

IUT Nancy Charlemagne
Université de Lorraine
2 ter boulevard Charlemagne
BP 55227
54052 Nancy Cedex

Département Informatique

Étude d'une solution d'indexation des messages échangés sur les listes de diffusion de l'IVOA



Observatoire astronomique
de Strasbourg



CENTRE DE DONNÉES
ASTRONOMIQUES DE STRASBOURG



Rapport de stage DUT Informatique
Entreprise : CNRS-INSU Observatoire Astronomique de Strasbourg

Enzo Cisternino
Tuteur : Laurent Michel
Année universitaire 2019-2020



IUT Nancy Charlemagne
Université de Lorraine
2 ter boulevard Charlemagne
BP 55227
54052 Nancy Cedex

Département Informatique

Étude d'une solution d'indexation des messages échangés sur les listes de diffusion de l'IVOA



Rapport de stage DUT Informatique
Entreprise : CNRS-INSU Observatoire Astronomique de Strasbourg

Enzo Cisternino
Tuteur : Laurent Michel
Année universitaire 2019-2020

Remerciements

Tout d'abord, je souhaite remercier l'équipe GALHECOS de m'avoir accueilli pendant ces 10 semaines de stage ainsi que le Centre de Données Astronomiques de Strasbourg et Mark Allen (Directeur du CDS) qui ont permis le financement de mon stage.

Je remercie M. Michel d'avoir dirigé mes travaux tout au long de ces semaines, d'avoir pris le temps chaque jour de me recevoir en vidéoconférence, d'avoir répondu à tous les messages tout au long de mon stage, d'avoir pris le temps de m'aider à rédiger mon rapport et à construire ma présentation.

Je remercie également Andre Schaaff d'avoir organisé la présentation des différents métiers au sein de l'observatoire, ainsi que la présentation par les stagiaires de leurs travaux.

Je remercie Christophe Saillard et Thomas Keller du service informatique de l'observatoire pour avoir mis en place Big Blue Button (site de vidéoconférences) et Rocket.Chat (chat instantané) et avoir assuré la maintenance de ces services.

Je remercie tous les membres du VO, et plus spécialement tous ceux ayant fait des retours sur l'outil. Voir son travail reconnu ou critiqué par d'autres personnes passionnées d'informatique est un honneur.

Je remercie M. Roegel d'avoir suivi tout au long de ces 10 semaines le suivi de mon stage, et d'avoir organisé la soutenance de celui-ci.

Enfin, je remercie mes parents et ma famille de m'avoir soutenu et de s'être intéressé à mon travail pendant ces semaines.

Table des matières

I.	Introduction	7
II.	Présentation de l'entreprise.....	8
A.	Histoire de l'observatoire.....	8
B.	Évolution scientifique	8
C.	Activités	8
D.	Recherche et équipe	9
E.	Présentation de l'OV (Observatoire Virtuel)	9
III.	Conditions de travail	11
A.	Environnement et organisation.....	11
B.	Méthode de travail.....	11
IV.	Présentation du stage.....	12
A.	Objectifs globaux	12
B.	L'existant.....	12
C.	Les exigences de haut niveau	13
D.	Environnement et logiciels.....	14
E.	Plan du stage (Gantt).....	15
V.	Récupération des archives.....	16
A.	Archivage par Pipermail.....	16
B.	Téléchargement des archives	17
C.	Présentation du format Mbox	17
D.	Parcours des archives	18
E.	Récupération des messages d'une archive	18
VI.	Mise en forme des messages.....	18
A.	Objectifs de la mise en forme.....	18
B.	Présentation Elastic Search	19
C.	Présentation du document pour ES	19
D.	Les pièces jointes	21
E.	Les problèmes d'encodage	22
F.	La reconstitution des dates	23
VII.	Indexation des messages	23
A.	Objectifs de l'indexation	23
B.	Choix du nom de l'index	24

C.	Le paramètre pour les index	24
D.	Indexation d'un message	25
E.	Reconstitution des fils de discussion.....	26
F.	Disponibilité des index sous maintenance	27
VIII.	Création de la solution web.....	27
A.	Objectifs de la solution web	27
B.	Création de l'architecture HTML.....	28
C.	Choix graphiques.....	31
D.	Implémentation du formulaire.....	31
E.	Création d'une requête	32
F.	Traitemet et exécution d'une requête.....	34
G.	Création des threads	35
H.	Coloration Syntaxique.....	36
I.	Téléchargement et affichage des pièces-jointes.....	37
J.	Lien par mail et requête via URL	38
IX.	Déploiement.....	38
A.	Intérêt	38
B.	Config.json	39
C.	Mise en ligne du site web.....	40
D.	Disponibilité de l'outil sous mise à jour.....	40
E.	Configuration Serveur	40
X.	Validation par l'observatoire virtuel.....	41
A.	Mise en ligne et intérêt	41
B.	Retours et Activité.....	41
C.	Prise en compte des retours.....	43
XI.	Conclusion	43
XII.	Annexes.....	44
XIII.	Références et Bibliographie	45

I. Introduction

Dans le cadre de ma deuxième et dernière année en DUT Informatique à l'IUT (Institut Universitaire de Technologie) Charlemagne à Nancy, j'ai effectué un stage de 10 semaines en tant que chargé d'études au sein de l'observatoire de Strasbourg du 14 avril 2020 au 23 juin 2020. Ce stage est ma première expérience professionnelle dans le monde de l'informatique.

Un OV (Observatoire Virtuel) est la vision selon laquelle les ensembles de données astronomiques et d'autres ressources devraient fonctionner comme un tout homogène.

L'IVOA (International Virtual Observatory Alliance) [\[IVOA,2020\]](#) est une organisation qui débat et accepte les normes techniques qui doivent être mises en place pour rendre l'Observatoire Virtuel possible. Elle sert aussi de point de convergence des nouvelles idées pour le VOL, de cadre pour discuter et partager des idées et des technologies du VO, et d'organe pour promouvoir et faire connaître le VO dans le monde.

Au sein de l'IVOA, les discussions menant à l'élaboration de standards et de protocoles se font sur des listes de diffusion organisées par groupes de travail (working group) ou par centres d'intérêts (interest group). L'archivage de ces mails est géré par Pipermail, un outil permettant de stocker ces mails dans le temps. Cet archivage datant de plus de 17 ans, y retrouver un mail s'y avère compliqué et fastidieux, c'est ainsi que le sujet de mon stage a vu le jour : indexer les messages de toutes les listes de diffusion pour permettre la recherche de mails ainsi que la construction d'un outil d'affichage avancé pour reconstruire les fils de discussion ainsi que mettre à disposition les pièces jointes de chaque courriel.

Ce rapport sera construit suivant plusieurs axes. En premier lieu, je présenterai l'observatoire de Strasbourg et l'IVOA, puis je développerai les principaux axes de mon stage, et terminerai par une conclusion sur mon expérience tout au long du stage.

II. Présentation de l'entreprise

A. Histoire de l'observatoire

L'Observatoire astronomique de Strasbourg est situé sur le campus historique de l'Université de Strasbourg. Il a été fondé en 1881. C'est un établissement de recherche et d'enseignement, qui contient également le Centre de données astronomiques de Strasbourg.

Il est en réalité le troisième observatoire de Strasbourg, le premier ayant été construit en 1673 sur une des tours d'enceinte de la ville, et le second en 1828 sur le toit des bâtiments de l'Académie. L'existence de l'Observatoire n'est dû qu'à une forte décision politique : lorsque l'Alsace-Moselle est cédée à l'Allemagne après la guerre franco-prussienne de 1870, l'empereur Guillaume Ier d'Allemagne décide de faire de Strasbourg une vitrine : triplant la superficie de la ville, il y fait construire une université comprenant un jardin botanique ainsi que l'Observatoire astronomique. [\[Wikipédia, 2020\]](#)

B. Évolution scientifique

La vocation initiale concerne pour partie l'astronomie de position, et l'observation de comètes, de météorites, d'étoiles variables. Vint ensuite la photométrie de nébuleuses, l'observation d'étoiles doubles. Lors du retour de Strasbourg à la France, l'observatoire est modernisé. Il y est installé l'électricité, téléphone, machines-outils. Par la suite, l'observation au sol étant jugée comme "ayant atteint ses limites instrumentales", l'observation satellitaire est privilégiée et l'archivage informatique est développé, ce qui donnera naissance au centre de données astronomiques de Strasbourg

C. Activités

L'établissement est un observatoire des sciences de l'univers de l'Institut national des sciences de l'univers (INSU), et une unité mixte de recherche du Centre national de la recherche scientifique (CNRS) et de l'Université de Strasbourg. Comme pour tout observatoire des sciences de l'univers, plusieurs missions doivent être remplies: recherche, enseignement, services d'observation et diffusion des connaissances.

D. Recherche et équipe

L'unité de recherche est divisée en deux équipes : [\[Astro Unistra,2020\]](#)

- L'équipe GALHECOS (Galaxies High Energy Cosmology Compact Objects & Stars) étudie la formation et l'évolution des galaxies dans un contexte cosmologique, au travers de leurs populations stellaires, de la dynamique des étoiles et de la matière noire, et des effets de rétroaction liés notamment à l'activité de leur trou noir central. Elle s'intéresse aux sources galactiques et extragalactiques émettrices en rayons X, objets compacts (étoiles à neutrons, naines blanches, etc...) et noyaux actifs de la galaxie. Elle héberge le SSC-XMM, un consortium international de laboratoires sélectionnés par l'ESA et labellisés par l'INSU comme Service d'Observation, qui contribue au traitement continu des données du satellite XMM-Newton et qui a la charge de fournir des catalogues complets de sources X observées à la communauté internationale.
- Le CDS (Centre de données astronomiques de Strasbourg), qui est à la fois une équipe de recherche et un service d'observation. Les services de bases de données (Simbad, Vizier) et de visualisation (Aladin Sky Atlas) développés par le CDS sont utilisés par l'ensemble de la communauté astronomique mondiale. Celui-ci est l'un des acteurs majeurs du développement de l'Observatoire Virtuel International en Astronomie. Fin 2008, le CDS a été labellisé TGIR (Très Grande Infrastructure de Recherche) par le Ministère de l'Enseignement Supérieur et de la Recherche, confirmé comme Infrastructure de Recherche en 2012, ce qui le range au même niveau que des infrastructures internationales comme l'European Southern Observatory ou RENATER à l'échelon national.

Vous pouvez retrouver en annexe l'organigramme du CDS.

E. Présentation de l'OV (Observatoire Virtuel)

L'objectif de l'Observatoire Virtuel est plus facilement décrit par analogie avec Internet. Lors de la navigation d'un lien à un autre, un utilisateur peut accéder à une vaste gamme de documents. Ceux-ci sont construits par de nombreuses personnes et organisations différentes et se propagent dans le monde entier. Internet est transparent. Le but de l'OV est d'obtenir le même sentiment pour les données astronomiques, qui est d'assurer l'interopérabilité entre les outils et la base de données. Comme pour Internet, l'OV n'est pas un système fixe, mais plutôt une vision, façon de faire les choses.

L'OV permet aux astronomes d'interroger plusieurs centres de données de manière transparente, fournit de nouveaux outils d'analyse et de visualisation puissants au sein de ce système et offre aux centres de données un cadre standard pour publier et fournir des services. Ceci est rendu possible par l'élaboration de standard de données et métadonnées, par la standardisation des méthodes d'échanges et par l'utilisation d'un registre, qui répertorie les services disponibles et ce qui peut être fait avec eux.

L'Alliance Internationale des observatoires virtuels (IVOA) a été créée en juin 2002 avec pour mission de "faciliter la coordination et la collaboration internationale nécessaire au développement et au déploiement des outils, des systèmes et des structures organisationnelles nécessaires pour permettre l'utilisation internationale des archives astronomiques en tant qu'Observatoire virtuel interopérant. L'IVOA comprend désormais 20 membres répartis sur tout le globe.



Le travail de l'IVOA se concentre sur l'élaboration de normes. Les groupes de travail sont constitués de membres inter-projets dans les domaines où les normes et technologies d'interopérabilité doivent être définies et approuvées. Les groupes de travail élaborent des normes en utilisant un processus calqué sur le World Wide Web Consortium, dans lequel les projets de travaux progressent vers les recommandations proposées. Les recommandations sont finalement approuvées par le Groupe de travail de l'Observatoire Virtuel de la commission 5 (Astronomical Data) de l'Union Astronomique Internationale. L'IVOA a également des groupes d'intérêt qui discutent des expériences d'utilisation des technologies OV et fournissent des commentaires aux groupes de travail.

Ces communications se font par le biais de services indépendants ou bien par mails, grâce à des listes de diffusions, ces listes de diffusions sont pour l'instant au nombre de 21, parmi toutes ces listes de diffusions, certaines ont plus de 15 ans (2003). [\[IVOA,2020\]](#)

III. Conditions de travail

A. Environnement et organisation

Mon stage s'est déroulé du 12 avril au 23 juin 2020, période pendant laquelle la France était en crise sanitaire dû à la pandémie mondiale du coronavirus.

A cause de cette situation, l'ensemble de mon stage s'est déroulé en télétravail à mon domicile. J'avais à disposition mon ordinateur personnel et trois écrans, me permettant d'avoir un environnement viable pour un stage.

La communication s'en est trouvée réduite, néanmoins, je pouvais communiquer avec L. Michel via Rocket Chat, qui, à la manière de Slack, me permettait de lui poser des questions et d'avoir des réponses et inversement.

Je travaillais 7H par journée, parfois un peu plus, afin de garder le même rythme de travail qu'en entreprise s'il n'y avait pas eu de crise sanitaire.

B. Méthode de travail

J'ai travaillé selon une méthode qui m'a été proposée (et non imposée) par M.Michel, j'ai décidé de l'adopter car elle me semblait viable et très intéressante.

Elle se composait de la manière suivante, chaque matin, nous nous donnions rendez-vous pour une visio-conférence où je présentais tout le travail que j'avais fait la veille.

C'était l'occasion pour moi d'avoir des retours de mon tuteur, ainsi que des nouvelles tâches à faire pour la journée qui commençait.

Chaque fin de journée, il m'était demandé d'écrire un rapport journalier sur wiki du Github [[GitHub,2020](#)] du projet. J'y résumais les tâches que j'avais effectuées, des commentaires personnels, ainsi qu'une idée personnelle des tâches que je pourrais avoir à faire le lendemain. Cette organisation se rapproche d'une méthode AGILE avec des sprints quotidiens. A niveau plus global, j'alternais les développements entre la partie serveur et la partie clients afin de pouvoir faire des validations croisées. Je pouvais ainsi voir l'application se construire pas à pas comme un ensemble cohérent.

Cette méthode a porté ses fruits car elle m'a permis d'avancer au jour le jour, et de ne pas me retrouver bloqué une semaine sur un problème spécifique. Sur la fin du projet, mon tuteur m'imposait plus de "deadline" car il voulait que telle ou telle partie soit terminée avant la mise en ligne du projet.

IV. Présentation du stage

A. Objectifs globaux

Le but de ce stage est la création d'un outil de recherche se rapprochant de celui de Google en termes de simplicité, mais proposant des fonctions plus avancées selon les besoins. Il permettra de faire des recherches de courriels dans les listes de diffusion de l'IVOA : requête par expéditeur, receveur, date d'envoi, sujet, contenu, type de pièces jointes, etc... , tout en ayant un aspect jeune et raffiné permettant d'avoir une interface conviviale pour chaque utilisateur.

L'objectif était de créer un service pour l'indexation de ces messages, leur mise en forme, et de la création de la solution web de cet outil.

B. L'existant

Les personnes travaillant au sein de l'IVOA pour l'OV communiquent par mails sur des listes de diffusions, ces listes sont nombreuses et anciennes et contiennent des milliers de mails.

Tous ces mails sont stockés par l'IVOA [\[IVOA, 2020\]](#) et on peut y accéder par mailing list, par mois, par fils de discussions, etc...

Malheureusement, cette interface est très vétuste et n'est ni effective ni ergonomique pour retrouver un mail précis dans toutes ces listes.

En effet, il n'y a aucun champ de recherche. Le seul moyen de retrouver un courriel voulu et de savoir le mois approximatif de l'envoi du mail, de se rendre sur la page du mois en question, et d'essayer de trouver parmi tous les courriels existants, lequel vous intéresse.

June 2020 Archives by thread

- [Messages sorted by: \[subject\] \[author\] \[date\]](#)
- [More info on this list...](#)

Starting: *Mon Jun 1 16:33:55 CEST 2020*

Ending: *Fri Jun 5 12:22:19 CEST 2020*

Messages: 5

- [Primitive types in the ivoa package](#) *CresitelloDittmar, Mark*
 - [Primitive types in the ivoa package](#) *Laurent MICHEL*
- [SpecDM/SEDDM status](#) *Igor Chilingarian*
 - [SpecDM/SEDDM status](#) *Markus Demleitner*
 - [SpecDM/SEDDM status](#) *Igor Chilingarian*

Last message date: *Fri Jun 5 12:22:19 CEST 2020*

Archived on: *Fri Jun 5 12:22:34 CEST 2020*

- [Messages sorted by: \[subject\] \[author\] \[date\]](#)
- [More info on this list...](#)

This archive was generated by Pipermail 0.09 (Mailman edition).

Sur cette capture d'écran, on peut voir l'ensemble des mails envoyés durant le mois de Juin 2020 sur la liste de diffusion DM (Data Model). Pour l'instant, cette page reste lisible et on peut retrouver un mail, mais pour certaines pages, on peut y retrouver plusieurs centaines de mails avec cette même interface, et toujours, sans aucun filtre de recherche.

Heureusement, Pipermail permet de télécharger une archive contenant pour chaque liste de diffusion, l'ensemble des mails contenus dans celle-ci. Cette archive est au format Mbox⁽¹⁾, qui est un format de stockage de mails, et qui associe à chaque courriel un ensemble d'attributs le décrivant.

Ces attributs sont nombreux et décrivent toutes les parties d'un courriel, que ce soit le destinataire ou l'expéditeur, jusqu'aux pièces jointes, à la date d'envoi, l'encodage, etc...

C. Les exigences de haut niveau

M. Michel a bien spécifié les exigences du stage, et quels étaient mes tâches et objectifs à réaliser/atteindre.

Les trois objectifs globaux sont donc les suivants :

- Création d'un programme d'indexation des courriels

- Création d'une solution web de recherche avancée de courriels
- Livraison d'un outil fonctionnel tant côté client que serveur

Ces trois objectifs se sont donc divisés en plusieurs sous-tâches qui ont dirigé mon travail :

- La récupération des archives : il s'agit ici de télécharger les archives Mbox et de pouvoir les parcourir librement afin d'extraire les mails qu'elles contiennent.
- La mise en forme des messages : c'est la partie nécessaire pour construire des objets compatibles avec la base de données qui stockera les courriels
- L'indexation des messages : tout simplement ajouter tous les mails des archives dans une base de données pour avoir un accès par requêtes
- Création de la solution web : la partie la plus importante, la création de l'outil de recherche qui permettra de faire le lien avec les besoins de l'utilisateur et les mails contenus dans la base de données
- Déploiement des services : il était obligatoire que je mette en ligne la solution web avant la fin du stage afin d'avoir des retours, il a donc fallu héberger Elastic Search sur un serveur public créer des configurations par défaut, etc...

D. Environnement et logiciels

Dans cette partie, je présenterai brièvement les langages, les environnements de développement et les logiciels que j'ai utilisés pendant mon stage.

HTML (Hypertext Markup Language) : C'est le langage de balisage le plus connu conçu pour représenter des pages Web.

CSS (Cascading Style Sheets) : Ce sont des feuilles de style en cascade, qui forment un langage informatique qui décrit la présentation des documents HTML, c'est ce qui permet de styliser une page Web.

Javascript : Javascript est un langage de programmation de scripts employées dans les pages Web pour manipuler l'arbre DOM du navigateur.

JQuery : JQuery est une bibliothèque Javascript libre et multiplateforme créée pour faciliter l'écriture de scripts côté client des pages Web.

JSON (JavaScript Object Notation) : JSON est un 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 par exemple [\[Wikipédia, 2020\]](#).

PyCharm : PyCharm est un environnement de développement intégré utilisé pour programmer en Python, il est développé par l'entreprise JetBrains.

WebStorm : WebStorm est un environnement de développement intégré pour les langages Web (HTML, CSS et JavaScript), développé par l'entreprise JetBrains.

Oracle VM VirtualBox : Oracle VM est un logiciel libre de virtualisation publié par Oracle. Je l'ai utilisé pour simuler un OS Linux pour coder, ne pouvant pas faire de dualboot sur mon ordinateur personnel.

GitHub : GitHub est un outil en ligne qui permet d'héberger ses « repositories » de code. GitHub est un outil gratuit pour héberger du code open source, et proposer également des plans payants pour les projets de code privé.

Elastic Search : Elastic Search est un serveur utilisant Lucene pour l'indexation et la recherche de données, il fournit un moteur de recherche distribué et multi-entité à travers une interface REST, qui permet de construire des requêtes précises selon les besoins de chaque utilisateur.

Rocket.chat : Rocket.chat est un logiciel de communication en équipe : on peut échanger avec les autres membres sur des fils publics, ou des groupes de discussions privés.

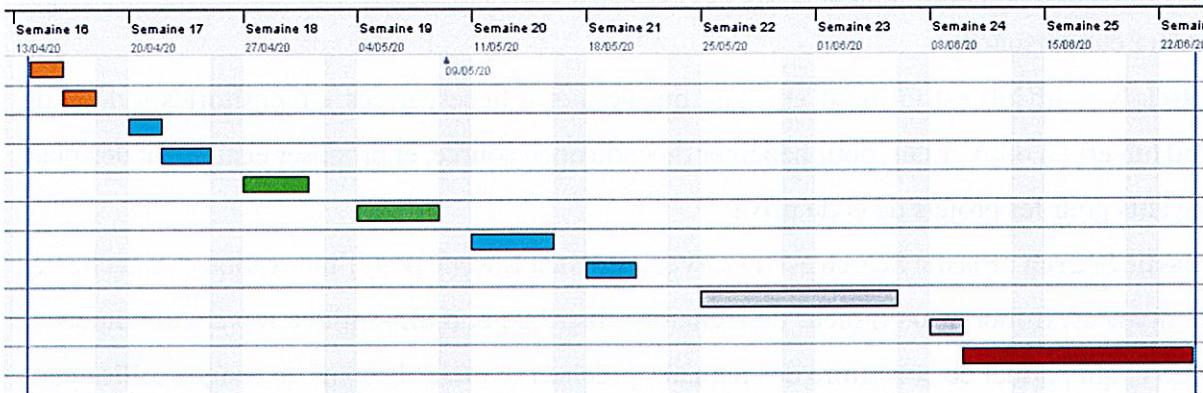
BigBlueButton : BBB est un site en ligne de visioconférence hébergé par l'observatoire de Strasbourg qui nous a permis de faire de nombreuses vidéoconférences tout au long de ce confinement et de ce télétravail.

Prism : Prism est un module Javascript permettant de faire de la coloration syntaxique, il détecte notamment les langages de programmation et colorie les mots-clés du code affiché dans la page HTML (Python, Java, XML, etc...)

E. Plan du stage (Gantt)

Concernant le plan du stage, voici un diagramme de type “GANTT” qui récapitule l'organisation générale de mon stage.

Nom	Date de début	Date de fin
• Prise en main de l'outil Perceval	14/04/20	15/04/20
• Prise en main de Elastic Search	16/04/20	17/04/20
• Indexation des mails dans Elastic Search	20/04/20	21/04/20
• Construction des Threads	22/04/20	24/04/20
• Création du prototype Web	27/04/20	30/04/20
• Création du Back-End du site Web	04/05/20	08/05/20
• Modification Création Threads et implémentation	11/05/20	15/05/20
• Création du script Python de chargement automatique	18/05/20	20/05/20
• Ajout de fonctionnalités et corrections de bugs	25/05/20	05/06/20
• Mise en ligne et corrections mineures	08/06/20	09/06/20
• Préparation Rapport et Présentation	10/06/20	23/06/20



J'ai passé environ une semaine à prendre en main les outils que j'ai utilisé pendant le stage, ainsi que créer quelques programmes de test, puis j'ai passé le reste du temps à développer les différents services demandés.

V. Récupération des archives

A. Archivage par Pipermail

Pipermail est un logiciel d'archivage de mails, qui génère une page Web pour chaque email, fil de discussion, mois, etc...

Son interface vétuste n'est pas adaptée à de la recherche, néanmoins, Pipermail permet de télécharger l'ensemble des mails envoyés à une mailing list par le téléchargement d'une archive au format Mbox.

Ces archives sont conséquentes en poids, atteignant plusieurs dizaines de méga-octets, allant jusqu'à 90 Mo pour la plus grande (Data Model).

The dm Archives

You can get [more information about this list](#) or you can [download the full raw archive](#) (77 MB).

B. Téléchargement des archives

Pour pouvoir récupérer les mails, il faut donc télécharger toutes les archives voulues, pour cela j'utilise le module url lib et plus précisément la méthode urllib.request.

Cette méthode prend en paramètre l'URL complète d'où télécharger le fichier, le chemin avec le nom du nouveau fichier à sauvegarder, et facultativement, une méthode pour suivre le téléchargement.

Chaque lien de téléchargement est construit de la manière suivante par Pipermail :

- L'URI de l'IVOA : "<http://mail.ivoa.net/pipermail>"
- La partie de l'URL pour chaque archive : "nomListe.mbox/nomListe.mbox"

Par exemple pour "Data Model": "<http://mail.ivoa.net/pipermail/dm.mbox/dm.mbox>"

Une fois la méthode exécutée, l'archive est sauvegardée à l'endroit spécifié dans les paramètres, et peut être utilisée par la suite. Au préalable, je m'assure de supprimer les archives téléchargées auparavant, afin de n'avoir que les archives les plus récentes dans nos dossiers.

Cette fonction est assurée par le module "downloader.py" dans le projet.

C. Présentation du format Mbox

Voici une liste non exhaustive des attributs que présente Mbox pour chaque mail :

To	Nom + Prénom + Email de la/les personne/s à qui a été envoyé le mail courant.
From	Nom + Prénom + Email de la personne qui a envoyé le mail.
Subject	C'est le sujet du message courant.
Message-ID	C'est un identifiant unique propre à chaque mail.
Date	C'est la date d'envoi du mail, ex : 26 May 2020 14:24:36 +0200
In-Reply-To	En cas de réponse à un mail, ce champ contient l'id du mail auquel le mail courant a répondu
References	En cas de réponse à d'autres mails, ce champ contient l'ensemble des ids des mails qui constituent le Thread (fil de discussion)

L'attribut "Content-Type" spécifie si le mail est "multipart" ou "text/plain".

S'il est "text/plain" cela veut dire qu'il est composé d'une seule partie, qui est en fait le contenu du mail, il n'y a donc pas de pièces jointes et cela est plus simple d'obtenir le contenu du message. S'il est "multipart" cela spécifie qu'il est composé de plusieurs parties, d'une partie

“text/plain” qui contient le contenu du message, ainsi que d’autres types décrivant les pièces jointes, par exemple “image/gif” est une image au format .gif.

D. Parcours des archives

Pour pouvoir utiliser le contenu d’une archive Mbox, il faut créer un objet Mbox en python via le module “mailbox” de Python. Pour cela, j’importe le module python et, pour chaque maillist, je crée un objet mailbox via le constructeur : `mailbox.mbox(path)` où path est le chemin de l’archive.

Cette méthode nous retourne donc un objet `mbox`, qui va nous permettre par la suite de parcourir tous les messages contenus dans celui-ci.

Cette fonction est implémentée par la méthode “`buildReposAndIndex`” de `percevaler.py`, entre autres elle permet, pour chaque mailing list, de créer l’objet Mbox (en mémoire) via la méthode “`createRepo`” et ainsi pouvoir l’utiliser par la suite.

E. Récupération des messages d’une archive

Pour parcourir ces archives, il suffit de faire une boucle for pour parcourir tous les messages : “`for message in mbox`”, et donc on obtient un objet message où l’on peut récupérer l’ensemble des attributs spécifiés dessus via plusieurs solutions :

- `message[“From”]` : méthode la plus simple mais qui peut générer des exceptions
- `message.get(“From”,failobj=”None”)` : méthode qui retourne dans tous les cas un résultat sans générer d’exceptions, car si l’attribut n’existe pas, on peut spécifier la valeur de retour.

Il est donc maintenant possible de parcourir l’ensemble des mails de chaque archive et ainsi commencer la mise en forme de chaque message pour l’indexation.

VI. Mise en forme des messages

A. Objectifs de la mise en forme

La mise en forme des messages est obligatoire pour la suite du processus, car elle permet de construire un objet qui pourra être indexé dans la base de données.

Il est nécessaire de récupérer les informations voulues des messages des archives, de gérer l'encodage ainsi que le format pour construire un objet propre à l'insertion, qui n'aura pas besoin d'être modifié sur la solution web.

B. Présentation Elastic Search

Elastic Search est une base de données no-SQL [[Wikipédia,2020](#)] qui permet de stocker de grosses quantités d'éléments, le principe est d'utiliser le format JSON, que ce soit pour indexer un élément ou pour effectuer une requête.

ES est hébergé et ses paramètres sont configurables. De base, il est accessible via le port 9200. On peut également avoir des informations générales sur la version d'Elastic Search en se rendant sur la page “localhost:9200”.

Programmant en python, j'ai installé le module Elastic Search, et ainsi pu créer une instance d'un objet Elastic Search, pour ajouter des index/éléments, les supprimer, les modifier, ou exécuter des requêtes.

Pour cela on déclare un objet Elastic Search dans le bloc voulu par la méthode suivante :

```
es = Elasticsearch([es_url]) où es_url est l'url de communication avec Elastic  
Search, dans notre cas : es_url = localhost:9200
```

Il faut savoir que Elastic Search peut contenir plusieurs index, ou un seul, sachant qu'un index n'a que seule limite de taille la limite de stockage de l'hébergeur, et par défaut on ne peut sélectionner que 10000 éléments à la fois.

Pour de très grandes requêtes, il sera donc nécessaire de modifier ce paramètre, ce qui ne nous intéresse pas pour le cas du site web, mais qui pourrait s'avérer utile pour faire du Big Data par exemple pour faire des analyses statistiques sur l'ensemble des mails de l'IVOA

C. Présentation du document pour ES

Comme dit ci-dessus, Elastic Search se repose sur le format JSON, il faut donc créer un fichier JSON pour ajouter un élément à un index. En python, un JSON est plus généralement représenté par un dictionnaire qui reprend exactement la même structure qu'un fichier JSON, chaque clé/champ de ce JSON sera un attribut sur lequel on pourra faire des recherches si le format le permet.

Je déclare donc un nouveau dictionnaire dans ma boucle “for”, pour chaque nouveau mail. Ce dictionnaire est appelé “item” et se compose de cette manière :

- Voici ses attributs qui sont récupérés de chaque mail avec la méthode message.get() :
 - from, to, in-reply-to, id , references, subject
- Il contient également d'autres attributs que j'ajoute et qui fournissent des informations utiles pour la solution web :
 - timestamp : représentant la conversion de la date du mail en timestamp (temps UNIX) , il permet de faire des requêtes triées par date
 - body : c'est le corps du message, le contenu que l'utilisateur a saisi
 - num : c'est un numéro unique pour chaque mail, mais plus simple que l'id, il est composé du raccourci de la liste de diffusion auquel il appartient, et d'un chiffre
 - maillist: c'est tout simplement le raccourci du nom de la liste de diffusion, il sera utilisé pour faire des requêtes sur une liste de diffusion précise
 - numThread: c'est le numéro de thread (fil de discussion) auquel appartient le mail courant
 - attachements_name : pour être bref, il contient l'ensemble des types et noms des pièces jointes s'il y en a
 - attachements : pour être bref, il contient l'ensemble des encodages des pièces jointes, et pour cela je l'ai tout d'abord fait dans un dictionnaire, et je stocke la représentation JSON de ce dictionnaire dans ce champ

entication-Results: spf=pass (sender IP is 194.254.240.32)
p.mailfrom=astro.unistra.fr; hostname.fr; dkim=none (message not signed)
d[dkim]=none;hmail.fr; dmarc=bestguesspass action=none
der.from=astro.unistra.fr;compauth=pass reason=109
lived-SPF: Pass (protection.outlook.com: domain of astro.unistra.fr
ignates 194.254.240.32 as permitted sender)
eiver=protection.outlook.com; client-ip=194.254.240.32;
o=smtpout81-ext1.partage.renater.fr;
lived: from ssmtpout81-ext1.partage.renater.fr (194.254.240.32) by
EUR06FT016.mail.protection.outlook.com (10.13.6.237) with Microsoft SMTP
ver id 15.20.3821.23 via Frontend Transport; Tue, 26 May 2020 12:24:37
00
comingTopHeaderMarker:
ginalChecksum:10ABDFC18E23FEA83E6FACB6C9C466C45594FD4AAAC4EC0D3A7F85A2582D9C64;Upp
ived: from zmtaauth01.partage.renater.fr (zmtaauth01.partage.renater.fr [194.254.2
smtpout10.partage.renater.fr (Postfix) with ESMTP id D087F661C8E
r <zenzo_cisternino@hotmail.fr>; Tue, 26 May 2020 14:24:36 +0200 (CEST)
ived: from zmtaauth01.partage.renater.fr (localhost [127.0.0.1])
zmtaauth01.partage.renater.fr (Postfix) with ESMTPS id CF29C1400B5;
e, 26 May 2020 14:24:36 +0200 (CEST)
ived: from localhost (localhost [127.0.0.1])
zmtaauth01.partage.renater.fr (Postfix) with ESMTP id BFD7A1400B9;
e, 26 May 2020 14:24:36 +0200 (CEST)
rus-Scanned: amavisd-new at zmtaauth01.partage.renater.fr
ived: from zmtaauth01.partage.renater.fr ((127.0.0.1))
localhost (zmtaauth01.partage.renater.fr [127.0.0.1]) (amavisd-new, port 10026)
t: ESMTP id XHXLIIuK0HF; Tue, 26 May 2020 14:24:36 +0200 (CEST)
ived: from 82.229.258.224 (unknown [194.254.241.251])
zmtaauth01.partage.renater.fr (Postfix) with ESMTPA id 83D761400B5;
e, 26 May 2020 14:24:36 +0200 (CEST)
cisternino enzo <zenzo_cisternino@hotmail.fr>
: Laurent MICHEL <laurient.michel@astro.unistra.fr>
ect: SQL
age-ID: <ddff0add9-db91-e7fa-61fd-4e44545f9e7a@astro.unistra.fr>
: Tue, 26 May 2020 14:24:36 +0200
-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.12; rv:68.0)
ko/20100101 Thunderbird/68.8.0
ent-Type: multipart/mixed;
ndary="-----09658FE434D36264E12A5B16"
ent-Language: fr
comingHeaderCount: 15
rn-Path: laurent.michel@astro.unistra.fr
-Exchange-Organization-ExpirationStartTime: 26 May 2020 12:24:38.8173
C)
-Exchange-Organization-ExpirationStartTimeReason: OriginalSubmit
-Exchange-Organization-ExpirationInterval: 1:00:00:00.000000
-Exchange-Organization-ExpirationIntervalReason: OriginalSubmit

```
{  
    "from": "Laurent MICHEL <laurent.michel@arcor.de>","body": "code\r\n-- \r\n---- English version of the message\r\n\"in-reply-to\": \"none\",  
"timestamp": 1590495876.0,  
"id": "<ddf9add9-db91-ef7a-61fd-4e44546f9d1c>","references": "none",  
"subject": "SQL",  
"data": "none",  
"to": "cisternino enzo <enzo_cisternino@hsn.de>","num": "test1",  
"maillist": "test",  
"numThread": -1,  
"attachements": {  
        "text/plain_0": "U0VMRUNUICogRlJPTSE="},  
    "date": "Tue, 26 May 2020 14:24:36 +0200"  
}
```

A gauche, on voit une partie d'un mail dans le fichier Mbox, on voit plein d'attributs qui ne sont point utiles à la recherche. Il y a trop d'informations superflues.

A droite, c'est une capture d'écran du dictionnaire python qui est généré par la méthode saveMailsToElastic, il est tout de suite plus attractif, car il contient moins d'informations superflues et où chacun des champs pourra faire l'objet d'une requête.

D. Les pièces jointes

Il peut y avoir plusieurs pièces jointes au sein d'un mail, que ce soit des images, des documents, des vidéos, ou des fichiers audios, ces pièces jointes sont encodées soit en base64 : qui transforme un média en une chaîne de caractères pouvant être décodée par la suite pour reconstruire le fichier, soit dans d'autres encodages pour les formats textuels.

Il est nécessaire de savoir si le message en question en contient, et pour cela on utilise le module email, on utilise donc la méthode message.walk(), qui va séparer toutes les “grosses” parties d’un mail dans une liste que l’on parcourt avec une boucle “for”.

Pour chaque partie, on teste plusieurs choses. On récupère déjà le type de la partie.

```
type = part.get_content_maintype()
```

Cela nous permet de savoir si cette partie est intéressante ou non, voici les différentes valeurs qu’elle peut prendre :

- “multipart” : on peut ignorer cette partie, car elle ne sert qu’à spécifier si le mail contient des pièces jointes ou non, et n’a aucun autre intérêt
- “text/plain” : si parmi le parcours des parties, c’est la première fois que l’on rencontre ce type, c’est qu’il s’agit du contenu du mail, dans ce cas on le stocke dans l’objet destiné à être insérer dans Elastic Search : item[“body”] =
`part.get_payload(decode=True)`, `get_payload()` retournant le contenu de la partie
- Tout autre type signifie la présence d’une pièce jointe, il est donc important de la stocker.

Pour cela on va créer une clé pour le dictionnaire, qui sera constituée du type de la pièce jointe, de son encodage, et de son nom. On récupère l’encodage d’une partie d’un mail via la méthode : part.get(“Content-Transfer-Encoding”)

On récupère le type de la pièce jointe avec la méthode part.get_content_type(), et on récupère son nom avec la méthode part.get_filename().

Par exemple, pour stocker une pièce jointe qui est au format png :

```
item[“attachements”][part.get_content_type().__base64__image.png] = part.get_payload()
```

A la suite duquel j’ajoute dans le champ “attachements_name”, l’ensemble des extensions des pièces jointes, ce qui permettra de faire des recherches de mails contenant un type de fichier précis (par exemple ne retourner que les mails ayant des fichiers XML en pièces jointes).

E. Les problèmes d’encodage

L’encodage a été un problème majeur durant ce stage, il faut savoir qu’à l’intérieur d’un même fichier Mbox, l’encodage est variable.

Elastic Search ne supporte qu’un seul type d’encodage de texte (utf 8,ascii...) pour chaque champ d’un élément. Par défaut, chaque champ est encodé en UTF-8, j’ai donc décidé de décoder toutes les parties des messages que je sauvegarde en UTF-8.

Lors de la méthode saveMailsToElastic qui génère le dictionnaire à insérer dans ES, pour chaque attribut, je décode son contenu que je réencode par la suite, pour avoir un format UTF-8.

Par exemple, pour récupérer le sujet d'un message :

```
item[“subject”] = message.get(“Subject”).encode(“utf-8”, “ignore”).decode(“utf-8”, “ignore”)
```

Malheureusement, il y a quelques cas spéciaux qu'il a été jugé facultatif de corriger car trop rares, en effet certains attributs de mails commencent par une séquence étrange (ex =?UTF-8=ED=ED=ED?=), qui représente la tentative de décodage de Mbox à la sauvegarde du fichier, nous avons donc décidé pour ces cas précis (qui arrivent majoritairement que dans ces cas précis : From, To), d'utiliser une expression régulière pour supprimer ce type de contenu, et de faire avec ce qui est proprement décodé.

F. La reconstitution des dates

Pour stocker le timestamp (UNIX) de chaque message, j'ai dû utiliser le champ “Date” du mail, je pensais au départ l'utiliser directement pour le mettre dans la base, mais il s'est avéré que le format n'est pas le même pour chaque mail, j'ai donc décidé de calculer le timestamp Unix de chaque message.

En plus de ne pas se soucier du format différent de chaque date, il permet à Elastic Search de faire des tris par date, car on ne peut pas faire de tri sur des chaînes de caractères via ES, et donc vu que le timestamp est stocké comme un entier, plus aucun problème n'est présent.

VII. Indexation des messages

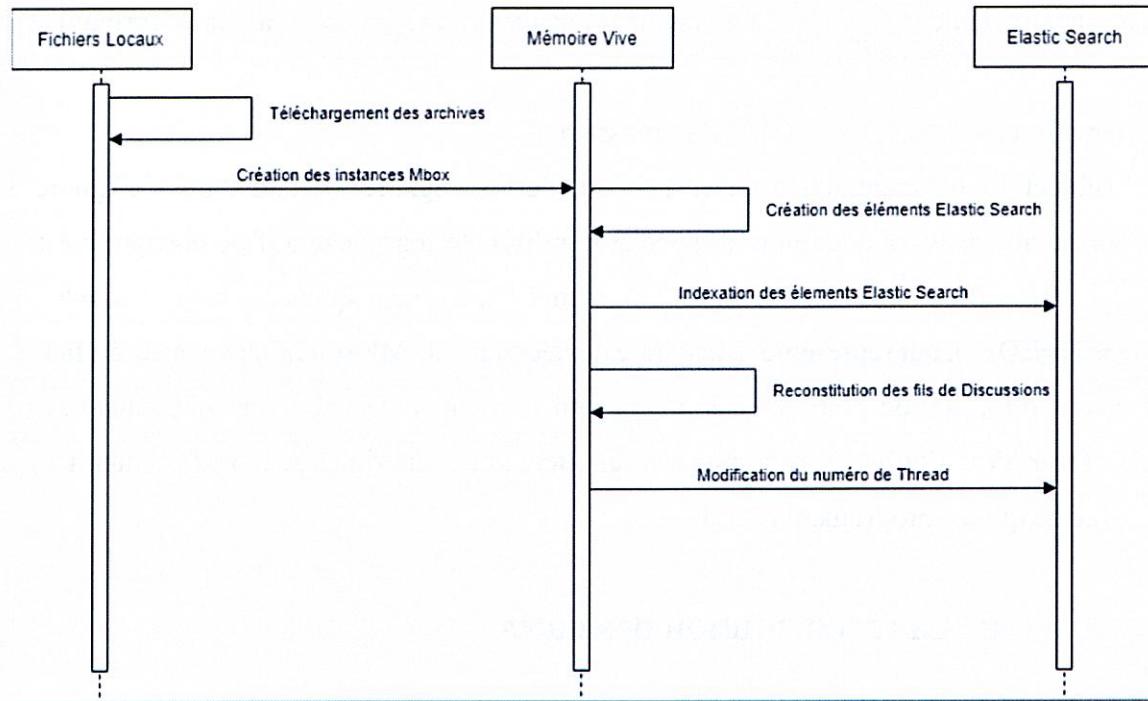
A. Objectifs de l'indexation

L'indexation de chaque message est critique dans le processus complet de création de l'outil de recherche, chaque message doit être indexé correctement, et que l'on soit sûr de son indexation. Il est nécessaire d'avoir des index propres sans duplicates de mails et sans informations superflues qui pourrait perturber l'affichage des résultats de requêtes.

Dans le cadre où l'outil devra être mis en ligne, il est nécessaire qu'à chaque mise à jour de la base de données (nouveaux mails envoyés et reçus par Pipermail) l'ensemble des données stockées restent accessibles durant le processus. Cela permet de n'avoir aucune coupure et de proposer un système viable et performant.

Il est aussi important de paramétriser les index pour gérer le nombre de requêtes acceptées.

Voici un résumé du processus :



B. Choix du nom de l'index

Nous avons choisi de proposer deux choix d'indexation, soit l'utilisateur souhaite un seul index pour tous les mails de toutes les archives, soit il souhaite un index global.

Dans le cas d'un multi-index, le nom de l'index est composé du nom de la liste de diffusion suivi d'un suffixe "temp" ou "new".

En effet, au premier lancement du programme il créera un index dm_new par exemple et si par la suite, on relance le programme, alors cet index se nommera dm_temp.

Dans le cas d'un index global, tous les mails sont stockés au même endroit, et le seul moyen de les différencier est d'aller voir la donnée stockée dans le champ "maillist" des éléments.

Le choix d'un index global est justifié dans le cadre d'une requête sur plusieurs listes de diffusions, ce qui permettra de faire une requête sur une liste précise, ou alors sur toutes les listes de diffusions stockées dans Elastic Search, c'est une fonctionnalité supplémentaire qui n'est pas possible avec de multiples index. Ce fonctionnement est celui retenu pour le site en ligne, mais l'inverse est implémenté et peut être utilisé par une autre personne.

C. Le paramètre pour les index

Il est possible de spécifier des paramètres à la création d'un index dans Elastic Search, des utilisateurs avancés peuvent modifier en profondeur le fonctionnement de chaque index,

Ici, nous nous sommes servis de cette fonctionnalité pour limiter la taille des requêtes. En effet en spécifiant la valeur max_result_window on peut empêcher un utilisateur d'effectuer des requêtes demandant un nombre d'éléments supérieur à la limite, c'est notamment utilisé pour empêcher des personnes malveillantes d'effectuer des requêtes qui pourraient mener à mal l'état du serveur sur lequel Elastic Search est hébergé.

Pour modifier un paramètre ou ajouter des mappings (configurations par défaut), on joint à la création d'un index via la méthode es.indices.create, un paramètre nommé body qui est en fait un dictionnaire représentant les paramètres que l'on souhaite ajouter.

D. Indexation d'un message

Une fois l'élément créé par le module Elasticer, et les index mis en place, il nous suffit d'indexer chaque message un à un pour remplir totalement chaque index.

Il suffit d'une méthode pour ajouter un élément à un index :

```
es.index(nom_index,doc_type,body,timeout)
```

doc_type correspond au type de document que l'on ajoute, il est obligatoire pour la fonction mais n'a pas d'intérêt si on ne configure pas des mappings à la création de l'index, j'ai donc spécifié le type "keyword" mais cela n'a aucun effet pour la suite.

body est tout simplement l'élément construit à partir d'un mail dans la méthode saveMailsToElastic.

request_timeout est le temps maximum que peut prendre une insertion, ce paramètre est facultatif, mais je l'augmente par défaut pour éviter des problèmes de délai liés à l'indexation d'un très grand nombre de nouveaux mails.

Je me suis assuré en amont que je n'ajoute qu'une seule fois un mail précis par liste de diffusion, cela consiste à créer un tableau de tous les ids déjà indexés, et, pendant le parcours de la boucle for, ne pas indexer un mail si l'id du message courant est déjà présent dans le tableau.

E. Reconstitution des fils de discussion

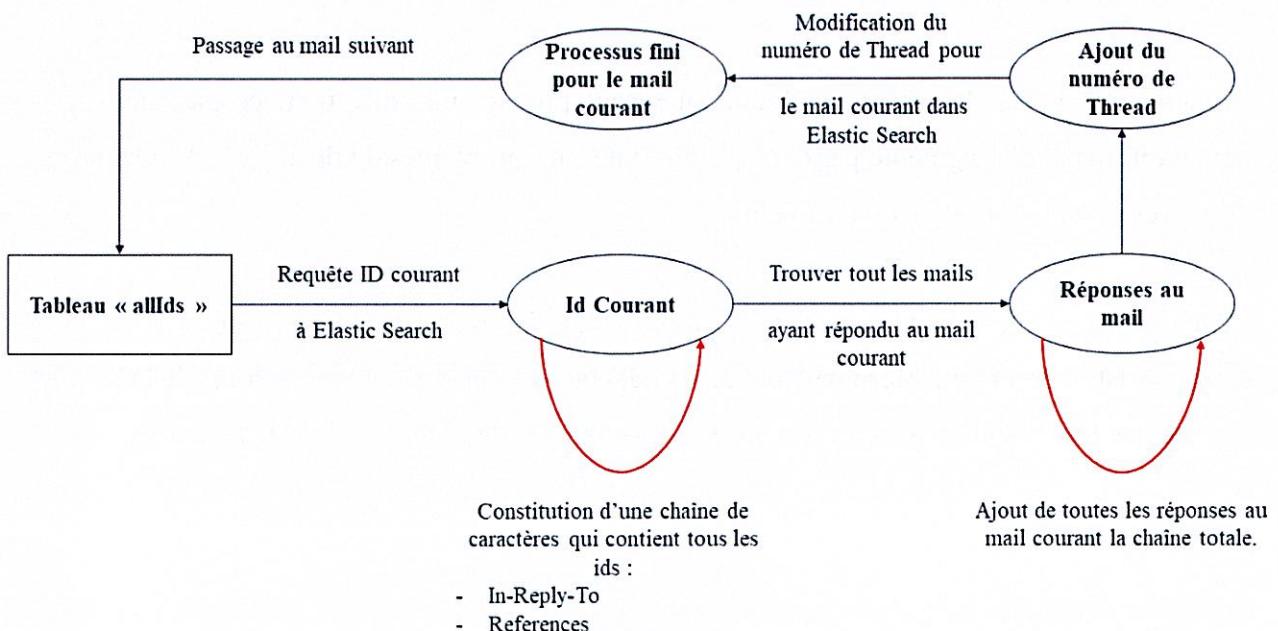
Un fil de discussion est un ensemble de mails qui constituent un “Thread” : discussion sur un sujet, cette notion de Thread n'est pas directement gérée par Mbox mais doit être reconstruite par le client.

Il est important de reconstruire ces Threads pour la suite, car c'est une fonctionnalité majeure pour la création de l'outil web, il ne sert à rien de trouver un mail si on ne peut pas retrouver à quoi ce mail a répondu, à qui, etc...

Il est possible de reconstruire les fils de discussion avec les informations de 3 attributs de chaque mail : Message-ID , In-Reply-To , References

A partir de ces 3 informations, on peut reconstruire les fils de discussions de chaque Thread, et donc renseigner le “numThread” présent dans chaque élément d’Elastic Search. La construction des threads s'effectue après l'indexation de tous les messages avec la méthode addThreads(allIds,indexName,es_url,mailList)

“allIds” est l'ensemble des ids des messages que j'ai indexés auparavant. “indexName” est le nom de l'index qui contient les mails, “es_url” est l'url pour construire l'instance d'Elastic Search, “mailList” est le nom de la liste de diffusion sur laquelle on travaille, elle peut être la même que index_name dans le cas des multi-index, mais ils ne sont pas les mêmes pour un index global. Voici ci-dessous le processus entier pour « Reconstruire les Threads » :



A la fin de ce processus, on a donc tous les threads d'une liste de diffusion, correctement reliés, et bien indexés malgré quelque fois des problèmes dû au fait que les personnes changent de sujet mais sur le même fil de discussion ou continuent le message sur une autre liste de diffusion.

F. Disponibilité des index sous maintenance

Il est important que les index restent accessibles quand le programme de mise à jour des données s'effectue. L'outil en ligne reste donc disponible pendant que l'on met à jour les mails. On se place dans le cadre d'un index global car c'est le mode par défaut du site. Pour cela, on choisit dynamiquement le nom des index en testant s'il existe déjà. On teste si ivoa_all_new existe, si oui, alors on crée un index ivoa_all_temp et inversement. Cela permet de ne supprimer aucun mail ou index pendant la mise à jour.

Une fois que ces index sont créés, les messages s'indexent via le processus décrit plus haut, puis c'est au tour des Threads d'être construits. Une fois tout le processus terminé, c'est ici que ça se complique, on doit supprimer les vieux index en les remplaçant par les nouveaux.

Le problème est qu'il est impossible de renommer un index dans Elastic Search, la solution que j'ai trouvée est donc de créer des alias pour chaque index. Chaque index se nomme soit ivoa_all_new ou ivoa_all_temp, mais leur alias pointeront toujours vers "ivoa_all".

Cela permet de ne pas avoir à modifier les requêtes et de toujours demander des informations au même nom d'index alors que techniquement, ce n'est plus le même. Cette opération est gérée par la méthode mergeIndex, qui s'occupe de supprimer l'index ancien et d'ajouter l'alias au nouvel index créé.

VIII. Création de la solution web

A. Objectifs de la solution web

La solution web est le lien entre le travail effectué par l'indexation des messages et les besoins de l'utilisateur. Il se doit d'être rapide, efficace et précis, et graphiquement agréable pour que l'utilisateur souhaite l'utiliser.

En terme d'efficacité, il doit être capable de retrouver les mails à partir de mots-clés, de les afficher de façon pertinente, et d'en présenter l'ensemble des informations (pièces jointes, threads,etc...). Il doit aussi être agréable à l'œil car il doit à la base ,améliorer l'affichage des messages par Pipermail.

B. Création de l'architecture HTML

Il a donc fallu construire une architecture fiable qui permette de retranscrire au plus simple l'affichage et la recherche de mails, pour l'aspect graphique, j'ai décidé de travailler avec Bootstrap. Cette interface doit permettre le parcours suivant pour un utilisateur : arrivé sur le site, il tape un mot dans la barre de recherche, il exécute la requête, et pour toute autre action plus ciblée, il utilise des widgets spécifiques.

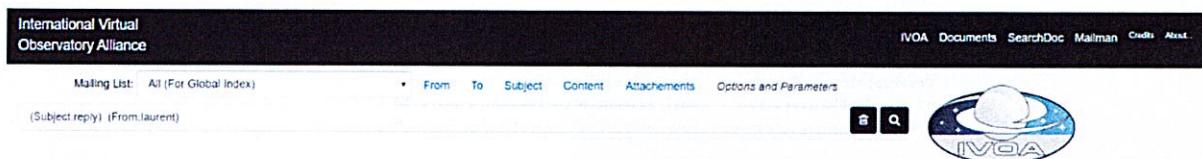
Tout d'abord, j'ai décidé de faire une barre de navigation qui comprendra l'ensemble des liens qui peuvent s'avérer utiles : la page d'accueil de l'IVOA, les listes de diffusion, un outil de recherche documentaire, les crédits, etc...

Ensuite, le formulaire, il a pour but de réunir l'ensemble des champs nécessaire à la construction d'une requête, voici une description de ces éléments :

- Choix de la mailing list avec une liste déroulante qui contient toutes les listes de diffusions (archivées) de l'IVOA
- 5 boutons cliquables qui seront utiles pour la création d'une requête : From, To, Subject, Content, Attachements
- Un bouton “Options and Parameters” qui permet de dérouler un sous-menu de fonctionnalités avancées, dans ce menu se trouve :
 - Un menu déroulant qui permet de choisir l'opérateur booléen pour la construction de la requête : OR ou AND
 - Un menu déroulant qui permet de choisir la méthode de tri des messages : par relevance, par date ascendante, par date descendante
 - Un groupe de boutons « checkbox » permettant de choisir le nombre de mails par requête
 - Deux « DatePicker » permettant de délimiter la période d'un mail
 - Un lien « Reset Default » qui reset la barre de recherche

- Un lien « See Query » qui affiche dans une boîte modale le contenu de la requête qui va être effectuée
- Un lien « executeQuery » qui lance la requête
- Un champ pour taper le texte que l'on souhaite rechercher suivi de deux icônes : une corbeille et une loupe, qui respectivement suppriment le contenu de la barre de recherche et lance la recherche

J'ai également ajouté le logo de l'IVOA sur la droite du formulaire.



Voici pour la partie présente en permanence sur le site. L'affichage des mails est géré d'une manière différente.

J'ai utilisé l'ensemble « fieldset + legend » pour l'affichage un mail et des boîtes modales pour les threads de chaque mail.

L'ensemble « fieldset + legend + Bootstrap » collapse permet de créer des sections déroulantes pour chaque mail qui se déroulent en cliquant sur le titre du mail, en voici un exemple

N°:1/3 18/05/2020 -DM- From : Laurent MICHEL Subject : Please do not reply

Voici l'aspect d'un mail quand il n'est pas déroulé. Cela permet de gagner de la place et de ne pas afficher le contenu de tous les mails courants.

N°:1/3 18/05/2020 -DM- From : Laurent MICHEL Subject : Please do not reply

[View Thread](#) [Display Mail URL](#)

FROM : Laurent MICHEL **TO :** "dm"
SUBJECT : Please do not reply **DATE :** 18/05/2020

Dear DM,

This is a message sent to test a pipermail search engine developed by an intern.

PLEASE DO NOT REPLY

--

---- English version:

<https://www.deepl.com/>

---- Laurent MICHEL Tel (33 0) 3 68 85 24 37
Observatoire de Strasbourg Fax (33 0) 3 68 85 24 32
11 Rue de l'Université Mail laurent.michel@astro.unistra.fr
67000 Strasbourg (France) Web <http://astro.u-strashbg.fr/~michel>

[Close Mail](#)

Et voici l'aspect d'un mail déroulé. Le principe de Collapse permet de dérouler ou d'enrouler une div pour gagner de l'espace et éviter de surcharger l'œil de l'utilisateur.

Finalement, voici l'aspect total du site avec quelques mails.

International Virtual Observatory Alliance

Mailing List All (For Global Index) From To Subject Content Attachments Options and Parameters (Subject:reply) (From:laurent)

5 mails found

N°:1/5 18/05/2020 -DM- From : Laurent MICHEL Subject : Please do not reply

N°:2/5 28/05/2020 -DM- From : Laurent MICHEL Subject : Test mail #2 DO NOT REPLY

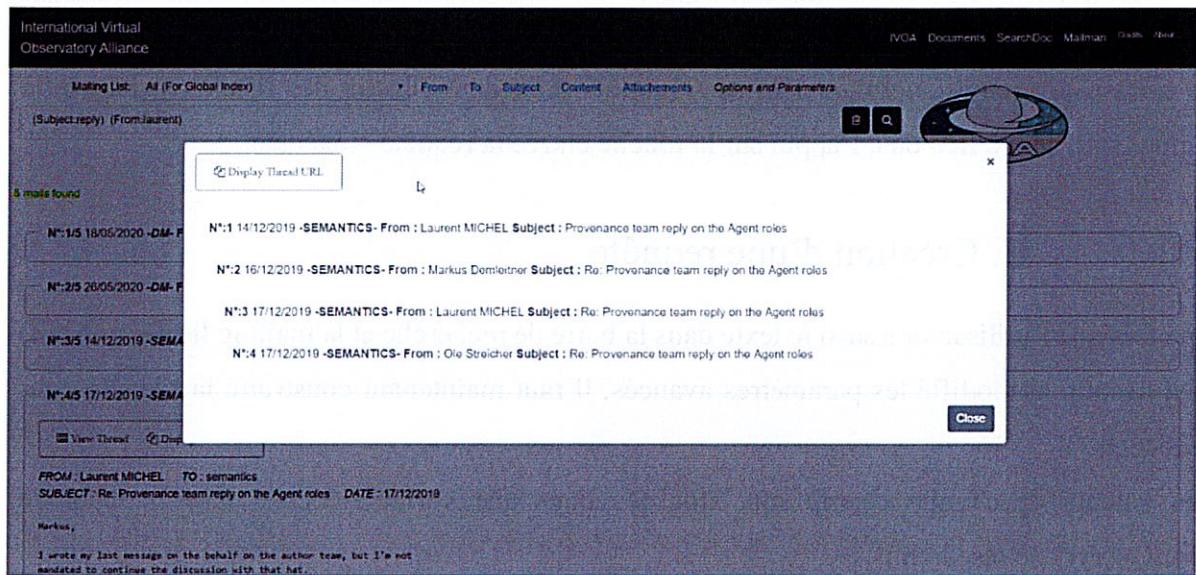
N°:3/5 14/12/2019 -SEMANTICS- From : Laurent MICHEL Subject : Provenance team reply on the Agent roles

N°:4/5 17/12/2019 -SEMANTICS- From : Laurent MICHEL Subject : Re: Provenance team reply on the Agent roles

N°:5/5 25/05/2020 -DM- From : Laurent MICHEL Subject : Test mail DO NOT REPLY



Et voici l'aspect du site quand on affiche le Thread d'un mail précis :



C. Choix graphiques

Concernant la partie graphique, nous avons décidé d'utiliser Bootstrap, qui permet de créer facilement le style d'une page Web. Nous avons décidé de reprendre le style graphique du site de l'IVOA, qui est une alternance de noir et de blanc, sans beaucoup d'éléments graphiques (images).

Tout est léger, avec des bordures fines et du texte non agressif, le but est de produire une interface légère. La présence de bordures arrondies permet de réduire l'agressivité du site, et la présence d'icônes représentatives des actions permet de faciliter l'utilisation pour des utilisateurs qui ont peu d'expérience Web.

D. Implémentation du formulaire

Le formulaire est l'élément le plus important dans la création des requêtes sur le site Web. La barre de recherche contient tous les éléments textuels que doit trouver la requête, comme à la manière de google, il suffit de taper des mots pour effectuer des recherches qui contiennent ces mots, mais il est possible de faire des recherches plus ciblées selon les besoins.

Les liens "From" "To" "Subject" "Content" "Attachments", permettent d'insérer du texte automatiquement dans la barre de recherche. Par exemple pour le lien "Content" : l'appui sur le bouton ajoute dans la barre de recherche (Content:), il suffit de taper du texte après les deux

points et entre les parenthèses pour spécifier que le terme doit être contenu dans le champ “Content” des mails retournés.

Le bouton “Execute Query” et l’icône « loupe » possèdent chacun des listeners, pour qu’à l’appui sur l’icône/lien ou à l’appui sur la touche entrée la requête s’exécute.

E. Création d’une requête

Une fois que l’utilisateur a saisi le texte dans la barre de recherche et la mailing list (si non All) et si besoin, ai modifié les paramètres avancés, il faut maintenant construire la requête pour l’exécuter.

Les valeurs de chaque champ sont stockées dans une variable locale dans la méthode formQuery() de research.js.

On a donc un ensemble d’informations :

- Le nom de la liste de diffusion
- Le contenu de la barre de recherche
- La méthode logique de la requête
- L’ordre de tri des mails
- Le nombre de mails dans la réponse
- L’éventuelle période des mails

Une requête Elastic Search se construit avec un fichier JSON :

- le champ “Size” représente le nombre de mails max que peut retourner la requête
- le champ “Query” contient le corps de la requête
- le champ “Sort” définit l’ordre de tri des mails

Les requêtes fonctionnent de manière logique. Elles sont constituées de mots clés définissant la logique de chacune.

```
{  
    "size":10,  
    "query":{  
        "bool":{  
            "must":{  
                "term":{  
                    "from":{"value":"Enzo"}  
                },  
                "term":{  
                    "to":{"value":"Laurent"}  
                }  
            }  
        },  
        "sort":{  
            "timestamp":{"order":"asc"}  
        }  
    }  
}
```

Cette requête retourne un maximum de 10 mails dont le destinataire est “Laurent” et l’expéditeur est “Enzo”, on trie les résultats par date et par ordre croissant.

Le mot-clé “bool” spécifie que c’est une condition logique qui doit être valide pour que le mail soit considéré comme valide.

Le mot-clé “must” spécifie que tous les termes contenus dans le champ must doivent être vérifiés sinon le mail ne sera pas retourné. L’idée générale du “must” c’est qu’il représente l’opérateur logique “AND”.

Si on remplace le mot-clé “must” par “should” alors la méthode logique de la requête est “OR”.

En jonglant avec ces champs-clés et en les imbriquant, on peut créer des requêtes complexes avec plusieurs champs à vérifier ou non.

Pour revenir à la création de la requête, tout d’abord, on teste si la barre de recherche est vide ou non, si elle est vide, alors on retourne un échantillon de N mails sans critères de recherche (on peut tout de même trier par date les mails retournés) , sinon on entame le processus de création de requête.

On récupère le timestamp des DatePicker, s'ils sont spécifiés, on les garde en mémoire, sinon on les ignore. On récupère ensuite la mailing list, si le champ est laissé sur "All", alors il n'y a pas de mailing list spécifiée et la requête se fait sur toutes les listes de diffusions. La taille de la requête est choisie en fonction du checkbox qui est coché (par défaut : 20) dans les paramètres avancés. Ensuite on récupère la méthode logique choisie pour la requête, ce qui va nous permettre de construire le corps de la requête.

En parallèle, on utilise des expressions régulières pour tous les attributs, ce qui nous donne une liste de mots pour chaque attribut.

On parcourt chaque liste et on ajoute une clause pour chaque attribut spécifié, ces clauses sont appelées "wildcard" et se composent de cette forme : "wildcard" : {"from" : "*laurent*"}, ce qui permet de rechercher les champs qui contiennent une partie du mot ou le mot entier.

En fonction de la méthode logique, on ajoute chaque clause dans un "must" ou dans un "should". Si l'utilisateur avait saisi du texte sans spécifier d'attributs (donc sans parenthèses), alors le mot est recherché parmi tous les attributs du mail (From,To,Content,Subject).

Une fois tout cela constitué, on peut passer à l'exécution et au traitement de la requête.

F. Traitement et exécution d'une requête

La méthode executeQuery() permet d'effectuer la requête passée en paramètres, si on est dans le cas d'un index global alors on effectue la requête sur ivoa_all, sinon, on l'effectue avec la mailing list choisie dans le menu déroulant.

J'utilise le module Axios pour effectuer les requêtes asynchrones à Elastic Search, qui permet de faire des ".then" et ".catch" en cas de succès ou d'échec de la requête.

En cas de succès on appelle la méthode traitementMessage(), et dans l'autre cas, on affiche une alerte avec l'erreur. La méthode traitementMessage parcourt l'ensemble des résultats de la requête et pour chacun, ajoute au DOM un nouvel ensemble fieldset+legend.

La méthode ajoute aussi un bouton "View Thread" et "Display URL" dont les fonctions seront expliquées par la suite. Le titre de chaque mail est une combinaison du numéro du mail (1/50 ,

2/50,etc...), de la date du mail, du nom de la liste de diffusion, de l'expéditeur, ainsi que le sujet du mail.

Dans le contenu de chaque mail affiché : on retrouve l'expéditeur, le destinataire, la date, le sujet, et un bouton “View” et “Download” pour afficher/télécharger chaque pièce jointe.

On a donc à ce point une “div” où tous les messages sont contenus et sont “déroulables”, les informations importantes de chaque mail y sont présentes et les boutons d’actions y sont présent sous formes d’icônes.

G. Création des threads

Le bouton View Thread créé précédemment permet d’ouvrir la boîte modale qui va être créée plus tard. La création des threads se fait via le module threads.js, qui, pour chaque mail ajouté, reconstruit le thread dans une boîte modale avec le même principe de fieldset+legend en utilisant le numéro Thread attribué auparavant.

On appelle la méthode addModal qui ajoute une boîte modale pour le thread courant, puis on appelle la méthode addModalThread avec le numéro de Thread du mail courant, qui nous permet de trouver avec une simple requête, tous les mails ayant le même numéro de thread que lui, après cela, on affiche tous les mails par ordre chronologique dans la boîte modale.



H. Coloration Syntaxique

La coloration syntaxique permet de changer par exemple la couleur d'un contenu textuel en ne ciblant que quelques parties, c'est notamment utilisé par tous les IDE pour modifier la couleur des mots-clés dans le code.

J'ai dû implémenter cette coloration avec deux méthodes : celle faite par ma part pour le contenu des mails, puis celle en utilisant Prism pour du code.

A chaque ajout de contenu de mail, ce contenu passe par une méthode highlight() qui va tout simplement changer la couleur de certaines lignes du message, ici on change la couleur des lignes qui citent d'anciens mails :

```
> The nature of these tag is not defined yet, but I'm convinced that we
> need it.
> The breaks a little bit the standard modelling approach but I'm afraid
> that this is necessary to get some interoperability level for archival
> data.
>
>
> Laurent
>
>
>
> Le 21/02/2020 à 09:51, Markus Demleitner a écrit :
>> On Thu, Feb 20, 2020 at 02:11:16PM -0500, CresitelloDittmar, Mark wrote:
>>> *) Domain specific Measurement types ("Time", "Position",
>>> "Polarization") vs "GenericMeasure"
>>> *RESPONSE:* We feel there is a distinct advantage to being
>>> able to
>>> define that Field X is a "Time" and Field Y is a "Position", rather
>>> than
>>> having everything be a generic measurement type.
```

Cela permet de mieux visualiser le contenu d'un mail et de mieux se repérer dans les parties importantes de celui-ci.

Prism quant à lui est utilisé pour la coloration syntaxique de code dans les pièces jointes, pour l'utiliser, il suffit d'insérer le code dans un élément "<pre><code>" dont l'élément "code" a pour classe le type de langage utilisé.

Pour savoir le type de langage utilisé, il suffit de récupérer le nom de la pièce jointe et de la diviser avec comme séparateur ".", ainsi on retrouve l'extension du fichier qui permet à Prism de colorier adéquatement le code.

I. Téléchargement et affichage des pièces-jointes

Pour chaque mail, il est nécessaire de présenter les pièces jointes, que l'on pourra soit voir dans une boîte modale, soit télécharger sur son ordinateur.

Pour commencer, il faut récupérer le champ “attachements” du mail et le transformer en objet JSON avec la méthode `JSON.parse(attachements)`, puis on parcourt chaque élément (qui constitue une pièce jointe). Pour chaque pièce-jointe on ajoute un lien de téléchargement, et pour les formats textuels ou d’images, il est proposé d’avoir un affichage dans une boîte modale de ces pièces jointes.

Pour générer le lien de téléchargement, on crée un élément “`<a>`” avec une balise `href` de la forme suivante :

- “data:” + type MIME + encodage + contenu de la pièce jointe

Le mot clé data signifie que la suite représente un objet. Le type MIME est un identifiant de format de données sur internet. Par exemple pour un fichier au format .txt cela donnera : `text/plain`. Pour une image au format .png ce sera : `image/png`. On récupère ce champ en parsant la clé du dictionnaire “attachements” dont chaque clé contient, le type MIME, l’encodage et le nom complet de chaque pièce jointe.

Le contenu de la pièce jointe est donc spécifié en récupérant la valeur associée à la clé dans le dictionnaire “attachements”, mais il faut l’encoder auparavant avec la méthode `encodeURI()`.

Une fois tous les liens de téléchargement faits, il est nécessaire de créer les boîtes modales pour afficher chaque pièce jointe. Le site permet donc d'afficher les pièces jointes suivantes : les pdf, les images, et les formats textuels.

Pour les images et les textes, on remplace simplement la balise “`<a>`” par une balise “`<pre><code>`” ou “``” suivant le format de celles-ci.

Pour les pdf, il est nécessaire d’imbriquer le pdf dans un ensemble “`<object><embed>`” mais la déclaration de la pièce jointes reste la même mais l’attribut `href` est remplacé par un attribut `src`.

J. Lien par mail et requête via URL

Chaque mail contient un identifiant unique “num”. Il était nécessaire de pouvoir partager un mail, de façon qu'une personne trouvant une information puisse envoyer un lien de partage à une autre personne, qui pourrait consulter à son tour le mail en question.

Pour cela, il est ajouté sur chaque mail, chaque thread, et chaque mail de thread, un bouton permettant d'afficher dans une boîte modale le lien contenant en paramètres le numéro du mail ou du thread.

Ce qui permet à l'utilisateur de copier-coller pour pouvoir le partager, l'utilisateur rentrant cette url dans la barre de recherche arrivera sur le site avec directement le mail ou le thread d'affiché.

Pour cela, on crée pour chaque mail, chaque thread un bouton qui, à l'appui, appelle la méthode qui affiche la boîte modale en en modifiant le contenu avec le paramètre fourni.

L'url est construite de la façon suivante :

- location.protocol + location.host + location.pathname

Cela construit la base de l'url en supprimant les paramètres s'il y en avait déjà auparavant. A la suite duquel on ajoute un paramètre “num” ou “thread” en fonction de si l'on décide d'afficher un mail ou un thread.

A l'arrivée sur le site web, celui-ci part à la recherche de paramètres dans l'url, s'il trouve un des deux paramètres pris en compte, alors il construit la requête en fonction du paramètre fourni, et affiche le mail dans le cas d'un mail, ou le premier mail d'un thread + sa boîte modale de Thread affichée par défaut dans le cas d'un Thread.

IX. Déploiement

A. Intérêt

Pour le déploiement des services, il est nécessaire d'avoir un déploiement simple et facile d'utilisation, ce qui a permis pour la fin de stage, de ne pas rencontrer de problèmes de compatibilité dûs à des paramètres codés “en dur” dans le code de chaque programme.

B. Config.json

Config.json représente l'ensemble des paramètres nécessaires pour l'indexation des mails et la récupération des archives.

```
{  
    "download_uri" : "http://mail.ivoa.net/pipermail/",  
    "elastic_search_url" : "http://localhost:9200",  
    "mbox_dir": "../mbox",  
    "reset_index": 1,  
    "global_index": 1,  
    "limit_request": 50,  
    "mailing_lists": [  
        {  
            "label": "Data Model",  
            "index_name": "dm",  
            "prefix": "dm_",  
            "download_mbox": "dm.mbox/dm.mbox"  
        }  
    ]  
}
```

“download_uri” renseigne l'url par défaut du site d'où l'on télécharge les archives MBOX.

“elastic_search_url” renseigne l'url d'hébergement du Elastic Search.

“mbox_dir” est le chemin où sera stocké toutes les archives MBOX.

“reset_index” est un entier qui décrit si l'on souhaite supprimer tous les index au démarrage ou non, globalement, veut-on rendre disponible le service pendant la mise à jour ou non.

“global_index” est aussi un entier spécifiant si l'on souhaite avoir un index par mailing list, ou un index global pour toutes.

“limit_request” est la limite de chaque requête pour un index, c'est le paramètre de l'index qui empêche de faire des requêtes de taille non raisonnable.

“mailing_lists” est un tableau qui stocke toutes les mailing list à télécharger et indexer dans Elastic Search, pour chaque élément :

- “label” représente le nom entier de la liste de diffusion
- “index_name” décrit le nom de l'index Elastic Search, si le choix est un multi-index,
- “prefix” représente le préfixe de chaque numéro unique de mail, ce qui permet de mettre en place le lien de partage
- “download_mbox” est la suite de l'url de l'IVOA pour télécharger chaque archive.

A l'aide de ce fichier de configuration, on choisit tous les paramètres pour l'indexation, on peut par exemple tout mettre pour une utilisation complète, ou que quelques listes de diffusion pour du débogage.

C. Mise en ligne du site web

Étant en télétravail, je codais chaque jour sur mon IDE personnel, et quand nous souhaitions mettre en ligne le site, il fallait que je modifie quelques paramètres pour que le site s'adapte au changement d'hébergeur, pour cela, nous avons déclaré dans le fichier index.html deux variables représentant l'adresse d'Elastic Search, à chaque mise en ligne du site, on alternait la mise en commentaire de mon adresse ou de celle de M. Michel.

D. Disponibilité de l'outil sous mise à jour

Comme expliqué ci-dessus, nous avons mis en place la fonction de disponibilité sous mise à jour, ce qui permet de maintenir les recherches de mails alors que l'on met à jour tous les mails stockés dans Elastic Search.

M. Michel a programmé sur son serveur, le lancement automatique du script de téléchargement et d'indexation des messages . Chaque lancement est effectué toutes les 4 heures, cela veut dire que toutes les 4 heures, les mails sont mis à jour dans Elastic Search, tout en maintenant le service actif.

E. Configuration Serveur

Le côté serveur a quelques configurations qu'il est important de spécifier :

- notamment la limitation appels HTTP, pour les limiter seulement en GET et POST et bloquer les requêtes DELETE
- ajout dans la “cron table” l'exécution du script d'indexation des mails

```

<Location "/elasticsearch">
    ProxyPass http://galhecos:9200
    ProxyPassReverse http://galhecos:9200
    Header add "Access-Control-Allow-Origin" "*"
    <Limit GET POST OPTIONS>
        Order allow,deny
        Allow from all
    </Limit>

0 */4 * * * /bin/sh -c 'cd /databox/databases/MailIndexation/vo-grimoire/serveur/python/launcher &&
/usr/bin/python3.6 main.py > /databox/databases/MailIndexation/logs/main.logs 2>&1'

```

En haut on retrouve la configuration du serveur pour bloquer ces requêtes, et en bas on retrouve la programmation du script qui toutes les 4 heures exécute la mise à jour des mails.

X. Validation par l'observatoire virtuel

A. Mise en ligne et intérêt

Depuis le début et même avant, il était bien spécifié que le stage devait arriver sur une solution viable et complète qui devra être mis en ligne avant la fin du stage, afin d'avoir des retours quant à son utilisation.

L'intérêt de la mise en ligne du projet est tout simplement de pouvoir avoir des retours et notamment de pouvoir en prendre en compte dans le développement, sachant que dans le futur, ce projet restera en ligne.

B. Retours et Activité

Le site a été mis en ligne le 9 Juin 2020, de nombreux utilisateurs ont pu consulter et utiliser le site et faire des recherches.



Voici une capture d'écran de l'activité du site, on peut voir des utilisateurs de différents pays qui restent plus ou moins longtemps, en parallèle de ça, M. Michel et moi recevions des mails de personnes faisant des retours / des suggestions / des questions.

Nous avons eu des retours de différents pays : Etats-Unis, Chine, France, Allemagne.

C. Prise en compte des retours

M. Streicher est la personne qui nous a envoyé le plus de retours et de suggestions, notamment des incohérences dans les Labels des boutons, tout simplement par exemple : suggérer de modifier “Advanced Parameters” par “Parameters and Options” ou encore modifier les “placeholder” dans les champs des formulaires.

Il nous a fait également remarquer des problèmes dans la visualisation de la requête, que j’ai corrigés par la suite. M. Cui nous a demandé, quant à lui, de masquer les adresses mails, modification que j’ai apportée pour éviter que certaines personnes récupèrent des adresses pour faire des spams.

Sa remarque nous a montré que l’outil est puissant, car de base tous les mails sont disponibles librement sans authentification, et l’outil développé permet de retrouver très facilement les mails, il y a donc plusieurs personnes qui se sont inquiétées de la confidentialité du site, alors que toutes les données utilisées sont publiques.

M. Mantelet quant à lui, nous a posé plusieurs questions de curiosité (taille des index, type d’index), auxquelles j’ai répondu, car c’est très intéressant de communiquer avec les personnes utilisant l’outil.

XI. Conclusion

Enrichissant. C’est le mot-clé qui résume le plus mon ressenti personnel quant à ce stage, étant dans un contexte de crise sanitaire, j’étais sceptique quant au bon déroulement du stage, et également le fait que je ne connaissais pas du tout les outils utilisés : Elastic Search ainsi que le langage Python par exemple.

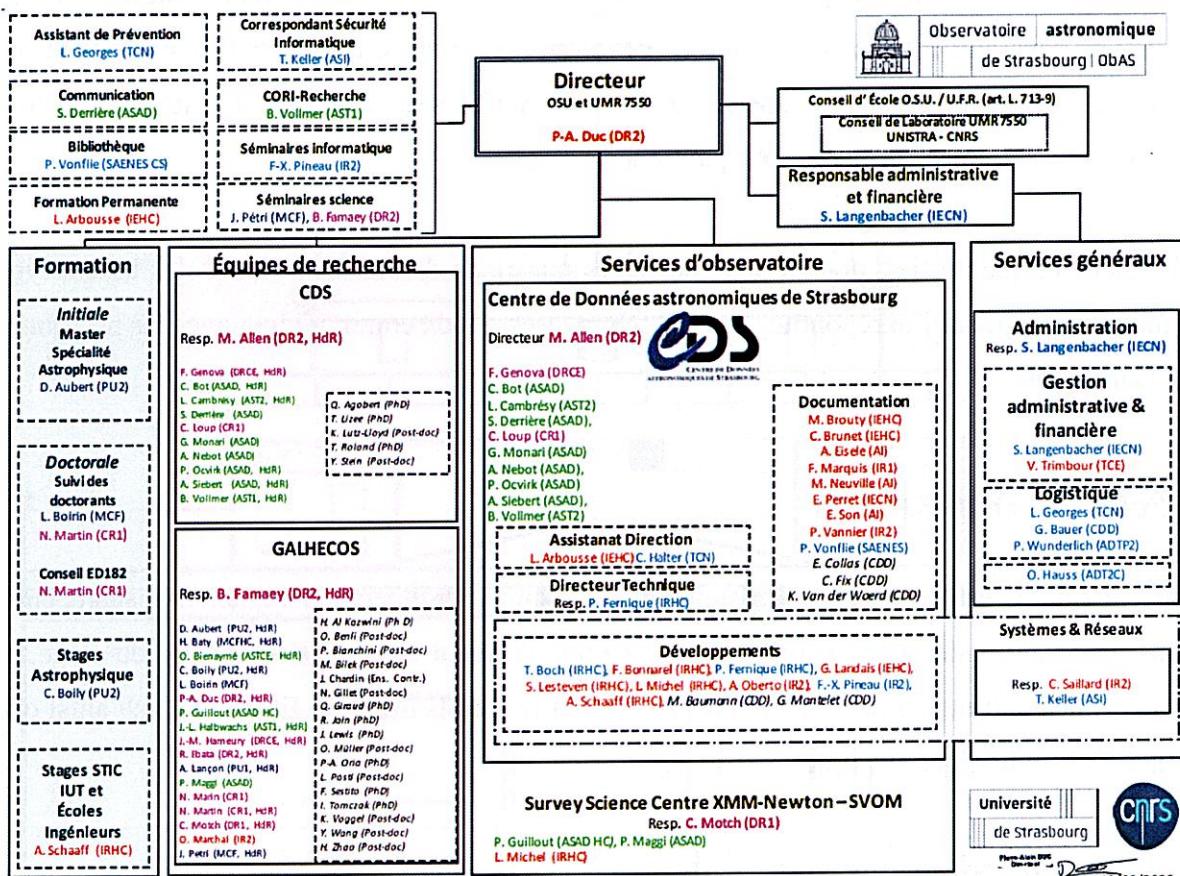
La découverte de nouveaux outils, le développement de programmes à caractère professionnel et voués à être utilisé ont été une grande source de motivation pour moi.

La création du script d’indexation et le site web ont été très enrichissants car devant être adaptables et facilement modifiable.

Quelques points négatifs comme les problèmes d'encodage ou la reconstitution des Threads sont apparus mais ont vite été réglés . Je suis néanmoins déçu de ne pas avoir pu travailler au sein de l'Observatoire, mais les conditions de mon télétravail étaient parfaites et donc cela n'a pas entaché mon quotidien.

Pour conclure, ce stage a été une ouverture d'esprit quant au monde professionnel ainsi qu'à la satisfaction de voir un produit fini et fonctionnel, ce sentiment améliore énormément ma confiance en soi, et conforte mon souhait de travailler dans le domaine de l'informatique, et je remercie sincèrement M. Michel de son investissement dans ce stage en tant que tuteur.

XII. Annexes



Annexe N°1 : Organigramme de l'Observatoire Astronomique de Strasbourg.

XIII. Références et Bibliographie

- [IVOA,2020] IVOA, <http://ivoa.net>
- [Wikipédia, 2020] Wikipédia : Observatoire astronomique de Strasbourg,
https://fr.wikipedia.org/wiki/Observatoire_astronomique_de_Strasbourg
- [Astro Unistra, 2020] Astro Unistra : Recherche, <https://astro.unistra.fr/recherche/>
- [IVOA,2020] IVOA, http://ivoa.netastronomers/using_the_vo.html
- [Github,2020] Github : Diary lmichel/vo-grimoire, <https://github.com/lmichel/vo-grimoire/wiki/diary>
- [IVOA,2020] IVOA : Working Groups, <http://ivoa.net/members/index.html>
- [Wikipédia,2020] Wikipédia : JSON, https://fr.wikipedia.org/wiki/JavaScript_Object_Notation
- [Wikipédia,2020] Wikipédia : NoSQL, <https://fr.wikipedia.org/wiki/NoSQL>