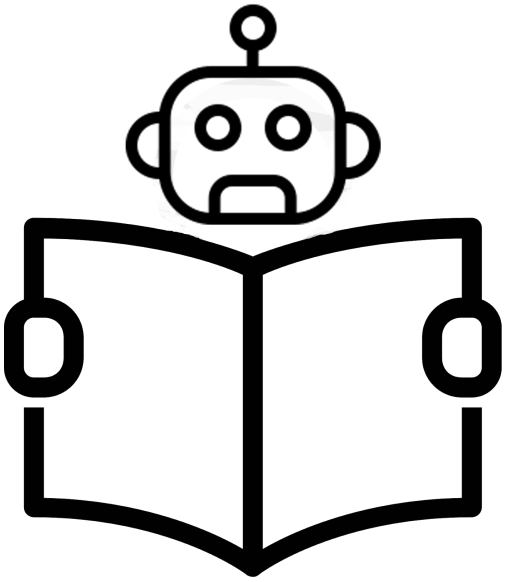
Projet informatique individuel : Solution de génération automatique de sitographie

---

Rapport final

version du 25/04/19



https://github.com/Fabien-Couthouis/projet_individuel_S8

Projet	Solution de génération automatique de sitographie
Auteur	Fabien Couthouis
Tuteur	M. Benjamin Clément
Date de début	25/01/2019
Date de fin	25/04/2019

Bordeaux INP ENSC

SOMMAIRE

I. Introduction

1. Contexte et rappel du projet

Lors du travail de rédaction d'un mémoire, d'une thèse, d'un article, et plus généralement d'un document s'inspirant de sources préexistantes, nous sommes amenés à écrire une bibliographie et/ou une sitographie. Celle-ci est indispensable afin de donner de la crédibilité à nos arguments, en citant les sources et les auteurs sur lesquelles nous nous appuyons pour avancer nos propos. Pour certains travaux, (thèse, article scientifique, rapport, ...), la bibliographie se doit d'obéir à certaines normes (APA, AFNOR, VANCOUVER).

Worland, J. (27 juillet 2015). Le risque d'inondation aux États-Unis pourrait être pire que nous le pensions. Consulté sur http://time.com/3973256/flooding-risk-coastal-cities/

Exemple d'une référence à un article correctement rédigée suivant la norme APA

La rédaction d'une bibliographie normée est un travail long, fastidieux, répétitif …. mais parfois indispensable. En partant du lien de l'article Internet en question, un humain peut rechercher sur la page web le nom de l'auteur, la date de publication, *etc*, afin de générer la références de l'article puis de réitérer sur l'ensemble des articles qu'il souhaite référencer afin de construire sa sitographie. Pourquoi ne pas demander à un ordinateur d'automatiser ce processus ?

Le but initial du projet était de créer un site Internet visant à générer automatiquement une sitographie normée à partir d'une liste d'urls fournie par l'utilisateur, en allant chercher les informations nécessaires à chaque référence (nom de l'auteur, titre, date de publication). Le projet a ensuite évolué afin de permettre également la gestion de ses webographies : création de comptes utilisateurs, création de webographie, ajout de référence, édition, ...

Objectif initiale : entrée fournie par l'utilisateur et sortie générée par le site

2. Technologies employées

Le langage Python est un langage que je trouve agréable à utiliser. De plus, de nombreuses librairies open-source sont disponibles dans ce langage. C'est pourquoi je me suis naturellement tourné vers le framework Django afin de développer le site. Bootstrap et Javascript ont été utilisés pour le front-end, puisque déjà utilisés pour le projet web du S5.

3. Existant

Il existe des logiciels d'aide à la rédaction bibliographique (Zotero, Mendeley, Endnote). Ces logiciels doivent être installés sur un ordinateur et il est difficile d'avoir accès à ses sources sur plusieurs appareils. Un seul outil en ligne a été trouvé pour rédiger les références de sources en français et en anglais, mais cet outil ne permet pas d'opérer sur une liste de liens et certaines actions doivent être exécutés manuellement. De plus, cet outil ne permet pas la gestion de ses webographies. Il est de plus impossible de récupérer les sources au format Bibtex. L'algorithme ne permet en outre pas de traiter les articles au format pdf.

II. Spécification générale

1. Analyse fonctionnelle

Les exigences fonctionnelles remplies par le site sont résumées dans le tableau ci-dessous :

Code	Description
EF_01	La sitographie en sortie est rédigée selon le format choisie par l'utilisateur (APA ou Bibtex)
EF_02	Le site permet la gestion des références web (page html sur Internet) et de références pdf (article au format pdf sur Internet)
EF_03	Le site permet la gestion de comptes utilisateurs (inscription/connexion)
EF_04	Le site permet la gestion de ses webographies et de ses références : - ajout/renommage/suppression de webographie - ajout/édition/suppression de références
EF_05	L'utilisateur doit pouvoir entrer indifféremment des sources en français ou en anglais
EF_06	En cas d'information manquante, le processus n'est pas interrompu et l'information est remplacée par une chaîne vide
EF_07	L'interface web est intuitive et simple d'utilisation (voir 2. Analyse des besoins pour l'interface utilisateur)

2. Liste des modèles Django utilisés

Modèle	Description
Webography	Modèle utilisé pour la gestion des webographies. Permet notamment de générer un webographie à partir d'une chaîne de caractères contenant des urls. Chaque webographie est associée à un utilisateur (qui peut être *Null*).
Reference	Classe permettant d'abstraire les méthodes communes aux références web et pdf.
ReferenceWeb	Dérive de Reference. Permet la recherche d'informations (nom de l'auteur, date de publication, titre) dans une page web sur Internet. Utilise notamment BeautifulSoup afin d'explorer le fichier html récupéré sur Internet grâce au module Request de Python.
ReferencePDF	Dérive de Reference. Permet de récupérer directement la référence d'un article pdf au format BibTex. Pour se faire, on fait appel à la librairie pdftitle pour trouver le titre de l'article. On utilise ensuite la librairie Sscholar, permettant de rechercher ce titre dans le moteur de recherche académique SemanticScholar et de récupérer la référence, via web-scrapping (et la librairie Selenium).

3. Schémas et diagrammes UML

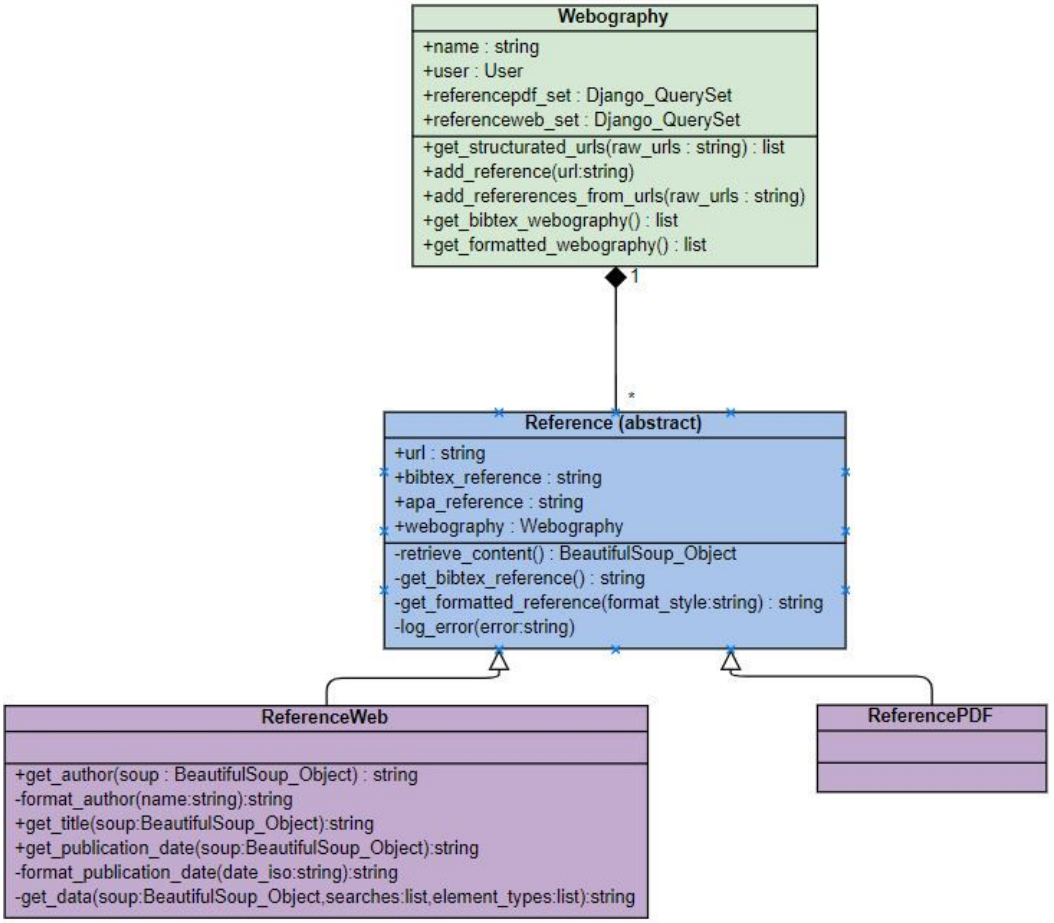


Diagramme des classes (Django models)

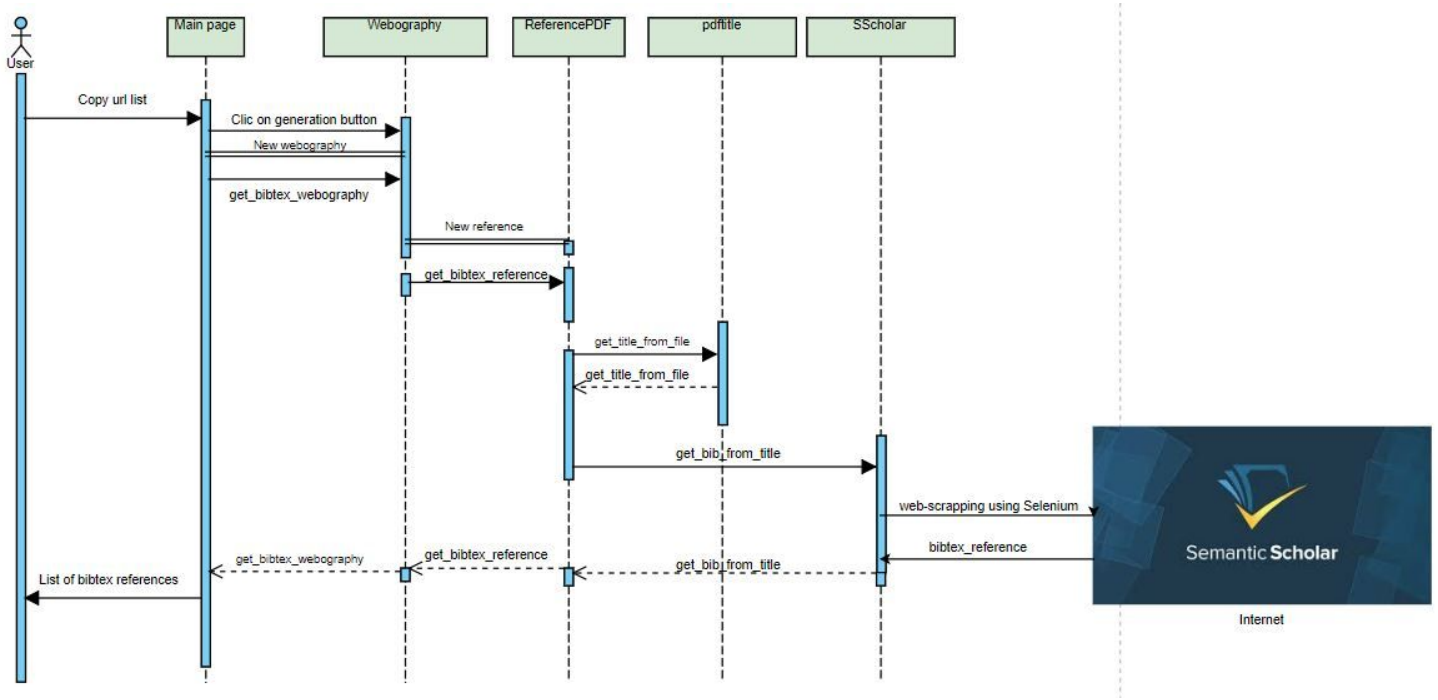


Diagramme de séquence : création d'une webographie depuis la page principale (liste d'urls), cas d'une référence pdf

Ce diagramme illustre la génération d'une webographie, depuis l'entrée d'une liste d'urls depuis la page principale. Il clique ensuite sur le bouton "Générer". Une webographie est créée. Pour chaque url de la liste entrée, une nouvelle référence est créée (le type web ou pdf est déterminée automatiquement, ici on considère une référence pdf). Pour cette référence pdf, on détermine son titre via la librairie pdftitle. Puis via la librairie Sscholar on effectue du web-scrapping pour créer une requête avec ledit titre et ainsi obtenir la référence bibtex sur le site. Le navigateur est émulé en utilisant la librairie Selenium.

Le fonctionnement pour une référence web est quasiment similaire. Seule la fin change : au lieu d'utiliser la librairie Sscholar, on utilise BeautifulSoup afin de faire une recherche dans le document html et de trouver les informations manuellement.
Pour plus d'informations sur les librairies utilisées, voir : III.1 Dépendances et librairies externes.

3. Analyse des besoins pour l'interface utilisateur

Les utilisateurs finaux sont les personnes rédigeant tout type de rapport devant faire apparaître des sources normées (mémoire ou thèse par exemple). Néanmoins, dans ce projet informatique individuel, par mesure de simplicité, je me suis limité à ma propre utilisation du site, en tant qu'utilisateur final aisément accessible. Le site est donc avant tout pensé pour mon utilisation mais il est tout à fait possible que d'autres utilisateurs aient la même utilisation que la mienne.
Citer ses sources au format approprié n'est pas forcément une partie de plaisir, c'est pourquoi le site doit permettre de le faire en minimisant le temps passé et les efforts de l'utilisateur.

Ces constatations se traduisent dans l'interface par :

- une information accessible sans trop effort (utilisation d'icônes)
- un export facilité (bouton de copie rapide dans le presse papier)
- la détection automatique du type de source (pdf/web)
- un passage aisé d'un format à l'autre (onglet pour passer du format APA à Bibtex)
- un design sobre afin que l'attention se focalise sur les informations essentielles
- peu de pages et une utilisation de popups afin que l'utilisateur ne se perde pas sur le site
- le même type d'écrans pour l'ajout/édition de webographie ou de référence afin de minimiser le temps d'apprentissage de l'interface

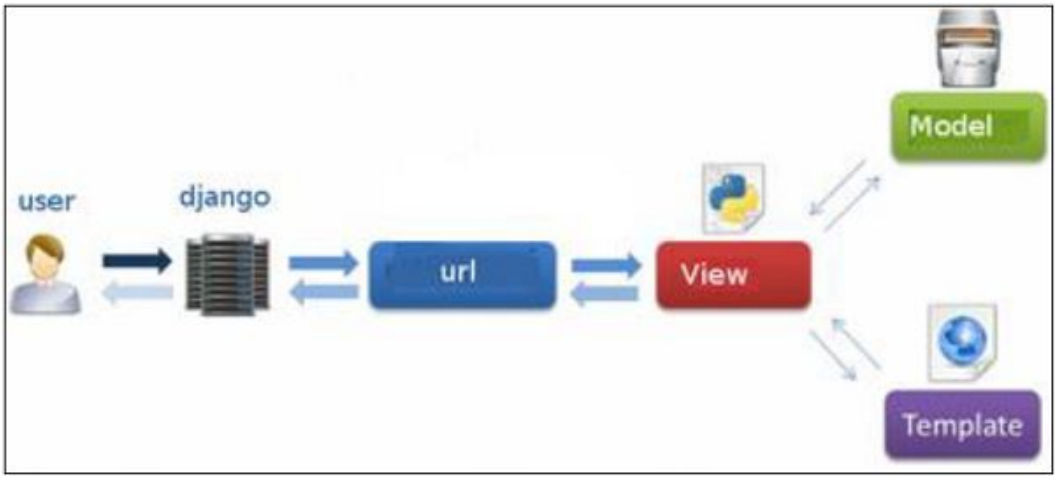
voir V.4 Captures des moments clés

Cette analyse du besoin a été établie en se basant sur mon ressenti personnel. Si le produit avait pour objectif d'être commercialisé, il aurait été intéressant de pousser celle-ci et de se confronter à d'autres utilisateurs potentiels, dans le cadre d'une conception centrée utilisateur (réalisation de questionnaires, focus groups, tests utilisateurs, …) .

4. Django et l'architecture MVT

L'architecture des sites dévelopés sous Django suivent le modèle MVT (pour *Model View Template*), dérivé du plus classique modèle MVC (*Model View Controller*).
La principale différence entre les deux modèles est la gestion de la partie Controller en interne dans Django (permettant les interactions *Model /View*). *Template* est un fichier html un peu différent, puisque intégrant le langage de Template propre à Django (*Django Template Language*).

En pratique, ce modèle offre un fichier html proche d'un fichier php, mais plus aisément maintenable, puisque le code de l'application se trouve dans le *Model*, la jonction se faisant dans la *View*. La séparation des responsabilité imposée par Django offre ainsi un développement plus agréable selon moi : pas besoin de réfléchir à une architecture, le code est facilement modifiable et organisé.



Architecture MVT (Model View Template) utilisée sous Django

III. Spécification détaillée

1. Dépendances et librairies externes

- Django : Framework web utilisé pour développer le site.

- beautifulsoup4 : Librairie pour parcourir les données d'un fichier html ou xml. Utilisé pour récupérer les informations d'une référence sur Internet (ReferenceWeb).

- gscholar : Librairie requettant Google Scholar afin de trouver une référence au format BibTex à partir du nom de l'article. Utilisée en première instance, je me suis rendu compte que le nombre de requêtes sur Google Scholar était limité afin d'éviter l'utilisation de robots. La librairie est néanmoins toujours utilisée si l'article n'est pas trouvé sur Semantic Scholar (cas rare).

- pdftitle : Librairie permettant de trouver le titre d'un article au format pdf. Utilise la librairie pdfminer afin de parcourir le texte du fichier. L'algorithme explore la première page du fichier et compare la taille et la police de chaque paragraphe. Le titre est à priori le paragraphe écrit en plus gros, en gras et situé le plus en haut de la première page. Les résultats offerts par cette librairie sont assez impressionnants : le titre est toujours trouvé pour les articles testés, le tout assez rapidement. À la suite d'une mise à jour de la librairie, une erreur est apparue : le titre était dépourvu de ses espaces. Comme cela me gênait pour la requête sur SemanticScholar, j'ai dû travailler sur un petit correctif, qui a été validé par l'auteur de la librairie et qui en fait désormais partie.

- pybtex et pybtex-apa-style : Permet la conversion du format BibTe au format Apa.

- requests : Permet les requêtes http depuis du code Python. Sert à récupérer le fichier pdf ou la page html à partir d'un lien.

- Sscholar : Librairie développée pour le projet permettant d'effectuer des requêtes sur le moteur de recherche académique Sscholar et de retrouver une référence BibTex à partir du titre d'un article. L'API fournie par le site ne permet pas la recherche d'article par site, d'où la nécessité de cette librairie. Utilise Selenium.

- Selenium : Librairie de web-scrapping permettant de simuler le fonctionnement d'un navigateur Internet (clic sur un bouton, etc…).

- urlextract : Permet d'extraire des urls à partir d'une chaîne de caractères.

Des librairies internes à Python ont également servies pour ce projet (Re pour les expressions régulières, Tempfile pour la gestion des fichiers pdf en tant que fichiers temporaires, …)

2. Justification des choix de conception

- Les modèles ont été placés dans des fichiers séparés afin d'en faciliter la relecture.

- Un modèle Reference, parent de ReferenceWeb et ReferencePDF a été créé pour éviter les redondances entre les classes (notamment les attributs et la conversion du format BibTex en APA).

- Les fichiers pdf téléchargés par le serveur sont stockés sous forme de fichiers temporaires (via tempfile). Ainsi, leur suppression gérée automatiquement après d'utilisation.

- La plupart des formulaires Django (excepté les formulaires d'inscription/connexion) ont été créés manuellement (ce ne sont pas des ModelForms) afin de pouvoir les personnaliser à mon gré via Bootstrap.

- L'API de SemanticScholar, bien qu'existante, n'a pas été utilisée car elle ne permet pas la requête via le titre d'un article.

- Dans le modèle ReferenceWeb, certaines informations ont nécessité un formatage (passage d'une date iso au format jour / mois / année, suppression des mots tels que "By" , "and", "with" pour le nom de l'auteur, …).

- Toujours dans ReferenceWeb, des blocs d'exception ont été utilisés pour détecter les erreurs. En cas d'erreur sur l'obtention d'une information, on continue le processus en cours et on renvoie une chaîne vide. L'utilisateur se chargera de compléter au besoin. Cela permet d'éviter une ribambelle de boucles "if" et de ne pas perdre le travail fait avant l'apparition d'une erreur.

- Si un article n'est pas trouvé sur SemanticScholar, la recherche est faire sur Google Scholar. Cela permet d'augmenter le taux de réussite en cas de référence présente sur un site et pas un autre. Comme cette situation n'est pas fréquente, le risque que Google Scholar bloque l'adresse IP est moindre.

IV. Gestion de projet

1. Un travail planifié

Un planning prévisionnel assez général et peu détaillé a été établi au démarrage du projet. Après une première semaine passée sur le projet, un planning prévisionnel plus détaillé a pu voir le jour. Ce planning prévisionnel a servi de fil rouge tout le long du projet et a été bénéfique pour plusieurs raisons :

- étude de faisabilité en début de projet
- réflexion sur l'ordre de réalisation des tâches en début de projet
- suivi de l'avancement
- source de motivation extrinsèque ("je dois finir X pour tenir le planning")
- respect des deadlines

Voir Annexe : Plannings.

Le planning a globalement été tenu tout le long du projet. Néanmoins, les améliorations finales et le design ont duré un peu plus longtemps que prévu (7 au lieu de 3 jours, dû à une mauvaise évaluation de la durée de la tâche). Ce temps a été rattrapé sur le bonus, qui a duré moins longtemps que prévu (voir V.2 Bonus : catégorisation automatique des références).

2. Une agilité inspirée des méthodes Scrum et Kanban

Ce projet étant individuel, la gestion de projet a été flexible et personnelle. Je me suis tout de même inspiré de certaines méthodes vues en cours, que j'ai adapté pour faciliter mon travail :

- **Méthode Scrum** : organisation en sprints. Chaque sprint doit se clôturer par l'achèvement d'un petit nombre de tâches à une date définie. Tout ce qui sort des tâches associées à un sprint doit attendre un prochain sprint.
- **Méthode Kanban** : amélioration continue. Refactorisation du code à chaque fin de sprint. Tableau présentant les états du flux de travail, avec historique du travail accompli, ainsi que le travail futur.

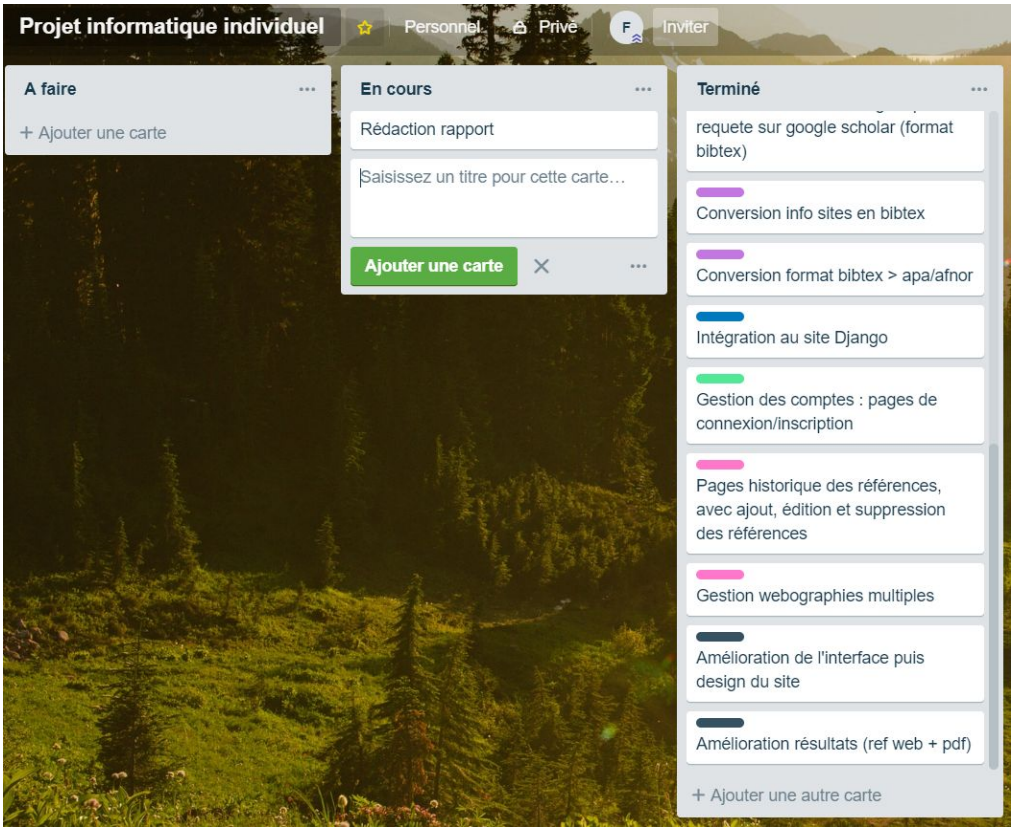


Tableau utilisé pour le suivi de projet (via Trello)

Le tableau reprend les tâches présentes dans le planning, présentées dans un format plus digeste et aisément manipulable. Chaque sprint correspond à une couleur d'étiquette.

3. Un suivi régulier

2 rendez-vous avec mon tuteur ont été organisés. Des échanges par mail et des états d'avancement réguliers ont également été mis en place. Cela a permis un retour extérieur de la part de mon tuteur, qui m'a notamment donné l'idée de la gestion de webographies, non prévu initialement.

4. Une gestion de projet efficace ?

Cette organisation s'est révélée assez libre mais tout de même structurée, ce qui m'a permis d'avancer à mon rythme, tout en restant dans les temps. Cependant, un peu trop de temps a été passé pour l'apprentissages des librairies de traitement du langage naturel, finalement pas utilisées puisque j'ai pu obtenir les références BibTex en requêtant un moteur de recherche académique, plutôt qu'en analysant le document. Même si ces connaissances sont toujours bonnes à prendre, une sur-anticipation peut se révéler être une perte de temps dans un projet. Heureusement ici, le temps de travail était libre et les délais plutôt larges, le projet n'a donc pas été impacté.

Cette organisation libre ne permet néanmoins pas de mesurer le temps effectif passé sur le projet.
Enfin, le risque de changement sur une brique externe du programme n'a pas été pris en compte en début de projet (ici changements sur la librairie pdftitle, voir III.1 Dépendances et librairies externes).

V. Tests et résultats

1. Protocoles de test

➢ Tests unitaires durant le développement

Certaines fonctionnalités jugées "à risque" - *comme l'obtention de référence BibTex* - ont été développées suivant une approche TDD (Test-Driven Development) afin de gagner du temps sur le débogage. Ces tests unitaires ont ensuite servi de tests de non régression lors de la modification de certaines parties du code (par exemple lors du passage de Google Scholar à SemanticScholar).

```
class ReferencePDFTest(TestCase):
    testUrl1 = "https://arxiv.org/pdf/1706.05507.pdf"
    testUrl2 = "https://arxiv.org/pdf/1212.5701.pdf"
    # no metadata
    testUrl3 = "https://nutritionj.biomedcentral.com/track/pdf/10.1186/s12937-019-0433-7"
    testUrl4 = "https://arxiv.org/pdf/1902.06865.pdf"
    testUrl5 = "https://arxiv.org/pdf/1711.08416.pdf"
    testUrl6 = "http://ciir.cs.umass.edu/pubfiles/ir-329.pdf"
    webography = Webography()
    webography.save()

    urls = [testUrl1, testUrl2, testUrl3, testUrl4, testUrl5, testUrl6]

    def test_get_bibtex_reference(self):
        article = ReferencePDF(webography_id=ReferencePDFTest().webography.id,
                               url=ReferencePDFTest().testUrl1)
        print(article.url)

        expected = '@article{alami2019factors, \ntitle = {Factors that influence dietary behavior toward irc
        self.assertEqual(article.bibtex_reference, expected)

    def test_get_apa_reference(self):
        article = ReferencePDF(url=ReferencePDFTest().testUrl3)
        expected = "[Alami et al., 2019] Alami, A., Sany, S. B. T., Lael-Monfared, E., Ferns, G. A., Tatari,
        print(expected)
        print(article.apa_reference)
        self.assertEqual(
            article.apa_reference, expected)
```

Exemple de tests unitaires effectués sur le modèle ReferencePDF

➢ **Test d'interface avec des scénarios d'utilisation**

- **Protocole de test de génération d'une webographie à partir d'une liste de liens sur la page principale**

- Se rendre sur la page principale du site.
- Copier une liste d'urls provenant de différents médias d'information et d'articles au format pdf.
- Cliquer sur le bouton "Générer".
- Vérifier les résultats obtenus pour les formats APA et BibTex

- **Protocole de test de gestion des références**

- Se rendre sur la page de gestion des références
- Créer une nouvelle webographie
- Ajouter une référence manuellement à celle-ci
- Ajouter 2 références par lot, sans compléter les champs, en laissant l'obtention automatique des informations
- Vérifier les résultats
- Éditer une référence
- La supprimer
- Renommer la webographie
- La supprimer

Ces différents tests ont permis un débogage rapide, une détection des erreurs aux endroits clés, la validation d'une nouvelle fonctionnalité et un gain de temps non négligeable (surtout les tests unitaires)

2. Bonus : catégorisation automatique des références

Cette partie bonus avait pour objectif de classer thématiquement des références à partir de leur titre et leur abstract. La stratégie imaginée était la suivante :

1. Récupération des données (*abstract* de chaque article des données d'entraînement)
2. Labélisation thématique manuelle des données selon plusieurs thèmes : (mathématique, biologie, physique, informatique, chimie, médecine, autre)
3. Normalisation des données (abstract + titre): tokenisation, retirer les mots courants (stop words : "the", "by", …), lemmatisation
4. Trouver les *x mots* (x étant un entier à définir) les plus courants grâce à la méthode TF-IDF : ce seront nos données d'entraînement une fois encodées (encodage *one-hot)*
5. Utiliser un algorithme de machine learning / réseau de neurone pour tenter une classification automatique en fonction des mots les plus courants de l'abstract.

Cependant, lors de l'étape 2., je me suis rendu compte qu'il était déjà difficile pour un humain d'effectuer cette catégorisation. En effet, les distinctions entre les thèmes sont parfois difficiles à faire puisque la plupart des articles sont pluridisciplinaires (par exemple, un article ayant pour thème la médecine se rapportera également à la chimie et la biologie dans la plupart des cas). De la catégorisation multi-classes aurait été peut être envisageable mais le temps nécessaire à la labellisation manuelle était trop important pour entrer dans le cadre de ce projet. Ces raisons m'ont poussé à me concentrer sur les améliorations du site plutôt que sur cette fonctionnalité bonus.

3. Lancement du site

Le site a été développé et testé dans un environnement Linux. Par conséquent, il n'est pas assuré de tourner sous un environnement différent.

Installation (depuis le terminal)

1. Cloner le dépôt
   `git clone https : //github.com/F abien − Couthouis/projet_individuel_S8.git`

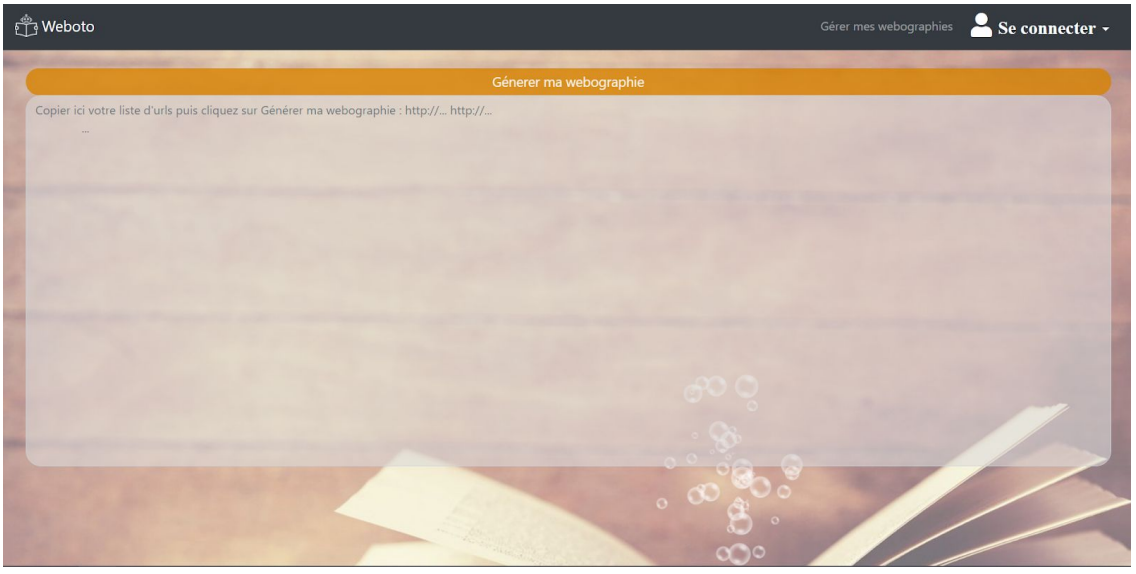
2. Entrer dans le dépôt :
   `cd projet_individuel_S8`

3. Installer les dépendances :
   `pip install − r requirements.txt`

Lancement

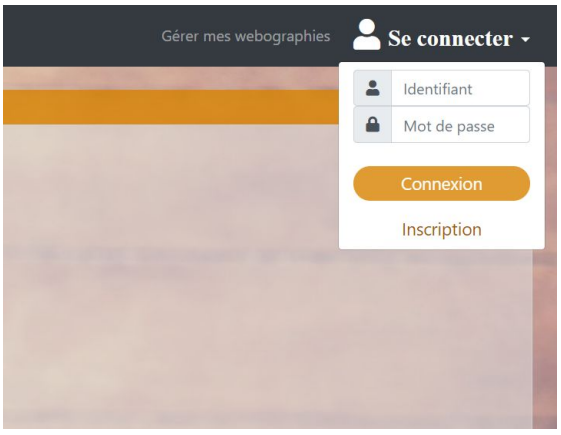
`python manage.py runserver`

4. Captures d'écran des moments clés

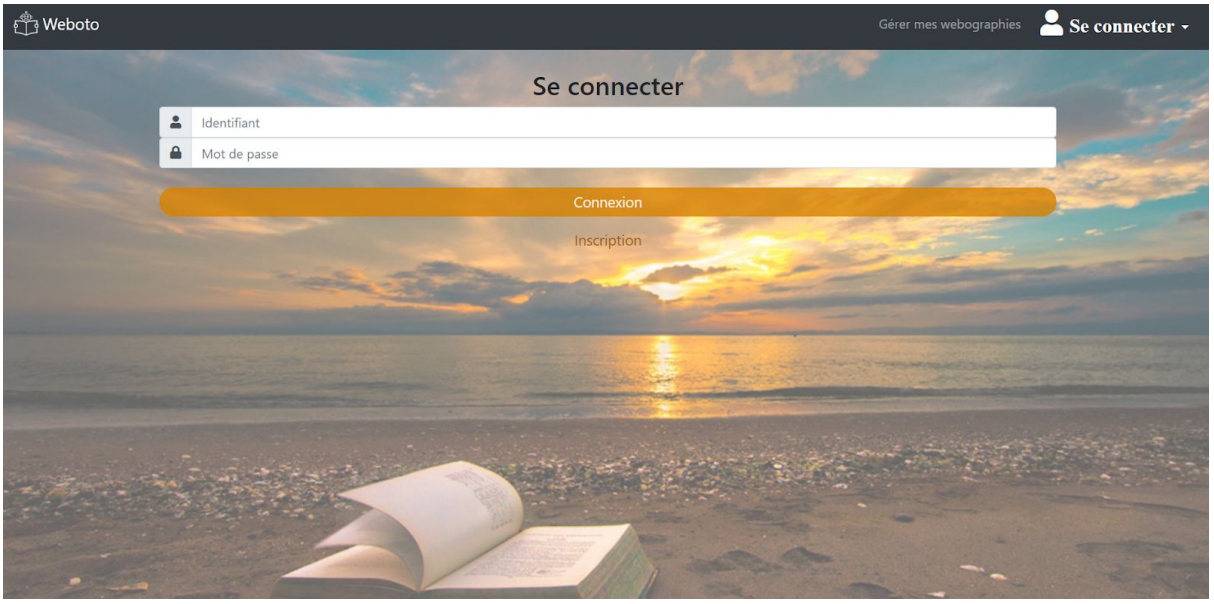


Page d'accueil

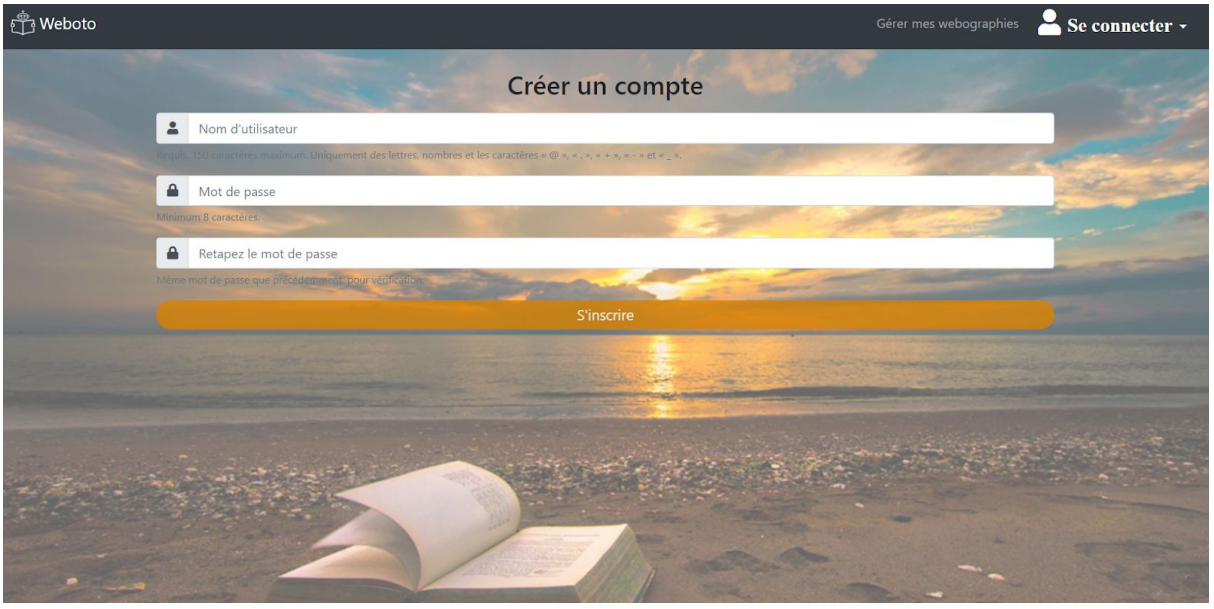
Après entrée d'une liste de liens et clic sur le bouton "Générer"



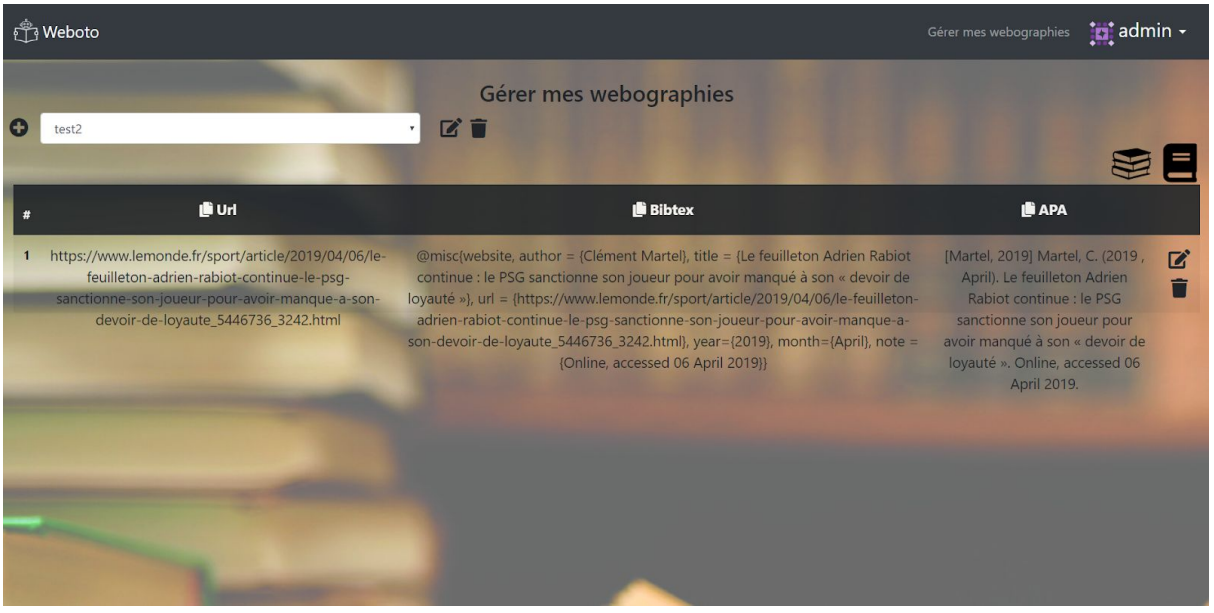
Connexion facile via la barre de navigation



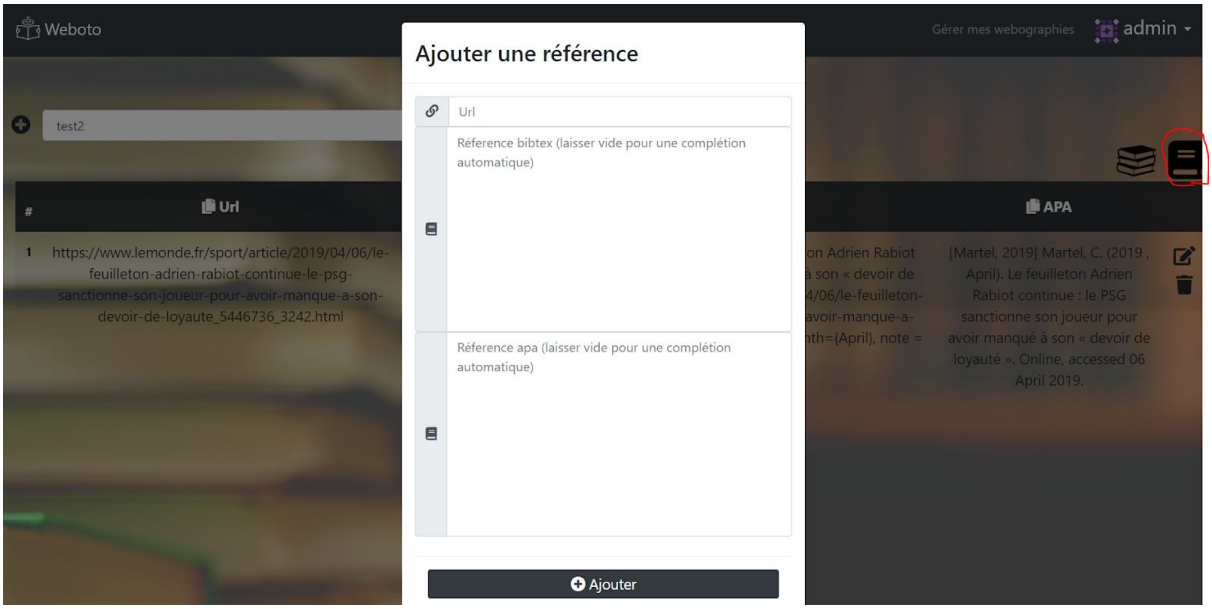
Connexion via page dédiée



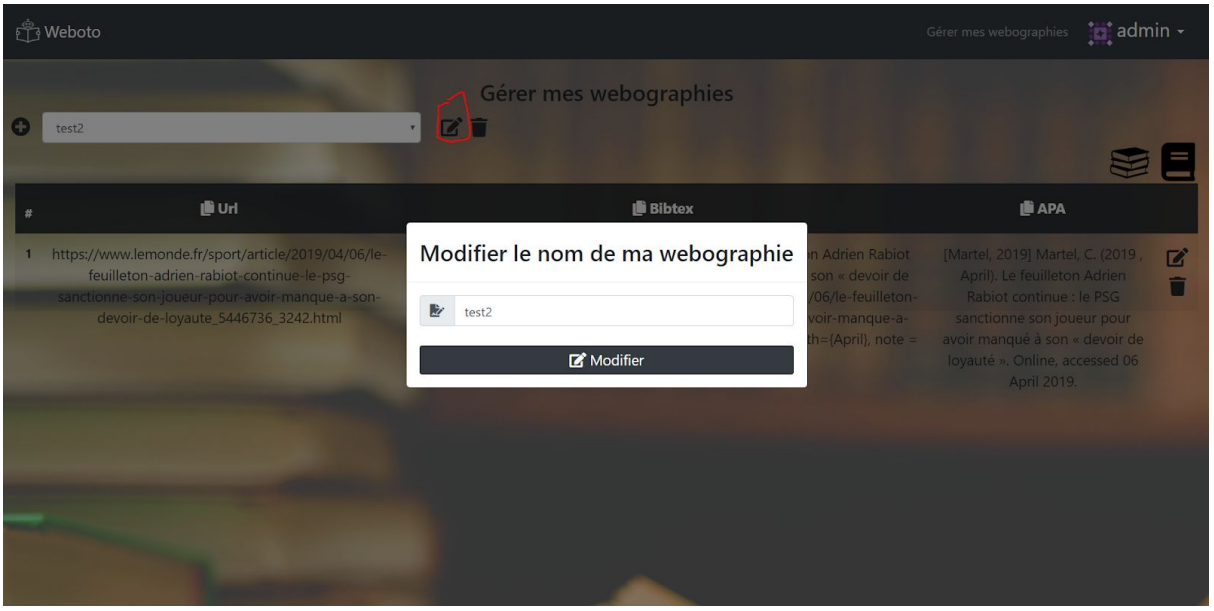
Page de création de compte



Page de gestion des webographies (tous les éléments sont cliquables pour les copier dans le presse papier)



Ajout d'une référence (via bouton en rouge)



Édition d'une webographie (via bouton en rouge)

VI. Bilan et perspectives

1. Résumé du travail fourni

1.	Apprentissage de Django
2.	Documentation sur les différentes librairies utiles au projet
3.	Familiarisation au traitement du langage naturel et aux expressions régulières
4.	Réalisation d'un site web complet sous Django, avec gestion de comptes utilisateurs (connexion/inscription) et gestion des références (utilisation du système de bases de données de Django). Tous les composants de base de Django ont été utilisés (système de base de données, formulaires, gestion des utilisateurs, page administrateur, langage de template, ….)
5.	Design et UX du site (Bootstrap + javascript)
6.	Récupération des données d'un article web en utilisant BeautifulSoup, puis conversion de ces données en une source au format Bibtex.
7.	Création d'une librairie permettant de requêter le site SemanticScholar
8.	Apport d'un correctif à la librairie pdftitle
9.	Réalisation de tests
10.	Rédaction d'un cahier des charges, de 3 état d'avancement, d'un rapport final et d'une présentation orale de 10min.

2. Bilan

Ce projet s'est révélé être intéressant pour moi et m'a permis de découvrir le framework Django, que j'ai trouvé plaisant à utiliser. Je réutiliserais volontier Django si je dois développer un site web dans le futur. En effet, la plupart des fonctionnalités de base d'un site web sont pré-codées et facilement utilisables dans Django (système de gestion des utilisateurs, page administrateur, ORM, …) et la documentation est très claire et exhaustive. De plus, il est aisé de lancer un serveur de test. Ces points permettent un gain de temps conséquent par rapport à un site web développé en php.

J'ai également pu également utiliser des techniques d'exploration de fichier html et de web-scrapping, dans le but d'automatiser la recherche d'informations présentes dans des fichiers. Ces compétences acquises pourront se révéler être un plus en entreprise, en vue d'automatiser des processus métier (par exemple technologie RPA, pour *Robotic Automation Process*).

Enfin, j'ai essayé autant que possible d'intégrer des techniques de traitement du langage naturel à mon projet, afin d'avoir une expérience dans le domaine en vue des stages. Le projet ne s'y prêtait finalement pas : mieux vaut privilégier les résultats à l'utilisation d'une technologie particulière.

Points positifs	Points négatifs
Apprentissage du framework Django	Peu d'utilisation de technologies de traitement du langage naturel (excepté les expressions régulières)
Réutilisation des compétences acquises en développement web et en Python	Délai un peu long pour l'obtention d'une référence pdf (nécessité de télécharger le fichier, d'obtenir son titre puis d'effectuer une requête sur Semantic Scholar)
Découverte et utilisation de nombreuses librairies Python	Parfois un manque d'informations pour les références web (pour certains sites dont le fichier html n'est pas très bien organisé)
Apport d'une aide à un autre développeur (librairie pdftitle)	Sur-anticipation (même si non pénalisante, voir IV.4 Une organisation efficace ?)
Résultat final satisfaisant. L'objectif final est rempli : la génération d'une webographie à partir d'une liste d'url fonctionne globalement correctement. Le site permet en plus de gérer ses webographies.	
Planning cohérent et organisation globalement efficace	
Réutilisation de concepts vus en cours de Génie Logiciel (UML, tests unitaires, patrons de conception, modèle MVC)	

Tableau de synthèse sur les points positifs et négatifs de ce projet informatique individuel

3. Évolutions possibles

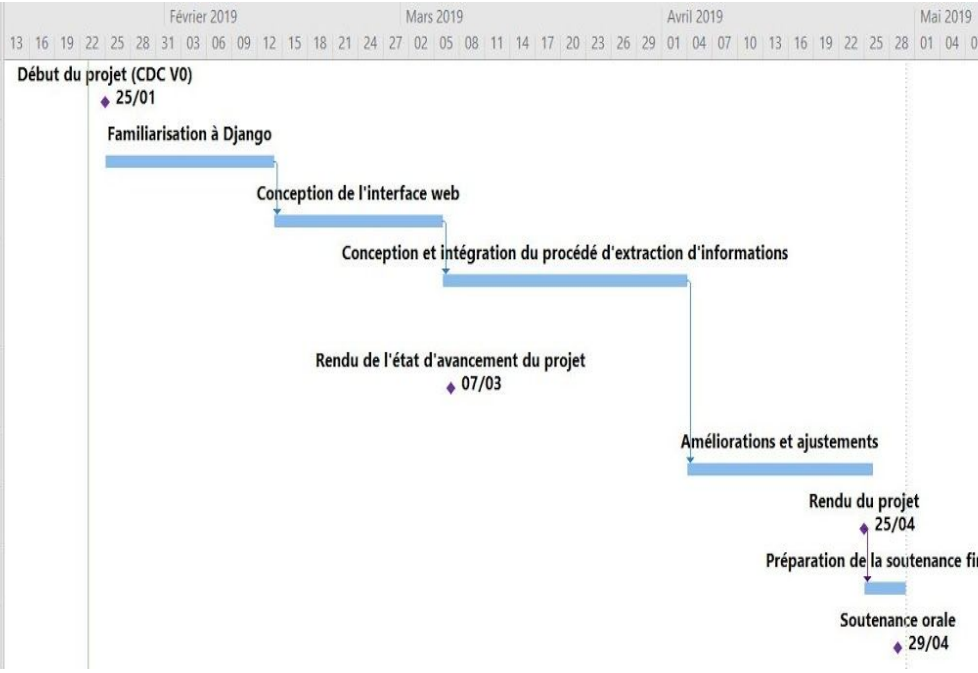
Même si le résultat final est satisfaisant, il est toujours possible d'améliorer le projet. Voici une liste d'idées pouvant être mises en place en vue de cet objectif :

- Webographies collaboratives entre plusieurs utilisateurs
- Plus de formats pour les références
- Recommandations de références pour une webographie donnée
- Solution pour alerter l'utilisateur en cas d'information incomplète
- …

VII. Annexes : plannings

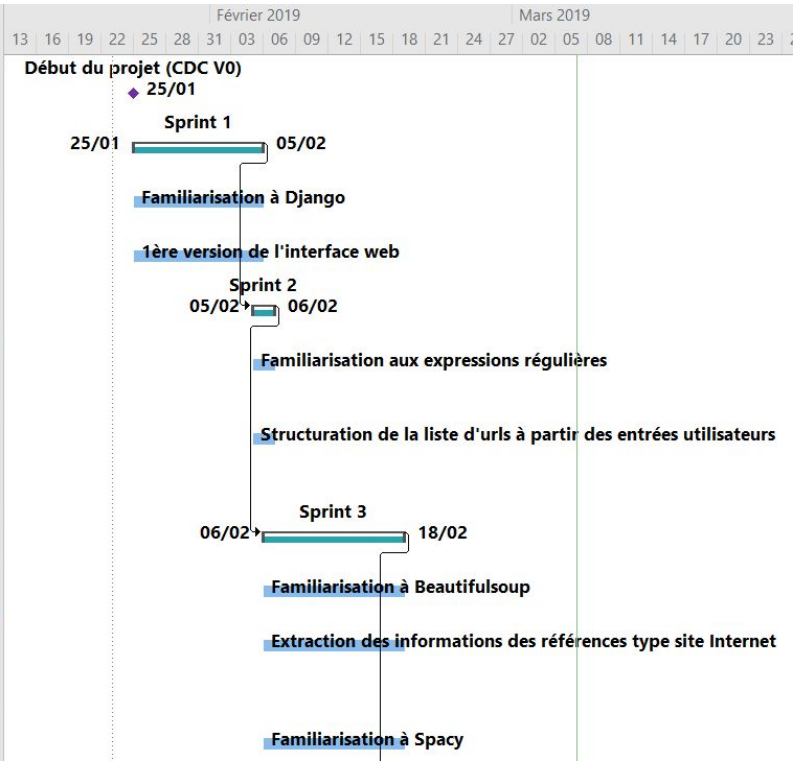
Planning prévisionnel général

	Nom de la tâche	Durée	Début	Fin	Prédécesseurs
1	Début du projet (CDC V0)	0 jr	Ven 25/01/19	Ven 25/01/19	
5	Familiarisation à Django	14 jrs	Ven 25/01/19	Mer 13/02/19	
6	Conception de l'interface web	14 jrs	Jeu 14/02/19	Mar 05/03/19	5
7	Conception et intégration du procédé d'extraction d'informations	21 jrs	Mer 06/03/19	Mer 03/04/19	6
2	Rendu de l'état d'avancement du projet	0 jr	Jeu 07/03/19	Jeu 07/03/19	
8	Améliorations et ajustements	16 jrs	Jeu 04/04/19	Jeu 25/04/19	7
3	Rendu du projet	0 jr	Jeu 25/04/19	Jeu 25/04/19	
9	Préparation de la soutenance finale	3 jrs	Jeu 25/04/19	Lun 29/04/19	3
4	Soutenance orale	0 jr	Lun 29/04/19	Lun 29/04/19	

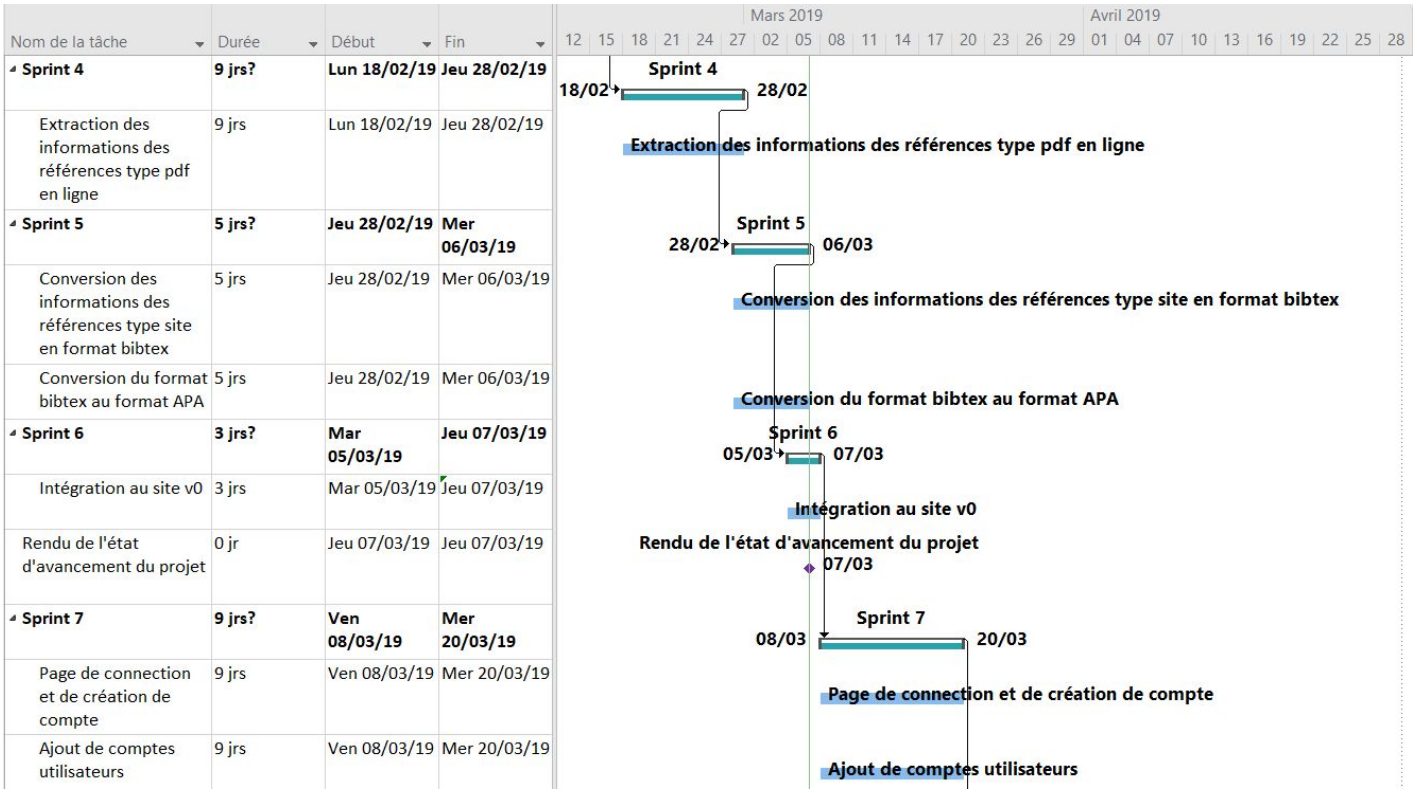


Planning prévisionnel détaillé

Nom de la tâche	Durée	Début	Fin
Début du projet (CDC V0)	0 jr	Ven 25/01/19	Ven 25/01/19
Sprint 1	**8 jrs?**	**Ven 25/01/19**	**Mar 05/02/19**
Familiarisation à Django	8 jrs	Ven 25/01/19	Mar 05/02/19
1ère version de l'interface web	8 jrs	Ven 25/01/19	Mar 05/02/19
Sprint 2	**2 jrs?**	**Mar 05/02/19**	**Mer 06/02/19**
Familiarisation aux expressions régulières	2 jrs	Mar 05/02/19	Mer 06/02/19
Structuration de la liste d'urls à partir des entrées utilisateurs	2 jrs	Mar 05/02/19	Mer 06/02/19
Sprint 3	**9 jrs?**	**Mer 06/02/19**	**Lun 18/02/19**
Familiarisation à Beautifulsoup	9 jrs	Mer 06/02/19	Lun 18/02/19
Extraction des informations des références type site Internet	9 jrs	Mer 06/02/19	Lun 18/02/19
Familiarisation à Spacy	9 jrs	Mer 06/02/19	Lun 18/02/19

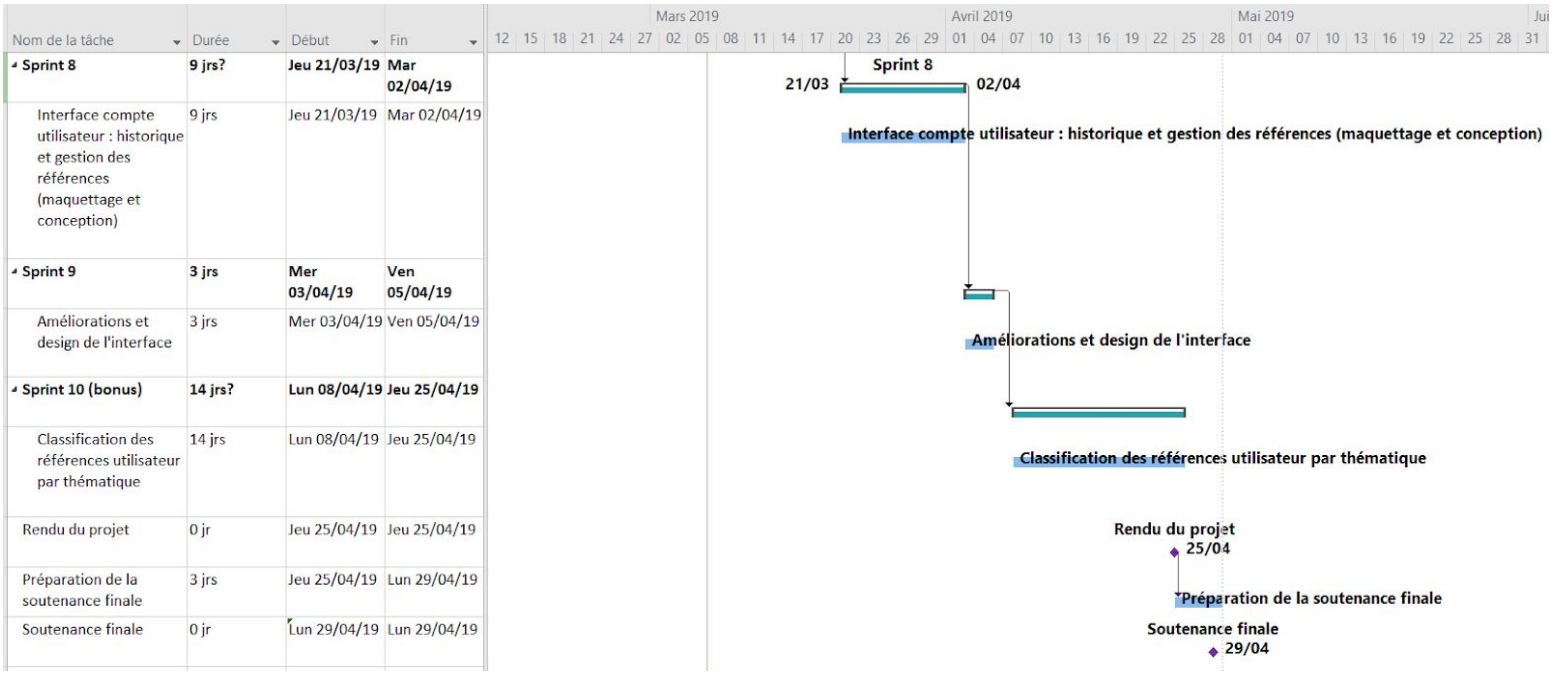


Planning prévisionnel détaillé 1/3



Nom de la tâche	Durée	Début	Fin
Sprint 4	**9 jrs?**	**Lun 18/02/19**	**Jeu 28/02/19**
Extraction des informations des références type pdf en ligne	9 jrs	Lun 18/02/19	Jeu 28/02/19
Sprint 5	**5 jrs?**	**Jeu 28/02/19**	**Mer 06/03/19**
Conversion des informations des références type site en format bibtex	5 jrs	Jeu 28/02/19	Mer 06/03/19
Conversion du format bibtex au format APA	5 jrs	Jeu 28/02/19	Mer 06/03/19
Sprint 6	**3 jrs?**	**Mar 05/03/19**	**Jeu 07/03/19**
Intégration au site v0	3 jrs	Mar 05/03/19	Jeu 07/03/19
Rendu de l'état d'avancement du projet	0 jr	Jeu 07/03/19	Jeu 07/03/19
Sprint 7	**9 jrs?**	**Ven 08/03/19**	**Mer 20/03/19**
Page de connection et de création de compte	9 jrs	Ven 08/03/19	Mer 20/03/19
Ajout de comptes utilisateurs	9 jrs	Ven 08/03/19	Mer 20/03/19

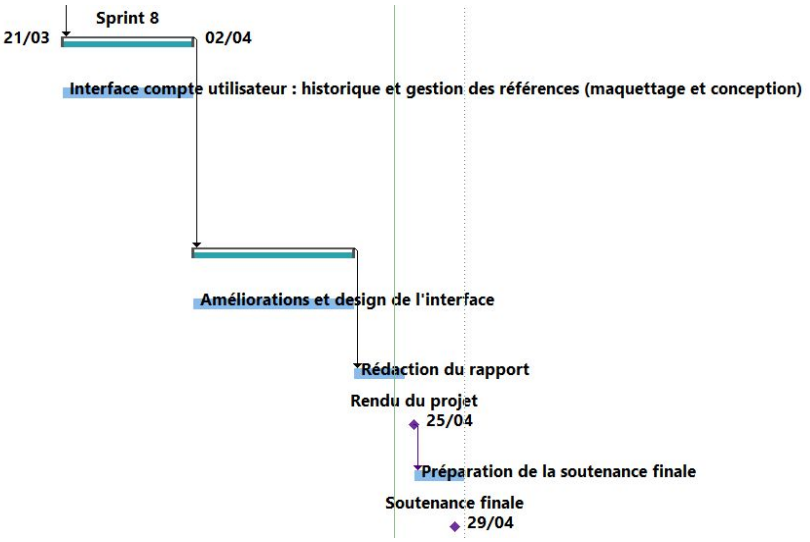
Planning prévisionnel détaillé 2/3



Nom de la tâche	Durée	Début	Fin
Sprint 8	**9 jrs?**	**Jeu 21/03/19**	**Mar 02/04/19**
Interface compte utilisateur : historique et gestion des références (maquettage et conception)	9 jrs	Jeu 21/03/19	Mar 02/04/19
Sprint 9	**3 jrs**	**Mer 03/04/19**	**Ven 05/04/19**
Améliorations et design de l'interface	3 jrs	Mer 03/04/19	Ven 05/04/19
Sprint 10 (bonus)	**14 jrs?**	**Lun 08/04/19**	**Jeu 25/04/19**
Classification des références utilisateur par thématique	14 jrs	Lun 08/04/19	Jeu 25/04/19
Rendu du projet	0 jr	Jeu 25/04/19	Jeu 25/04/19
Préparation de la soutenance finale	3 jrs	Jeu 25/04/19	Lun 29/04/19
Soutenance finale	0 jr	Lun 29/04/19	Lun 29/04/19

Planning prévisionnel détaillé 3/3

Sprint 8	**9 jrs?**	**Jeu 21/03/**
Interface compte utilisateur : historique et gestion des références (maquettage et conception)	9 jrs	Jeu 21/03/1
Sprint 9	**12 jrs**	**Mer 03/04/19**
Améliorations et design de l'interface	12 jrs	Mer 03/04/
Rédaction du rapport	3 jrs	Ven 19/04/
Rendu du projet	0 jr	Jeu 25/04/1
Préparation de la soutenance finale	3 jrs	Jeu 25/04/1
Soutenance finale	0 jr	Lun 29/04/1



Changements apportés au planning prévisionnel détaillé 3/3 (en fin de projet)