UEL313 : Bibliothèques logicielles - GROUPE 6 - S3

Membres du groupe

Etudiant.e	Alias
Mathilde C.	Clouddy23
Kamo G.	Spaghette5
Mathieu L.	mathleys
Filippos K.	filkat34

Dépôt public

Le projet est hébergé sur Github : https://github.com/filkat34/UEL313-Groupe6-S3

Objectifs

- ☑ Assurer la maintenance corrective et évolutive d'une application existante
- ☑ Savoir utiliser le client git et la plateforme Github en vue de collaborer au sein d'une équipe de développement.

Environnement de développement

- Cloner ce dépôt GIT sur votre machine.
- Suivre le guide d'installation de l'environnement docker (pdf fourni dans les ressources de l'UE) en remplaçant lors de l'étape "Run a new container" le "Host path" par le chemin vers le dossier cloné du dépôt sur votre machine.

Principe général de collaboration

Répartition du travail

Flux RSS	Page Links	Refonte UI
Filippos	Mathilde & Mathieu	Kamo & Mathieu

Calendrier

Une réunion d'équipe est prévue à chaque fin d'échéance.

Echéance	Objectif
11/12	Phase de documentation et de réflexion sur la façon d'implémenter la fonctionnalité. Décrire le choix d'implémentation retenu ci-dessous.
13/12	Phase de développement de chaque fonctionnalité sur une branche distincte.

Echéance	Objectif
14/12	Relecture des branches, fusion et tests manuels fonctionnels.

Phase de documentation et de réflexion

Ci-dessous sont explicitées les implémentations choisies pour chaque intervention évolutive sur l'application.

Flux RSS

Les cadriciels fournissent souvent des modules spécifiques pour la génération de flux RSS comme sfeed pour Symfony. Vu la simplicité du fonctionnement de cette application, nous avons décidé de ne pas avoir recours à l'un de ces modules mais de mettre en place nous-mêmes le flux RSS en suivant le protocole d'implémentation suivant :

- Création d'une nouvelle route pour le flux RSS `/feed` qui servira le flux RSS.
- Ajout d'une méthode DAO pour récupérer les 15 derniers liens dans `LinkDAO.php`
- Création d'un nouveau contrôleur `RssFeedController.php` qui récupère les 15 derniers liens grâce à la méthode DAO précédemment implémentée et qui génère le fichier xml du flux à partir des liens récupérés. Dans le *content-type* de la réponse du contrôleur, nous avons choisi de mettre *application/rss+xml* plutôt que *'text/xml* qui est plus universel car le premier a une meilleur compatibilité avec les agrégateurs de flux rss.

Idéalement, il aurait fallu également modifier la table des données de l'applications pour ajouter un champ *createdat* car, en l'état actuel, les derniers liens sont récupérés en fonction de leur clé primaire qui est automatiqument incrémentée à chaque ajout de lien dans la base.

Page de liens

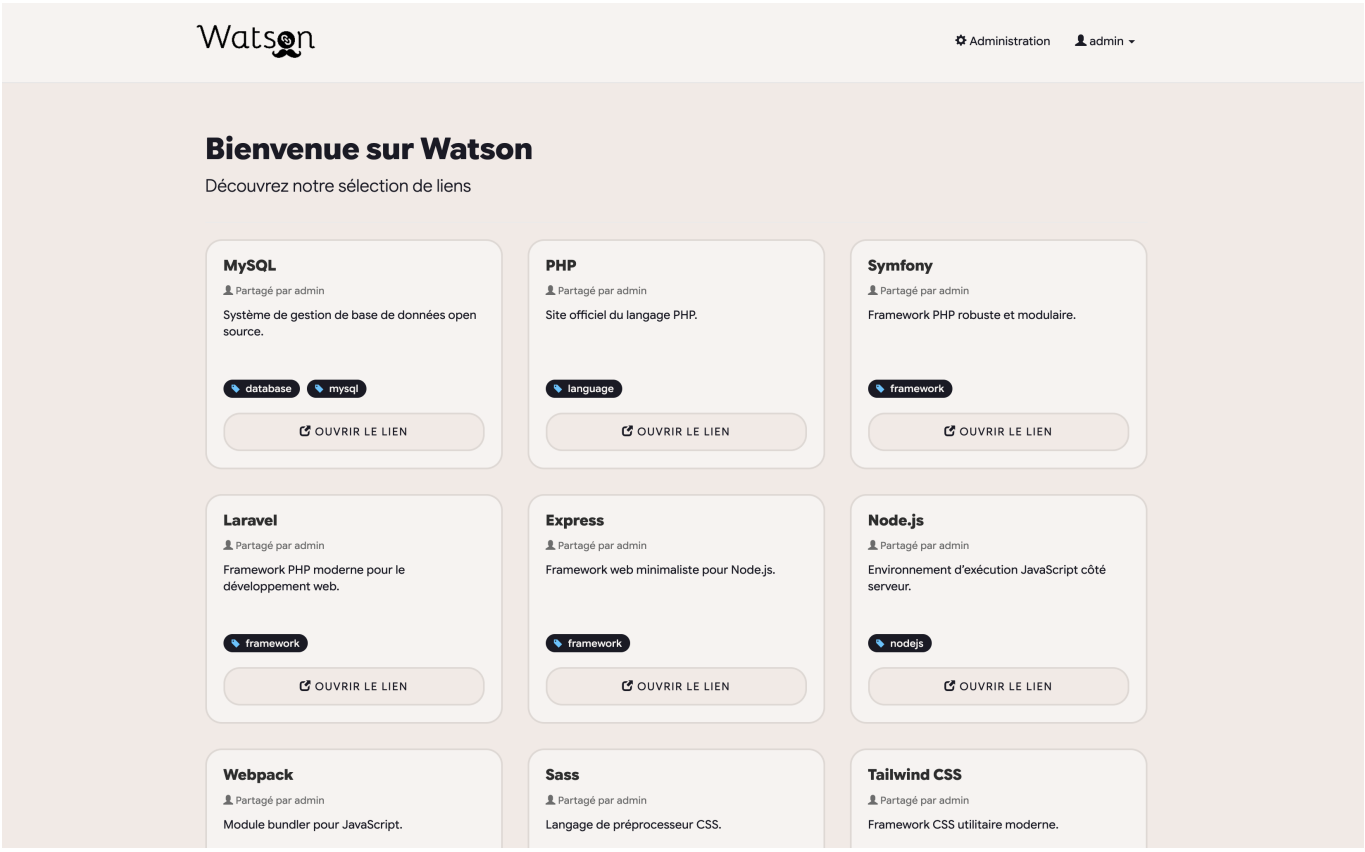
Pour la pagination du back-office, on a choisit une approche côté serveur (PHP/SQL) plutôt qu'en JavaScript avec architecture MVC (DAO pour les données, contrôleur pour la logique métier, vue pour le rendu HTML avec Twig).

Objectif : Limiter la quantité de données chargées (éviter le chargement de toute la table tl_liens) et respecter la contrainte de 15 liens/page directement au niveau de la BDD.

- Côté DAO (LinkDAO) : Ajout de la méthode countAll() pour renvoyer le nombre total de liens présents + Ajout de la méthode findByPage() qui calcule un offset (décalage) selon le numéro de page, exécute une requête avec tri descendant et qui transforme les lignes SQL en objets via la méthode qui existe déjà buildDomainObject().
- Côté contrôleur (AdminController::indexAction) : Ajout de l'objet Request pour pouvoir lire le paramètre ?page= dans l'URL + Récupération du numéro de page + Appel de countAll() pour compter le nombre total de liens et donc du nombre total de pages + Vérification que page demandée ne dépasse pas la dernière page (sinon on donne la dernière page) + Remplacement de findAll() par findByPage($page, $limit) pour récupérer que les 15 liens de la page courante + Passage à la view Twig des variables links, page et totalPages.
- Côté vue (admin.html.twig) : Réutilisation du tableau pour afficher les liens avec ajout de la class pagination de Bootstrap pour afficher proprement les liens vers les pages 1 jusqu'à la dernière, pour indiquer la page courante comme active, pour désactiver les boutons "précédent" et "suivant" si on est

sur la page 1 ou la dernière page et pour génèrer les URLs avec pour rester cohérent avec route /admin.

Refonte UI



Pour la refonte de l'interface de Watson, il a été choisi de conserver Bootstrap dans sa version 3.3 tout en modernisant l'apparence générale de l'application.

Objectif : Améliorer l'expérience utilisateur avec une interface plus moderne, épurée et cohérente tout en conservant la structure HTML/CSS existante.

Modifications apportées :

- **Palette de couleurs** : Mise en place d'une charte graphique cohérente avec une couleur principale utilisée pour les éléments interactifs (boutons, liens actifs, badges)
- **Navigation** : Refonte de la barre de navigation avec un fond blanc, des effets de survol subtils et une meilleure hiérarchie visuelle
- **Cartes de liens** : Transformation des liens en cartes modernes avec ombres portées, coins arrondis (20px) et animations au survol (translation verticale + changement d'ombre)
- **Formulaires** : Création d'un style unifié pour les pages de connexion et d'éditions
- **Espace administration** :
    - Onglets modernisés avec fond blanc, radius cohérent et transition fluide entre les onglets
    - Tableaux épurés avec lignes au survol et boutons d'action colorés
    - Modales de confirmation centrées avec icône d'avertissement et design aligné sur les formulaires
- **Footer** : Simplification avec icône RSS en plus du texte "Flux RSS" et effet au survol
- **Pagination** : Style moderne avec coins arrondis, couleurs cohérentes et états désactivés visuellement distincts

Principes de design appliqués :

- Utilisation intensive de `border-radius` pour adoucir l'interface
- Ombres portées (`box-shadow`) pour créer de la profondeur
- Transitions CSS pour des interactions fluides
- Espacement généreux pour améliorer la lisibilité
- Couleurs cohérentes avec la charte graphique

Phase de développement

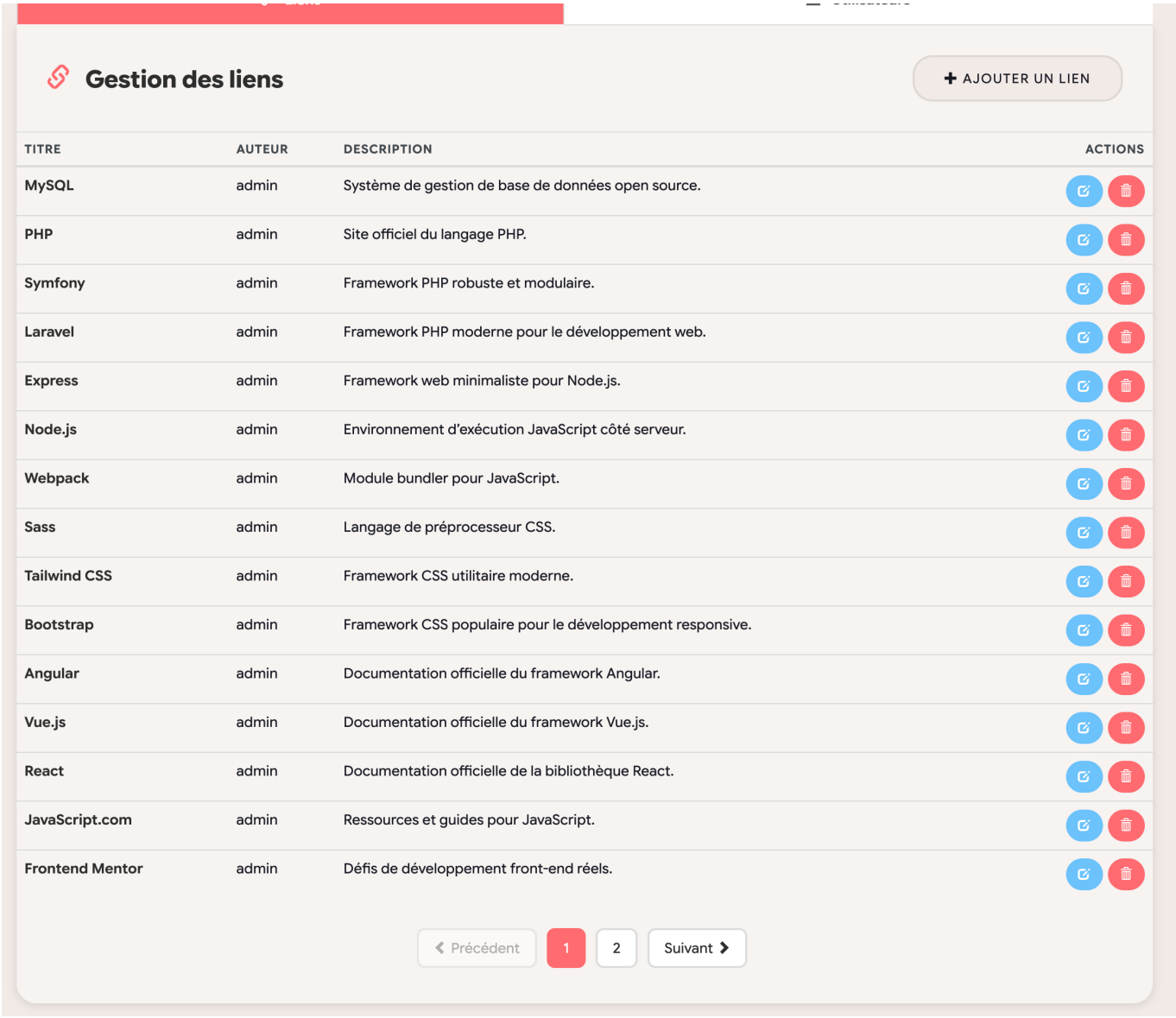
Plusieurs issues ont été identifiées en fonction des fonctionnalités à implémenter :

1. Chaque membre de l'équipe s'assigne une issue en fonction de son choix dans la répartition du travail.
2. Il crée une branche sur laquelle il travaille sur l'issue choisie en lui donnant un nom correspondant à ce qu'il implémente. Exemples : `feature/fluxRSS`, `feature/pagelinks`, etc.
3. Une fois son travail fini, il fait une demande de tirage et dans la description, ne pas oublier de lier la demande à une issue en mettant "Fixes #[numéro de l'issue concernée]" (par exemple : "Fixes #11"). Github se chargera de fermer l'issue en question une fois la fusion de la demande faite.

Tests manuels fonctionnels

Pagination des liens

Pour la page de liens, nous avons ajouté un système de pagination dans l’espace d’administration, limitant l’affichage à 15 liens par page. Cela permet de fluidifier la navigation et d’éviter l’affichage d’une liste trop longue.

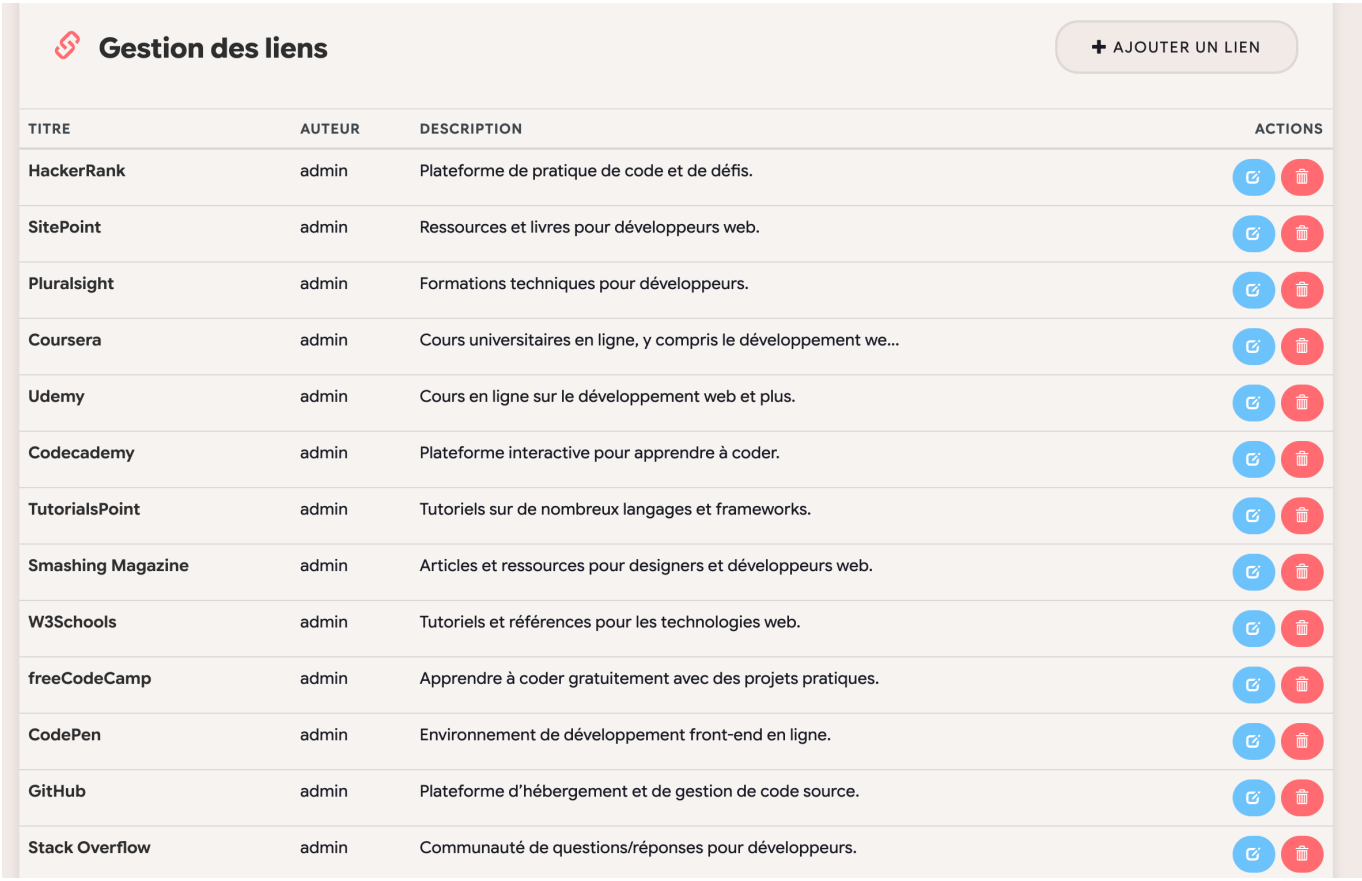


Gestion des liens

+ AJOUTER UN LIEN

TITRE	AUTEUR	DESCRIPTION	ACTIONS
MySQL	admin	Système de gestion de base de données open source.	
PHP	admin	Site officiel du langage PHP.	
Symfony	admin	Framework PHP robuste et modulaire.	
Laravel	admin	Framework PHP moderne pour le développement web.	
Express	admin	Framework web minimaliste pour Node.js.	
Node.js	admin	Environnement d'exécution JavaScript côté serveur.	
Webpack	admin	Module bundler pour JavaScript.	
Sass	admin	Langage de préprocesseur CSS.	
Tailwind CSS	admin	Framework CSS utilitaire moderne.	
Bootstrap	admin	Framework CSS populaire pour le développement responsive.	
Angular	admin	Documentation officielle du framework Angular.	
Vue.js	admin	Documentation officielle du framework Vue.js.	
React	admin	Documentation officielle de la bibliothèque React.	
JavaScript.com	admin	Ressources et guides pour JavaScript.	
Frontend Mentor	admin	Défis de développement front-end réels.	

Précédent 1 2 Suivant



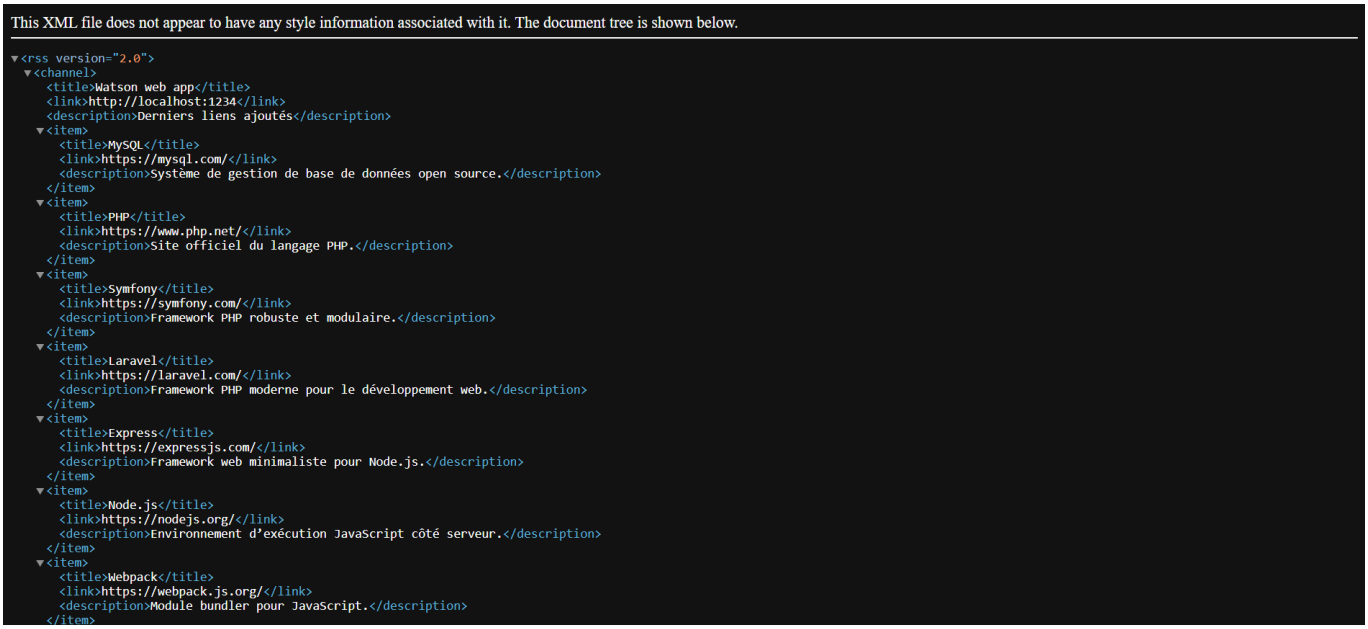
Gestion des liens

+ AJOUTER UN LIEN

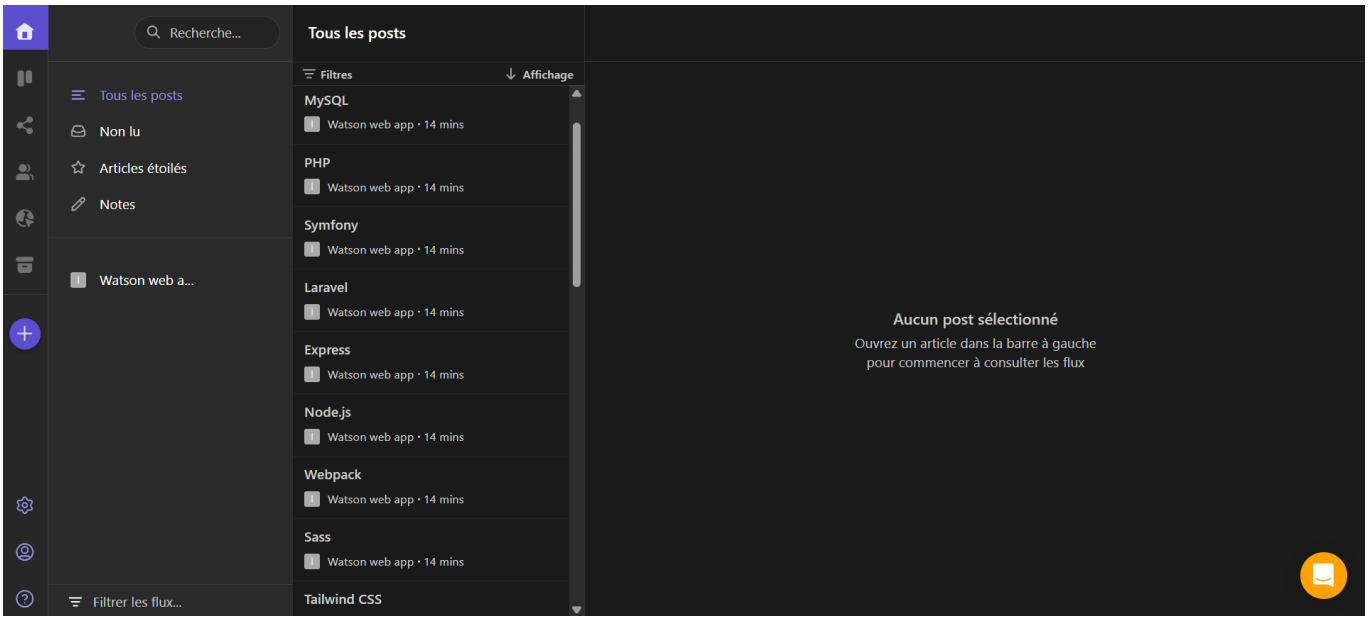
TITRE	AUTEUR	DESCRIPTION	ACTIONS
HackerRank	admin	Plateforme de pratique de code et de défis.	
SitePoint	admin	Ressources et livres pour développeurs web.	
Pluralsight	admin	Formations techniques pour développeurs.	
Coursera	admin	Cours universitaires en ligne, y compris le développement we...	
Udemy	admin	Cours en ligne sur le développement web et plus.	
Codecademy	admin	Plateforme interactive pour apprendre à coder.	
TutorialsPoint	admin	Tutoriels sur de nombreux langages et frameworks.	
Smashing Magazine	admin	Articles et ressources pour designers et développeurs web.	
W3Schools	admin	Tutoriels et références pour les technologies web.	
freeCodeCamp	admin	Apprendre à coder gratuitement avec des projets pratiques.	
CodePen	admin	Environnement de développement front-end en ligne.	
GitHub	admin	Plateforme d'hébergement et de gestion de code source.	
Stack Overflow	admin	Communauté de questions/réponses pour développeurs.	

Flux RSS

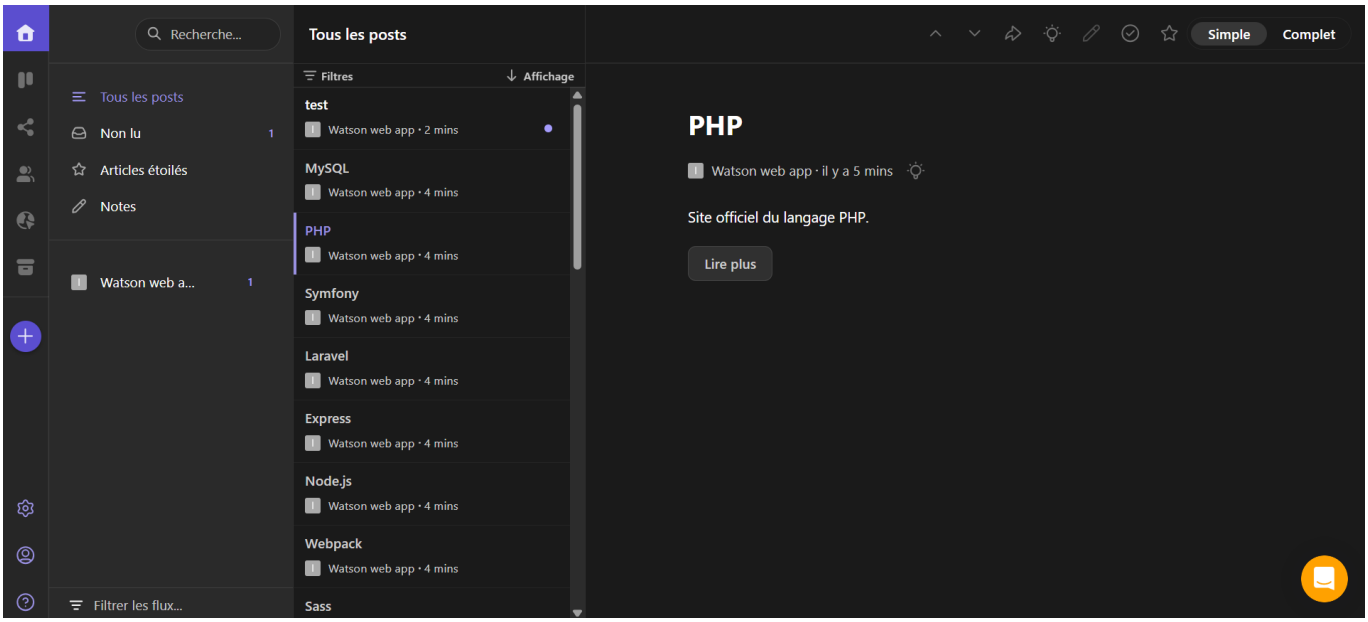
Pour tester, le bon fonctionnement du flux RSS, nous avons d'abord consulté le fichier *xml* disponible sur `/feed`.



Ensuite nous avons installé l'extention *Feeder* sur le navigateur *Chrome* et nous nous sommes rendus sur la même URL. Les 15 derniers liens ajoutés y apparaissaient correctement sur l'interface de l'application.



Nous avons ajouté ensuite un nouveau lien "test" dans l'application Watson pour vérifier si l'affichage était bien dynamique : une notification concernant la publication de ce nouveau lien est correctement apparue.


Recherche...
Tous les posts
Non lu 1
Articles étoilés
Notes
Watson web a... 1
Filtrer les flux...
Tous les posts
Filtres
Affichage
test
Watson web app • 2 mins
MySQL
Watson web app • 4 mins
PHP
Watson web app • 4 mins
Symfony
Watson web app • 4 mins
Laravel
Watson web app • 4 mins
Express
Watson web app • 4 mins
Node.js
Watson web app • 4 mins
Webpack
Watson web app • 4 mins
Sass
Simple
Complet
PHP
Watson web app · il y a 5 mins
Site officiel du langage PHP.
Lire plus