

Projet tutoré



Encadré par :

Lardeux Frédéric, Enseignant Chercheur, Faculté des sciences -Université d'Angers

Daucé Bruno, Maître de conférences, Faculté de Droit-Économie-Gestion - Université d'Angers -

Réalisé par :

Agharbi Ayman

Coest jordan

Mamori Nabil

Kos Boris

Licence professionnelle Métiers de l'informatique / Logiciels libres

Université d'Angers

2018-2019

Table des matières

Introduction.....	3
Remerciement.....	3
Partie 1 – Organisation et informations générales.....	4
I) Présentation du projet.....	4
A. Contexte.....	4
B. Répartition des tâches et organisation.....	5
II) Architecture de l'application.....	7
A. Choix de développement.....	7
B. Persistance des données.....	8
C. Une application modulaire.....	10
Partie 2 – La collecte des données : quelles solutions techniques ?.....	12
I) Élaboration de la base de données « revues ».....	12
A. Quelles sources ?.....	12
B. Comment mettre à jour cette base de données ?.....	13
II) Récupération des « Call for papers ».....	15
A. Le <i>parsing</i> de données.....	15
B. Les technologies.....	15
C. Faire le lien entre les <i>calls</i> et les revues (fonction de taux de vraisemblance).....	18
D. L'automatisation.....	20
Partie 3 – Exploitation des données : les livrables.....	23
I) Affichage des données : une page web.....	23
A. Template ou pas template.....	23
B. Recherche utilisateur : quel mécanisme (requêtes SQL) ?.....	23
C. Hébergement.....	24
II) Améliorations futures.....	24
Conclusion.....	27
Annexe 1 : Manuel utilisateur / administrateur.....	28
Annexe 2 : Manuel développeur.....	33

Introduction

Le présent rapport a pour objectif de présenter et décrire le déroulement d'un projet tutoré de troisième année de licence professionnelle (logiciels libres) à l'Université d'Angers. Ce projet s'est déroulé entre le 17 décembre 2018 et le 28 mars 2019 : il a été réalisé par un groupe de quatre étudiants ; cinq semaines ont été dédiées à ce projet dans l'emploi du temps durant cette période.

Ce rapport présente, de manière synthétique, les principaux éléments qui ressortent de cette expérience de gestion de projet. Il tentera de répondre aux questions suivantes :

- **Quelles ont été les mises en œuvre concrètes pour mener à bien ce projet ?**
- **Comment justifier ces mises en œuvre, ces choix ?**

Une première partie sera consacrée à une présentation générale du sujet et de la gestion du projet. Dans un deuxième temps, il s'agira d'expliquer les solutions techniques qui ont été utilisées pour le *parsing* et le stockage des données. Enfin, une troisième et dernière partie évoquera les livrables du projet et leurs évolutions futures possibles.

Remerciement

Nous profitons de ce moment du rapport pour remercier M. Frédéric Lardeux pour nous avoir accompagné tout au long de ce projet : ses conseils ont su nous motiver et nous aiguiller. Nous remercions également M. Bruno Daucé pour avoir apporté la problématique du projet : nous avons tous, les membres du groupe, grandement apprécié travailler sur ce sujet.

Partie 1 – Organisation et informations générales

I) Présentation du projet

A. Contexte

En recherche académique, les résultats obtenus sont publiés dans des revues scientifiques. Certains numéros de revues sont consacrés à un problème particulier. Afin de communiquer aux différents chercheurs la possibilité de publier leurs travaux de recherche, des appels à publication sont diffusés. Le problème pour les chercheurs est de trouver l'appel à contributions correspondant à leurs travaux de recherche. En effet, ces appels sont souvent diffusés par courrier électronique, sur les pages Web des revues, sur les pages Facebook... Cette multitude de canaux de communication dilue l'information. De plus, tous les magazines n'ont pas la même réputation. Pour que le travail soit valorisé, il doit être publié dans des revues de bon niveau. Dans notre cas nous prendrons le domaine de la gestion, différents classements existent : celui de la Fondation nationale pour l'enseignement de la gestion des entreprises (FNEGE), celui du CNRS ainsi que celui du Haut Conseil de l'évaluation de la recherche et de l'enseignement supérieur (HCERES). L'objectif de ce projet est de proposer un outil permettant d'analyser les différents appels à publication et de les référencer sur une page Web indiquant leur appartenance (ou non) aux différents classements.

L'intérêt d'une telle réalisation est double. D'une part, comme l'explique le paragraphe précédent, il s'agit d'un besoin pour les enseignants chercheurs de centralisation et de qualification de l'information. D'autre part, cet outil peut également présenter un intérêt certain pour les éditeurs. En effet, on peut imaginer qu'une telle page web, si elle est utilisée par un grand nombre d'enseignants chercheurs, devienne une sorte de vitrine pour ces derniers. Nous avons pris en compte cet aspect en intégrant le logo de l'éditeur à l'affichage.

L'objectif du projet est donc la création d'une application de recueil d'appels à publication qui permet, pour les enseignants chercheurs, de trouver "*The Good Call*" (de l'anglais *call for paper*) : celui dont la thématique correspond au secteur d'expertise et qui est demandé par une revue "sérieuse". Ainsi, nous proposons, avec M. Bruno Daucé, cette expression comme nom de projet. L'anglicisme de ce nom est bien entendu choisi afin de toucher un public de chercheurs en France, mais aussi à l'étranger.

B. Répartition des tâches et organisation

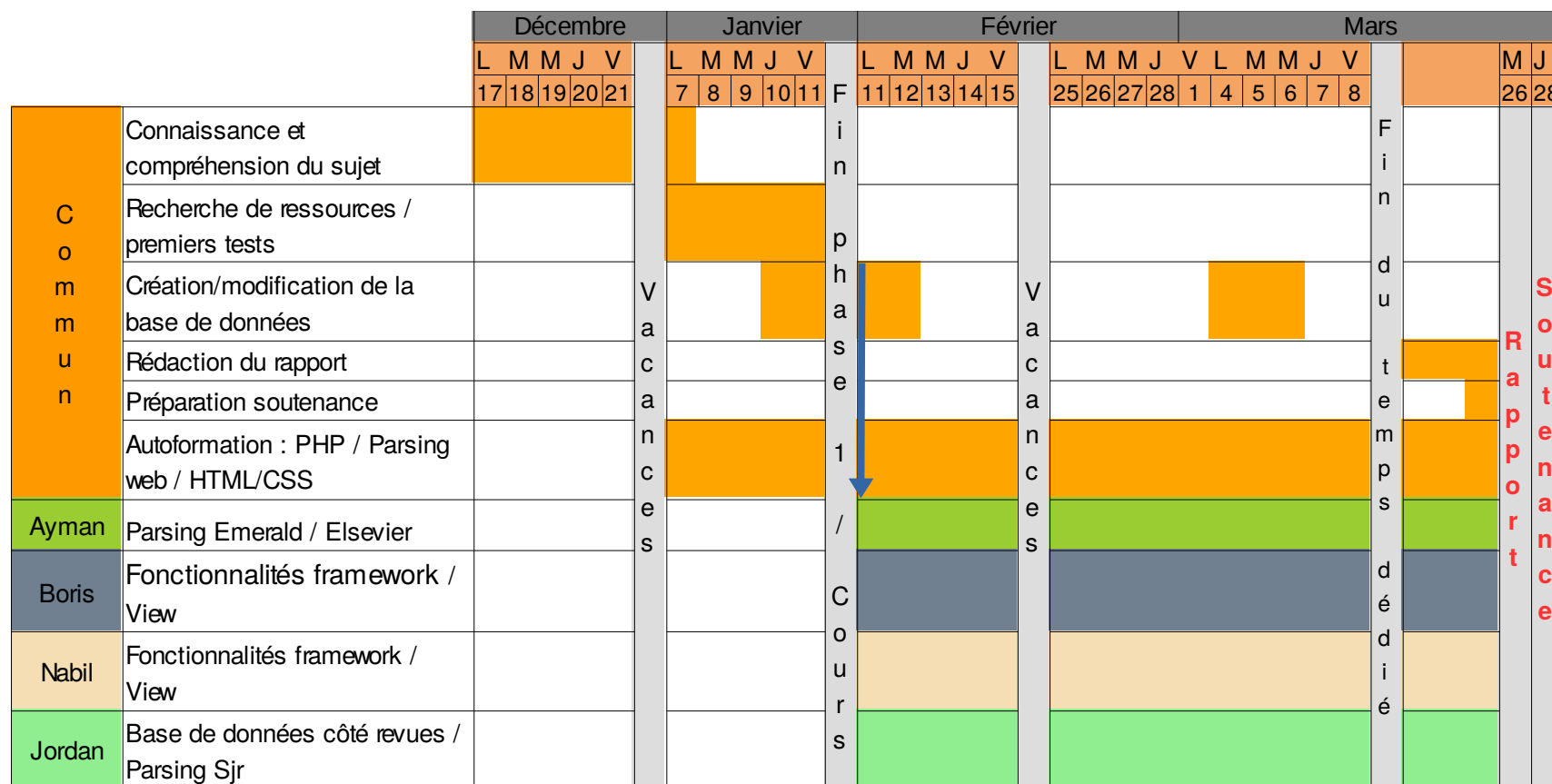


Figure 1: Diagramme de Gantt simplifié (du lundi 17/12/18 au jeudi 28/03/19)

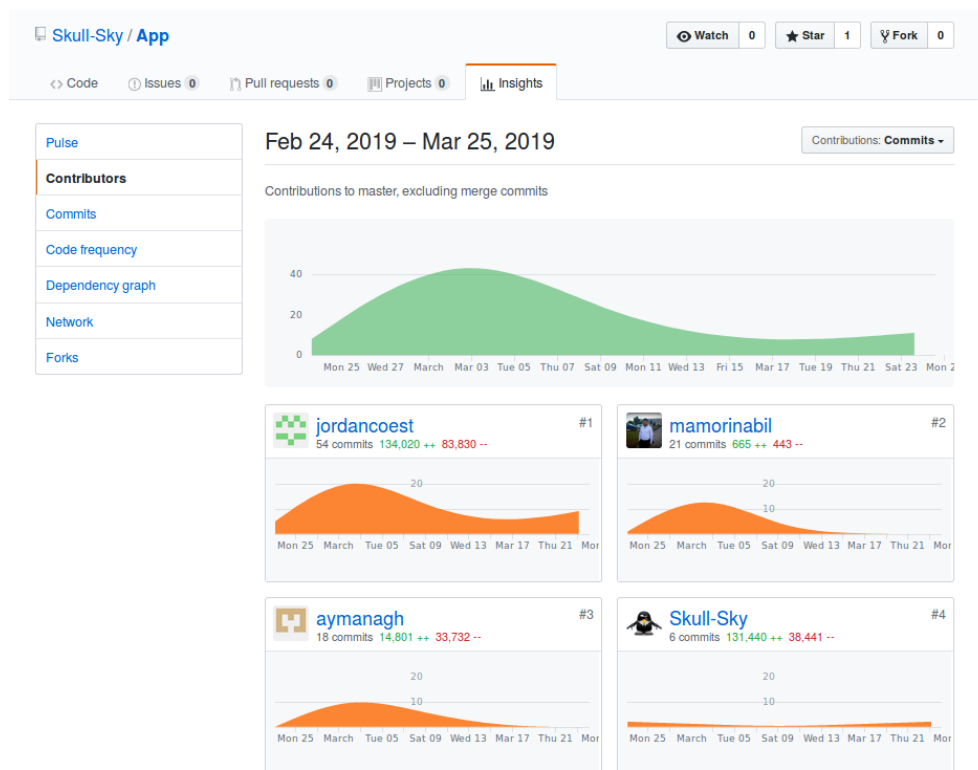


Figure 2: L'interface GitHub qui nous a permis de gérer le partage du code

Le diagramme de Gantt simplifié (page précédente) permet de visualiser l'organisation de notre travail tout au long du projet. Bien entendu, la réalisation des tâches ne s'est pas faite de manière "linéaire" : nous avons effectué de nombreux aller-retour pour rectifier ou améliorer certains aspects du développement. De plus, bien que nous ayons tenté de séparer les axes de travail entre chaque membre du groupe, il a très souvent été utile de s'entraider pour faire avancer les choses.

Afin de faciliter le travail en équipe, nous avons également utilisé le logiciel de version *GIT*, et le service web d'hébergement *GitHub*. Cette technologie permet de se tenir à jour régulièrement des modifications de code et de récupérer des versions précédentes en cas de besoin : <https://github.com/Skull-Sky/App>.

II) Architecture de l'application

A. Choix de développement

S'agissant d'une application web faisant appel à une base de données, le choix du langage de programmation s'est très vite tourné vers PHP.

Nous avons mis cette solution en concurrence avec le langage JAVA mais nous n'avons finalement pas retenu ce dernier car PHP semblait nous apporter de meilleurs arguments. Il s'agit en effet d'un langage très populaire, bien documenté et qui était certainement le plus "maîtrisé" par les membres du groupe. En effet, les premiers jours du projet nous ont permis de faire un listing des différentes tâches/fonctions à réaliser et le langage PHP nous a semblé le plus pertinent, car disposant déjà de nombreuses bibliothèques (pour le *parsing* de pages web notamment). L'aspect plus procédural de PHP (en comparaison à JAVA qui s'apparente plus à un langage orienté objet) nous semblait également plus adapté pour ce type de projet : l'exploration et l'apprentissage sont rendus très rapide avec PHP (un simple rafraîchissement de page sur le navigateur permet de vérifier, en grande partie, la fonctionnalité ou non du code). Cependant, cette fin d'année de licence professionnelle nous a permis de découvrir le "*framework*" Jave EE, pour *Java Enterprise Edition* qui facilite le développement d'applications web. Sans aucun doute, la réflexion sur le choix du langage aurait été différente s'il avait débuté à cette période de l'année.

Nous avons également pris la décision de ne pas utiliser de *framework* pour le développement de notre application. Toujours dans un souci d'exploration, nous avons en effet préféré ne pas nous contraindre à un *pattern*. De plus, maîtriser un framework prend du temps, et peut donc parfois mettre à l'écart certains membres d'un groupe (en cas de différences dans sa compréhension). Également, nous avons tenté de respecter deux principes de développement : YAGNI et KISS (pour *You Ain't Gonna Use It* et *Keep It Simple, Stupid*) qui préconisent de mettre toujours en œuvre les choses au moment où on en a réellement besoin – YAGNI - et de ne pas compliquer les choses - KISS. En fait, nous avons, au fur et à mesure du projet, ajouté certaines fonctionnalités (fonctions PHP) dans un fichier nommé `framework.php`. Il ne s'agit bien entendu pas d'une infrastructure de développement complète comme pourrait l'être par exemple *Symphony*, mais ce fichier nous a facilité le travail de groupe comme l'aurait fait un *framework*.

Afin de structurer le code et donner une ligne directrice à notre développement, nous avons néanmoins tenté de nous approcher d'une structure MVC (*Model View Controler*). Notre application est ainsi articulée en 3 dossiers :

- un dossier *view* qui correspond à l’affichage de l’application.
- un dossier *model* qui fait le lien avec la base de données dans un sous-dossier *Bd* et qui contient la logique de récupération des données (*parsing*), ainsi qu’un dossier *admin*.
- un dossier *framework* qui, principalement, modifie les vues en fonction des besoins et gère les demandes des utilisateurs de l’application.

Bien entendu, un fichier `index.php` permet un *routing* (minimaliste) de l’application lors de son lancement.

B. Persistance des données

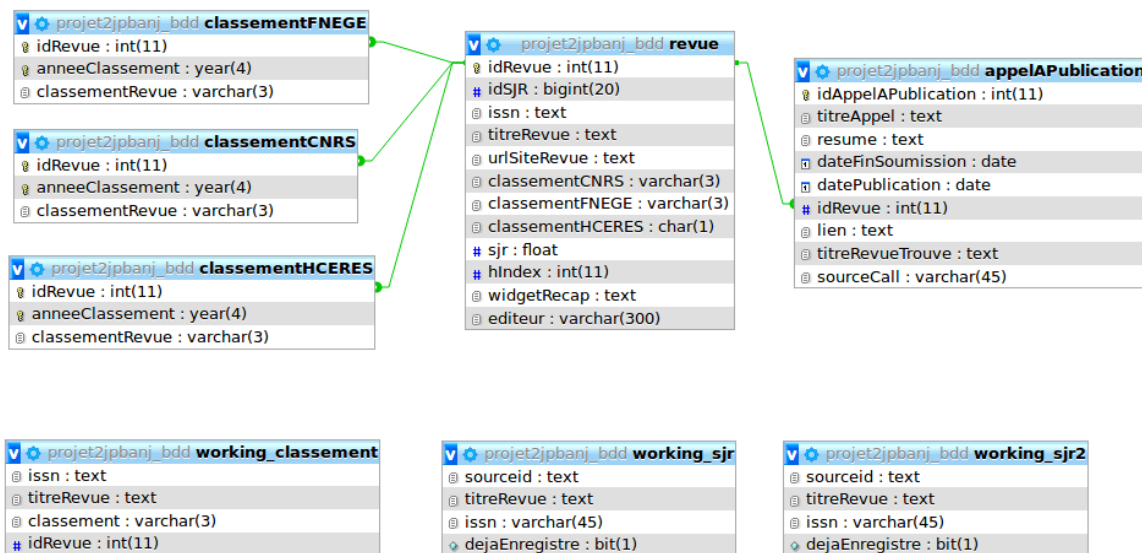


Figure 3: Représentation de notre structure de base de données

« *The Good Call* » repose en grande partie sur la capacité de l’application à faire le lien entre un appel à publication et la revue scientifique à laquelle il est associé. Nous avons donc créé dans un premier temple les deux tables `revue` et `appelAPublication` dans une base de données. Les parties suivantes permettront de détailler les informations contenues dans ces tables. On peut cependant déjà préciser que l’identifiant de la revue (`idRevue`) est clé étrangère de la table qui enregistre les appels à publication, et permet donc de faire le lien lors des requêtes SQL.

Il nous a été demandé de pouvoir enregistrer l'historique des différents classements pour les revues. Le classement SJR comprend plusieurs dizaines de milliers de revues et l'historique de ce classement est facilement consultable via le site internet www.scimagojr.com. Nous avons donc jugé que l'historisation de cette "note" était trop coûteuse dans notre base de données. En revanche, et cela correspondait à la demande initiale, nous avons intégré un moyen d'enregistrer l'historique des classements HCERES, CNRS et FNEGE (environ 900 revues concernées). Les tables `classementHCERES`, `classementFNEGE` et `classementCNRS` répondent à ce besoin. Elles sont identifiées par une clé primaire qui est la concaténation de l'identifiant de la revue et l'année du classement en question. Néanmoins, pour chaque revue, nous avons fait le choix de recopier la valeur du classement en cours dans la table `revue`. En effet, nous avons jugé qu'il était trop coûteux d'effectuer une requête complexe (*jointure + group by*) à chaque fois qu'un utilisateur demande l'affichage de la page web, plutôt que de mettre à jour le classement en question dans la table `revue` à chaque mise à jour de ce dernier (soit une fois par an maximum : le classement 2016 du FNEGE, par exemple, sera mis à jour cette année seulement). Il s'agit alors d'une relative redondance de données, identifiée et intentionnelle.

Enfin, la base de données se compose également de trois "tables de travail" qui sont utilisées dans un script SQL au moment de la mise à jour des données.

Comme dit précédemment, notre application est composée d'un dossier `Db` qui gère les requêtes SQL. On peut s'attarder sur la fonction d'insertion des appels à publication qui y est définie (figure suivante).

```
function insertCall($conn, $title, $description, $link, $datefin, $datePu, $idRevue, $revue, $sourceCall){
    try{
        // On vérifie l'existence du call dans la base (titre et date)
        $existsql = "SELECT EXISTS(
            SELECT * FROM appelAPublication
            WHERE titreAppel = '$title'
            AND dateFinSoumission = '$datefin')";
        $stmt = $conn->query($existsql);
        $count = $stmt->fetch();

        // Si le call n'est pas en base, on l'insère
        if ($count[0] == 0){
            $sql = "INSERT INTO appelAPublication (sourceCall, titreAppel, resume, lien, dateFinSoumission, dateP"
            VALUES (
                '$sourceCall',";
                (isset($title)) ? $sql .= " '$title'," : $sql .= " null,";
                (isset($description)) ? $sql .= " '$description'," : $sql .= " null,";
                (isset($link)) ? $sql .= " '$link'," : $sql .= " null,";
                (isset($datefin)) ? $sql .= " '$datefin'," : $sql .= " null,";
                (isset($datePu)) ? $sql .= " '$datePu'," : $sql .= " null,";
                (isset($idRevue)) ? $sql .= " '$idRevue'," : $sql .= " null,";
                (isset($revue)) ? $sql .= " '$revue'," : $sql .= " null,";
            );
            $conn->query($sql);
        }
    } catch (Exception $e) {
        // Gestion des erreurs
    }
}
```

Figure 4: Fonction d'insertion des appels à publication

Comme le décrit le code ci-dessus, on vérifie en fait l'existence de l'appel à publication qui a été récupéré avant de l'insérer en base. Pour ce faire, on vérifie si le titre de l'appel et la date de

fin de soumission correspondent déjà ou non à une ligne de la table `appelAPublication` en base. En effet, un appel à publication peut être récupéré via plusieurs sources (expliqué par la suite) et il est donc nécessaire de vérifier que l'on ne va pas enregistrer le même appel plusieurs fois dans la base. On prend en compte ce couple d'éléments seulement car les informations peuvent sensiblement varier en fonction des sources mais il semble raisonnable d'espérer trouver le même titre (si-non quoi, un programme pourra difficilement juger si un appel est différent d'un autre ou non).

De la même manière, un ensemble de requêtes est défini dans ce même dossier : connexion à la base en PDO, mise à jour des données SJR, mise à jour des classements, récupération de la liste des call et des revues.

C. Une application modulaire

L'application développée se veut modulaire. Cela signifie qu'elle se compose d'une partie plutôt fixe et d'une partie plus variable, à laquelle on pourra ajouter des éléments facilement à l'avenir. Cette notion peut s'expliquer par le schéma suivant.

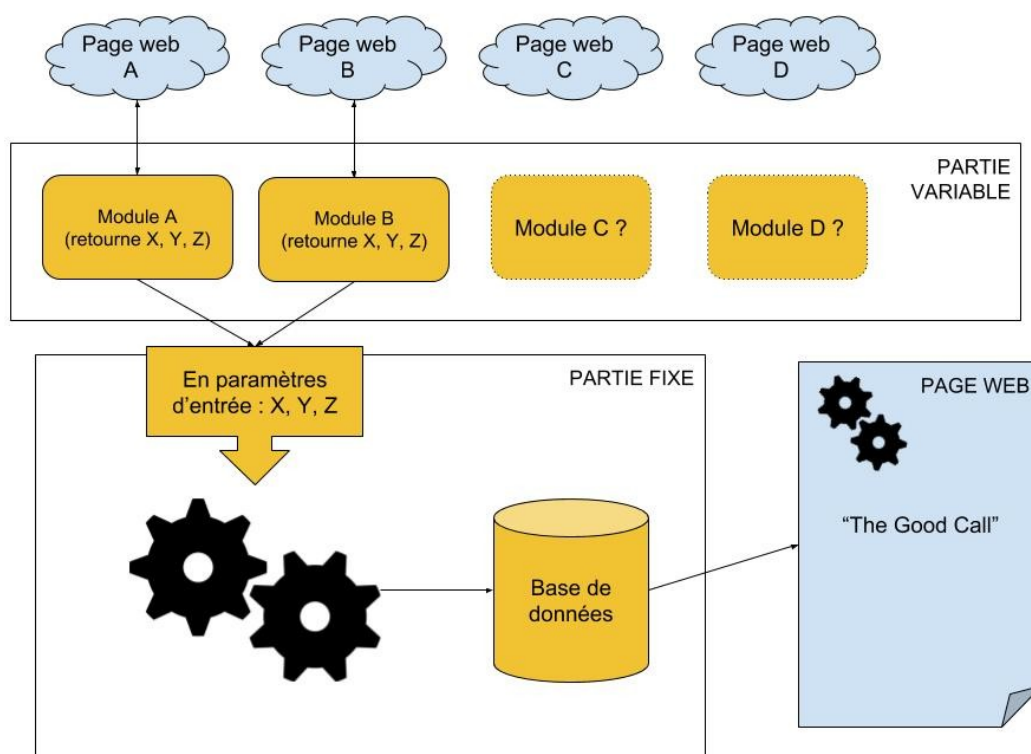


Figure 5: Visualisation de l'architecture modulaire

La partie variable correspond aux modules qui permettent le *parsing* des pages web. Chaque module est spécifique à une page web précise (à ce jour, un module pour Emerald et un module pour Elsevier). On pourrait tout à fait imaginer ajouter des modules supplémentaires pour analyser d'autres page web. Le principe de base est le suivant : chaque module, pour ajouter des éléments à la base de données, doit appeler la fonction `insertCall` comme ce qui suit :

```
insertCall($conn, $title, $description, $link, $datefin, $datePu, $idRevue, $revue, $editeur);
```

où les paramètres de la fonction correspondent à ce qui suit :

- `$conn` = une connexion PDO à la base de donnée concernée
- `$title` = le titre de l'appel tel qu'il sera enregistré en base
- `$description` = les 1000 premiers caractères de l'appel pour donner un aperçu de celui-ci à l'affichage
- `$link` = le lien vers l'appel
- `$datefin` = la date de fin de soumission de l'appel à publication
- `$datePu` = la date de publication de l'appel (il s'agit souvent de la date de *parsing*)
- `$idRevue` = l'identifiant de la revue qui est associée au *call* (voir partie sur la fonction de vraisemblance)
- `$revue` = le nom de la revue tel qu'il a été trouvé dans l'appel (au cas où on ne trouve pas de lien avec une revue présente en base)
- `$editeur` = le nom de l'éditeur, ou la source du *call* (afin d'afficher le logo de celui-ci)

De l'autre côté, la partie fixe, correspond au mécanisme qui prend en entrée les variables précisées plus haut, qui les intègre dans la base de données et qui affiche la page web. Un développeur qui souhaiterait intégrer de nouveaux éditeurs / sites internet n'aurait alors qu'à intégrer un nouveau module, sans s'occuper du reste du mécanisme de l'application. On pourrait même imaginer demander aux éditeurs de mettre en place des services web qui mettraient à disposition d'un administrateur de "The Good Call" les données dans le format désiré.

Partie 2 – La collecte des données : quelles solutions techniques ?

I) Élaboration de la base de données « revues »

A. Quelles sources ?

Nous nous sommes d'abord intéressés à la liste des revues et des produits de la recherche HCERES pour le domaine SHS1 « ÉCONOMIE et GESTION ». Cette liste « comprend 912 revues parmi lesquelles 465 revues de gestion proprement dites provenant de la liste du Collège scientifique de la FNEGE, et 832 figurant sur la liste du CoNRS. Les 912 revues classées se répartissent en 349 revues classées A (38%), 331 classées B (36%), et 232 classées C (26%) » [Source : Classement HCERES, mis à jour le 25/01/2018].

Après avoir intégrées ces premières revues en base de données, nous avons réalisé les premiers tests de mise en relation avec les appels à publication trouvés. Le résultat a été plutôt décevant puisque environ 60 % des *calls* ne faisaient pas partie du classement HCERES (et donc des classements FNEGE et CNRS) : il était alors impossible de "qualifier" ces *calls*.

Nous avons alors pris la décision de stocker en base l'ensemble des revues "SJR" (<https://www.scimagojr.com/journalrank.php>), soit plus de 34K lignes. Cela nous semble intéressant puisqu'on peut en effet récupérer, pour chacune de ces revues, un bon nombre d'information. L'exemple suivant correspond à la revue "Academy of Management Journal" :



Figure 6: Widget récapitulatif des informations relatives à la revue

"H-index : 266

Lien vers l'image récap : https://www.scimagojr.com/journal_img.php?id=20191

SJR : 8.548

Nom de l'éditeur : Academy of Management

Lien vers l'éditeur : <http://aom.org/amj/>

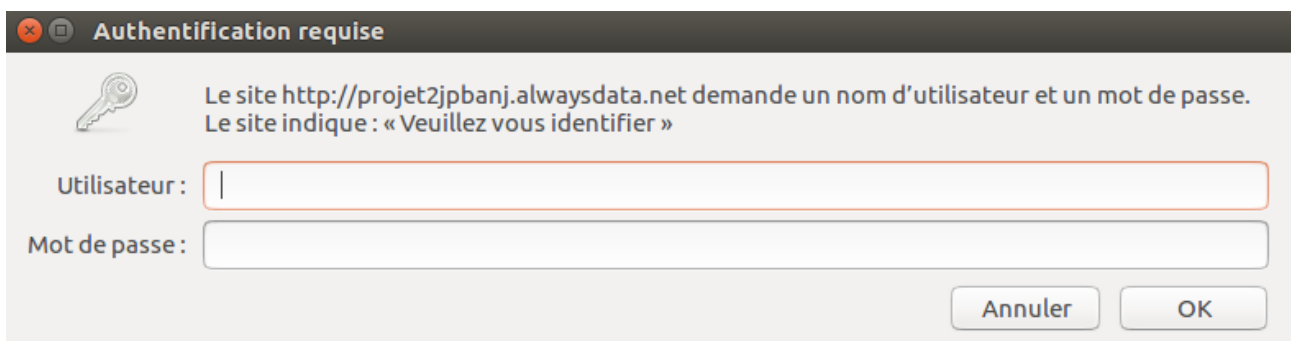
[Source : <https://www.scimagojr.com/journalsearch.php?q=20191&tip=sid&clean=0>]

En croisant ces informations avec le classement HCERES, notre ratio d'association est beaucoup plus élevé : à ce jour, nous associons 92 % des appels à une revue enregistrée en base (et, finalement, 67 % des appels stockés ne sont pas liés au classement HCERES). Il faut noter qu'un faible pourcentage d'erreur est possible lors de cette mise en relation mais ce pourcentage reste négligeable (voir partie sur la fonction de vraisemblance).

B. Comment mettre à jour cette base de données ?

Étant donné la périodicité des classements, il faut prévoir une manière de les mettre à jour, et intégrer de nouveaux classements à la base.

Côté utilisateur/administrateur, cela est rendu possible via l'url : <http://projet2jpbanj.alwaysdata.net/model/admin/updateJournal.php>.



Authentification requise

Le site <http://projet2jpbanj.alwaysdata.net> demande un nom d'utilisateur et un mot de passe. Le site indique : « Veuillez vous identifier »

Utilisateur :

Mot de passe :

Annuler OK

Figure 7: L'authentification est rendu obligatoire pour le dossier "admin"

Cette partie du site demande une identification (gérée par un couple de fichier .htaccess et .htpasswd qui sécurise la partie admin de l'application).

What do you want to update ?

☐ HCERES ranking ☐ CNRS ranking ☐ FNEGE ranking ☐ SJR data (H-Index, SJR)

Please choose your file, which must be a .csv

Be careful : HCERES, CNRS and FNEGE can only be updated once a year.

Last updates :
HCERES - 2018
CNRS - 2017
FNEGE - 2016

Be also aware that updating sjr can take a while.

Aucun fichier sélectionné.

Figure 8: Partie administrateur qui permet la mise à jour des classements

La mise à jour de ces classements est détaillée dans le manuel utilisateur / administrateur (Annexe 1 de ce rapport).

Il peut cependant être intéressant de détailler les différentes étapes du script SQL de mise à jour pour un de ces classements. Prenons l'exemple du classement HCERES par exemple :

1. Le formulaire ajoute le fichier dans le dossier `model/admin/csv`
2. Création de la table `working_classement`
3. Ajout des données du fichier csv correspondant à la table `working_classement`
4. Formatage des données : on ne garde que les caractères numériques pour l'issn (et on supprime les 0 en début de chaîne) et on met à NULL les champs vides
5. Certains issn sont en doublons (erreur de saisie par l'auteur du classement?) : on les supprime. NB : Par la suite, il pourrait être judicieux de se pencher sur cette question et de gérer différemment ces doublons
6. Test de présence des revues en base avec l'issn et le titre. Si une revue existe déjà dans la table `revue`, on ajoute son `idRevue` dans la table `working_classement`
7. Ajout dans la table `revue` des revues qui ne sont pas déjà en base (pour lesquelles l'`idRevue` est NULL dans la table `working_classement`)

8. Ajout des nouveaux `idRevue` affectés dans la table `working_classement`
9. Copie des données de la table `working_classement` dans la table `classementHCERES` pour faire l'historique
10. Mise à jour de la table `revue` avec le classement nouvellement enregistré en base

II) Récupération des « Call for papers »

A. Le *parsing* de données

Le *parsing*, en français analyse syntaxique, est une technique qui consiste à analyser un flux de caractères, fourni en entrée. Le flux entré est fractionné en bloc de données plus petits en suivant un ensemble de règles, de sorte qu'il puisse être interprété, géré et transmis plus facilement par un ordinateur. Dans notre cas nous utilisons en entrée la liste des *calls*. Nous séparerons ensuite ces *calls* un par un afin de récupérer les informations de chacun d'entre eux.

Le *parsing* va donc être très utile pour :

- Tester l'existence de données
- Manipuler un flux de données
- Extraire des informations choisies dans ce flux de données.

Comme le *parsing* repose en tout premier lieu sur les routines de gestion de chaînes, il est important de choisir un langage de programmation ayant des fonctions souples et puissantes à cette fin. Pour ce faire nous avons choisi le langage Php (comme expliqué plus haut).

Pour commencer à *parser* un document il faut choisir les technologies adaptées au type de celui-ci (par exemple on utilise DOM pour les pages Html). Ensuite, il faut analyser la page, afin d'établir une liste des points de repère exploitables. Il s'agit de mots-clés que l'automate de *parsing* devra localiser à l'intérieur de cette page (comme des balises par exemple).

B. Les technologies

Curl :

PHP supporte *libcurl*, une bibliothèque créée par Daniel Stenberg, qui permet de se connecter et de communiquer avec différents types de serveurs. Afin de pouvoir utiliser les fonctions cURL dans PHP, le paquet *libcurl* est nécessaire. L'extension cURL permet d'interagir en PHP avec tous

ces protocoles que nous employons de manière quotidienne sans avoir à gérer la connexion ou encore sans se soucier de la manière dont il faut écrire la requête ou en recevoir la réponse.

[Sources :

<https://www.julp.fr/articles/15-php-l-extension-curl.html>

<https://secure.php.net/manual/fr/intro.curl.php>]

DOM :

Document Object Model (traduisez modèle objet de document) est une spécification du W3C (World Wide Web Consortium) définissant la structure d'un document sous forme d'une hiérarchie d'objets, afin de simplifier l'accès aux éléments constitutifs du document. Plus exactement DOM est un langage normalisé d'interface (API, Application Programming Interface), indépendant de toute plateforme et de tout langage, permettant à une application de parcourir la structure du document et d'agir dynamiquement sur celui-ci.

[Source : <https://www.commentcamarche.net/>]

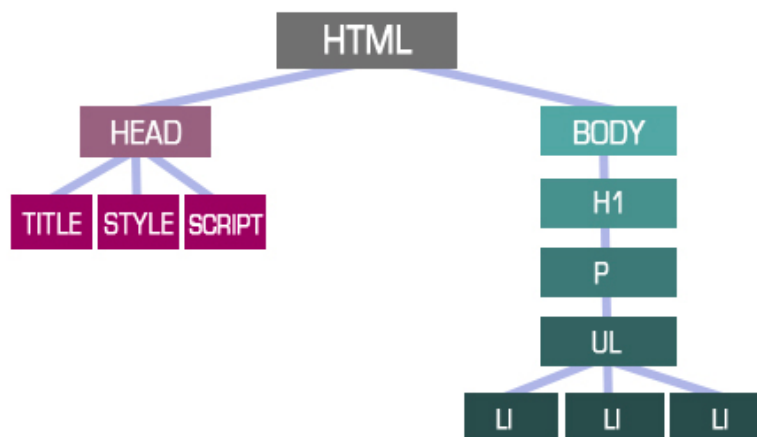


Figure 9: DOM permet une représentation d'une page web par arbre

SAX :

SAX est une API basée sur un modèle événementiel, cela signifie que SAX permet de déclencher des événements au cours de l'analyse du document XML. Une application utilisant SAX implémente généralement des gestionnaires d'événements, lui permettant d'effectuer des opérations selon le type d'élément rencontré.

Ainsi, une application basée sur SAX peut gérer uniquement les éléments dont elle a besoin sans avoir à construire en mémoire une structure contenant l'intégralité du document.

[Source : <https://www.commentcamarche.net/contents/1329-dom-document-object-model-et-sax-simple-api-for-xml/>]

Simple HTML DOM Parser :

Lors de notre recherche sur les technologie disponible pour répondre a notre besoin, nous avons trouver Simple HTML DOM Parser : une classe PHP qui permet de sélectionner des éléments de l'arbre DOM à l'aide de sélecteurs. Il nous suffit juste de télécharger cette bibliothèque à l'URL suivante : <http://sourceforge.net/projects/simplehtmldom/files/> et de l'inclure à notre fichier dont nous avons besoin.

[Source : <http://petit-dev.com/parsez-le-contenu-dun-site-avec-simple-html-dom-parser/>]

Les méthodes SAX et DOM adoptent chacune une stratégie très différente pour analyser la syntaxe des documents XML, elles s'utilisent donc dans des contextes différents. DOM charge l'intégralité d'un document XML dans une structure de données, qui peut alors être manipulée puis reconvertie en XML. Cependant pour cela, il faut que la taille de la structure représentant le document XML ne soit pas supérieure (ou pas trop) à ce que peut contenir la mémoire vive. La méthode SAX apporte alors une alternative dans les cas de figure où les documents XML sont de taille très importante (on parle alors d'adaptation au volume de données, en anglais scalability).

Les pages webs que l'on analyse dans le cadre de notre application sont tout à fait manipulables par DOM (structure peu importante). Par soucis de simplicité, nous avons donc utilisé cette technologie. Plus précisément, dans le cas du *parsing* des appels à publication Emerald, nous utilisons les flux RSS de l'éditeur, qui génère une page XML rigoureusement structurée (voir figure suivante).

```
<title>Policing Cybercrime</title>
<link>http://www.emeraldgrouppublishing.com/authors/writing/calls.htm?id=8349</link>
<description>from Policing, final submission on 1st August 2019</description>
</item>
<item>
<title>Tourism in Indian Cities</title>
<link>http://www.emeraldgrouppublishing.com/authors/writing/calls.htm?id=8353</link>
<description>from International Journal of Tourism Cities, final submission on 16th August 2019</description>
</item>
</item>
```

Figure 10: Code source du flux RSS des calls Emerald

Nous avons donc pu, dans ce cas bien précis, utilisé la technologie CURL qui permet un *parsing* simplifié dans le cas de balises bien structurées.

C. Faire le lien entre les *calls* et les revues (fonction de taux de vraisemblance)

L'un des enjeux principal du début du projet a été de trouver un moyen efficace et robuste de faire le lien entre les revues et les appels à publication enregistrés en base.

Notre première idée s'est portée sur le numéro ISSN des revues (un code de 8 chiffres servant à identifier les journaux, revues, magazines, périodiques de toute nature et sur tous supports, papier comme électronique). Cependant, les appels à publication en ligne ne donnent pratiquement jamais le numéro issn de la revue associée. En fait, l'unique moyen de tester si un appel à publication fait référence à une revue que l'on a enregistré en base est de tester le nom de la revue.

Logistical Challenges for Sharing Economies

Special issue call for papers from International Journal of Physical Distribution & Logistics Management

Special issue Call for Papers for International Journal of Physical Distribution & Logistics Management

Logistical Challenges for Sharing Economies

Submission Deadline: October 31, 2019

(All papers should be submitted to the [IJPDLM Manuscript Central website](#) between Oct 1 and Oct 31, 2019: please choose the Logistical Challenges for Sharing Economies option when submitting)

Figure 11: Exemple d'appel à publication

L'exemple ci-dessous présente un cas typique. Le *parsing* de ce *call* (Emerald) nous donne l'information que la revue associée a pour titre : « International Journal of Physical Distribution & Logistics Management ». L'idée serait donc de chercher en base la revue portant ce nom. Seulement, si nous faisons ainsi, la requête ne nous donnerait aucun résultat (la revue en question existe pourtant bien !). Pour quelle raison ?

On peut remarquer dans le titre trouvé lors du *parsing* le « & » qui correspond à un « and » en anglais. En fait, c'est bien ce type de différence qu'il a fallu détecter et tester pour faire le lien. Dans ce cas bien précis, la revue enregistrée en base porte en fait le nom suivant : « International Journal of Physical Distribution and Logistics Management ». Il a donc été nécessaire de créer une fonction de test de vraisemblance.

Description de la fonction `similar_text()`.

La fonction la plus efficace que l'on ait trouvée pour comparer deux chaînes de caractères est la fonction `similar_text()`. Cette fonction prend deux chaînes de caractères en paramètres et nous renvoie le pourcentage de similarité entre ces deux chaînes.

```
similar_text ( string $chaine_1, string $chaine_2, float $pourcent  
 ) : int
```

- `$chaine_1` : Titre de revue de l'appel à publication après le parsing.
- `$chaine_2` : Titre de la revue dans la base de données.
- `$pourcent` : Pourcent est le troisième argument. `similar_text()` va calculer la similarité en pourcentage, en divisant le résultat de `similar_text()` par la moyenne de la longueur des chaînes de caractères fournies.

Fonctionnement de la fonction `similar_text()`

Nous utilisons la fonction `similar_text()` pour récupérer l'identifiant de la revue qui a le pourcentage de similarité le plus élevé dans notre base de données avec le titre de revue de l'appel à publication après le *parsing*. On associe à cette fonction `removeSpecialCharacter()` qui permet de supprimer les caractères et expressions spéciaux qui pourraient biaiser la comparaison (page suivante).

Cette fonction est très importante puisqu'elle garantit la véracité du lien entre le *call* et la revue, ce qui est le cœur de ce projet. Après différents tests, nous avons jugé qu'un taux de vraisemblance à 90 % permettait une minimisation des erreurs. Malheureusement, il arrive encore que certains liens erronés se fassent : une revue *parsée* « European Business Review » mise en relation avec une revue « European Business Law Review » enregistrée en base de données par exemple. Au fur et à mesure des développements futurs, il semblerait donc pertinent d'affiner encore un peu plus cette fonction primordiale.

```

/* ----- Fonction pour enlever les caractères spéciaux ----- */
function removeSpecialCharacter($text) {
    $utf8 = array(
        '/[àâäåªä]/u' => 'a',
        '/[ÄÅÃÄ]/u' => 'A',
        '/[íîï]/u' => 'I',
        '/[ìïî]/u' => 'i',
        '/[èêë]/u' => 'e',
        '/[ÉÊË]/u' => 'E',
        '/[óôõöªö]/u' => 'o',
        '/[ÖÛÜÜ]/u' => 'O',
        '/[ùûüü]/u' => 'u',
        '/[ÚÛÜÜ]/u' => 'U',
        '/[!;,:*@]/u' => '',
        '/ç/' => 'c',
        '/&/' => 'and',
        '/ of /' => ' ',
        '/ the /' => ' ',
        '/Ç/' => 'C',
        '/ñ/' => 'n',
        '/Ñ/' => 'N',
        '/' => ' ', // conversion d'un tiret UTF-8 en un tiret simple
        '/[«»]/u' => ' ', // guillemet double
        '/ /' => ' ', // espace insécable (équival. à 0x160)
    );
    return preg_replace(array_keys($utf8), array_values($utf8), $text);
}

```

Figure 12: Fonction de suppression des caractères spéciaux

D. L'automatisation

Crontab est un outil qui permet de lancer des applications de façon régulière, notamment sur un serveur pour y lancer des scripts de sauvegardes, etc.

L'installation :

Bien qu'il soit souvent installé par défaut, voici les commandes pour l'installer sur Ubuntu/Debian:
apt-get install cron

La configuration :

Pour être autorisé à utiliser la commande `crontab`, il faut que l'utilisateur soit présent dans le groupe `cron`. Les fichiers `/etc/cron.allow` et `/etc/cron.deny` permettent de définir les droits d'utilisation sur `crontab`.

Si le fichier `/etc/cron.allow` existe, alors vous devez être présent dans ce fichier pour être autorisé à utiliser cette commande. Si le fichier `/etc/cron.allow` n'existe pas mais que `/etc/cron.deny` existe,

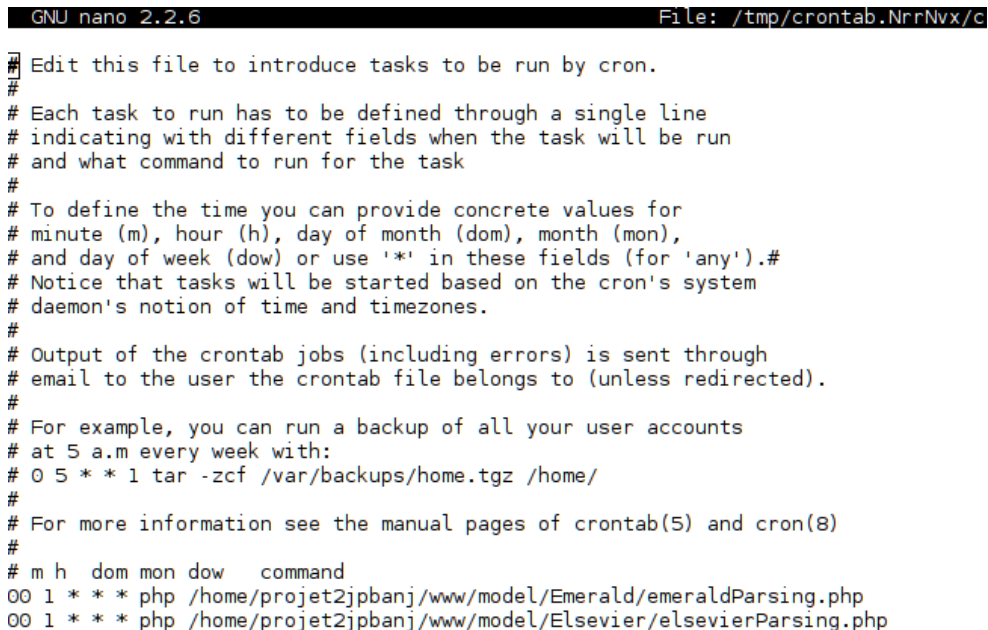
alors vous ne devez pas être mentionné dans le fichier **/etc/cron.deny** afin de pouvoir utiliser cette commande.

Il est conseillé de ne créer qu'un seul des deux fichiers et ensuite placer le nom des utilisateurs auxquels vous voulez donner/refuser l'accès à la commande **crontab**.

Les fichiers **/etc/cron.allow** et **/etc/cron.deny** gèrent seulement l'accès à la commande **crontab**. Cela signifie que si un utilisateur possède un crontab et qu'il est ensuite supprimé de la liste de **/etc/cron.allow**, les tâches seront quand même effectuées. Vous devez supprimer le crontab de cet utilisateur situé dans **/var/spool/cron/crontabs** pour annuler son exécution.

Automatisation d'un script php :

Pour éditer les actions du fichiers crontab, il faut lancer la commande : **crontab -e**



```
GNU nano 2.2.6 File: /tmp/crontab.NrrNvx/c
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow   command
00 1 * * * php /home/projet2jpbanj/www/model/Emerald/emeraldParsing.php
00 1 * * * php /home/projet2jpbanj/www/model/Elsevier/elsevierParsing.php
```

Figure 13: Lancement de la commande **crontab -e**

Afin d'automatiser des tâches, il faut ajouter, en bas du fichier, les commandes qui doivent être effectuées, en respectant une syntaxe précise.

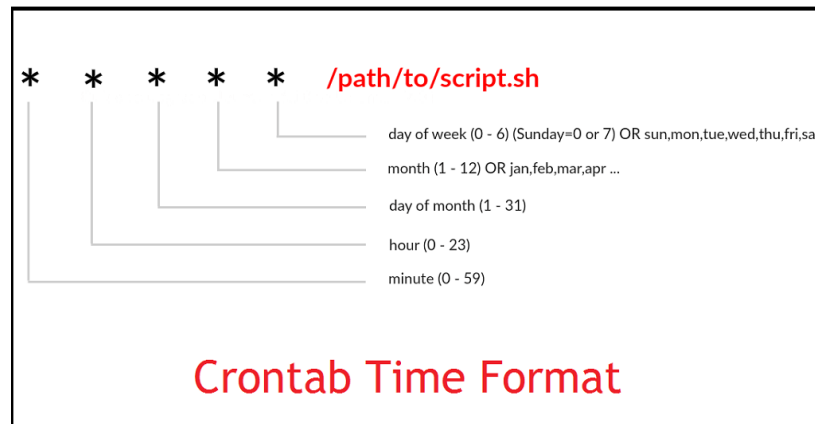


Figure 14: Syntaxe pour l'automatisation des tâches dans le crontab

NB : Si on souhaite éditer le crontab avec l'éditeur nano par défaut (plutôt que vim), il faut lancer la commande bash suivante :

```
projet2jpbanj@ssh4:~$ export EDITOR=nano
```

Dans notre cas précis, nous avons édité le crontab comme ce qui suit, afin d'exécuter les fichiers de parsing de revues emeraldParsing.php et elsevierParsing.php, tous les jours à 1h du matin.

```
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow  command
00 1 * * * php /home/projet2jpbanj/www/model/Emerald/emeraldParsing.php
00 1 * * * php /home/projet2jpbanj/www/model/Elsevier/elsevierParsing.php
```

Pour plus d'information : <https://doc.ubuntu-fr.org/cron>.

Partie 3 – Exploitation des données : les livrables

I) Affichage des données : une page web

A. Template ou pas template

La finalité du projet correspond à un affichage des données enregistrées via une page web. Nous nous sommes, pour ce faire, inspirés d'un template déjà créé : *JobListing* par ColorLib (disponible à l'adresse suivante : <https://colorlib.com/wp/template/joblisting/>). Ce *template* est disponible sous licence CC BY 3.0 : cela signifie que le *footer* de notre page doit y faire référence (ce que nous avons respecté). N'étant pas des spécialistes du *front-end* en web, nous avons en effet préféré partir de l'existant pour, ensuite, faire des modifications. Cependant, cette utilisation de *template* nous a posé quelques soucis. Nous avons en effet vite réalisé que de très nombreuses fonctionnalités étaient présentes (plusieurs dizaines de fichiers d'utilité "inconnue") et qu'il était parfois difficile de mettre en place une modification du visuel ; alors même que cette modification aurait pu paraître évidente (changement de couleur ou d'emplacement d'une div par exemple).

À de nombreuses reprises, nous nous sommes donc posés la question de l'intérêt de l'utilisation de ce *template*, en opposition à la création d'une page web créée de toute pièce. Était-ce réellement un gain de temps ? En parallèle des premières configurations de la page, nous avons même commencé à développer notre propre *design*. Néanmoins, nous avons finalement pris la décision de conserver le *template* car l'aspect visuel est plus en adéquation avec les demandes actuelles sur le web.

Il pourrait donc être intéressant, pour une personne spécialisée en développement front-end, de se poser la question de l'efficacité de ce *template* vis-à-vis d'un développement d'une application dont le visuel serait "monté de toute pièce". En effet, en utilisant ce *template*, respectons-nous bien les principes YAGNI et KISS évoqués plus tôt dans ce rapport ?

B. Recherche utilisateur : quel mécanisme (requêtes SQL) ?

Les pages web créées permettent l'interrogation de la base de données via des requêtes SQL : possibilité de tris et de recherches par mots clés. Au commencement du projet nous avions

préconisé les recherches via le serveur PHP. Seulement, nous avons découvert que les requêtes SQL sont beaucoup plus puissantes ; et nous avons donc réorganisé notre code en fonction.

Ainsi, c'est la fonction `recoveryListCalls($tri="", $str="", $op="")` qui permet la prise en compte des requêtes, puisqu'elle a pour rôle de sélectionner les valeurs désirées de la base en fonction des paramètres donnés par l'utilisateur.

→ Plus d'informations sur ces fonctionnalités dans le manuel utilisateur en annexe.

C. Hébergement

Pour l'hébergement du site web et de la base de données, nous utilisons un hébergeur en ligne Alwaysdata (<https://www.alwaysdata.com/fr/>) qui nous met à disposition un serveur apache et une base de données Mysql gratuitement.

Cet hébergement nous donne accès à 100mo gratuitement. À ce jour, notre application utilise 20,8mo de cet espace disponible. Nous estimons que cet espace sera largement suffisant pour les prochaines années de développement : la base de données et les fichiers actuels représentent en effet le plus "gros" du code (tel que l'application est pensée aujourd'hui) : viendront s'ajouter peu à peu les appels à publication, mais il ne s'agit que de données textuelles peu coûteuses.

Site web actuel : <http://projet2jpbanj.alwaysdata.net/>

En cas de désir/besoin de changement d'hébergement, il sera utile de se référer au manuel du développeur en annexe.

II) Améliorations futures

De nombreuses perspectives d'évolutions sont envisageables à l'avenir. En voici quelques unes.

- Afin de d'optimiser l'application et faciliter son développement / maintien, il serait utile d'utiliser un framework (exemple Symfony) pour mieux structurer le code et avoir une architecture plus propre.

- Effectuer plus de test afin de rendre l'application plus robuste (notamment sur la partie mise à jour des différents classements).
- Ajouter les catégories des revues afin de rendre possible la recherche par catégorie. Pour information, nous avons déjà commencé ce travail en harmonisant les catégories du FNEGE et du CNRS mais nous avons laissé en suspens cette question au moment de l'ajout des revues SJR (pour lesquelles la catégorisation est bien plus complexe).

Il faudrait, du coup, maintenant trouver une manière d'harmoniser ce classement avec les revues SJR.

nomCat	designationFR	designationEN	equivalentFNEGE
AgrEnEnv	Economie de l'agriculture, de l'envi...	Agricultural, environmental and energy econo...	OTH
CPT	Comptabilité et contrôle de gestion	Accounting and auditing	ACC
DevTrans	Economie du développement et tra...	Development and transition economics	OTH
EcoDroit	Droit et économie	Law and economics	OTH
EcoPub	Economie publique et choix collectifs	Public economics and public choice	PUB SEC
Fin	Finance et assurance	Finance and insurance	FIN
GEN	Revue généralistes en économie ...	General economics, general management	GEN MAN
GRH	Gestion des ressources humaines	Human resources management	HRM
HPEA	Histoire de la pensée économique, ...	History of economic thought, economic and b...	BUS HIST
Innov	Innovation et entrepreneuriat	Innovation and entrepreneurship	INNOV
LOG	Logistique et production	Production and operations management	LOG
Macro	Macroéconomie, économie internati...	Macroeconomics, international and monetary ...	OTH
Metrie	Econometrie	Econometrics	OTH
MgPub	Management public	Public management	PUB SEC
MKG	Marketing	Marketing	MKG
OrgInd	Organisation industrielle	Industrial organization	ORG STUD
RO	Recherche opérationnelle	Operations research	ORG STUD
SANT	Economie et gestion de la santé	Health economics and management	HLTH
SI	Systèmes d'information	Management information systems	MIS
Spatiale	Economie spatiale, économie géog...	Urban, spatial and regional economics, trans...	OTH
StratInt	Stratégie et management internatio...	Business strategy and international manage...	OTH
ThEco	Théorie économique, théorie des je...	Economic theory, game and decision theory ...	OTH
ThO	Théorie des organisations	Organization studies	GEN MAN
TravPop	Economie du travail et de la popula...	Labor and population economics	ORG STUD

Figure 15: Début de travail sur l'harmonisation des catégories des revues

- Développer la fonctionnalité de *parsing* en ajoutant d'autres sites d'éditeurs : Sage, Taylor & Francis, Wiley, etc.
- Créer un *back-end* avec une possibilité de s'inscrire en tant que chercheur ou simple visiteur, afin d'avoir accès à des fonctionnalités plus développées, comme :
 - Système de panier qui sauvegarde les calls favoris et synthétise un certain nombre d'informations
 - Système d'envoi d'email à chaque nouveau call (*newsletters*)
 - Ajouter un compte administrateur qui pourra mettre à jour la base de données via une interface web plus performante
- Continuer le développement de la page « Liste des revues » en proposant une zone de recherche en fonction de l'année et/ou de l'éditeur ; ainsi qu'en proposant un service de statistiques concernant les *calls* qui y ont participé

Il faut imaginer notre rendu de projet comme une version *alpha* de « The Good Call » qui tente de montrer la voie des possibles.

Conclusion

« The Good Call » est un projet de développement complet. D'une part, il demande la mobilisation de nombreux aspects techniques : parsing de pages web, hébergement, automatisation de tâches, scripts SQL, développement de fonctions php, etc. D'autre part, la compréhension du contexte et de l'environnement dans lequel l'application sera utilisée est primordiale pour répondre aux attentes des futurs utilisateurs. Au terme de ces quelques semaines dédiées au projet, nous présentons donc une application qui répond à la demande initiale : une page web qui recense des appels à publication (deux sources à ce jour : Emerald et Elsevier) et qui les met en relation avec une base de données de revues scientifiques. Les chercheurs ont en effet la possibilité de trouver des appels à publication qui correspondent à leur domaine de recherche (recherche par mots-clés notamment) et qui sont associés à une revue dont la réputation leur convient.

Au cours de ces semaines, des demandes supplémentaires ont émanées et des perspectives de développement complémentaires sont apparues. Bien entendu, il serait possible de continuer ce projet encore plusieurs semaines afin de l'affiner, de le compléter mais nous laissons cette tâche aux futurs étudiants qui prendront ce relais.

Ce projet tutoré a été très formateur pour les membres du groupe. Sans aucun doute, il nous a permis de progresser sur des points techniques et de découvrir de nouvelles technologies. Il est également très intéressant de se confronter à une gestion de projet en groupe, et de respecter un délai. Nous sommes convaincu de pouvoir remobiliser ces connaissances et compétences à l'avenir dans le monde professionnel.

Annexe 1 : Manuel utilisateur / administrateur

Partie 1 : Page principale

Notre page principale est composée d'un menu fixe en haut de la page, suivi par la partie de tri des calls ; viens ensuite la liste des calls et enfin un bas de page.

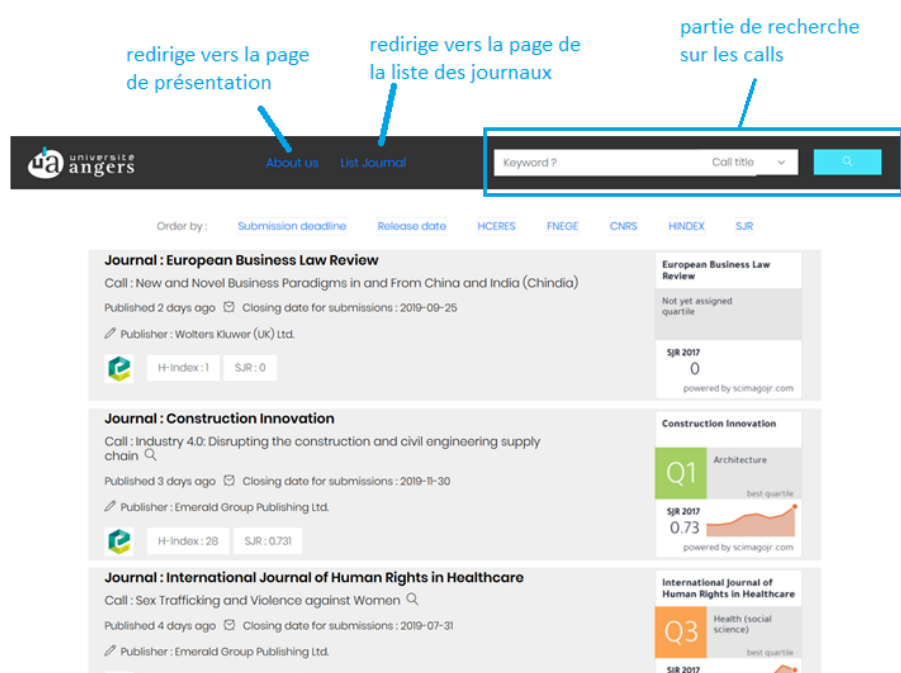


Figure 16: Page d'accueil

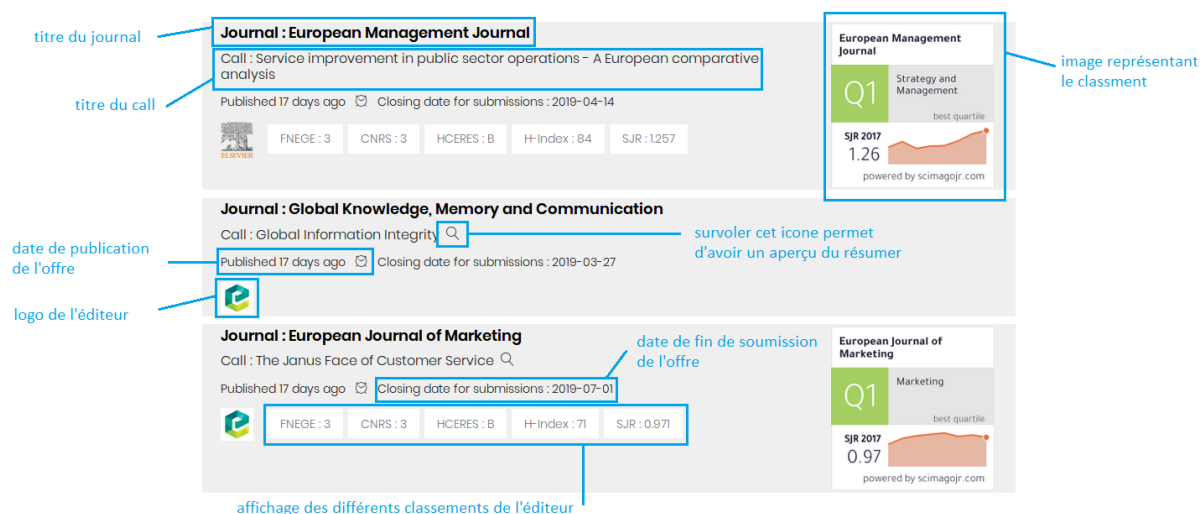


Figure 17: Affichage des calls en fonction des informations disponible

Partie 2 : Recherche de call

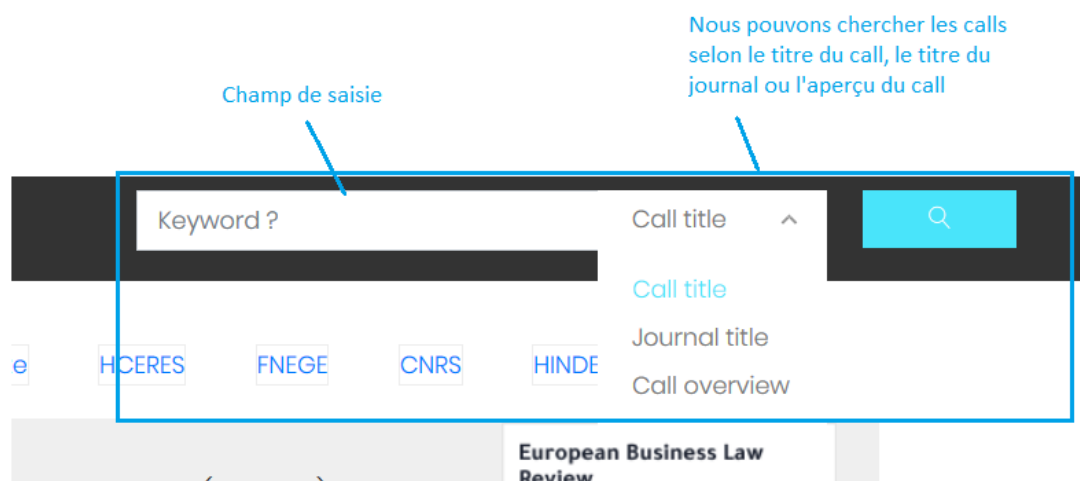


Figure 18: Possibilité de recherches par mots-clés

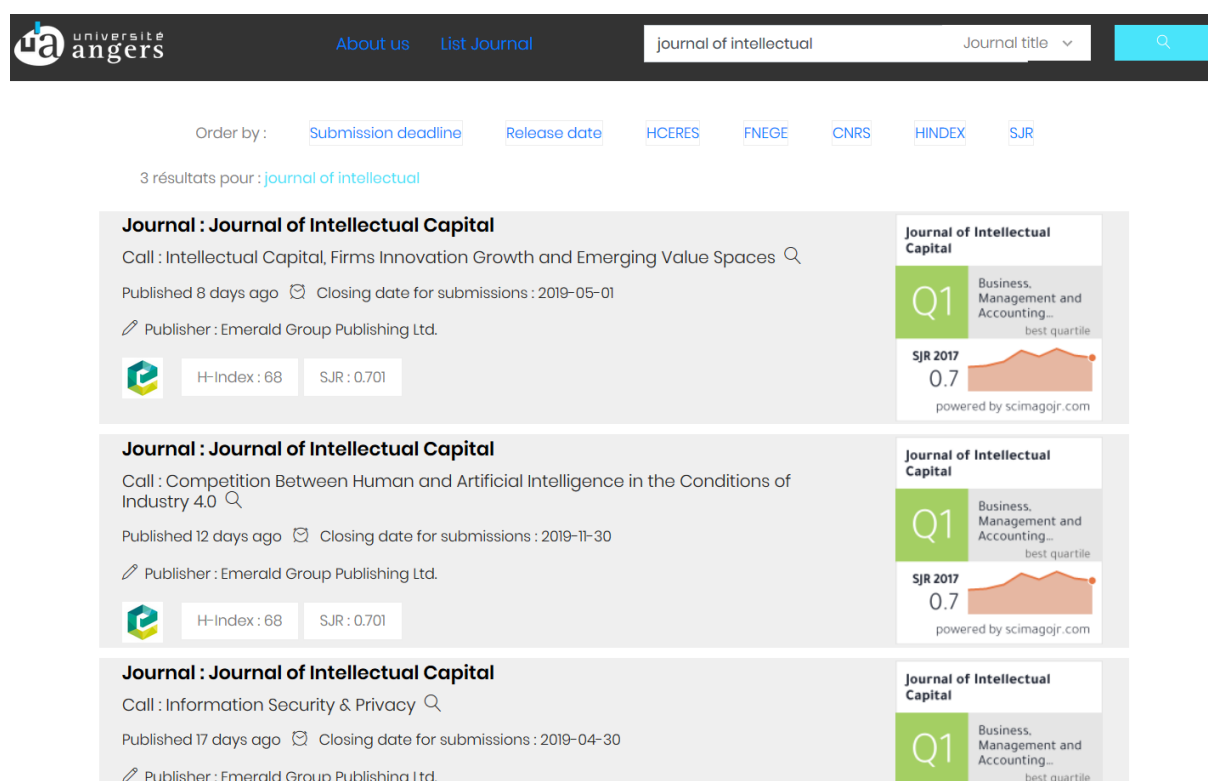


Figure 19: Exemple de recherche avec les mots « journal of intellectual » dans le titre du journal, on peut voir le nombre de r sultat trouv  par la recherche et on peut trier ces calls

Partie 3 : Tri des calls

Nous pouvons trier les calls selon 7 critères : la date de fin de soumission, la date de publication et par les différents classements. On peut effectuer qu'un seul tri à la fois.

Partie 4 : Remonter en haut de la page



Figure 20: Pour remonter en haut de la page il suffit de cliquer sur le bouton en bas à droite de la page. Le bouton apparaît une fois que l'on défile vers le bas.

Partie 5 : Page liste des journaux

Show 10 entries Search:

Titre de la Revue	Lien	Classement CNRS	Classement FNEGE	Classement HCERES	Sjr
Advances in International Accounting	Q	3	3	B	
Annales d'Economie et de Statistique	Q	2		A	
Antitrust Bulletin	Q	2		A	
Antitrust Law and Economics Review	Q	3		B	
Applied Psychology An International Review	Q	3		B	
Autrepart	Q	4		C	
Bankers Markets and Investors	Q	4	3	B	
Bulletin Francais d'Actuariat	Q	4	4	C	
Business and Economic History	Q	4		C	
Business and Information Systems Engineering	Q	4	4	C	

Showing 1 to 10 of 5,000 entries

Previous [1](#) [2](#) [3](#) [4](#) [5](#) ... [500](#) Next

Figure 21: Nous avons une page qui nous affiche la liste des journaux et permet de les trier selon les colonnes

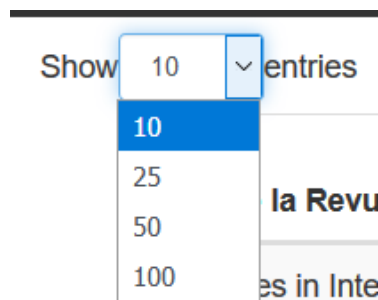


Figure 22: Nous pouvons afficher un certain nombre de journaux par page, il suffit de changer le nombre présent en haut de la page à gauche

Partie 6 : Page d'information



Figure 23: La page « about » regroupe les informations sur notre projet, il s'agit des crédits et mentions égales. Pour y accéder il suffit de cliquer sur le lien « about us » présent dans le menu en haut des autres pages du site

Partie 7 : Admin – Mise à jour des classements

L'URL suivante permet d'accéder à une partie admin du site web, qui permet une mise à jour des classements : <http://projet2jpbanj.alwaysdata.net/model/admin/updateJournal.php>

Cette page est sécurisée par une identification. Pour ajouter ou modifier un mot de passe, il convient de contacter l'administrateur du site.

Pour ajouter un classement, il convient simplement de respecter certaines conventions concernant les fichiers de classement à charger.

Pour les classement HCERES / FNEGE / CNRS : 3 colonnes

ISSN / Titre de la revue / Classement

Pour le classement SJR (à télécharger via <https://www.scimagojr.com/journalrank.php>) : 3 colonnes

Source ID (ou identifiant SJR) / Titre de la revue / ISSN

NB1 : Il faut penser à supprimer l'entête de ces colonnes

NB2 : Le chargement d'un classement SJR est très long (plusieurs heures) car il implique le *parsing* de toutes les revues pour une mise à jour des données. Une manière plus efficace d'effectuer cette mise à jour est peut-être de charger le classement SJR via l'URL donnée précédemment puis de lancer le fichier de *parsing* : `model/Sjr/sjrParsing.php` via une exécution en ligne de commande SSH. Cela permet de gérer et visualiser les potentielles erreurs plus facilement que sur un navigateur web.

Hôte SSH : `ssh-projet2jpbanj.alwaysdata.net`

Id : `projet2jpbanj` / Mdp : `projetBANJ`

	A	B	C
1	2034-9130	@GRH	3
2	1619-4500	4OR: A Quarterly Journal of Operations Research	
3	0001-3072	Abacus: A Journal of Accounting, Finance and Business Studies	2
4	1941-6520	Academy of Management Annals	1
5	0001-4273	Academy of Management Journal	1
6	1537-260X	Academy of Management Learning and Education	2
7	1558-9080	Academy of Management Perspectives	1
8	0363-7425	Academy of Management Review	1*
9	0001-4788	Accounting and Business Research	3
10	0810-5391	Accounting and Finance	3
11	0155-9982	Accounting Forum	
12	0148-4184	Accounting Historians Journal	4
13	1032-3732	Accounting History	3
14	2155-2851	Accounting History Review (ex Accounting, Business and Financial History)	3
15	0888-7993	Accounting Horizons	2
16	1744-9480	Accounting in Europe	4
17	0951-3574	Accounting, Auditing & Accountability Journal	3
18	0361-3682	Accounting, Organization and Society	1
19	1073-0516	ACM Transactions on Computer-Human Interaction	3
20	0155-9982	Actes de la Recherche en Sciences Sociales	
21	1476-7503	Action Research	4
22	0095-3997	Administration and Society	3
23	0001-8392	Administrative Science Quarterly	1*
24	0897-3660	Advances in International Accounting	3

Figure 24: Exemple d'un fichier csv bien formaté pour la mise à jour du classement FNEGE

Annexe 2 : Manuel développeur

Introduction

Ce manuel a pour vocation d'aider à la compréhension de l'application afin de pouvoir développer et améliorer ses fonctionnalités.

Partie 1 : Quels langages utilisé(s)?

L'application est développée en PHP. Le framework *bootstrap* facilite la création du design. La bibliothèque *Jquery* de *Javascript* permet d'écrire plus facilement toutes les animations sur les pages d'affichage.

Partie 2 : Quelle architecture pour le code ?

L'application est organisée en 4 dossiers :

(i) framework :

Dans le fichier `frameworkApp.php` devront être placées toutes les nouvelles fonctions écrites pour les nouvelles fonctionnalités. On placera également toutes les librairies externes dans ces dossiers (exemple : fichier `simple_html_dom.php`).

(ii) model :

Dans le dossier `model`, chaque nouveau fichier de code doit être placé dans le dossier qui lui correspond (exemple : un fichier qui effectue des requêtes doit se situer dans le dossier `Bd`).

(iii) test :

Chaque test concernant l'application doit se trouver dans le dossier `test` afin de toujours garder la même logique (robustness, test de code, etc.).

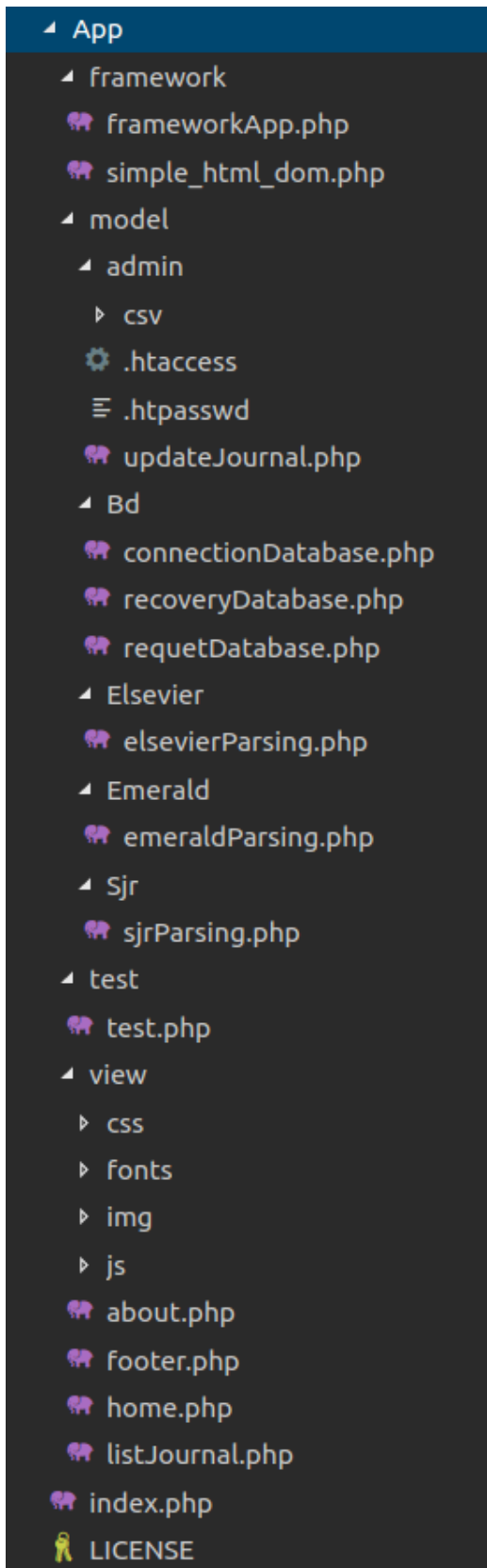


Figure 25: Architecture de l'application

(iv) view :

Le dossier view correspond à un dossier public (éventuellement d'une autre application). Il est organisé afin de séparer le css , js, ... ainsi que les vues qui sont situées à la racine du dossier (exemple : pour une nouvelle page d'affichage il faut ajouter un fichier dans le dossier view à la racine pour garder la même organisation).

(+) index.php :

Le fichier `index.php` sert de routeur. L'application a une navigation simple, par des liens de page en page.

Partie 3 : Comment créer une fonctionnalité ?

L'application se veut modulaire. Pour créer une fonctionnalité il faut tout d'abord l'organiser et la placer dans le dossier model. Le dossier model, comme expliqué dans la partie précédente, est composé de plusieurs sous-dossiers. Pour une nouvelle fonctionnalité, il faut créer un dossier avec un nom explicite. Le dossier comportera les fichiers propres à cette fonctionnalité avec une possibilité d'externaliser certaines fonctions dans le dossier framework. En effet, si ces fonctions ont pour vocation d'être réutilisées dans un autre fichier, elles doivent être écrites dans le fichier `frameworkApp.php`.

Partie 4 : Comment ajouter une nouvelle vue ?

Pour la création d'une nouvelle page, le fichier de la vue doit être sous format `.php` ou `.html`. Il doit se situer à la racine du dossier view. Si on trouve dans la page des animations en Javascript, les scripts doivent être ajoutés dans le dossier js qui se trouve dans view (voir explication dans la

partie 2 si besoin). Pour respecter le principe d’affichage il est conseillé d’ajouter le même header et footer que les autres pages. La mise en page doit se faire en harmonie avec le thème existant.

Parite 5 : Comment automatiser un script ?

Pour automatiser un script php (*parsing* d’une nouvelle page web d’appels à publication notamment), il suffit d’ajouter en fin de crontab la commande correspondante.

```
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow   command
00 1 * * * php /home/projet2jpbanj/www/model/Emerald/emeraldParsing.php
00 1 * * * php /home/projet2jpbanj/www/model/Elsevier/elsevierParsing.php
```

Parite 5 : Exportation de la base sur un nouveau serveur

```
function databaseConnection()
{
    try {
        $pdo = new PDO( dsn: 'mysql:host=mysql-projet2jpbanj.alwaysdata.net;
                           dbname=projet2jpbanj_bdd;charset=utf8', username: '176186',
                           passwd: 'projetBANJ', array(PDO::MYSQL_ATTR_LOCAL_INFILE => true));
        // force le lancement d'exception en cas d'erreurs d'exécution de requêtes SQL
        // via eg. $pdo->query()
        $pdo->setAttribute( attribute: PDO::ATTR_ERRMODE, value: PDO::ERRMODE_EXCEPTION);
        return $pdo;
    } catch (PDOException $e) {
        displayException($e);
        exit;
    }
}
```

Figure 26: Fonction de connexion à la base de données en PDO

Pour l’installation de l’application sur un nouveau serveur il suffit de copier le dossier à la racine du serveur et d’ajuster la configuration pour faire pointer vers le fichier index.php. Il est recommandé d’exporter les données de la base de données (*dump*) avant d’exporter l’application sur un autre serveur. Il ne faut pas oublier d’adapter la configuration de la fonction de connexion (voir capture d’écran). Également, il faudra penser à changer le chemin du fichier `.htaccess` dans le dossier admin (afin de gérer la sécurisation du dossier).

NB : Il est possible d’ajouter à souhait un utilisateur du serveur via la partie « Espace client → Permissions » de la partie admin de AlwaysData (<https://admin.alwaysdata.com/user/>).