Accompagnement et visualisations de la Science Ouverte (SoVisu)

David Reymond, Alaric Tabariès, Lena Bara

▶ **To cite this version:**

David Reymond, Alaric Tabariès, Lena Bara. Accompagnement et visualisations de la Science Ouverte (SoVisu). 2022. hal-04103412

HAL Id: hal-04103412
https://hal.science/hal-04103412

Submitted on 23 May 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

SoVisu

Version 1.0.2

David Reymond, Alaric Tabariès, Lena Bara

mai 22, 2023

Contenu

1 Description **1**
- 1.1 Utilisation personnelle . . . 1
- 1.2 Utilisation pilotage . . . 1

2 Généralités **3**
- 2.1 Technologie . . . 3
- 2.2 Licence . . . 3
- 2.3 Code source . . . 3
- 2.4 Documentation . . . 3

3 Guide de l'utilisateur **5**
- 3.1 Vérification des données . . . 5
  - 3.1.1 Identifiants . . . 5
  - 3.1.2 Notices . . . 6
  - 3.1.3 Expertise . . . 6
  - 3.1.4 Domaines . . . 7
  - 3.1.5 Descriptifs de recherche . . . 9
- 3.2 Synthèses . . . 9
  - 3.2.1 Tableau de bord . . . 9
  - 3.2.2 Références . . . 10
- 3.3 Façonner sa représentation lexicale . . . 10
  - 3.3.1 Lexiques extraits . . . 10
  - 3.3.2 Sélecteur de langues . . . 11
  - 3.3.3 Interface d'une langue . . . 12
  - 3.3.4 Mode opératoire suggéré . . . 13

4 Principes de fonctionnement **15**
- 4.1 Initialisation et connexion au SI . . . 15
- 4.2 Validations des Id et intégration des notices . . . 16
- 4.3 Enrichissements et synthèses . . . 17
- 4.4 Interfaces . . . 17

5 Installation **19**
- 5.1 Prérequis . . . 19
- 5.2 Configuration de l'environnement . . . 19
- 5.3 Mise en route . . . 21
  - 5.3.1 Initialisation des processus dans la partie Admin . . . 21

5.3.1.1 Spécificités de la mise en route de SoVisu pour le développement . . . 25
5.3.2 Initialisation des visualisations dans Kibana . . . 25
5.4 Mise en production . . . 25

6 Développement **27**
6.1 Application ElasticHal . . . 27
6.1.1 Sous-paquets . . . 27
6.1.1.1 Paquet elasticHal.libs . . . 27
6.1.2 Modules . . . 29
6.1.2.1 elasticHal.admin . . . 29
6.1.2.2 elasticHal.apps . . . 30
6.1.2.3 elasticHal.clean_researchers_from_es_base . . . 30
6.1.2.4 elasticHal.collect_from_HAL . . . 30
6.1.2.5 elasticHal.insert_entities . . . 30
6.1.2.6 elasticHal.models . . . 31
6.1.2.7 elasticHal.views . . . 33
6.2 Application SoVisuhal . . . 33
6.2.1 Sous-paquets . . . 33
6.2.1.1 Paquet sovisuhal.libs . . . 33
6.2.2 Modules . . . 35
6.2.2.1 sovisuhal.forms . . . 35
6.2.2.2 sovisuhal.views . . . 35
6.2.2.3 sovisuhal.viewsActions . . . 35

Index des modules Python **37**

Index **39**

CHAPITRE 1

Description

SoVisu est un dispositif flexible et ouvert créé pour accompagner la communauté d'un établissement à la Science Ouverte (SO) en lui fournissant les instruments guides de son pilotage, tout en permettant de cartographier l'expertise des chercheurs, faciliter l'autoarchivage et apprécier la qualité de l'indexation. Ce dispositif prend source sur HAL pour s'inscrire dans l'écosystème de la SO côté usager en donnant la possibilité aux chercheurs d'aligner leurs identifiants, de qualifier/vérifier et compléter leurs données bibliographiques, leurs domaines et leurs mot-clés « experts ».

1.1 Utilisation personnelle

Pour un chercheur, SoVisu accompagne la vérification et le recensement des Id (idRef, OrcId, idHal), récupère sur [HAL](https://hal.science) les notices bibliographiques associées à sa production pour produire une synthèse et des cartographies lexicales (comparable à celles produites par les index). L'interface laisse la possibilité d'exclure de son profil certaines productions. SoVisu produit alors des tableaux de bord sous forme de visualisations dynamiques et interactives à partir des travaux (indicateurs bibliométriques interactifs, extractions lexicales), des liens directs vers HAL pour chacune des notices afin de corriger/compléter certaines données et des synthèses : chacun peut apprécier la qualité de sa représentation sur les index et éventuellement corriger.

1.2 Utilisation pilotage

Pour un laboratoire, ou groupe de, côté gouvernance les mêmes principes et fonctions précédentes par agrégations, extractions de la production au format HCERES (4 premiers volets avec calculs réalisés : interrogation unpaywal, appartenance d'un doctorant dans les co-publiants notamment, attribution à des axes/équipes spécifiques). Les agrégations permettent alors d'opérer sur des données bibliographiques validées par les chercheurs.

CHAPITRE 2

Généralités

2.1 Technologie

SoVisu est une application web Python/Django se basant sur les données disponibles sur https://hal.science et constitué de deux applications distinctes :

- **elasticHal** : chargée de récupérer les données des laboratoires et chercheurs sur HAL, puis de les enrichir, afin de les incorporer à une base de données Elastic. Accessible via l'interface administration de Django,
- **sovisuhal** : chargée d'afficher les données aux utilisateurs et se basant sur un environnement Django. Présente les données HAL récupérées puis stockées dans Elastic.

2.2 Licence

EUPL_v1.2_fr.pdf

2.3 Code source

Sur Git-hub : https://github.com/Patent2net/SoVisu

2.4 Documentation

http://sovisu.readthedocs.io/

CHAPITRE 3

Guide de l'utilisateur

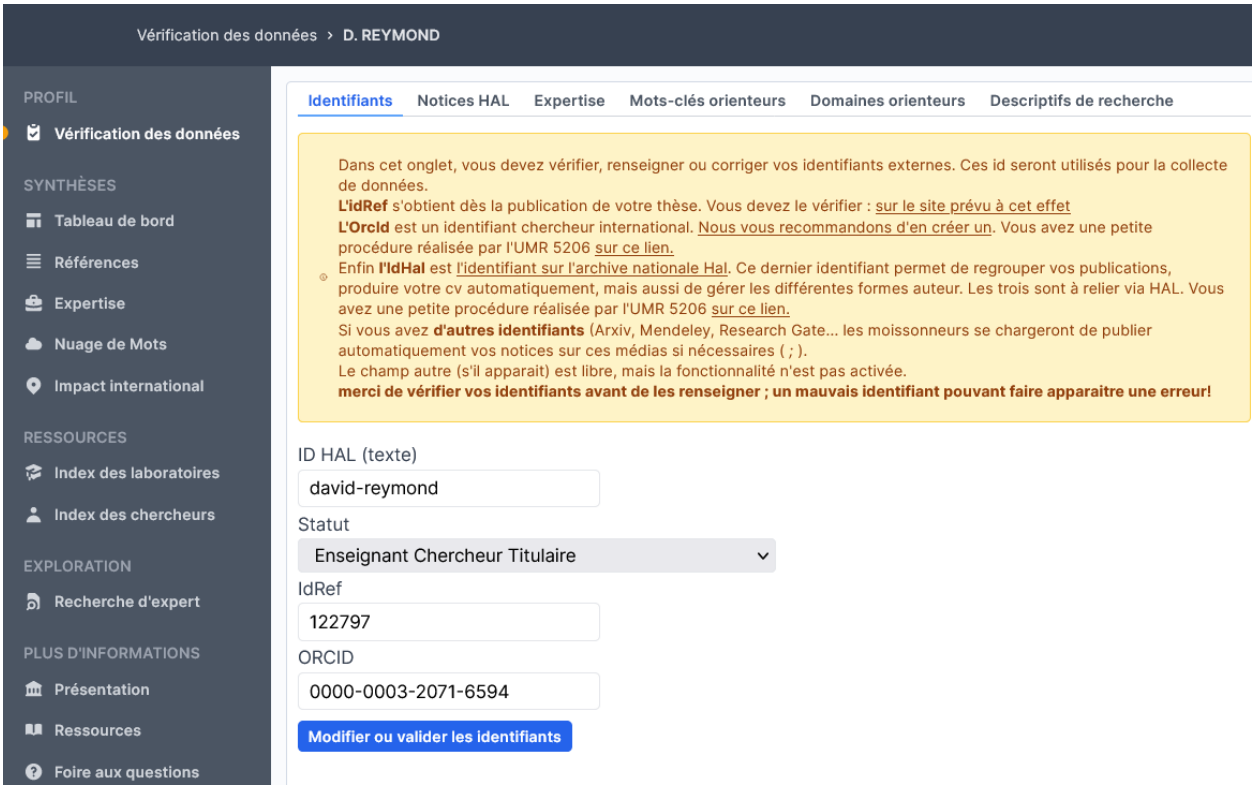
SoVisu est fait pour accompagner l'auto-archivage des travaux de publication. Il permet de situer l'état de ses publications (sont elles présentes, ouvertes, les fichiers complets peuvent ils être déposés ?). SoVisu construit de surcroît leur représentation lexicales dans les index (quels sont les mots qui permettent de retrouver mes productions ?) en anglais et en français. Ceci s'effectue en quelques clics. Les informations renseignées dans votre profil vous aideront par la suite à compléter et étendre cette représentation lexicale en vous suggérant de nouveaux mots-clés. Après authentification, SoVisu vous dirige vers la vérification des données de votre profil.

3.1 Vérification des données

3.1.1 Identifiants

Partie incontournable pour entrer du bon pied dans la science ouverte. Vous rapportez vos IdRefs, IdHal , OrcId (à créer le cas échéant).

Avertissement : Si les identifiants rapportés sont faux rien de ce qui suit ne fonctionnera. Si l'idhal vient d'être créé, terminez demain ! !



Seul l'idhal est utilisé pour télécharger les notices. .. note :

```
Le formulaire peut aisément intégrer d'autres Ids (scopusId, researcherId...)
```

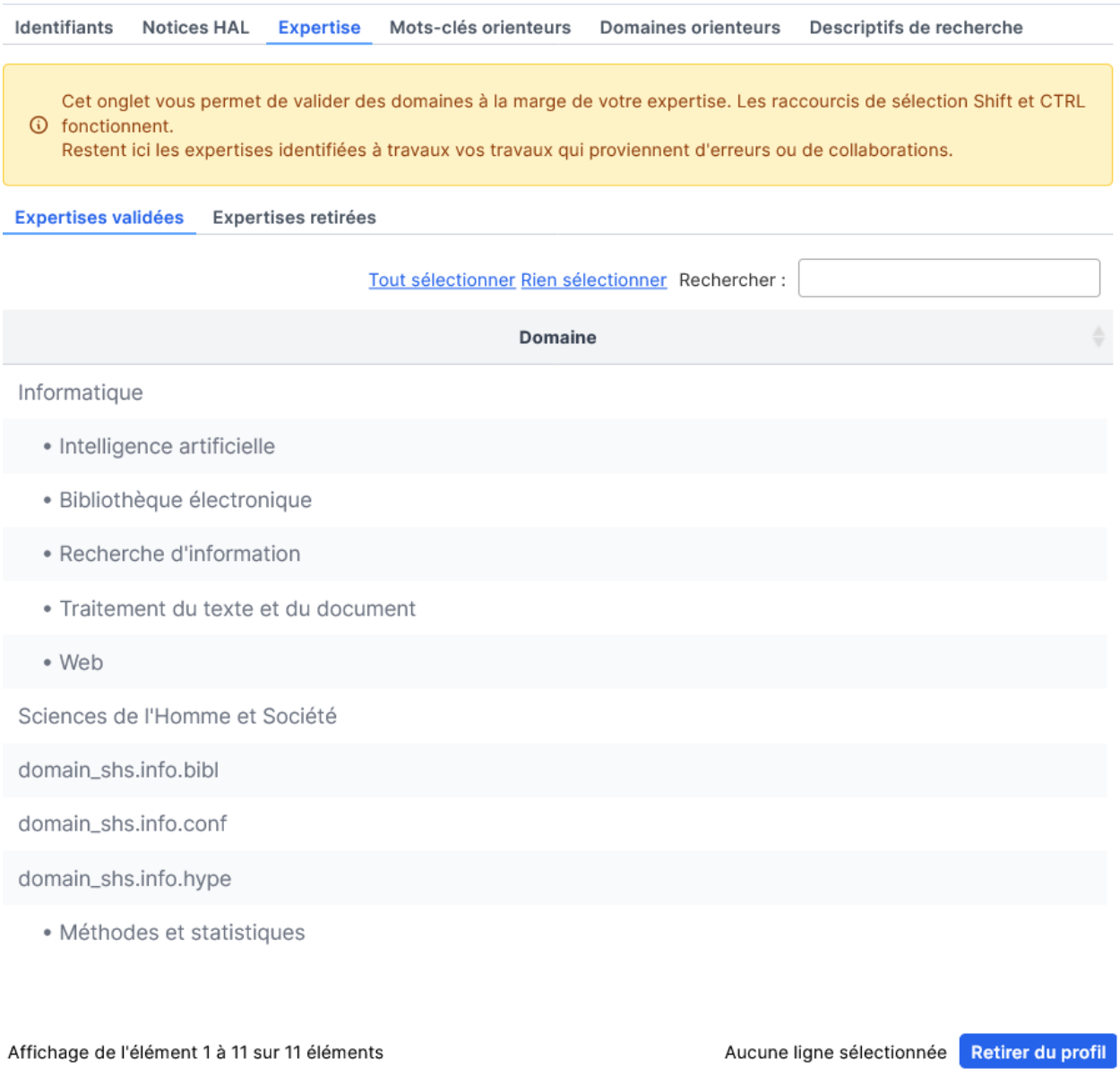
3.1.2 Notices

Vous affiche la liste de vos publications et votre position d'auteur (que vous pouvez modifier à l'aide du bouton « Mise à jour de l'autorat », si mal calculée). Vous pouvez éventuellement retirer de votre profil celles que vous considérez comme marginales avec l'autre bouton. Celle-ci basculera dans le volet des notices retirées. Au plus les productions sont en phase avec votre expertise au mieux le lexique issus de ces dernières ciblera votre profil dans le moteur de recherches.

Astuce : Si vous avez récemment modifié, le bouton « Collecter (ou mettre à jour) de nouvelles références » permet de re-télécharger les notices associées à votre idhal.

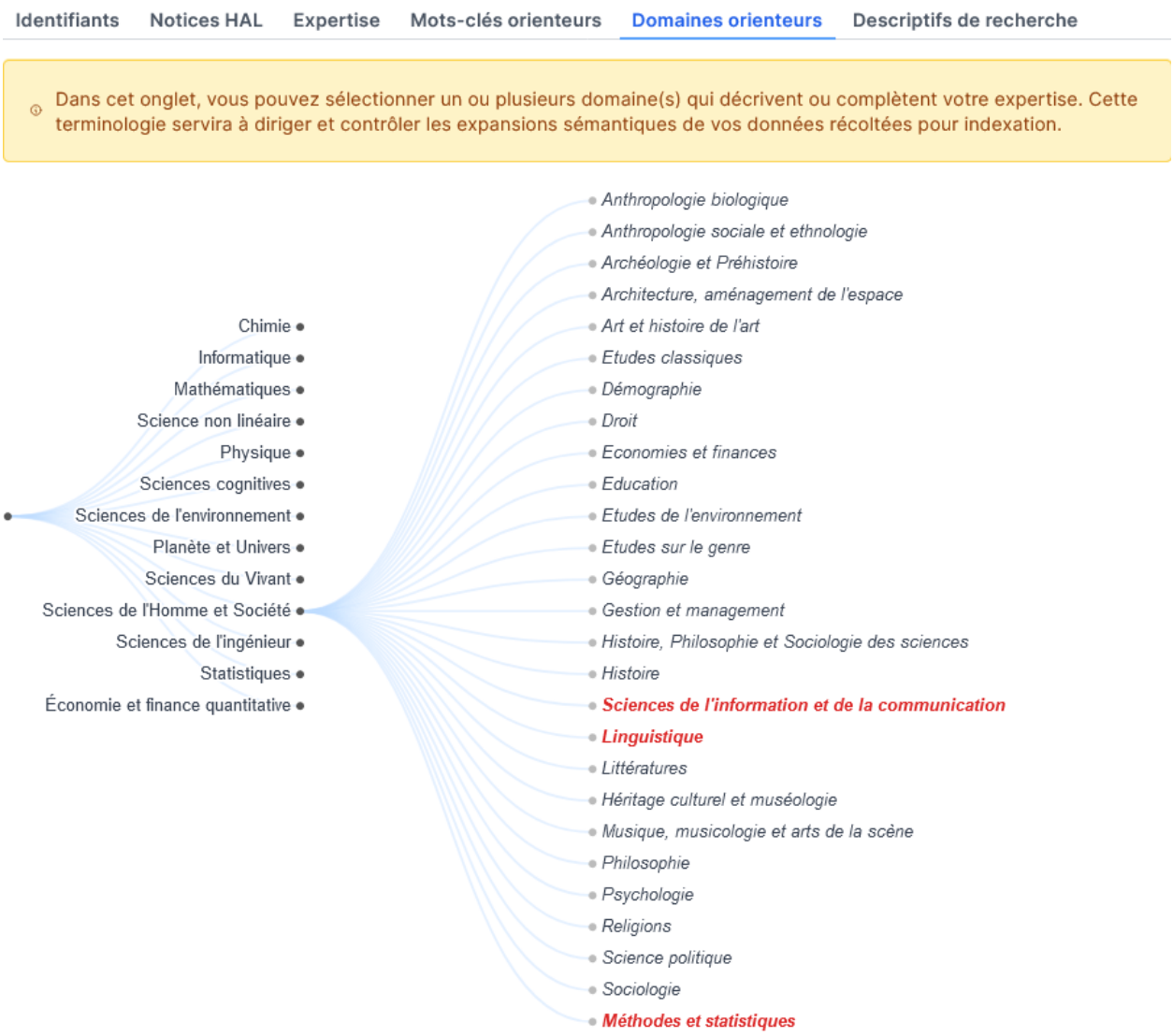
3.1.3 Expertise

Cet onglet vous permet de sélectionner les domaines scientifiques issus par calcul de votre production. Vous pouvez en retirer de votre profil pour ne conserver que celles qui représentent le mieux vos travaux. Au mieux ces éléments cernent votre expertise au plus les enrichissements seront pertinents.



3.1.4 Domaines

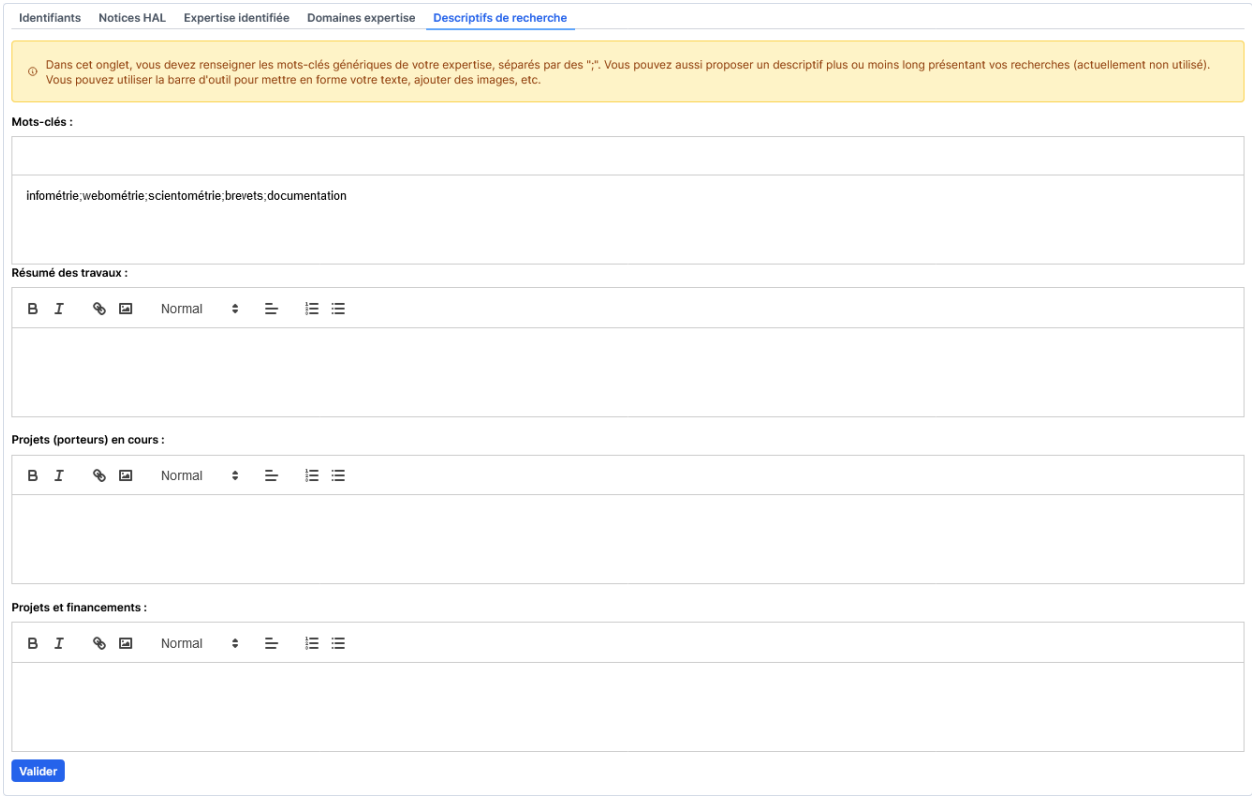
Même principe que précédemment. L'interface vous permet de sélectionner dans la nomenclature AureHAL le ou les domaines de vos travaux. Encore une fois, au mieux ces domaines décrivent votre expertise au plus les enrichissements seront pertinents.



Astuce : Le menu déroulant vous permet de parcourir l'arborescence des domaines disciplinaires pour y sélectionner les noeuds les plus adéquats. N'oubliez pas de valider.

3.1.5 Descriptifs de recherche

Dans cet onglet vous devez renseigner les mots-clés génériques de votre expertise.



Les autres champs permettent de spécifier vos projets, travaux, etc.

Note : Ces derniers sont non utilisés dans cette version.

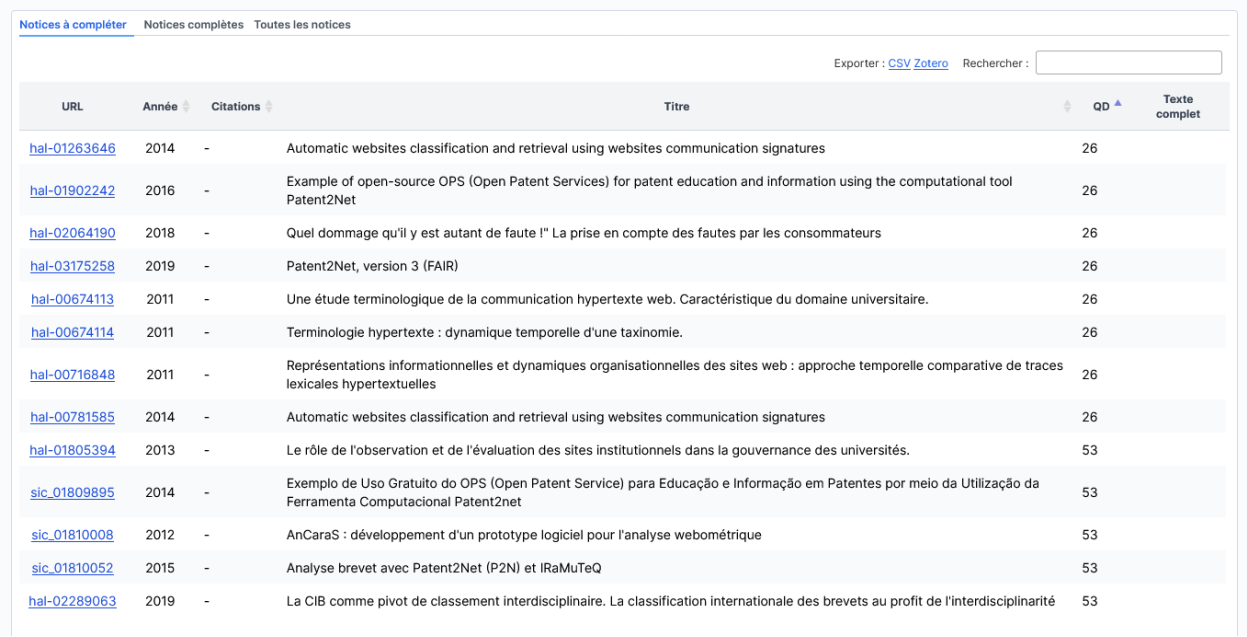
3.2 Synthèses

3.2.1 Tableau de bord

Différentes cartographies de vos publications. Le commutateur validé permet d'inclure ou pas vos productions que vous auriez invalidées. Le baromètre de vos publications vous donne la tendance d'ouverture fonctions de vos choix d'éditeurs. Si la courbe indécidable diminue, que la courbe ouvert augmente, continuez vous êtes sur la bonne voie !

3.2.2 Références

Permet d'afficher vos publications en trois rubriques : les notices qui mériteraient d'être complétées (champs manquants), celles qui sont complètes et l'ensemble de vos publications (hors celles retirées volontairement). Le dispositif souligne également si la publication complète peut être archivée avec le texte complet (en fichier pdf ou autre) : notez qu'il convient toujours de demander aux co-auteurs bien sûr, mais les conditions juridiques par rapport aux éditeurs ont, dans ce cas, été levées. Un score de 0 à 100 attribue une note à ces métadonnées, clés du référencement et de l'indexation pour vous inciter à améliorer ce point.


Notices à compléter | Notices complètes | Toutes les notices

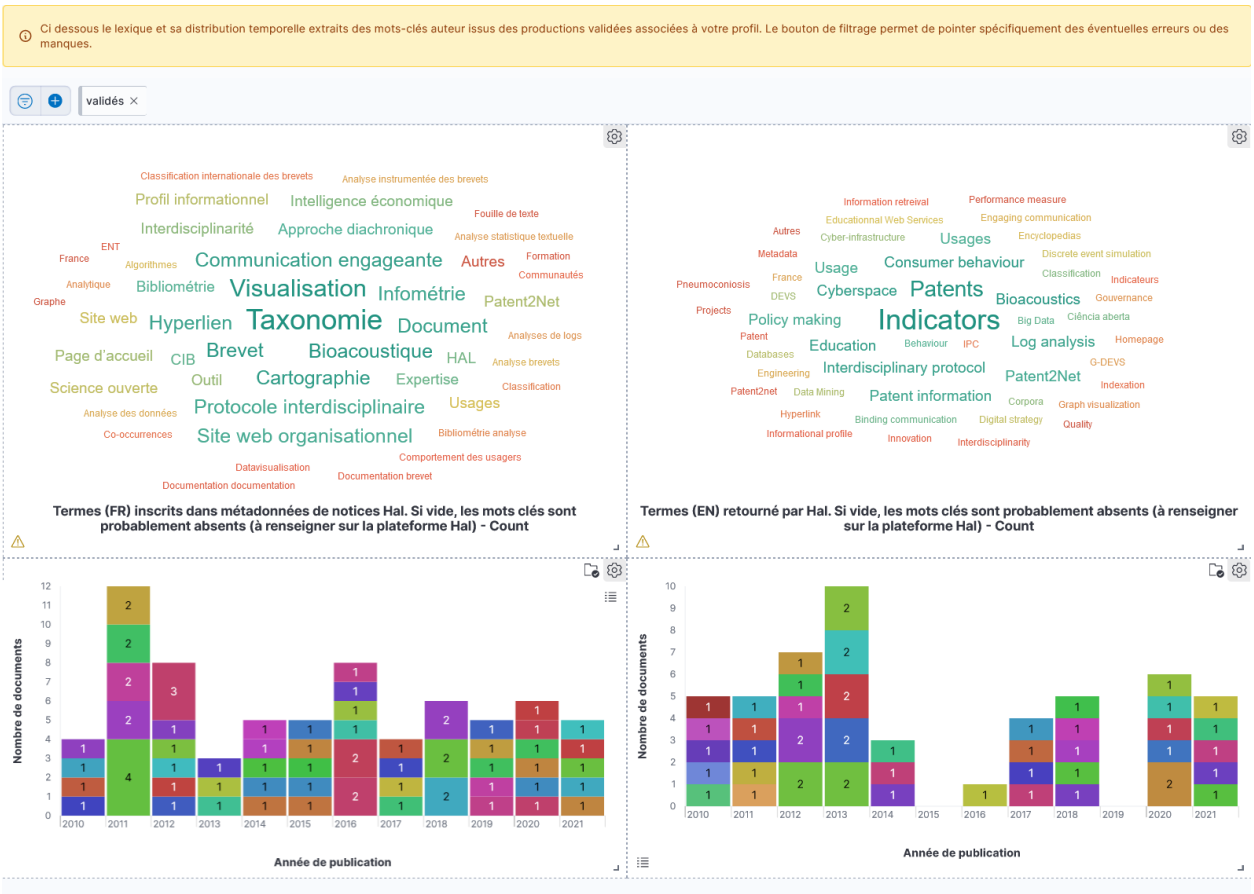
Exporter : CSV Zotero Rechercher :

URL	Année	Citations	Titre	QD	Texte complet
hal-01263646	2014	-	Automatic websites classification and retrieval using websites communication signatures	26	
hal-01902242	2016	-	Example of open-source OPS (Open Patent Services) for patent education and information using the computational tool Patent2Net	26	
hal-02064190	2018	-	Quel dommage qu'il y est autant de faute !" La prise en compte des fautes par les consommateurs	26	
hal-03175258	2019	-	Patent2Net, version 3 (FAIR)	26	
hal-00674113	2011	-	Une étude terminologique de la communication hypertexte web. Caractéristique du domaine universitaire.	26	
hal-00674114	2011	-	Terminologie hypertexte : dynamique temporelle d'une taxinomie.	26	
hal-00716848	2011	-	Représentations informationnelles et dynamiques organisationnelles des sites web : approche temporelle comparative de traces lexicales hypertextuelles	26	
hal-00781585	2014	-	Automatic websites classification and retrieval using websites communication signatures	26	
hal-01805394	2013	-	Le rôle de l'observation et de l'évaluation des sites institutionnels dans la gouvernance des universités.	53	
sic_01809895	2014	-	Exemplo de Uso Gratuito do OPS (Open Patent Service) para Educação e Informação em Patentes por meio da Utilização da Ferramenta Computacional Patent2net	53	
sic_01810008	2012	-	AnCaraS : développement d'un prototype logiciel pour l'analyse webométrique	53	
sic_01810052	2015	-	Analyse brevet avec Patent2Net (P2N) et IRaMuTeQ	53	
hal-02289063	2019	-	La CIB comme pivot de classement interdisciplinaire. La classification internationale des brevets au profit de l'interdisciplinarité	53	

3.3 Façonner sa représentation lexicale

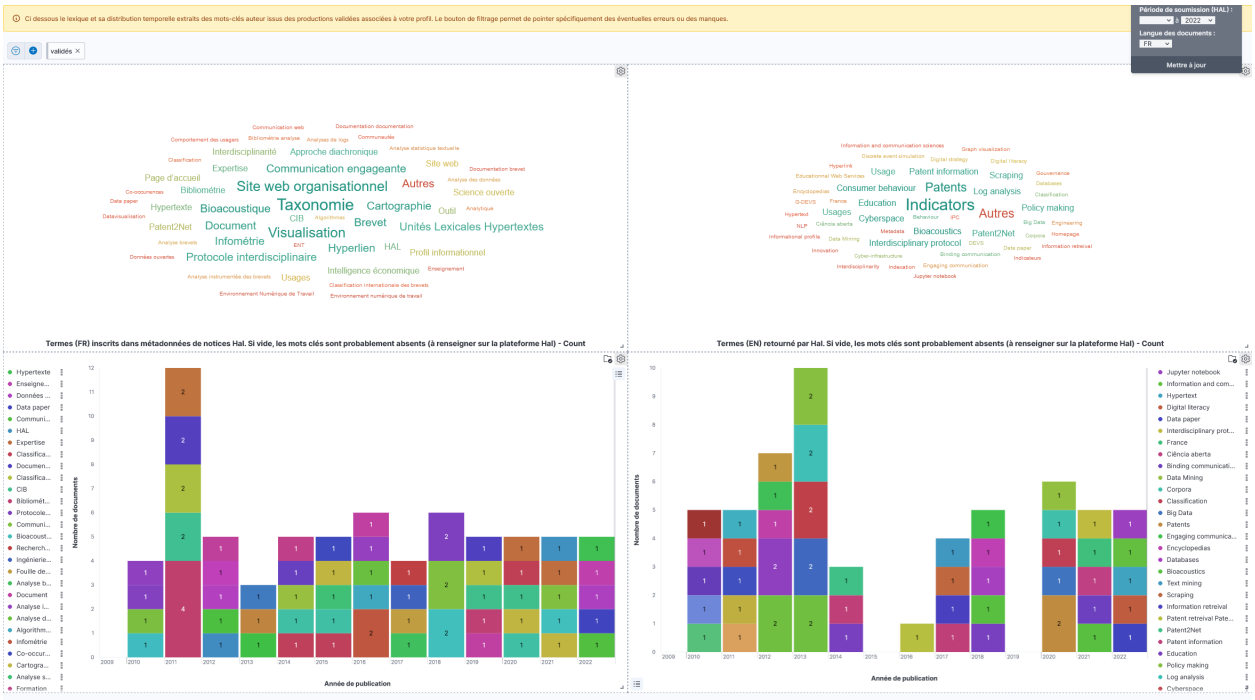
3.3.1 Lexiques extraits

SoVisu vous affiche la représentation lexicale de vos travaux en deux langues à partir des mots-clés des notices et sa distribution historique.



3.3.2 Sélecteur de langues

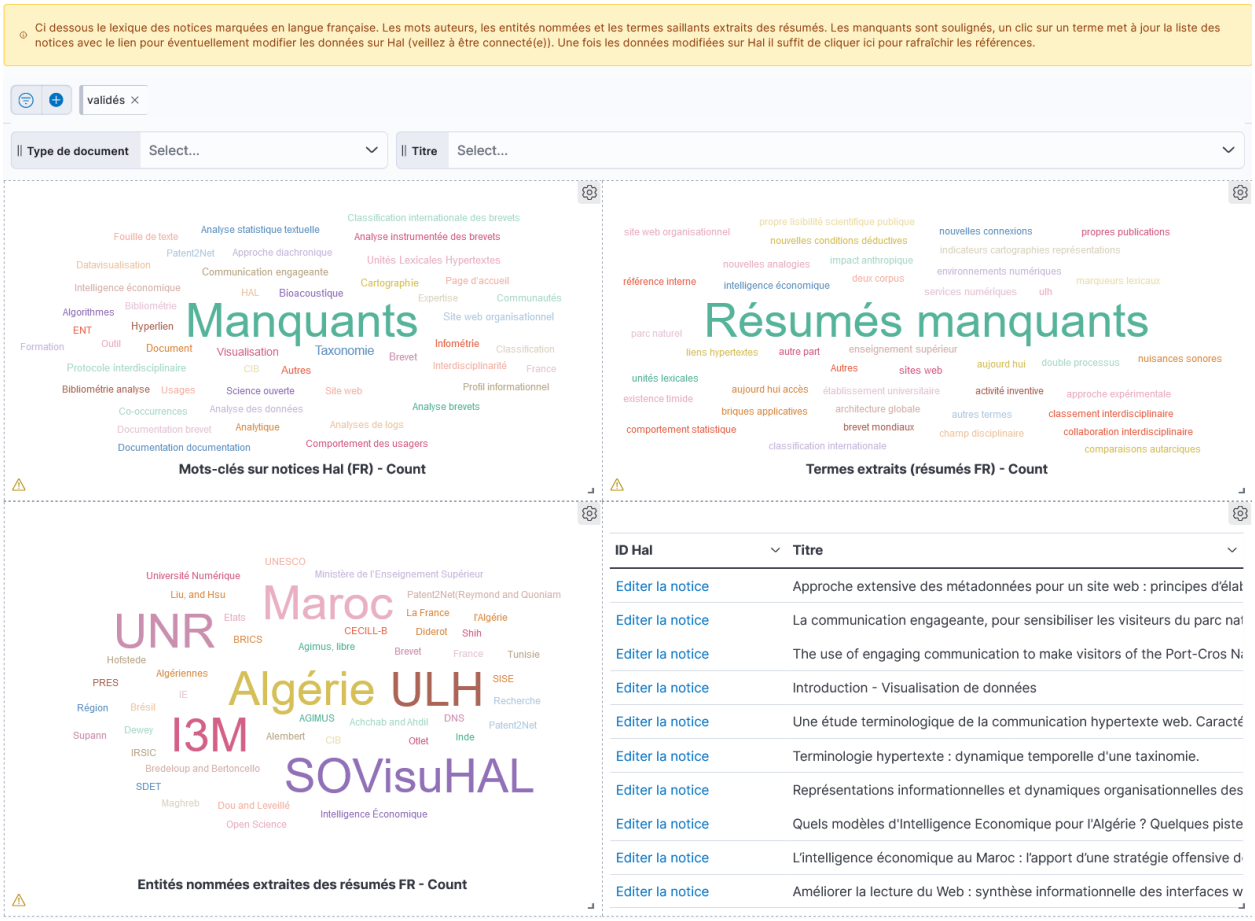
Le sélecteur des langues vous permet de filtrer sur une langue en particulier. Sélectionnez et mettez à jour.



3.3.3 Interface d'une langue

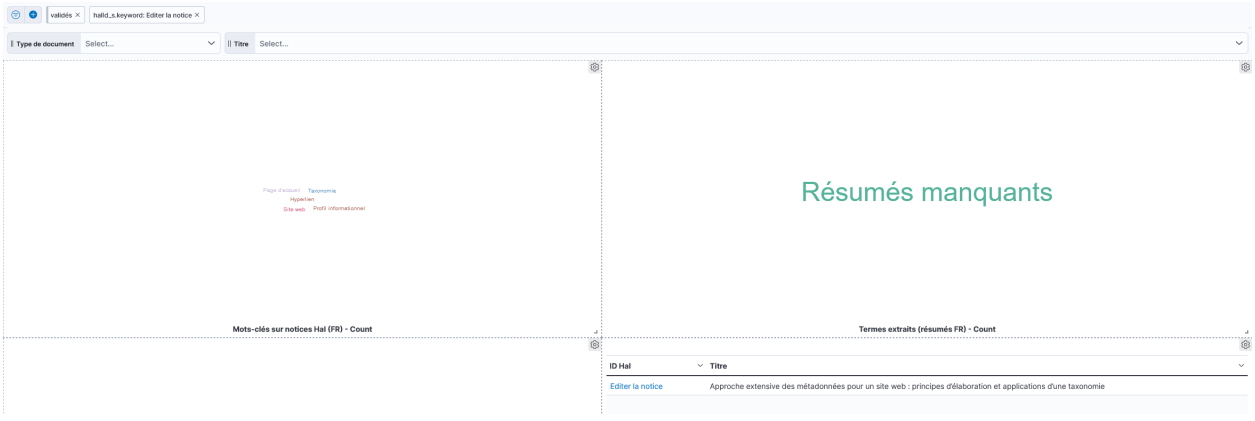
Vous avez alors accès à une représentation plus complète dans la langue choisie :

1. les mots clés auteurs, ceux qui sont présents sur les notices. Les notices sans mots-clés sont marquées « manquants »,
2. à droite des précédents, les mots extraits automatiquement des résumés à partir du service de l'inist term-extraction
3. En bas à gauche de l'interface, les entités nommées identifiées (dates, noms propres, lieux) par traitement des résumés à l'aide d'outils de traitement automatique du langage (Spacy)
4. Vos références avec un lien pour éditer la notice directement sur HAL (il faudra vous authentifier la première fois).



3.3.4 Mode opératoire suggéré

Grâce à cette interface, vous pourrez modifier la représentation lexicale de vos travaux, appuyer certains termes (par synonymie), en rajouter d'autres. A terme, SoVisu vous suggèrera d'apposer d'autres termes. En sélectionnant les manquants vous filtrez les notices qui n'auraient pas de mots-clés ou celles sans résumés (celles ci sont moins *lisibles*). En sélectionnant des termes qui vous paraissent étranges, vous ciblez les notices dont sont extraits ces termes. Chaque fois, la sélection provoque une mise à jour du filtre de requête en haut à gauche de l'interface que vous pouvez supprimer (la croix) pour revenir en arrière.



Astuce : N'hésitez pas non plus à déposer le texte complet (pdf), mais n'oubliez pas de renseigner mots-clés et résumés utilisés par les index pour appuyer votre référencement.

CHAPITRE 4

Principes de fonctionnement

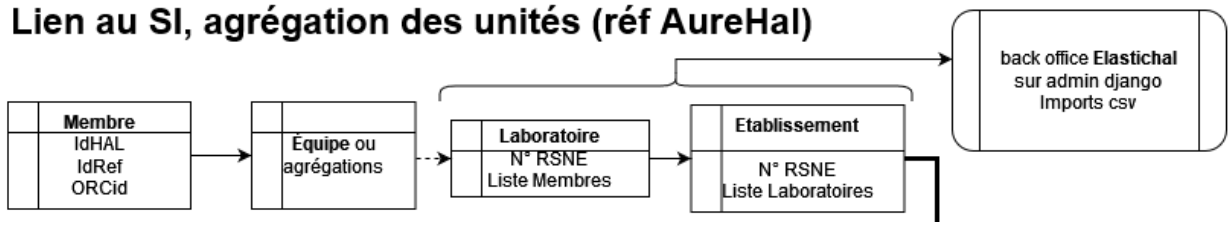
SoVisu est un logiciel open source, opérationnel et installable dans un établissement universitaire. SoVisu est issue d'un projet initié en 2019 à l'origine pour cartographier les compétences des membres de l'Université de Toulon. SoViSu est destiné à l'ensemble de la communauté universitaire pour faciliter l'autoarchivage et la gestion individuelle de sa production sur HAL. SoVisu affecte un score de qualité calculé sur la quantité des métadonnées des notices, et extrait automatiquement un lexique propre à chaque document pour l'agréger en un profil informationnel individuel ou collectif. Chacun peut ainsi apprécier la représentation de ses travaux sur les index, et l'adapter pour parfaire sa lisibilité en plus de sa visibilité.

4.1 Initialisation et connexion au SI

L'initialisation comprend l'intégration des données chercheurs (issues de ldap) et laboratoires (labHalid) rattachés à une structure (identifiant SIREN). Le laboratoire est décrit par son intitulé, son acronyme, les membres le composant et une liste d'e-mails. Chaque élément est une liste ce qui permet d'associer des dénominations multiples et suivre de fait les évolutions historiques. Les membres sont eux-mêmes décrits par leur nom et prénom.

Avertissement : Les données ldap sont manuellement introduites dans SoVisu mais avec l'authentification CAS il devrait être possible de les puiser directement lors de la première authentification.

Avertissement : le ldapid est utilisé pour désigner le profil utilisateur. Il est prévu de basculer sur une authentification ORCId. A terme, l'utilisation du ORCId en Id interne à SoVisu devrait permettre de s'extraire du SI d'un établissement. Toutefois, les données d'annuaires (chercheur, doctorant, etc.) sont utiles pour renseigner les exports HCERES.



Cette étape construit dans Elastic les index suivant :

- **SIREN-structures** # données de structure
- **SIREN-labHalid-laboratories** # données de laboratoire
- **SIREN-labHalid-laboratories-documents** # collection de laboratoire
- **SIREN-labHalid-researchers** # chercheurs du laboratoire
- **SIREN-labHalid-researchers-ldapid-documents** # documents du ldapid de chercheur

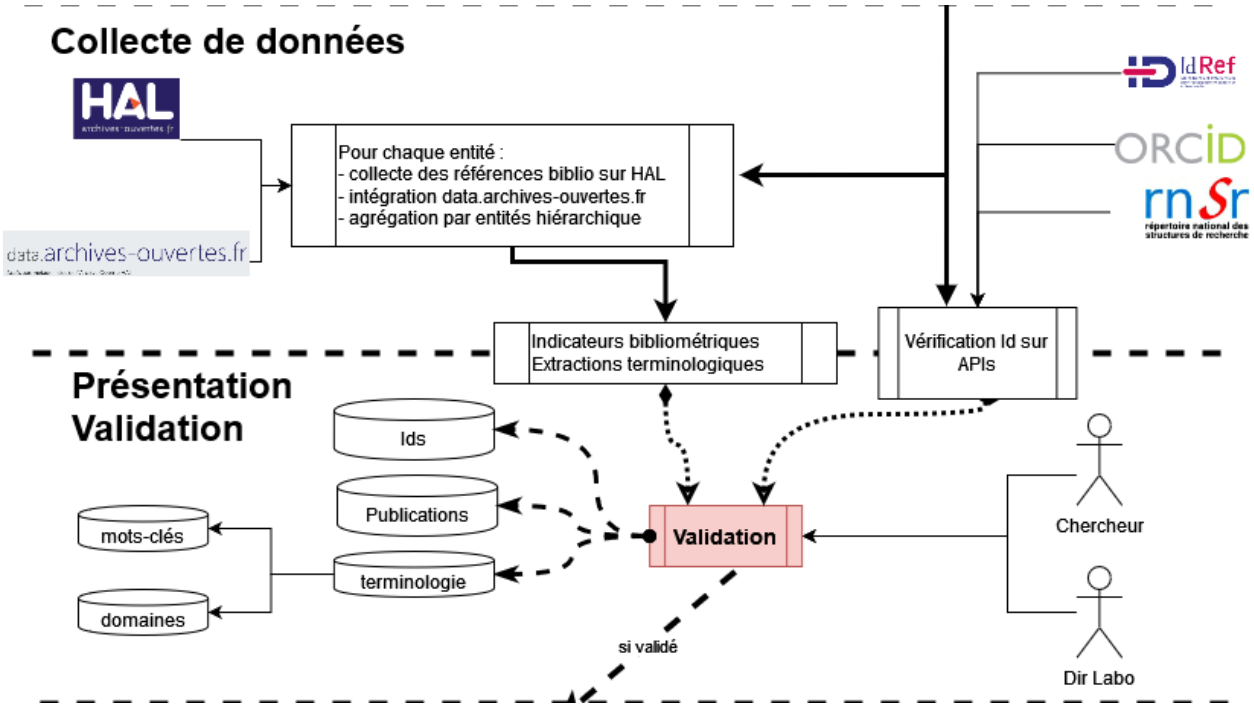
Astuce : Le labHalid Non-labo est créé pour les chercheurs non rattachés.

4.2 Validations des Id et intégration des notices

Lors de l'authentification le chercheur doit renseigner ses identifiants de chercheur, les créer le cas échéant :

- idRef : obligatoire mais non utilisé. Mis ici pour informer de son existence et pour que chacun vérifie (problèmes d'homonymie, de changement de noms, ou si la thèse n'a pas été soutenu en France, cet Id n'existe pas)
- idHal : primordial, si faux rien ne marche.
- ORCId : optionnel (non utilisé). Mis ici pour suggérer aux chercheurs de s'en créer un. A terme devrait être l'id principal de SoVisu pour les chercheurs.

Cette étape de validation permet à SoVisu de collecter les notices associées à l'idHal sur Hal. Celles-ci sont alors indexées dans son profil : *SIREN-labHalid-researchers-ldapid-documents*.



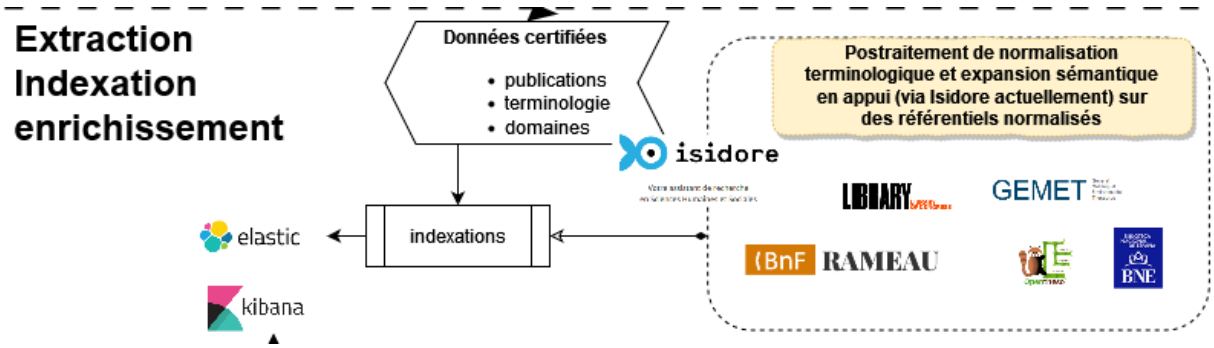
SoVisu calcule le score de *qualité des données* des notices sur la base du nombre de métadonnées renseignées.

Les mots-clés auteurs (renseignés par l'éditeur de la notice) sont complétés par un lexique issus de l'extraction automatique sur les titres et résumés (bibliothèque Spacy) : entités nommées et champs nominaux complètent ainsi la représentation des productions.

4.3 Enrichissements et synthèses

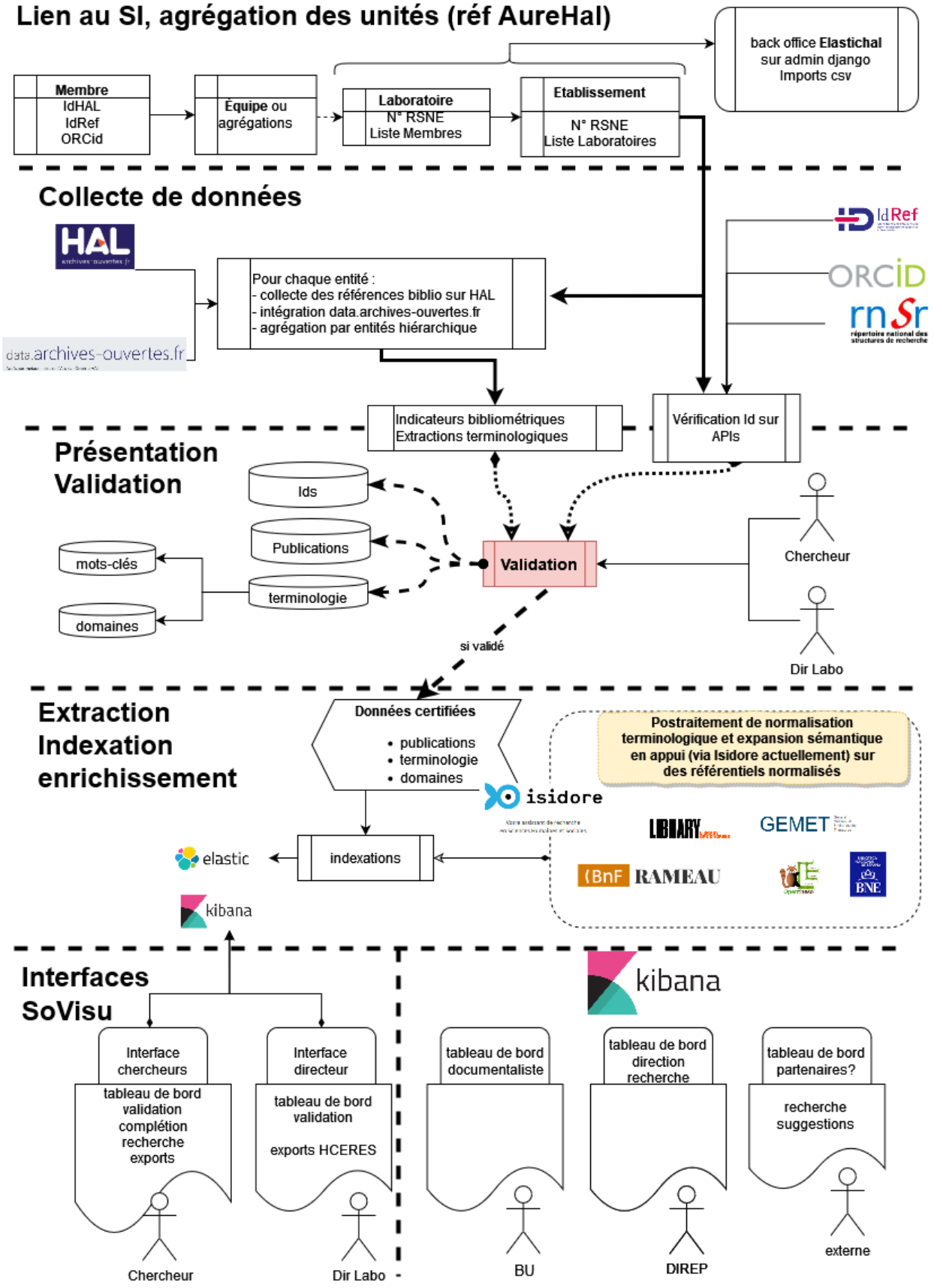
Avertissement : Les tests d'enrichissement sont encore en cours.

Le principe est de s'appuyer sur les données validées par les chercheurs (mots clés, domaines et vocabulaire issu de ses productions validées) pour enrichir la représentation lexicale (synonymes, hyperonymes, etc.). Isidore a fait parti des premiers tests plutôt concluant mais des travaux d'affinage sont nécessaires.



4.4 Interfaces

Constituées principalement pour intégrer des tableaux de bord Kibana, les formulaires de validation et d'interaction avec les usagers. Actuellement seuls deux profils sont ciblés : les chercheurs et les directeurs de laboratoire. Un moteur d'exploration pour fournir une vue au travers de cartographies interactives des contenus vise aussi les usagers « externe ». Les autres usagers potentiels (documentaliste ou direction de la recherche) disposent de tableaux de bords spécifiques dans Kibana.


Lien au SI, agrégation des unités (réf AureHal)
Membre
IdHAL
IdRef
ORCid
Équipe ou agrégations
Laboratoire
N° RSNE
Liste Membres
Etablissement
N° RSNE
Liste Laboratoires
back office Elastichal
sur admin django
Imports csv
Collecte de données
HAL
data.archives-ouvertes.fr
Pour chaque entité :
- collecte des références biblio sur HAL
- intégration data.archives-ouvertes.fr
- agrégation par entités hiérarchique
IdRef
ORCID
rnsr
Indicateurs bibliométriques
Extractions terminologiques
Vérification Id sur APIs
Présentation
Validation
Ids
Publications
mots-clés
terminologie
domaines
Validation
Chercheur
Dir Labo
si validé
Extraction
Indexation
enrichissement
Données certifiées
• publications
• terminologie
• domaines
isidore
Postraitement de normalisation terminologique et expansion sémantique en appui (via Isidore actuellement) sur des référentiels normalisés
LIBRARY
GEMET
BnF RAMEAU
BNE
elastic
indexations
kibana
Interfaces
SoVisu
kibana
Interface chercheurs
tableau de bord validation complétion recherche exports
Interface directeur
tableau de bord validation
exports HCERES
tableau de bord documentaliste
tableau de bord direction recherche
tableau de bord partenaires?
recherche suggestions
Chercheur
Dir Labo
BU
DIREP
externe

CHAPITRE 5

Installation

SoVisu est un projet s'appuyant sur le framework Django et qui est constitué de 2 applications :

— elasticHal, qui est chargée de récupérer les informations les structures et les laboratoires.

— sovisuhal qui collecte les identifiants chercheurs, les métadonnées de leurs publications sur Hal pour présenter l

Les données récupérées sont stockées sur un moteur de recherche Elasticsearch, couplé à l'interface utilisateur Kibana qui est utilisée pour créer les tableaux de bord proposés aux utilisateurs.

5.1 Prérequis

Afin d'initialiser le projet, il est nécessaire de disposer d'une machine équipée de Python (version 3.9=~), de Docker, ainsi que de Git.

Il n'est pas nécessaire de créer des index pour Elasticsearch, ces derniers sont créés par elasticHal lors de la collecte.

Le code de SoVisu est accessible sur https://github.com/Patent2net/SoVisu . Ce qui suit suppose que vous ayez installé un environnement virtuel python noté `venv` par la suite avec les librairies du fichier requirements.txt (point 2 de la procédure).

5.2 Configuration de l'environnement

1. Clonez le répertoire

```
(.venv) $ git clone https://github.com/Patent2net/SoVisu/
```

2. installez les requirements dans le projet SoVisu

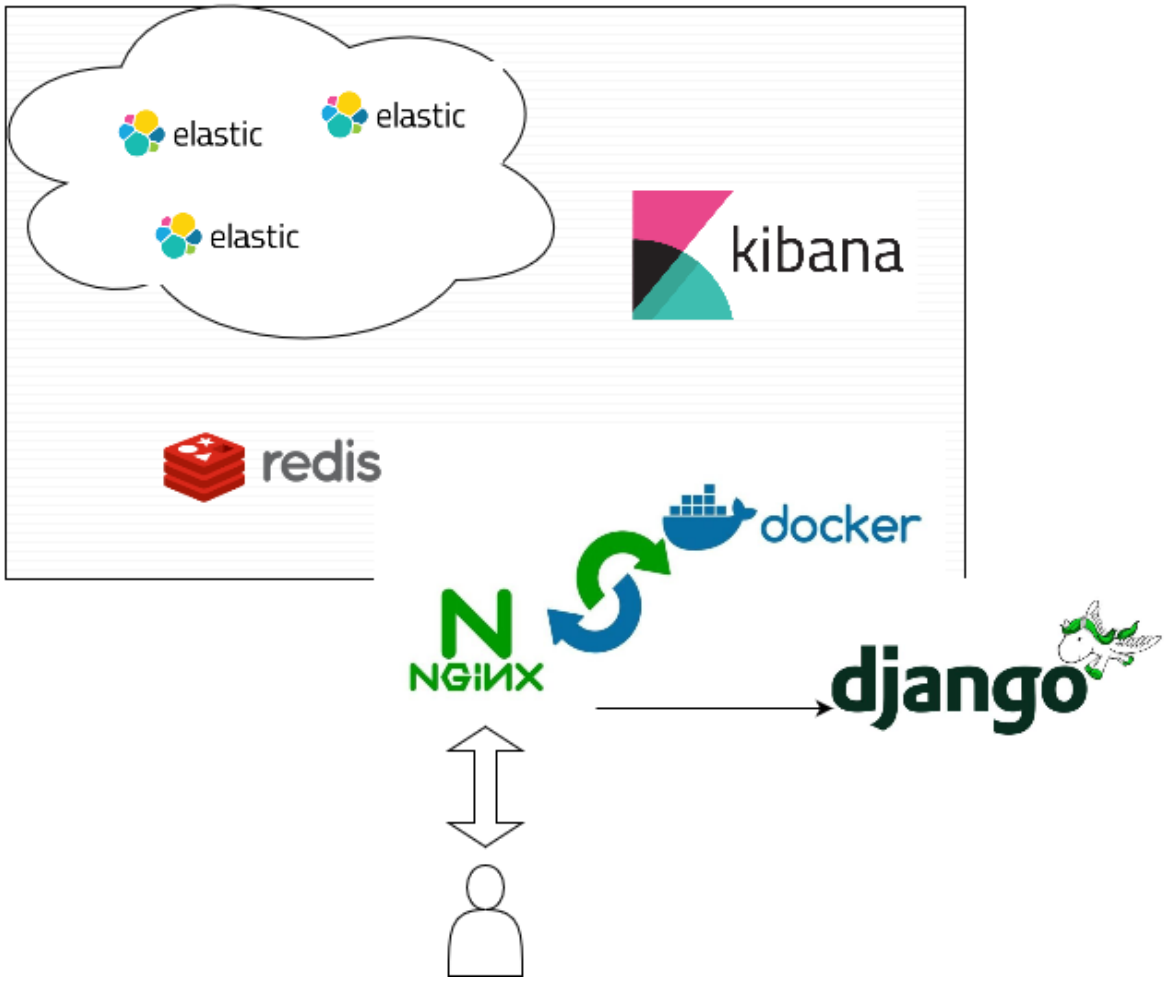
```
(.venv) $ pip install -r requirements.txt
```

3. Dans doc/VisuStack, créer le dossier `volume` ainsi que les sous-dossiers `backup1`, `data1`, `data2`, `data3` et `MB-data`.
4. Exécutez la commande suivante pour installer l'environnement serveur utilisé par le projet :

```
(.venv) $ docker-compose doc/VisuStack/docker-compose.yml
```

Avertissement : Cette commande peut prendre du temps à s'exécuter, car elle télécharge les images Docker de Kibana, ElasticSearch et Redis qui sont nécessaires au fonctionnement du projet.

SoVisu s'appuie sur une architecture telle la figure ci-dessous : Docker héberge tous les services supports excepté le serveur Django. Nginx sert de frontal de sécurité à tous les services y compris SoVisu.



5. Initialisez les migrations de SoVisu :

5.1 Adapter les variables d'environnement 5.2 Migration .. code-block : : console

(.venv) $ python manage.py migrate

6. Désignez le serveur CAS de l'institution :

```
(.venv) $ python manage.py add_institution "nom de l'institution" https://cas.exemple.fr
```

7. Créez un profil administrateur :

```
(.venv) $ python manage.py createsuperuser
```

Astuce : il vous sera demandé de rentrer un identifiant, une adresse mail et un mot de passe. Bien que l'identifiant et le mot de passe soient obligatoires, le champ adresse mail est optionnel.

5.3 Mise en route

Avertissement : Avant toute mise en route de SoVisu, vérifiez que l'instance elastic avec lequel le projet interagit est active. Si ce n'est pas le cas, SoVisu renverra un message d'erreur au lieu de s'initialiser normalement.

5.3.1 Initialisation des processus dans la partie Admin

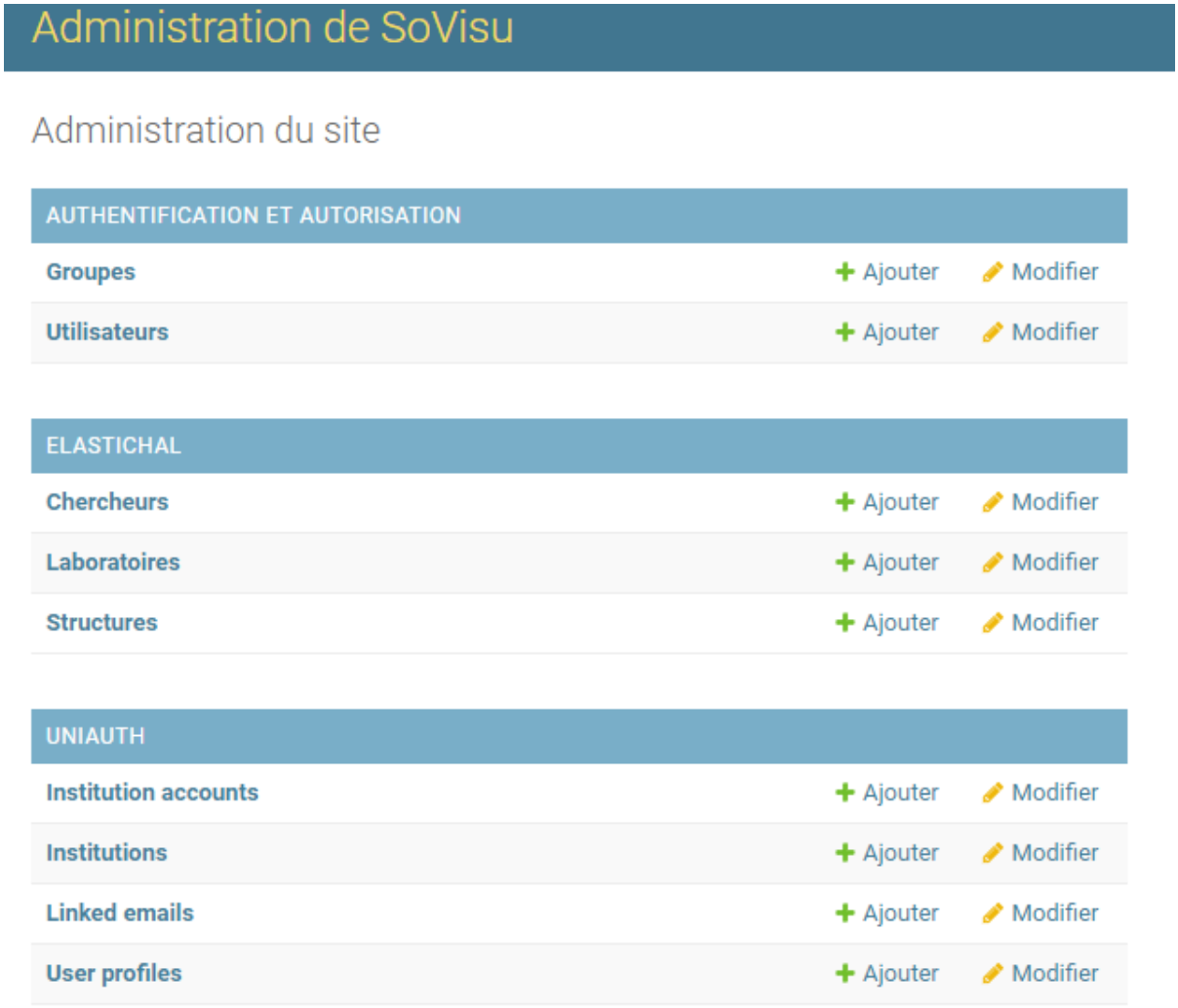
1. Lancez SoVisu :

```
(.venv) $ python manage.py runserver
```

2. Visitez l'adresse suivante : http://127.0.0.1:8000/admin/
3. Renseignez les identifiants administrateur créés précédemment.

En arrivant dans l'interface d'administration, vous pouvez voir les différentes Applications de SoVisu :

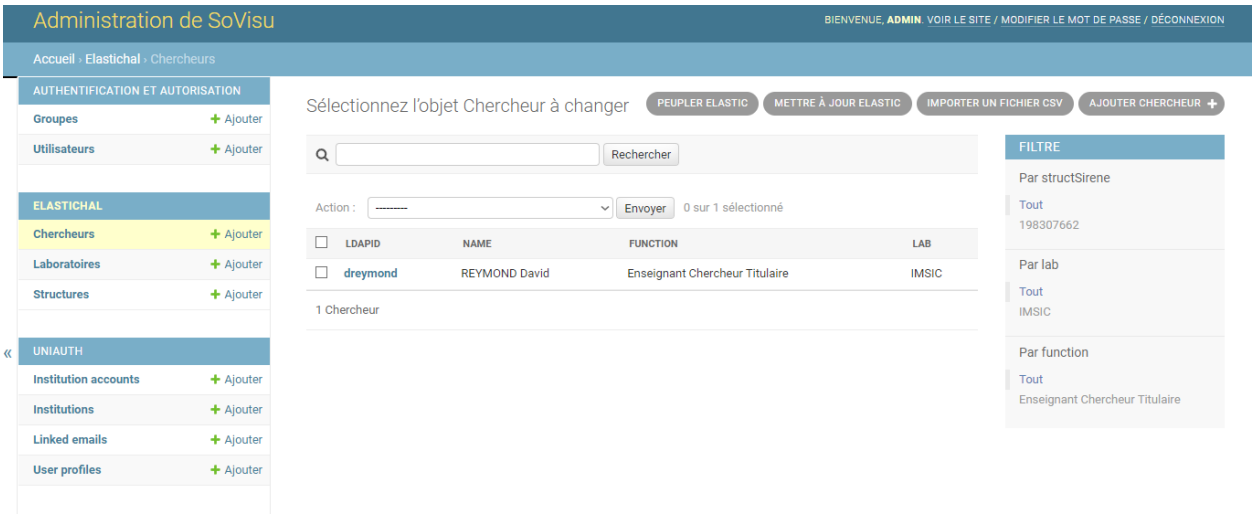
— « Authentification et autorisation », qui est présente par défaut dans Django. Elle permet de gérer les utilisateurs se connectant avec des identifiants créés depuis l'interface administrateur ou avec des commandes depuis manage.py.
— « Elastichal », qui permet d'initialiser la base de données Elasticsearch.
— « Uniauth », qui permet dans le cas d'une installation dans une institution de gérer la connection à partir des identifiants CAS.



Dans la partie « Elastichal », trois modèles sont disponibles :

- « Chercheurs », qui permet de stocker les informations de base concernant les objets chercheurs.
- « Laboratoires », qui permet de stocker les informations de base concernant les objets laboratoires.
- « Structures », qui permet de stocker les informations de base concernant les objets structures institutionnelles dont dépendent les laboratoires.

4. Cliquez sur « Chercheurs » dans l'onglet « Elastichal ».



Le menu des modèles présent dans l'application Elastichal vous propose plusieurs options disponibles en cliquant sur les boutons situés en haut à droite de l'écran :

- « Peupler Elastic », permet d'initialiser la base de données Elasticsearch à partir des données présentes dans les modèles Elastichal.
- « Mettre à jour Elastic », permet de mettre à jour les données présentes dans Elasticsearch à partir des données présentes dans les modèles Elastichal.
- « Importer des données », permet d'importer des données dans le modèle à partir d'un fichier CSV.
- « Ajouter chercheur », permet d'ajouter un chercheur manuellement dans le modèle.

Dans le cas de la mise en route de SoVisu, nous allons importer des données à partir d'un fichier CSV.

5. Cliquez sur « Importer des données ». Sur la page suivante il vous sera demandé d'importer un fichier. Il est important que celui ci soit au format CSV, et qu'il contienne des champs précis en fonction du modèle à remplir (voir les astuces ci dessous pour plus de détails).

Astuce : Dans le cas du modèle « Chercheurs », le fichier csv devra contenir les colonnes suivante : *structSirene, ldapId, name, type, function, mail, lab, supannAffectation, supannEntiteAffectationPrincipale, halId_s, labHalId, idRef, structDomain, firstName, lastName, aurehalId*

exemple de fichier CSV Chercheurs :

structSirene	ldapId	name	type	function	mail	lab	supannAffectation	supannEntiteAffectationPrincipale	halId_s	labHalId	idRef	structDomain	firstName	lastName	aurehalId
1983	dreymonc	REYMON David	Personnel	Enseignant Chercheur Titulaire	david.r tln.fr	IMSIC	IMSIC	IUT TC	david reym	5270	1227	univ-tln.fr	David	REYMON	599503

Astuce : Dans le cas du modèle « Laboratoires », le fichier csv devra contenir les colonnes suivante : *structSirene ;*

acronym ; label ; halStructId ; rsnr ; idRef

exemple de fichier CSV Laboratoires :

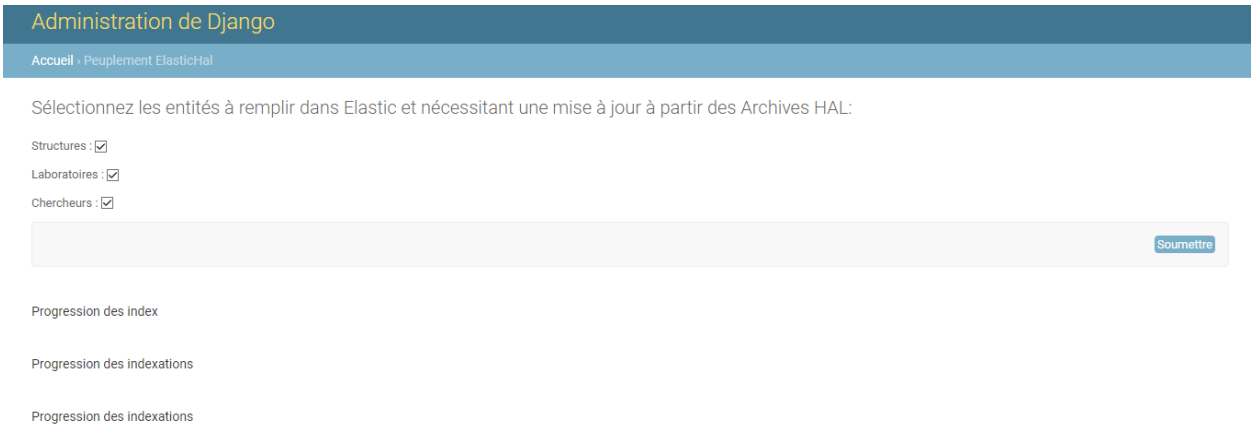
structSirene	acronym	label	halStructId	rsnr	idRef
198307662	IMSIC	Institut mediterranéen des sciences de l'information et de la communication	527028	527028	233736085

Astuce : Dans le cas du modèle « Structures », le fichier csv devra contenir les colonnes suivante : *structSirene, label, acronym, domain*

exemple de fichier CSV structures :

structSirene	label	acronym	domain
198307662	Université de Toulon	UTLN	univ-tln.fr

6. Répétez l'opération pour les autres modèles dans Elastichal.
7. Une fois les trois modèles complétés, retournez sur le menu d'un des modèles Elastichal et cliquez sur « Peupler Elastic ».



8. **Par défaut, « Peupler elastic » propose de remplir la base de données Elasticsearch avec les données présentes dans les modèles Structures, Laboratoires et Chercheurs.**
   Cliquez sur « soumettre » afin de lancer le processus de collecte, Il est possible de voir l'état de la collecte dans la partie « Progression » de la page.

Avertissement : Lors d'une première mise en route, il est impératif de lancer « peupler elastic » avec l'ensemble des modèles remplis. Le modèle structure permet de délimiter la récupération dans le cas ou les laboratoires et/ou chercheurs seraient recensés dans plusieurs structures. La récupération des données dans Elasticsearch est longue, et peut prendre du temps.

Astuce : La fonction « peupler elastic » peut également être utilisée par la suite pour mettre à jour en masse l'ensemble des données d'un ou de l'ensemble des modèles proposés par SoVisu : celle ci se base sur les données des modèles

présent dans Django mais également les éléments déjà importés dans Elasticsearch. Pour cela, il suffit de sélectionner la partie qui doit être mise à jour pour lancer un processus allégé.

5.3.1.1 Spécificités de la mise en route de SoVisu pour le développement

Dans le cas d'une installation de SoVisu sur une machine de développement n'ayant pas accès à l'authentification par CAS, il est nécessaire de définir un profil utilisateur afin d'accéder à l'application. Dans le cas d'un profil chercheur, celui ci est identifié par SoVisu grâce à son identifiant ldapId. Le nom d'utilisateur est le même que l'identifiant ldapId.

Il est cependant possible de définir un profil utilisateur nommé « adminlab », reconnu par SoVisu comme un administrateur du laboratoire et ayant donc accès complet à l'application.

1. Dans l'interface d'administration de SoVisu, cliquez sur « Utilisateurs » dans « Authentification et autorisations ».
2. Cliquez sur « Ajouter utilisateur ».
3. Créez un utilisateur ayant pour nom d'utilisateur « adminlab », le mot de passe est libre de choix.
4. Cliquez sur « Enregistrer ».

5.3.2 Initialisation des visualisations dans Kibana

Dans Kibana, il est possible de créer des visualisations à partir des données présentes dans Elasticsearch. Par défaut SoVisu propose des visualisations qui sont disponibles dans les dossiers du projet : SoVisu/doc/Dashboards/

Afin de les rendre disponible il est nécessaire de les importer dans l'instance Kibana dont dépend votre installation.

1. Dans Kibana, ouvrez le menu.
2. Dans la section Management, cliquez sur « Stack Management »
3. Sur la nouvelle page affichée, allez dans « Saved Objects »(dans la section Kibana).
4. Cliquez sur « Import » et importez les fichiers disponibles dans le dossier SoVisu/doc/Dashboards/, en sélectionnant les options « check for existing objects » et « automatically overwrite conflicts ».
5. Cliquez sur « import ».
6. Kibana vous signale l'ensemble des objets modifiés ; Cliquez sur « Done »
7. répétez les points 4 à 6 pour l'ensemble des fichiers dans le dossier.

5.4 Mise en production

Avertissement : à compléter

CHAPITRE 6

Développement

6.1 Application ElasticHal

6.1.1 Sous-paquets

6.1.1.1 Paquet elasticHal.libs

Modules

elasticHal.libs.archivesOuvertes

elasticHal.libs.archivesOuvertes.**explain_domains**(*dom_uri*)
Recherche le domaine parent d'un domaine dom_uri

elasticHal.libs.archivesOuvertes.**explore_broader**(*uri*)
Recherche le domaine parent d'un domaine donné

elasticHal.libs.archivesOuvertes.**extrait_sujets_domaines**(*data*)
À partir des données d'un l'article, extrait les sujets et domaines

elasticHal.libs.archivesOuvertes.**get_aurehalId**(*authIdHal_s*)
get the aurehalId (authIdHal_i) of the searcher with authIdHal_s (halId_s)

elasticHal.libs.archivesOuvertes.**get_concepts_and_keywords**(*aurehalid*)
Récupère les concepts et mots-clés d'un auteur à partir de son aurehalid

elasticHal.libs.archivesOuvertes.**get_halid_s**(*aurehal_id*)
Récupération du authidhal_s associé au aurehal_id depuis HAL

elasticHal.libs.archivesOuvertes.**get_label**(*label*, *lang*)
Récupére le nom complet d'un label en fonction de la langue associée

elasticHal.libs.archivesOuvertes.**recup_individu**(*authidhal_s*)

recupération des données d'un individu à partir de son authidhal_s

elasticHal.libs.dimensions

elasticHal.libs.dimensions.**getCitations**(*doi*)

Récupération des citations d'un article

elasticHal.libs.doi_enrichissement

elasticHal.libs.doi_enrichissement.**check_doi**(*doi*)

La fonction plancte avec une erreur sur certains URL qui n'est pas attrapée par le try except// Vérifie si le doi renseigné existe dans la base de données de doi.org

elasticHal.libs.doi_enrichissement.**docs_enrichissement_doi**(*doc*)

Enrichissement des documents avec les informations provenant du DOI

elasticHal.libs.hal

elasticHal.libs.hal.**find_publications**(*idhal*, *field*, *increment=0*)

Cherche les publications d'un auteur dans HAL à partir de son IDHAL

elasticHal.libs.hal.**get_content**(*hal_url*)

Récupère le contenu d'un article HAL à partir de son URL

elasticHal.libs.keyword_enrichissement

elasticHal.libs.keyword_enrichissement.**keyword_from_teeft**(*txt*, *lang*)

Enrichissement des mots clés avec les entités trouvées dans les résumés à partir de TEEFT

elasticHal.libs.keyword_enrichissement.**return_entities**(*txt*, *lang*)

Enrichissement des documents avec les entités trouvées dans les résumés à partir de la terminologie de loterre

elasticHal.libs.location_docs

elasticHal.libs.location_docs.**generate_countrys_fields**(*doc*)

Enrichi les notices avec les champs country_collaboration qui fusionne :

« deptStructCountry_s », –> Structure/regroupement d'équipes : Pays « labStructCountry_s », –> Structure/laboratoire : Pays « structCountry_s », –> Structure/regroupement d'institutions : Pays « structCountry_t »,–> Structure : Pays (copie de ce champ : structCountry_s) ! ! ! ! « rgrpInstStructCountry_s »,–> Structure/regroupement d'institutions : Pays « rgrpLabStructCountry_s »–> Structure/regroupement de laboratoires : Pays

el le champs country_origin fusionnant :

country_s, –> Pays (Code ISO 3166) rteamStructCountry_s –> Structure/équipe de recherche : Pays instStructCountry_s –> Structure/institution : Pays

en se basant sur les métadonnées Hal du document

elasticHal.libs.utils

elasticHal.libs.utils.**append_to_tree**(*scope*, *rsr*, *tree*, *state*)

Rajoute un domaine d'expertise à un arbre d'expertise

elasticHal.libs.utils.**calculate_mds**(*doc*)

Attribue un score à la qualité de description d'une notice.

elasticHal.libs.utils.**filter_concepts**(*concepts*, *validated_ids*)

Filtre les concepts qui ne sont pas dans la liste des concepts validés

elasticHal.libs.utils.**remove_page**(*pdf_file*, *pages*)

Supprime des pages d'un pdf

elasticHal.libs.utils.**should_be_open**(*doc*)

Détermine si une notice devrait être ouverte

6.1.2 Modules

6.1.2.1 elasticHal.admin

class elasticHal.admin.**ElasticActions**

Bases : object

Actions pour l'index Elasticsearch

static export_to_elastic(*request*)

Initialise la création des index Elasticsearch à partir des csv enregistrés

static update_elastic(*request*)

Met à jour les données dans les index Elasticsearch sélectionnées

class elasticHal.admin.**ExportCsv**

Bases : object

export_as_csv(*request*, *queryset*)

Exporte les données sélectionnées dans un fichier CSV

class elasticHal.admin.**LaboratoryAdmin**(*model*, *admin_site*)

Bases : ModelAdmin, *ExportCsv*

Modèle de l'administration des laboratoires

actions = ['export_as_csv']

get_urls()

Initialise les urls du modèle LaboratoryAdmin

list_display = ('acronym', 'label', 'halStructId', 'idRef', 'structSirene')

list_filter = ('structSirene',)

property media

static upload_csv(*request*)

Permet de charger un fichier CSV dans la base de données du modèle Laboratory

class elasticHal.admin.**StructureAdmin**(*model*, *admin_site*)

Bases : ModelAdmin, *ExportCsv*

Modèle de l'administration des structures

actions = ['export_as_csv']

get_urls()

Initialise les urls du modèle StructureAdmin

list_display = ('structSirene', 'acronym', 'label')

property media

static upload_csv(*request*)

Permet de charger un fichier CSV dans la base de données du modèle Structure

6.1.2.2 elasticHal.apps

class elasticHal.apps.**ElastichalConfig**(*app_name*, *app_module*)

Bases : AppConfig

Configuration de l'application elasticHal

default_auto_field = 'django.db.models.BigAutoField'

name = 'elasticHal'

6.1.2.3 elasticHal.clean_researchers_from_es_base

6.1.2.4 elasticHal.collect_from_HAL

elasticHal.collect_from_HAL.**TrouveChercheurs**(*struct*)

elasticHal.collect_from_HAL.**collect_data**(*laboratories=False*, *researcher=False*, *django_enabler=None*)

Collecte les données d'HAL et les indexe dans ElasticSearch

elasticHal.collect_from_HAL.**get_structid_list**()

Récupère la liste des structSirene des structures recensées dans ElasticSearch

elasticHal.collect_from_HAL.**init_labo**()

Initialise les données de l'index

6.1.2.5 elasticHal.insert_entities

elasticHal.insert_entities.**create_laboratories_index**(*pg*)

Créé les index pour les laboratoires

elasticHal.insert_entities.**create_structures_index**(*pg*)

Crée les index de structures dans Elasticsearch

elasticHal.insert_entities.**get_labo_list**()

Récupère la liste des laboratoires dans Elasticsearch et/ou Django DB, afin de les stocker dans une variable globale

elasticHal.insert_entities.**get_structid_list**()

Récupère la liste des structures dans Elasticsearch et / ou Django DB, afin de les stocker dans une variable globale

elasticHal.insert_entities.**temp_laboratories**(*row*)

Nettoie les données provenant de Django et les compare à celles d'Elastic pour complétion

6.1.2.6 elasticHal.models

class elasticHal.models.**Laboratory**(**args*, ***kwargs*)

Bases : Model

Définition du modèle Laboratory dans Django

exception DoesNotExist

Bases : ObjectDoesNotExist

exception MultipleObjectsReturned

Bases : MultipleObjectsReturned

acronym

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

halStructId

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

idRef

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

label

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

objects = <django.db.models.manager.Manager object>

rsnr

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

structSirene

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

class elasticHal.models.**Researcher**(**args*, ***kwargs*)

Bases : Model

Définition du modèle Researcher dans Django

exception DoesNotExist

Bases : ObjectDoesNotExist

exception MultipleObjectsReturned

Bases : MultipleObjectsReturned

aurehalId

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

firstName

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

function

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

halId_s

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

idRef

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

lab

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

labHalId

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

lastName

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

ldapId

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

mail

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

name

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

objects = <django.db.models.manager.Manager object>

structDomain

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

structSirene

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

supannAffectation

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

supannEntiteAffectationPrincipale

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

type

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

class elasticHal.models.**Structure**(**args*, ***kwargs*)

Bases : Model

Définition du modèle Structure dans Django

exception DoesNotExist

Bases : ObjectDoesNotExist

exception MultipleObjectsReturned

Bases : MultipleObjectsReturned

acronym

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

domain

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

label

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

objects = <django.db.models.manager.Manager object>

structSirene

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

6.1.2.7 elasticHal.views

elasticHal.views.**get_index_list**()

Récupère la liste des index laboratoires dans elasticsearch

6.2 Application SoVisuhal

6.2.1 Sous-paquets

6.2.1.1 Paquet sovisuhal.libs

Modules

sovisuhal.libs.elastichal

sovisuhal.libs.elastichal.**indexe_chercheur**(*ldapid*, *labo_accro*, *labhalid*, *idhal*, *idref*, *orcid*)

Indexe un chercheur dans Elasticsearch

sovisuhal.libs.esActions

sovisuhal.libs.esActions.**confirm_p**(*scope_field*, *scope_value*, *validate*)

Retourne un ensemble de documents spécifique en fonction d'un filtre, qui ont leur champ validated à une certaine valeur.

sovisuhal.libs.esActions.**date_all**()

Retourne tous les documents, triés par date de publication

sovisuhal.libs.esActions.**date_p**(*scope_field*, *scope_value*)

Retourne un ensemble de documents spécifique en fonction d'un filtre, triés par date de publication

sovisuhal.libs.esActions.**es_connector**(*mode='Prod'*)

Assure la connexion de SoVisu à l'instance ElasticSearch

sovisuhal.libs.esActions.**mode = 'Prod'**

try :

from decouple import config from ldap3 import ALL, Connection, Server from uniauth.decorators import login_required

mode = config(« mode ») # Prod –> mode = "Prod" en env Var

except :

from django.contrib.auth.decorators import login_required

mode = « Dev »

sovisuhal.libs.esActions.**ref_p**(*scope_bool_type*, *scope_field*, *scope_value*, *validate*, *date_range_type*, *scope_date_from*, *scope_date_to*)

Retourne un ensemble de documents spécifique en fonction de différents filtres, dans une période donnée

sovisuhal.libs.esActions.**ref_p_filter**(*p_filter*, *scope_bool_type*, *scope_field*, *scope_value*, *validate*, *date_range_type*, *scope_date_from*, *scope_date_to*)

Retourne un ensemble de documents spécifique en fonction de différents filtres, dans une période donnée et d'un filtre p_filter(« uncomplete », »complete », « all »).

sovisuhal.libs.esActions.**scope_all**()

Paramètre pour les requêtes ElasticSearch, retourne tous les documents

sovisuhal.libs.esActions.**scope_p**(*scope_field*, *scope_value*)

Retourne un ensemble de documents spécifique en fonction d'un filtre

sovisuhal.libs.halConcepts

sovisuhal.libs.halConcepts.**concepts**()

Retourne la liste des concepts pouvant être par la suite assignés à un chercheur

sovisuhal.libs.hceres

sovisuhal.libs.hceres.**common_data**(*list1*, *list2*)

Renvoie la liste des éléments communs entre deux listes

sovisuhal.libs.hceres.**sort_references**(*articles*, *halstructid*)

Classe les références récupérées dans un ordre défini pour le HCERES

6.2.2 Modules

6.2.2.1 sovisuhal.forms

6.2.2.2 sovisuhal.views

6.2.2.3 sovisuhal.viewsActions

sovisuhal.viewsActions.**admin_access_login**(*request*)

Fonction gérant les accès à SoVisu

sovisuhal.viewsActions.**export_hceres_xls**(*request*)

Export des données de l'HCERES d'un laboratoire sous fichier Excel (XLS)

sovisuhal.viewsActions.**idhal_checkout**(*idhal*)

Vérifie si le halId renseigné existe

sovisuhal.viewsActions.**refresh_aurehal_id**(*request*)

Mise à jour de l'id aurehal

sovisuhal.viewsActions.**update_authorship**(*request*)

Met à jour l'autorat des documents d'un utlisateur après vérification de ce dernier

sovisuhal.viewsActions.**update_members**(*request*)

Permet la mise à jour du profil utilisateur

sovisuhal.viewsActions.**validate_credentials**(*request*)

Validation des identifiants

sovisuhal.viewsActions.**validate_expertise**(*request*)

Validation des domaines d'expertise

sovisuhal.viewsActions.**validate_guiding_domains**(*request*)

Validation des domaines de guidance

sovisuhal.viewsActions.**validate_references**(*request*)

Validation des références HAL

sovisuhal.viewsActions.**validate_research_description**(*request*)

Validation de la description de recherche

sovisuhal.viewsActions.**vizualisation_url**()

Permet d'ajuster l'affichage des visualisations Kibana À intégrer dans les consts

Index des modules Python

e

elasticHal.admin, 29
elasticHal.apps, 30
elasticHal.collect_from_HAL, 30
elasticHal.insert_entities, 30
elasticHal.libs.archivesOuvertes, 27
elasticHal.libs.dimensions, 28
elasticHal.libs.doi_enrichissement, 28
elasticHal.libs.hal, 28
elasticHal.libs.keyword_enrichissement, 28
elasticHal.libs.location_docs, 28
elasticHal.libs.utils, 29
elasticHal.models, 31
elasticHal.views, 33

s

sovisuhal.libs.elastichal, 33
sovisuhal.libs.esActions, 33
sovisuhal.libs.halConcepts, 34
sovisuhal.libs.hceres, 34
sovisuhal.viewsActions, 35

Index

A

acronym (*attribut elasticHal.models.Laboratory*), 31
acronym (*attribut elasticHal.models.Structure*), 33
actions (*attribut elasticHal.admin.LaboratoryAdmin*), 29
actions (*attribut elasticHal.admin.StructureAdmin*), 29
admin_access_login() (*dans le module sovisuhal.viewsActions*), 35
append_to_tree() (*dans le module elasticHal.libs.utils*), 29
aurehalId (*attribut elasticHal.models.Researcher*), 31

C

calculate_mds() (*dans le module elasticHal.libs.utils*), 29
check_doi() (*dans le module elasticHal.libs.doi_enrichissement*), 28
collect_data() (*dans le module elasticHal.collect_from_HAL*), 30
common_data() (*dans le module sovisuhal.libs.hceres*), 34
concepts() (*dans le module sovisuhal.libs.halConcepts*), 34
confirm_p() (*dans le module sovisuhal.libs.esActions*), 33
create_laboratories_index() (*dans le module elasticHal.insert_entities*), 30
create_structures_index() (*dans le module elasticHal.insert_entities*), 30

D

date_all() (*dans le module sovisuhal.libs.esActions*), 33
date_p() (*dans le module sovisuhal.libs.esActions*), 33
default_auto_field (*attribut elasticHal.apps.ElastichalConfig*), 30
docs_enrichissement_doi() (*dans le module elasticHal.libs.doi_enrichissement*), 28
domain (*attribut elasticHal.models.Structure*), 33

E

ElasticActions (*classe dans elasticHal.admin*), 29
elasticHal.admin
    module, 29
elasticHal.apps
    module, 30
elasticHal.collect_from_HAL
    module, 30
elasticHal.insert_entities
    module, 30
elasticHal.libs.archivesOuvertes
    module, 27
elasticHal.libs.dimensions
    module, 28
elasticHal.libs.doi_enrichissement
    module, 28
elasticHal.libs.hal
    module, 28
elasticHal.libs.keyword_enrichissement
    module, 28
elasticHal.libs.location_docs
    module, 28
elasticHal.libs.utils
    module, 29
elasticHal.models
    module, 31
elasticHal.views
    module, 33
ElastichalConfig (*classe dans elasticHal.apps*), 30
es_connector() (*dans le module sovisuhal.libs.esActions*), 34
explain_domains() (*dans le module elasticHal.libs.archivesOuvertes*), 27
explore_broader() (*dans le module elasticHal.libs.archivesOuvertes*), 27
export_as_csv() (*méthode elasticHal.admin.ExportCsv*), 29
export_hceres_xls() (*dans le module sovisuhal.viewsActions*), 35

export_to_elastic() (*méthode statique elasticHal.admin.ElasticActions*), 29
ExportCsv (*classe dans elasticHal.admin*), 29
extrait_sujets_domaines() (*dans le module elasticHal.libs.archivesOuvertes*), 27

F

filter_concepts() (*dans le module elasticHal.libs.utils*), 29
find_publications() (*dans le module elasticHal.libs.hal*), 28
firstName (*attribut elasticHal.models.Researcher*), 31
function (*attribut elasticHal.models.Researcher*), 31

G

generate_countrys_fields() (*dans le module elasticHal.libs.location_docs*), 28
get_aurehalId() (*dans le module elasticHal.libs.archivesOuvertes*), 27
get_concepts_and_keywords() (*dans le module elasticHal.libs.archivesOuvertes*), 27
get_content() (*dans le module elasticHal.libs.hal*), 28
get_halid_s() (*dans le module elasticHal.libs.archivesOuvertes*), 27
get_index_list() (*dans le module elasticHal.views*), 33
get_label() (*dans le module elasticHal.libs.archivesOuvertes*), 27
get_labo_list() (*dans le module elasticHal.insert_entities*), 30
get_structid_list() (*dans le module elasticHal.collect_from_HAL*), 30
get_structid_list() (*dans le module elasticHal.insert_entities*), 30
get_urls() (*méthode elasticHal.admin.LaboratoryAdmin*), 29
get_urls() (*méthode elasticHal.admin.StructureAdmin*), 29
getCitations() (*dans le module elasticHal.libs.dimensions*), 28

H

halId_s (*attribut elasticHal.models.Researcher*), 31
halStructId (*attribut elasticHal.models.Laboratory*), 31

I

id (*attribut elasticHal.models.Laboratory*), 31
id (*attribut elasticHal.models.Researcher*), 32
id (*attribut elasticHal.models.Structure*), 33
idhal_checkout() (*dans le module sovisuhal.viewsActions*), 35
idRef (*attribut elasticHal.models.Laboratory*), 31
idRef (*attribut elasticHal.models.Researcher*), 32
indexe_chercheur() (*dans le module sovisuhal.libs.elastichal*), 33
init_labo() (*dans le module elasticHal.collect_from_HAL*), 30

K

keyword_from_teeft() (*dans le module elasticHal.libs.keyword_enrichissement*), 28

L

lab (*attribut elasticHal.models.Researcher*), 32
label (*attribut elasticHal.models.Laboratory*), 31
label (*attribut elasticHal.models.Structure*), 33
labHalId (*attribut elasticHal.models.Researcher*), 32
Laboratory (*classe dans elasticHal.models*), 31
Laboratory.DoesNotExist, 31
Laboratory.MultipleObjectsReturned, 31
LaboratoryAdmin (*classe dans elasticHal.admin*), 29
lastName (*attribut elasticHal.models.Researcher*), 32
ldapId (*attribut elasticHal.models.Researcher*), 32
list_display (*attribut elasticHal.admin.LaboratoryAdmin*), 29
list_display (*attribut elasticHal.admin.StructureAdmin*), 30
list_filter (*attribut elasticHal.admin.LaboratoryAdmin*), 29

M

mail (*attribut elasticHal.models.Researcher*), 32
media (*propriété elasticHal.admin.LaboratoryAdmin*), 29
media (*propriété elasticHal.admin.StructureAdmin*), 30
mode (*dans le module sovisuhal.libs.esActions*), 34
module
    elasticHal.admin, 29
    elasticHal.apps, 30
    elasticHal.collect_from_HAL, 30
    elasticHal.insert_entities, 30
    elasticHal.libs.archivesOuvertes, 27
    elasticHal.libs.dimensions, 28
    elasticHal.libs.doi_enrichissement, 28
    elasticHal.libs.hal, 28
    elasticHal.libs.keyword_enrichissement, 28
    elasticHal.libs.location_docs, 28
    elasticHal.libs.utils, 29
    elasticHal.models, 31
    elasticHal.views, 33
    sovisuhal.libs.elastichal, 33
    sovisuhal.libs.esActions, 33
    sovisuhal.libs.halConcepts, 34
    sovisuhal.libs.hceres, 34
    sovisuhal.viewsActions, 35

N

name (*attribut elasticHal.apps.ElastichalConfig*), 30
name (*attribut elasticHal.models.Researcher*), 32

O

objects (*attribut elasticHal.models.Laboratory*), 31
objects (*attribut elasticHal.models.Researcher*), 32
objects (*attribut elasticHal.models.Structure*), 33

R

recup_individu() (*dans le module elasticHal.libs.archivesOuvertes*), 27
ref_p() (*dans le module sovisuhal.libs.esActions*), 34
ref_p_filter() (*dans le module sovisuhal.libs.esActions*), 34
refresh_aurehal_id() (*dans le module sovisuhal.viewsActions*), 35
remove_page() (*dans le module elasticHal.libs.utils*), 29
Researcher (*classe dans elasticHal.models*), 31
Researcher.DoesNotExist, 31
Researcher.MultipleObjectsReturned, 31
return_entities() (*dans le module elasticHal.libs.keyword_enrichissement*), 28
rsnr (*attribut elasticHal.models.Laboratory*), 31

S

scope_all() (*dans le module sovisuhal.libs.esActions*), 34
scope_p() (*dans le module sovisuhal.libs.esActions*), 34
should_be_open() (*dans le module elasticHal.libs.utils*), 29
sort_references() (*dans le module sovisuhal.libs.hceres*), 34
sovisuhal.libs.elastichal
    module, 33
sovisuhal.libs.esActions
    module, 33
sovisuhal.libs.halConcepts
    module, 34
sovisuhal.libs.hceres
    module, 34
sovisuhal.viewsActions
    module, 35
structDomain (*attribut elasticHal.models.Researcher*), 32
structSirene (*attribut elasticHal.models.Laboratory*), 31
structSirene (*attribut elasticHal.models.Researcher*), 32
structSirene (*attribut elasticHal.models.Structure*), 33
Structure (*classe dans elasticHal.models*), 32
Structure.DoesNotExist, 32
Structure.MultipleObjectsReturned, 32
StructureAdmin (*classe dans elasticHal.admin*), 29
supannAffectation (*attribut elasticHal.models.Researcher*), 32
supannEntiteAffectationPrincipale (*attribut elasticHal.models.Researcher*), 32

T

temp_laboratories() (*dans le module elasticHal.insert_entities*), 30
TrouveChercheurs() (*dans le module elasticHal.collect_from_HAL*), 30
type (*attribut elasticHal.models.Researcher*), 32

U

update_authorship() (*dans le module sovisuhal.viewsActions*), 35
update_elastic() (*méthode statique elasticHal.admin.ElasticActions*), 29
update_members() (*dans le module sovisuhal.viewsActions*), 35
upload_csv() (*méthode statique elasticHal.admin.LaboratoryAdmin*), 29
upload_csv() (*méthode statique elasticHal.admin.StructureAdmin*), 30

V

validate_credentials() (*dans le module sovisuhal.viewsActions*), 35
validate_expertise() (*dans le module sovisuhal.viewsActions*), 35
validate_guiding_domains() (*dans le module sovisuhal.viewsActions*), 35
validate_references() (*dans le module sovisuhal.viewsActions*), 35
validate_research_description() (*dans le module sovisuhal.viewsActions*), 35
vizualisation_url() (*dans le module sovisuhal.viewsActions*), 35