COMPTE-RENDU

# Design

Design de Lucas validé.

# Architecture des sources

Lucas la crée sur GitHub, les fichiers seront organisés en plusieurs dossiers, globalement :

* Web
  + Documentation
  + Node.js
  + …
* src
  + Documentation
  + App : sources cpp
  + …
* Ressources
  + Algorithmes
  + Documentation
  + Medias
  + CR réunions
  + …
* …

# Algorithmes

Les algorithmes sont disponibles dès ce soir dans le dossier /Ressources/Algorithmes sur GitHub.

## Corentin : Crawler

Algorithme validé sur le principe, gestion du FTP et du http à ajouter. Téléchargement de la page doit être inclus mais pas besoin de le revoir.

A CODER

## Marin : Keywords

Revue :

* Checking des doublons une fois tous les mots clés extraits pour éviter de perdre du temps à chaque fois à lire tout le tableau
* Enlever l’étape de stockage intermédiaire dans Html\_content pour ne pas surcharger le processeur
* Enlever les espaces une fois que les mots auront été extraits du flux (double espaces, tab etc)
* H6 et description à ajouter dans les éléments retenus, augmenter la valeur du H1.
* Suppression des mots simples lors de la lecture d’extraction du document, pas avant. On ne le lit qu’une fois.

A MODIFIER

## Mehdi : Rankings

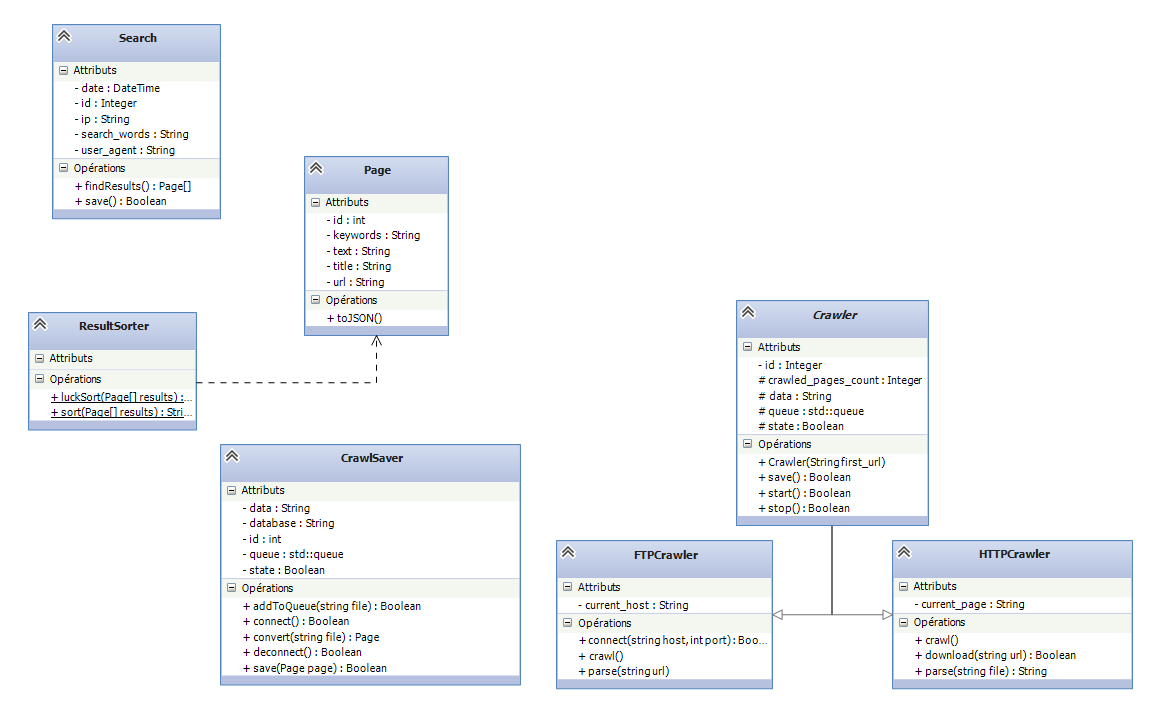
* Revoir l’algorithme en fonction du diagramme de classe suivant
* Attention au poids de la page pour pondérer le résultat, une page plus longue a plus de chance d’avoir chaque mot plusieurs fois. Pourquoi ne pas calculer en pourcentages de points au final plutôt qu’en points ? A voir
* BONUS : Eventualité de faire du cache en enregistrant les 1000 premiers résultats d’une recherche

A MODIFIER

# Diagrammes

## Diagramme de classe

Le groupe adopte celui de Lucas ci-joint : <http://puu.sh/6Zww3.png> (image à la suite, zoomer pour voir)



ResultSorter est une classe statique servant à trier les résultats d’une recherche.

CrawlSaver réagit à un évènement lancé par un crawler à la fin du crawl d’une page, elle agit de manière parallèle.

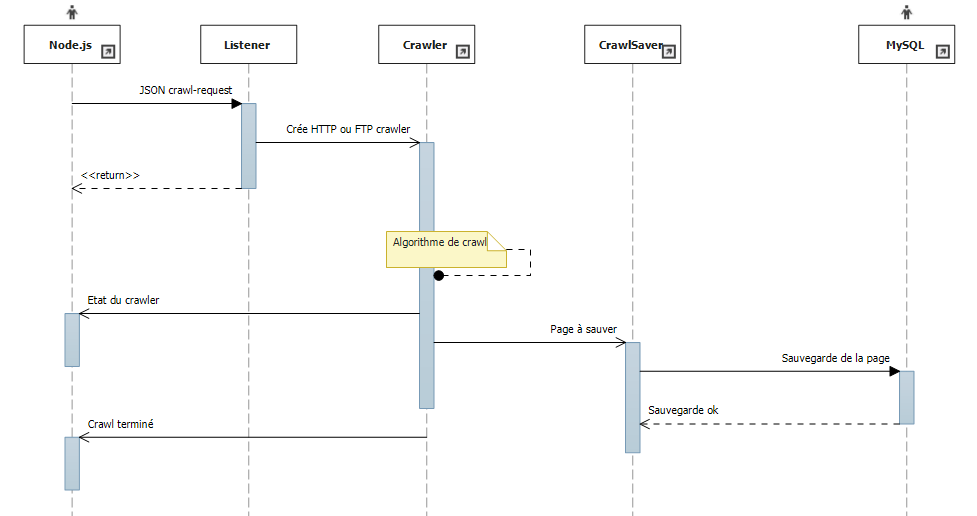
Crawler est une classe abstraite. Pour tout détail, demander à Lucas.

Propositions :

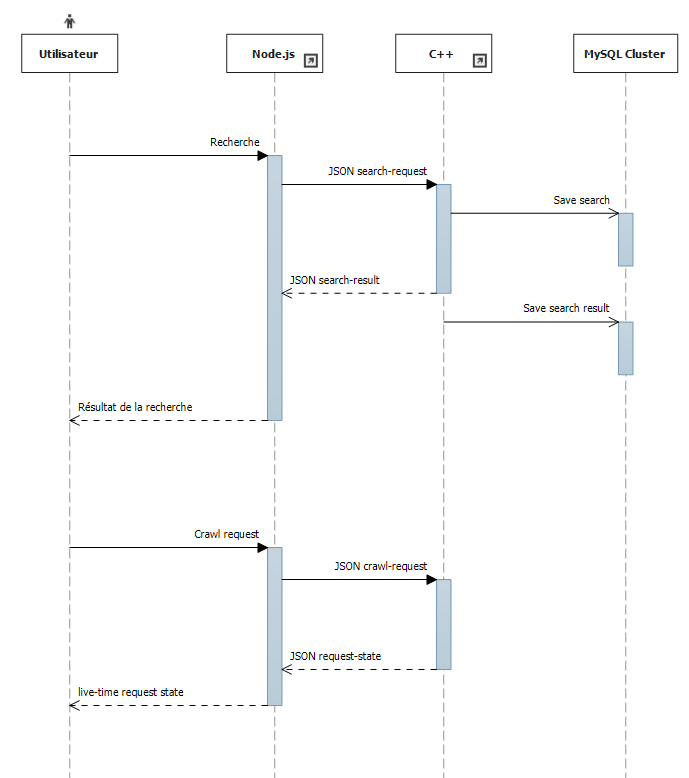
* Ajout d’une classe keywords – **Rejeté par manque d’utilité**
* Groupement de pages – **Rejeté pour le moment**
* Ajouter un champ avec le nombre d’URL pointant vers la page – **À voir en bonus**

## Diagramme de séquences

Tout le monde avait globalement les mêmes idées pour le diagramme de séquence, Lucas joint son diagramme pour le crawl, même principe pour la recherche.



Un diagramme global



# En base de données

Trois tables pour le moment

* Users
  + Id : int
  + Username : string
  + Email : string
  + Password : string (à voir en fonction du système de cryptage)
  + Picture
  + …
* Page
  + Id : int
  + Title : string
  + Text : text
  + Keywords : string (tableau JSON)
  + url : string
* Search
  + Id : int
  + Date : DateTime
  + Ip : string
  + Text : string
  + User\_agent : string
  + Results : string

# Architecture matérielle

4 machines virtuelles Debian pur à 1Go de RAM, un serveur DNS, un reverse proxy pour le web.

Chacun aura les 4 machines sur son ordinateur de travail, elles seront sur un dépôt dès que Lucas les aura configurées avec le cluster MySQL etc…

# Environnement de travail

Les sources seront sur GitHub, Lucas configurera les machines virtuelles pour que tout le monde puisse travailler facilement.

Chaque sprint prévu dans le planning du projet doit entrainer la création d’une nouvelle branche ayant le nom de la fonctionnalité sur git.

La branche master doit rester à tout moment COMPILABLE et PROPRE, ne faire un checkout que lorsque le build passe et la feature est fonctionnelle.

Une fois les VM téléchargées, chacun gère ses machines comme il l’entend, pour tout ajout de paquet/programme important, consulter le groupe.

# Code : Bonnes pratiques

* Nommage des variables : nom\_variable
* Nommage des fonctions : nomFonction()
* Code en anglais, commentaires en français
* Bien espacer et indenter le code, exemple : if (i == 3 && j != 2) …
* Commentaire toutes les 10 lignes maximum, et sur chaque passage non évident (pas besoin de commenter un while (true))
* Un .h et un .cpp par classe. Les noms de classe sont en anglais de la frome NomClass.

Tâches planifiées jusqu’au 22 mars. Réunions le 7 et le 22 en théorie, à voir plus tard pour la date/heure exacte.

Au boulot ;)