PIXIMIND
Web & Mobile Solutions

Rapport de stage

développement d'une solution de prédiction de l'activité d'une entreprise

Réalisé par :

Emna Mkaouar

Chedy Chaaben

Table of contents

List of Figures

Introduction générale

Quel que soit le domaine, et peu importe l'activité, l'intelligence artificielle fait incontestablement avancer notre quotidien. Elle fait référence à des machines et des systèmes qui sont inspirés par l'intelligence humaine dans le but d'automatiser les tâches les plus simples pour nous faciliter la vie, améliorer notre confort et optimiser le temps et l'argent.

En effet, dans les entreprises, les responsables de partenariat perdent beaucoup de temps pour trouver des collaborateurs qui partagent les mêmes centres d'intérêt. Ainsi, dans le domaine informatique, les agences de développement cherchent des partenaires selon les technologies et les services, et cette tâche peut perdre du temps.

C'est le cadre choisi pour notre projet de stage d'été. L'objectif de ce sujet est de développer une application pour la prédiction de l'activité d'une entreprise à travers le contenu de son site web en automatisant le processus de la recherche des partenaires pour l'entreprise Piximind.

Notre application permet aux responsables de partenariat de chercher des agences de développements qui partagent les mêmes technologies de Piximind en se basant sur des liens. Cette plateforme permet de classifier les agences de développements et donner comme résultats ses contacts (emails, télé...).

Pour la présentation de ce projet, nous avons divisé notre projet en deux chapitres : Dans le premier chapitre, on va identifier les besoins fonctionnelles et non fonctionnelles de notre projet puis présenter la conception en définissant les architectures utilisées.
Le deuxiéme chapitre est dédié pour l'étude concrète du projet. Ce chapitre expose le travail réalisé sur la création des interfaces.

Chapitre 1

Spécification des besoins et conception

Pour faire émerger les concepts qui permettront d'apporter des solutions aux différents problèmes, il faut identifier les divers acteurs de notre système et les fonctionnalités de l'application qui seront détaillées dans les besoins fonctionnels et non fonctionnels de notre projet , puis présenter les différents architectures utilisées.

1.1 Spécifications des besoins

1.1.1 Acteurs

Les acteurs sont définis en tant qu'entités externes ayant une interaction avec le système.
Nous pouvons distinguer dans notre application un seul acteurs :

— **le responsable de partenariats** : c'est la personne qui est chargée d'analyser le marché lié aux services proposés par son entreprise , à rechercher et identifier des partenaires.

1.1.2 Les besoins fonctionnels

Les exigences fonctionnelles de notre projet représentent ce qu'attend chaque intervenant de l'application pour satisfaire les besoins du l'utilisateur . Ces besoins devront être persistants, précis et faisables.
Notre application est basée sur les fonctionnalités suivantes :

— Prédire l'activité d'une entreprise en se basant sur son site : le responsable de partenariat donne un lien d'une entreprise par suite notre application affichera comme résultat si cette entreprise est une agence de développement informatique ou non .

— Permettre de savoir le centre d'intérêt de l'entreprise : l'application permettre de comparer les technologies de l'agence de développement trouvés dans son site avec celles de Piximind.

— Présenter comme résultat les contacts : notre application affichera les contacts de l'entreprise donnée comme les numéros de téléphone , les emails ou les liens de contact .

1.1.3 Les besoins non fonctionnels

Les exigences non fonctionnelles correspondent aux caractéristiques du système, à la fiabilité, aux dépendances de la plateforme et aux contraintes d'environnement et de mise en œuvre.

— **Performance** : L'application doit être fiable et toujours opérationnelle.
— **Flexibilité** : L'application doit être configurable et flexible.
— **Simplicité** : L'application doit avoir une interface compréhensible et facile à prendre en main.
— **Rapidité** : L'application doit assurer la rapidité en tant que temps de réponse.

1.1.4 Diagramme de cas d'utilisation

La figure suivante représente le diagramme de cas d'utilisation global qui donne une vue générale de l'acteur de l'application ainsi que des différentes fonctionnalités requises

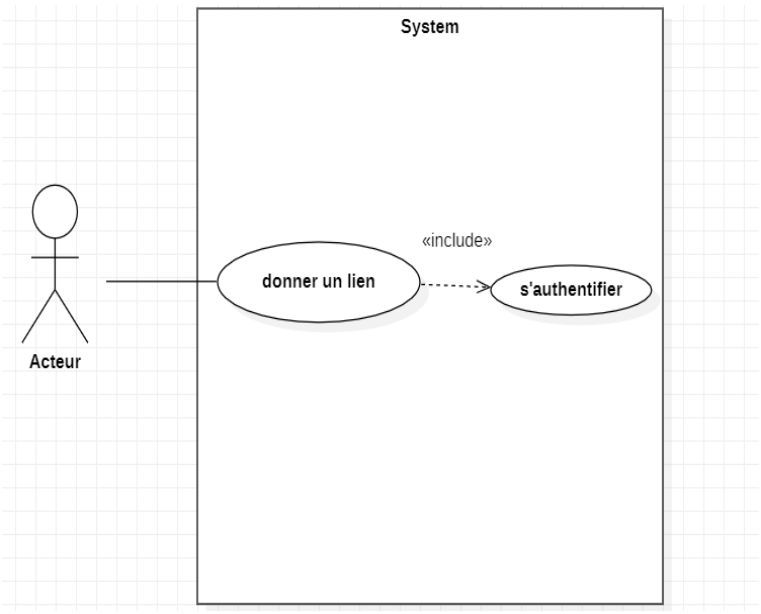

FIGURE 1.1 – Diagramme de cas d'utilisation

1.2 Conception

Pour donner suite à la spécification des besoins auxquelles notre application doit répondre dans la partie précédente, nous consacrons cette partie à la conception. Tout d'abord, on va présenter l'architecture générale de notre application, puis nous détaillons l'architecture logique.

1.2.1 Conception globale

Dans cette partie, on va identifier l'architecture physique et logique qui sont les plus adaptées à notre application.

1.2.1.1 Architecture Physique

Dans notre projet, nous avons choisi d'utiliser l'architecture 4-tiers comme architecture physique puisqu'elle présente les quatre niveaux suivants :

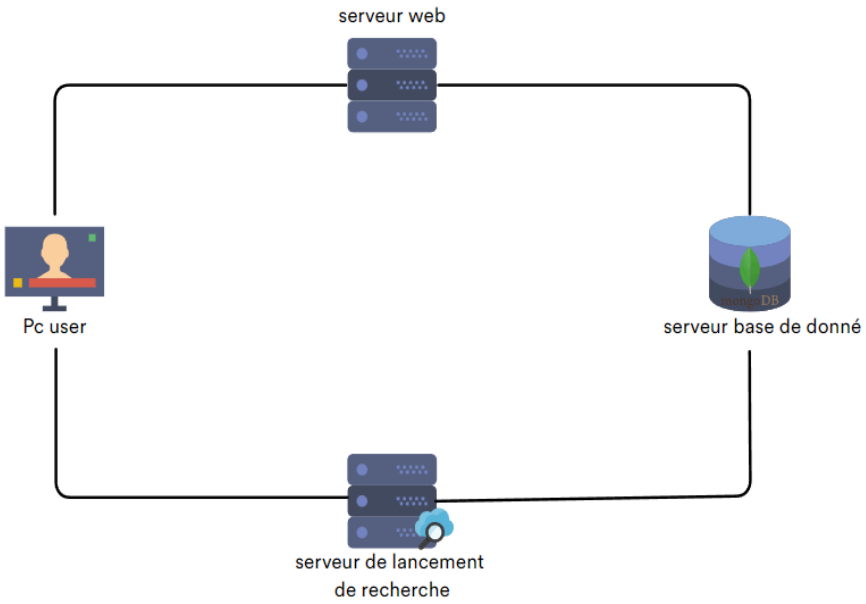

FIGURE 1.2 – Diagramme de déploiement

— Serveur de recherche : c'est le serveur qui permet à l'utilisateur d'effectuer la recherche.
— Serveur web : Il s'agit du moyen qui garantit l'interaction entre les différents composants.
— Serveur base de données : Un serveur de base de données sert au stockage, à la récupération et à la gestion des données dans une base de données.
— Client léger :Il s'agit du moyen par lequel le client accéder à l'application via un navigateur Web.

1.2.1.2 Architecture logique

L'architecture logique, fait partie des étapes de la conception globale. Elle décrit les différentes composantes de notre système et les relations entre eux.

L'architecture logique utilisée dans notre application est l'architecture multi-modèle. Nous avons modélisé notre projet en trois packages où chaque package possède sa propre architecture :

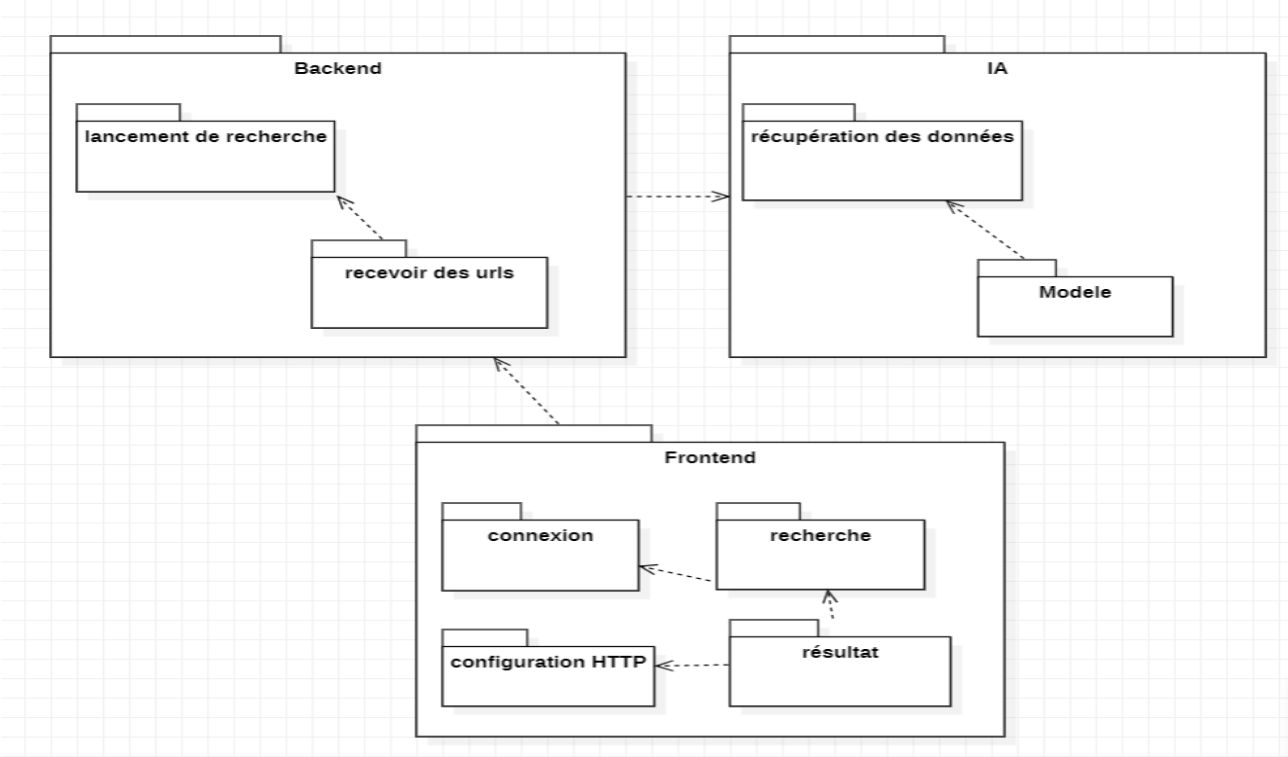

FIGURE 1.3 – Diagramme de paquet

— **Paquet « IA »** : ce package contient la récupération des informations ainsi les modèles pour la classification.
Nous avons choisi l'architecture pipeline puisque chaque composante a un input et un output au composante suivante. Au début, nous récupérons les données d'un site de l'entreprise en utilisant un algorithme de web scraping, ensuite les entrons dans un modèle d'extraction des keywords. Ces keywords seront traités selon un modèle, qui recherche la similarité entre ces derniers et les mots clé qui définissent une agence de développement.

FIGURE 1.4 – Architecture Pipeline

— **Paquet « backend »** : ce package comprend les détails du composant fonctionnel de l'application.
Nous avons choisi l'architecture REST puisque nous avons développé la partie backend par Fastapi. Cette architecture est basé sur trois composants : Le Serveur REST , le composant API RESTFUL et le composant application client.
En effet, l'implémentation de REST est extrêmement facile puisqu'elle repose principalement sur le protocole HTTP et fournit des réponses dans le format JSON. Les API RESTful supportent les méthodes « http » les plus courantes (POST, GET, DELETE et PUT).

— **Paquet « frontend »** : ce package est développer par React.JS. Nous avons choisi l'architecture en couche puisque nous avons décomposer notre application en 4 couches :
La couche « connexion » pour l'authentification du l'utilisateur, la couche « recherche » pour donner le lien et lancer la recherche, la couche « configuration » pour la traduction des requête http et finalement la couche « résultat » pour afficher les résultats.

1.2.2 Conception détaillée

1.2.2.1 Diagramme de séquence

On utilise le diagramme de séquence pour décrire la coopération entre les objets le long de l'axe chronologique. Dans le diagramme de séquence, l'accent est mis sur l'ordre dans lequel les messages sont envoyés. Le temps avance du haut vers le bas dans le diagramme.

Diagramme de séquence de lancement de recherche :

Cette information alors sera envoyée au serveur FastApi, puis ce serveur pourra accéder a ce lien pour récupérer le contenu de ce site à travers le web Scraping . Ces données seront traitées par un Modèle de Keywrods puis le Modèle Bert qui donnera comme résultat un score.
Selon ce score le serveur FastApi sera traité la résultat :

— si l'entreprise est agence de développement le serveur terminera le web Scraping des contacts et des technologies qui seront stockés dans notre base donnée et affichera « c'est une agence de développement »,

— sinon il sera affiché comme résultat « c'est pas une agence de développement » sans stocker les données.

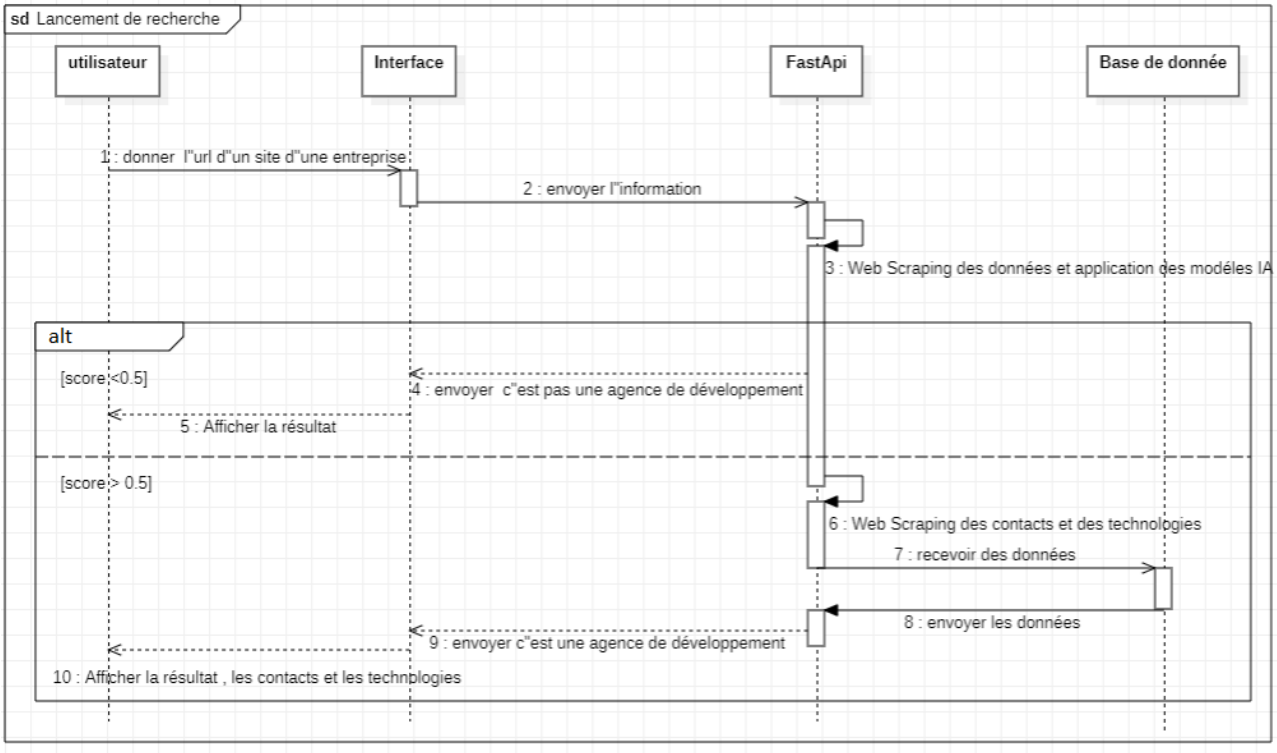

FIGURE 1.5 – Diagramme de séquence

Conclusion

Dans ce chapitre, on a spécifié notre acteur du système et on a identifié son rôle, puis on a présenté les besoins fonctionnelles et non fonctionnelles du système ainsi que les diagrammes nécessaires auxquels nous pouvons concevoir et développer le système. Nous avons ensuite effectué la conception qui sera suivie ultérieurement pour le développement de notre application. On passe maintenant à la phase de réalisation.

Chapitre 2

Réalisation

Après avoir défini la conception du projet, nous présentons ici la partie finale de notre rapport, qui vise à montrer les interfaces conçues et développées. Nous présenterons d'abord l'environnement matériel et logiciel avec lesquels on a travaillé durant la phase de programmation, puis les différentes interfaces et fonctionnalités réalisées seront présentées et aussi les tests effectués.

2.1 Environnement de travail

2.1.1 Environnement matériel

Au cours de la mise en œuvre du projet, les matériaux suivants ont été utilisés :

Le développement était réalisé à l'aide d'un PC possédant les caractéristiques décrites ci-dessous :

- Système d'exploitation : Windows 10
- Mémoire (RAM) : 24 Go
- Processeur : Intel Core i5-8250

2.1.2 Environnement logiciel

Dans cette section on va présenter les différents outils logiciels qu'on a utilisées dans la réalisation de notre projet :

- **Jupyter Notebook :**
  Jupyter Notebook est une outil open source qui permettre d'écrire des lignes de code informatique et le partager en collaboration. On a l'utilisé dans la partie Intelligence artificielle afin de développer le code écrit en python.

- **Visual studio code :**
  VSC est un éditeur open source développer par Microsoft. Il est utilisé avec des

différents langages de programmation.

— **MongoDB :**
MongoDB est un système de base de donnée NoSQL. Il peut offrir de nombreux avantages à une équipe des développeurs puisque son schéma flexible facilite l'évolution et le stockage des données.

— **MongoDB Compass :**
MongoDB Compass est l'interface utilisateur graphique pour MongoDB.
Cet outil permettre de visualiser, effectuer ,modifier des requêtes ou des regroupements sur la base de données.

— **Star UML :**
Star UML est le logiciel qu'on a utilisé pour réaliser les divers diagrammes de notre application.

2.1.3 Technologies

Dans cette section on va présenter les technologies et le Framework qu'on a utilisées dans la réalisation de notre projet.

2.1.3.1 Langages de programmation

— **python :**
Python est un langage de programmation open source le plus utilisé par les informaticiens. Il sert à la programmation d'applications , l'analyse des données et dans le domaine IA . Nous avons l'utiliser dans le web scraping et l'application des modèles IA.

2.1.3.2 Frameworks et librairies

— **ReactJs :**
ReactJs est l'une des bibliothèque JavaScript les plus populaire pour le développement des interfaces graphique. Nous avons l'utilisé pour construire la front-end de notre application.

— **Redux :**
Redux est un conteneur à état prévisible pour les applications Javascript. Elle permet de gérer l'état de l'application en un seul endroit et de garder les changements dans l'application plus prévisibles et traçables.

— **FastApi :**
FastApi est un Framework moderne et rapide utilisé pour les créations des API

avec Python. Nous avons l'utilisé pour le développement de notre back-end.

— **Beautiful Soup :**
Beautiful Soup est une bibliothèque python qui assure l'analyse syntaxique des documents XML et HTML. Cette bibliothèque est utilisé pour extraire les données d'une page web de manière lisible.

— **Scikit-learn :**
Scikit-learn est une librairie Python open source pour le machine Learning.
Elle offre dans sa structure des nombreuses bibliothèques d'algorithmes à implémenter.

2.2 Implementation et résultats

Dans cette section nous allons présenter notre solution détaillée en identifiant les étapes pour arriver au résultat.

2.2.1 Extraire les données

Nous avons collecté les données à partir de site proposé par l'utilisateur. Donc nous avons choisi d'utiliser le Beautiful Soup pour le wab scraping et l'extraction du texte de la page web donnée.
La figure montre l'algorithme utilisée pour scrapper le contenu de n'importe quel lien donné.

```
def collectData(soup,url):
    li = []
    text = ''
    find = soup.find_all(['p', 'span'])
    nom=name(url)
    reg = '(\w?\/n\s?)'
    listrem=['copyright','phone','télé',"mail",'contact','menu',"immeuble","voir plus" ,"powered by",'afficher plus' ,"tél",'déc
    listrep=["+","  ", reg , '\n ','\n',nom ,"#" ,'\xa0' ,'\\','%','_','/','\t','*','facebook','linkedin','youtube','tous droits
    for i in find:
        j = i.text.lower()
        j = ''.join(i for i in j if not i.isdigit())
        for k in listrep:
            j=j.replace(k,'')
        if (j != ''):
            li.append(j)
        for k in listrem :
            if k in j :
                if j in li:
                    li.remove(j)
```

FIGURE 2.1 – Code du Web Scrapping du contenu

2.2.2 Nettoyage des données

Dans cette partie , nous avons effectué quelques étapes pour améliorer la valeur , la fiabilité et la cohérence des informations extraites par le web scrpaing , donc nous avons :

— modifié tous les mots en minuscules,

— supprimé les « stopwords » et quelques mot qui ne sont pas nécessaires dans notre data,

— changé les verbes à l'infinitif en utilisant Lemmatisation.

```
text = " ".join(li)
text = re.sub(r'[^\w\s]', '', str(text).lower().strip())
lst_text = text.split()
if lst_stopwords is not None:
    lst_text = [word for word in lst_text if word not in lst_stopwords]
lem = nltk.stem.wordnet.WordNetLemmatizer()
lst_text = [lem.lemmatize(word) for word in lst_text]
text = " ".join(lst_text)
return(text)
```

FIGURE 2.2 – Code du nettoyage des données

2.2.3 Extraction des mots clés

Après l'extraction et le nettoyage des données , nous avons obtenu un texte qui contient des informations sur l'entreprise , mais ce texte est volumineux .
Donc afin d'optimiser notre résultat et diminuer le risque d'erreur, nous avons proposé un modèle d'extraction des Keywords « KeyBert ».
KeyBert est un modèle qui est capable d'extraire les mots clés ou les phrases clés en se basent sur Bert.Il trouve les sous-phrases dans un document qui sont les plus similaires au document lui-même.

```
from keybert import KeyBERT
def keywordsModel(text):
    kw_model = KeyBERT()
    keywords=kw_model.extract_keywords(text, keyphrase_ngram_range=(1, 3), stop_words=None,top_n=15,use_maxsum=True)
    l=''
    for kw in keywords:
        l=l+" "+kw[0]
    return(l)
```

```
[('offshoring teamwork', 0.4663), ('dapplications mobile site', 0.4878), ('web web design', 0.4908), ('offshoring teamwork di
t', 0.4918), ('mobile site web', 0.5011), ('boite développement offshore', 0.5206), ('application mobile web', 0.5416), ('offsh
ore tunisie', 0.5475), ('web design technology', 0.5677), ('offshore application', 0.5693)]
```

FIGURE 2.3 – Modéle KeyBert

2.2.4 Modèle BERT

BERT est un Modèle IA , appelée Masked-Language Modeling. Il est caractérisé par l'application de l'entraînement bidirectionnel de Transformer qui comprend le mécanisme d'attention.
Ce modèle applique une méthode de masquage aléatoire des mots dans la phrase , puis cherche à les prédire . Ce qui signifie qu'il regarde dans les deux sens gauche et droite afin d'utiliser tout le contexte de la phrase.
Donc , dans notre projet , nous avons choisi d'utiliser bert , alors nous avons proposé des mots clés qui caractérise une agence de développement pour chercher la similarité entre ces mots et les autres mots clé obtenu par le Modèle KeyBert.

```
from sentence_transformers import SentenceTransformerdef keywordsModel(text):
def SimilarityModel(string):
    sentences=['agence de développement entreprise solution télecom développement mobile web site web degital innovation informat
    model_name='bert-base-nli-mean-tokens'
    model= SentenceTransformer(model_name)
    sentence_embeddings = model.encode(sentences)
    from sklearn.metrics.pairwise import cosine_similarity
    similarity_score = cosine_similarity(
        [sentence_embeddings[0]],
        sentence_embeddings[1:]
    ).flatten()
    return(similarity_score)
```

	Sentence	Mot clé	Similarity_Score
0	offshoring teamwork dapplications mobile site...	agence de développement entreprise solution té...	0.794914

FIGURE 2.4 – Modéle Bert

2.2.5 Extraire des informations

Après la résultat obtenir du modèle Bert ,nous pouvons finalement extraire les informations demandées .
Donc nous avons utilisé le web scraping pour l'extraction des emails ,des numéros de téléphones et des technologies.
Les figures suivantes montrent les codes du web scraping des informations :

— **Contacts :**

```
def FindMail(soup):
    reg = '(\w+@\w+\.+\w+)'
    mail = re.findall(reg, soup.text)
    return(mail)

def FindPhone(soup):

    reg = "(\(?\+?\(?\d{1,3}\)?\s?\d{1,3}\s?\d{1,3}\s?\d{2,3}\s?\d{1,4}\s?\d{1,4}\s?\d{1,4})"
    phone = re.findall(reg, soup.text)
    return(phone)
```

FIGURE 2.5 – Code du Web Scrapping des contacts

— **Technologies :**

```
techno = serializeList(techno_collection.find())
list_techno = []
for i in techno:
    list_techno.append(i["name"])
techno = list_techno
l = [0 for i in range(35)]
for lien in list_liens:
    try:
        result = requests.get(lien)
        soup2 = BeautifulSoup(result.text, 'html.parser')
        text = soup2.text.lower()

        for i in techno:
            if i in text:
                l[techno.index(i)] = 1
    except requests.exceptions.ConnectTimeout or requests.exceptions.ReadTimeout:
        continue

technologie = []
for i in range(0, 35):
        if l[i] == 1:
        technologie.append(techno[i])

return(technologie)
```

FIGURE 2.6 – Code du Web Scrapping des technologies

2.3 Les Scénarios d'exécution

Dans cette section, nous allons présenter quelques captures d'écrans afin d'avoir un aperçu qui montre les scénarios d'exécution des interfaces de notre projet.

2.3.1 Connexion

La figure suivante représente la page qui permet à l'utilisateur de s'authentifier et accéder à notre plateforme en saisissant son nom et son mot de passe.

FIGURE 2.7 – La page de connexion

2.3.2 Sign up

Cette interface permet à l'utilisateur de s'inscrire et créer un compte en remplissant les champs suivants.

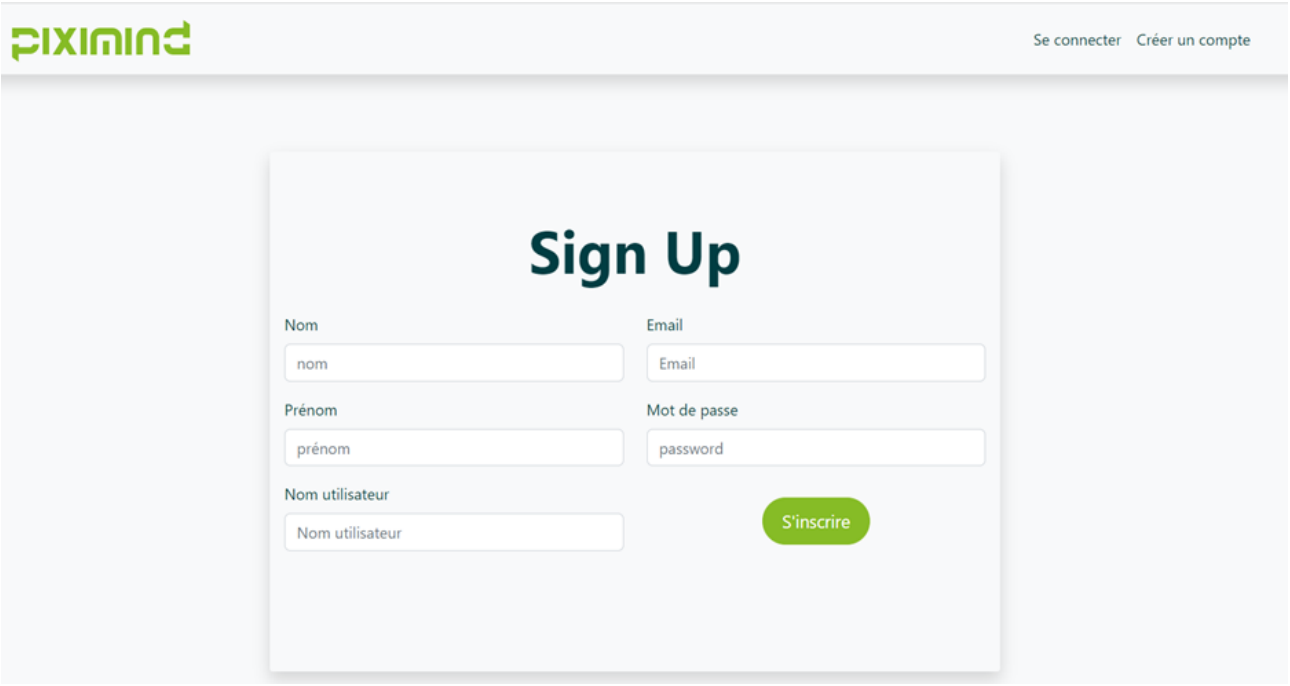

FIGURE 2.8 – La page de Sign up

2.3.3 Recherche

L'interface suivante permet à l'utilisateur de donner l'url d'un site d'une entreprise puis de cliquer sur le bouton de recherche pour obtenir le résultat.

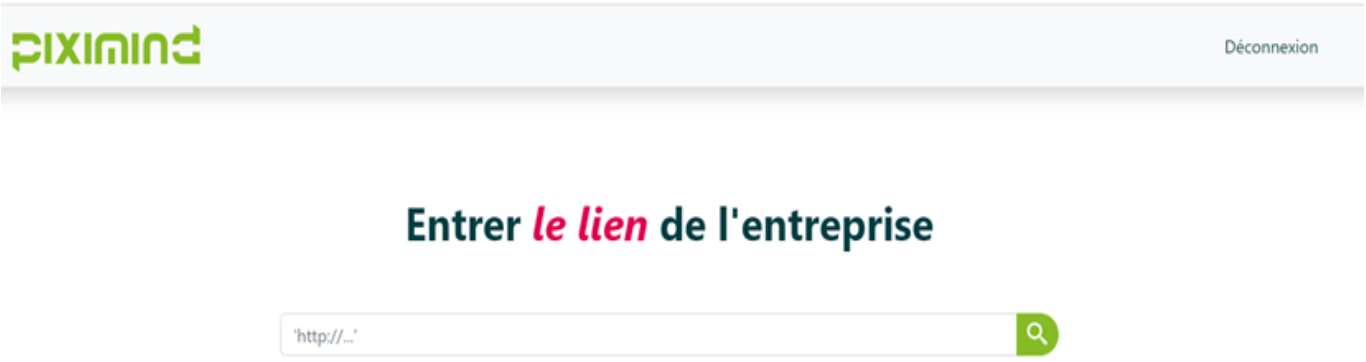

FIGURE 2.9 – La page de recheche

2.3.4 Résultat

Après le lancement de recherche , l'application affiche les résultat .
Alors il existe deux types de résultats :

— Si l'entreprise est une agence de développement ,l'application donne comme résultat les technologie et les contacts :

FIGURE 2.10 – Résultat

— Si l'entreprise n'est pas une agence de développement :

FIGURE 2.11 – Résultat

Conclusion

L'objet de ce chapitre est la phase de mise en œuvre du projet. Après l'achèvement de la phase de spécification des besoins et conception , on a présenté la réalisation de l'application. Tout d'abord, nous avons présenté une description et une spécification de l'environnement matériel de même que les différentes technologies utilisées. Puis, nous avons exposé des captures d'écran des principales interfaces de notre application. Finalement, nous avons effectué des tests unitaires pour prouver la validité des services web et leur fonctionnement correct.