LOG795

Identifier l’écart d’un cas par rapport au ratio

PFE-006
Version : 1.0

Date d'émission : 2019-08-14
Date de révision : 2019-08-14

Rapport Technique Final

Auteurs :
Sohaib Ouarrak
Ibrahim Bchouti
Hamzet Abid
Ayman Osman
Jérémie Beaulieu
François-Olivier April

Présenté à :
Alain April
Philippe April
Mathieu Dupuis

RÉSUMÉ

Plusieurs centaines de cas de cours existent pour les mêmes codes civils. Souvent, ces cas de cours se ressemblent beaucoup et ont toujours des attributs et décisions très semblables. Les avocats ont souvent besoin de parcourir des centaines de cas de cours similaires afin de ressortir l'information pertinente qui les aidera durant leur évaluation. Notre projet sert à automatiser la tâche en analysant les cas de cours afin de pouvoir identifier les détails les plus pertinents (ratio) qui expliquent le raisonnement derrière la décision du juge.

Table des matières

Historique des révisions

Date	Version	Description	Auteur
16-07-2019	0.1	Première version suite à la rencontre du 12/07/2019	Sohaib Ouarrak
19-07-2019	0.2	Modification et ajout des éléments du BRS	Sohaib Ouarrak
26-07-2019	0.3	Ajout des besoins des utilisateurs	Ibrahim Bchouti
2-08-2019	0.4	Ajout Annexe A et B	Ibrahim Bchouti
9-08-2019	0.5	Vue d'ensemble et de l'analyse par intelligence artificielle	Jérémie Beaulieu
10-08-2019	0.6	Vue d'ensemble du fonctionnement et structure du Backend	Ibrahim Bchouti
11-08-2019	0.7	Vue d'ensemble du fonctionnement et structure du Frontend	Ayman Osman Hamzet Abid
12-08-2019	0.8	Caractéristique et attribue des caractéristique	Ibrahim Bchouti
13-08-2019	0.9	Méthodologie, enjeux environnementaux et conclusion	Ibrahim Bchouti
14-08-2019	1.0	Formatage et correction du rapport	Sohaib Ouarrak

1 Introduction

1.1 Objectif

Ce document collige, analyse, et définit les besoins fonctionnels ainsi que les caractéristiques non-fonctionnelles, de haut niveau, du système PFE-006. Il se concentre sur les fonctionnalités recherchées par le client. Il est intentionnellement écrit dans le langage de l'utilisateur afin qu'il puisse bien saisir et ainsi clarifier ses exigences.

1.2 Portée

Ce document technique permet essentiellement de démontrer le cheminement du projet de conception logiciel de son début jusqu'à sa fin en présentant le contenu du projet ainsi que les résultats obtenus.

1.3 Références

Aucune référence n'a été utilisée

1.4 Langue de rédaction

Tous les documents relatifs au logiciel développé sont écrits en français, à l'exception du code source qui est écrit en anglais.

1.5 Définitions, acronymes et abréviations

Terme	Définition
AWS	Amazon Web Service est un service infonuagique à la demande
Back-End	La couche d'accès aux données du logiciel
Front-End	La couche de présentation du logiciel
API	Application Programming Interface permet aux applications de communiquer les unes avec les autres.
PFE	Projets de fin d'études
Ratio	Résumé d'un cas de cours qui explique la raison du verdict final
BD	Base de données
Cognito	Service d'authentification d'AWS
DynamoDB	Base de données NoSQL d'AWS
S3	Service de stockage de fichiers d'AWS
Lambda	Service d'AWS permettant d'exécuter du code sans avoir à mettre en service ou gérer des serveurs

2 Positionnement

2.1 Énoncé du problème

Le problème est	pour chaque cas de cours, les avocats doivent analyser les cas précédents pour connaître les différents ratios résultants afin de construire leurs cas.
Cela affecte	les spécialistes de la justice
Dont l'impact est	ils doivent lire plusieurs centaines de pages de cas de cours pour trouver les ratios de chaque cas.
Une bonne solution serait	d'avoir un programme qui extrait automatiquement les ratios des cas de cours.

2.2 Positionnement du produit

Pour	les spécialistes de la justice.
qui	désire faciliter la prise de décision d'un cas de justice ayant des cas similaires
PFE-006	est une plateforme web d'aide pour les spécialistes de la justice
qui	permet d'analyser plusieurs cas de cours pour en ressortir et classifier les ratios de chacun d'eux.
Contrairement à	lire manuellement les cas de cours et extraire les ratios à la main
Le produit final	est facile à utiliser, sauve du temps précieux aux spécialistes de la justice en ressortant les informations nécessaires pour analyser les cas antécédents d'un cas de cours

3 Description des intervenants et des utilisateurs

3.1 Résumé des intervenants

Nom	Représente	Rôle
P. April	Propriétaire du projet	Client
A.April	École de Technologie Supérieure	Superviseur du projet dans le cadre de LOG795
M.Dupuis	École de Technologie Supérieure	Superviseur du projet dans le cadre de LOG795
S. Ouarrak	École de Technologie Supérieure	Études, dans le cadre de LOG795, Analyse, conception et implémentation
I. Bchouti	École de Technologie Supérieure	Études, dans le cadre de LOG795, Analyse, conception et implémentation
A.Osman	École de Technologie Supérieure	Études, dans le cadre de LOG795, Analyse, conception et implémentation
H.Abid	École de Technologie Supérieure	Études, dans le cadre de LOG795, Analyse, conception et implémentation
J.Beaulieu	École de Technologie Supérieure	Études, dans le cadre de GTI795, Analyse, conception et implémentation
F-O.April	École de Technologie Supérieure	Études, dans le cadre de LOG795, Analyse, conception et implémentation

3.2 Résumé des utilisateurs

Nom	Description	Responsabilité	intervenant
P. April	Propriétaire du projet	Client	Émettent leurs besoins et tests/acceptent les solutions proposées
-	Spécialistes de la justice	Développement et conception du produit	Utilisent le logiciel pour leur travail ou bien pour avoir des idées lors de la prise de décision

3.3 Environnement utilisateur

- Poste de travail ayant le fureteur Chrome et une connexion Internet

3.4 Principaux besoins des intervenants et utilisateurs

Besoin	Priorité	Préoccupations	Solution actuelle	Solution proposée
B01 - Obtenir un ratio pour un nouveau cas de cours automatiquement	1	Pouvoir rapidement savoir le résultat d'un cas de cours	Passer manuellement à travers le cas de cours pour ressortir l'information importante	Site web qui permet d'extraire un ratio d'un fichier de cas de cours
B02 - Pouvoir regrouper les cas de cours similaires	2	Éviter les répétitions pour des cas de cours similaires et avoir une meilleure organisation	Aucune solution, tâche faite manuellement	Permettre à l'utilisateur de créer des projets où il peut regrouper des cas de cours similaires.
B03 - Être capable de s'authentifier et de protéger son	2	Perte de données et piratage d'information	Aucune solution	Service d'authentification sécurisé avec Cognito de

information		sensible		Amazon Web Services.
B04 - Obtenir des ratios de cas de cours précis et valides	1	Pouvoir prendre de bonnes décisions à partir des ratios obtenus	Fait manuellement	Utilisation d'une intelligence artificielle afin d'obtenir les meilleurs résultats
B05 - Pouvoir sauvegarder les cas de cours les plus importants	3	Pouvoir accéder rapidement aux cas de cours les plus importants	Aucune solution	Permettre à l'utilisateur de choisir les cas de cours les plus importants et les afficher dans la section des favoris
B06 - Être capable d'exporter les résultats de l'analyse	3	Pouvoir facilement partager les résultats durant un procès	Aucune solution	Permettre à l'utilisateur de télécharger les résultats en forme de PDF et de permettre de les imprimer

4 Vue d'ensemble du produit

4.1 Vue d'ensemble du produit à l'aide de l'architecture AWS

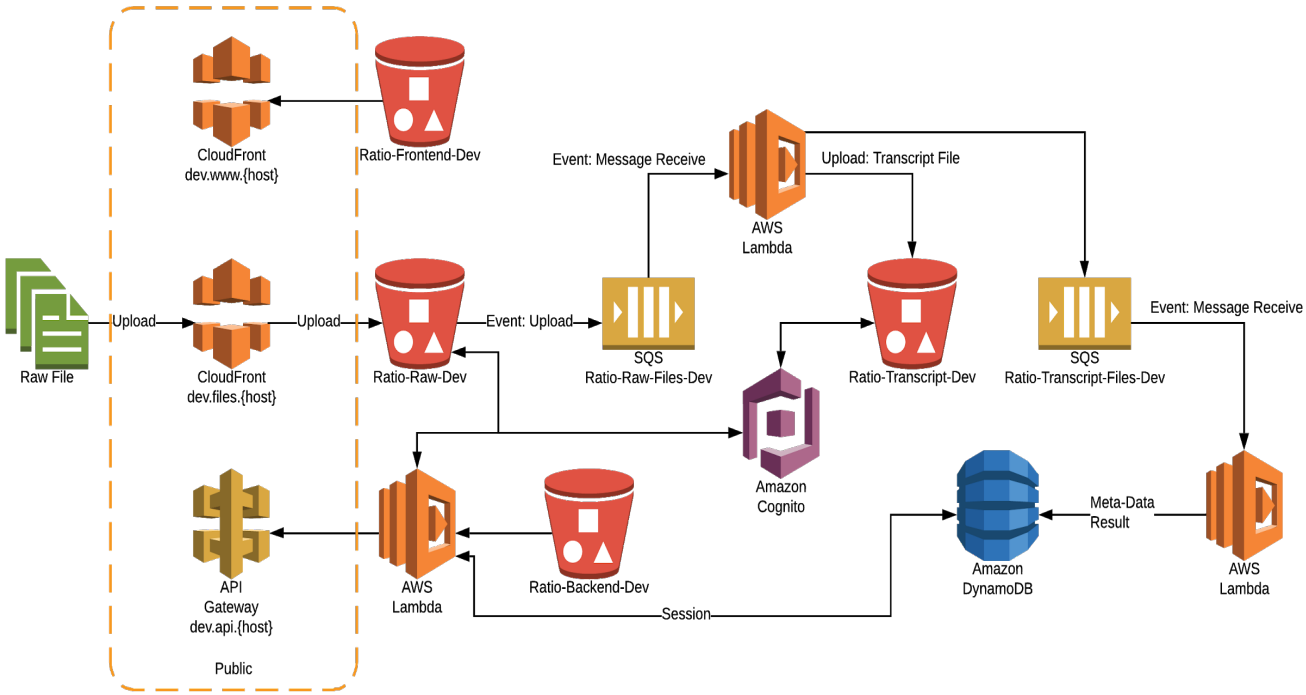

Figure 1 *- Proposition du service AWS concernant la nouvelle plateforme à développer*

Nous pouvons observer les différents services d'AWS utilisés tout au long de notre projet et comment ils peuvent communiquer entre eux dans notre architecture. Ceci est une architecture générale de notre système complète.

L'architecture est principalement séparée autour des trois grands services d'AWS que nous utilisons : Cognito, DynamoDB et S3. En effet ces trois services interagissent entre eux grâce aux fonctions lambda et aux fils SQS.

Cette structure nous permettra de facilement implémenter une application complètement « serverless ».

4.2 Vue d'ensemble de l'analyse par intelligence artificielle

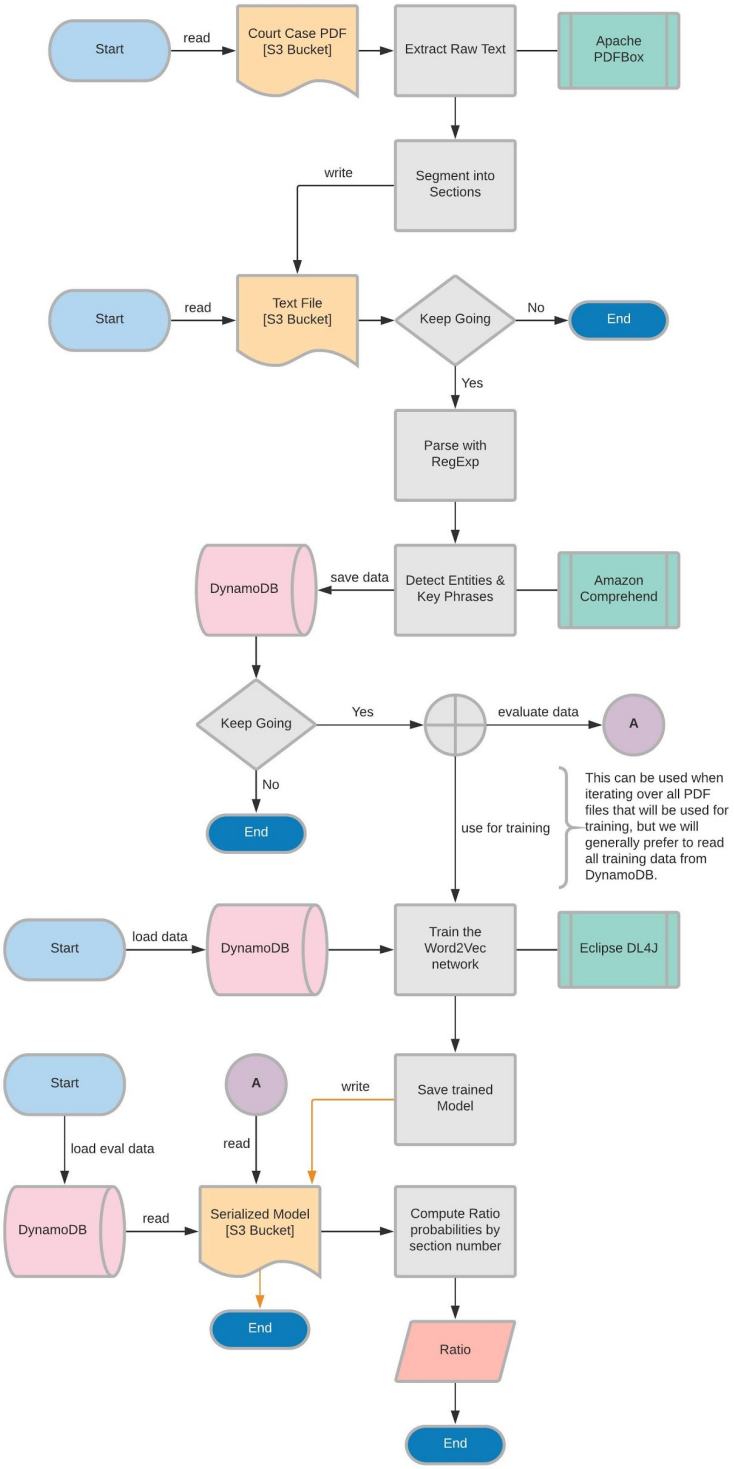

Figure 2 *- Diagramme de flux de la solution d'analyse par intelligence artificielle*

Le diagramme de la *Figure 2* ci-dessus illustre l'architecture globale de la solution d'analyse de cas de cours par voie d'intelligence artificielle, qui a été développée dans le cadre de ce projet de fin d'études. Ladite solution logicielle devait être en mesure d'effectuer une analyse sémantique complexe d'un cas de cour donné sous forme de document PDF, et d'utiliser les extrants issus de l'analyse pour évaluer le ratio. Pour ce faire, les fonctionnalités suivantes ont dû être implémentées : extraction et segmentation du texte brut, analyse initiale par expressions régulières, génération d'entités logiques et de phrases clés, entraînement des réseaux neuronaux, et évaluation probabiliste du ratio.

Vue d'ensemble par section :

- Extraction et segmentation du texte brut
    - L'extraction du texte brute se fait à l'aide de la librairie PDFBox d'Apache.
    - Le document PDF donné en entrée peut être lu d'un « Bucket » S3 d'Amazon, ou d'un dossier local sur la machine d'un développeur.
    - Le texte brut extrait peut être enregistré localement sous forme d'un fichier .txt. Il serait également facile de l'enregistrer dans un « Bucket » S3 d'Amazon si l'on désirait avoir la possibilité d'utiliser l'infrastructure infonuagique pour effectuer de nouvelles analyses à partir du fichier en question.
    - Un algorithme de segmentation faisant appel à des expressions régulières et à un algorithme de tri permet de diviser le texte en sections, conformément aux numéros que l'on retrouve entre crochets sur le cas de cour donné en entrée.
- Analyse initiale par expressions régulières
    - Cette procédure permet l'extraction des éléments d'intérêts suivants : juge, demandeur/demanderesse, défendeur/défenderesse, décisions (accueille, rejette, condamne, acquitte).
    - Les éléments ainsi obtenus ne sont présentement pas utilisés, mais pourraient facilement être intégrés au « front-end » dans une itération future.
- Génération d'entités logiques et de phrases clés
    - Utilisation de Amazon Comprehend, soit un API de traitement de langage naturel, pour la détection de différents types d'entités et de phrases clés.
    - Chaque section de texte est traitée individuellement afin de respecter les contraintes de taille (« input size ») de la librairie et ainsi en assurer l'efficacité.
    - Les données extraites sont enregistrées dans la base de données DynamoDB, afin que l'on puisse s'en servir pour la détection du ratio ou toute autre application future.
- Entraînement des réseaux neuronaux
    - Un modèle de type Word2Vec est généré à l'aide de « Eclipse DL4J » et est entraîné avec les phrases clés d'un ensemble de cas de cour dont on connaît le ratio.

    - Les phrases clés faisant partie d'une section qui constitue un ratio sont altérées avant d'être ajoutées aux vecteurs afin d'induire une similarité logique avec le mot ratio.
    - Une fois l'entraînement complété, le modèle est sérialisé et enregistré dans un « Bucket » S3 d'Amazon, afin d'être utilisé par après pour l'évaluation d'un cas de cour.
- Évaluation probabiliste du ratio
    - Un cas de cour ayant subi les étapes d'analyse préalables peut être évalué à l'aide des phrases clés qui en ont été extraites.
    - L'évaluation se base sur trois critères d'évaluation pondérés :
        - Le score (niveau de confiance) attribué par Amazon Comprehend pour chacune des phrases clés.
        - La similarité logique entre les mots qui constituent un ratio et ceux qui composent les phrases clés évaluées.
            - Cette étape requiert la lecture du modèle de type Word2Vec, entraîné au préalable.
        - La proximité physique entre la section qui comporte le jugement final est celle qui est en cours d'évaluation.
    - Un niveau de confiance entre 0 et 1 est attribué à chaque section de texte évaluée. La section dont cette valeur est la plus élevée sera considérée comme étant le ratio.
    - Une fois trouvé, le ratio est enregistré dans la base de données DynamoDB afin de servir le « back-end ».

<u>Forces de l'architecture :</u>

- Différents points de début « Start » et de fin « End » que l'on peut observer sur le diagramme de flux de la *Figure 2*. Ceux-ci mettent en évidence les différentes sections du code qui peuvent être exécutées, maintenues et mises à jour de manière indépendante. Il est ainsi possible, par exemple, d'entraîner à nouveau le réseau de neurones utile au calcul du ratio, sans avoir à se préoccuper des fichiers d'entraînement qui ont été analysés au préalable.
- Performance générale.
    - L'extraction du texte l'analyse par traitement de langage naturel, ainsi que l'entraînement et l'évaluation de nos réseaux neuronaux prennent tous très peu de temps en regard de leur complexité respective.

<u>Améliorations possibles :</u>

- Gérer la détection des lois mentionnées dans un cas de cour et leur incidence sur le jugement final. Un objet LawDTO prévu à cet effet a déjà été implémenté, mais ce dernier n'est jamais utilisé.

- Faire usage de tous les types d'entités analysés pour le calcul du ratio. Seules les entités de type « KeyPhrase » sont présentement utiles à cette fin.

4.3 Vue d'ensemble du fonctionnement et structure du Back-End

Tout d'abord le Back-End est constitué de plusieurs sections qui communiquent avec les parties les plus importantes de notre architecture. Le Back-End est au centre du fonctionnement général de notre application. Les services offerts à partir du Back-End nous permettent d'accomplir les tâches de base de notre application et d'obtenir un produit fonctionnel.

Le Back-End a été construit avec NodeJS et Express et il communique principalement avec les services d'AWS et le Front-End. Les services utilisés d'AWS sont : Cognito pour l'authentification, DynamoDB en tant que base de données NoSQL et S3 pour stocker et récupérer des fichiers.

<u>Vue d'ensemble par section :</u>

- Authentification avec Cognito d'AWS
    - L'utilisateur peut se créer un nouveau compte sécurisé avec courriel, nom et mot de passe
    - Le mot de passe de l'utilisateur doit suivre les règles que nous avons appliquées dans Cognito (8 caractères minimum, chiffre, majuscule)
    - L'utilisateur doit confirmer son compte après s'être enregistré afin de s'assurer qu'il est bien le propriétaire de l'adresse courriel. Il peut confirmer son compte avec un code reçu via courriel
    - Lorsque le compte de l'utilisateur est créé et confirmé, il peut se connecter à l'application grâce à son courriel et mot de passe.
    - Une option de récupération de mot de passe oubliée et disponible si l'utilisateur ne se rappelle plus de son mot de passe. Il recevra un code de confirmation à son courriel afin de pouvoir changer son mot de passe.
    - L'utilisateur, lorsque connecté à l'application, a le choix de pouvoir changer son mot de passe quand il veut
- Utilisation de base de données DynamoDB
    - Les fonctionnalités et routes du Back-End nous permet de faire la création de nouveaux projets avec (id, date, nom, userID)
    - Nous pouvons ensuite obtenir tous les projets ou un projet spécifique grâce à des routes séparées
    - Le Back-End offre aussi la possibilité de sauvegarder une session d'un utilisateur. La session sera mise dans une table de DynamoDB et elle pourra être chargée lors du retour de l'utilisateur
- Stockage de fichiers avec S3 d'AWS
    - Le Back-End offre l'option d'ajouter de nouveaux fichiers dans le « Bucket » S3 (déplacer dans le Front-End)

    - Nous pouvons aussi obtenir et lire n'importe quel fichier PDF qui se retrouve dans notre « Bucket » S3
    - Analyser et obtenir le ratio d'un fichier PDF sur S3
- Analyse d'un cas de cours directement fait dans le Back-End
    - Analyser le cas de cours d'un fichier PDF sur S3
    - Obtenir le texte brut du cas de cours
    - Utilisation d'une table DynamoDB avec des poids pour chaque mot
    - Calculer, à l'aide de la table des poids sur DynamoDB, les paragraphes avec le poids le plus élevé
    - Trouver le ratio du cas de cours analysé et renvoyer le ratio trouvé au Front-End

Forces de l'architecture :

- Tout d'abord, nous cherchions à produire une application serverless. C'était un des besoins importants du client. Ceci a beaucoup affecté notre choix de technologies pour le Back-End. Nous avons utilisé NodeJS avec Express afin de pouvoir facilement transitionnel vers une structure serverless avec les fonctions lambda d'AWS. En effet nous pouvons utiliser des solutions comme ClaudiaJS qui nous permettent de déployer des services Express directement sur lambda.
- L'utilisation des services d'AWS nous offre une très bonne sécurité pour les comptes d'utilisateur et les informations dans la base de données et des fichiers stocker sur S3.
- Nous pouvons aussi très facilement agrandir et accommoder plus de requête et d'utilisateur grâce au modèle « Pay as you go » d'AWS. Nous ne devons pas nous occuper de gérer nos propres serveurs et les limites qui s'y attachent.
    - Performance générale
    - Le système d'authentification fonctionne parfaitement et nous permet de bien établir des règles pour les utilisateurs et leur compte. Les utilisateurs peuvent facilement créer de nouveaux comptes et gérer leurs comptes.
    - La base de données DynamoDB est très rapide et facile à utiliser pour les besoins que l'on a. Nous avons des données qui ont peu de relations entre eux, donc il est plus simple d'utiliser une base de données NoSQL.
    - Le système de stockage S3 nous permet de rapidement stocker et obtenir des fichiers PDF qui sont les seuls types de fichiers que l'on accepte

Améliorations possibles :

- Il serait important d'avoir plus de sécurité pour certaines des requêtes de notre Back-End. En effet, toutes les requêtes qui causent un changement dans la base de données DynamoDB ou S3 sont protégées par le fait que l'utilisateur doit être authentifié. Mais, il faut vérifier que l'utilisateur est le propriétaire du projet. Un utilisateur ne devrait pas avoir le droit d'ajouter des fichiers dans le projet d'un autre utilisateur.
- Il serait intéressant d'ajouter un système de cache avec base de données DynamoDB. Ceci nous permettrait de sauvegarder les requêtes avec leur réponse

dans une table. De cette façon, nous pourrions réutiliser la réponse sans refaire une nouvelle requête. Les requêtes sauvegardées dans la table auraient chacun une date d'expiration afin de pouvoir rafraîchir les réponses obtenues.

- Il serait très important pour le Back-End d'ajouter des tests et de la documentation afin de pouvoir faciliter la tâche des nouveaux développeurs qui cherche à améliorer l'application

4.4 Vue d'ensemble du fonctionnement et structure du Front-End

Le Front-End est une application web écrite en JavaScript à l'aide de la librairie React. L'application est coupée en trois principales sections : les vues, les services, et l'état (« state »). Les vues ont la responsabilité d'afficher le contenu à l'utilisateur, les services ont la responsabilité de communiquer avec le Back-End et de lire et modifier les données de l'application, finalement, la section d'état à la responsabilité de garder les données en mémoire.

Les 3 sections communiquent ensemble afin d'avoir un système fonctionnel. Plus précisément, les services vont communiquer avec le Back-End via HTTP, une fois les données reçues, celles-ci sont entreposées dans l'état de l'application. Ensuite, à l'aide de Redux, nous connectons les vues avec l'état de l'application, de façon à ce que les vues soient rafraîchies à chaque fois que les données sont modifiées.

<u>Vue d'ensemble par section :</u>

Vues :

Il y a 4 principales vues dans l'application. Tout d'abord, il y a la page de connexion, dans celle-ci l'utilisateur se connecte, ou s'enregistre. Ensuite, il y a la liste de projets, dans laquelle l'utilisateur consulte ses projets anciennement créés. Ensuite, il y a le détail d'un projet, sur cette page l'utilisateur peut téléverser ses documents, et consulter le ratio pour chacun de ceux-ci. Finalement, il y a la page profilée, qui est utilisée par l'utilisateur afin de consulter les informations liées à son profil.

Services :

L'application possède deux services, « AuthService » et « ProjectService ». Le premier est utilisé pour les fonctionnalités liées à l'authentification de l'utilisateur ainsi que la gestion de son profil. Le deuxième est utilisé pour toutes les fonctionnalités reliées au Projet. C'est-à-dire, la création et le chargement de projets et le téléversement de documents. Les services sont appelés des « sagas » qui sont un terme lié à la librairie « Redux-Saga » qui a d'ailleurs été utilisé dans ce projet.

État :

L'état de l'application est géré par ce qui est appelé des « reducers ». Tout comme les services, il y en a deux. Le premier s'occupe des informations liées à

l'utilisateur et le deuxième s'occupe des informations liées aux projets, tels que la liste des projets chargés dans l'application et le détail du projet couramment sélectionné.

Fonctionnalités :

Essentiellement, le Front-End est seulement responsable d'afficher les données et de communiquer avec le Back-End via HTTP afin d'effectuer les actions. Par contre, il y a certaines fonctionnalités qui ont été déléguées au Front-End. Dans cette section, nous expliquerons comment les grandes fonctionnalités de l'application sont exécutées.

- Connexion d'un utilisateur :

  Lors de la connexion, le service « AuthService » va appeler notre Back-End via l'URL : /login. Dans le Body de la requête, nous envoyant un objet possédant le courriel et le mot de passe de l'utilisateur, afin que le Back-End puisse vérifier si la connexion est valide. Une fois validé, le Back-End nous retournera un token JWT. Par la suite, toute requête va envoyer dans ses « headers » le token JWT en question afin que le Back-End puisse savoir de quel utilisateur il s'agit.

- Extraction d'un ratio :

  Le Front-End n'a aucune responsabilité d'analyse de fichier, son rôle est de contacter le Back-End en donnant la clé du fichier à analyser, et celui-ci va faire l'analyse et ensuite redonner le résultat au Front-End.

- Téléversement de fichiers PDF sur un « Bucket » S3 :

  Pour le téléversement des fichiers, nous avons utilisé le SDK offert par AWS pour JavaScript. Ainsi, c'est le Front-End qui communique avec S3 et lui envoie le fichier sans passer par le Back-End. Le SDK offert par AWS offre toutes les fonctions nécessaires pour le téléversement. Tout ce qu'on a à faire c'est appeler la fonction « upload » offerte par le SDK en lui passant la clé du fichier, et le nom du « Bucket » ainsi que le fichier à téléverser.

Forces de l'architecture :

- La séparation des vues et des services fait en sorte que les vues n'ont aucune logique autre celle d'afficher les données. Tout ce qui est en lien avec les fonctionnalités ou bien les appels vers le Back-End est géré par les services. Ainsi, on réduit le couplage et il est très facile d'ajouter et enlever des fonctionnalités sans toucher aux vues.
- La persistance du jeton JWT permet une gestion simple de la session d'un utilisateur et très sécuritaire, car il n'est plus nécessaire de garder en mémoire le courriel et le mot de passe de l'utilisateur.

- L'utilisation d'un service rend simple la gestion de l'ordre des appels API, parfois, certains appels doivent attendre qu'un autre appel soit terminé avant d'être exécuté. L'usage de services avec la libraire Redux-Saga permet de facilement attendre qu'un appel préalable termine avant d'effectuer l'appel en question.
- Le projet sera maintenu sur une longue durée, ainsi le choix de la librairie React est très sensé, car celui-ci est très populaire, puis possède beaucoup de documentation et une grande communauté en ligne.

Améliorations possibles :

- Le front-end n'utilise pas les fichiers de configurations pour obtenir le lien vers l'api. Ainsi, il peut être difficile de changer d'environnement, par exemple, dans le cas ou il y aurait un serveur Back-End par environnement (production, développement) il faudrait faire des changements au niveau du code de l'application pour faire le changement entre chacun des environnements.
- Pour le téléversement de fichier, nous utilisons la clé secrète du compte AWS, celle-ci se retrouve dans le code du Front-End. Ceci peut-être dangereux, car n'importe qui y ayant accès pourrait faire des appels malicieux vers nos services AWS. Il faudrait utiliser le jeton JWT au lieu de la clé secrète. Puisque le jeton provient de Cognito, un autre service de AWS, la gestion des droits vers S3 peut aussi être gérée par ce jeton.
- Il serait intéressant de mettre dans l'état de l'application une propriété « loading » qui change à True à chaque fois qu'un appel vers l'API est en cours afin de pouvoir facilement gérer l'affichage d'un indicateur de chargement.

4.5 Automatisation de l'extraction des données

L'automatisation de l'extraction des données consiste en l'utilisation de fonctions Lambda afin d'extraire le texte brut ainsi que le ratio des fichiers PDF de cas de cours dès leur téléversement dans le compartiment S3. Les deux fonctions utilisées sont nécessairement écrites en Java, puisqu'elles incorporent respectivement l'extraction du texte brut et l'analyse par intelligence artificielle. Des files de messagerie SQS sont utilisées afin de déclencher les fonctions, telles qu'illustrées dans la *Figure 1*.

Il est à noter que l'automatisation de l'extraction des données n'est actuellement pas fonctionnelle. Divers problèmes reliés au SDK d'AWS pour Java ont retardé cette section du projet. Principalement, la version 1 du kit de développement, étant bien mieux documentée que la version 2, a été initialement choisie pour le développement des fonctions Lambda. Cependant, le fat JAR résultant des dépendances du SDK était trop volumineux pour être utilisé par Lambda. Il fut donc nécessaire d'utiliser la version 2, n'ayant pas ce problème.

Le fil conducteur de l'extraction des données va comme suit:

- Téléversement d'un cas de cour sous format PDF au compartiment Ratio-Raw

- Envoi de message à la file Ratio-Raw-Files
- Activation de la première fonction Lambda, qui charge le fichier PDF
- Extraction du texte brut en sauvegarde locale
- Téléversement du texte brut au compartiment Ratio-Transcript
- Envoi de message à la file Ratio-Transcript-Files
- Activation de la seconde fonction Lambda, qui charge le fichier texte
- Extraction du Ratio
- Ajout du Ratio dans la base de données

Vue d'ensemble par section :

- Fonctions Lambda
    - Le code Java utilisé constituant les fonctions Lambda doit être sous forme de fat JAR ou de fichier ZIP contenant le JAR principal ainsi que les dépendances. Il est aussi possible d'utiliser le système de couches contenant un groupe de dépendances prédéterminées. Celles-ci sont mises en ligne séparément. Le code utilisé est sous forme de fat JAR.
    - La fonction *Transcript* est déclenchée lorsqu'un message est envoyé à la file d'attente Ratio-Raw-Files. Le message en question contient l'emplacement et le nom du fichier PDF à récupérer dans le compartiment S3. La fonction télécharge le fichier PDF dans l'emplacement temporaire /tmp de Lambda, en extrait le texte brut, puis le téléverse sous forme de fichier texte dans le compartiment Ratio-Transcript
    - La fonction *ratio* est déclenchée lorsqu'un message est envoyé à la file d'attente Ratio-Transcript. La fonction télécharge le fichier texte dans l'emplacement temporaire /tmp de Lambda puis en extrait le ratio en utilisant les réseaux neuronaux à l'aide de l'analyse par intelligence artificielle décrite dans la section 4.2. Ensuite, le ratio est placé dans la base de données DynamoDB
- Files d'attente SQS
    - La file d'attente Ratio-Raw-Files reçoit un message lorsqu'un téléversement est fait dans le compartiment S3 Ratio-Raw. S'il n'a pas été traité, après un temps d'attente prédéterminé, le message est acheminé vers la file Ratio-Raw-Files-DL, afin de ne pas bloquer la file principale en cas de problème et afin de laisser une trace si un fichier n'a pas pu suivre le procédé normal.
    - La file d'attente Ratio-Transcript reçoit un message lorsqu'un téléversement est fait dans le compartiment S3 Ratio-Transcript. Pour les mêmes raisons que dans le cas de la file d'attente précédente, si les messages n'ont pas été traités passé un délai prédéterminé, ceux-ci se retrouvent dans la file Ratio-Transcript-DL.
- Stockage de fichier avec S3
    - Le compartiment Ratio-Raw est utilisé pour le téléversement initial des fichiers PDF de cas de cours. Il est configuré pour automatiquement

envoyer un message vers la file SQS Ratio-Raw-Files dès qu'un téléversement est fait.
    - Le compartiment Ratio-Transcript est utilisé par la fonction Lambda *Transcript* pour le téléversement des fichiers de texte brut. Il est configuré pour automatiquement envoyer un message vers la file SQS Ratio-Transcript dès qu'un téléversement est fait.

Forces de l'architecture:

- L'utilisation des fonctions Lambda constitue une architecture sans serveur. En effet, aucun frais n'est appliqué lorsque le service n'est pas utilisé. Le coût du service est donc calculé par temps de calcul consommé. De plus, la gestion d'un serveur et de ses ressources est ainsi éliminée.
- Lambda communique facilement avec les autres services d'AWS. Considérant le fait que tous les services en lignes utilisés pour le projet sont propres à Amazon, l'intégration de Lambda s'en voit facilitée.
- L'utilisation des fonctions Lambda est indépendante du Frontend et du Backend de l'application. Il n'est jamais nécessaire de communiquer directement avec l'un ou l'autre. Cela facilite grandement les modifications et agit comme une forme d'encapsulation pour l'ensemble de l'extraction des données.
- L'extraction est entièrement automatisée. Dès qu'un utilisateur téléverse un fichier PDF sur S3, l'entièreté du processus d'extraction est faite sans contribution additionnelle de sa part.

Améliorations possibles:

- Il est possible de déclencher une fonction Lambda directement lorsqu'un événement prédéterminé, tel que le téléversement d'un fichier, se produit sur un compartiment S3. Cela rendrait possible l'élimination des files d'attente SQS dans l'automatisation de l'extraction. Toutefois, mois de traces de l'utilisation serait laissée et une partie de l'information transmise dans les messages SQS serait perdue.
- L'utilisation de couches Lambda aurait facilité la gestion problématique des dépendances et aurait permis un Debugging plus facile. En effet, il était nécessaire de compiler le code incluant toutes les dépendances et de téléverser le fichier JAR en résultant à chaque modification, aussi petite soit-elle. Avec les couches Lambda, il aurait été possible de mettre en ligne les dépendances n'ayant aucune chance de changer au cours du projet, puis de téléverser un JAR allégé lors des changements.
- Il serait possible de combiner les deux fonctions Lambda pour ainsi passer directement du fichier PDF au ratio, sans sauvegarder le texte brut, sauvant potentiellement du temps d'exécution, selon les besoins des utilisateurs.

5 Décisions prises tout au long du projet

Besoin	Décision #1	Décision #2	Décision finale
Un Back-End pour acheminer des requêtes et interagir avec le Front-End et les services AWS	Un Back-End NodeJS avec Expresse	Un Back-End avec Spring	Back-End NodeJS avec Expresse, car il est plus simple à implémenter et les membres de l'équipe ont plus d'expérience reliée à NodeJS.
Système d'authentification	Utiliser Cognito de AWS	Utiliser les services offerts par Google	Cognito de AWS, car nous avons décidé d'utiliser lorsque possible les services offerts sur AWS pour garder une bonne cohésion.
Base de données pour conserver les données importantes	Utiliser DynamoDB de AWS	Utiliser une base de données SQL de AWS	DynamoDB de AWS, car nous trouvons plus d'avantages à utiliser une base de données NOSQL
Entreposage pour les fichiers importants	Utiliser S3 de AWS	Utiliser les services offerts par Google	S3 de AWS, car nous avons décidé d'utiliser lorsque possible les services offerts sur AWS pour garder une bonne cohésion.
Protection de l'accès et secret Key d'AWS	Placer les Keys dans le fichier de credential de l'utilisateur	Créer des variables d'environnement sur AWS	Placer les Keys dans le fichier de credentials, car cela permet à chaque utilisateur de facilement modifier les paramètres qu'il veut.
Téléversement des fichiers au « Bucket	Faire le téléversement à	Faire le téléversement à	Téléversement à partir du Front-End,

» S3	partir du Front-End	partir du Back-End	car cela permet de réduire le nombre de requêtes et sauve de l'argent
Intégration d'un service de NLP (« natural language processing »)	Amazon Comprehend	Rosette Text Analytics	Amazon Comprehend, en raison de sa performance et facilité d'intégration à notre architecture AWS.
Sélection d'une librairie pour l'extraction de données	Apache PDFBox	iText	Les deux librairies répondaient facilement à nos besoins fonctionnels, mais nous étions plus familiers avec PDFBox et aimions le fait qu'il utilise la licence permissive *Apache License 2.0*.
Sélection d'une librairie pour les réseaux neuronaux	Eclipse DL4J	TensorFlow	DL4J, puisque sa documentation rendait très claires les procédures à suivre pour l'entraînement, la sérialisation, ainsi que l'évaluation de différents types de réseaux neuronaux.
Sélection d'une version du SDK d'AWS pour Java	SDK version 1	SDK version 2	La version 2 a été choisie, car la version 1 est beaucoup plus volumineuse, ce qui est problématique lors de la création du fichier JAR utilisé dans les fonctions Lambda
Sélection d'un	Angular	ReactJS avec	Bien que les deux

framework pour l'écriture de l'application Front-End		React-Redux	soient excellents et offrent les mêmes avantages. React est un peu plus flexible et l'équipe avait plus d'expérience avec celui-ci donc nous avons décidé de le choisir.
Sélection d'une librairie de composants pour le Front-End	Bootstrap	Material-UI	Après avoir comparé les deux un contre l'autre, Material-UI semblait plus populaire, car il a plus de contributeurs sur Github. Ainsi, nous avons décidé de l'utiliser.
Sélection d'une librairie pour les appels asynchrones du Front-End	Fetch	Axios	Bien que les deux soient très populaires, Fetch semblait plus facile d'utilisation. Donc nous avons décidé de l'utiliser. Aussi, un des membres de l'équipe était à l'aise avec celle-ci donc, il pouvait nous aider en cas de problème.

6 Caractéristiques du produit

6.1 CAR01 – Importation des cas de cours

Le logiciel doit permettre l'import et afficher les cas de cours à analyser.

6.2 CAR02 – Afficher la progression de l'analyse.

Le logiciel doit afficher la progression de l'analyse du cas de cours à l'utilisateur afin de voir l'état de l'analyse.

6.3 CAR03 – Permettre de regrouper les cas de cours

Le logiciel doit permettre de regrouper les cas de cours par analyse.

6.4 CAR04 – Afficher le ratio du cas

Le logiciel doit pouvoir afficher le ratio de chaque cas soumis par l'utilisateur et le classer selon la partie qui a gagné (défense ou plaignant) de façons automatiques.

6.5 CAR05 - Authentification de l'utilisateur

Permettre à l'utilisateur de créer un compte sécurisé et de pouvoir s'authentifier pour accéder à son compte.

6.6 CAR06 - Récupération de mot de passe oublié

Permettre à l'utilisateur de récupérer et changer son mot de passe s'il ne se rappelle plus son mot de passe.

6.7 CAR07 - Changer de mot de passe d'utilisateur

Permettre à l'utilisateur de modifier son mot de passe lorsqu'il est authentifié sur la plateforme.

7 Caractéristiques non fonctionnelles

7.1 ENF01 - Sécurité

Tous les cas de cours importés par l'utilisateur sont accessibles que par cette utilisatrice. L'utilisateur peut accéder à son espace de travail en utilisant son courriel et un mot de passe.

7.2 ENF02 - Performance

Le logiciel doit répondre en moins de 100ms pour toute requête du client. Par contre, les résultats de l'analyse d'un cas de cours peuvent être retournés de façon asynchrone.

7.3 ENF03 - Réutilisabilité

Le système du logiciel doit contenir des commentaires pertinents afin que les futurs développeurs puissent comprendre le code créé.

7.4 ENF03 - Portabilité

Le logiciel est développé et testé pour être utilisé avec le fureteur Chrome.

8 Contraintes

8.1 CC01 - Type de fichier

Le logiciel doit seulement accepter les cas de cours en format PDF.

8.2 CC02 - Hébergement

Le logiciel doit utiliser les services web d'Amazon (S3, DynamoDB).

8.3 CC03 - Type d'application infonuagique

Le logiciel doit être une plateforme web serverless.

9 Attributs des caractéristiques

Caractéristiques	État	Bénéfice	Effort	Risque
CAR01 - Importation des cas de cours	Complété	Élevé	Bas	Bas
CAR02 - Afficher la progression de l'analyse	Approuvé	Moyen	Bas	Bas
CAR03 - Permettre de regrouper les cas de cours	Complété	Élevé	Moyen	Moyen
CAR04 - Afficher le ratio du cas	Complété	Élevé	Haut	Élevé
CAR05 - Authentification de l'utilisateur	Complété	Élevé	Haut	Élevé
CAR06 - Récupération de mot de passe oublié	Complété	Moyen	Moyen	Bas
CAR07 - Changer de mot de passe d'utilisateur	Complété	Moyen	Bas	Bas

10 Échéancier du projet

Livrable	Date de début estimée	Date de fin estimée
Choix de technologie	20/05/2019	27/05/2019
Structure de l'architecture	27/05/2019	03/06/2019
Ajout d'un système d'authentification	03/06/2019	17/06/2019
Fonctionnalité de base (création de projets, téléversement de cas de cours)	03/06/2019	08/07/2019
Automatisation de l'extraction des données	03/06/2019	02/08/2019
Algorithme d'analyse de ratio	22/06/2019	02/08/2019
Intelligence artificielle pour déterminer le ratio et les prédictions	03/06/2019	02/08/2019

11 Méthodologie de travail

11.1 Gestion

Nous avons dès le début identifié les technologies et les services que nous allions utiliser. Nous avons décidé d'utiliser les services d'AWS. Une fois que le choix a été accepté par l'équipe, nous avons conçu une architecture détaillée qui nous permet de bien comprendre les tâches à accomplir et de comprendre comment chaque service communiquera entre elles.

11.2 Outils

Google Drive

Nous avons utilisé Google Drive afin de garder tous les fichiers importants au même endroit. Nous avons des répertoires pour les différentes ressources que nous avons eues durant la session. Nous avons aussi gardé tous les exemples de cas de cours sur le Drive afin d'avoir un accès simple.

Trello

L'utilisation de la plateforme Trello nous a servi à planifier les tâches à accomplir et faire un suivi durant toute la session.

Slack

La plateforme Slack nous a permis de communiquer facilement et rapidement entre nous.

Lucidchart

Lucidchart nous a permis de construire des diagrammes et schémas.

11.3 Technologies

Java

Le langage Java nous a principalement été utile pour le développement des parseurs de fichier et des fonctions lambda.

AWS

Nous avons utilisé les services d'AWS (Cognito, DynamoDB, Lambda, Comprehend, S3) afin de pouvoir accomplir les tâches et les exigences.

ReactJS

ReactJS a été utilisé afin de construire le Front-End et toutes les vues que les utilisateurs peuvent interagir avec.

NodeJS

NodeJS a été utilisé afin de construire le Back-End et les divers services offerts par le Back-End.

12 Enjeux environnementaux

Tout d'abord, notre projet n'était complètement technologique avec aucune partie physique qui pourrait avoir un effet néfaste à l'environnement. Les projets technologiques logiciels ont rarement des impacts directs sur l'environnement, car nous développons des applications qui seront utilisées sur des ordinateurs.

Nous avons par contre le choix des services technologique que nous avons utilisé dans notre application. La totalité des services provient d'AWS. Amazon cherche toujours à réduire le plus possible leur empreinte écologique avec les choix des centres de données et des infrastructures qu'ils construisent.

- En effet, Amazon s'assure d'utiliser le plus possible de l'énergie renouvelable et cherche à réduire les émissions de carbone. Voici un extrait tiré directement du site web d'AWS : *« Nous prenons également en compte des considérations de durabilité dans la conception de nos centres de données. AWS est engagé depuis longtemps dans l'utilisation d'une énergie 100 % renouvelable. Lorsque des entreprises passent d'une infrastructure sur site au cloud AWS, elles réduisent généralement leurs émissions de carbone de 88 %, car nos centres de données peuvent offrir des économies d'échelle environnementales. Les entreprises utilisent généralement 77 % de serveurs en moins, 84 % d'électricité en moins et 28 % de mélange électricité solaire - électricité éolienne en plus dans le cloud AWS par rapport à leurs propres centres de données. » -* *https://aws.amazon.com/fr/compliance/data-center/environmental-layer/?nc1=h_ls*

Nous pouvons observer que le choix d'utiliser les services d'AWS est une très bonne décision en termes de développement durable et cela nous permet de réduire au maximum notre empreinte écologique.

13 Conclusion et recommandations

Le projet était un tout nouveau projet de fin d'études ce qui apportait beaucoup d'avantage et de désavantage. En termes d'avantage, nous avons eu la chance de choisir la structure que l'on voulait et de décider quelle technologie nous allions utiliser pour chaque partie de l'application. Mais les désavantages étaient que nous n'avions pas de chemins clairs et les exigences ou spécifications changeaient de semaine en semaine.

Ce projet nous a permis de découvrir de nouvelles technologies et d'apprendre beaucoup de choses par rapport au domaine de droits et cas de cours. Nous avons eu la chance de travailler en étroite collaboration avec des professionnels afin de développer un logiciel professionnel qui atteint tous leurs objectifs.

Nous pouvons, sans aucun doute, dire que nous avons atteint nos objectifs. Bien sûr, comme le projet était nouveau, il n'était pas attendu de notre part de compléter ou de finir l'application en une seule session. Mais nous avons réussi à accomplir les tâches les plus importantes reliées au projet.

En effet, notre solution permet à l'utilisateur de se créer un compte sécurisé, de créer des projets, d'ajouter des cas de cours au projet et d'analyser chaque cas de cours afin d'obtenir le ratio. Ceci est exactement ce que nous devions accomplir durant notre projet et encore plus.

Il reste encore beaucoup d'aspects à améliorer avant de pouvoir déployer le projet en tant que produit final. Il faudrait principalement terminer les tâches d'automatisation avec les fonctions lambda et de regrouper les différents services (Front-End, Back-End et AI) ensemble afin de pouvoir utiliser toute la puissance de chacun des services.

14 Annexe

14.1 Annexe A - Diagramme de cas d'utilisation

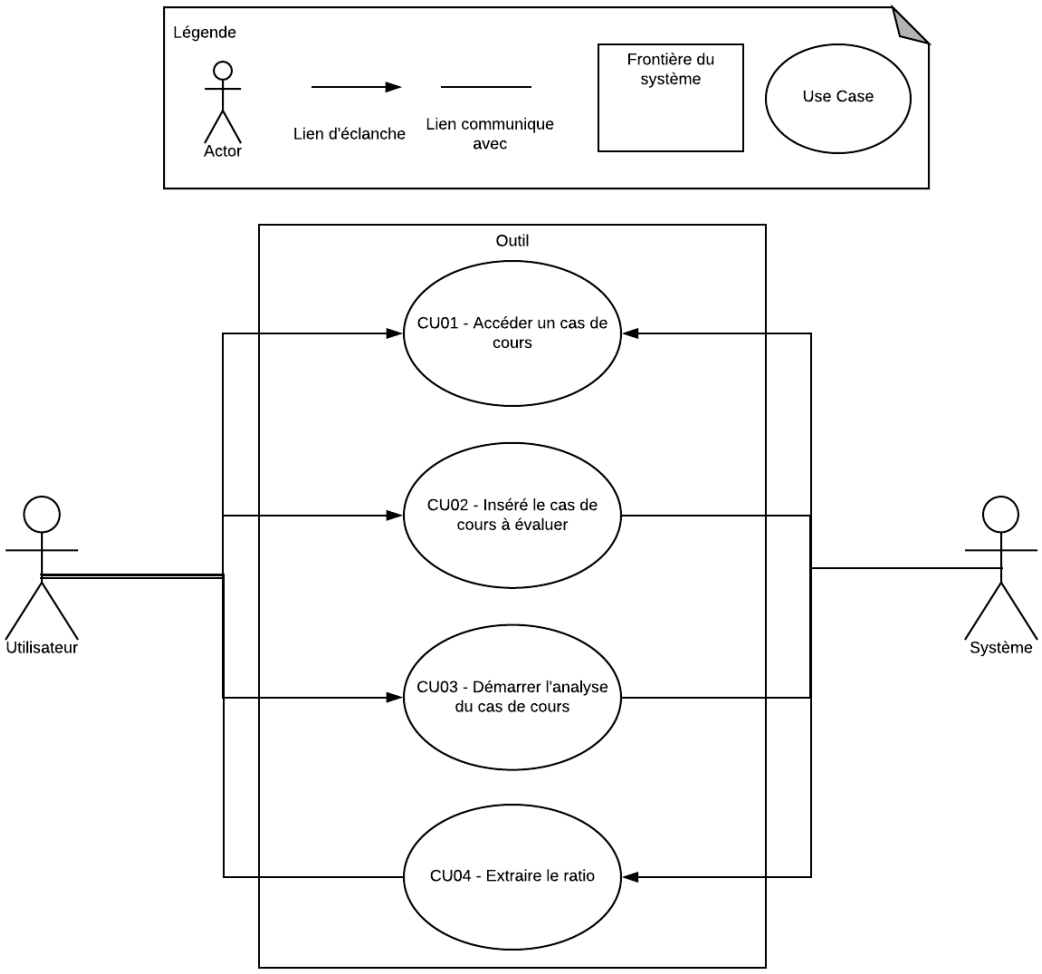

Figure 1 *- Cas d'utilisations concernant la nouvelle plateforme à développer*

Dans l'image ci-dessus, nous avons le diagramme des cas d'utilisation qui définit la manière d'utiliser le système pour les utilisateurs.

14.2 Annexe B - Attributs des caractéristiques

Cette section présente la définition des attributs que nous avons associés aux différentes caractéristiques du prototype logiciel.

État

Proposé	La caractéristique est proposée.
Approuvé	La caractéristique a été approuvée par les parties prenantes.
Complété	La caractéristique est complétée et incluse dans le système.
Annulé	La caractéristique a été annulée.

Bénéfice

Faible	La caractéristique apporte un bénéfice faible au système. La valeur ajoutée n'est pas nécessaire au bon fonctionnement du système.
Moyen	La valeur ajoutée de la caractéristique n'est pas critique au bon fonctionnement du système.
Élevé	La valeur ajoutée de la caractéristique est très élevée ou essentielle au système.

Effort

Bas	Ce niveau indique que la quantité d'effort requis est en dessous d'une semaine de travail.
Moyen	Ce niveau indique que la quantité d'effort requis est entre une et deux semaines de travail.
Haut	Ce niveau indique que la quantité d'effort requis est d'au moins deux semaines de travail.

Risque

Faible	La technologie est bien connue, la méthode d'implémentation est peu complexe, ou la conception est simple à effectuer.
Moyen	La technologie est récente ou a peu de documentation, la méthode d'implémentation demande une attention particulière, ou la conception requiert une analyse assez complète.
Élevé	La technologie est peu connue, la méthode d'implémentation est complexe, ou la conception requiert une analyse très complète.