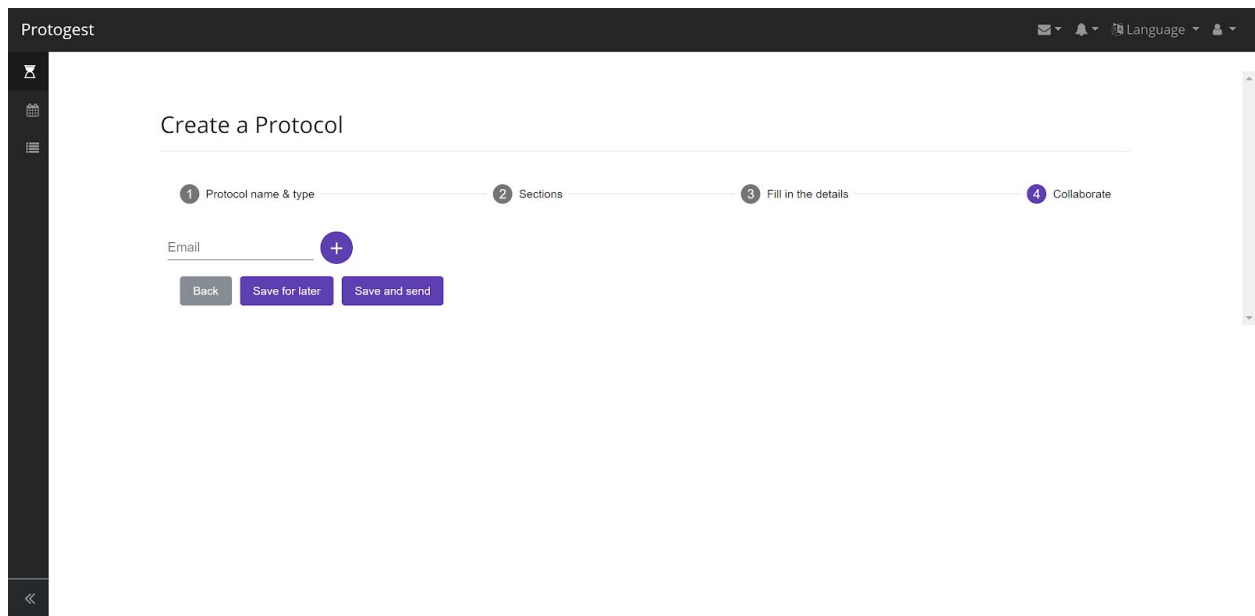


Figure 7 - Création de protocole - Sauvegarde et envoi

Liste de protocole

En termes de fonctionnalités, la page de liste de protocole comportait déjà la plupart des informations nécessaires en affichant la liste des protocoles reçus et ceux envoyés. Par contre, pour ce qui est des fonctionnalités plus spécifiques ainsi que l’affichage, la page comportait plusieurs lacunes. Non seulement l’affichage de la liste était très compact et difficile à comprendre, mais il y avait aussi un problème au niveau du défilement de la page puisque les deux listes étaient empilées l’une par-dessus l’autre.

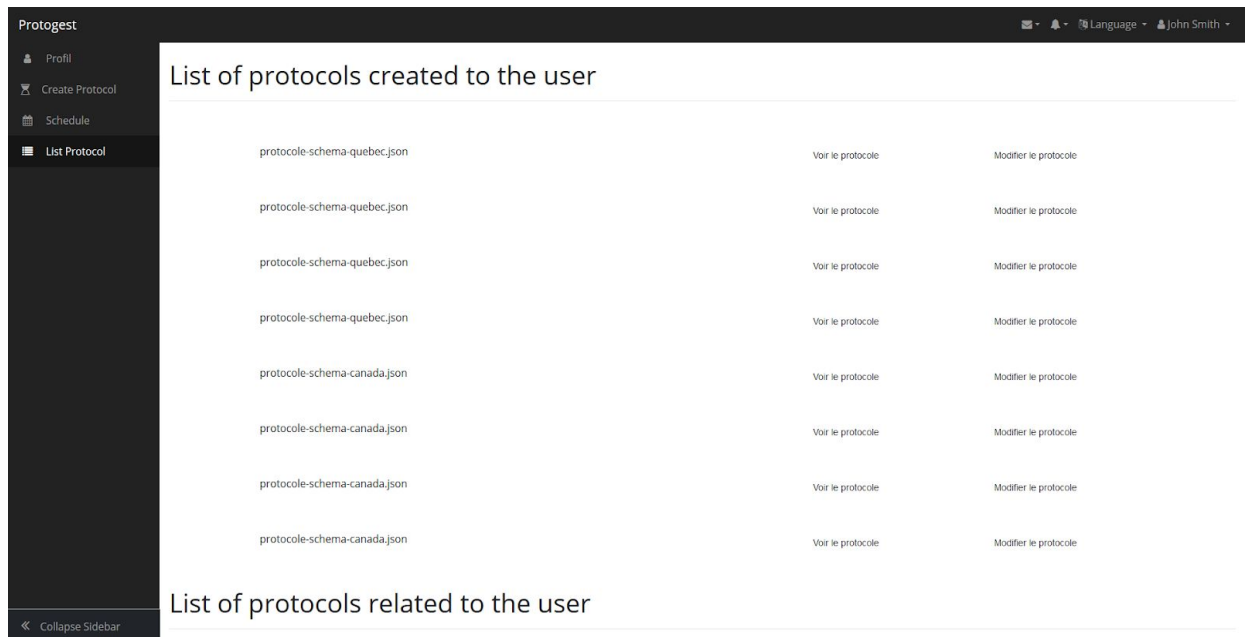


Figure 8 - liste des protocoles original

Tout d'abord, en regardant cette version il aurait été compliqué pour un utilisateur de faire la différence entre les différents protocole affichés. En effet, à l'exception du type de protocole, aucune information utile permettant de différencier les protocoles n'apparaissait dans la liste ce qui rendait la tâche de l'utilisateur plus difficile que nécessaire pour faire la différence entre deux protocoles. De plus, il n'y avait aucune indication pour montrer à l'utilisateur à quelle étape chaque protocole était rendu. Finalement, les boutons textuels sont encombrants et il est difficile de savoir que ce sont des boutons puisqu'ils n'ont pas une couleur différente du reste de la page. Après discussion et délibération, une nouvelle version de l'affichage qui répondait plus aux besoins du client fut produite.

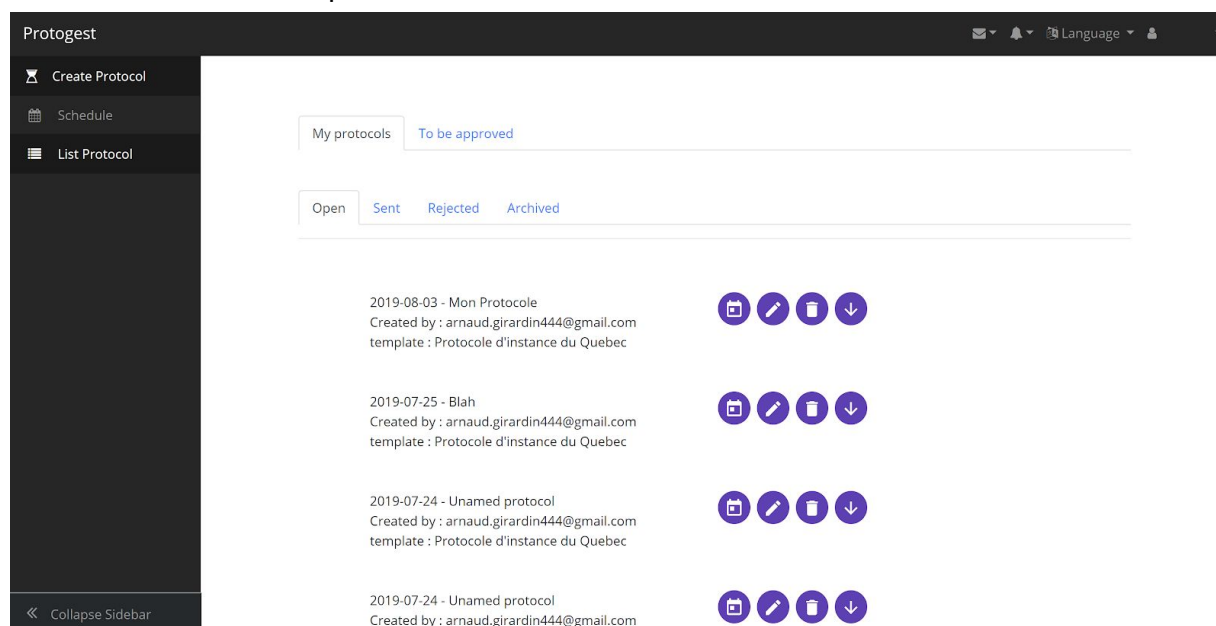


Figure 9 - Liste des protocoles modifier

Pour résoudre le problème, plusieurs options de refonte auraient pu être entreprises. Les listes auraient pu être divisées sur plusieurs pages ou bien il serait possible de créer une structure d'affichage divisant mieux les protocoles. Diviser les protocoles en deux pages auraient été redondant autant en termes de design que de fonctionnalité. Sinon, créer une nouvelle structure complète est inutile puisque toutes les fonctions d'affichage existaient déjà, sinon cela aurait aussi été demandant en termes de temps. Une refonte de la page était la meilleure option puisqu'elle permettait un affichage plus simple à utiliser et répondant plus au besoin du client.

Tout d'abord, l'affichage de la page a été divisé avec des onglets. De cette façon, il est plus facile savoir quels protocoles appartiennent à l'utilisateur. La page est donc divisée en deux grandes sections : celle des protocoles créés par l'utilisateur et ceux reçus par l'utilisateur.

Par la suite, des sous-onglets ont été rajoutés afin de pouvoir filtrer les protocoles selon leur statut. Ces onglets représentent principalement le cycle de vie du protocole, et seront expliqués plus en détail dans le tableau suivant.

| Statut | Description |
|----------|--|
| Pending | Après la création du protocole, celui-ci est automatiquement mis au statut 'pending'. Ceci est la seule fonctionnalité de statut qui est implémenté à ce jour. Représente que le protocole est créé, mais n'est pas terminé ou envoyé. |
| Actif | Le statut indiquera que le protocole est complété, envoyer et en attente d'acceptation du receveur. |
| Rejected | Le statut indiquera que le protocole envoyé a été refusé par le receveur et est en attente de nouvelle date. |
| Archived | Le statut indiquera que le protocole est finalisé et envoyé à la cour. Il apparaîtra dans la section archiver et permettra à l'utilisateur d'y avoir accès si celui-ci désire le consulter. |

Tableau 6 - cycle de vie d'un protocole

Il est important de noter que la fonctionnalité pour faire un changement permanent de statut n'est pas encore implémentée. Le changement se produit seulement au niveau de l'affichage pour le moment.

De plus une variable appelée «Approved» a été ajoutée à la base de données pour indiquer si un protocole possédant le statut «Active» est prêt à être envoyé ou non. Celui-ci indique si l'utilisateur qui a reçu un protocole a approuvé celui-ci en indiquant que toutes les dates des instances ne rentrent pas en conflit avec son horaire. Ainsi lorsque le protocole est approuvé il sera prêt à être envoyé. Pour le moment la fonctionnalité d'affichage qui montre que le protocole est en attente d'envoi ou approuvée est opérationnelle, mais le bouton pour l'envoi final à la cour ou la fonctionnalité d'acceptation ou refus de date n'est pas encore implémenté dans l'application. Le tout a été créé seulement pour donner l'impression de faire l'envoi final et mettre le statut à «Archived», afin que le client puisse comprendre la fonctionnalité. Bref, le tout est une première itération du cycle de vie et devra être modifié et ajusté pour les besoins futurs du client par la prochaine équipe travaillant sur le projet.

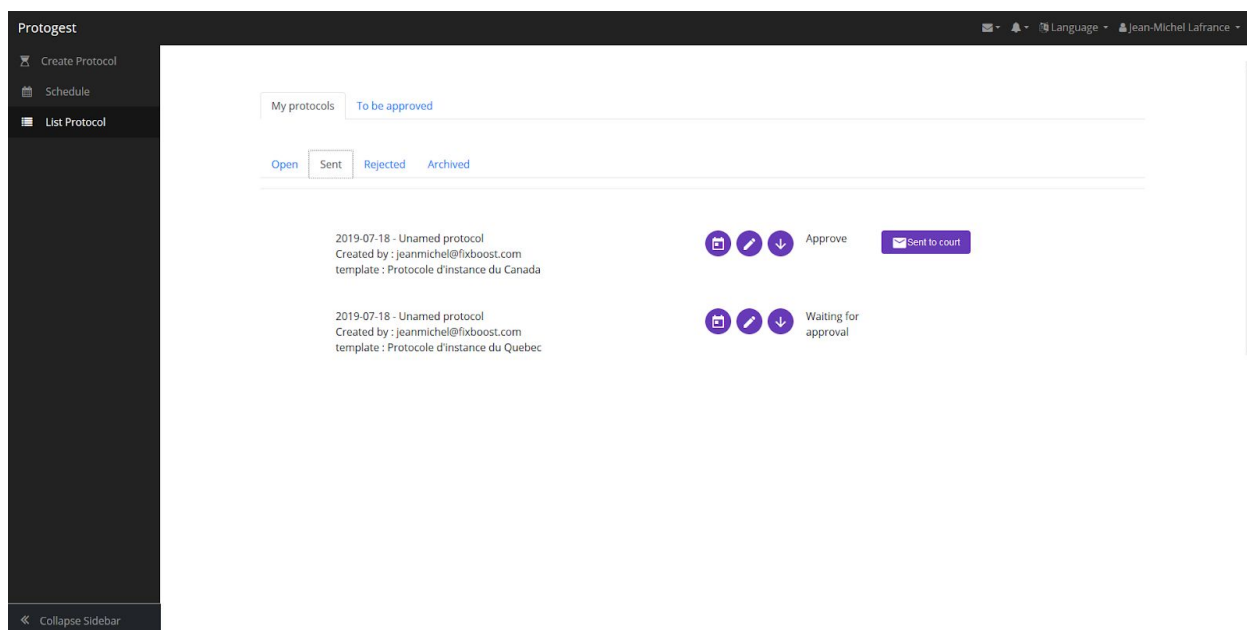


Figure 10 - section sent de la liste de protocol

L'affichage des informations de la liste de protocole a aussi été modifié. En effet, seulement le nom du gabarit était affiché auparavant ce qui rendait l'identification des protocoles difficile. Pour remédier à la situation, l'ajout d'autres informations pertinentes comme le nom du protocole, la date de création et l'utilisateur ayant créé le protocole fut indispensable. De plus, des noms plus précis furent donnés aux gabarits de protocole au lieu d'un nom de fichier pour ainsi afficher un nom plus représentatif. Finalement, la liste est triée selon la date de création, puis selon le nom du protocole.

Pour terminer, les boutons difficiles à voir et simples à confondre avec du texte ont été enlevés. Ils ont été remplacés par des boutons de couleur suivant le design de l'application et le texte a été remplacé par des icônes représentant le mieux possible la fonctionnalité associée au bouton.

Outil de résumé

Toujours dans l'esprit de rendre l'expérience du client plus agréable et simpliste, un outil permettant à l'utilisateur de voir un résumé de la liste de protocole a été ajouté à la barre de menu. Ainsi lorsque l'utilisateur accèdera la fonction en cliquant l'icône de cloche dans le menu un sous-menu apparaîtra indiquant le nombre de protocoles pour chaque statut l'attendant. De cette façon, l'avocat pourra voir rapidement dès sa connexion combien de protocoles sont en attente de réponse, acceptés ou refusés par un intervenant. Lui offrant aussi un raccourci simple et rapide à utiliser pour arriver directement sur la page.

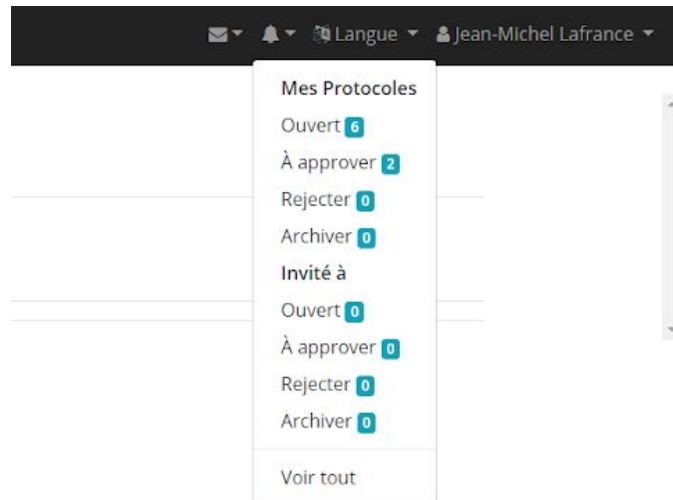


Figure 11 - outil de récapitulation de la liste des protocoles

Affichage d'un protocole

Le concept de cette page semble assez simple, cependant l'affichage ne fonctionnait pas lors de la réception du code source. En effet, lors de la première tentative de visualiser les dates d'un protocole d'instance en particulier, la seule chose visible était une page blanche comportant une liste vide d'événements.

La page a donc été réparée afin de permettre l'affichage du calendrier ainsi que les événements (dates sélectionnées) dans celui-ci. Un fois l'affichage fonctionnel, les membres de l'équipe se sont rendu compte que les événements du calendrier n'avaient pas de description précise permettant de les différencier. FullCalendar, le module de calendrier utilisé permet d'ajouter une description à un événement, cependant la seule information présente dans les données de protocole était le id de la section ainsi que la date. Une certaine logique a dû être écrite afin de permettre de trouver la description d'une section en fonction de son id dans le schéma (gabarit de protocole) parent. Une fois la description trouvée, il fut trivial de la faire afficher dans le calendrier comme il est possible de le voir sur la figure suivante.

Protogest

Créer un protocole

Calendrier

Protocole de liste

Réduire la barre latérale

<

>

today

August 2019

month

week

day

| Sun | Mon | Tue | Wed | Thu | Fri | Sat |
|-----|-----|---|--|-----|-----|-----|
| 28 | 29 | 30 | 31 | 1 | 2 | 3 |
| | | Referral to competent court or dismissal (C.C.P., a. 167) 8 | Other exception (with a reference to the C.C.P. article) 9 | | | |
| 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 11 | 12 | 13 | 14 | 15 | 16 | 17 |
| 18 | 19 | 20 | 21 | 22 | 23 | 24 |
| 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |

Events

8

Referral to competent court or dismissal (C.C.P., a. 167)

2019-07-30

9

Other exception (with a reference to the C.C.P. article)

2019-07-31

Figure 12 - Affichage d'un protocole

Backend

Architecture

Au début du projet, le backend se composait initialement de deux modules, soit les modules service et Entity. Le module Service s'occupe de la logique d'affaire de l'application, alors que Entity contient le modèle. Le but de maintenir deux modules était de partager la librairie Entity entre plusieurs service. Cependant le projet ne contient qu'un service, donc sépare les deux modules sans raison particulière et on ne fait qu'introduire plus de complexité au projet. Le projet est un Monolith, et l'architecture de microservices avait été mis au rebut par l'équipe de la dernière session.

Dès que le début du projet, plusieurs problèmes ont été observés avec le backend.

Base de données

Initialement le projet utilisait deux bases de données: MySQL et DynamoDB. Les deux bases de données géraient les mêmes informations. Ceci rendait leur travail redondant. De plus, la connexion vers DynamoDB ne s'était pas faite et le code était erroné, c.-à-d. le code comportait des problèmes sévères autant au niveau de la structure que de son utilisation.

Par ailleurs, il y avait plus de cinq requêtes SQL écrites à la main. Il est certain qu'il existe plusieurs opinions sur la valeur et l'utilité d'écrire du SQL à la main, certains pensent que ce n'est pas bon, alors que d'autres pensent que c'est normal de le faire. Pour le cas du projet, les équipes précédentes auraient pu éviter d'écrire du SQL et utiliser les méthodes déjà implémentées par JPA.

À titre d'exemple, une requête SQL a été écrite pour chercher les protocoles d'instances créées par un certain utilisateur. Cette requête aurait facilement été faite en utilisant l'API programmatiquement et ne requiert pas l'écriture de la requête à la main. Par ailleurs, cela représente un risque de sécurité et les développeurs doivent prendre cet aspect en compte lors de la maintenance du code.

Ensuite, le module Entity n'était utilisé que par le module Service. Puisqu'ils ont choisi une architecture Monolith pour le backend, ceci fait qu'un seul module, en l'occurrence le module Service, s'en sert. Ceci rend leur décision d'utiliser deux modules inutiles.

Vérification d'authentification:

Un autre problème que rencontré vient du fait que le backend ne vérifie jamais la validité de l'authentification des utilisateurs. En effet, le système d'authentification ne se résumait qu'à une seule variable dans le frontend qui détermine si un utilisateur est connecté ou non. Ceci n'est pas une mesure très sécuritaire, puisque si la validation est manquante, tout utilisateur inscrit ou pas pourrait facilement accéder à des ressources du site qui sont normalement réservées que pour les personnes authentifiées, par exemple, créer des protocoles, lister des protocoles, etc.

Code mal écrit et ne suit les bonnes lignes directrices

En outre, le code écrit ne suivait pas les bonnes lignes directrices établies par Spring boot, à titre d'exemples:

- Des Singletons étaient utilisés où des classes 'Component' auraient dû être utilisées.
- Aucune utilisation des 'Beans' n'a été faite, alors que le projet aurait bénéficié de leur utilisation.
- Beaucoup de configuration avait été hardcodé à plusieurs endroits dans le code.

Solutions et Architecture actuelle

En prenant en considération tous ces problèmes, plusieurs changements ont été apportés au backend tout en gardant certaines décisions faites durant les itérations antérieures.

Voici un diagramme représentant un résumé du backend final, avec les explications qui en suivent.

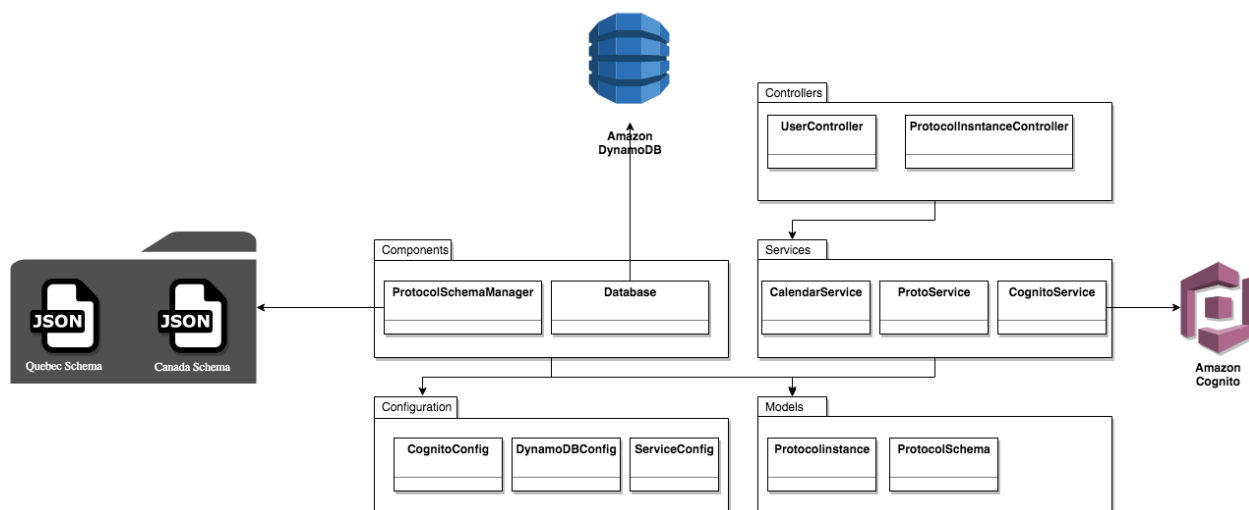


Figure 13 - Diagramme simplifié de l'architecture du backend

Monolith

Le backend a été continué en utilisant une structure selon le modèle de Monolith. L'idée d'utiliser des microservices a été explorée, mais avec le temps de développement et nombre de développeurs en backend, la conclusion fut que ce n'était pas optimal.

API Rest

Le backend et le frontend communiquent toujours en utilisant un REST API. Aucune des nomenclatures choisies par les anciennes équipes n'a été modifiée, mais les règles SOP ont été suivies pour créer les nouvelles requêtes.

Déplacement de Cognito vers le backend

Toute la logique de Cognito a été déplacée dans le backend, facilitant ainsi le contrôle du flux de connexion et d'inscription. La communication est établie avec Cognito à partir du backend et le front-end ne s'occupe que de sa communication avec le backend. Il y a une bien meilleure division des responsabilités. ([Plus d'informations](#))

Migrer vers DynamoDB

Durant le projet, la décision a été prise de garder une seule base de données et de n'utiliser que DynamoDB. En effet une interface de connexion vers Amazon AWS a été implémentée et une logique permettant de vérifier et créer les tables automatiquement. ([Plus d'informations](#))

Vérification de l'authentification

Une solution de vérifications des requêtes a été implémentée afin de faire vérification des utilisateurs demandant une ressource qui devrait être authentifiée. Ceci se fait par la vérification d'un token utilisée par Cognito. Si le token en question est valide, la ressource est délivrée à l'utilisateur. Cependant, si ce n'est pas valide, l'utilisateur est redirigé vers la page de connexion. ([Plus d'informations](#))

Ci-dessous est un diagramme simplifié illustrant la vérification faite par le backend.

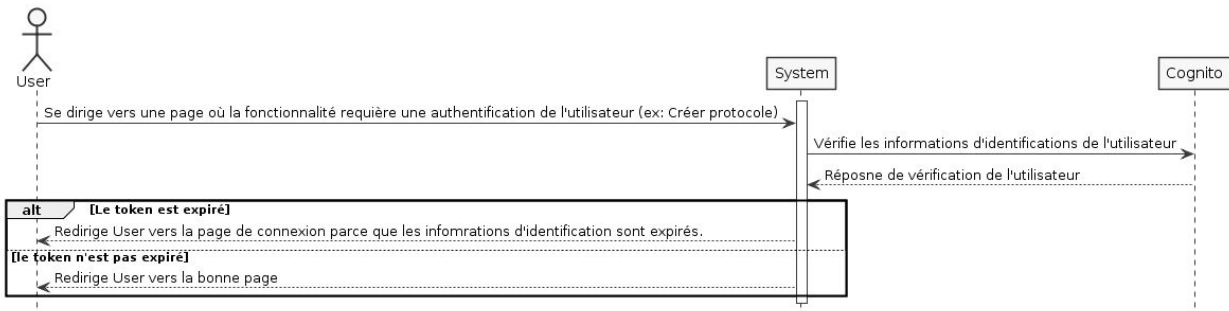


Figure 14 - Diagramme simplifié illustrant la solution de la base de données

Gestion des protocoles

La gestion des protocoles était virtuellement inexistante dans l'ancienne version. De plus, l'équipe d'avant avait écrit la plupart de la logique avec SQL à la main, ceci est un très mauvais choix et causant les problèmes suivants, c.-à-d. avec la migration à DynamoDB, toute la logique de gestion de protocoles a dû être faite.

Voici certaines tâches faites en rapport avec la gestion des protocoles :

- **Création d'un protocole:** Le frontend et le backend ont été modifiés afin de supporter le nouveau mécanisme de création de protocoles. Le protocole créé requiert deux

éléments : le Token de l'utilisateur et le contenu du rapport. Une vérification du token est faite au préalable pour s'assurer que l'utilisateur a le droit de créer un rapport. Ensuite, un rapport est créé et stocké dans la base de données avec les informations envoyées par le frontend. Finalement, tous les utilisateurs invités sont notifiés avec un système d'envoi de courriels.

- **Lister les protocoles:** Le backend expose une API Rest permettant de lister tous les protocoles d'un utilisateur. Un utilisateur peut être lié à plusieurs protocoles et il peut être le créateur ou un invité. Les deux requêtes sont différentes, mais seulement au niveau du backend où la complexité est abstraite du frontend. Pour avoir une liste de protocoles d'un utilisateur, le frontend communique avec le backend en ajoutant le token de l'utilisateur dans le *header*. Le backend retourne un tableau avec les protocoles d'instances dans le format de JSON. Chaque protocole d'instance retourné comporte une indication permettant de déterminer s'il est entamé, archivé, etc.
- **Invitation de membres aux protocoles:** Avec cette itération, il est maintenant possible d'inviter plusieurs membres lorsqu'on crée un protocole. Le frontend envoie les informations du protocole, le *token* ainsi que les courriels des personnes à inviter. Après avoir vérifié la véracité du *token* et l'authentification du client, le backend itère sur les courriels des invités et leur envoie une invitation. Les invités sont enregistré dans la base de données dans le protocole d'instance créer. De ce fait, un frontend peut aussi demander de voir les protocoles où un utilisateur a été invité.

Stocker les schémas de protocoles d'instance dans DynamoDB

À l'état initial, les schémas de protocoles sont directement stockés et transmis dans le frontend. Ceci fait que le premier chargement de la page est plus long et les changements faits requièrent un redéploiement du frontend. Un déplacement des schémas dans DynamoDB fut nécessaire et une implémentation d'une requête REST API permettant de chercher et retourner les schémas de protocoles du backend.

Au niveau du backend, un mécanisme permettant d'itérer sur les fichiers locaux d'un dossier fut implémenté. Ces fichiers doivent être du type JSON et le système vérifie qu'ils ont été créés dans la base de données. Cette vérification est faite à chaque démarrage du backend.

Voici une illustration du mécanisme implémenté au backend.

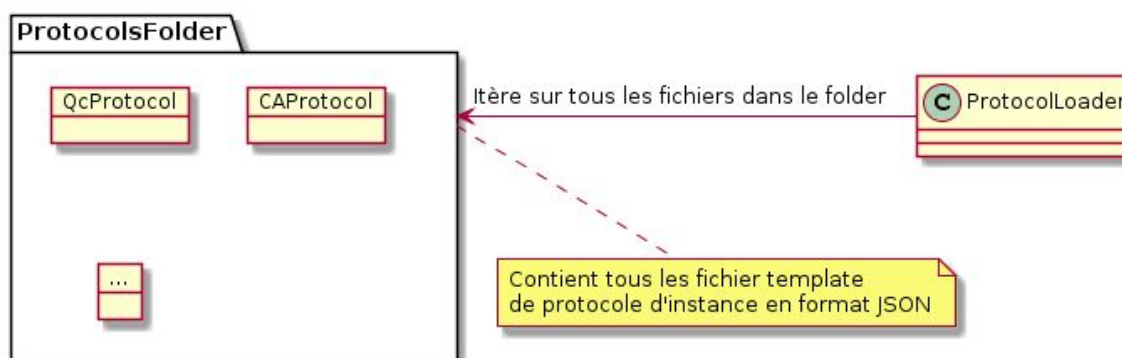


Figure 15 - Diagramme simplifié illustrant la solution de la base de données

Finalement, à la fin du projet, un backend pleinement fonctionnel a été conçu. Des changements majeurs furent implémentés à l'architecture. Des solutions plus simple et facilement maintenable furent conceptualisées, rendant le backend plus utilisable.

Base de données

Au début du mandat, le projet comportait deux bases de données: MySQL et DynamoDB. Dans plusieurs projets, le fait d'avoir deux bases de données est justifiable si chacune des bases de données avait une responsabilité différente de l'autre. Par contre, les deux bases de données utilisées dans l'application se partageaient la même responsabilité. Par ailleurs, la connexion vers la base de données DynamoDB ne se faisait pas et tout le travail se faisait dans une 'sandbox'. De plus, il y avait un grand nombre de SQL écrit à la main dans l'application, ce qui rendait la migration vers une base de données NoSQL encore plus difficile.

Compte tenu de ces faits, une seule base de données fut choisie pour migrer toute notre logique de stockage dans celle-ci. La base de données choisie fut DynamoDB, service offert par Amazon AWS, parce qu'elle utilise du NoSQL ce qui constitue un avantage certain pour le stockage des protocoles d'instance, c.-à-d. les protocoles d'instances peuvent différer dans leur structure d'un protocole à l'autre. Ceci offre une structure difficile à répliquer en base de données normalisée, c'est pourquoi utiliser une dénormalisée répondait plus au besoin du projet.

La migration a été faite, mais cela ne c'est pas fait sans problème, surtout lors du transfert du code SQL écrit dans les versions antérieures.

Au niveau de l'architecture, un mécanisme permettant d'enregistrer des tables et de les vérifier à chaque démarrage de l'application fut implémenté. Pour ce faire, une classe *Component* (classe Database), qui s'initialise au démarrage de Spring Boot, itère sur les tables à vérifier et applique la stratégie de vérification.

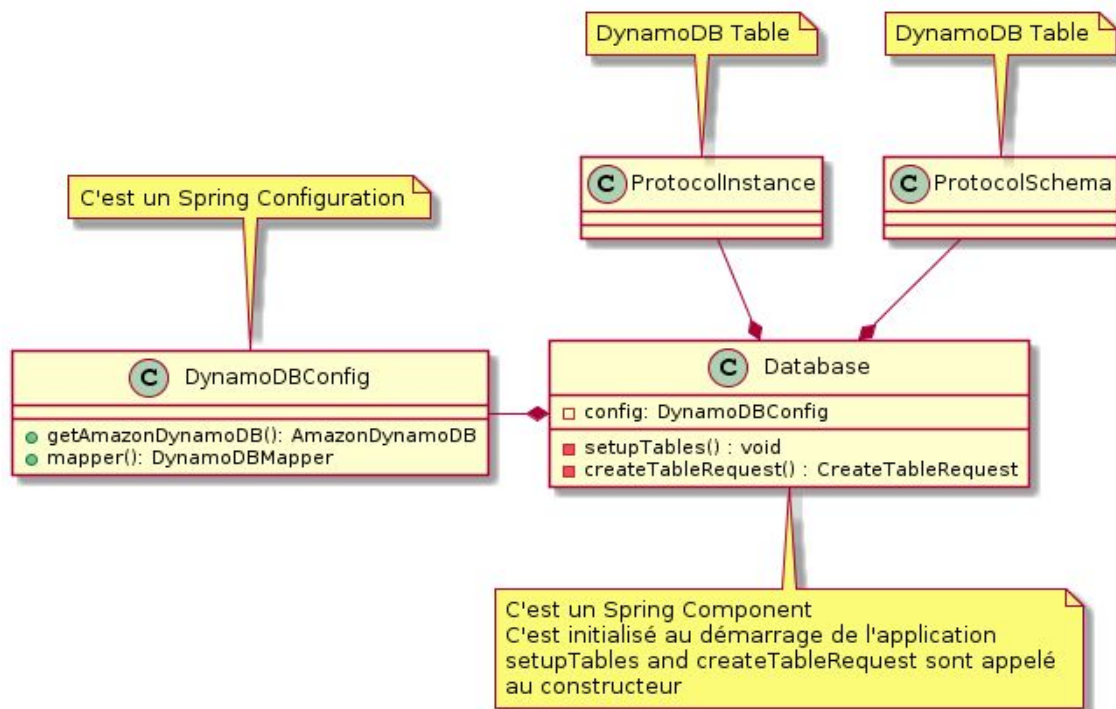


Figure 16 - Diagramme simplifié illustrant la solution de la base de données

Cette architecture est facilement maintenable et permet une modification futures simple puisque la vérification de toutes les tables est centralisé dans la classe Database.

Cognito

Transfert cognito vers le backend

Cognito avait été utilisé durant la dernière itération et l'application permettait déjà l'inscription et la connexion des utilisateurs. Cependant, durant les réunions hebdomadaires, des discussions ont été tenues sur les changements pouvant être apportés à l'architecture pour la rendre plus solide et maintenable. Plusieurs idées en sont ressorties, mais seulement certaines d'entre elles ont été choisies pour investiguer.

Les deux idées sont :

- Idée 1: Déplacer la logique de Cognito vers le backend (et l'enlever du frontend)
- Idée 2: Utiliser la page de connexion/Inscription qu'Amazon offre directement.

Solution impéementée

Après investigation, une décision a été prise pour déplacer Cognito au backend pour deux raisons:

- Premièrement, en mettant l'authentification et l'inscription directement au frontend, la chances est manquée de pouvoir contrôler le cas d'utilisation de l'authentification. Par exemple, enregistrer le temps auquel un utilisateur s'est inscrit ou s'est connecté serait une fonction intéressante. En laissant la connexion et inscription au frontend, ce contrôle du flux de données devra être abandonné.
- Deuxièmement, le fait d'avoir Cognito en frontend ne constitue pas une faille de sécurité de soi. Cependant, le PoolId et clientId sont exposés directement à tous les utilisateurs, laissant la possibilité qu'un utilisateur malveillant utilise les mêmes données d'authentification sur un autre site. Ceci permettrait à des utilisateurs de s'inscrire dans Cognito sans même visiter le site web.

La première raison a aidé la décision, mais c'est la deuxième raison qui a renforcé la décision finale. Par ailleurs, pour ce qui est de l'option d'utiliser les pages faites par Amazon, cela ne constituerait pas une solution pour le contrôle du flux d'inscription et de connexion, ce n'est donc pas une option qui a été retenue.

Ci-dessous se trouve un exemple représentant le système où on peut visualiser le flux de données qui passe du frontend vers le backend vers Cognito.

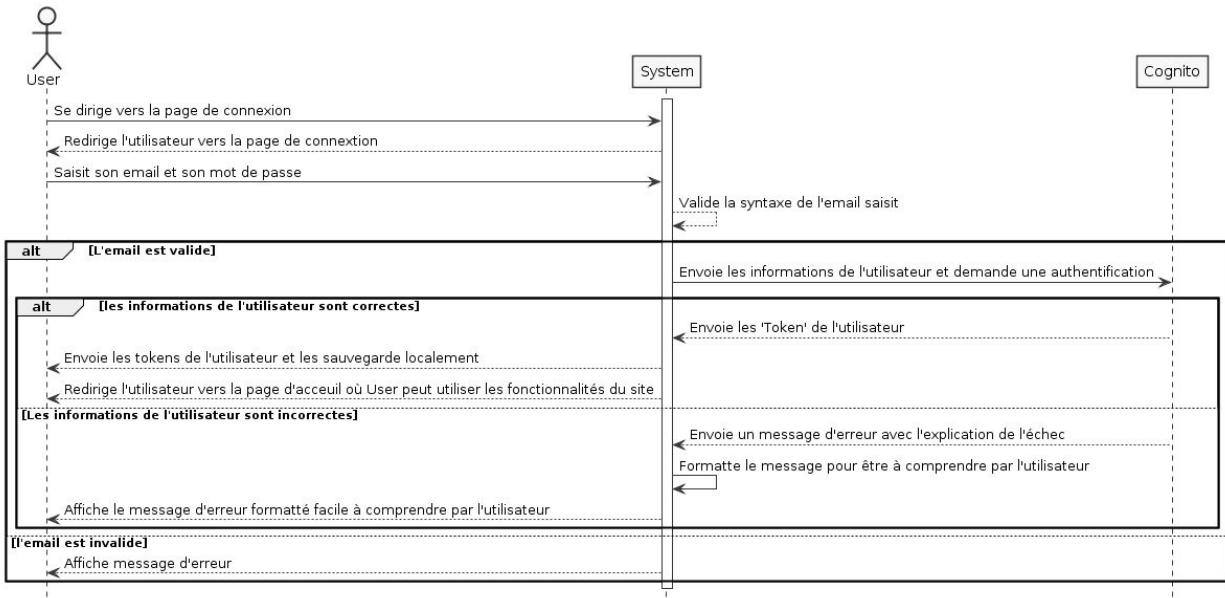


Figure 17 - Diagramme simplifié illustrant la solution de la base de données

Validation des tokens

Dans l'itération précédente, la vérification des requêtes des utilisateurs était virtuellement inexistante. Une seule variable au frontend permettait de 'connecté' un utilisateur (Cette variable est facilement changeable par un utilisateur en utilisant n'importe quel navigateur internet, ex Firefox ou Google Chrome) et les tokens générés au niveau d'Amazon n'étaient pas utilisés.

De plus les informations de l'utilisateur étaient hardcodé lors de la création d'un protocole; par exemple, le courriel était toujours 'test@test.com' quel que soit l'utilisateur connecté. Tout ceci fait en sorte qu'un utilisateur qui n'a jamais été inscrit sur le site pouvait non seulement accéder aux pages réservés aux utilisateurs authentifiés, mais aussi créer des protocoles et envoyer des requêtes sur les protocoles au backend sans même être inscrit et les étapes pour le faire était extrêmement facile.

Solution implémentée

Avec la nouvelle architecture, un mécanisme de vérification a été implémenté permettant de s'assurer que les requêtes sont bien faites par des utilisateurs enregistrés et connecté. Si un utilisateur non authentifié fait une requête d'une page qui requiert une authentification, l'utilisateur est redirigé vers la page de connexion. Par ailleurs, chaque requête pour lister les protocoles ou créer les protocoles est accompagnée par le token de l'utilisateur. Si le token est manquant, non valide ou expiré, l'utilisateur est redirigé vers la page de connexion.

Oubli de mot de passe et Profil

Au cours de l'analyse du frontend, un manque crucial de fonctionnalité pour la qualité de vie de l'utilisateur apparaissait lors de l'utilisation de l'application. Par exemple, l'application ne pouvait pas modifier ou récupérer le mot de passe d'un utilisateur, après la création. Ceci est un problème pour toutes les futures versions surtout lorsque des clients pourront commencer à l'utiliser puisque beaucoup d'entre eux risquent de perdre leur mot de passe. Ainsi, une nouvelle fonctionnalité devra être créée offrant au client un moyen de pouvoir facilement effectuer ces actions

Solution impémenteée.

Dans un premier lieu, une nouvelle page fut créée permettant à l'utilisateur de changer son mot de passe lorsque celui-ci l'oublie. Celle-ci utilise les fonctionnalités de Cognito se trouvant dans le backend en envoyant toutes les données nécessaires au courriel du compte. Ceci permet à l'utilisateur de récupérer un code à partir de son courriel qu'il pourra par la suite utiliser pour modifier son mot de passe.

Dans un deuxième temps, lorsque l'avocat sera connecté à la plateforme il aura besoin d'une fonction afin de modifier ces informations personnelles ou son mot de passe. Pour remédier à cela une nouvelle page, Profil, fut créée permettant au client de changer son nom, prénom ou mot de passe. Pour pouvoir effectuer le changement de mot de passe, l'utilisateur doit simplement entrer son ancien mot de passe, puis mettre un nouveau avec sa confirmation. Le système va valider les informations puis envoyer une requête vers le backend qui va communiquer avec Cognito pour faire le changement final.

Ajout des informations utilisateur

Dans la page d'inscription, l'application demandait aux utilisateurs de saisir leur nom et prénom, mais ne faisait rien avec ces informations après. Peut-être que les équipes précédentes ne savaient pas comment enregistrer ces éléments dans Cognito.

Solution impémenteée

Pour cette raison, une fonctionnalité a été ajoutée dans le backend afin d'enregistrer le nom et prénom des utilisateurs lors de leur inscription. Par la suite, le frontend peut avoir accès aux noms et prénom en faisant un appel à une fonctionnalité du backend pour récupérer ceux-ci.

Ces informations n'étaient pas utilisées originellement par le frontend ou le backend seulement le courriel était nécessaire afin de lier les protocoles au bon utilisateur. Par contre, après certaines modifications dans le frontend il était maintenant possible de voir dans l'en-tête de la page et dans la liste de protocole le nom complet de l'avocat. Pour ce qui est de l'en-tête, les changements servent à personnaliser l'application. Alors que pour la liste de protocole ces changements servent à indiquer au client de qui provient le protocole qu'il a reçu.

Discussions

Angular vs ReactJS

Dès le début du projet, une décision devait être prise entre les cadres React et Angular pour l'implémentation du frontend. Heureusement, une implémentation pour chacun des cadres existait déjà permettant ainsi d'orienter le choix. Après avoir parcouru le code ainsi que discuté des options, le choix fut pris de continuer avec Angular au lieu de React. Plusieurs raisons ont motivé ce choix technologique, mais la raison principale est que le code Angular était beaucoup plus avancé que celui fait en React. En effet, la version Angular étant plus récente, elle avait beaucoup de fonctionnalités déjà implémentées et la structure du code permettait d'ajouter de nouvelles fonctionnalités sans trop de problèmes. De plus le but principal du projet était l'avancement de celui-ci et non recommencer le frontend une autre fois, ceci aurait été contre-productif.

Ceci dit, ce fut aussi l'un des plus grands regrets de l'équipe puisque même si l'application existait déjà en Angular, le manque de connaissance et les limitations d'Angular ont démontré que de recommencer l'application en React aurait été un meilleur choix pour notre situation.

MySQL vs DynamoDB

MySQL est une base de données relationnelle, alors que DynamoDB est une base de données NoSQL utilisant des documents pour stocker les données. Dans ce cas, une base de données non relationnelle est beaucoup plus intéressante, parce qu'une ressource de protocole d'instance peut être différente de toutes les autres. Ceci favorise DynamoDB grandement parce que ses tables ne requièrent pas que les données suivent un schéma particulier. Par ailleurs, un protocole d'instance est une entité dont les données ne dépendent pas réellement d'autres, ce qui fait que la présence de liens vers d'autres tables est inutile. Ceci favorise aussi l'utilisation de DynamoDB au lieu de MySQL. Finalement, bien que l'hébergement de base de données n'était pas pris en compte par l'équipe, le coût de DynamoDB est bas par rapport à celui de MySQL. En conclusion, l'application sera construite utilisant DynamoDB, car les tables ne requièrent pas de schéma, aucun lien entre les tables (joins) n'est nécessaire et le coût de maintenance de la base de données est plus bas.

Problèmes rencontrés

Pour commencer, le plus gros problème rencontré pendant la durée du projet est le manque de connaissance dans la technologie Angular. Au début du mandat, une décision devait être prise entre utiliser le langage React ou Angular pour continuer la plateforme. Après discussion entre les membres de l'équipe, la décision a été prise de continuer avec le code Angular qui avait eue deux itérations à ce point. Ceci était plus efficace que de prendre la vieille version en React qui aurait dû être modifiée afin que tout soit mis à jour. Par contre, le manque de connaissance du groupe et les limitations du langage ont ralenti l'avancement du projet.

Par la suite, les problèmes de structure et les changements apportés à la base de données dans le backend n'étaient pas prévus et on prit un certain temps à résoudre pour créer une version solide et efficace. Ceci était un changement pour le mieux. Par contre, ce changement a causé certains délais qui ont dû être résolus dans l'application avant de commencer à implémenter de nouvelles fonctionnalités à celle-ci.

Par ailleurs, au niveau du backend, l'architecture était incomplète et le code ne suivait pas les lignes directrices du cadrage (Singleton, écriture de SQL à la main, etc.). La compréhension de cette architecture fut difficile, mais nécessaire pour changer et améliorer le backend avec la nouvelle architecture.

Ensuite, au cours du projet, une fonctionnalité qui a été considérée fut de garder une sauvegarde des données de session du serveur Tomcat dans la base de données DynamoDB. Après discussions avec Mathieu le spécialiste AWS, un module répondant au besoin de la fonctionnalité a été mis de l'avant. Par contre, le plug-in n'était plus supporté par l'équipe de développement à cause de problèmes de compatibilité avec Tomcat 8.5. Ainsi il sera impossible d'entreposer les sessions du serveur comme nous voulions pour aller vers serveurs "state-less". Donc, le manque de temps pour trouver une solution pour résoudre le problème fait en sorte que le problème devra être résolu dans une prochaine session.

Finalement, un des plus grands problèmes que l'équipe a vécu au cours de la session est le départ d'un de ces membres, Xavier Reid, celui-ci a dû quitter en milieu de session. Le départ a été fait de façon soudaine, mais malgré les problèmes que cela a causés les membres de l'équipe ont su réagir à la situation de façon calme et efficace. Les tâches assignées à celui-ci ont été divisées par l'équipe puis assignées aux personnes concernées, ne causant pas de retard majeur. Cela a par contre réduit la main-d'oeuvre et l'espérance de progression finales.

Modifications futures

Ce projet étant en constante évolution, plusieurs fonctionnalités restent à être développées. Dans la section, frontend plusieurs fonctionnalités de prise d'erreur ont été mises en place afin d'afficher à l'utilisateur si une erreur arrive ou non à l'exécution. Par contre, chacune d'entre elles utilisait l'affichage d'erreur déjà existante sur la page. Une des améliorations à faire serait de standardiser l'affichage de ces erreurs.

Comme expliqué plus haut, les enregistrements des sessions utilisateurs sont sauvegardés à même le serveur et une migration pour les sauvegarder dans la base de données DynamoDB a été considéré pour rendre le serveur "state-less". Par contre, le plugin pour effectuer cette manoeuvre sur un serveur Tomcat vers DynamoDB n'est plus soutenu. Il existe plusieurs solutions différentes, comme changer de serveur (ex.: Netty) ou encore changer le service de stockage de données (ex.: MongoDB, Elasticache).

Par la suite, dans les fonctions les plus importantes pour l'utilisateur, se trouve l'édition d'un protocole. Dans la version actuelle, un bouton existe pour éditer un protocole déjà existant, mais la fonction ne marche pas encore. Étant une fonction très importante pour l'utilisateur, celle-ci devrait être un des premiers mandats de la prochaine équipe.

De plus la gestion du cycle de vie du protocole n'a pas été implémentée. Ceci inclut les fonctionnalités suivantes : compléter et envoyer un protocole aux intervenants, accepter les dates proposées, refuser les dates proposées et finalement l'envoi du protocole final qui archivera le protocole.

Pour ce qui est de la fonction d'envoi final, celle-ci n'est pas implémentée dans la version actuelle. La fonctionnalité devra permettre d'envoyer un courriel à tous les participants indiquant que le protocole est prêt à l'envoi avec une copie PDF du document final. Dans la même optique, une fonctionnalité pour télécharger le formulaire final en version PDF doit être implémentée sur le bouton télécharger déjà existant dans la liste de protocole. Finalement, une fonctionnalité de signature en ligne pour compléter le protocole pourrait intéresser le client.

Aussi à travers le projet un nouveau concept a été conceptualisé afin de permettre à plus d'un intervenant d'être invité durant la création. Avec cette nouvelle fonctionnalité, certains des invités pourront collaborer à la complétion du protocole d'instance. Ainsi des rôles devront être assignés aux différents utilisateurs afin de les distinguer et de leur offrir certaines options tout en fonction de leurs droits. Par exemple, le droit d'éditer un protocole en cour de création ou de modifier les dates d'invitations.

Finalement, il serait pertinent permettre d'importer les dates d'un protocole terminer dans le calendrier Outlook de l'utilisateur et des invités pour éviter que ceux-ci aient à les rentrer manuellement dans leur agenda.

Enjeux Environnementaux

Notre projet permet une collaboration entre des avocats sur des protocoles d'instances. Protogest permet de faire cette collaboration virtuellement ce qui fait en sorte que les avocats impriment et utilisent moins de papiers que lorsqu'ils n'utilisent pas Protogest. La diminution de la consommation de papiers a un impacte très positif sur notre environnement et notre planète.

Cependant, la cour n'accepte toujours pas des versions électroniques des protocoles d'instances. Par contre, selon nos informations, la cour s'apprête à implémenter un système informatique dans un avenir proche. Lorsque cela se fera, une des fonctionnalités que l'on aimerait ajoutée au projet est l'envoi de la version finale du protocole d'instance à la cour sans l'imprimer. Ceci ferait certainement économies sur encore plus de papiers.

CONCLUSION

Pour conclure, le projet fut un succès et le client est très satisfait des avancements qu'il a pu voir. Une certaine partie du projet a dû être consacrée à l'amélioration du code et renforcement des fonctionnalités déjà existantes. Par contre, celui-ci en ressort plus fort et mieux construit offrant une meilleure plate-forme pour toutes les équipes futures qui développeront sur le projet. De plus, le départ soudain d'un membre de l'équipe à cause des inconvénients et délais qui ont ralenti le projet. Par contre, cela a renforcé l'équipe qui a su démontrer sa force lors d'une situation hors de son contrôle. Finalement, le changement de direction des besoins a nécessité une amélioration du contenu de la plate-forme pour mieux répondre au client. Malgré les problèmes encourus, des nouveautés au niveau du fonctionnement de l'application ont su être apportées.

Le projet a permis à tous les membres de l'équipe d'en apprendre davantage sur le cadriciel Angular, Spring Boot ainsi que sur les services offerts par Amazon. De plus, le projet a donné la chance aux membres de l'équipe de travailler pour un vrai client offrant ainsi plusieurs défis intéressants à découvrir et surmonter. Par ailleurs, ce projet aura aussi permis de développer l'esprit critique des membres de l'équipe ainsi que de renforcer leur capacité à prendre les bonnes décisions, et ce, dans un court délai.

Au niveau du Frontend, l'équipe a su maintenir le code hérité et développer plusieurs nouvelles interfaces à utiliser. Au Backend, l'équipe a solidifié la structure avec une nouvelle architecture plus robuste et plusieurs nouvelles fonctionnalités ont été ajoutées.

Bref, la version actuelle de Protogest n'est pas la dernière, mais les modifications apportées durant la session permettront au projet de continuer dans la bonne direction. Grâce aux changements apportés, il est fort probable que la prochaine équipe n'ait qu'à se concentrer sur les fonctionnalités au lieu de faire du réusinage.

RÉFÉRENCES

- [1] Louisa(2018), Utilisation de sessions avec DynamoDB
<https://forums.aws.amazon.com/thread.jspa?threadID=275425>
- [2] Henri Yandell (2018), Amazon DynamoDB Session Manager pour Apache Tomcat
<https://github.com/amazon-archives/aws-dynamodb-session-tomcat>
- [3] Phillip Webb, Dave Syer, Josh Long, Stéphane Nicoll, Rob Winch, Andy Wilkinson, Marcel Overdijk, Christian Dupuis, Sébastien Deleuze, Michael Simons, Vedran Pavić, Jay Bryant, Madhura Bhavé (2019), Documentation Spring boot
<https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/>
- [4] Martin Grotzke(2018), memcached gestionnaire de session
<https://github.com/magro/memcached-session-manager>
- [5] Amazon (2019), Ressources Amazon pour développeurs
<https://aws.amazon.com/cognito/dev-resources/>
- [6] Amazon (2019) Exemple DynamoDB en Java
<https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/CodeSamples.Java.html>
- [7] Jean-Michel Lafrance, Arnaud Girardin et Yanis Sofiane Moussaoui (2019), Document de vision du projet
<https://drive.google.com/file/d/12WqLEAtv1Bx86BCT7C6uOqDb21r1f1z1/view?usp=sharing>
- [8] Google (2019), Documentation Material Design Angular
<https://material.angular.io/>
- [9] FullCalendar (2019), Documentation de FullCalendar
<https://fullcalendar.io/docs>
- [10] Google (2019), Documentation d'Angular
<https://angular.io/docs>

Annexe 1 -

Cas d'utilisations

| Cas d'utilisation | Contenu | |
|--|--|--|
| CU01 - Inscription de l'utilisateur | L'utilisateur peut créer un nouveau compte dans le système. | |
| | Pré-conditions | Avoir un accès à un courriel valide |
| | Flux de base | <ol style="list-style-type: none"> 1. L'utilisateur arrive sur la page Login, clique sur le bouton s'enregistrer 2. Rempli le formulaire d'enregistrement 3. Clique sur le bouton 'Register' 4. L'utilisateur est retourner vers la page de connection ou celui-ci peut entrer les informations de connection 5. Lors de la première connection |
| | Post-conditions | L'utilisateur peut maintenant se connecter sur l'application |
| CU02 - Connexion de l'utilisateur | L'utilisateur peut se connecter en utilisant ses informations d'identification | |
| | Pré-conditions | L'utilisateur est inscrit (CU01) |
| | Flux de base | <ol style="list-style-type: none"> 1. L'utilisateur arrive à la page login 2. Saisit son courriel et son mot de passe 3. L'utilisateur clique sur le bouton 'Login' 4. L'utilisateur est dirigé vers la page listant tous les protocoles. |
| | Post-conditions | L'utilisateur est redirigé dans l'application avec accès au information de son compte. |
| CU03 - Récupération d'un mot de passe oublié par l'utilisateur | L'utilisateur peut récupérer son mot de passe si celui-ci à oublier l'original | |
| | Pré-conditions | L'utilisateur est inscrit (CU01) |
| | Flux de base | <ol style="list-style-type: none"> 1. L'utilisateur arrive à la page de login 2. Clique sur le bouton 'forget password' 3. L'utilisateur saisit son courriel |

| | | |
|--|--|--|
| | | <ol style="list-style-type: none"> 4. Cliquez sur le bouton 'Send' 5. L'utilisateur reçoit un courriel contenant un code 6. L'utilisateur saisit le nouveau mot et le code reçu par courriel sur la page 'forget password', ou de nouvelle section son maintenant afficher 7. Cliquez sur le bouton 'Change password' 8. L'utilisateur est redirigé vers la page de login |
| | Post-conditions | L'utilisateur peut maintenant accéder à la plateforme en utilisant le nouveau mot de passe |
| | | |
| CU04 - Vérification de l'authentification et redirection | L'utilisateur doit être authentifié afin de visiter certaines pages. | |
| | Pré-conditions | L'utilisateur est connecté (CU02) |
| | Flux de base | <ol style="list-style-type: none"> 1. L'utilisateur demande une page qui requiert une authentification, e.g Page de listes des protocoles 2. Le système vérifie les informations d'identification de l'utilisateur 3. L'utilisateur est dirigé vers la page demandée |
| | Post-conditions | |
| CU05 - Modifier ces informations de profil | L'utilisateur peut changer ces informations personnelle après être connecter au frontend | |
| | Pré-conditions | L'utilisateur est connecter à la plateforme (CU02) |
| | Flux de base | <ol style="list-style-type: none"> 1. L'utilisateur clique sur le bouton 'Profil' dans le menu 2. La page de modification de profile s'affiche 3. L'utilisateur saisit les nouvelles informations 4. Cliquez sur le bouton 'Save' 5. Les nouvelles informations de l'utilisateur sont sauvegardés |

| | | | | | | | | | |
|--|---|--|---|----------------|--|--------------|--|-----------------|---|
| | <table> <tr> <td>Post-conditions</td><td>L'utilisateur à changer ces informations personnelles, celle-ci apparaisse différente dans l'en-tête de l'application</td></tr> </table> | Post-conditions | L'utilisateur à changer ces informations personnelles, celle-ci apparaisse différente dans l'en-tête de l'application | | | | | | |
| Post-conditions | L'utilisateur à changer ces informations personnelles, celle-ci apparaisse différente dans l'en-tête de l'application | | | | | | | | |
| CU06 - Modifier le mot de passe | <table> <tr> <td colspan="2">L'utilisateur peut changer son mot de passe pour un nouveau après s'être connecter à la plateforme</td></tr> <tr> <td>Pré-conditions</td><td>L'utilisateur est connecter à la plateforme (CU02) Connaître le mot de passe original</td></tr> <tr> <td>Flux de base</td><td></td></tr> <tr> <td>Post-conditions</td><td>L'utilisateur peut se connecter avec son nouveau mot de passe</td></tr> </table> | L'utilisateur peut changer son mot de passe pour un nouveau après s'être connecter à la plateforme | | Pré-conditions | L'utilisateur est connecter à la plateforme (CU02) Connaître le mot de passe original | Flux de base | | Post-conditions | L'utilisateur peut se connecter avec son nouveau mot de passe |
| L'utilisateur peut changer son mot de passe pour un nouveau après s'être connecter à la plateforme | | | | | | | | | |
| Pré-conditions | L'utilisateur est connecter à la plateforme (CU02) Connaître le mot de passe original | | | | | | | | |
| Flux de base | | | | | | | | | |
| Post-conditions | L'utilisateur peut se connecter avec son nouveau mot de passe | | | | | | | | |
| CU07 - Lister les protocoles d'un utilisateur | <table> <tr> <td colspan="2">L'utilisateur peut voir la liste des protocoles d'instance reliés à son compte autant ceux reçu que envoyé</td></tr> <tr> <td>Pré-conditions</td><td>L'utilisateur est connecter à la plateforme (CU02) Les informations d'authentications de l'utilisateur sont valide (CU04)</td></tr> <tr> <td>Flux de base</td><td> <ol style="list-style-type: none"> 1. L'utilisateur clique sur le bouton 'List' dans le menu de gauche 2. Les protocoles que l'utilisateur a créer sont affiché dans un onglet 3. Les protocoles dont l'utilisateur a été invité pour participer s'affiche dans un autre onglet 4. Les protocoles qui sont archivés s'affiche sur un autre onglet. </td></tr> <tr> <td>Post-conditions</td><td>Les protocoles sont affichés dans la page.</td></tr> </table> | L'utilisateur peut voir la liste des protocoles d'instance reliés à son compte autant ceux reçu que envoyé | | Pré-conditions | L'utilisateur est connecter à la plateforme (CU02) Les informations d'authentications de l'utilisateur sont valide (CU04) | Flux de base | <ol style="list-style-type: none"> 1. L'utilisateur clique sur le bouton 'List' dans le menu de gauche 2. Les protocoles que l'utilisateur a créer sont affiché dans un onglet 3. Les protocoles dont l'utilisateur a été invité pour participer s'affiche dans un autre onglet 4. Les protocoles qui sont archivés s'affiche sur un autre onglet. | Post-conditions | Les protocoles sont affichés dans la page. |
| L'utilisateur peut voir la liste des protocoles d'instance reliés à son compte autant ceux reçu que envoyé | | | | | | | | | |
| Pré-conditions | L'utilisateur est connecter à la plateforme (CU02) Les informations d'authentications de l'utilisateur sont valide (CU04) | | | | | | | | |
| Flux de base | <ol style="list-style-type: none"> 1. L'utilisateur clique sur le bouton 'List' dans le menu de gauche 2. Les protocoles que l'utilisateur a créer sont affiché dans un onglet 3. Les protocoles dont l'utilisateur a été invité pour participer s'affiche dans un autre onglet 4. Les protocoles qui sont archivés s'affiche sur un autre onglet. | | | | | | | | |
| Post-conditions | Les protocoles sont affichés dans la page. | | | | | | | | |
| CU08 - Créer un protocole d'instance | <table> <tr> <td colspan="2">L'utilisateur peut pouvoir créer un protocole d'instance avec les dates qu'il a choisit</td></tr> <tr> <td>Pré-conditions</td><td>L'utilisateur est connecter à la plateforme (CU02)</td></tr> </table> | L'utilisateur peut pouvoir créer un protocole d'instance avec les dates qu'il a choisit | | Pré-conditions | L'utilisateur est connecter à la plateforme (CU02) | | | | |
| L'utilisateur peut pouvoir créer un protocole d'instance avec les dates qu'il a choisit | | | | | | | | | |
| Pré-conditions | L'utilisateur est connecter à la plateforme (CU02) | | | | | | | | |

| | | |
|---------------------------------------|--|---|
| | | Les informations d'authentications de l'utilisateur sont valide (CU04) |
| | Flux de base | <ol style="list-style-type: none"> 1. L'utilisateur clique sur le bouton 'Create Protocol' dans le menu de gauche 2. L'utilisateur choisi le type de protocole qu'il veut créer 3. Clique sur le bouton 'Next' 4. L'utilisateur sélectionne les sections du protocoles qu'il veut remplir 5. Clique sur le bouton 'Next' 6. L'utilisateur saisit les dates dans une section 7. Clique 'Next' et refait l'étape 6 et 7 jusqu'à ce qu'il n'y a plus de section 8. L'utilisateur saisit l'email de toutes les personnes qu'il souhaite inviter 9. Utilisateur clique sur 'Save' |
| | Post-conditions | Un nouveau protocole d'instance est créé et lié au compte de l'utilisateur |
| CU09 - Afficher les conflits de dates | L'utilisateur peut récupérer les dates occupées de son calendrier personnelle | |
| | Pré-conditions | L'utilisateur est connecter à la plateforme (CU02) |
| | Flux de base | <ol style="list-style-type: none"> 1. L'utilisateur se rend sur la page de création de protocole 2. Le système va demander la permission de l'utilisateur pour récupérer les dates relire à son compte 3. Les dates occupées sont grisé dans le calendrier de sélection |
| | Post-conditions | - |
| CU10 - Créer un Gabarit personnalisé | L'utilisateur à la possibilité de créer une gabarit de protocole d'instance personnalisé et de le sauvegarder pour futur utilisation | |
| | Pré-conditions | L'utilisateur est connecter à la plateforme (CU02) |

| | | |
|---|--|---|
| | Flux de base | <ol style="list-style-type: none"> 1. L'utilisateur se rend sur la page du gabarit 2. Clique sur le bouton créer gabarit 3. Choisit les sections qu'il désire afficher dans le gabarit 4. Clique sur le bouton sauvegarder |
| | Post-conditions | Un nouveau gabarit personnalisé est créé et lié au compte de l'utilisateur |
| | | |
| CU11 - Éditer un Gabarit personnalisé | L'utilisateur a la possibilité d'éditer un gabarit de protocole d'instance personnalisé et de le sauvegarder pour future utilisation | |
| | Pré-conditions | L'utilisateur est connecté à la plateforme (CU02) |
| | Flux de base | <ol style="list-style-type: none"> 1. L'utilisateur se rend sur la page du gabarit 2. Clique sur le bouton éditer un gabarit 3. Effectue des modifications sur les sections qu'il désire afficher dans le gabarit 4. Clique sur le bouton sauvegarder |
| | Post-conditions | Les modifications du gabarit personnalisé sont enregistrées sur le compte de l'utilisateur |
| CU12 - Inviter des utilisateurs à un protocole d'instance | L'utilisateur a la possibilité d'inviter plusieurs utilisateurs sur un protocole d'instance | |
| | Pré-conditions | L'utilisateur est connecté à la plateforme (CU02) |
| | Flux de base | <ol style="list-style-type: none"> 1. L'utilisateur crée ou modifie un protocole d'instance 2. L'utilisateur arrive dans la section 'collaborer' 3. Ou il peut rentrer plusieurs adresses courriel utilisant le bouton de + 4. Puis clique, sur le bouton 'Sauvegarder et envoyer' 5. Le système envoie un courriel à tous les nouveaux intervenants |

| | | |
|---|--|--|
| | Post-conditions | Un courriel est envoyé à tous les nouveau intervenant et ceux-ci ont maintenant accès au protocole d'instance dans leur liste de protocol |
| | | |
| CU13 - Accepter les instances d'un protocol | L'utilisateur doit pouvoir accepter une ou plusieurs dates d'un protocol d'instance reçu | |
| | Pré-conditions | L'utilisateur est connecter à la plateforme (CU02) L'utilisateur doit avoir un protocol créé et lié à son compte (CU08) |
| | Flux de base | <ol style="list-style-type: none"> 1. L'utilisateur se rend dans la page 'Liste de protocol' 2. Choisi un protocol à valider 3. Arrive dans la liste des dates 4. Accepte les différentes instances 5. Le système sauvegarde les changement et change le sous-statut pour approuver |
| | Post-conditions | Le protocole d'instance change de sous-statut pour approuver |
| | | |
| CU13 - Refuser les instances d'un protocol | L'utilisateur doit pouvoir refuser une ou plusieurs dates d'un protocol d'instance reçu | |
| | Pré-conditions | L'utilisateur est connecter à la plateforme (CU02) L'utilisateur doit avoir un protocol créé et lié à son compte (CU08) |
| | Flux de base | <ol style="list-style-type: none"> 1. L'utilisateur se rend dans la page 'Liste de protocol' 2. Choisi un protocol à valider 3. Arrive dans la liste des dates 4. Refuse les différentes instances 5. Le système sauvegarde les changement et change le sous-statut pour approuver |
| | Post-conditions | Le protocole d'instance change de statut pour 'Rejected' |
| | | |

| | | |
|--|--|--|
| CU14 - Editer un protocol | L'utilisateur peut éditer un protocole d'instance à n'importe quel étape | |
| | Pré-conditions | L'utilisateur est connecter à la plateforme (CU02) L'utilisateur doit avoir un protocol créé et lié à son compte (CU08) |
| | Flux de base | <ol style="list-style-type: none"> 1. L'utilisateur se rend dans la page 'Liste de protocol' 2. Choisi un protocol à éditer 3. Une page similaire à celle de création avec toutes les dates qui avaient été rempli est affiché 4. L'utilisateur peut effectuer les modifications nécessaire 5. Cliquer sur le bouton 'sauvegarder' 6. Le système enregistre les modification apportées au protocol |
| | Post-conditions | Les modification du protocole son enregistré |
| CU15 - Supprimer un protocol | L'utilisateur peut supprimer un protocole d'instance à n'importe quel étape, excepté archivé | |
| | Pré-conditions | L'utilisateur est connecter à la plateforme (CU02) L'utilisateur doit avoir un protocol créé et lié à son compte (CU08) |
| | Flux de base | <ol style="list-style-type: none"> 1. L'utilisateur se rend dans la page 'Liste de protocol' 2. Choisi un protocol à supprimer 3. Le système confirme la suppression 4. Le système supprime le protocol d'instance |
| | Post-conditions | Le protocole d'instance est supprimé de la plateforme et la base de donnée |
| CU16 - Effectuer l'envoi final d'un protocol | L'utilisateur peut effectuer un envoie final du protocol lorsque celui-ci est accepté par tous les intervenants. | |
| | Pré-conditions | L'utilisateur est connecter à la plateforme |

| | | |
|---|--|---|
| | | (CU02) L'utilisateur doit avoir un protocole créé et lié à son compte (CU08) L'utilisateur dont les instances ont été acceptées (CU13) |
| | Flux de base | <ol style="list-style-type: none"> 1. L'utilisateur se rend dans la page 'Liste de protocole' 2. Choisi un protocole prêt à envoyer 3. Clique sur le bouton 'Envoyé à la cour' 4. Le système envoie un courriel à tous les intervenants et change le statut pour 'archiver' |
| | Post-conditions | Un email est envoyé à tous les intervenants avec une copie du pdf |
| | | |
| CU17 - Afficher le protocole dans le calendrier | L'utilisateur peut afficher un calendrier avec toutes les dates d'instances du protocole | |
| | Pré-conditions | L'utilisateur est connecter à la plateforme (CU02) L'utilisateur doit avoir un protocole créé et lié à son compte (CU08) |
| | Flux de base | <ol style="list-style-type: none"> 1. L'utilisateur se rend dans la page 'Liste de protocole' 2. Choisi un protocole à afficher 3. La page de calendrier va s'afficher avec toutes les dates d'instance |
| | Post-conditions | - |