المعهد العالي للتصرف بسوسة
Institut Supérieur de Gestion de Sousse
Sousse Institute of Management

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

Amelioration de recherche et intégration de Recherche Intelligent

Borgi Aaron et Ben Hassine Ghaithallah
3éme LIG

Encadreur de stage : Sghaier Hamza
Référent universitaire : Haddad Ahmed
Année Universitaire : 2023-2024

Remerciements

Nous voudrons 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 project, 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

1 Introduction Générale **1**

1.1 Étude du projet 3

1.1.1 Introduction 3

1.1.2 Cadre du projet 3

1.2 Présentation de l'entreprise 3

1.2.1 Secteurs d'activité 4

1.3 Etude de l'éxistant 5

1.3.1 Analyse de l'existant 5

1.3.2 Critique de l'éxistant 6

1.4 Les Solutions 7

1.5 Méthodologie de développement 8

1.5.1 Introduction 8

1.5.2 Méthode Adoptée : SCRUM 8

1.5.3 Le Langage UML 10

1.6 Conclusion 10

2 Approches Proposés **11**

2.1 Introduction 12

2.2 Les solutions possibles 12

2.2.1 Recherche Régulière 12

2.2.2 Utilisaton de base de donnèes Cassandra 12

2.3 Choix d'approche de Recherche Vectorielle 13

2.4 Recherche Vectorielle 13
2.5 Elasticsearch 15
2.6 Utilisation d'un modéle Sentence-Transformers 16
2.6.1 Comment le modéle intérpréte les phrases ? 16
2.6.2 Pourquoi utiliser Mean Pooling plutôt qu'un autre type de pooling ? 20
2.7 Conclusion 21

3 Sprint 0 : Plannification du projet 22

3.1 Introduction 23
3.2 Identification des besoins 23
3.2.1 Les besoins fonctionnels 23
3.2.2 Les besoins non fonctionnels 24
3.3 Diagramme de cas d'utilisation global 25
3.3.1 Introduction 25
3.4 Backlog De Produit 26
3.5 Plannification des sprints 27
3.6 La Plannification De Release 28
3.7 Architecture du systéme 28
3.7.1 Architecture "MVC Avec Microservices" 29
3.7.2 Architecture "Clean Architecture" 32
3.8 L'environnement de développement et choix techniques : 34
3.8.1 Introduction 34
3.8.2 L'environnement matériel 34
3.8.3 L'environnement logiciel 35
3.8.4 Logiciel de modèlisation UML 42
3.9 Diagramme de deploiement 43
3.10 Conclusion 43

4 Étude et réalisation du Sprint 1 **44**

4.1 Introduction 45

4.2 Backlog du Sprint 1 45

4.3 Spécification fonctionnelle 45

4.3.1 Diagramme de cas d'utilisation général 45

4.3.2 Description textuelle du CU « Rechercher produit en Français » . 46

4.3.3 Description textuelle du CU « Voir suggestions » 47

4.3.4 Diagramme de séquence detaillé 49

4.4 Architecture de la base de données 51

4.4.1 Diagramme de classes 51

4.4.2 Schéma de la base de données 53

4.4.3 Tables de base de données MySQL 53

4.4.4 Tables de base de données Elasticsearch 54

4.5 Réalisation 56

4.5.1 La création des colonnes des vecteurs 56

4.5.2 Le modéle Sentence-Transformers et encodage des phrases 58

4.5.3 Préparation des données dans Elasticsearch pour recherche vectorielle 63

4.5.4 Elasticsearch et Similarité Cosinus 65

4.6 Conclusion 71

5 Étude et réalisation du Sprint 2 **72**

5.1 Introduction 74

5.2 Backlog du Sprint 2 74

5.3 Spécification fonctionnelle 74

5.4 Diagramme de cas d'utilisation général 74

5.5 Description textuelle des cas d'utilisations 75

5.5.1 Description textuelle du CU « Rechercher produit en Arabe traditionnel » 76

5.5.2 Description textuelle du CU « Rechercher produit en Arabe en dialecte Tunisien » 77
5.5.3 Description textuelle du CU « Voir suggestions » 78
5.6 Conception 79
5.6.1 Diagramme de séquence détaillé 79
5.7 Réalisation 81
5.7.1 Les premières approches 81
5.7.2 Création de notre propre classe traducteur 83
5.8 Processus complèt de recherche vectorielle avec la traduction 94
5.8.1 Rèpartition dètaillèe de processus de recherche vectorielle complèt 94
5.8.2 Interface de recherche et affichage des suggestions 96
5.9 Conclusion 97

6 Étude et réalisation du Sprint 3 98

6.1 Introduction 99
6.2 Backlog du Sprint 3 99
6.3 Spéficiation fonctionnelle 99
6.3.1 Diagramme de cas d'utilisation du CU général 100
6.3.2 Description textuelle des cas d'utilisations 100
6.4 Conception 105
6.4.1 Diagrammes de séquence détaillé 105
6.5 Tableau de bord 108
6.5.1 Les indicateurs de performance 109
6.6 Réalisation 110
6.7 Conclusion 113

Bibliographie

Table des figures

1.1 Logo de l'entreprise d'Axam. 4
1.2 Présentation du processus SCRUM 9

2.1 Présentation du similarité cosinus 14
2.2 Présentation du processus de recherche avec recherche vectorielle 15
2.3 Presentation du processus de tokenisation 17
2.4 Exemple du processus de tokenisation 18
2.5 Présentation du processus de Mean Pooling 19
2.6 Présentation du processus de Max Pooling 20
2.7 Présentation du processus de Min Pooling 21

3.1 Présentation de Diagramme de Cas D'Utilisation global 25
3.2 L'architecture MVC avec microservices 31
3.3 Échange d'informations entre les éléments MVC 32
3.4 Architecture "Clean Architecture" dans ASP. NET Core 33
3.5 Logo officiel du logiciel Visual Studio 35
3.6 Logo officiel du logiciel Visual Studio Code 36
3.7 Logo officiel du langage de programmation C# 36
3.8 Logo officiel du framework ASP .NET Core 37
3.9 Logo officiel du langage de programmation Python 37
3.10 Logo officiel du framework Jupyter 38
3.11 Logo officiel du framework Flask 38
3.12 Logo officiel du bibilothéque React 39

3.13 Logo officiel du langage Typescript 39
3.14 Logo officiel du Docker 40
3.15 Logo officiel d'Elasticsearch 40
3.16 Logo officiel du Kibana 41
3.17 Logo officiel du logiciel Postman 41
3.18 Logo officiel du Overleaf 41
3.19 Logo officiel du LaTeX 42
3.20 Logo officiel du Visual Paradigm 42
3.21 Présentation de Diagramme de deploiement global 43

4.1 Diagramme de cas d'utilisation général du Sprint 1 46
4.2 Diagramme de séquence de cas d'utilisation « Rechercher produits en Français » 50
4.3 Diagramme de classes globale 52
4.4 Code d'index de Produit 57
4.5 Code d'importation de modéle Sentence-Transformers 58
4.6 Exemple de processus de Tokenisation 59
4.7 Exemple de Mean Pooling 61
4.8 Code de méthode mean_pooling 62
4.9 Méthode encode_sentence_and_normalise 63
4.10 Encodage des deux colonnes Bréve Déscription et SEO Label Produit . . 64
4.11 les deux nouvelles colonnes « descriptionvecteur » et « labelproduitvecteur » 64
4.12 Insértion des données dans Elasticsearch 65
4.13 Méthode KnnSearchAsync 66
4.14 Processus de recherche véctorielle Elasticsearch 68
4.15 Encodage de terme de recherche et recherche vectorielle dans Elasticsearch 69
4.16 Recherche à partir de Postman 69

4.17 Interface de recherche de client 70

5.1 Diagramme de cas d'utilisation général du Sprint 2 75
5.2 Diagramme de séquence des cas d'utilisations « Rechercher produits en Arabe Traditionnel » et « Rechercher produits en Arabe en dialecte Tunisien » 80
5.3 Code d'initialisation de classe et du dictionnaire 84
5.4 Code de méthode __is_latin 87
5.5 Code de méthode __find_best_matches 88
5.6 Code de méthode translate 92
5.7 Code de endpoint "/encode" 93
5.8 Processus complèt de recherche vectorielle avec la traduction 94
5.9 Interface de recherche de client et visiteur 96
5.10 Affichage des suggestions des noms des produits similaires 97

6.1 Diagramme de cas d'utilisation général de Sprint 3 100
6.3 Diagramme de séquence de CU "Modifier paramétres de recherche" . . . 106
6.2 Diagramme de séquence de CU "Consulter tableau de bord" 107
6.4 Diagramme de séquence de CU "S'authentifier" 108
6.5 Interface de pour l'authentification de l'admin 111
6.6 Interface de modification de paramétres de recherche pour l'admin 112
6.7 Interface de Tableau de bord de l'administrateur 113

Liste des tableaux

1.1 Fiche d'information sur Axam 4
1.2 Comparaison des marketplaces Tunisiennes 6

3.1 Table des fonctionnalités et scénarios. 26
3.2 Spécifications des ordinateurs utilisés 34

4.1 Backlog du Sprint 1 45
4.2 Table Client 53
4.3 Table Keyword 53
4.4 Table Tags 54
4.5 Table Recherche 54
4.6 Table Produit 54

5.1 Backlog du Sprint 2 74

6.1 Backlog du Sprint 2 99
6.2 Tableau des KPIs 110

Chapitre 1

Introduction Générale

Sommaire

1.1 Étude du projet 3

1.1.1 Introduction 3

1.1.2 Cadre du projet 3

1.2 Présentation de l'entreprise 3

1.2.1 Secteurs d'activité 4

1.3 Etude de l'éxistant 5

1.3.1 Analyse de l'existant 5

1.3.2 Critique de l'éxistant 6

1.4 Les Solutions 7

1.5 Méthodologie de développement 8

1.5.1 Introduction 8

1.5.2 Méthode Adoptée : SCRUM 8

1.5.3 Le Langage UML 10

1.6 Conclusion 10

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

Également, le Business Intelligence (BI) joue aussi un rôle trés important pour l'analyse, nettoyage des données des clients et des produits de l'entreprise, et les stocker, ces étapes résultent dans une methode 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 premiér 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 probléme, 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' ´etudes au sein de l'Institut Supérieur de Gestion de Sousse pour l'obtention du diplome de licence en informatique de gestion (Business Intelligence). Il a été réalisé au sein de la start-up Axam. Ce travail est destiné a 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 d´eveloppement 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.

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

Tableau 1.1 – Fiche d'information sur Axam

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

1.2.1 Secteurs d'activité

Axam est une boite 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'éxistant

Cette première ètape de notre projet consitera en une analyse de l'existant, suivie d'une critique de l'existant afain de proposer notre solution comme solution aux problèmes identifiès. Nous baserons notre recherche sur des solutions existantes et opèrationnelles des marchès ecommerce 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 le ur les processus métiers implémentés chez les autres entreprises d'ecommerce, en examinant l'efficacité de ces processus. Cette analyse nous permettra de comprendre comment améliorer les solutions existantes.

Logo	Description	Avantages	Inconvénients
JUMIA	Jumia est un market-place compléte avec un ecosystéme 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 تونيزيانت	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	Wamia est un market-place Tunisienne compléte 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 marketplaces Tunisiennes

1.3.2 Critique de l'éxistant

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

- L'absence de la recherche des produits en utilisant la langue Arabe en dialecte Tunisien (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 traditionnel.
- 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 modéle Sentence Transformer, en s'appuyant sur la plateforme Elasticsearch pour permettre les clients 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 langage Française.
- Pouvoir rechercher un ou plusieurs produits dans la langage Arabe en dialecte Tunisien.
- Pouvoir rechercher un ou plusieurs produits dans la langage Arabe Traditionnel.

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 d´eveloppement 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 systéme 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 accomplis par l'équipe de développement au cours d'une période donnée (sprint).
- **Livraision**, 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

Sommaire

2.1 Introduction 12

2.2 Les solutions possibles 12

2.2.1 Recherche Régulière 12

2.2.2 Utilisaton de base de donnèes Cassandra 12

2.3 Choix d'approche de Recherche Vectorielle 13

2.4 Recherche Vectorielle 13

2.5 Elasticsearch 15

2.6 Utilisation d'un modéle Sentence-Transformers 16

2.6.1 Comment le modéle intérpréte les phrases ? 16

2.6.2 Pourquoi utiliser Mean Pooling plutôt qu'un autre type de pooling ? 20

2.7 Conclusion 21

2.1 Introduction

Dans ce chapitre 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 utilisées 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 donnèes 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 déja 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 Utilisaton de base de donnèes Cassandra

Nous aurions pu utiliser une différente base de donnèes SQL plus optimisé et performante comme Cassandra pour l'optimisation de vitesse de recherche et la prècision des resultats renvoyès. Avec cette approche la migration des donnèes existantes va être beaucoup plus complexe aussi que l'intègration de cette base de donnèes dans notre

application. Mais les rèsultate vont être plus prècis et rapide, mais il reste la limitation la plus importante, qui est la recherche en Arabe avec le dialecte Tunisien, et l'Arabe Traditionnel. Pour cette raison on a dècidè de prendre l'approche d'Elasticsearch avec le recherche vectorielle qui sont dans les sections ci-dessous.

2.3 Choix d'approche de Recherche Vectorielle

En raison des défauts des autres solutions que nous avons mentionnées dans la section précédente, nous avons décidé d'opter pour l'approche de recherche vectorielle, que nous expliquerons plus en détail dans la section suivante.

2.4 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 significante, précis 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 θ s'obtient en prenant leur produit scalaire divisé par le produit

de leurs normes. Le valeur de cette angle est compris entire [-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.

Figure 2.1 – Présentation du similarité cosinus

Comme la figure 2.2 montre, d'abord le client va saisir son terme de recherche (mot clé) qui est ensuite envoyé à notre contrôleur ASP .NET Core, qui ensuite envoie cette mot ou phrase à notre microservice en Flask qui contient notre modéle Sentence-Transformers(all-mpnet-base-v2) dans notre cas, qui ensuite verifient si le mot est en Français, si c'est le cas, en fait l'encodage du phrase directe, sinon, en la traduit soit en utilisant Google Traduction si la phrase est en Arabe Traditionnel, soit en utilisant notre dictionnaire propre des mots du dialecte Tunsienne et aprés faire l'encodage, enfin, le vecteur est renvoyé au contrôleur, qui l'utilise pour faire le recherche vectorielle dans Elasticsearch, et renvoyer les résultats, qui sont les produits dans notre cas.

Figure 2.2 – Présentation du processus de recherche avec recherche vectorielle

2.5 Elasticsearch

Elasticsearch est un outil d'analyse de donnèes distribuès open source et hautement èvolutif. Il est conçu pour stocker, rechercher et analyser et rechercher de grands volumes de données de manière rapide et efficace en utilisant des diffèrents mèthodes 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 sa, 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.

2.6 Utilisation d'un modéle Sentence-Transformers

Notre modéle de choix est un modéle appelé « all-mpnet-base-v2 » qui est basé sur le modèle mpnet-base de Microsoft, et puisqu'il est déja pré-entraînée pour comprendre la langue Française. Il sert a convertir des 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 ([1])

2.6.1 Comment le modéle intérpréte les phrases ?

Pour interpréter les phrases, le modéle est besoin d'un « Tokenizer » puisqu'il ne peux 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 peux pas comprendre le texte brut. Au lieu de cela, il travail avec des représentations numériques de jetons. La figure 2.3 présente le processus de tokenisation.

Figure 2.3 – Presentation du processus de tokenisation

La figure 2.4 présente un exemple du processus de tokenisation.

Figure 2.4 – 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 aprés 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 2.5 présente ce processus.

Figure 2.5 – Présentation du processus de Mean Pooling

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

Puisque notre modéle génère un vecteur pour chaque token aprés 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 do 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. La probléme avec cette stratégie est ce que elle peut donner plus importance à une mot ou phrase plus que les autres et alterner 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 2.6 présente ce processus.

Figure 2.6 – 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 probléme 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 2.7 présente ce processus.

Figure 2.7 – Présentation du processus de Min Pooling

2.7 Conclusion

Dans ce chapitre on a presenté et expliqué les approches qu'on a décidé de prendre ainsi que les technologies qu'on a utilisé et leurs importance, pertinence et les benéfices qu'ils apportent a notre projet tels que le Recherche Vectorielle, l'Elasticsearch et le modéle sentence-transformers.

Chapitre 3

Sprint 0 : Plannification du projet

Sommaire

3.1 Introduction 23

3.2 Identification des besoins 23

3.2.1 Les besoins fonctionnels 23

3.2.2 Les besoins non fonctionnels 24

3.3 Diagramme de cas d'utilisation global 25

3.3.1 Introduction 25

3.4 Backlog De Produit 26

3.5 Plannification des sprints 27

3.6 La Plannification De Release 28

3.7 Architecture du systéme 28

3.7.1 Architecture "MVC Avec Microservices" 29

3.7.2 Architecture "Clean Architecture" 32

3.8 L'environnement de développement et choix techniques : 34

3.8.1 Introduction 34

3.8.2 L'environnement matériel 34

3.8.3 L'environnement logiciel 35

3.8.4 Logiciel de modèlisation UML 42
3.9 Diagramme de deploiement 43
3.10 Conclusion 43

3.1 Introduction

Dans ce chapitre, on va présenter la premiére phase de la méthode SCRUM, qui est le Sprint 0, qui ce 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, en va présnter l'architecture du systéme pour laquelle nous avons opté. Et enfin, en 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 systéme doit livrer aux utilisateurs. L'outil n'est considéré comme opérationnel que si sa disponibilité fonctionnelle est garantie. Dans le cas du notre systéme, 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 a rechercher les produits dans la langage Française.
- **Recherche en Arabe Tunisienne :** Permettre le client a rechercher les produits dans la langage 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 systéme 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 systéme doit repondre 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'accés 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 systéme de façon formelle à l'aide du diagramme de cas d'utilisation du langage de modélisation UML. D'abord nous identifions les acteurs qui intéragissent avec notre systéme, qui sont :

Le Visiteur : C'est le personne qui va accéder à notre systéme pour rechercher le(s) produit(s) dans notre systéme.

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

L'Admin : C'est le personne qui va accéder à notre systéme pour rechercher le(s) produit(s) dans notre systéme et modifier les paramétres 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 fonctionalités 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 en dialecte Tunisien	En tant qu'un client, visiteur, je veux rechercher un ou plusieurs produits dans la langue Tunisienne.
2	Recherche en Arabe Classique	En tant qu'un client, visiteur, je veux rechercher un ou plusieurs produits en Arabe Classique.
3	Recherche en Français	En tant qu'un client, visiteur, je veux rechercher un ou plusieurs produits en Français.
4	Suggestion des produits	En tant qu'un client, visteur, si le produit exacte que je cherche n'existe pas, je veux voir des suggestions des produits similaires.
5	Modifier paramètres de recherche	En tant qu'un administrateur, je veux modifier mes paramètres de recherche des produits.
6	Consulter tableau de bord	En tant qu'un administrateur, je veux s'authentifier et accéder à mon tableau de bord.

Tableau 3.1 – Table des fonctionnalités et scénarios.

3.5 Plannification des sprints

Une fois que le Product Backlog a été établi, la réunion de planification peut être tenue. Le but de cette réunion est de préparer le plan de travail. Le résultat de cette réunion est le choix des durées des Sprints. La liste suivante détaille la planification des Sprints :

Sprint 1 :

1. Rechercher des produits en Français
2. Voir suggestions

Sprint 2 :

1. Rechercher des produits en Arabe Traditionnel
2. Rechercher des produits en Arabe en dialecte Tunisien
3. Voir suggestions

Sprint 3 :

1. Modifier paramètres de recherche
2. Consulter tableau de bord

3.6 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 le membres du projet sont plannifiés 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.

3.7 Architecture du systéme

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

C'est pour cette raison, qu'on a opté pour l'architecture MVC avec des microservices (Modèle, Vue, Contrôleur, Modèle IA, et Elasticsearch) comme l'architecture global du

projet, et le modèle « Clean Architecture », dans le backend, qui seront trés pratique pour gérer les intéractions entre les différents composants de notre application. Nous décrirons ces architectures dans les prochaines sections.

3.7.1 Architecture "MVC Avec Microservices"

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 donnèes.

Modèle IA

Notre modèle IA a une et une seule responsabilité, qui est de faire l'encodage de la requête de l'utilisateur qui est envoyé au contrôleur et retourner le résultat encodé.

Elasticseach

Elasticsearch agit comme notre base de données de recherche de vecteurs qui contient notre index de produits contenant 2 colonnes de vecteurs recherchables à l'aide du vecteur codé renvoyé par le modèle AI.

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

Figure 3.2 – L'architecture MVC avec microservices

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

3.7.2 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 context 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 donnés 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 reprèsente les couches de « Clean Architecture »

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

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

— **Vitesse d'implementation :** 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.

— **Indépendence des systémes externes :** Cette architecture permet de changer de système externe sans affecter le cœur du système.

— **Testabilité améliorée du code :** Dans un environnement qui depends 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 vitement créer un systéme bien performant, organisé, testable, scalable, et robuste.

3.8 L'environnement de développement et choix techniques :

3.8.1 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 ([2])

3.8.2 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

3.8.3 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 beacoup des outils qui aide et accélére et améliore l'expérience de développement comme une complétion automatique intelligente, création des classes/interfaces intelligente et un débogeur 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 déboggage, 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, nottament 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 au développeurs, il prend en charge principalement le développement des APIs web RESTful qui peuvent être déployé sur Windows, Linux et MacOS. Il offre une variété des fonctionallités 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 differents 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 trés minimalistique 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 bibilothéque 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 fournisaant 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 prèsente le logo de Docker.

Figure 3.14 – Logo officiel du Docker

Elasticsearch

Elasticsearch est un outil d'analyse de donnèes distribuès open source et hautement èvolutif. Il est conçu pour stocker, rechercher et analyser et rechercher de grands volumes de données de manière rapide et efficace en utilisant des diffèrents mèthodes 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 prèsente 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 indexes stockées dans Elasticsearch ainsi que l'analyse des données de grandes volumes. La figure 3.16 prèsente 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 régles typographiques et la mise en page appuyé. La figure 3.19 présente le logo de LaTeX

Figure 3.19 – Logo officiel du LaTeX

3.8.4 Logiciel de modèlisation UML

Visual Paradigm

Visual Paradigm est un outil de conception de diagrammes UML.Il est capable de prendre en charge de nombreux diagrammes commer-ciaux et techniques comme UML et BPMN. Cette plateform possedéde une interface graphique simplifiant la manipulation de ces fonctionallités comme (Drag & Drop). La figure 3.20 présente le logo de Visual Paradigm.

Figure 3.20 – Logo officiel du Visual Paradigm

3.9 Diagramme de deploiement

Un diagramme de déploiement est une représentation visuelle de la configuration matérielle et logicielle d'une application, ainsi que des connexions entre les différents composants. Il permet de comprendre comment les différents éléments de l'application sont déployés et interagissent les uns avec les autres. La figure 3.21 présente le diagramme de deploiement de notre application. On peut observer que notre serveur d'application est connecté a un serveur web Kestrel de .NET, aussi que deux bases de données, Elasticsearch, et MySQL, et qu'on a deux services API, ASP .NET Core et Flask qui contient notre modéle Sentence-Transformers.

Figure 3.21 – Présentation de Diagramme de deploiement global

3.10 Conclusion

Dans ce chapitre, nous avons préparé notre plan de travail. Nous avons capturé les besoins fonctionnels et non fonctionnels de notre applcation 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

Sommaire

4.1 Introduction 45

4.2 Backlog du Sprint 1 45

4.3 Spécification fonctionnelle 45

4.3.1 Diagramme de cas d'utilisation général 45

4.3.2 Description textuelle du CU « Rechercher produit en Français » . 46

4.3.3 Description textuelle du CU « Voir suggestions » 47

4.3.4 Diagramme de séquence detaillé 49

4.4 Architecture de la base de données 51

4.4.1 Diagramme de classes 51

4.4.2 Schéma de la base de données 53

4.4.3 Tables de base de données MySQL 53

4.4.4 Tables de base de données Elasticsearch 54

4.5 Réalisation 56

4.5.1 La création des colonnes des vecteurs 56

4.5.2 Le modéle Sentence-Transformers et encodage des phrases 58

4.5.3 Préparation des données dans Elasticsearch pour recherche vectorielle 63

4.5.4 Elasticsearch et Similarité Cosinus 65
4.6 Conclusion 71

4.1 Introduction

Après avoir examiné l'étude technique de notre projet d'amélioration de recherche et intégration de recherche intelligente, nous entamons maintenant le Sprint 1, qui ce concentra sur la création d'un systéme 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, visiteur, je veux saisir ma terme de recherche en Français pour chercher le(s) produit(s)	1	10
2	En tant qu'un client, visiteur, si le produit exacte que je cherche n'existe pas, je veux voir des suggestions des produits similaires.	2	8

Tableau 4.1 – Backlog du Sprint 1

4.3 Spécification fonctionnelle

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

4.3.1 Diagramme de cas d'utilisation général

La figure 4.1 illustre le diagramme de cas d'utilisation gènèral de notre premier sprint, qui est la recherche en Français.

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 boutton pour rechercher le(s) produit(s) qu'il veut chercher.

Acteur Principal : Client, Visiteur

Précondition :

1. Le client (ou le visiteur) est sur la page de recherche
2. Le client (ou le visiteur) a saisi son terme de recherche
3. Le client (ou le visiteur) a cliqué sur "Rechercher"

Postcondition : Le(s) produit(s) que le client cherche est renvoyé, si'il n'existe pas,

le systéme renvoie des produits similaires comme suggestion.

Scénario de base :

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

Scénario alternatifs :

1. Le terme de recherche est vide :
   (a) Le système affiche un message d'erreur informant le client que le 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 systéme essaie de renvoyer les produits les plus similaires comme des suggestions.
   (b) Retour à l'étape 1 du scénario de base.

4.3.3 Description textuelle du CU « Voir suggestions »

Titre : Rechercher produit en Français

Résumé : Aprés que le client a cherché un produit, le systéme essaie de lui suggérer des produits similaires.

Acteur Principal : Client

Précondition :

1. Le client est authentifié.
2. Le client à déja cherché un produit.

Postcondition : Le(s) produit(s) que le client cherche est renvoyé, et le systéme renvoie des produits similaires comme suggestions.

Scénario de base :

1. Le client cherche pour un produit.
2. Le systéme cherche le produit.
3. Le systéme renvoie les produits ainsi que des suggestions des produits similaires.

Scénario alternatifs :

1. Le(s) produit(s) que le client cherche n'existe pas.
   (a) Le systéme essaie de renvoyer les produits les plus similaires comme des suggestions.
   (b) Retour à l'étape 1 du scénario de base.

4.3.4 Diagramme de séquence detaillé

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 génére un ensemble de tables indispnesables 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

Les diagrammes de classes sont l'un des types de diagrammes UML les plus utiles, car ils décrivent clairement la structure d'un système particulier en modélisant ses classes, ses attributs, ses opérations et les relations entre ses objets. *Diagramme de classes* ([3]).

La figure 4.3 montre la conception UML diagramme de classe gènèral de notre projet.

User
-Id : Int
-Email : String
-Password : String
-Role : String
+getId()
+setId(Id) : void
+getEmail()
+setEmail(Email) : void
+getPassword()
+setPassword(Password) : vo
+getRole()
+setRole(Role) : void
+RegisterUser(User user)
+FindByEmail(string email)
Recherche
1
*
faire
Produit
-CodeInterne : String
-ImageProduit : String
-CodeABarre : String
-REFERENCE : String
-SKU : String
-LabelProduit : String
-SEOLabelProduit : String
-Categorie : String
-SousCategorie : String
-SousSousCategorie : String
-CategorieId : int
-Collection : String
-BreveDescription : String
-Description : String
-Tags : String
-FicheTechnique : String
-AltImage : String
-Link : String
-MetaDescription : String
-MetaTitle : String
-OldOptimizationGrade : String
-NewOptimizationGrade : String
-Poids : Float
-Couleur : String
-ColorId : Int?
-Marque : String
-MarqueId : Int
-Garantie : String
-Stock : Float
-FabriqueEn : String
-EstRetournable : String
-PrixVendeur : Float
-PrixBrute : Float
-PrixPromo : Float
-LienWebEtVideo : String
-ImagePrincipale : String
-ImagesSecondaires : String
-SellerId : String
-CreatedBy : String
-DescriptionVecteur : List<Float>
-ProduitLabelVecteur : List<Float>
+getCodeInterne()
+setCodeInterne(CodeInterne) : void
+getImageProduit()
+setImageProduit(ImageProduit) : void
+getCodeABarre()
+setCodeABarre(CodeABarre) : void
+getREFERENCE()
+setREFERENCE(REFERENCE) : void
+getSKU()
+setSKU(SKU) : void
+getLabelProduit()
+setLabelProduit(LabelProduit) : void
+getSEOLabelProduit()
+setSEOLabelProduit(SEOLabelProduit) : void
+getCategorie()
+setCategorie(Categorie) : void
+getSousCategorie()
+setSousCategorie(SousCategorie) : void
+getSousSousCategorie()
+setSousSousCategorie(SousSousCategorie) : void
+getCategorieId()
+setCategorieId(CategorieId) : void
+getCollection()
+setCollection(Collection) : void
+getBreveDescription()
+setBreveDescription(BreveDescription) : void
+getDescription()
Keyword
-Id : Int
-Word : String
-Count : String
+getId()
+setId(Id) : void
+getWord()
+setWord(word) : void
+getCount()
+setCount(Count) : void
+GetKeywordAsync(string keyword)
+AddKeywordAsync(string keyword)
+UpdateKeywordCountAsync(string keyword)
+FindAllAsync(CancellationToken cancellationTo
avoir
*
1
Recherche
-Id : Int
-Date : Date
+getId()
+setId(id) : void
+getDate()
+setDate(Date) : void
*
avoir
1
1
1
Tags
-Id : Int
-Libellé : String
+getId()
+setId(id) : void
+getLibellé()
+setLibellé(libellé) : void

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
recherche_id	Int	Clé étrangère

Tableau 4.3 – Table Keyword

Champs	Type	Contrainte
id	Auto-incrément	Clé primaire
title	String	Non null
product_id	Int	Unique,
Clé étrangère		

Tableau 4.4 – Table Tags

Champs	Type	Contrainte
id	Auto-incrément	Clé primaire
date	DateTime	Non null

Tableau 4.5 – Table Recherche

4.4.4 Tables de base de données Elasticsearch

Tableau 4.6 – 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	

Tableau 4.6 – Suivie de la page précédente

Champs	Type	Contrainte
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	
Marque	keyword	
marque_id	integer	
garantie	keyword	
Stock	float	
fabriqué en	keyword	
est retournable	keyword	
Prix vendeur	float	
Prix brute	float	

Tableau 4.6 – Suivie de la page précédente

Champs	Type	Contrainte
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 4.4 montre le code nécessaire pour créer ce index.

```
index_mapping = {
    "properties": {
        """
          ...
          Le reste des colonnes qui sont dans
          la section de scheméma de base de données
        """
        "LabelProduitVecteur": {
            "type": "dense_vector",
            "dims": 768,
            "index": True,
            "similarity": "cosine",
        },
        "DescriptionVecteur": {
            "type": "dense_vector",
            "dims": 768,
            "index": True,
            "similarity": "cosine",
        },
    }
}
```

Figure 4.4 – 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 4.6. 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 modéle Sentence-Transformers et encodage des phrases

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

```
class AllMpnetBaseV2:
    def __init__(self) -> None:
        self.tokenizer = AutoTokenizer.from_pretrained(
            "sentence-transformers/all-mpnet-base-v2"
        )
        self.model = AutoModel.from_pretrained(
            "sentence-transformers/all-mpnet-base-v2"
        )
```

Figure 4.5 – Code d'importation de modéle Sentence-Transformers

On utilise AutoTokenizer pour importer le Tokeniser du modéle, aussi que AutoModel pour importer le modèle. Le modéle est besoin d'un « Tokenizer » puisqu'il ne peux 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 es 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 » à un identifiant de 101, et « PAD » à un identifiant de 0. La figure 4.6 illustre un exemple simple de Tokenisation.

Figure 4.6 – Exemple de processus de Tokenisation

4.5.2.1 Comment le modèle encode une phrase ?

On a crée une méthode appelé "encode_sentence_and_normalise" pour faire l'encodage en faisant les étapes mentionné dans la partie précédente en ajoutant une autre étape qui est trés importante, qui est le Mean Pooling.
D'abord, on utilise le tokeniser pour effectuer la tokenisation et le padding sur la phrase, en a effectué « truncation » a True au cas où la phrase dépasse la limite des mots par phrase pour notre modéle qui est 368 mots, il va seulement prendre les 368 premiéres mots si la phrase dépasse la limite. Ensuite, on utilise la méthode « no_grad » de Pytorch, pour désactiver les « Gradients » et passer les séquences tokenisés au modéle pour faire l'encodage.

4.5.2.2 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 dèsactive les calculs des « gradients » pour un nombre des raisons importantes qui sont citès dans *no_grad* ([4]), 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 modéle 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 ».

4.5.2.3 Le Mean Pooling

Le Mean Pooling est un processus qui calcule efficacement la moyenne de la sequence obtenu aprés 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 4.7 illustre un exemple.

Figure 4.7 – Exemple de Mean Pooling

Nous prenons la moyenne de chaque vecteur et on le mettons dans un seul vecteur. La figure 4.8 présente le code nécessaire pour cette methode.

```python
def __mean_pooling(
    self, model_output: torch.Tensor, attention_mask
) -> torch.Tensor:
    # Le premier élément de l'output' du modèle contient toutes les vecteurs de tokens
    token_embeddings = model_output[0]

    input_mask_expanded = (
        attention_mask.unsqueeze(-1).expand(token_embeddings.size()).float()
    )

    return torch.sum(token_embeddings * input_mask_expanded, 1) / torch.clamp(
        input_mask_expanded.sum(1), min=1e-9
    )
```

Figure 4.8 – Code de méthode mean_pooling

Cette méthode consiste de trois étapes, qui sont :

1. **L'éxtraction 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.

Aprés l'étape de Mean Pooling, on obtient un vecteur qui reprèsente la phrase, que l'on ensuite normalise avec la fonction normalize de Pytorch. La figure 4.9 illustre la méthode compléte pour l'encodage d'une phrase.

```python
def encode_sentence_and_normalise(self, sentence: str) -> list:
    encoded_input = self.tokenizer(
        sentence, padding=True, truncation=True, return_tensors="pt"
    )

    # Calculer les intégrations de jetons
    with torch.no_grad():
        model_output = self.model(**encoded_input)

    # Performer le "Mean Pooling"
    sentence_embedding = self.__mean_pooling(
        model_output, attention_mask=encoded_input["attention_mask"]
    )

    # Noramliser les intégrations
    sentence_embedding = F.normalize(sentence_embedding, p=2, dim=1)

    # Presser(.squeeze()) le tenseur dans un tenseur unidimensionnel, puis le retourner sous forme de liste
    return sentence_embedding.squeeze().tolist()
```

Figure 4.9 – Méthode encode_sentence_and_normalise

4.5.3 Préparation des données dans Elasticsearch pour recherche vectorielle

Pour effectuer notre méthode de recherche qui est le 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 crée « encode_sentence_and_normalise » pour générer les vecteurs, mais d'abord, on doit créer une instance de notre modéle, on a appelé cette classe

AllMpnetBaseV2. Dans notre cas on veux utiliser les colonnes « Bréve Description » et « SEO Label Produit », donc on va faire l'encodage pour chaque ligne dans deux nouveaux colonnes « DescriptionVecteur » et « LabelProduitVecteur ».
La figure 4.10 montre le code pour cette étape.

```
model = AllMpnetBaseV2()
df["LabelProduitVecteur"] = df["SEO label produit"].apply(lambda x: model.encode_sentence_and_normalise(x))
df["DescriptionVecteur"] = df["Brève description"].apply(lambda x: model.encode_sentence_and_normalise(x))
```

Figure 4.10 – Encodage des deux colonnes Bréve Déscription 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 4.11.

LabelProduitVecteur	DescriptionVecteur
[-0.01078800205141306, -0.06222750246524811, -...	[0.05318637937307358, -0.03993595764040947, -0...

Figure 4.11 – 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 4.12 montre le code nécessaire pour cette étape.

```python
from elasticsearch import Elasticsearch
from index_mapping import index_mapping

es = Elasticsearch(hosts=["http://localhost:9200/"], basic_auth=("elastic", "123456789"))

es.indices.create(index="axam_products", mappings=index_mapping)

record_list = df.to_dict("records")


for record in record_list:
    record['seller-id'] = str(record['seller-id'])
    record['code a barre'] = str(record['code a barre'])
    record['images secondaires'] = str(record['images secondaires'])
    record['old_optimization grade'] = str(record['old_optimization grade'])
    record['new_optimization grade'] = str(record['new_optimization grade'])
    try:
        es.index(index="axam_products", document=record, id=record["code interne"])
    except Exception as e:
        print(e)
```

Figure 4.12 – Insértion des données dans Elasticsearch

4.5.4 Elasticsearch et Similarité Cosinus

Aprés qu'on a préparé notre données dans Elasticsearch, en ajoutant les colonnes des vecteurs qu'on va comparer contre, on peut maintenant performer la recherche vectorielle en utilisant Elasticsearch.

Quand un client saisi son terme de recherche à travers notre interface, il est envoyé à notre API ASP .NET Core, qui ensuite l'envoie à notre API Flask qui contient notre

modéle Sentence-Transformers, pour effectuer les étapes nécessaires pour l'encodage du phrase en vecteur, et ensuite le renvoyer au contrôleur pour l'utiliser dans un Elasticsearch query knn, pour ça on a crée la méthode « KnnSearchAsync », qui est montré dans la figure 4.13.

```
private async Task<ISearchResponse<T>> KnnSearchAsync<T>(List<float> queryVector, KnnSearchRequest knnSearchRequest) where T : class
{
    var query = new
    {
        knn = new[]
        {
            new
            {
                field = "LabelProduitVecteur",
                query_vector = queryVector,
                k = knnSearchRequest.TopResProdLabel,
                num_candidates = knnSearchRequest.NumCandidatesProdLabel,
            },
            new
            {
                field = "DescriptionVecteur",
                query_vector = queryVector,
                k = knnSearchRequest.TopResDesc,
                num_candidates = knnSearchRequest.NumCandidatesDesc,
            },
        },
    };

    var response = await _elasticClient.LowLevel.SearchAsync<SearchResponse<T>>(IndexName, PostData.Serializable(query));
    return response;
}
```

Figure 4.13 – Méthode KnnSearchAsync

Cette méthode prend 2 paramètres qui sont :

- **queryVector :** Qui est une liste des réels représentant le vecteur encodé.
- **knnSearchRequest :** Qui est une classe qui prend 4 attributs utilisé dans note query knn :
  1. **TopResProdLabel :** Un entier représentant le nombre des meilleurs résultats par recherche vectorielle de « LabelProduit », valeur par dèfaut : 5.
  2. **TopResDesc :** Un entier représentant le nombre des meilleurs résultats par recherche vectorielle de « Description », valeur par dèfaut : 5.
  3. **NumCandidatesDesc :** Un entier représentant le nombre total des produits que nous souhaitons rechercher par la description, valeur par dèfaut : 20.
  4. **NumCandidatesProdLabel :** Un entier représentant le nombre total des produits que nous souhaitons rechercher par la label produit, valeur par dèfaut : 10.

Cette recherche trouve les principales correspondances vectorielles globales k = 25

pour le vecteur « LabelProduit » et la valeur globale k = 25 aussi pour le vecteur « Description ». Ces valeurs principales sont ensuite combinées avec les correspondances de la requête de correspondance et les 10 premiers documents sont renvoyés si k > 10, sinon le nombre de résultats renvoyés sont calculé comme ça :

$$kTopResProdLabel + kTopResDescription$$

$$\text{Si } kTopResProdLabel < 10 \text{ Et } kTopResDescription < 10$$

Les multiples entrées knn et les correspondances de requêtes sont combinées via une disjonction, comme si vous preniez un booléen ou entre elles. Les k premiers résultats vectoriels représentent les voisins globaux les plus proches sur toutes les partitions d'index. La notation d'un document avec les améliorations configurées ci-dessus serait comme le suivant, notons :

- $\mathbf{q}$ comme vecteur de requête.
- $\mathbf{v}_{\text{label}}$ and $\mathbf{v}_{\text{desc}}$ comme vecteurs de document dans les champs "LabelProduitVecteur" et "DescriptionVecteur", respectivement.
- $\text{score}_{\text{label}}$ and $\text{score}_{\text{desc}}$ comme les scores basés sur ces vecteurs.

En utilisant la similarité cosinus, le score de chaque composant peut être calculé comme suit :

$$\text{score}_{\text{label}}(\mathbf{q}, \mathbf{v}_{\text{label}}) = \frac{\mathbf{q} \cdot \mathbf{v}_{\text{label}}}{\|\mathbf{q}\| \|\mathbf{v}_{\text{label}}\|} = \frac{\sum_{i=1}^{n} q_i v_{\text{label},i}}{\sqrt{\sum_{i=1}^{n} q_i^2} \sqrt{\sum_{i=1}^{n} v_{\text{label},i}^2}}$$

$$\text{score}_{\text{desc}}(\mathbf{q}, \mathbf{v}_{\text{desc}}) = \frac{\mathbf{q} \cdot \mathbf{v}_{\text{desc}}}{\|\mathbf{q}\| \|\mathbf{v}_{\text{desc}}\|} = \frac{\sum_{i=1}^{n} q_i v_{\text{desc},i}}{\sqrt{\sum_{i=1}^{n} q_i^2} \sqrt{\sum_{i=1}^{n} v_{\text{desc},i}^2}}$$

$$\text{Score Combinée} = \alpha \cdot \text{score}_{\text{label}} + \beta \cdot \text{score}_{\text{desc}}$$

Ici :

- $\cdot$ représente le produit scalaire.
- $\|\mathbf{q}\|$ and $\|\mathbf{v}\|$ désignent les normes (grandeurs) des vecteurs, qui sont calculées comme la racine carrée de la somme des composantes au carré des vecteurs.

- α and β sont des pondérations qui pourraient être utilisées pour équilibrer l'importance des scores des deux domaines (en supposant une importance égale, les deux pourraient être fixés à 0,5 par défaut, ou ajustés en fonction de besoins spécifiques).

La figure 4.14 illustre le processus complét de recherche.

Figure 4.14 – Processus de recherche véctorielle Elasticsearch

L'implementation de ce processus se produit dans la fonction suivante :

```
1  public async Task<ISearchResponse<AxamProduct>> Handle(GetProductsQuery request, CancellationToken cancellationToken)
2  {
3      await _keywordRepository.AddKeywordAsync(request.SearchQuery);
4      var encodeCommand = new EncodeSentenceCommand(request.SearchQuery);
5      var encodedQuery = await _sender.Send(encodeCommand, cancellationToken);
6
7      var response = await KnnSearchAsync<AxamProduct>(encodedQuery, request.KnnSearchRequest);
8
9      return response;
10 }
```

Figure 4.15 – Encodage de terme de recherche et recherche vectorielle dans Elasticsearch

La figure 4.16 illustre la recherche des produits à partir de Postman.

Figure 4.16 – Recherche à partir de Postman

La figure 4.17 illustre l'interface de recherche de client, ou il a cherché pour « lambaa taa tawla » qui se traduit en « lampe de table », avec des produits renvoyès ainsi que des suggestions des produits potentiellement similaires.

Figure 4.17 – Interface de recherche de client

4.6 Conclusion

Dans ce chapitre, nous avons couvert toutes les étapes nécessaires pour performer le recherche vectorielle dans notre projet, en commençant par la création de la fonction d'encodage « encode_sentence_and_normalise »à l'aide de notre modèle et de son tokeniser, ensuite la préparation de notre données pour ensuite insérez-les dans Elasticsearch pour avoir la capabilité de recherche vectorielle. Enfin nous préparons notre contrôleur ASP .NET Core pour atteindre l'endpoint « /encode » afin d'encoder le mot-clé de recherche saisi par l'utilisateur, et l'utiliser dans une query knn à l'aide de la méthode « KnnSearchAsync » pour performer une query Elasticsearch knn pour renvoyer des produits et des suggestions. Grâce à ces étapes, le travail pour le reste des sprints a été rendu beaucoup plus facile et évolutif.

Chapitre 5

Étude et réalisation du Sprint 2

Sommaire

5.1 Introduction 74

5.2 Backlog du Sprint 2 74

5.3 Spécification fonctionnelle 74

5.4 Diagramme de cas d'utilisation général 74

5.5 Description textuelle des cas d'utilisations 75

5.5.1 Description textuelle du CU « Rechercher produit en Arabe traditionnel » 76

5.5.2 Description textuelle du CU « Rechercher produit en Arabe en dialecte Tunisien » 77

5.5.3 Description textuelle du CU « Voir suggestions » 78

5.6 Conception 79

5.6.1 Diagramme de séquence détaillé 79

5.7 Réalisation 81

5.7.1 Les premières approches 81

5.7.2 Création de notre propre classe traducteur 83

5.8 Processus complèt de recherche vectorielle avec la traduction 94

5.8.1 Rèpartition dètaillèe de processus de recherche vectorielle complèt 94
5.8.2 Interface de recherche et affichage des suggestions 96
5.9 Conclusion 97

5.1 Introduction

Aprés avoir terminé le Sprint 1 du chapitre 4 qui tratait la recherche des produits dans la langue Française, nous entamons maintenant le Sprint 2 qui va aborder la recherche en Arabe en dialecte Tunisien, et ensuine l'Arabe Traditionnel.

5.2 Backlog du Sprint 2

ID	Scénario	Priorité	Complexité
1	En tant qu'un client, visiteur, je veux saisir ma terme de recherche en Arabe en dialecte Tunisien pour chercher le(s) produit(s)	1	7
2	En tant qu'un client, visiteur, je veux saisir ma terme de recherche en Arabe Traditionnel pour chercher le(s) produit(s)	2	6
3	En tant qu'un client, visiteur, je veux voir des produits similaires comme des suggestions.	3	6

Tableau 5.1 – Backlog du Sprint 2

5.3 Spécification fonctionnelle

Dans cette partie, on va expliquer les différentes fonctionnelités du Sprint 2 à travers le diagramme de cas d'utilisation. Puis en va exposer les différents scénarios de notre cas d'utilisation à travers des descriptions textuelles.

5.4 Diagramme de cas d'utilisation général

La figure 5.1 illustre le diagramme de cas d'utilisation gènèral de notre premier sprint, qui est la recherche en Français.

Figure 5.1 – Diagramme de cas d'utilisation général du Sprint 2

5.5 Description textuelle des cas d'utilisations

Une fois les divers cas d'utilisation présentés, nous examinerons de plus près certains d'entre eux en fournissant la description textuelle de certains d'entre eux.

5.5.1 Description textuelle du CU « Rechercher produit en Arabe traditionnel »

Titre : Rechercher produit en Arabe traditionnel

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

Acteur Principal : Client

Précondition :

1. Le client (ou le visiteur) sont sur la page de recherche
2. Le client (ou le visiteur) a saisi son terme de recherche en Arabe traditionnel
3. Le client (ou le visiteur) a cliqué sur "Rechercher"

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

Scénario de base :

1. Le client saisit son terme de recherche.
2. Le client clique sur la boutton "Rechercher"
3. Le système prend le terme de recherche, en vérifiant que c'est en Arabe traditionnel, et le traduit en Français.
4. Le systéme prend cette terme de recherche, et performe les étapes nécessaires pour la convertir en vecteur.
5. Le systéme compare cette vecteur contre les vecteurs dans Elasticsearch.
6. Le systéme renvoie les produits.

Scénario alternatifs :

1. Le terme de recherche est vide :
   (a) Le système affiche un message d'erreur informant le client que le terme de recherche est requis.
   (b) Retour à l'étape 1 du scénario de base.
2. Le terme de recherche n'est pas en Arabe traditionnel :
   (a) Le système suppose que le terme recherché est en Français.
   (b) Passer à la 4ème étape des scénarios de base.
3. Le(s) produit(s) que le client cherche n'existe pas.
   (a) Le systéme essaie de renvoyer les produits les plus similaires comme des suggestions.
   (b) Retour à l'étape 1 du scénario de base.

5.5.2 Description textuelle du CU « Rechercher produit en Arabe en dialecte Tunisien »

Titre : Rechercher produit en Arabe en dialecte Tunisien

Résumé : Le client saisit son terme de recherche en Arabe en dialecte Tunisien, en cliquant sur la boutton pour rechercher le(s) produit(s) qu'il veut chercher.

Acteur Principal : Client

Précondition :

1. Le client (ou le visiteur) sont sur la page de recherche
2. Le client (ou le visiteur) a saisi son terme de recherche en Arabe en dialecte Tunisien
3. Le client (ou le visiteur) a cliqué sur "Rechercher"

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

Scénario de base :

1. Le client saisit son terme de recherche en Arabe en dialecte Tunisien.
2. Le client clique sur la boutton "Rechercher"
3. Le système prend le terme de recherche, en vérifiant que c'est en Arabe en dialecte Tunisien, et le traduit en Français.
4. Le systéme prend cette terme de recherche, et performe les étapes nécessaires pour la convertir en vecteur.
5. Le systéme compare cette vecteur contre les vecteurs dans Elasticsearch.
6. Le systéme renvoie les produits.

Scénario alternatifs :

1. Le terme de recherche est vide :
   (a) Le système affiche un message d'erreur informant le client que le terme de recherche est requis.
   (b) Retour à l'étape 1 du scénario de base.
2. Le terme de recherche n'est pas en Arabe en dialecte Tunisien :
   (a) Le système suppose que le terme recherché est en Français.
   (b) Passer à la 4ème étape des scénarios de base.
3. Le(s) produit(s) que le client cherche n'existe pas.
   (a) Le systéme essaie de renvoyer les produits les plus similaires comme des suggestions.
   (b) Retour à l'étape 1 du scénario de base.

5.5.3 Description textuelle du CU « Voir suggestions »

Titre : Rechercher produit en Français

Résumé : Aprés que le client a cherché un produit, le systéme essaie de lui suggérer

des produits similaires.

Acteur Principal : Client

Précondition :

1. Le client est authentifié.
2. Le client à déja cherché un produit.

Postcondition : Le(s) produit(s) que le client cherche est renvoyé, et le systéme renvoie des produits similaires comme suggestions.

Scénario de base :

1. Le client cherche pour un produit.
2. Le systéme cherche le produit.
3. Le systéme renvoie les produits ainsi que des suggestions des produits similaires.

Scénario alternatifs :

1. Le(s) produit(s) que le client cherche n'existe pas.
   (a) Le systéme essaie de renvoyer les produits les plus similaires comme des suggestions.
   (b) Retour à l'étape 1 du scénario de base.

5.6 Conception

Dans cette partie, nous allons présenter le diagramme de séquence correspondant à notre diagramme cas d'utilisation précédemment présentés dans les descriptions textuelles pour le deuxième Sprint.

5.6.1 Diagramme de séquence détaillé

Nous avons regroupé les deux cas d'utilisation « Rechercher produits en Arabe traditionnel » et « Rechercher produits en Arabe en dialecte Tunisien » dans un seul diagramme de séquence présenté dans la figure 5.2.

Figure 5.2 – Diagramme de séquence des cas d'utilisations « Rechercher produits en Arabe Traditionnel » et « Rechercher produits en Arabe en dialecte Tunisien »

5.7 Réalisation

Cette partie est consacrée à la présentation des étapes nécessaires pour réaliser le travail nécessaire pour satisfaire notres cas d'utilisations, qui consiste à permettre le client ou le visiteur à rechercher notres produits en Arabe traditionnel et Arabe en dialecte tunisien tout en améliorant l'expérience de recherche en utilisant la traduction et la recherche vectorielle via Elasticsearch.

5.7.1 Les premières approches

Puisque notre modèle actuel n'est formé que sur les langues latines, nous avons eu l'idée de l'entraîner à la fois sur l'arabe tunisien et l'arabe traditionnel, mais un certain nombre de limitations nous empêchaient de le faire :

1. Absence totale de jeux de données sur la langue arabe en dialecte Tunisien.
2. Absence totale de jeux de données sur la langue Arabe Traditionnel.
3. Le processus de l'entraînement nécessite une machine beaucoup plus puissante et une période de temps très longue afin de traiter correctement les 2 langues que nous avons citées.

Nous avons également testé avec le « Fine-Tuning » pour entraîner notre modéle, nous définissons ce processus comme suit :
Le « Fine-Tuning » consiste à prendre un modèle d'apprentissage automatique pré-entraîné et à le former davantage sur un ensemble de données plus petit et ciblé. L'objectif du réglage fin est de conserver les capacités d'origine d'un modèle pré-entraîné tout en l'adaptant à des cas d'utilisation plus spécialisés.
What is fine-tuning in machine learning and AI ? ([5])

Mais ce processus nécessitait un ensemble de données beaucoup plus volumineux que celui que nous avions préparé et prenait trop de temps, nous avons donc décidé d'adopter l'approche mentionnée ci-dessous.

5.7.2 Création de notre propre classe traducteur

Avec les limitations de l'approche « Fine-Tuning » que nous avons mentionnée précédemment, nous avons décidé de créer notre propre traducteur qui va :

1. Vérifier si le terme de recherche est en Arabe traditionnel, et le traduire en Français à partir de l'API Google Traduction.
2. Vérifier si le terme de recherche est en Arabe en dialecte Tunisien à partir de notre propre dictionnaire des mots Tunisiens et leur equivalent en Français, si il n'y a pas d'équivalent exacte, il essaie de trouver l'équivalent en calculant un pourcentage, et s'il n'y a pas d'équivalent même après avoir calculé le pourcentage, le mot reste tel quel en supposant qu'il soit en Français

5.7.2.1 La création du dictionnaire

D'abord, nous commençons par préparer notre classe, que nous avons nommée « TunisianTranslator » et initialiser un attribut privé, qui est notre dictionnaire. La figure 5.3 montre le code nécessaire pour cette étape.

```python
class TunisianTranslator:
    __translation_dict = {
        "tawla": "table",
        "korsi": "chaise",
        "telifoun": "telephone",
        "mtaa": "de",
        "kartabla": "cartable",
        "9raya": "scolarite",
        "9aleya": "poêle",
        "mekina": "machine",
        "9ahwa": "cafe",
        "5obz": "pain",
        # ... reste des mots
```

Figure 5.3 – Code d'initialisation de classe et du dictionnaire

Ensuite, nous continuons en définissant notre première méthode dans la classe qui est une méthode privée nommée `__is_latin`. Cette méthode est conçue pour vérifier si une chaîne donnée s contient uniquement des caractères des blocs Unicode Basic Latin et Latin-1 Supplement, elle renvoie True si tous les caractères de la chaîne « s » se trouvent dans la plage Unicode U+0000 à U+00FF, ce qui correspond aux blocs Basic Latin et Latin-1 Supplement. Si un caractère se situe en dehors de cette plage, la méthode renvoie False.

5.7.2.2 Analyse des expressions régulières

L'expression régulière utilisée dans notre méthode est :

$$[\backslash u0000-\backslash u00FF]$$

- `[...]` : Spécifie une classe de caractères, correspondant à n'importe quel caractère unique inclus entre parenthèses.
- `^` : Entre parenthèses de classe de caractères, cela annule la classe, donc elle correspond à n'importe quel caractère *non* répertorié entre parenthèses.
- `\u0000-\u00FF` : Définit une plage de caractères Unicode de $U + 0000$ à $U + 00FF$, qui comprend à la fois les blocs Basic Latin et Latin-1 Supplement.

Le `Latin-1 Supplement` est défini comme suit :

Le bloc Latin-1 Supplement fait partie de la norme Unicode, couvrant la plage de $U + 0080$ *jusqu'à* $U + 00FF$*. Il complète le bloc Basic Latin (ASCII), qui couvre de* $U + 0000$ *jusqu'à* $U + 007F$*. Ce bloc comprend des caractères supplémentaires utilisés dans diverses langues occidentales, comprenant des lettres accentuées, des signes de ponctuation, des symboles monétaires et d'autres symboles typographiques, représentant la moitié supérieure du codage de caractères ISO/IEC 8859-1.*

Latin-1 Supplement ([6])

5.7.2.3 Logique de méthode

La méthode `re.search(r"[\u0000-\u00FF]", s)` recherche dans la chaîne s, recherchant tout caractère en dehors de plage $U+0000$ jusqu'à $U+00FF$:

- Si un tel caractère est trouvé, `re.search` renvoie un objet de correspondance, qui est évalué à `True`.
- Si aucun caractère de ce type n'est trouvé (c'est-à-dire que tous les caractères sont dans la plage spécifiée), il renvoie `None`, qui est évalué à `False`.

L'utilisation de l'opérateur `not` inverse le résultat de `re.search`. Ainsi, la méthode renvoie `False` si des caractères se trouvent en dehors de la plage Unicode spécifiée, et `True` si tous les caractères s'y trouvent.

La figure 5.4 illustre le code nécessaire pour cette méthode.

```python
def __is_latin(self, s):
    # Cette expression régulière correspond à tout caractère qui ne fait pas partie de la plage de base Latin ou Latin-1 Supplement.
    return not re.search(r"[^\u0000-\u00FF]", s)
```

Figure 5.4 – Code de méthode ___is_latin

Cette méthode étant terminée, nous passons maintenant à notre deuxième méthode, qui est une méthode privée appelée `__find_best_match` pour vérifier le mot correspondant le plus proche dans notre dictionnaire Tunisien. Pour avoir cette fonctionnalité, on va utiliser la bibliothéque `difflib` du langage de programmation Python, qui fournit des aides pour calculer les différences entre les séquences, y compris des outils pour trouver des séquences correspondant étroitement à une séquence d'entrée donnée.

5.7.2.4 Répartition des fonctionnalités

Pour trouver cette mot correspondant, on va utiliser la méthode `get_closest_matches` qui prends 4 paramétres :

1. `tunisian_text:` La chaîne de caractères pour laquelle des correspondances proches sont recherchées.
2. `self.__translation_dict.keys():` La séquence à laquelle le tunisian_text est comparé. Ici, ce sont les clés du notre dictionnaire Tunisien ___translation_dict.
3. `n=1:` Ce paramètre spécifie le nombre de correspondances proches à renvoyer. Le réglage n=1 signifie que la fonction renverra au plus une

correspondance la plus proche.

4. `cutoff=0.7:` C'est le seuil de similarité. Les correspondances avec un score de similarité inférieur à 0,7 (où 1,0 correspond à une correspondance exacte et 0 à une absence de similarité) ne sont pas incluses. Une valeur de 0,7 implique que la fonction prendra en compte les correspondances qui sont similaires à au moins 70 à tunisian_text.

5.7.2.5 Renvoie du rèsultat

La fonction sauveguarde les résultats de `get_close_matches` dans la variable `matches`, elle vérifie ensuite si des correspondances ont été trouvées. Si `matches` contient des éléments, `matches[0]` (la meilleure correspondance due à n=1) est renvoyé. Si `matches` est vide, ce qui indique qu'aucune correspondance suffisamment proche n'a été trouvée, la méthode renvoie None. La figure 5.5 illustre le code nècessaire pour cette mèhode.

```
from difflib import get_close_matches

def __find_best_match(self, tunisian_text):
        matches = get_close_matches(
            tunisian_text, self.__translation_dict.keys(), n=1, cutoff=0.7
        )
        return matches[0] if matches else None
```

Figure 5.5 – Code de méthode ___find_best_matches

Une fois les 2 méthodes précédentes sont terminées, nous pouvons maintenant passer à notre méthode la plus importante, qui utilisera ces méthodes pour gérer la logique de traduction et renverra le terme de recherche traduit. La méthode est appelé `translate`, et sa visibilité est publique.

5.7.2.6 Aperçu de la méthode

But : Traduire du texte du tunisien ou ou du texte en Arabe traditionnel (non latin) vers le Français,, en utilisant à la fois des services de traduction automatisée et un dictionnaire de traduction prédéfini.

Input : `text:` Une chaîne de caractères contenant le texte à traduire.

Output : Le texte traduit en Français, ou un message d'erreur si la traduction échoue.

5.7.2.7 Répartition détaillée

1. Initialisation des variables :
   - `mots = []` : Initialise une liste vide qui contiendra plus tard des mots individuels du texte saisi.
2. Vérifiez les caractères non latins :
   - On utilise la méthode `__is_latin` pour vérifier si le texte est en latin ou non. Si cette méthode renvoie `False`, nous utilisons Google Traduction pour traduire ce texte en Français, sinon, nous vérifions si le texte est en Tunisien en vérifiant si il y'a une correspondance dans notre dictionnaire.
3. Utilisation de Google Traduction
   - Création de l'instance de l'objet `Translator` importé de la bibliothéque `googletrans`.
   - La méthode essaie de traduire le texte en Français (dest="fr"). En cas de succès, le texte du résultat traduit remplace le texte original.
4. Traduction en utilisation de notre dictionnaire
   - Nous initialisons une liste vide pour stocker les mots traduits (translated_words = []).
   - Ensuite, nous performons une itération sur les mots de terme de recherche.

- Recherche de la meilleure correspondance : pour chaque mot, il appelle self.__find_best_match(word) pour trouver la clé correspondante la plus proche dans le dictionnaire de traduction __translation_dict.
- Si une correspondance est trouvée (la clé n'est pas None), la méthode ajoute la valeur correspondante de __translation_dict à translated_words.
- Si aucune correspondance n'est trouvée, le mot original est ajouté à translated_words car elle suppose que le mot n'a pas besoin de traduction ou qu'aucune traduction appropriée n'existe dans le dictionnaire, supposons qu'elle est en Français.

5. Construction de la phrase traduite finale en joignant la liste translated_words en une seule chaîne avec des mots séparés par des espaces (" ".join(translated_words)) et renvoie cette chaîne comme phrase traduite finale.

La figure 5.6 illustre le code nècessaire pour créer cette méthode.

```python
def translate(self, text):
    words = []
    # Vérifiez si le texte n'est pas latin et essayez de le traduire si nécessaire
    if not self.__is_latin(text):
        try:
            translator = Translator()
            text = translator.translate(text, dest="fr").text
        except Exception as e:
            print(f"Translation failed: {e}")
            return "Translation error"

    words = text.split()

    translated_words = []

    for word in words:
        key = self.__find_best_match(word)
        if key:
            translated_words.append(self.__translation_dict[key])
        else:
            translated_words.append(word)

    return " ".join(translated_words)
```

Figure 5.6 – Code de méthode translate

Enfin, nous doivent exposer une endpoint Flask `"/encode"` comme une endpoint de microservice publique pour regrouper toutes le fonctionnalités qui sont les suivants :

1. Extraire le terme de recherche de la requête.
2. Essayez de le traduire en Français si nécessaire.
3. Utiliser la méthode encode_sentence_and_normalise de notre modéle Sentence-Transformers pour faire l'encodage du terme de recherche en

vecteur.

4. Renvoyer ce vecteur au contrôleur ASP .NET Core sous forme JSON.

La figure 5.7 illustre le code complèt du endpoint `"/encode"`.

```python
from flask import Flask, request, jsonify
from flask_cors import cross_origin
from model import model
from translator import translator

app = Flask(__name__)


@app.route("/encode", methods=["POST"])
@cross_origin()
def encode():
    data = request.json
    sentence = data.get("sentence")
    translated_sentence = translator.translate(sentence)
    encoded_sentence = model.encode_sentence_and_normalise(translated_sentence)
    return jsonify({"query_vector": encoded_sentence})
```

Figure 5.7 – Code de endpoint "/encode"

5.8 Processus complèt de recherche vectorielle avec la traduction

La figure 5.8 illustre le processus complèt de recherche vectorielle avec la traduction.

Figure 5.8 – Processus complèt de recherche vectorielle avec la traduction

5.8.1 Rèpartition dètaillèe de processus de recherche vectorielle complèt

1. `Intèraction du client:` Le client saisit son terme de recherche à travers l'interface de recherche.
2. `Contrôleur ASP.NET :` Contrôleur ASP.NET : le terme de recherche est envoyé du client à un contrôleur backend implémenté dans ASP.NET,

qui transmet ensuite le terme de recherche à une API Flask.

3. **`Analyse des mots clés:`** l'API Flask détermine si le terme de recherche est en latin. Si tel est le cas, le processus passe directement à l'encodage du terme en utilisant notre modéle Sentence-Transformers pour qu'il soit adapté à la recherche vectorielle dans Elasticsearch. Si le terme n'est pas en latin, il vérifie s'il est en arabe traditionnel ou en dialecte tunisien, sinon, il suppose qu'il est en Français.
4. **`Traduction selon la langue détectée:`** Les termes arabes traditionnels sont traduits à l'aide des services de traduction Google. Les termes du dialecte tunisien sont traduits à l'aide de notre dictionnaire tunisien.
5. **`Encodage`** : Les termes traduits sont éventuellement codés à l'aide de notre modéle Sentence-Transformers.
6. **`Recherche vectorielle dans Elasticsearch:`** Les vecteurs codés sont utilisés pour interroger Elasticsearch, qui recherche des vecteurs (documents) similaires stockés dans son index, qui sont notre produits dans notre cas.
7. **`Récupération des résultats:`** Les résultats de la recherche sont récupérés et renvoyés via le contrôleur ASP.NET au client, complétant ainsi le processus de recherche.

5.8.2 Interface de recherche et affichage des suggestions

Cette section est consacrée pour la réalisation de l'interface de recherche pour le client et le visiteur ainsi que l'affichage des suggestions des produits similaires a son terme de recherche.

La figure 5.9 illustre l'interface de recherche, dans ce cas un visiteur a cherché pour parfum en Arabe traditionnel, et les résultats sont renvoyés, ainsi que des suggestions au bas de la page qui sont dans la figure 5.10.

Figure 5.9 – Interface de recherche de client et visiteur

Figure 5.10 – Affichage des suggestions des noms des produits similaires

5.9 Conclusion

Dans ce chapitre, nous avons couvert toutes les étapes nécéssaires pour permettre le client à rechercher notres produits en Arabe traditionnel et en dialecte Tunisien a travers le processus de traduction et traitement du mot clé qu'il cherche en utilisant les divers techniques, fonctions et bibliothéque mentionné tout au long de notre chapitre.

Chapitre 6

Étude et réalisation du Sprint 3

Sommaire

6.1 Introduction 99
6.2 Backlog du Sprint 3 99
6.3 Spéficiation fonctionnelle 99
6.3.1 Diagramme de cas d'utilisation du CU général 100
6.3.2 Description textuelle des cas d'utilisations 100
6.4 Conception 105
6.4.1 Diagrammes de séquence détaillé 105
6.5 Tableau de bord 108
6.5.1 Les indicateurs de performance 109
6.6 Réalisation 110
6.7 Conclusion 113

6.1 Introduction

Aprés avoir terminé le Sprint 2 du chapitre 5 qui tratait la recherche des produits dans la langue Arabe traditionnel et en dialecte Tunisien, nous entamons maintenant le Sprint 2 qui va aborder le dashboard de l'admin qui lui permettra de modifier ces paramétres de recherche.

6.2 Backlog du Sprint 3

ID	Scénario	Priorité	Complexité
1	En tant qu'un admin, je veux modifier mes paramétres de recherche, pour chercher le(s) produit(s)	2	7
2	En tant qu'un admin, je veux consulter le tableau de bord des produits avec le meilleur score de similarité.	1	10

Tableau 6.1 – Backlog du Sprint 2

6.3 Spéficiation fonctionnelle

Au cours de cette partie, nous mettrons en avant les différentes fonctionnalités du Sprint 3 à travers le diagramme de cas d'utilisation. Par la suite, nous détaillerons quelques scénarios de cas d'utilisation grâce à des descriptions textuelles.

6.3.1 Diagramme de cas d'utilisation du CU général

Figure 6.1 – Diagramme de cas d'utilisation général de Sprint 3

6.3.2 Description textuelle des cas d'utilisations

Suite à l'exposition des cas d'utilisations, nous approfondirons leurs descriptions textuelle.

6.3.2.1 Description textuelle du CU "Modifier paramétres de recherche"

Titre : Modifier paramétres de recherche

Résumé : L'admin saisit son terme de recherche en Arabe traditionnel, dialecte Tunisien ou en Français, aussi que modifier un ou plusieurs paramétres de recherche, en cliquant sur la boutton pour rechercher le(s) produit(s) qu'il

veut chercher.

Acteur Principal : Admin

Précondition :

1. L'admin est sur la page de recherche de l'admin, et il est authentifié.
2. L'admin a saisi son terme de recherche en l'un des trois langages mentionnées.
3. L'admin a modifié l'un des quatres paramétres de recherche (TopResProdLabel, TopResDesc, NumCandidatesProdLabel, NumCandidatesDesc).
4. L'admin a cliqué sur "Rechercher"

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

Scénario de base :

1. L'admin saisit son terme de recherche.
2. L'admin modifie l'un des quatres paramétres de recherche (TopResProdLabel, TopResDesc, NumCandidatesProdLabel, NumCandidatesDesc).
3. L'admin clique sur la boutton "Rechercher"
4. Le système prend le terme de recherche, en vérifiant que c'est valide.
5. Le systéme prend cette terme de recherche, et performe les étapes nécessaires pour la convertir en vecteur.
6. Le systéme compare cette vecteur contre les vecteurs dans Elasticsearch en utilisant les paramétres de recherche que l'admin a saisi.
7. Le systéme renvoie les produits.

Scénario alternatifs :

1. Le terme de recherche est vide :
   (a) Le système affiche un message d'erreur informant le client que le terme de recherche est requis.
   (b) Retour à l'étape 1 du scénario de base.
2. Le terme de recherche n'est pas valide :
   (a) Le système suppose que le terme recherché est en Français.
   (b) Passer à la 6ème étape des scénarios de base.
3. L'admin n'as pas modifié l'un des quatres paramétres de recherche :
   (a) Le système utilise les valeurs par dèfauts pour la recherche :
      - NumCandidatesProdLabel : 25
      - NumCandidatesDesc : 25
      - TopResProdLabel : 10
      - TopResDesc : 20
   (b) Passer à la 6ème étape des scénarios de base.
4. Le(s) produit(s) que l'admin cherche n'existe pas.
   (a) Le systéme essaie de renvoyer les produits les plus similaires comme des suggestions.
   (b) Retour à l'étape 1 du scénario de base.

6.3.2.2 Description textuelle du CU "Consulter tableau de bord"

Titre : Consulter tableau de bord des produits

Résumé : L'admin consulte le tableau de bord pour visualiser les produits renvoyés avec le meilleur score de similarité.

Acteur Principal : Admin

Précondition :

1. L'admin est sur la page de recherche de l'admin, et il est authentifié.
2. L'admin a saisi son terme de recherche en l'un des trois langages mentionnées.
3. L'admin a modifié l'un des quatres paramétres de recherche (TopResProdLabel, TopResDesc, NumCandidatesProdLabel, NumCandidatesDesc).
4. L'admin a cliqué sur "Rechercher"

Postcondition : Le(s) produit(s) que l'admin cherche est renvoyé, avec leurs scores et visualisation.

Scénario de base :

1. L'admin accède à son tableau de bord
2. L'admin saisit son terme de recherche.
3. L'admin modifie l'un des quatres paramétres de recherche (TopResProdLabel, TopResDesc, NumCandidatesProdLabel, NumCandidatesDesc).
4. L'admin clique sur la boutton "Rechercher"
5. Le système prend le terme de recherche, en vérifiant que c'est valide.

6. Le systéme prend cette terme de recherche, et performe les étapes nécessaires pour la convertir en vecteur.
7. Le systéme compare cette vecteur contre les vecteurs dans Elasticsearch en utilisant les paramétres de recherche que l'admin a saisi.
8. Le systéme renvoie les produits avec les scores et les visualise.

6.4 Conception

Au cours de cette section, nous examinerons les diagrammes de séquence en relation avec les descriptions textuelles des cas d'utilisation précédemment exposés correspondant au troisième sprint.

6.4.1 Diagrammes de séquence détaillé

Dans cette partie, nous aborderons les diagrammes de séquence pour les cas d'utilisation suivants :

- Modifier paramétres de recherche
- Consulter tableau de bord
- S'authentifier

6.4.1.2 Diagramme de séquence "Modifier paramétres de recherche"

Figure 6.3 – Diagramme de séquence de CU "Modifier paramétres de recherche"

6.4.1.1 Diagramme de séquence "Consulter tableau de bord"

Figure 6.2 – Diagramme de séquence de CU "Consulter tableau de bord"

6.4.1.3 Diagramme de séquence "S'authentifier"

Figure 6.4 – Diagramme de séquence de CU "S'authentifier"

6.5 Tableau de bord

Le tableau de bord (ou dashboard en anglais) est un outil qui permet à plusieurs collaborateurs de suivre l'avancée d'un projet. Il présente de manière synthétique tous les éléments et indicateurs clés qui permettent à la fois :

- D'organiser le jalonnage, lister les tâches et de les assigner aux différentes parties prenantes au projet.
- De prendre des décisions et de mettre des correctifs en place quand des alertes sont remontées.

6.5.1 Les indicateurs de performance

Un indicateur de performance est un mesure ou un ensemble de mesures d'un aspect spécifique et essentiel de la performance globale de l'organisation. Dans cette section, nous allons présenter les différents indicateurs que nous avons choisis pour le tableau de bord de l'admin. Nous remplissons le tableau 6.2 avec :

- **Indicateur de performance (KPI) :** Le nom de l'indicateur
- **Objectif :** l'importance ou la signification de l'indicateur
- **Visualisation :** la manière dont l'indicateur sera affiché et le type de graphique approprié pour cet indicateur.
- **Spécifications :** Les détails descriptifs concernant chaque KPI.

KPI	Objectif	Visualisation	Spécification
Score de similarité de recherche	Mesurer le score de similarité entre le terme de recherche et les produits	Diagramme à Bandes	Similarité entre le terme de recherche et les produits
Les mots les plus recherché	Mesure les mots les plus recherché dans le site	Diagramme Circulaire	Nombre de mots (en pourcentage) recherché par rapport au total des recherches

Tableau 6.2 – Tableau des KPIs

6.6 Réalisation

Dans cette section nous présentons le tableau de bord pour notre administrateur développé durant le Sprint 3, ainsi que l'interface de modification des paramétres de recherche des produits, et bien sûr l'interface d'authentification.

La figure 6.5 illustre le formulaire d'authentification pour l'administrateur.

Figure 6.5 – Interface de pour l'authentification de l'admin

La figure 6.6 illustre le formulaire de modification de paramétres de recherche pour l'administrateur.

Figure 6.6 – Interface de modification de paramétres de recherche pour l'admin

La figure 6.7 illustre le tableau de bord de l'administrateur aprés qu'il a cherché pour « parfum », à gauche, sont les produits avec leurs scores (similarités) a la terme de recherche de l'admin, et à droite, les mots les plus

cherché sur notre site.

Figure 6.7 – Interface de Tableau de bord de l'administrateur

6.7 Conclusion

Au terme du troisième sprint, nous avons réussi à intégrer un tableau de bord pour l'administrateur dans notre application web. Cela permet à l'administrateur de mesurer et voir quelles termes de recherche donne les meilleurs résultats, ainsi que les termes de recherche les plus recherché sur notre site. Ainsi que modifier les paramétres de recherche des produits pour potntiellement améliorer la précision des résultats de recherche. Notre application est maintenant enrichi des fonctionnalités Business Intelligence.

Conclusion et perspectives

Ce rapport expose le travail accompli au cours de ce PFE au sein de la société Axam. Notre mission consistait à utiliser le recherche vectorielle à l'aide de Elasticsearch ainsi que un modèle Sentence-Transformers pré-entraîné, pour l'encodage des termes des recherches et la compréhension de la langage Arabe traditionnel et le dialect Tunisien. Toutes ces technologies nous ont permis d'améliorer considérablement la précision de nos résultats de recherche ainsi que la rapidité de renvoie des résultats, ainsi que la recherche dans 3 langages.

Tout au long du développement de ce projet, nos compétences techniques ont été renforcé grâce à l'utilisation des différents frameworks et langages de programmation tel que ASP .NET Core avec C#, et Flask, Pytorch avec Python pour le développement backend et l'utilisation du modéle Sentence-Transformers, ainsi que des technologies DevOps comme Docker pour l'utilisation d'Elasticsearch, qui est une nouvelle technologie adapté, et React avec Typescript pour le développement d'un interface simple pour l'administrateur et client, ainsi que le tableau de bord.

Malgré les défis rencontrés, tels que l'utilisation des nouvelles technologies que nous avons j'amais utilisés, nous avons dépassé nos limites et avons réussi à tout apprendre assez rapidement pour développer le projet que nous venons de terminer.

Bibliographie

[1] Pinecone. *All-Mpnet-Base-V2*. 2024. URL : https://www.pinecone.io/models/ll-mpnet-base-v2/.

[2] Elastic. *ECE Hardware Prereq*. 2024. URL : https://www.elastic.co/guide/en/cloud-enterprise/current/ece-hardware-prereq.html.

[3] Lucidcharts. *Diagramme de classes*. 2024. URL : https://www.lucidchart.com/pages/fr/diagramme-de-classes-uml.

[4] Pytorch. *no_grad*. 2023. URL : https://pytorch.org/docs/stable/generated/torch.no_grad.html.

[5] TechTarget. *What is fine-tuning in machine learning and AI ?* 2024. URL : https://www.techtarget.com/searchenterpriseai/definition/fine-tuning.

[6] Symbl. *Latin-1 Supplement*. 2024. URL : https://symbl.cc/en/unicode/blocks/latin-1-supplement/.

[7] Microsoft. *Vecteurs dans Recherche*. 2024. URL : https://learn.microsoft.com/fr-fr/azure/search/vector-search-overview.

[8] Edin Šahbaz. *Getting Started with Clean Architecture in .NET Core*. 2023. URL : https://medium.com/@edin.sahbaz/getting-started-with-clean-architecture-in-net-core-fa9151bc5918.

[9] Microsoft. *Visual Studio*. 2024. URL : https://visualstudio.microsoft.com/.

[10] Microsoft. *Visual Studio Code*. 2024. URL : https://code.visualstudio.com/.

[11] Meta. *React*. 2024. URL : https://react.dev/.

[12] Elastic. *Elasticsearch*. 2024. URL : https://www.elastic.co/.

[13] Postman. *What is Postman ?* 2024. URL : https://www.postman.com/product/what-is-postman/.

[14] Docker. *Docker Overview*. 2024. URL : https://docs.docker.com/get-started/overview/.

[15] Microsoft. *A tour of the C# language*. 2024. URL : https://learn.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/.

[16] Microsoft. *What is ASP .NET Core ?* 2024. URL : https://dotnet.microsoft.com/en-us/learn/aspnet/what-is-aspnet-core.

[17] Openclassrooms. *Adoptez une architecture MVC*. 2022. URL : https://openclassrooms.com/fr/courses/4670706-adoptez-une-architecture-mvc-en-php.

[18] Elastic. *knn-search*. 2024. URL : https://www.elastic.co/guide/en/elasticsearch/reference/current/knn-search.html.

Amelioration de recherche et intégration de Recherche Intelligent

Borgi Aaron et Ben Hassine Ghaithallah

Resumé

Ce projet a été réalisé au sein de la société Axam dans le cadre d'un projet de fin d'études pour l'obtention du diplôme de licence en informatique de gestion, spécialité "Business Intelligence", à l'Institut Supérieur de Gestion de Sousse (ISGS) pour l'année universitaire 2023/2024.

L'objectif principal du projet consiste à utiliser une nouvelle approche de recherche dans les stites E-Commerce afin d'optimiser à la fois la vitesse et la précision de la recherche dans plusieurs langages tels que le Français, Arabe traditionnel, et Arabe en dialect Tunisien.

Ce rapport présente les différentes étapes de la conception et du développement de cette application, mettant en avant les méthodologies et technologies employées pour atteindre les objectifs fixés.

Mots clés— Recherche Vectorielle, Elasticsearch, Sentence-Transformers, Fine-Tuning, Similarité Cosinus, MVC, ASP .NET Core, Agile, SCRUM