le cnam
école d'ingénieur·e·s

CONSERVATOIRE NATIONAL DES ARTS ET METIERS

CENTRE REGIONAL PACA

MEMOIRE

Présenté en vue d'obtenir

Le DIPLOME D'INGENIEUR CNAM

SPECIALITE : Informatique option Multimédia

Par

Thomas LAURE

Extraction de données sur des sites web de manière semi-automatisée

Soutenu le 26 août 2021

JURY

PRESIDENT : Annick RAZET

MEMBRES :

Damien LEPELLETIER	Maître d'apprentissage
Ricard MARXER	Tuteur académique
Jean-Marc ROBERT	2ème membre
Olivia PAPINI	3ème membre

Remerciements

Je tiens à remercier mon maître d'apprentissage, Damien Lepelletier, pour sa pédagogie, sa disponibilité et sa confiance au cours de ces trois années d'alternance pour préparer ce projet de fin d'études, et Vincent Boisard, président de Coexel, pour m'avoir accueilli au sein de son équipe et de m'avoir fait confiance sur l'ensemble des projets qui m'ont été assignés.

Je tiens aussi à remercier mon tuteur académique, Ricard Marxer, pour ses précieux conseils qui m'ont permis de rédiger au mieux ce mémoire, ainsi que Olivia PAPINI, coordinatrice pédagogique du CNAM, pour avoir répondu à toutes mes questions d'organisation.

Je remercie aussi tous les membres du jury, les enseignants du CNAM et Laurence Gaillard-de-Villaine, directrice du service Formation Tout au Long de La Vie de l'Université de Toulon, qui m'ont aidé et aiguillé durant ce long cursus, et d'avoir fait de leur mieux pour aider notre promotion à suivre les enseignements dans les meilleures conditions possibles.

Merci pour la relecture à Damien Lepelletier, Ricard Marxer et Peter Duvauchelle.

Note sur le mémoire

Tous les mots suivis d'un astérisque (*) seront définis dans le glossaire au début des annexes.

Aussi quand j'évoquerai l'API, cela sous-entendra « API PHP ».

Liste des variables

articlesContent : *tableau d'objets* dans lequel chaque objet représente le contenu d'un article, avec le titre, la date et son URL.

cssComponents : *objet* JavaScript qui stocke les chemins CSS qui pointent vers le titre, l'URL et la date d'un article. Ses données vont servir à créer le contenu d'un plugin.

fileName : *chaîne de caractères* représentant le nom du plugin à créer. Ce nom est composé du nom de domaine du site, suffixé par un timestamp, afin de s'assurer de l'unicité des noms de fichier.

pluginId : *entier* représentant l'identifiant du plugin qui a été inséré en base de données et lié à la thématique du client. Cet identifiant est une clé primaire, et donc est unique.

pluginValidate : *booléen* permettant de retenir le fait que le plugin en cours de création a déjà été validé, pour éviter que l'utilisateur ne sature l'espace disque de notre serveur en cliquant plusieurs fois sur le bouton de validation.

selectedThematic : *entier* récupéré dans le champ de sélection sur l'interface utilisateur. Cet entier est lié à une et une seule thématique. Il est récupéré en base de données, ainsi que le nom de la thématique, pour être affichés dans le champ de sélection.

Liste des abréviations

AFNOR : Association Française de Normalisation

AJAX : Asynchronous JavaScript And XML

API : Application Programming Interface

CI/CD : Continuous Integration/Continuous Deployment

CSS : Cascading Style Sheets

DOM : Document Object Model

ECMA : European Computer Manufacturers Association

HTML : HyperText Markup Language

IHM : Interface Homme Machine

ISO : International Organization for Standardization

JSON : JavaScript Object Notation

NAS : Network Attached Storage

PHP : PHP Hypertext Preprocessor

PSR : PHP Standards Recommendations

RGPD : Règlement Général sur la Protection des Données

RSS : Really Simple Syndication

SGBD : Système de Gestion de Base de Données

URL : Uniform Resource Locator

XML : eXtensible Markup Language

Glossaire des termes techniques

AJAX : Asynchronous JavaScript + XML. Il ne s'agit pas d'une technologie à proprement parler, mais d'une approche qui utilise un ensemble de technologies existantes dont l'objectif est de faire des mises à jour rapides de l'IHM sans avoir à recharger l'intégralité de la page (par exemple le chargement de données en base de données).

Algorithme automatisé : Algorithme ne nécessitant pas d'intervention humaine pour fonctionner.

Algorithme semi-automatisé : Algorithme nécessitant une intervention humaine minime pour remplir sa mission.

API : Ou interface de programmation d'application, est un ensemble normalisé de classes, de méthodes, de fonctions et de constantes qui sert de façade par laquelle un logiciel offre des services à d'autres logiciels. Elle est offerte par une bibliothèque logicielle ou un service web, le plus souvent accompagnée d'une description qui spécifie comment des programmes consommateurs peuvent se servir des fonctionnalités du programme fournisseur (Wikipédia).

Application mobile-first : Il s'agit d'une application qui a été développée pour s'adapter en premier lieu à l'affichage sur plateforme mobile, les règles CSS s'écrasant au fur-et-à-mesure de leur interprétation par le navigateur, et les plateformes mobiles étant dans leur ensemble plus limitées en performances que des stations fixes, cela permet d'augmenter la performance de l'application sur ces premières.

Application responsive : Il s'agit d'une application conçue pour que son interface s'adapte à tous les formats d'écran.

Back-end : Désigne l'opposé du front-end. Il définit, entre autres, l'ensemble des opérations permettant le traitement des données. De manière plus générale il représente la partie immergée de l'iceberg, ou la partie non visible d'un logiciel.

Big Data : Désigne les ressources d'informations dont les caractéristiques en termes de volume, de vélocité et de variété imposent l'utilisation de technologies et de méthodes analytiques particulières pour générer de la valeur, et qui dépassent en général les capacités d'une seule et unique machine et nécessitent des traitements parallélisés (Wikipédia).

Bootstrap : Bootstrap est une librairie CSS et JavaScript responsive et mobile-first qui contient un ensemble d'outils uniformisés et personnalisables qui permettent de créer les interfaces des applications web. Ce genre de librairie permet de faire gagner beaucoup de temps dans le développement d'interfaces web.

CI/CD : L'approche CI/CD garantit une automatisation et une surveillance continues tout au long du cycle de vie des applications, des phases d'intégration et de test jusqu'à la distribution et au déploiement. Ensemble, ces pratiques sont souvent désignées par l'expression « pipeline CI/CD » et elles reposent sur une collaboration agile entre les équipes de développement et d'exploitation (DevOps) (RedHat). Le CI représente l'intégration continue (Continuous Integration) et le CD représente le déploiement continu (Continuous Deployment).

CSS : Les Cascading Style Sheets ou Feuilles de Styles en Cascade, forment un langage informatique qui décrit la présentation ou la forme des documents HTML grâce à un ensemble de règles.

Crawler : Brique de l'application chargée de parcourir les différentes sources afin de récupérer les données et de les insérer dans la base de données.

DOM **:** Document Object Model. Il s'agit d'une interface de programmation normalisée par le W3C, qui permet à des scripts d'examiner et de modifier le contenu du navigateur web (Wikipédia).

ECMA International : Organisme de standardisation active dans le domaine de l'informatique.

ECMA Script : Ensemble de normes concernant les langages de programmation de type script et standardisées par Ecma International dans le cadre de la spécification ECMA-262. Il s'agit donc d'un standard, dont les spécifications sont mises en œuvre dans différents langages de script, comme JavaScript ou ActionScript. C'est un langage de programmation orienté prototype (Wikipédia).

Ecoconception : Démarche préventive et innovante qui permet de réduire les impacts négatifs du produit, service ou bâtiment sur l'environnement sur l'ensemble de son cycle de vie (ACV), tout en conservant ses qualités d'usage (Wikipédia). Tout en allégeant les traitements pour de meilleures performances.

Fonction de rappel : fonction passée dans une autre fonction en tant qu'argument, qui est ensuite invoquée à l'intérieur de la fonction externe pour accomplir une sorte de routine ou d'action (MDN).

Framework PHP : Un framework PHP est un environnement de développement basé sur le langage PHP. Un tel environnement fournit une structure applicative et un ensemble de composants et d'outils pour le développement d'applications web.

Front-end : Le développement web frontal correspond aux productions HTML, CSS et JavaScript d'une page Internet ou d'une application qu'un utilisateur peut voir et avec lesquelles il peut interagir directement (Wikipédia).

Git : Git est un logiciel open source de gestion de version créé par Linus Torvalds.

Machine Learning : Ou apprentissage machine, est un champ d'étude de l'intelligence artificielle qui se fonde sur des approches mathématiques et statistiques pour donner aux ordinateur la capacité d'apprendre à partir de données, c'est-à-dire améliorer leurs performances à résoudre des tâches sans être explicitement programmé pour chacune (Wikipédia).

Méthode Agile : Méthode de gestion de projet ayant pour objectif de placer le client au centre des transactions, et qui suit des valeurs et principes définis dans le Manifeste Agile. Exemples de méthodes de gestion de projet Agile : Scrum, Kanban, Scrumban.

NAS : Network Attached Storage ou serveur de stockage en réseau, il s'agit d'un boîtier constituant un serveur de stockage de fichiers relié à un réseau et centralisé, permettant aux acteurs autorisés d'en consulter le contenu, de sauvegarder des documents.

Norme : Document de référence élaboré par un organisme de normalisation reconnu comme l'ISO et l'AFNOR. Une norme définit les règles, caractéristiques, bonnes pratiques et recommandations applicables à des produits, services, méthodes, processus ou organisations.

Plugin : Correspond à la transformation du contenu d'un site web en flux RSS exploitable.

Principes SOLID : Ensemble de bonnes pratiques de conception à utiliser pour faciliter la maintenance, la compréhension, la flexibilité d'infrastructures logicielles.
Les différents principes :

- *Responsabilité unique* (Single responsibility) : Une classe, une fonction ou une méthode doit avoir une et une seule responsabilité.
- *Ouvert/fermé* (Open/closed) : une entité applicative doit être fermée à la modification directe mais ouverte à l'extension.
- *Substitution de Liskov* (Liskov substitution) : une instance de type T doit pouvoir être remplacée par une instance de type G, tel que G sous-type de T, sans que cela ne modifie la cohérence du programme.
- *Ségrégation des interfaces* (Interface segregation) : préférer plusieurs interfaces spécifiques pour chaque client plutôt qu'une seule interface générale.
- *Inversion des dépendances* (Dependency inversion) : il faut dépendre des abstractions, pas des implémentations. (Wikipédia)

Product Backlog : Il s'agit d'un document partagé et élaboré avec l'ensemble de l'équipe du projet, il contient une liste ordonnée de user stories* priorisées.

PSR : PHP Standards Recommandations, il s'agit des standards de programmation en PHP tels que la documentation, la gestion des espaces de noms, les conventions de nommages, etc. Elles sont disponibles à cette adresse : https://www.php-fig.org/psr/

Puppeteer : Bibliothèque développée par Google qui simule le lancement d'un navigateur web pour réaliser différentes tâches dans celui-ci comme des captures d'écran, de la récupération de données, etc… C'est l'équivalent de Selenium mais réalise les tâches de manières asynchrones et bénéficie de l'aspect orienté client de JavaScript.

Retex : Le retour d'expérience est l'enrichissement des connaissances pour un individu ou une organisation apprenante (Wikipédia).

RGPD : Règlement Général sur la Protection des Données, ce règlement encadre le traitement des données personnelles sur le territoire de l'Union Européenne (CNIL).

Session de crawl : À Coexel, une session de crawl correspond à une période dans la journée où nous récupérons les informations de nos différentes sources à l'aide de programmes automatisés. Généralement la nuit

Standard : Format élaboré par un petit nombre d'acteurs et adopté par des consortiums. Un standard peut être ouvert (diffusé librement) ou fermé (les spécifications ne sont pas rendues publiques et/ou l'utilisation est restreinte par son propriétaire).

User story : Il s'agit de l'ensemble des fonctionnalités d'un projet décrites individuellement. Elles contiennent :

- Un ID
- Un libellé : pour nommer la tâche.
- Une description : le descriptif de la fonctionnalité attendue par le client.
- Une importance : un entier qui fixe la priorité de la story. Elle peut être changée en cours de réalisation du projet.
- Le sprint associé à la user story
- Les dépendances : user stories nécessaires pour pouvoir passer à la user story courante.

Auxquelles s'ajoutent parfois un jeu de tests pour la validation, ainsi que des notes si nécessaire.

Z-index : Lorsque des éléments se chevauchent, le *z-order* détermine l'ordre des différentes couches que formeront les éléments. Généralement, un élément couvrira un autre élément si sa valeur de `z-index` est supérieure à celle du deuxième élément (MDN).

Table des matières

Introduction

Ce mémoire se place dans la continuité des rapports d'activité rédigés les semestres précédents et aussi dans la continuité du projet principal que j'ai mené depuis mon arrivée à Coexel.

Dans le cadre de son activité de récupération d'informations sur Internet, Coexel crée différents types de connecteurs entre nos services et la toile pour extraire les données sur les sources que nos clients souhaitent suivre. Ces sources peuvent être de différentes natures : réseaux sociaux, brevets, appels d'offre, sites web, etc.

Nous allons plus nous intéresser aux sites web. Un site web peut proposer des flux RSS pour suivre ses actualités, ou pas. Dans le second cas, nous sommes obligés de créer un connecteur, appelé plugin*, dont le rôle sera de transformer le contenu d'un site web en un flux RSS exploitable par Mytwip. En effet, notre application s'appuie sur les flux RSS pour récupérer les informations et les insérer en base de données pour que celles-ci soient redirigées vers le client via son interface.

Un plugin fonctionne de la manière suivante : nous devons lui spécifier la page que le client souhaite suivre, ensuite nous précisons les chemins CSS qui permettent d'extraire les titres, les dates, les URL et les résumés des articles, ces deux derniers sont facultatifs car certains sites n'en proposent pas. Une fois le plugin créé, nous avons un moteur, que nous appelons scraper, qui va appeler le plugin, celui-ci étant stocké sur nos serveurs, et parcourir les chemins CSS* spécifiés pour récupérer et mettre en forme les données afin de les stocker en base de données, en évitant les doublons. Dans les phases de récupération de données quotidiennes, le scraper va appeler les plugins un par un, ce qui prend quelques heures par rapport au nombre de plugins que nous avons (voir Annexe 1).

Actuellement j'ai la charge de réaliser des plugins et de la maintenance de notre scraper. Le problème avec cette approche est qu'elle me fait perdre du temps sur les développements parallèles, mais elle fait aussi perdre du temps à notre équipe de production qui reçoit les demandes clients et qui est chargée d'ajouter manuellement les plugins à notre logiciel.

L'objectif de ce projet est donc de mettre en place une solution semi-automatisée* qui doit simplifier le processus qui consiste à recevoir un plugin, le développer et l'intégrer à notre outil de veille Mytwip.

Notre projet se nomme **Plugins Auto**.

Le développement de ce mémoire se fera dans l'ordre des chapitres suivants :

- *Etat de l'art* : présentation des travaux qui ont été réalisés par le passé dans différentes études dans le domaine de la récupération de données de manière automatisée*. L'état de l'art rassemble aussi des informations sur les moyens de traitement semi-automatisés.
- *La problématique* : expression du besoin qui a mené à ce projet, les contraintes qui y sont associées, que ce soit au niveau des ressources ou au niveau des contraintes techniques, les méthodes utilisées pour développer et mener le projet dans son ensemble. Dans cette partie sera également exposée l'analyse des risques.
- *Analyse des principaux choix* : dans cette partie seront justifiés les choix méthodologiques et techniques exprimés dans la partie précédente.
- *Solutions proposées* : expression des solutions techniques mises en œuvre pour répondre au besoin qui a mené au développement de ce produit. Un avis critique sur la mise en place de la solution clôturera cette partie.
- *Résultats* : exposition des résultats et comparaison du projet final dans son ensemble par rapport à ce qui a été prévu initialement avec une analyse critique.
- *Conclusion* : bilan du travail réalisé par l'équipe, perspectives à court, moyen et long terme, et bilan personnel.

Après ces différentes parties, suivront les sources qui m'ont aidé à la rédaction de l'état de l'art, les annexes qui contiennent des éléments techniques détaillés, et pour conclure, un résumé en français et en anglais de ce mémoire.

Etat de l'art

I. Approche automatisée

a. Extraction de contenu via l'occurrence de patterns

La littérature propose différentes manières d'extraire ces actualités de manière automatisée, notamment Yu-Chieh Wu dans [5], qui se base sur un algorithme d'extraction de texte dans des vidéos, pour proposer un moyen d'extraire le contenu textuel d'articles en faisant de la détection de textes via des patterns qui se répètent sur la page en question. Ces patterns se basent sur des éléments comme la hauteur des blocs d'actualités, leur largeur, leur style CSS, etc… La présence d'éléments indésirables tels que les publicités commerciales et les barres de navigation complexifie cette extraction, d'autant plus que les sites ne possèdent pas une structure HTML commune, et celle-ci change avec le temps selon les nouvelles technologies, tendances du web et pratiques de développement individuelles. Pour pallier ce problème visuel, certains algorithmes proposent d'analyser l'arborescence du DOM* (Document Object Model) pour extraire plus facilement le contenu textuel en supprimant les éléments indésirables. Selon les tests réalisés, cet algorithme parvient à extraire le contenu des actualités dans 96.38% des cas. Ces tests ont été réalisés avec 3506 pages web couvrant 15 régions géographiques avec 11 langues différentes. Cet algorithme est à perfectionner pour voir s'il est capable de prendre en compte plus de langues, et pour qu'il soit capable de récupérer des éléments supplémentaires comme des dates.

b. Extraction de contenu grâce à des expressions régulières

Dans ce même article, une approche consistant à mettre en place des regex (expressions régulières) pour repérer les éventuels patterns est proposée, mais cette approche nécessite une mise à jour et/ou une maintenance régulière du programme, donc une intervention humaine, pour intégrer des regex qui correspondent aux nouveaux cas rencontrés.

c. Repérage des actualités via la similarité entre les arbres du DOM

Reis et al. décrivent un algorithme dans [3], et illustré dans la figure 1, qui établit des arbres à partir d'une analyse des éléments du DOM pour les comparer entre eux. Cette opération est réalisée afin de déterminer un ensemble d'arbres qui possèdent une structure commune. Cela permet donc d'éliminer les arbres ponctuels, à savoir les contenus indésirables dans notre recherche d'articles d'actualité. D'après les résultats de leurs tests, 87.71% de l'ensemble des

articles seraient extraits sur 4088 pages analysées. Cette technique a plusieurs limites. Elle ne permet pas d'extraire le contenu des actualités dont la structure interne ne respecte pas certains standards du web, et elle est incapable d'extraire des articles provenant de sites dont les actualités sont insérées à la main à cause d'une potentielle hétérogénéité dans la structure de ceux-ci, et aura plus de mal à déterminer des arbres communs.

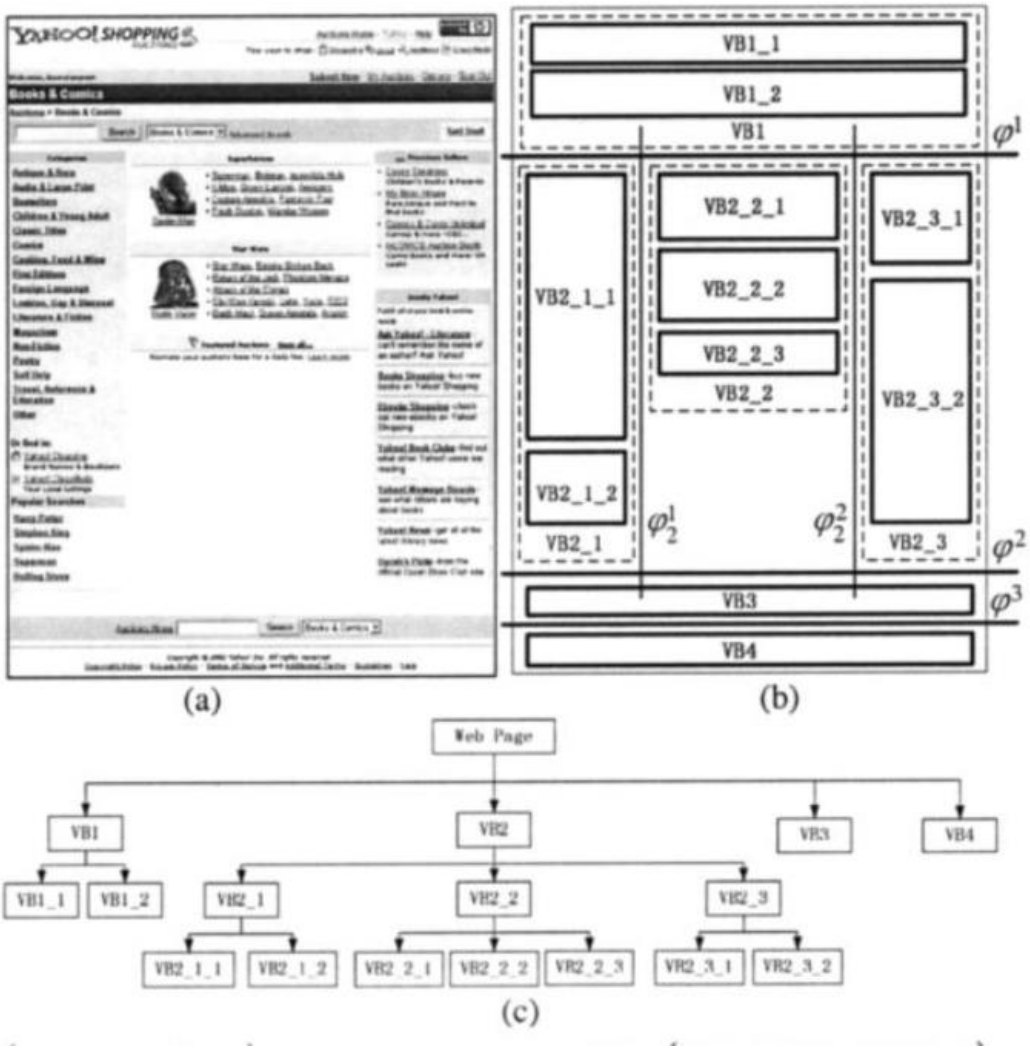


FIGURE 1 : SEGMENTATION D'UNE PAGE WEB SELON LES ARBRES DU DOM

d. Extraction de contenu via un algorithme de Machine Learning dans les éléments segmentés

Le document de Yu-Chieh Wu se réfère à [2], qui propose de segmenter une page web selon les régions clés de celle-ci telles que le titre des actualités, leur date de publication et leur résumé. La méthode présentée ici s'appuie sur un algorithme de Machine Learning capable de détecter le contenu recherché dans les éléments segmentés qui semblent posséder certaines similitudes. Les auteurs de l'article ont créé leur propre parseur CSS pour ne pas

utiliser de moteur de rendu, ce qui augmenterait le temps de traitement par rapport à ce qu'ils souhaitent faire de ces données car ils devaient traiter beaucoup de pages en même temps.

Dans notre cas, nous pourrons utiliser un web scraper JavaScript, Puppeteer*, qui va simuler l'utilisation d'un navigateur web, couplé à un serveur Node.js, car ces briques sont déjà mises en place et utilisées au sein de l'entreprise car le volume de données que nous gérons le permet. Et le développement d'un tel outil n'entre pas dans le cadre de ce PFE. La méthode décrite dans ce document pourrait présenter quelques limites lorsque nous essaierons de récupérer des articles sur des sites dont la structure du DOM n'est pas normalisée et/ou trop hétérogène pour pouvoir établir une segmentation nette.

e. Extraction par la reconstitution d'une page suivant la perception visuelle

Pour pallier ce problème, l'algorithme VIPS (VIsion-based Page Segmentation), évoqué dans [4], propose de reconstruire la structure d'une page selon la perception visuelle qu'un humain a de cette page. C'est-à-dire que l'algorithme est capable d'interpréter les blocs d'une page et les séparateurs de ces blocs, indépendamment de la structure HTML de cette page, afin de reconstruire cette structure selon un ensemble de normes et/ou de standards définis. Ce qui faciliterait l'extraction de contenu à partir d'une page HTML correctement structurée. Un autre avantage de l'algorithme VIPS est qu'il permet de contourner certains protocoles de sécurités empêchant des robots de collecter le code HTML d'une page donnée, car un tel algorithme prend en entrée le rendu de la page.

Cette technique comporte un défaut qui est le fait qu'elle met l'intégralité du contenu d'une actualité présentée dans un seul bloc visuel. En fonction de la forme du contenu, associé, l'algorithme peut présenter des difficultés à extraire les informations utiles en oubliant certains éléments du résumé, ou alors considérer le titre comme étant un résumé, ou inversement. De plus, si le template de la page change, il faut réentraîner l'algorithme pour qu'il le prenne en compte. C'est à cette problématique que répond Junfeng Wang et al. dans [6] avec leur algorithme de Machine Learning, qui ne s'appuie ni sur la structure HTML de la page, ni sur le rendu visuel de celle-ci. Il se base sur le fait que si nous, en tant qu'humains, sommes capables de différencier le titre et le résumé d'un article, c'est parce que ces deux éléments comportent certaines caractéristiques propres, ainsi que certaines qui les mettent en corrélation. C'est grâce à ces caractéristiques qu'ils ont pu atteindre un score moyen de

98.1% de réussite dans l'extraction d'articles d'actualité, qui plus est avec peu d'échantillons pour entraîner leur algorithme.

f. Extraction d'actualités depuis des flux RSS

Une approche différente est proposée par trois membres du département d'informatique de l'institut de technologie de Tokyo à travers [1]. Ils appuient eux aussi sur la difficulté d'adapter des algorithmes aux changements relatifs à l'apparence des pages web, ce sur quoi s'appuie l'algorithme VIPS, à la structure du DOM vis-à-vis du calcul de distance entre des nœuds qui présentent des similarités structurelles, et que ce type d'approches n'est pas encore utilisé à grande échelle à cause d'une grande nécessité d'intervention humaine dans la maintenance de ce genre de projets. Les membres de l'institut de technologie de Tokyo proposent donc un algorithme indépendant de la structure du DOM et de l'apparence d'un site web d'actualités, en se basant sur les flux RSS fournis par de tels sites. En effet, ils affirment que 97% des principaux sites web d'actualités américains proposent des flux RSS, et que généralement ces sites s'appuient sur ceux-ci pour afficher leurs informations, et qu'ils s'en servent également pour réaliser des newsletters pour les abonnés. Il existe différents formats pour ce type de flux, mais ils possèdent généralement une structure et des balises en commun, telles que les balises <title> et <link>. Grâce aux balises citées précédemment, pour chaque élément <item> du flux RSS, leur algorithme va ouvrir la page associée à l'URL contenue dans la balise <link>, détecter le titre de la page grâce au contenu de la balise <title>, afin d'établir une liste de mots clés permettant, lors du parcours de la page ouverte, d'identifier le contenu textuel en lien avec ces mots clés, pour reconstituer un article, comme montré dans la figure 2.

Cet algorithme est pleinement fonctionnel sur des sites web qui se basent sur des flux RSS, mais ne fonctionne pas sur des plus petits sites n'utilisant pas ce genre de flux.

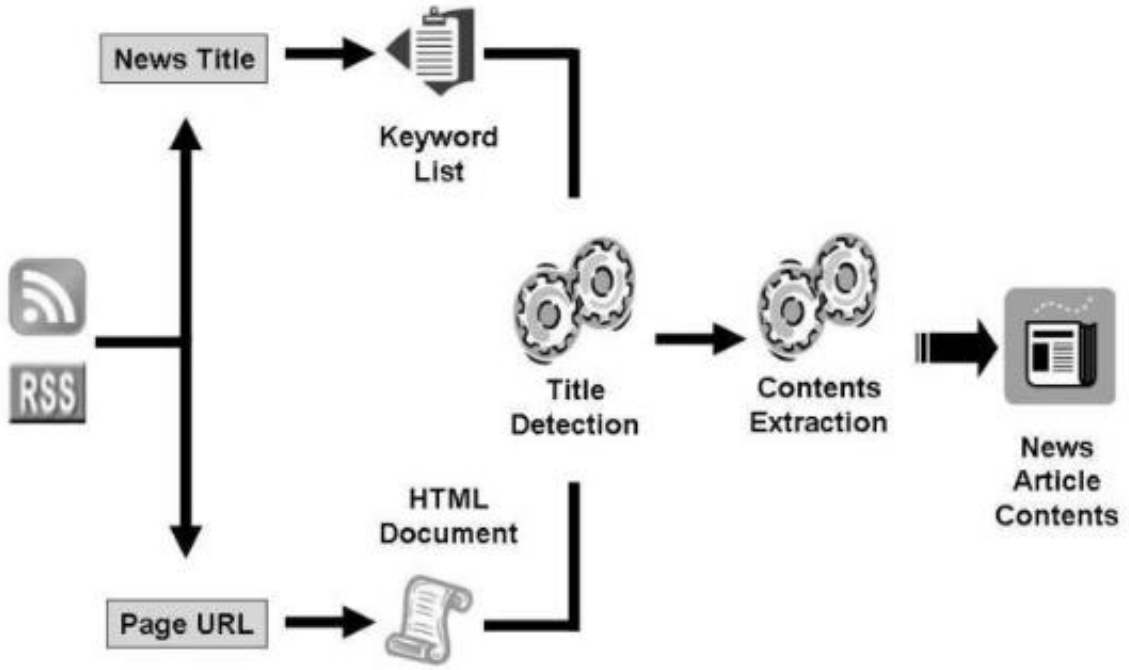


FIGURE 2 : RECONSTITUTION D'ARTICLES DEPUIS UN FLUX RSS

g. Extraction d'actualités depuis une API

La classification d'articles est l'objectif des travaux présentés par Iglesias et al. dans [7]. Ils proposent d'utiliser l'API* NYT API du New York Times pour récupérer l'ensemble des actualités du site. Cette API leur permet d'éviter d'utiliser ou de créer un algorithme dépendant de l'apparence du site et/ou de la structure du DOM. Pour la classification des articles, les auteurs utilisent le service externe RapidMiner qui va générer un ensemble de mots clé depuis le contenu de l'article parcouru, puis nettoyer les mots clé inutiles par un filtre permettant d'enlever ceux dont le nombre d'apparitions est trop faible. Cette technique d'extraire des articles depuis une API fournie par le site en question est sans issue du fait que seul un petit nombre de sites d'actualités propose ce genre d'outil.

II. Approche semi-automatisée

a. Introduction

L'approche semi-automatisée consiste à mélanger une intervention humaine et l'efficacité d'un algorithme. Celui-ci ne sera pas un algorithme de Machine Learning supervisé, nécessitant la mise en forme d'un jeu de données. Dans ce cas nous sortons d'un contexte d'utilisation de l'intelligence artificielle et de publications scientifiques, pour étudier le fonctionnement d'outils déjà ouverts au public. La plupart d'entre eux sont des applications de surveillance de pages web.

b. Outils

Parmi ces outils, Distill (figure 3) se présente comme une extension pour navigateur, compatible avec Firefox, Chrome et Opera, et une application mobile. Ce module permet de sélectionner sur des pages web, un ou plusieurs éléments que nous souhaitons surveiller, et lorsque ces éléments seront modifiés, nous recevons une notification avec le nouveau contenu. Il ne s'agit donc pas d'un outil permettant d'extraire le contenu d'un site, mais de surveiller ce contenu et de notifier lors d'un changement. Avec ceci nous pouvons donc sélectionner, sur une page d'actualités, le conteneur contenant l'ensemble des articles. Lorsque nous sommes notifiés, cela pourra signifier deux choses : soit l'apparence du site a été modifiée, soit de nouveaux éléments sont sur la page. Etant donné que sur ce genre de sites il peut être assez long de modifier la structure d'une page à cause des changements que ça implique côté back-end*, il est plus probable que la notification soit due à de nouveaux articles sur le site.

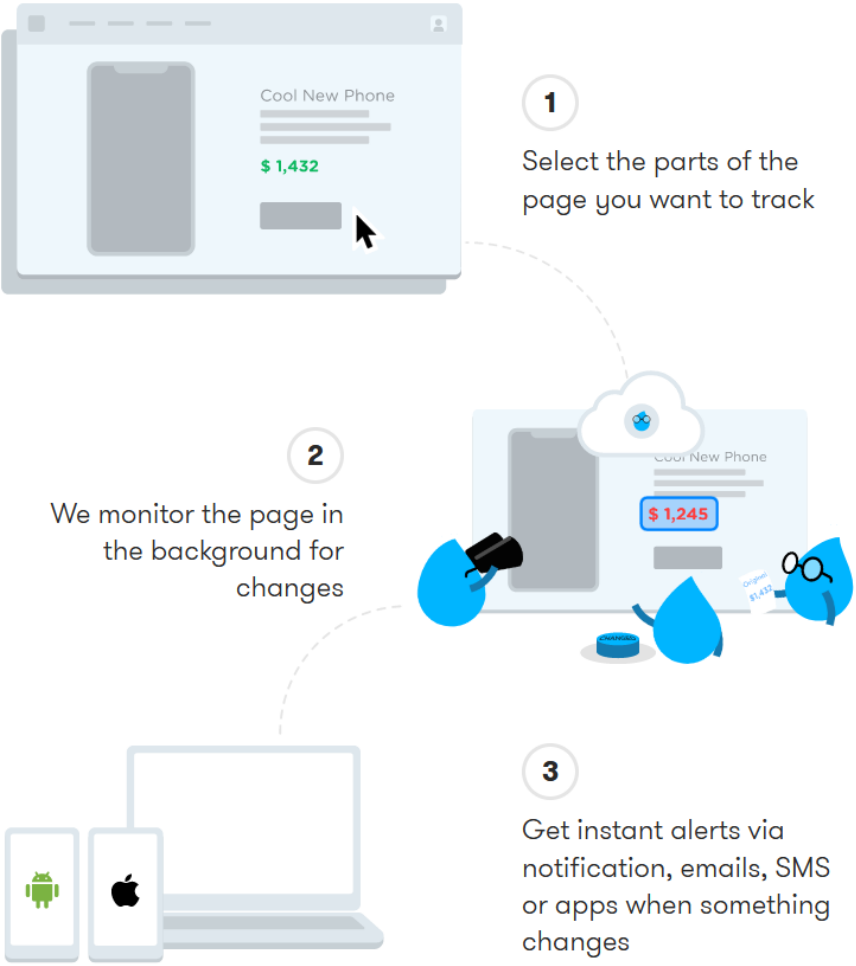


FIGURE 3 : FONCTIONNEMENT DE DISTILL (HTTPS://DISTILL.IO)

Visualping est lui aussi une extension pour navigateur et propose sensiblement les mêmes fonctionnalités que Distill, mais est plus limité et gère moins de formats, tels que les documents PDF et certains sites ne sont pas accessibles, ceux qui utilisent AJAX* par exemple.

D'autres outils du même genre existent tels que ChangeTower, Wachete, ou encore Versionista, et se différencient via leurs possibilités de personnalisation, leur fonctionnalités accessibles gratuitement et leur manière de notifier

III. Bilan de l'état de l'art

D'après les éléments étudiés, de nombreuses approches automatisées sont proposées, mais peu d'entre elles sont pleinement fonctionnelles et/ou utilisables dans notre cas d'usage, et par rapport au temps dont nous disposons pour mettre en place de telles méthodes.

Les plus viables sont celles qui se basent sur les flux RSS sont utilisées par de nombreux outils, mais elles nécessitent d'être améliorées pour prendre en compte les sites qui ne disposent pas de flux RSS. Cette amélioration pourrait être le fruit d'un mélange de différentes méthodes. Les méthodes automatisées sont donc déjà réalisées, mais nécessitent d'être améliorées ou adaptées.

D'un autre côté, les méthodes semi-automatisées avec la surveillance de sites web sont bien en place, sont plus accessibles et plus simples à déployer. Leur grand avantage est qu'elles ne nécessitent pas de mise en forme de données contrairement aux méthodes qui s'appuient sur le Machine Learning.

IV. Conclusions

Après avoir établi cet état de l'art et après avoir analysé nos besoins et nos contraintes, nous allons nous tourner vers une solution semi-automatisée, car celles-ci sont à ce jour plus faciles d'accès, nécessite moins de temps d'auto-formation, mais sont surtout plus matures que les solutions automatisées du fait de leur simplicité.

L'idée de faire sous-traiter l'extraction d'actualités par un prestataire externe nous a déjà traversé l'esprit, mais après avoir contacté le prestataire en question et établi une période de test, qui n'a pas été concluante à cause d'un manque d'efficacité et de communication.

En ce qui concerne le langage de programmation choisi pour réaliser ce travail, nous allons nous tourner vers le JavaScript pour sa flexibilité et sa facilité d'utilisation et de déploiement

dans les navigateurs web. Mais aussi parce qu'il s'agit d'une technologie que nous utilisons au quotidien dans nos développements.

V. SWOT

SWOT de la mise en place d'une méthode automatisée :

SWOT de la mise en place d'une méthode semi-automatisée :

Problématique

I. Besoin

Le besoin principal est de simplifier notre processus qui consiste à ce que le client constitue une liste de sites pour lesquels il souhaite suivre les actualités, il soumet cette liste à notre équipe de production qui va vérifier s'il existe des plugins valides pour chacun des sites, ceux pour lesquels il n'y en a pas sont envoyés à l'équipe de développement qui va réaliser les plugins correspondants, puis les envoyer à l'équipe de production qui va les ajouter manuellement à notre outil de veille Mytwip (figure 4). À l'issue de ce processus, les clients pourront voir dans leur espace utilisateur les actualités récupérées depuis les sites soumis.

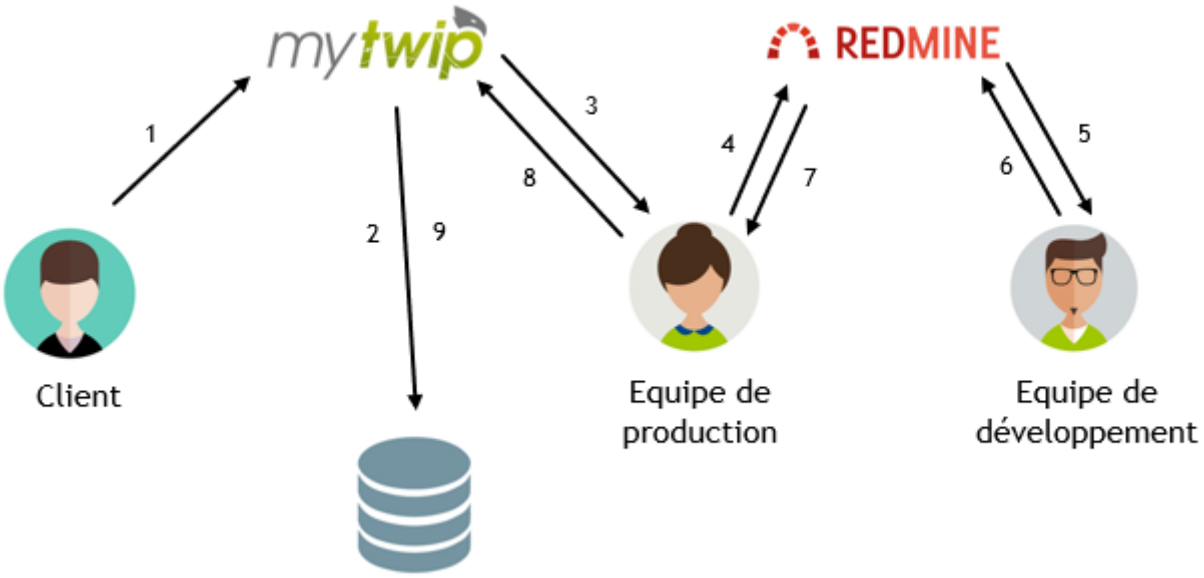


FIGURE 4 : PROCESSUS DE SOUMISSION DE PLUGINS (PLUS DE DETAILS EN ANNEXE 2)

Le problème est que ce processus est composé de plusieurs étapes qui nous prennent un temps non négligeable qui peut être investi ailleurs, surtout dans le cas de la réalisation des plugins.

Le besoin est donc de concevoir un outil qui permet au client de concevoir lui-même ses plugins grâce à une interface sur laquelle il soumet le site et pourrait créer son plugin grâce aux options disponibles.

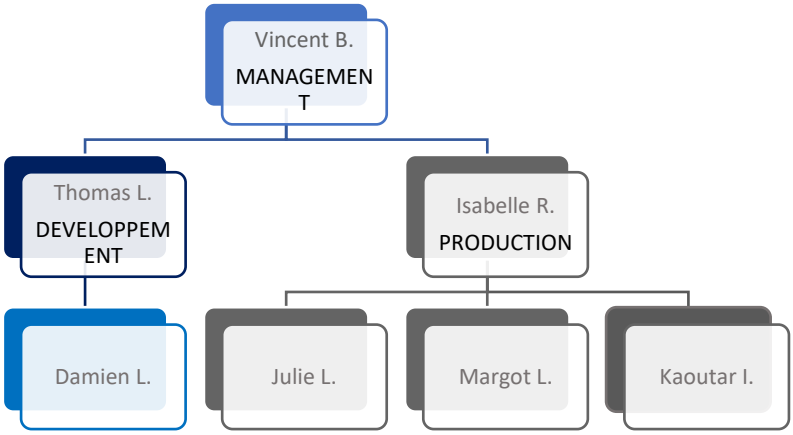
Etant donné que nous allons utiliser une méthode d'extraction semi-automatisée, la problématique dégagée est la suivante : **Comment extraire des informations depuis des sites web d'actualités de manière semi-automatisée ?**

II. Contraintes

a. Moyens humains

Les différents acteurs qui interviennent tout au long de ce projet sont :

- L'équipe de production de Coexel composée de 4 personnes qui feront office de clients
- Vincent BOISARD en tant que product owner
- Thomas LAURE pour le développement et la gestion du projet
- Damien LEPELLETIER en tant que développeur en soutien sur les tâches plus complexes



b. Moyens techniques

Infrastructure :

- Un serveur web Apache 2
- Les bases de données gérées avec le SGBD MySQL 5.7
- Un serveur web Node.js avec le framework Express

Langages de programmation orientés serveur :

- PHP 7
- JavaScript (avec Node.js) et la librairie Puppeteer

Langages de programmation orientés client :

- HTML 5
- CSS 3/Bootstrap 5
- JavaScript
- jQuery

Liste du matériel utilisé pour développer l'application :

- Deux PC fixes d'une valeur approximative de 500€ l'unité.
- Deux ultrabooks pour le télétravail d'une valeur approximative de 1000€ l'unité.
- Le serveur dédié utilisé pour héberger notre solution.
- Le serveur qui héberge notre SGBD.

Nous aurons les contraintes fonctionnelles suivantes :

- L'application devra être responsive* et mobile-first*.
- Il s'agira d'une Single Page Application, c'est-à-dire que tous les éléments de l'interface se trouveront sur la même page.

Exigences	Contraintes	Solutions
Facilité de maintenance	Temps	Utiliser les technologies classiques de développement web : HTML, CSS, JS et PHP
Application responsive et mobile-first	Peu de ressources humaines	Bootstrap pour la partie responsive et mobile-first
Simplicité de déploiement	L'application doit être modulaire	Utilisation de technologies déjà utilisées à Coexel.
Single Page Application	Minimiser le nombre d'éléments d'interaction	Utilisation d'AJAX ou de Fetch API

III. Méthodologies utilisées

Pour mener à bien ce projet, nous utilisons la méthodologie Agile* avec la méthode Scrumban, mélange entre la méthodologie Scrum et la méthodologie Kanban, à travers l'outil collaboratif Notion. Tous les documents liés au projet seront accessibles depuis cet outil pour les centraliser. Nous aurons une sauvegarde de ceux-ci sur un dispositif de stockage externe en cas de panne : coupure Internet, défaillance du service distant, perte de données, etc.

Ce projet sera versionné avec Git* sur les machines et sur le dépôt distant GitHub de Coexel. Nous travaillons en équipe, il est donc nécessaire d'établir ou d'utiliser des normes déjà existantes à ce niveau. Ici nous allons utiliser le Gitflow (figure 4).

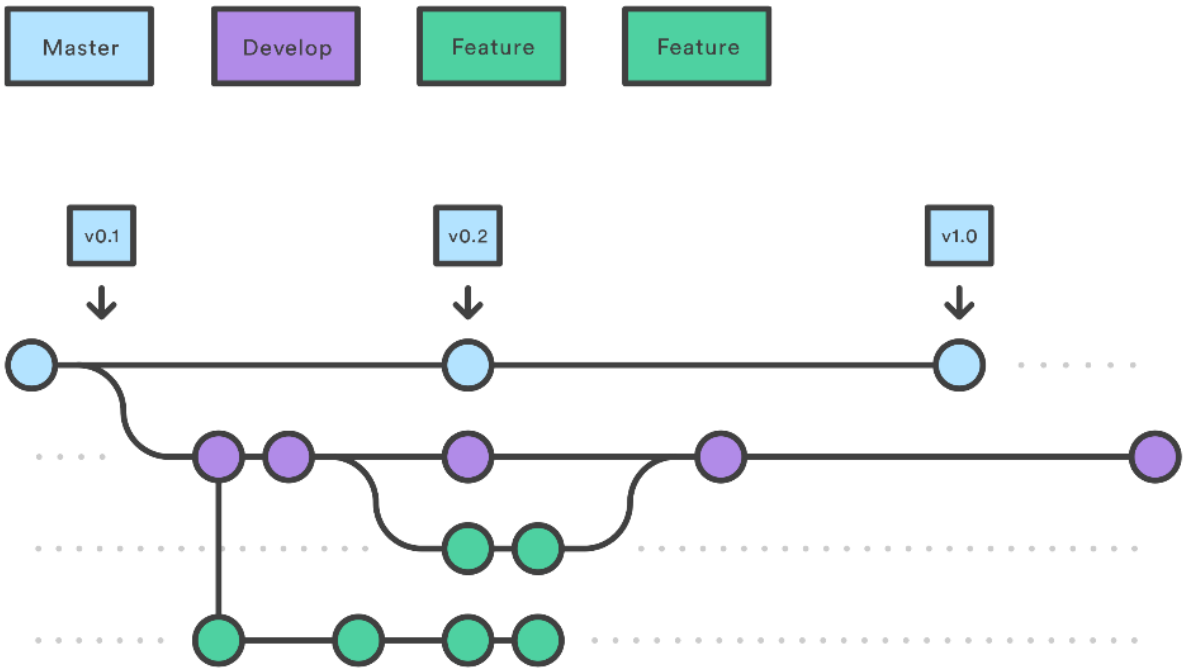


FIGURE 5 : GITFLOW (HTTPS://WWW.ATLASSIAN.COM)

IV. Analyse des risques

Pour l'analyse des risques qui va suivre, je vais me baser sur les moyens d'identifications suivants afin d'établir un tableau qui soit le plus clair possible :

- Le **type de risque** :
    - *Technique* : Architecture mal maîtrisée, technologies trop innovantes, manque d'expérience dans un langage, matériels, logiciels, configuration, performances, manque de décision entre plusieurs choix possibles, conception trop complexe, etc…
    - *Organisationnel* : Manque de coordination dans le projet, communication insuffisante entre acteurs, structures défaillantes, procédures incohérentes de gestion de projet, trop de responsables, etc…
    - *Fonctionnel* : IHM, ergonomie, services rendus, fonctionnalités, etc…
    - *Projet* : Quantité de moyens insuffisants, ressources humaines inappropriées, problème de pilotage, planification mal conçue, etc…
    - *Règlementaire* : Evolution de la règlementation en vigueur, législation sur l'environnement, connaissance des codes et règlements insuffisants, etc…
    - *Economique* : Législations fiscales, taux de changes, inflation, etc…
    - *Humain* : maladie, décès, etc…
    - *Environnementaux* : catastrophes naturelles, incendies, inondations, etc…
- **Probabilité d'apparition** :
    - Faible
    - Moyenne
    - Forte
    - Très forte
- Quantification du **niveau d'impact** :
    - Mineur
    - Moyen
    - Important
    - Majeur

- Calcul de **criticité des risques** avec leur niveau d'impact :
    - Risque faible
    - Risque à suivre
    - Risque à réduire
    - Risque inacceptable

Niveau d'impact

Probabilité d'apparition	Mineur	Moyen	Important	Majeur
Faible				
Moyenne				
Forte				
Très forte				

Tableau de l'analyse des risques :

Risque	Conséquence	Type	Probabilité	Impact	Criticité
Inondation	Arrêt des PC, destruction de notes papiers et du mobilier, pannes, perte des travaux effectués depuis la dernière sauvegarde	Technique	Faible	Majeur	À suivre
Décès	Augmentation de la durée ou arrêt du projet, potentielle influence sur le moral de l'équipe	Humaine	Faible	Important	Faible
Maladie	Augmentation de la durée du projet	Humaine	Moyenne	Moyen	À suivre
Utilisation de logiciels sans licence	Amendes et dégradation de l'image de marque	Règlementaire	Faible	Faible	Faible
Altération accidentelle des données	Perte des travaux effectués depuis la dernière sauvegarde	Technique	Moyenne	Important	À suivre
Vol d'équipement informatique	Matériel à racheter et à reconfigurer	Projet	Faible	Important	Faible
Bogue dans une application	Arrêt de l'utilisation du logiciel	Fonctionnel	Forte	Important	À réduire
Effacement de données et/ou de configurations par un virus	Perte des travaux effectués depuis la dernière sauvegarde	Technique	Moyenne	Important	À suivre
Perte de l'accès à nos données sur Notion	Frein à l'avancée du projet	Technique	Faible	Important	Faible
Perte de l'accès à nos serveurs	Frein ou blocage du projet	Technique	Faible	Majeur	À réduire
Algorithme peu fiable	Perte de temps, perte de la fidélisation de clients	Fonctionnel	Forte	Majeur	Inacceptable

V. Objectifs fixés initialement

L'objectif fixé initialement est d'avoir une interface fonctionnelle, un algorithme fonctionnel qui retourne les différents articles, c'est-à-dire réussir à déterminer les différentes actualités et à en extraire le titre, la date et l'URL. L'objectif initial prend aussi en compte la création du plugin, sa suppression, et son attribution au compte de l'utilisateur.

La date limite pour le développement de la première version de l'outil est fixée au 30 juin pour une réunion de démonstration le 1er juillet. À la suite de cette réunion, une période de test, d'affinage et d'optimisation de l'algorithme durera deux semaines, jusqu'au 16 juillet, suivi d'une période de deux semaines également, jusqu'au 30 juillet, avec l'équipe de production pour vérifier le bon fonctionnement de l'application en conditions réelles. Cette deuxième période de test ne peut débuter sans que les tests unitaires, les tests fonctionnels et les tests d'IHM ne passent avec succès. En effet, le mois de juillet est utilisé pour améliorer notre algorithme et pour mettre en place des métriques permettant de mesurer sa fiabilité suivant les cas rencontrés, il est donc indispensable que les tests préalables soient passés et validés. Les métriques seront rédigées sur un tableur Google Sheets disponible en annexe.

Au niveau de ces métriques, l'objectif initial est que notre outil fonctionne sur 65% des sites minimum.

L'intégration de Plugins Auto sur Mytwip ne fait pas partie du périmètre initial du projet.

Analyse des principaux choix

I. Matériel

Chaque développeur possède un PC fixe au bureau de Coexel, mais en raison du télétravail nous travaillons tous les deux sur des ultrabooks dimensionnés convenablement pour pouvoir réaliser toutes les tâches habituelles, à savoir 16 Go de RAM, un SSD de 512 Go et un processeur disposant de 8 cœurs. En cas de besoins nous utilisons une application de bureau à distance pour accéder aux ressources présentes sur nos PC fixes et ainsi maximiser les ressources disponibles pour les tâches plus lourdes qui doivent tourner en tâche de fond la nuit, et ne pas avoir à le faire chez nous.

Nous utilisons les serveurs habituels car la nouvelle application sera intégrée à Mytwip, et parce que nous aurons besoin d'accéder aux données propres à nos clients. Il n'y a donc pas besoin d'une nouvelle infrastructure pour le futur déploiement de ce projet.

II. Méthodologies

Nous avons choisi la méthodologie Agile pour son cycle incrémental et son processus d'amélioration continue du produit. Comme précisé en partie Problématique – III, l'équipe de production de Coexel fera office de clients dans les tests de l'application, et ce seront eux qui nous feront les retours après tests, que nous inclurons dans notre fichier de retour d'expérience sur Notion, afin de les ajouter sous forme de nouvelles user stories dans le product backlog et revoir notre planification des tâches et la répartition des ressources au besoin.

Parmi les frameworks Agiles, nous avons choisi Scrumban (mélange de Scrum et Kanban), toujours sur l'outil Notion, pour sa facilité de déploiement, d'interprétation et de visualisation. Nous sommes une petite équipe qui n'a pas beaucoup d'expérience en Agile, il était donc important d'utiliser une méthode qui soit la plus visuelle et concise possible, et Kanban s'y prête parfaitement, d'autant plus qu'elle permet de plus se focaliser sur les tâches que sur les processus, ce dont nous avons l'habitude à Coexel, et qui présente une parfaite méthodologie de transition. Notre méthodologie a aussi un aspect Srum parce que nous planifions les tâches sur des sprints de deux semaines, et à la fin d'un de ces sprints, nous consultons

collectivement le Kanban pour déterminer quelles seront les tâches à réaliser pour les deux semaines suivantes.

De même que pour le GitFlow (figure 5), celui-ci a été choisi parce qu'il est simple à mettre en place et à comprendre pour des personnes novices avec Git. En effet, Coexel a adopté Git depuis moins de cinq ans, et ne l'a pas encore beaucoup expérimenté en raison de l'architecture lourde de Mytwip. Notre choix s'est porté sur ce workflow aussi en raison du fait que je l'ai déjà utilisé dans divers projets personnels et scolaires.

Le principe du GitFlow est le suivant : une branche *develop* est créée depuis la branche *master*, et des branches *feature* seront créées depuis la branche *develop*. Une branche *feature* sera créée pour développer une fonctionnalité précise. Une fois la fonctionnalité terminée et testée, elle sera fusionnée avec la branche *develop* qui contiendra les évolutions précédentes. À la fin de chaque sprint, et dès que les vérifications nécessaires du bon fonctionnement de l'existant sont passées, la branche *develop* est fusionnée avec la branche *master*, et constitue donc un livrable à faire tester par notre équipe de production. Pour commencer un nouveau sprint, nous retournons sur la branche *develop*, créons les branches *feature* nécessaires et ainsi de suite jusqu'à la fin du projet.

III. Outils

Nous avons utilisé Notion pour la rédaction de tous les documents du projet, ainsi que la gestion de projet : Product Backlog, Kanban, planification, comptes-rendus de réunions, dossier de conception, document de retour d'expérience (retex*), etc. (figure 6).

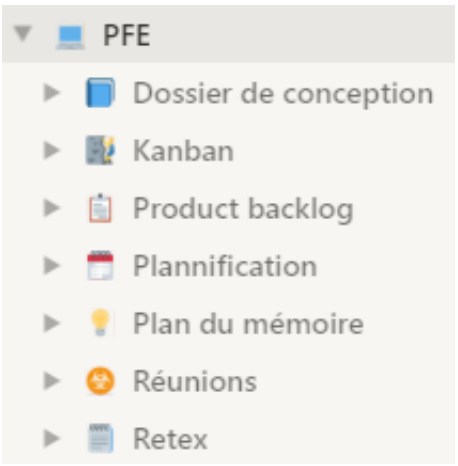


FIGURE 6 : ENSEMBLE DES DOCUMENTS DU PROJET SUR NOTION

Cet outil nous permet de tout centraliser sur une même plateforme collaborative pour éviter de nous disperser sur divers outils. L'avantage de Notion est qu'il s'agit d'un outil collaboratif sur lequel on peut partager des documents à un ensemble d'individus, et gérer les privilèges de ceux-ci en fonction de leur rôle dans le projet. Par exemple l'équipe de développement aura les droits d'édition sur le product backlog, privilège que n'aura pas l'équipe de production, et l'équipe de développement n'aura pas le privilège d'édition sur le Kanban, qui sera réservé au product owner et au chef de projet.

Un autre avantage de Notion est qu'il propose une mise en forme unifiée des documents avec un ensemble d'outils proposés, dont des images, des fichiers sous différents formats (PDF, Word, audios, etc.), et qu'il est compatible avec des services externes tels que Figma, que j'ai utilisé pour réaliser la charte graphique et les maquettes du projet (voir annexe 3). Cette compatibilité permet d'afficher le contenu réalisé sur ces outils sur Notion sans avoir à faire de captures d'écran (par exemple), et augmente notre productivité.

Notion est aussi compatible avec Windows, les systèmes GNU/Linux, MacOS, Android et iOS, ce qui permet de partager les documents avec un maximum de personnes et de pouvoir les consulter depuis n'importe où et n'importe quand, l'accessibilité étant un argument important dans un contexte de télétravail.

L'inconvénient de n'utiliser qu'un seul outil est que cela crée une dépendance au bon fonctionnement de cette plateforme. Autrement dit, si le service vient à ne plus fonctionner, nous ne pouvons plus l'utiliser le temps qu'il soit remis en service, et pourrait potentiellement geler l'avancée du projet, sans compter le risque de perte de données.

Pour pallier ce désagrément, nous faisons chaque semaine une sauvegarde de nos documents sur notre NAS* pour toujours y avoir accès, mais sans pouvoir interagir avec leur contenu. Ce qui constitue une contrainte, mais qui est bien plus acceptable que de perdre l'accès à nos données, ou de les perdre lors d'une erreur de maintenance sur la plateforme. Nous avons donc fait en sorte de conserver l'intégrité des données.

IV. Techniques

L'application sera déployée sur un serveur Apache 2 supportant PHP 7. L'application devra donc être développée en langage PHP pour la partie back-end, tout en s'appuyant sur notre scraper JavaScript et le serveur Node.js associé.

Côté front-end*, dans la même optique d'évolutivité et de maintenance vis-à-vis de l'existant, les technologies employées devront être natives dans la mesure du possible, mais jQuery étant présente dans l'existant, ce sera la seule librairie JavaScript autorisée, d'autant plus qu'elle facilite l'implémentation des appels AJAX sur notre API interne.

Pour le CSS nous utiliserons Bootstrap* car nous souhaitons l'intégrer à Mytwip et il nous semble juste de commencer à l'intégrer avec ce projet.

L'ensemble de nos applications utilisent le SGBD* MySQL, ce sera donc celui-ci que nous allons utiliser pour les transactions avec les bases de données.

Nous utiliserons le serveur Node.js qui a été mis en place lors des précédentes années avec notre web scraper basé sur Puppeteer. Cet ensemble couplé à un script PHP est celui qui est chargé de récupérer les données sur les sites web une fois les plugins créés (voir annexe 1).

Le fait d'utiliser des technologies qui soient les plus natives possibles, nous permet dans nos développements de maîtriser au maximum nos algorithmes, de les concevoir dans une démarche d'écoconception*, et de maîtriser le poids de notre application.

V. Normes employées

Pour simplifier les développements, au sein de Coexel nous utilisons en grande majorité les normes* et standards* établis par la communauté des langages et technologies utilisés, en plus de ceux que nous avons en interne. Nous avons choisi d'utiliser ceux déjà existants pour nous éviter de prendre du temps à établir les nôtres, mais aussi pour anticiper le fait que de futurs développeurs pourront travailler sur ce projet à l'avenir pour y apporter des évolutions, et plus nous utilisons des normes et standards déjà existants, plus facile et rapide sera l'intégration des développeurs sur ce projet.

a. Langage PHP

Nous allons suivre les recommandations des PSR* suivantes :

- PSR-1 : Basic Coding Standards https://www.php-fig.org/psr/psr-1/
- PSR-2 : Coding Style Guide https://www.php-fig.org/psr/psr-2/
- PSR-5 : PHPDoc Standard https://github.com/php-fig/fig-standards/blob/master/proposed/phpdoc.md
- PSR-12 : Extended Coding Style https://www.php-fig.org/psr/psr-12/

- PSR-19 : PHPDoc tags https://github.com/php-fig/fig-standards/blob/master/proposed/phpdoc-tags.md

Ces PSR donnent les standards pour les conventions de nommages, les standards d'indentation et l'écriture de la documentation PHP.

Pour la nomenclature que nous utilisons à Coexel, lorsque nous travaillons sur notre API, par exemple lorsque nous souhaitons supprimer un plugin, nous appelons la route */delete-plugin*, et cette route déterminée dans un fichier JSON va appeler la fonction qui y est associée : *deletePlugin()*. Celle-ci sera chargée de gérer les paramètres POST ou GET qui lui sont envoyés et d'appeler la fonction du même nom mais préfixée d'un underscore (_) qui elle, exécutera la commande de suppression du plugin. Donc la fonction non préfixée de l'underscore se chargera de gérer les paramètres envoyés via l'API, et la fonction préfixée sera appelée par la fonction précédente qui lui transmettra les paramètres POST ou GET afin d'exécuter l'action souhaitée.

b. Langage JavaScript

Pour ce langage nous utiliserons également des normes et standards établis par l'organisation ECMA International* au travers de l'ECMA Script* dont je vais décrire les principales utilisées ici :

- Convention de nommage :
    - camelCase (variables et fonctions)
    - Les variables globales écrites en majuscules
- Indentation : 4 espaces
- Utiliser le mot clé `const` pour toutes les variables dont la valeur ne sera pas amenée à être modifiée.
- Eviter le plus possible d'utiliser le mot clé `var`.
- Utilisation des guillemets simples le plus possible pour des raisons d'écoconception.
- Documentation explicative sur des portions de code : utilisation du double slash `//`
- Documentation sur les fonctions : voir figure 7

```
/**
 * Represents a book.
 * @constructor
 * @param {string} title - The title of the book.
 * @param {string} author - The author of the book.
 */
function Book(title, author) {
}
```

FIGURE 7 : DOCUMENTATION D'UNE FONCTION (HTTPS://JSDOC.APP/)

En interne nous avons juste une spécification qui est de préfixer d'un $ le nom des variables JavaScript qui stockent un objet jQuery, c'est-à-dire un nœud du DOM, mais façon jQuery. De la même manière, les variables booléennes sont préfixées par *is*, par exemple, pour savoir si un plugin a bien été supprimé, la variable associée est nommée *isDeleted*.

c. Langage HTML

Nous vérifions la bonne conformité du document HTML avec le W3C Validator : https://validator.w3.org/

Convention de nommage : kebab-case (class, id, data-attribute)

Indentation : 4 espaces

d. Langage CSS

Ici nous utilisons les standards utilisés par WordPress qui sont les plus communes dans l'écriture de fichiers CSS : https://make.wordpress.org/core/handbook/best-practices/coding-standards/css/

Solutions proposées

I. Les différentes fonctionnalités

Comme précisé plus tôt dans le mémoire, nous développons une Single Page Application parce que notre application va être intégrée à Mytwip. Cela permet également une meilleure ergonomie et une meilleure prise en main par l'utilisateur, surtout que le nombre de fonctionnalités nous permet de tout mettre facilement sur une même interface.

Dans cette partie je vais exposer les différentes fonctionnalités de la partie front-end, de l'algorithme et de la partie back-end. Je tiens à séparer le fonctionnement de l'algorithme du front-end et du back-end parce que son fonctionnement est ce qui nous intéresse le plus dans ce projet.

a. Fonctionnalités front-end

L'interface est découpée en deux parties distinctes : le quart de la page à gauche est réservé aux fonctionnalités pour l'utilisateur, le reste est occupé par l'iframe qui affiche le site renseigné par le client. Pour avoir un aperçu de l'IHM, voir en annexe 3.

Les fonctionnalités de la partie gauche sont les suivantes, de haut en bas et de gauche à droite :

- Un champ texte permettant de saisir le site souhaité pour qu'il soit affiché dans l'iframe.
- Un spinner qui renseigne à l'utilisateur le fait que l'iframe est en train de charger la page voulue.
- Un bouton d'exécution de l'algorithme, une fois que l'utilisateur a sélectionné dans l'iframe la zone dans laquelle se trouve les articles qu'il souhaite extraire.
- Un bouton de réinitialisation qui offre la possibilité de réinitialiser le contenu sélectionné et retenu en mémoire pour la création du plugin, si jamais l'utilisateur n'a pas sélectionné ce qu'il souhaitait.
- Un bouton permettant de supprimer le contenu empêchant l'accès aux articles, comme les pop-up RGPD.
- Une zone permettant d'afficher un aperçu du flux RSS grâce aux informations récupérées par l'algorithme.

- Un champ de sélection déroulant qui permet au client de choisir la thématique de son espace utilisateur à laquelle il veut associer le flux RSS du plugin.
- Un bouton de validation permettant de créer le plugin.
- Un bouton de suppression du plugin offrant la possibilité de le supprimer si jamais il ne convient finalement pas au client, et d'un autre côté évite de saturer l'espace disque du serveur au minimum.

b. Fonctionnalités back-end

Les fonctionnalités du back-end sont pour beaucoup développées en PHP et sont accessibles depuis notre API grâce à des appels AJAX. D'autres sont accessibles via notre API Node.js pour accéder aux fonctionnalités permettant de manipuler les plugins du côté serveur.

Les fonctionnalités back-end sont les suivantes :

- Récupération du code source du site souhaité pour l'afficher dans l'iframe. Ici il est nécessaire de noter que nous n'affichons pas le site directement une fois celui-ci saisi dans le champ prévu à cet effet. Il nous faut au préalable passer par un de nos programmes qui va récupérer le code source de la page, désactiver les éléments cliquables, ainsi que lui injecter des dépendances supplémentaires, dont jQuery pour nous permettre d'interagir avec le contenu pour faire fonctionner notre algorithme. C'est pour cette raison qu'il existe un délai de chargement du contenu représenté par le spinner dans le front-end.
- Récupération des thématiques du client en fonction de son espace.
- Création du plugin sur le serveur.
- Association en base de données du plugin à la thématique sélectionnée.
- Suppression du plugin sur le serveur.
- Suppression en base de données du lien entre le plugin et la thématique sélectionnée une fois que celui-ci est supprimé du serveur.

c. Fonctionnalités de l'algorithme

Le rôle principal de notre algorithme est d'extraire les informations des articles d'actualités présents dans la zone sélectionnée par l'utilisateur. Pour cela, il faut donc envoyer en entrée de l'algorithme cette zone que nous appellerons « *conteneur* ». À partir de cette entrée, il va réaliser un certain nombre d'étapes et d'opérations pour retourner à l'utilisateur un aperçu du flux RSS à partir des éléments récupérés. Toutes les étapes décrites se passent dans

l'iframe, à l'exception de l'affichage de l'aperçu du flux RSS, qui sera affiché dans une zone spécifique de notre application.

Ces traitements sont les suivants :

1. Repérage des différents liens dans le conteneur, et on leur crée un attribut data-code qui contient le chemin CSS du lien.
2. On regroupe dans des tableaux les éléments présentant des attributs data-code en commun, c'est-à-dire qu'il y aura autant de tableaux que de data-code différents.
3. On parcourt chaque tableau de data-code pour essayer de remonter le plus haut dans le DOM, dans la limite du conteneur, afin de trouver leur parent commun le plus haut (en jaune dans la figure 8). C'est-à-dire que nous remontons progressivement afin que le nombre d'éléments ne soit plus égal au nombre d'éléments disposant d'un data-code. Généralement ce nombre est égal à 1, car il s'agit du parent commun le plus haut dans le DOM, comme expliqué précédemment. Ce fonctionnement est illustré dans la figure 8 ci-dessous. Lorsque nous rencontrons ce parent commun, nous redescendons d'un niveau dans le DOM pour retenir finalement l'élément qui détermine les différents articles (en vert dans la figure 8), étant donné que tous les articles, dans une grande majorité de cas, possèdent une structure commune. Notons toutefois que dans le cas schématisé dans la figure 8, le conteneur n'est pas la balise *ul* (liste désordonnée), mais qu'il se peut que cet élément soit celui cliqué par l'utilisateur, ce qui n'influe en rien sur le résultat final.

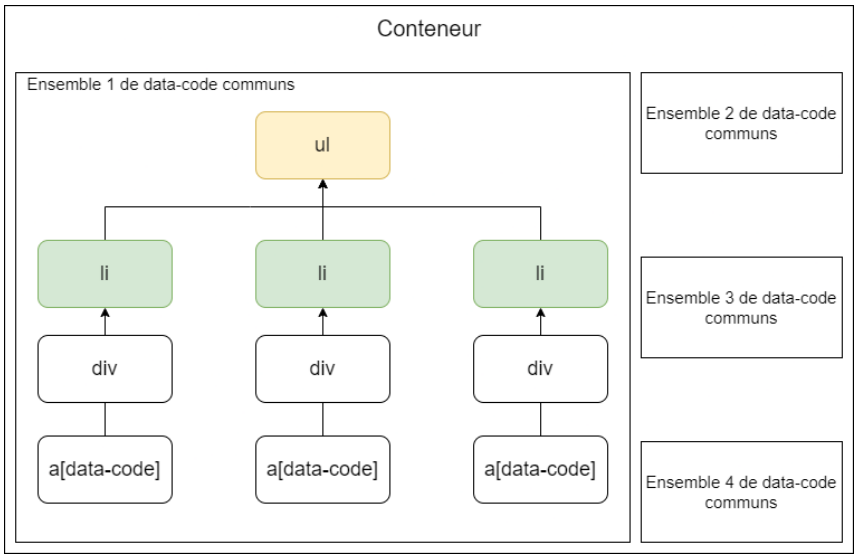


FIGURE 8 : REMONTEE DANS LE DOM A LA RECHERCHE DU PARENT COMMUN LE PLUS HAUT

4. Lorsque l'algorithme est parvenu à extraire chaque délimiteur d'article, il leur attribue le même data-code que leur lien. Ces éléments sont appelés « *candidats* ».
5. Ensuite ces tableaux de candidats sont tous stockés dans un tableau commun pour qu'ils puissent être triés par ordre décroissant de nombre de candidats. Le tableau de candidats retenu est celui en ayant le plus. Celui-ci sera appelé « *articles* », même s'il y a une probabilité que celui-ci renvoie autre chose, soit dû à une mauvaise manipulation de l'utilisateur, soit dû à une erreur dans l'algorithme, ou alors une page web mal structurée. Cette probabilité reste relativement faible et pour faciliter les explications de cette partie, nous partons du principe que ce tableau renvoie bien les articles souhaités. Chaque article est encadré en rouge dans l'iframe, comme sur la figure 9, pour que l'utilisateur ait un retour visuel. La zone en gris foncé est le conteneur, encore une fois pour que l'utilisateur ait un aperçu de la zone cliquée.

FIGURE 9 : RETOUR VISUEL DANS L'IFRAME

6. L'algorithme parcourt ce tableau d'articles, et pour chaque article il va analyser le contenu de son DOM et de son contenu textuel pour extraire le titre, l'URL et la date, s'il y en a une, ainsi que leur chemin CSS. Nous ne récupérons pas le résumé même si celui-ci est présent parce que cela représente une complexité relativement grande que de différencier le contenu textuel du titre et celui du résumé. D'autant plus que nous

avons lors de chaque session de crawl* des programmes qui se chargent de récupérer le contenu textuel intégral des articles depuis leur URL, ce qui nous permet dans un second temps de reconstituer le résumé avec un nombre de caractères restreint.

7. Une fois que nous avons récupéré les chemins CSS des composants des articles, nous les stockons dans un objet appelé « *cssComponents* », et le contenu du titre, de la date et de l'URL sont stockés dans un tableau d'objets « *articlesContent* ». Ces objets sont ensuite récupérés depuis le contexte de la page principale de notre application.
8. L'objet *cssComponents* est récupéré pour être envoyé à notre API Node.js pour la création du plugin, et le tableau *articlesContent* est récupéré pour constituer un aperçu du flux RSS pour l'utilisateur, voir figure 10 ci-dessous.

FIGURE 10 : RECONSTITUTION DES ACTUALITES DU SITE DE COEXEL SUR L'INTERFACE DE NOTRE APPLICATION

Remarque :

Il est aussi possible pour l'utilisateur de sélectionner le corps d'un article, ce qui aura pour conséquence de sélectionner l'ensemble des éléments possédant le même chemin CSS que celui sélectionné. Dans le traitement n°3, le nombre d'éléments considérés comme conteneurs sera supérieur à 1 dans ce cas, et le programme considérera chaque élément comme étant un article. Nous avons fait le choix de laisser cette opportunité à l'utilisateur pour lui offrir plus de flexibilité dans son utilisation. La suite du traitement reste la même.

II. Développement

a. Déroulé du projet

Comme précisé dans la partie *Analyse des principaux choix - II* nous avons choisi de travailler les fonctionnalités sur des périodes de deux semaines. J'ai été seul à développer pendant ces sprints, mais Damien Lepelletier était disponible pour m'apporter de l'aide sur les tâches les plus exigeantes, ainsi que des conseils. À la fin de chaque sprint l'équipe de production teste l'application pour apporter ses retours pour que je puisse améliorer le livrable.

Sprint 0 :

Nous avons commencé le projet avec un sprint 0, de deux semaines, réservé à la préparation du travail avec la rédaction d'un dossier de conception, mais aussi pour rédiger un *product backlog* avec les différentes *user stories* (voir annexe 5), créer le Kanban sur Notion et mettre en place une première ébauche de planification pour la réalisation du projet avec les différentes échéances.

Sprint 1 :

Lors du sprint 1 nous nous sommes concentrés sur la visualisation du site web dans l'iframe, ainsi que sur la sélection du conteneur.

Sprint 2 :

L'algorithme principal avait été commencé par Damien Lepelletier quelques années auparavant avant d'être mis de côté par manque de temps pour le poursuivre. Je l'ai récupéré pour l'adapter à notre besoin et ajouter des fonctionnalités.

Sprint 3 :

Lors de ce sprint j'ai pu récupérer des fonctionnalités que j'avais développées l'année précédente dans un autre projet. Il s'agit de la validation du plugin, c'est-à-dire la récupération des chemins CSS pour créer le plugin sur le serveur, mais aussi la fonctionnalité de réinitialisation des éléments sélectionnées, que ce soit les chemins CSS ou le conteneur. Bien entendu j'ai dû adapter ces fonctionnalités aussi à nos cas d'usage.

Sprint 4 :

Mise en place de la suppression du plugin sur le serveur, avec la création des points API nécessaires.

Sprint 5 :

Ici je me suis focalisé sur l'association du plugin à la thématique du client lors de la création du plugin, et sur la suppression du lien entre celui-ci et la thématique sélectionnée.

Sprint 6 :

Il s'agit de la période de test de deux semaines que je réalise seul pour affiner et optimiser l'algorithme pour qu'il corresponde plus à la diversité de sites et de cas que nous pouvons rencontrer en utilisation réelle.

Sprint 7 :

Sprint réalisé avec l'équipe de production qui m'aide à tester le produit en conditions réelles, et me faire un retour entre le nombre de sites pour lesquels l'application a réussi à créer le plugin souhaité et le nombre de plugins où il a échoué. Ceci me permet de mesurer la performance et la fiabilité de l'application, mais aussi de voir quels cas sont problématiques et continuer à le faire évoluer.

b. Sécurité

Dans le contexte de Plugins Auto, différentes failles de sécurité peuvent être présentes étant donné que nous faisons appel à notre API qui est indépendante de notre logique métier, les points API peuvent être appelés depuis presque n'importe où depuis l'application, et avec une couche de sécurité, ou pas. Dans notre cas nous avons veillé à ce que les accès nécessitent une authentification de la part du client. Cette authentification se traduit de deux manières différentes à Coexel. Nous avons une liste d'adresses IP de confiance qui peuvent utiliser certaines fonctionnalités, typiquement celles qui ne sont pas définies dans Mytwip et qui représentent des tâches internes à l'entreprise, pour nous assurer que seuls les membres de l'équipe puissent les utiliser. L'autre moyen est l'utilisation d'un jeton ou *token*. Ce jeton est généré lorsqu'un client se connecte à Mytwip, il lui est propre et possède une date d'expiration. Ce moyen nous permet de nous assurer que pour utiliser la fonctionnalité, un utilisateur doit se connecter à son espace Mytwip.

Mais pour utiliser une fonctionnalité, il est nécessaire d'avoir un jeton valide, de connaître le chemin de l'API ainsi que les paramètres adéquats. Ce qui signifie qu'une personne ayant accès à ces informations sans posséder de compte Mytwip, par exemple en « volant » un jeton, peut avoir accès à certaines fonctionnalités. Des routines de contrôle se chargent d'analyser les logs d'utilisation et de détecter ces appels frauduleux en comparant le contexte d'appel, la durée de session et les adresses IP utilisées. Dans ce cas-là, le jeton est désactivé et un mail d'alerte est envoyé aux administrateurs du système.

Du côté de l'interface, pour éviter qu'un utilisateur non averti ne joue avec les différents boutons (Valider, Supprimer, Réinitialiser), je les ai désactivés à des moments stratégiques en fonction des actions réalisées précédemment. Ceci permet également de guider l'utilisateur lors de son parcours d'utilisation, voir figure 11 ci-dessous.

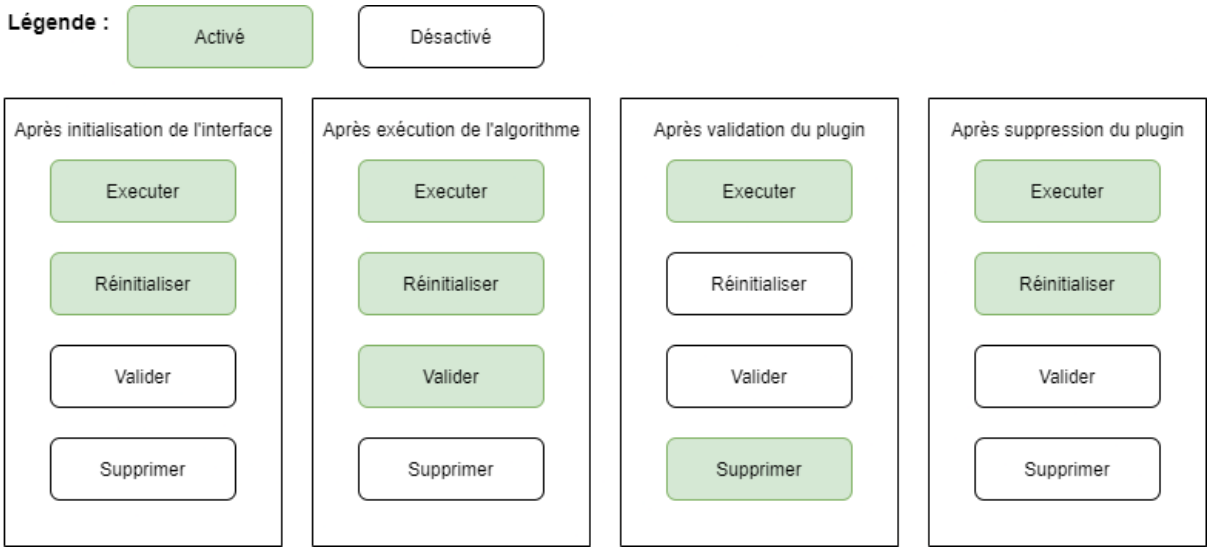


FIGURE 11 : ETAT DES BOUTONS EN FONCTION DE L'ETAPE

Nous sommes conscients du fait qu'un utilisateur averti pourra enlever dans le DOM de la page, via la console de son navigateur, les attributs *disabled* des boutons. C'est pour cela que le programme opère d'autres contrôles en arrière-plan.

Du côté des injections SQL, beaucoup de contrôles passent par l'API également, étant le point d'entrée de nos fonctionnalités. Sur notre API nous avons toute une série de contrôles notamment sur le format des données envoyées. La quasi-totalité des paramètres envoyés sont des entiers positifs, il est donc aisé de contrôler si ceux-ci sont des chaînes de caractères, à l'instar des requêtes SQL ou de potentielles données frauduleuses.

Du côté applicatif, dans Plugins Auto le client a la possibilité de créer un plugin avec les informations récupérées. Pour éviter qu'il ne clique plusieurs fois sur le bouton de validation, et ainsi créer plusieurs fois le même plugin et saturer l'espace disque de notre serveur. Pour gérer cette problématique nous utilisons une variable booléenne *pluginValidate* ayant la valeur « faux » par défaut, et qui stockera la valeur « vrai » lorsque le plugin aura été validé par l'utilisateur et créé sur le serveur. De cette manière le programme saura que lorsque cette valeur sera à « vrai » le client ne pourra pas valider le plugin de nouveau, voir figure 12, page suivante. Pour utiliser de nouveau cette fonctionnalité il devra soit réinitialiser ses choix, soit supprimer le plugin avec la fonctionnalité accessible, soit soumettre un nouveau site web dans le champ prévu à cet effet.

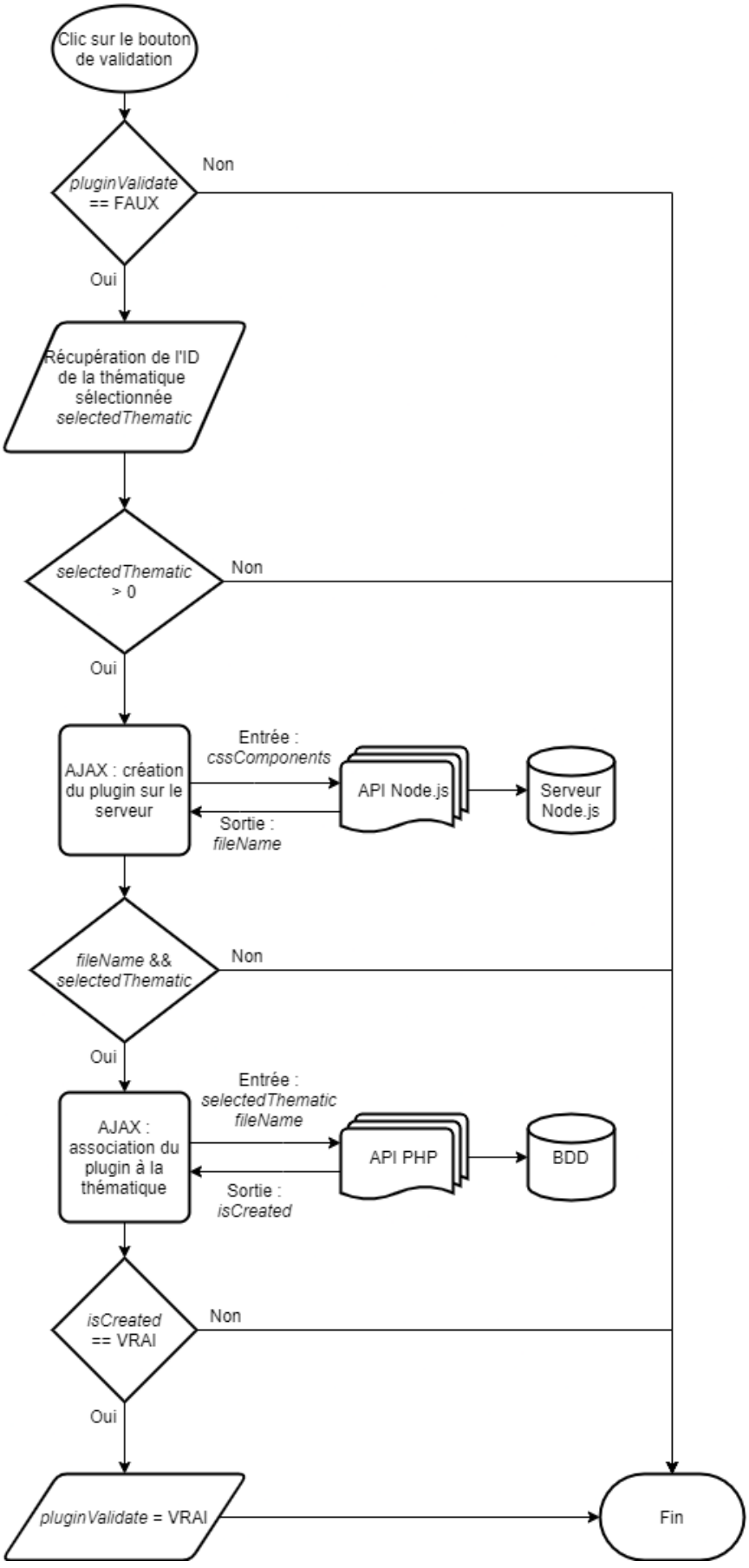


FIGURE 12 : PROCESSUS DE VALIDATION DU PLUGIN

Nous avons également une problématique sur la suppression du plugin. Nous devons nous assurer que l'utilisateur ne puisse supprimer que le plugin qu'il vient de créer et pas d'autres plugins. En effet, il pourrait accéder à notre API avec un jeton et tenter d'envoyer des paramètres pour supprimer des plugins à la chaîne, mais dans notre API, nous contrôlons le fait qu'un plugin ne peut être supprimé qu'une demi-heure après sa création, ce qui limite l'impact d'une telle action. D'un autre côté, la fonction PHP appelée par notre API fait aussi appel à notre API Node.js en bout de chaîne, car c'est celle-ci qui supprime le plugin du serveur, voir figure 13 ci-dessous. Ici aussi nous avons une couche de sécurité, seules des adresses IP de confiance, à savoir celles de certains de nos serveurs, peuvent y accéder, donc un utilisateur externe ne peut faire appel à notre API sans être connecté à notre serveur.

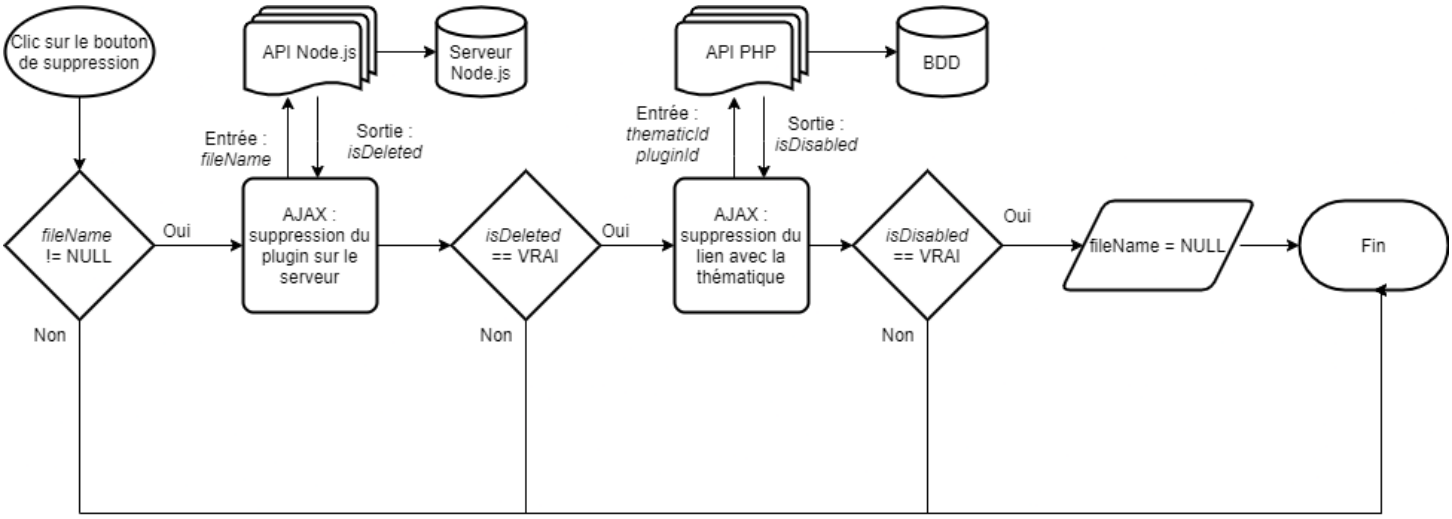


FIGURE 13 : FONCTIONNEMENT DE LA SUPPRESSION D'UN PLUGIN PAR APPEL API

c. Tests

Au cours de ce projet nous avons réalisé différents tests logiciels avant de nous lancer dans les tests en conditions réelles. À Coexel il n'y a pas de pipeline CI/CD*, nous sommes libres d'utiliser le framework de tests unitaires que nous souhaitons. Dans mon cas j'ai écrit les tests unitaires à la main et j'ai suivi les principes de TDD (Tests Driven Development), à savoir écrire les tests, puis passer au code afin de n'écrire que le minimum de code possible pour faire fonctionner les tests, ce qui permet de gagner du temps et de mieux optimiser les fonctionnalités concernées par ces tests.

Les tests ont été menés dans l'ordre suivant :

1. *Tests unitaires* : pour tester les unités de code. Tous les tests unitaires doivent être passés avec succès avant de pouvoir passer aux tests d'intégration.
2. *Tests d'intégration* : pour tester si les unités de code fonctionnent bien ensemble. Tous les tests d'intégration doivent être passés avec succès avant de pouvoir passer aux tests fonctionnels.
3. *Tests fonctionnels* : simulation du comportement de l'utilisateur sur notre produit.

Tests unitaires

Les tests unitaires ont été principalement utilisés pour le module de conversion des dates, pour contrôler que le format des dates récupérées sur les articles est bien interprété, puis pour vérifier que la date, dans le format retenu, est bien convertie en un format de date compréhensible par l'application sous forme de chaîne de caractères.

Ces tests sont écrits dans un fichier à part faisant appel aux dépendances nécessaires pour tester les fonctions.

Liste des tests unitaires :

- Conversion des dates en chaînes de caractères exploitables par l'application.
- Réinitialisation des variables.

Tests d'intégration

Ces tests ont principalement fait l'objet d'observations sur nos fonctions AJAX. En effet la plupart de nos fonctionnalités dépendent d'appels AJAX et d'échanges avec le serveur à travers des fonctions de rappel* (ou *callback*). C'est-à-dire qu'une fois qu'une fonction a terminé son traitement, elle va lancer la fonction passée en argument qui va se charger d'exploiter le résultat de sortie de la première.

Nous utilisons des fonctions de rappel car les fonctions AJAX sont des fonctions asynchrones.

Liste des tests d'intégration :

- Association du plugin à la thématique du client après la création du plugin.
- Récupération des bons chemins CSS par l'application.
- Vérification de la suppression du plugin sur le serveur.

- Vérification de la suppression en base de données du lien entre le plugin et la thématique du client.
- Vérification de la bonne récupération des thématiques de l'espace de l'utilisateur courant dans le champ de sélection.

Tests fonctionnels

Sur notre interface nous utilisons une iframe, et les actions qui se déroulent dans celle-ci sont dans un autre contexte que la page parente de l'iframe. C'est-à-dire que l'iframe possède son propre contexte et qu'il est difficile d'automatiser ces tests via un automate. Nous avons donc réalisé ces tests à la main.

Liste des tests fonctionnels :

- Vérifier l'ouverture du site dans l'iframe lors de l'insertion de l'URL dans le champ texte.
- Vérifier le changement de couleur de l'élément choisi comme conteneur.
- Vérifier l'encadrement en rouge des articles repérés.
- Vérifier la désactivation des boutons Valider, Supprimer et Réinitialiser quand ils doivent l'être.
- Vérifier que le contenu affiché en tant qu'aperçu du flux RSS dans l'espace dédié corresponde bien au contenu des articles encadrés en rouge.
- Vérifier la bonne réinitialisation des éléments de l'IHM lors de l'appui sur le bouton de réinitialisation.
- Vérifier la suppression de la pop-up RGPD lors de l'appui sur le bouton prévu à cet effet.
- Vérifier la bonne construction des liens des articles.

III. Déroulé critique du projet

Au cours de ce projet j'ai dû utiliser plusieurs modules qui ont été développés et utilisés dans d'autres cas d'usage. Le problème est que ces modules n'ont pas forcément été conçus pour être utilisés dans d'autres contextes. En effet, j'ai dû réécrire partiellement certains d'entre eux pour qu'ils prennent en compte nos nouveaux besoins, ce qui a eu pour conséquence de me prendre plus de temps que nécessaire dans le développement de certaines fonctionnalités en plus du risque de porter atteinte au bon fonctionnement des autres programmes qui utilisent ces modules.

Je souhaite souligner ici l'importance des principes SOLID* et la mise en place d'une API indépendante de la logique métier. Ce dernier point est bien en place à Coexel et est utilisé quotidiennement. En revanche l'application des principes SOLID peut se faire dans les nouveaux développements, mais les choix de conception qui ont été faits en amont, dans les débuts de la société Coexel, ne les appliquent pas forcément et entraînent des complications lors de la réutilisation de modules par leur manque de flexibilité, d'ouverture aux extensions et le manque de délimitation du périmètre des implémentations en ce qui concerne leur rôle.

J'avais pour projet d'utiliser des principes d'écoconception dans ce projet, mais les problématiques liées à la conception des modules ont aussi représenté des contraintes de ce côté. J'ai pu néanmoins mettre en place certains de ces principes au niveau de l'écriture du code mais pas au niveau de l'architecture logicielle ou de l'optimisation des ressources chargées par le serveur.

Du point de vue de la gestion de projet, la flexibilité de Coexel dans ses processus m'a permis sans problème d'utiliser la méthodologie Agile, et l'ouverture d'esprit des collaborateurs s'est révélée utile pour que je leur explique pourquoi j'ai choisi cette méthodologie et comment la mettre en pratique. Cette flexibilité m'a aussi fait gagner beaucoup de temps dans ce projet lorsque par exemple j'avais besoin de l'aide de Damien Lepelletier ou alors de faire une démonstration à Vincent Boisard, product owner, sans perturber l'avancée d'autres projets.

Au cours de nos tests et à partir de notre premier prototype, nous avons pu observer que les sites qui disposent d'une pop-up RGPD*, représentant 47,3% (71 sites sur les 150) des sites testés, constituent une problématique car cette pop-up bloque l'affichage du contenu du site souhaité et ne permet pas à l'utilisateur d'interagir avec les articles contenus dans l'iframe, et

donc bloque la bonne utilisation de Plugins Auto. Suite à ce constat, nous avons dû retravailler notre produit pour ajouter une fonctionnalité supplémentaire permettant d'effacer cette pop-up. Mais lors de son implémentation, nous avons observé différents cas particuliers nécessitant notre attention. Nous avons pensé en premier lieu à intégrer un bouton permettant à l'utilisateur d'interagir avec le contenu de l'iframe, par exemple pour fermer les pop-ups indésirables. Compte tenu du fait que nous devions permettre à l'utilisateur d'interagir à la souris avec le contenu de l'iframe, nous ne pouvions mettre le bouton que dans l'iframe. Nous nous sommes donc confrontés à une problématique d'ergonomie :

- Est-ce logique de mettre un bouton d'interaction dans l'iframe qui est censée être réservée à l'affichage de la page souhaitée ?
- Ne risque-t-on pas de perdre l'utilisateur au niveau de la logique de l'interface ? Et donc de nuire à son expérience ?

Pour afficher ce bouton nous avons dû définir dans le CSS une position absolue (*absolute*) et un *z-index** maximum pour qu'il soit prioritaire au niveau de l'affichage sur la page. Nous devions donc opérer une insertion dans le DOM de la page souhaitée par l'utilisateur qui est une opération relativement coûteuse en ressources, et aussi supposer le fait qu'il se pourrait que le *z-index* de notre bouton vienne en concurrence avec celui d'un élément natif de la page en fonction du choix des développeurs de celle-ci.

Le but de notre produit est aussi de limiter au maximum l'interaction de l'utilisateur. Dans ce cas, nous rajouterions un élément d'interaction, de la complexité dans le code, de potentiels problèmes d'affichage du bouton, sans compter que nous lui autorisions à interagir avec les liens la page, ce qui pourrait entraîner l'utilisateur à oublier d'appuyer de nouveau sur le bouton pour désactiver ces interactions, ce qui le conduirait à cliquer sur des liens dans l'iframe, et se retrouver sur des pages différentes, sans possibilité de revenir en arrière d'une autre manière que de recharger l'intégralité de l'interface. Ceci réduirait considérablement l'expérience utilisateur.

C'est pourquoi nous avons opté pour une solution plus simple : un bouton sur la page parente qui permet de supprimer dans l'interface le contenu dont la position est définie en fixée (*fixed*) dans le CSS (voir figure 14). Cette fonctionnalité a supprimé la pop-up dans 100% des cas. Mais parmi ces cas, seuls 5 sites rencontraient le problème suivant : la barre de navigation verticale n'était pas affichée, ce qui empêchait de sélectionner des articles. Nous avons dû ajouter une

fonctionnalité lors du clic sur le bouton de suppression de pop-up pour forcer l'affichage de cette barre de navigation (voir figure 15).

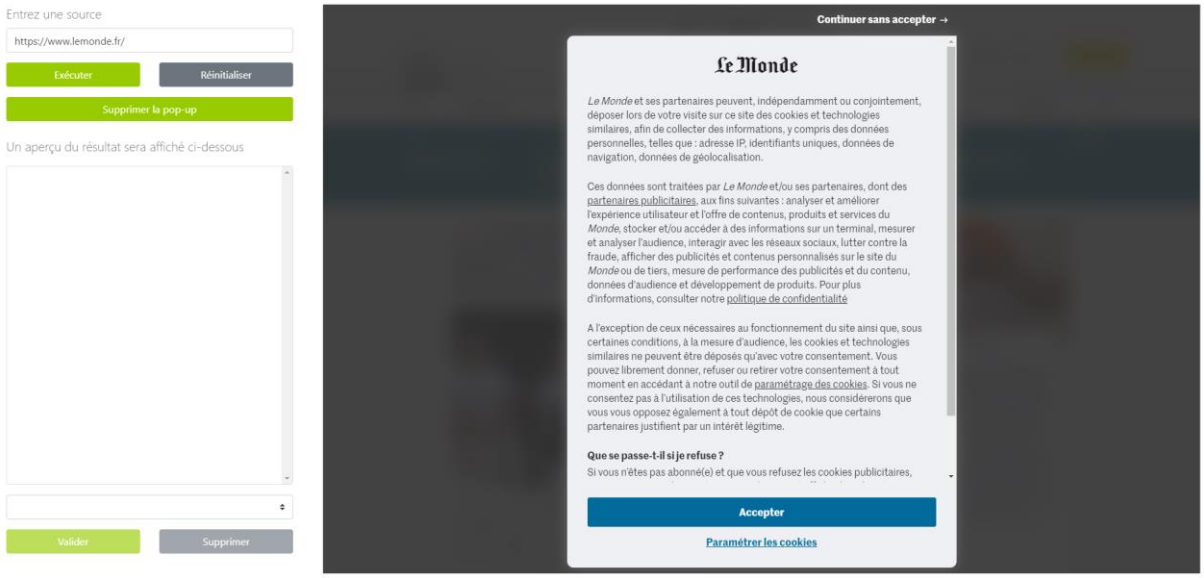


FIGURE 14 : AVANT LE CLIC SUR LE BOUTON DE SUPPRESSION DE LA POP-UP

FIGURE 15 : APRES LE CLIC SUR LE BOUTON DE SUPPRESSION DE LA POP-UP

Nous avons aussi remarqué des erreurs dans la construction des liens des articles. En effet il est récurrent de rencontrer sur des sites d'actualités, des articles dont le lien est présenté comme sur la figure 16 (voir le nœud « a » du DOM surligné en bleu) :

FIGURE 16 : CONSTRUCTION D'UN LIEN D'ARTICLE SANS NOM DE DOMAINE SUR LE SITE HTTPS://WWW.FRANCETVINFO.FR/

On peut voir que le lien en question n'est pas construit avec le nom de domaine. Mais lorsque que nous récupérons l'URL avec JavaScript, le langage est capable de la reconstruire avec la partie manquante à partir du nom de domaine de la page courante. Mais avec l'utilisation de Plugins Auto, les liens reconstruits l'étaient avec le nom de domaine de notre outil, à savoir « *http://api.mytwip.com/* », ce qui donnait par exemple le résultat suivant :

http://api.mytwip.com/*sante/maladie/coronavirus/pass-sanitaire/covid-19-les-regles-d-isolement-pour-les-cas-contacts-pourraient-evoluer-pour-les-personnes-vaccinees-suggere-olivier-veran_4709097.html*.

Donc le lien ne renvoyait pas vers l'article souhaité. Pour répondre à cette problématique nous avons dû récupérer à l'aide d'une expression régulière le nom de domaine et créer dans le DOM de Plugins Auto une balise HTML *base* ayant comme valeur dans l'attribut *href* le nom de domaine récupéré.

Le principe de cette balise étant de spécifier la base des URLs présentes dans les éléments enfants du DOM où elle est déclarée, voir sur la figure 17 l'élément surligné en bleu :

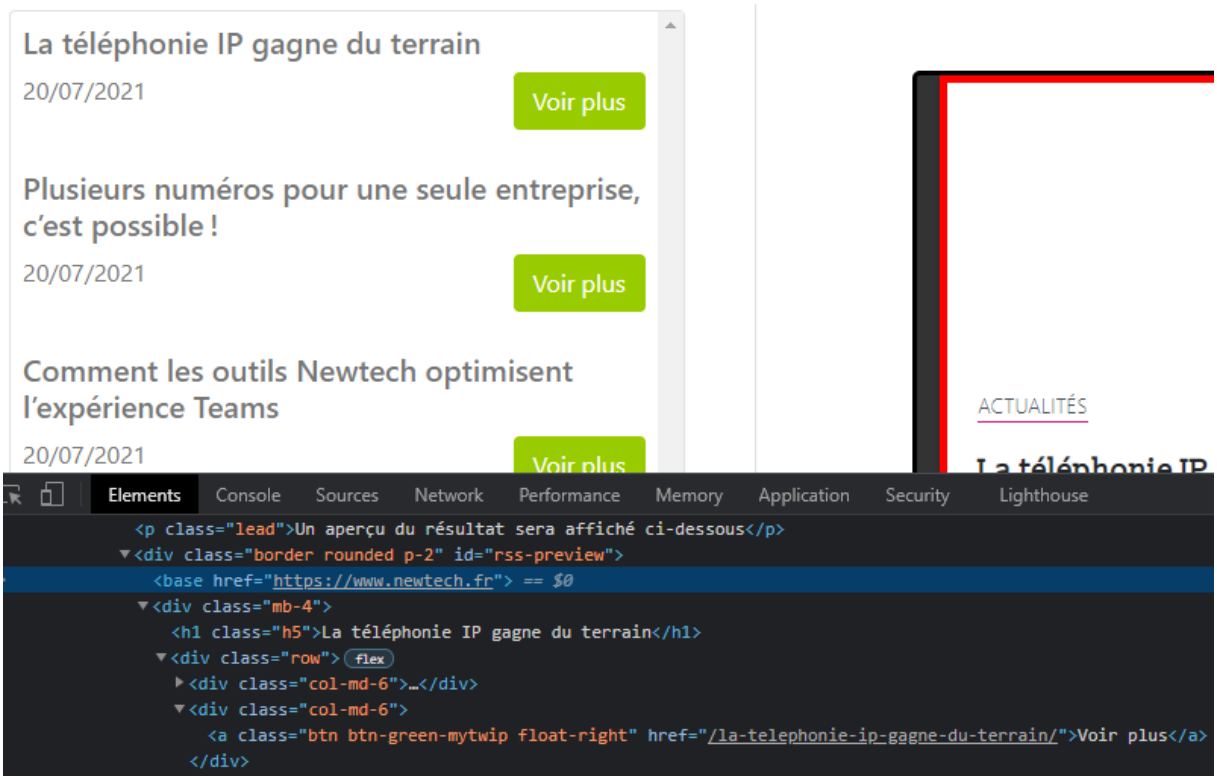


FIGURE 17 : CONSTRUCTION DE LA BALISE HTML BASE DANS PLUGINS AUTO

Dans le cas présent l'URL sera donc : *https://www.newtech.fr/la-telephonie-ip-gagne-du-terrain/*.

Résultats

Exposé des résultats

Dans le cadre du projet Plugins Auto, nous avons mené les tests sur un jeu de 100 sites web d'actualités. Ces sites présentent une diversité représentative de ce que nous avons l'habitude de voir dans les demandes de nos clients, et cette liste a été alimentée avec le temps par des demandes clients réelles afin d'avoir plus de diversité dans nos tests, nous amenant à une liste de 150 sites web à la fin de la période de tests. Une présentation du tableur sur lequel nous avons listé les sites est disponible en annexe 6.

Sur les 150 sites testés, 104 ont eu un plugin fonctionnel créé, ce qui représente 69,33% de réussite, l'objectif déterminé au commencement du projet était de 65%. Néanmoins il nous reste une marge de progression car sur 19 sites nous parvenons bien à supprimer la pop-up et à sélectionner le conteneur, mais lorsque nous exécutons l'algorithme, il semblerait que celui-ci ne détermine pas les bons articles, ce qui pourrait être dû soit à une mauvaise structuration du DOM de la page, soit à des cas que nous ne gérons pas encore avec notre algorithme. 19 sites sur 150 représentent une part non négligeable de notre jeu d'essai, il sera nécessaire de prendre plus de temps pour déterminer concrètement ce qui fausse le repérage des articles pour augmenter la fiabilité de Plugins Auto. Si nous parvenons à gérer ces cas et à créer des plugins fonctionnels, cela ferait monter notre pourcentage de réussite à 82% (123/150). Ces sites sont appelés « la marge ».

Les 18% (27/150) restants, représentent des sites qui sont des cas particuliers et sur lesquels nous ne pouvons pas réaliser de plugins fonctionnels avec Plugins Auto. Sur ces 27 sites, nous sommes capables de faire des plugins fonctionnels à la main sur 21 d'entre eux. Nous sommes capables de les faire de cette manière et pas avec Plugins Auto, car certains sites possèdent des articles n'ayant pas de lien, ou alors dont l'architecture du DOM est trop complexe pour un outil générique tel que le nôtre, mais que nous pouvons faire en temps qu'humains, car nous avons la possibilité de gérer les cas particuliers en allant chercher dans le DOM les chemins CSS qui nous intéressent de manière « sur mesure », contrairement à l'algorithme. Il y a aussi certains sites qui présentent une sécurité contre l'extraction de données relativement élevée, notamment grâce à un système de blacklist lorsque le site détecte une navigation sur son site très rapide, signifiant qu'un programme automatisé opère des opérations sur celui-ci.

Pour observer les résultats de manière plus visuelle, voir le tableau des résultats figure 18 et 19.

Sur ces plugins non réalisables avec Plugins Auto, il en reste alors 4 qui ne sont pas faisables, quelle que soit la méthode utilisée.

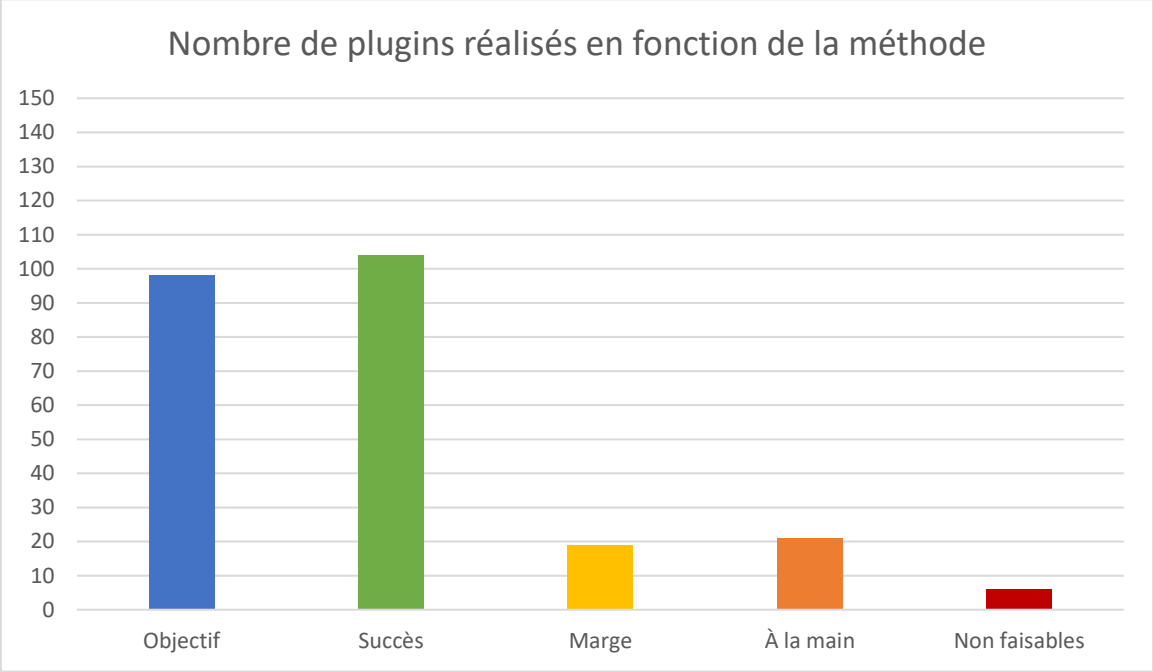


FIGURE 18 : NOMBRE DE PLUGINS REALISES EN FONCTION DE LA METHODE

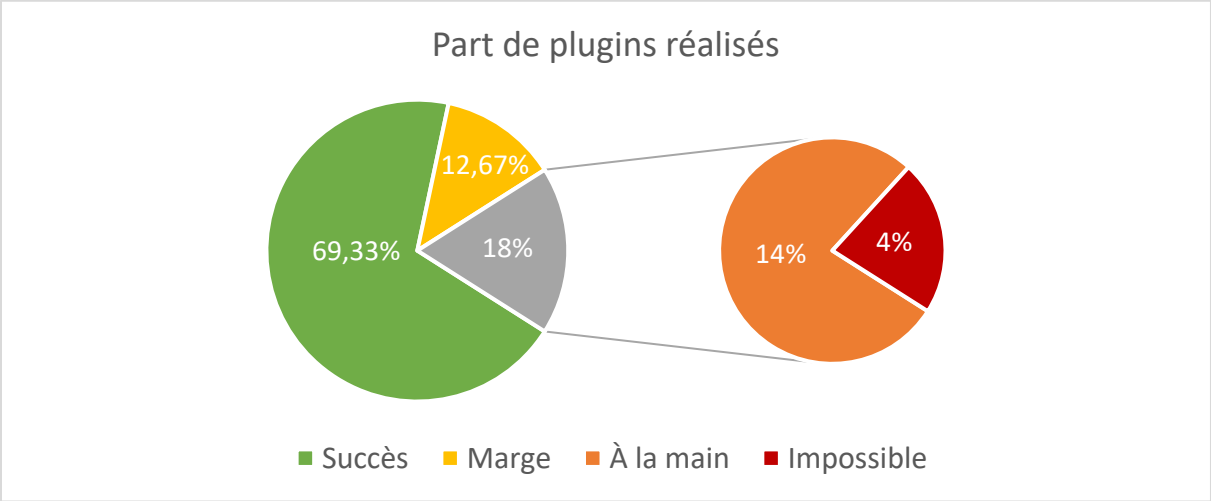


FIGURE 19 : PART DE PLUGINS REALISABLES EN FONCTION DE LA METHODE

Analyse critique

Quelques limites résident du côté de l'utilisation. En effet, nous devons nous fier à la compréhension de l'utilisateur de la structure d'un site. Sans cela, il risque de ne pas comprendre pourquoi notre algorithme ne fonctionne pas sur les sites qui sont à l'origine très difficile à faire pour nous, et finalement, qu'il se lasse d'utiliser notre produit et finisse par nous envoyer l'intégralité de ses listes de sites. Il s'agit sensiblement de la même problématique pour le temps de chargement du site dans l'iframe. Celui-ci peut prendre plus ou moins de temps en fonction du site donné et de sa sécurité. Actuellement un site met en moyenne, sur un éventail de 150 sites, 4.75 secondes à se charger dans l'iframe. Selon diverses études, près de 50% des internautes s'attendent à ce qu'une page web soit chargée en moins de 2 secondes, et au-delà de 3 secondes, près de 80% des utilisateurs jugent qu'il s'agit d'une mauvaise expérience, d'après le site SEO Trench. Le risque est donc que l'utilisateur nous envoie ses listes de plugins s'il juge notre outil pas assez performant. C'est pourquoi nous allons devoir faire preuve de pédagogie auprès de notre clientèle pour leur expliquer les avantages et les limites de notre produit, mais aussi améliorer Plugins Auto avec le temps.

Conclusion

Une majeure partie des prestations de Coexel concerne le suivi des actualités sur différents sites, et leur restitution depuis notre solution Mytwip via l'interface client grâce à l'interprétation de flux RSS. Coexel souhaite consacrer plus de temps sur des projets plus exigeants en termes de ressources humaines et de moyens employés, mais aussi pour libérer plus de temps à l'équipe de production pour augmenter sa réactivité et sa disponibilité vis-à-vis de la clientèle afin d'accroître l'avantage concurrentiel que nous avons de ce point de vue.

Les solutions proposées dans la littérature permettent de montrer l'importance de la demande en solutions automatisées ou semi-automatisées pour augmenter l'autonomie du client dans son activité de veille.

Le travail réalisé tout au long de ce projet montre qu'il est possible de mettre en place un algorithme semi-automatisé face à la diversité des sites web que nous sommes amenés à rencontrer habituellement, que ce soit au niveau de la présentation des articles, de la structure du site web au niveau du DOM, ainsi que sur des traitements particuliers à chaque site pouvant potentiellement entraver la bonne récupération des articles lors de l'utilisation de Plugins Auto.

Le cahier des charges initial a bien été respecté, ainsi que l'objectif de 65% de réussite de plugins fonctionnels, mais une marge de 12,67% reste atteignable pour perfectionner notre produit. Toutefois, il reste 18% de plugins non réalisables avec Plugins Auto d'après le jeu de test que nous avons.

À court terme, Plugins Auto va être amélioré pour gérer un maximum de cas parmi la marge, afin d'augmenter sa fiabilité. Après cela, l'interface sera intégrée à notre solution de veille Mytwip pour que les clients puissent réaliser eux-mêmes un maximum de plugins.

À moyen terme des évolutions mineures pourront être apportées pour améliorer l'ergonomie, et/ou améliorer l'algorithme pour que celui-ci gère plus de sites et augmenter l'autonomie des clients dans ce processus.

À long terme, un autre projet pourrait être envisagé, celui de créer un projet similaire mais basé sur un algorithme d'intelligence artificielle ou automatisé. À l'origine Plugins Auto devait utiliser un algorithme de Machine Learning, mais le manque de temps et de ressources

humaines compétentes dans la discipline face au caractère ambitieux de ce projet ne l'ont pas permis. Un tel projet entrainerait un avantage concurrentiel certain pour Coexel, car à ce jour, aucun de nos principaux concurrents ne propose une telle offre.

Bilan personnel

Ce projet m'a permis d'être force de proposition au niveau des différentes méthodologies à mettre en œuvre. J'ai également pu perfectionner mes connaissances dans les langages utilisés, notamment JavaScript.

Il est toujours passionnant de pouvoir participer à l'évolution d'un logiciel existant, surtout sur des projets relativement ambitieux tels que Plugins Auto, projet qui représente une suite logique par rapport aux travaux réalisés les semestres précédents. Ce projet m'a permis de découvrir les notions d'algorithmes automatisés et semi-automatisés, ainsi que les enjeux qui gravitent autour, mais aussi de profiter d'une formation d'introduction sur le Machine Learning avec un organisme de formation.

J'ai particulièrement apprécié travailler sur l'algorithme en l'optimisant suivant les différents cas rencontrés, ce qui m'a permis de sortir de ma zone de confort, et de ressentir une certaine satisfaction et un sentiment d'accomplissement lorsque je voyais les différentes problématiques détectées au cours de la phase de tests résolues, voyant une réelle avancée dans la fiabilité du produit.

Du point de vue de l'organisation du travail, il n'est pas évident de mettre en place de nouvelles méthodologies au sein d'une entreprise, mais la capacité d'adaptation de l'équipe et la flexibilité de Coexel sont de vrais avantages pour tout apprenti ingénieur souhaitant faire ses premières marques dans la gestion de projet ou souhaitant apporter ses connaissances personnelles à une entreprise.

Mon seul regret est de ne pas avoir pu approfondir les éléments d'écoconception que je souhaitais implémenter, notamment au niveau de l'optimisation du chargement des ressources par le serveur en modifiant le fichier .htaccess, ou alors au niveau de l'architecture du code, et en réduisant la dépendance à un maximum de librairies externes, mais cela montre finalement bien la complexité d'une telle mise en œuvre dans un outil historique d'une entreprise, ainsi que les contraintes techniques associées. Mettre en place une réelle démarche d'écoconception peut se faire à n'importe quel moment dans un projet et à

différents niveaux, mais le plus important est de le faire au début, ce qui laisse une marge de manœuvre certaine pour poser les bases de conception nécessaires.

Ce mémoire marque l'aboutissement de ma formation d'ingénieur informatique et multimédia au Conservatoire National des Arts et Métiers. Coexel m'a très bien préparé à la suite de mon parcours professionnel, et je mettrai en œuvre au maximum les compétences et connaissances acquises lors de mon parcours.

Sources

I. Bibliographie

[1] H. Han, T. Noro, et T. Tokuda, « An automatic web news article contents extraction system based on RSS feeds », *Journal of web engineering*, Vol. 8, No. 3, 2009, pp. 268-284.

[2] E. Cardoso, I. Jabour, E. Laber, R. Rodrigues, et P. Cardoso, « An efficient language-independent method to extract content from news webpages », *Proceedings of the 11th ACM symposium on Document engineering*, Septembre 2011, pp. 121-128.

[3] D. C. Reis, P. B. Golgher, A. S. Silva, et A. F. Laender, « Automatic web news extraction using tree edit distance », *Proceedings of the 13th conference on World Wide Web*, Mai 2004, pp. 502 - 511.

[4] D. Cai, S. Yu, J.-R. Wen, et W.-Y. Ma, « Extracting Content Structure for Web Pages Based on Visual Representation », *Web Technologies and Applications*, 2003, pp. 406-417.

[5] Y. Wu, « Language independent web news extraction system based on text detection framework », *Information Sciences*, Vol. 342, 10 mai 2016, pp. 132-149.

[6] J. Bu, C. Chen, Z. Guan, X. He, G. Lu, J. Pei, J. Wang, « News article extraction with template-independent wrapper », *Proceedings of the 18th International Conference on World Wide Web*, Avril 2009, pp. 1085-1086.

[7] J. A. Iglesias, A. Tiemblo, A. Ledezma, et A. Sanchis, « Web news mining in an evolving framework », *Information Fusion*, Vol. 28, mars 2016, pp. 90-98.

II. Liste des sites consultés

- https://scholar.google.fr/
- https://www.sciencedirect.com/science/article/abs/pii/S0020025515009147
- https://link.springer.com/chapter/10.1007%2F3-540-36901-5_42
- https://www.riverpublishers.com/journal/journal_articles/RP_Journal_1540-9589_833.pdf
- https://www.sciencedirect.com/science/article/abs/pii/S156625351500069X?casa_token=K6o1sVb-xD0AAAAA:ayLTH_iYoCvHoL-a3eJFIT38-sZcCslqTK3JdF2ZqDQvZz3FIZTeuEJUpTCcn53UMj_Jk-amTQ
- https://dl.acm.org/doi/abs/10.1145/988672.988740
- https://ieeexplore.ieee.org/abstract/document/7583910
- https://dl.acm.org/doi/abs/10.1145/3242587.3242661
- https://arxiv.org/abs/1804.04635
- https://distill.io/features
- https://www.slant.co/versus/22371/28613/~visualping_vs_distill-web-monitor
- https://developer.mozilla.org/fr/docs/Web/JavaScript/Language_Resources
- https://www.php-fig.org/psr/
- https://jsdoc.app/
- https://make.wordpress.org/core/handbook/best-practices/coding-standards/css/
- https://wprock.fr/blog/conventions-nommage-programmation/
- https://seo-trench.com/site-web-3-secondes

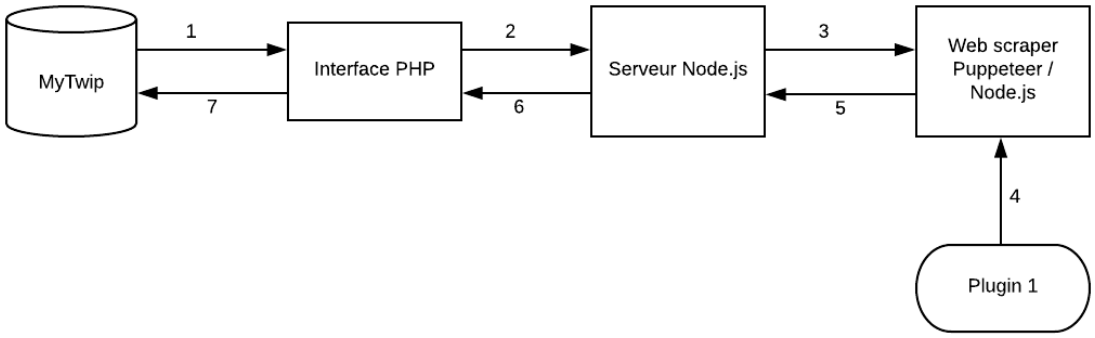
Table des annexes

Annexe 1 : Fonctionnement d'un plugin

Exemple de plugin JavaScript :

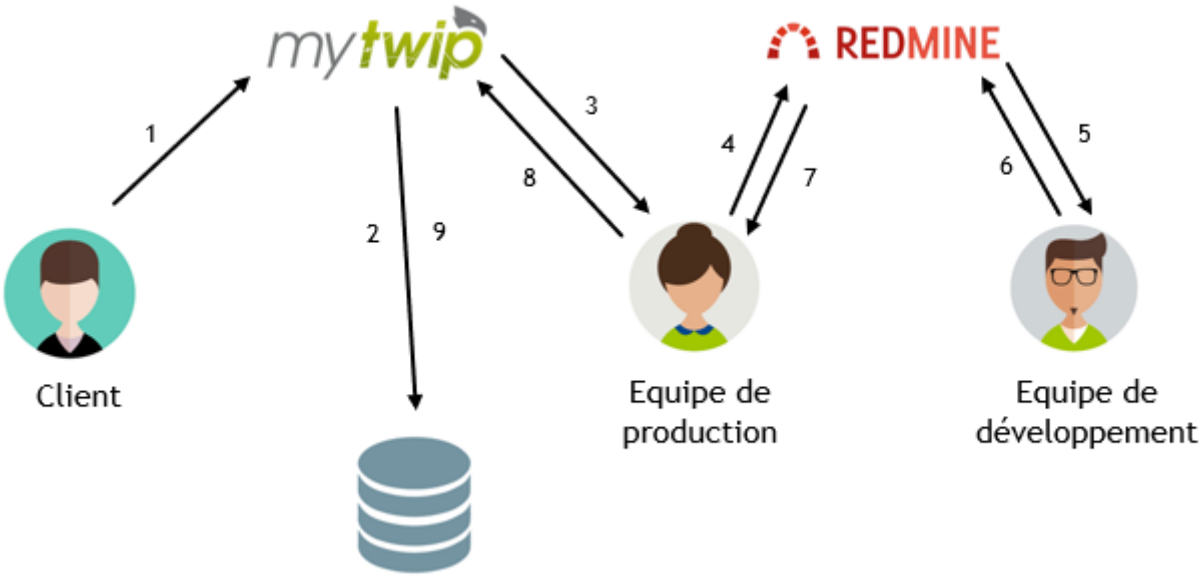
```
module.exports.site = 'https://norminfo.afnor.org/search?term=';
module.exports.selectorToWait = '#Liste-Resultats > .resultats-rech tbody > .standard';
module.exports.rootSelector = '#Liste-Resultats > .resultats-rech tbody > .standard';
module.exports.title = 'a';
module.exports.url = 'a';
module.exports.date = '';
module.exports.abstract = 'td:nth-child(2)';
module.exports.bottom = 5;
```

Schéma du fonctionnement général du web scraper pour un seul plugin :



1) Appel du crawler la nuit par l'application
2) L'interface appelle le serveur Node.js grâce à une URL
3) Le serveur Node.js appelle le scraper grâce à une route définie « /scraper »
4) Le scraper appelle le plugin afin de récupérer les données liées à celui-ci et les met au format JSON
5) Le scraper envoie les données
6) Le serveur les fait transiter à l'interface PHP
7) L'interface PHP extrait les données du fichier JSON pour convertir les données du plugin en un flux RSS exploitable par Mytwip et fait appel au crawler pour les insérer en base de données afin de les intégrer à la solution.

Annexe 2 : Processus de soumission de plugins

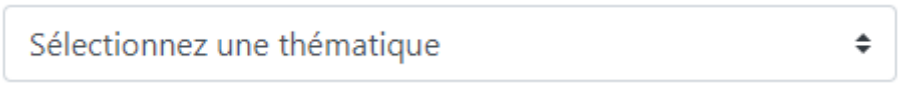


1) Le client soumet ses sources à Mytwip.
2) Mytwip vérifie si les sources ont déjà été soumises précédemment. Si c'est le cas, la solution va directement proposer au client une liste de flux RSS pouvant correspondre à sa demande.
3) Si ce n'est pas le cas, un e-mail est envoyé à l'équipe de production avec la liste des sources du client.
4) L'équipe de production crée une nouvelle tâche sur Redmine à l'attention de l'équipe de développement.
5) L'équipe de développement est notifiée lors de la création de la nouvelle tâche, et crée les plugins nécessaires.
6) L'équipe de développement met à jour Redmine avec l'URL des plugins réalisés.
7) L'équipe de production est notifiée de la mise à jour sur Redmine.
8) L'équipe de production ajoute les nouveaux flux RSS dans Mytwip.
9) Mytwip enregistre les données des plugins dans la base de données.

Annexe 3 : Charte graphique

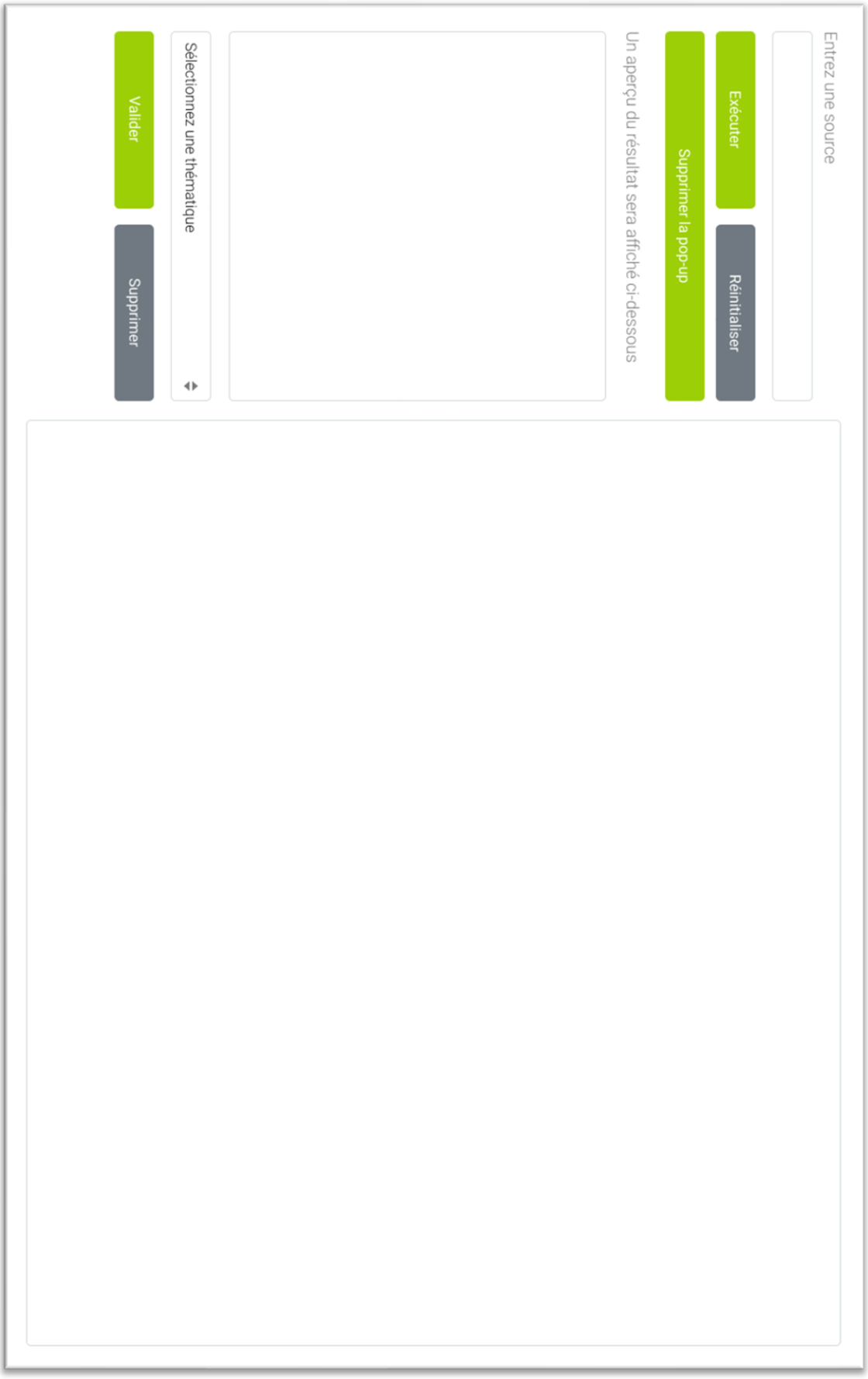
- Police basique sans-serif, qui est intégrée à tous les systèmes d'exploitation et connue de tous les navigateurs : Segoe UI et Arial. Pour faire en sorte que n'importe quel appareil puisse ouvrir l'application.
- Couleur du texte et des bordures : #767676 (gris)
- Couleur des boutons #99cc00 (vert) et #646464 (gris foncé)
- Couleur du fond : blanc #ffffff
- Champ texte (Bootstrap 4) :

- Champ de sélection (Bootstrap 4) :



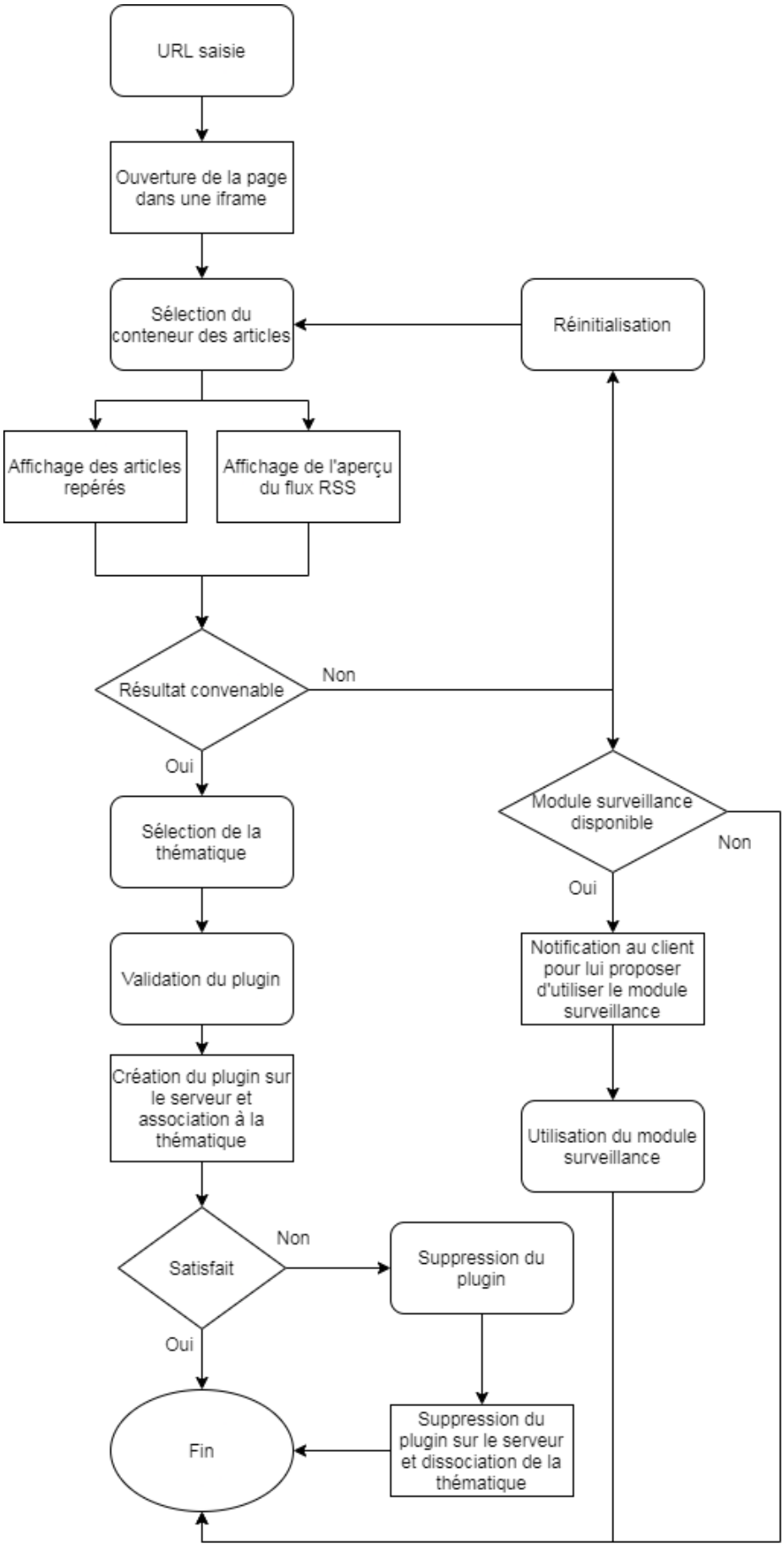
- Boutons (Bootstrap 4) :

- Maquette (réalisée avec Figma) :



Annexe 4 : Parcours utilisateur

1. L'utilisateur souhaite suivre les actualités du site https://www.coexel.com/
2. L'utilisateur va copier l'URL du site, ouvrir Plugins Auto, et coller l'adresse dans le champ prévu à cet effet
3. La page s'ouvre dans une iframe, l'utilisateur peut la visualiser. Sur cette page, les clics d'interaction sont désactivés
4. L'utilisateur sélectionne le conteneur où se trouvent les articles
5. L'utilisateur exécute l'algorithme du Plugins Auto grâce au bouton Exécuter
6. L'algorithme va encadrer en rouge les articles repérés et afficher une prévisualisation du flux RSS dans la zone prévue à cet effet.
7. S'il le souhaite, l'utilisateur peut réinitialiser ce qu'il a sélectionné.
8. L'utilisateur choisit la thématique de son compte à laquelle il veut associer le flux RSS.
9. Il valide son choix, le plugin est créé sur le serveur et le flux RSS est associé en base de données à la thématique associée pour qu'il puisse le retrouver sur son espace client.
10. Si toutefois la réalisation d'un plugin n'est pas possible sur le site souhaité et si dans son abonnement le client possède le module Surveillance, une notification lui sera envoyée pour lui proposer d'utiliser ce module. Sinon il pourra contacter notre service pour que nous lui fassions le plugin.


URL saisie
Ouverture de la page dans une iframe
Sélection du conteneur des articles
Réinitialisation
Affichage des articles repérés
Affichage de l'aperçu du flux RSS
Résultat convenable
Non
Oui
Module surveillance disponible
Non
Oui
Sélection de la thématique
Notification au client pour lui proposer d'utiliser le module surveillance
Validation du plugin
Création du plugin sur le serveur et association à la thématique
Utilisation du module surveillance
Satisfait
Non
Suppression du plugin
Oui
Fin
Suppression du plugin sur le serveur et dissociation de la thématique

Annexe 5 : Product backlog

Les différentes priorités sont : 1 – Importante / 2 – Haute / 3 – Moyenne / 4 – Faible

User story 1 :

- Libellé : Visualisation du site
- Description : En tant que client, je souhaite saisir une URL dans l'application, afin de pouvoir visualiser le site souhaité.
- Sprint : 1
- Priorité : 1
- Dépendances : aucune

User story 2 :

- Libellé : Sélection du conteneur des articles
- Description : En tant que client, je souhaite sélectionner le conteneur où se trouvent les articles, afin de pouvoir le visualiser.
- Sprint : 1
- Priorité : 1
- Dépendances : user story 1

User story 3 :

- Libellé : Exécution de l'algorithme
- Description : En tant que client, je souhaite exécuter l'algorithme pour qu'il récupère les informations que je souhaite.
- Sprint : 2
- Priorité : 1
- Dépendances : user story 2

User story 4 :

- Libellé : Validation du plugin
- Description : En tant que client, je souhaite pouvoir valider mon plugin, afin qu'il soit créé sur le serveur.
- Sprint : 3
- Priorité : 1
- Dépendances : user story 3

User story 5 :

- Libellé : Aperçu du plugin
- Description : En tant que client, je souhaite avoir un aperçu des éléments récupérés, afin de contrôler mon plugin.
- Sprint : 3
- Priorité : 1
- Dépendances : user story 3

User story 6 :

- Libellé : Association du plugin
- Description : En tant que client, je souhaite que le plugin soit associé à mon espace/thématique, afin d'accéder directement aux actualités.
- Sprint : 5
- Priorité : 1
- Dépendances : user story 4

User story 7 :

- Libellé : Suppression du plugin
- Description : En tant que client, je souhaite supprimer le plugin, parce que celui-ci ne me convient pas.
- Sprint : 4
- Priorité : 2
- Dépendances : user story 4

User story 8 :

- Libellé : Réinitialisation du conteneur
- Description : En tant que client, je souhaite réinitialiser les informations retournées si celles-ci ne me conviennent pas.
- Sprint : 3
- Priorité : 3
- Dépendances : user story 3

User story 9 :

- Libellé : Désassociation du plugin à la thématique
- Description : En tant qu'administrateur, je souhaite que le plugin soit délié de la thématique du client en base de données lors de la suppression du plugin, afin de savoir quelles sont ceux qui sont désactivés.
- Sprint : 5
- Priorité : 2
- Dépendances : user story 6, user story 7

Annexe 6 : Tableur des tests

Site	Description du problème
https://www.maddyness.com/hashtag/maddymoney/	
https://www.keyyo.com/	Pop-up RGPD
https://www.newtech.fr/	
https://www.numerique.gouv.fr/actualites/	
https://www.europol.europa.eu/	
http://www.minascent.com/news-downloads.htm	Site mal construit
https://provencebusiness.fr/	Le site ne se charge pas entièrement, ce qui rend impossible l'extraction d'actualités
https://www.francechimie.fr/presse	
https://www.lesechos.fr/industrie-services	Pop-up RGPD, récupère des articles, mais pas les bons
https://www.lesechos.fr/start-up	Pop-up RGPD, récupère des articles, mais pas les bons
https://www.sudinfo.be/2023/sections/actualite	Pop-up RGPD
http://www.uirr.com/en/media-centre/press-releases-and-position-papers/2021.html	
https://www.dfds.com/en/about/media/news	Pas possible : frame-ancestors 'none'
https://transportnytt.se/	
https://acpr.banque-france.fr/news/communique-de-presse	
https://www.fdanews.com/	
https://www.pharmavoice.com/	Je ne peux pas sélectionner de zone
https://www.agribusinessglobal.com/	
https://www.mypharma-editions.com/	
https://www.hi-paris.fr/news/	
https://moto-station.com/tag/electrique?univers=scooter-station	Pop-up RGPD, ne parvient pas à récupérer les articles
https://www.frisonscooter.com/blogs/news/tagged/scooter-electrique	
https://auto.orange.fr/ecomobilite/news/	Pop-up RGPD

https://www.frandroid.com/produits-android/automobile/scooters-electriques	Pop-up RGPD
https://www.francebleu.fr/provence-alpes-cote-dazur/infos/economie-social	Pop-up RGPD
https://www.lecampus.online/conferences	L'algo ne parvient pas à identifier les différents articles
https://www.mobilitesmagazine.com/	
https://www.cae-eco.fr/Focus-CAE	
https://www.cae-eco.fr/Actualites-CAE	
https://www.sauramps.com/	Trop compliqué
https://www.cae-eco.fr/Notes-CAE	
https://www.mollat.com/economie-droit	Trop compliqué
https://www.mollat.com/sciences-humaines-histoire	Trop compliqué
https://www.futuribles.com/fr/vigie/	
https://www.socialter.fr/	Site non chargé
https://www.ih2ef.gouv.fr/actualites#tab-news-item-1	
https://lawportal.com/law-firm-news/	
https://wayback.archive-it.org/12090/20210412125123/https://ec.europa.eu/easme/en/news	
https://www.kuraray.com/news	L'algo ne parvient pas à identifier les différents articles
https://www.energie-rs2e.com/fr/flashes	Pas possible
https://www.agra.fr/agra-presse/	Pop-up RGPD
http://www.bilto.fr/jeux/pid348-actualites.html	Pop-up RGPD
https://www.bsc.news/category/bsc-news	
https://capitalfinance.lesechos.fr/	Pop-up RGPD, ne récupère pas les bons articles
https://www.darchitectures.com/actus-breves-c4.html	Pop-up
https://www.decisions-hpa.com/categorie/actualite/	Pop-up RGPD
https://www.diapasonmag.fr/a-la-une	Pop-up RGPD
https://ecobretagne.com/	Pas possible de cette manière, mauvaise structure
https://www.info-entreprise.com/blog	Possible, mais tous les articles ne s'affichent pas

Table des illustrations

Résumé

L'activité d'extraction d'actualités prend beaucoup de temps aux équipes de Coexel qui pourrait être mis à profit d'autres projets plus importants et à l'amélioration de son service client, qui fait sa force vis-à-vis de la concurrence.

Ce mémoire propose une méthode semi-automatisée qui permet aux clients de réaliser eux-mêmes leurs plugins depuis une interface, en sélectionnant la zone du site qui contient les actualités, et dans un second temps en exécutant un algorithme semi-automatisé pour récupérer leurs informations depuis la structure de la page web.

Les tests réalisés montrent la possibilité d'extraire des actualités sur environ 80% des sites web.

Mots-clés : flux RSS, algorithme semi-automatisé, web scraping

---

The news extraction activity takes up a lot of Coexel's time which could be used for other more important projects and for improving its customer service, which is its strength in relation to the competition.

This thesis proposes a semi-automated method that allows customers to make their own plugins from an interface, selecting the area of the site that contains the news, and then running a semi-automated algorithm to retrieve their information from the web page structure.

The tests carried out show the possibility of extracting news from about 80% of the websites.

Keywords : RSS feed, semi-automated algorithm, web scraping