

École polytechnique de Louvain

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

Auteurs: **Alexandre DEWIT, Jacques YAKOUB**

Promoteurs: **Kim MENS, Olivier GOLETTI**

Lecteur: **Christine JACQMOT**

Année académique 2019–2020

Master [120] en sciences informatiques

TABLE DES MATIÈRES

Table des matières	i
Glossaire	iv
Remerciements	vi
Préambule	vii
1 Introduction	1
2 Problématique	2
2.1 Situation actuelle	2
2.2 Problème	19
2.3 Défis à relever	23
3 Approche	25
3.1 Méthodologie	25
3.2 Planning	27
3.3 Organisation du travail	28
4 Analyse des besoins	30
4.1 Analyse fonctionnelle	30
4.2 Analyse non fonctionnelle	36
4.3 Contraintes	37
5 Solution	39
5.1 Choix technologiques	39
5.2 Client	49
5.2.1 Connexion et création de comptes	50
5.2.2 Exercices	51
5.2.3 Gestion	57
5.2.4 Administration	63
5.3 API	69
5.3.1 Architecture	69
5.3.2 Points clés de l'implémentation	69
6 Analyse critique	70
6.1 Validation	70
6.1.1 Validation externe	70
6.2 Métriques	75
6.3 État du projet	75
7 Pour aller plus loin	76
8 Conclusion	77
Bibliographie	78
Annexes	79
A Analyse Bibliographique	80

TABLE DES FIGURES

2.1 Practice-it : page principale pour rechercher un problème	3
2.2 Practice-it : page d'un problème	4
2.3 Practice-it : recherche avancée	4
2.4 Hackerrank : tableau de bord	6
2.5 Hackerrank : page pour rechercher un problème	6
2.6 Hackerrank : page d'un challenge	7
2.7 Leetcode : page de recherche de problèmes	9
2.8 Leetcode : quelques mots-clés disponibles sur le côté droit	10
2.9 Leetcode : page d'un problème	10
2.10 Codeforces : page de recherche de problèmes	12
2.11 Codeforces : page d'un challenge	13
2.12 Codechef : page de recherche de challenges	15
2.13 Codechef : page de filtres	16
2.14 Codechef : page d'un challenge	16
2.15 Coderbyte : page de recherche de challenges	18
2.16 Coderbyte : page d'un challenge	18
2.17 La page Bibliothèque	21
2.18 Représentation du système de filtres avec les mots-clés	21
2.19 Page d'une fiche	22
3.1 Exemple d'interface Github Project Board	27
3.2 Exemple d'issues sur Github	27
3.3 Diagramme de Gantt simplifié	28
4.1 Diagramme UML à états pour l'état d'une fiche	34
4.2 Diagramme UML à états pour l'état d'un mot-clé	35
4.3 Propriétés ACID	37
4.4 Triangle CAP avec quelques bases de données	38
5.1 Du JavaScript aussi bien côté serveur que client	39
5.2 Compilation et interprétation - mode de fonctionnement	40
5.3 Bloquant/non bloquant : un exemple pour l'illustrer	40
5.4 VeeValidate : Exemple d'utilisation	44
5.5 Tiptap : Exemple d'utilisation	45
5.6 Axios : Exemple d'utilisation	46
5.7 Finalité d'un ORM	47
5.8 Vue globale de la librairie openapi-enforcer	48
5.9 SourceCode : Représentation des URL de premier niveau	49
5.10 SourceCode : interface de création de comptes	50
5.11 SourceCode : interface de connexion	51
5.12 SourceCode : partie bibliothèque	51
5.13 SourceCode : page bibliothèque	52
5.14 SourceCode : preview d'une ressource informatique	52
5.15 SourceCode : recherche dans la bibliothèque	54

5.16	SourceCode : Ajouter aux favoris depuis le panneau "Filtres"	54
5.17	SourceCode : le panneau "Historique"	55
5.18	SourceCode : le panneau "Favoris"	55
5.19	SourceCode : la représentation d'une fiche	56
5.20	SourceCode : module de gestion (URL)	57
5.21	SourceCode : gestion des ressources informatiques personnelles	58
5.22	SourceCode : créer une ressource informatique	59
5.23	SourceCode : gestion des favoris	61
5.24	SourceCode : création/modification des favoris	62
5.25	SourceCode : profil	63
5.26	SourceCode : partie administration (URL)	63
5.27	SourceCode : administration des ressources informatiques	64
5.28	SourceCode : importation des ressources informatiques	65
5.29	SourceCode : gestion des mots-clés	66
5.30	SourceCode : création/modification des mots-clés	67
5.31	SourceCode : gestion des catégories de mots-clés	68
5.32	SourceCode : gestion des utilisateurs	69

LISTE DES TABLEAUX

2.1	Les différentes plateformes analysées	2
4.1	Problématique de la recherche par mots-clés	33
5.1	Comparatif de quelques frameworks pour le Frontend	42
5.2	Comparatif de quelques frameworks pour l'API	42
5.3	Comparatif de quelques frameworks pour le CLI	43

TABLE DES EXTRAITS DE CODE

5.1	Exemple des fonctionnalités de Lodash	43
-----	---	----

GLOSSAIRE

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) : notre API.
Catégorie de mots-clés	Rubrique contenant des mots-clés de même nature.
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"(Create), "Lire" (Read), "Mettre à jour"(Update), "Supprimer"(Delete). 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.
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.
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.
Métadonnée	Type de données permettant de caractériser et structurer des ressources informatiques.
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.

Terme	Description
OER	Acronyme anglais pour "Open Educational Resources" (dans sa version française, "REL" pour "Ressources éducatives libres"). La définition officielle de l'UNESCO[6] 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 " Qui ? Quoi ? Où ? Quand ? Comment ? Combien ? Pourquoi ?". 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 " R Epresentational S tate T ransfer". Style d'architecture logicielle avec des contraintes/conventions techniques pour concevoir des services communiquant par le web.

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 Rucquoys, Céline Deknop, Christine Jacqmot, Yves Deville, Kim Mens, Guillaume Maudoux, Adrien Widart, Olivier Goletti, Sami Bosch et Isabelle Boulogne, Ophélie Hempel* qui ont pris leur temps pour tester notre prototype et nous fournir de précieux avis et retours constructifs.

PRÉAMBULE

Comme l'illustrent de nombreux médias, l'informatique se révèle être une discipline prometteuse enseignée par de plus en plus d'institutions pédagogiques. Toutefois, la recherche et le référencement d'exercices de programmation de qualité sont une tâche chronophage. Dans ce mémoire, nous allons présenter notre solution/prototype pour pallier ce problème par le biais de notre application web. Nous commencerons par exposer la situation actuelle et les défis à relever. Nous poursuivrons par nos idées, propositions et réalisations en justifiant nos choix technologiques et d'implémentation. Nous traiterons par la suite les différentes validations réalisées sur l'application ainsi que des retours d'utilisateurs ayant contribué à l'outil. Nous conclurons ce mémoire par une analyse critique ainsi que les possibilités futures de notre solution/prototype.

L'ensemble de notre solution/prototype est disponible sur GitHub sous un compte spécialement dédié à ce mémoire : SourceCodeOER (<https://github.com/SourceCodeOER>).

- <https://github.com/SourceCodeOER/sourcecode-front> pour la partie Frontend
- https://github.com/SourceCodeOER/sourcecode_api pour la partie API
- <https://github.com/SourceCodeOER/cli> pour la partie CLI
- <https://github.com/SourceCodeOER/miscellaneous> pour notre base de données de test (pour la validation de notre solution par de vrais utilisateurs) et nos fichiers de configuration paramétrée pour déployer l'ensemble de notre prototype en mode production/démonstration avec Docker Compose et ce n'importe où comme à l'UCLouvain.

Notre solution/prototype est hébergée par l'UCLouvain en mode démonstration à l'adresse suivante : <http://tfe-dewit-yakoub.info.ucl.ac.be/>

CHAPITRE 1

INTRODUCTION

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 [5]). Au-delà de l’aspect théorique, la programmation se découvre aussi par la pratique, ce pourquoi 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**.

À l’instar des *Open Educational Resources (OER)*, notre plateforme offre 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.

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 (voir 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 SourceCode.

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 : SourceCode.

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.

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 (voir section TODO).

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.*



[Main Page](#) → [Problems](#)

Click a category below to view its available problems.

[**Building Java Programs, 5th edition**](#) (637)

[**Building Java Programs, 4th edition**](#) (621)

[**Building Java Programs, 3rd edition**](#) (617)

[**University of Washington CSE 142 \(CS1\)**](#) (336)

Problems used in the CS1 course at Washington, an objects-late introduction to Java.

[**CS1 Sections**](#) (56)

Problems solved during our weekly discussion sessions led by TAs.

[**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)

[**University of Washington CSE 143 \(CS2\)**](#) (427)

[**University of Washington CSE 373**](#) (53)

[**JavaScript**](#) (84)

FIGURE 2.1 – Page principale pour rechercher un problème



[Main Page](#) → [Problems](#) → **Solve a Problem**

[Circle >](#)

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 2.2 – Page d'un problème



[Main Page](#) → [Problems](#) → **Search**

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.](#)

© University of Washington 2019

FIGURE 2.3 – Recherche avancée

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.

The screenshot shows the Hackerrank dashboard. At the top, there's a navigation bar with links for Practice, Compete, Jobs, and Leaderboard, along with a search bar, a 'Hiring developers?' link, and buttons for Log In and Sign Up. Below the navigation is a section titled 'Dashboard' with a 'Your Skills' heading. It features two main cards: 'Interview Preparation Kit' (NEW) and 'Add your first skill'. The 'Interview Preparation Kit' card includes a brief description and a 'View' button. The 'Add your first skill' card includes a brief description and a 'Explore Skills' button. Below these cards is a section titled 'Skills Available For Practice' containing a grid of ten skill categories: Algorithms, Data Structures, Mathematics, C, C++, Java, Python, Ruby, Linux Shell, Functional Programming, Artificial Intelligence, SQL, Databases, and Regex. Each category has a small icon and a brief description.

FIGURE 2.4 – Tableau de bord

The screenshot shows a search results page for Java challenges. At the top, there's a navigation bar with links for Practice, Compete, Jobs, and Leaderboard, along with a search bar, a 'Hiring developers?' link, and buttons for Log In and Sign Up. Below the navigation, it says 'Practice > Java' and the word 'Java' is highlighted. The main content area displays two challenge cards: 'Java Exception Handling (Try-catch)' and 'Java Exception Handling'. Each card shows the title, difficulty level (Easy), max score (10 or 15), success rate (92.87% or 95.07%), and a 'Solve Challenge' button. To the right of the challenges are three filter sections: 'STATUS' (Solved, Unsolved), 'DIFFICULTY' (Easy, Medium, Hard), and 'SUBDOMAINS' (Introduction, Strings, BigNumber, Data Structures, Object Oriented Programming, Exception Handling, Advanced). The 'Exception Handling' subdomain is checked.

FIGURE 2.5 – Page pour rechercher un problème

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:

```
18
3
```

Sample Output 0:

```
3
```

Sample Input 1:

```
18
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](#)

FIGURE 2.6 – Page d'un challenge

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.

The screenshot shows the Leetcode homepage with the following sections:

- Announcement:** Due to some upstream issue, users won't receive any email notifications from the discuss posts. Sorry for the inconvenience!
- Category - All:** Categories include Algorithms (orange), Database (blue), Shell (green), and Concurrency (purple, marked as New).
- Solved Status:** 0/1411 Solved (Easy 0, Medium 0, Hard 0). A "Pick One" button is available.
- Search Bar:** Search question titles, description or IDs.
- Difficulty, Status, Lists, Tags filters.**
- Problem List:** A table of 14 problems with columns: #, Title, Solution, Acceptance, Difficulty, Frequency, and Lock icon.
- Challenges:** Daily Coding Challenge (30-Day LeetCoding Challenge, Day 14).
- Contests:** Weekly Contest 185 (Sunday, April 19th, 4:30am-6:00am local time) and New Biweekly Contest (14:30 UTC every other Saturday, Starting June 1, 2019).
- Your Progress:** Session: Anonymous Session.

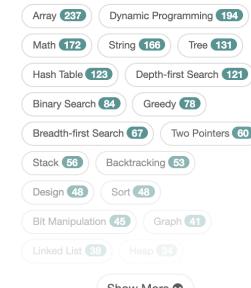
FIGURE 2.7 – Page de recherche de problèmes

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.

1 2 3 4 5 >

Topics



Companies

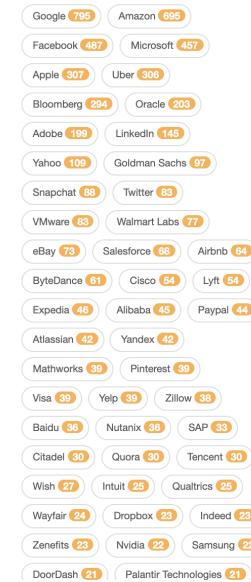


FIGURE 2.8 – Quelques mots-clés disponibles sur le côté droit

The screenshot shows a LeetCode problem page for "31. Next Permutation". The page includes the problem title, difficulty level (Medium), acceptance rate (336,132 submissions, 1,051,767 total), and a code editor with sample JavaScript code. The code is as follows:

```

1  /*
2  * @param {number[]} nums
3  * @return {void} Do not return anything, modify nums in-place instead.
4  */
5  var nextPermutation = function(nums) {
6  };
7

```

The page also features navigation links like "Problems", "Pick One", "Prev", "Next", "Console", "Contribute", "Run Code", and "Submit".

FIGURE 2.9 – Page d'un problème

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
 - Concurrence
- 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.

The screenshot shows the Codeforces website's problem search interface. At the top, there's a navigation bar with links for HOME, TOP, CONTESTS, GYM, PROBLEMSET, GROUPS, RATING, API, CALENDAR, HELP, and a 10 YEARS! link. A search bar is also present. On the left, a sidebar lists categories: MAIN, ACMGURU, PROBLEMS (which is selected), SUBMIT, STATUS, STANDINGS, and CUSTOM TEST. The main content area displays a table of problems:

#	Name	Tags	Rating	Solver Count
1339B	Sorted Adjacent Differences	constructive algorithms, sortings	1200	x11498
1339A	Filling Diamonds	brute force, dp, implementation, math	1000	x13824
1338E	JYPnation	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

On the right side, there are two boxes: "Pay attention" which lists upcoming contests, and "Filter Problems" which allows users to filter by difficulty and apply tags.

FIGURE 2.10 – 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	<input type="button" value="Copy"/>
2	
6	
5 -2 4 8 6 5	
4	
8 1 4 2	
output	<input type="button" value="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 have fun and learn 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 2.11 – Page d'un challenge

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)

The screenshot shows the CodeChef website's practice challenges section. At the top, there is a navigation bar with links for PRACTICE, COMPETE, DISCUSS, COMMUNITY, HELP, and ABOUT, along with social media icons for Google+, Twitter, and Facebook, and a 'New User' button. Below the navigation is a search bar with fields for Username and Password, and a 'Login' button. A 'Forgot Password' link is also present.

The main content area is titled 'Practice' and features a sub-section titled 'This is the place to hone your skills.' It includes an illustration of a chef wearing a toque blanche and holding dumbbells labeled 'JAVA' and 'AXX'. Below this, there is a brief description of the challenge: '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.'

Below this section is a table listing various challenges:

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 2.12 – Page de recherche de challenges

FIGURE 2.13 – Page de filtres

This problem was part of the [CodeChef April Challenge](#). All user submissions for this contest problem are publicly available [here](#).

Shaheen has bought some gifts for a friend, which are wrapped up in several boxes of different sizes (all of which are full). She will need to carry the gifts a long way to her friend's house, so she would prefer to add some extra packing, and accommodate everything in one extra box. Moreover, to avoid damaging the contents, she does not wish to place more than two boxes directly within any box; however, boxes can be placed within boxes which contain other boxes, etc. A box which is used for holding two boxes of sizes a and b will have size $a+b$, and will cost Shaheen $a+b$ coins at the local store. Please help Shaheen determine the minimum cost required to achieve the complete packing, and the number of different possible packings (arrangements of boxes within each other) having such a minimal cost.

Input

The first line of input is $n \leq 2000$, the number of boxes with Shaheen's gifts. The next n lines of input contain one positive integer each, not greater than 10^6 , representing the sizes of the successive boxes.

Output

Print to output exactly 2 numbers separated by spaces: the cost of the optimal solution, and the number of distinct ways of achieving this solution (modulo 10^9).

Example

Input:
5
3

All Submissions
Successful Submissions

FIGURE 2.14 – Page d'un challenge

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.

The screenshot shows the Coderbyte Challenges page. At the top, there is a navigation bar with links for Challenges, Courses, and Coderbyte for Employers, along with Upgrade and Register/Login buttons. Below the navigation bar, the title "Challenges" is displayed, followed by a teal button labeled "UPGRADE TO UNLOCK CHALLENGES". The main content area lists various challenges:

- Find Intersection**: Not Completed | Easy | Solutions: 5890. Buttons: View all user solutions, Discussion.
- Questions Marks**: Not Completed | Easy | Solutions: 11766. Buttons: View all user solutions, Discussion.
- Binary Reversal**: Not Completed | Easy | Solutions: 2494. Buttons: View all user solutions, Discussion.
- Letter Changes**: Not Completed | Easy | Solutions: 121826. Buttons: View all user solutions, Discussion.
- Longest Word**: Not Completed | Easy | Solutions: 118070. Buttons: View all user solutions, Discussion.
- First Factorial**: Not Completed | Easy | Solutions: 177177. Buttons: View all user solutions, Discussion.
- First Reverse**: Not Completed | Easy | Solutions: 192651. Buttons: View all user solutions, Discussion.
- NEW React Button Toggle**: Not Completed | Easy | Solutions: 1355. Buttons: View all user solutions, Discussion.
- NEW SQL Member Count**: Not Completed | Medium | Solutions: 1460. Buttons: View all user solutions, Discussion.
- NEW Bash Print Memory**: Not Completed | Medium | Solutions: 1. Buttons: View all user solutions, Discussion.

A purple banner at the bottom of the challenge list says "FLASH SALE! Get 30% off any premium plan!"

On the right side, there is a sidebar titled "Filters" with sections for DIFFICULTY (Easy 111, Medium 96, Hard 61), COMPANIES (Google 46, Facebook 24, Amazon 12, Microsoft 21), and TAGS (string manipulation 64, free 9, recursion 13, math fundamentals 59, searching 102, regular expression 13, sorting 8, array 75, sequences 7, dynamic programming 20).

FIGURE 2.15 – Page de recherche de challenges

The screenshot shows the details for the "Find Intersection" challenge. At the top, there are tabs for easy, Solutions, and Discussion. The challenge 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`".

Below the description, there are Examples and Tags:

Examples

```
Input: ["1, 3, 4, 7, 13", "1, 2, 4, 13, 15"]
Output: 1,4,13
```

```
Input: ["1, 3, 9, 10, 17, 18", "1, 4, 9, 10"]
Output: 1,9,10
```

Tags

array, free

The main area contains a code editor with the following JavaScript code:

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

Below the code editor, there are buttons for Run Code, Run Test Cases, and Submit. A log window shows the output: "1, 3, 4, 7, 13", "1, 2, 4, 13, 15". There are also Clear Log and Auto-clear buttons.

FIGURE 2.16 – Page d'un challenge

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 **SourceCode**.

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 **SourceCode** 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 **SourceCode**. 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 priviléges 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 SourceCode** mais plutôt une mise sur papier de notre conception de l'application web au vu des critères exposés précédemment.

<SourceCode/>

rechercher

Ressource d'exercices

Résultats : 42

Langage

Java Python C

Difficulté

Facile Difficile

Cotation

> 3 étoiles

ajouter un filtre

White loop

4,7 ★

Facile | Java | QCM | Listes chaînées

[consulter](#)

Introduction to recursion

4 ★

Facile | Python | Fill the code | Récursion

[consulter](#)

Les pointeurs à gogo

3,9 ★

Difficile | C | Fill the code | Pointeurs

[consulter](#)

Le défi I

3 ★

Difficile | C | projet | Réseau

[consulter](#)

Le défi II

3 ★

Difficile | C | QCM | pointeurs

[consulter](#)

FIGURE 2.17 – La page Bibliothèque

<SourceCode/>

rechercher

Tous les filtres

effacer annuler [appliquer](#)

Langage	Difficulté	Type d'exercice	Plateforme
<input type="button" value="Rechercher un filtre"/>	<input type="button" value="Rechercher un filtre"/>	<input type="button" value="Rechercher un filtre"/>	<input type="button" value="Rechercher un filtre"/>
<input type="checkbox"/> Java <input type="checkbox"/> Python <input type="checkbox"/> Scala <input type="checkbox"/> C <input type="checkbox"/> Non défini	<input type="checkbox"/> Facile <input type="checkbox"/> Normal <input type="checkbox"/> Difficile	<input type="checkbox"/> QCM <input type="checkbox"/> Exercice papier <input type="checkbox"/> projet <input type="checkbox"/> Fill the code <input type="checkbox"/> Mission	<input type="checkbox"/> Inginious <input type="checkbox"/> Facebook <input type="checkbox"/> Google <input type="checkbox"/> Non défini
Auteur	Cotation	Thématische	Institution
<input type="button" value="Rechercher un filtre"/>	<input type="radio"/> 5 étoiles <input type="radio"/> plus de 4 étoiles <input type="radio"/> plus de 3 étoiles <input type="radio"/> moins de 3 étoiles <input checked="" type="radio"/> par défaut	<input type="button" value="Rechercher un filtre"/>	<input type="button" value="Rechercher un filtre"/>
<input type="checkbox"/> Kim Jong Un <input type="checkbox"/> Rambo <input type="checkbox"/> Mark Zuckerberg <input type="checkbox"/> John Doe <input type="checkbox"/> Anonyme		<input type="checkbox"/> Paradigmes de programmation <input type="checkbox"/> Listes chaînées <input type="checkbox"/> Pointeurs <input type="checkbox"/> Récursion <input type="checkbox"/> Boucles	<input type="checkbox"/> UCLouvain <input type="checkbox"/> ULB <input type="checkbox"/> UHasselt <input type="checkbox"/> IPET

FIGURE 2.18 – Représentation du système de filtres avec les mots-clés

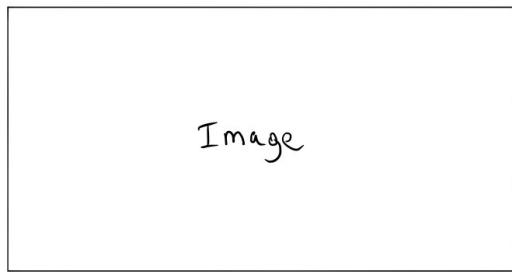


While loop

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 Kim Jong Un

Difficulté Facile

Type d'exercice QCM

Langage Java

Thématique Listes charnées

Plateforme Inginious

Institution UCLouvain

Source[↓ Télécharger la source](#)[↗ Lien vers la source](#)**Votre cotation**

★ ★ ★ ★ ★

FIGURE 2.19 – 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 introductory à l'analyse fonctionnelle relatée dans la section 4.1, mais nous avions besoin de cette étape pour avoir une vue au global de SourceCode 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.

SourceCode 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 (voir figure 3.3), nous devions 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'évoquerons 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 rencontrions 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 6.1 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 TODO), 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-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 issues². (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 informe de l'avancement général du projet
- Avec les issues (figure 3.2), chaque tâche/problème dispose des éléments suivants :

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>

- 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)

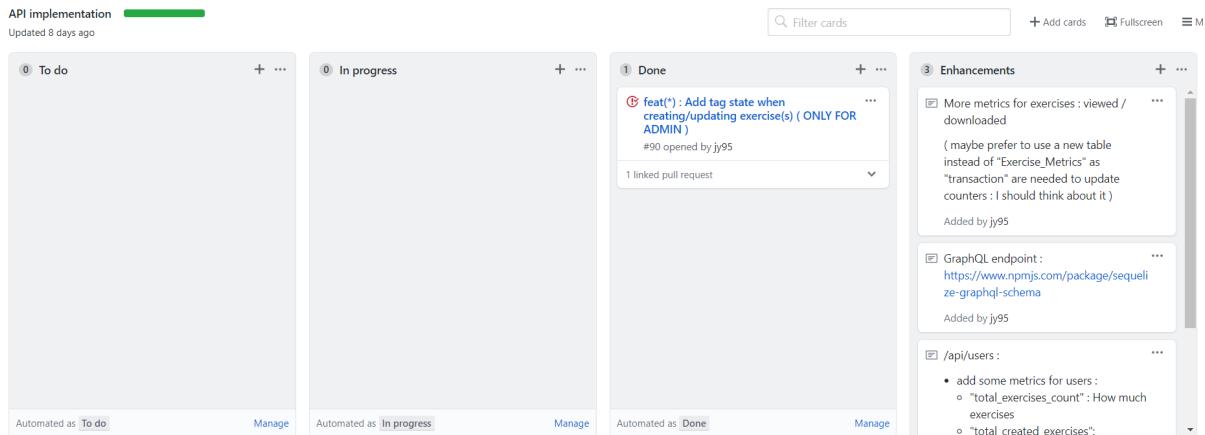


FIGURE 3.1 – Exemple d’interface Github Project Board

<input type="checkbox"/>	fix(police): bug in safari for rendering font	bug	
	#8 by dewita was closed on 12 Dec 2019		
<input type="checkbox"/>	feat(FilterPanel): add possibility to choose all the filters in one category	enhancement	
	#7 by dewita was closed on 2 Jan		
<input type="checkbox"/>	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		
<input type="checkbox"/>	feat(Exercise): add rating system when user is connected	enhancement	
	#4 by dewita was closed on 5 Jan		
<input type="checkbox"/>	docs: explanation of the project setup	documentation	
	#3 by dewita was closed on 12 Dec 2019		
<input type="checkbox"/>	feat(ExercisePanel): true data on the window	enhancement	
	#2 by dewita was closed on 26 Nov 2019		
<input type="checkbox"/>	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.

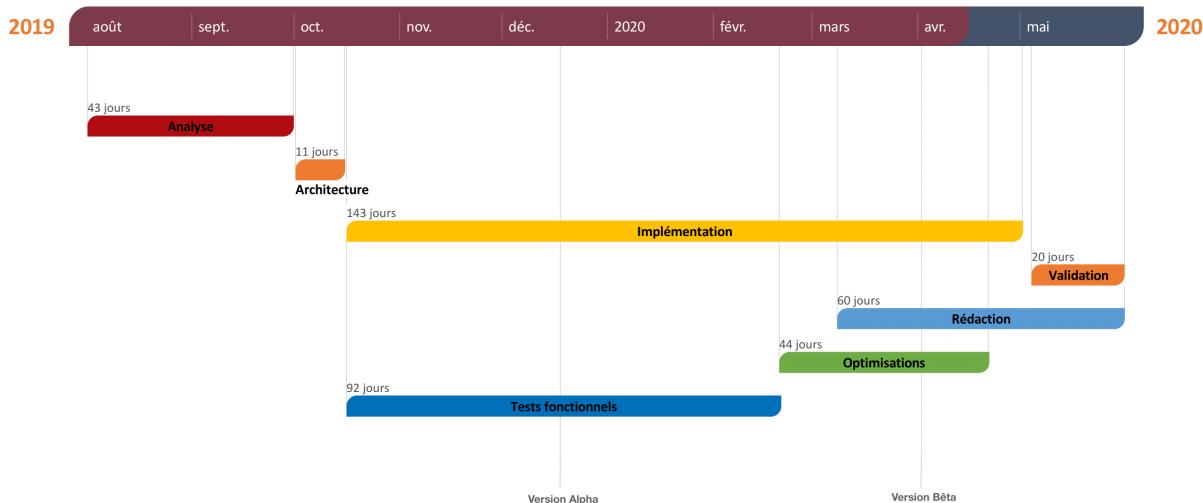


FIGURE 3.3 – Diagramme de Gantt simplifié

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 A).

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

- 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 à 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 dénoué 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 QQOQCCP 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 A).

4.1 Analyse fonctionnelle

Les différents types d'utilisateurs

SourceCode 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 priviléges 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***[8]. Celle-ci consiste à classifier 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" (**luxe**)

À titre informatif, le choix du **JSON** se justifie par la norme que nous avons élaborée (cf annexe A), 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
- 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
- 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 (des restrictions que nous détaillerons dans la section TODO sont présentes pour éviter des dérives)

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
- Créer automatiquement les mots-clés non existants dans le système (avec éventuellement un état de départ)
- 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)

MModifier un mot-clé

MModifier une catégorie de mots-clés

MChanger l'état d'un ou plusieurs mot(s) clé(s)

MModifier les informations d'une fiche

MChanger l'état d'une fiche (aucune restriction)

MCréer 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 :

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

Nous reviendrons d'ailleurs ultérieurement (cf section TODO, figure 5.15, ...) sur les façons dont nous avons mis en place cette réflexion dans l'implémentation de **SourceCode**.

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. Le diagramme UML à états ci-dessous représente ces états et les principales transitions entre états (pour ne pas surcharger celui-ci) :

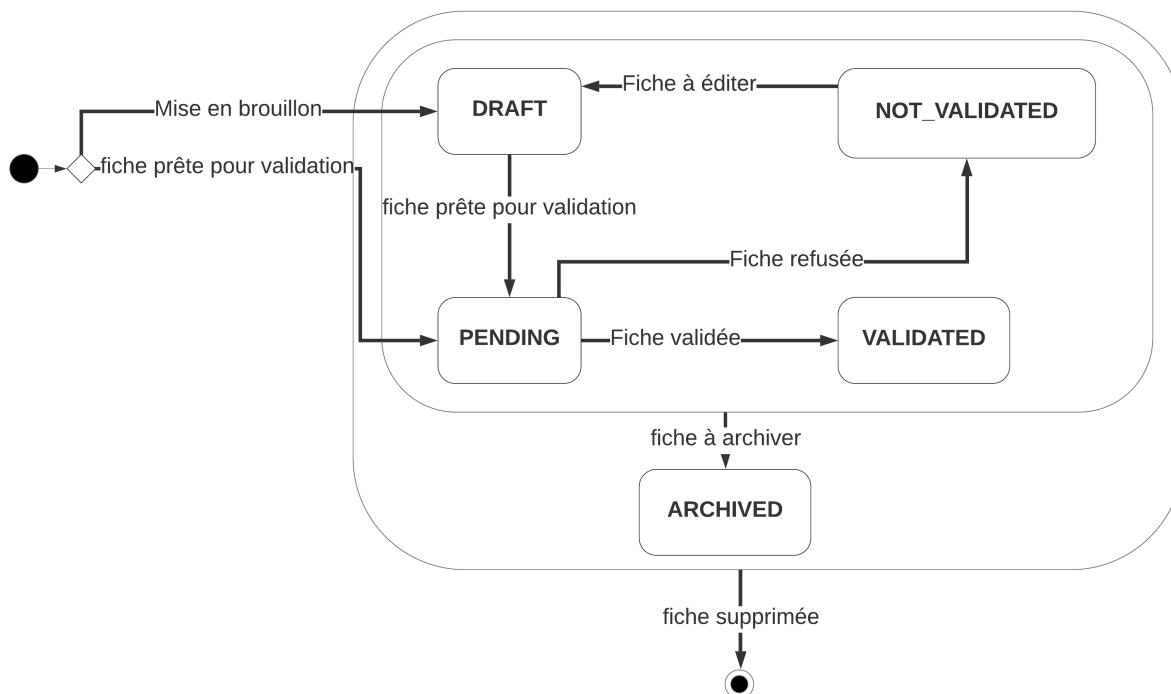


FIGURE 4.1 – Diagramme UML à états pour l'état d'une fiche

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. Le diagramme UML à états ci-dessous représente ces états et les principales transitions entre états (pour ne pas surcharger celui-ci) :

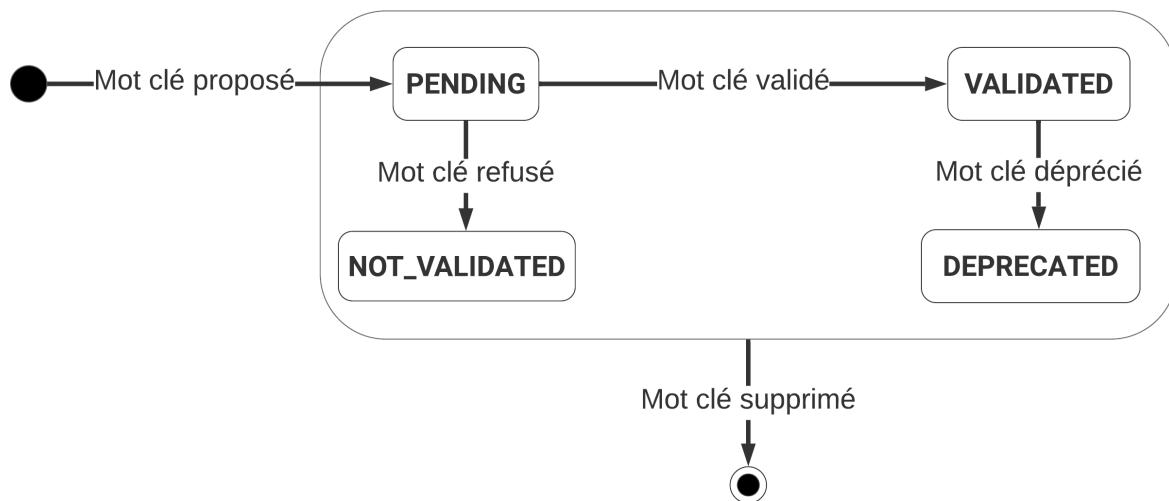


FIGURE 4.2 – Diagramme UML à états pour l'état d'un mot-clé

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 QOQCCP). 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 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 *Pr. 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 "Authentication/Authorization/Accounting", 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 **SourceCode**.

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 (par souci de clarté, l'abréviation **MAJ** désigne le terme "mise(s) à jour") :



FIGURE 4.3 – Propriétés **ACID** [3]

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

⁵<https://www.postgresql.org/>

⁶https://fr.wikipedia.org/wiki/Propri%C3%A9t%C3%A9s_ACID

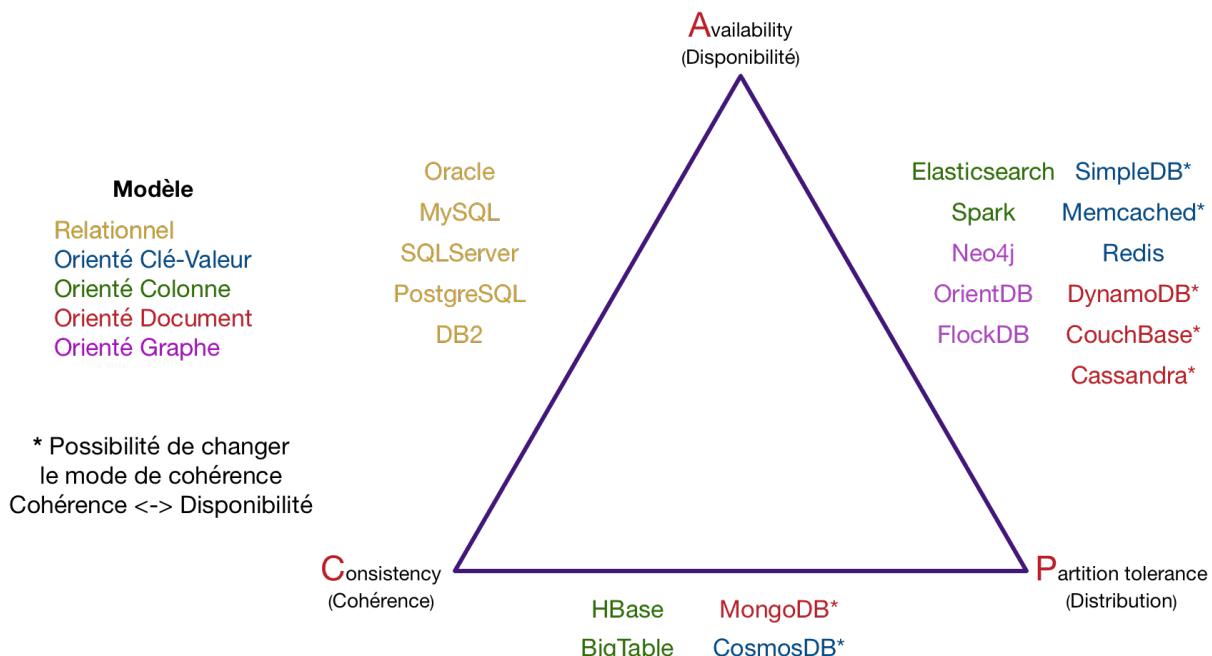
Étant donné que ces propriétés sont assez contraignantes, le lecteur attentif (ou lectrice attentive) que vous êtes est en droit de remettre en question cette contrainte technique et il est donc utile de recourir au théorème **CAP** pour la justifier. Comme expliqué par un article [2], chaque lettre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

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

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 justifier l'obligation de PostgreSQL par rapport à des solutions alternatives, comme le montre la figure 4.4 :



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 plate-forme 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.

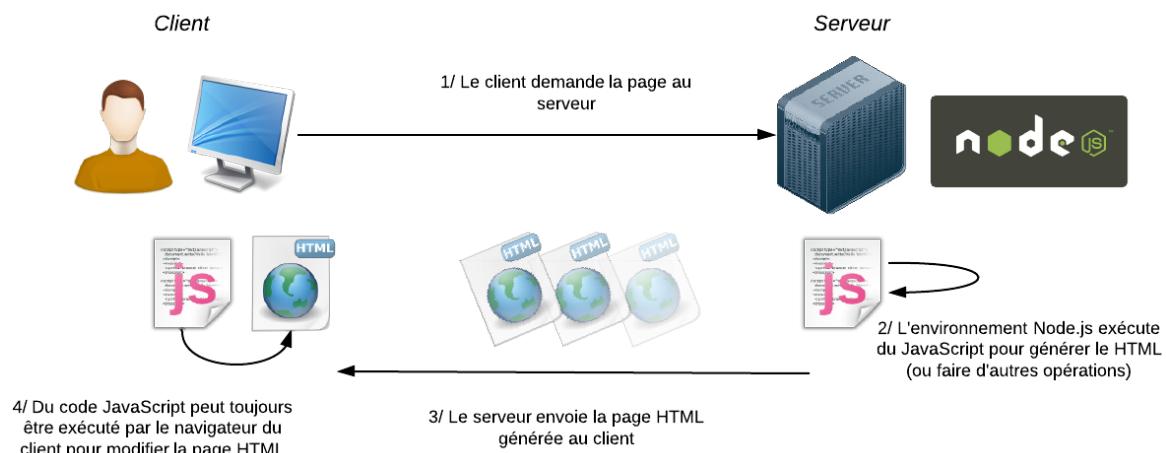


FIGURE 5.1 – Du JavaScript aussi bien côté serveur que client [4]

Deux points sont élémentaires pour expliquer l'intérêt et la rapidité de Nodejs : 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 :

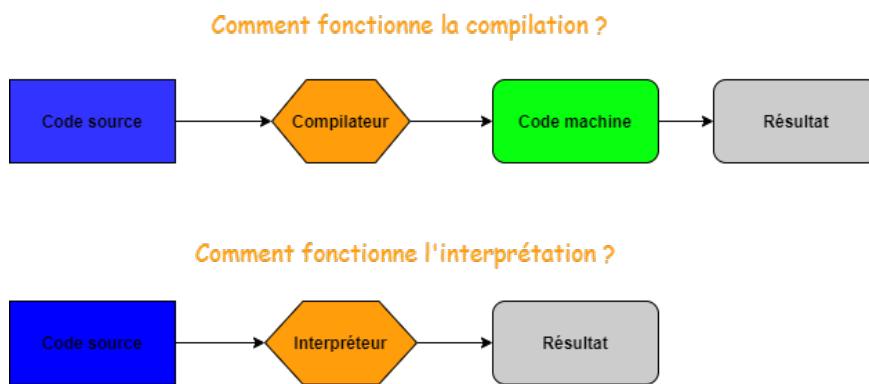


FIGURE 5.2 – Compilation et interprétation - mode de fonctionnement

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.

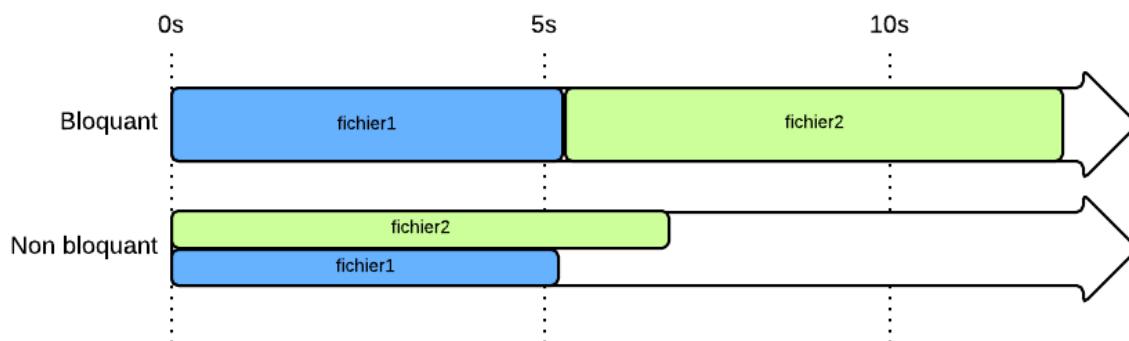


FIGURE 5.3 – Bloquant/non bloquant : un exemple pour l'illustrer [4]

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 perçu 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 aisément 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 pour le CLI

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

Comme expliqué par Wikipédia[7], le terme librairie désigne "une collection de fonctions utilitaires prêtes à être utilisées par des programmes". En effet, 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 = _.clonedepth(autres);
12 // etc...
```

Listing 5.1 – Exemple des fonctionnalités de Lodash

5.1.0.0.2 Librairies pour le Frontend

VeeValidate

Un excellent framework, construit pour VueJS, responsable de la validation des différents formulaires présents sur [SourceCode](#). 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 :

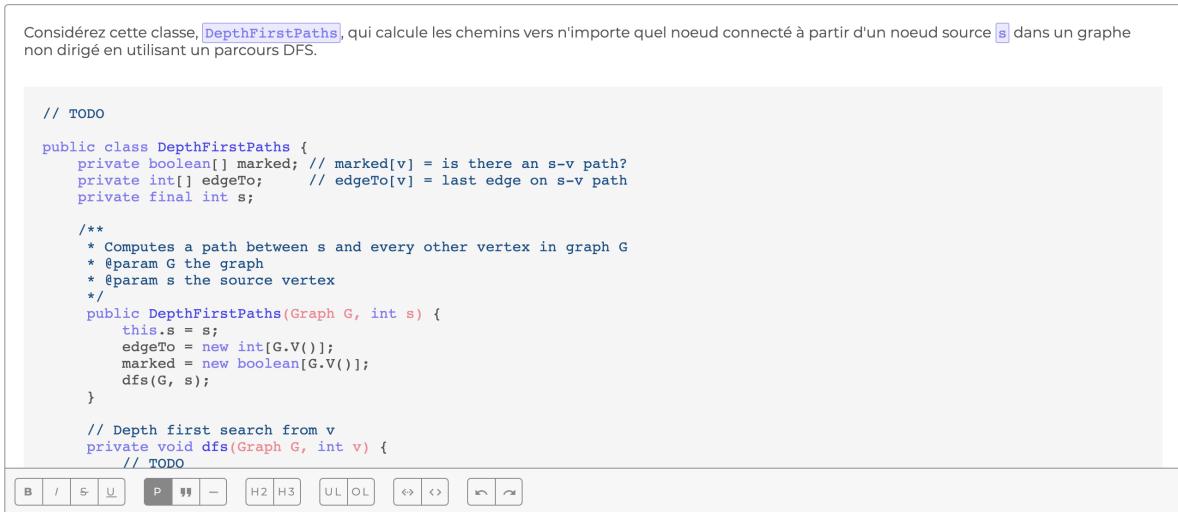


FIGURE 5.5 – Exemple d'un éditeur de texte

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 (voir 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 :

```

try {
  const response: SearchExerciseResponse = await this.app.$axios.$post( url: '/api/search', newSearchRequest);
  commit('INIT', response);
} catch(e) {
  const errorAxios = e as AxiosError;

  if (errorAxios.response) {
    const status: number = errorAxios.response.status;

    if (status === 400) {
      this.$displayError( injectee: "Une erreur est survenue lors du chargement des exercices.", payload: {statusCode: status});
    } else if (status === 401) {
      this.$displayError( injectee: `Vous devez vous connecter pour charger ces exercices.`, payload: {statusCode: status});
    } else if (status === 403) {
      this.$displayError( injectee: `Vous n'êtes pas autorisé à effectuer cette action.`, payload: {statusCode: status});
    } else if (status === 500) {
      this.$displayError( injectee: `Une erreur est survenue depuis nos serveurs, veuillez-nous en excuser.`, payload: {statusCode: status});
    } else {
      this.$displayError( injectee: "Une erreur est survenue lors du chargement des exercices.", payload: {statusCode: status});
    }
  } else {
    this.$displayError( injectee: `Une erreur est survenue lors du chargement des exercices.`, payload: {statusCode: 400});
  }

  commit('RESET')
}

```

FIGURE 5.6 – Exemple 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 Librairies 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 librairie Sequelize qui propose du mapping objet-relationnel² (à titre informatif, d'autres alternatives similaires existent, notamment dans Loopback que nous avions 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 vons proposons d'ailleurs une illustration récapitulatif en figure 5.7.

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).

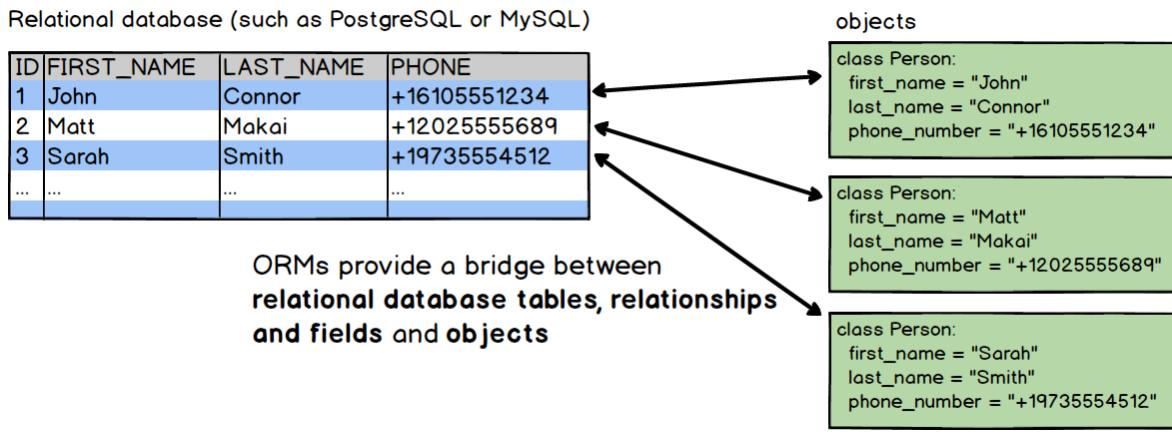


FIGURE 5.7 – Finalité d'un ORM

De ce fait, cela permet de simplifier les interactions 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**⁵. 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 5.8 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-enforcer⁶ : le cœur de la librairie
- openapi-enforcer-middleware⁷ : le middleware de la librairie
- openapi-enforcer-multer⁸ : un middleware optionnel qui s'occupe automatique de l'upload de fichiers (cf section 4.1)

⁵Brigham Young University - <https://www.byu.edu/about>

⁶<https://github.com/byu-oit/openapi-enforcer>

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

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

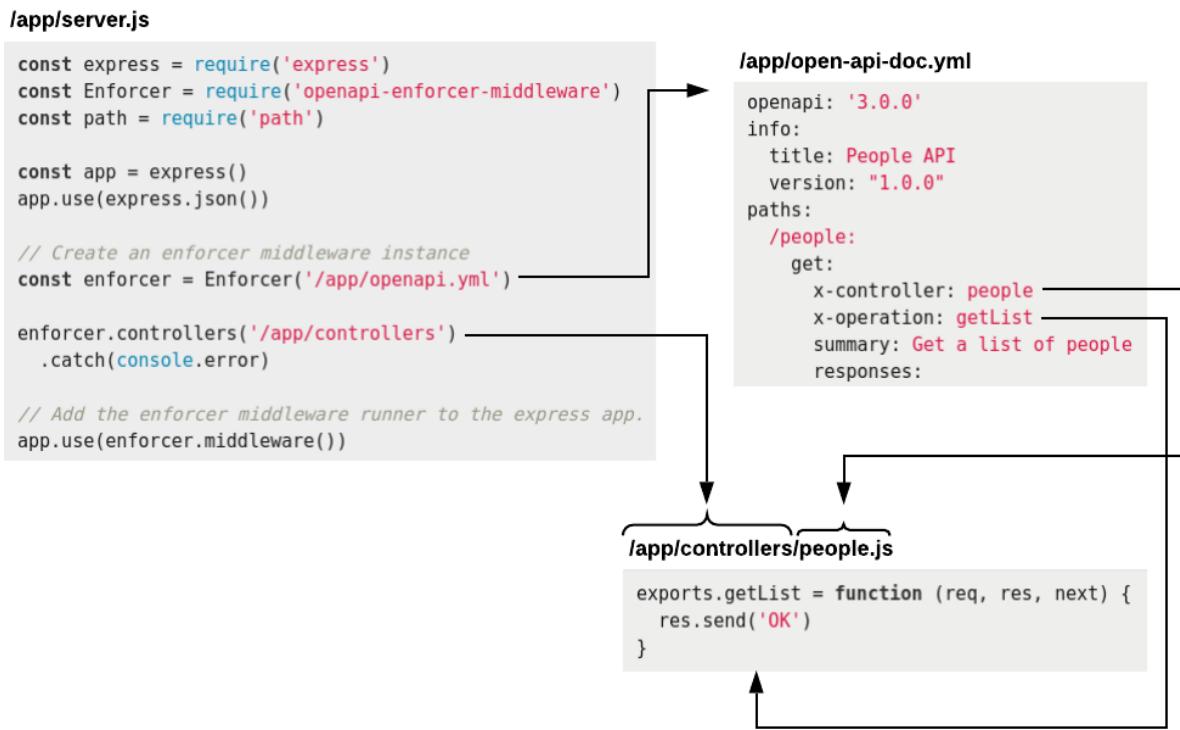


FIGURE 5.8 – Vue globale de la librairie openapi-enforcer⁹

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-json¹⁰ : pour authentifier l'utilisateur lors de sa première visite
- passport-jwt¹¹ : pour implémenter un mécanisme d'authentification **JWT**¹², 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 : Jest¹³. (Nous reviendrons sur la dimension des tests de manière plus approfondie dans le chapitre ???(validation))

⁹<https://byu-oit.github.io/openapi-enforcer-middleware/getting-started#setup>

¹⁰<https://github.com/JJamesMGreene/passport-json>

¹¹<http://www.passportjs.org/packages/passport-jwt/>

¹²<https://jwt.io/>

¹³<https://jestjs.io/>

5.2 Client

Cette section est entièrement dédiée à la conception du Frontend de **SourceCode**. 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 **SourceCode**.



FIGURE 5.9 – 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.

Attention : Lorsqu'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`.

Les deux premières URL de la figure désignent les interfaces de connexion à **SourceCode** 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 **SourceCode** 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 priviléges sur **SourceCode**. 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 **SourceCode**. 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 **SourceCode**. Une liste plus exhaustive des fonctionnalités à intégrer se trouve en section 4.1.

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.

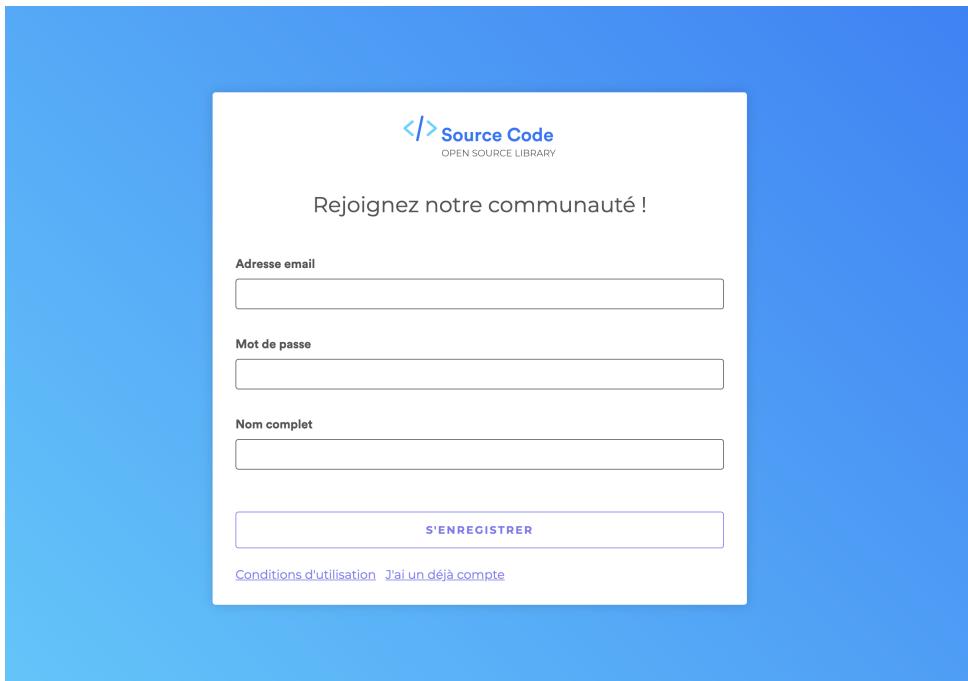


FIGURE 5.10 – Interface de création de comptes

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.

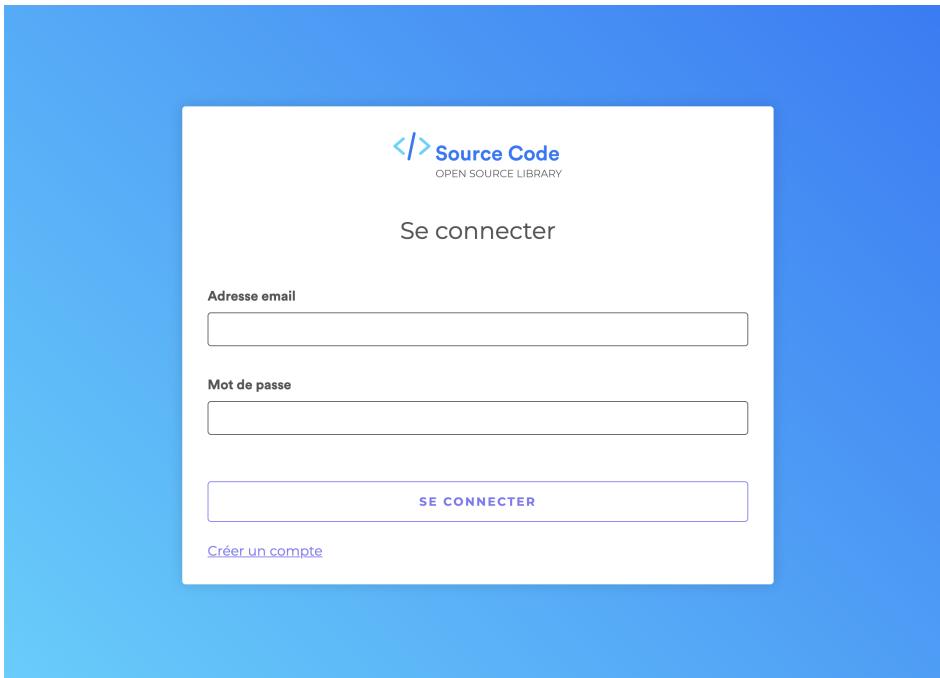


FIGURE 5.11 – Interface de connexion

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 (voir section ???) mais nous avions surtout choisi ce timing pour la séance de validation de notre plateforme (section ???).

5.2.2 Exercices

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



FIGURE 5.12 – 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 priviléges utilisateurs sur cette page ainsi que les choix ergonomiques.
- La fiche d'une ressource informatique (accessible depuis l'URL */exercices/_id*) : 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 (voir section 2.1). Il était important pour nous de conserver une interface simple et déjà utilisée par certaines applications web.

The screenshot shows the 'Bibliothèque' (Library) page. On the left, there's a sidebar with links for 'se connecter', 'Accueil', 'Bibliothèque' (which is the active tab), and 'Tutorial'. Below these are buttons for 'crée un compte' and 'Source Code' with a 'OPEN SOURCE LICENSE' link. The main area has a search bar with placeholder 'Rechercher' and a 'filtres' button. A message says '202 résultats - réinitialiser la recherche'. A list of exercises follows:

- Median** [LSINF121] Algorithmique et structures de données | Java | code [VOIR L'EXERCICE](#)
- Union find** [LSINF121] Algorithmique et structures de données | match [VOIR L'EXERCICE](#)
- Trier une liste chainée** [LSINF1252] Systèmes informatiques | C | code | pointeurs [VOIR L'EXERCICE](#)
- strlen, strcat et strcasecmp** [LSINF1252] Systèmes informatiques | C | code | pointeurs [VOIR L'EXERCICE](#)
- Traduction de code assembleur** [LSINF1252] Systèmes informatiques | C | code [VOIR L'EXERCICE](#)
- Bubble Sort Invariant (MCQ)** [LEPL1402] Informatique 2 | QCM [VOIR L'EXERCICE](#)
- Printing data** [LSINF1252] Systèmes informatiques | C | code [VOIR L'EXERCICE](#)
- Les conditions** Algorithmes et recettes de cuisine | match [VOIR L'EXERCICE](#)
- Merge Sort (implémenté)** [LSINF121] Algorithmique et structures de données | Java | code [VOIR L'EXERCICE](#)

FIGURE 5.13 – page bibliothèque

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.

This screenshot shows a preview of the 'Improved strcpy' exercise. It includes the exercise title, its category, and a 'VOIR L'EXERCICE' button. Below it, another exercise is partially visible.

Heap	[LSINF121] Algorithmique et structures de données match	VOIR L'EXERCICE
Improved strcpy	[LSINF1252] Systèmes informatiques C code malloc	VOIR L'EXERCICE

FIGURE 5.14 – Preview d'une ressource informatique

Le preview d'une ressource contient les informations suivantes :

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

Les autres mots-clés seront consultables depuis l'interface de la fiche (voir section 5.2.2.2)

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

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 5.15, le panneau de filtres n'est plus à la même position que ce que nous avions imaginé avec notre patchwork 2.2. Dans ce dernier, nous avions 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 avions 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 ?? ?.

The screenshot shows the Source Code library interface. On the left, there's a sidebar with navigation links like Accueil, Bibliothèque, Tutoriel, Administration, Gestion, and Mon profil. The main area has a search bar at the top with a 'Rechercher' button. Below it, a search result summary says '33 résultats - réinitialiser la recherche'. A 'Java' filter is selected. The results list includes items like 'Median', 'Merge Sort (implém.)', 'Connected Components', etc., each with a 'VOIR L'EXERCICE' button. On the left, a 'filtres' panel is open, showing various categories like Platforme, Cours, Type D'exercice, Source, and Langage. Under 'Langage', 'Java' is checked. Other languages listed include C, Makefile, Javascript, HTML/CSS, TypeScript, R, C++, Go, and Erlang.

FIGURE 5.15 – Recherche dans la bibliothèque

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.

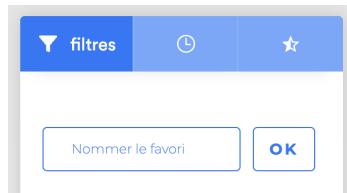


FIGURE 5.16 – Ajouter aux favoris depuis le panneau "Filtres"

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.

The screenshot shows the 'Historique' (History) panel. On the left is a sidebar with navigation links: Accueil, Bibliothèque, Tutoriel, Administration, Gestion, and Déconnexion. The main area has a header 'Bibliothèque' with a search bar. Below it, a message says '33 résultats - réinitialiser la recherche'. A 'Java' tag is selected. The results list contains 10 items, each with a title, a brief description, and a 'VOIR L'EXERCICE' button.

Titre	Description	Action
Median	[LSINF12] Algorithmique et structures de données Java code	VOIR L'EXERCICE
Merge Sort (implen)	[LSINF12] Algorithmique et structures de données Java code	VOIR L'EXERCICE
Connected Components	[LSINF12] Algorithmique et structures de données Java code	VOIR L'EXERCICE
Incremental Hash	[LSINF12] Algorithmique et structures de données Java code	VOIR L'EXERCICE
Global Warming (implen)	[LSINF12] Algorithmique et structures de données Java code	VOIR L'EXERCICE
Circular LinkedList	[LSINF12] Algorithmique et structures de données Java code	VOIR L'EXERCICE
Digraph (implen)	[LSINF12] Algorithmique et structures de données Java code	VOIR L'EXERCICE
Global Warming	[LSINF12] Algorithmique et structures de données Java code	VOIR L'EXERCICE
Binary Search Tree (implen)	[LSINF12] Algorithmique et structures de données Java code	VOIR L'EXERCICE

FIGURE 5.17 – Le panneau "Historique"

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.

The screenshot shows the 'Favoris' (Favorites) panel. The sidebar includes 'Bibliothèque' and 'favoris'. The main area shows a search bar with 'Rechercher' and a message '10 résultats - réinitialiser la recherche'. It lists 10 results from 'John Aoga' and 'Java', with a 'Java' tag selected. Each result has a 'VOIR L'EXERCICE' button.

Titre	Description	Action
Global Warming (implen)	[LSINF12] Algorithmique et structures de données Java code	VOIR L'EXERCICE
Binary Search Tree (implen)	[LSINF12] Algorithmique et structures de données Java code	VOIR L'EXERCICE
PART 6 : Maze (implen)	[LSINF12] Algorithmique et structures de données Java code	VOIR L'EXERCICE
Connected Components (implen)	[LSINF12] Algorithmique et structures de données Java code	VOIR L'EXERCICE
Global Warming (implen)	[LSINF12] Algorithmique et structures de données Java code	VOIR L'EXERCICE
Word Transformation Shortest Path (implen)	[LSINF12] Algorithmique et structures de données Java code	VOIR L'EXERCICE
PART 6 - Breadth First Paths (implen)	[LSINF12] Algorithmique et structures de données Java code	VOIR L'EXERCICE
Binary Search Tree Iterator (implen)	[LSINF12] Algorithmique et structures de données Java code	VOIR L'EXERCICE
Depth First Paths (implen)	[LSINF12] Algorithmique et structures de données Java code	VOIR L'EXERCICE

FIGURE 5.18 – Le panneau "Favoris"

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 (voir section 5.2.3.2).
- **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 **SourceCode**. 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 :

The screenshot shows the SourceCode platform's interface for viewing a resource. On the left is a vertical sidebar with a navigation menu for users (Eric Cartman), including sections like Accueil, Bibliothèque, Tutoriel, Administration, Exercices, Importer, Catégories, Tags, Utilisateurs, Gestion, Mes exercices, Mes favoris, and Mon profil. At the bottom of the sidebar is a Déconnexion link. The main content area has a header 'Exercices > Connected Components (implém)' and a 'Retour à la recherche' link. The central part of the screen displays the resource details for 'Connected Components (implém)'. It includes a 'détails' button, the author (psc, John Aoga), course information ([LSINF112]), language (Java), license (CC-BY-NC-SA-4.0), platform (INGINIOUS), source (https://github.com/UCL-INGI/LSINF112-Data-Structures-And-Algorithms), and type (match code). Below this, there is a code editor showing Java code for implementing connected components. A 'MODIFIER L'EXERCICE' button is located in the top right corner of the main content area. The code in the editor is:

```
public class ConnectedComponents {
    /**
     * return the number of connected components in g
     */
    public static int numberOfConnectedComponents(Graph g) {
        // TODO
        return 0;
    }
}
```

At the bottom of the code editor, it says 'Le projet IntelliJ est disponible ici.'

FIGURE 5.19 – 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 (voir section 5.2.3.1 pour la gestion côté utilisateur ou voir section 5.2.4.1 pour la gestion côté (super-)administrateur).

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'invalider la ressource ou de la laisser passer, sans grande chance d'être retrouvée par d'autres utilisateurs de la plateforme.

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.

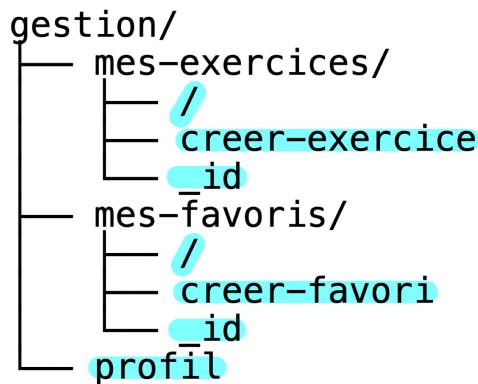


FIGURE 5.20 – Module de gestion (URL)

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.

Titre	Nb de votes	Moyenne	Mis à jour	Fichier(s)	Status
Liste doubllement 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	✗	✓
FLList 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 5.21 – 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

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 A) :

- **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.
- ✗ **Invalidé**
 - 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'un ressource informatique se retrouve en figure 4.1.

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 apparaitra à côté du bouton de création d'exercices avec les options suivantes :

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

Rechercher		PLUS D'ACTIONS		CRÉER UN EXERCICE +	
Titre	Nb de votes	Moyenne	Mis à jour	Fichier(s)	Status
<input checked="" type="checkbox"/> Liste doublement chaînée	1	3	26/04/20 à 9:53	✗	✓

FIGURE 5.22 – Gestion des ressources informatiques personnelles (creer-exercice ou __id)

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)

Créer un exercice

Titre *
Entrez le nom de votre exercice

Description *
Décrivez l'énoncé de votre exercice et expliquez ce que contient votre archive zip (si présente). Un énoncé bien écrit aura bien plus de retours positifs qu'un descriptif vaste et sans explication pour utiliser le contenu de votre archive.

Tags *
COMMENCER LA SÉLECTION

Un tag **ne figure pas** dans la liste ?
Pas de problème, vous pouvez toujours nous en proposer [ici](#) !

Code couleur

- valide
- en attente
- obsolète
- non valide

Uploadez votre archive (zip)

Url vers l'exercice
Entrez l'url absolue vers votre 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.

12 résultat(s)

Binary Search Tree
Nb de votes : 0
Moyenne : -

Binary Search Tree Iterator (implém)
Nb de votes : 0
Moyenne : -

Reading integers in a binary file
Nb de votes : 0
Moyenne : -

Soumission de la ressource

Deux options de soumission sont disponibles :

- **Brouillon** : Vous estimatez 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 estimatez 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 estimatez que la ressource informatique n'est pas correcte, car elle comporte des défauts, des incohérences ou est un dupliqué d'une autre ressource informatique.
- **Archiver** : Vous estimatez 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 estimatez 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.

5.2.3.2 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.

Titre de recherche	Description	Nombre de tags sélectionnés	Tags
Cours de Java avec John	Titre de recherche : Algorithme et structures de données	3	[LSINF121] Algorithme et structures de données John Aoga java
Recettes et algorithmes	Titre de recherche : Recettes et algorithmes	4	INGINIOUS match Charline Outters Algorithmes et recettes de cuisine
Plateforme Google	Titre de recherche : Google	2	Google INGINIOUS
Exos en C et Java	Titre de recherche : code	2	java c
Test de favori	Titre de recherche : Depth	5	Maxime Mawat c java makefile John Aoga
Mon premier	Titre de recherche : mon	1	facile

FIGURE 5.23 – Gestion des favoris (mes-favoris/)

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.

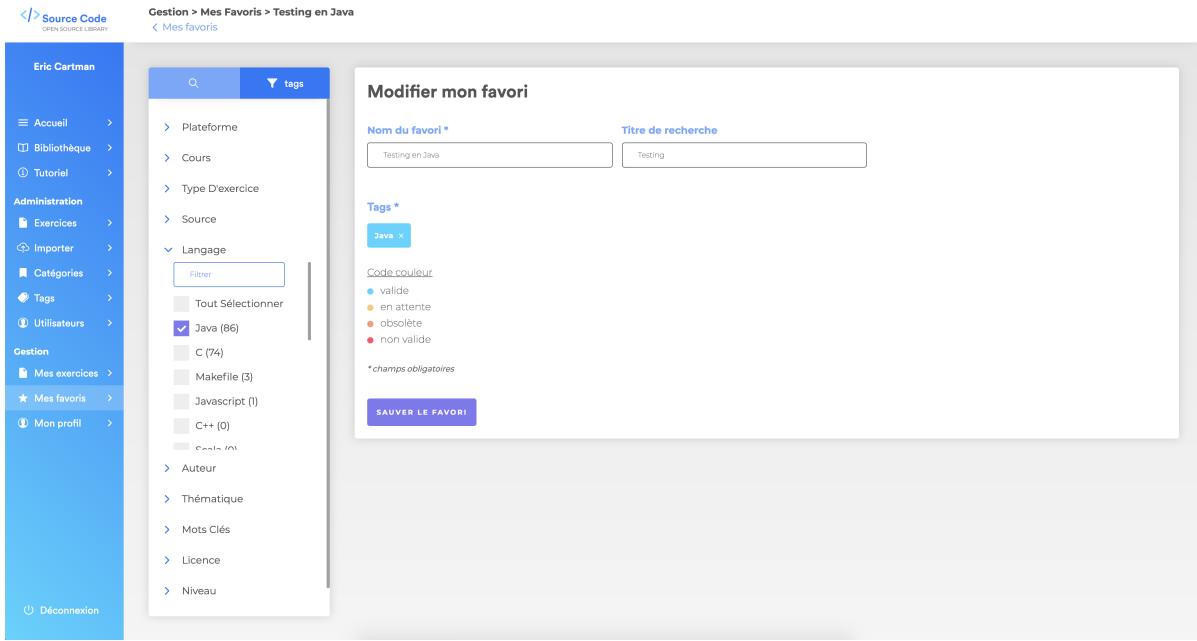


FIGURE 5.24 – Création/modification des favoris (creer-favori ou __id)

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 A. :

- 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.

5.2.3.3 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 (voir section ???).

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

Eric Cartman

- ≡ Accueil >
- Bibliothèque >
- ⓘ Tutoriel >
- Administration**
 - ❑ Exercices > **Exercices**
 - ▷ Importer >
 - Catégories >
 - ⌚ Tags >
 - 👤 Utilisateurs >
- Gestion**
 - ❑ Mes exercices >
 - ★ Mes favoris >
 - 👤 Mon profil > **Mon profil**

Déconnexion

FIGURE 5.25 – Consultation du profil (profil)

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.

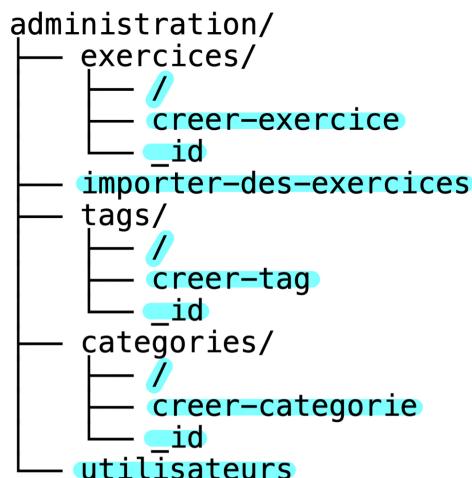


FIGURE 5.26 – Partie administration (URL)

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.

Titre	Créeur	Nb de votes	Moyenne	Mis à jour	Fichier(s)	Statut
Decision Making in Java - if-else	Eric Cartman	0	-	28/04/20 à 15:25	X	✓
Lorem	adwid	0	-	28/04/20 à 15:20	X	⚠
Decision Making in Java - nested-if	Eric Cartman	1	4	28/04/20 à 15:09	X	✓
Coverage Testing	Eric Cartman	0	-	28/04/20 à 15:03	X	✗
Loops - do while	Alexandre Rucquoys	0	-	28/04/20 à 14:43	X	⚠
Loops - for loop	adwid	0	-	28/04/20 à 14:41	X	⚠
Decision Making in Java - if-else-if ladder	Eric Cartman	1	3	28/04/20 à 14:38	X	✓
Loops - while loop	avdb	0	-	28/04/20 à 14:38	X	⚠
Circular linkedlist (Implem)	Eric Cartman	0	-	28/04/20 à 14:27	X	⚠
Circular linkedlist (Implem)	Eric Cartman	0	-	28/04/20 à 14:25	X	✓
Circular Linked list	Eric Cartman	0	-	28/04/20 à 14:24	X	✓
Liste doublement chaînée	Eric Cartman	1	3	26/04/20 à 9:53	X	✓

FIGURE 5.27 – Administration des ressources informatiques (exercices/)

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

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).

Actions supplémentaires Par rapport à un utilisateur classique, vous pouvez changer l'état d'une ressource informatique de manière plus large avec les états suivants :

- Valide
- Invalide
- En attente
- Archive
- Brouillon

Pour connaître la signification de ces états, reportez-vous à la section 5.2.3.1 concernant la partie : statuts d'une ressource informatique

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 ???, car nous aurions voulu intégrer un système de feedback 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 **SourceCode**. Nous avons alors développé une interface prenant en charge l'importation de ressources informatiques.

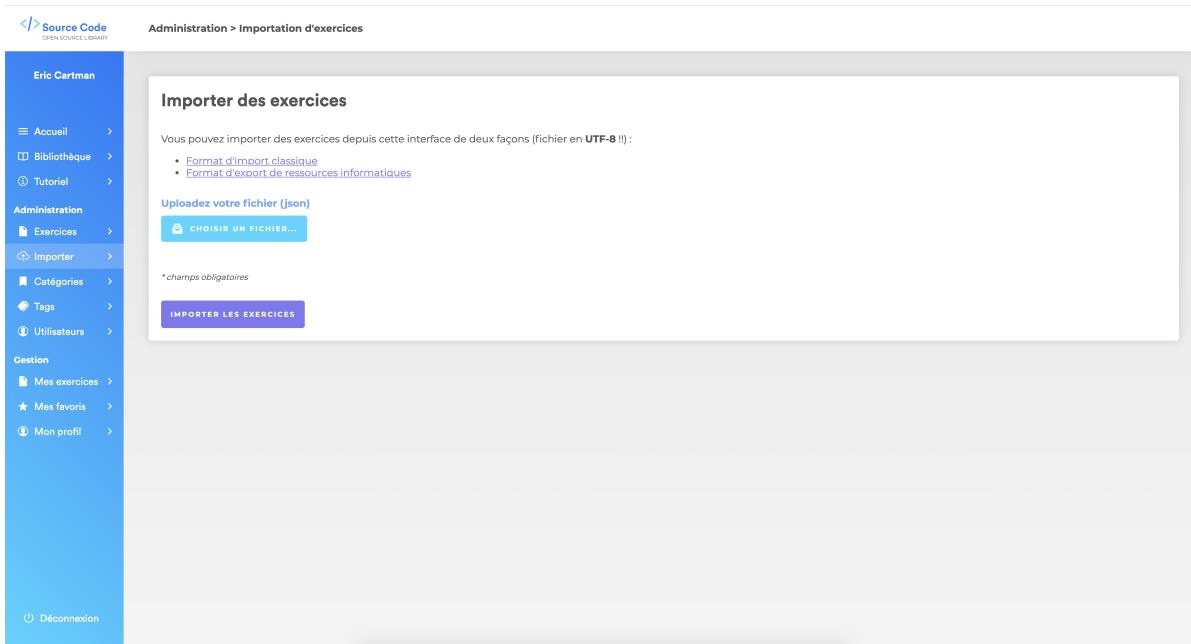


FIGURE 5.28 – Importation des ressources informatiques (importer-des-exercices)

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, ...). À ce propos, la section **? ? ?** de la partie backend explique de manière plus concise notre choix concernant les priviléges de cette fonctionnalité.

5.2.4.3 La gestion des mots-clés

Les (super-)administrateurs ont la possibilité de gérer les mots-clés présents sur la plate-forme. 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 (voir 5.2.4.1), mis à part que les filtres se limitent aux catégories de mots-clés et au statut des mots-clés.

Nom	Catégorie	Exercices tagués	Version	Statut
INGINIOUS	plateforme	204	9	✓
Google	plateforme	0	9	✓
[LSINF125] Systèmes informatiques	cours	90	1	✓
[LEPL1402] Informatique 2	cours	62	1	✓
[LSINF112] Algorithmique et structures de données	cours	47	1	✓
validation mémoire	cours	8	0	✓
Algorithmes et recettes de cuisine	cours	5	1	✓
code	type d'exercice	164	1	✓
match	type d'exercice	37	4	✗
examen	type d'exercice	17	1	✓
QCM	type d'exercice	16	1	✓
didacticiel	type d'exercice	7	0	✓

FIGURE 5.29 – Gestion des mots-clés (tags/)

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 SourceCode.

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 (voir analyse bibliographique en annexe A).

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 SourceCode.

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.

The screenshot shows the 'Créer un tag' (Create a tag) form. The title is 'Créer un tag'. There are two main input fields: 'Titre *' (Title *) and 'Catégorie *' (Category *). Below the 'Titre' field is a placeholder 'Entrez le nom du tag' (Enter the name of the tag). To the right of the 'Catégorie' field is a blue button labeled 'CHOISIR UNE CATÉGORIE'. A note at the bottom left of the form area says '* champs obligatoires' (mandatory fields). At the bottom of the form are four buttons: 'INVALIDER' (red), 'MARQUER COMME OBSOLETE' (orange), 'METTRE EN ATTENTE' (yellow), and 'VALIDER' (blue). On the left side of the page is a sidebar with a user profile 'Eric Cartman' and various navigation links: Accueil, Bibliothèque, Tutoriel, Administration (with sub-links for Exercices, Importer, Catégories, Tags, Utilisateurs), Gestion (with sub-links for Mes exercices, Mes favoris, Mon profil), and Déconnexion.

FIGURE 5.30 – Création/modification des mots-clés (creer-tag/ ou _id)

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).

Nom	Nb de tags	Nb de tags valides	Nb de tags invalides	Nb de tags en attente	Nb de tags Obsolètes
niveau	4	4	0	0	0
type d'exercice	7	5	1	0	1
public ciblé	0	0	0	0	0
thématique	8	8	0	0	0
Misconception	0	0	0	0	0
licence	3	3	0	0	0
Mots clés	25	20	4	0	1
plateforme	2	2	0	0	0
langage	16	16	0	0	0
source	4	4	0	0	0
cours	5	5	0	0	0
learning time	0	0	0	0	0
difficulté	4	4	0	0	0
auteur	51	40	0	11	0

FIGURE 5.31 – Gestion des catégories de mots-clés (categories/)

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 SourceCode. 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.

ID	Nom	Email	Rôle
1	Eric Cartman	yolo24@uclouvain.be	Super Administrateur
4	Admin A	adminA@sourcecode.be	Administrateur
5	Admin B	adminB@sourcecode.be	Administrateur
6	Admin C	adminC@sourcecode.be	Administrateur
7	Admin D	adminD@sourcecode.be	Administrateur
8	Admin E	adminE@sourcecode.be	Administrateur
9	Admin F	adminF@sourcecode.be	Administrateur
10	Admin G	adminG@sourcecode.be	Administrateur
11	Admin H	adminH@sourcecode.be	Administrateur
12	Admin I	adminI@sourcecode.be	Administrateur
13	Admin J	adminJ@sourcecode.be	Administrateur
2	Alexandre D.	test@sourcecode.be	Super Administrateur
14	GROOT	giqdyseacgzmpqgg@awdrt.com	Utilisateur
15	adwid	ycwuilovgqcgbtgxfg@ttirv.com	Utilisateur

FIGURE 5.32 – gestion des utilisateurs (utilisateurs)

5.3 API

5.3.1 Architecture

5.3.2 Points clés de l'implémentation

Spécification OAS

Prenons maintenant un peu de recul sur ce projet...

Au chapitre 2, nous avons d'abord défini les besoins clés d'une application de partage de ressources informatiques en analysant une série de plateformes web existantes.

Après cela, en chapitre 4, nous nous sommes concentrés sur une analyse plus profonde de notre plateforme en y relatant les fonctionnalités à intégrer avec la méthode ***MoSCoW***, les besoins à satisfaire pour que la plateforme puisse être utilisée par le grand public, ainsi que les contraintes qui nous ont été imposées (en incluant celles que nous nous sommes imposées).

Dans le chapitre 5, nous avons enfin expliqué notre solution à la problématique du partage de ressources informatiques en détaillant l'architecture de **SourceCode** aussi bien du côté Frontend que du Backend.

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

6.1 Validation

6.1.1 Validation externe

Pour valider notre travail, nous avions initialement planifié deux séances de validation afin de récolter les avis d'un panel d'utilisateurs suffisamment large. Pour cause de COVID-19, il n'a été possible d'organiser qu'une seule, avec sept utilisateurs. Chaque personne se voyait attribuer deux rôles différents (visiteur-utilisateur ou visiteur-administrateur) afin de prendre connaissance des fonctionnalités les plus importantes de **SourceCode**. Notre évaluation a pu être réalisée grâce à Google Forms (réponses, graphiques) et vous retrouverez l'intégralité de ce sondage dans l'annexe ???.

Parmi les participants, six d'entre eux font partie de l'*UCLouvain* de la faculté *EPL*, 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 **SourceCode**.

Les sous-sections suivantes résument les critiques émises sur SourceCode. 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)**
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)
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 ???.

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)

- 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.
- Certaines personnes se plaignaient que la boîte d'édition était trop petite (ou trop grande). Dans son état actuel, **SourceCode** n'est pas responsive, et dans la perspective où elle le serait, ce problème sera certainement réglé pour tout écran.
- 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 changé 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)

- 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.

- 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)

1. **SourceCode** 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, **SourceCode** 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 (voir chapitre ???).
3. C'est un point intéressant pour nous. **SourceCode** 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. Il faut que la plateforme puisse évoluer en termes d'ergonomie pour rendre l'interface plus accessible aux utilisateurs.

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)
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 ???

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)

1. **SourceCode** 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, **SourceCode** 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 inginious 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.
2. 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'avions 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 disposition. (1)

1.

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 SourceCode pour l'améliorer le plus possible. Pour celles qui n'ont pas été ajoutées, nous les avons référencées dans le chapitre ???.

6.2 Métriques

6.3 État du projet

CHAPITRE 7

POUR ALLER PLUS LOIN

CHAPITRE 8

CONCLUSION

BIBLIOGRAPHIE

- [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] 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/>.
- [4] 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>.
- [5] DELMÉE PATRICK. Dans : *PROF* (6 déc. 2019). URL : <http://www.enseignement.be/index.php?page=27203&id=2922>.
- [6] UNESCO. *Ressources éducatives libres*. URL : <https://fr.unesco.org/themes/tic-education/rel>.
- [7] WIKIPEDIA. *Bibliothèque logicielle*. URL : https://fr.wikipedia.org/wiki/Biblioth%C3%A8que_logicielle.
- [8] WIKIPEDIA. *Méthode MoSCoW*. URL : https://fr.wikipedia.org/wiki/M%C3%A9thode_MoSCoW.

Annexes

ANNEXE A

ANALYSE BIBLIOGRAPHIQUE

TODO includegraphics pdf

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain

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