Outils d'Analyse Sémantique

Rapport de projet
Conception et Développement Logiciel
Master 1 Informatique
2019 / 2020

Membres:

Killian MARCHAND
Romane BUON
Artem TARASENKO

Responsables :

Nathalie CAMELIN
Antoine LAURENT

Table des matières

I - Présentation du projet

Ce projet est réalisé dans le cadre de la première année de Master Informatique à Le Mans Université.
Il s'intègre au projet de recherche sur la "reconnaissance de la parole" du Laboratoire d'Informatique de le Mans Université (LIUM).

L'équipe Language and Speech Technology (LST) du LIUM travaille sur des activités de recherche dans le domaine du traitement automatique du langage et de la parole.

Leurs travaux de recherche se concentrent sur trois axes :

- La reconnaissance de la parole :
  Transformation d'un signal en une suite de mots afin d'effectuer de la détection d'opinions, de concepts applicatifs, d'entités nommées...
- La caractérisation de la parole :
  Segmentation et regroupement en locuteurs, identification et vérification du locuteur, identification de la langue et détection des émotions.
- La traduction automatique :
  Traduction d'un texte ou d'un enregistrement sonore grâce à un programme informatique sans l'intervention humaine.

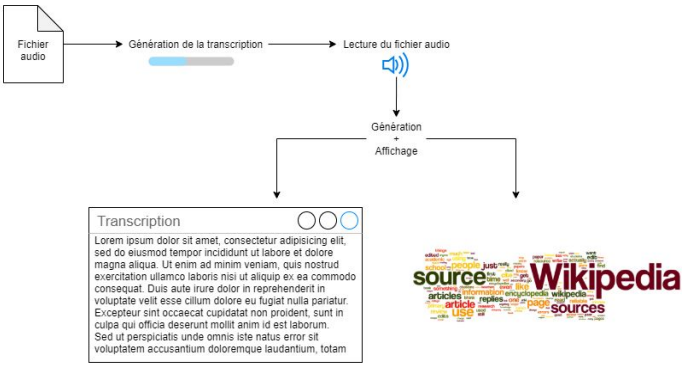
Afin de permettre à l'équipe LST d'effectuer des démonstrations de leurs travaux lors de conférences ou de présentations, nous allons mettre en place un service web permettant d'exposer ces travaux.

Ce service web doit permettre de :

• Lancer un décodage (soumission d'un fichier audio, sélection de la langue (français / anglais), affichage de l'avancement, récupération de la transcription automatique),

• Envoyer le résultat du décodage vers un module d'extraction d'informations sémantiques,

• Afficher la transcription automatique « enrichie » ainsi que le nuage de mots correspondant.



II - Analyse technique et fonctionnelle

II.1 - Besoins fonctionnels

Afin de mener à bien ce projet, nous avons défini plusieurs User Stories correspondant aux différents besoins fonctionnels.

US1	En tant qu'utilisateur, je souhaite générer la transcription d'un fichier audio.
US2	En tant qu'utilisateur, je souhaite afficher la transcription d'un fichier audio.
US3	En tant qu'utilisateur, je souhaite générer le nuage de mots correspondant à la transcription.
US4	En tant qu'utilisateur je souhaite afficher le nuage de mots correspondant à la transcription.
US5	En tant qu'utilisateur, je souhaite pouvoir m'identifier.
US6	En tant qu'administrateur, je souhaite pouvoir gérer les différents modules.
US7	En tant qu'utilisateur, je souhaite pouvoir gérer l'affichage des modules.

II.2 - Maquette

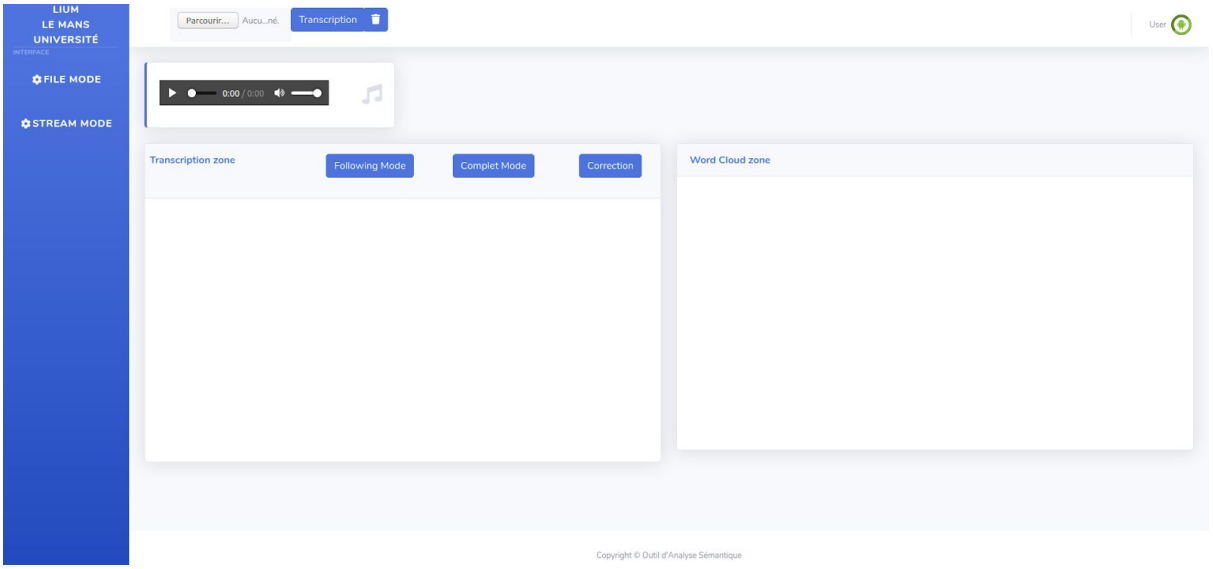


Figure 2.1 - Première version du service web

II.3 - Choix de conception

Nous avons effectué différents choix de conception lors de ce projet. Premièrement, afin de pouvoir nous concentrer sur la partie importante du projet, nous avons décidé d'utiliser un template Bootstrap. Ce qui nous a permis de gagner du temps sur la partie front-end et de nous concentrer sur le back-end.

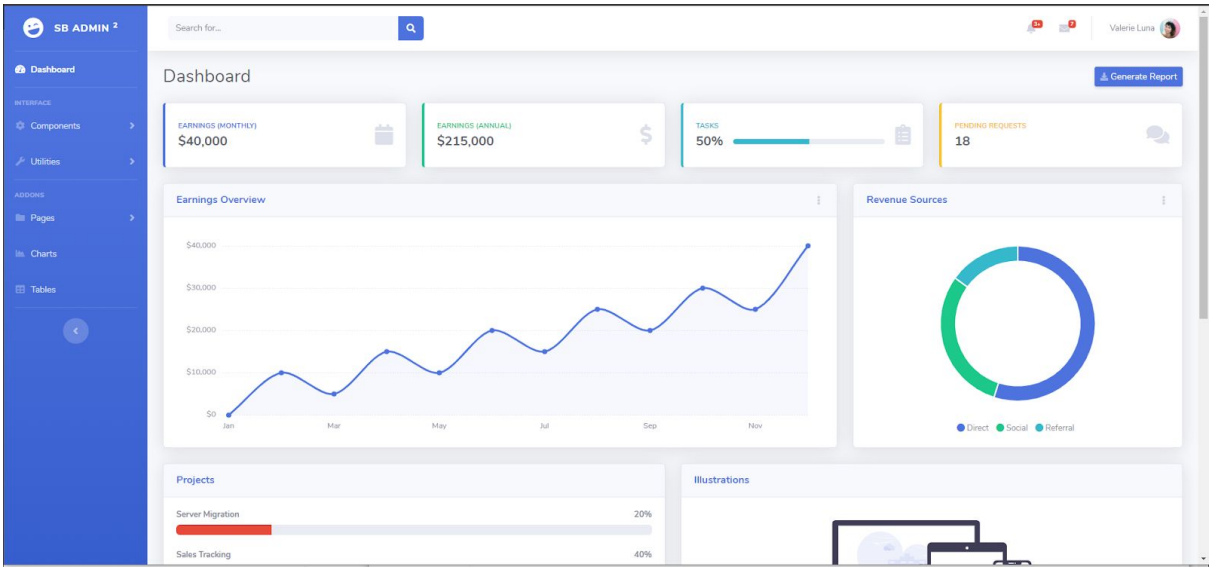


Figure 2.2 - Template Bootstrap initial

Afin de mettre en place ce service web, nous avons accès à l'API de traitement d'un fichier audio, mise en place par l'équipe LST.
Cette API permet de transcrire un fichier audio en version texte. Nous récupérons à partir de celle-ci des fichiers contenant le texte transcrit (en format .xml, .txt, …).

De plus, un projet sur la génération de nuages de mots nous a été fourni. Ce projet a été développé par des étudiants de Master 1 Informatique de l'année 2018/2019 et permet de générer un nuage de mots à partir de fichiers .cvs et .xlsx.
Nous avons donc retravaillé ce projet afin de l'adapter au service web et qu'il puisse générer des nuages de mots à partir des fichiers (.xml, .txt) que nous retourne l'API de transcription.
Ce projet est majoritairement codé en JavaScript.

De notre côté, pour mettre en place le service web nous travaillons principalement avec du JavaScript pour développer les différentes fonctionnalités.
Nous appelons l'API de transcription pour envoyer et récupérer la transcription du fichier audio choisi (via la technologie REST). Pour générer le nuage de mots, nous utilisons le projet nuage de mots (voir ci-dessus) que nous avons adapté pour ce service web.

L'accès aux différentes fonctionnalités du service web est réglementé par deux statuts distincts : administrateur et utilisateur.
Une personne avec le statut d'administrateur aura la possibilité de gérer les différents modules. Il pourra changer les adresses des services utilisés, supprimer des modules/services ou encore ajouter de nouveaux modules.
Contrairement à un administrateur, un utilisateur aura seulement la possibilité, dans les paramètres, de cocher l'option d'affichage des modules sur la page principale.
Les deux statuts n'ont pas de restriction d'utilisation sur la page principale. Ils pourront tous deux transcrire un fichier audio ou un streaming et afficher les informations demandées.

III - Conception

III.1 - API & Transcription

Nous utilisons l'API mise à notre disposition par le LIUM par le biais de requêtes AJAX. Ce qui nous permet d'envoyer des fichiers (ici, de type audio) au serveur, mais également de supprimer un fichier, ou encore de récupérer le contenu de la transcription.

Exemple d'une requête AJAX :

```
$.ajax({
  url: 'http://lst-demo.univ-lemans.fr:8000/api/v1.1/files',
  type:'POST',

  dataType: 'json',
  enctype: 'application/json',
  autoUpload: true,
  processData: false,
  contentType: false,//"{'start':true, 'asr_model_name':'french.studio.fr_FR'}",
  start: true,
  asr_model_name: 'french.studio.fr_FR',
  data: form,

  headers: {
    "Authentication-Token": token,
  },

  success: function(resultat){
    console.log('Upload: success')
    //loadDoc(resultat.id);
    //modifAudio(resultat.id);

  },
  error: function(resultat){
    console.log('Upload: Error')
  },
});
```

Ici nous avons un exemple d'une requête AJAX permettant d'envoyer un fichier via la méthode POST.

Ensuite, en fonction des évènements (success, error, onload…) retournés par la requête, on peut appeler des méthodes pour récupérer les données sous format texte ou XML.

Comme vu précédemment nous récupérons donc la transcription sur le serveur du LIUM. Ensuite nous récupérons les données (ici le texte correspondant au fichier audio) que nous affichons sur l'écran de l'utilisateur.
Pour cela nous devons transformer les données reçues en XML. En effet, la requête AJAX nous renvoie les données soit en version Text, soit en String.

On peut ajouter que, dans le premier mode mis en place, si l'utilisateur lance le fichier audio, un curseur met en avant le segment de texte correspondant au temps d'avancement de l'audio. Ce qui permet aux autres utilisateurs de suivre l'avancée sur le texte plus facilement.
Pour mettre en place ce suivi, nous avons imité le principe des karaokés où le texte suit l'avancement de la chanson.

III.2 - Nuage de mots

Pour la partie nuage de mots, nous utilisons le projet développé par des étudiants de Master 1 Informatique de l'année 2018/2019. Ce projet utilise l'outil TreeTagger, qui permet de lemmatiser les différents mots. Il utilise également le package *WordCloud2* pour générer un nuage de mots. Ce dernier permet d'obtenir un résultat comme sur la figure 3.1.

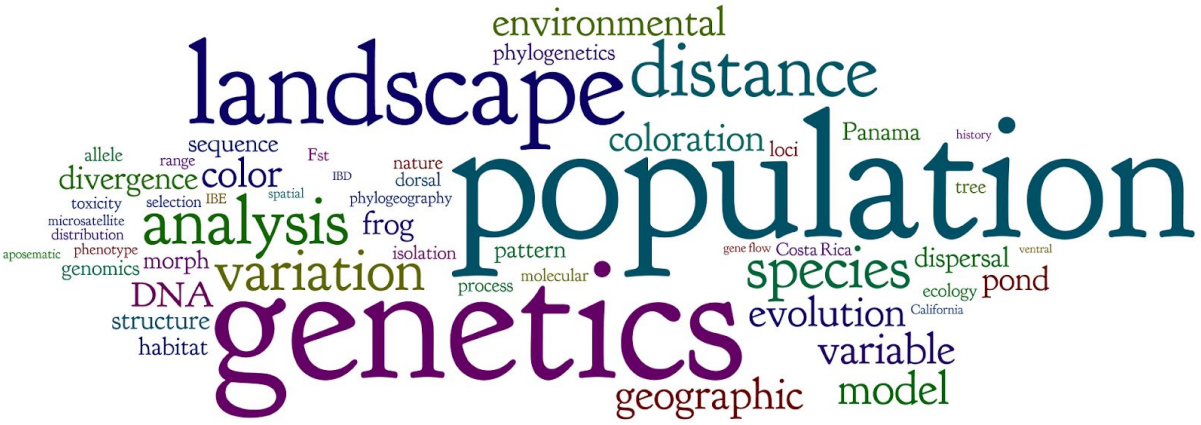


Figure 3.1 - Nuage de mots

Dans un premier temps, nous avons parcouru le code du projet des étudiants de l'année précédente, afin de comprendre son fonctionnement. Dans ce projet, le texte entré est d'abord lemmatisé, ensuite le nombre d'occurrences de chaque lemme est compté, afin d'attribuer une taille de police à chacun. Puis le nuage de mots est créé.

C'est lors de la fin du second sprint que nous avons réussi à mettre en place le nuage de mots. Dans un premier temps, nous avons intégré le nuage de mots sous forme d'une API et en JavaScript. Ensuite nous avons dû changer de direction dans notre conception, puisque nous devions utiliser le module du nuage de mots comme un service web. Nous sommes donc passés du Javascript au Python pour gérer ce module. Nous avons également mis en place une blacklist, afin d'empêcher l'apparition de certains mots dans le nuage de mots. Celle-ci se présente sous la forme d'une fichier *blacklist.txt* dans le code source du projet et peut être modifié afin d'ajouter ou de retirer des mots en fonction des besoins.

Sous Javascript, nous utilisons l'API *WordCloud2*, qui était utilisée dans le projet des étudiants de l'année précédente.
Sous Python, nous utilisons l'API *wordcloud*, pour générer le nuage de mots. Nous utilisons également la librairie "Flask" pour gérer la partie service web. L'API *wordcloud* est une adaptation de l'API *WordCloud2* en Python.

III.3 - Identification et paramétrages

For the authentification system, a skeleton for the server and client side has been created. In the beginning, I explored several new frameworks that I needed to create a further application. First, I prepared the server part which receives and processes requests from the client part. Queries mostly work with the database. I used MongoDB as the database, and Node.js, Passport.js, passport-jwt (to create a user session), Mongoose, jsonwebtoken, CORS, express and bcryptjs (for password hashing) to work with it. For the client side, I used the Angular CLI and some of its internal libraries. I divided the project into components so that the code would look more clear and more convenient to use. Inside each component, I created functions that send requests to the server when the client clicks certain buttons. Then, I took the HTML page made by my partners and integrated it into the client part. Inserting click functions, inside the buttons of the page.

Malheureusement, en raison de différents problèmes, cette partie n'a pas pu être intégrée au produit final.

Les paramètres ont été mis en place sous forme d'un fichier JSON qui stocke les informations de chaques modules.
L'administrateur, une fois devant les paramètres, peut modifier l'urls de chaque module comme il le souhaite. Par la suite, les changements seront enregistrés dans le fichier JSON. Ceci permet d'avoir une sauvegarde permanente des paramètres, et également de faciliter l'ajout et la suppression de nouveaux modules dans le futur.

IV - Gestion de Projet

IV.1 - Méthode Agile

Ce projet a été réalisé en suivant une méthode Agile nommée Scrum. L'objectif de cette méthode est de découper le temps imparti pour la réalisation du projet en plusieurs *sprints* (d'une durée d'une semaine à un mois). A la fin de chaque *sprint*, une démonstration fonctionnelle du projet doit être présentée au client, afin de s'assurer que l'avancement du projet correspond aux attentes et aux besoins du client. Les retours du client sont ensuite pris en compte pour les *sprints* suivants.

Dans le cahier des charges, et durant les réunions avec les responsables du projet, nous avons donné des priorités (de 1 à 4) à nos user stories. Chaque priorité correspond initialement à chacun des *sprints*.

IV.2 - Planning

Ce projet a été découpé en plusieurs *sprints* d'une durée d'un mois. Chaque *sprint* correspond à une "version" du site.
Le premier *sprint*, se terminant fin janvier, doit permettre d'avoir un projet fonctionnel avec les features définies avec une priorité 1 dans le cahier des charges et ainsi de suite pour les autres *sprints*.

Figure 4.1 - Diagramme de GANTT prévisionnel

Le travail effectué par chaque membre du groupe lors du projet est représenté ci-dessous sous forme d'un diagramme de GANTT.

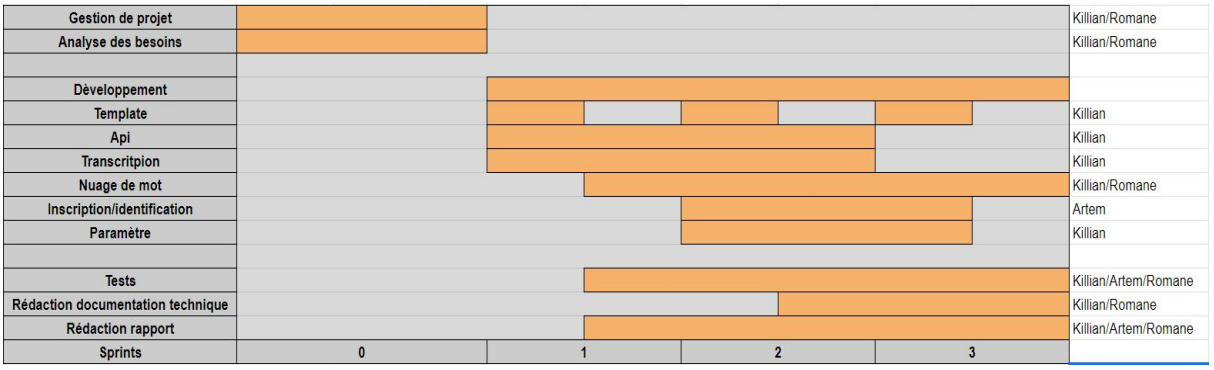


Figure 4.2 - Diagramme de GANTT effectif

IV.3 - Outils de gestion de projet

Afin de planifier les différentes tâches et d'avoir une vision globale de l'avancée du projet, nous avons utilisé Trello, qui est un outil de gestion de projet en ligne.

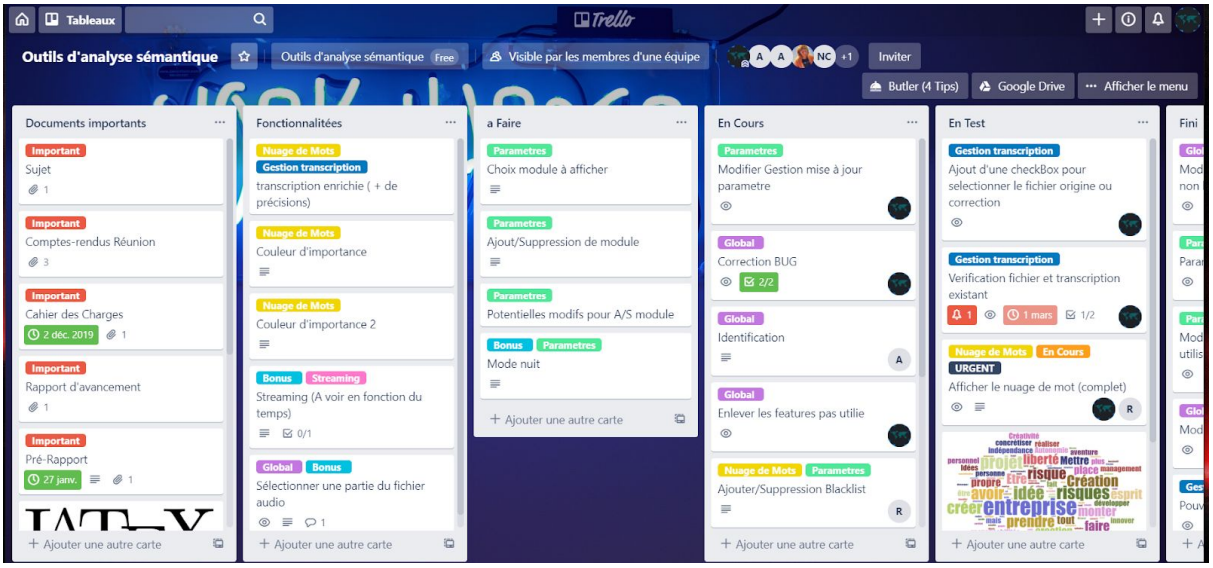


Figure 4.3 - Trello

De plus, de façon à réunir le travail de chaque membre du groupe, nous avons utilisé GitHub, ce qui a permis de centraliser tout le code du projet.
En ce qui concerne l'écriture du cahier des charges, du rapport ainsi que du manuel utilisateur, nous avons utilisé l'outil Google Drive.
Un Discord a été mis en place afin de pouvoir communiquer entre nous et pouvoir s'entraider à distance en cas de besoin.

IV.4 - Répartition des tâches

Les différentes tâches ont été réparties entre les différents membres du groupe en fonction de leurs compétences. Le tableau ci-dessous montre le degré de participation à l'avancement du projet de chaque membre du groupe.

Activité	K.Marchand	R.Buon	A.Tarasenko	*Total*
Gestion de projet	*50*	*50*	*0*	100
Analyse des besoins	*50*	*50*	*0*	100
Analyse & conception	*50*	*50*	*0*	100
Codage	*42.5*	*42.5*	*15*	100
Réalisation BD, requêtes	.	.	.	.
Réalisation IHM, CSS, images	60	30	10	100
Tests	*40*	*40*	*20*	100
Rédaction documentation technique	50	50	0	100
Rédaction rapport	*50*	*50*	*0*	100
Travail global réalisé	**49.6**	**45.3**	**5.1**	100

Figure 4.4 - Répartition des tâches

IV.5 - Retours sur les sprints

IV.5.A - Sprint 1

Le mois de janvier correspondait à notre premier sprint, où nous devions mettre en place les "features" de priorité 1, qui composent la première version du projet. Nous sommes parvenus à mettre en place une grande partie des features que nous avions prévu de développer pendant ce *sprint*. En effet, beaucoup d'entre elles ont été mises en place mais certaines ont pris plus de temps que prévu pour être développées.

Premièrement, le nuage de mots n'a pas pu être mis en place. En effet, la prise de connaissance du projet mis à notre disposition et la transformation de ce dernier nous ont pris plus de temps que prévu.

Deuxièmement, du côté de la transcription, nous avons pu mettre en place les fonctionnalités principales. La seule difficulté que nous avons rencontré sur cette partie venait de la connexion avec l'API. C'est-à-dire, que nous arrivions à envoyer un fichier sur le serveur, mais la transcription ne se lançait pas directement. Pour contrer ceci, dans ce premier *sprint*, nous avons pris la décision de mettre les

fichiers utilisés en dur dans le code et de remettre à plus tard la possibilité de choisir un fichier dans l'ordinateur de l'utilisateur.

IV.5.B - Sprint 2

Pour ce sprint, il était prévu d'ajouter diverses fonctionnalités. Premièrement, l'affichage de la transcription et du nuage de mots en pas à pas (c'est-à-dire que la partie affichée correspond seulement à la partie du fichier audio déjà jouée). Ensuite, la gestion de différents paramètres, l'identification ainsi que le curseur de suivi sur la transcription.

Au début de ce sprint, nous avons pu mettre en place des éléments concernant les paramètres. De plus, comme le nuage de mots n'était pas fonctionnel à la fin du premier sprint, nous avons décidé de mettre la priorité sur cet aspect afin de rattraper le retard.

En fin de sprint, une première version du nuage de mots à pu être intégrée en JavaScript, ce qui nous permettait de générer les nuages de mots dont nous avions besoin et qui correspondaient à la transcription du fichier audio.
De plus, le mode de suivi de la transcription a été modifié. Nous n'affichons plus le texte mot à mot, mais segment par segment. Ce qui nous a permis de contrer quelques bugs.

IV.5.C - Sprint 3

Ce dernier sprint a commencé par la mise en place du nuage de mots en tant que service web. Nous avons donc dû revoir l'architecture de ce dernier. Nous sommes passés de JavaScript à Python (Flask) pour créer ce nouveau service web.
De plus, nous avons ajouté une "blacklist", qui permet d'empêcher l'affichage de certains mots dans le nuage de mots.

En parallèle, la partie transcription est arrivée à sa fin. Cependant, il y avait quelques bugs à régler et des modifications à apporter. Le mode correction a été intégré assez rapidement. Le mode de suppression des fichiers sur le serveur est modifié, il passe de la suppression d'un seul fichier à la possibilité de supprimer plusieurs fichiers simultanément.

A présent, un utilisateur pourra choisir n'importe quel fichier audio, puisque nous récupérons ce fichier que nous téléchargeons directement sur la machine pour que notre lecteur audio puisse être mis à jour.

V – Problèmes rencontrés

Durant ce projet, nous avons rencontré quelques problèmes, ce qui nous a fait prendre du retard que nous avons dû rattraper par la suite.

V.A - Gestion de projet

Du point de vue gestion de projet, nous avons commencé le projet au nombre de deux. Où nous avons (Killian et Romane) rédigé le cahier des charges, en analysant les besoins exprimés par nos responsables via le sujet et les réunions passées. En ce qui concerne le cahier des charges, il y a eu plusieurs aller-retours entre les responsables du projet et le groupe, afin de corriger les différentes erreurs, ainsi que pour avoir le cahier des charges le plus représentatifs possible du produit final.
Lors de la phase de développement, le groupe est passé de deux à trois membres.

V.B - Transcription et API

Les quelques soucis rencontrés étaient des problèmes liés à l'API, principalement avec des soucis liés au CORS.

Nous avons également dû faire un choix sur l'affichage du texte et du curseur, où nous avions un bug lié à l'appel de l'événement avec l'audio, où les informations ne s'affichaient pas forcément. Pour remédier à cela, nous avons choisi non pas d'afficher le texte mot par mot mais segment par segment.

Un des plus gros problèmes que nous avons rencontré était lié au lecteur audio, où pour modifier le fichier audio en fonction du fichier choisi il fallait que ce fichier audio soit en local dans notre dossier source du projet.

Concernant l'audio, nous avons dû mettre en place un système de stockage en local dans les dossiers du projet car nous n'arrivions pas à récupérer le PATH de notre fichier. Du coup, nous copions notre fichier dans le dossier "audio" puis nous modifions notre source du fichier audio par ce fichier.

V.C - Nuage de mots

Le nuage de mots à tardé à être mis en place, puisque nous avons dans un premier temps voulu chercher un système/API sur internet, ce qui nous aurait permis de gagner du temps et de nous concentrer sur le passage en service web.

Nous nous sommes concentrés dans un second temps sur le projet effectué par des étudiants de l'année précédente qui avait été mis à notre disposition. Nous avons passé un peu de temps à le comprendre. Ce qui nous a permis de mettre en place une première version du nuage de mots en version API et en JavaScript à la fin du *sprint* 2.

Dès le début du *sprint* 3, nous nous sommes concentrés sur la transformation du nuage de mots déjà développé, pour le passer sous Python et mettre en place le service web.

Nous avons travaillé avec *bokeh_wordcloud2* qui est une API qui génère un nuage de mots. Le problème était que nous ne pouvions pas récupérer une image correcte du nuage de mots pour l'afficher dans notre application via le service web. C'est pourquoi nous avons trouvé une nouvelle API nommée *word_cloud*, qui nous offre la possibilité d'obtenir notre nuage de mots sous le format d'une image en PNG.

Pour effectuer l'envoi par le service web, nous avons dû convertir l'image au format Base64 pour qu'elle puisse être reconstruite correctement en sortie via notre application en JavaScript.

V.D - Choix

Lors du troisième sprint, nous avons dû choisir quelles fonctionnalités prioriser, car nous n'aurions pas le temps de tout mettre en place. Nous avions, lors du sprint 0, donné des priorités à chaque User Story dans le cahier des charges. Ainsi, lors de la prise de décision du troisième sprint nous avons décidé de développer en priorité les User Stories ayant une priorité plus importante.

VI – Glossaire

API : Interface de programmation d'application qui sert de façade par laquelle un logiciel offre des services à d'autres logiciels. (Wikipédia)
AJAX : Permet de construire des applications Web et des sites web dynamiques interactifs sur le poste client en se servant de différentes technologies des navigateurs web. (Wikipédia)
Angular CLI : Interface en lignes de commandes pour Angular. (cli.angular.io)
Back-end : Désigne un étage de sortie d'un logiciel devant produire un résultat. On l'oppose au front-end qui lui est la partie visible de l'iceberg. (Wikipédia)
Bootstrap : Collection d'outils utiles à la création du design de sites et d'applications web. (Wikipédia)
CORS : Mécanisme qui permet à des ressources restreintes d'une page web, d'être récupérées par un autre domaine extérieur au domaine à partir duquel la première ressource a été servie. (Wikipédia)
Features : Caractéristiques.
Flask : Framework open-source de développement web en Python. (Wikipédia)
Front-end : Correspond aux productions HTML, CSS et JavaScript d'une page internet ou d'une application qu'un utilisateur peut voir et avec lesquelles il peut interagir directement. (Wikipédia)
JSON : format de données textuelles dérivé de la notation des objets du langage JavaScript. Il permet de représenter de l'information structurée comme le permet XML. (Wikipédia)
Lemmatiser : Opération qui consiste à regrouper les occurrences d'un texte sous des adresses lexicales. (universalis.fr)
Lemme : Fait référence à un mot, de quelque nature que ce soit, qu'il soit composé ou simple, à condition qu'il puisse être référencé dans un dictionnaire. (linternaute.fr)
LIUM : Laboratoire d'Informatique de le Mans Université.
LST : Équipe Language and Speech Technology du LIUM.
Méthode Agile : Groupes de pratiques de pilotage et de réalisation de projets. (Wikipédia)
MongoDB : Système de gestion de bases de données orienté documents. (Wikipédia)
Node.js : Plateforme logicielle libre en JavaScript orientée vers les applications réseau événementielles hautement concurrentes qui doivent pouvoir monter en charge. (Wikipédia)
Scrum : Méthode de gestion de projet Agile.
Sprint : Phase de développement de la méthode Scrum, d'une durée déterminée.
Service web : Protocole d'interface informatique de la famille des technologies web permettant la communication et l'échange de données entre applications et systèmes hétérogènes dans des environnements distribués. (Wikipédia)

User Story : Description simple d'un besoin ou d'une attente exprimée par un utilisateur et utilisée dans le domaine du développement de logiciels et de la conception de nouveaux produits pour déterminer les fonctionnalités à développer. (Wikipédia)
XML : Métalangage informatique de balisage générique qui est un sous-ensemble du Standard Generalized Markup Language. (Wikipédia)

VII – Bibliographie

Bokeh_wordcloud2 [en ligne]. 2019.
Disponible sur : https://bokeh-wordcloud2.readthedocs.io

CodePen [en ligne].
Disponible sur : https://codepen.io/

CodePunker [en ligne]. 2013.
Disponible sur : https://www.codepunker.com/

CodeSpeedy [en ligne].
Disponible sur : https://www.codespeedy.com

Developpez.com [en ligne]. 2000.
Disponible sur : https://www.developpez.com/

Flask [en ligne]. 2010.
Disponible sur : https://flask.palletsprojects.com

MDN Web Docs [en ligne]. 2005. Web technology for developers.
Disponible sur : https://developer.mozilla.org/en-US/docs/Web

Open Classrooms [en ligne]. 1999.
Disponible sur : https://openclassrooms.com/

Programmation360 [en ligne].
Disponible sur : https://programmation360.com/

Programming Historian [en ligne].
Disponible sur : https://programminghistorian.org

Python.Doctor [en ligne].
Disponible sur : https://python.doctor/

Stack Overflow [en ligne]. 2008.
Disponible sur : https://stackoverflow.com/

W3Schools [en ligne]. 1999.
Disponible sur : https://www.w3schools.com

WayToLearnX [en ligne]. 2018.
Disponible sur : https://waytolearnx.com/

Wikipédia [en ligne]. 2001.
Disponible sur : https://fr.wikipedia.org/

VIII – Annexes

- Fiche d’avancement : https://docs.google.com/spreadsheets/d/1NqyGIg1klNGacTAKtnYtdhkT1C_Ttny48UOASV__VWc/edit?usp=sharing

- Dépôt GitHub : https://github.com/kilmhd/Outil_Analyse_Semantique