  


**SAÉ 3.01 A&D : Développement et Gestion de Projet**

**Dictionnaire des données – Groupe 22**

**Tuteur : BOGGIA Analia**

**PROJET VINYLOG**

**ARBELBIDE Eneko – HERRMANN Anthony – MONTOURO Maxime – MORANCE Kyllian – PICOULET--SONDER Alexandre**

# Tables des matières

[Tables des matières 1](#_Toc1432405269)

[1. Dictionnaire des éléments 1](#_Toc738068559)

[2. Spécifications afin de réaliser l’algorithme 1](#_Toc1300732718)

[2.1 Algorithme de Damerau-Levenshtein : 1](#_Toc739366471)

[2.2 Algorithme de distance de mots suivant la disposition physique des touches : 1](#_Toc1900240759)

[3. Diagramme des classes 1](#_Toc1107177457)

# 1. Dictionnaire des éléments

Ici sont les éléments présents dans le bloc “Obtenir l'ensemble des mots ayant une distance (Damerau-Levenshtein) de distanceMax ou moins”.

|  |  |  |
| --- | --- | --- |
| Nom | Type | Signification |
| mot | String | Mot entré par un utilisateur pouvant contenir des erreurs |
| distanceMax | Int | Distance max de ressemblance entre deux mots |
| ensembleMots[] | String | Tableau de chaine de caractères contenant l’ensemble des mots de |
| alphabet | String | Chaine de caractère contenant les lettres de l’alphabet |

2. Spécifications afin de réaliser l’algorithme

Afin de réaliser plus facilement notre algorithme de recherche de cohérence de deux chaînes de caractères. En effet, nous utiliserons l’algorithme de Distance de Damerau-Levenshtein.

## 2.1 Algorithme de Damerau-Levenshtein :

Cet algorithme consiste à calculer le nombre minimum d’opérations nécessaires pour transformer une chaîne de caractère en une autre. Une opération peut être défini par 4 opérations :

* L'insertion
* La suppression d’un simple caractère
* La substitution d’un simple caractère
* Transposition de deux caractères adjacents

Ainsi grâce à cet algorithme, nous pourrons détecter si un mot entré par un utilisateur comportant des erreurs peut correspondre à un élément présent dans la base de données. Ce dernier pourrait être un style de musique, le nom d’un artiste ou encore le nom d’un vinyle.

Nous utiliserons cet algorithme qui sera dérouler au plus 5 fois car nous partons dans l’optique qu’il y aura au plus 5 erreurs lors d’une saisie clavier par un utilisateurs.

## 2.2 Algorithme de distance de mots suivant la disposition physique des touches :

Le but de cet algorithme est d’évaluer la similarité entre deux mots en prenant en compte la disposition physique des touches sur le clavier. Il mesure la proximité des caractères en considérant les déplacements entre les touches voisines pour chaque mot, permettant ainsi d'identifier des mots similaires malgré des erreurs de frappe dues à des caractères adjacents sur le clavier.

3. Diagramme des classes

Différentes classes possibles dans la solution algorithmique :

* Mot
* Voisins
* Clavier
* Dictionnaire