

École polytechnique de Louvain

Source Code : *un catalogue open source d'exercices informatiques*

Auteurs: **Alexandre DEWIT**, **Jacques YAKOUB**
Promoteurs: **Kim MENS**, **Olivier GOLETTI**
Lectrice: **Christine JACQMOT**
Année académique 2019–2020
Master [120] en sciences informatiques

TABLE DES MATIÈRES

Résumé	ii	6.3 Conclusion	65
Remerciements	iii	7 Pour aller plus loin	66
Glossaire	iv	8 Conclusion	68
1 Introduction	1	Bibliographie	69
2 Problématique	3	Annexes	70
2.1 Situation actuelle	3	A Problématique : Images des plate-	formes
2.2 Problème	8	A.1 Practice-it	71
2.3 Défis à relever	11	A.2 Hackerrank	73
3 Approche	13	A.3 Leetcode	75
3.1 Méthodologie	13	A.4 Codeforces	77
3.2 Planning	15	A.5 Codechef	79
3.3 Organisation du travail	16	A.6 Coderbyte	81
4 Analyse des besoins	18	B Analyse Bibliographique	82
4.1 Analyse fonctionnelle	18	C Google Form : validation du projet	102
4.2 Analyse non fonctionnelle	24	C.1 Choix de vos rôles	102
4.3 Contraintes	25	C.2 Rôles : Visiteur et utilisateur	102
5 Solution	27	C.3 Visiteur et Administrateur (mo-	dérateur)
5.1 Choix technologiques	27	C.4 Questions ouvertes	109
5.2 Client	37	D Extraits de codes	112
5.3 Serveur	53	Table des illustrations	116
5.4 Divers	56	Table des illustrations des annexes	117
6 Analyse critique	59		
6.1 Validation externe	59		
6.2 Métriques	64		

Ce travail de fin d'études a été réalisé dans le cadre de notre Master en Sciences Informatiques à l'EPL de l'UCLouvain, durant l'année académique 2019-2020.

De nos jours, l'informatique se révèle être une discipline prometteuse enseignée par un nombre grandissant d'institutions pédagogiques. Toutefois, l'obtention d'exercices de programmation qualitatifs et variés constitue un problème majeur, car cela implique l'existence d'une plateforme fiable mêlant confort de recherche et référencement cohérent des ressources informatiques. Contrairement à d'autres domaines, il ne semble pas exister un consensus suffisamment partagé sur ce type de ressource. En effet, une grande partie de la problématique est occultée par des plateformes existantes : comment référencer / partager / rechercher des ressources informatiques provenant de diverses sources ? Dans ce mémoire, nous allons présenter notre solution pour pallier ce problème grâce à une plateforme basée sur un format novateur et ouvert. Ce travail consiste en 3 étapes importantes : une phase de recherche et d'inventaire constituant les besoins clés de l'application, suivi du développement de celle-ci et une phase d'analyse critique impliquant aussi bien une validation interne qu'externe afin d'en attester sa cohérence et son utilité. À défaut d'être reprise par une communauté pour en continuer le développement, nous espérons que notre approche open source puisse nourrir des réflexions ultérieures autour de cette problématique.

L'ensemble de notre prototype est disponible sur GitHub sous un compte spécialement dédié à ce mémoire : SourceCodeOER (<https://github.com/SourceCodeOER>). Notre prototype est hébergé par l'UCLouvain en mode démonstration à l'adresse suivante : <http://tfe-dewit-yakoub.info.ucl.ac.be/>

REMERCIEMENTS

Le présent travail est le fruit de la collaboration d'une multitude de personnes que nous souhaiterions chaleureusement remercier :

Merci au professeur *Kim Mens* et son doctorant *Olivier Goletti* pour leurs innombrables conseils et contributions durant l'ensemble du travail.

Merci à *Christine Jacqmot*, membre du Louvain Learning Lab, de nous avoir partagé son expérience des OER (notamment avec l'outil OER UCLouvain), de ses nombreux apports et conseils sur notre application ainsi que d'avoir accepté de lire et d'assister à la défense orale du présent travail.

Merci à *l'équipe INGI de l'UCLouvain* de nous avoir fourni une machine virtuelle, permettant ainsi d'héberger notre prototype pour le tester avec des utilisateurs.

Merci à *Jérémie Obsomer* de nous avoir donné quelques pistes de design pour la plateforme et pour la réalisation d'un poster décalé présentant notre projet.

Merci à *Alexandre Rucquoy, Alexis Van Den Bogaert, Céline Deknop, Kim Mens, Guillaume Maudoux, Adrien Widart, Olivier Goletti, Sami Bosch Isabelle Boulogne et Ophélie Hempel* qui ont pris leur temps pour tester notre prototype et nous fournir de précieux avis et retours constructifs.

Terme	Description
API	Acronyme anglais pour "Application Programming Interface". Solution informatique permettant à des applications de communiquer et d'échanger des données ou des services entre elles.
Backend	Il s'agit de la partie "immergée" de l'application (comparable à un iceberg) : l'API.
Catégorie de mots-clés CI/CD	Rubrique contenant des mots-clés de même nature. Abréviation anglophone pour "Continuous integration / Continuous Delivery", ce qui peut se traduire comme intégration continue / déploiement continu. Il s'agit donc de l'association de ces 2 concepts. L'intégration continue consistant à vérifier que chaque modification du code source n'engendre pas de défauts, notamment par l'usage de tests. Le déploiement continu consistant à réaliser le déploiement à chaque modification du code source, par le biais d'un système automatisé.
CLI	Acronyme anglais pour "Command Line Interface". Solution informatique permettant à un utilisateur de communiquer avec sa machine de manière textuelle.
CRUD	Acronyme anglais pour "Créer"(C reate), "Lire" (R ead), "Mettre à jour"(U ppdate), "Supprimer"(D elete). Il décrit les opérations de base pour la persistance des données (comme les bases de données). Un lien avec REST peut être fait par les méthodes HTTP POST, GET, PUT, DELETE respectivement.
Déploiement	Ensemble d'activités permettant la mise en service du logiciel afin qu'il soit prêt à être utilisé.
Fiche	Document décrivant une ressource informatique ainsi que les métadonnées liées à elle. Parmi les informations renseignées, on trouve notamment des mots-clés.
Frontend	Il s'agit de la partie "émergée" de l'application (comparable à un iceberg). C'est l'interface graphique avec laquelle les utilisateurs finaux interagissent.
Librairie	Ce terme désigne, comme expliqué par Wikipédia [8], "une collection de fonctions utilitaires prêtes à être utilisées par des programmes".
Maintenance	Ensemble d'activités permettant d'assurer le bon fonctionnement du logiciel.
Métadonnée	Type de données permettant de caractériser et structurer notamment des ressources informatiques.

Terme	Description
Middleware	Terme générique pour désigner tout logiciel permettant de mettre en relation plusieurs applications. Ce terme existe également sous une version francisée : le "logiciel médiateur".
Mot-clé	Mot (ou groupe de mots) permettant de caractériser une ou plusieurs fiche(s). Un mot-clé est toujours lié à une catégorie de mots-clés.
OAS	Acronyme anglais pour "OpenAPI Specification". Il s'agit d'une manière de décrire des API de type REST qui soit à la fois compréhensible pour les ordinateurs et les humains, sans que ces derniers ne doivent inspecter le code source ou jouer aux devinettes.
OER	Acronyme anglais pour "Open Educational Resources" (dans sa version française, "REL" pour "Ressources éducatives libres"). La définition officielle de l'UNESCO[7] les désigne comme étant "des matériaux d'enseignement, d'apprentissage ou de recherche appartenant au domaine public ou publiés avec une licence de propriété intellectuelle permettant leur utilisation, adaptation et distribution à titre gratuit".
QQOQCCP	Acronyme francophone pour " Q ui ? Q uoi ? O ù ? Q uand ? C omment ? C ombien ? P ourquoi ?". Celui-ci donne un cadre de questions à poser, notamment pour l'analyse.
Ressource informatique	Désigne tout matériel ayant pour attrait le domaine de l'informatique. Exemple : des exercices de programmation, des tutoriels en informatique, des algorithmes, des slides ...
REST	Acronyme anglais pour " RE presentational S tate T ransfer". Style d'architecture logicielle avec des contraintes/conventions techniques pour concevoir des services communiquant par le web.
Variable d'environnement	Ce terme désigne, comme expliqué par Wikipédia [10], "une variable dynamique utilisée par les différents processus d'un système d'exploitation".

Contexte

Dans une époque dominée par la technologie, la programmation se révèle aujourd'hui être un atout irréfutable pour se bâtir une perspective professionnelle durable. Compte tenu de cela, de plus en plus d'institutions pédagogiques intègrent une section informatique afin d'enseigner cette discipline prometteuse (comme l'illustre cet article de la Fédération Wallonie-Bruxelles [6]). Au-delà de l'aspect théorique, la programmation se découvre aussi par la pratique, ce pour quoi les chargés de cours constituent dans leur cursus un *catalogue* d'exercices de programmation pour que les élèves puissent les résoudre au cours de l'année.

Créer des énoncés est une activité chronophage et bien souvent, nous ne faisons que réinventer la roue, car une autre personne a très certainement déjà envisagé ce même exercice par le passé. Dans une ère où l'information se trouve à quelques clics de souris, pourquoi ne pas proposer un service entièrement basé sur la *contribution d'un catalogue* « *Open Source* » ? C'est donc ici que démarre notre projet : **Source Code**.

Problématique

Ce mémoire se concentre exclusivement sur la problématique de partage de ressources informatiques. Cette dernière implique un effort considérable dans la création d'une bibliothèque de ressources informatiques, car celle-ci constitue les fondations de la plateforme. En effet, partant de la bibliothèque, d'autres éléments interdépendants seront nécessaires pour assurer sa qualité et sa praticité (critères de recherche, modération, référencement ...).

Motivation

Ce mémoire, réalisé dans le cadre de notre formation, nous permet de mettre en œuvre ce qui a été enseigné durant notre cursus académique. Partant de rien, si ce n'est qu'avec la vision et les objectifs de nos promoteurs, nous devons en toute autonomie sortir de notre zone de confort pour apporter une solution au problème des ressources informatiques partagées.

Projet axé développement oblige, nous sommes donc confrontés à choisir et utiliser des technologies pour créer la plateforme. En ce sens, nous apprenons au travers de ce mémoire à parfaire nos compétences en développement et à nous familiariser avec de nouveaux outils de développement. Enfin, nous espérons que notre travail puisse être profitable au domaine des ressources partagées. Afin de renforcer cette volonté, ce projet est totalement Open Source.

Objectif

À l'instar des *Open Educational Resources (OER)*, notre plateforme souhaite offrir la possibilité à des équipes pédagogiques de collaborer sur la problématique de partage d'exercices. Cette dernière consiste à référencer des ressources informatiques, en permettant à un public varié de profiter de toutes contributions.

Approche

Avant de démarrer la réalisation de la plateforme web, nous devons impérativement définir les besoins d'une plateforme de partage de ressources informatiques. Cette phase d'analyse s'est déroulée en deux temps.

Le premier temps consistait à analyser des plateformes similaires existantes afin d'en extraire des éléments visuels ou fonctionnels pertinents pour ce projet tandis que pour le second, il nous fallait effectuer d'une part, une analyse fonctionnelle regroupant l'organisation de la plateforme et les fonctionnalités à intégrer avec la méthode **MoSCoW**, et d'autre part une analyse non fonctionnelle afin d'y relater les diverses contraintes du projet, les critères de sécurité et de maintenance.

Après cette phase d'analyse, nous nous sommes attelés à l'architecture de **Source Code** à l'aide d'un patchwork présentant les fonctionnalités essentielles de l'application côté frontend et de schémas UML pour la base de données.

Nous avons ensuite pu travailler sur l'implémentation de **Source Code** pour intégrer les fonctionnalités listées dans l'analyse fonctionnelle. Cette phase s'est déroulée durant quasiment toute l'année académique, sous l'œil avisé de nos promoteurs pour garantir une cohérence dans notre travail.

L'étape finale fut la validation du projet, où nous avons convié des utilisateurs pour tester l'application dans son entièreté. Nous voulions nous assurer que notre mémoire fasse sens et soit utile pour la problématique que nous visons. Suite aux diverses remarques, nous avons pu effectuer une dernière itération de développement afin de prendre en considération les commentaires reçus.

Contribution

Au travers de ce mémoire, nous espérons avoir contribué à la problématique des ressources informatiques partagées en proposant une solution impliquant recherche et développement dans le domaine. Aujourd'hui et plus que jamais, l'informatique se veut de plus en plus accessible, et pour répondre à cette demande grandissante, le partage de ressources, à l'instar des OER semble répondre à ce besoin en fournissant une banque de données qualitative et accessible publiquement.

Plan

Nous allons d'abord définir les besoins clés d'une application de partage de ressources informatiques en analysant une série de plateformes web existantes (cf. chapitre 2). Après cela, nous allons nous concentrer sur une analyse plus poussée de notre plateforme en y relatant les fonctionnalités à intégrer, les besoins à satisfaire pour que la plateforme puisse être utilisée par le grand public, ainsi que les contraintes qui ont été imposées (cf. chapitre 4).

Nous allons par la suite expliquer notre solution à la problématique du partage de ressources informatiques en détaillant l'architecture de **Source Code** aussi bien du côté frontend que du backend (cf. chapitre 5). Pour poursuivre, nous prendrons du recul sur le projet en évoquant la phase de validation et les métriques (cf. chapitre 6). Enfin, nous discuterons du futur de la plateforme à travers un roadmap de fonctionnalités à intégrer pour une future itération de la plateforme (cf. chapitre 7).

Dans ce chapitre, nous allons nous concentrer sur l'analyse d'applications web existantes. Cette dernière est importante, car elle nous permettra de définir les besoins clés d'une application de partage de ressources informatiques. Cette étape correspond à la première phase de notre planning (cf. figure 3.3 en rouge). Vous pouvez considérer ce chapitre comme une introduction au chapitre 4, où nous développerons plus en détail les besoins spécifiques de **Source Code**.

2.1 Situation actuelle

Pour mieux comprendre l'univers des ressources informatiques, nous avons analysé différentes plateformes. Bien que ces dernières n'apportent pas de solution précisément axée sur notre problématique, elles nous ont tout de même permis d'extraire des éléments pertinents à la création de notre plateforme : **Source Code**.

Le tableau ci-dessous regroupe les différents sites web parcourus. Dans les sous-sections suivantes, nous listons les qualités et les défauts de chacune de ces plateformes. Vous retrouverez les captures d'écran représentant les différentes plateformes dans l'annexe A.

Plateforme	Lien
Practice-it	https://practiceit.cs.washington.edu/problem/list
Hackerrank	https://www.hackerrank.com/dashboard
Leetcode	https://leetcode.com/problemset/all/
Codeforces	https://codeforces.com/problemset/
Codechef	https://www.codechef.com/problems/challenge/
Coderbyte	https://coderbyte.com/challenges/

TABLE 2.1 – Les différentes plateformes analysées

Avant d'aller plus loin, nous tenons à mentionner l'outil OER UCLouvain hors du champ de notre analyse. En effet, notre projet est une version simplifiée de leur application, car nous ne traitons qu'un seul domaine des OER : les ressources informatiques. Par conséquent, le système de recherche que le site propose ne correspond pas à la vision que nous avons pour notre type de plateforme étant donné qu'il est beaucoup plus complexe que ce que nous voulons mettre en place : un catalogue de ressources informatiques. Néanmoins, nous avons pu discuter avec *Christine Jacqmot*, membre du *Louvain Learning Lab*, afin de lui présenter notre projet et de voir comment se comporte l'outil OER UCLouvain côté administration. Ses conseils nous ont aidés à améliorer notre prototype et dans le cas échéant, à rajouter dans le chapitre 7 des précieuses pistes d'améliorations.

Practice-it

Practice-it est une plateforme permettant de résoudre des problèmes en Java en ligne. Comme le site le relate : *la plupart des problèmes viennent des cours d'introduction en Java de l'université de Washington.*

Qualité(s)

- Les problèmes peuvent être résolus de manière interactive depuis la plateforme.
- La progression d'exercices résolus est sauvegardée.
- Les problèmes sont organisés sur plusieurs niveaux : par cours ou par l'année d'édition du livre contenant ces exercices -> Chapitres -> Thématiques -> Exercices.
- Accès à une recherche avancée par mots clés ou titre de recherche.
- Un exercice contient les informations suivantes :
 - Titre
 - Auteur
 - Date de modification/création
 - Langage de programmation

Défaut(s)

- Interface très simpliste.
- **On doit se connecter** pour accéder à la recherche avancée.
- Pas de catégorie de mots-clés pour la recherche avancée : les éléments sont affichés de manière pêle-mêle dans une liste déroulante conséquente.
- La recherche de problèmes sans la recherche avancée n'est pas pratique.

Hackerrank

Cette plateforme a la volonté d'aider les développeurs à améliorer leurs compétences en programmation. Elle est aussi faite pour que les grandes entreprises puissent facilement trouver des développeurs ayant les compétences requises. Seule la partie "*Practice*" du site est intéressante pour notre problématique.

Qualité(s)

- Le tableau de bord est séparé en plusieurs sections pratiques :
 - Compétences de l'utilisateur
 - Les différentes compétences proposées (langage de programmation, mathématiques, ...)
 - Tutoriel
- Une barre de recherche est disponible pour trouver des exercices / challenges par leur titre.
- Lorsqu'on choisit un langage de programmation, l'interface de recherche présente trois catégories de mots-clés :
 - Statut (résolu ou non)

- Difficulté
- Sous-domaine (thématique)
- Parmi les mots-clés, on peut en sélectionner plusieurs dans la même catégorie (OU logique) tout en sélectionnant d'autres mots-clés dans une autre catégorie (ET logique entre les différentes catégories).
- On peut résoudre de manière interactive le challenge depuis la plateforme (à condition d'être connecté).
- On peut noter le challenge sur 5 étoiles.
- Les tests cases et énoncés du challenge peuvent être téléchargés.
- Une section "*Discussions*" est mise à disposition pour chaque challenge.

Défaut(s)

- Peu de catégories et de mots-clés pour rechercher un challenge.
- Il faut obligatoirement se créer un compte pour suivre un tutoriel ou soumettre un challenge.
- La moyenne de note d'un challenge ne figure pas dans l'interface. On peut noter un exercice, mais on ne peut pas connaître "l'avis" général.

Leetcode

Cette plateforme cherche à améliorer les compétences des développeurs en proposant des exercices et tutoriels sur des thématiques variées. Elle permet d'étendre ses connaissances en programmation en vue de préparer un prochain entretien avec une entreprise.

Qualité(s)

- Une barre de recherche est disponible pour trouver des exercices par leur titre, description ou identifiant (ID).
- Choix parmi différentes catégories :
 - Algorithmes
 - Base de données
 - Terminal
 - Concurrency
- Deux grandes catégories de mots-clés sont présentes :
 - Topics
 - Entreprises
- Chacun des mots-clés est associé à un nombre correspondant au nombre d'exercices contenant ce mot-clé.
- Possibilité de filtrer les exercices en fonction de la difficulté, du statut (résolu, à faire, essayé).

- On peut résoudre de manière interactive le challenge depuis la plateforme (à condition d'être connecté).
- On peut liker ou disliker un exercice.
- Une section "*Discussions*" est mise à disposition pour chaque exercice.
- Chaque exercice peut se résoudre avec son langage de prédilection.

Défaut(s)

- Malgré un choix varié de mots-clés, ces derniers ne sont pas classés dans des catégories différentes.
- Les mots-clés ne sont pas directement accessibles depuis l'interface, car il faut scroller.
- Aucune spécificité au niveau des langages de programmation (exemple : des exercices en C pour la gestion de la mémoire).
- Il faut obligatoirement se connecter pour s'exercer avec la plateforme.

Codeforces

CodeForces est une plateforme créée par une communauté de programmation axée sur la compétition et les concours.

Qualité(s)

- On peut voir le nombre d'utilisateurs ayant résolu le challenge.

Défaut(s)

Cette plateforme est un exemple concret de site web que nous ne voulons pas créer tant l'ergonomie et les différentes fonctionnalités ne sont pas assez matures ou pertinentes à notre égard.

- La recherche n'est pas claire du tout. Pour afficher la barre de recherche, il faut cliquer sur une flèche bleue (pas logique).
- On peut filtrer les challenges en cliquant sur un des mots-clés présents sur un des exercices, mais on ne peut pas y rajouter un autre mot-clé (cela efface la précédente sélection).
- Possibilité de rajouter des mots-clés avec un panneau latéral (sur la droite), mais les options sont très sommaires (difficultés, mots-clés dans une liste déroulante conséquente). C'est aussi le seul moyen de prendre connaissance avec la panoplie de mots-clés proposés.
- On peut filtrer les exercices par niveau de difficulté, mais il n'y a aucune indication pour nous dire ce que cela représente. Concrètement, il faut entrer un nombre entre plancher X et plafond Y (le plus grand nombre enregistré pour identifier la difficulté d'un exercice était de 3800).
- Aucun moyen de réinitialiser la recherche. La seule manière est de retirer manuellement les mots-clés sélectionnés et le titre de recherche.

Codechef

CodeChef est un site de programmation compétitif. Il s'agit d'une initiative éducative à but non lucratif de Directi, qui vise à fournir une plateforme pour les étudiants, les jeunes professionnels du logiciel, afin qu'ils puissent s'exercer et affiner leurs compétences en programmation grâce à des concours en ligne. En outre, le programme "CodeChef For Schools" vise à atteindre les jeunes étudiants et à inculquer une culture de la programmation dans les écoles indiennes. (texte tiré de Wikipedia)

Qualité(s)

- La recherche est basée sur le niveau de difficulté : débutant, facile, normal, difficile et challenge.
- On interagit de manière interactive avec la plateforme pour soumettre son code.
- On peut filtrer des exercices par mots-clés en utilisant une interface dédiée.
- On peut connaître le nombre d'exercices se rapportant à un mot-clé.
- Possibilité de filtrer par auteur.

Défaut(s)

- Il faut obligatoirement se connecter pour soumettre un exercice.
- Il n'y a pas de catégorie de mots-clés, ils sont tous affichés de manière pêle-mêle depuis l'interface.
- Pas de barre de recherche pour trouver un exercice en fonction de son titre.

Coderbyte

CoderByte est une plateforme proposant divers challenges de programmation en vue d'améliorer les compétences des développeurs et les préparer à une future interview avec une entreprise.

Qualité(s)

- La plateforme propose un système de mots-clés avec 3 catégories : difficulté, entreprise, mots-clés. Il se situe dans un panneau latéral accessible sur le côté droit de l'écran.
- Parmi les mots-clés, on peut en sélectionner plusieurs dans la même catégorie (OU logique) tout en sélectionnant d'autres mots-clés dans une autre catégorie (ET logique entre les différentes catégories).
- On peut résoudre les challenges directement depuis la plateforme et sélectionner son langage de prédilection.
- Pas besoin de se connecter pour résoudre un challenge.
- On peut participer à une discussion autour d'un challenge particulier.

Défaut(s)

- Pas de barre de recherche pour trouver un exercice en fonction de son titre.
- Classification sommaire des mots-clés (les langages de programmation sont mélangés avec les thématiques, ...).

Conclusion sur l'analyse

Cette analyse nous a permis de partir sur de bonnes bases pour la création de notre application web. Nous voulions nous inspirer de certaines interfaces proposant de bonnes idées au niveau de la recherche pure, car cette dernière sera l'une des fonctionnalités principales de **Source Code**.

Certaines fonctionnalités comme l'espace de discussion sur un challenge spécifique ou encore la soumission de sa solution directement sur la plateforme demeurent de bonnes idées, mais pas nécessaires à l'élaboration d'une application comme **Source Code**, car l'objectif premier est le référencement de ressources informatiques avant tout. Dans une perspective future, l'intégration de ce genre de fonctionnalités pourrait bien sûr être envisageable, ce qui rendrait ce projet encore plus flexible.

Avec le support de cette analyse, nous avons pu constituer une liste des éléments importants devant être intégrés au projet. Cela a aussi servi à mettre sur papier nos idées à travers un patchwork que vous pourrez consulter dans la section 2.2.

2.2 Problème

Compte tenu de l'analyse des plateformes précédentes, nous allons nous intéresser aux obstacles à franchir pour créer une plateforme de partage de ressources informatiques, en particulier la nôtre.

La recherche de ressources informatiques demeure la base de **Source Code**. Par conséquent, l'élaboration d'une **bibliothèque** proposant une recherche efficace et pratique est un premier jalon à traverser. Il faut que l'utilisateur comprenne assez rapidement comment interagir avec l'interface pour trouver des ressources informatiques correspondant à ses critères de recherche (exemples : exercices avec le langage C, thématique sur les pointeurs, ...).

Se pose ensuite la question du **contenu d'une fiche**. Cette dernière contiendrait probablement un titre et une description (énoncé d'un exercice, tutoriel), mais le plus important reste son **référencement** à travers l'application. Il s'agirait alors d'intégrer un système de mots-clés cohérent permettant de retrouver rapidement une fiche dans la bibliothèque.

Le **système de mots-clés** serait incontestablement le cœur de l'interface de recherche de ressources informatiques. Non seulement il devrait être facilement accessible dans l'interface (ergonomie), mais il devrait aussi proposer une *taxonomie* compréhensible par les utilisateurs de la plateforme. Nous entendons par *taxonomie*, une catégorisation des différents mots-clés pour faciliter la recherche (exemples : difficulté, cours, langage, ...).

Pour qu'une application de cette envergure soit facilement maintenable en termes de contenu, un autre besoin à satisfaire serait la création d'une **interface de gestion** permettant d'ajouter des ressources, de créer des mots-clés, ... Sans cette interface, l'intérêt de la plateforme serait moindre, car l'ajout de ressources serait rendu difficile, voire impossible à réaliser pour le grand public.

Étant donné qu'une multitude d'utilisateurs ajouteraient du contenu sur la plateforme, un système de **modération** semble important à mettre en place. Dans cette perspective, il est alors question de déterminer une organisation sous *différents types d'utilisateurs* avec des privilèges propres (simple visiteur, utilisateur avec un compte, administrateur). Sans cette modération, la plateforme ne pourrait pas prétendre à du contenu de qualité, car tout type de ressource serait alors publié dans la bibliothèque.

Pour poursuivre avec le critère de *qualité* du contenu, il serait judicieux d'élaborer un **système de statuts** pour les ressources publiées. Les ressources validées seraient ainsi visibles dans la bibliothèque (accessible à tout public) tandis que les autres (non valides, en attente, ...) seraient uniquement visibles depuis les interfaces de gestion. Reste à déterminer le nombre de statuts et leur utilité.

Toujours dans l'optique d'améliorer la qualité du contenu de la plateforme, un **système de vote** devrait être mis en place afin de laisser la communauté s'exprimer sur la pertinence d'une ressource informatique. La modération est une première instance de validation des ressources informatiques ajoutées sur la plateforme, mais nous estimons que la communauté doit aussi pouvoir donner son avis. Cela permettrait d'une part au créateur de la ressource informatique d'améliorer sa fiche et d'autre part aux administrateurs d'émettre une nouvelle évaluation de la ressource informatique.

Afin de vous donner une petite idée de ce que nous avons imaginé après la première analyse des plateformes, nous vous proposons quelques illustrations de notre patchwork pour la partie bibliothèque et la consultation d'une fiche. **Ceci n'est pas un design final de Source Code**, mais plutôt une mise sur papier de notre conception de l'application web au vu des critères exposés précédemment.

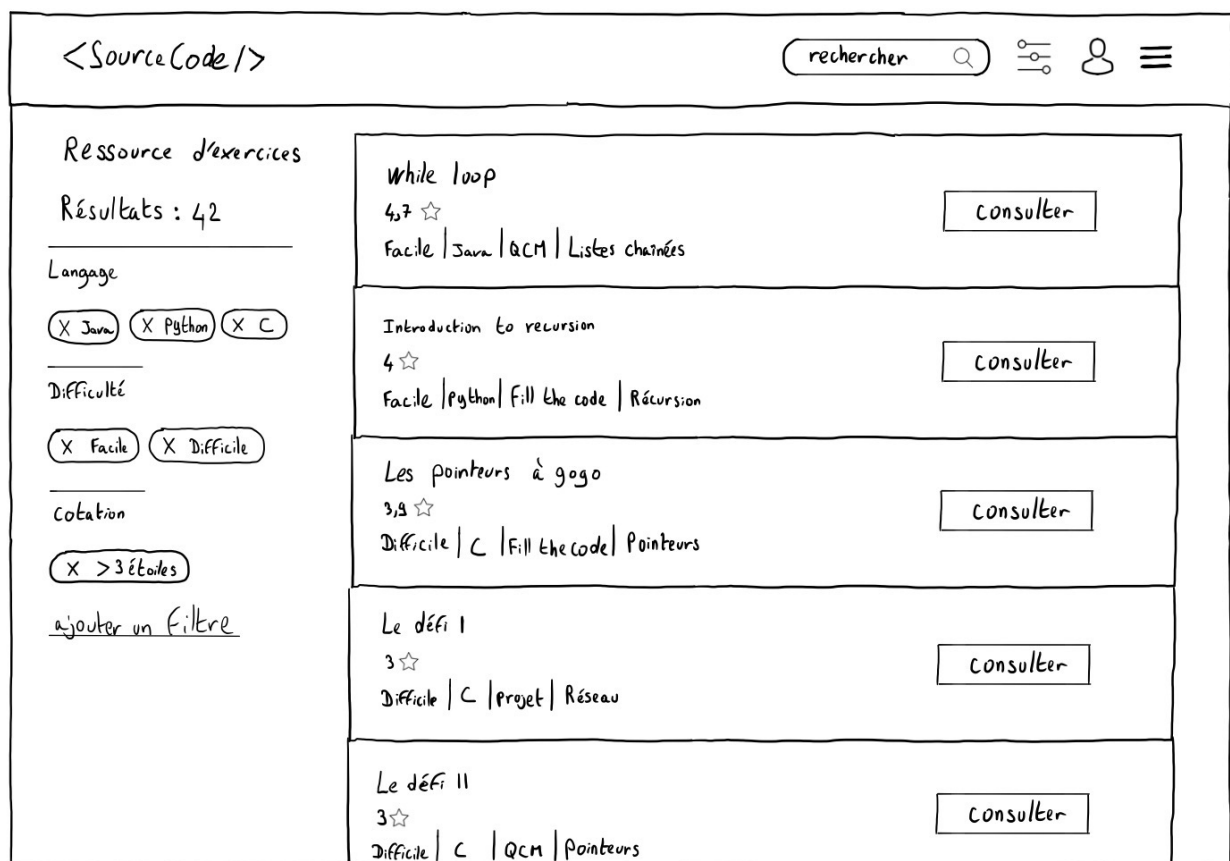


FIGURE 2.1 – La page Bibliothèque

<SourceCode/>
rechercher

Tous les filtres
effacer 0
annuler
appliquer

Langage

Rechercher un filtre

☐ Java
☐ Python
☐ Scala
☐ C
☐ Non défini

Difficulté

Rechercher un filtre

☐ Facile
☐ Normal
☐ Difficile

Type d'exercice

Rechercher un filtre

☐ QCM
☐ Exercice papier
☐ projet
☐ Fill the code
☐ Mission

Plateforme

Rechercher un filtre

☐ Ingenious
☐ Facebook
☐ Google
☐ Non défini

Auteur

Rechercher un filtre

☐ Kim Jong Un
☐ Rambo
☐ Mark Zuckerberg
☐ John Doe
☐ Anonyme

Cotation

☐ 5 étoiles
☐ plus de 4 étoiles
☐ plus de 3 étoiles
☒ moins de 3 étoiles
☒ par défaut

Thématique

Rechercher un filtre

☐ Paradigmes de programmation
☐ Listes chaînées
☐ pointeurs
☐ Récursion
☐ Boucles

Institution

Rechercher un filtre

☐ UCLouvain
☐ ULB
☐ UMONS
☐ IPET

FIGURE 2.2 – Représentation du système de filtres avec les mots-clés

<SourceCode/>
rechercher

While loop
4.7 ☆

Description

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Image

Tags

Auteur

Difficulté

Type d'exercice

Langage

Thématique

Plateforme

Institution

Kim Jong Un

Facile

QCM

Java

Listes chaînées

Ingenious

UCLouvain

Source

⬇ Télécharger la source

🔗 Lien vers la source

Votre cotation

☆☆☆☆☆

FIGURE 2.3 – Page d'une fiche

2.3 Défis à relever

Après avoir formalisé les besoins d’une application de partage de ressources informatiques, nous allons maintenant lister ce que nous allons mettre en place pour réaliser notre plateforme. Notez que cette liste est introductive à l’analyse fonctionnelle relatée dans la section 4.1, mais nous avons besoin de cette étape pour avoir une vue au global de **Source Code** avant d’aller un peu plus dans les détails.

- Proposer une interface simple et ergonomique à destination d’un public varié. Pour rappel, la cible principale est l’équipe pédagogique (professeurs, doctorants, formateurs, ...), le reste étant majoritairement des étudiants. Peu importe le type d’utilisateur, l’outil doit être suffisamment clair que pour comprendre les fonctionnalités clés de l’application (recherche de ressources informatiques, création de fiches, gestion des favoris,...).
- Créer un système de mots-clés évolutif et extensible. Nous entendons par là la possibilité de créer/modifier/proposer des mots-clés, créer/modifier des catégories de mots-clés, ... Cela permettra ainsi de ne pas figer l’application sur une poignée de mots-clés mais aussi de s’adapter à de nouvelles conventions en renommant certains d’entre eux.
- Proposer une interface de modération complète pour gérer les mots-clés, les catégories de mots-clés, les ressources informatiques et les utilisateurs. En outre, les interfaces d’administration ne seraient disponibles qu’à partir du moment où l’utilisateur possède un compte éligible d’administrateur. Les autres interfaces de gestion pour ses propres ressources informatiques devraient être accessibles aux utilisateurs possédant un simple compte utilisateur.
- La recherche de ressources informatiques doit fonctionner de manière efficace. À ce propos, le site *coderByte* propose une bonne façon de rechercher des exercices, avec un panneau latéral comportant des mots-clés rangés dans des catégories. Nous allons donc nous inspirer de cette pratique pour naviguer dans la bibliothèque, tout en ajoutant une barre de recherche pour naviguer dans les titres des ressources informatiques.
- Créer et intégrer à l’application un outil pour importer/exporter rapidement des ressources informatiques, des mots-clés et catégories de mots-clés. Ajouter des ressources informatiques peut être chronophage dans le cas où il y aurait quelques dizaines de fiches à créer depuis l’interface de gestion. Nous souhaitons donc automatiser le processus de création en proposant un format d’import acceptant les éléments précédemment cités. À l’inverse, exporter des ressources informatiques depuis notre plateforme devrait être rendu possible afin de partager les ressources avec d’autres plateformes par exemple.
- Une piste non explorée par les différentes plateformes précédemment analysées est de proposer une recherche de ressources informatiques plus attractive. Nous pensons donc à intégrer un historique et un système de favoris pour naviguer plus aisément dans les recherches effectuées antérieurement.

Pour conclure ce chapitre et cette section, il est important de noter que la création d’une application web à destination d’un public varié demande un investissement conséquent en termes de temps. Il s’agit de créer une interface satisfaisant les attentes de chaque groupe d’utilisateurs tout en proposant un projet maintenable et évolutif pouvant satisfaire les besoins futurs.

Chaque page doit être réfléchie en matière d’ergonomie et de design pour que l’utilisateur ait envie de continuer à naviguer sur la plateforme. Côté développement, l’ensemble du code doit être suffisamment modulaire afin d’apporter de nouvelles fonctionnalités de manière aisée (évolution de l’application). Au plus du temps sera investi dans un tel projet, au plus ce dernier sera mature et agréable à prendre en main pour chaque classe d’utilisateur, car nous aurons appris à comprendre leurs besoins. Nous pensons particulièrement à la recherche de ressources informatiques et à la gestion de ces mêmes ressources informatiques, mots-clés, catégories de mots-clés.

Source Code n’aura pas la prétention d’être en version finale, il se positionnera plus sur la dénomination de **prototype**. En effet, notre volonté principale est avant tout de contribuer au domaine des ressources partagées, car nous pensons que ce dernier est porteur d’une riche communauté d’entre-aide. C’est pourquoi le titre de notre mémoire contient les mots ***open source***, car nous prenons exemple sur ce mode d’organisation pour créer cet outil de partage.

« Les plans sont inutiles, mais la planification est indispensable »

Dwight David Eisenhower

En parallèle à nos différentes phases de planning (cf. figure 3.3), nous devons nous appliquer à mettre en place une méthode de travail efficace. Que ce soit au niveau de notre binôme ou des meetings prévus chaque semaine avec nos deux promoteurs, il fallait trouver une manière de s'organiser afin de mener notre mémoire sur la bonne voie. Ce chapitre a donc pour but d'expliquer notre méthodologie de travail (méthode utilisée, phases du travail, communication), l'élaboration d'un planning et l'organisation du travail en interne.

3.1 Méthodologie

De par la nature ouverte du projet à accueillir continuellement des nouvelles fonctionnalités et changements, une méthode Agile a été plébiscitée (qui s'oppose aux méthodes classiques plus rigides comme *waterfall*). Comme expliqué par le manifeste Agile[1], un certain nombre de principes sont présents et nous n'évoquons ici que les plus importants :

Satisfaction du client avant tout Il s'agit du principe sacro-saint de la méthode Agile[1].

Au lieu d'avoir un planning bien ficelé, il faut accepter l'ajout de nouvelles fonctionnalités et des changements (même tardifs) pour viser ce but.

Itérations courtes En méthode Agile[1], il est préconisé d'adopter des cycles de développement de faible durée afin de notamment identifier et corriger très tôt des bugs (une expression anglophone y est dédiée : "fail fast, fail often"). Puisque nous rencontrons nos promoteurs à peu près toutes les semaines, c'est naturellement ce rythme qui s'est imposé pour nos itérations.

Livraison continue Tout au long du projet, de nombreux utilisateurs dont nos promoteurs ont utilisé intensivement l'application et nous ont livré quantité d'avis et retours. Il était donc nécessaire de déployer des versions de l'application très régulièrement pour montrer son avancement et donner la possibilité d'essayer de nouvelles fonctionnalités par les utilisateurs : nous avons fini par automatiser une grande partie de ce processus (nous expliquerons en détail dans la section 6 comment cela a été réalisé concrètement).

Phases du travail

Pouvoir baliser les grandes étapes de ce travail est indispensable, car cela permet de suivre en outre l'évolution générale tout au long de ce projet. Bien que l'étape de développement (implémentation) suive les principes d'une méthode Agile[1], il est inévitable de passer par les différentes étapes du développement logiciel (analyse, conception, implémentation, validation et déploiement).

Ceci a été motivé par de multiples raisons. On citera notamment le vaste contexte très ouvert de la recherche et référencement des ressources informatiques, pour laquelle il a fallu accorder

une période de réflexion et d'analyse assez conséquente.

Un autre facteur a été que de nouveaux besoins et idées ont émergé au fur et à mesure de la construction de notre solution/prototype. Ces circonstances nous ont donc contraints à revoir constamment notre approche et notre organisation. Une manière de l'illustrer est de considérer la phase de validation (cf. section 6), consistant à certifier que la solution ainsi développée correspond bien aux critères techniques qu'aux critères de satisfaction client. Le versionnage (c.-à-d. le fait de gérer plusieurs versions) de notre solution/prototype a permis de récolter de précieux retours d'utilisateurs au plus vite, tout en permettant d'identifier d'éventuelles régressions entre une version et une autre.

Communication

En méthode Agile[1], il y a de manière générale 2 questions récurrentes, et ce peu importe la taille et la nature des projets :

- Où on en est ?
 - À quelle vitesse on progresse ?
 - Qu'avons-nous réalisé ?
- Quand on délivre le produit ?
 - Que reste-t-il à faire ?
 - Comment estimer le temps/travail nécessaire pour réaliser les différentes tâches ?

Pour y répondre, il convient d'utiliser des moyens/outils de communication suffisamment explicites pour tous les acteurs de ce projet. Étant donné que notre solution est hébergée sur Github, nous avons décidé d'utiliser les fonctionnalités suivantes de Github : le "Project Board" ¹ et les issues ². (dont vous pourrez trouver une illustration respectivement aux figures 3.1 et 3.2)

Comme nous pouvons le voir, par l'usage de ces outils/moyens, l'état et la progression générale du projet sont accessibles au plus grand nombre :

- Avec le "Project Board", on représente les tâches sous la forme de cartes amovibles dans un tableau, avec les caractéristiques suivantes :
 - Chaque tâche est catégorisée dans une colonne bien précise ("to do" / à réaliser, "in progress" / en cours, "done" / réalisée, "enhancements" / améliorations)
 - Une barre de progression nous notifie de l'avancement général du projet
- Avec les issues (figure 3.2), chaque tâche/problème dispose des éléments suivants :
 - un titre pour expliquer brièvement celle-ci
 - une étiquette en couleur pour notifier sa nature (problème, amélioration, documentation, etc.)
 - la personne assignée sur celle-ci
 - le statut de celle-ci (ouverte/fermée)

¹<https://help.github.com/en/github/managing-your-work-on-github/about-project-boards>

²<https://help.github.com/en/github/managing-your-work-on-github/about-issues>

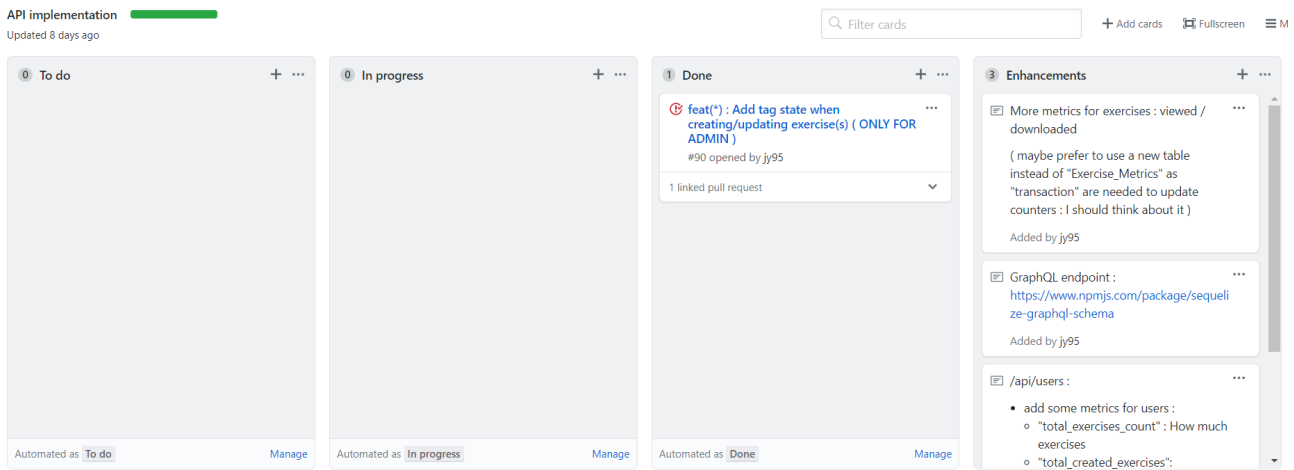


FIGURE 3.1 – Exemple d’interface Github Project Board

☐	fix(police): bug in safari for rendering font bug	
	#8 by dewita was closed on 12 Dec 2019	
☐	feat(FilterPanel): add possibility to choose all the filters in one category enhancement	
	#7 by dewita was closed on 2 Jan	
☐	fix(Exercises): bug when getting back to the main page of exercises after a particular exercise bug	
	#5 by dewita was closed on 12 Dec 2019	
☐	feat(Exercise): add rating system when user is connected enhancement	
	#4 by dewita was closed on 5 Jan	
☐	docs: explanation of the project setup documentation	
	#3 by dewita was closed on 12 Dec 2019	
☐	feat(ExercisePanel): true data on the window enhancement	
	#2 by dewita was closed on 26 Nov 2019	
☐	docs(store): explanations for the search module documentation	
	#1 by dewita was closed on 12 Dec 2019	

FIGURE 3.2 – Exemple d’issues sur Github

3.2 Planning

Comme illustré par la citation en tête de ce chapitre, avoir un planning est plus utile qu’avoir un plan bien ficelé, car de nombreux obstacles imprévus peuvent nuire à celui-ci (dont nous avons cité quelques-unes au point 3.1)

Vous pourrez retrouver ci-dessous une représentation simplifiée du diagramme de Gantt³ finale sous la forme d’une ligne du temps sur laquelle sont placées les grandes étapes du projet ainsi que les deux versions majeures (alpha et beta) que nous avons présentées aux différents acteurs du projet.

³Un diagramme de Gantt permet de représenter en outre la planification des diverses tâches du projet en fonction du temps

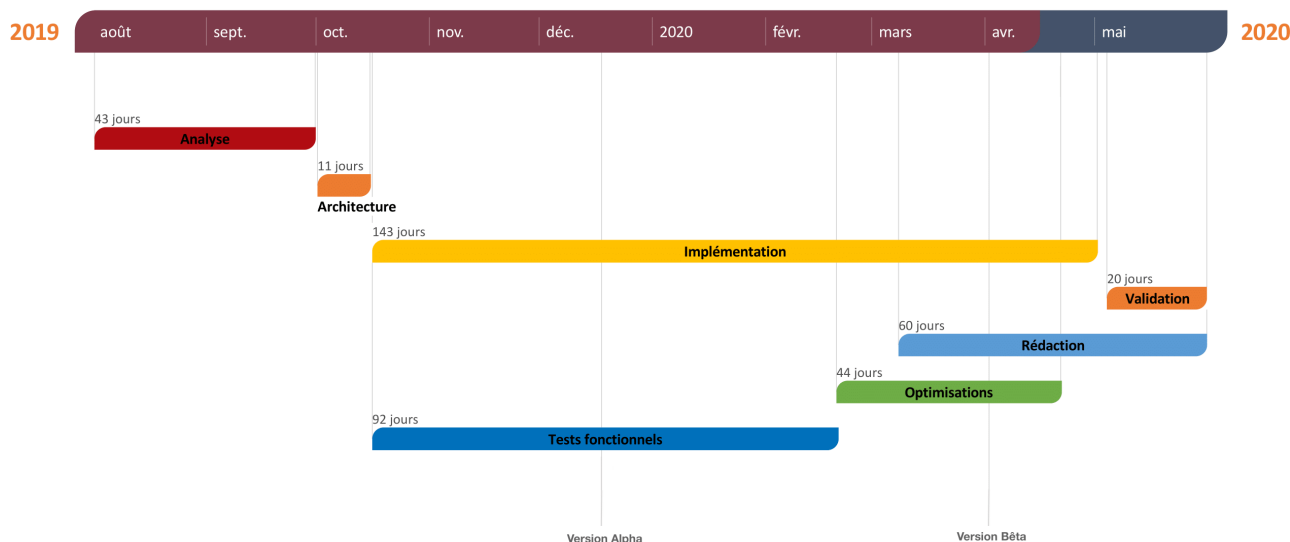


FIGURE 3.3 – Diagramme de Gantt simplifié

Comparé au plan initial, on note des changements plus ou moins conséquents. Il était prévu d'accorder plus de temps aux phases primaires du projet (analyse et conception) pour partir sur de bonnes bases. Cela s'est avéré impossible tant l'usage de données concrètes dans nos ébauches semblait indispensable pour avoir des retours constructifs sur notre solution au plus tôt. De nombreuses fonctionnalités ont été rajoutées au fur et à mesure de l'avancement du projet, ce qui a prolongé la phase d'implémentation jusqu'au mois d'avril (sans oublier son démarrage précoce en octobre).

Fort heureusement, le démarrage très précoce du projet nous a permis de surmonter les nombreux obstacles et imprévus (grâce à notre méthodologie expliquée à la section 3.1) sur notre chemin et de réaliser la majorité des tâches initialement prévues ainsi qu'une part conséquente des suggestions apportées tout au long du projet.

3.3 Organisation du travail

Dès le départ, nous avons tiré à notre avantage les capacités et centres d'intérêt de chaque membre de notre groupe. Par chance, la complémentarité de notre duo a été telle que la répartition des tâches n'a pas posé de soucis, sans oublier une aide mutuelle en cas de besoin. Dans cette section, nous aborderons notre organisation et ses implications lors des différentes étapes de ce travail.

Analyse

Cette phase consistait en ces 3 actions fondamentales :

- Effectuer des recherches bibliographiques concernant l'étiquetage de ressources informatiques (dont vous pouvez consulter la synthèse de celle-ci en annexe B).
- Étudier des solutions similaires, présentes sur le web (que nous avons consigné en table 2.1) pour en dresser les forces et faiblesses de chacune d'entre elles.
- Formaliser les besoins (exprimés en section 2.3) en fonctionnalités.

De par sa nature essentielle, chaque membre de notre équipe a été impliqué dans l'analyse. Si cette étape s'est principalement déroulée d'août à fin septembre, il a été nécessaire de réaliser régulièrement des phases d'analyses complémentaires pour compléter ou réaliser des fonctionnalités.

Conception et implémentation

Contrairement à l'analyse, la conception et l'implémentation ont été réalisées en fonction du domaine spécifique de chacun :

Jacques : responsable du backend et du CLI : base de données, extraction et nettoyage des données, gestion de la performance et la sécurité.

Alexandre : responsable du frontend : prototypage de l'application, interface graphique.

Le principal problème avec cette approche est la "compartimentation" qui favorise une connaissance limitée de la totalité des technologies utilisées. Ceci risque de provoquer un arrêt qui, temporairement ou non, nuit à la bonne réalisation de nos objectifs dans les délais fixés si un membre de notre équipe se retrouve dans l'incapacité sa part de travail pour l'une ou l'autre raison.

Une solution fréquemment utilisée en entreprise pour pallier ce problème est d'impliquer plus d'acteurs sur chaque partie du développement ou la totalité du code (avec un système d'affectation des tâches, rotatif ou non).

Ce choix n'en est pas dénué d'avantages, car en outre de permettre une haute maîtrise des technologies utilisées dans chaque partie, on a un interlocuteur unique à qui poser des questions spécifiques.

Rédaction

L'écriture de ce présent manuscrit a rassemblé l'ensemble de notre équipe. En tant que document qui représente l'aboutissement du travail commun accompli, la présence de chacun est indispensable dans sa réalisation. Dès le départ, nous avons accordé une grande importance à celui-ci lors de la planification du travail (représenté par la figure 3.3), car l'ampleur de cette tâche ne s'improvise pas en dernière minute.

L'élaboration de ce document s'est faite par étapes. Tout d'abord, nous avons réalisé un plan de rédaction simplifiée consistant à représenter les divers sujets et points du contenu à aborder autour d'un fil rouge conducteur. Ensuite, un premier jet à réaliser a été attribué au mémorant des différentes parties sous sa responsabilité. Enfin, nous sommes passés par un laborieux processus répété de relecture et d'améliorations continues du texte, notamment orientés par les nombreux commentaires de nos promoteurs.

« Les choses paraissent simples
jusqu'à ce qu'on commence à les
analyser »

Audrey Niffenegger

La phase d'analyse est dans la vie d'un projet l'une des phases les plus critiques, car celle-ci formalise les besoins des clients sous tous ses aspects afin que toutes les parties (dont les développeurs) puissent se mettre d'accord. Pour pallier les nombreuses difficultés inhérentes au contexte, plusieurs méthodes reconnues existent (notamment QOOQCCP et UML).

La principale fonctionnalité attendue de l'application web est la recherche de fiches sur base d'une combinaison de critères divers et variés (dont notamment des mots-clés) pourrait sembler à première vue simple, mais il n'en est rien. En effet, la question sous-jacente est le problème du référencement de ressources informatiques. Contrairement à d'autres domaines et l'existence de normes pour organiser les informations d'une ressource informatique (*Dublin Core / LOM* pour ne citer que quelques unes), il n'existe pas de consensus sur l'arborescence / la taxonomie des mots-clés (cf. annexe B).

4.1 Analyse fonctionnelle

Les différents types d'utilisateurs

Source Code repose sur la participation active d'acteurs variés. Cela nécessite la mise en place d'une organisation sous forme de rôles afin de maintenir cette plateforme de ressources *cohérente* et *qualitative*. Notre analyse nous a permis d'isoler **4 types d'utilisateurs** disposant de *privileges additionnels* (en plus de leurs privilèges propres, chacun hérite aussi de ceux du précédent de la présente liste) :

- **Visiteur** : Il s'agit d'un utilisateur non inscrit à notre plateforme. Il ne peut rechercher et consulter que les fiches de ressources validées par un administrateur.
- **Utilisateur** : Il s'agit d'un utilisateur inscrit à notre plateforme. Celui-ci peut proposer de nouvelles ressources et maintenir les fiches dont il est le créateur.
- **Administrateur** : Celui-ci crée/modifie des ressources de toute nature (fiches proposées par les utilisateurs, mots-clés et catégories de mots-clés), en plus de classifier les fiches.
- **Super Administrateur** : Celui-ci dispose du droit de supprimer de manière définitive les différentes ressources de notre plateforme (y compris les mots-clés et catégories de mots-clés).

Fonctionnalités

Pour réaliser les défis expliqués à la section 2.3, il convient de faire l'inventaire des nombreuses fonctionnalités à réaliser de manière formelle. Afin de les prioriser en fonction de leurs importances respectives, nous avons utilisé la méthode *MoSCoW* [9]. Celle-ci consiste à classer chaque tâche sous l'une des 4 lettres en gras qui composent cet acronyme en anglais dont voici la signification pour chacune :

- **Must** : C'est ce qui "doit être fait" (**vital**)
- **Should** : C'est ce qui "devrait être fait dans la mesure du possible" (**essentiel**)
- **Could** : C'est ce qui "pourrait être fait dans la mesure où cela n'a pas d'impact sur les autres tâches" (**confort**)
- **Won't** : C'est ce qui "ne sera pas fait cette fois, mais sera fait plus tard" (**luxé**)

À titre informatif, le choix du **JSON** se justifie par la norme que nous avons élaborée (cf. annexe B), qui l'utilise afin de pouvoir manipuler plus aisément les données contenues dans celui-ci. Des explications complémentaires sur d'autres aspects de cet inventaire seront apportées dans les prochaines sections de ce chapitre.

Tout visiteur peut :

- M** Rechercher des fiches dans l'application. La recherche se porte sur une combinaison libre des critères suivants :
 - M** Rechercher dans les mots-clés des fiches
 - M** Rechercher dans le titre des fiches
 - M** Rechercher les fiches par leurs états (cf. figure 4.1)
 - M** Rechercher les fiches sur base d'un certain seuil sur le résultat moyen accordé par les utilisateurs
 - M** Rechercher les fiches par leurs créateurs
 - M** Rechercher les fiches par leurs identifiants
- M** Consulter une fiche
- M** Consulter les mots-clés et catégories de mots-clés existants (optionnellement avec des statistiques d'utilisation de ceux-ci)
- M** S'inscrire à l'application
- M** Se connecter à l'application via une adresse mail et un mot de passe
- S** Ordonner les fiches lors de la recherche avec une libre combinaison des paramètres suivants (en précisant pour chacun l'ordre du tri : ascendant / descendant) :
 - le résultat moyen des votants pour une fiche
 - le nombre de votants pour une fiche
 - la date de la dernière modification de la fiche
 - le titre de la fiche
 - l'état de la fiche (cf. figure 4.1)
 - l'identifiant de la fiche
- C** Choisir quelles propriétés des fiches que l'on souhaite consulter
- C** Télécharger la source d'une fiche

Tout utilisateur peut :

- M** Mettre en ligne une fiche (avec éventuellement un fichier)
- M** Modifier les informations d'une fiche lui appartenant
- M** Changer l'état d'une fiche lui appartenant (cf. figure 4.1 : pour éviter des dérives, seuls les états **DRAFT**, **PENDING** et **ARCHIVED** sont possibles pour ce rôle spécifique)
- M** Proposer un nouveau ou plusieurs mot(s) clé(s)
- S** Gérer des favoris
- S** Utiliser des favoris dans la recherche de fiches
- S** Évaluer une fiche
- S** Consulter / Modifier ses informations personnelles

Tout administrateur peut :

- M** Importer plusieurs fiches sur base d'un fichier JSON. Cette fonctionnalité comprend les fonctionnalités sous-jacentes suivantes :
 - Ajouter éventuellement un état spécifique pour chacune des fiches à importer (cf. figure 4.1)
 - Créer automatiquement les mots-clés non existants dans le système, avec éventuellement un état de départ (cf. figure 4.2)
 - Créer automatiquement les catégories de mots-clés non existantes dans le système
 - Associer les mots-clés à leurs fiches respectives.
- M** Exporter des fiches au format JSON
- M** Proposer un nouveau ou plusieurs mot(s) clé(s), avec éventuellement un état spécifique pour chacun (cf. figure 4.2)
- M** Modifier un mot-clé
- M** Modifier une catégorie de mots-clés
- M** Changer l'état d'un ou plusieurs mot(s) clé(s) (cf. figure 4.2)
- M** Modifier les informations d'une fiche
- M** Changer l'état d'une fiche (cf. figure 4.1 : aucune restriction)
- M** Créer ou trouver des catégories de mots-clés
- S** Lister tous les utilisateurs de l'application (sans distinction)

Tout super administrateur peut :

- M** Supprimer des fiches / mots-clés / catégories de mots-clés
- S** Modifier le type d'un utilisateur
- W** Créer de nouveaux utilisateurs

Recherche par mots-clés

Une interrogation légitime est de savoir comment exprimer la recherche par mots-clés, ce qui n'est bien entendu pas chose aisée. Pour l'illustrer, prenons l'exemple de quelques fiches et mots-clés, que nous noterons resp. $F\mathbf{X}$ et $M\mathbf{X}$ ($\mathbf{X}$ étant remplacé par un numéro pour les distinguer), avec quelques exemples de requêtes.

	M1	M2	M3
F1	X	X	
F2		X	X
F3	X		

TABLE 4.1 – Problématique de la recherche par mots-clés

- Quelles sont les fiches qui disposent des mots-clés M1 ou M2 ?
- Quelles sont les fiches qui disposent du mot-clé M2, mais pas le M1 et le M2 réunis ?
- Quelles sont les fiches qui ne disposent pas du mot-clé M3 ?

Comme vous pouvez le constater, ces requêtes, souvent particulièrement verbeuses, peuvent viser des réalités différentes et donc provoquer des situations confuses aussi bien pour un être humain que pour une machine (par exemple, si on combine des "et" et "ou" dans une même phrase, sans ponctuation adaptée). Ceci pose de facto la question de la manière de représenter de telles requêtes.

Afin d'exprimer le plus clairement possible cette grande diversité des requêtes, nous avons fait le choix de la **FNC**¹. De ce fait, nous pouvons ainsi exprimer respectivement par une forme non ambiguë les exemples de requêtes précédentes ² :

- $M1 \vee M2$
- $M2 \wedge \neg M1$
- $\neg M3$

Dès lors, il est plus aisé d'identifier quelle(s) fiche(s) corresponde(nt) à nos critères de recherche (à des fins de clarté, nous utiliserons le symbole ✓ ici, mais également dans la table 5.4).

recherche / fiche	F1	F2	F3
$M1 \vee M2$	✓	✓	
$M2 \wedge \neg M1$		✓	
$\neg M3$	✓		✓

TABLE 4.2 – Solution théorique de la recherche par mots-clés

Nous reviendrons d'ailleurs ultérieurement (cf. section 5.3.2.4, figure 5.11...) sur les façons dont nous avons mis en place cette réflexion dans l'implémentation de **Source Code**. Vous pourrez noter, qu'en dépit de conventions d'écriture différentes (car inhérentes au contexte), les tables 4.2 et 5.4 expriment les mêmes principes.

¹Forme normale conjonctive - https://fr.wikipedia.org/wiki/Forme_normale_conjonctive

²Par souci de simplicité, nous n'allons pas introduire de nouvelles notations pour les littéraux : $M\mathbf{X}$ devant ici est compris comme "la fiche dispose du mot-clé n°X"

État d'une fiche

Au cours de notre analyse des processus de partage/modération de ressources informatiques, nous avons été confrontés à de nombreuses lacunes dans des systèmes similaires dont notamment :

- Une manière très simplifiée de considérer les fiches : il n'y a pas de nuance claire pour différencier la situation d'une fiche par rapport à une autre. Pour l'illustrer, prenons l'exemple d'une fiche qui est en cours de rédaction (et donc n'est pas encore prête à être publique) : bien souvent, cette information est absente.
- Le manque d'évolutivité technique dans la gestion des fiches, que nous pouvons sans doute imputer à une analyse très restrictive. Conséquence du point précédent, cette lacune rend difficile l'ajout de nouvelles fonctionnalités telles que l'archivage numérique de ces ressources.

C'est ainsi que nous avons décidé d'associer un état à chaque fiche, permettant ainsi de distinguer la situation de chacune au cours de ses processus (cf. annexe B - table 4.1 : élément "state"). Le diagramme UML à états ci-dessous représente ces états et les principales transitions entre états (pour ne pas surcharger celui-ci) :

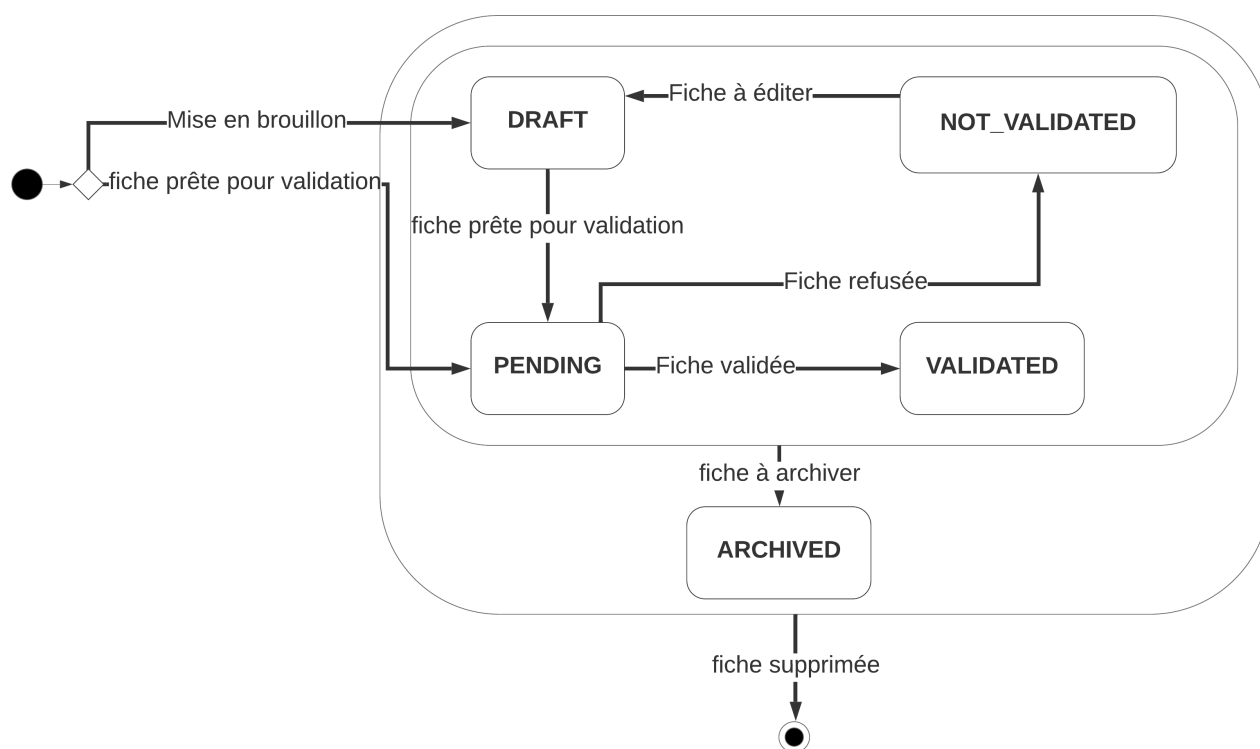


FIGURE 4.1 – Diagramme UML à états pour l'état d'une fiche

État d'un mot-clé

Une des questions annexes au sujet des fiches est la gestion des mots-clés. En effet, les mots-clés étant des éléments indispensables d'une fiche de qualité pour un meilleur référencement, il convient d'établir une stratégie précise pour exploiter au mieux ceux-ci.

Durant notre analyse, nous avons pu constater deux écoles de pensée bien distinctes (comparable à ce qui existe en économie) :

- laissez-faire : Il s'agit de donner une liberté totale en matière de marquage (en utilisant aussi bien des mots-clés existants que non). Bien que cette approche a le mérite de faire émerger de nouveaux mots-clés par les contributions d'utilisateurs, cela restreint les possibilités de modération.
- l'interventionnisme : Il s'agit de restreindre le choix en matière de marquage (exclusivement des mots-clés existants). Bien que cette approche rend la modération facile, cela restreint les possibilités de s'adapter à une réalité changeante.

Nous avons remarqué qu'aucune de ces deux possibilités ne se distinguait suffisamment de l'autre pour répondre de manière optimale à la problématique. C'est pourquoi nous avons fait le choix d'une 3e voie, qui se situe donc entre ces 2 manières de penser. Tout comme les fiches, il s'agit d'associer un état à chaque mot-clé pour distinguer sa situation propre (cf. annexe B - table 4.2 : élément "state"). Le diagramme UML à états ci-dessous représente ces états et les principales transitions entre états (pour ne pas surcharger celui-ci) :

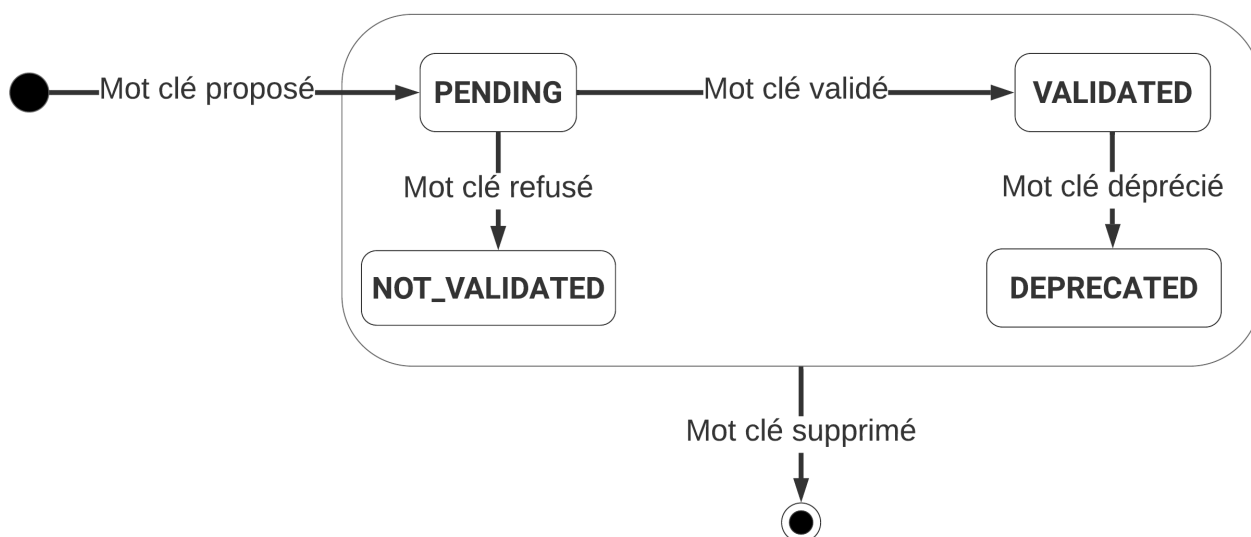


FIGURE 4.2 – Diagramme UML à états pour l'état d'un mot-clé

Maintenabilité aisée des données

Lors de notre analyse, un besoin particulièrement criant s'est présenté à nous : sans être informaticien ou connaître les technologies réalisant le stockage de données, il faut disposer d'un moyen de lire/modifier ces données avec aisance.

Cela implique dès lors de proposer des opérations pour manipuler les différentes ressources de notre application (fiches, mots-clés, catégories de mots-clés, etc...). Pour compléter la panoplie, nous avons conçu un moyen d'importer / exporter les fiches, notamment pour répondre aux besoins d'archivage.

4.2 Analyse non fonctionnelle

Sécurité

La sécurité d'une application doit faire l'objet d'une attention non négligeable, particulièrement dans le contexte de notre époque mouvementée dans bien des domaines. Il est donc essentiel de se poser les bonnes questions (par exemple en utilisant QQQQCCP). Pour relever ce défi, une méthode possible est la **AAA**³ consistant en 3 piliers :

Authentification : Il s'agit du fait de prouver l'utilisateur que l'on prétend être. Une manière fréquemment utilisée sur de nombreux sites consiste en l'association d'un nom d'utilisateur et d'un mot de passe.

Autorisation : Il s'agit du fait de vérifier quelles ressources accessibles et les opérations (par exemple les CRUD) permises sur celles-ci pour l'utilisateur authentifié.

Traçabilité : Il s'agit du fait d'enregistrer tous les faits et gestes des utilisateurs authentifiés. Les informations ainsi collectées permettent principalement de prévenir/comprendre des problèmes.

Maintenance et évolution

Ce travail s'est déroulé dans le cadre d'un mémoire universitaire. Étant donné que la problématique du mémoire ne risque pas de faiblir d'intensité dans les années à venir, une reprise/évolution du projet par une équipe ultérieure inconnue est une possibilité à ne pas exclure. Il convient donc de mettre en place les dispositions nécessaires pour faciliter la maintenance de notre application.

Pour cela, nous avons sélectionné des technologies relativement bien documentées et populaires de telle sorte que nos successeurs n'éprouvent pas de difficulté à se former à celles-ci. Vous pourrez retrouver des explications plus détaillées à ce sujet dans la section 5.1.

Ergonomie

Une application web à destination d'un public varié et conséquent se doit d'avoir une interface simple et efficace pour ne pas perdre ses utilisateurs et gagner en popularité. On pourrait considérer que le design d'une application est subjectif, mais un point à ne sûrement pas négliger se situe autour de l'ergonomie. La question à se poser pour chaque interface est alors de savoir comment disposer les éléments afin de rendre la navigation la plus claire possible. Nous nous sommes donc concentrés sur cette problématique en élaborant un patchwork que vous pouvez consulter à la fin de la section 2.2.

Au fur et à mesure de nos meetings avec le professeur *Kim Mens* et *Olivier Goletti*, les conseils et recommandations ont souvent été pris en compte pour améliorer l'expérience utilisateur. Par la même occasion, nous avons fait tester l'application à différents utilisateurs (amis, designer) tout au long de la phase de développement afin de nous faire part de leur ressenti.

³Acronyme anglophone pour "**A**uthentication/**A**uthorization/**A**ccounting", que l'on peut traduire en français par "authentification/autorisation/traçabilité"

4.3 Contraintes

Une première contrainte évidente est indubitablement le temps. Nous avons consacré une année académique entière sur ce mémoire, mais ce n'est certainement pas assez pour rendre mature `Source Code`.

Pour la partie frontend, nous avons décidé de nous consacrer uniquement sur le design et l'ergonomie d'un desktop ayant une largeur minimale de 1550 pixels. Pour l'instant, nous avons recommandé aux utilisateurs de l'application ayant une petite résolution de modifier l'échelle du navigateur internet afin d'avoir une expérience optimale malgré notre contrainte. Créer les autres déclinaisons pour petits desktop, tablettes et smartphones ne nous auraient pas permis d'intégrer toutes les fonctionnalités qu'on avait prévu d'ajouter sur la plateforme, car cela aurait pris trop de temps (modifier la disposition des éléments, créer de nouveaux éléments, changer le layout ...).

Pour la partie backend, il nous a été imposé de travailler avec une base de données relationnelle⁴ dont PostgreSQL⁵ a été le système de gestion imposé. Ce type de base de données dispose de propriétés très intéressantes dont notamment les **ACID**⁶, que vous retrouvez sur la figure 4.3 :



FIGURE 4.3 – Propriétés **ACID** [4]

⁴https://fr.wikipedia.org/wiki/Base_de_données_relationnelle

⁵<https://www.postgresql.org/>

⁶https://fr.wikipedia.org/wiki/Propriétés_ACID

Étant donné que ces propriétés sont assez contraignantes, le lecteur attentif (ou lectrice attentive) que vous êtes est en droit de remettre en question cette contrainte technique et il est donc utile de recourir au théorème **CAP** pour mieux la justifier. Comme expliqué par un article [2], chaque lettre de cette anagramme représente une certaine propriété :

Consistency (Cohérence) : "Une donnée n'a qu'un seul état visible, quel que soit le nombre de répliquas"

Availability (Disponibilité) : "Tant que le système tourne (distribué ou non), la donnée doit être disponible"

Partition Tolerance (Distribution) : "Quel que soit le nombre de serveurs, toute requête doit fournir un résultat correct"

Comme souligné dans ce même article [2], le théorème dit que "dans toute base de données, vous ne pouvez respecter au plus que 2 propriétés". Grâce à ce théorème, nous pouvons mieux comparer PostgreSQL par rapport à des solutions alternatives, comme le montre la figure 4.4 :

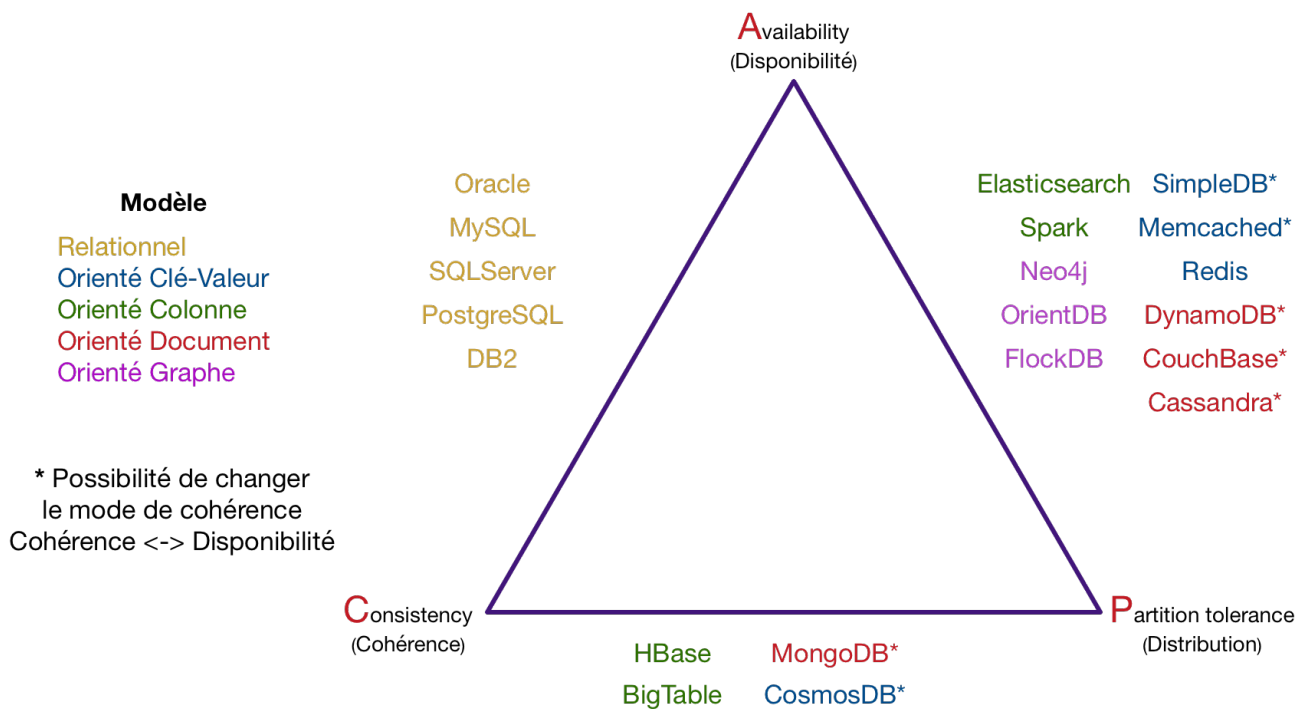


FIGURE 4.4 – Triangle **CAP** avec quelques bases de données [2]

En effet, pour remplir les objectifs expliqués en section 2.3, il convient de sélectionner les propriétés de cohérence et de disponibilité. Comme vous pouvez le constater, PostgreSQL est l'un des candidats possibles qui correspondent le mieux à nos besoins. À titre informatif, il va de soi qu'une évolution des besoins à l'avenir peut se produire et entraîner une remise en question.

5.1 Choix technologiques

Choix de l'environnement de programmation

Parmi la myriade de langages de programmation (avec leurs différentes plateformes logicielles existantes) possibles, nous avons opté pour le langage JavaScript et la plateforme Node.js pour toutes les parties de notre solution (backend, frontend et CLI).

Ce langage ne vous est peut-être pas tout à fait inconnu en effet, il est souvent utilisé conjointement avec les interfaces HTML du côté client. Le JavaScript est souvent mal considéré par les utilisateurs, lesquels mettent en cause ses difficultés.

Néanmoins, ce langage riche n'en reste pas pour la cause dépourvu d'avantages et pour illustrer ce panel, nous n'en citerons que quelques points : JavaScript est un langage basé sur les événements ; il permet ainsi de mettre à jour dynamiquement l'interface qu'il assiste, et c'est notamment le cas pour la quasi-totalité des fameuses messageries instantanées utilisées partout.

Node.js nous permet désormais, à l'instar du PHP, d'écrire du code JavaScript du côté serveur qui servira à répondre aux requêtes du client, tout en bénéficiant des avantages du JavaScript.

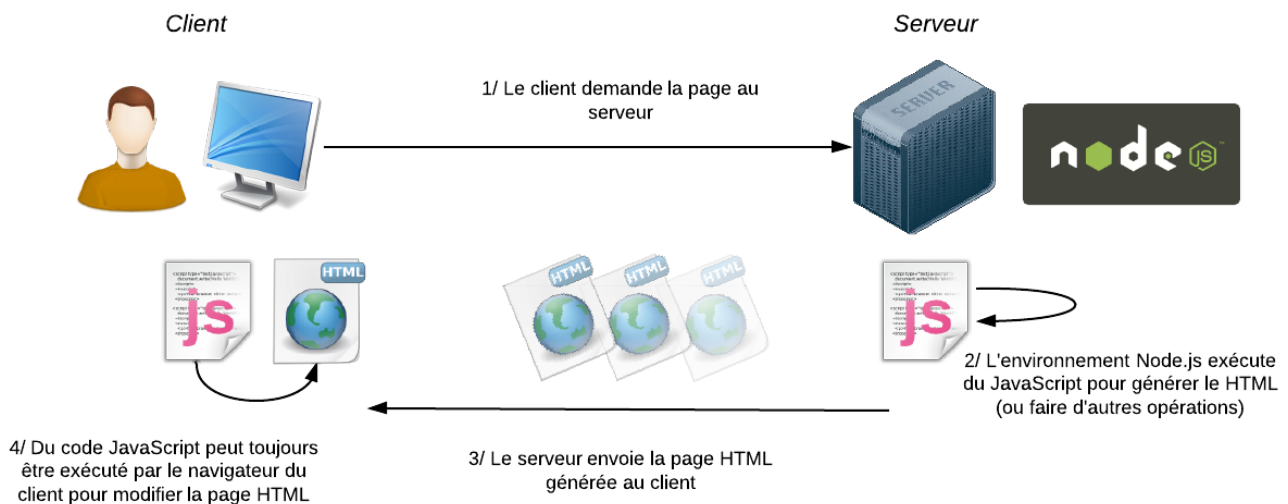


FIGURE 5.1 – Du JavaScript aussi bien côté serveur que client [5]

Deux points sont élémentaires pour expliquer l'intérêt et la rapidité de Node.js : le moteur JavaScript V8 et son fonctionnement non bloquant.

Ce moteur, développé par Google, très performant et optimisé propose la compilation à la volée (en anglais, JIT , just-in-time compilation) , approche hybride entre la compilation et l'interprétation, dont nous pouvons schématiser leur mode de fonctionnement individuel par la figure suivante :

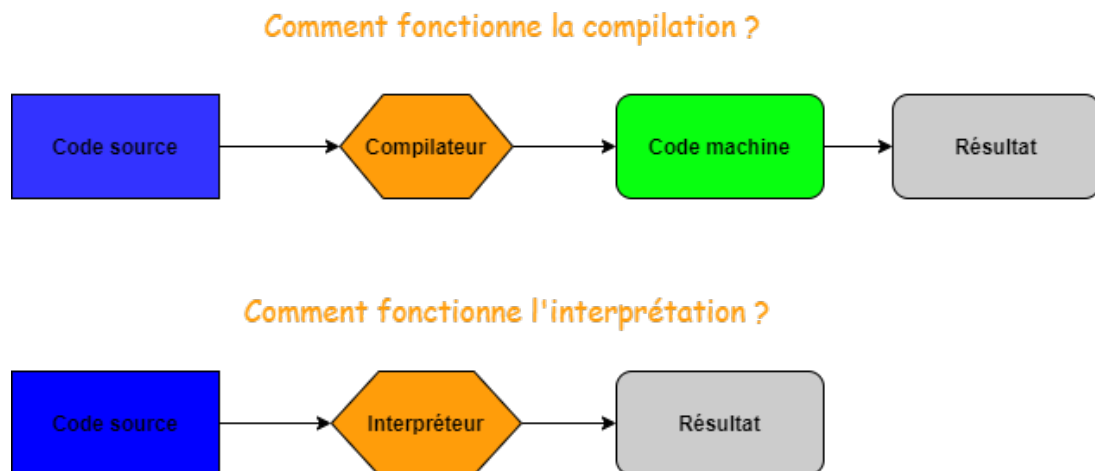


FIGURE 5.2 – Compilation et interprétation - mode de fonctionnement

Malheureusement, la compilation et l'interprétation ont chacune leurs qualités et défauts, que l'on peut synthétiser en 4 grands points :

- Une fois compilé, on ne peut plus changer le programme (contrairement à l'interprétation)
- En cas d'erreur(s) dans le programme, la compilation les affiche dans la phase "compilation" contrairement à l'interprétation qui les affiche à l'exécution.
- La compilation permet d'optimiser les performances d'un programme (car disposant de l'intégralité du code source au départ), mais nécessite un certain temps pour traduire le programme en code machine.
- L'interprétation possède moins d'étapes que la compilation.

Dans tout système, il existe des tâches coûteuses en temps : des appels à la base de données, etc. Dans un mode de fonctionnement bloquant, il convient d'attendre la fin de ces tâches avant de réaliser autre chose. Ceci est un gâchis dans la mesure où des tâches moins gourmandes en temps sont ainsi bloquées. Node.js permet, par son mode de fonctionnement non bloquant, des gains de performances, comme illustré ci-dessous.

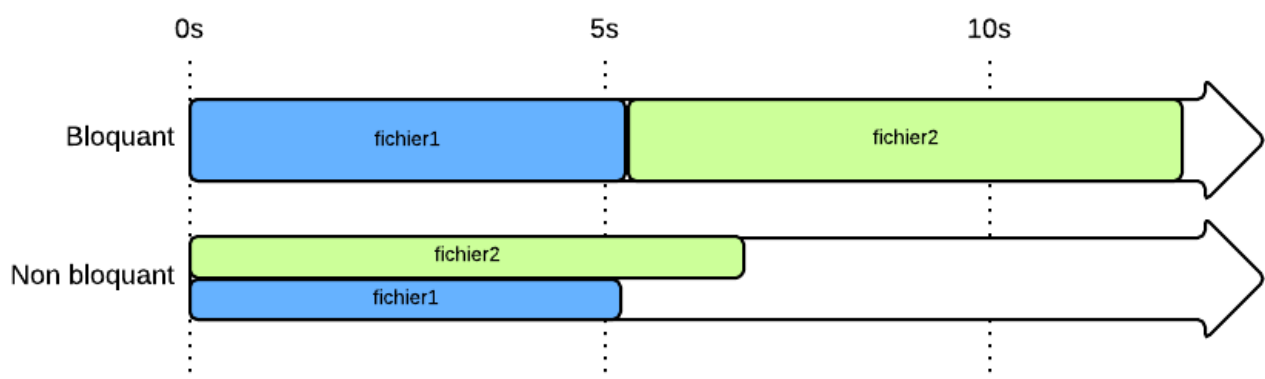


FIGURE 5.3 – Bloquant/non bloquant : un exemple pour l'illustrer [5]

Choix des frameworks

De manière générale, un framework peut être considéré comme une boîte à outils très pratique nous permettant de développer une application conséquente en disposant de fonctionnalités et d'une structure de base. Nous pouvons citer 2 avantages majeurs à son usage :

- Une architecture spécialement prévue (et souvent éprouvée) pour résoudre des classes de problèmes précises permettant ainsi une maintenabilité / évolutivité de l'application
- Une standardisation de la programmation permettant ainsi d'interchanger, d'injecter et/ou réutiliser du code existant pour ne pas "réinventer la roue".

Dès lors, ce choix, qui possède aussi son lot d'inconvénients, ne saurait être pris à la légère, car il constitue le squelette de l'application. C'est pourquoi vous pourrez retrouver ci-dessous une liste non exhaustive des frameworks (dans les tables 5.1, 5.2 et 5.3) que nous avons étudiés ainsi que les critères retenus pour les départager.

Critères - [M]auvais, [S]ufficient, [B]on

Doc Documentation :

La présence d'une documentation suffisamment claire et explicite pour répondre aux principales questions sur son utilisation. La qualifier n'est pas une tâche aisée, puisque souvent soumise exclusivement à notre subjectivité personnelle : en effet, des mesures quantitatives telles que le ratio entre le nombre de lignes de commentaires (**CLOC**) et le nombre total de lignes de code (**LOC**) n'aident pas, car quantité n'est pas synonyme de qualité ...

Fcts Fonctionnalités :

Le nombre de fonctions utilitaires ainsi que de leur degré réciproque d'utilité / de praticité perçue par les utilisateurs. Un choix binaire ne peut dès lors pas s'appliquer : une solution minimaliste en nombre de fonctionnalités pourrait davantage répondre à nos besoins.

Maint Maintenabilité d'une application créée avec ce framework :

Il s'agit du degré de facilité avec laquelle la maintenance du code est accomplie. Plusieurs paramètres dont la structure et la complexité du code impactent ce critère. Il s'agit dès lors de ne pas négliger ce critère, car rien n'exclut l'ajout de nouvelles fonctionnalités ou la découverte de bugs à l'avenir.

Pop Popularité :

Le fait d'être connu et d'être utilisé par un grand nombre d'utilisateurs. Puisque notre solution est entièrement basée sur Node.js, nous pouvons consulter des données publiques du registre par défaut (NPM) notamment par des sites comme NPM Trends.

Perfs Performances :

Elles se mesurent en fonction du temps de réponse à une requête client. Il convient cependant d'être prudent avec cette explication simpliste : certaines requêtes peuvent nécessiter plus ou moins de ressources. De ce fait, les performances d'un framework sont principalement influencées par les technologies utilisées.

Pour calculer le résultat total pour un framework précis, il s'agit de la somme des résultats obtenus pour chacun des critères avec notre système de notation : **M**, **S** et **B** (resp. 0, 0.5 et 1).

Framework pour le front-end

Framework	Doc	Fcts	Maint	Pop	Perfs	Total
Vue.js ¹	B	B	B	S	B	4.5
Angular ²	B	B	B	S	B	4.5
React ³	B	B	S	B	B	4.5

¹<https://vuejs.org> ²<https://angular.io>

³<https://fr.reactjs.org>

TABLE 5.1 – Comparatif de quelques frameworks pour le frontend

Le choix entre ces trois frameworks s'apparente à une décision plus que subjective, car ce sont tous les trois d'excellents outils de développement.

Puisqu'il fallait bien trancher, *Vue.js* a été la solution retenue pour la partie front-end car nous l'avions déjà utilisé par le passé. Ce framework est aussi populaire qu'*Angular*, mais moins que *React*. Vue détient cependant le plus grand nombre de "*stars*" sur GitHub, ceci traduisant un intérêt non négligeable pour ce framework au sein de la communauté des développeurs.

Au-dessus de *Vue.js*, nous avons utilisé le framework *Nuxt.js*^a qui permet d'encapsuler la logique des routes, des stores, des plugins et des middlewares de telle manière à ce que le code soit plus facilement maintenable. En outre, il ajoute aussi le mode *SSR* (Server Side Rendering), qui permet à l'application d'être performante, tout en profitant d'un boost pour le référencement naturel (SEO).

Framework pour l'API

Framework	Doc	Fcts	Maint	Pop	Perfs	Total
Express ¹	B	S	S	B	B	4
LoopBack ²	B	B	S	M	S	3
Feathers ³	B	B	S	M	S	3

¹<https://expressjs.com/> ²<https://loopback.io/>

³<https://feathersjs.com/>

TABLE 5.2 – Comparatif de quelques frameworks pour l'API

Express a été la solution que nous avons retenue, car celui-ci offre, bien qu'ayant un nombre de fonctionnalités restreint par rapport à ses concurrents, des possibilités de configuration adaptées dans le contexte de ce projet. Il est en effet plus aisé d'y incorporer du code écrit par autrui tandis que les autres étudiés se concentrent principalement sur eux-mêmes.

^a<https://fr.nuxtjs.org/>

Framework	Doc	Fcts	Maint	Pop	Perfs	Total
Yargs ¹	B	B	B	S	B	4.5
Commander.js ²	S	S	M	S	S	2
Inquirer.js ³	S	B	S	S	S	3

¹ <http://yargs.js.org/>

² <https://github.com/tj/commander.js>

³ <https://github.com/SBoudrias/Inquirer.js>

TABLE 5.3 – Comparatif de quelques frameworks pour le CLI

Yargs a été la solution que nous avons retenue, car celui-ci surclasse de manière incontestable les autres solutions existantes. En effet, la documentation (très claire et complète) et le nombre de fonctionnalités permettent une rapide adoption pour construire une solution dans les plus brefs délais.

Choix des librairies externes

De nombreuses applications, y compris la nôtre, partagent des besoins communs : de ce fait, il est donc adéquat de réutiliser le travail d'autrui plutôt que de "réinventer la roue".

Compte tenu du nombre relativement élevé de librairies que nous avons utilisées dans les diverses parties de notre solution, nous évoquerons ici uniquement que les plus conséquentes.

5.1.0.0.1 Librairies communes

Lodash

Cette librairie propose un grand nombre de fonctionnalités dont nous allons vous montrer un infime échantillon au moyen de cet exemple :

```
1 let _ = require('lodash');
2 let array = [1, 1,
3   {"category_id": 42, "text": "foo"},
4   {"category_id": 42, "text": "FOO"},
5 ]
6 // pour distinguer les nombres des objets
7 const [nombres, autres] = _.partition(array, Number.isInteger);
8 // pour obtenir une liste d'éléments distinct
9 const nombres_distints = _.uniqWith(nombres, _.isEqual)
10 // pour avoir une copie identique d'éléments contenus dans un tableau
11 const autres_bis = _.clonedeep(autres);
12 // etc...
```

Code 5.1 – Exemple des fonctionnalités de Lodash

5.1.0.0.2 Bibliothèques pour le frontend

VeeValidate

Un excellent framework, construit pour VueJS, responsable de la validation des différents formulaires présents sur **Source Code**. Très facilement, nous pouvons importer des règles de validation (*min* pour minimum de caractères, *max* pour maximum de caractère, *required* pour champs obligatoire ...) depuis VeeValidate et les ajouter au composant responsable de la validation du champ.

Voici un petit exemple d'utilisation tiré du fichier `login.vue`, responsable d'afficher la page de connexion à un compte :

```
<ValidationObserver ref="observer" tag="form" v-slot="{ valid }" @submit.prevent="validateBeforeSubmit()">
  <ValidationProvider tag="label"
    name="email"
    rules="required|email"
    v-slot="{ errors }">
    Adresse email
    <input id="Email" name="email" v-model="form.email" class="input--grey" type="email">
    <span class="error-message">{{errors[0]}}</span>
  </ValidationProvider>

  <ValidationProvider tag="label"
    name="mot de passe"
    rules="required"
    v-slot="{ errors }">
    Mot de passe
    <input id="Password" name="password" v-model="form.password" class="input--grey" type="password">
    <span class="error-message">{{errors[0]}}</span>
  </ValidationProvider>

  <div class="button-wrapper">
    <button :class="{ 'button--ternary-color': !valid, 'button--ternary-color-reverse': valid}" type="submit">
      Se connecter
    </button>
  </div>
</ValidationObserver>
```

FIGURE 5.4 – Exemple sur la page login

ValidationObserver et ValidationProvider sont deux composants de VeeValidate permettant de gérer un formulaire. Le premier fait office de conteneur, auquel on attache un ou plusieurs ValidationProvider responsables d'un champ en particulier (texte, checkbox ...). Pour chaque ValidationProvider, il suffit d'ajouter une règle de validation comme c'est le cas pour le champ email : `required|email` (champ obligatoire et doit être un email).

Lorsque l'utilisateur cliquera sur le bouton de connexion, une fonction de validation va être lancée pour dans un premier temps valider le formulaire. Si toutes les règles sont respectées, alors les données sont envoyées à l'API, sinon, un message d'erreur avec l'explication de la règle non respectée s'affichera juste en dessous du champ incorrect.

Tiptap

Tiptap est un éditeur de texte riche "renderless" (ce terme signifie que la conception du design nous appartient). Cette librairie prévue pour fonctionner avec VueJS permet de mettre en place assez rapidement un éditeur de texte complet et flexible. On peut aisément y rajouter des options comme des triggers que l'on peut attacher à des boutons pour styliser le texte plus facilement. Pour exemple, voici à quoi ressemble l'éditeur de texte de la description d'une ressource informatique :

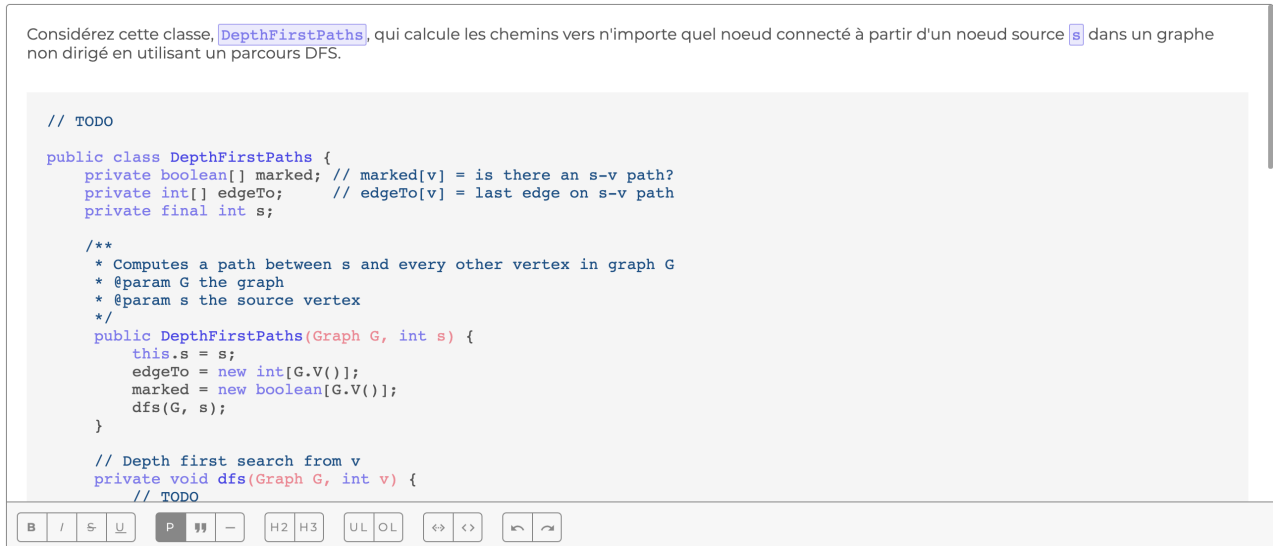


FIGURE 5.5 – Exemple d'un éditeur de texte

Les boutons en dessous du champ texte permettent de formater le texte (gras, italique, souligné, barré). En cliquant sur un d'eux, le trigger attaché au bouton va lancer une des méthodes appartenant à Tiptap pour modifier le texte et rendre les modifications automatiquement visibles pour l'utilisateur dans l'éditeur de texte.

Highlight.js

Librairie permettant de détecter automatiquement le langage de programmation dans un texte et de lui appliquer la coloration syntaxique adéquate. Highlight.js est utilisé conjointement avec la librairie Tiptap pour écrire du code dans l'éditeur et profiter de la coloration syntaxique (cf. image précédente).

Axios

Librairie facilitant de manière drastique la conception de requêtes HTTP. Nous utilisons donc Axios pour communiquer avec notre API. Voici un exemple d'utilisation :

```
1 try {
2     const response: SearchExerciseResponse = await this.app.$axios.$post('/api
   ↪ /search', newSearchRequest);
3     commit('INIT', response);
4 } catch (e) {
5     const errorAxios = e as AxiosError;
```

```

6      const status : number = (errorAxios.response) ? errorAxios.response.status
      ↪ : 400;
7      const errors : any = {
8          400: "Une erreur est survenue lors du chargement des exercices.",
9          401: 'Vous devez vous connecter pour charger ces exercices.',
10         403: 'Vous n'êtes pas autorisé à effectuer cette action.',
11         500: 'Une erreur est survenue depuis nos serveurs, veuillez-nous en
      ↪ excuser.',
12     }
13
14     // fetch the message for this error
15     const message : string = (errors.hasOwnProperty(status)) ? errors[status]
      ↪ : errors[400];
16     this.$displayError(message, {statusCode: status});
17
18     commit('RESET')
19 }

```

Code 5.2 – Axios : Exemple d'utilisation d'une POST request

Axios contient plusieurs méthodes dont `$post` qui est utilisée dans l'exemple. Il suffit de donner en premier argument le endpoint (`/api/search`) et le payload que l'on veut envoyer en deuxième argument.

Chacune de ces méthodes retourne une Promesse contenant la réponse ou une erreur si la requête ne s'est pas bien exécutée. Une bonne pratique consiste donc à entourer la requête avec un bloc `try/catch` pour gérer les deux cas.

5.1.0.0.3 Bibliothèques pour le backend

Sequelize

Afin d'interagir avec la base de données en Postgresql (cf. section 4.3), nous avons fait le choix d'utiliser la bibliothèque Sequelize qui propose du mapping objet-relationnel² (à titre informatif, d'autres alternatives similaires existent, notamment dans Loopback que nous avons étudié en table 5.2).

Cette approche, qui s'inscrit dans la lignée de la **POO**³, consiste à interagir avec la base de données à travers un logiciel qui s'occupe de faire le lien entre les données stockées et les objets⁴ qui les représentent, nous épargnant ainsi l'usage du **SQL** (ou un autre langage complexe) et les conversions manuelles de données de chaque côté. Nous vous proposons d'ailleurs une illustration récapitulative en figure 5.6.

²En anglais, le terme utilisé est **Object-Relational Mapping** ou **ORM**.

³Programmation orientée objet - https://fr.wikipedia.org/wiki/Programmation_orientée_objet

⁴En **POO**, le terme "objet" désigne un type abstrait de données, pouvant représenter des choses du monde réel ou non, pouvant contenir un état (c.-à-d. leurs propriétés / caractéristiques) et/ou des comportements (c.-à-d. du code).

Relational database (such as PostgreSQL or MySQL)

ID	FIRST_NAME	LAST_NAME	PHONE
1	John	Connor	+16105551234
2	Matt	Makai	+12025555689
3	Sarah	Smith	+19735554512
...	...	...	...

objects

```
class Person:
    first_name = "John"
    last_name = "Connor"
    phone_number = "+16105551234"

class Person:
    first_name = "Matt"
    last_name = "Makai"
    phone_number = "+12025555689"

class Person:
    first_name = "Sarah"
    last_name = "Smith"
    phone_number = "+19735554512"
```

ORMs provide a bridge between
relational database tables, relationships
and fields and objects

FIGURE 5.6 – Finalité d'un ORM

De ce fait, cela permet de simplifier les interactions⁵ entre le backend et la base de données, afin de pouvoir changer vers un autre système de gestion de base de données théoriquement sans modifier la logique interne (quelques restrictions / spécificités / différences techniques peuvent néanmoins se présenter entre le système d'origine et de destination ...).

OpenAPI-Enforcer

Il s'agit d'une librairie développée et massivement utilisée par la **BYU**⁶. Elle permet, sur base de document(s) respectant la OAS, d'enrichir notre API de nombreuses fonctionnalités très intéressantes dont voici quelques-unes :

- Analyse, désérialisation et validation des paramètres de requêtes **HTTP**.
- Mécanisme de réponse automatique en cas d'erreur
- Association automatique des chemins au code

Nous vous invitons à découvrir la figure 5.7 qui propose une explication condensée de la librairie. Nous reviendrons plus en détail sur son intégration dans la section 5.3 : cette sous-section ne constituant qu'une brève introduction. À titre informatif, cette librairie se décline en 3 sous-librairies :

- `openapi-enforcer`⁷ : le cœur de la librairie
- `openapi-enforcer-middleware`⁸ : le middleware de la librairie
- `openapi-enforcer-multer`⁹ : un middleware optionnel qui s'occupe automatique de l'upload de fichiers (cf. section 4.1)

⁵Par exemple, de nombreux **ORMs**, y compris Sequelize, intègrent la notion de "scopes queries" (<https://sequelize.org/master/manual/scopes.html>), permettant de réutiliser du code pour des requêtes fréquemment utilisées.

⁶Brigham Young University - <https://www.byu.edu/about>

⁷<https://github.com/byu-oit/openapi-enforcer>

⁸<https://github.com/byu-oit/openapi-enforcer-middleware>

⁹<https://github.com/byu-oit/openapi-enforcer-multer>

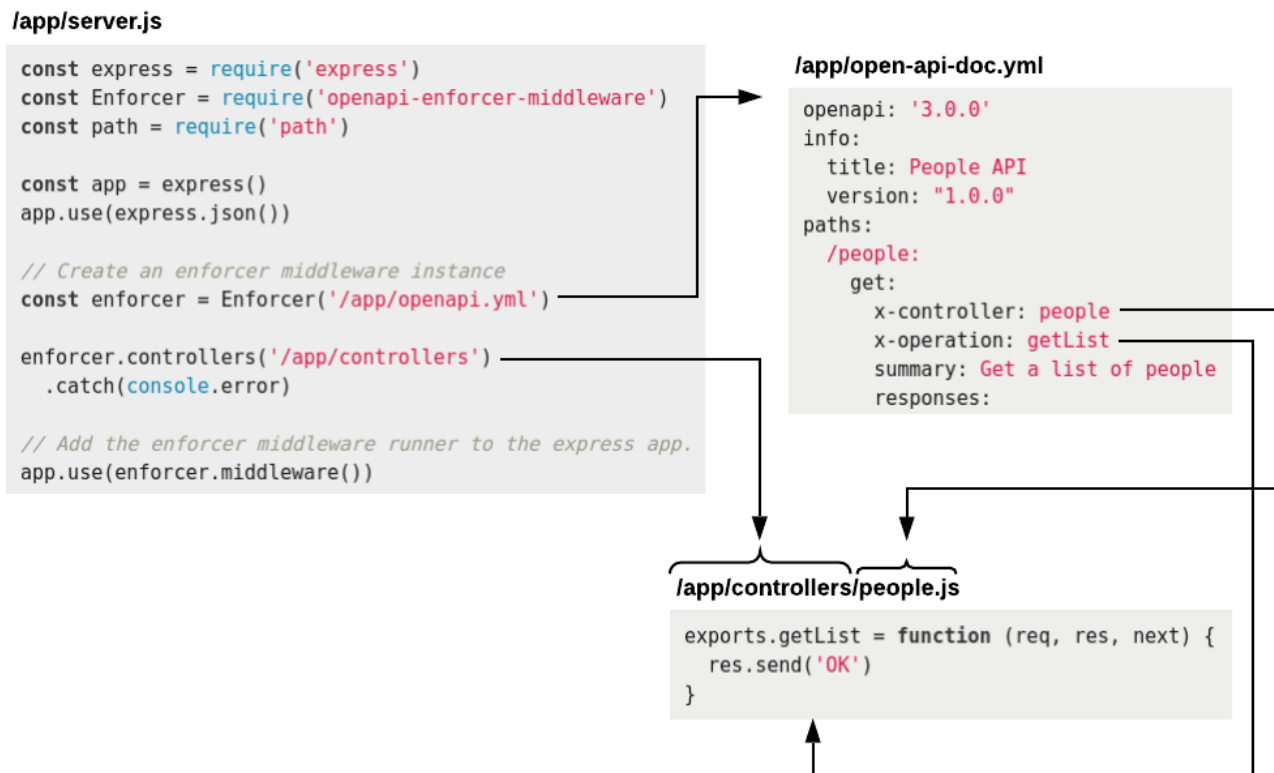


FIGURE 5.7 – Vue globale de la librairie OpenAPI-Enforcer ¹⁰

Passport.js

Comme abordé en section 4.1, une authentification des utilisateurs est requise pour accéder à certaines fonctionnalités. Il est cependant adéquat de ne pas se limiter, car comme expliqué en section 4.2 (partie "Sécurité"), l'association d'un nom d'utilisateur et d'un mot de passe ne constitue que l'une des très nombreuses stratégies pour s'authentifier (par exemple via un réseau social). C'est précisément à cette fin que répond ce middleware d'authentification, permettant ainsi l'intégration de stratégies multiples. En effet, nous avons utilisé dans notre solution 2 stratégies :

- passport-json¹¹ : pour authentifier l'utilisateur lors de sa première visite
- passport-jwt¹² : pour implémenter un mécanisme d'authentification **JWT**¹³, afin de ne pas redemander à l'utilisateur de se ré-authentifier à chaque requête

Supertest

Librairie permettant de réaliser des assertions en HTTP de manière plus aisée. Une de ses particularités est d'être "framework-agnostic", ce qui permet d'utiliser le framework de test de notre choix, dont notamment celui de Facebook : Jest¹⁴. (nous reviendrons sur la dimension des tests de manière plus approfondie dans la section 6.2)

¹⁰<https://byu-oit.github.io/openapi-enforcer-middleware/getting-started#setup>

¹¹<https://github.com/JamesMGreene/passport-json>

¹²<http://www.passportjs.org/packages/passport-jwt/>

¹³<https://jwt.io/>

¹⁴<https://jestjs.io/>

5.2 Client

Cette section est entièrement dédiée à la conception du frontend de **Source Code**. Étant donné qu'il s'agit de la partie visuelle et ergonomique du site, nous aborderons cette thématique de manière plus illustrative.

Pour ce faire, nous allons naviguer à travers les différentes pages de la plateforme en expliquant nos choix d'implémentation et les fonctionnalités qui ont été mises en place.

Afin de vous donner une idée plus claire de ce qui va être abordé, voici une représentation du groupe d'URL de premier niveau que contient **Source Code**.

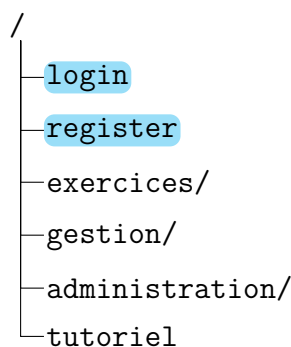


FIGURE 5.8 – URL de premier niveau

Cette figure sera notre fil conducteur tout au long de cette section, car nous allons analyser chaque partie en profondeur. Les URL en **noir** signifient qu'elles renferment un niveau supplémentaire d'URL tandis que les **bleus** désignent une page de contenu.

Les deux premières URL de la figure désignent les interfaces de connexion à **Source Code** ainsi que la création d'un compte.

L'URL */exercices* représente la bibliothèque de ressources informatiques. C'est dans cet espace que le coeur de **Source Code** réside, car c'est là que le système de recherche a été pensé pour tout le reste de la plateforme. En outre, cette même URL contient aussi la consultation de fiches lorsque les critères de recherche de l'utilisateur sont satisfaits. À noter que tout type d'utilisateur peut accéder à ces pages.

L'URL */gestion* renferme tout ce dont un utilisateur (connecté avec son compte) peut effectuer sur la plateforme pour enrichir son utilisation. On y répertorie la gestion/création/modification de ses propres ressources informatiques, la gestion/création/modification de ses favoris pour faciliter la recherche et enfin, la consultation de son profil. Cette partie est accessible aux utilisateurs et (super-)administrateurs.

L'URL */administration* contient les fonctionnalités exigeant le plus haut niveau de privilèges sur **Source Code**. Cette partie est donc entièrement dédiée aux administrateurs et super-administrateurs. Ils peuvent gérer absolument toutes les ressources informatiques disponibles sur la plateforme (impliquant aussi la modification et création de celles-ci), l'importation/exportation de ressources informatiques, la gestion/création/modification de mots-clés, la gestion/création/modification des catégories de mots-clés et finalement la gestion des utilisateurs de la plateforme.

L'URL */tutoriel* concerne un aspect plus ludique de **Source Code**. À travers différentes pages, une thématique de l'application est décortiquée afin que l'utilisateur puisse améliorer sa prise en main avec la plateforme.

Avant d'aller plus loin, considérez la section 2.3 afin de vous remettre en tête ce que nous voulions apporter à **Source Code**.

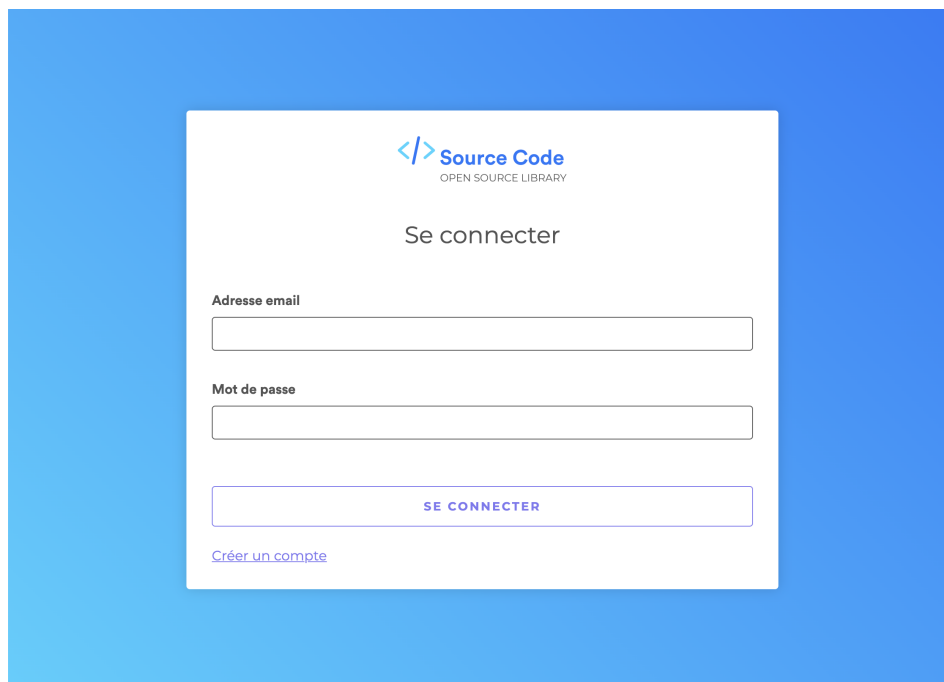
5.2.1 Connexion et création de comptes

La création d'un compte permet d'accéder à des fonctionnalités supplémentaires comme la création de favoris ou de ressources informatiques.

Par défaut, un compte créé offre les privilèges d'un **utilisateur**. Voici pour rappel (section 4.1) les différents types d'utilisateurs pouvant accéder à la plateforme :

- **Visiteur** : C'est la forme la plus simple. Les droits de ce type d'utilisateur se limitent à la navigation dans la bibliothèque de ressources informatiques. Le système de favoris n'est accessible qu'à partir du moment où vous êtes membre de la plateforme. Nous entendons par là l'utilisateur ou l'administrateur.
- **Utilisateur** : L'utilisateur est membre de l'application. Il possède donc un compte et peut participer au partage de ressources informatiques. L'utilisateur a la possibilité de créer des favoris qu'il pourra ainsi utiliser dans la bibliothèque ou depuis ses interfaces de gestion.
- **(Super-)administrateur** : L'administrateur est un rôle capital, car lui seul permet de garantir du contenu de qualité. Il est chargé de valider les ressources informatiques soumises par les utilisateurs, de valider les mots-clés et de créer de nouvelles catégories de mots-clés. En plus de ces droits-ci, l'administrateur possède bien évidemment tous les droits de l'utilisateur !

Le formulaire de création nécessite une adresse email (unique), un mot de passe et un nom complet.



The image shows a login interface for 'Source Code OPEN SOURCE LIBRARY'. The background is a solid blue. In the center, there is a white rectangular box. At the top of this box is the logo, which consists of a blue icon of a code tag (</>) followed by the text 'Source Code' in blue and 'OPEN SOURCE LIBRARY' in smaller, grey capital letters. Below the logo, the text 'Se connecter' is centered. Underneath, there are two input fields: the first is labeled 'Adresse email' and the second is labeled 'Mot de passe'. Both labels are in a small, grey font. Below the password field is a button with the text 'SE CONNECTER' in blue, all-caps. At the very bottom of the white box, there is a link that says 'Créer un compte' in blue.

FIGURE 5.9 – Interface de connexion

Lorsque l'utilisateur possède déjà un compte, il lui suffit de renseigner son adresse email et son mot de passe. Pour l'instant, une session d'une heure lui sera attribuée pour gérer ses ressources, favoris, ... Après quoi il sera redirigé vers la page de login pour entrer à nouveau ses identifiants. Ce comportement est bien évidemment modifiable depuis l'api (cf. section 5.3.2.1), mais nous avons surtout choisi ce timing pour la séance de validation de notre plateforme (cf. C).

5.2.2 Exercices

La bibliothèque constitue l'élément central de **Source Code** pour le partage de ressources. C'est dans cet espace que les ressources informatiques validées par la communauté apparaîtront.

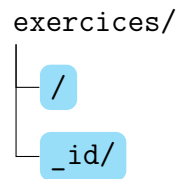


FIGURE 5.10 – Partie bibliothèque (URL)

Cette section peut se découper en deux parties distinctes :

- La bibliothèque (accessible depuis l'URL `/exercices`) : nous y aborderons le système de recherche mis en place, les privilèges utilisateurs sur cette page ainsi que les choix ergonomiques.
- La fiche d'une ressource informatique (accessible depuis l'URL `/exercices/_id15`) : nous y expliquerons le référencement de la fiche afin d'établir un lien avec de la bibliothèque.

5.2.2.1 La bibliothèque

La page *bibliothèque* contient toutes les ressources informatiques ajoutées par la communauté et ayant été validées par les administrateurs de la plateforme. Elle est accessible publiquement, pour tout type d'utilisateurs.

Concernant l'interface, nous nous sommes inspirés de *Coderbyte* et *Hackerrank* pour le panneau de filtres et le preview des ressources informatiques (cf. section 2.1). Il était important pour nous de conserver une interface simple et déjà utilisée par certaines applications web.

¹⁵Lorsqu'une URL contient `_id`, cela peut être remplacé par un nombre naturel $n \in \mathbb{N} \setminus \{0\}$. Les id's représentent l'id d'une entité stockée dans la base de données. Exemple : pour récupérer l'exercice ayant l'id 42 (pour autant qu'il existe), il suffit de renseigner l'URL `/exercices/42`.

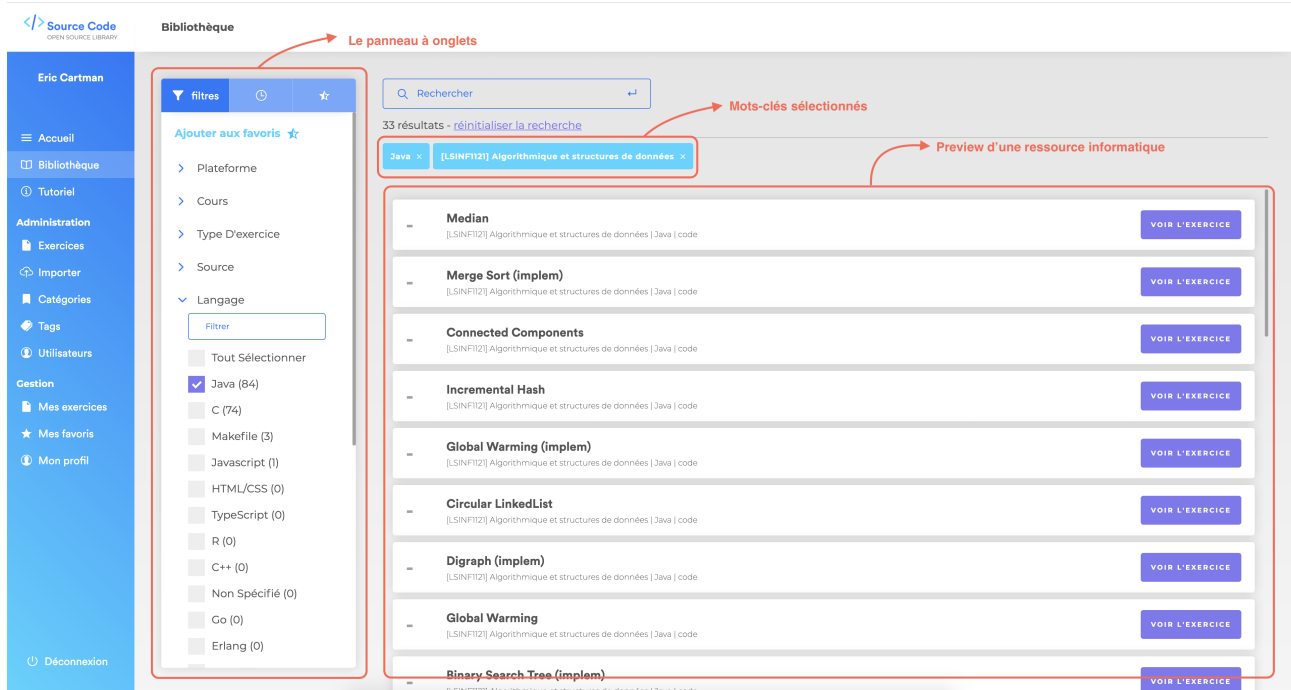


FIGURE 5.11 – page bibliothèque

5.2.2.1.1 Preview d'une ressource informatique

La section de droite représente tous les exercices correspondants aux critères de recherche. Il suffit de scroller dans cette section afin de charger les ressources informatiques suivantes.

Le preview d'une ressource contient les informations suivantes :

- Le titre de la ressource
- Quelques mots-clés que nous jugeons récurrents et pertinents pour mieux identifier le type de ressource (si référencés sur celle-ci) :
 - Le cours
 - Le langage
 - La difficulté
 - La thématique
 - Le type d'exercice

Nous avons choisi 5 mots-clés maximum à afficher pour que le preview de la ressource soit toujours facilement visible. Pour la justification du choix des mots-clés, nous l'expliquons en annexe B, table 4.3 avec quelques éléments récurrents que nous avons pu identifier avec les autres normes. Les autres mots-clés seront consultables depuis l'interface de la fiche (cf. section 5.2.2.2)

- La cotation de la ressource (sur 5). Nous détaillons d'ailleurs son utilité dans la section 5.2.2.2.

5.2.2.1.2 Le panneau à onglets

Pour effectuer une recherche dans la bibliothèque, il suffit de taper un titre dans la barre de recherche et/ou d'utiliser le panneau à onglets (élément central de la recherche).

Il existe dès lors 3 types d'onglets :

Filtres Ils permettent d'affiner la recherche de ressources informatiques. Les mots-clés cochés apparaîtront juste au-dessus des résultats. Si un mot-clé ne convient plus à la recherche, il suffit de décocher un des tags en question ou de cliquer sur la croix du label représentant le mot-clé sélectionné.

Comme vous pourrez le constater sur la figure 5.11, le panneau de filtres n'est plus à la même position que ce que nous avons imaginé avec notre patchwork 2.2. Dans ce dernier, nous avons intégré le filtrage de mots-clés dans une interface dédiée à cet effet.

Dans l'interface actuelle, nous voulions que tout soit facilement accessible compte tenu du fait que nous allions intégrer l'historique et le système de favori au même endroit (ce que nous n'avions pas encore décidé au stade du patchwork).

La page que nous avons prévue sur le patchwork pour les filtres (figure 2.2) s'apparente plutôt à une interface avancée. Nous expliquerons d'ailleurs les quelques pistes d'améliorations qui pourraient être apportées à ce sujet dans une prochaine mise à jour de la plateforme en chapitre 7.

Le système de filtres fonctionne comme *Hackerrank* et *Coderbyte* (section 2.1) :

- Plusieurs mots-clés peuvent être choisis dans une même catégorie. Dans ce cas, la recherche donnera des résultats comprenant au moins un des mots-clés sélectionnés.
- Lorsqu'un mot-clé est sélectionné dans deux catégories différentes, la recherche prendra en compte toutes les ressources informatiques comprenant au moins un mot-clé sélectionné dans chacune des catégories.

Si vous possédez un compte, vous pouvez enregistrer les critères de recherche actuels en cliquant sur "Ajouter aux favoris". Après avoir fourni un nom pour le favori, ce dernier apparaîtra désormais dans l'onglet favoris.

Historique L'historique vous permet de naviguer à travers les recherches que vous avez précédemment effectuées. Il restera disponible tout au long de la session, après quoi il sera réinitialisé.

L'historique affiche les recherches précédemment effectuées, de la plus récente à la plus ancienne. Le titre de recherche s'affichera en bleu tandis que les mots-clés sélectionnés seront affichés en mode "pêle-mêle", séparés par une barre verticale (|).

Favoris (seulement lorsqu'on est connecté) Les favoris créés apparaîtront dans ce panneau. Pour utiliser un des favoris, il suffit de cliquer sur l'un d'eux et la recherche s'actualisera depuis la même page avec les nouveaux critères de recherche pris en compte.

Vous pouvez supprimer ou modifier les favoris depuis ce panneau en passant la souris sur un favori, puis en sélectionnant une des deux icônes qui s'affichent.

- Le **crayon** permet de modifier le favori. Il mènera donc sur la page de modification de ce favori (cf. section 5.2.3.3).
- La **poubelle** permet de supprimer le favori directement depuis le panneau.

5.2.2.2 La fiche d'une ressource informatique

Les ressources informatiques sont le cœur de **Source Code**. Elles représentent toutes les contributions émises par la communauté. Elles peuvent être consultées depuis la bibliothèque en cliquant sur "voir l'exercice".

Voici à quoi ressemble la structure d'une fiche :

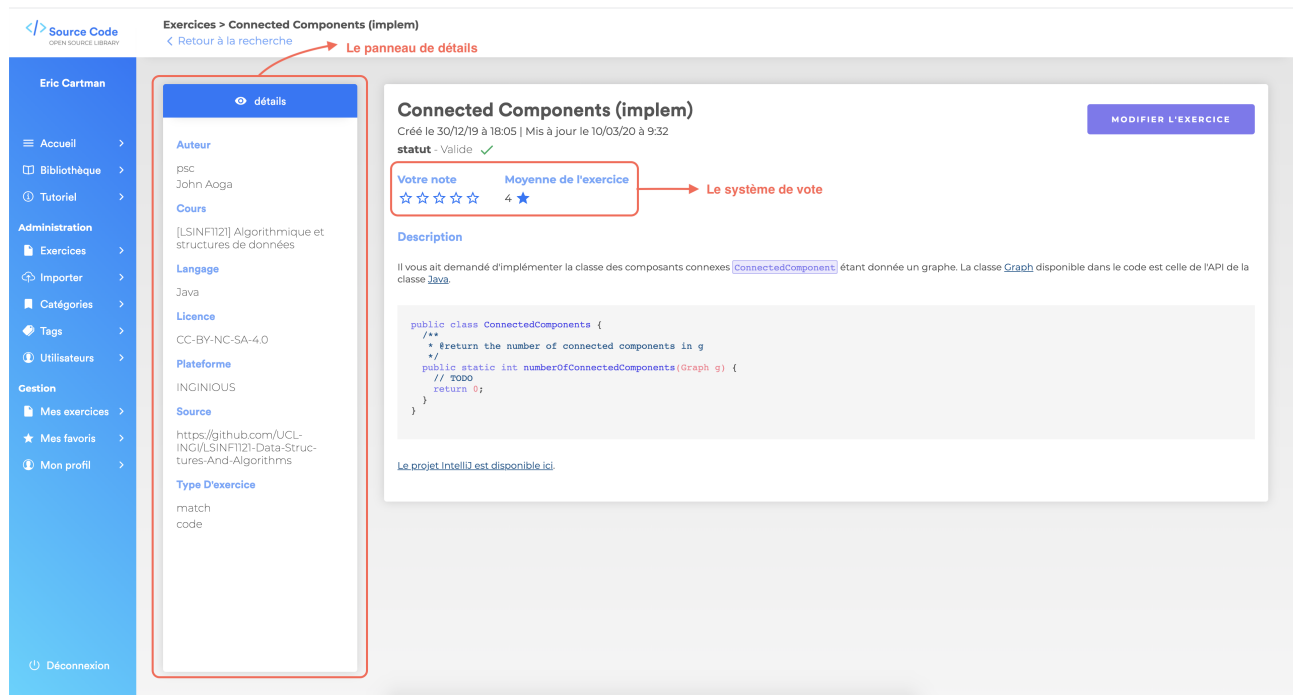


FIGURE 5.12 – la représentation d'une fiche

Lorsque la ressource vous appartient ou que vous êtes un administrateur, un bouton de modification apparaît dans le coin supérieur droit de la fiche de la ressource. Cliquer sur le bouton mènera sur le formulaire de modification de la ressource (cf. section 5.2.3.1 pour la gestion côté utilisateur ou section 5.2.4.1 pour la gestion côté (super-)administrateur).

5.2.2.2.1 Panneau de détails

Le panneau latéral contient tous les mots-clés qui ont été ajoutés pour identifier la ressource. Plus il y en a, plus cette dernière sera favorablement référencée sur la plateforme.

En effet, la recherche dans la bibliothèque fonctionne avec les filtres (mots-clés) et/ou le titre de recherche. Plus le titre de la ressource sera pertinent, plus facile elle pourra être retrouvée en tapant une partie du titre dans la barre de recherche. De la même manière, plus la ressource informatique contient de mots-clés dans des catégories de mots-clés différentes, plus il sera facile de la retrouver avec le panneau de filtres. Il est donc important que le créateur de la fiche fasse preuve d'exhaustivité dans le choix des mots-clés et de pertinence. Le cas échéant, les administrateurs se réservent le droit d'invalidier la ressource ou de la laisser passer, sans grande chance d'être retrouvée par d'autres utilisateurs de la plateforme.

5.2.2.2.2 Système de vote

Sur la fiche de la ressource informatique, vous pouvez donner une note sur la qualité de la ressource sur 5 étoiles. Il suffit de cliquer sur une des étoiles pour donner un score (seulement si vous possédez un compte).

Là aussi, le vote joue un rôle important dans le référencement de la ressource informatique, car les utilisateurs peuvent filtrer les ressources informatiques en fonction de leur note.

Nous avons choisi un système de vote sur 5 étoiles, car notre analyse (section 2.1) a révélé cette caractéristique commune sur de nombreux sites. La plateforme *Hackerrank* utilise d'ailleurs ce système.

5.2.3 Gestion

Le module de gestion comprend :

- La gestion de ressources informatiques personnelles (*mes-exercices/*)
- La gestion de favoris (*mes-favoris/*)
- La consultation du profil (*profil*)

Ce module est accessible aux utilisateurs (possédant un compte) et aux administrateurs.

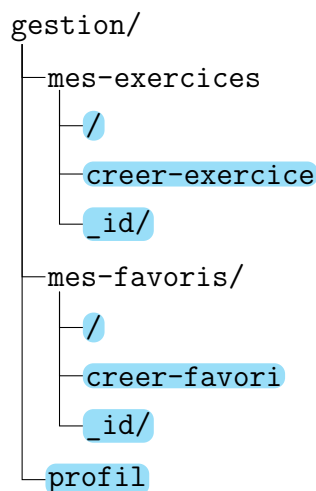


FIGURE 5.13 – Module de gestion (URL)

5.2.3.1 Gestion de ressources informatiques personnelles

La recherche fonctionne de la même manière que dans la bibliothèque, à la différence qu'elle est effectuée dans les ressources informatiques personnelles seulement. Pour comprendre le fonctionnement du panneau à onglets, consultez la section 5.2.2.1.2.

Source Code
OPEN SOURCE LIBRARY

Gestion > Mes exercices

Le panneau à onglets

Le statut d'une ressource informatique

Eric Cartman

Accueil >
Bibliothèque >
Tutoriel >
Administration >
Exercices >
Importer >
Catégories >
Tags >
Utilisateurs >
Gestion >
Mes exercices >
Mes favoris >
Mon profil >
Déconnexion

filtrer

Ajouter aux favoris

Plateforme
Cours
Type D'exercice
Source
Langage
Auteur
Thématique
Mots Clés
Licence
Niveau
Difficulté
Cotation
Statut

Mes exercices

Nombre de résultat(s) : 208 - réinitialiser la recherche

Rechercher

CRÉER UN EXERCICE +

Titre	Nb de votes	Moyenne	Mis à jour	Fichier(s)	Status
Liste doublement chaînée	1	3	26/04/20 à 9:53	×	✓
Stockage d'un vecteur de réels dans un fichier	0	-	26/04/20 à 9:53	×	✓
ASCII Decoder	0	-	26/04/20 à 9:53	×	✓
FList MergeSort	0	-	26/04/20 à 9:53	×	✓
Infinite Streams	0	-	26/04/20 à 9:53	×	✓
Depth First Paths	2	4.5	26/04/20 à 9:53	×	✓
Manipulate the memory	0	-	26/04/20 à 9:53	×	✓
Trier une liste chaînée	0	-	26/04/20 à 9:53	×	✓
Règles de participation aux cours	0	-	26/04/20 à 9:53	×	✓
Breadth First Paths	0	-	26/04/20 à 9:53	×	✓
Parcours de fichiers	1	2	26/04/20 à 9:53	×	✓
Reading arguments	0	-	26/04/20 à 9:53	×	✓

FIGURE 5.14 – Gestion des ressources informatiques personnelles (mes-exercices/)

Plusieurs informations sont disponibles pour identifier les ressources informatiques :

- Le titre de la ressource
- Le nombre de votes émis par la communauté
- La moyenne (sur 5) des votes émis par la communauté
- La date de mise à jour de la ressource informatique
- La présence d'une archive zip pour la ressource informatique
- Le statut de la ressource

5.2.3.1.1 Statuts d'une ressource informatique

Nous prêtons attention à la qualité du contenu sur la plateforme. Nous avons donc mis en place un système de statuts pour donner une indication sur l'état d'une ressource informatique.

Voici les différents états que peut prendre la ressource informatique (découle de notre analyse bibliographique en annexe B) :

-  **Brouillon**
 - Signifie que la ressource n'est pas encore prête à être validée par l'administration. La ressource n'est donc pas disponible depuis la bibliothèque et uniquement accessible depuis l'interface de gestion du créateur de la fiche et des administrateurs.
-  **En attente de validation**
 - La ressource est mise en attente pour révision. Un administrateur se chargera de la valider ultérieurement.

-  **Valide**
 - Lorsque la ressource est acceptée par l’administration, la ressource est alors validée et disponible publiquement depuis la bibliothèque.
-  **Invalide**
 - Lorsque la ressource n’est pas considérée comme valide (mauvaise qualité d’écriture, doublons, incohérences), l’administrateur se réserve le droit d’invalider la ressource. Elle pourra néanmoins être modifiée pour la refaire passer en phase de révision.
-  **Archive**
 - Quand une ressource est archivée, elle est alors uniquement accessible par les administrateurs et par son créateur.

Le diagramme UML pour l’état d’une ressource informatique se retrouve en figure 4.1.

5.2.3.1.2 Gestion du statut de la ressource

Pour modifier l’état d’une ou plusieurs ressources, il faut dans un premier temps les cocher. Une liste déroulante d’actions apparaîtra à côté du bouton de création d’exercices avec les options suivantes :

- Publier (envoyer pour révision)
- Mettre en brouillon
- Archiver

Plus d’options seront disponibles pour les administrateurs, car il est évident qu’un utilisateur ne puisse pas valider sa propre ressource par exemple.

Rechercher		PLUS D’ACTIONS ▼		CRÉER UN EXERCICE +	
Titre	Nb de votes	Moyenne	Mis à jour	Fichier(s)	Status
☒ Liste doublement chaînée	1	3	26/04/20 à 9:53		

FIGURE 5.15 – Gestion des ressources informatiques personnelles (creer-exercice ou _id)

5.2.3.2 Création/modification d’une ressource

La fiche d’une ressource informatique est composée des éléments suivants :

- Un titre (obligatoire)
- Une description (obligatoire)
- Des mots-clés (obligatoire)
- Une archive zip en guise de fichier
- Un lien externe (URL)

Un point important sur ce formulaire est la possibilité de proposer un mot-clé à la plateforme. Il peut arriver que l'application ne propose pas tous les mots-clés auxquels les utilisateurs aspirent. De la même manière que ces derniers contribuent à la plateforme en ajoutant des ressources, pourquoi ne pourraient-ils pas aussi contribuer à étendre la liste des mots-clés ? Il suffit donc de sélectionner une catégorie particulière et fournir un nom pour le mot-clé.

Si ce dernier n'existe pas encore dans le système, il sera alors ajouté dans l'interface de gestion des mots-clés avec le statut "en attente" afin que les administrateurs puissent le prendre en charge.

The screenshot shows the 'Créer un exercice' page. On the left is a sidebar with the user's name 'Eric Cartman' and navigation links: Accueil, Bibliothèque, Tutoriel, Administration (Exercices, Importer, Catégories, Tags, Utilisateurs), and Gestion (Mes exercices, Mes favoris, Mon profil). The main content area has a breadcrumb 'Gestion > Mes exercices > Créer un exercice' and a search bar. The form itself has sections for 'Titre *', 'Description *', 'Tags *', and 'Code couleur'. There are buttons for 'COMMENCER LA SÉLECTION', 'CHOISIR UN FICHIER...', and 'Uri vers l'exercice'.

Lorsque vous remplissez le champ titre et/ou ajoutez des tags, le panneau à onglets "exercices" affiche toutes les ressources informatiques similaires (au sens strict) à la fiche que vous créez. Cela signifie que toutes les ressources informatiques contenant le titre entré et TOUS les mots-clés que vous avez référencés seront ainsi présents dans ce panneau.

The screenshot shows the 'exercices' tab with a search bar and a dropdown menu. Below the search bar, it says '12 résultat(s)'. The results are listed as follows:

- Binary Search Tree**
Nb de votes : 0
Moyenne : -
- Binary Search Tree Iterator (implem)**
Nb de votes : 0
Moyenne : -
- Reading integers in a binary file**
Nb de votes : 0
Moyenne : -

5.2.3.2.1 Soumission de la ressource

Deux options de soumission sont disponibles :

- **Brouillon** : Vous estimez que la ressource informatique n'est pas encore prête pour être validée, vous la laissez donc en stand-by pour la modifier plus tard. À noter que vous **DEVEZ** remplir les champs obligatoires pour l'enregistrer sous cet état. Il s'agit d'un choix technique décidé au niveau de l'API, pour conserver un seuil minimal de qualité dans les contributions.
- **Soumettre** : Vous estimez que la ressource informatique est prête pour inspection. En cliquant sur soumettre, la ressource se met automatiquement en mode "en attente", jusqu'à ce qu'un administrateur la prenne en charge pour validation. Ce texte est remplacé par **valider** dans le cas de l'administrateur. Le statut de la ressource sera ainsi valide.

Dans le cas de l'administrateur, ce dernier possède les options suivantes, en plus des précédentes :

- **Invalider** : Vous estimez que la ressource informatique n'est pas correcte, car elle comporte des défauts, des incohérences ou est un duplicaté d'une autre ressource informatique.
- **Archiver** : Vous estimez que la ressource informatique n'a plus d'utilité à demeurer dans la bibliothèque. En choisissant cette option, seuls les administrateurs et le créateur de la ressource pourront consulter cette ressource dans le futur.
- **Mettre en attente** : Vous estimez que la ressource informatique est prête pour inspection. En cliquant sur soumettre, la ressource se met automatiquement en mode "en attente", jusqu'à ce qu'un administrateur la prenne en charge pour validation.

5.2.3.3 Gestion de favoris

Les favoris permettent d'enregistrer des critères de recherche (titre de recherche et filtres) pour pouvoir les réutiliser par après, sans devoir parcourir le panneau de filtres pour effectuer la même recherche.

Pour gérer les favoris, il suffit de passer la souris sur l'un d'eux. Deux icônes apparaîtront, le crayon permettant de modifier le favori et la poubelle permettant de le supprimer.

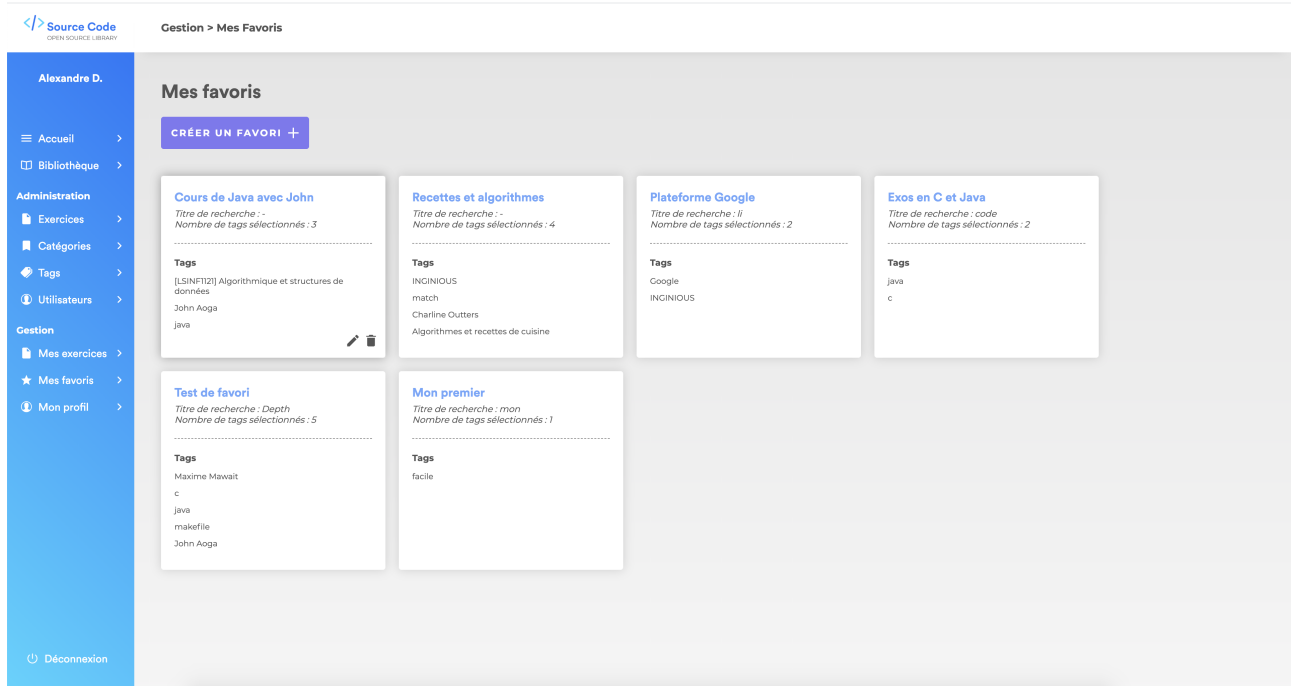


FIGURE 5.16 – Gestion des favoris (mes-favorites/)

5.2.3.3.1 Formulaire de création/modification d'un favori

Voici les différents champs à remplir pour créer un favori :

- **Nom du favori (obligatoire) :** donner un nom au favori.
- **Titre de la sélection :** titre de recherche.
- **Mots-clés :** les tags qui vont être appliqués pour votre recherche.

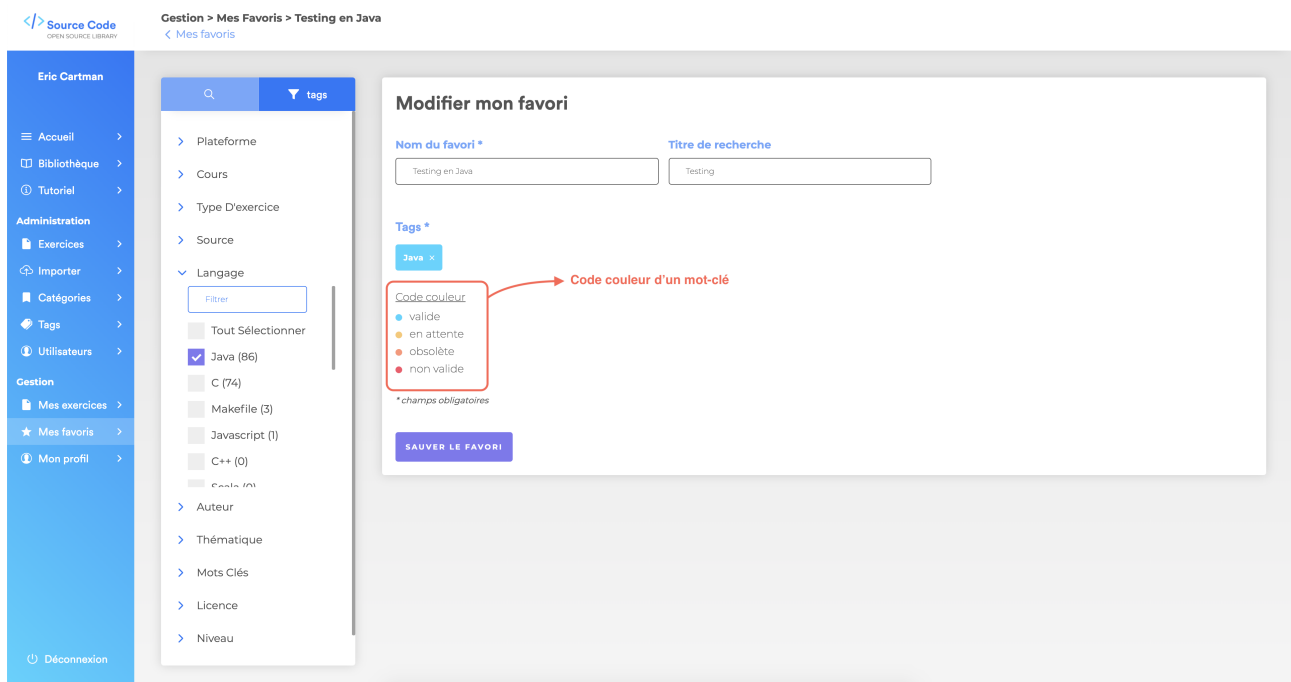


FIGURE 5.17 – Création/modification des favoris (creer-favori ou _id)

En ce qui concerne l'ajout de mots-clés, un code couleur représentant leur statut est présenté (figure 4.2 pour le schéma UML à états). À noter que ce choix de statuts découle de notre analyse bibliographique en annexe B. :

- Un mot-clé **valide** pourra être utilisé depuis la bibliothèque pour filtrer les ressources informatiques. Ceci garantit un meilleur référencement pour la ressource.
- Un mot-clé **obsolète** risque d'être remplacé par un autre mot-clé ou d'être supprimé ultérieurement.
- Un mot-clé **en attente** n'a pas encore passé la validation d'un administrateur.
- Un mot-clé **non valide** n'a pas été accepté l'administration. Il se peut qu'il soit modifié ultérieurement.

Nous parlons du statut des mots-clés de manière plus concise dans la section 5.2.4.3 concernant la gestion des mots-clés côté administrateur.

5.2.3.4 La consultation de profil

À ce stade-ci du développement, le frontend autorise seulement la consultation du profil et non la modification de celui-ci par l'utilisateur. En revanche, cette fonctionnalité est déjà disponible côté backend (cf. documentation : https://sourcecodeoer.github.io/sourcecode_api/#operation/updateUser).

Dans le chapitre 7, nous développons quelques pistes d'améliorations concernant cette page afin de la rendre encore plus utile.

5.2.4 Administration

Le module d'administration comprend :

- La gestion des ressources informatiques (*exercices/*)
- L'importation de ressources informatiques (*importer-des-exercices*)
- La gestion des mots-clés (*tags/*)
- La gestion des catégories de mots-clés (*categories/*)
- La gestion des utilisateurs (*utilisateurs*)

Ce module est uniquement accessible aux (super-)administrateurs.

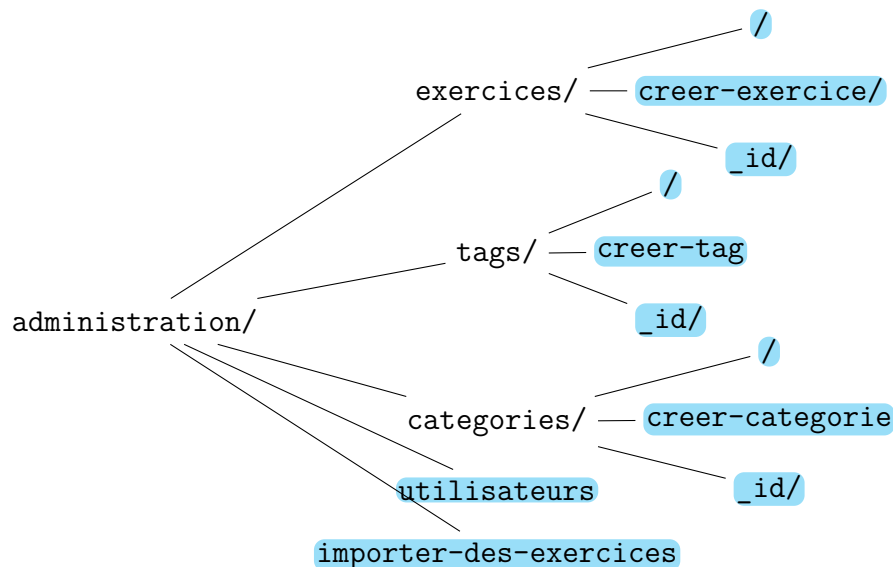


FIGURE 5.18 – Partie administration (URL)

5.2.4.1 La gestion des ressources informatiques

La gestion des ressources informatiques fonctionne exactement comme la gestion de ses propres ressources informatiques. Reportez-vous donc à la section 5.2.3.1 pour connaître les fonctionnalités communes aux utilisateurs et aux administrateurs.

L'administrateur possède en outre les fonctionnalités additionnelles suivantes pour la gestion des ressources informatiques :

5.2.4.1.1 Export des résultats de recherche

En filtrant les ressources informatiques par mots-clés et/ou par titre de recherche, vous pouvez exporter tous les résultats correspondant à ce filtrage dans un fichier JSON. Le format de ce fichier est décrit sur l'api (https://sourcecodeoer.github.io/sourcecode_api/#operation/ExportExercises).

5.2.4.1.2 Actions supplémentaires

Par rapport à un utilisateur classique, vous pouvez changer l'état d'une ressource informatique de manière plus large comme l'archivage ou la validation.

Pour connaître ces états et leur utilité, reportez-vous à la section 5.2.3.1.1 et la section 4.1 pour l'état d'une fiche.

5.2.4.1.3 Champ supplémentaire dans le tableau des ressources

Le créateur de la ressource informatique. À ce stade du développement, nous ne pouvons pas contacter l'utilisateur directement depuis la plateforme. C'est définitivement une piste que nous allons explorer dans le chapitre 7, car nous aurions voulu intégrer un système de tickets complétant la validation par statut d'une ressource.

5.2.4.2 L'importation de ressources informatiques

Améliorer le processus de création de ressources informatiques est une de nos principales préoccupations pour **Source Code**. Nous avons alors développé une interface prenant en charge l'importation de ressources informatiques.

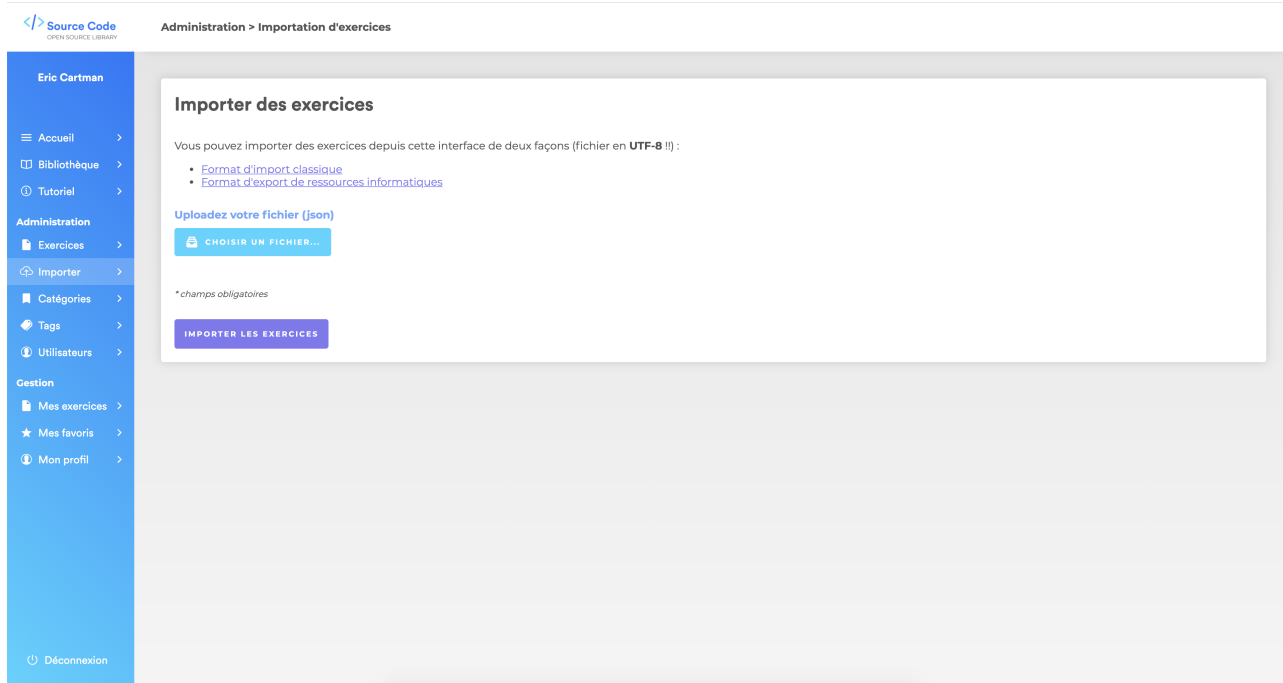


FIGURE 5.19 – Importation des ressources informatiques (importer-des-exercices)

Les ressources informatiques doivent être importées avec un fichier JSON respectant un des deux formats que nous vous invitons à consulter :

- Format d'import classique (https://sourcecodeoer.github.io/sourcecode_api/#operation/createMultipleExercises)
- Format d'export de ressources informatiques (https://sourcecodeoer.github.io/sourcecode_api/#operation/ExportExercises)

Nous avons choisi de réserver cette fonctionnalité aux (super-)administrateurs, car il pourrait y avoir des dérives au niveau des simples utilisateurs (attaques DDOS sur l'API, ...).

5.2.4.3 La gestion des mots-clés

Les (super-)administrateurs ont la possibilité de gérer les mots-clés présents sur la plateforme. Ils peuvent donc créer des mots-clés, les modifier et changer le statut de ceux-ci.

L'interface de gestion prévue à cet effet est semblable à celle de la gestion des ressources informatiques (cf. 5.2.4.1), mis à part que les filtres se limitent aux catégories de mots-clés et au statut des mots-clés.

5.2.4.3.1 Statuts des mots-clés

De la même manière que pour les ressources informatiques, nous proposons un système de statuts pour donner une indication sur l'état d'un mot-clé (la figure 4.2 représente ces états sur un diagramme UML). Nous voulons que le système de mots-clés soit évolutif et flexible. Nous laissons donc les utilisateurs de la plateforme ajouter leurs ressources informatiques, mais aussi des propositions de mots-clés afin d'agrandir le vocabulaire de **Source Code**.

Cela passe bien évidemment par la validation des administrateurs, qui agissent de concert pour offrir un vocabulaire contrôlé aux utilisateurs, évoluant ainsi en fonction des besoins (cf. analyse bibliographique en annexe B).

Voici la liste des différents statuts que peut prendre un mot-clé :

- **En attente de validation**

- Le mot-clé est mis en attente pour révision. Un administrateur se chargera de le valider ultérieurement.

- **Valide**

- Lorsque le mot-clé est accepté par l'administration, il est alors validé et disponible publiquement dans la bibliothèque.

- **Invalide**

- Lorsque le mot-clé n'est pas considéré comme valide (faute de frappe, doublons, incohérences), l'administrateur se réserve le droit d'invalider le tag. Il pourra néanmoins être modifié (par un admin) pour le revalider plus tard.

- **Archive**

- Quand un mot-clé n'est plus d'utilité ou est remplacé par un autre.

La suppression physique d'un mot-clé est possible, mais elle est réservée au super-administrateur car c'est une opération irréversible. C'est pour cela que l'existence d'un statut "obsolète" a été mis en place : garder une trace sur l'évolution du vocabulaire de **Source Code**.

5.2.4.3.2 Création/modification de mots-clés

Le formulaire de création/modification d'un mot-clé requiert les éléments suivants pour être publié :

- **Un nom** : le nom du mot-clé
- **Une catégorie** : la catégorie de mots-clés auquel il se rapporte

La publication du mot-clé s'effectue en choisissant un des 4 statuts expliqués précédemment.

5.2.4.4 La gestion des catégories de mots-clés

La gestion des catégories de mots-clés se limite uniquement à la modification du titre ou à la création d'une nouvelle catégorie. La suppression est réservée au super-administrateur car elle implique la suppression de tous les mots-clés qui se rapportent à cette catégorie (opération dangereuse et irréversible).

5.2.4.5 La gestion des utilisateurs

Les administrateurs ont accès à la liste de tout type d'utilisateur ayant un compte enregistré sur **Source Code**. Ils peuvent accéder aux informations suivantes :

- Le nom de l'utilisateur
- L'email de l'utilisateur
- Le rôle de l'utilisateur (simple utilisateur, administrateur, super-administrateur)

Depuis cette interface, il est possible de modifier le rôle de certains utilisateurs pour les promouvoir ou les destituer d'un rôle. Cette fonctionnalité est uniquement disponible pour le super-administrateur car lui seul devrait garder le contrôle permanent de l'application pour éviter tout débordement.

5.3 Serveur

Cette section est entièrement dédiée à la conception du backend de **Source Code**. Puisque cette partie est l'extrême opposé de la précédente (cf. section 5.2), nous l'aborderons par sa réalisation technique, en nous concentrant sur l'essentiel.

5.3.1 Structure du projet

Pour répondre aux besoins exprimés en chapitre 4 et sur base de nos choix (cf. section 5.1), nous avons élaboré la présente structure :

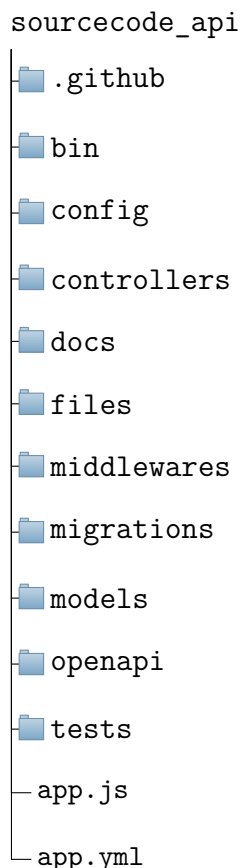


FIGURE 5.20 – Arborescence de l'API

Cette structure ne vous est peut-être pas inconnue, car il s'agit de la structure officielle de Sequelize¹⁶, sur laquelle nous avons ajouté quelques dossiers et fichiers :

- `openapi` : nos fichiers OAS, utilisés par OpenAPI-Enforcer (cf. figure 5.7) (le point d'entrée principal étant `app.yml`)
- `controllers` : nos fichiers chargés de répondre aux requêtes **HTTP**, utilisés par OpenAPI-Enforcer (cf. figure 5.7)
- `tests` : nos tests pour valider l'implémentation (cf. section 6)
- `docs` : la documentation de l'API en **HTML**¹⁷ (cf. figure 5.21 pour un exemple)
- `upload / files` : resp. lieu de stockage des fichiers importés / stockés
- `.github` : nos scripts d'automatisation de type Github Actions (cf. section 5.4.3)
- `bin` : du code n'entrant dans aucun autre dossier
- `app.js` : le point d'entrée principal de l'API, construite sur Express (cf. section 5.1)

¹⁶<https://github.com/sequelize/cli> - commande "sequelize init"

¹⁷Celle-ci est automatiquement publiée sur https://sourcecodeoer.github.io/sourcecode_api/. Pour la générer en local, cela se fait via la commande : `npx redoc-cli bundle api.yml -o docs/index.html`

5.3.2 Points clés de l'implémentation

5.3.2.1 Paramétrisation de l'API

Comme révélé en section 4.2, il convient d'accorder un certain contrôle aux utilisateurs finaux. Une manière de configurer une telle application est d'utiliser des variables d'environnement, qui permettent, contrairement à d'autres approches (comme les fichiers de configuration **JSON**, **YAML**, ...), de changer de valeur directement, sans éteindre/redémarrer l'application.

Nous avons accordé les possibilités suivantes (la liste détaillée des variables d'environnement est disponible dans le fichier README.MD) :

- Changer le port utilisé par l'API
- Changer l'URI pour se connecter à la base de données
- Changer la politique de traçabilité (cf. section 4.2 - partie "Sécurité")
- Changer la durée de validité du token **JWT** (cf. section 5.1)

5.3.2.2 Documentation de l'API

De manière synthétique, OAS¹⁸ a été plébiscité pour les raisons suivantes :

- Peut être utilisé par des outils divers pour de multiples finalités :
 - apporter des fonctionnalités supplémentaires / construire un projet plus rapidement, comme OpenAPI-Enforcer (cf. figure 5.7)
 - pour générer de la documentation (cf. figure 5.21), comme Redoc¹⁹
- Non dépendant du code : de ce fait, plus facile à maintenir et à faire évoluer

Change the status of given exercises

Change the status of given exercises.

The following restrictions are applied on simple user(s) (no restriction for admin):

1. He/She can only modify his/her own exercises
2. Only the following states are allowed :
 - DRAFT
 - PENDING
 - ARCHIVED

AUTHORIZATIONS: bearerAuth

REQUEST BODY SCHEMA: application/json

exercises	Array of integers non-empty An array of exercises IDs
state	string Enum: "DRAFT" "PENDING" "VALIDATED" "NOT_VALIDATED" "ARCHIVED" Status of the exercise into the system. Currently, 5 states are possibles : 1. DRAFT : The default state (for example when an exercise is inserted into the system) 2. PENDING : When an exercise is ready for review 3. VALIDATED : When an exercise is validated by an admin 4. NOT_VALIDATED : When an exercise is refused by an admin 5. ARCHIVED : When an exercise is archived / soft deleted

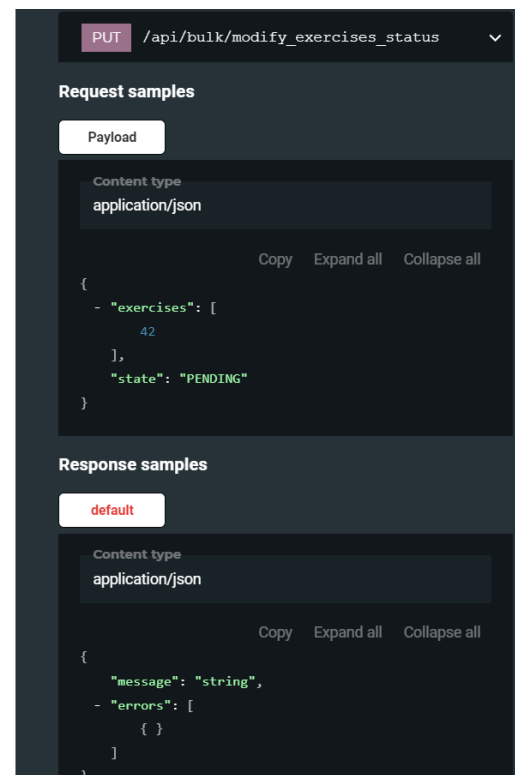


FIGURE 5.21 – Exemple de documentation **HTML** générée sur base de fichiers OAS

¹⁸Pour des exemples concrets, veuillez vous référer au fichier `api.yml` et le dossier `openapi`

¹⁹<https://github.com/Redocly/redoc>

5.3.2.3 Middlewares

Pour rappel (cf. section 5.1), Express a été le "squelette" sur lequel nous avons construit notre application. Une de ses particularités est l'usage de middlewares²⁰ dont nous pouvons expliquer le fonctionnement général par le biais de cette illustration :

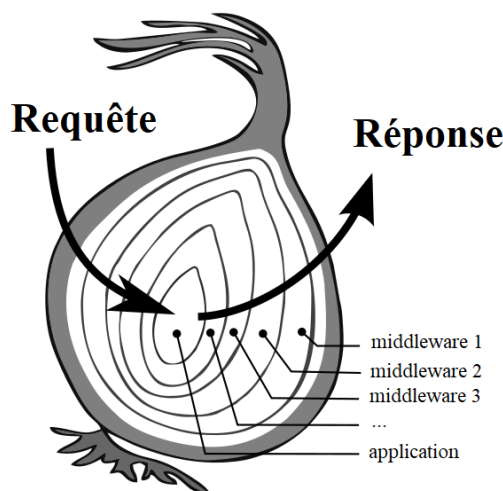


FIGURE 5.22 – Middlewares : modèle de l'oignon

- La requête est interceptée par le serveur.
- La requête traverse successivement chaque "couche" (middleware 1, 2, ...) jusqu'à atteindre le cœur de l'application, à condition qu'il n'y ait pas eu d'erreurs avant.
- Quelle que soit la situation, une réponse est toujours envoyée au client de la requête.

La grande force de cette approche est qu'elle permet de déléguer²¹ et se concentrer ainsi sur l'essentiel. Il est également possible de combiner plusieurs middlewares en un seul dans une même "couche", par des bibliothèques comme connect-chain-if²².

5.3.2.4 Recherche par mots-clés

Pour rappel, nous avons abordé au chapitre 4 (cf. table 4.1) la problématique de la recherche et de la solution théorique que nous envisagions. Notre implémentation repose sur un champ précalculé²³, de type tableau d'entiers (ex. [1,2,3] signifie que la fiche dispose des mots-clés avec les identifiants 1, 2 et 3). Avec notre implémentation (cf. code D.3), notre format de requête²⁴ et les "scopes" en Sequelize (cf. section 5.1), il est possible de l'appliquer à l'exemple en table 4.1, avec les 3 mêmes requêtes que nous avons utilisées :

recherche / fiche	F1	F2	F3
[[1, 2]]	✓	✓	
[2, -1]		✓	
[-3]	✓		✓

TABLE 5.4 – Solution technique de la recherche par mots-clés

²⁰Par exemple, aux middlewares en section 5.1 : Passport.js et OpenAPI-Enforcer

²¹Par exemple, l'authentification (cf. section 4.2), la validation des requêtes (cf. figure 5.7) ...

²²<https://www.npmjs.com/package/connect-chain-if>

²³Par souci de performances, il est préférable d'avoir cette information directement, plutôt que la calculer sur la table intermédiaire entre les fiches et les mots-clés à chaque fois.

²⁴Les crochets ayant la même signification que les parenthèses de la **FNC** ; le $\neg$ d'un entier s'encode avec le même entier, mais avec un symbole négatif ; le $\wedge$ s'effectue sur les éléments du premier niveau tandis que le $\vee$ s'effectue lui sur le 2e niveau

5.4 Divers

Dans cette section, nous aborderons quelques éléments secondaires de **Source Code**. En effet, en dehors du développement, il existe bien d'autres facettes importantes du cycle de vie²⁵ d'un logiciel : le déploiement / la maintenance ...

5.4.1 CLI

Comme nous l'évoquions précédemment (cf. annexe B), il convient de disposer d'outils pour indexer / mettre en ligne un nombre conséquent de ressources informatiques, provenant de diverses sources, en un minimum de temps. C'est pour répondre à ce besoin que nous avons développé un CLI, sur base du framework Yargs (cf. section 5.1)²⁶.

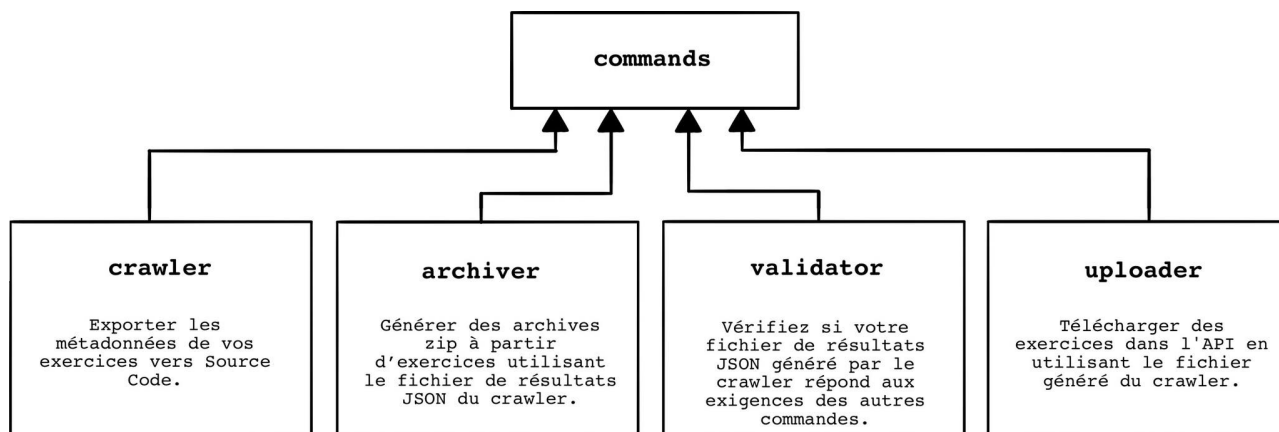


FIGURE 5.23 – CLI - inventaire des commandes disponibles

Chaque commande disposant naturellement de ses propres règles en matière d'arguments et/ou d'options. Fort heureusement, Yargs possède un éventail de fonctionnalités très diversifié, que nous allons illustrer un échantillon représentatif sur la commande "crawler" (cf. annexe D.2) :

- La possibilité d'indiquer si une option est nécessaire ou non (ligne 28)
- La possibilité de restreindre un choix uniquement à certaines valeurs (ligne 16)
- La possibilité d'identifier quels options et arguments ont été donnés à la commande (lignes 59-62)
- La possibilité de charger un fichier de configuration (lignes 39-40), permettant ainsi de ne pas devoir réécrire l'entièreté de la ligne de commande pour l'invoquer.

Veuillez noter que nous avons appliqué le "principe ouvert/fermé"²⁷ dans cette commande : en effet, il est possible d'utiliser des stratégies d'indexation déjà implémentées ou non (ex. "inginius-git" qui s'occupe de ressources informatiques au format INGINIOUS²⁸, disponibles via Git²⁹), tout en conservant le même comportement, quelle que soit la situation.

Par respect de la norme que nous avons créée (cf. annexe B), nous avons également adopté le **JSON** et les mêmes conventions d'écriture pour les fichiers d'entrée / de sortie des différentes commandes, à quelques exceptions près que nous avons détaillé dans le fichier README.MD.

²⁵En anglais, ce terme est connu sous le nom de "software lifecycle". (<https://web.maths.unsw.edu.au/~lafaye/CCM/genie-logiciel/cycle-de-vie.htm>)

²⁶Comme le montre le point d'entrée du CLI, à savoir "index.js" (que nous avons inclus dans l'annexe D.1), il sera aisé d'ajouter de nouvelles commandes à l'avenir.

²⁷La lettre **O** dans l'anagramme **SOLID** ([https://fr.wikipedia.org/wiki/SOLID_\(informatique\)](https://fr.wikipedia.org/wiki/SOLID_(informatique))), qui présente quelques principes pour la programmation orientée objet

²⁸<https://inginius.info.ucl.ac.be/>

²⁹<https://git-scm.com/>

5.4.2 Docker

Comme expliqué par son site officiel³⁰, il s'agit d'une plateforme de conteneurisation permettant de paqueter une application. Dans un conteneur sont contenus une application ainsi que tous les éléments dont elle a besoin pour fonctionner (code source, bibliothèques, ...). Le déploiement pouvant être réalisé sur n'importe quelle machine, tout en restant plus léger que des alternatives existantes³¹. Voici un récapitulatif des concepts clés :

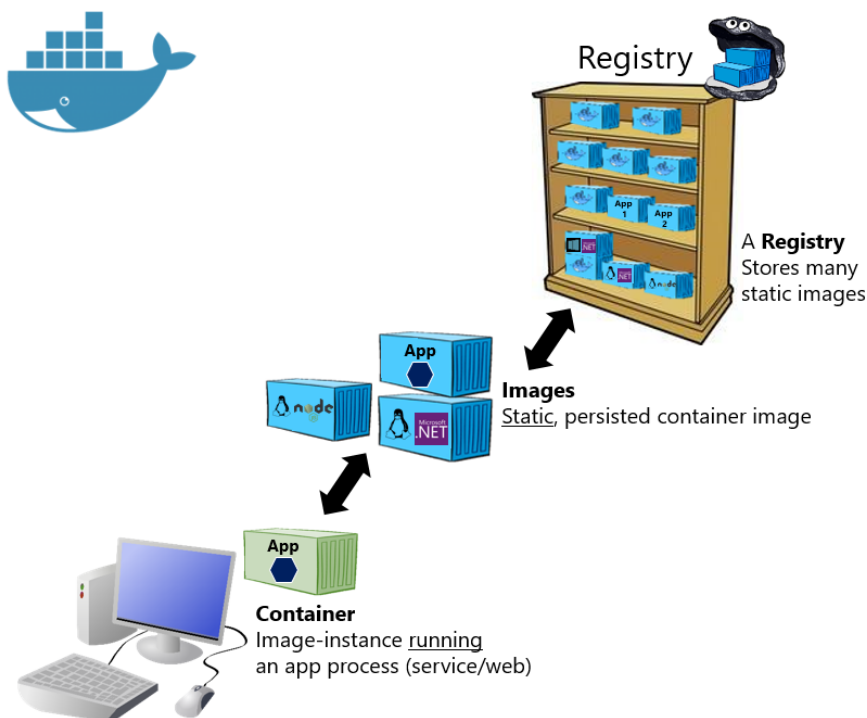


FIGURE 5.24 – Taxonomie des termes et concepts Docker³²

Nous avons ainsi découpé notre application en 3 images³³ :

- sourcecode-front : le frontend de Source Code
- sourcecode_api : le backend de Source Code
- sourcecode-postgres : une base de données en Postgresql, contenant des données extraites par le CLI, pour la démonstration de notre solution (cf. section 6)

jy95 / sourcecode-front Updated 14 days ago	☆ 0	📄 904	🌐 PUBLIC
jy95 / sourcecode_api Updated 14 days ago	☆ 0	📄 572	🌐 PUBLIC
jy95 / sourcecode-postgres Updated 17 days ago	☆ 0	📄 268	🌐 PUBLIC

FIGURE 5.25 – Docker Hub - des images prêtes à l'emploi

³⁰<https://www.docker.com/why-docker>

³¹Des explications sont disponibles sur <https://www.docker.com/resources/what-container>

³²<https://docs.microsoft.com/fr-fr/dotnet/architecture/microservices/container-docker-introduction/docker-containers-images-registries>

³³Elles sont disponibles sur Docker Hub (<https://hub.docker.com/>) et générées sur base de fichiers Dockerfile, présents dans chaque repository Github de code. Notons enfin la présence dans le repository miscellaneous de fichiers docker-compose-***.yaml pour déployer automatiquement l'entière de Source Code avec Docker Compose, en fonction de l'environnement souhaité (plus d'informations dans le README.MD).

5.4.3 Github Actions

Vous avez peut-être noté la présence d'un dossier `.github` dans chacun des repositories composant notre solution. Pour rappel (cf. section 5.3), ce dossier contient des scripts permettant d'automatiser un ensemble de tâches récurrentes, par le biais de Github Actions. Comme expliqué par Github³⁴, Github Actions est une fonctionnalité permettant d'automatiser nos workflows³⁵ directement sur base du code hébergé chez Github (dont notamment du CI/CD). Ceci nous permettant de maintenir constamment un niveau de qualité, via les quelques workflows que nous avons conçus :

- Créer une nouvelle image Docker, à chaque modification du code source (cf. section 5.4.2)
- Tester le bon fonctionnement de l'API, à chaque modification du code source, par le biais de tests (cf. chapitre 6)
- Vérifier la validité de la documentation en OAS, à chaque modification du code source
- Générer la documentation en **HTML** (cf. figure 5.21)
- Détecter et identifier les changements de l'API suggestibles d'impacter les clients l'utilisant, dont notamment le frontend
- Compiler les sources **LaTeX** du mémoire pour générer un **PDF**

feat: Allow configurations to have empty tag array #100

Merged jy95 merged 2 commits into master from configuration-rules-less-strict 14 days ago

Conversation 4 Commits 2 Checks 6 Files changed 4

✓ chore: Add tests and deal with every case c445bf6

▼ **OAS Documentation**
on: push

- ✓ validate-oas-spec
- ⊗ deploy-oas-spec

▼ **Source Code API Breaking Changes**
on: pull_request

- ✓ build-oas-artefacts
- ✓ build-report

▼ **Source Code CI/CD**
on: pull_request

- ✓ ci
- ✓ **cd**

Source Code CI/CD / cd
succeeded 16 days ago in 47s

- ▶ ✓ Set up job
- ▶ ✓ Run actions/checkout@v2
- ▶ ✓ Docker login
- ▶ ✓ Build
- ▶ ✓ Tags
- ▶ ✓ Push version
- ⊗ Push release (only on master)
- ▶ ✓ Post Run actions/checkout@v2
- ▶ ✓ Complete job

FIGURE 5.26 – Github Actions - exemple de workflows

³⁴<https://github.com/features/actions>

³⁵Si ce terme vous êtes inconnu, nous vous invitons à le découvrir sur <https://fr.wikipedia.org/wiki/Workflow>

Dans ce chapitre-ci, nous nous efforçons d'avoir un regard critique sur **Source Code**. Par la validation du projet (test avec des utilisateurs externes, test de la qualité du code) et les différentes métriques du projet, nous voulons nous assurer que notre prototype ait un sens et contribue utilement à la problématique des ressources informatiques.

6.1 Validation externe

Pour valider notre travail, nous avons initialement planifié deux séances de validation afin de récolter les avis d'un panel d'utilisateurs suffisamment large et diversifié. Pour cause de COVID-19, nous n'avons pu organiser qu'une seule séance via Teams avec les utilisateurs disponibles : la partie professorale du panel ne disposant généralement plus d'assez de disponibilités à nous accorder. Chaque personne se voyait attribuer deux rôles différents (visiteur-utilisateur ou visiteur-administrateur) afin de prendre connaissance des fonctionnalités les plus importantes de **Source Code**. Notre évaluation a pu être réalisée grâce à Google Forms (réponses, graphiques) et vous retrouverez l'intégralité de ce sondage dans l'annexe C.

Parmi les participants, six d'entre eux font partie de la faculté *EPL* de l'*UCLouvain*, dont quatre étudiants de master en sciences informatiques et deux doctorants. La septième personne est une professeure en section informatique à la *HELHa* de Mons.

Un des premiers objectifs de cette séance fut de faire comprendre le but de la plateforme, son utilité. Cela semblait être acquis par la majorité sauf pour une personne. Il n'est pas exclu de penser que cette dernière était biaisée par la plateforme *INGInious* de l'*UCLouvain*, car elle l'a prise comme point de référence pour juger **Source Code**.

Les sous-sections suivantes résument les critiques émises sur **Source Code**. Certaines remarques suggéraient des améliorations et corrections. Avec le temps qui nous restait, nous avons essayé de les prendre en compte pour rendre l'utilisation de la plateforme un peu plus immersive et plus fonctionnelle. Vous retrouverez alors un "(corrigé)" quand nous avons pu prendre en considération la remarque.

Bibliothèque

- Système de recherche intuitif, rapide et riche en filtres.
- La barre de recherche n'est pas bien placée, car elle laisse penser que c'est pour lancer une recherche sur tout le domaine du site web. **(corrigé)**
- Dans la création de favoris depuis le panneau, au lieu de cliquer sur OK, appuyer sur enter pour valider le nom du nouveau favori. **(corrigé)**
- Ajouter quelques tags + titre de recherche plutôt que juste le nom du favori dans la partie onglet "favoris". **(corrigé)**
- Recherche : Gestion des fautes de frappe et/ou recherche approximative. (1)
- Le texte de la barre de recherche est trop petit, et en bleu trop clair. **(corrigé)**

- Un peu difficile de scroller dans la liste des tags (jusqu'à 3 scrollbars). (**corrigé**)

Nos observations :

1. La gestion des fautes de frappe ou recherche approximative est une fonctionnalité qui pourrait être très utile pour la bibliothèque (le but étant de faciliter la recherche au maximum).

Favoris

- Pas nécessaire d'avoir obligatoirement un tag pour créer un favori. (**corrigé**)
- Quand on crée un favori puis le modifie, difficile de se souvenir de quelle catégorie appartient le tag qu'on a sélectionné précédemment : soit renommer le tag, soit créer un moyen de savoir de quelle catégorie le tag appartient. (1)

Nos observations :

1. Lors de l'édition d'un favori ou d'une ressource informatique, les mots-clés ajoutés sont affichés dans une section du formulaire en mode pêle-mêle (sans les catégories auxquels ils se rapportent). Le seul moyen de vérifier la catégorie à laquelle un mot-clé appartient est de naviguer dans le panneau à onglets. Nous discuterons des futures améliorations possibles dans le chapitre 7.

Création de ressources informatiques

- Frustrant de ne pas pouvoir créer un bloc de code directement depuis plusieurs lignes sélectionnées. (1)
- Ajouter un <tab> pour l'édition du code. (**corrigé**)
- Possibilité d'agrandir la boîte d'édition de texte. (2)
- Un peu difficile d'utiliser le panneau des tags (c'est une première) (3)

Nos observations :

1. Actuellement, il faut d'abord créer un bloc de code puis copier-coller le code que l'on veut formater dedans. La librairie d'édition de texte que nous utilisons (Tiptap) ne gère pas nativement ce qui a été soulevé dans la remarque. En revanche, cette même librairie est extensible et autorise la modification/l'ajout de fonctionnalités de manière aisée. Il est donc tout à fait envisageable de corriger le tir dans une prochaine mise à jour de l'application.
2. Certaines personnes se plaignaient que la boîte d'édition était trop petite (ou trop grande). Dans son état actuel, **Source Code** n'est pas responsive, et dans la perspective où elle le serait, ce problème sera certainement réglé pour tout écran.
3. Comme cela a déjà été soulevé dans la section concernant la bibliothèque, le panneau des mots-clés est un peu lourd à l'utilisation. Il y avait jusqu'à 3 scrollbars pour naviguer dans le panneau, ce qui diminuait considérablement l'accessibilité. Nous avons donc corrigé ce comportement en faisant en sorte d'afficher une seule scrollbar, avec un accès plus clair à la barre de recherche sous chaque catégories de mots-clés.

Esthétique et ergonomie

- Très chouette.
- Le bouton retour dans le header est un peu perturbant, car on ne le remarque pas avant d'avoir vraiment pris en main le site. (**corrigé**)
- Un dark mode (1)

Nos observations :

1. Le dark mode n'est pas une fonctionnalité nécessaire à la plateforme, mais elle pourrait ajouter un plus côté attractivité.

Points forts

- Ergonomique et riche en filtres de recherche.
- Très complète.
- Très utile, s'il y a une grande communauté et qu'il y a beaucoup d'exercices. (1)
- La sélection dynamique est top (pour la recherche).
- Design, visuel, fluidité.
- L'idée de rassembler toutes les ressources informatiques au même endroit semble pertinente et utile.
- Le système de tags (utilisé correctement par les utilisateurs) semble flexible et potentiellement très utile.
- Visuellement attrayant.

Nos observations :

1. La grande communauté dépendra surtout de la qualité de l'application et de son utilité. Les exercices que nous avons utilisés pour la séance de validation étaient importés à partir de quatre cours hébergés sur la plateforme *INGInious*. Nous pouvons donc aisément fournir plus de contenu rien qu'en important tous les cours stockés sur cette même plateforme avec l'avantage d'un système de mots-clés plus avancé.

Points faibles

- S'il y a peu d'exercices, la plateforme aura peu de sens. (1)
- S'il y a trop d'exercices, les administrateurs pourraient peut-être demeurer débordés pour la modération. (2)
- L'application est très complète, mais le souci vient justement de cela : la courbe d'apprentissage est assez pentue, mais après cela, l'outil deviendrait très puissant (difficulté pour les professeurs un peu moins "tech savvy"). (3)

Nos observations :

1. **Source Code** pourra très rapidement posséder une riche bibliothèque de ressources informatiques rien qu'avec les exercices stockés sur la plateforme *INGInious*. Avec la promotion de la plateforme auprès d'autres institutions scolaires, **Source Code** pourrait alors prétendre à une popularité certaine dans le monde des ressources partagées. Il s'agit alors de savoir vendre son produit avec les bons arguments...
2. Nous n'avions pas pensé à cela pour le prototype. Nous voulions d'abord créer un système cohérent et fonctionnel au niveau de la gestion des ressources informatiques et mots-clés. Les différents statuts attribuables à ces mêmes ressources informatiques et mots-clés jouent déjà un rôle majeur dans la gestion (ex. trier les ressources informatiques non valides ou en attente de validation ...). En termes d'améliorations, nous pourrions automatiser le processus de validation afin que les administrateurs ne soient pas débordés (cf. chapitre 7).
3. C'est un point intéressant pour nous. **Source Code** est une application qui tente de faciliter le plus possible la recherche et la gestion de ressources informatiques (système de filtres, historique, favoris ...). Malheureusement, ajouter une pléthore de fonctionnalités peut aussi devenir une barrière à l'utilisation, car l'utilisateur doit d'abord apprendre à les maîtriser. Une des premières solutions que nous avons mises en place était la création d'une section tutoriel sur la plateforme, mais cela n'est qu'une solution de contournement. Une autre idée serait de prévoir une interface en fonction du public ciblé (débutant, "tech savvy" ...).

Que pouvons-nous faire pour améliorer l'application ?

- Il s'agit d'un produit bien abouti ! Vous êtes à un stade où ça sera les utilisateurs de la plateforme qui vous feront des feedbacks sur ce qui serait bien à ajouter/améliorer.
- Clarifier l'interface. L'UI représente cependant beaucoup de travail. Vous avez priorisé les bonnes choses.
- Améliorer l'agencement des menus. (1)
- Une fonction mot de passe oublié pour les utilisateurs. (2)

Nos observations :

1. Dans les retours d'utilisation, les testeurs étaient perturbés avec le menu principal et le panneau latéral. La confusion venait du fait que le menu principal comportait des chevrons, ce qui laissait penser à une structure à plusieurs niveaux (listes). Le panneau de filtres contenait aussi ces mêmes chevrons, alors nous avons supprimé ces derniers sur le menu principal pour enlever cette confusion.
2. C'est une fonctionnalité importante pour ce genre d'application, mais nous n'avons pas pris le temps de l'intégrer. Nous l'ajoutons dans le chapitre 7.

Que manquerait-il pour que cette plateforme ne soit plus un prototype en termes de fonctionnalités ?

- Je ne vous rejoins pas sur l'idée que **Source Code** en est au stade de "prototype". Ça ressemble bien plus à un produit fini, en une première version, qui peut être concernée par d'éventuelles mises à jour par la suite.
- Pas grand-chose, et beaucoup à la fois. Il s'agit du lent travail de lisser les bugs et les surprises que peuvent rencontrer les utilisateurs.

- Peut-être la complétion automatique dans la barre de recherche
- Pour moi c'est plus qu'un prototype au stade où elle en est. S'il fallait vraiment une fonctionnalité en plus, ça serait de pouvoir ajouter ses solutions sur la plateforme. (1)
- Une manière d'interagir entre les étudiants et les "créateurs d'exercices" ? Histoire de pouvoir au moins poser des questions et ne pas être aussi détachés. (2)

Nos observations :

1. **Source Code** n'est pas prévu pour résoudre des exercices depuis la plateforme. À ce stade-ci, on peut la considérer comme un référencier de ressources informatiques qui faciliterait la recherche de ressources particulière avec un système de recherche complet. Dans une perspective future, **Source Code** pourrait avoir son propre correcteur d'exercices si la ressource nécessite une résolution. Dans ce cas, cela profiterait aux étudiants pour s'évaluer.
2. Certaines plateformes comme Hackerrank ou Coderbyte ont prévu une section forum pour chaque exercice/challenge. C'est une fonctionnalité intéressante, mais elle va de pair avec le point précédent.

Pensez-vous que cette application ait un réel impact dans le monde des ressources partagées ?

- Oui (de la part de la majorité)
- Il faudrait juste voir comment gérer l'accès entre votre plateforme et celles qui permettent la correction des exercices (Inginious). Par exemple un moyen de linker votre plateforme avec les autres, pour qu'en un clic on arrive directement sur l'exercice inginius en étant déjà connecté dessus.
- Sans être intégrée à une plateforme automatisée de correction de code, et sans une gestion de parcours et de lien entre les exercices, sans doute pas beaucoup. Elle sera limitée à une utilisation par les enseignants pour créer leurs propres cours. (1)

Nos observations :

1. La plateforme est d'abord créée pour l'équipe pédagogique, prônant le partage de ressources informatiques dans une bibliothèque triée par mots-clés. Une prochaine étape serait alors d'intégrer un système de correction d'exercices directement depuis la plateforme. Nous n'avons pas eu le temps d'intégrer une telle fonctionnalité, mais cela pourrait définitivement être utile aux étudiants.

Utiliseriez-vous cette plateforme ? Quelle serait votre activité principale ?

- Je pense que pour des professeurs, cela peut être très utile pour trouver et donner des exercices à des étudiants (ou d'avoir des idées).
- Trouver des idées d'exercices pour compléter mes cours.
- Je pense que ça s'appliquerait surtout au domaine éducatif.
- Oui, en tant que professeur/enseignant, pourvu que les solutions soient mises à dispositions.

Conclusion

Cette séance de validation a globalement été positive et instructive. Le message d’une plateforme de partage de ressources informatiques à destination d’une équipe pédagogique semble être acquis par la majorité. Nous avons appris que la plateforme dispose d’un design et d’une ergonomie satisfaisants. Il faut néanmoins y apporter quelques mises à jour pour rendre le projet mature, notamment au niveau de l’ergonomie. Certaines des fonctionnalités mentionnées dans les critiques ont pu être intégrées à **Source Code** pour l’améliorer le plus possible. Pour celles qui n’ont pas été ajoutées, nous les avons référencées dans le chapitre 7.

6.2 Métriques

Cette section traduit notre implication sur ce projet. Pour ce faire, nous avons repris des statistiques des répertoires GitHub du frontend et de l’API.

6.2.1 Statistiques

- Nombre de lignes de code (frontend + backend) : $21685 + 10158 = 31843$
 - Le nombre de lignes de code a été calculé en retirant tous les fichiers générés automatiquement (ex : npm install qui va créer un fichier *package-lock.json*).
- Nombre de pull requests résolues (frontend + backend) : $49 + 79 = 128$
 - Nous avons beaucoup utilisé les pull requests pour garder une trace des mises à jour importantes effectuées. Cela a été extrêmement utile avec l’intégration continue en *Docker* pour tester rapidement l’application avec l’une d’elles (cf. section 5.4.2).
- Nombre de tests fonctionnels (backend) : 48
 - Plutôt que de tester individuellement chaque partie de l’application (comme le feraient des tests unitaires¹), nous avons choisi cette approche, de type boîte noire (c.-à-d. sans connaître le code source), qui consiste à vérifier si les spécifications (cf. section 4.1) sont respectées par le logiciel ainsi développé. Cette vérification est notamment très utile en CI/CD (cf. figure 5.26).
- Couverture du code (backend) : 100%
 - Il s’agit de mesurer la proportion du code source exécutée lors de tests² : nous considérons ici la définition générale, car celle-ci peut se décliner sous d’autres formes³. Veuillez noter que nous avons exclu du calcul quelques lignes difficiles, voire impossibles à tester afin d’avoir un taux plus réaliste.
- Note Codacy de qualité du code (backend / frontend) : A / B
 - Il s’agit d’une plateforme qui analyse le code source pour produire diverses mesures (pourcentage de code dupliqué, complexité du code ...). Elle vérifie / analyse également la lisibilité du code en fonction de conventions de codage courantes⁴.

6.2.2 Graphe d’activité

Source Code a connu un développement plus ou moins constant durant l’année académique 2019-2020. Étant donné que le frontend et le backend sont intimement liés, vous pourrez constater sur le graphe 6.1 et 6.2 que la forte activité de l’un compense l’activité au ralenti de l’autre répertoire.

¹https://fr.wikipedia.org/wiki/Test_unitaire

²Vous pouvez consulter les résultats depuis https://codecov.io/gh/SourceCodeOER/sourcecode_api/branch/master

³https://fr.wikipedia.org/wiki/Couverture_de_code, [3], ...

⁴Vous pouvez consulter ces rapports sur https://app.codacy.com/gh/SourceCodeOER/sourcecode_api/dashboard et <https://app.codacy.com/gh/SourceCodeOER/sourcecode-front/dashboard>

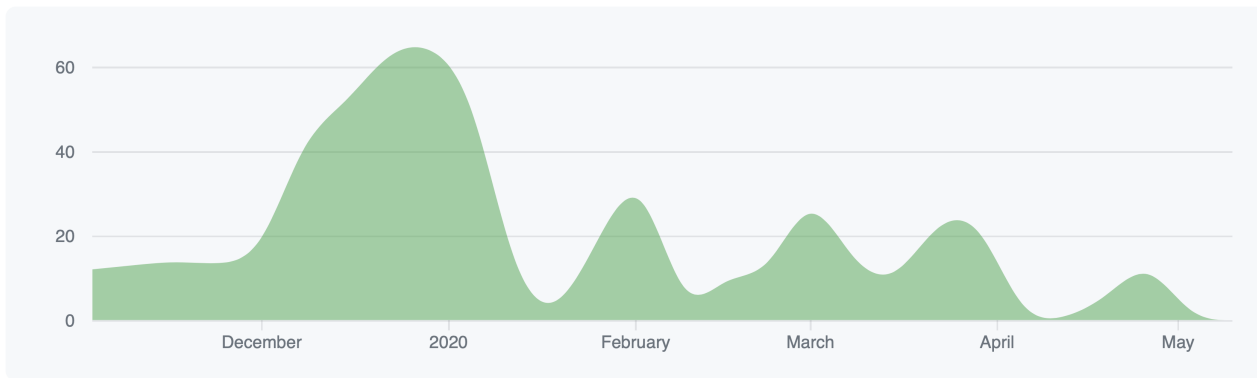


FIGURE 6.1 – GitHub : graphe d’activité (frontend)

Les fonctionnalités les plus importantes comme la recherche dans la bibliothèque, la consultation d’une fiche et la gestion des ressources informatiques ont été intégrées durant la période d’octobre à janvier, après quoi nous avons amélioré l’expérience d’utilisation de l’application en appliquant quelques mises à jour et corrections de bugs jusqu’au mois de mai.

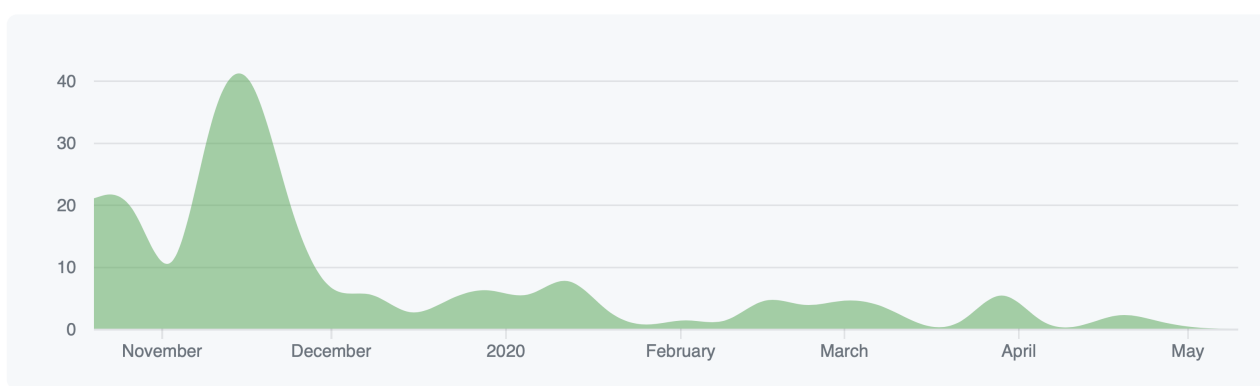


FIGURE 6.2 – GitHub : graphe d’activité (backend)

Globalement, nous avons respecté le planning 3.3 que nous nous sommes imposé. Nous avons cependant effectué une dernière mise à jour prenant en compte un maximum des remarques relatives en section 6.1, car nous voulions prouver que **Source Code** est capable de s’adapter aux besoins des utilisateurs.

6.3 Conclusion

À travers ce chapitre, nous avons pu constater si ce projet tient la route à son stade actuel.

La séance de validation nous a été extrêmement bénéfique, car les remarques étaient constructives. Cela nous conforte dans l’idée que notre plateforme a des chances de devenir mature et utilisable par une communauté. Quoi qu’il en soit, notre projet est *Open Source* et pourra donc toujours évoluer avec sa communauté. C’est d’ailleurs notre plus grand souhait pour cette plateforme. La qualité technique du code est assurée par les différentes métriques énoncées précédemment (cf. section 6.2).

Au niveau du planning, il semble que nous ayons respecté ce qui était initialement mis en place au vu de nos graphes d’activité. Nous avons tenté d’être constants tout au long du projet, car ce dernier demande beaucoup d’investissement en matière de recherche et de développement. Il semblerait que cela ait porté ses fruits.

Ce chapitre tente de rassembler tous les futurs travaux qui pourront être apportés à **Source Code**.

Durant notre année académique, nos promoteurs ont gardé un œil sur le projet afin de nous proposer quelques pistes pour améliorer l'application. Parallèlement, nous avons pu rencontrer *Christine Jacqmot*, membre du Louvain Learning Lab, qui nous a partagé son expérience des OER avec quelques conseils pour notre plateforme. Grâce à ces précieux avis, à notre prise de recul et aux remarques récoltées lors de la séance de validation (cf. 6) avec des utilisateurs, nous avons dressé une liste des améliorations possibles pour **Source Code**.

7.1 Liste des améliorations

- Améliorer le système de validation des ressources informatiques en ajoutant un système de tickets (issues) comme GitHub. Cela permettra au créateur de la ressource de communiquer avec les administrateurs en sachant ce qui ne va pas avec sa fiche.
- Faciliter la modération de ressources informatiques en automatisant la recherche de ressources similaires à d'autres ou non pertinentes (contenu injurieux, mal écrit,...).
- Ajouter des métriques supplémentaires pour les ressources informatiques :
 1. Nombre de fois où la ressource informatique a été consultée.
 2. Nombre de fois où l'archive zip a été téléchargée.
- Prévoir l'intégration avec de multiples plateformes dont Ingenious, Google Scholar pour améliorer le référencement des ressources informatiques.
- Ajouter le multi-upload de fichier dans le formulaire de création/modification d'une ressource informatique.
- Ajouter le preview de fichier quand c'est possible (image, pdf, word,...) dans le formulaire de création/modification de ressources informatiques.
- Avoir la possibilité de rendre certains favoris publics afin de les partager avec les autres et les sauvegarder dans ses propres favoris.
- Ajouter la compatibilité avec les normes Dublin Core et LOM (cf. annexe B, pratique pour l'export sous un certain format et l'import capable de convertir les formats). Cela permettra à **Source Code** de s'ouvrir à d'autres plateformes qui utilisent d'autres normes.
- Gestion de la recherche approximative dans la bibliothèque avec propositions de recherche.
- Améliorer la lisibilité des mots-clés dans le formulaire de création de ressources informatiques ou de favoris. Il faut pouvoir distinguer à quelle catégorie de mots-clés un mot-clé appartient, sans devoir rechercher dans le panneau de filtres.
- Améliorer l'éditeur de texte pour la création/modification d'une ressource informatique.

1. Créer un bloc de code directement en sélectionnant plusieurs lignes au lieu de créer un bloc puis coller le code dedans.
 2. Agrandir la boîte d'édition de texte en fonction de l'écran.
- Ajouter un dark mode.
 - Ajouter une fonction de mot de passe oublié.
 - Ajouter d'autres possibilités d'authentification (par exemple réseaux sociaux ou autres).
 - Pouvoir résoudre certains exercices directement depuis la plateforme avec l'automatisation de la correction du code.
 - Créer un espace de discussion autour d'une ressource informatique. Cela peut être utile pour poser des questions sur la résolution d'un exercice.
 - Ajouter un support complet du responsive sur la plateforme (cf. section 4.3)
 - Ajouter une recherche avancée dans la bibliothèque avec des conditions (NOT, AND, OR). Cela a déjà été intégré côté backend, il reste donc à l'ajouter dans le frontend.
 - Pouvoir modifier son profil et consulter certaines statistiques comme :
 1. La moyenne totale des ressources informatiques créées
 2. Le nombre de ressources informatiques créées
 - Intégrer GraphQL à l'API pour permettre de charger les données que l'on souhaite, sans forcément multiplier les endpoints en REST.
 - Mettre en place des limites d'utilisation sur l'API (ex. un utilisateur ne pourrait pas importer plus que n fiches par quart d'heure).

Nous avons pris plaisir à développer cette plateforme web pour une cause qui nous concerne dans le domaine informatique : le partage, comme pour les projets open source. Durant l'année académique, le temps fut notre ennemi, mais nous avons fait en sorte de fournir une plateforme fonctionnelle, à défaut qu'elle ne soit pas totalement complète.

La liste évoquée plus haut peut donc être considérée comme un prochain roadmap de développement. Il y a encore pas mal à réaliser, mais nous insistons sur le fait que ce projet est open source et utilise des technologies connues (cf. 5.1) afin que ça ne devienne pas une barrière au développement.

Ce mémoire a pour but principal d'apporter une solution à la problématique des ressources informatiques par le biais d'une plateforme web devant réaliser des objectifs ambitieux, notamment en matière d'expérience utilisateur et de référencement, avec une durée de développement restreinte pour un projet de cette ampleur.

En dépit de sa jeunesse, notre solution Open Source semble avoir rencontré ces critères, car nous avons pu la faire valider autant par un panel diversifié d'utilisateurs qu'au niveau de sa réalisation technique à proprement parler (tests, métriques sur la qualité du code ...). En outre, des processus automatisés (cf. section 5.4.3) permettent notamment le déploiement aisé et rapide d'un logiciel fonctionnel : il n'est donc pas exclu d'espérer une reprise ultérieure de Source Code ou à défaut la création d'une nouvelle solution inspirée de la nôtre.

Sur le niveau personnel, ce projet a été une source intarissable d'expériences. Celui-ci a été le plus conséquent que nous ayons pu connaître lors de notre parcours universitaire et nous avons beaucoup appris sur l'organisation du travail et sur l'utilisation de nouvelles technologies et outils de développement. Nous restons cependant conscients que notre solution reste perfectible et critiquable dans son état actuel (cf. 6.1 / 7), ce pour quoi nous avons réalisé un roadmap de développement prenant en compte les diverses remarques et potentielles fonctionnalités.

Finalement, c'est avec un certain sentiment du devoir accompli, mais aussi d'un désir inassouvi d'améliorations possibles à tout niveau que nous concluons ce présent mémoire.

- [1] Kent BECK et al. *Manifeste pour le développement Agile de logiciels*. URL : <https://agilemanifesto.org/iso/fr/manifesto.html>.
- [2] CENTRALESUPÉLEC. *Maîtrisez les bases de données NoSQL*. URL : <https://openclassrooms.com/fr/courses/4462426-maitrisez-les-bases-de-donnees-nosql/4462471-maitrisez-le-theoreme-de-cap>.
- [3] R. AL-EKRAM et Kostas KONTOGIANNIS. "Source code modularization using lattice of concept slices". Dans : avr. 2004, p. 195-203. ISBN : 0-7695-2107-X. DOI : 10.1109/CSMR.2004.1281420.
- [4] Anne-Sophie LAUGIER et Louis-Baptiste FRANCE. *Technologies Business Intelligence et technologies Big Data, quelles différences*. URL : <https://www.illustradata.com/technologies-bi-et-big-data-differences/>.
- [5] NEBRA MATHIEU. *Node.js : mais à quoi ça sert ? - Des applications ultra-rapides avec Node.js*. URL : <https://openclassrooms.com/en/courses/1056721-des-applications-ultra-rapides-avec-node-js/1056866-node-js-mais-a-quoi-ca-sert>.
- [6] DELMÉE PATRICK. Dans : *PROF* (6 déc. 2019). URL : <http://www.enseignement.be/index.php?page=27203&id=2922>.
- [7] UNESCO. *Ressources éducatives libres*. URL : <https://fr.unesco.org/themes/tic-education/rel>.
- [8] WIKIPEDIA. *Bibliothèque logicielle*. URL : https://fr.wikipedia.org/wiki/Biblioth%C3%A8que_logicielle.
- [9] WIKIPEDIA. *Méthode MoSCoW*. URL : https://fr.wikipedia.org/wiki/M%C3%A9thode_MoSCoW.
- [10] WIKIPEDIA. *Variable d'environnement*. URL : https://fr.wikipedia.org/wiki/Variable_d%27environnement.

Annexes

A.1 Practice-it



[Main Page](#) → **Problems**

Click a category below to view its available problems.

<u>Building Java Programs, 5th edition</u>	(637)
<u>Building Java Programs, 4th edition</u>	(621)
<u>Building Java Programs, 3rd edition</u>	(617)
<u>University of Washington CSE 142 (CS1)</u>	(336)
<i>Problems used in the CS1 course at Washington, an objects-late introduction to Java.</i>	
CS1 Sections	(56)
<i>Problems solved during our weekly discussion sessions led by TAs.</i>	
Section 1 (printing, methods)	(5)
Mantra	
DoubleSlash	
Difference	
Confusing	
StarFigures	
Section 2 (expressions, for loops, constants)	(8)
Section 3 (parameters, graphics)	(7)
Section 4 (ifelse, return, Scanner)	(6)
Section 5 (while, Random, boolean)	(8)
Section 6 (file processing)	(5)
Section 7 (arrays)	(11)
Section 8 (classes and objects)	(2)
TimeSpan	
Circle	
Section 9 (Critters)	(4)
CS1 Labs	(59)
CS1 Exams	(221)
<u>University of Washington CSE 143 (CS2)</u>	(427)
<u>University of Washington CSE 373</u>	(53)
<u>JavaScript</u>	(84)

FIGURE A.1 – Page principale pour rechercher un problème

TimeSpan

Language/Type:  Java [classes](#) [constructors](#) [fields](#) [implementing](#) [instance methods](#)
Author: Marty Stepp

Define a class named `TimeSpan`. A `TimeSpan` object stores a span of time in hours and minutes (for example, the time span between 8:00am and 10:30am is 2 hours, 30 minutes). Each `TimeSpan` object should have the following public methods:

- `TimeSpan(hours, minutes)`
Constructs a `TimeSpan` object storing the given time span of hours and minutes.
- `getHours()`
Returns the number of hours in this time span.
- `getMinutes()`
Returns the number of minutes in this time span, between 0 and 59.
- `add(hours, minutes)`
Adds the given amount of time to the span. For example, (2 hours, 15 min) + (1 hour, 45 min) = (4 hours). Assume that the parameters are valid: the hours are non-negative, and the minutes are between 0 and 59.
- `add(timespan)`
Adds the given amount of time (stored as a time span) to the current time span.
- `getTotalHours()`
Returns the total time in this time span as the real number of hours, such as 9.75 for (9 hours, 45 min).
- `toString()`
Returns a string representation of the time span of hours and minutes, such as "28h46m".

The minutes should always be reported as being in the range of 0 to 59. That means that you may have to "carry" 60 minutes into a full hour.

Type your solution here:

solution code goes here

FIGURE A.2 – Page d'un problème



Search Problems by Keyword:

(choose a keyword)  (all languages) 

Search Problems by Substring:

substring to search for (all languages) 

Is there a problem? [Contact a site administrator.](#)

FIGURE A.3 – Recherche avancée

A.2 Hackerrank

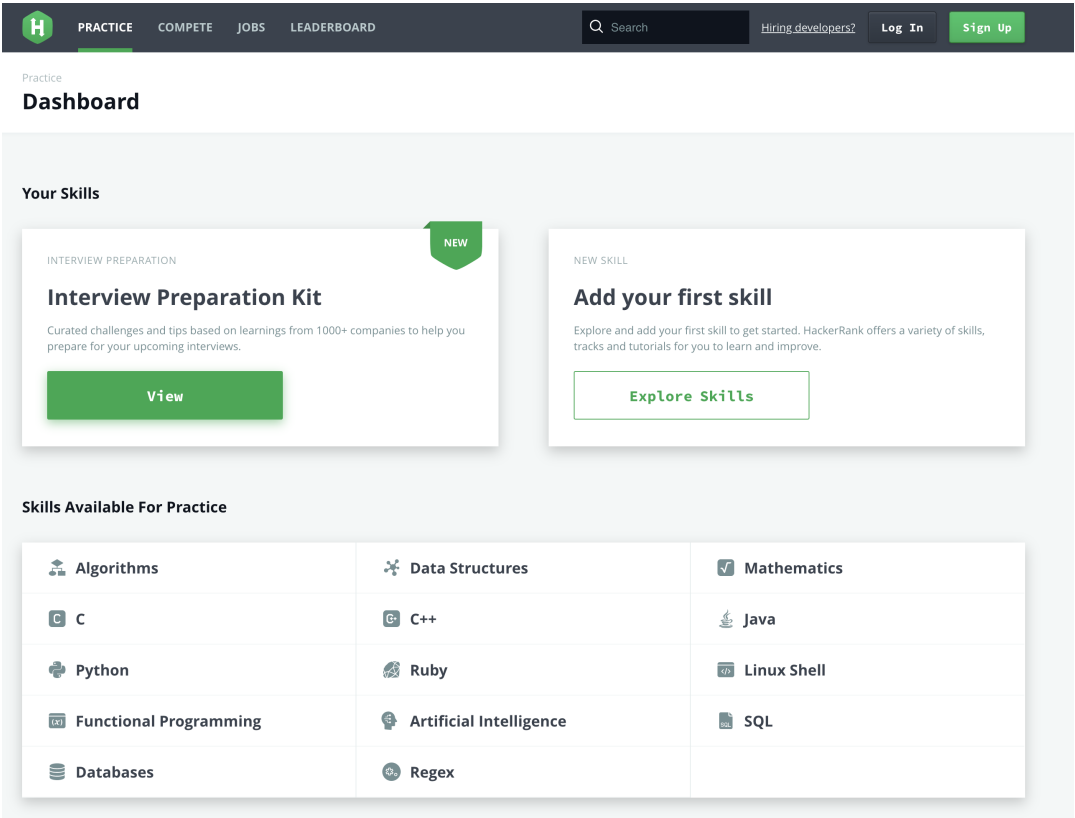


FIGURE A.4 – Tableau de bord

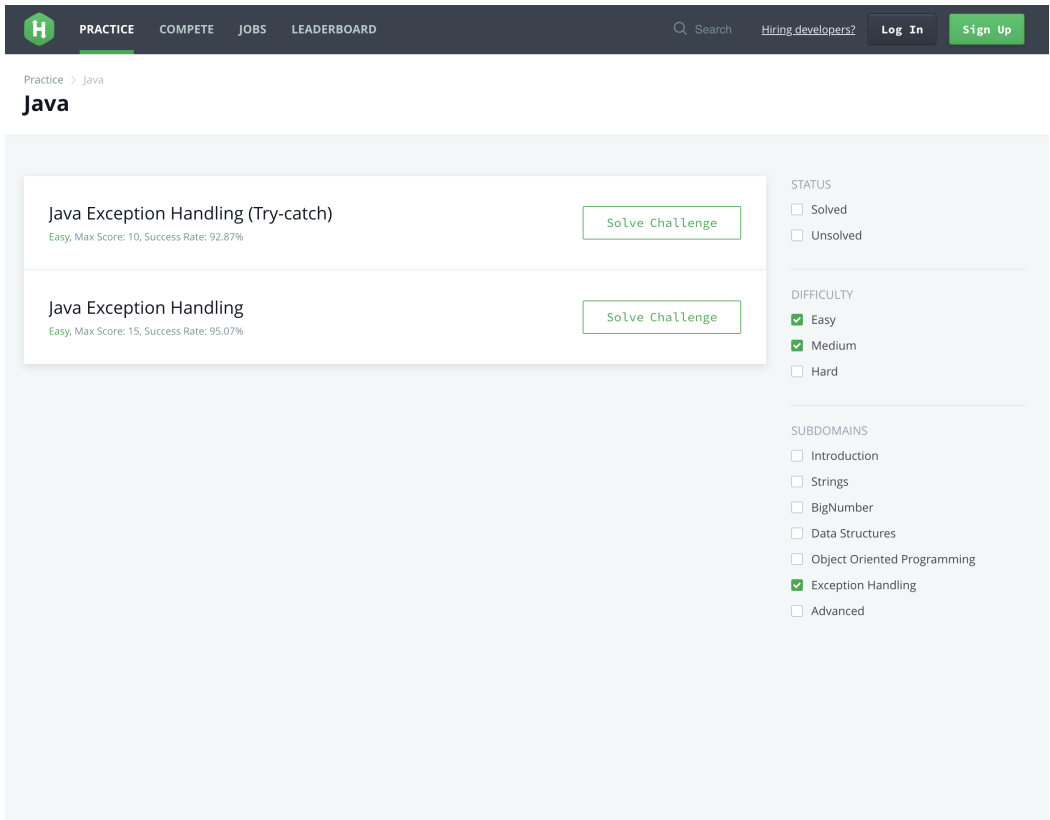


FIGURE A.5 – Page pour rechercher un problème

Practice > Java > Exception Handling > Java Exception Handling (Try-catch)

Java Exception Handling (Try-catch)

Problem

[Submissions](#)[Leaderboard](#)[Discussions](#)[Editorial](#)

Exception handling is the process of responding to the occurrence, during computation, of exceptions – anomalous or exceptional conditions requiring special processing – often changing the normal flow of program execution. (Wikipedia)

Java has built-in mechanism to handle exceptions. Using the try statement we can test a block of code for errors. The catch block contains the code that says what to do if exception occurs.

This problem will test your knowledge on try-catch block.

You will be given two integers x and y as input, you have to compute x/y . If x and y are not 32 bit signed integers or if y is zero, exception will occur and you have to report it. Read sample Input/Output to know what to report in case of exceptions.

Sample Input 0:

```
10
3
```

Sample Output 0:

```
3
```

Sample Input 1:

```
10
Hello
```

Sample Output 1:

```
java.util.InputMismatchException
```

Sample Input 2:

Author [Shafaet](#)
Difficulty [Easy](#)
Max Score 10
Submitted By 50901

NEED HELP?

[View discussions](#)
[View editorial](#)
[View top submissions](#)

RATE THIS CHALLENGE

☆☆☆☆

MORE DETAILS

[Download problem statement](#)
[Download sample test cases](#)
[Suggest Edits](#)

[f](#) [t](#) [in](#)

FIGURE A.6 – Page d'un challenge

A.3 Leetcode

LeetCode

Explore Day 14 Problems Mock Contest Articles Discuss Store

Premium

Announcement

Due to some upstream issue, users won't receive any email notifications from the discuss posts. Sorry for the inconvenience!

Category - All

Algorithms Database Shell Concurrency New

0/1411 Solved - Easy 0 Medium 0 Hard 0

Pick One

Search question titles, description or IDs

Difficulty Status Lists Tags

#	Title	Solution	Acceptance	Difficulty	Frequency
1	Two Sum		45.2%	Easy	
2	Add Two Numbers		33.1%	Medium	
3	Longest Substring Without Repeating Characters		29.8%	Medium	
4	Median of Two Sorted Arrays		28.7%	Hard	
5	Longest Palindromic Substring		29.0%	Medium	
6	ZigZag Conversion		35.2%	Medium	
7	Reverse Integer		25.6%	Easy	
8	String to Integer (atoi)		15.1%	Medium	
9	Palindrome Number		47.2%	Easy	
10	Regular Expression Matching		26.4%	Hard	
11	Container With Most Water		49.5%	Medium	
12	Integer to Roman		54.0%	Medium	
13	Roman to Integer		54.9%	Easy	
14	Longest Common Prefix		34.9%	Easy	

Daily Coding Challenge

30-Day LeetCode Challenge Day 14

Weekly Contest 185 Sunday, April 19th 4:30am-6:00am local time Register

New Biweekly Contest 14:30 UTC every other Saturday Starting June 1, 2019 Register

Your Progress

Session: Anonymous Session

FIGURE A.7 – Page de recherche de problèmes

28	Implement strStr()		34.0%	Easy	
29	Divide Two Integers		16.2%	Medium	
30	Substring with Concatenation of All Words		24.9%	Hard	
31	Next Permutation		32.0%	Medium	
32	Longest Valid Parentheses		27.6%	Hard	
33	Search in Rotated Sorted Array		33.7%	Medium	
34	Find First and Last Position of Element in Sorted Array		35.4%	Medium	
35	Search Insert Position		41.6%	Easy	
36	Valid Sudoku		47.3%	Medium	
37	Sudoku Solver		41.6%	Hard	
38	Count and Say		43.5%	Easy	
39	Combination Sum		54.1%	Medium	
40	Combination Sum II		46.3%	Medium	
41	First Missing Positive		31.1%	Hard	
42	Trapping Rain Water		47.3%	Hard	
43	Multiply Strings		33.0%	Medium	
44	Wildcard Matching		24.2%	Hard	
45	Jump Game II		29.9%	Hard	
46	Permutations		61.2%	Medium	
47	Permutations II		44.8%	Medium	
48	Rotate Image		54.3%	Medium	
49	Group Anagrams		55.1%	Medium	
50	Pow(x, n)		29.2%	Medium	

50 rows per page.

12345>

Topics

Array	237	Dynamic Programming	194
Math	172	String	166
Tree	131	Hash Table	123
Depth-first Search	121	Binary Search	84
Greedy	78	Breadth-first Search	67
Two Pointers	60	Stack	56
Backtracking	53	Design	48
Sort	48	Bit Manipulation	45
Graph	41	Linked List	38
Heap	33		
Show More			

Companies

Google	795	Amazon	695
Facebook	487	Microsoft	457
Apple	307	Uber	306
Bloomberg	294	Oracle	203
Adobe	199	LinkedIn	145
Yahoo	109	Goldman Sachs	97
Snapshot	88	Twitter	83
VMware	63	Walmart Labs	77
eBay	73	Salesforce	68
Airbnb	64	ByteDance	61
Cisco	54	Lyft	54
Expedia	46	Alibaba	45
Paypal	44	Atlassian	42
Yandex	42	Mathworks	39
Pinterest	39	Visa	39
Yelp	39	Zillow	38
Baidu	38	Nutanix	36
SAP	33	Citadel	30
Quora	30	Tencent	30
Wish	27	Intuit	25
Qualtrics	25	Wayfair	24
Dropbox	23	Indeed	23
Zenefits	23	Nvidia	22
Samsung	22	DoorDash	21
Palantir Technologies	21		

FIGURE A.8 – Quelques mots-clés disponibles sur le côté droit

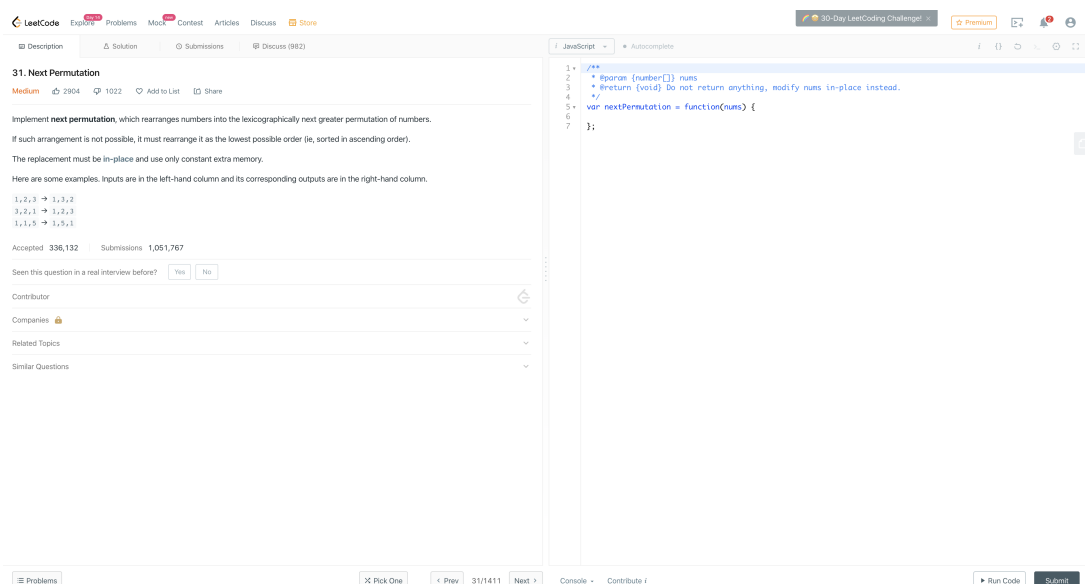


FIGURE A.9 – Page d'un problème

A.4 Codeforces

 **CODE FORCES**
Sponsored by Telegram

Enter | Register

HOME TOP CONTESTS GYM **PROBLEMSET** GROUPS RATING API CALENDAR HELP 10 YEARS! 🎉

MAIN ACMGURU | **PROBLEMS** SUBMIT STATUS STANDINGS CUSTOM TEST

Problems

#	Name		🚀	🏆
1339B	Sorted Adjacent Differences	constructive algorithms, sortings	🚀	1200 x11498
1339A	Filling Diamonds	brute force, dp, implementation, math	🚀	1000 x13824
1338E	JYPnaton	graphs	🚀	3500 x34
1338D	Nested Rubber Bands	constructive algorithms, dfs and similar, dp, math, trees	🚀	2700 x299
1338C	Perfect Triples	bitmasks, brute force, constructive algorithms, divide and conquer, math	🚀	2200 x1634
1338B	Edge Weight Assignment	bitmasks, constructive algorithms, dfs and similar, greedy, math, trees	🚀	1800 x4191
1338A	Powered Addition	greedy, math	🚀	1400 x10127
1335F	Robots on a Grid	dfs and similar, dp, dsu, graphs, greedy, trees	🚀	2600 x758
1335E2	Three Blocks Palindrome (hard version)	brute force, data structures, dp, two pointers	🚀	2000 x3324
1335E1	Three Blocks Palindrome (easy version)	binary search, brute force, data structures, dp, two pointers	🚀	1700 x4859
1335D	Anti-Sudoku	constructive algorithms, implementation	🚀	1300 x12244
1335C	Two Teams Composing	binary search, greedy, implementation, sortings	🚀	1100 x14499
1335B	Construct the String	constructive algorithms	🚀	1000 x16299
1335A	Candies and Two Sisters	math	🚀	800 x20550
1334G	Substring Search	bitmasks, brute force, fft	🚀	2900 x136
1334F	Strange Function	binary search, data structures, dp, greedy	🚀	2600 x407
1334E	Divisor Paths	combinatorics, graphs, greedy, math, number theory	🚀	2200 x1276
1334D	Minimum Euler Cycle	constructive algorithms, graphs, greedy, implementation	🚀	1800 x4034
1334C	Circle of Monsters	brute force, constructive algorithms, greedy, math	🚀	1600 x8222
1334B	Middle Class	greedy, sortings	🚀	1100 x13333
1334A	Level Statistics	implementation, math	🚀	1200 x13191

→ Pay attention

Before contest
Codeforces Round #635 (Div. 1)
07:48:44
[Register now »](#)
*has extra registration?

Before contest
Codeforces Round #635 (Div. 2)
07:48:43
[Register now »](#)
*has extra registration?

 **Like** 212 people like this. Be the first of your friends.

→ Filter Problems

Difficulty: —
[Add tag](#)

FIGURE A.10 – Page de recherche de problèmes

B. Sorted Adjacent Differences

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

You have array of n numbers $a_1, a_2, \dots, a_n$.

Rearrange these numbers to satisfy $|a_1 - a_2| \leq |a_2 - a_3| \leq \dots \leq |a_{n-1} - a_n|$, where $|x|$ denotes absolute value of x . It's always possible to find such rearrangement.

Note that all numbers in a are not necessarily different. In other words, some numbers of a may be same.

You have to answer independent t test cases.

Input

The first line contains a single integer t ($1 \leq t \leq 10^4$) — the number of test cases.

The first line of each test case contains single integer n ($3 \leq n \leq 10^5$) — the length of array a . It is guaranteed that the sum of values of n over all test cases in the input does not exceed 10^5 .

The second line of each test case contains n integers $a_1, a_2, \dots, a_n$ ($-10^9 \leq a_i \leq 10^9$).

Output

For each test case, print the rearranged version of array a which satisfies given condition. If there are multiple valid rearrangements, print any of them.

Example

input	Copy
2	
6	
5 -2 4 8 6 5	
4	
8 1 4 2	
output	Copy
5 5 4 6 8 -2	
1 2 4 8	

Note

In the first test case, after given rearrangement, $|a_1 - a_2| = 0 \leq |a_2 - a_3| = 1 \leq |a_3 - a_4| = 2 \leq |a_4 - a_5| = 2 \leq |a_5 - a_6| = 10$. There are other possible answers like "5 4 5 6 -2 8".

In the second test case, after given rearrangement, $|a_1 - a_2| = 1 \leq |a_2 - a_3| = 2 \leq |a_3 - a_4| = 4$. There are other possible answers like "2 4 8 1".

Codeforces Round #633 (Div. 2)

Finished

→ Virtual participation

Virtual contest is a way to take part in past contest, as close as possible to participation on time. It is supported only ICPC mode for virtual contests. If you've seen these problems, a virtual contest is not for you - solve these problems in the archive. If you just want to solve some problem from a contest, a virtual contest is not for you - solve this problem in the archive. Never use someone else's code, read the tutorials or communicate with other person during a virtual contest.

[Start virtual contest](#)

→ Problem tags

constructive algorithms **sortings** *1200
No tag edit access

→ Contest materials

- Announcement (en) ✕
- Tutorial (en) ✕

FIGURE A.11 – Page d'un challenge

A.5 Codechef



[g+](#)[t](#)[f](#)

Forgot Password

[PRACTICE](#) [COMPETE](#) [DISCUSS](#) [COMMUNITY](#) [HELP](#) [ABOUT](#)

Home » Practice(challenge)

Practice



This is the place to hone your skills.

Try your hand at one of the practice problems, and submit your solution in the language of your choice (our judge accepts solutions in over 35+ languages). Receive points, and move up through the CodeChef ranks. Get better, and better prepare yourself for the competitions

Beginner	Easy	Medium	Hard	Challenge	Peer
Name	Code	Successful Submission	Accuracy		
The Turing Music Box	TMBOX	7	2.12		
(Challenge) 300 IQ Ants	ADANTS	11	36.36		
Lights Off, Revisited!	EX	12	8.11		
Table Tilt	TILT	13	26.44		
Scrabble on a Graph	SCRABBLE	13	23.33		
(CH) Serejs and Billiards	SEABIL	13	39.39		
Chef in a Mysterious Land (Challenge)	MYST	13	36.84		
(Challenge) Chef Exchange	CHEFXCHG	13	46.67		
(Challenge) Codechef War II	CHFWAR2	13	60		
(Challenge) Bacteria Synthesis	SYNBAC	13	38.64		
Chef and New Game	CHNWGM	14	49.21		
Santa Delivering Problem	DSANTAP	14	55.43		

FIGURE A.12 – Page de recherche de challenges

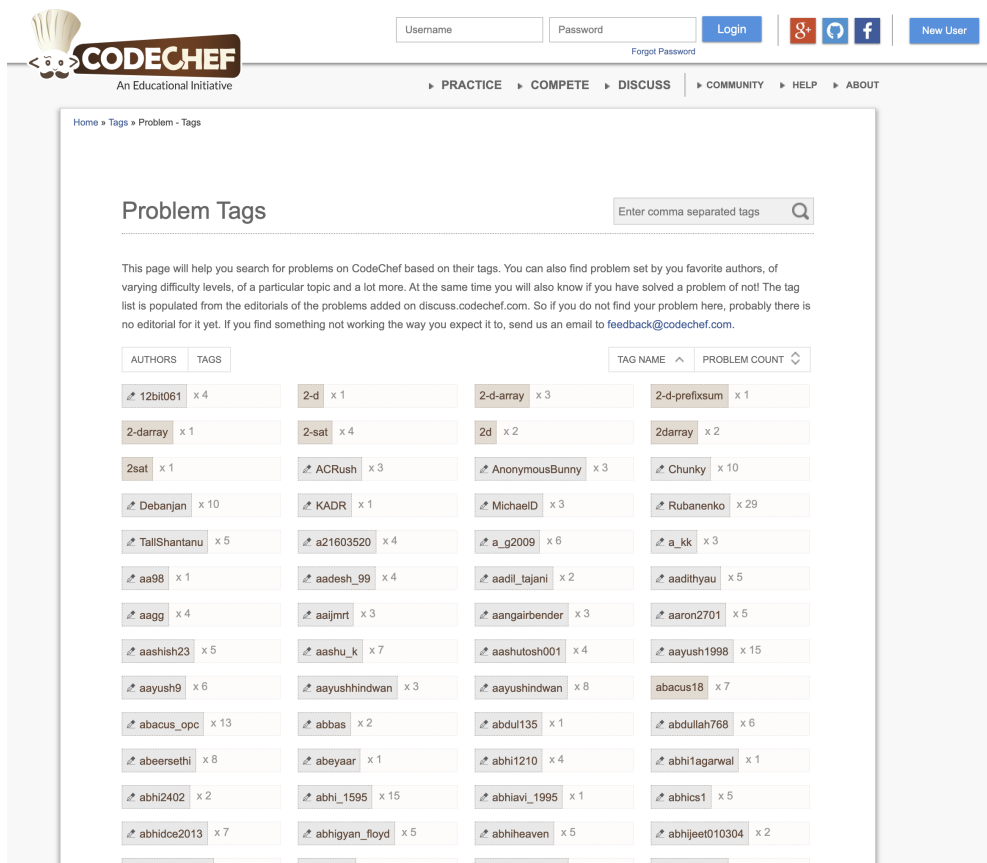


FIGURE A.13 – Page de filtres



FIGURE A.14 – Page d'un challenge

A.6 Coderbyte

The screenshot shows the Coderbyte website's 'Challenges' page. At the top, there's a navigation bar with the Coderbyte logo, links for 'Challenges', 'Courses', and 'Coderbyte for Employers', and buttons for 'UPGRADE' and 'REGISTER / LOGIN'. The main heading is 'Challenges'. Below it, a list of challenges is displayed, each with a title, status (e.g., 'Not Completed'), difficulty (e.g., 'Easy'), and the number of solutions (e.g., 'Solutions: 5890'). Each challenge has links to 'View all user solutions' and 'Discussion'. The challenges listed are: Find Intersection, Questions Marks, Binary Reversal, Letter Changes, Longest Word, First Factorial, First Reverse, React Button Toggle (marked as new), SQL Member Count (marked as new), and Bash Print Memory (marked as new). On the right, a 'Filters' sidebar allows filtering by 'DIFFICULTY' (Easy, Medium, Hard), 'COMPANIES' (Google, Facebook, Amazon, Microsoft), and 'TAGS' (string manipulation, free, recursion, math fundamentals, searching, regular expression, sorting, array, sequences, dynamic programming). At the bottom, a purple banner reads '*FLASH SALE* Get 30% off any premium plan!'.

FIGURE A.15 – Page de recherche de challenges

The screenshot shows the Coderbyte challenge page for 'Find Intersection'. The challenge is marked as 'easy'. The description states: 'Have the function FindIntersection(strArr) read the array of strings stored in strArr which will contain 2 elements: the first element will represent a list of comma-separated numbers sorted in ascending order, the second element will represent a second list of comma-separated numbers (also sorted). Your goal is to return a comma-separated string containing the numbers that occur in elements of strArr in sorted order. If there is no intersection, return the string false.' Examples provided show inputs like ['1, 3, 4, 7, 13', '1, 2, 4, 13, 15'] resulting in '1,4,13'. Tags include 'array' and 'free'. The code editor shows a JavaScript function template:

```
1 function FindIntersection(strArr) {
2
3   // code goes here
4   return strArr;
5 }
6
7 // keep this function call here
8 console.log(FindIntersection(readline()));
9
```

 The test runner on the right shows the input ['1, 3, 4, 7, 13', '1, 2, 4, 13, 15'] and the output logs will appear here.

FIGURE A.16 – Page d'un challenge



École polytechnique de Louvain

Analyse bibliographique

Auteur: **Jacques YAKOUB**

Année académique 2019–2020
Master [120] en sciences informatiques

TABLE DES FIGURES

2.1	Les 15 éléments du Dublin Core	4
2.2	Les 9 catégories d'éléments du LOM	5
2.3	Représentation hiérarchique du LOM	6
2.4	Correspondance des éléments entre le Dublin Core et le LOM	6
2.5	MLR - Spécification d'un élément de données	8
2.6	MLR - Exemple de spécification d'un élément de données	8
2.7	Représentation simplifiée du MLR	9
3.1	Exemple de taxonomie pour les exercices de programmation	10
3.2	Liste des sous-domaines de l'informatique selon le CC2005	11
3.3	Exemple de hiérarchie de compétences	12
3.4	Catégories d'exercices informatiques	12

LISTE DES TABLEAUX

2.1	Les 5 types de données du LOM	7
4.1	Éléments de base de notre norme	14
4.2	Éléments d'un mot-clé de notre norme	15
4.3	Quelques éléments additionnels de notre norme	15

TABLE DES EXTRAITS DE CODE

4.1	Exemple d'une ressource information avec notre norme	16
-----	--	----

Après avoir délimité la problématique de notre mémoire (à savoir le référencement et recherche des ressources informatiques¹), nous avons parcouru la toile à la recherche de divers travaux à ce sujet pour tenter d'apporter des réponses à celle-ci. En effet, bien qu'il soit peu probable d'y trouver une solution prête à l'emploi pour couvrir la totalité de la problématique, cette étape n'en demeure pas moins essentielle dans la mesure où des éléments connexes (voire des approches analogues dans d'autres domaines) à notre recherche peuvent émerger et nous permettre de formuler judicieusement notre réponse.

Afin de synthétiser et vous présenter les résultats de notre recherche, nous utiliserons ce fil rouge :

- l'étude de diverses normes pour décrire une ressource
- la recherche de taxonomie pour classer des ressources informatiques

Enfin, nous concluons la présente recherche par les enseignements qu'elle nous a apportés sur notre mémoire par l'élaboration de notre propre norme pour établir une solution équilibrée.

¹Une ressource informatique désigne tout matériel ayant pour attrait le domaine de l'informatique. Exemple : des exercices de programmation, des tutoriels en informatique, des algorithmes, des slides ...

Se poser la question de la représentation d'une ressource est loin d'être anodin dans la mesure où celle-ci doit faire l'objet d'une démarche scientifique valable pour être recevable :

- Quels sont les critères/caractéristiques à notre disposition pour la décrire ?
- Quelle définition (claire et précise) associer à chaque terme spécifique à celle-ci ?

Notre recherche a fait émerger de très nombreuses normes utilisées par le monde pour décrire des ressources éducatives (informatiques ou non). Bien qu'il serait instructif d'étudier l'ensemble des normes existantes pour en déduire les similitudes et différences entre elles, cela peut s'avérer être une tâche chronophage avec une utilité relativement limitée. En effet, nous avons remarqué les faits suivants :

- Certaines normes ne sont que des versions spécialisées d'une autre (par exemple, le **LOMFR** n'est qu'une déclinaison du **LOM**)
- Certaines normes étendent simplement le travail réalisé par une autre norme, en apportant un nombre restreint de changements (comme la **IMS**¹ qui a commencé son développement au départ du **LOM**)
- La norme la plus ancienne que nous avons trouvée date de 1995, ce qui nous indique que la réflexion sur le référencement sur ce type de ressources est relativement récente par rapport à d'autres domaines. À titre informatif, en 1973, il y avait déjà la norme ISO 2709 pour décrire un objet (livre y compris) dans une bibliothèque.

De ce fait, nous n'avons donc retenu ici que les 3 les plus reconnues au niveau mondial (c.-à-d. celles ayant été validées par des organisations fiables comme l'ISO² et la IEEE³) et intéressantes à présenter. Voici donc la liste des normes choisies, ordonnée en fonction de l'année de leur première version, ainsi que l'abréviation en gras que nous utiliserons dès maintenant pour les désigner respectivement :

- la "Dublin Core" (**DC**), officiellement connue sous le nom de "*ISO 15836*"
- la "Learning Object Metadata" (**LOM**), officiellement connue sous le nom de "*IEEE 1484.12.1 – 2002 Standard for Learning Object Metadata*"
- la "Metadata for learning resources" (**MLR**), officiellement connue sous le nom de "*ISO/IEC 19788 Information technology – Learning, education and training – Metadata for learning resources*"

¹<http://www.imspjproject.org/>

²International Organization for Standardization - <https://www.iso.org/fr/home.html>

³Institute of Electrical and Electronics Engineers - <https://www.ieee.org/>

Nous allons débiter par expliquer le **DC** dont l'organisation **DCMI** [7] en est l'agence de maintenance. Comme expliqué par la Bibliothèque nationale de France [5], cette norme consiste principalement en "15 éléments tous facultatifs et tous répétables" (que l'on peut classer dans 3 grandes catégories de description : contenu, propriété intellectuelle et l'instance de la ressource). Vous pouvez trouver ci-dessous une illustration expliquant chacun de ces éléments (le terme entre guillemets est le terme d'origine et sa traduction française est entre parenthèses) :

<contributor> (contributeur)	Contributeur (auteur secondaire) du contenu de la ressource : personne physique ou collectivité. [On peut conseiller de saisir cet élément de façon normalisée].
<coverage> (couverture)	Couverture spatio-temporelle ou juridictionnelle du contenu de la ressource. Le contenu de cet élément est, le plus souvent, un nom de lieu et/ou des dates en utilisant un vocabulaire contrôlé : par exemple « France » ; « 19 ^e siècle »...
<creator> (créateur)	Créateur du contenu de la ressource : personne physique ou collectivité. [On peut conseiller de saisir cet élément de façon normalisée].
<date> (date)	Date de la création du document. Il est fortement conseillé d'utiliser la norme ISO 8601 qui code la date sous la forme AAAA-MM-JJ. Il est possible de « raffiner » cet élément pour indiquer une date de modification, de sélection de la ressource...
<description> (description)	Description du contenu de la ressource. Le contenu de cet élément peut être un résumé, une table des matières ou encore une description en texte libre.
<format> (format)	Manifestation physique ou numérique de la ressource. Le contenu de cet élément est, le plus souvent, le support ou/et le type d'application informatique nécessaire à sa lecture, par exemple image/jpeg ou text/pdf. On peut également donner des indications concernant la taille du document (par exemple nombre d'octets). L'utilisation de la liste MIME (liste des « types de média définis sur Internet ») est recommandée.
<identifier> (identifiant)	Référence unique et non ambiguë de la ressource. Il est recommandé d'utiliser un système d'identification, comme URI (<i>Uniform Resource Identifier</i>) ou URL (<i>Uniform Resource Locator</i>).
<language> (langue)	Langue du contenu de la ressource. L'utilisation de la norme ISO 639 est recommandée.
<publisher> (éditeur)	Éditeur (au sens commercial) de la ressource : personne physique ou collectivité.
<relation> (relation)	Ressource(s) liée(s) à la ressource décrite (autres versions, formats...). L'utilisation d'un système d'identification (comme URI, URL...) est recommandée.
<rights> (droits)	Informations sur les droits concernant la ressource décrite (gestion des droits, droits d'auteur...).
<source> (source)	Référence d'origine de la ressource décrite. L'utilisation d'un système d'identification est recommandée.
<subject> (sujet)	Sujet du document. L'utilisation d'un système de mots-clefs ou d'indices de classification est recommandée, en particulier les classifications reconnues ou les vocabulaires contrôlés.
<title> (titre)	Titre principal de la ressource, le plus souvent donné par son créateur. [Même s'il n'en est pas question dans la norme, on peut recommander de rendre cet élément obligatoire pour la bonne gestion des systèmes et des échanges de données.]
<type> (type)	Information sur la nature ou le genre de la ressource (texte, image...). L'utilisation d'un terme choisi dans une liste contrôlée, par exemple celle proposée sur le site Dublin Core, est recommandée.

FIGURE 2.1 – Les 15 éléments du Dublin Core [18]

Le lecteur/lectrice attentif/attentive que vous êtes aura noté la présence du terme "vocabulaire contrôlé" [22] ("controlled vocabulary" dans le texte d'origine de la norme [7]). Comme l'explique Wikipédia [22], il s'agit d'un "lexique dont le but est de rendre possible l'organisation des connaissances afin d'optimiser la recherche d'information." Étant donné qu'il impose l'usage de termes prédéfinis, celui-ci s'oppose au "langage naturel, où aucune restriction n'existe quant au choix du vocabulaire". Illustrons ceci par un petit exemple sur l'élément "format" avec la même information, mais exprimée différemment :

- vocabulaire contrôlé (par exemple : MIME⁴) : application/pdf;charset=utf-8
- langage naturel : fichier PDF en UTF-8

La norme **DC** n'impose pas des normes précises (mais recommande explicitement [7] l'usage des "vocabulaires contrôlés" [22]) pour les divers éléments expliqués ci-dessous : en effet, des organisations telles que la Bibliothèque nationale de France [5] se doivent d'avoir des exigences strictes de référencement pour optimiser la recherche d'informations.

Enfin, la norme **DC** inclut la notion de "profil d'application" (de son nom technique "Dublin Core™ Application Profile" [8]). Derrière ce nom se cache un constat assez simple à établir : tenter d'établir une norme unique et stricte est une procédure qui ne peut aboutir. En effet, chaque organisation a des besoins et des exigences différents sur les divers éléments de la norme **DC**. Comme expliqué par la **DCMI** [8], le profil d'application consiste donc en une déclaration qui spécifie quels éléments de la norme **DC** et quelles normes supplémentaires/spécifiques dessus sont d'application.

Nous allons poursuivre par le **LOM** dont le comité **IEEE-LTSC-LOM** [4] (dépendant donc du IEEE que nous avons expliqué précédemment) est responsable. Utilisant le **DC** comme point de départ, celle-ci propose comme expliqué par Wikipédia [20] et "9 catégories regroupant 68 éléments" dont voici un petit descriptif :

Description des catégories du LOM	
Catégories	Description
1. General	Regroupe les caractéristiques d'une ressource qui sont indépendantes du contexte d'utilisation (titre, langue, description, etc.).
2. Life cycle	Décrit l'état actuel d'une ressource et qui y a contribué durant son évolution (version, entités contributives, etc.).
3. Meta-Metadata	Rassemble les données détaillant la fiche descriptive elle-même plutôt qu'une ressource.
4. Technical	Assemble les caractéristiques techniques d'une ressource (format, taille, localisation...).
5. Educational	Décrit les caractéristiques pédagogiques d'une ressource (type de ressource, rôle de l'utilisateur, contexte d'utilisation...).
6. Rights	Spécifie les conditions d'utilisation d'une ressource (coûts, droits d'auteur, etc.).
7. Relation	Décrit, s'il y a lieu, la relation entre une ressource et d'autres ressources.
8. Annotation	Permet des commentaires sur l'utilisation pédagogique d'une ressource.
9. Classification	Décrit la localisation d'une ressource dans un certain système de classification.

FIGURE 2.2 – Les 9 catégories d'éléments du **LOM** [20]

Nous vous proposons dès lors une illustration pour représenter concrètement les divers éléments du **LOM** dans ces catégories dans la figure 2.3.

⁴RFC 6838 - <https://tools.ietf.org/html/rfc6838>

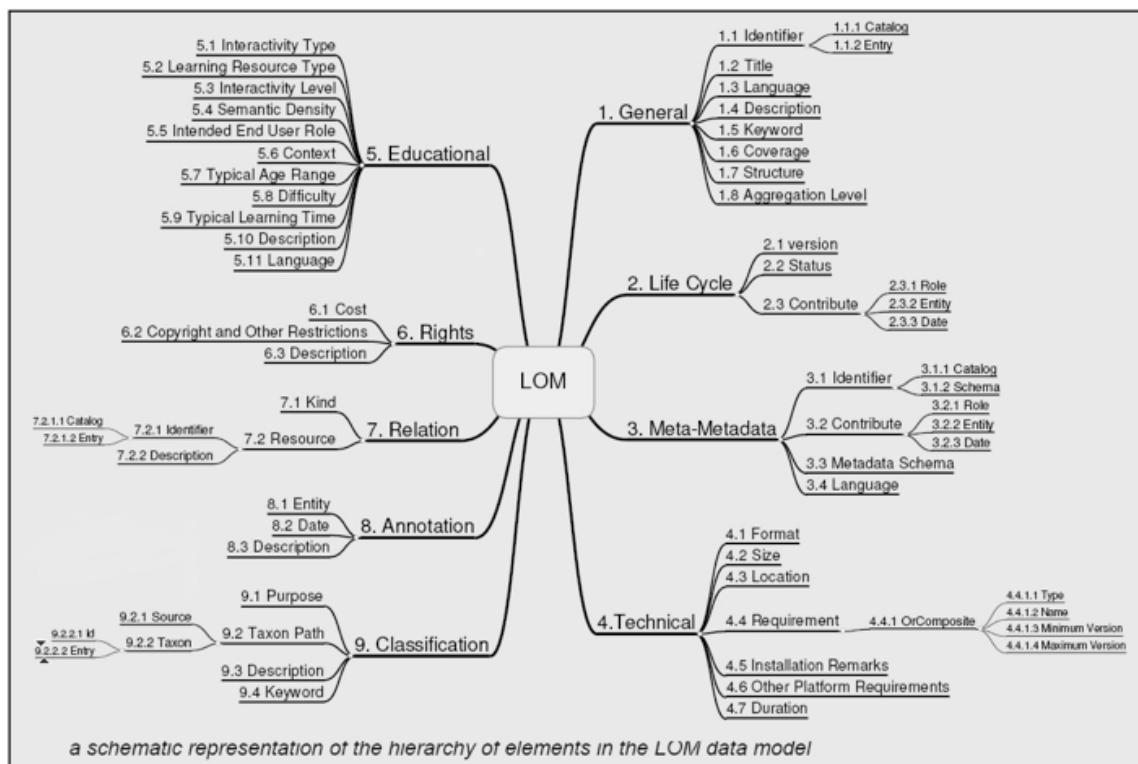


FIGURE 2.3 – Représentation hiérarchique du LOM [10]

Cette norme, qui dispose aussi des profils d'application que nous avons expliqués ci-dessus, a aussi prévu un processus de correspondance entre chaque élément du **DC** vers son équivalent du **LOM** afin de faciliter la transition vers celle-ci.

<i>Dublin Core</i>	<i>IEEE LOM</i>	
Data Element	Identifiant	Data Element
DC.Identifier	1.1.2	General.Identifier.Entry
DC.Title	1.2	General.Title
DC.Language	1.3	General.Language
DC.Description	1.4	General.Description
DC.Subject	1.5	General.Keyword
DC.Coverage	1.6	General.Coverage
DC.Type	5.2	Educational.LearningResourceType
DC.Date	2.3.3	LifeCycle.Contribute.Date
DC.Creator	2.3.2	LifeCycle.Contribute.Entity
DC.OtherContributor	2.3.2	LifeCycle.Contribute.Entity
DC.Publisher	2.3.2	LifeCycle.Contribute.Entity
DC.Format	4.1	Technical.Format
DC.Rights	6.3	Rights.Description
DC.Relation	7.2.2	Relation.Resource.Description
DC.Source	7.2	Relation.Resource

FIGURE 2.4 – Correspondance des éléments entre le Dublin Core et le LOM [9]

Enfin, tout comme la norme **DC**, le **LOM** ne fait que recommander de bonnes pratiques. Une des recommandations spécifiques du **LOM** est de préciser un type de donnée spécifique pour chaque élément du **LOM**. En effet, dans le **DC**, tous les éléments étaient encodés soit dans le langage naturel soit dans un vocabulaire contrôlé. Cette liberté rend l'indexation et recherche de ressources très difficile à gérer. Vous pouvez trouver ci-dessous une explication sur ces 5 types de données :

Type de donnée	But	Exemple
CharacterString (CS)	Valeur servant d'identifiant formel ou de nom	application/pdf
DateTime (DT)	Exprime la date et l'heure sous une forme prescrite et formatée	2001-12-31T12 :59 :60,999+02 :30
Duration (DU)	Exprime la durée sous une forme normalisée et formatée	P2Y1M2DT1H20M25.55S
LangString (LS)	Exprime une combinaison d'un indicateur de langue et d'une chaîne de caractères formulée dans celle-ci	FR -
Vocabulary (Vs, Ve)	Exprime le besoin d'un vocabulaire contrôlé ou à des listes prédéterminées de termes ou de nombres	Très faible, faible, moyen, élevé, très élevé

TABLE 2.1 – Les 5 types de données du **LOM** [20]

Nous allons conclure par le **MLR** dont l'**ISO/IEC JTC1** (dépendant de l'ISO que nous avons expliqué précédemment) est responsable. Cette norme a pour vocation à unifier **DC** et **LOM** (et possède donc un processus de correspondance permettant de migrer du **LOM** au **MLR**). Comme expliqué par Wikipédia [19], les normes **DC** et **LOM** ont plusieurs problèmes en matière d'interopérabilité ⁵ :

- Les bonnes pratiques de convention sont uniquement recommandées et de ce fait pas obligatoires. (par exemple, un élément de type date en **DC** peut être libellé de manière intraitable pour des recherches)
- L'existence de définitions ambiguës et donc soumises à notre subjectivité personnelle. Par exemple :
 - Dans le **DC**, l'élément date peut être lié à une date de sa création ou de sa publication selon le contexte.
 - Dans le **LOM**, la multitude des profils d'application fait émerger des vocabulaires contrôlés parfois difficilement conciliables entre eux.

⁵(Définition Larousse) <https://www.larousse.fr/dictionnaires/francais/interopérabilité/43787>

De ce fait, le **MLR** tente de résoudre ces problèmes par le biais des "Data Element Specification (DES)" [19] dont vous pouvez trouver des explications détaillées sur chaque champ par les ressources que nous avons utilisées ([1] et [19]), illustrés par un exemple :

Spécification d'un élément de données (DES)	
Identifiant (obligatoire)	
Attributs de l'élément de données	
Nom de la propriété (obligatoire)	
Définition (obligatoire)	
Indicateur linguistique (obligatoire)	
Domaine (obligatoire)	
Image (obligatoire)	
Règles de contenu (conditionnel)	
Raffine (conditionnel)	
Exemple(s) (facultatif)	
Note(s) (facultatif)	

FIGURE 2.5 – MLR - Spécification d'un élément de données [1]

Spécification d'un élément de données (DES)	
Identifiant	ISO_IEC_19788-3:2010::DES0300
Attributs de l'élément de données	
Nom de la propriété	format (eng) format (fra)
Définition	file format, physical medium, or dimensions of the learning resource (eng) format de fichier, type de média ou dimensions de la ressource pédagogique (fra)
Indicateur linguistique	non-linguistique
Domaine	ISO_IEC_19788-1:2010::RC0002 (Ressource d'apprentissage)
Image	Littéral
Règles de contenu	RS_DES0004
Raffine	ISO_IEC_19788-2:2010::DES0900 (format)
Exemple(s)	video/mpeg text/html
Note(s)	-

FIGURE 2.6 – MLR - Exemple de spécification d'un élément de données [1]

Si cette norme peut paraître complexe à comprendre par rapport aux normes précédentes, notre recherche bibliographique nous a permis de trouver divers documents pour résorber ce problème dont notamment ceux du doctorant Nikos Palavitsinis ([13] , [12] ainsi que sa présentation Slideshare [11]) :

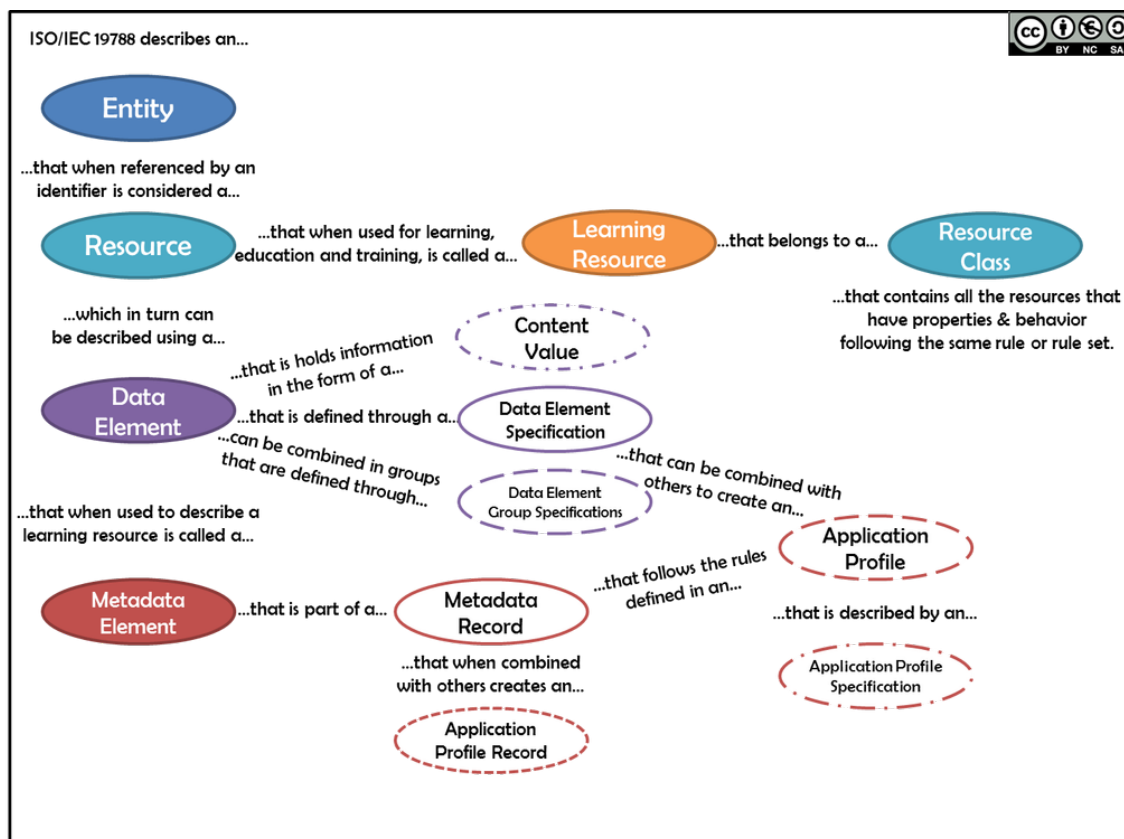


FIGURE 2.7 – Représentation simplifiée du MLR [13]

Pour conclure ce chapitre, nous avons retiré des enseignements de cette étude des normes, qui nous guideront notamment au chapitre 4 :

- Beaucoup d'éléments dans les diverses normes étudiées sont optionnels. De ce fait, certains éléments sont plus appropriés à un contexte que pour un autre.
- Quelle que soit la classification choisie des éléments, il convient d'utiliser quand cela est possible des vocabulaires contrôlés pour répondre à des besoins de recherche dans les ressources.
- Un soin tout particulier doit être accordé à la rédaction des différents éléments d'une norme. Une mauvaise interprétation peut conduire à un nivellement par le bas des objectifs à atteindre.

Dans le chapitre suivant, nous allons nous intéresser aux manières possibles de classer des ressources informatiques puisque notre recherche n'a pas fait émerger une quelconque norme spécialement prévue pour ce contexte.

CHAPITRE 3

TAXONOMIE DE L'INFORMATIQUE

Après avoir abordé la représentation d'une ressource, il convient d'aborder la question de la taxonomie¹ de l'informatique. De manière similaire au chapitre précédent, des interrogations analogiques peuvent se poser :

- Quelles "catégories"² avec leurs caractéristiques/valeurs propres devons-nous utiliser ?
- Quelle définition (claire et précise) associer à chaque terme ?

Notre recherche n'a pas permis de trouver un consensus suffisamment partagé au niveau mondial sur la taxonomie. En revanche, un début de réponse se trouve dans les normes étudiées précédemment telles que le **LOM** (cf. les figures 2.2 et 2.3). En effet, elles permettent de catégoriser une ressource, qu'elle soit informatique ou non. Le **LOM** permet en outre, par le biais de la catégorie n°9, de décrire la ressource dans un système de classification (propre à chaque organisation), tel que celui ci-dessous :

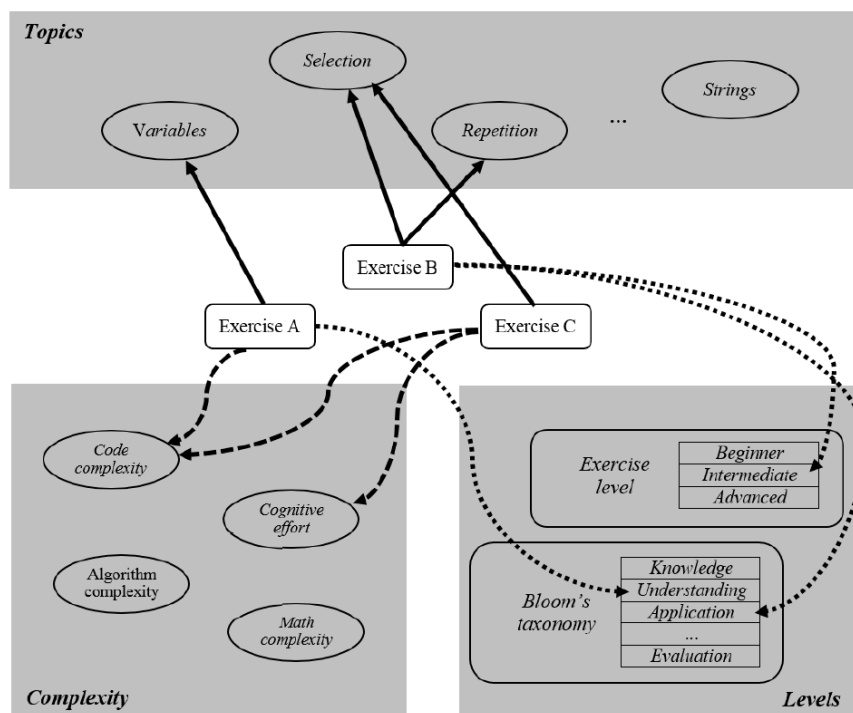


FIGURE 3.1 – Exemple de taxonomie pour les exercices de programmation [14]

¹Comme expliqué par Wikipédia[21], ce terme désigne une "méthode de classification des informations dans une architecture structurée de manière évolutive".

²Le vrai terme dans la littérature scientifique est "taxon" que nous invitons à découvrir par vos propres moyens

Comme expliqué par un rapport [2], l'informatique est une discipline qui se divise en plusieurs (sous) domaines/sujets que vous retrouverez ci-dessous. Bien que cette liste soit l'aboutissement de longues années de travail, elle ne doit pas être considérée comme définitive, car comme l'illustrent ces articles ([17] et [2]), de nouveaux domaines (ou des définitions modifiées) peuvent émerger au fur et à mesure que le temps passe, notamment avec l'arrivée du **CC2005** [15] (Computing Curricula 2005).

Knowledge Area
Programming Fundamentals
Integrative Programming
Algorithms and Complexity
Computer Architecture and Organization
Operating Systems Principles & Design
Operating Systems Configuration & Use
Net Centric Principles and Design
Net Centric Use and configuration
Platform technologies
Theory of Programming Languages
Human-Computer Interaction
Graphics and Visualization
Intelligent Systems (AI)
Information Management (DB) Theory
Information Management (DB) Practice
Scientific computing (Numerical mthds)
Legal / Professional / Ethics / Society
Information Systems Development
Analysis of Business Requirements
E-business
Analysis of Technical Requirements
Engineering Foundations for SW
Engineering Economics for SW
Software Modeling and Analysis
Software Design
Software Verification and Validation
Software Evolution (maintenance)
Software Process
Software Quality
Comp Systems Engineering
Digital logic
Embedded Systems
Distributed Systems
Security: issues and principles
Security: implementation and mgt
Systems administration
Management of Info Systems Org.
Systems integration
Digital media development
Technical support

FIGURE 3.2 – Liste des sous-domaines de l'informatique selon le **CC2005** [3]

Un fait important relevé par un rapport [2] est que l'informatique se concentre autour de 3 processus de base (théorie, abstraction et conception). Ceci pourrait permettre d'avoir un critère supplémentaire pour classifier des ressources informatiques au lieu d'inférer cela par le contexte.

Cette recherche bibliographique sur les ressources informatiques a également révélé des tentatives précédentes de classification de compétences : le **CC2005** [15] que nous avons évoquée plus tôt (utilisable par exemple conjointement avec une norme [3] dérivée du **DC** que nous avons trouvé) ou des propositions de modèles dont notamment celle que nous avons trouvé dans un article de recherche [6] qui proposait de décomposer le concept de compétence en principalement 3 termes connexes, connus sous l'anagramme **KSD** ("Knowledge, Skill, et Disposition" que l'on pourrait respectivement traduire en français, en respectant le sens exprimé de l'article, par "savoirs théoriques, savoirs pratiques et dispositions") dont vous pourrez trouver une mise en situation en figure 3.3.

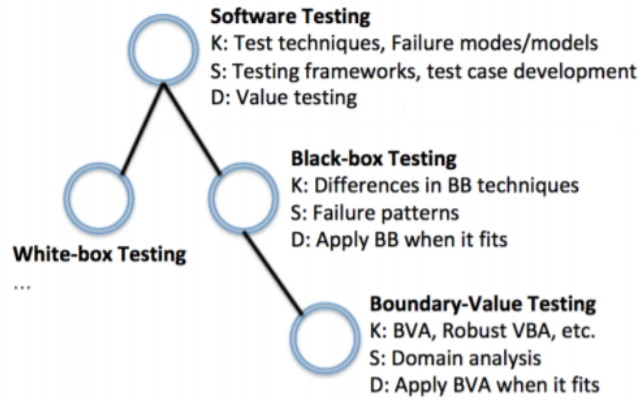


FIGURE 3.3 – Exemple de hiérarchie de compétences [6]

Enfin, dans le cas spécifique d'exercice de programmation, il convient d'avoir une idée du type d'exercice sans forcément lire sa description pour comprendre de quoi il en retourne. Ceci permet en autres de favoriser une diversification des compétences. L'analyse a fait surgir un document récent qui synthétise les plus fréquemment rencontrés :

Concept	Description
Blank sheet	This kind of exercise provides a blank sheet for the student to write his/her solution source code from the scratch.
Code extension	This kind of exercise provides partially finished solution source code (the provided parts are not subject to change by the student) which the student has to complete.
Code improvement	This kind of exercise provides correct initial source code which does not yet achieve all the goals specified in the exercise specification, so the student has to modify it to solve the exercise.
Buggy code	This kind of exercise provides code with bugs (and failed tests) to foster the student to find the right code.
Fill-in the gap	This kind of exercise provides code with missing parts and asks students to fill them with the right code.
Mixed code	This kind of exercise breaks a solution into several blocks of code, mixes them, and asks students to sort them (e.g., quicksort algorithm).
Show me	This kind of exercise defines a small set of primitives that can be used to solve the challenge (e.g., <code>move('red', 'left')</code> meaning move red block to left stack) and provides a visual animation of the code execution.
Spot the bug	This kind of exercise provides code with bugs and asks students to merely indicate the location of the bugs.

FIGURE 3.4 – Catégories d'exercices informatiques [16]

Malgré tous nos efforts pour trouver une taxonomie cohérente à notre situation, aucune d'entre elles ne semble correspondre à nos besoins spécifiques concernant les ressources informatiques (en plus de ne pas susciter un grand intérêt). Dans notre cas, nous aurions besoin d'une taxonomie laissant la possibilité à chaque organisation de faire ce qu'elles veulent, par l'usage d'un vocabulaire flexible et contrôlé. Ce dernier pourrait ainsi évoluer avec le temps et la plateforme qui l'utilise afin de répondre aux besoins des différents utilisateurs de ce même vocabulaire. Nous évoquons cette piste dans le chapitre 4 en proposant notre propre taxonomie adéquate au domaine des ressources informatiques.

Après étude approfondie de la situation, nous avons jugé utile de créer notre propre norme et terminologie pour les raisons suivantes :

- Les 3 normes que nous avons étudiées (**DC**, **LOM** et **MLR**) proposaient des éléments dits "général" sur des ressources (auteur, copyright, date de création/modification ...) et de ce fait ne tiennent pas compte d'éléments spécifiques des ressources informatiques (langage de programmation, thématique ...). Les quelques normes dérivées de celles-ci, orientées au contexte des ressources informatiques (telles que [3] et [16]), proposent certes des avantages, mais aussi des choix discutables sur certains aspects de leur taxonomie.
- Les "vocabulaires contrôlés" [22] sur les éléments sont au coeur de la problématique. En effet, dépendre exclusivement de champs libres rendra l'indexation et la recherche de ressources de qualité impossible. Comme expliqué au chapitre 2, ce vocabulaire se doit être bien défini et clair or l'analyse bibliographique des différents documents a illustré le fait que certains vocabulaires contrôlés n'étaient pas assez génériques (c.-à-d. trop orientés vers un contexte très particulier), et de ce fait difficilement applicables à d'autres organisations.
- La question de la gestion des vocabulaires contrôlés personnalisés (c.-à-d. créés par nos soins) n'est presque jamais soulevée. La recommandation générale dans les normes étudiées était de prendre une sémantique standardisée telle que certains RFC¹ (comme **RFC 6838**), quand cela est possible. Pour certains éléments (comme le langage de programmation), il n'est pas toujours possible d'avoir une liste exhaustive des valeurs possibles. C'est pourquoi nous avons pris la décision d'avoir des vocabulaires contrôlés "extensibles" (c.-à-d. y associer un statut pour encourager la découverte et la modération de valeurs reconnues, dépréciées ...).

Maintenant que nous avons éclairci ces points, nous pouvons passer aux explications sur notre norme. Celle-ci s'est construite sur base de notre réflexion principalement autour des normes **DC** et **LOM**. Contrairement aux normes étudiées, notre norme a fait le choix du schéma **JSON**² (par opposition au **XML**³), car bien qu'étant moins extensible (notamment sur le nombre limité de types de données) que son rival, celui-ci offre une grande facilité de manipulation et de syntaxe aussi bien pour un être humain qu'une machine. Un aspect très important était de proposer une norme ouverte proposant des lignes directrices adaptées aux visions différentes en termes de taxonomie, que nous allons expliquer à la page suivante. En effet, dans le but de référencer un maximum de ressources de sources différentes en un minimum de temps, des robots d'indexation⁴ seront plus que certainement utilisés et il convient dès lors de proposer une structure en conséquence.

¹Requests For Comments - https://fr.wikipedia.org/wiki/Request_for_comments

²JavaScript Object Notation (RFC-8259)

³Extensible Markup Language - <https://www.w3.org/TR/2006/REC-xml11-20060816/>

⁴En anglais, le terme officiel est crawler

Par similarité avec les autres normes, les termes techniques de notre norme seront en anglais, mais nous les présenterons en français. Débutons par les éléments de base de notre norme :

Élément	Cardinalité	Description
title	1	Il s'agit du titre de la ressource. S'agissant d'un champ libre, nous recommandons de se référer à l'élément "title" du DC ou 1.2 du LOM
description	1	Il s'agit d'une description de la ressource. S'agissant d'un champ libre, nous recommandons de se référer à l'élément "description" du DC ou 1.4 du LOM
createdAt	1	Il s'agit de la date de la création de la ressource. Contrairement au type DT du LOM (que nous avons évoqué dans la table 2.1), on impose l'usage exclusif de la norme ISO 8601 .
updatedAt	1	Il s'agit de la date de la dernière mise à jour de la ressource. Contrairement au type DT du LOM (que nous avons évoqué dans la table 2.1), on impose l'usage exclusif de la norme ISO 8601 .
state	1	Il s'agit de l'état de la ressource dans le système. Nous recommandons l'usage de ce vocabulaire contrôlé : • DRAFT : l'état par défaut (par exemple la ressource vient d'être insérée dans un système) • PENDING : la ressource est prête pour validation • VALIDATED : la ressource est validée par un administrateur • NOT_VALIDATED : la ressource est refusée par un administrateur • ARCHIVED : la ressource est archivée
version	1	Il s'agit de la version de la ressource. Contrairement au LOM , nous imposons l'usage d'un type numérique positif pour représenter cette information. (0 : version initiale de la ressource, 1 : première modification de la ressource ...)
url	0-1	URL qui référence la ressource. Cette adresse doit être conforme au RFC 3986 .

TABLE 4.1 – Éléments de base de notre norme

Ensuite, tout comme les normes **DC** et **LOM**, un certain nombre d'éléments facultatifs et répétables sont prévus. Comme nous l'avons déjà expliqué, la taxonomie et terminologie de l'informatique étant propre à chaque organisation, nous devons trouver un moyen de concilier tout ceci (cf. vocabulaires contrôlés "extensibles"). L'idée retenue a été de les consigner sous une clé "tags" représentant un tableau dont chaque cellule est constituée des éléments suivants :

Élément	Description
text	Champ textuel libre. Nous recommandons quand cela est possible de suivre une norme reconnue (par exemple l'ISO 639 pour les langues).
category	Identifiant permettant d'identifier à quel élément le mot-clé se rapporte

state	Il s'agit de l'état du mot-clé dans le système. Nous recommandons l'usage de ce vocabulaire contrôlé : • PENDING : l'état par défaut (par exemple le mot-clé est en cours de validation) • NOT_VALIDATED : le mot-clé n'est officiellement pas reconnu • DEPRECATED : le mot-clé devrait plus être utilisé ultérieurement • VALIDATED : le mot-clé est officiellement reconnu
-------	---

TABLE 4.2 – Éléments d'un mot-clé de notre norme

De cette manière, nous proposons une alternative flexible permettant de réutiliser des éléments du **LOM** tout en rajoutant des nouveaux et d'implémenter de manière plus simplifiée une recherche multicritère⁵. Voici quelques éléments récurrents que nous avons pu identifier durant la conception de notre solution :

Identifiant	Élément	Description
2	cours	Il s'agit de décrire à quel(s) domaine(s) de l'informatique (ou cours) la ressource fait attrait. Nous recommandons de partir de la liste des domaines en figure 3.2 avant de créer de nouvelles entrées.
3	type d'exercice	Il s'agit de décrire la catégorie d'une ressource informatique de type "exercice" auquel elle se rapporte. Nous recommandons de partir de la liste en figure 3.4 avant de créer une nouvelle catégorie.
6	auteur	Il s'agit de décrire le(s) auteur(s) de la ressource. Nous recommandons pour ce champ libre d'adopter une convention claire d'écriture (par exemple le prénom suivi d'un espace et puis le nom), mais nous restons tributaires de la coopération du créateur de la ressource.
8	thématique	Il s'agit de décrire le(s) thème(s) liés à la ressource informatique. Puisque nous n'avons trouvé aucun consensus suffisamment partagé sur ce domaine, ce champ libre est impacté par la subjectivité du créateur de la ressource (exemples : "listes chaînées", "pointeurs", "CUnit" ...).
9	mots-clés	Il s'agit de référencer les mots-clés n'entrant dans aucune des catégories d'éléments existantes. Ce champ libre est exclusivement impacté par la subjectivité du créateur de la ressource (exemples : "Module 3", "Mission 42", "Septembre 2016").
10	licence	Il s'agit de décrire à quel(s) license(s) logiciels la ressource est soumise. Nous recommandons de partir des identifiants du standard SPDX ⁶ avant de créer de nouvelles entrées.

TABLE 4.3 – Quelques éléments additionnels de notre norme

⁵Par exemple : "Je veux rechercher toutes les ressources traitant d'intelligence artificielle, créées par l'auteur X ou l'auteur Y, réalisables sur la plateforme Z"

⁶<https://spdx.org/licenses/>

Pour illustrer l'encodage d'une ressource informatique avec notre norme, nous vous proposons l'exemple suivant :

Listing 4.1 – Exemple d'une ressource information avec notre norme

```

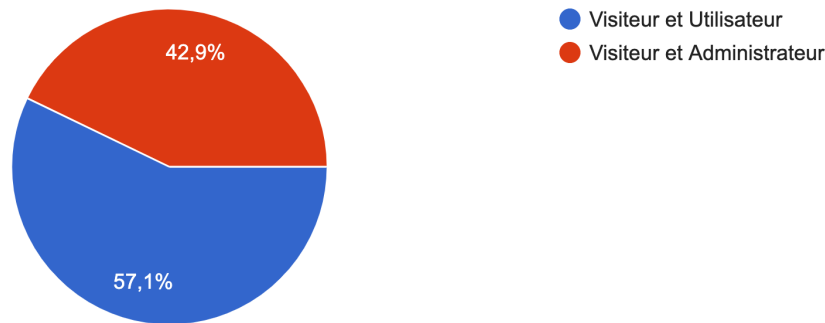
1 {
2   "title": "Opérations sur les bits",
3   "description": "...",
4   "url": "https://inginius.info.ucl.ac.be/course/LINGI1122/
      execute",
5   "createdAt": "2019-12-22T15:18:31.090Z",
6   "updatedAt": "2019-12-22T15:19:33.473Z",
7   "state": "DRAFT",
8   "version": 1,
9   "tags": [
10    {
11      "text": "[LSINF1121] Algorithmique et structures de
          données",
12      "state": "VALIDATED",
13      "category": 2
14    },
15    {
16      "text": "code",
17      "state": "DEPRECATED",
18      "category": 3
19    },
20    {
21      "text": "Mathieu Xhonneux",
22      "state": "PENDING",
23      "category": 6
24    },
25    {
26      "text": "John Aoga",
27      "state": "PENDING",
28      "category": 6
29    },
30    {
31      "text": "CC-BY-NC-4.0",
32      "state": "VALIDATED",
33      "category": 10
34    }
35  ]
36 }
```

Nous concluons le présent travail en précisant que la présente norme (et la taxonomie liée) possède bien entendu des faiblesses en dépit de sa structure ouverte : comme les efforts précédents que nous avons expliqués aux deux derniers chapitres (**LOM**, la proposition d'arborescence de compétences en figure 3.3 ...), l'internationalisation de la description d'une ressource est le principal problème. En effet, des divergences manifestes de terminologie et/ou de description existent (par exemple la vision de l'enseignement aux États-Unis comparé en Europe) et des normes tentant de résorber ce problème (telles que le **MLR**) ne semblent pas encore suffisamment adoptées et/ou connues.

- [1] Yolaine BOURDA et al. “Métadonnées pour ressources d’apprentissage (MLR) : nouvelle norme ISO de description de ressources pédagogiques”. Dans : *STICEF (Sciences et Technologies de l’Information et de la Communication pour l’Éducation et la Formation)* 17 (2010), 11 pages. URL : <https://hal.archives-ouvertes.fr/hal-00696450>.
- [2] Peter J. DENNING et al. “Computing as a Discipline”. Dans : *IEEE Computer* 22.2 (1989), p. 63-70. DOI : 10.1109/2.19833. URL : <https://doi.org/10.1109/2.19833>.
- [3] Stephen H. EDWARDS et al. “Developing a Common Format for Sharing Programming Assignments”. Dans : *SIGCSE Bull.* 40.4 (nov. 2008), p. 167-182. ISSN : 0097-8418. DOI : 10.1145/1473195.1473240. URL : <https://doi.org/10.1145/1473195.1473240>.
- [4] Institute of ELECTRICAL et Electronics ENGINEERS. “IEEE Standard for Learning Object Metadata”. Dans : *IEEE Std 1484.12.1-2002* (2002), p. 1-40.
- [5] Bibliothèque nationale de FRANCE. *Dublin Core*. URL : <https://www.bnf.fr/fr/dublin-core#bnf-dublin-core-et-bonnes-pratiques>.
- [6] Stephen FREZZA et al. “Modelling competencies for computing education beyond 2020 : a research based approach to defining competencies in the computing disciplines”. Dans : *Proceedings Companion of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education, ITiCSE 2018, Larnaca, Cyprus, July 02-04, 2018*. Sous la dir. de Guido RÖSSLING et Bruce SCHARLAU. ACM, 2018, p. 148-174. DOI : 10.1145/3293881.3295782. URL : <https://doi.org/10.1145/3293881.3295782>.
- [7] Dublin Core™ Metadata INITIATIVE. *DCMI Metadata Terms*. URL : <https://www.dublincore.org/specifications/dublin-core/dcmi-terms/>.
- [8] Dublin Core™ Metadata INITIATIVE. *Dublin Core™ Application Profile Guidelines*. URL : <https://www.dublincore.org/specifications/dublin-core/application-profile-guidelines/>.
- [9] Pythagoras KARAPIPERIS et Demetrios SAMPSON. “Enhancing Educational Metadata Management Systems to Support Interoperable Learning Object Repositories”. Dans : jan. 2003, p. 214-218. DOI : 10.1109/ICALT.2003.1215059. URL : https://www.researchgate.net/publication/221425064_Enhancing_Educational_Metadata_Management_Systems_to_Support_Interoperable_Learning_Object_Repositories.
- [10] Dimitrios A. KOUTSOMITROPOULOS et al. *The Use of Metadata for Educational Resources in Digital Repositories : Practices and Perspectives*. URL : <http://www.dlib.org/dlib/january10/kout/01kout.print.html>.
- [11] Nikos PALAVITSINIS. *Making Sense of ISO/IEC 19788*. Juin 2016. URL : <https://www.slideshare.net/nikospala/making-sense-of-isoiec-19788>.
- [12] Nikos PALAVITSINIS. *Making Sense of ISO/IEC 19788 (part I)*. URL : <https://palavitsinis.wordpress.com/2016/06/23/making-sense-of-isoiec-19788-part-i/>.
- [13] Nikos PALAVITSINIS. *Making Sense of ISO/IEC 19788 (part II)*. URL : <https://palavitsinis.wordpress.com/2016/07/04/making-sense-of-isoiec-19788-part-ii/>.

- [14] Álvaro SANTOS, Anabela GOMES et Antonio MENDES. “A taxonomy of exercises to support individual learning paths in initial programming learning”. Dans : oct. 2013, p. 87-93. DOI : 10.1109/FIE.2013.6684794.
- [15] Russell SHACKELFORD et al. “Computing Curricula 2005 : The Overview Report”. Dans : *SIGCSE Bull.* 38.1 (mar. 2006), p. 456-457. ISSN : 0097-8418. DOI : 10.1145/1124706.1121482. URL : <https://doi.org/10.1145/1124706.1121482>.
- [16] Jakub SWACHA et al. “Defining Requirements for a Gamified Programming Exercises Format”. Dans : *Knowledge-Based and Intelligent Information & Engineering Systems : Proceedings of the 23rd International Conference KES-2019, Budapest, Hungary, 4-6 September 2019*. Sous la dir. d’Imre J. RUDAS et al. T. 159. Procedia Computer Science. Elsevier, 2019, p. 2502-2511. DOI : 10.1016/j.procs.2019.09.425. URL : <https://doi.org/10.1016/j.procs.2019.09.425>.
- [17] Allen B. TUCKER et Peter WEGNER. “Computer Science and Engineering : The Discipline and Its Impact”. Dans : *The Computer Science and Engineering Handbook*. Sous la dir. d’Allen B. TUCKER. CRC Press, 1997, p. 1-15.
- [18] Isabelle WESTEEL. “Indexer, structurer, échanger : métadonnées et interopérabilité”. Dans : *OpenEdition Books* (2010), p. 100-123. DOI : <https://doi.org/10.4000/books.pressesenssib.434>. URL : <https://books.openedition.org/pressesenssib/434>.
- [19] WIKIMEDIA. *ISO/IEC 19788*. Avr. 2020. URL : https://en.wikipedia.org/wiki/ISO/IEC_19788.
- [20] WIKIPEDIA. *Learning Object Metadata*. Avr. 2020. URL : https://fr.wikipedia.org/wiki/Learning_Object_Metadata.
- [21] WIKIPEDIA. *Taxonomie (homonymie)*. Fév. 2020. URL : [https://fr.wikipedia.org/wiki/Taxonomie_\(homonymie\)](https://fr.wikipedia.org/wiki/Taxonomie_(homonymie)).
- [22] WIKIPEDIA. *Vocabulaire contrôlé*. Avr. 2020. URL : https://fr.wikipedia.org/wiki/Vocabulaire_contr%C3%B4l%C3%A9.

C.1 Choix de vos rôles



C.2 Rôles : Visiteur et utilisateur

C.2.1 Premier rôle : Visiteur

Des remarques sur votre expérience avec ce rôle ?

- Système intuitif, clair et très riche en filtres, permettant de cibler/retrouver rapidement l'exo potentiellement recherché.
Petite remarque, mais pas importante et peut-être personnelle : la "barre de recherche" laisse penser qu'il s'agit d'une recherche à travers tout le site et non pas directement au sein de la bibliothèque (peut-être parce qu'elle est placée dans le header et non pas au sein de la "page" bibliothèque ?)
- Simple et intuitif
- La barre de recherche textuelle est placée en haut du site, comme si c'était pour rechercher sur le domaine entier. Je pense qu'elle serait mieux mise un peu plus bas, à droite de "réinitialiser ma recherche" par exemple.

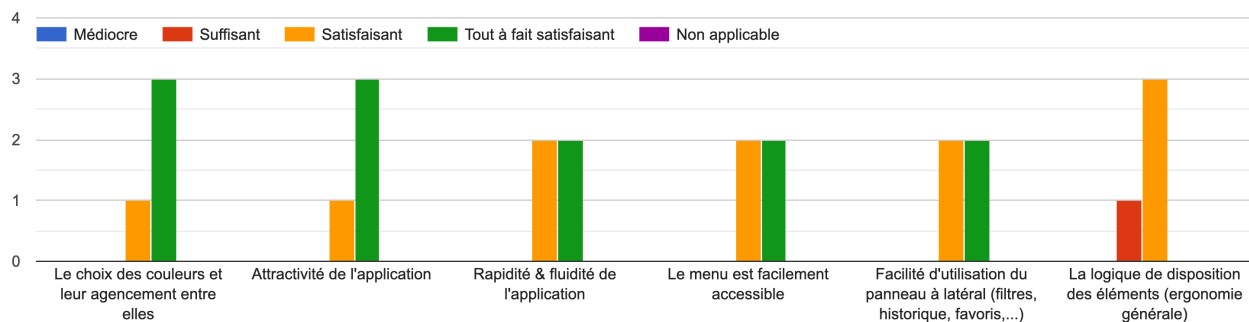
C.2.2 Deuxième rôle : Utilisateur

Des remarques sur votre expérience avec ce rôle ?

- Pas de remarque

C.2.3 Questionnaire

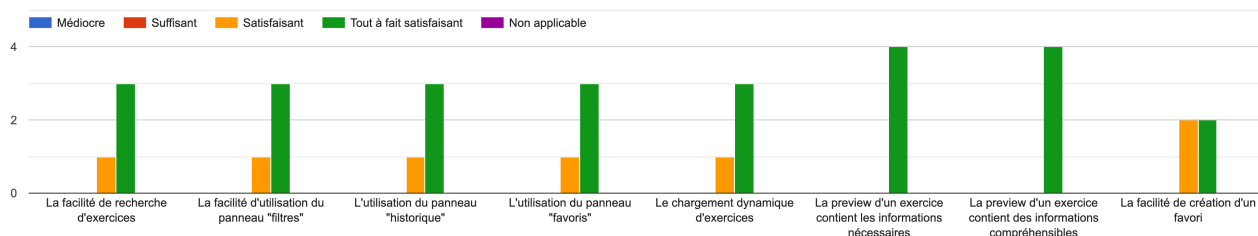
Esthétique & Ergonomie



Des remarques ?

- ++ Très chouette ! Remarque similaire à la précédente, avec la barre de recherche dans le header.
Je dois avouer que le bouton "retour" dans le header est également un peu perturbant, je ne l'ai découvert qu'après un petit moment d'utilisation du site. Son emplacement est peut-être moins intuitif.
- peut être la barre de recherche pour les exercices et les chevrons dans le menu de gauche pas nécessaires
- Je pense tout a été dit dans l'appel :)
- Un dark mode ? Personnaliser le thème de couleurs ?

Page bibliothèque



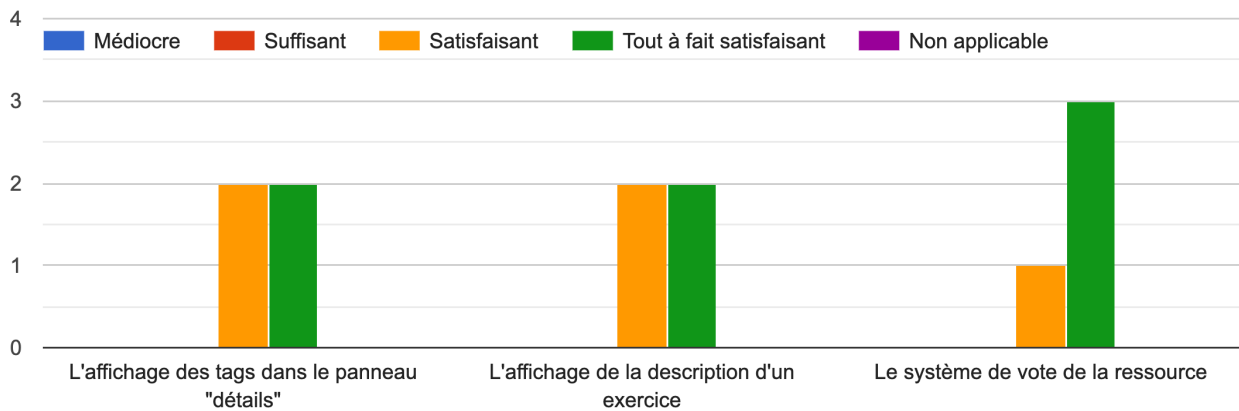
Des remarques ?

- Vraiment un détails mais à la création d'un favori, si une fois le nom entré, la touche "enter" permettait de directement créer le favori au lieu de devoir cliquer, ça pourrait être sympa (mais de nouveau, on est dans le détails)
- Petites remarques :
 - Si je suis dans la bibliothèque, que je clique sur l'onglet Favoris et que je choisis d'en modifier un, (j'arrive donc sur la page de modification du favori) le bouton de retour (dans le header) me propose de retourner dans "Mes favoris", alors que je venais de la bibliothèque.

- Toujours en étant dans l'onglet Favoris de la Bibliothèque, si je clique sur le "stylet" me permettant d'aller modifier le favori, ça exécute le favori (ça lance la recherche) avant de me transférer à la page de modification du favori.

- Juste je pense pas nécessaire d'avoir un tag pour créer un favori

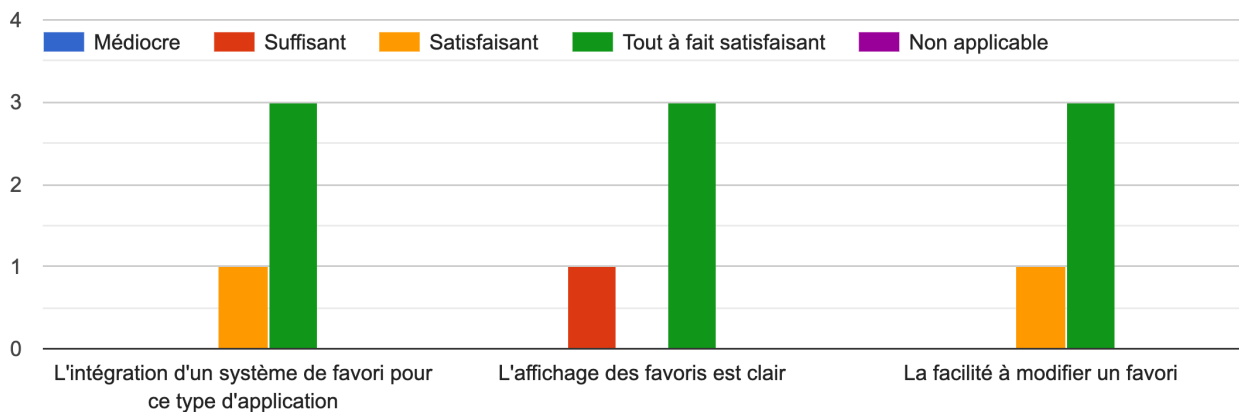
Page d'une ressource informatique



Des remarques ?

- Il n'y a actuellement aucune réponse à cette question.

Page des favoris

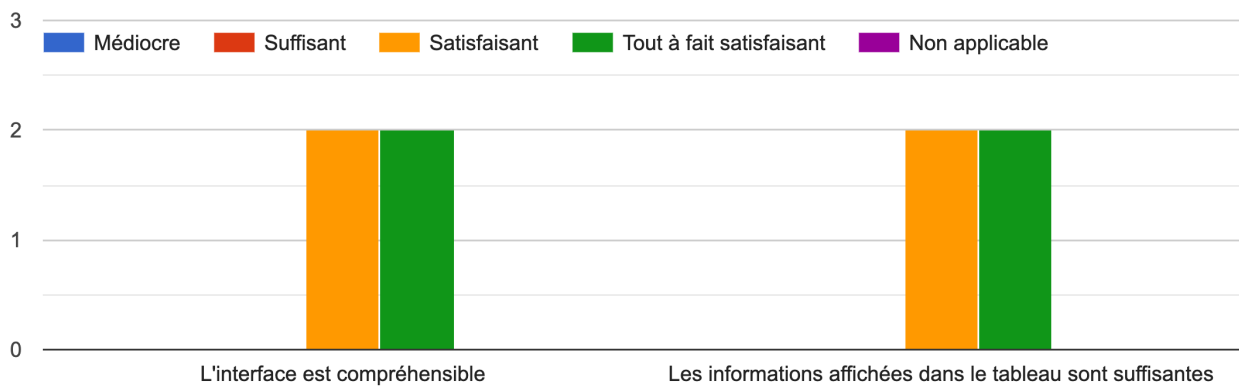


Des remarques ?

- Petit détails mais, je crée un favori en sélectionnant plusieurs tags de plusieurs catégories. Parmi eux, je choisis "non spécifié" (Langage). Je reviens par la suite pour modifier mon favori, je revois donc ma liste de tags choisis et je vois "non spécifié", sauf que.. Difficile de se souvenir de quelle catégorie il s'agit. Peut-être renommer des tags pas assez parlant, ou alors lorsque l'on est sur la page de modification du favori et que l'on clique sur un des tags déjà présents, ça ouvre le menu déroulant des tags au bon endroit (nous permettant de directement savoir de quelle catégorie il s'agit)

- Manque d'informations dans les carte favoris de la bibilothèques, on ne devrait pas avoir à retourner dans la liste de fav pour voir ce que fait le favoris

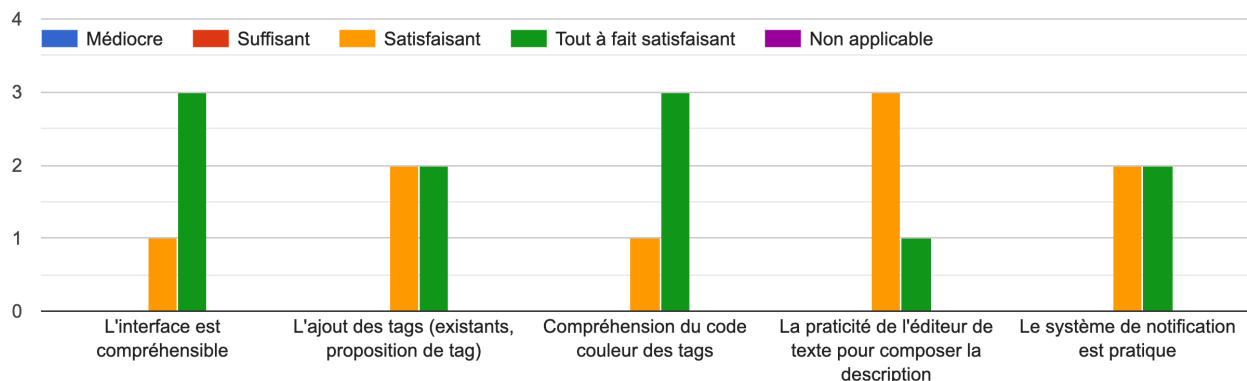
Page de gestion des ressources informatiques



Des remarques ?

- Il n'y a actuellement aucune réponse à cette question.

Page de création/modification d'une ressource informatique



Des remarques ?

- (((Interligne peut-être un peu grande)))
Frustrant de pas pouvoir créer un bloc de code directement depuis plusieurs lignes sélectionnées. (Peut-être ne pas trop miser sur la page 'Tutoriel', je suis pas certain que les gens ont tendance à se rendre sur des manuels pour comprendre un outils)
- Glitch UI : quand on est sur "mes exercices" et qu'on clique sur "créer un exercice" les exercices disparaissent.

C.3 Visiteur et Administrateur (modérateur)

C.3.1 Premier rôle : Visiteur

Des remarques sur votre expérience avec ce rôle ?

- Quelques remarques :
 - proposition : Gestion des fautes de frappes et/ou de la recherche approximative.
 - Le texte de la barre de recherche est trop petit, et en bleu trop clair.
 - La barre de recherche d'exercices est dans la barre principale, et donne l'impression qu'elle recherche dans tout le site. Elle est trop loin des filtres.
 - Cf. image de github pour avoir des catégories de tags (i.e. filtres)
- Plutôt fluide. Eu un peu de mal à scroller dans les listes de tags par filtres (Cliquer sur "languages" puis scroll jusqu'à Java). Il y a bien deux scroll bars mais ce n'est pas super intuitif
- Tout s'est bien passé. 2 petites remarques d'ergonomie : la double liste déroulante dans le panneau des tags lorsqu'une catégorie est sélectionnée m'a un peu perturbée. Et je n'ai pas trouvé tout de suite dans la barre de recherche (action 2)

C.3.2 Deuxième rôle : Administrateur

Des remarques sur votre expérience avec ce rôle ?

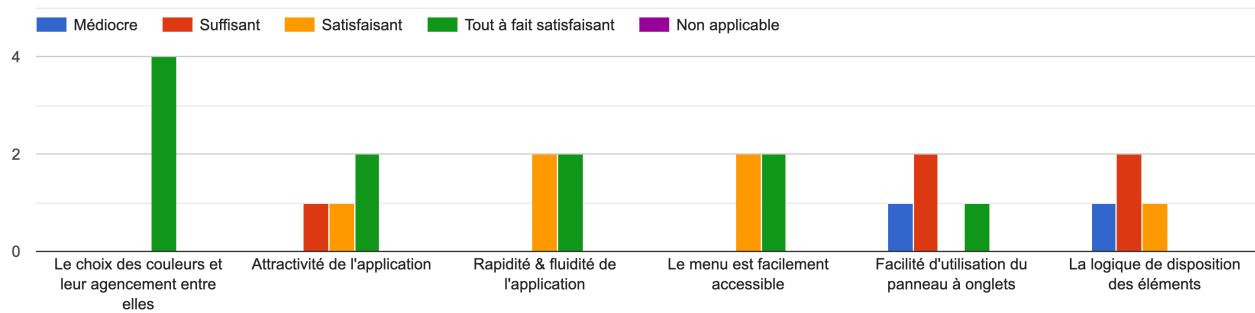
- Editeur de texte!!!!
 - Pas de <tab>
 - Sélectionner plusieurs lignes pour en faire un bloc de code génère un bloc de code par ligne.
 - Possibilité d'agrandir la boîte de texte si besoin (handle)
 - Petit bug avec le copier-coller et les fins de ligne (unix/windows)

Les tags représentés par un filtre, sur le côté, c'est louche. Pas fait le 5 de admin.

- J'ai malencontreusement supprimer le tag admin en attente sans faire exprès et n'ai pas pu tester ça, mais à part l'absence de possibilité d'indenter mon code dans l'éditeur, ça va. Encore une fois, le tab des tags est un peu difficile à utiliser, mais dès la seconde utilisation ça passe/
- Pas de souci majeur rencontré dans le cadre de cette expérience.

C.3.3 Questionnaire

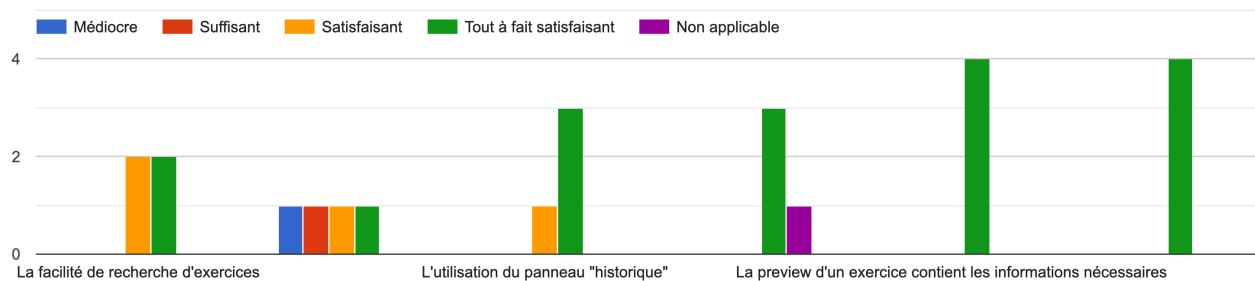
Esthétique & Ergonomie



Des remarques ?

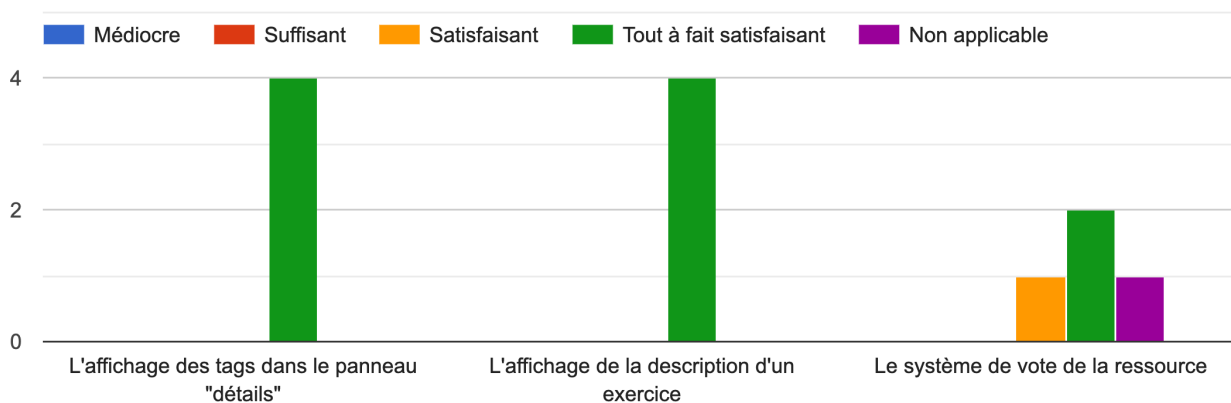
- Bien joué !
- Tout a été dit dans l'appel je pense :)
- On en a parlé, mais le menu filtres/tags est un peu trop détaché de ce qu'on est en train de faire
- Attention à la zone de recherche de la page bibliothèque qui semble décalée de tout le reste. De plus, les 2 icônes s'y trouvant semblent beaucoup plus petites que le reste de la page.

Bibliothèque



Des remarques ?

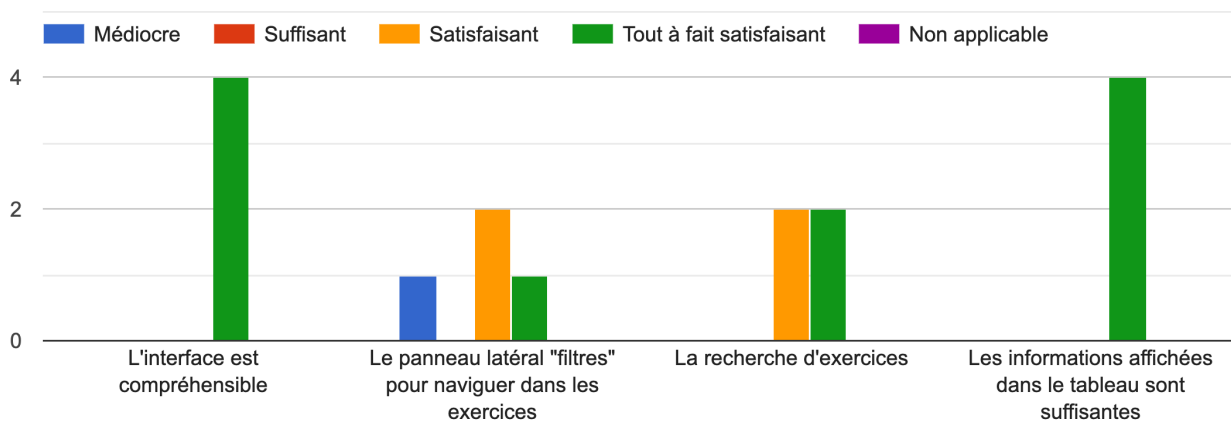
- Il n'y a actuellement aucune réponse à cette question.



Des remarques ?

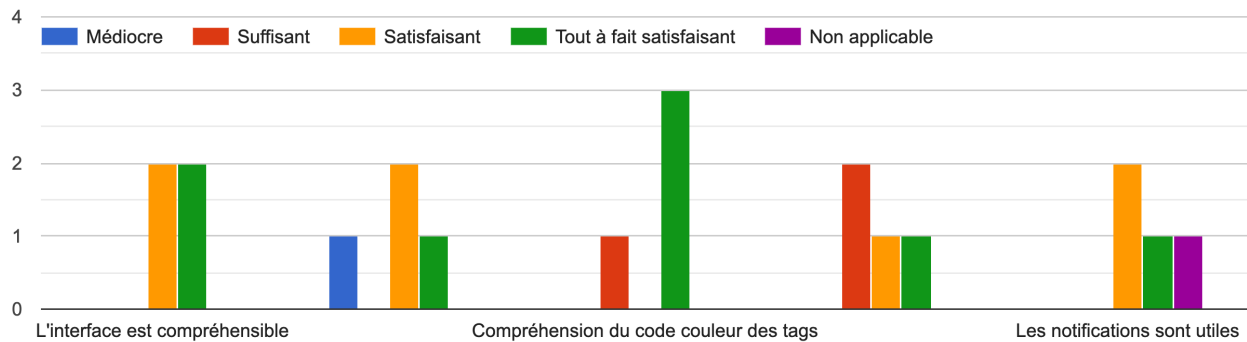
- C'est beau ! Ca me semble être la page la plus accomplie ;)

Page de gestion des ressources informatiques



Des remarques ?

- Il n'y a actuellement aucune réponse à cette question.



Des remarques ?

- Beaucoup de travail pour la gestion d'un tag

C.4 Questions ouvertes

Quels sont les points forts de cette application ?

- Ergonomique et riche en filtres de recherche (utile puisque le but est de rassembler une grande quantité d'exos)
- Elle est très complète
- Très utile si il y a une grande communauté et qu'il y a beaucoup d'exercice. Et la sélection dynamique est top.
- Design visuel, fluidité
- Son design, très ergonomique, lisible et claire, bon agencement des composants
- L'idée de rassembler toutes les ressources informatiques au même endroit me semble pertinent et utile. En plus, le système de tags (utilisés correctement par vos utilisateurs, ça vous ne pouvez pas garantir) me semble flexible et potentiellement très utile
- Application très complète et visuellement attrayante

Quels sont les points faibles de cette application ?

- C.f. remarques et détails de la partie précédente
- Beaucoup d'information pas toujours facilement identifiable.
- Si il n'y a que peu d'exercice, la plateforme aura peu d'intérêts. Et il faut aussi voir si les admins ne risquent pas d'être débordés de requetes en cas de trop grande affluence
- Clarté et association visuelle
- Le nombre d'actions à faire pour une recherche, de manière générale

- Beaucoup de menus et de scroll bars, la learning curve sera donc un peu compliquée pour les utilisateurs les moins "tech savvy" (vous seriez surpris du niveau des profs, même d'informatique parfois!)
- La gestion des tags m'a semblé un peu lourde et complexe. Beaucoup d'éléments dans les menus sur le côté, ce qui est un point faible au début de la prise en main de l'application mais qui deviendra un point fort après avoir bien pris en main l'application.

Quels sont les points faibles de cette application ?

- Il s'agit d'un produit bien abouti! Je dirais que c'est un stade où c'est une masse d'utilisateurs et de feedback qui permettraient de déterminer les différentes voies d'amélioration. (D'ailleurs, tout ce qui a été relevé pendant le call relève davantage du détails et de petites mises à jour que de réels problèmes à fix.)
- Clarifier l'interface. L'UI représente cependant beaucoup de travail. Je cois que vous avez priorisé les bonnes choses.
- Permettre une vérification par la communauté et par des utilisateurs vérifiés, ou l'ajout d'un tag " non vérifié "
- Améliorer l'agencement des menus surtout
- Un fonction "mdp oublié" pour les users. Certaines librairies JS font ça assez bien, ça n'est pas spécialement énormément de boulot en plus. Egalement, la possibilité pour ceux qui créent l'exercice d'ajouter une "solution" a l'exercice.
- Principalement écouter le feedback des utilisateurs qui font vos validations. Vous êtes bons niveau performances/etc ; il manque surtout du polish interface graphique
- Gérer cette zone de recherche perdue en haut autrement. C'est ce qui m'a perturbé le plus dans l'utilisation de l'appli.

Que manquerait-il pour que cette application ne soit plus un prototype en terme de fonctionnalités ?

- Je ne vous rejoins pas sur l'idée que Source Code en est au stade de "prototype". Ca ressemble bien plus à un produit fini, en une première version, qui peut être concernée par d'éventuelles mises à jour par la suite.
- Pas grand chose, et beaucoup à la fois. Il s'agit du lent travail de lisser les bugs et les surprises que peuvent rencontrer les utilisateurs.
- peut etre un complétion " automatique " dans la barre de recherche et permettre la recherche de mot clés dans le titre et pas juste tout le titre genre pour trouver les while loops pouvoir taper " loop"
- Rien. Pour moi c'est + qu'un prototype au stade où elle en est. Cela dit, s'il fallait vraiment ajouter UNE seule feature, je mettrais celle où, le créateur d'un exercice peut "ajouter une solution" et un autre utilisateur pourrait "proposer une solution" qui serait soumise à approbation de celui qui a créé l'exercice.
- Une manière d'interagir entre les étudiants et les "créateurs d'exercices" ? Histoire de pouvoir au moins poser des questions et pas être aussi détachés
- Rien

Dans une perspective où ce projet ne serait plus un prototype et ne souffrirait plus de ces points faibles : Pensez-vous que cette application ait un réel impact dans le monde des ressources partagées ?

- Oui
- Oui, très bonne idée. Il faudrait juste voir comment gérer l'accès entre votre plateforme et celles qui permettent la correction des exos (expl Inginious). Par exemple un moyen de linker votre plateforme avec les autres, pour qu'en un clic on arrive directement sur l'exo inginiuous en étant déjà connecté dessus.
- Sans être intégrée à une plateforme automatisée de correction de code, et sans une gestion de parcours et de lien entre les exercices, sans doute pas beaucoup. Elle sera limitée à une utilisation par les enseignants pour créer leurs propres cours.
- Oui je pense que ça peut être utile
- Je ne sais pas, car d'autres plateformes similaires existent et ont la fonctionnalité "ajouter une solution" ou même de "soumettre en ligne".
- Oui, cela peut avoir une belle valeur ajoutée.

Avez-vous eu recours à la section tutoriel ?

- Non
- Oui pour l'édit de texte
- Très sommairement.
- Avant de commencer la session de tests, pas après

Utiliseriez-vous cette plateforme ? Si oui, quel serait alors votre activité principale ? Si non, pourquoi ?

- Je pense que pour des professeurs, cela peut être très utile pour trouver et donner des exos à des étudiants (ou d'avoir des idées)
- Trouver des idées d'exercices pour compléter mes cours.
- non, je pense que ça s'appliquerait surtout au domaine éducatif
- Oui
- Oui, en tant que Professeur / enseignant, pourvu que les solutions soient mises à dispositions.
- Sans doute pas, dans la mesure où INGINIOUS fait des choses similaires tout en étant open source et présent dans tous les environnements auxquels je suis confrontée.
- Oui, je pourrais l'utiliser. Recherche d'idées dans divers domaines.

```

1 #!/usr/bin/env node
2
3 require('yargs')
4   .command(require('./commands/crawler'))
5   .command(require('./commands/uploader'))
6   .command(require("./commands/archiver"))
7   .command(require("./commands/validator"))
8   .help()
9   .argv;

```

Code D.1 – Point d'entrée d'une application, sur base de Yargs

```

1 #!/usr/bin/env node
2 const path = require('path');
3 const {promises: fs, readFileSync} = require("fs");
4 const PATH_FOR_STRATEGY = path.resolve(__dirname, "..", "./strategies");
5 const DEBUG_FILE = path.resolve("./results.json");
6
7 exports = module.exports = {
8   "command": "crawler",
9   "describe": "export metadata from your exercises to Source Code",
10  "builder": function (y) {
11    return y
12      .option("strategy", {
13        alias: "s",
14        type: "string",
15        description: "Name of the already implemented strategy you
16        ↪ want to use",
17        choices: ["inginiuous-git"]
18      })
19      .option("custom_strategy", {
20        alias: "cs",
21        type: "string",
22        conflicts: "strategy",
23        description: "Absolute path to a JS file implementing your
24        ↪ strategy (see docs for more info)"
25      })
26      .option("workingDirectory", {
27        alias: "w",
28        type: "string",
29        description: "Absolute path to a folder where the crawler can
30        ↪ do its stuff",
31        demandOption: true
32      })
33  }
34 }

```

```

30     .option("resultFile", {
31         alias: "out",
32         type: "string",
33         description: "Absolute path to a JSON file the crawler could
↪ write its results",
34         default: DEBUG_FILE
35     })
36     .coerce("workingDirectory", (arg) => {
37         return path.resolve(arg);
38     })
39     .config("settings", "Absolute path to a JSON config file for
↪ strategy + crawler", (configPath) => {
40         return JSON.parse(readFileSync(path.resolve(configPath), 'utf
↪ -8'));
41     })
42     .help()
43     .argv;
44 },
45 "handler": function (argv) {
46     fs
47         .mkdir(argv.workingDirectory, {recursive: true})
48         .then(() => fetch_and_save_results(argv))
49         .catch((err) => {
50             console.error(err);
51         });
52 }
53 };
54
55 async function fetch_and_save_results(argv) {
56     // If custom script, invoke this to get results
57     try {
58         const results =
59             (argv.hasOwnProperty("custom_strategy"))
60             ? await require(argv.custom_strategy)(argv)
61             : await require(path.resolve(PATH_FOR_STRATEGY, argv.strategy)
↪ )(argv);
62
63         await fs.writeFile(argv.resultFile, JSON.stringify(results, null, 4));
64         console.log("SUCCESSFULLY SAVED THE RESULTS");
65         return await Promise.resolve(results)
66     } catch (e) {
67         return await Promise.reject(e);
68     }
69 }

```

Code D.2 – Exemple d'une implémentation d'une commande "Yargs"

```

1 // Extracted from models/Exercises (it should be renamed to "Fiche" one day)
2 // The interesting method is tagsConditionsBuilder
3 const Sequelize = require("sequelize");
4 const Op = Sequelize.Op;

```

```

5
6 const simpleCase = (array, mustBePresent) => {
7   // positive tags : 2 OR 3 OR 4 , etc...
8   // negative tags : NOT 2 OR NOT 3 , etc...
9   return Sequelize.where(
10    // for negative check, we should use contains instead
11    // For example (using the examples above) :
12    //   For positive check : tags_ids && [2,3,4]
13    //   For negative check : tags_ids @> [2,3]
14    Sequelize.where(
15      Sequelize.col("tags_ids"),
16      (mustBePresent) ? Op.overlap : Op.contains,
17      array
18    ),
19    Op.is,
20    mustBePresent
21  )
22 };
23
24 const tagsWhereBuilder = {
25   "simpleCase": simpleCase,
26   "complexCase": (must_have, must_not) => ({
27     [Op.or]: [
28       simpleCase(must_have, true),
29       simpleCase(must_not, false),
30     ]
31   })
32 };
33
34 // tag condition builder
35 function tagsConditionsBuilder(tags) {
36
37   const conditions = tags.map(tagOrTags => {
38
39     // filter negative/positive integer(s) into array ( more efficient to
40     ↪ check that way )
41     const must_have = Array.isArray(tagOrTags)
42       ? tagOrTags.filter(tag => tag >= 0)
43       : tagOrTags >= 0
44       ? [tagOrTags]
45       : [];
46     const must_not = Array.isArray(tagOrTags)
47       ? tagOrTags.filter(tag => !(tag >= 0)).map(tag => -tag)
48       : tagOrTags >= 0
49       ? []
50       : [-tagOrTags];
51
52     // multiple case can occur because of the mixin of must_have /
53     ↪ must_not checks
54     // One is these case is mandatory true

```

```

53     const kind = (must_not.length > 0 && must_have.length > 0) ? "
↳ complexCase" : "simpleCase";
54
55     // in simple case, it's straightforward
56     if (kind === "simpleCase") {
57         return tagsWhereBuilder[kind](
58             (must_have.length > 0) ? must_have : must_not,
59             // simplification : if must_have is not empty, mustOverlap
↳ will be true , otherwise false (must_not)
60             must_have.length > 0
61         )
62     } else {
63         // the most horrible case
64         return tagsWhereBuilder[kind](must_have, must_not)
65     }
66
67 });
68
69 // as the expression is in Conjunctive Normal Form, we know we can combine
↳ AND and OR formulas
70 return {
71     [Op.and]: conditions
72 };
73 }

```

Code D.3 – Implémentation de la recherche par mots-clés

TABLE DES ILLUSTRATIONS

Table des figures

2.1	La page Bibliothèque	9
2.2	Représentation du système de filtres avec les mots-clés	10
2.3	Page d'une fiche	10
3.1	Exemple d'interface Github Project Board	15
3.2	Exemple d'issues sur Github	15
3.3	Diagramme de Gantt simplifié	16
4.1	Diagramme UML à états pour l'état d'une fiche	22
4.2	Diagramme UML à états pour l'état d'un mot-clé	23
4.3	Propriétés ACID	25
4.4	Triangle CAP avec quelques bases de données	26
5.1	Du JavaScript aussi bien côté serveur que client	27
5.2	Compilation et interprétation - mode de fonctionnement	28
5.3	Bloquant/non bloquant : un exemple pour l'illustrer	28
5.4	VeeValidate : Exemple d'utilisation	32
5.5	Tiptap : Exemple d'utilisation	33
5.6	Finalité d'un ORM	35
5.7	Vue globale de la librairie OpenAPI-Enforcer	36
5.8	Source Code : Représentation des URL de premier niveau	37
5.9	Source Code : interface de connexion	38
5.10	Source Code : partie bibliothèque	39
5.11	Source Code : page bibliothèque	40
5.12	Source Code : la représentation d'une fiche	42
5.13	Source Code : module de gestion (URL)	43
5.14	Source Code : gestion des ressources informatiques personnelles	44
5.15	Source Code : créer une ressource informatique	45
5.16	Source Code : gestion des favoris	48
5.17	Source Code : création/modification des favoris	48
5.18	Source Code : partie administration (URL)	49
5.19	Source Code : importation des ressources informatiques	51
5.20	Source Code : Arborescence de l'API	53
5.21	Exemple de documentation HTML générée sur base de fichiers OAS	54
5.22	Middlewares : modèle de l'oignon	55
5.23	CLI - inventaire des commandes disponibles	56
5.24	Taxonomie des termes et concepts Docker	57
5.25	Docker Hub - des images prêtes à l'emploi	57
5.26	Github Actions - exemple de workflows	58
6.1	GitHub : graphe d'activité (frontend)	65
6.2	GitHub : graphe d'activité (backend)	65

Liste des tableaux

2.1	Les différentes plateformes analysées	3
4.1	Problématique de la recherche par mots-clés	21
4.2	Solution théorique de la recherche par mots-clés	21
5.1	Comparatif de quelques frameworks pour le frontend	30
5.2	Comparatif de quelques frameworks pour l'API	30
5.3	Comparatif de quelques frameworks pour le CLI	31
5.4	Solution technique de la recherche par mots-clés	55

Table des extraits de code

5.1	Exemple des fonctionnalités de Lodash	31
5.2	Axios : Exemple d'utilisation d'une POST request	33

TABLE DES ILLUSTRATIONS DES ANNEXES

Table des figures

A.1	Practice-it : page principale pour rechercher un problème	71
A.2	Practice-it : page d'un problème	72
A.3	Practice-it : recherche avancée	72
A.4	Hackerrank : tableau de bord	73
A.5	Hackerrank : page pour rechercher un problème	73
A.6	Hackerrank : page d'un challenge	74
A.7	Leetcode : page de recherche de problèmes	75
A.8	Leetcode : quelques mots-clés disponibles sur le côté droit	76
A.9	Leetcode : page d'un problème	76
A.10	Codeforces : page de recherche de problèmes	77
A.11	Codeforces : page d'un challenge	78
A.12	Codechef : page de recherche de challenges	79
A.13	Codechef : page de filtres	80
A.14	Codechef : page d'un challenge	80
A.15	Coderbyte : page de recherche de challenges	81
A.16	Coderbyte : page d'un challenge	81
B.1	Les 15 éléments du Dublin Core	86
B.2	Les 9 catégories d'éléments du LOM	87
B.3	Représentation hiérarchique du LOM	88
B.4	Correspondance des éléments entre le Dublin Core et le LOM	88
B.5	MLR - Spécification d'un élément de données	90
B.6	MLR - Exemple de spécification d'un élément de données	90
B.7	Représentation simplifiée du MLR	91
B.8	Exemple de taxonomie pour les exercices de programmation	92
B.9	Liste des sous-domaines de l'informatique selon le CC2005	93
B.10	Exemple de hiérarchie de compétences	94
B.11	Catégories d'exercices informatiques	94

Liste des tableaux

B.1	Les 5 types de données du LOM	89
B.2	Éléments de base de notre norme	96
B.3	Éléments d'un mot-clé de notre norme	97
B.4	Quelques éléments additionnels de notre norme	97

Table des extraits de code

B.1	Exemple d'une ressource information avec notre norme	98
D.1	Point d'entrée d'une application, sur base de Yargs	112
D.2	Exemple d'une implémentation d'une commande "Yargs"	112
D.3	Implémentation de la recherche par mots-clés	113

UNIVERSITÉ CATHOLIQUE DE LOUVAIN

École polytechnique de Louvain

Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | www.uclouvain.be/epl