

**BUT Informatique 2023-2024 – Semestre 4**

**SAÉ 4.01** [**Intégration d'une application et gestion de projet**](https://elearn.univ-pau.fr/course/view.php?id=19005)

**Pôle Communication**

**Parcours D (IAMSI)**

**Groupe 22**

**Tuteur : BOGGIA Analia**

**PROJET VINYLOG**

**MONTOURO Maxime**

Table des matières

[I - / Présentation du projet 3](#_Toc162601717)

[I.I - / Contexte de l’équipe : 3](#_Toc162601718)

[I.II -/ Contexte de l’application : 3](#_Toc162601719)

[II - / Présentation de l’algo : 4](#_Toc162601720)

[II.I - / Contexte : 4](#_Toc162601721)

[II.II - / Fonctionnement de l’algorithme : 4](#_Toc162601722)

[II.III - / Les technologies intervenus : 5](#_Toc162601723)

[III - / Bilan : 6](#_Toc162601724)

[III.I - / Bilan technique : 6](#_Toc162601725)

[III.II - / Bilan Général : 6](#_Toc162601726)

[IV - / Annexes : 7](#_Toc162601727)

[IV.I - / Annexe 1 : 7](#_Toc162601728)

[IV.II - / Annexe 2 : 8](#_Toc162601729)

[IV.III - / Annexe 3 : 9](#_Toc162601730)

Tables des figures :

[Figure 1: Démonstration de l'algorithme 7](#_Toc162601074)

[Figure 2: Racine du github 8](#_Toc162601075)

[Figure 3: Fichiers de code 9](#_Toc162601076)

# I - / Présentation du projet

## I.I - / Contexte de l’équipe :

Vinylog est une application web imaginé tout d’abord dans le cadre de la SAE 2.05 par un groupe d’étudiant. Au semestre 3, seulement Alexandre Picoulet—Sonder a continué le projet. Par conséquent, d’autres étudiants ont rejoint le projet, ces derniers sont :

- Eneko Arbelbide (Développeur)

- Anthony Hermann (Alternant)

- Maxime Montouro (Développeur et Chef de projet)

-Kyllian Morrance (Développeur)

Notre méthode de gestion de projet était du SCRUM But, une méthode de gestion de projet agile.

Etant seulement 4, nous devions réorganiser SCRUM (qui prévoit une équipe de 8 Développeurs à plein temps, 1 Scrum Master et 1 Product Owner). Par conséquent, nous étions tous Scrum Master, Product Owner et Développeur.

## I.II -/ Contexte de l’application :

Vinylog est une donc une application web de catalogue sur le monde du vinyle, permettant à un utilisateur de rechercher un vinyle, obtenir des conseils sur l’entretien des vinyles ou bien encore vendre et acheter des vinyles vendus par d’autres utilisateurs.

  Nous souhaiterons simplifier la compréhension de cet univers et son accès à tous. Grâce à une plateforme centralisée où on aura accès à toutes les possibilités qui s’offrent. C’est un catalogue répertoriant un maximum de vinyles existants et leurs points de vente, avec une description détaillée des produits, par tous les amateurs de vinyles. Il leur permettra d’obtenir facilement des nouveaux vinyles tout en bénéficiant de conseils pratiques utiles pour une écoute optimale. Le catalogue proposera aussi des produits d’entretien pour les appareils adaptés au monde du vinyle.

Notre application vise à offrir une expérience complète et adaptée à tous les profils de visiteurs afin de valoriser la culture musicale du vinyle et de perpétuer la communauté de passionnés de vinyles.

Plus besoin de chercher pendant des heures, tout se trouve sur notre application web.

# II - / Présentation de l’algo :

## II.I - / Contexte :

Notre but étant de pouvoir offrir une expérience à tout type d’utilisateur, du plus confirmé au débutant. Par conséquent, un novice dans le domaine ne sait pas forcément quoi chercher et comment le faire.

C’est ici qu’intervient l’algorithme de correction d’erreurs. Ce dernier se déclenche lorsqu’une saisie est effectuer dans une barre de recherche. Nous partons dans l’optique que chacun des mots saisis est erroné.

## II.II - / Fonctionnement de l’algorithme :

Mais du coup comment fonctionne cet algorithme ?

Dans un premier temps, vous allez devoir saisir une information dans la barre de recherche, ainsi que deux entiers. Ces deux entiers seront utiles dans la suite de l’algorithme. (cf Annexe IV.1).

La chaine saisie sera analysée avec chacun des mots stockés dans notre base de données. Mais notre base de données étant conséquentes, un premier filtrage sera effectué. Le meilleur moyen de filtrer est d’enlever les mots qui ne sont pas de la même taille que le mot saisi.

Et c’est parti nous rentrons dans l’algorithme !

Nous allons récupérer votre saisie, celle-ci sera analysé. Pour l’analyse, la chaîne de caractères va passer par deux sections. Tout d’abord, nous utilisons l’algorithme de Damarau-Levenshtein. L’algorithme est utilisé pour mesurer la similarité entre deux chaînes de caractères. Il se base sur quatre points :

* Insertion d'un caractère
* Suppression d'un caractère
* Remplacement d'un caractère
* Transposition de deux caractères adjacents

Ces quatre points ne sont pas forcément traités car comme préciser dans le fonctionnement de l’algorithme, seuls les mots de même longueur sont gardés donc l’insertion et la suppression de caractère ne sont pas utilisées (malgré leur implémentation). En même temps que la chaîne sont saisis deux entiers, intervient le premier entier. En effet, chacun des changements (donc remplacement et transposition de caractères) vaut un point. Si le nombre de changement devient supérieur alors le mot n’est pas gardé. Dans le cas contraire, le mot est gardé dans une liste. Lorsque tous les mots ont fini d’être analysé, une liste avec les mots les plus probables seront retournés (qui peut d’ailleurs être vide).

Dans la seconde partie de l’algorithme, nous allons calculer la distance physique du clavier entre deux lettres. Ce dernier se base sur le fait qu’en tapant une lettre au clavier, notre doigt peut appuyer sur une autre lettre proche. Et ici, intervient le second entier, qui comme précédemment, est une limite de distance afin de déterminer si un mot est proche (qui peut aussi être vide).

Pour conclure, un dernier algorithme entre en jeu. Ce dernier va analyser les deux listes de mots précédemment créés afin de déterminer le mot le plus probable. Pour se faire, une analyse de chacun des mots de la liste de Damarau-Levenshtein va être comparé à chacun des mots de la liste de « Saisie Clavier ». Ici, il y a plusieurs cas de figures possibles :

* Un mot est contenu dans les deux listes alors il s’agit du mot le plus probable
* Une des deux listes et vide, alors on affiche la liste de l’autre algorithme
* Les deux listes sont vides alors le mot a donc trop d’erreurs pour être possible.

## II.III - / Les technologies intervenus :

Cet algorithme a uniquement été développé en JavaScript, un Langage Web qui permet la dynamisation des pages.

Tout cet algorithme a utilisé le paradigme POO ou Programmation Orientée Objet. C’est une façon de programmer proche du réel en créant des objets qui possède des attributs, des fonctions spécifiques. (ex : dans un jeu vidéo on ne créer pas chacune des voitures à la main mais on sait qu’elles ont des spécificités comme 4 roues des portes etc. mais des différences entre une Honda et une Peugeot). Chacune des classes créer possède des attributs, des fonctions propres à eux. (pour comprendre l’arborescence du Github, cf. Annexe IV.II et IV.III).

# III - / Bilan :

## III.I - / Bilan technique :

1. Mon apprentissage pour le développement de cet algorithme a été important. En effet, d’une part, l’apprentissage de nouvelles technologies comme Javascript fût riche. Mais aussi sur la gestion d’erreur et/ou bug. De plus, lors des différentes phases de tests, de nombreux bugs sont apparus et dans un temps très court, j’ai dû en régler un grand nombre. L’apprentissage été donc surtout cibler sur les compétences C1 (Réaliser) et C2 (Optimiser).

## III.II - / Bilan Général :

Pour conclure, mon apprentissage était surtout ciblé sur C1 et C2 mais aussi dans la gestion d’équipe. En effet, mon axe d’amélioration n’était pas que sur la programmation mais aussi sur tout ce qui gravite autour comme la gestion de projet. En parallèle du développement de l’algorithme, je travaillais aussi sur les autres pôles de compétences. Je ne peux donc pas résumer mon apprentissage qu’aux compétences 1 et 2.

# IV - / Annexes :

## IV.I - / Annexe 1 :

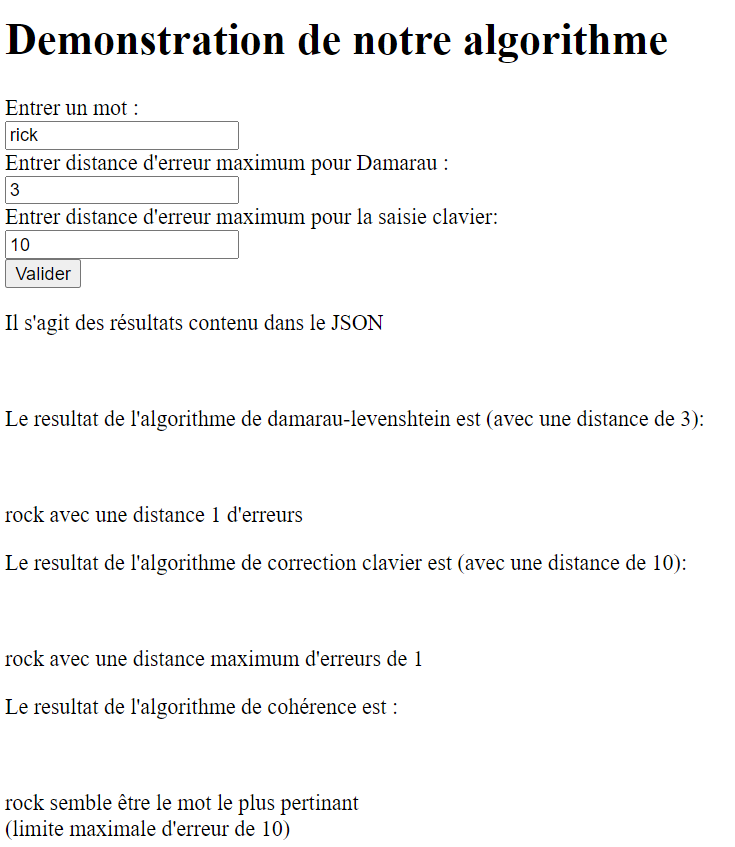
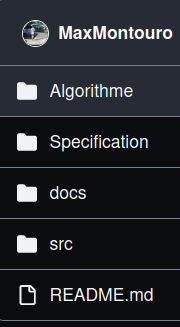


Figure 1: Démonstration de l'algorithme

Ici, nous pouvons voir que le mot ‘rick’ a été saisie, ainsi que 3 pour le nombre d’erreur maximum pour l’algorithme de Damarau-Levenshtein et 10 pour la saisie clavier.

Nous pouvons voir le déroulement complet de l’algorithme avec en finalité, le mot le plus probable par rapport à la saisie : Rock.

## IV.II - / Annexe 2 :



Ici, il s’agit de l’arborescence à la racine du projet, avec les différents sous-dossiers.

Figure 2: Racine du github

## IV.III - / Annexe 3 :



Ici, nous pouvons voir les différents fichiers contenants

tout le code de l’algorithme.

Figure 3: Fichiers de code