



Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

\*\*\*

Université de Sousse

\*\*\*

Institut Supérieur de Gestion de Sousse

License en Informatique de Gestion (Business Intelligence)

Rapport de Stage de Fin D'Études

Amélioration de recherche et intégration de Recherche Intelligent

Borgi Aaron et Ben Hassine Ghaithallah  
3ème LIG

Date

Soutenance orale  
le *insert date*

Encadreur de stage : Haddad Ahmed  
Réfèrent universitaire : Haddad Ahmed  
Année Universitaire : 2023-2024

## Remerciements

Nous voudrions remercier tous ceux qui ont participé à mener ce projet. Nous tenons spécialement à remercier :

Monsieur, Ahmed Haddad, pour son encadrement, son soutien, ses conseils qui nous ont été d'une grande utilité pour rédiger, pour l'aide et le support tout au long de ce projet, ses conseils ont été d'un grand apport.

Nos remerciements vont aussi à tous les professeurs de l'ISGS, pour leurs générosités et la patience dont ils ont su faire preuve.

## Dédicace

A nos parents, qui nous ont aidé à devenir ce que nous sommes aujourd'hui,

A nos amis et à nos proches,

A tout ceux qui ont nous aidé à accomplir ce projet...

Toute notre reconnaissance.

# Table des matières

<b>1</b>	<b>Introduction Générale</b>	<b>1</b>
1.1	Étude du projet . . . . .	2
1.1.1	Introduction . . . . .	2
1.1.2	Cadre du projet . . . . .	2
1.2	Présentation de l'entreprise . . . . .	2
1.2.1	Secteurs d'activité . . . . .	4
1.3	Etude de l'existant . . . . .	4
1.3.1	Analyse de l'existant . . . . .	4
1.3.2	Critique de l'existant . . . . .	5
1.4	Les Solutions . . . . .	6
1.5	Méthodologie de développement . . . . .	7
1.5.1	Introduction . . . . .	7
1.5.2	Méthode Adoptée : SCRUM . . . . .	7
1.5.3	Le Langage UML . . . . .	9
1.6	Conclusion . . . . .	9
<b>2</b>	<b>Approches Proposés</b>	<b>10</b>
2.1	Introduction . . . . .	10
2.2	Les solutions possibles . . . . .	10
2.2.1	Recherche Régulière . . . . .	10
2.2.2	Utilisaton d'une base de données SQL altèrnatif et performante	11
2.3	Recherche Vectorielle . . . . .	11

2.4	Elasticsearch . . . . .	12
2.5	Utilisation d'un modèle Sentence-Transformers . . . . .	13
2.5.1	Comment le modèle interprète les phrases ? . . . . .	13
2.5.2	Pourquoi utiliser Mean Pooling plutôt qu'un autre type de pooling ? . . . . .	17
2.6	Conclusion . . . . .	18
<b>3</b>	<b>Sprint 0 : Plannification du projet</b>	<b>19</b>
3.1	Introduction . . . . .	19
3.2	Identification des besoins . . . . .	19
3.2.1	Les besoins fonctionnels . . . . .	19
3.2.2	Les besoins non fonctionnels . . . . .	21
3.3	Diagramme de cas d'utilisation global . . . . .	22
3.3.1	Introduction . . . . .	22
3.4	Backlog De Produit . . . . .	22
3.5	La Plannification De Release . . . . .	23
3.6	Architecture du système . . . . .	24
3.6.1	Architecture "MVC" . . . . .	24
3.6.2	Architecture "Clean Architecture" . . . . .	26
3.7	L'environnement de développement et choix techniques : . . . . .	29
3.7.1	Introduction . . . . .	29
3.7.2	L'environnement matériel . . . . .	29
3.7.3	L'environnement logiciel . . . . .	30
3.7.4	Logiciel de modélisation UML . . . . .	37
3.8	Conclusion . . . . .	37
<b>4</b>	<b>Étude et réalisation du Sprint 1</b>	<b>38</b>
4.1	Introduction . . . . .	38
4.2	Backlog du Sprint 1 . . . . .	38
4.3	Spécification fonctionnelle . . . . .	38

4.3.1	Diagramme de cas d'utilisation général . . . . .	39
4.3.2	Description textuelle du CU « Rechercher produit en Français »	39
4.3.3	Diagramme de séquence détaillé . . . . .	40
4.4	Architecture de la base de données . . . . .	41
4.4.1	Diagramme de classes . . . . .	41
4.4.2	Schéma de la base de données . . . . .	41
4.4.3	Tables de base de données MySQL . . . . .	41
4.4.4	Tables de base de données Elasticsearch . . . . .	42
4.5	Réalisation . . . . .	43
4.5.1	La création des colonnes des vecteurs . . . . .	44
4.5.2	Le modèle Sentence-Transformers et encodage des phrases . . .	45
4.5.3	Elasticsearch et utilisation de la similarité cosinus dans le re- cherche vectorielle . . . . .	51

<b>Bibliographie</b>	<b>54</b>
----------------------	-----------

# Table des figures

1.1	Logo de l'entreprise d'Axam. . . . .	2
1.2	Présentation du processus SCRUM . . . . .	8
2.1	Présentation du similarité cosinus . . . . .	12
2.2	Presentation du processus de tokenisation . . . . .	14
2.3	Exemple du processus de tokenisation . . . . .	15
2.4	Présentation du processus de Mean Pooling . . . . .	16
2.5	Présentation du processus de Max Pooling . . . . .	17
2.6	Présentation du processus de Min Pooling . . . . .	18
3.1	Présentation de Diagramme de Cas D'Utilisation global . . . . .	22
3.2	L'architecture MVC . . . . .	25
3.3	Échange d'informations entre les éléments MVC . . . . .	26
3.4	Architecture "Clean Architecture" dans ASP. NET Core . . . . .	27
3.5	Logo officiel du logiciel Visual Studio . . . . .	30
3.6	Logo officiel du logiciel Visual Studio Code . . . . .	30
3.7	Logo officiel du langage de programmation C# . . . . .	31
3.8	Logo officiel du framework ASP .NET Core . . . . .	32
3.9	Logo officiel du langage de programmation Python . . . . .	32
3.10	Logo officiel du framework Jupyter . . . . .	33
3.11	Logo officiel du framework Flask . . . . .	33
3.12	Logo officiel du bibilothèque React . . . . .	34
3.13	Logo officiel du langage Typescript . . . . .	34

3.14	Logo officiel du Docker . . . . .	35
3.15	Logo officiel d'Elasticsearch . . . . .	35
3.16	Logo officiel du Kibana . . . . .	35
3.17	Logo officiel du logiciel Postman . . . . .	36
3.18	Logo officiel du Overleaf . . . . .	36
3.19	Logo officiel du LaTeX . . . . .	36
3.20	Logo officiel du Visual Paradigm . . . . .	37
4.1	Diagramme de cas d'utilisation général du Sprint 1 . . . . .	39
4.2	Diagramme de séquence de cas d'utilisation « Rechercher produits en Français » . . . . .	40
4.3	Code d'index de Produit . . . . .	44
4.4	Code d'importation de modèle Sentence-Transformers . . . . .	45
4.5	Exemple de processus de Tokenisation . . . . .	46
4.6	Exemple de Mean Pooling . . . . .	49
4.7	Code de méthode mean_pooling . . . . .	49
4.8	Méthode encode_sentence_and_normalise . . . . .	51
4.9	Encodage des deux colonnes Brève Description et SEO Label Produit .	52
4.10	les deux nouvelles colonnes « descriptionvecteur » et « labelproduitvecteur » . . . . .	52
4.11	Insertion des données dans Elasticsearch . . . . .	53



# Liste des tableaux

1.1	Fiche d'information sur Axam . . . . .	3
1.2	Comparaison des compagnies d'assurance . . . . .	5
3.1	Table des fonctionnalités et scénarios. . . . .	23
3.2	Spécifications des ordinateurs utilisés . . . . .	29
4.1	Backlog du Sprint 1 . . . . .	38
4.2	Table Client . . . . .	41
4.3	Table Keyword . . . . .	41
4.4	Table Produit . . . . .	42

# Chapitre 1

## Introduction Générale

L'intelligence Artificielle (IA) a radicalement transformé de nombreux aspects de notre vie moderne dans les dernières années. De nos jours, l'intelligence Artificielle a devenu l'une des plus grandes solutions pour les sociétés de toutes tailles, spécifiquement ceux qui spécialisent dans l'E-commerce.

Également, le Business Intelligence (BI) joue aussi un rôle très important pour l'analyse, nettoyage des données des clients et des produits de l'entreprise, et les stocker, ces étapes résultent dans une méthode de recherche plus intelligente et précise.

De plus, les sites de commerce électronique utilisent également l'IA pour améliorer la recherche et la recommandation de produits. Les algorithmes d'apprentissage automatique analysent le comportement des utilisateurs, les historiques d'achat et les données contextuelles pour personnaliser les résultats de recherche et suggérer des produits pertinents, ce qui améliore l'expérience d'achat et stimule les ventes.

## 1.1 Étude du projet

### 1.1.1 Introduction

Dans ce premier chapitre, nous mettrons l'accent sur le cadre du projet. Par conséquent, nous commencerons par présenter l'entreprise d'accueil Axam. Par la suite, nous présenterons en détail le sujet du projet, en introduisant une vue d'ensemble du problème, une étude de l'existant et la solution proposée. ainsi la présentation de la méthode de travail pour laquelle nous avons opté.

### 1.1.2 Cadre du projet

Ce projet s'inscrit dans le cadre de projets de fin d'études au sein de l'Institut Supérieur de Gestion de Sousse pour l'obtention du diplôme de licence en informatique de gestion (Business Intelligence). Il a été réalisé au sein de la start-up Axam. Ce travail est destiné à l'entreprise elle-même pour l'amélioration de sa méthode de recherche des produits dans sa site-web.

## 1.2 Présentation de l'entreprise

Notre stage de fin d'études c'est déroulé au sein de l'entreprise « Axam ». C'est un bureau informatique professionnel en développement des sites web, des applications mobiles, et l'E-commerce.

La figure 1.1 représente le logo de Axam.



**Figure 1.1** – Logo de l'entreprise d'Axam.

<b>Nom de l'entreprise</b>	Axam
<b>Date de création</b>	2022
<b>Secteurs d'activité</b>	Activités informatique / E-commerce
<b>Siège</b>	Sahline, Monastir
<b>Téléphone</b>	+216 55 220 306
<b>Adresse e-mail</b>	contact@axam.tn

**Tableau 1.1** – Fiche d'information sur Axam

Axam a été créée en 2022. La table 1.1 présente une fiche d'information sur cette entreprise.

### **1.2.1 Secteurs d'activité**

Axam est une boîte de développement, de services informatiques, et d'E-commerce spécialisée dans :




- la création et développement des applications web.
- la création des applications mobiles (Android et IOS).
- le design (web et mobile)
- la vente des produits sur son site e-commerce.

## **1.3 Etude de l'existant**

Cette première étape de notre projet consistera en une analyse de l'existant, suivie d'une critique de l'existant afin de proposer notre solution comme solution aux problèmes identifiés. Nous baserons notre recherche sur des solutions existantes et opérationnelles des marchés e-commerce existantes et cherchons à améliorer ce processus.

### **1.3.1 Analyse de l'existant**

Il est essentiel avant de commencer à réaliser le projet, de bien étudier et analyser les points forts et faibles des solutions existantes et envisager des améliorations. Nous nous concentrons sur les processus métiers implémentés chez les autres entreprises d'e-commerce, en examinant l'efficacité de ces processus. Cette analyse nous permettra de comprendre comment améliorer les solutions existantes.

Logo	Description	Avantages	Inconvénients
	Jumia est un marketplace complète avec un écosystème de paiement et livraison global	Processus de recherche généralisé et rapide pour la recherche des produits existantes.	Recherche que en Français, absence des suggestions si le produit n'existe pas.
	Tunisianet est un marketplace Tunisienne des produits informatiques proposant une variété des produits et services aux clients.	Processus de recherche rapide.	Recherche que en Français, absence des suggestions si le produit n'existe pas, et résultats non précis.
	Wamia est un marketplace Tunisienne complète qui offre des différents produits et services aux clients.	Processus de recherche rapide.	Recherche que en Français, absence des suggestions si le produit n'existe pas, et résultats non précis.

**Tableau 1.2** – Comparaison des compagnies d'assurance

### 1.3.2 Critique de l'existant

Après une analyse de l'existant nous avons relevé quelques problèmes tels que :

- L'absence de la recherche des produits en utilisant la langue Tunisienne (Derja)
- Vitesse de recherche lente lors de la recherche d'un ou plusieurs produits.
- L'absence de la recherche des produits en utilisant la langue Arabe traditionnelle.
- Résultats de recherche non précis la plupart du temps.

## 1.4 Les Solutions

Pour résoudre les problèmes mentionnés ci-dessus, la société « Axam » nous a proposé de développer et utiliser un modèle Sentence Transformer, en s'appuyant sur la plateforme Elasticsearch pour les clients mettre de :

- Améliorer la vitesse de recherche lors de la recherche d'un ou plusieurs produits.
- Améliorer la précision des résultats de recherche pour qu'ils correspondent exactement à ce que recherche le client dans les langues mentionnées ci-dessus.
- Pouvoir rechercher un ou plusieurs produits dans la langue Française.
- Pouvoir rechercher un ou plusieurs produits dans la langue Arabe en dialecte Tunisienne.
- Pouvoir rechercher un ou plusieurs produits dans la langue Arabe Traditionnelle.

## 1.5 Méthodologie de développement

### 1.5.1 Introduction

Toute réalisation du projet doit être précédée d'une méthode de développement et un langage d'analyse et de conception, qui ont pour but de permettre de définir les étapes préliminaires de développement d'un projet afin de rendre ce développement plus facile et plus flexible aux besoins et d'éviter tout retard au niveau du délai. En se basant sur ce constat, et pour une organisation adéquate de développement du projet et pour faciliter et accélérer la transformation des besoins des utilisateurs en un système logiciel, nous avons opté pour l'approche SCRUM comme processus de travail et UML comme langage de modélisation.

### 1.5.2 Méthode Adoptée : SCRUM

#### *a. Définition*

Le principe de la méthode agile SCRUM est de concentrer l'équipe de développement sur un ensemble de fonctionnalités à réaliser de façon itérative, dans des itérations d'une durée de deux à quatre semaines, appelées des Sprints. Chaque Sprint commence par une estimation suivie d'une planification opérationnelle et se termine par une démonstration de ce qui a été achevé.

#### *b. Pourquoi SCRUM ?*

Avant, le client ne voyait pas l'évolution du travail et n'était pratiquement pas impliqué pendant l'exécution de son projet. En effet, il ne pouvait pas commencer à faire des tests avant que tout soit presque fini. Par contre en Scrum, le client est de plus en plus impliqué dans le processus où il sera appelé à valider les livrables. À chaque livrable, les fonctionnalités sont en amélioration continue et le client voit régulièrement l'évolution des travaux et peut intervenir pour ajuster ses besoins.

C'est pourquoi, Scrum vise essentiellement à optimiser la prévisibilité d'un projet et à mieux contrôler les risques.

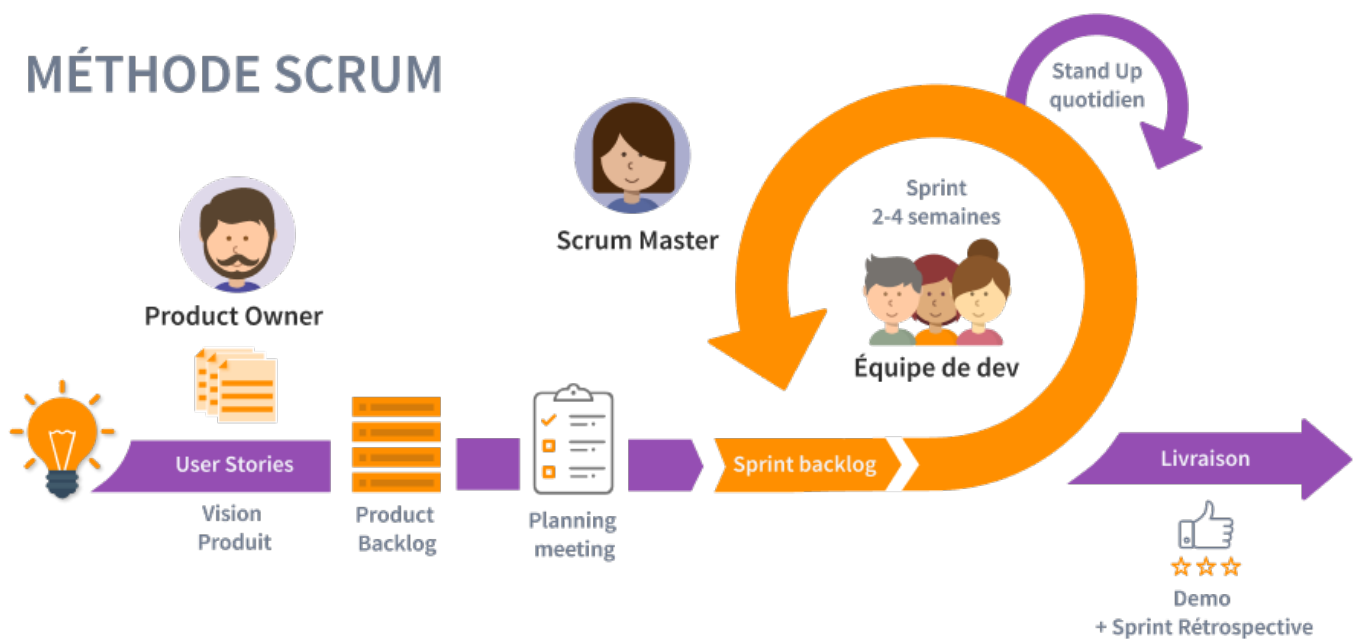


### c. Les Artefacts SCRUM :

Les artefacts Scrum sont des éléments du cadre Scrum qui matérialisent le travail à faire ou la valeur à apporter, ainsi que l'avancement du travail réalisé par l'équipe. Il existe trois artefacts tels que décrit dans le guide :

- **Le Backlog Produit**, ou (*Product backlog* en anglais) : est une liste ordonnée et en constante évolution du travail à réaliser afin d'améliorer le produit.
- **Le Backlog Sprint**, ou (*Sprint backlog* en anglais) : Le Sprint backlog est une liste contenant les spécifications techniques pour les tâches devant être accomplies par l'équipe de développement au cours d'une période donnée (sprint).
- **Livraison**, ou (*Delivery* en anglais) : la version du produit livrée aux parties prenantes après chaque sprint.

La Figure 1.2 représente méthodologie SCRUM.



**Figure 1.2** – Présentation du processus SCRUM

#### *d. Les rôles de la méthode Scrum :*

L'adoption de Scrum requiert la mise en place de 3 rôles spécifiques. Il s'agit du : Product owner, de l'équipe de développement et du Scrum master.

- **Product Owner :** Le product owner (PO) représente le client ou l'utilisateur final. Son rôle est de veiller à ce que le produit soit conforme aux attentes du client que ce soit en termes de qualité ou de valeur ajoutée.
- **Scrum Master :** Il est le leader de l'équipe, il s'assure que le scrum est correctement appliquée et respectée et enlève les obstacles qui peuvent perturber l'avancement des travaux.
- **l'équipe de Développement :** L'équipe de développement est chargée de la mise en œuvre des solutions techniques et de la réalisation des développements requis.

### **1.5.3 Le Langage UML**

Après le choix de la méthodologie de développement, nous avons besoins d'un langage de modélisation unifiée pour la modélisation de notre projet. Pour concevoir notre système, nous avons choisi UML (Unified Modeling Language) comme un langage de modélisation qui couvre les différents vues du projet. Notre choix s'est basé sur les points forts de ce langage notamment sa standardisation et les divers diagrammes qu'il propose.

## **1.6 Conclusion**

Dans ce chapitre, nous avons présenté le cadre général de notre projet en déterminant la problématique et en proposant une solution. Nous avons dévoilé la méthodologie de développement et la méthode de conception qui seront utilisés dans les prochains chapitres de ce rapport et nous avons argumenté notre choix.

# Chapitre 2

## Approches Proposés

### 2.1 Introduction

Dans cette section on va explorer les différentes solutions potentielles et les différents approches qu'on a pris pour améliorer la recherche dans notre projet. Nous aborderons chacune des technologies utilisés en expliquant leurs fonctions principales, leurs utilisations courantes et leurs avantages. Nous discuterons également de l'importance de ces outils dans le domaine de l'analyse de texte et du traitement du langage naturel. En comprenant ces différentes approches, nous serons en mesure d'explorer en détail leurs fonctionnalités spécifiques et leur pertinence pour notre projet de PFE.

### 2.2 Les solutions possibles

Dans cette section on va explorer les solutions potentielles qu'on peut adopter pour notre projet tel que les différentes techniques de recherches et les bases de données qu'on peut utiliser.

#### 2.2.1 Recherche Régulière

Nous aurions pu utiliser la recherche régulière simplement en faisant correspondre le terme de recherche avec les phrases disponibles dans notre base de données SQL déjà existante via des requêtes SQL beaucoup plus optimisés et rapides. Avec cette

approche, il y a moins de travail et complexité, mais des résultats beaucoup moins précis.

### 2.2.2 Utilisation d'une base de données SQL alternatif et performante

Nous aurions pu utiliser une différente base de données SQL plus optimisé et performante comme Cassandra pour l'optimisation de vitesse de recherche et la précision des résultats renvoyés. Avec cette approche la migration des données existantes va être beaucoup plus complexe aussi que l'intégration de cette base de données dans notre application. Mais les résultats vont être plus précis et rapide, mais il reste la limitation la plus importante, qui est la recherche en Arabe avec le dialecte Tunisien, et l'Arabe Traditionnelle. Pour cette raison on a décidé de prendre l'approche d'Elasticsearch avec la recherche vectorielle qui sont dans les sections ci-dessous.

## 2.3 Recherche Vectorielle

La recherche vectorielle est une approche de la récupération des informations qui prend en charge l'indexation et l'exécution des requêtes sur des représentations numériques du contenu. Étant donné que le contenu est numérique plutôt que texte brut, la correspondance est basée sur des vecteurs qui sont les plus similaires au vecteur de requête, ce qui permet la correspondance entre :

- La ressemblance sémantique ou conceptuelle (« chien » et « canine », conceptuellement similaire mais linguistiquement distincte)
- contenu multilingue (« dog » en anglais et « chien » en Français)

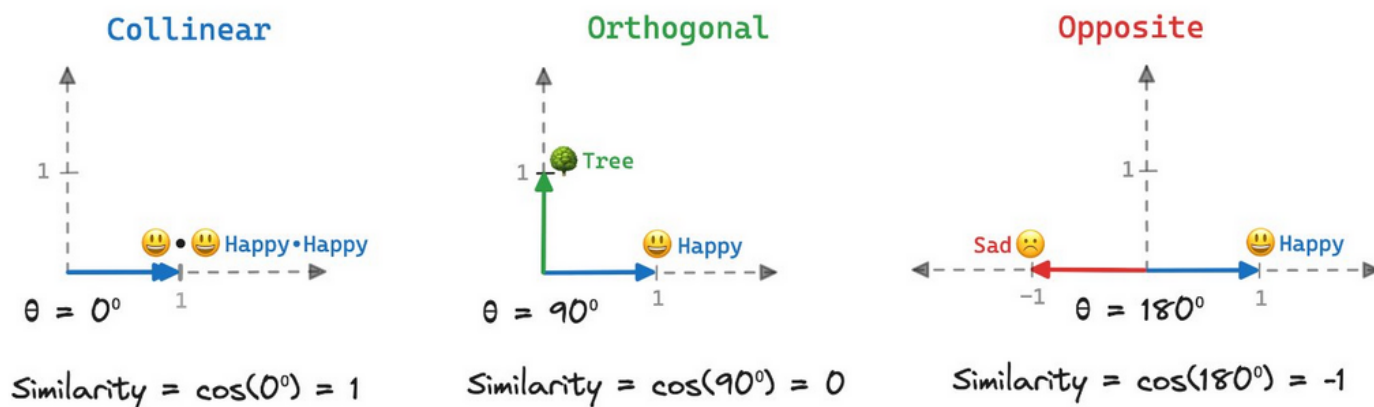
Pour calculer cette similarité Il y a plusieurs approches de calcul, la plus significative, précise et populaire c'est la similarité cosinus qui utilise cette formule mathématique :

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}}$$

Soit  $A$  et  $B$  deux vecteurs de n'importe quelles dimensions, la similarité cosinus mesure l'angle entre deux vecteurs. Plus l'angle est petit, plus les vecteurs sont similaires, le

cosinus de leur angle  $\theta$  s'obtient en prenant leur produit scalaire divisé par le produit de leurs normes. La valeur de cet angle est comprise entre  $[-1, 1]$ , la valeur de -1 indique des vecteurs opposés, la valeur de 0 des vecteurs indépendants (orthogonaux) et la valeur de 1 des vecteurs colinéaires de coefficient positif. Plus que cette valeur s'approche de 1, le plus que les deux vecteurs sont similaires.

### A Geometric intuition to vector similarity! 🚀



**Collinear: The vectors/words are same!**

**Orthogonal: The vectors/words are totally un-related!**

**Opposite: The vectors/words are exact opposites!**

**Figure 2.1** – Présentation du similarité cosinus

## 2.4 Elasticsearch

Elasticsearch est un outil d'analyse de données distribuées open source et hautement évolutif. Il est conçu pour stocker, rechercher et analyser et rechercher de grands volumes de données de manière rapide et efficace en utilisant des différentes méthodes comme Knn Search. Il utilise une structure de données de type index inversé pour indexer et rechercher rapidement des documents. Il prend en charge une variété de types de données, notamment le texte, les nombres, les dates, et les vecteurs par exemple le type dense vector. Avec tout ça, Elasticsearch facilite et optimise la recherche sémantique(vectorielle) en utilisant des indices inversés qui nous permet de

stocker des données de type dense vector (vecteur), qui sont extrêmement efficaces pour la recherche de texte complet. Au lieu de parcourir chaque document et de vérifier la correspondance avec les termes de la requête, Elasticsearch utilise cet indice inversé pour trouver rapidement les documents contenant les termes de recherche. Cela accélère considérablement le processus de recherche, car l'indice fonctionne comme un système de référence rapide.

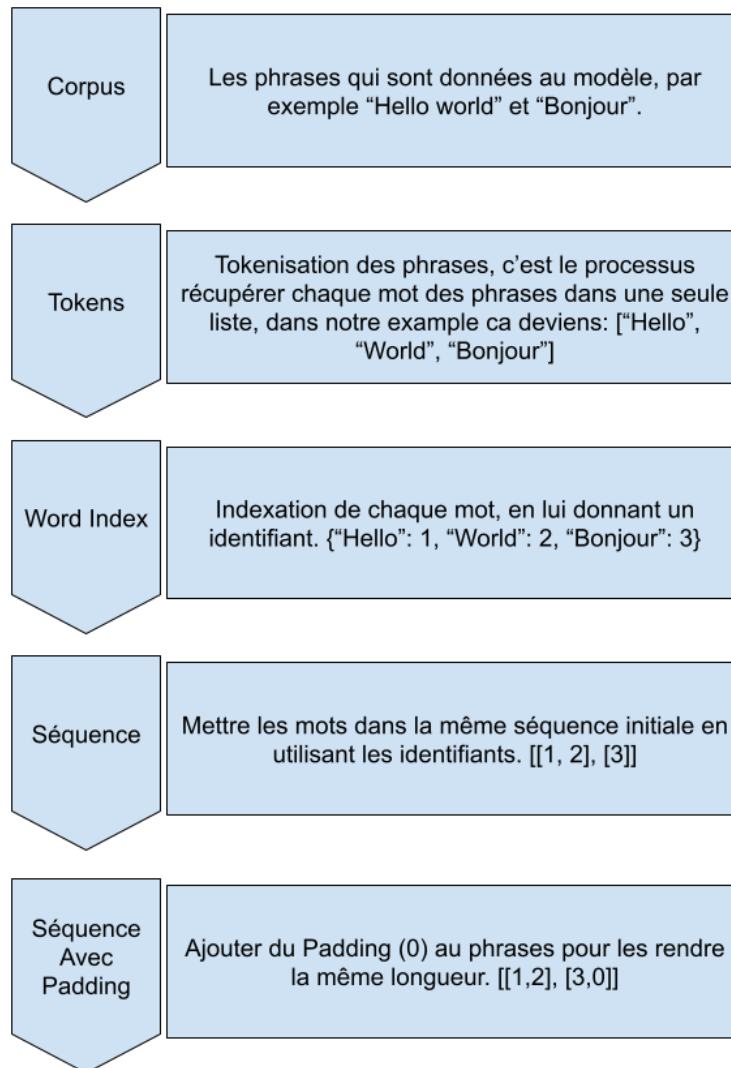
## 2.5 Utilisation d'un modèle Sentence-Transformers

Notre modèle de choix est un modèle appelé « all-mpnet-base-v2 » qui est basé sur le modèle mpnet-base de Microsoft, et puisqu'il est déjà pré-entraînée pour comprendre la langue Française. Il sert à convertir des phrases et des paragraphes en vecteurs sémantiques de 768 dimensions dans un espace vectoriel, avec une limite des paragraphes de 384 mots au maximum. Il est spécifiquement affiné pour établir des correspondances sémantiques précises entre les phrases en les plaçant dans un espace où la similarité cosinus peut être calculée efficacement pour des tâches comme la recherche sémantique, et la mesure de la similarité des phrases.

« All-Mpnet-Base-V2 » (PINECONE, [2024](#))

### 2.5.1 Comment le modèle interprète les phrases ?

Pour interpréter les phrases, le modèle a besoin d'un « Tokenizer » puisqu'il ne peut pas comprendre du texte, on doit convertir chaque phrase en une représentation numérique. Le processus est appelé la « tokenisation » qui est le processus de conversion d'une séquence de caractères en une séquence de jetons(tokens). En PNL, un jeton(token) représente généralement un mot, mais il peut également représenter des sous-mots, des caractères ou d'autres unités, selon le tokeniseur. Ce processus est nécessaire car notre modèle ne peut pas comprendre le texte brut. Au lieu de cela, il travaille avec des représentations numériques de jetons. La figure 2.2 présente le processus de tokenisation.



**Figure 2.2** – Presentation du processus de tokenisation

La figure 2.3 présente un exemple du processus de tokenisation.

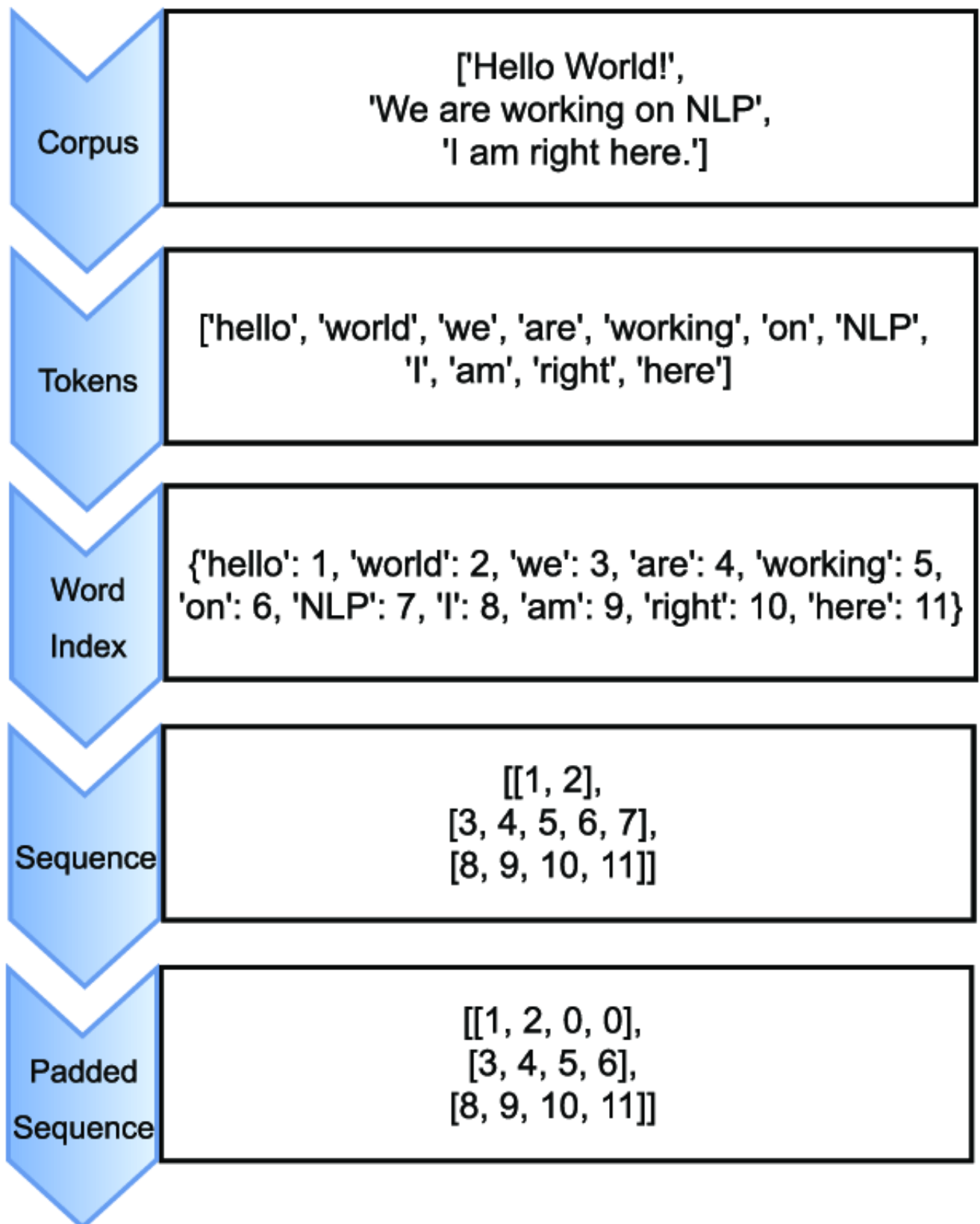


Figure 2.3 – Exemple du processus de tokenisation



### *a. Padding*

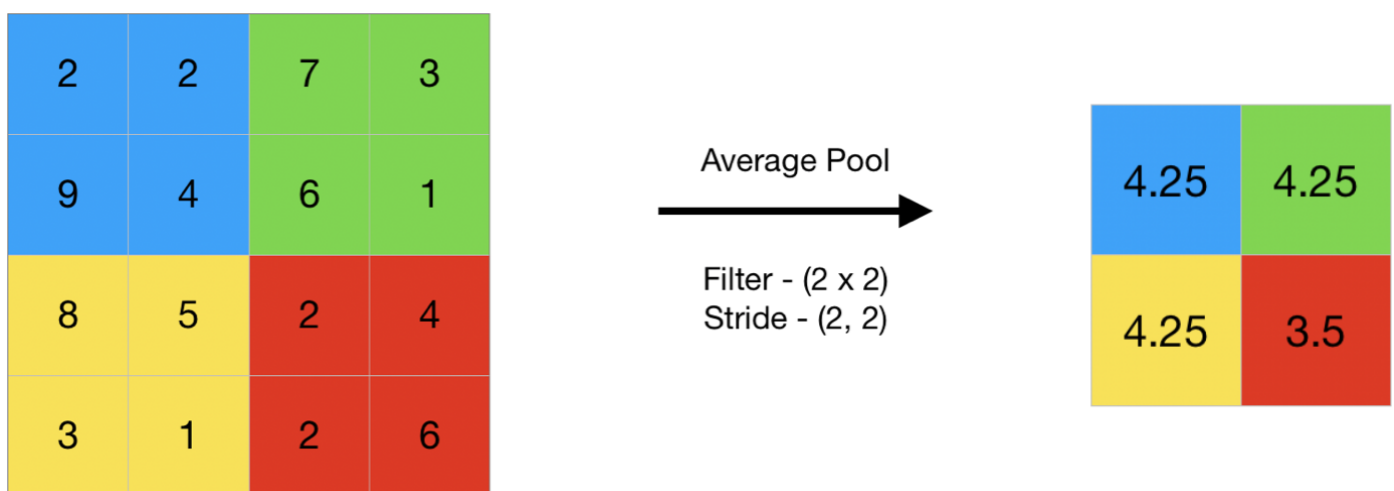
Le remplissage(padding) est appliqué pour garantir que toutes les séquences d'un lot ont la même longueur, ce qui est une exigence pour de nombreuses architectures de réseaux neuronaux. Des jetons de remplissage sont ajoutés à la fin de la séquence pour étendre sa longueur jusqu'à un maximum prédéfini, qui est de 384 dans notre cas.

### *b. Attention Mask (Masque d'attention)*

C'est simplement un masque qui fait la différence entre le contenu et les jetons de remplissage.

### *c. Mean Pooling (Regroupement des moyennes)*

C'est un processus qui calcule efficacement la moyenne de la sequence obtenu après le padding tout en ignorant les jetons de remplissage (padding tokens) qui ont une valeur de 0, ce qui donne un seul vecteur d'intégration qui représente la phrase entière. Ce processus ne prend en compte que les jetons sans remplissage dans la phrase, en utilisant un masque d'attention. Le masque d'attention a la même longueur que la séquence de jetons, « 1 » indiquant les jetons sans remplissage et « 0 » pour les jetons de remplissage. Cela garantit que l'incorporation de phrase résultante est calculée uniquement à partir du contenu significatif de la séquence. La figure 2.4 présente ce processus.



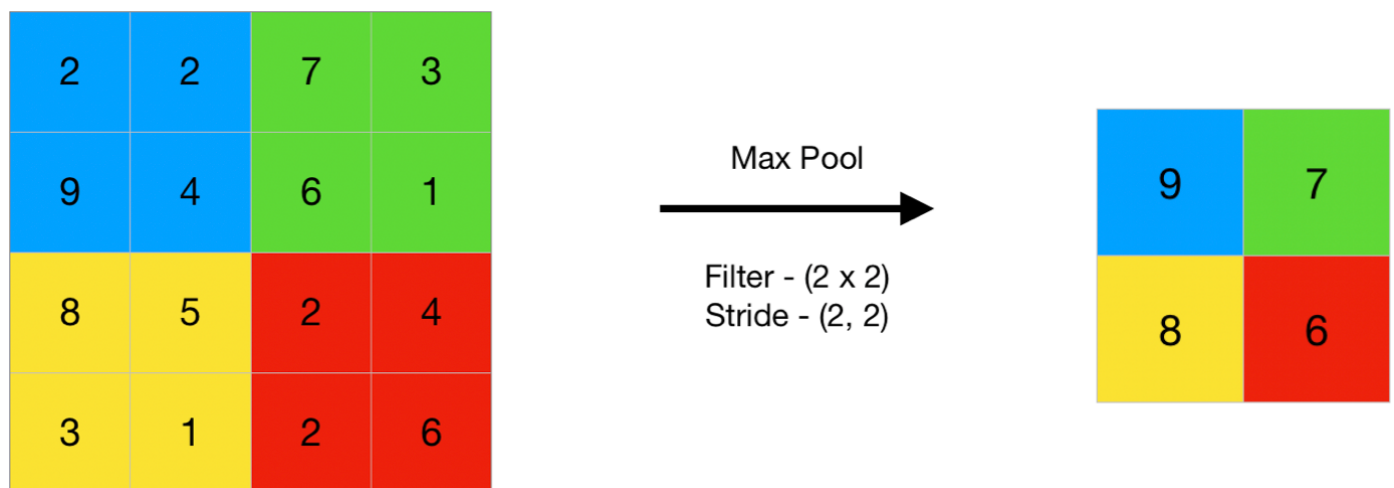
**Figure 2.4** – Présentation du processus de Mean Pooling

## 2.5.2 Pourquoi utiliser Mean Pooling plutôt qu'un autre type de pooling ?

Puisque notre modèle génère un vecteur pour chaque token après les étapes précédentes, le mean pooling consiste à prendre la moyenne de ces vecteurs sur tous les tokens, ce qui résulte en un unique vecteur qui est censé capturer l'essence sémantique de l'ensemble de la phrase, et donner importance à tous les mots de phrase, tous en nous permettant de prendre et interpréter le contexte de toute la phrase. Il existe deux autres types de pooling qu'on a testé sans obtenir de résultats aussi bons :

### *a. Max Pooling*

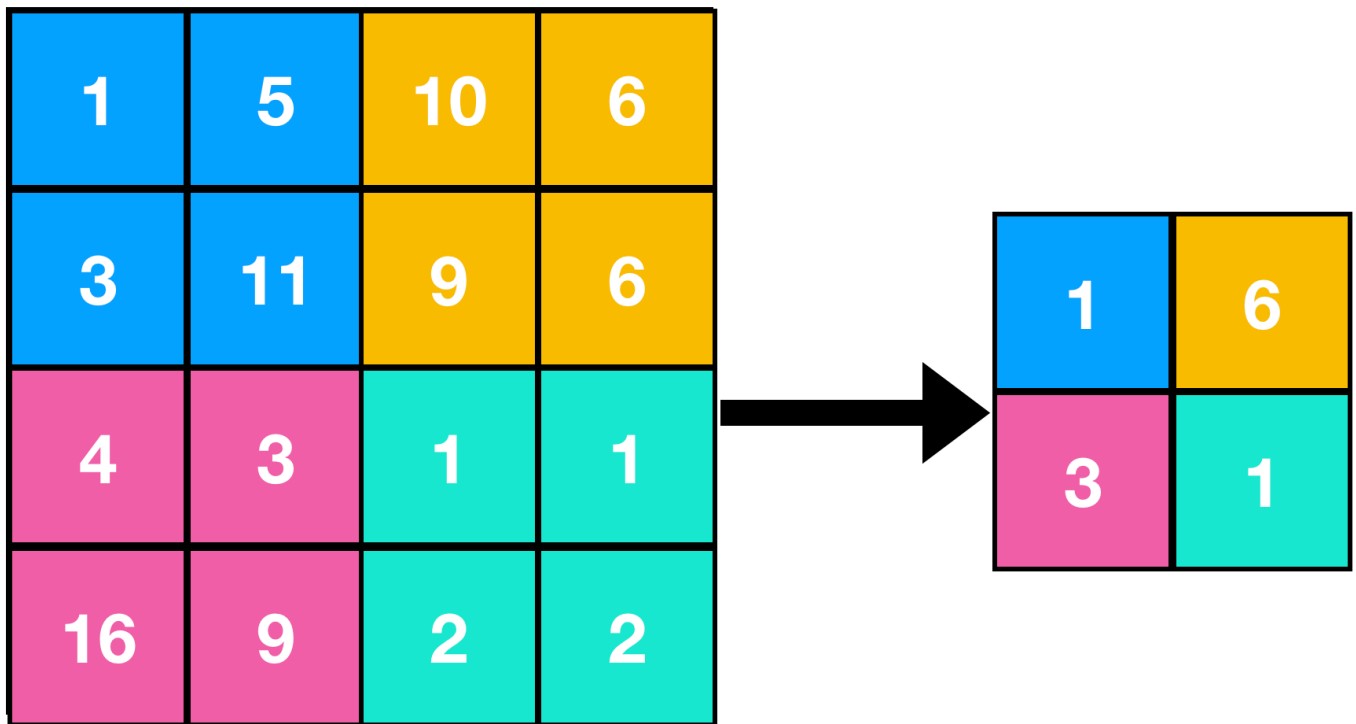
Ce processus consiste à prendre la valeur maximum de ces vecteurs sur tous les tokens au lieu de la moyenne. Le problème avec cette stratégie est ce que elle peut donner plus importance à une mot ou phrase plus que les autres et altérer les résultats de manière négative, ce qui entraîne moins de précision. Par exemple, si un personne cherche pour un « T-shirt bleu », avec le max pooling, le mot « bleu » peut avoir plus d'importance que t-shirt, donc il peut retourner tous les produits bleus, au lieu des t-shirts. La figure 2.5 présente ce processus.



**Figure 2.5** – Présentation du processus de Max Pooling

### *b. Min Pooling*

Ce processus consiste à prendre la valeur minimum de ces vecteurs sur tous les tokens au lieu de la moyenne. La problème avec cette stratégie est ce que elle peut donner moins importance à une mot ou phrase clé moins que les autres et alterner les résultats de manière négative, ce qui aussi entraîne moins de précision. Prenant le même exemple précédent, si un personne cherche pour un « T-shirt bleu », avec le min pooling, le mot « bleu » peut avoir moins d'importance que t-shirt, donc il peut retourner tous les t-shirts sauf les bleus. La figure 2.6 présente ce processus.



**Figure 2.6** – Présentation du processus de Min Pooling

## **2.6 Conclusion**

Dans cette section on a présenté et expliqué les approches qu'on a décidé de prendre ainsi que les technologies qu'on a utilisé et leurs importance, pertinence et les bénéfices qu'ils apportent a notre projet tels que le Recherche Vectorielle, l'Elasticsearch et le modèle sentence-transformers.

# Chapitre 3

## Sprint 0 : Plannification du projet

### 3.1 Introduction

Dans cette section, on va présenter la première phase de la méthode SCRUM, qui est le Sprint 0, qui se commence par l'identification des besoins. Par la suite, nous faisons une analyse globale de notre projet en identifiant les acteurs et ensuite le Diagramme de Cas D'Utilisation globale. De plus, nous planifions le reste des sprints de notre projet. Puis, on va présenter l'architecture du système pour laquelle nous avons opté. Et enfin, on va présenter les outils de développement utilisés pour réaliser ce projet.

### 3.2 Identification des besoins

#### 3.2.1 Les besoins fonctionnels

Les besoins fonctionnels sont les fonctionnalités que le système doit livrer aux utilisateurs. L'outil n'est considéré comme opérationnel que si sa disponibilité fonctionnelle est garantie. Dans le cas de notre système, ces besoins se concentrent sur :

- **Vitesse de recherche :** Améliorer les vitesses de recherche au maximum afin de renvoyer des résultats précis au client.
- **Recherche en Français :** Permettre le client à rechercher les produits dans la langue Française.
- **Recherche en Arabe Tunisienne :** Permettre le client à rechercher les produits dans la langue Arabe Tunisienne.

- **Recherche en Arabe Traditionnel :** Permettre le client a rechercher les produits dans la langage Arabe Classique.
- **Suggestion des produits :** Si le produit recherché par le client n'existe pas, le système tentera de suggérer des produits similaires en prenant le contexte du terme de recherche.

### 3.2.2 Les besoins non fonctionnels

Les exigences non fonctionnels décrivent les objectifs liés aux performances du système et d'autres aspects cruciaux du système qui ne sont pas directement liés à ses fonctionnalités spécifiques. Ils définissent les critères de qualité que le système doit respecter pour répondre aux attentes des utilisateurs. Notre système doit répondre aux exigences non fonctionnels suivantes :

- **La Fiabilité :** L'application doit être fonctionnelle sans détection des erreurs afin de satisfaire les besoins de client.
- **La Sécurité :** Vu que l'application contient des données confidentielles, tous l'accès des produits doivent être protégées par un privilege d'accées.
- **La Disponibilité :** Les services offerts par notre application sont disponibles pendant les 24 heures et durant toute la semaine.
- **La Performance :** L'application doit être rapide et robuste (Vitesse de réponse rapide et précision des résultats lors du recherche des produits dans les langues différents).

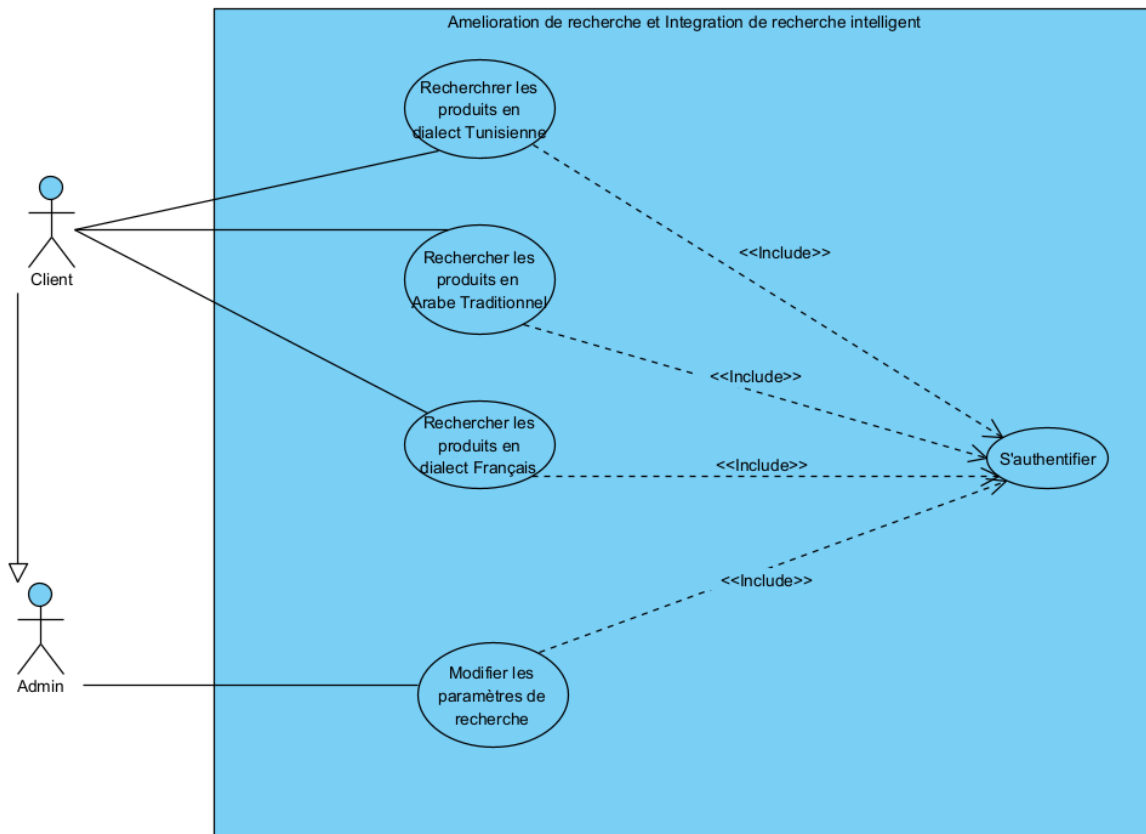
## 3.3 Diagramme de cas d'utilisation global

### 3.3.1 Introduction

Dans cette partie, on va présenter les besoins de notre système de façon formelle à l'aide du diagramme de cas d'utilisation du langage de modélisation UML. D'abord nous identifions les acteurs qui interagissent avec notre système, qui sont :

**Le Client :** C'est le personne qui va accéder à notre système pour rechercher le(s) produit(s) de ses besoins.

**L'Admin :** C'est le personne qui va accéder à notre système pour rechercher le(s) produit(s) dans notre système et modifier les paramètres de son recherche.



**Figure 3.1** – Présentation de Diagramme de Cas D'Utilisation global

## 3.4 Backlog De Produit

Le backlog de produit est une liste de fonctionnalités à réaliser. Ces fonctionnalités sont exprimées sous formes des besoins et sont priorisées par le Product Owner ce qui permet d'établir un ordre de réalisation à respecter.

Notre backlog est composé de trois colonnes :

**ID** : C'est l'identifiant du scénario

**Fonctionnalité** : Permet de mieux ordonner les scénarios.

**Scénario** : Comporte la description des scénarios suivant le forme « En tant que ...  
Je veux »

ID	Fonctionnalité	Scénario
1	Recherche en Arabe Tunisienne	En tant qu'un client, je veux rechercher un ou plusieurs produits dans la langue Tunisienne.
2	Recherche en Arabe Classique	En tant qu'un client, je veux rechercher un ou plusieurs produits en Arabe Classique.
3	Recherche en Français	En tant qu'un client, je veux rechercher un ou plusieurs produits en Français.
4	Vitesse de recherche	En tant qu'un client, je veux avoir une vitesse de recherche rapide.
5	Suggestion des produits	En tant qu'un client, si le produit que je cherche n'existe pas, je veux avoir des suggestions des produits similaires.

**Tableau 3.1** – Table des fonctionnalités et scénarios.

## 3.5 La Plannification De Release

Une fois que le Product Owner a fini de construire le Product Backlog, l'équipe SCRUM et l'environnement technique du projet est bien mis en œuvre, et tous les membres du projet sont planifiés le sprint ensemble aussi que les fonctionnalités à développer pour chaque Sprint, la durée de la prévision de chaque sprint est estimée à quatre semaines.



## 3.6 Architecture du système

Il est indispensable à la conception de tout système informatique de choisir le modèle d'architecture adéquat et pourra assurer le bon fonctionnement, une meilleure performance, ainsi que la scalabilité, la fiabilité et très important, la productivité de développement.

C'est pour cette raison, qu'on a opté pour l'architecture MVC (Modèle, Vue, Contrôleur) comme l'architecture global du projet, et le modèle « Clean Architecture », dans le backend, qui seront très pratique pour gérer les interactions entre les différents composants de notre application. Nous décrirons ces architectures dans les prochaines sections.

### 3.6.1 Architecture "MVC"

Le modèle MVC permet de bien organiser son code source. Il va nous aider à savoir quels fichiers créer, mais surtout à définir leur rôle. Le but de MVC est justement de séparer la logique du code en trois parties que l'on retrouve dans des fichiers distincts.

#### *Modèle*

Cette partie gère ce qu'on appelle la logique métier de notre site. Elle comprend notamment la gestion des données qui sont stockées, mais aussi tout le code qui prend des décisions autour de ces données. Son objectif est de fournir une interface d'action la plus simple possible au contrôleur. On y trouve donc entre autres des algorithmes complexes et des requêtes SQL, et Elasticsearch dans notre cas.

#### *Vue*

Cette partie se concentre sur l'affichage. Elle ne fait presque aucun calcul et se contente de récupérer des variables pour savoir ce qu'elle doit afficher. On y trouve essentiellement du code HTML mais aussi quelques boucles et conditions Javascript très simples, pour afficher par exemple une liste de messages. Il est développé avec React et Typescript.

## Contrôleur

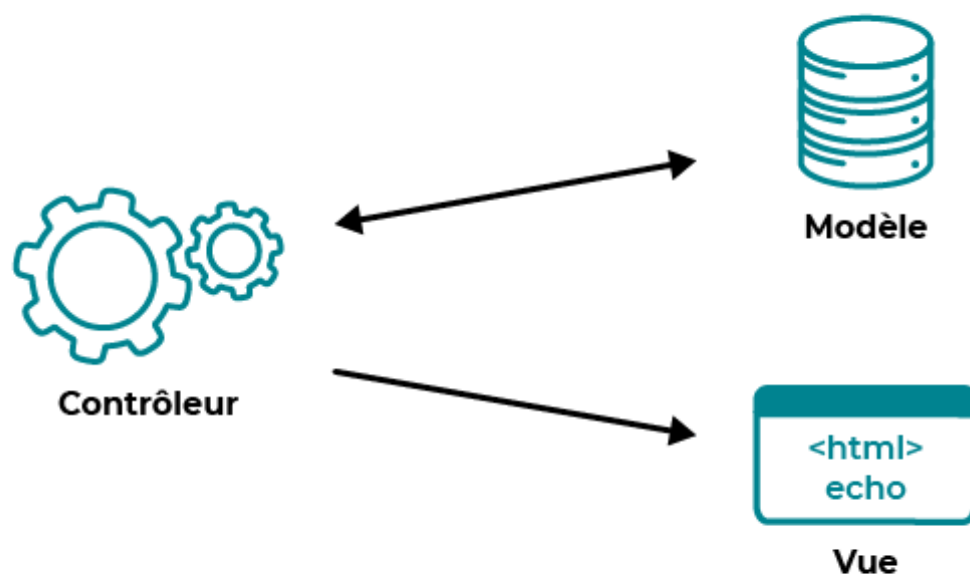
Cette partie gère les échanges avec l'utilisateur. C'est en quelque sorte l'intermédiaire entre l'utilisateur, le modèle et la vue. Le contrôleur va recevoir des requêtes de l'utilisateur. Pour chacune, il va demander au modèle d'effectuer certaines actions (demander les produits en fonction d'un mot-clé de recherche) et de lui renvoyer les résultats (la liste des produits). Puis il va adapter ce résultat et le donner à la vue. Enfin, il va renvoyer la nouvelle page HTML, générée par la vue, à l'utilisateur. Il est implémenté avec ASP .NET Core et C# et utilise des requêtes SQL et Elasticsearch pour manipuler les données.

La figure 3.2 schématise le rôle de chacun de ces éléments.



**Figure 3.2** – L'architecture MVC

Il est important de bien comprendre comment ces éléments s'agencent et communiquent entre eux. La figure 3.3 montre cette communication.



**Figure 3.3** – Échange d'informations entre les éléments MVC

### 3.6.2 Architecture "Clean Architecture"

"Clean Architecture" est un modèle architectural introduit par Robert C. Martin, également connu sous le nom d'Oncle Bob. Il favorise une séparation claire des préoccupations en divisant l'application en couches concentriques, chaque couche ayant ses responsabilités et ses dépendances. Le principe fondamental derrière l'architecture propre est la règle de dépendance, qui stipule que les dépendances doivent toujours pointer vers l'intérieur vers les couches les plus stables et abstraites, plutôt que vers l'extérieur vers des couches plus concrètes et volatiles.

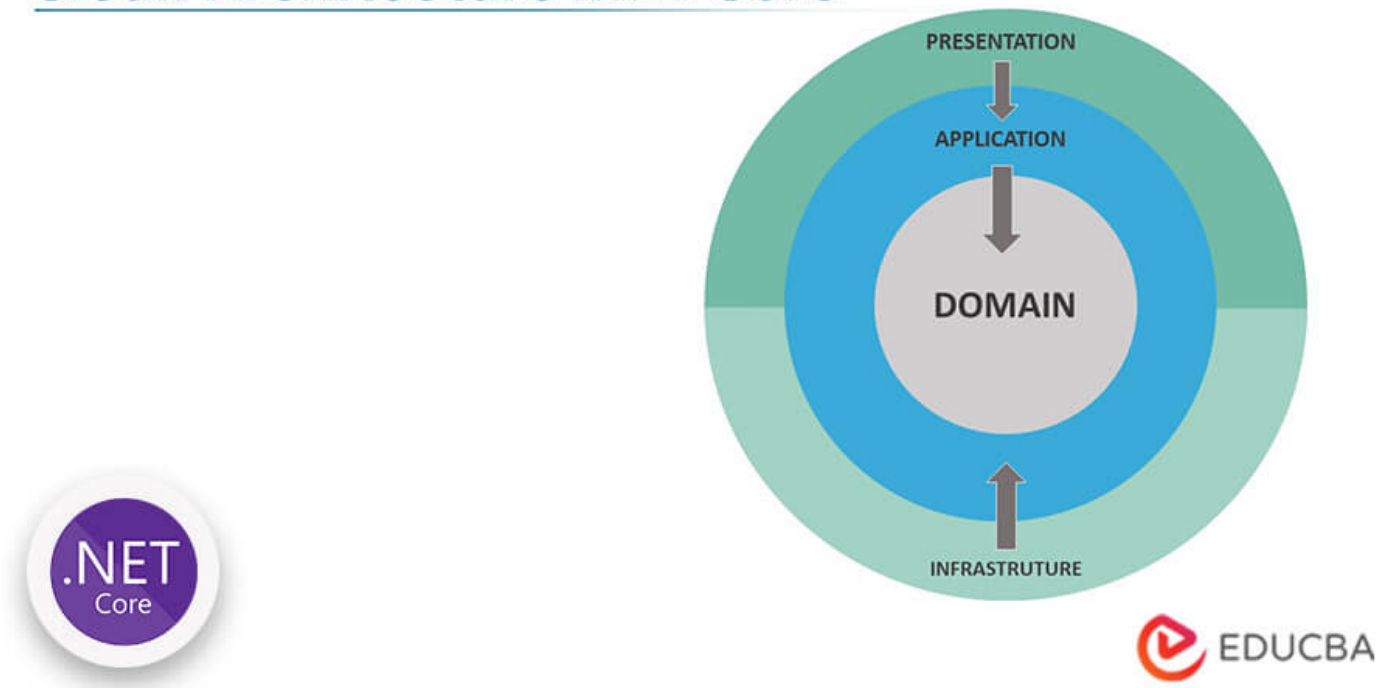
Cette architecture se compose généralement des couches suivantes :

- **La couche Presentation** : Cette couche est responsable de la gestion des interactions des utilisateurs et de la fourniture des données à l'interface utilisateur. Dans notre contexte d'une API Web .NET Core, cette couche comprend les contrôleurs et autres composants qui gèrent les requêtes et les réponses HTTP.

- **La couche Application** : La couche Application contient la logique métier et les cas d'utilisation de l'application. Il agit comme intermédiaire entre la couche présentation et la couche domaine. Cette couche est indépendante de tout problème spécifique d'interface utilisateur ou d'infrastructure.
- **La couche Domain** : La couche Domain représente le noyau de l'application, encapsulant les règles métier, les entités, les interfaces (Orienté Objet) des repositories des bases de données et la logique spécifique au domaine. Il doit être indépendant de la technologie et ne contenir aucune dépendance vis-à-vis de frameworks ou de bibliothèques externes.
- **La couche Infrastructure** : La couche infrastructure traite des problèmes externes tels que les bases de données, les services externes et les frameworks. Il contient des implémentations d'interfaces définies dans la couche application et interagit avec des ressources externes.

La figure Figure 3.4 représente les couches de « Clean Architecture »

## Clean Architecture .NET Core



**Figure 3.4** – Architecture "Clean Architecture" dans ASP. NET Core

Les avantages de l'utilisation de « Clean Architecture » :

- **Vitesse d'implémentation** : La mise en œuvre immédiate vous permet d'implémenter cette architecture avec n'importe quel langage de programmation.
- **Les couches de Domain et Application comme noyau du système** : Les couches Domain et Application sont toujours au centre de la conception et sont connus comme le cœur du système, c'est pourquoi le cœur du système ne dépend pas de systèmes externes.

- **Indépendance des systèmes externes :** Cette architecture permet de changer de système externe sans affecter le cœur du système.
- **Testabilité améliorée du code :** Dans un environnement qui dépend hautement des tests (unitaires et d'intégration), vous pouvez tester votre code rapidement et facilement.
- **Création de produits scalables, robustes et de haute qualité :** Vous pouvez vite créer un système bien performant, organisé, testable, scalable, et robuste.

## 3.7 L’environnement de développement et choix techniques :

### 3.7.1 Introduction

Le développement de notre projet nécessite un ordinateur avec des spécifications puissantes en raison de choses telles que le développement de C# dans Visual Studio, le lancement des containers Docker, le lancement et le stockage de données dans Elasticsearch, l’importation et l’utilisation de notre modèle Sentence-Transformers, et surtout l’exécution de la recherche vectorielle. C’est pour cette raison que nous disposons d’ordinateurs avec les spécifications mentionnées dans la section de l’environnement matériel ci-dessous.

« ECE Hardware Prereq » (ELASTIC, [2024a](#))

### 3.7.2 L’environnement matériel

Dans la table 3.2 nous mentionnons les spécifications des ordinateurs utilisés pour le développement de notre application :

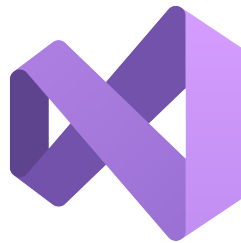
Processeur	Mémoire	Disque dur	Système d’exploitation
11th Gen Intel Core i7 @ 2.304GHz	32Go	1.5To SSD	Windows 11-64bits
9th Gen Intel Core i7 @ 1.8GHz	32Go	1To SSD 1To HDD	Windows 10-64bits

**Tableau 3.2** – Spécifications des ordinateurs utilisés

### 3.7.3 L'environnement logiciel

#### *Visual Studio*

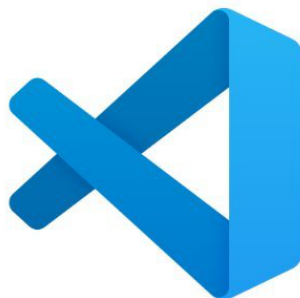
Visual Studio est un environnement de développement intégré (IDE) créé par Microsoft. Il est principalement utilisé pour le développement de logiciels, notamment pour les langages de programmation tels que C#, C++. Pour C# il offre beaucoup des outils qui aide et accélère et améliore l'expérience de développement comme une complétion automatique intelligente, création des classes/interfaces intelligente et un débogueur puissant pour détecter et résoudre les erreurs. La figure 3.5 présente le logo de Visual Studio.



**Figure 3.5** – Logo officiel du logiciel Visual Studio

#### *Visual Studio Code*

Visual Studio Code (VS Code) est un éditeur de code source gratuit et open source développé par Microsoft connu par sa légèreté, rapidité et personnalisation à travers les extensions qu'il fournit pour une diversité des langages de programmation. Il fournit beaucoup de choses qu'un IDE fait comme la coloration syntaxique, la complétion automatique, et le débogage, et la gestion de versions intégré. La figure 3.6 présente le logo de Visual Studio Code.



**Figure 3.6** – Logo officiel du logiciel Visual Studio Code

## *C#*

C# (prononcé "C sharp") est un langage de programmation de haut niveau, orienté objet, développé par Microsoft dans le cadre de sa plateforme .NET. Lancé en 2000, C# a été conçu pour être simple, moderne, sûr et évolutif. C# est utilisé pour le développement d'une variété des applications, notamment des applications backend et des REST API web avec ASP .NET Core. La figure 3.7 présente le logo de C#.



**Figure 3.7** – Logo officiel du langage de programmation C#

## *ASP .NET Core*

ASP.NET Core est un framework open source développé par Microsoft pour la création d'applications web modernes. Il constitue la prochaine évolution de la plateforme ASP.NET, offrant une architecture modulaire, légère et hautement performante. Il offre une large flexibilité et une modularité aux développeurs, il prend en charge principalement le développement des APIs web RESTful qui peuvent être déployées sur Windows, Linux et MacOS. Il offre une variété des fonctionnalités comme le traitement asynchrone des requêtes, le middleware et le pipeline de requêtes personnalisable. La figure 3.8 présente le logo de ASP .NET Core.

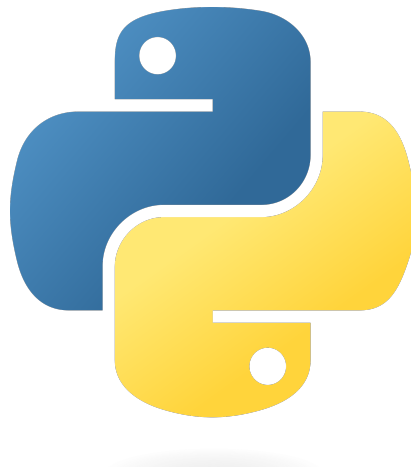




**Figure 3.8** – Logo officiel du framework ASP .NET Core

### *Python*

Python est un langage de programmation interprété, de haut niveau, polyvalent puisqu'il est utilisé dans plusieurs domaines notamment l'analyse de données et le machine learning (avec des bibliothèques telles que NumPy, Pandas, et PyTorch). La figure 3.9 présente le logo de Python.



**Figure 3.9** – Logo officiel du langage de programmation Python

### *Jupyter Notebook*

Jupyter Notebook est une application web open source qui permet de créer et de partager des documents interactifs contenant du code qui sont composés des cellules de code qui peuvent être exécuté individuellement, et facilement partagé à travers des différents sites comme Kaggle, Google Collab. La figure 3.10 présente le logo de Jupyter.



**Figure 3.10** – Logo officiel du framework Jupyter

### *Flask*

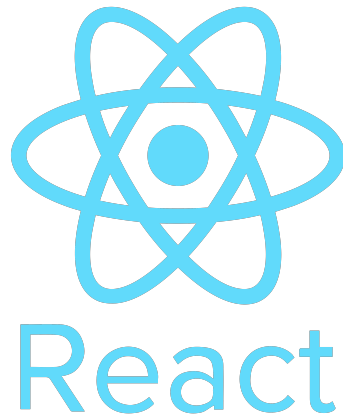
Flask est un framework Python très minimaliste pour développer des REST APIs en Python. Il est très facile de configurer et de faire fonctionner une API. Et pour cette raison, nous l'avons utilisé pour exposer un point de terminaison d'API REST unique afin d'exposer notre modèle Sentence-Transformers. La figure 3.11 présente le logo de Flask.



**Figure 3.11** – Logo officiel du framework Flask

### *React*

React est un framework Javascript gratuit et open-source développé et maintenu par méta(anciennement sous le nom Facebook) en 2013 utilisée principalement pour le développement des interfaces utilisateurs web complexes à travers des "components", ou composants en Français. La figure 3.12 présente le logo de React.



**Figure 3.12** – Logo officiel du bibilothèque React

### *Typescript*

Typescript est un langage de programmation gratuit et open-source développé et maintenu par Microsoft principalement pour améliorer l'expérience de développement pour les développeurs Javascript en fournissant des types statiques, des erreurs directement dans l'éditeur de code, et support totale pour la programmation Orienté Objet. La figure 3.13 présente le logo de Typescript.



**Figure 3.13** – Logo officiel du langage Typescript

### *Docker*

Docker est une plateforme open source qui permet de développer, de déployer et d'exécuter des applications de manière efficace en utilisant des conteneurs logiciels. Les conteneurs sont des unités d'exécution légères et autonomes qui encapsulent une application et tous ses composants, y compris les bibliothèques, la version de langages de programmation utilisè(s), les ports exposès... la figure 3.14 prè sente le logo de Docker.



**Figure 3.14** – Logo officiel du Docker

### *Elasticsearch*

Elasticsearch est un outil d'analyse de données distribuées open source et hautement évolutif. Il est conçu pour stocker, rechercher et analyser et rechercher de grands volumes de données de manière rapide et efficace en utilisant des différentes méthodes comme Knn Search. Il utilise une structure de données de type index inversé pour indexer et rechercher rapidement des documents. Il prend en charge une variété de types de données, notamment le texte, les nombres, les dates, et les vecteurs par exemple le type dense vector. La figure 3.15 présente le logo d'Elasticsearch.



**Figure 3.15** – Logo officiel d'Elasticsearch

### *Kibana*

Kibana est un logiciel de visualisation de données liées à Elasticsearch. Il permet de visualiser et manipuler les index stockés dans Elasticsearch ainsi que l'analyse des données de grandes volumes. La figure 3.16 présente le logo de Kibana.



**Figure 3.16** – Logo officiel du Kibana

### *Postman*

Postman est une plateforme API qui permet de construire, tester et utiliser des APIs en simplifiant et organisant les étapes nécessaires comme la création des workspaces, qui permet un utilisateur de regrouper plusieurs endpoints API dans un workspace aussi que le support de différents types de "body". La figure 3.17 présente le logo de Postman



**Figure 3.17** – Logo officiel du logiciel Postman

### *Overleaf*

Overleaf est une plateforme en ligne permettant un éditeur de texte pour LaTeX sans aucun téléchargement de logiciel, aussi connu comme un SaaS (Software as a Service). Il aussi permet l'écriture collaborative des documents comme celui-ci. La figure 3.18 présente le logo d'Overleaf.



**Figure 3.18** – Logo officiel du Overleaf

### *LaTeX*

LaTeX utilise des commandes de texte pour indiquer comment le document doit être structuré et formaté, plutôt que ce concentrer sur la présentation visuelle. Le document est ensuite compilé en un fichier de format PDF en appliquant les règles typographiques et la mise en page appuyé. La figure 3.19 présente le logo de LaTeX



**Figure 3.19** – Logo officiel du LaTeX

### 3.7.4 Logiciel de modélisation UML

#### *Visual Paradigm*

Visual Paradigm est un outil de conception de diagrammes UML. Il est capable de prendre en charge de nombreux diagrammes commerciaux et techniques comme UML et BPMN. Cette plateforme possède une interface graphique simplifiant la manipulation de ces fonctionnalités comme (Drag & Drop). La figure 3.20 présente le logo de Visual Paradigm.



**Figure 3.20** – Logo officiel du Visual Paradigm

## 3.8 Conclusion

Dans cette section, nous avons préparé notre plan de travail. Nous avons capturé les besoins fonctionnels et non fonctionnels de notre application et nous avons fixé nos choix techniques. Dans le chapitre qui suit nous allons présenter le premier sprint.

# Chapitre 4

## Étude et réalisation du Sprint 1

### 4.1 Introduction

Après avoir examiné l'étude technique de notre projet d'assurance automobile en ligne, nous entamons maintenant le Sprint 1, qui se concentre sur la création d'un système de recherche des produits dans la langue Française.

### 4.2 Backlog du Sprint 1

Le backlog de notre premier Sprint est présenté dans le tableau 4.1.

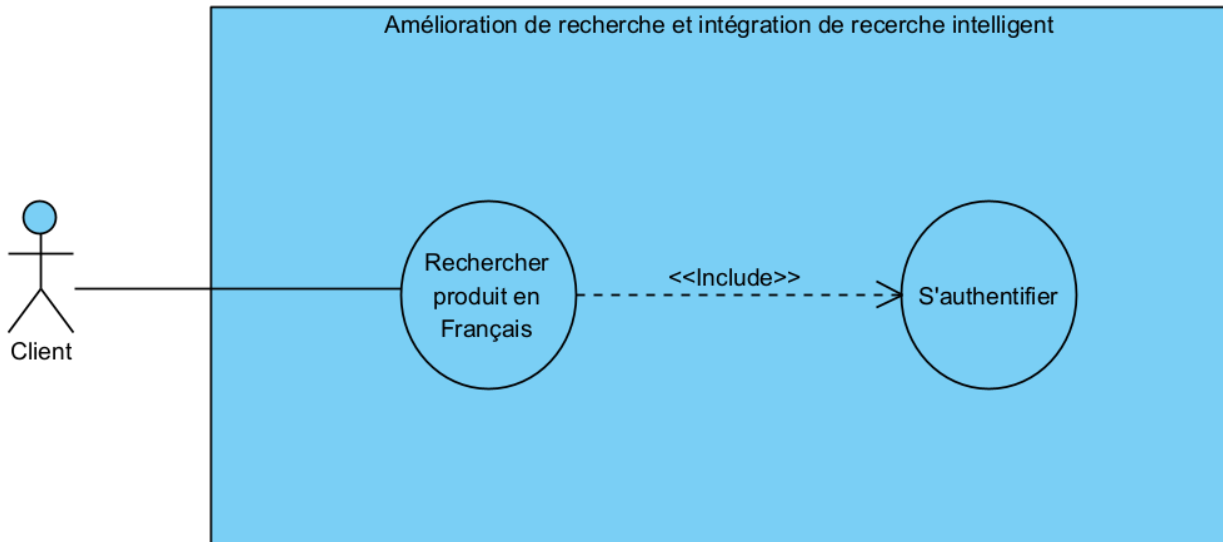
ID	Scénario	Priorité	Complexité
1	En tant qu'un client, je veux saisir ma terme de recherche en Français pour chercher le(s) produit(s)	1	10

**Tableau 4.1** – Backlog du Sprint 1

### 4.3 Spécification fonctionnelle

Dans cette partie, on va expliquer les différentes fonctionnalités du Sprint 1 à travers le diagramme de cas d'utilisation. Puis on va exposer les différents scénarios de notre cas d'utilisation à travers des descriptions textuelles.

### 4.3.1 Diagramme de cas d'utilisation général



**Figure 4.1** – Diagramme de cas d'utilisation général du Sprint 1

### 4.3.2 Description textuelle du CU « Rechercher produit en Français »

**Titre :** Rechercher produit en Français

**Résumé :** Le client saisit son terme de recherche (en Français), en cliquant sur la bouton pour rechercher le(s) produit(s) qu'il veut chercher.

**Acteur Principal :** Client

**Précondition :** Le client est authentifié

**Postcondition :** Le(s) produit(s) que le client cherche est renvoyé, si'il n'existe pas, le système renvoie des produits similaires comme suggestion.

**Scénario de base :**

1. Le client saisit son terme de recherche.
2. Le client clique sur la bouton "Rechercher"
3. Le système prend cette terme de recherche, et performe les étapes nécessaires pour la convertir en vecteur.
4. Le système compare cette vecteur contre les vecteurs dans Elasticsearch.
5. Le système renvoie les produits.

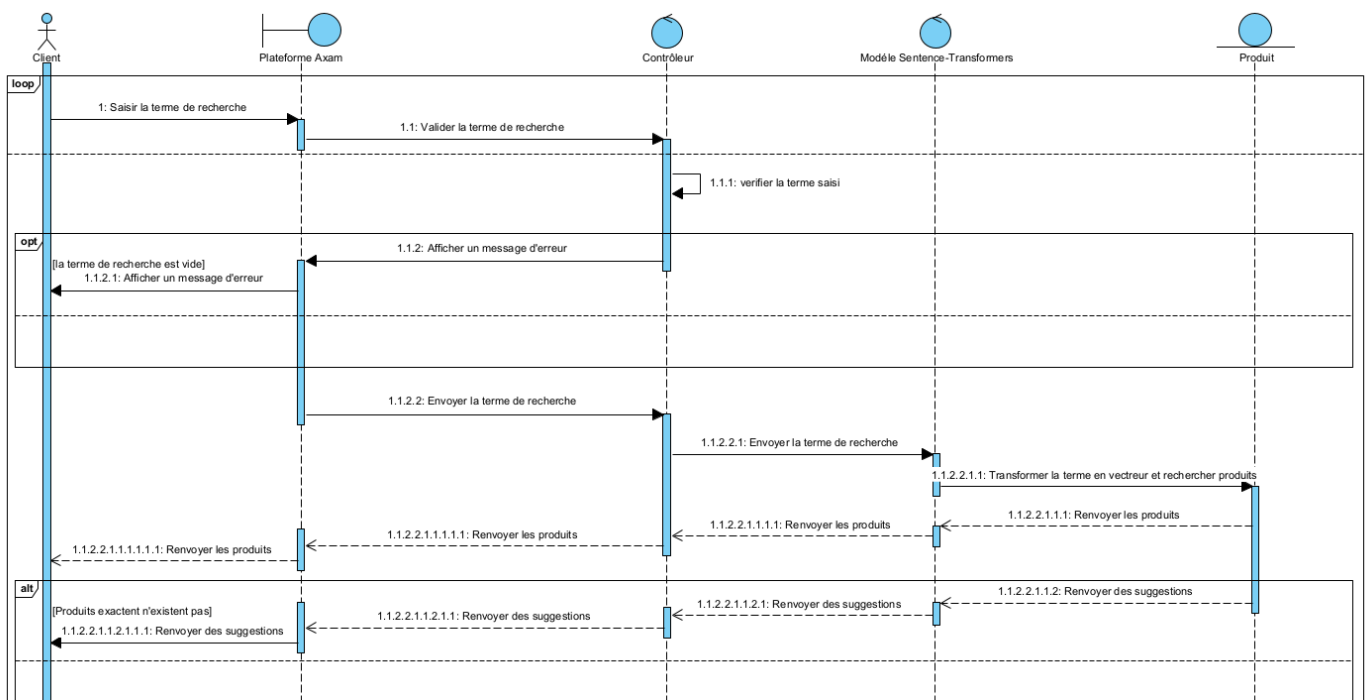


## Scénario alternatifs :

1. La terme de recherche est vide :
  - (a) Le système affiche un message d'erreur informant le client que la terme de recherche est requis.
  - (b) Retour à l'étape 1 du scénario de base.
2. Le(s) produit(s) que le client cherche n'existe pas.
  - (a) Le système essaie de renvoyer les produits les plus similaires comme des suggestions.
  - (b) Retour à l'étape 1 du scénario de base.

### 4.3.3 Diagramme de séquence détaillé

En adoptant l'architecture MVC dans le chapitre précédent, nous avons choisi de suivre ce modèle pour simplifier la création des diagrammes de séquence. Cette section présentera le diagramme de séquence de cas d'utilisation « Rechercher produits en Français » qui est présenté dans la figure 4.2.



**Figure 4.2** – Diagramme de séquence de cas d'utilisation « Rechercher produits en Français »

## 4.4 Architecture de la base de données

Dans le cadre de notre projet, l'intégration d'Elasticsearch et MySQL génère un ensemble de tables indispensables pour garantir son fonctionnement interne. Cependant, nous avons décidé de mettre l'accent uniquement sur les tables spécifiques à notre projet durant la conception.

### 4.4.1 Diagramme de classes

### 4.4.2 Schéma de la base de données

Suite à l'exploration du modèle conceptuel de la base de données pour le premier sprint, nous allons maintenant décrire sa transposition en modèle logique, illustré par les tables dans les sections suivantes.

### 4.4.3 Tables de base de données MySQL

Champs	Type	Contrainte
id	Auto-incrément	Clé primaire
email	String	Non nul
password	String	Non nul
role	String	Non nul

**Tableau 4.2** – Table Client

Champs	Type	Contrainte
id	Auto-incrément	Clé primaire
word	String	Non nul
count	Int	Non nul

**Tableau 4.3** – Table Keyword

#### 4.4.4 Tables de base de données Elasticsearch

**Tableau 4.4** – Table Produit

Champs	Type	Contrainte
code interne	keyword	
image produit	keyword	
code a barre	keyword	
REFERENCE	keyword	
SKU	keyword	
label produit	text	
SEO label produit	text	
categorie	keyword	
sous-categorie	keyword	
sous-sous-categorie	keyword	
categorie_id	integer	
collection	text	
Brève description	text	
Description	text	
Tags	text	
fiche technique	text	
alt image(71 caracteres)	text	
link	keyword	
meta-description	text	
meta title	text	
old_optimization grade	keyword	
new_optimization grade	keyword	
Poids	float	
Couleur	keyword	
color_id	integer	

**Tableau 4.4 – Suivre de la page précédente**

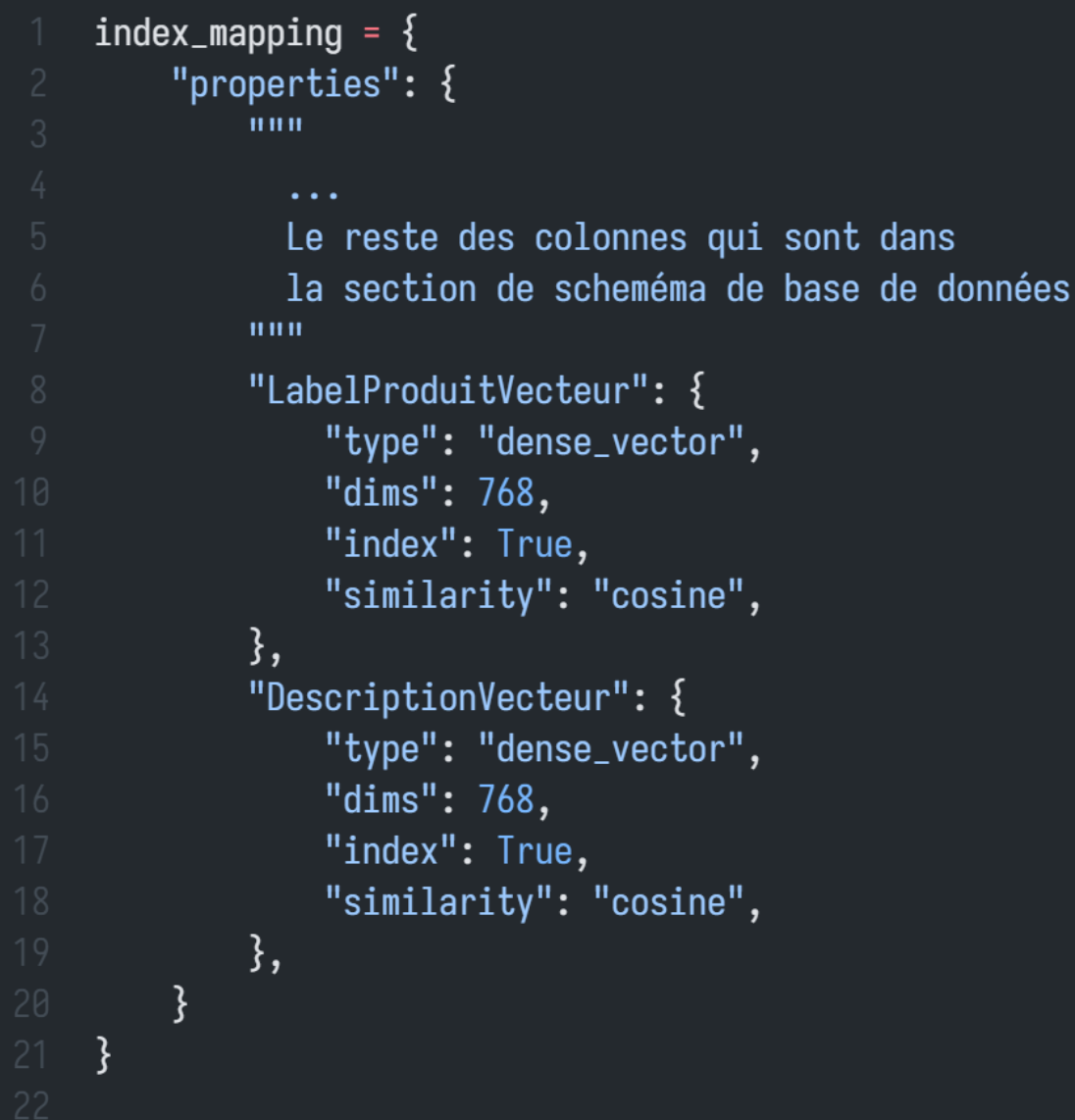
Champs	Type	Contrainte
Marque	keyword	
marque_id	integer	
garantie	keyword	
Stock	float	
fabriqué en	keyword	
est retournable	keyword	
Prix vendeur	float	
Prix brute	float	
Prix Promo	float	
lien (web et video)	keyword	
lien	keyword	
image principale	keyword	
images secondaires	keyword	
seller-id	keyword	
Created by	text	
LabelProduitVecteur	dense_vector	dims : 768, index : true, similarity : cosine
DescriptionVecteur	dense_vector	dims : 768, index : true, similarity : cosine

## 4.5 Réalisation

Cette partie est consacrée à la présentation de l'interface de recherche pour le client et à approfondir les détails de fonctionnement de recherche et Elasticsearch.

### 4.5.1 La création des colonnes des vecteurs

D'abord on a commencé par la création de notre index (table) de produit pour Elasticsearch et insérer les données la. La figure 4.3 montre le code nécessaire pour créer ce index.



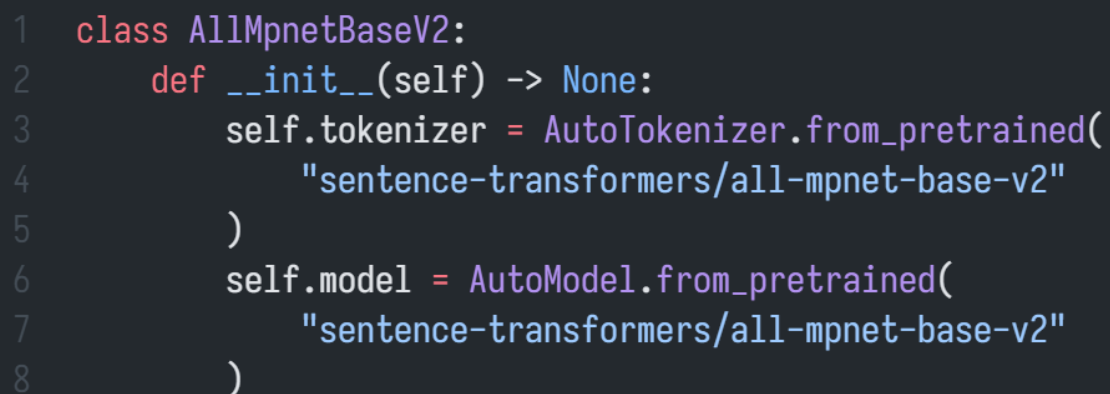
```
1  index_mapping = {
2      "properties": {
3          ""
4          ...
5          Le reste des colonnes qui sont dans
6          la section de schéma de base de données
7          ""
8          "LabelProduitVecteur": {
9              "type": "dense_vector",
10             "dims": 768,
11             "index": True,
12             "similarity": "cosine",
13         },
14         "DescriptionVecteur": {
15             "type": "dense_vector",
16             "dims": 768,
17             "index": True,
18             "similarity": "cosine",
19         },
20     }
21 }
22
```

**Figure 4.3** – Code d'index de Produit

On commence par la création d'un objet « `index_mapping` » qui contient un objet « propriétés » contenant toutes les colonnes de l'index dans Elasticsearch, le reste des colonnes sont les mêmes que dans le tableau 4.4. Concentrons-nous sur les 2 dernières colonnes, qui sont les colonnes les plus importantes, les vecteurs que nous allons utiliser pour notre recherche. Les deux colonnes sont de type `dense_vector`, indiquant qu'elles sont des vecteurs, la contrainte « `dims` » qui a une valeur de 768, indique que ces vecteurs sont à 768 dimensions, « `index` » qui a la valeur `True`, indique qu'ils sont indexables, c'est à dire qu'on peut utiliser cette colonne pour effectuer une comparaison du similarité, qui est de type « `cosine` » qui indique que Elasticsearch va effectuer une similarité cosinus, qu'on va expliquer en plus de détails dans les sections suivantes.

#### 4.5.2 Le modèle Sentence-Transformers et encodage des phrases

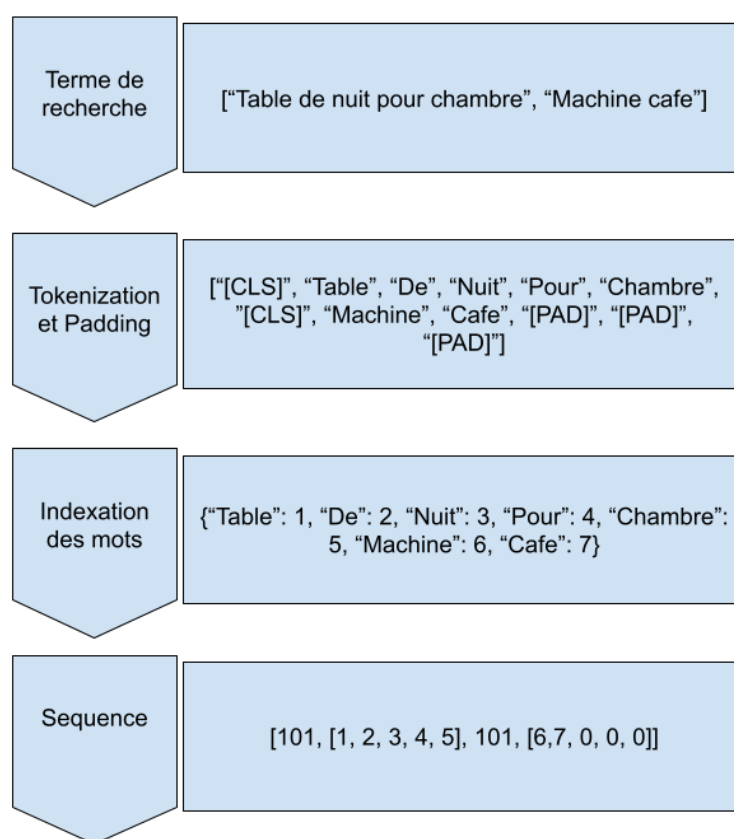
Comme on a mentionné, le modèle Sentence-Transformers qu'on va utiliser est « `all-mpnet-base-v2` », qu'on l'importe de cette façon :



```
1 class AllMpnetBaseV2:
2     def __init__(self) -> None:
3         self.tokenizer = AutoTokenizer.from_pretrained(
4             "sentence-transformers/all-mpnet-base-v2"
5         )
6         self.model = AutoModel.from_pretrained(
7             "sentence-transformers/all-mpnet-base-v2"
8         )
```

**Figure 4.4** – Code d'importation de modèle Sentence-Transformers

On utilise `AutoTokenizer` pour importer le Tokeniser du modèle, aussi que `AutoModel` pour importer le modèle. Le modèle a besoin d'un « Tokenizer » puisqu'il ne peut pas comprendre du texte, on doit convertir chaque phrase en une représentation numérique. Le processus est appelé la « tokenisation » qui est le processus de conversion d'une séquence de caractères en une séquence de jetons(tokens), ce token représente généralement un mot, il y'a aussi des tokens comme le token « CLS » qui est le « Classify Token », il est mis au debut, pour marquer que c'est une phrase, et le token « PAD », qui est utilisé quand il y a plusieurs phrases a tokeniser, et pour ça, il est besoin de rendre toutes les phrases de même longueur. Le token « CLS » a un identifiant de 101, et « PAD » a un identifiant de 0. La figure 4.5 illustre un exemple simple de Tokenisation.



**Figure 4.5** – Exemple de processus de Tokenisation

## Comment le modèle encode une phrase ?

On a créé une méthode appelée "encode\_sentence\_and\_normalise" pour faire l'encodage en faisant les étapes mentionnées dans la partie précédente en ajoutant une autre étape qui est très importante, qui est le Mean Pooling.

D'abord, on utilise le tokenizer pour effectuer la tokenisation et le padding sur la phrase, on a effectué « truncation » à True au cas où la phrase dépasse la limite des mots par phrase pour notre modèle qui est 368 mots, il va seulement prendre les 368 premiers mots si la phrase dépasse la limite. Ensuite, on utilise la méthode « no\_grad » de Pytorch, pour désactiver les « Gradients » et passer les séquences tokenisées au modèle pour faire l'encodage.



## Que'est ce qu'un « Gradient » ?

Un gradient consiste à mettre à jour les poids de chaque neurone de la dernière couche vers la première. Il vise à corriger les erreurs selon l'importance de la contribution de chaque élément à celles-ci. Mais dans notre cas, on désactive les calculs des « gradients » pour un nombre des raisons importantes qui sont cités dans « no\_grad » (PYTORCH, 2023), tels que :

1. **Contrôle du calcul du gradient** : Dans notre cas, le modèle est utilisé pour l'inférence, c'est-à-dire pour générer des vecteurs pour une phrase d'entrée donnée. Puisqu'il n'est pas nécessaire de calculer les gradients pendant l'inférence, l'utilisation de `torch.no_grad()` évite une consommation inutile de mémoire et une surcharge de calcul en désactivant le suivi des gradients.
2. **Optimisation de la mémoire** : Lors de l'inférence, il n'est pas nécessaire de calculer les gradients car les paramètres du modèle ne sont pas mis à jour. En désactivant le calcul du gradient, nous économisons de la mémoire qui serait autrement utilisée pour stocker les informations sur le dégradé. Cela peut être particulièrement important pour les grands modèles ou lorsqu'il s'agit de longues séquences.
3. **Optimisation de la vitesse** : La désactivation du calcul du gradient accélère également le processus, car le framework n'a pas besoin d'effectuer les calculs supplémentaires requis pour le suivi du gradient.

Après que le modèle fais l'encodage de phrase, il nous donne un output qui consiste de plusieurs vecteurs qui représente chaque mot de la phrase. De coup, on a plusieurs vecteurs, c'est à dire chaque mot est isolée, donc on est besoin d'une méthode pour regrouper ces mots, et prendre le contexte de toute la phrase, c'est là qu'intervient la méthode de « Mean Pooling ».

## Le Mean Pooling

Le Mean Pooling est un processus qui calcule efficacement la moyenne de la sequence obtenu après le padding et la tokenisation tout en ignorant les jetons de remplissage (padding tokens) qui ont une valeur de 0, ce qui donne un seul vecteur qui représente la phrase entière. La figure 4.6 illustre un exemple.

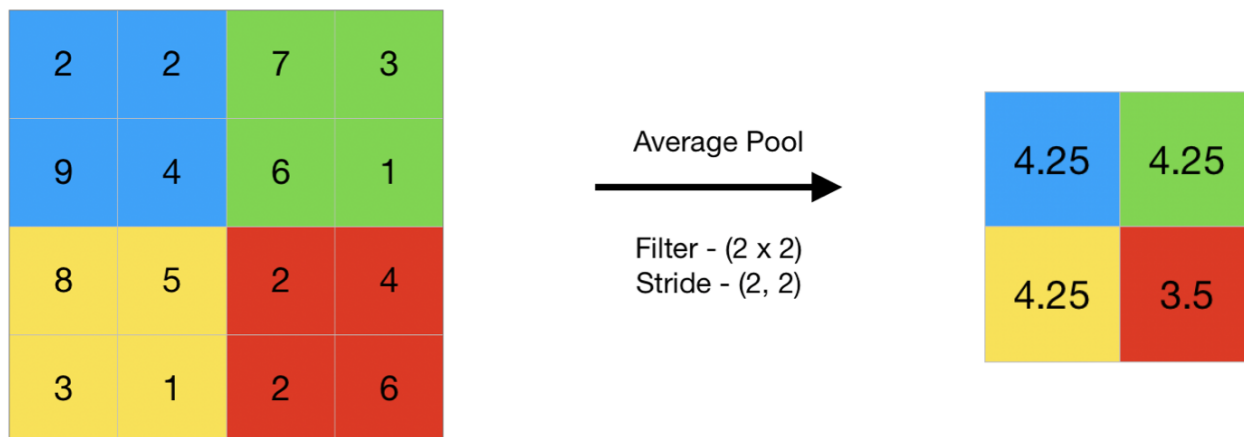


Figure 4.6 – Exemple de Mean Pooling

Nous prenons la moyenne de chaque vecteur et on le mettons dans un seul vecteur. La figure 4.7 présente le code nécessaire pour cette methode.

```
1 def __mean_pooling(  
2     self, model_output: torch.Tensor, attention_mask  
3 ) -> torch.Tensor:  
4     # Le premier élément de l'output du modèle contient toutes les vecteurs de tokens  
5     token_embeddings = model_output[0]  
6  
7     input_mask_expanded = (  
8         attention_mask.unsqueeze(-1).expand(token_embeddings.size()).float()  
9     )  
10  
11     return torch.sum(token_embeddings * input_mask_expanded, 1) / torch.clamp(  
12         input_mask_expanded.sum(1), min=1e-9  
13     )
```

Figure 4.7 – Code de méthode mean\_pooling

Cette méthode consiste de trois étapes, qui sont :

1. **L'extraction des vecteurs des tokens** : `token_embeddings = model_output[0]` : cette ligne récupère les vecteurs de tokens à partir de l'output du modèle. Généralement, pour les modèles de Sentence Transformers, le premier élément de la sortie (`model_output[0]`) contient les intégrations de tous les tokens de la séquence d'entrée.

2. **Extension du masque d'attention** : `input_mask_expanded = attention_mask.unsqueeze(-1).expand(token_embeddings.size()).float()` :

Cette ligne traite le `attention_mask`. Le masque d'attention est simplement un masque qui fait la différence entre le contenu et les jetons de remplissage.

- `unsqueeze(-1)` ajoute une dimension supplémentaire à la fin du `attention_mask`, le rendant compatible en dimensions avec `token_embeddings` lorsque nous appliquons `expand()`.
- `expand(token_embeddings.size())` ajuste le masque pour qu'il corresponde aux dimensions de `token_embeddings`, répétant efficacement le masque pour chaque dimension vecteur.
- `.float()` convertit le masque en float, facilitant les opérations mathématiques ultérieures avec `token_embeddings`.

3. **Application du masque et calcul de Mean Pooling** :

- `torch.sum(token_embeddings * input_mask_expanded, 1)` : ceci calcule la somme des vecteurs dans la dimension de séquence (dimension 1), mais uniquement pour les vecteurs correspondant aux jetons de données réels (pas de padding(0)), comme indiqué par `input_mask_expanded`.
- `torch.clamp(input_mask_expanded.sum(1), min=1e-9)` : La somme des vecteurs est ensuite divisée par la somme de `input_mask_expanded` le long de la dimension de séquence, ce qui donne le nombre de tokens sans padding. `torch.clamp` garantit que nous ne divisons pas par zéro en définissant une valeur minimale (`1e-9`), empêchant ainsi les erreurs de division par zéro.

La figure 4.8 illustre la méthode complète pour l’encodage d’une phrase.

```
1 def encode_sentence_and_normalise(self, sentence: str) -> list:
2     encoded_input = self.tokenizer(
3         sentence, padding=True, truncation=True, return_tensors="pt"
4     )
5
6     # Calculer les intégrations de jetons
7     with torch.no_grad():
8         model_output = self.model(**encoded_input)
9
10    # Performer le "Mean Pooling"
11    sentence_embedding = self.__mean_pooling(
12        model_output, attention_mask=encoded_input["attention_mask"]
13    )
14
15    # Normaliser les intégrations
16    sentence_embedding = F.normalize(sentence_embedding, p=2, dim=1)
17
18    # Presser(.squeeze()) le tenseur dans un tenseur unidimensionnel, puis le retourner sous forme de liste
19    return sentence_embedding.squeeze().tolist()
```

**Figure 4.8** – Méthode `encode_sentence_and_normalise`

### 4.5.3 Elasticsearch et utilisation de la similarité cosinus dans le recherche vectorielle

Pour effectuer notre méthode de recherche qui est la recherche vectorielle, il faut d’abord ajouter les vecteurs avec lesquels nous voulons comparer, et les ajouter dans notre base de données qui est Elasticsearch. Pour effectuer ça, on va utiliser la méthode qu’on a créée « `encode_sentence_and_normalise` » pour générer les vecteurs, mais d’abord, on doit créer une instance de notre modèle, on a appelé cette classe `AllMpnetBaseV2`. Dans notre cas on veut utiliser les colonnes « Brève Description » et « SEO Label Produit », donc on va faire l’encodage pour chaque ligne dans deux nouvelles colonnes « `DescriptionVecteur` » et « `LabelProduitVecteur` ».

La figure 4.9 montre le code pour cette étape.

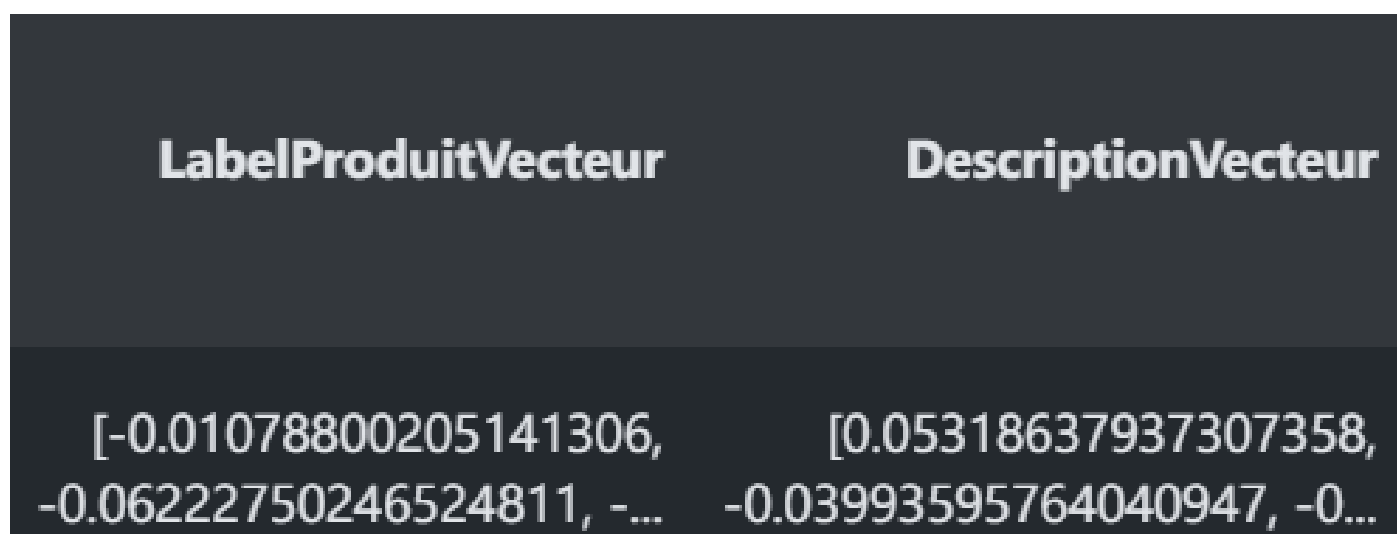
```

1 model = AllMpnnetBaseV2()
2 df["LabelProduitVecteur"] = df["SEO label produit"].apply(lambda x: model.encode_sentence_and_normalise(x))
3 df["DescriptionVecteur"] = df["Brève description"].apply(lambda x: model.encode_sentence_and_normalise(x))

```

**Figure 4.9** – Encodage des deux colonnes Brève Description et SEO Label Produit

Le résultat de cette étape c'est qu'on obtient 2 nouvelles colonnes, « DescriptionVecteur » et « LabelProduitVecteur » qui sont montré dans la figure 4.10.



**Figure 4.10** – les deux nouvelles colonnes « descriptionvecteur » et « labelproduitvecteur »

L'étape suivante consiste de faire une connexion à Elasticsearch, et insérer la les données des produits. D'abord on établit une connexion à notre cluster Elasticsearch qu'on a lancé à partir de Docker, en créant une instance de class Elasticsearch et spécifiant le host, et le basic auth qui consiste de nom utilisateur et mot de passe généré par Kibana. Ensuite, nous créons notre index Elasticsearch en spécifiant notre index\_mapping qu'on a mentionné au debut de ce chapitre à travers la méthode « es.indices.create » qui prends deux paramètres, le nom de l'index et son mapping. Enfin, nous convertissons notre CSV en object Python à travers la méthode « to\_dict » et nous insérons chaque ligne dans Elasticsearch à travers la méthode « index », qui prends trois paramètres « index », « document » et « id ».

La figure 4.11 montre le code nécessaire pour cette étape.

```
1 from elasticsearch import Elasticsearch
2 from index_mapping import index_mapping
3
4 es = Elasticsearch(hosts=["http://localhost:9200/"], basic_auth=("elastic", "123456789"))
5
6 es.indices.create(index="axam_products", mappings=index_mapping)
7
8 record_list = df.to_dict("records")
9
10
11 for record in record_list:
12     record['seller-id'] = str(record['seller-id'])
13     record['code a barre'] = str(record['code a barre'])
14     record['images secondaires'] = str(record['images secondaires'])
15     record['old_optimization grade'] = str(record['old_optimization grade'])
16     record['new_optimization grade'] = str(record['new_optimization grade'])
17     try:
18         es.index(index="axam_products", document=record, id=record["code interne"])
19     except Exception as e:
20         print(e)
```

Figure 4.11 – Insertion des données dans Elasticsearch

# Bibliographie

- DOCKER. (2024). Docker Overview. <https://docs.docker.com/get-started/overview/>
- EDIN ŠAHBAZ. (2023). Getting Started with Clean Architecture in .NET Core. <https://medium.com/@edin.sahbaz/getting-started-with-clean-architecture-in-net-core-fa9151bc5918>
- ELASTIC. (2024a). ECE Hardware Prereq. <https://www.elastic.co/guide/en/cloud-enterprise/current/ece-hardware-prereq.html>
- ELASTIC. (2024b). Elasticsearch. <https://www.elastic.co/>
- META. (2024). React. <https://react.dev/>
- MICROSOFT. (2024a). A tour of the C# language. <https://learn.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/>
- MICROSOFT. (2024b). Vecteurs dans Recherche. <https://learn.microsoft.com/fr-fr/azure/search/vector-search-overview>
- MICROSOFT. (2024c). Visual Studio. <https://visualstudio.microsoft.com/>
- MICROSOFT. (2024d). Visual Studio Code. <https://code.visualstudio.com/>
- MICROSOFT. (2024e). What is ASP .NET Core? <https://dotnet.microsoft.com/en-us/learn/aspnet/what-is-aspnet-core>
- OPENCCLASSROOMS. (2022). Adoptez une architecture MVC. <https://openclassrooms.com/fr/courses/4670706-adoptez-une-architecture-mvc-en-php>
- PINECONE. (2024). All-Mpnet-Base-V2. <https://www.pinecone.io/models/ll-mpnet-base-v2/>
- POSTMAN. (2024). What is Postman? <https://www.postman.com/product/what-is-postman/>
- PYTORCH. (2023). no\_grad. [https://pytorch.org/docs/stable/generated/torch.no\\_grad.html](https://pytorch.org/docs/stable/generated/torch.no_grad.html)