



The logo for alBaraka features the brand name "alBaraka" in a large, bold, dark grey sans-serif font. To the right of the text is a graphic element consisting of two stylized, upward-pointing arrows. The top arrow is orange and the bottom one is red, both with white outlines.

alBaraka



Auteur	Description
Mouhamed Ibrahim Abdelbeki	Etudiant Esprit Ecole Ingenieur, Stagiaire Al Baraka Bank (Siège Social Al Baraka Bank) - 5 juin -> 4 Aout 2023

Table Des Matières

01

Contexte

- Présentation du Projet.
- Avantages

02

Architecture

- Technologie Utilisée.
- Architecture du Chatbot.

03

Environnement

04

Base De Donnée

- Structure AlBaraka_Chatbot
- Structure q_and_a
- Documments et Réponses

05

Rattachement Documents

- | | |
|---------------|----------------------|
| • Html | • app.js |
| • chatbot.css | • Analyse_chatbot.py |
| • chatbot.js | • q_and_a.py |
| • db.js | • Com_sw.py |

06

Interfaces

- | | |
|-----------------------|-------------------|
| • Site Web | • Com Software |
| • Chatbot | • Analyse Chatbot |
| • Interaction Chatbot | • plots |

1 - Contexte

1.1 Présentation du Projet

- Le projet vise à développer et intégrer un chatbot au site Web d'AlBaraka Bank en Tunisie.
- Le chatbot a pour but d'améliorer l'expérience client en fournissant des services de soutien client automatisés et accessibles en tout temps.
- Le chatbot est capable de comprendre les demandes des utilisateurs en langage naturel, et de fournir des réponses appropriées en temps réel.
- De plus, le chatbot peut effectuer diverses tâches, telles que fournir des informations de compte, aider à effectuer des transactions, et répondre aux questions courantes sur les produits et services de la banque.

1.2 Avantages

L'implémentation de ce chatbot présente plusieurs avantages pour AlBaraka Bank :

- Amélioration de l'efficacité du service à la clientèle : Le chatbot peut répondre aux questions des clients 24/7, ce qui réduit les temps d'attente et améliore l'expérience globale des clients.
- Réduction des coûts opérationnels : En automatisant les tâches de support client courantes, le chatbot permet à la banque d'économiser sur les coûts de personnel.
- Analyse des interactions client : Le chatbot peut collecter et analyser les données des interactions client, fournissant des informations précieuses pour améliorer les produits et services de la banque.
- Extensibilité : Le chatbot peut être facilement mis à jour pour inclure de nouvelles fonctionnalités à mesure que les besoins de la banque évoluent.



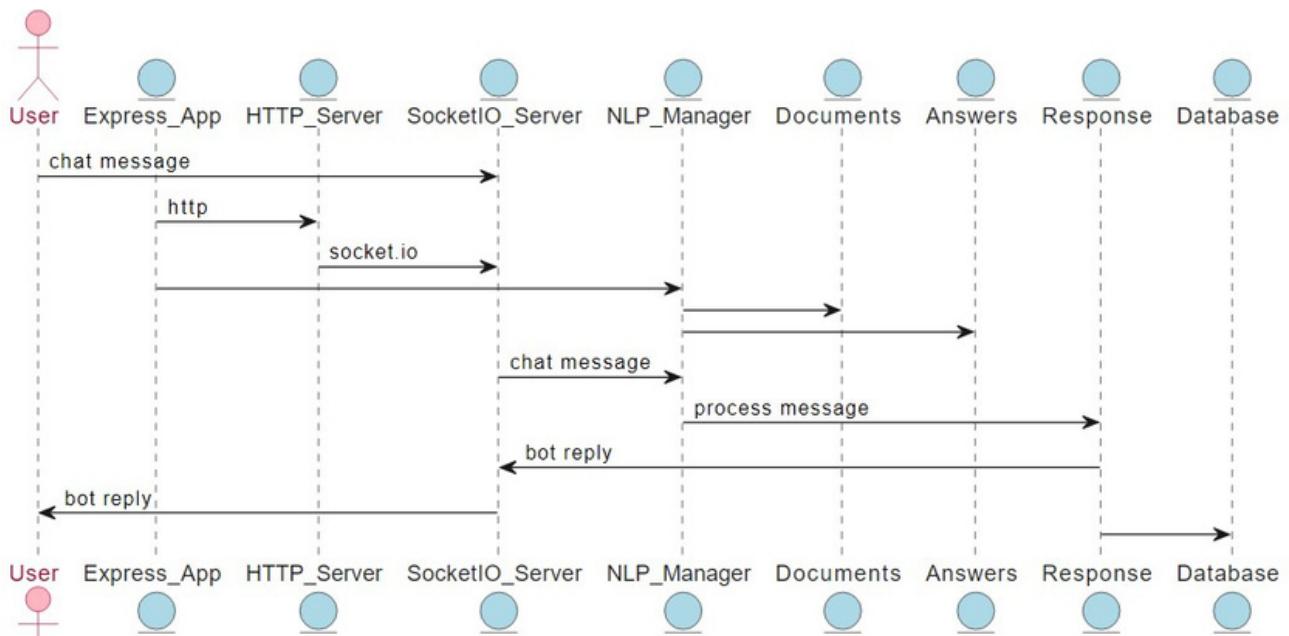
2 - Architecture

2.1 Technologie Utilisée

- Node.js : Environnement d'exécution JavaScript côté serveur qui permet le développement d'applications réseau hautement performantes. Il est non bloquant, ce qui signifie qu'il peut gérer simultanément un grand nombre de requêtes sans ralentir.
- Express.js : Un framework pour Node.js qui facilite la création d'applications web. Express.js simplifie l'écriture du code serveur et offre une structure pour organiser votre application. Il fournit également un ensemble robuste de fonctionnalités pour les applications web et mobiles.
- Socket.IO : Une bibliothèque JavaScript pour les applications web en temps réel. Elle permet des communications bidirectionnelles entre les navigateurs web et les serveurs. Avec Socket.IO, vous pouvez envoyer des données en temps réel à un client et recevoir des données en temps réel d'un client.
- Node-nlp : Une bibliothèque de traitement du langage naturel pour Node.js. Elle permet aux machines de comprendre le langage humain, ce qui est essentiel pour construire des chatbots. Node-nlp offre des fonctionnalités telles que le stemming, la lemmatisation, le POS tagging, le nom de l'entité recognition, et plus encore.
- HTTP : C'est un protocole de communication client-serveur qui permet d'échanger des données sur le web. Il est utilisé dans votre projet pour créer un serveur HTTP qui peut accepter et répondre aux requêtes HTTP.



2.2 Diagramme du Chatbot



Dans ce diagramme :

1. L'Utilisateur (en rose) envoie un 'message de chat' au Serveur SocketIO.
2. L'Application Express communique avec le Serveur HTTP en utilisant le protocole HTTP.
3. Le Serveur HTTP communique avec le Serveur SocketIO en utilisant socket.io.
4. L'Application Express communique également avec le Gestionnaire NLP.
5. Le Gestionnaire NLP interagit avec divers Documents et Réponses.
6. Le Serveur SocketIO transmet le 'message de chat' de l'Utilisateur au Gestionnaire NLP.
7. Le Gestionnaire NLP traite le message et génère une Réponse.
8. La Réponse est renvoyée au Serveur SocketIO sous forme d'événement 'réponse du bot'.
9. Le Serveur SocketIO renvoie la 'réponse du bot' à l'Utilisateur.
10. La Réponse est également sauvegardée dans la Base de Données.



3 - Environement

- Le projet de chatbot a été développé localement en utilisant les ressources du site Web d'AlBaraka téléchargées à l'aide de l'extension Chrome "Save All Resources". Ces ressources permettent de travailler sur une copie du site Web d'AlBaraka en local, sans perturber le site Web en production.
- Le chatbot est intégré à la page d'accueil du site Web par l'ajout de plusieurs éléments HTML et l'inclusion de fichiers CSS et JavaScript.

4 - Base de Donnée

4.1 AlBaraka Chatbot DataBase

- La base de données **AlBaraka_Chatbot.db** et la table **AlBaraka_chatbot** ont été créées dans le but de stocker et de gérer les informations relatives aux interactions entre les utilisateurs et le chatbot d'Al Baraka.
- Chaque interaction est enregistrée comme un enregistrement unique dans la table **AlBaraka_chatbot**. Les champs tels que **message**, **response**, **success**, **language**, et **documentType** permettent de stocker des informations détaillées sur chaque interaction, notamment le message de l'utilisateur, la réponse du chatbot, si la réponse a été réussie ou non, la langue utilisée et le type de document concerné.
- Cela permet à Al Baraka d'analyser les performances du chatbot, d'identifier les domaines dans lesquels le chatbot réussit ou échoue, et d'utiliser ces informations pour améliorer le service. Par exemple, le champ **success** permet de voir à quelle fréquence le chatbot est capable de fournir une réponse réussie, tandis que le champ **documentType** donne une idée des types de documents que les utilisateurs demandent le plus souvent.



- En outre, l'interface utilisateur créée avec le script Python permet à Al Baraka de visualiser les données en temps réel, de voir les interactions à mesure qu'elles se produisent et d'obtenir un aperçu instantané des performances du chatbot.

Structure de la table AlBaraka_chatbot :

Colonne	Type	Description
id	INTEGER	Identifiant unique pour chaque interaction entre l'utilisateur et le chatbot.
message	TEXT	Le message original que l'utilisateur a envoyé au chatbot.
response	TEXT	La réponse que le chatbot a donnée à l'utilisateur.
success	INTEGER	Un indicateur de si la réponse du chatbot était correcte ou non (1 pour succès, 0 pour échec).
language	TEXT	La langue dans laquelle l'utilisateur a interagi avec le chatbot.
documentType	TEXT	Le type de document que l'utilisateur demandait ou sur lequel la conversation portait.



4.2 q_and_a DataBase

La table q_and_a stocke les questions et réponses du chatbot, avec des colonnes pour l'ID unique, le sujet, la question et la réponse. Elle est cruciale pour stocker et récupérer les informations nécessaires pour répondre aux questions des utilisateurs.

Structure de la table q_and_a:

Colonne	Type	Description
id	INTEGER	Clé primaire, identifiant unique de la question
sujet	TEXT	Sujet de la question
question	TEXT	Texte de la question posée par l'utilisateur
reponse	TEXT	Texte de la réponse fournie par le chatbot



4.3 Fichiers Documents et Réponses

Les fichiers de documents et de réponses ont été créés pour le chatbot pour plusieurs raisons :

- **Formation du modèle de langue :** Les documents sont utilisés pour former le modèle de langage du chatbot. Ils représentent différents énoncés que les utilisateurs pourraient utiliser lorsqu'ils interagissent avec le chatbot. Par exemple, pour l'intention de "salutation", les documents pourraient comprendre des phrases comme "Bonjour", "Salut", "Comment ça va ?", etc. Ces documents aident le chatbot à comprendre le contexte et à identifier l'intention derrière l'énoncé de l'utilisateur.
- **Réponses pré-configurées :** Les réponses, d'autre part, sont des réponses pré-configurées que le chatbot peut donner en fonction de l'intention identifiée. Par exemple, si l'intention identifiée est "salutation", alors le chatbot pourrait répondre avec "Bonjour, comment puis-je vous aider aujourd'hui ?". Ces réponses permettent au chatbot de donner des réponses cohérentes et pertinentes.
- **Amélioration continue :** En enregistrant les interactions entre l'utilisateur et le chatbot dans une base de données, il est possible d'analyser plus tard ces données pour améliorer le chatbot. Par exemple, si certaines réponses ne sont pas correctes, elles peuvent être identifiées et corrigées. De plus, de nouveaux documents et réponses peuvent être ajoutés au fil du temps pour permettre au chatbot de gérer de nouvelles intentions ou de nouveaux énoncés.
- **Personnalisation du chatbot :** Avoir des fichiers de documents et de réponses distincts permet également de personnaliser facilement le chatbot pour différentes langues ou domaines. Par exemple, si vous voulez créer un chatbot en anglais, vous pouvez simplement créer des fichiers de documents et de réponses en anglais.



4.4 Documents

Document	Description
Autofinancement_Voiture4CV	Document contenant des énoncés ou des phrases qu'un utilisateur pourrait utiliser lorsqu'il s'informe sur le financement d'une voiture 4 CV. Ces phrases pourraient inclure des questions sur les conditions de financement, les taux d'intérêt, les échéances de paiement, etc.
A_propos_AlBaraka	Document contenant des énoncés que les utilisateurs pourraient utiliser lorsqu'ils demandent des informations générales sur AlBaraka. Cela peut inclure des questions sur l'histoire de la banque, les services qu'elle offre, sa mission et sa vision, etc.
Carte_Allocation_Touristique	Document contenant des énoncés que les utilisateurs pourraient utiliser lorsqu'ils s'interrogent sur la carte d'allocation touristique. Cela peut inclure des questions sur la façon d'obtenir la carte, ses avantages, les restrictions, etc.
demande_pret_automobile	Document contenant des énoncés que les utilisateurs pourraient utiliser lorsqu'ils demandent un prêt automobile. Cela peut inclure des questions sur les taux d'intérêt, les conditions de prêt, la documentation nécessaire, le processus de demande, etc.



demande_rdvz_conseillerFinancier

Document contenant des énoncés que les utilisateurs pourraient utiliser lorsqu'ils demandent un rendez-vous avec un conseiller financier. Cela peut inclure des demandes pour fixer un rendez-vous, connaître la disponibilité du conseiller, préparer un ordre du jour pour la réunion, etc.

horaires_ouverture

Document contenant des énoncés que les utilisateurs pourraient utiliser lorsqu'ils demandent les heures d'ouverture de la banque. Cela peut inclure des questions sur les heures d'ouverture et de fermeture en semaine, pendant le week-end, pendant les vacances, etc.

mood

Document contenant des énoncés qui expriment l'humeur ou l'émotion de l'utilisateur. Il sert à entraîner le chatbot pour détecter et répondre de manière appropriée à l'état émotionnel de l'utilisateur, en fournissant par exemple du soutien en cas de frustration, de colère, de joie, etc.

offre_etudiant

Document contenant des énoncés que les utilisateurs pourraient utiliser lorsqu'ils s'informent sur les offres destinées aux étudiants. Cela peut inclure des questions sur les comptes d'épargne pour étudiants, les prêts étudiants, les cartes de crédit pour étudiants, etc.

ouverture_compte

Document contenant des énoncés que les utilisateurs pourraient utiliser lorsqu'ils souhaitent ouvrir un compte. Cela peut inclure des questions sur les types de comptes disponibles, la documentation nécessaire, le processus d'ouverture de compte, etc.



salutation

Document contenant des énoncés utilisés pour saluer. Cela inclut différents types de salutations comme "Bonjour", "Bonsoir", etc. Il permet au chatbot de comprendre et de répondre correctement à une salutation de l'utilisateur.

solde_compte

Document contenant des énoncés que les utilisateurs pourraient utiliser lorsqu'ils demandent des informations sur le solde de leur compte. Cela peut inclure des demandes de mise à jour du solde, des questions sur les transactions récentes, etc.

tx_change

Document contenant des énoncés que les utilisateurs pourraient utiliser lorsqu'ils demandent des informations sur les taux de change. Cela peut inclure des questions sur les taux de change actuels, la conversion de devises, etc.

whomi

Document contenant des énoncés demandant des informations sur le chatbot, comme "Qui es-tu ?", "Que peux-tu faire ?", etc. Ces phrases aident le chatbot à se présenter et à expliquer ses fonctionnalités à l'utilisateur.

mdp_oublie_bq_online

Document contenant des énoncés que les utilisateurs pourraient utiliser lorsqu'ils ont oublié leur mot de passe pour accéder à leur compte en ligne. Cela peut inclure des demandes d'aide pour réinitialiser le mot de passe, des questions sur la sécurité du compte, etc.



4.5 Réponses

Document	Description
Autofinancement_Voiture4CV	Document contenant les réponses que le chatbot peut donner lorsqu'il s'agit de financer une voiture 4 CV. Ces réponses peuvent comprendre des informations sur les conditions de financement, les taux d'intérêt, les échéances de paiement, etc.
A_propos_AlBaraka	Document contenant les réponses que le chatbot peut donner lorsqu'un utilisateur demande des informations générales sur AlBaraka. Les réponses peuvent inclure des informations sur l'histoire de la banque, les services qu'elle offre, sa mission et sa vision, etc.
Carte_Allocation_Touristique	Document contenant les réponses que le chatbot peut donner lorsqu'un utilisateur demande des informations sur la carte d'allocation touristique. Les réponses peuvent inclure des informations sur comment obtenir la carte, ses avantages, les restrictions, etc.
demande_pret_automobile	Document contenant les réponses que le chatbot peut donner lorsqu'un utilisateur demande un prêt automobile. Les réponses peuvent inclure des informations sur les taux d'intérêt, les conditions de prêt, la documentation nécessaire, le processus de demande, etc.



demande_rdvz_conseillerFinancier

Document contenant les réponses que le chatbot peut donner lorsqu'un utilisateur demande un rendez-vous avec un conseiller financier. Les réponses peuvent inclure des informations sur la disponibilité du conseiller, comment fixer un rendez-vous, préparer un ordre du jour, etc.

horaires_ouverture

Document contenant les réponses que le chatbot peut donner lorsqu'un utilisateur demande les heures d'ouverture de la banque. Les réponses peuvent inclure des informations sur les heures d'ouverture et de fermeture en semaine, pendant le week-end, pendant les vacances, etc.

mood

Document contenant les réponses que le chatbot peut donner en fonction de l'humeur ou de l'émotion de l'utilisateur. Le chatbot peut offrir du soutien en cas de frustration, d'excitation, de colère, de joie, etc.

offre_etudiant

Document contenant les réponses que le chatbot peut donner lorsqu'un utilisateur demande des informations sur les offres pour étudiants. Les réponses peuvent inclure des informations sur les comptes d'épargne pour étudiants, les prêts étudiants, les cartes de crédit, etc.

ouverture_compte

Document contenant les réponses que le chatbot peut donner lorsqu'un utilisateur souhaite ouvrir un compte. Les réponses peuvent inclure des informations sur les types de comptes disponibles, la documentation nécessaire, le processus d'ouverture de compte, etc.



salutation

Document contenant les réponses que le chatbot peut donner lorsqu'il reçoit une salutation. Cela comprend différentes réponses à des salutations comme "Bonjour", "Bonsoir", etc.

solde_compte

Document contenant les réponses que le chatbot peut donner lorsqu'un utilisateur demande des informations sur le solde de son compte. Les réponses peuvent inclure des informations sur la mise à jour du solde, les transactions récentes, etc.

tx_change

Document contenant les réponses que le chatbot peut donner lorsqu'un utilisateur demande des informations sur les taux de change. Les réponses peuvent inclure des informations sur les taux de change actuels, comment convertir les devises, etc.

whomi

Document contenant les réponses que le chatbot peut donner lorsqu'un utilisateur demande des informations sur le chatbot. Les réponses peuvent inclure des informations sur les fonctionnalités du chatbot, ce qu'il peut faire, comment il peut aider, etc.

mdp_oublie_bq_online

Document contenant les réponses que le chatbot peut donner lorsqu'un utilisateur a oublié son mot de passe pour accéder à son compte en ligne. Les réponses peuvent inclure des instructions sur comment réinitialiser le mot de passe, des informations sur la sécurité du compte, etc.



5 - Rattachement Documents

5.1 Configuration de l'interface utilisateur du chatbot

- Cette section décrit l'installation de l'interface utilisateur du chatbot, y compris l'icône du chatbot, la fenêtre de chat, l'indicateur de frappe et la zone d'entrée de texte pour l'utilisateur. Ces éléments visuels sont essentiels pour permettre une interaction fluide entre l'utilisateur et le chatbot.

```
<!-- Début de l'icône du chatbot -->
<div id="chatbot-icon">
    <!-- Image de l'icône du chatbot -->
    
    <!-- Texte d'aide qui s'affiche à côté de l'icône du chatbot -->
    <div id="helper-text"></div>
</div>
<!-- Fin de l'icône du chatbot -->

<!-- Début de la fenêtre du chatbot, qui est initialement cachée -->
<div id="chatbot" style="display: none;">
    <!-- Zone où les messages du chat seront ajoutés dynamiquement -->
    <div id="chat-messages"></div>

    <!-- Indicateur de frappe du chatbot, qui est initialement caché -->
    <div id="typing-indicator" style="display: none;">...</div>

    <!-- Zone d'entrée du message, qui est initialement cachée et désactivée -->
    <div class="message-input" style="display: none;">
        <input id="input-box" type="text" disabled>
    </div>
</div>
<!-- Fin de la fenêtre du chatbot -->
```

5.2 Intégration du fichier CSS du chatbot

- Cette section concerne l'intégration du fichier CSS associé au chatbot. Le fichier CSS permet de définir et de contrôler le style de l'interface utilisateur du chatbot, notamment l'aspect visuel des éléments tels que l'icône du chatbot, la fenêtre de chat, l'indicateur de frappe et la zone d'entrée du message.

```
<link rel="stylesheet" href="core/modules/system/css/chatbot/chatbot.css">
```

5.3 Incorporation du fichier JavaScript du chatbot

- Cette partie fait référence à l'inclusion du fichier JavaScript du chatbot. Le fichier JavaScript est crucial pour la gestion de la dynamique et de l'interactivité du chatbot. Il gère notamment l'envoi et la réception de messages, la manipulation des éléments de la fenêtre de chat, ainsi que l'interaction entre le client et le serveur.

```
<script src="/themes/bootstrap/js/chatbot/chatbot.js"></script>
```

5.4 Intégration de Socket.io

- Cette ligne du code inclut le fichier JavaScript de Socket.io, une bibliothèque JavaScript pour les applications web en temps réel. Socket.io permet une communication bidirectionnelle en temps réel entre les navigateurs web et les serveurs. Dans le contexte du chatbot, Socket.io est utilisé pour faciliter les échanges en temps réel de messages entre le chatbot et l'utilisateur.

```
<script src="/socket.io/socket.io.js"></script>
```



5.5 Styles CSS pour l'interface du chatbot

- Cette section du code définit les styles CSS pour l'interface utilisateur du chatbot. Elle comprend le formatage du texte global, l'apparence de l'icône du chatbot, le style de la fenêtre de chat, l'aspect de la zone de texte d'entrée, l'indicateur de frappe du chatbot, ainsi que les différents styles pour les messages de l'utilisateur et du bot.
- En outre, elle comporte des définitions pour des animations comme l'animation de rebondissement de l'icône du chatbot, le clignotement du texte d'aide, etc. Les styles sont définis de manière à rendre l'interaction utilisateur-chatbot aussi agréable et conviviale que possible. Les polices, les couleurs, les espacements et les bordures ont été soigneusement sélectionnés pour offrir une expérience utilisateur harmonieuse et intuitive.

```
/* Définition des polices globales et du style du texte */
body {
    font-family: 'Orbitron', sans-serif;
}

#chatbot,
#helper-text {
    font-family: 'Orbitron', sans-serif; /* Définition de la police de caractères */
    font-size: 12.5px; /* Définition de la taille de la police */
    line-height: 1.5; /* Définition de l'interligne */
    letter-spacing: 1px; /* Espacement entre les lettres */
    color: #333; /* Couleur du texte */
}

/* Style de l'icône du chatbot */
#chatbot-icon {
    position: fixed; /* Position fixée par rapport à la fenêtre du navigateur */
    bottom: 20px; /* Position depuis le bas */
    right: 20px; /* Position depuis la droite */
    z-index: 100; /* Superposition de couches */
    cursor: pointer; /* Curseur de la souris comme un pointeur */
    width: 100px; /* Largeur de l'icône */
    height: 80px; /* Hauteur de l'icône */
    animation: bounce 2s infinite; /* Animation de rebondissement */
}

/* Style de la fenêtre de chat du chatbot */
#chatbot {
    position: fixed; /* Position fixée par rapport à la fenêtre du navigateur */
    bottom: 120px; /* Position depuis le bas */
    right: 20px; /* Position depuis la droite */
    z-index: 101; /* Superposition de couches */
    width: 300px; /* Largeur de la fenêtre de chat */
}
```

```

/* Style de la zone de texte d'entrée */
#input-box {
    position: relative; /* Position relative aux éléments environnants */
    bottom: 0;           /* Position depuis le bas */
    right: 20px;         /* Position depuis la droite */
    width: 280px;        /* Largeur de la zone de texte */
    height: 30px;        /* Hauteur de la zone de texte */
    border-radius: 5px;   /* Rayon des coins arrondis */
    border: 1px solid #ddd; /* Bordure */
    padding: 5px;         /* Espacement intérieur */
}

/* Indicateur de saisie (s'affiche lorsque le bot est "en train d'écrire") */
#typing-indicator {
    position: relative; /* Position relative aux éléments environnants */
    padding: 10px;       /* Espacement intérieur */
    background: #ddd;    /* Couleur de fond */
    border-radius: 10px;  /* Rayon des coins arrondis */
    width: 50px;         /* Largeur de l'indicateur */
    text-align: center;  /* Centrer le texte à l'intérieur */
    margin: auto;        /* Centrer l'élément dans son conteneur */
}

/* Styles de la zone de texte d'entrée */
#input-box::placeholder {
    color: #fff;          /* Couleur du texte indicatif */
    opacity: 0.8;          /* Transparence du texte indicatif */
}

/* Styles pour Internet Explorer 10-11 */
#input-box:-ms-input-placeholder {
    color: #fff;          /* Couleur du texte indicatif */
}

/* Animation de rebondissement de l'icône du chatbot */
@keyframes bounce {
    0%, 100% {transform: translateY(0); } /* Aucune translation verticale au début et à la fin */

    50% {transform: translateY(-20px);} /* Translation verticale de -20px à mi-chemin */
}

/* Style de l'icône du chatbot lorsqu'il cesse de rebondir */
#chatbot-icon.stop-bouncing {
    -webkit-animation: none; /* Arrêt de l'animation pour les navigateurs WebKit (Chrome, Safari) */
    animation: none;         /* Arrêt de l'animation pour tous les navigateurs */
}

#chatbot-icon.stop-bouncing::after {
    opacity: 0;
}

/* Style du texte d'aide */
#helper-text {
    position: absolute; /* Position absolue par rapport à son conteneur parent */
    top: -90px;          /* Position depuis le haut */
    right: 30px;          /* Position depuis la droite */
    width: 150px;         /* Largeur du texte d'aide */
    padding: 10px;         /* Espacement intérieur */
    font-size: 14px;       /* Taille de la police */
    font-weight: bold;     /* Graisse de la police */
    color: #000000;        /* Couleur du texte */
    text-align: center;    /* Centrer le texte à l'intérieur */
    background: #ffffff;   /* Couleur de fond */
    border-radius: 10px;    /* Rayon des coins arrondis */
    box-shadow: 0px 0px 10px rgba(0, 0, 0, 0.15); /* Ombre portée */
}

```

```

/* Animation de clignotement du texte */
@keyframes blinkText {
    0% {opacity: 0;}          /* Complètement transparent au début */
    50% {opacity: 1;}          /* Complètement opaque à mi-chemin */
    100% {opacity: 0;}         /* Complètement transparent à la fin */
}

/* Style de l'image de l'icône du chatbot */
#chatbot-icon img {
    width: 100%;             /* Largeur totale */
    height: auto;             /* Hauteur auto pour maintenir les proportions */
}

/* Style de la zone de saisie de texte pour l'utilisateur */
.message-input {
    position: relative;      /* Position relative aux éléments environnants */
    bottom: 0;                /* Position depuis le bas de l'élément parent */
    right: 15px;              /* Position depuis la droite de l'élément parent */
    z-index: 102;              /* Niveau de superposition sur l'axe z (au-dessus des autres éléments) */
    width: 280px;              /* Largeur de la zone de saisie */
    height: 30px;              /* Hauteur de la zone de saisie */
    border: 2px solid black;   /* Bordure solide de 2px de couleur noire */
    border-radius: 15px;        /* Rayon des coins arrondis */
    padding: 5px;              /* Espacement intérieur */
    color: black;              /* Couleur du texte entré par l'utilisateur */
    font-weight: bold;         /* Graisse du texte entré par l'utilisateur */
}

/* Styles des messages de l'utilisateur et du bot */
.message {
    max-width: 2000px;        /* Limiter la largeur des messages */
    margin-bottom: 3px;         /* Marge en dessous de chaque message */
}

/* Styles spécifiques aux messages de l'utilisateur */
.user-message {
    background-color: white;   /* Couleur de fond */
    color: #333;               /* Couleur du texte */
    order: 1;                  /* Ordre dans le flux flex */
    align-self: flex-end;       /* Alignement à droite */
    border-radius: 18px;        /* Rayon des coins arrondis */
    margin-left: 10px;          /* Marge à gauche */
    padding: 10px;              /* Espacement intérieur */
    border: 2px solid #333;     /* Bordure */
}

/* Styles spécifiques aux messages du bot */
.bot-message {
    background-color: #ddd;    /* Couleur de fond */
    color: #333;               /* Couleur du texte */
    order: 2;                  /* Ordre dans le flux flex */
    align-self: flex-start;     /* Alignement à gauche */
    border-radius: 18px;        /* Rayon des coins arrondis */
    margin-right: 10px;         /* Marge à droite */
    padding: 10px;              /* Espacement intérieur */
    border: 2px solid #333;     /* Bordure */
}

/* Style du texte à l'intérieur de chaque message */
.message-text {
    padding: 5px;              /* Espacement intérieur */
    border-radius: 5px;         /* Rayon des coins arrondis */
}

```



5.6 Initialisation et Gestion du Chatbot

Le code JavaScript se charge de l'initialisation et de la gestion d'un chatbot dans un navigateur. Il couvre :

- Récupération des éléments HTML requis.
- Affichage interactif du texte d'aide.
- Gestion des interactions utilisateur avec l'icône du chatbot.
- Prise en charge de l'envoi des messages de l'utilisateur.
- Affichage des messages du chatbot et de l'utilisateur.
- Communication en temps réel avec le serveur via Socket.io.
- Fonction pour redémarrer le chat.

```
window.onload = function () { // Quand la page est chargée, on execute la fonction suivante

    var chatbot = document.getElementById('chatbot'); // On récupère l'élément HTML ayant pour id "chatbot"
    var chatbotIcon = document.getElementById('chatbot-icon'); // On récupère l'icône du chatbot
    var inputBox = document.getElementById('input-box'); // On récupère la boite d'entrée du message
    var messageContainer = document.getElementById('chat-messages'); // On récupère le conteneur des messages du chat
    var helperTextElement = document.getElementById('helper-text'); // On récupère l'élément du texte d'aide

    var isFirstClick = true; // Indicateur du premier clic, initialement fixé à vrai
    var helperMessages = ["Hey! Besoin d'aide?", "Cliquez ici pour vous aider!", "Je suis sur que je peux vous aider!", "Hey! je suis là pour vous!", "Je suis à votre service!"]; // Messages d'aide
    var helperText = helperMessages[Math.floor(Math.random() * helperMessages.length)]; // Choix aléatoire d'un message d'aide à afficher
    var textIndex = 0; // Index du texte, initialement fixé à zéro
    var typingSpeed = 70; // Vitesse de frappe du texte, définie en millisecondes
    var helperTextInterval; // Variable qui sera utilisée pour arrêter l'intervalle de frappe du texte d'aide


    function typeHelperText() { // Fonction qui simule la frappe du texte d'aide

        if (textIndex < helperText.length) { // Si l'index du texte est inférieur à la longueur du texte d'aide
            helperTextElement.textContent = helperText.substring(0, textIndex + 1); // On affiche le texte d'aide caractère par caractère
            textIndex++; // On incrémente l'index du texte
        } else {
            clearInterval(helperTextInterval); // On arrête l'intervalle de frappe du texte d'aide
            textIndex = 0; // On réinitialise l'index du texte

            var nextHelperText; // Variable pour stocker le prochain texte d'aide
            do {
                nextHelperText = helperMessages[Math.floor(Math.random() * helperMessages.length)]; // On choisit un autre texte d'aide au hasard
            } while (nextHelperText === helperText); // On s'assure que le nouveau texte d'aide n'est pas le même que le précédent
            helperText = nextHelperText; // On met à jour le texte d'aide

            var fadeOutDuration = helperText.length * typingSpeed; // On calcule la durée de l'animation de fade-out basée sur la longueur du texte

            helperTextElement.style.animation = `fadeinout ${fadeOutDuration}ms infinite`; // On applique l'animation de fade-out au texte d'aide

            setTimeout(function () { // On attend une seconde avant de recommencer à taper le texte
                helperTextInterval = setInterval(typeHelperText, typingSpeed); // On recommence à taper le texte d'aide
            }, 1000); // On attend une seconde
        }
    }

}
```



```

function displayHelperText() { // Fonction pour afficher le texte d'aide
    textIndex = 0; // On réinitialise l'index du texte
    helperTextInterval = setInterval(typeHelperText, typingSpeed); // On commence à taper le texte d'aide
}

displayHelperText(); // On affiche le texte d'aide quand la page est chargée

chatbotIcon.addEventListener('click', function () { // Quand on clique sur l'icône du chatbot
    helperTextElement.style.display = 'none'; // On cache le texte d'aide
    clearInterval(helperTextInterval); // On arrête de taper le texte d'aide

    chatbotIcon.classList.add('stop-bouncing'); // On arrête l'animation de rebond de l'icône du chatbot
    inputBox.disabled = false; // On active la boîte d'entrée du message
    inputBox.style.display = 'block'; // on affiche la boîte d'entrée du message
    chatbot.style.display = 'block'; // On affiche le chatbot

    if (isFirstClick) { // Si c'est le premier clic sur l'icône du chatbot
        isFirstClick = false; // On met à jour l'indicateur du premier clic
        displayBotMessage("Bonjour, comment puis-je vous aider aujourd'hui?", 100); // On affiche le premier message du chatbot
    } else {
        restartChat(); // Sinon, on redémarre le chat
    }
});

inputBox.addEventListener('keyup', function (event) { // Quand on relâche une touche dans la boîte d'entrée du message
    if (event.keyCode === 13) { // Si la touche est la touche "Entrée"
        event.preventDefault(); // On empêche l'action par défaut de la touche
        sendMessage(inputBox, inputBox.value); // On envoie le message de l'utilisateur
    }
});

function createNewInput(helpText) { // Fonction pour créer une nouvelle boîte d'entrée du message
    var newInput = document.createElement('input'); // On crée un nouvel élément input
    newInput.type = 'text'; // On définit le type de l'input comme texte
    newInput.className = 'message-input'; // On donne une classe à l'input
    newInput.placeholder = helpText; // On définit le texte d'aide de l'input
    newInput.addEventListener('keyup', function (event) { // Quand on relâche une touche dans l'input
        if (event.keyCode === 13) { // Si la touche est la touche "Entrée"
            sendMessage(newInput, newInput.value); // On envoie le message de l'utilisateur
        }
    });
    messageContainer.appendChild(newInput); // On ajoute l'input au conteneur des messages
}

function displayUserMessage(messageText) { // Fonction pour afficher le message de l'utilisateur
    var userMessageContainer = document.createElement('div'); // On crée un nouvel élément div
    userMessageContainer.className = 'message user-message'; // On donne une classe à l'élément div

    var userMessageText = document.createElement('div'); // On crée un nouvel élément div pour le texte du message
    userMessageText.textContent = messageText; // On donne le texte du message à l'élément div
    userMessageText.className = 'message-text'; // On donne une classe à l'élément div du texte

    userMessageContainer.appendChild(userMessageText); // On ajoute l'élément div du texte à l'élément div du conteneur du message
    messageContainer.appendChild(userMessageContainer); // On ajoute l'élément div du conteneur du message au conteneur des messages
}

```

```

function displayBotMessage(messageText, typingSpeed) { // Fonction pour afficher le message du chatbot
    document.getElementById('typing-indicator').style.display = 'block'; // On affiche l'indicateur de frappe du chatbot

    var botMessageContainer = document.createElement('div'); // On crée un nouvel élément div pour le conteneur du message du chatbot
    botMessageContainer.className = 'message bot-message'; // On donne une classe à l'élément div

    var botMessageText = document.createElement('div'); // On crée un nouvel élément div pour le texte du message du chatbot
    botMessageText.className = 'message-text'; // On donne une classe à l'élément div

    botMessageContainer.appendChild(botMessageText); // On ajoute l'élément div du texte à l'élément div du conteneur du message
    messageContainer.appendChild(botMessageContainer); // On ajoute l'élément div du conteneur du message au conteneur des messages

    var typingIndex = 0; // On initialise l'index de frappe à zéro

    botTypingInterval = setInterval(function () { // On commence à taper le message du chatbot
        botMessageText.textContent = messageText.substring(0, typingIndex + 1); // On affiche le message du chatbot caractère par caractère
        typingIndex++; // On incrémente l'index de frappe

        if (typingIndex >= messageText.length) { // Si l'index de frappe est supérieur ou égal à la longueur du message
            clearInterval(botTypingInterval); // On arrête de taper le message du chatbot

            document.getElementById('typing-indicator').style.display = 'none'; // On cache l'indicateur de frappe du chatbot
            createNewInput('Ecrivez votre message...'); // On crée une nouvelle boîte d'entrée du message
        }
    }, typingSpeed); // On définit la vitesse de frappe du message du chatbot
}

const socket = io('http://localhost:3000'); // On crée une connexion socket avec le serveur

function sendMessage(input, messageText) { // Fonction pour envoyer le message de l'utilisateur
    messageContainer.innerHTML = ''; // On vide le conteneur des messages

    if (messageText.trim().length > 0) { // Si le message de l'utilisateur n'est pas vide
        displayUserMessage(messageText); // On affiche le message de l'utilisateur
        socket.emit('chat message', messageText); // On envoie le message de l'utilisateur au serveur
    } else {
        createNewInput('Ecrivez votre message...'); // On crée une nouvelle boîte d'entrée du message
    }

    input.disabled = true; // On désactive la boîte d'entrée du message après l'envoi du message
}

socket.on('bot reply', function (botResponse) { // Quand on reçoit une réponse du bot depuis le serveur
    displayBotMessage(botResponse, 70); // On affiche le message du bot
});

function restartChat() { // Fonction pour redémarrer le chat
    messageContainer.innerHTML = ''; // On vide le conteneur des messages
    displayBotMessage("Bonjour, comment puis-je vous aider aujourd'hui?", 100); // On affiche le premier message du bot
};

}

```

5.7 Mise en place du serveur et de l'entraînement du modèle NLP

5.7.1 Bibliothèques et serveur

- Dans cette partie du code, les bibliothèques nécessaires pour le chatbot sont importées. Ensuite, les documents et les réponses liés à différentes intentions du chatbot sont importés. Le gestionnaire du chatbot est créé avec les langues prédefinies. L'application Express, le serveur HTTP et Socket.io sont initialisés.
- Le serveur est configuré pour servir les fichiers statiques du site web. Les documents et les réponses sont ajoutés au gestionnaire de chatbot pour chaque intention. Le gestionnaire du chatbot est alors formé et le modèle est sauvegardé.
- Une fois que le chatbot a été formé, le serveur est mis à l'écoute des nouvelles connexions. Lorsqu'une connexion est établie, le chatbot est prêt à recevoir et à traiter les messages. Les messages traités sont renvoyés à l'utilisateur et sauvegardés dans la base de données.
- Enfin, le serveur est démarré sur le port 3000 et l'URL du serveur est affichée.

```
// Importer le module Express pour créer une application web
const express = require('express');

// Importer le module HTTP pour créer un serveur HTTP
const http = require('http');

// Importer Socket.IO pour faciliter les communications en temps réel entre le serveur et le client
const socketIo = require('socket.io');

// Importer NlpManager du module node-nlp pour le traitement du langage naturel
const { NlpManager } = require('node-nlp');

// Importer le module de base de données pour stocker les messages et les réponses du chatbot
const db = require('./db.js');
```



5.7.2 Importation des Documents et Réponses

- Cette partie du code est dédiée à l'importation des documents et des réponses correspondantes pour chaque intention identifiée du chatbot. Les documents et réponses importés sont stockés dans des constantes pour être utilisés ultérieurement lors de la formation du gestionnaire de traitement du langage naturel (NLP). C'est un processus essentiel qui permet au chatbot d'apprendre à reconnaître différentes intentions à partir des phrases saisies par les utilisateurs et à générer des réponses appropriées.

```
// Importer les documents et les réponses en français pour le chatbot
const FRAutofinancement_Voiture4CVDocuments = require('./documents/fr/Autofinancement_Voiture4CV');// pour l'intention "Autofinancement_Voiture4CV"
const FRAutofinancement_Voiture4CVAnswers = require('./answers/fr/Autofinancement_Voiture4CV');// pour l'intention "Autofinancement_Voiture4CV"

const FRA_propos_AlBarakaDocuments = require('./documents/fr/A_propos_AlBaraka');// pour l'intention "A_propos_AlBaraka"
const FRA_propos_AlBarakaAnswers = require('./answers/fr/A_propos_AlBaraka');// pour l'intention "A_propos_AlBaraka"

const FRCarte_Allocation_TouristiqueDocuments = require('./documents/fr/Carte_Allocation_Touristique');// pour l'intention "Carte_Allocation_Touristique"
const FRCarte_Allocation_TouristiqueAnswers = require('./answers/fr/Carte_Allocation_Touristique');// pour l'intention "Carte_Allocation_Touristique"

const FRdemande_pret_automobileDocuments = require('./documents/fr/demande_pret_automobile');// pour l'intention "demande_pret_automobile"
const FRdemande_pret_automobileAnswers = require('./answers/fr/demande_pret_automobile');// pour l'intention "demande_pret_automobile"

const FRdemande_rdvz_conseillerFinancierDocuments = require('./documents/fr/demande_rdvz_conseillerFinancier');// pour l'intention "demande_rdvz_conseillerFinancier"
const FRdemande_rdvz_conseillerFinancierAnswers = require('./answers/fr/demande_rdvz_conseillerFinancier');// pour l'intention "demande_rdvz_conseillerFinancier"

const FRhoraires_ouvertureDocuments = require('./documents/fr/horaires_ouverture');// pour l'intention "horaires_ouverture"
const FRhoraires_ouvertureAnswers = require('./answers/fr/horaires_ouverture');// pour l'intention "horaires_ouverture"

const FRmdp_oublie_bq_onlineDocuments = require('./documents/fr/mdp_oublie_bq_online');// pour l'intention "mdp_oublie_bq_online"
const FRmdp_oublie_bq_onlineAnswers = require('./answers/fr/mdp_oublie_bq_online');// pour l'intention "mdp_oublie_bq_online"

const FRmoodDocuments = require('./documents/fr/mood');// pour l'intention "mood"
const FRmoodAnswers = require('./answers/fr/mood');// pour l'intention "mood"

const FROffre_etudiantDocuments = require('./documents/fr/offre_etudiant');// pour l'intention "offre_etudiant"
const FROffre_etudiantAnswers = require('./answers/fr/offre_etudiant');// pour l'intention "offre_etudiant"

const FRouverture_compteDocuments = require('./documents/fr/ouverture_compte');// pour l'intention "ouverture_compte"
const FRouverture_compteAnswers = require('./answers/fr/ouverture_compte');// pour l'intention "ouverture_compte"

const FRsalutationDocuments = require('./documents/fr/salutation');// pour l'intention "salutation"
const FRsalutationAnswers = require('./answers/fr/salutation');// pour l'intention "salutation"

const FRsolde_compteDocuments = require('./documents/fr/solde_compte');// pour l'intention "solde_compte"
const FRsolde_compteAnswers = require('./answers/fr/solde_compte');// pour l'intention "solde_compte"

const FRtx_changeDocuments = require('./documents/fr/tx_change');// pour l'intention "tx_change"
const FRtx_changeAnswers = require('./answers/fr/tx_change');// pour l'intention "tx_change"

const FRwhomiDocuments = require('./documents/fr/whomi');// pour l'intention "whomi"
const FRwhomiAnswers = require('./answers/fr/whomi');// pour l'intention "whomi"
```

5.7.3 Formation du modèle

- Dans cette section du code, nous importons les documents et réponses associés à diverses intentions pour le chatbot. Les documents contiennent des expressions courantes que l'utilisateur peut dire, liées à une intention spécifique. Les réponses, quant à elles, comprennent les répliques possibles du chatbot lorsque cette intention est reconnue.
- Ces importations permettent de former le modèle de traitement du langage naturel, en associant les phrases des documents et les réponses à leurs intentions respectives grâce aux méthodes addDocument et addAnswer. Cela permet au chatbot de comprendre et de répondre de manière appropriée aux requêtes des utilisateurs.

```
// Créer un nouveau NlpManager pour le traitement du langage naturel en français
const manager = new NlpManager({ languages: ['fr'] });

// Créer une nouvelle application Express
const app = express();

// Créer un nouveau serveur HTTP à partir de l'application Express
const server = http.createServer(app);

// Initialiser Socket.IO sur le serveur HTTP
const io = socketIo(server);

// Définir le dossier des fichiers statiques pour le serveur Express
app.use(express.static('C:/Users/abdel/OneDrive/Bureau/Projects/Al Baraka WS/www.albaraka.com.tn'));


// pour l'intention "Autofinancement_Voiture4CV"
FRAutofinancement_Voiture4CVDocuments.forEach((document) => {manager.addDocument('fr', document.input, 'Autofinancement_Voiture4CV');});
// pour l'intention "Autofinancement_Voiture4CV"
FRAutofinancement_Voiture4CVAnswers.forEach((answer) => {manager.addAnswer('fr', 'Autofinancement_Voiture4CV', answer);});
// pour l'intention "A_propos_AlBaraka"
FRA_propos_AlBarakaDocuments.forEach((document) => {manager.addDocument('fr', document.input, 'A_propos_AlBaraka');});
// pour l'intention "A_propos_AlBaraka"
FRA_propos_AlBarakaAnswers.forEach((answer) => {manager.addAnswer('fr', 'A_propos_AlBaraka', answer);});
// pour l'intention "Carte_Allocation_Touristique"
FRCarte_Allocation_TouristiqueDocuments.forEach((document) => {manager.addDocument('fr', document.input, 'Carte_Allocation_Touristique');});
// pour l'intention "Carte_Allocation_Touristique"
FRCarte_Allocation_TouristiqueAnswers.forEach((answer) => {manager.addAnswer('fr', 'Carte_Allocation_Touristique', answer);});
// pour l'intention "demande_pret_automobile"
FRDemande_pret_automobileDocuments.forEach((document) => {manager.addDocument('fr', document.input, 'demande_pret_automobile');});
// pour l'intention "demande_pret_automobile"
FRDemande_pret_automobileAnswers.forEach((answer) => {manager.addAnswer('fr', 'demande_pret_automobile', answer);});
// pour l'intention "demande_rdvz_conseillerFinancier"
FRDemande_rdvz_conseillerFinancierDocuments.forEach((document) => {manager.addDocument('fr', document.input, 'demande_rdvz_conseillerFinancier');});
// pour l'intention "demande_rdvz_conseillerFinancier"
FRDemande_rdvz_conseillerFinancierAnswers.forEach((answer) => {manager.addAnswer('fr', 'demande_rdvz_conseillerFinancier', answer);});
// pour l'intention "horaires_ouverture"
FRhoraires_ouvertureDocuments.forEach((document) => {manager.addDocument('fr', document.input, 'horaires_ouverture');});
// pour l'intention "horaires_ouverture"
FRhoraires_ouvertureAnswers.forEach((answer) => {manager.addAnswer('fr', 'horaires_ouverture', answer);});
// pour l'intention "mdp_oublie_bq_online"
FRmdp_oublie_bq_onlineDocuments.forEach((document) => {manager.addDocument('fr', document.input, 'mdp_oublie_bq_online');});
// pour l'intention "mdp_oublie_bq_online"
FRmdp_oublie_bq_onlineAnswers.forEach((answer) => {manager.addAnswer('fr', 'mdp_oublie_bq_online', answer);});
// pour l'intention "mood"
FRmoodDocuments.forEach((document) => {manager.addDocument('fr', document.input, 'mood');});
// pour l'intention "mood"
FRmoodAnswers.forEach((answer) => {manager.addAnswer('fr', 'mood', answer);});
```

```

// pour l'intention "offre_etudiant"
FRoffre_etudiantDocuments.forEach((document) => {manager.addDocument('fr', document.input, 'offre_etudiant');});
// pour l'intention "offre_etudiant"
FRoffre_etudiantAnswers.forEach((answer) => {manager.addAnswer('fr', 'offre_etudiant', answer);});
// pour l'intention "ouverture_compte"
FRouverture_compteDocuments.forEach((document) => {manager.addDocument('fr', document.input, 'ouverture_compte');});
// pour l'intention "ouverture_compte"
FRouverture_compteAnswers.forEach((answer) => {manager.addAnswer('fr', 'ouverture_compte', answer);});
// pour l'intention "salutation"
FRsalutationDocuments.forEach((document) => {manager.addDocument('fr', document.input, 'salutation');});
// pour l'intention "salutation"
FRsalutationAnswers.forEach((answer) => {manager.addAnswer('fr', 'salutation', answer);});
// pour l'intention "solde_compte"
FRsolde_compteDocuments.forEach((document) => {manager.addDocument('fr', document.input, 'solde_compte');});
// pour l'intention "solde_compte"
FRsolde_compteAnswers.forEach((answer) => {manager.addAnswer('fr', 'solde_compte', answer);});
// pour l'intention "tx_change"
FRtx_changeDocuments.forEach((document) => {manager.addDocument('fr', document.input, 'tx_change');});
// pour l'intention "tx_change"
FRtx_changeAnswers.forEach((answer) => {manager.addAnswer('fr', 'tx_change', answer);});
// pour l'intention "whomi"
FRwhomiDocuments.forEach((document) => {manager.addDocument('fr', document.input, 'whomi');});
// pour l'intention "whomi"
FRwhomiAnswers.forEach((answer) => {manager.addAnswer('fr', 'whomi', answer);});

```

5.7.3 Entrainement modèle, Traitement et Démarrage serveur

- Dans cette section du code, le gestionnaire NLP est formé sur les documents et réponses fournis, puis le modèle entraîné est sauvegardé. Une fois que le modèle est prêt, le serveur commence à écouter les nouvelles connexions de Socket.IO.
- Lorsqu'un message de chat est reçu du client, il est traité par le modèle NLP. Si le modèle n'est pas capable de trouver une réponse, une réponse par défaut est définie. Le bot envoie ensuite la réponse au client.
- Chaque message, avec la réponse du bot et d'autres informations pertinentes, est sauvegardé dans la base de données.
- Enfin, le serveur HTTP est démarré sur le port 3000, et l'URL du serveur est affichée.

```

// Entrainer le modèle NLP avec les documents et les réponses, puis sauvegarder le modèle
manager.train().then(() => {
    manager.save();

    // Écouter les nouvelles connexions Socket.IO
    io.on('connection', (socket) => {

        // Écouter les messages de chat envoyés par le client
        socket.on('chat message', async (msg) => {
            let success = true;

            // Traiter le message reçu avec le modèle NLP
            let response = await manager.process('fr', msg);

            // Si le modèle NLP n'a pas pu trouver une réponse, définir une réponse par défaut
            if (!response.answer) {
                success = false;
                response.answer = 'Désolé, je n\'ai pas bien compris. Pouvez-vous reformuler votre phrase?';
            }

            // Envoyer la réponse du bot à l'utilisateur
            socket.emit('bot reply', response.answer);

            // Sauvegarder le message, la réponse et d'autres informations dans la base de données
            db.save({
                message: msg,
                response: response.answer,
                success: success,
                language: 'fr',
                documentType: response.intent
            });
        });
    });

    // Démarrer le serveur HTTP sur le port 3000 et afficher l'URL du serveur
    server.listen(3000, () => {
        const serverUrl = 'http://localhost:3000/';
        console.log(`Le serveur est en cours d'exécution sur le port 3000. Voici le lien : ${serverUrl}`);
    });
});

```

5.8 Connexion à la base de données et gestion des interactions

- Cette section du code gère la connexion à la base de données SQLite "AlBaraka_Chatbot.db" et définit également la structure de la table de la base de données "AlBaraka_chatbot". Cette table enregistre des informations telles que le message de l'utilisateur, la réponse du chatbot, le succès de l'interaction (a-t-elle été comprise ou non par le chatbot), la langue utilisée et le type de document utilisé pour former le chatbot.
- Le code comporte également une fonction "save" qui gère l'insertion de nouvelles entrées dans la table de la base de données. Les informations pertinentes sont recueillies et enregistrées dans la table chaque fois que le chatbot interagit avec un utilisateur.
- Finalement, la fonction "save" est exportée pour pouvoir être utilisée dans d'autres parties de l'application.



```

// Importer la bibliothèque sqlite3 en mode verbeux
const sqlite3 = require('sqlite3').verbose();

// Se connecter à la base de données SQLite nommée "AlBaraka_Chatbot.db" (la crée si elle n'existe pas)
let db = new sqlite3.Database('AlBaraka_Chatbot.db', (err) => {
    // Vérifier si une erreur s'est produite lors de la connexion à la base de données
    if (err) {
        // Afficher le message d'erreur
        console.error(err.message);
    }
    // Afficher un message indiquant que la connexion a été établie avec succès
    console.log('Connected to the AlBaraka_Chatbot database.');
});

// Exécuter une commande SQL pour créer une table nommée "AlBaraka_chatbot" si elle n'existe pas déjà
db.run(`
CREATE TABLE IF NOT EXISTS AlBaraka_chatbot(
    id INTEGER PRIMARY KEY,
    message TEXT,
    response TEXT,
    success INTEGER,
    language TEXT,
    documentType TEXT
)`);

// Fonction de rappel exécutée après la tentative de création de la table
(db) => {
    // Vérifier s'il y a une erreur lors de la création de la table
    if (err) {
        // Afficher le message d'erreur
        console.error(err.message);
    } else {
        // Afficher un message indiquant que la table a été créée avec succès ou qu'elle existait déjà
        console.log('Table AlBaraka_chatbot has been created or already exists.');
    }
};

// Définir une fonction nommée "save" pour enregistrer les données dans la table "AlBaraka_chatbot"
const save = (data) => {
    // Obtenir la date et l'heure actuelles au format ISO
    const datetime = new Date().toISOString();
    // Exécuter une commande SQL pour insérer un nouvel enregistrement dans la table "AlBaraka_chatbot"
    db.run(`INSERT INTO AlBaraka_chatbot(message, response, success, language, documentType) VALUES(?, ?, ?, ?, ?)`, [
        data.message, data.response, data.success ? 1 : 0, data.language, data.documentType || 'Unknown'
    ], function(err) {
        // Vérifier s'il y a une erreur lors de l'insertion de l'enregistrement
        if (err) {
            // Afficher le message d'erreur
            return console.error(err.message);
        }
        // Afficher un message indiquant que l'enregistrement a été inséré avec succès, avec l'identifiant de l'enregistrement
        console.log('A row has been inserted with rowid ${this.lastID}');
    });
};

// Exporter la fonction "save" pour qu'elle puisse être utilisée dans d'autres fichiers de ce projet
module.exports = { save };

```



5.9 Affichage des données du Chatbot

Ce script Python facilite l'analyse des performances du chatbot en utilisant une interface graphique et des graphiques interactifs. Il comprend trois principales fonctionnalités :

- La visualisation des interactions du chatbot directement depuis la base de données, mettant en évidence les réussites et les échecs.
- La présentation du taux de réussite du chatbot sous forme de graphique à barres, montrant les réponses réussies par rapport aux échecs.
- L'affichage de la distribution des types de documents des messages du chatbot sous forme de graphique à barres.
-

L'interface utilisateur inclut des boutons permettant de générer les graphiques susmentionnés à la demande.

```
# Importer les modules nécessaires
import sqlite3
import tkinter as tk
from tkinter import ttk
import matplotlib.pyplot as plt

# Définition de la fonction pour visualiser les données
def view_data():
    # Rendre ces variables globales pour pouvoir y accéder à l'extérieur de la fonction
    global success_index, document_type_index

    # Supprimer toutes les entrées précédentes
    for entry in data_frame.grid_slaves():
        entry.destroy()

    # Se connecter à la base de données
    conn = sqlite3.connect('C:/Users/abdel/OneDrive/Bureau/Projects/Al Baraka WS/www.albaraka.com.tn/chatbot/AlBaraka_chatbot.db')
    cursor = conn.cursor()

    # Récupérer les données
    cursor.execute("SELECT * FROM AlBaraka_chatbot")
    records = cursor.fetchall()

    # Obtenir les noms des colonnes à partir de la description du curseur
    columns = [column[0] for column in cursor.description]
    success_index = columns.index('success')
    document_type_index = columns.index('documentType')
```

```

# Afficher les données dans l'interface graphique
for i in range(len(records)):
    for j in range(len(records[0])):
        e = ttk.Entry(data_frame, width=20, font=('Arial', 16, 'bold'))
        e.grid(row=i, column=j)

        # Vérifier si le champ de succès est vrai, et si c'est le cas, rendre le texte vert, sinon rouge.
        if j == success_index:
            if records[i][j]:
                e.configure(foreground='green')
                e.insert(tk.END, 'Success')
            else:
                e.configure(foreground='red')
                e.insert(tk.END, 'Failure')
        else:
            e.insert(tk.END, '' if records[i][j] is None else records[i][j])

# Fermer la connexion
conn.close()

# Rappeler cette fonction après 1000ms (1 seconde)
root.after(1000, view_data)

# Définition de la fonction pour afficher le graphique du taux de succès
def show_success_rate_plot():
    # Déclarer cette variable comme globale
    global success_index

    # Se connecter à la base de données
    conn = sqlite3.connect('C:/Users/abdel/OneDrive/Bureau/Projects/Al Baraka WS/www.albaraka.com.tn/chatbot/AlBaraka_Chatbot.db')
    cursor = conn.cursor()

    # Récupérer les données
    cursor.execute("SELECT * FROM AlBaraka_chatbot")
    records = cursor.fetchall()

    # Fermer la connexion
    conn.close()

    # Calculer le taux de succès
    success_count = sum(record[success_index] for record in records)
    total_count = len(records)
    success_rate = success_count / total_count if total_count > 0 else 0

    # Créer et afficher le graphique du taux de succès
    plt.figure(figsize=(8, 6))
    plt.bar(['Success', 'Failure'], [success_count, total_count - success_count], color=['green', 'red'])
    plt.xlabel('Response')
    plt.ylabel('Count')
    plt.title('Chatbot Response Success Rate')
    plt.show()

# Créer la fenêtre principale
root = tk.TK()
root.title("AlBaraka Chatbot Data")

# Créer le cadre pour l'affichage des données
data_frame = ttk.Frame(root)
data_frame.grid(row=0, column=0)

# Bouton pour afficher le graphique du taux de succès
success_rate_button = ttk.Button(root, text="Show Success Rate Plot", command=show_success_rate_plot)
success_rate_button.grid(row=1, column=0, pady=5)

# Bouton pour afficher le graphique du type de document
document_type_button = ttk.Button(root, text="Show Document Type Plot", command=show_document_type_plot)
document_type_button.grid(row=2, column=0, pady=5)

# Appeler la fonction view_data immédiatement lorsque le script est exécuté
view_data()

# Commencer la boucle principale de tkinter
root.mainloop()

```



5.10 Implémentation de l'interface utilisateur de gestion Q&A

Cet extrait de code implémente une interface utilisateur graphique (GUI) pour la gestion des questions et réponses du chatbot. L'interface est construite en utilisant Tkinter, une bibliothèque Python pour la création d'interfaces utilisateur graphiques. Cette interface permet à l'administrateur d'ajouter, de mettre à jour ou de supprimer des questions et des réponses qui seront utilisées par le chatbot pour interagir avec les utilisateurs.

Plus précisément, l'interface graphique présente les fonctionnalités suivantes :

- Des champs de saisie pour le sujet, la question et la réponse, qui permettent à l'administrateur d'ajouter de nouvelles Q&R ou de modifier des Q&R existantes.
- Des boutons "Ajouter", "Mettre à jour" et "Supprimer" pour effectuer les actions correspondantes sur les Q&R.
- Un tableau affichant la liste des Q&R existantes.
- Un menu "Fichier" qui donne la possibilité d'importer des Q&R à partir d'un fichier texte.
- En coulisses, l'application stocke toutes les Q&R dans une base de données SQLite. Elle utilise également le système de fichiers pour enregistrer chaque paire de Q&R dans un fichier texte distinct.

```
# Import des modules nécessaires depuis tkinter et d'autres bibliothèques
import tkinter as tk
from tkinter import filedialog, simpledialog, messagebox, ttk
from tkinter.scrolledtext import ScrolledText
import tkinter.font as tkFont
import sqlite3
import os
```



```

# Créer une classe ChatbotQA pour l'application de chatbot de questions-réponses
class ChatbotQA:
    def __init__(self):
        # Créer la fenêtre principale
        self.window = tk.Tk()
        self.window.title("Chatbot Q&R")
        self.window.geometry("800x500")

        # Définir une police personnalisée pour l'application
        self.default_font = tk.font.nametofont("TkDefaultFont")
        self.default_font.configure(size=14)

        # Créer et configurer des styles pour les widgets de l'application
        self.style = ttk.Style()
        self.style.configure("Treeview.Heading", font=('Arial', 14)) # Changer la police de l'en-tête des colonnes
        self.style.configure("TButton", font=("Arial", 10), padding=10)
        self.style.configure("TEntry", font=("Arial", 14), padding=10)
        self.style.configure(" TLabel", font=("Arial", 14), padding=10)

        # Créer des étiquettes, des zones de saisie, des boutons et une Treeview pour afficher les données Q&R
        self.subject_label = ttk.Label(self.window, text="Sujet:")
        self.subject_entry = ttk.Entry(self.window, font=("Arial", 14), width=30)

        self.question_label = ttk.Label(self.window, text="Question:")
        self.question_entry = ScrolledText(self.window, font=("Arial", 14), width=120, height=5)

        self.answer_label = ttk.Label(self.window, text="Réponse:")
        self.answer_entry = ScrolledText(self.window, font=("Arial", 14), width=120, height=5)

        self.add_button = ttk.Button(self.window, text="Ajouter", command=self.add_qa)
        self.update_button = ttk.Button(self.window, text="Mettre à jour", command=self.update_qa)
        self.delete_button = ttk.Button(self.window, text="Supprimer", command=self.delete_qa)

        self.qa_list = ttk.Treeview(self.window, columns=("ID", "Sujet", "Question", "Réponse"), show="headings")
        self.qa_list.bind('<Double-1>', self.load_qa)

        # Définir les en-têtes des colonnes dans la Treeview
        self.qa_list.heading("ID", text="ID")
        self.qa_list.heading("Sujet", text="Sujet")
        self.qa_list.heading("Question", text="Question")
        self.qa_list.heading("Réponse", text="Réponse")

        # Définir les largeurs des colonnes dans la Treeview
        self.qa_list.column("ID", minwidth=0, width=50, stretch=tk.NO)
        self.qa_list.column("Sujet", minwidth=0, width=250, stretch=tk.YES)
        self.qa_list.column("Question", minwidth=0, width=250, stretch=tk.YES)
        self.qa_list.column("Réponse", minwidth=0, width=250, stretch=tk.YES)

        # Disposition en grille pour tous les widgets dans la fenêtre principale
        self.subject_label.grid(row=0, column=0)
        self.subject_entry.grid(row=0, column=1)
        self.question_label.grid(row=1, column=0)
        self.question_entry.grid(row=1, column=1)
        self.answer_label.grid(row=2, column=0)
        self.answer_entry.grid(row=2, column=1)
        self.add_button.grid(row=3, column=0)
        self.update_button.grid(row=3, column=1)
        self.delete_button.grid(row=4, column=0)
        self.qa_list.grid(row=5, column=0, columnspan=2, sticky="nsew")

```



```

# Configuration des poids de colonne et de ligne pour le redimensionnement
self.window.grid_columnconfigure(0, weight=1)
self.window.grid_columnconfigure(1, weight=1)
self.window.grid_rowconfigure(5, weight=1)

# Créer une barre de menu et un menu de fichiers pour l'importation de données
self.menu = tk.Menu(self.window)
self.window.config(menu=self.menu)
self.filemenu = tk.Menu(self.menu)
self.menu.add_cascade(label="Fichier", menu=self.filemenu)
self.filemenu.add_command(label="Importer...", command=self.import_file)

# Se connecter à la base de données SQLite et créer la table pour les données Q&R si elle n'existe pas
self.db = sqlite3.connect('q_and_a.db')
self.cursor = self.db.cursor()
self.cursor.execute('''
    CREATE TABLE IF NOT EXISTS q_and_a (
        id INTEGER PRIMARY KEY,
        sujet TEXT UNIQUE,
        question TEXT,
        reponse TEXT
    )
''')
self.db.commit()

# Charger les données existantes depuis la base de données dans la Treeview
self.load_data()

# Fonction auxiliaire pour formater le sujet à utiliser dans les noms de fichiers
def format_subject(self, subject):
    return subject.replace(' ', '_')

# Fonction auxiliaire pour enregistrer les données Q&R dans un fichier
def save_to_file(self, subject, question, answer):
    if not os.path.exists('Q&A_chatbot'):
        os.makedirs('Q&A_chatbot')

    subject = self.format_subject(subject)
    with open(f'Q&A_chatbot/{subject}.txt', 'w') as file:
        file.write(f'Q: {{ input: "{question}" , output: "{subject}" }},\n')
        file.write(f'A: "{answer}"')

# Méthode pour ajouter des données Q&R à la base de données et à la Treeview
def add_qa(self, subject=None, question=None, answer=None):
    if subject is None or question is None or answer is None:
        subject = self.subject_entry.get()
        question = self.question_entry.get("1.0", 'end-1c')
        answer = self.answer_entry.get("1.0", 'end-1c')

    # Vérifier si un champ est vide et afficher un message d'erreur le cas échéant
    if subject == "" or question == "" or answer == "":
        messagebox.showerror("Erreur de saisie", "Tous les champs doivent être remplis")
        return

    try:
        # Exécuter une instruction SQL INSERT pour ajouter les données Q&R à la base de données
        self.cursor.execute('''
            INSERT INTO q_and_a (sujet, question, reponse) VALUES (?, ?, ?)
        ''', (self.format_subject(subject), question, answer))
        values=row
    
```



```

# Méthode pour charger les données Q&R sélectionnées dans la Treeview dans les zones de saisie appropriées
def load_qa(self, event):
    row_id = self.qa_list.focus()
    data = self.qa_list.item(row_id)

    self.subject_entry.delete(0, tk.END)
    self.question_entry.delete('1.0', tk.END)
    self.answer_entry.delete('1.0', tk.END)

    self.subject_entry.insert(tk.END, data['values'][1].replace('_', ' '))
    self.question_entry.insert(tk.END, data['values'][2])
    self.answer_entry.insert(tk.END, data['values'][3])

# Méthode pour supprimer des données Q&R de la base de données et la Treeview
def delete_qa(self):
    row_id = self.qa_list.focus()
    data = self.qa_list.item(row_id)

    if not data:
        messagebox.showerror("Erreur de sélection", "Aucun enregistrement sélectionné")
        return

    self.cursor.execute('DELETE FROM q_and_a WHERE id = ?', (data['values'][0],))
    self.db.commit()

    subject_file = f"Q&A_chatbot/{data['values'][1]}.txt"
    if os.path.exists(subject_file):
        os.remove(subject_file)

    messagebox.showinfo("Succès", "Question et réponse supprimées avec succès")
    self.load_data()

# Méthode pour importer des données à partir d'un fichier texte dans la base de données et la Treeview
def import_file(self):
    filepath = filedialog.askopenfilename(filetypes=[("Fichiers texte", "*.txt")])

    if filepath:
        with open(filepath, 'r') as file:
            lines = file.readlines()

            for line in lines:
                line = line.strip()
                if line:
                    subject = simpledialog.askstring("Saisie", f"Sujet pour la question : '{line}' ?")
                    if subject is None:
                        return

                    answer = simpledialog.askstring("Saisie", f"Réponse pour la question : '{line}' ?", initialvalue="AUCUNE")
                    self.add_qa(subject, line, answer)

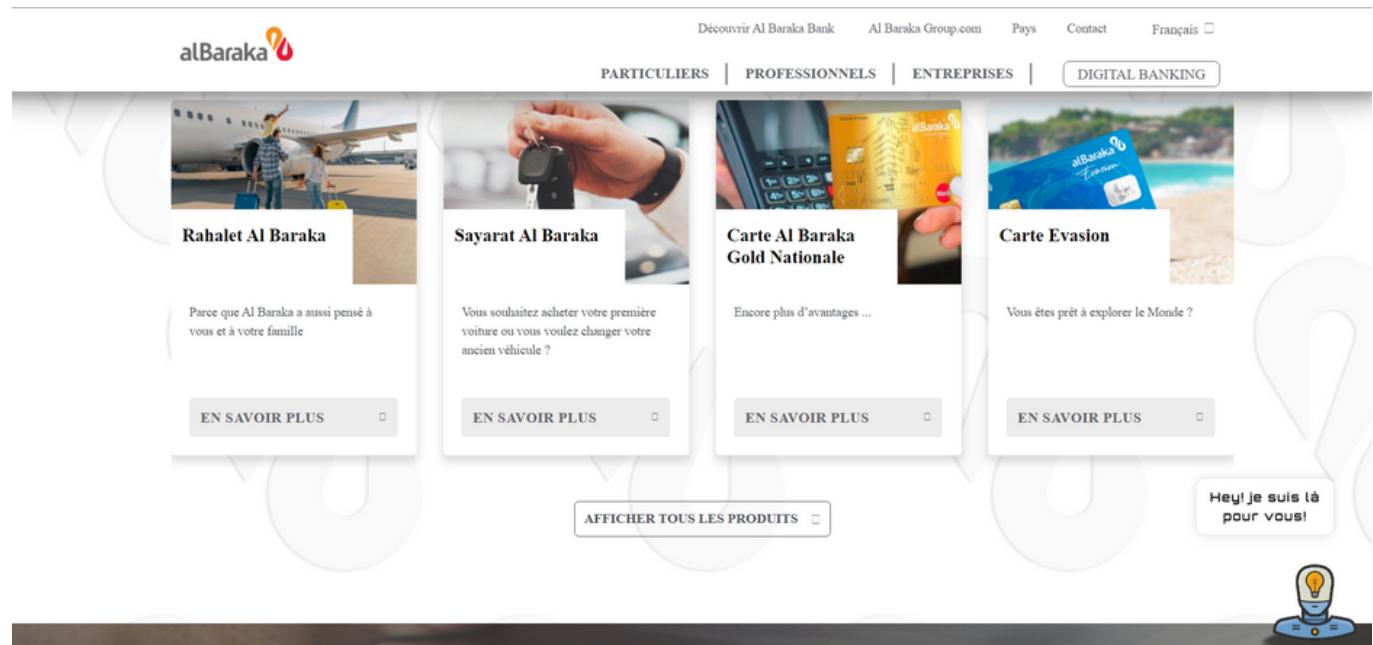
# Méthode pour exécuter l'application en mode boucle d'événements
def run(self):
    self.window.mainloop()

```



6 - Interfaces

6.1 Site web



6.2 Chatbot



6.3 Interaction Chatbot

6.3.1 Demande rendez vous avec un conseiller Financier



6.3.2 Horaires d'ouverture

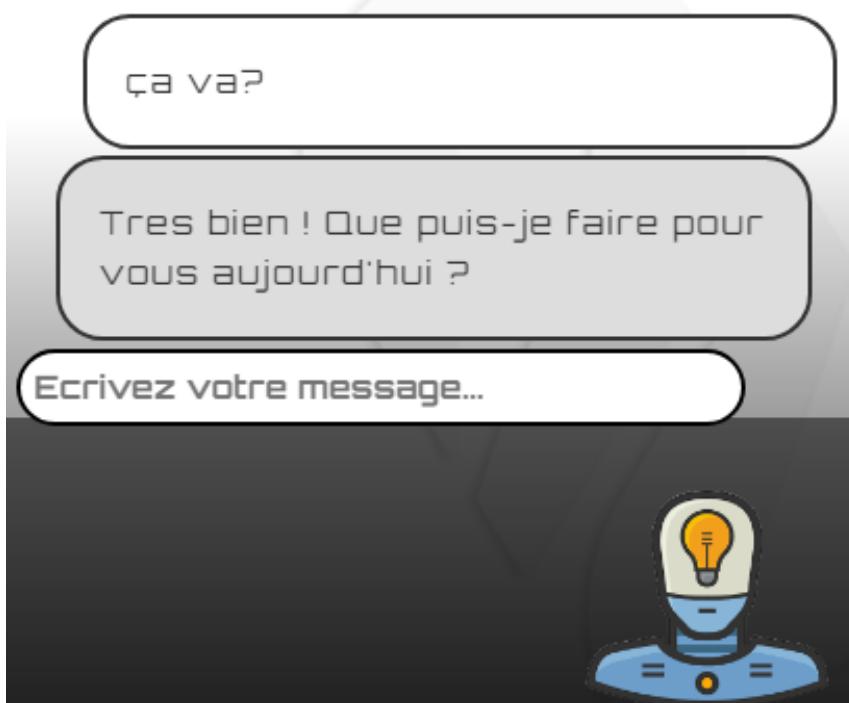
horaires

Nous attendons votre visite dans nos agences de 8h15 à 16h30 pendant la double séance, et de 8h15 à 13h30 pendant la séance unique.

Ecrivez votre message...



6.3.3 Interrogation sur le bien etre



6.3.4 Offre Etudiant

offre etudiant

Exactement, nous avons mis en place une offre bancaire dédiée aux étudiants. Vous pouvez consulter les détails en cliquant ici :
<https://www.albaraka.com.tn/fr/partner-etudiant>.

Ecrivez votre message...



6.3.5 Ouverture Compte



6.3.6 Salutation



6.3.7 Solde Compte

solde de mon compte

Obtenez rapidement et facilement le solde de votre compte grâce à notre application 'Banecti'. Nous sommes heureux de vous aider :)

Ecrivez votre message...



6.3.8 Taux de change

taux de change

Pour consulter les fluctuations
des taux de change, veuillez
cliquer ici :
<https://www.albaraka.com.tn/fr/cours-des-changes>.

Ecrivez votre message...



6.3.9 Question D'identification

The image shows a digital interface for a chatbot. At the top, there is a red and yellow decorative bar. Below it, a question is asked in a white speech bubble: "qui es tu?". A large, grey, rounded rectangular area contains a response from the AI: "En tant qu'intelligence artificielle d'Albaraka Bank, je suis là pour vous aider à gérer vos comptes, effectuer des paiements et bien plus encore. Que puis-je faire pour vous aujourd'hui ?". Below this, another white speech bubble says "Ecrivez votre message...". At the bottom right, there is a small blue robot head icon with a lightbulb inside its head, and a yellow speech bubble next to it containing the number "43".

qui es tu?

En tant qu'intelligence artificielle d'Albaraka Bank, je suis là pour vous aider à gérer vos comptes, effectuer des paiements et bien plus encore. Que puis-je faire pour vous aujourd'hui ?

Ecrivez votre message...

43

6.3.10 Mot de passe oublié



6.3.11 Autofinancement Voiture 4CV

Autofinancement voiture

Merci de vous rendre sur
<https://www.albaraka.com.tn/fr/simulation>
pour effectuer la simulation, puis
veuillez remplir le formulaire en
utilisant ce lien :
<https://www.albaraka.com.tn/fr/contactez-nous>.

Ecrivez votre message...



6.3.12 Demande d'information pour AlBaraka Bank

a propos albaraka

Al Baraka est une banque bien établie et reconnue dans le domaine de la finance islamique. Elle opère depuis de nombreuses années et compte une présence internationale. Cependant, la fiabilité d'une banque peut varier en fonction de différents facteurs tels que la réglementation, la gouvernance et les pratiques spécifiques à chaque pays. Il est recommandé de faire des recherches supplémentaires et de consulter des avis d'experts avant de prendre une décision financière.

Ecrivez votre message...



6.3.13 Carte Allocation Touristique

carte allocation

La carte évasion vous permet de dépenser votre allocation touristique en toute sécurité, sans vous inquiéter.

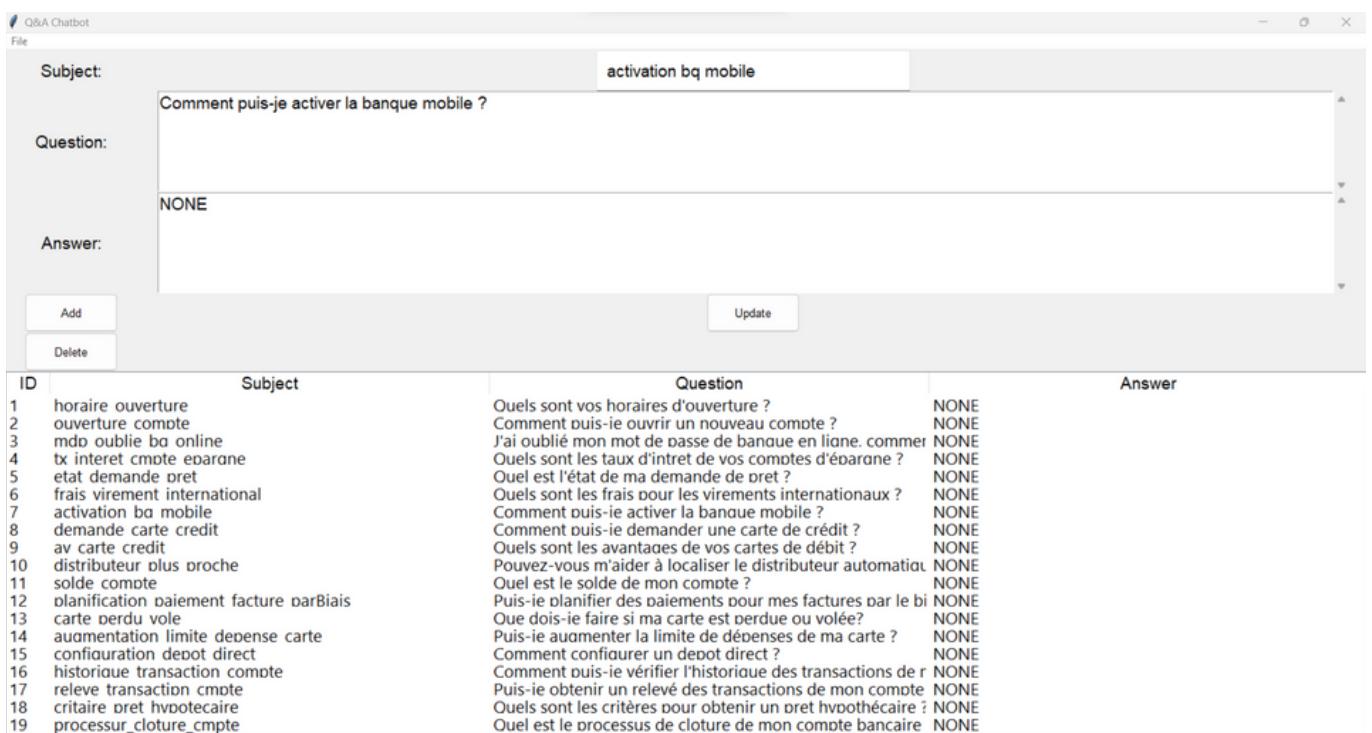
Ecrivez votre message...



6.3.14 Demande Pret Automobile



6.3 Com Software



The screenshot shows a software interface for managing a Q&A chatbot. At the top, there's a search bar with the text "activation bq mobile". Below it, a question is displayed: "Comment puis-je activer la banque mobile ?". The response field is labeled "NONE". The interface includes buttons for "Add" and "Delete" at the bottom left, and "Update" at the bottom right. A large table below lists 19 entries, each with an ID, subject, question, and answer.

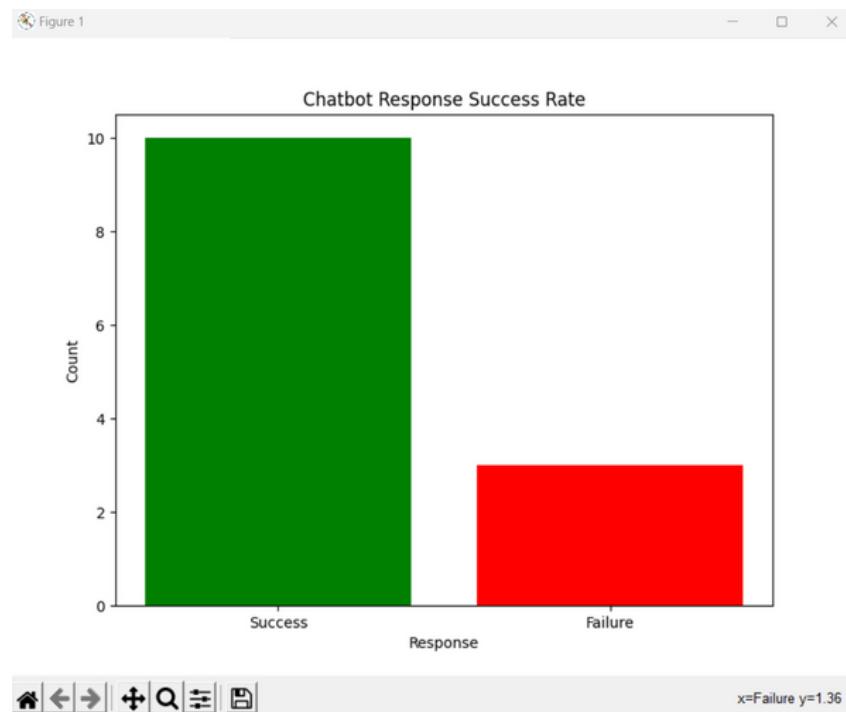
ID	Subject	Question	Answer
1	horaire ouverture	Quels sont vos horaires d'ouverture ?	NONE
2	ouverture compte	Comment puis-je ouvrir un nouveau compte ?	NONE
3	mdo oublier ba online	J'ai oublié mon mot de passe de banque en ligne. comment	NONE
4	tx interet cmpte eparane	Quels sont les taux d'intérêt de vos comptes d'épargne ?	NONE
5	etat demande pret	Quel est l'état de ma demande de prêt ?	NONE
6	frais virement international	Quels sont les frais pour les virements internationaux ?	NONE
7	activation ba mobile	Comment puis-je activer la banque mobile ?	NONE
8	demande carte credit	Comment puis-je demander une carte de crédit ?	NONE
9	av carte credit	Quels sont les avantages de vos cartes de débit ?	NONE
10	distributeur plus proche	Pouvez-vous m'aider à localiser le distributeur automatique ?	NONE
11	solde compte	Quel est le solde de mon compte ?	NONE
12	planification paiement facture parBiais	Puis-je planifier des paiements pour mes factures par le biais	NONE
13	carte perdu volée	Que dois-je faire si ma carte est perdue ou volée ?	NONE
14	augmentation limite depense carte	Puis-je augmenter la limite de dépenses de ma carte ?	NONE
15	configuration depot direct	Comment configurer un dépôt direct ?	NONE
16	historique transaction compte	Comment puis-je vérifier l'historique des transactions de mon compte ?	NONE
17	releve transaction cmpte	Puis-je obtenir un relevé des transactions de mon compte ?	NONE
18	critaire pret hypothecaire	Quels sont les critères pour obtenir un prêt hypothécaire ?	NONE
19	processus_cloture_cmpte	Quel est le processus de clôture de mon compte bancaire	NONE



6.3 Analyse Chatbot

AlBaraka Chatbot Data					
1	salut	Salutations ! Comment	Success	fr	salutation
2	vente voiture 4cv	Merci de bien vouloir ac	Success	fr	demande_pret_automol
3	salut	Salut comment puis-je v	Success	fr	salutation
4	vehivule 4cv	Désolé, je n'ai pas bien	Failure	fr	None
5	salut	Bonjour ! Que puis-je fa	Success	fr	salutation
6	t qui	Je suis là pour répondre	Success	fr	whomi
7	hjbfdsdqf	Désolé, je n'ai pas bien	Failure	fr	None
8	salut ça va?	Je vais bien, merci ! Co	Success	fr	mood
9	d'accord c'est quoi alba	Al Baraka a été fondée e	Success	fr	A_propos_AlBaraka
10	hello helloooo	Désolé, je n'ai pas bien	Failure	fr	None
11	non hahahah j'espere q	Je vais bien, merci ! Co	Success	fr	mood
12	je veux ouvrir un compt	Nous serions ravis de v	Success	fr	ouverture_compte
13	salut	Salut ! Que puis-je faire	Success	fr	salutation

6.3.1 Plot Success Rate



6.3.2 Plot Documents

