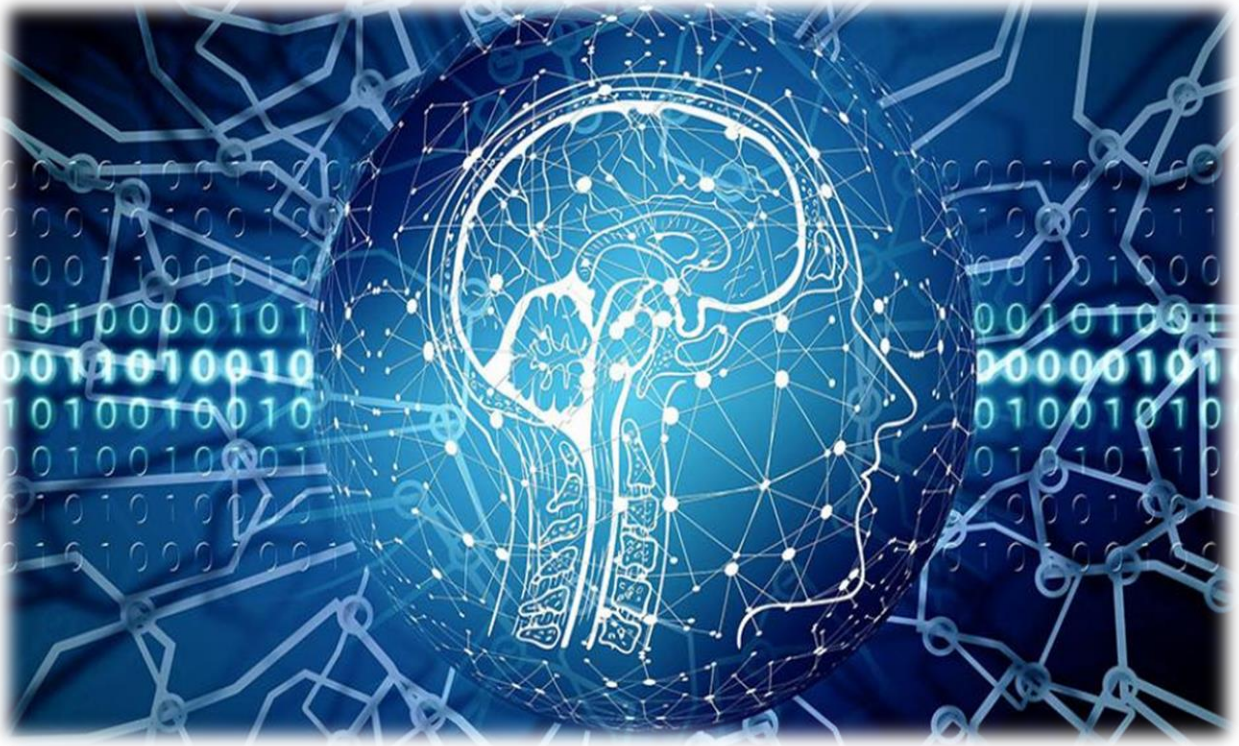
UNIVERSITÉ ABDELMALEK ESSAÂDI
FACULTÉ DES SCIENCES ET TECHNIQUES
TANGER

Rapport Projet

MODULE: Big Data Analytics

Section: MST-MBD S3 (2023-2024)



- **Réalisé par :**
  ELHAMAMI Chaimae
  EL YOUNSSI Oumaima
  ALAMI Youssra

- **Encadré par :**
  Lotfi EL AACHAK

I. Introduction :

L'évaluation automatique des courtes réponses (ASAG - Automatic Short Answer Grading) est une tâche informatique axée sur l'évaluation des réponses brèves en langage naturel à des questions objectives. Dans le domaine de l'éducation et de l'évaluation, l'ASAG joue un rôle crucial en automatisant le processus de notation des réponses courtes, offrant ainsi une solution évolutive et efficace pour les éducateurs. La recherche active dans ce domaine a considérablement augmenté récemment, avec plus de 80 articles répondant à la définition de l'ASAG.

Cette tâche a attiré une attention significative en raison de son potentiel pour révolutionner le paysage de l'évaluation, offrant des avantages tels que la rapidité, la cohérence et l'objectivité dans la notation. Les systèmes ASAG exploitent diverses techniques de traitement du langage naturel (NLP) et d'apprentissage automatique pour analyser et évaluer les réponses courtes, fournissant des informations précieuses sur la compréhension et les connaissances des étudiants.

Les différentes approches de l'ASAG impliquent souvent l'utilisation de réseaux neuronaux récurrents (RNN) et d'autres architectures d'apprentissage profond. Ces modèles sont formés sur des ensembles de données annotés, apprenant à reconnaître des motifs et des relations au sein des réponses courtes qui sont corrélés à différents niveaux de compréhension ou de justesse.

II. Description de projet :

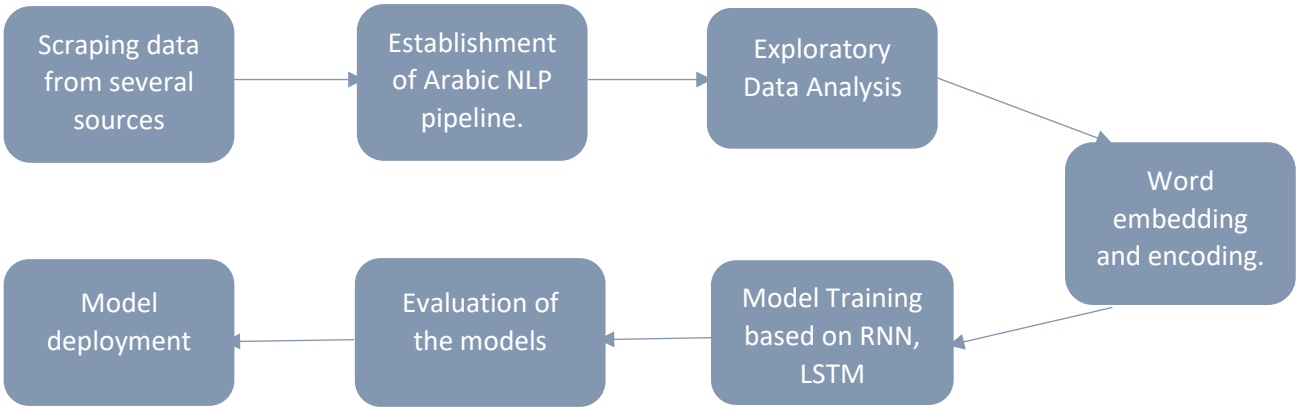
Dans le cadre de ce projet, nous avons pour objectif de développer un système automatisé de notation des courtes réponses en arabe, avec comme idée fondamentale l'assistance intelligente à l'éducation islamique des élèves. L'objectif principal est d'attribuer une note adéquate aux étudiants en fonction de leurs réponses (notes de 0 à 20). Le système est entièrement en arabe, et un jeu de données personnalisé a été créé en collectant des données brutes à partir de diverses sources à l'aide de techniques de scraping, en utilisant la bibliothèque BeautifulSoup.

Les étapes clés de ce projet comprennent la collecte de données provenant de diverses sources, notamment des sites web arabes, des ensembles de données et des ouvrages. Ensuite, un pipeline de traitement du langage naturel en arabe a été établi, suivi de l'application de techniques d'analyse exploratoire des données pour mieux appréhender les informations recueillies.

Le processus englobe également la création d'encodages et d'embeddings de mots en arabe, préalable à l'entraînement du modèle. Nous avons travaillé avec trois architectures de modèles distinctes, adaptées au traitement des langues naturelles en arabe : RNN, LSTM et Transformer. Nous avons exploré ces trois architectures pour observer les différences entre leurs résultats. Finalement, nous avons opté pour le modèle LSTM, que nous présenterons par la suite dans ce rapport.

➔ Schéma de projet :

Ce schéma montre les étapes cruciales que nous avons suivies pour réaliser ce projet :



III. Technologies :

1- Création de modèle de DL :

NLTK : signifie Natural Language Toolkit, qui est une plateforme populaire open-source pour construire des programmes Python pour travailler avec des données de langage humain. Il fournit une large gamme d'outils et de ressources pour diverses tâches de traitement de langage naturel, telles que la tokenisation, l'étiquetage des parties du discours, la reconnaissance des entités nommées, l'analyse de sentiment et l'apprentissage automatique pour la modélisation et la classification du langage.

Scikit-learn : est une bibliothèque open-source de machine learning spécialement conçue pour Python. Elle met à disposition une gamme complète d'outils simples et performants destinés à des tâches variées telles que la classification, la régression, le clustering, ainsi que la sélection de modèles en machine learning. Dans le cadre de ce projet, Scikit-learn joue un rôle essentiel en s'impliquant dans différentes phases du processus, notamment le traitement des données, la division des ensembles de données, l'évaluation des modèles, et bien d'autres aspects cruciaux.

Gensim : est une bibliothèque Open Source de traitement de langage naturel (NLP) en Python dont le but est de rendre la modélisation de sujet aussi facile d'accès et efficace que possible, Dans le cadre de ce projet, Gensim joue un rôle crucial dans le traitement des données textuelles en arabe, Grâce à son ensemble d'outils dédiés au traitement du langage naturel, Gensim permet d'effectuer des opérations telles que la création de modèles de vecteurs de mots (Word2Vec) et la génération de représentations vectorielles des mots en arabe.

BeautifulSoup : une bibliothèque Python puissante pour l'extraction d'informations à partir de pages web HTML et XML, est utilisée pour le processus de scraping de données. Cette étape cruciale consiste à collecter des données à partir de diverses sources en ligne, telles que des sites web arabes, des ensembles de données spécifiques, et d'autres ressources éducatives.

Pandas : est une bibliothèque open-source pour la manipulation et l'analyse de données en Python. Elle fournit des structures de données hautement performantes pour stocker et manipuler des données tabulaires et des séries temporelles.

Numpy : est une bibliothèque open-source pour le calcul numérique en Python. Elle fournit un objet de tableau multidimensionnel de haute performance (ndarray), ainsi que des fonctions pour travailler avec ces tableaux.

2- Déploiement de modèle :

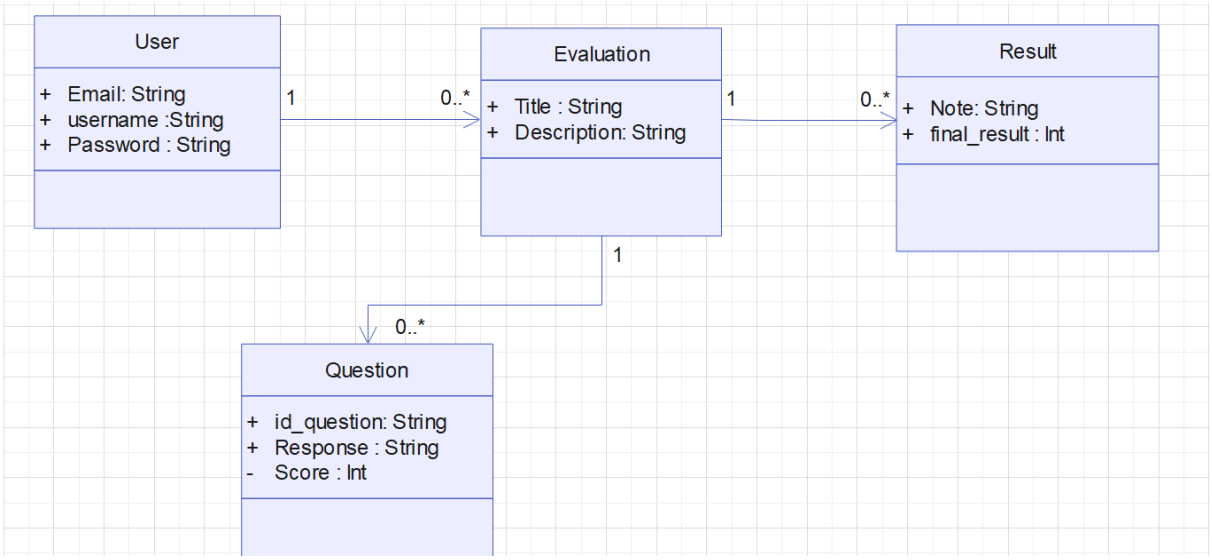
Angular : est un framework open source largement utilisé pour la création d'applications web dynamiques et interactives. Angular jouait un rôle central dans le projet en tant que gestionnaire de la logique et de l'interface utilisateur côté client. Il nous a permis de structurer le code de manière modulaire, facilitant ainsi la création de composants réutilisables et la gestion efficace de l'état de l'application.

Flask : un n framework web léger pour Python, joue un rôle essentiel dans le déploiement de l'application et la mise en œuvre de l'interface utilisateur, elle offre une structure simple et flexible pour construire des applications web, ce qui facilite l'intégration de notre système d'évaluation automatisée et d'assistance intelligente pour l'éducation islamique dans une interface conviviale. Grâce à Flask, nous pouvons créer des routes, définir des modèles, et gérer les interactions entre le backend et le frontend de manière efficace.

MongoDB : est un système de gestion de base de données orienté documents, il a été intégré au projet en tant que système de gestion de base de données NoSQL, offrant une solution flexible et évolutive pour le stockage des données. Sa structure orientée document a permis de stocker les informations de manière naturelle, en suivant le modèle JSON, facilitant ainsi l'intégration avec le traitement du langage naturel en arabe.

Docker : est une plateforme logicielle qui facilite la création, le déploiement et l'exécution d'applications dans des environnements isolés appelés conteneurs. Les conteneurs Docker encapsulent une application avec ses dépendances et son environnement d'exécution, assurant ainsi une portabilité et une cohérence entre les différents environnements.

IV. Modelisation :



V. Modèles d'apprentissage profond

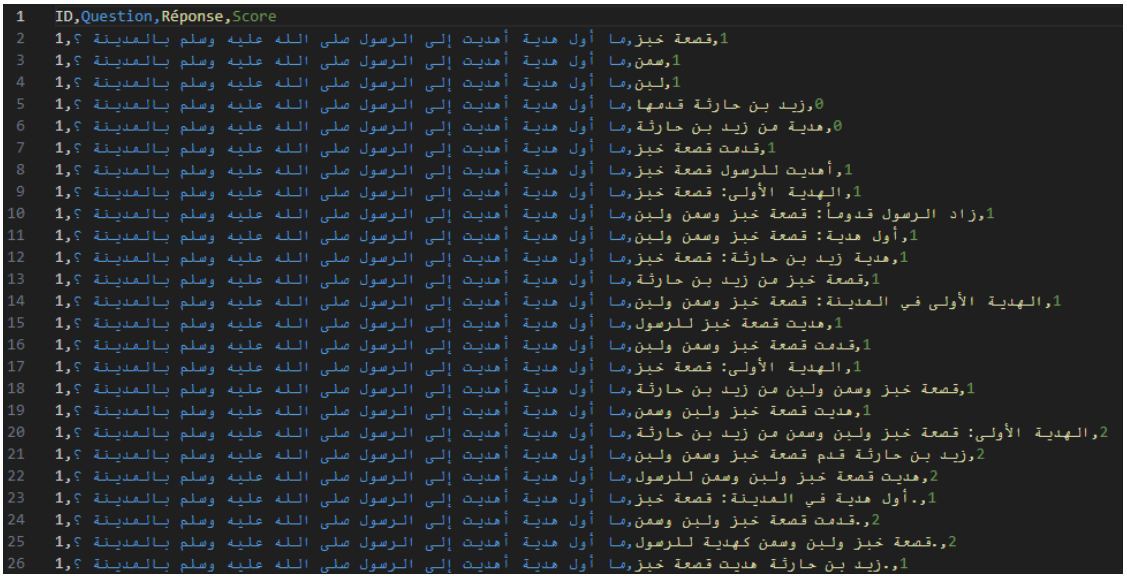
1. Collection de données :

Dans cette section, nous avons rassemblé des données provenant de diverses sources, notamment des sites web comportant des questions ainsi que leurs réponses en éducation islamique. Nous avons ensuite utilisé un script Python à cet effet.

scrap data.py > ...

```python
import requests
from bs4 import BeautifulSoup

# URL de la page contenant les questions en arabe
url = "https://mawdoo3.com/%D8%A3%D8%B3%D8%A6%D9%84%D8%A9_%D8%B9%D8%A7%D9%85%D8%A9_%D8%AF%D9%8A%D9%86%D9%8A%D8%A9_%

# Envoyer une requête GET à la page
response = requests.get(url)

# Vérifier si la requête a réussi (statut 200)
if response.status_code == 200:
    # Parser le contenu HTML avec BeautifulSoup
    soup = BeautifulSoup(response.text, 'html.parser')

    # Trouver les éléments contenant les questions (cet exemple suppose l'utilisation d'une classe CSS spécifique)
    question_elements = soup.find_all('div', class_='question-class')

    # Parcourir les éléments des questions et afficher le texte
    for question_element in question_elements:
        question_text = question_element.get_text(strip=True)
        print(question_text)

else:
    print(f"La requête a échoué avec le code d'état {response.status_code}")
```

Voilà les données collectées :



ID	Question	Réponse	Score
1	؟	أول هدية أهديت إلى الرسول صلى الله عليه وسلم بالمدينة ما	قصعة خبز,1
1	؟	أول هدية أهديت إلى الرسول صلى الله عليه وسلم بالمدينة ما	سمن,1
1	؟	أول هدية أهديت إلى الرسول صلى الله عليه وسلم بالمدينة ما	لبن,1
1	؟	أول هدية أهديت إلى الرسول صلى الله عليه وسلم بالمدينة ما	زيد بن حارثة قدمها,0
1	؟	أول هدية أهديت إلى الرسول صلى الله عليه وسلم بالمدينة ما	هدية من زيد بن حارثة,0
1	؟	أول هدية أهديت إلى الرسول صلى الله عليه وسلم بالمدينة ما	قدمت قصعة خبز,1
1	؟	أول هدية أهديت إلى الرسول صلى الله عليه وسلم بالمدينة ما	أهديت للرسول قصعة خبز,1
1	؟	أول هدية أهديت إلى الرسول صلى الله عليه وسلم بالمدينة ما	الهدية الأولى: قصعة خبز,1
1	؟	أول هدية أهديت إلى الرسول صلى الله عليه وسلم بالمدينة ما	زاد الرسول قدوماً: قصعة خبز وسمن ولبن,1
1	؟	أول هدية أهديت إلى الرسول صلى الله عليه وسلم بالمدينة ما	أول هدية: قصعة خبز وسمن ولبن,1
1	؟	أول هدية أهديت إلى الرسول صلى الله عليه وسلم بالمدينة ما	هدية زيد بن حارثة: قصعة خبز,1
1	؟	أول هدية أهديت إلى الرسول صلى الله عليه وسلم بالمدينة ما	قصعة خبز من زيد بن حارثة,1
1	؟	أول هدية أهديت إلى الرسول صلى الله عليه وسلم بالمدينة ما	الهدية الأولى في المدينة: قصعة خبز وسمن ولبن,1
1	؟	أول هدية أهديت إلى الرسول صلى الله عليه وسلم بالمدينة ما	هديت قصعة خبز للرسول,1
1	؟	أول هدية أهديت إلى الرسول صلى الله عليه وسلم بالمدينة ما	قدمت قصعة خبز وسمن ولبن,1
1	؟	أول هدية أهديت إلى الرسول صلى الله عليه وسلم بالمدينة ما	الهدية الأولى: قصعة خبز,1
1	؟	أول هدية أهديت إلى الرسول صلى الله عليه وسلم بالمدينة ما	قصعة خبز وسمن ولبن من زيد بن حارثة,1
1	؟	أول هدية أهديت إلى الرسول صلى الله عليه وسلم بالمدينة ما	هديت قصعة خبز ولبن وسمن,1
1	؟	أول هدية أهديت إلى الرسول صلى الله عليه وسلم بالمدينة ما	الهدية الأولى: قصعة خبز ولبن وسمن من زيد بن حارثة,2
1	؟	أول هدية أهديت إلى الرسول صلى الله عليه وسلم بالمدينة ما	زيد بن حارثة قدم قصعة خبز وسمن ولبن,2
1	؟	أول هدية أهديت إلى الرسول صلى الله عليه وسلم بالمدينة ما	هديت قصعة خبز ولبن وسمن للرسول,2
1	؟	أول هدية أهديت إلى الرسول صلى الله عليه وسلم بالمدينة ما	أول هدية في المدينة: قصعة خبز,1
1	؟	أول هدية أهديت إلى الرسول صلى الله عليه وسلم بالمدينة ما	قدمت قصعة خبز ولبن وسمن,2
1	؟	أول هدية أهديت إلى الرسول صلى الله عليه وسلم بالمدينة ما	قصعة خبز ولبن وسمن كهدية للرسول,2
1	؟	أول هدية أهديت إلى الرسول صلى الله عليه وسلم بالمدينة ما	زيد بن حارثة هديت قصعة خبز,1

Afin de former nos modèles, nous avons segmenté les données en plusieurs fichiers CSV, chacun contenant les réponses associées à une question spécifique. Cette approche nous a permis de former le modèle individuellement sur chaque question.

Question_1	1/1/2024 5:32 PM	Fichier source Co...	18 Ko
Question_2	1/1/2024 5:47 PM	Fichier source Co...	12 Ko
Question_3	1/1/2024 5:47 PM	Fichier source Co...	26 Ko
Question_4	1/1/2024 5:47 PM	Fichier source Co...	19 Ko
Question_5	1/1/2024 5:47 PM	Fichier source Co...	24 Ko
Question_6	1/1/2024 5:48 PM	Fichier source Co...	16 Ko
Question_7	1/1/2024 5:48 PM	Fichier source Co...	16 Ko
Question_8	1/1/2024 5:48 PM	Fichier source Co...	11 Ko
Question_9	1/10/2024 12:16 AM	Fichier source Co...	13 Ko
Question_10	1/1/2024 5:48 PM	Fichier source Co...	11 Ko
Question_11	1/1/2024 5:48 PM	Fichier source Co...	18 Ko
Question_12	1/1/2024 5:48 PM	Fichier source Co...	25 Ko
Question_13	1/1/2024 5:48 PM	Fichier source Co...	24 Ko
Question_14	1/1/2024 5:48 PM	Fichier source Co...	19 Ko
Question_15	1/1/2024 5:48 PM	Fichier source Co...	15 Ko

2. EDA

Cette étape nous a permis de plonger dans les détails de nos jeux de données, en identifiant les tendances, les modèles et les caractéristiques clés. À travers des visualisations graphiques, des statistiques descriptives, et d'autres techniques analytiques, nous avons pu mettre en lumière des informations pertinentes et significatives.

NB : On a procédé de la même manière pour toutes les questions.

→ Chargement de données :

```
question11 = pd.read_csv('/kaggle/input/datasetar/Data/Question_11.csv', sep=',')
question11.head()
```

	ID	Question	Réponse	Score
0	11	ما هي مهنة النّبيّ -صلّى الله عليه وسلّم- قب...	كان -عليه الصّلاة والسّلام- يعمل برعي الأغنام ...	2
1	11	ما هي مهنة النّبيّ -صلّى الله عليه وسلّم- قب...	كان راعيًا للأغنام	1
2	11	ما هي مهنة النّبيّ -صلّى الله عليه وسلّم- قب...	كان يمارس مهنة الرعي	1
3	11	ما هي مهنة النّبيّ -صلّى الله عليه وسلّم- قب...	كان يشتغل كراعي للمواشي	1
4	11	ما هي مهنة النّبيّ -صلّى الله عليه وسلّم- قب...	كان يعمل في رعاية الحيوانات	0

→ Vérification de la colonne Score

```
question11['Score'].unique()
```

```
array([2, 1, 0])
```

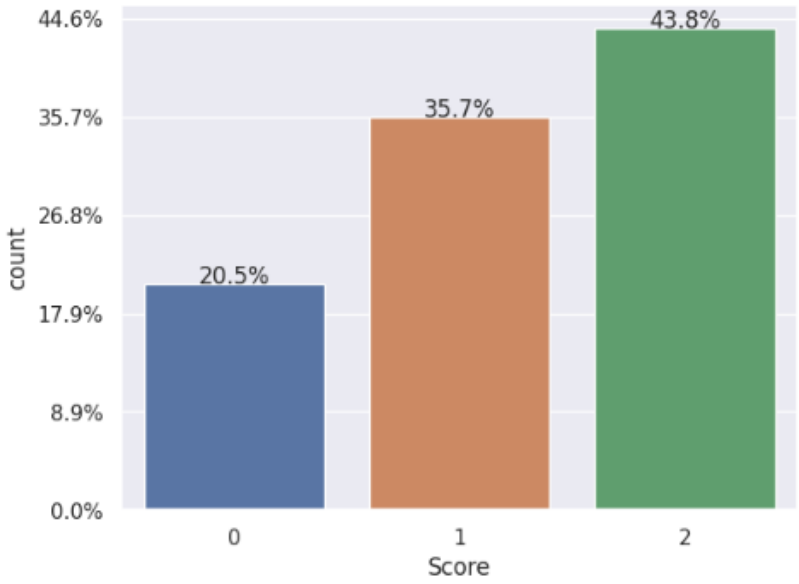
→ Vérification de l'absence de valeurs nulles.

```
question11.isnull().sum()
```

```
ID          0
Question    0
Réponse     0
Score       0
dtype: int64
```

→ Visualisation de données

```
sns.set(style="darkgrid")
ax = sns.countplot(data=question11, x='Score')
total = len(question11)
for p in ax.patches:
    height = p.get_height()
    ax.text(p.get_x() + p.get_width() / 2., height + 0.1,
            f'{height/total:.1%}', ha="center")
ax.yaxis.set_major_formatter(mtick.PercentFormatter(xmax=total))
plt.show()
```



On remarque que notre ensemble de données est équilibrée.

3- NLP pipeline

→ Suppression des Stopwords

```
#Drop stop words
from nltk.stem.arlstem import ARLSTem
stemmmer = ARLSTem()

def remove_stowords(elements):
    corps = []
    for string in elements :
        string = nltk.sent_tokenize(string.strip())
        string = [ stemmmer.stem(word) for word in string if not word in stop_words ]
        string = ''.join(string)
        corps.append(string)
    return corps
```

→ Tokenisation

```
tokenizer = Tokenizer(filters='''!"#$%&()+,-./:;<=>?@[\\]^{|}~\t\n÷×–“…"!|+¦~{}',.؟":/~][%^&()<>؛''''' )

# Fit the tokenizer on the training texts
tokenizer.fit_on_texts(corps11)

with open('TokenizerModel11.pkl', 'wb') as tokenizer_file:
    pickle.dump(tokenizer, tokenizer_file)

sequences = tokenizer.texts_to_sequences(corps11)

max_sequence_length = 35
sequences = pad_sequences(sequences, max_sequence_length)

vocab_dict = tokenizer.word_index

vocab_size = len(vocab_dict) + 1
```

→ Word Embedding

Cette étape vise à représenter les mots d'un vocabulaire sous forme de vecteurs numériques dans un espace continu. En utilisant le fameux modèle de vectorisation gensim.

```
from gensim.models import KeyedVectors
fasttext_model = KeyedVectors.load_word2vec_format("/kaggle/input/fast-text-translation-data/fast_text_files/vectors/wiki.ar.vec")
```

```
from keras.layers import Embedding
import numpy as np


EMBEDDING_DIM = 300
num_words = len(vocab_dict) + 1
count = 0

embedding_matrix = np.zeros((num_words, EMBEDDING_DIM))

for word, idx in vocab_dict.items():
    if word in fasttext_model:
        embedding_matrix[idx] = fasttext_model.get_vector(word)
    else:
        count += 1
        print("Word does not exist in the vocabulary ---> " + word)
        embedding_matrix[idx] = fasttext_model.get_vector("unk")
```

```
Word does not exist in the vocabulary ---> للاغنام
Word does not exist in the vocabulary ---> بالاغنام
Word does not exist in the vocabulary ---> وبالتجار
Word does not exist in the vocabulary ---> والمراعا
Word does not exist in the vocabulary ---> بيراع
```

→ Modélisation de données :

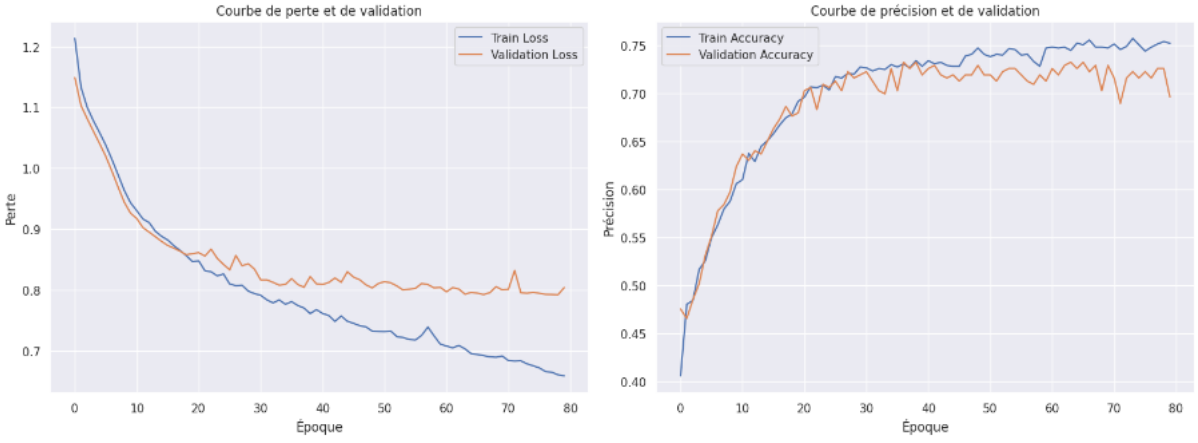
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

4- RNN

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, SimpleRNN, Dense,Dropout
from tensorflow.keras.regularizers import l2


input_dim=vocab_size
output_dim=33
max_sequence=33

model = Sequential([
Embedding(input_dim=input_dim, output_dim=output_dim,
mask_zero=True, input_length=max_sequence,
trainable=False, embeddings_initializer=tf.keras.initializers.
random_normal()),
SimpleRNN(units=10,kernel_regularizer=l2(0.01)),
Dense(3,activation='softmax')
])
model.summary()
```

→ Entraînement du modèle et visualisation des résultats

```
# Compile the model
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])

# Train the model
history = model.fit(X_train, y_train, validation_data=(X_test, y_test),
                    batch_size=64, epochs=80, verbose=1)



plt.figure(figsize=(16, 6))
# Courbe de perte
plt.subplot(1, 2, 1)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Courbe de perte et de validation')
plt.xlabel('Époque')
plt.ylabel('Perte')
plt.legend()


# Courbe de précision
plt.subplot(1, 2, 2)
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Courbe de précision et de validation')
plt.xlabel('Époque')
plt.ylabel('Précision')
plt.legend()

plt.tight_layout()
plt.show()
```




```
scores_trainig = model.evaluate(X_train, y_train, verbose=1)
print("Training Loss: %f%%" % (scores_trainig[0]))
print("Training Accuracy: %.2f%%" % (scores_trainig[1]*100))
```

```
38/38 [==============================] - 0s 7ms/step - loss: 0.6611 - accuracy: 0.7533
Training Loss: 0.661077%
Training Accuracy: 75.33%
```

```
scores_test = model.evaluate(X_test, y_test, verbose=1)
print("Test Loss: %f%%" % (scores_test[0]))
print("Test Accuracy: %.2f%%" % (scores_test[1]*100))
```

```
10/10 [==============================] - 0s 8ms/step - loss: 0.8037 - accuracy: 0.6964
Test Loss: 0.803703%
Test Accuracy: 69.64%
```

5- LSTM

```
from keras.models import Sequential
from keras.layers import LSTM, Dense, Embedding
from tensorflow.keras.regularizers import l2
from tensorflow.keras.regularizers import l1

input_dim=vocab_size
output_dim=35
input_length=35


# Define the LSTM model
model = Sequential()
model.add(Embedding(input_dim=input_dim, output_dim=output_dim, input_length=input_length))
model.add(LSTM(units=10,kernel_regularizer=l2(0.0001)))
model.add(Dense(3, activation='softmax'))
```

→ L’entraînement du modèle et l’affichage du résultats :

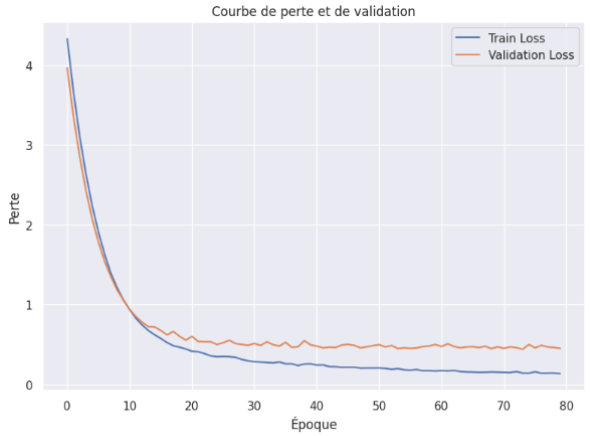
```
# Compile the model
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# Train the model
history = model.fit(X_train, y_train, validation_data=(X_test, y_test),
                    batch_size=64, epochs=80, verbose=1)

plt.figure(figsize=(16, 6))
# Courbe de perte
plt.subplot(1, 2, 1)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Courbe de perte et de validation')
plt.xlabel('Époque')
plt.ylabel('Perte')
plt.legend()

# Courbe de précision
plt.subplot(1, 2, 2)
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Courbe de précision et de validation')
plt.xlabel('Époque')
plt.ylabel('Précision')
plt.legend()

plt.tight_layout()
plt.show()
```

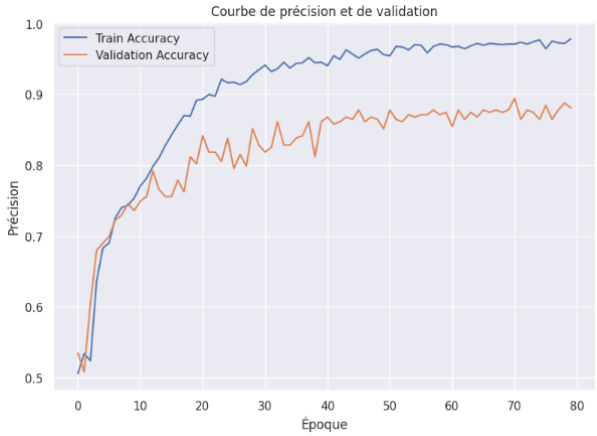
```
scores_trainig = model.evaluate(X_train, y_train, verbose=1)
print("Training Loss: %f%%" % (scores_trainig[0]))
print("Training Accuracy: %.2f%%" % (scores_trainig[1]*100))
```

```
3/3 [==============================] - 0s 6ms/step - loss: 0.1761 - accuracy: 1.0000
Training Loss: 0.176145%
Training Accuracy: 100.00%
```

```
scores_test = model.evaluate(X_test, y_test, verbose=1)
print("Test Loss: %f%%" % (scores_test[0]))
print("Test Accuracy: %.2f%%" % (scores_test[1]*100))
```

```
1/1 [==============================] - 0s 29ms/step - loss: 0.2733 - accuracy: 0.9565
Test Loss: 0.273258%
Test Accuracy: 95.65%
```





Ce modèle nous a donné des bons résultats c'est pour ça on a décidé de l'utiliser dans notre application

→ Sauvegarde du modèle :

```
from keras.models import load_model

# # Assuming 'model' is your Keras model
model.save('model11.h5')
```

6- Transformer

```
from tensorflow.keras import layers
class TokenAndPositionEmbedding(layers.Layer):
    def __init__(self, maxlen, vocab_size, embed_dim ,embedding_matrix):
        super(TokenAndPositionEmbedding, self).__init__()
        self.token_emb = layers.Embedding(input_dim=vocab_size,weights=[embedding_matrix], output_dim=embed_dim)
        self.pos_emb = layers.Embedding(input_dim=maxlen, output_dim=embed_dim)

    def call(self, x):
        maxlen = tf.shape(x)[-1]
        positions = tf.range(start=0, limit=maxlen, delta=1)
        positions = self.pos_emb(positions)
        x = self.token_emb(x)
        return x + positions
```

```python
class TransformerBlock(layers.Layer):
    def __init__(self, embed_dim, num_heads, ff_dim, rate=0.1):
        super(TransformerBlock, self).__init__()
        self.att = layers.MultiHeadAttention(num_heads=num_heads, key_dim=embed_dim)
        self.ffn = keras.Sequential(
            [layers.Dense(ff_dim, activation="relu"), layers.Dense(embed_dim),]
        )
        self.layernorm1 = layers.LayerNormalization(epsilon=1e-6)
        self.layernorm2 = layers.LayerNormalization(epsilon=1e-6)
        self.dropout1 = layers.Dropout(rate)
        self.dropout2 = layers.Dropout(rate)


    def call(self, inputs, training):
        attn_output = self.att(inputs, inputs)
        attn_output = self.dropout1(attn_output, training=training)
        out1 = self.layernorm1(inputs + attn_output)
        ffn_output = self.ffn(out1)
        ffn_output = self.dropout2(ffn_output, training=training)
        return self.layernorm2(out1 + ffn_output)
```

```python
import keras
num_heads = 1
ff_dim = 8

inputs = layers.Input(shape=(max_sequence_length,))
embedding_layer = TokenAndPositionEmbedding(max_sequence_length, vocab_size,
                                            EMBEDDING_DIM,embedding_matrix)
x = embedding_layer(inputs)
transformer_block = TransformerBlock(EMBEDDING_DIM, num_heads, ff_dim,0.3)
x = transformer_block(x)
x = layers.GlobalMaxPooling1D()(x)
x = layers.Dropout(0.2)(x)
outputs = layers.Dense(3, activation="softmax",
                       kernel_regularizer=tf.keras.regularizers.l2(0.0001))(x)

model = keras.Model(inputs=inputs, outputs=outputs)
```

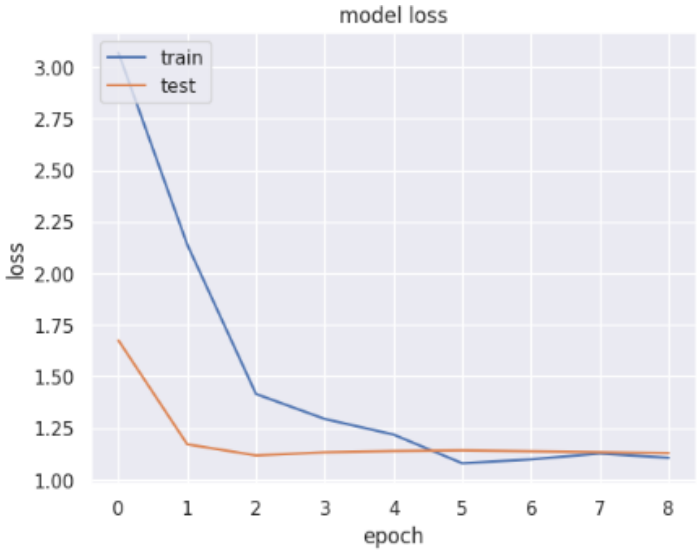
→ Résultat :

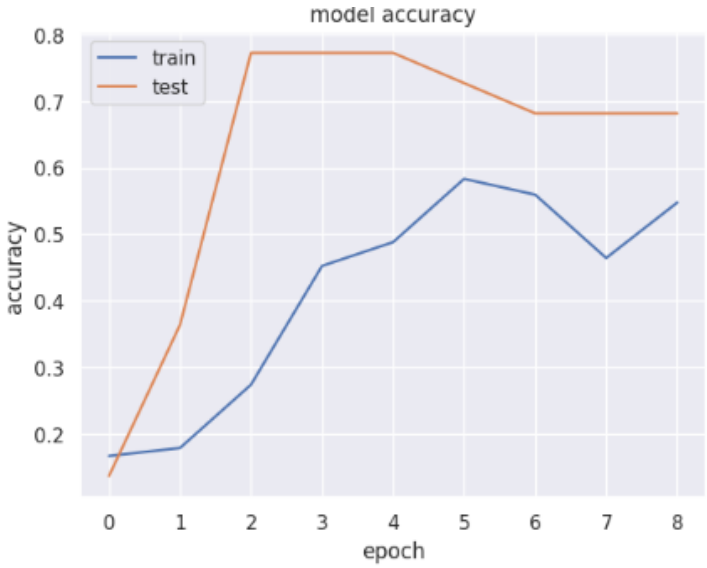
```python
scores_trainig = model.evaluate(X_train1, y_train1, verbose=1)
print("Training Loss: %f%%" % (scores_trainig[0]))
print("Training Accuracy: %.2f%%" % (scores_trainig[1]*100))
print("Training Precision: %.2f%%" % (scores_trainig[2]*100))
print("Training Recall: %.2f%%" % (scores_trainig[3]*100))
print("Training F1 Score: %.2f%%" % (scores_trainig[4]*100))
print("Training Cohen Kappa: %.2f%%" % (scores_trainig[5]*100))
```

```
3/3 [==============================] - 0s 15ms/step - loss: 0.2715 - accuracy: 0.9167 - precision_26: 0.9146 - recall_26: 0.8929 - f1_m: 0.9062 - cohen_kappa: 0.8555
Training Loss: 0.271452%
Training Accuracy: 91.67%
Training Precision: 91.46%
Training Recall: 89.29%
Training F1 Score: 90.62%
Training Cohen Kappa: 85.55%
```

```python
scores_test = model.evaluate(X_test, y_test, verbose=1)
print("Test Loss: %f%%" % (scores_test[0]))
print("Test Accuracy: %.2f%%" % (scores_test[1]*100))
print("Test Precision: %.2f%%" % (scores_test[2]*100))
print("Test Recall: %.2f%%" % (scores_test[3]*100))
print("Test F1 Score: %.2f%%" % (scores_test[4]*100))
print("Test Cohen Kappa: %.2f%%" % (scores_test[5]*100))
```

```
10/10 [==============================] - 1s 8ms/step - loss: 2.1749 - accuracy: 0.3630 - precision_26: 0.3740 - recall_26: 0.3036 - f1_m: 0.3466 - cohen_kappa: 0.0541
Test Loss: 2.174950%
Test Accuracy: 36.30%
Test Precision: 37.40%
Test Recall: 30.36%
Test F1 Score: 34.66%
Test Cohen Kappa: 5.41%
```





VI. Déploiement de l'application

→ Api Flask

```python
def remove_stowords(elements):
    stemmmer = ARLSTem()
    corps = []
    for string in elements :
        string = nltk.sent_tokenize(string.strip())
        string = [ stemmmer.stem(word) for word in string if not word in stop_words ]
        string = ''.join(string)
        corps.append(string)
    return corps
```

```python
Folder_tokenizers = './tokenizers'
def load_tokenizer_by_number(folder_path, target_number):
    # List files in the folder
    files = os.listdir(folder_path)

    # Filter files that end with the specified number

    matching_files = [file for file in files if file == f'{target_number}.pkl']

    if not matching_files:
        print(f"No files found ending with {target_number}.pkl")
        return None

    # Select the first matching file
    selected_file = matching_files[0]

    file_path = os.path.join(folder_path, selected_file)

    # Load the tokenizer from the file
    with open(file_path, 'rb') as tokenizer_file:
        tokenizer = pickle.load(tokenizer_file)

    return tokenizer
```

```python
Folder_models = './models'
def load_model_by_number(folder_path, target_number):
    # List files in the folder
    files = os.listdir(folder_path)

    # Filter files that end with the specified number
    matching_files = [file for file in files if file == f'{target_number}.h5']


    if not matching_files:
        print(f"No files found ending with {target_number}.h5")
        return None

    # Select the first matching file
    selected_file = matching_files[0]
    file_path = os.path.join(folder_path, selected_file)

    # Load the model from the file
    model = load_model(file_path)

    return model

```

```python
def predict_sequence(response,id):

    # Convert texts to sequences of integers
    tokenizer_charge = load_tokenizer_by_number(Folder_tokenizers, id)
    sequences = tokenizer_charge.texts_to_sequences([response])

    # Pad sequences to ensure uniform length
    max_sequence_length = 35
    sequences = pad_sequences(sequences, max_sequence_length)

    # Make predictions
    model = load_model_by_number(Folder_models, id)
    predicted_probs = model.predict(sequences)
    predicted_classe = np.argmax(predicted_probs, axis=1)

    return predicted_classe

```

```python
@app.route('/lists', methods=['POST'])
def rec_listes():
 Results=[]

 data = request.json
 Responses = data.get('Responses', [])
 ids = data.get('ids', [])

 #print(Responses)
 corps= remove_stowords(Responses)

 for response, id_value in zip(corps, ids):
    if not response.isspace() and response.strip() != "":
        result=predict_sequence(response,id_value)
        Results.append(result[0])

    else :
        result=0
        Results.append(result)

 Results = [item if isinstance(item, (int, float, str)) else item.item() for item in Results]

 return jsonify({'Results': Results})

```

→ Fichier Docker :

- Côté serveur :

Pour construire et exécuter notre application, nous avons utilisé Docker. Nous avons créé un fichier Dockerfile et un fichier docker-compose.yml. Le Dockerfile contient le code suivant :

```
FROM python

# Set the working directory in the container
WORKDIR /app

# Copy the requirements file to the container
COPY ./requirements.txt .

# Install dependencies
RUN pip install -r requirements.txt

# Copy the rest of the project files to the container
COPY . .

# Expose the port that the application will be running on
EXPOSE 8000
```

Le fichier docker-compose.yml contient le code suivant :

docker-compose.yml

```
version: '3.10'
services:
  app:
    build: .
    command: uvicorn main:app --host 0.0.0.0
    ports:
      - "8000:8000"
```

Pour construire et exécuter l'application dans Docker, nous avons utilisé la commande suivante : **docker-compose up**

- Côté client :

Pour construire et exécuter notre application frontend, nous avons créé un Dockerfile contenant le code suivant :

```
FROM node:18.13.0-alpine as build
RUN mkdir -p /app

WORKDIR /app
```

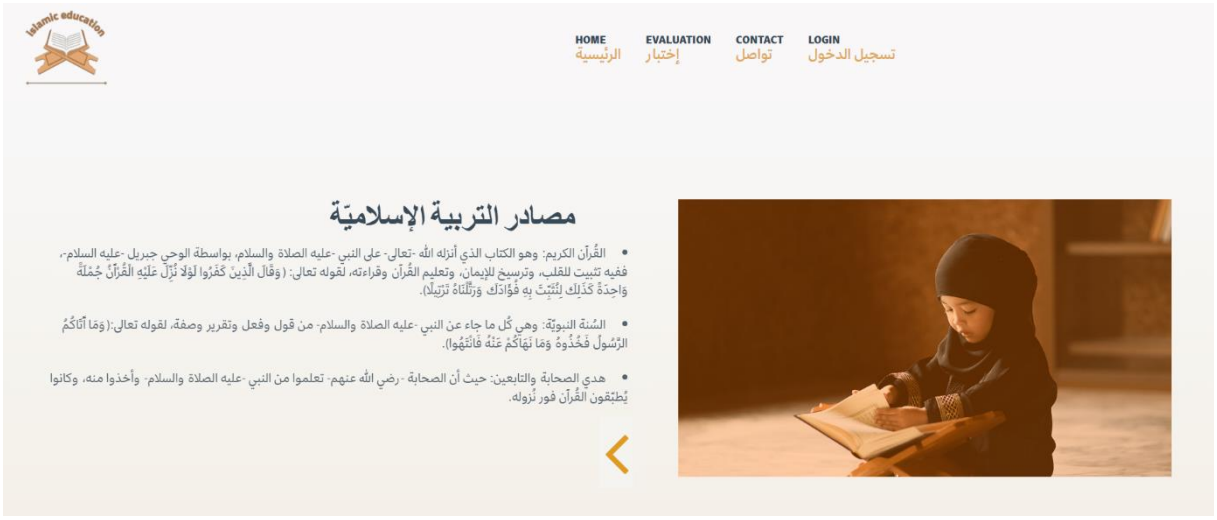
```
COPY package.json /app/
RUN npm install

COPY . /app/
RUN npm run build --prod

FROM nginx:alpine
COPY --from=build /app1/dist/UI /usr/share/nginx/html
```

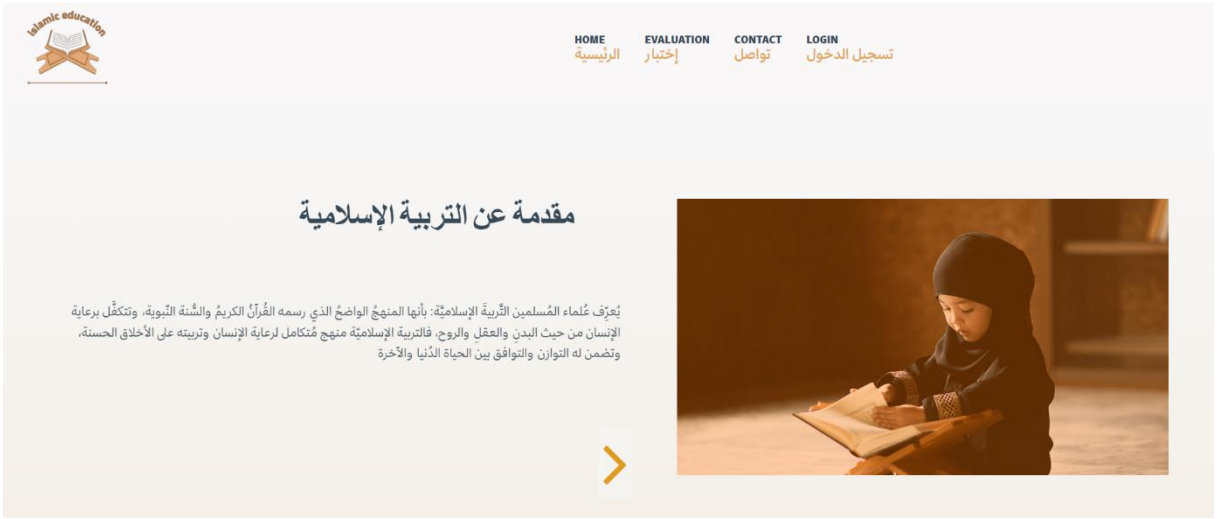
Pour construire l'application dans Docker, nous utilisons la commande suivante :

docker build --tag frontend.

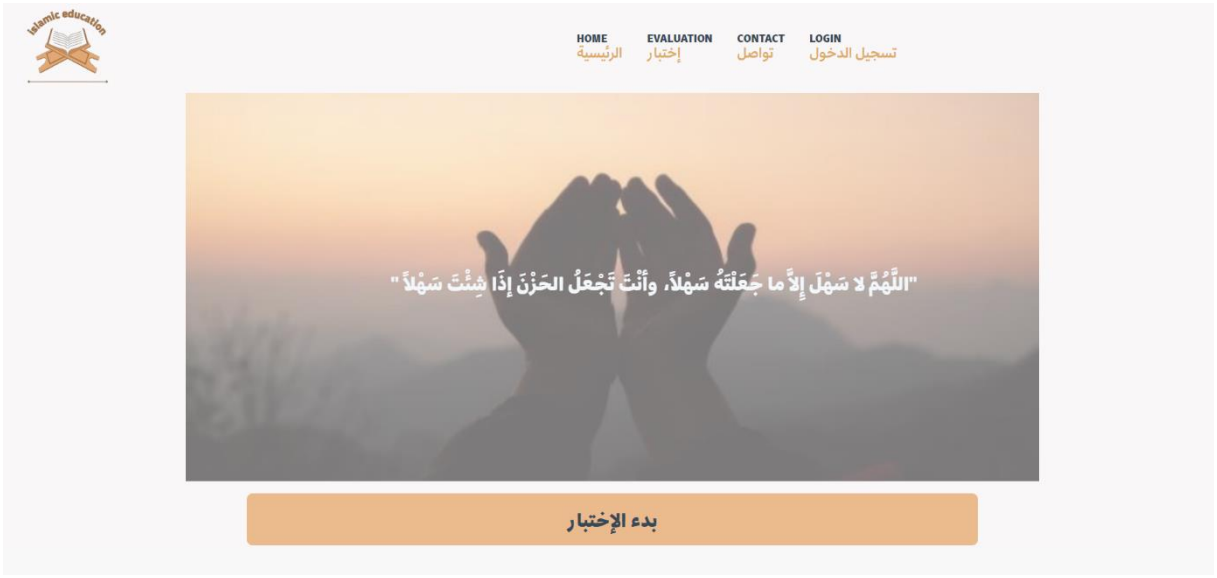
Pour l'exécuter, nous utilisons : **docker run -d -p 4200:80 --name frontend frontend**

→ Démonstration de l'application





L'évaluation commence quand l'élève clique sur le botton « بدئ الإختبار »



Quand il termine l’évaluation, le tableau de résultats sera afficher :

تقييم الإختبار

رقم السؤال	السؤال	الجواب	المعدل
1	ما أول هدية أهديت إلى الرسول صلى الله عليه وسلم بالمدينة ؟	هديت الرسول قصعة خبز ولبن وسمن	2
2	لماذا سمّي العام العاشر من البعثة عام الحزن؟	في عام الحزن، فقد النبي صلى الله عليه وسلم زوجته خديجة وعمه أبو طالب، مما ألقى بظلال الحزن على حياته	2
3	من أول من آمنت بالنبي من النساء؟ و من أول من آمن من الرجال	السيدة خديجة -رضي الله عنها- أول من آمن بالنبيّ -صلى الله عليه وسلم و أول من أسلم وآمن من الرّجال أبو بكر الصِّديق	2
4	ما هي أوّل غزوة خاضها المسلمون؟ وفي أي عام وقعت؟		0
5	ما هي السُّورة التي تعدُّ أمُّ الكتاب؟		0
6	من هو الصحابي الذي لقب بأسد الله ورسوله؟		0
7	ما هي مهنة النَّبيّ -صلّى الله عليه وسلّم- قبل البعثة؟	التِّجارة	1
8	كم كان عمر النَّبيُّ -صلّى الله وعليه وسلّم- عند وفاته، وفي أيّ عام كانت؟	كان عمره عند الوفاة ثلاث وستّين سنةً	1
9	كم كان عدد الصَّلوات التي فرضها الله- تعالى- على المسلمين ابتداءً؟	عدد الصلوات التي فرضها الله -تعالى- على المسلمين كانت خمسين صلاة	2
10	من هو مؤذّن رسول الله -صلّى الله عليه وسلّم-؟	المؤذّن الذي كان يُكلّف بالأذان لرسول الله -صلّى الله عليه وسلّم- هو بلال بن رباح -رضي الله عنه	2

المعدل النهائي : 12