



**ECOLE MAROCAINE DES
SCIENCES DE L'INGENIEUR**

Membre de

 HONORIS UNITED UNIVERSITIES

RAPPORT

TITRE DU PROJET :

**Gestion des réservations des tables pour des
restaurants**

Préparé par :

- **AMEZIANE Douaae**
- **TOUIEB Othmane**
- **KAMAL Aymane**

Encadré par :

- **M^{lle} HAZMAN Chaimaa**
- **M^{lle} SNIBA Farah**
- **M. SAFSOUF Yassine**

**Année universitaire :
2023-2024**

Sommaire

I. Introduction

- **Contexte du projet**
- **Objectifs du projet**

II. Cahier des charges

- **Description des besoins des administrateurs**
- **Description des besoins des clients**
- **Identification des fonctionnalités essentielles**

III. Conception de l'Application

- **Choix des technologies utilisées**

IV. Déploiement

- **Configuration du serveur et de la base de données**

V. Annexes

- **Diagrammes UML**
- **Requêtes SQL dans fonctions**
- **Captures d'écran de l'application**

VI. Remerciements

VII. Conclusion

I. Introduction :

1. Contexte du projet

Dans un marché de la restauration compétitif et en évolution constante, notre groupe de restaurants cherche à moderniser ses processus de réservation et de gestion des tables. Actuellement basées sur des réservations téléphoniques, ces méthodes présentent des inefficacités et des risques d'erreurs. Notre objectif est de développer une application de gestion des tables pour simplifier le processus de réservation pour les clients et optimiser la gestion pour le personnel administratif, améliorant ainsi l'expérience globale des clients et renforçant la compétitivité de notre groupe de restaurants.

2. Objectifs du projet

Développer une application de gestion des tables pour faciliter les réservations clients et optimiser la gestion opérationnelle des restaurants du groupe.

II. Cahier des charges :

1. Introduction :

L'objectif de ce projet est de développer une application de gestion des tables pour un groupe de restaurants, offrant à la fois aux clients la possibilité de faire des réservations et aux administrateurs les outils nécessaires pour gérer efficacement les restaurants et les réservations.

2. Contexte :

Avec l'expansion du groupe de restaurants, il est devenu essentiel d'avoir un système centralisé pour gérer les réservations et les restaurants.

3. Description du Projet :

L'application devra permettre :

- *Pour les Administrateurs :*
 - **Ajouter, modifier et supprimer** des restaurants avec leurs détails (nom, adresse, type de cuisine, etc.).
 - **Consulter** la liste des réservations pour chaque restaurant avec la possibilité de les modifier ou de les supprimer.
 - **Consulter** toutes les réservations effectuées dans tous les restaurants ou par identifiant de réservation.
 - **Consulter** les statistiques du restaurant telles que le taux d'occupation des tables, etc.

- *Pour les Clients :*
 - **Effectuer** une nouvelle réservation en sélectionnant le restaurant, la date, l'heure et le nombre de personnes.
 - **Modifier** ou **annuler** une réservation existante.
 - **Consulter** la liste de leurs réservations avec la possibilité de les modifier ou de les annuler.
 - **Vérifier** la disponibilité des tables dans un restaurant pour une date et une heure spécifique.
 - **Rechercher** un restaurant par nom, type de cuisine ou adresse.

4. Exigences :

Fonctionnelles :

- Interface utilisateur intuitive et conviviale pour les administrateurs et les clients.
- Intégration de fonctionnalités de recherche avancée pour faciliter la navigation.

Techniques :

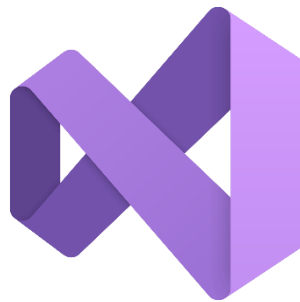
- Utilisation d'une base de données pour stocker les informations sur les restaurants et les réservations.

5. Contraintes :

- Le projet doit être achevé avant le 11 mai 2024 à 23h59.

III. Conception de l'Application :

1. Microsoft Visual Studio



Microsoft Visual Studio est un environnement de développement intégré (IDE) utilisé par les développeurs pour créer une variété d'applications logicielles, notamment des applications de bureau, des applications web, des applications mobiles et des services cloud. Il offre une suite complète d'outils de développement, y compris des éditeurs de code avancés, des

fonctionnalités de débogage, des outils de test et des capacités de déploiement.

L'environnement de développement intégré de Visual Studio prend en charge divers langages de programmation tels que C#, Visual Basic, C++, JavaScript, Python, et bien d'autres. Il permet aux développeurs de travailler efficacement en fournissant des fonctionnalités telles que la coloration syntaxique, l'achèvement automatique du code, la navigation dans le code, et la gestion des projets.

En outre, Visual Studio offre une intégration étroite avec d'autres outils et services de développement de Microsoft, tels qu'Azure DevOps pour la gestion de projet, Azure pour le déploiement dans le cloud, et GitHub pour le contrôle de version et la collaboration de code.

2. Microsoft SQL Server Management Studio (SSMS)



Microsoft SQL Server Management Studio (SSMS) est un outil graphique puissant utilisé pour gérer les instances de Microsoft SQL Server. Il fournit aux administrateurs de bases de données et aux développeurs un environnement intégré pour effectuer une variété de tâches de gestion de bases de données, telles que la création et la modification de bases de données, la conception de schémas, l'écriture et l'exécution de requêtes SQL, la gestion de la sécurité, et la surveillance des performances.

SSMS offre une interface utilisateur intuitive et conviviale, permettant aux utilisateurs de naviguer facilement à travers les bases de données, les tables et les procédures stockées. Il propose également des outils avancés de développement, tels que des éditeurs de requêtes graphiques, des

concepteurs de requêtes visuelles, et des outils de débogage pour aider les développeurs à optimiser et à déboguer leurs scripts SQL.

En outre, SSMS offre une intégration étroite avec d'autres outils et services Microsoft, tels qu'Azure SQL Database et Azure Synapse Analytics, permettant aux utilisateurs de gérer et de surveiller leurs bases de données dans le cloud de manière transparente.

3. StarUML



StarUML est un logiciel de modélisation UML (Unified Modeling Language) utilisé par les développeurs et les concepteurs logiciels pour créer des diagrammes et des modèles de conception pour leurs projets logiciels. Il offre une variété d'outils et de fonctionnalités pour aider les utilisateurs à visualiser, analyser et concevoir des systèmes logiciels complexes.

StarUML prend en charge les différents types de diagrammes UML, tels que les diagrammes de cas d'utilisation, les diagrammes de classes, les diagrammes de séquence, les diagrammes d'activité, et bien d'autres. Il permet aux utilisateurs de créer des diagrammes à l'aide d'une interface utilisateur intuitive et de partager leurs modèles avec d'autres membres de l'équipe de développement.

En outre, StarUML offre des fonctionnalités avancées telles que la génération de code à partir de modèles, l'ingénierie inverse pour importer du code source existant dans des diagrammes UML, et la compatibilité avec d'autres outils de modélisation et de développement.

4. C++



Le langage de programmation C++ est un langage de programmation polyvalent et puissant largement utilisé dans le développement logiciel. Créé à partir du langage C, C++ étend ce dernier avec des fonctionnalités orientées objet telles que l'encapsulation, l'héritage et le polymorphisme, tout en conservant sa performance et sa flexibilité.

C++ est largement utilisé dans divers domaines, y compris le développement d'applications système, les jeux vidéo, les applications embarquées, les logiciels de bureau et bien d'autres. Il offre aux développeurs un contrôle fin sur le matériel et les ressources système, tout en offrant un haut niveau d'abstraction grâce à ses fonctionnalités orientées objet.

Le langage C++ est pris en charge par de nombreux environnements de développement, tels que Microsoft Visual Studio, Eclipse CDT, et Xcode, et est largement utilisé dans l'industrie pour sa performance, sa portabilité et sa flexibilité.

IV. Déploiement :

1. Se connecter à la base de données :

Se connecter

History **Parcourir**

Tapez ici pour filtrer la liste

- Local
 - OTHMANE\SQLEXPRESS
 - MSSQLLocalDB
 - ProjectModels
- Réseau
- Azure

Nom du serveur : OTHMANE\SQLEXPRESS

Authentification : Authentification Windows

Nom d'utilisateur : OTHMANE\othma

Mot de passe :

☐ Se souvenir du mot de passe

Nom de la base de données : <par défaut>

Crypter : Obligatoire (Vrai)

Certificat de serveur de confiance : True

Avancé...

Se connecter Annuler

2. Création de la base de données :

```
create database RestaurantDb;
```

100 %

Aucun problème détecté

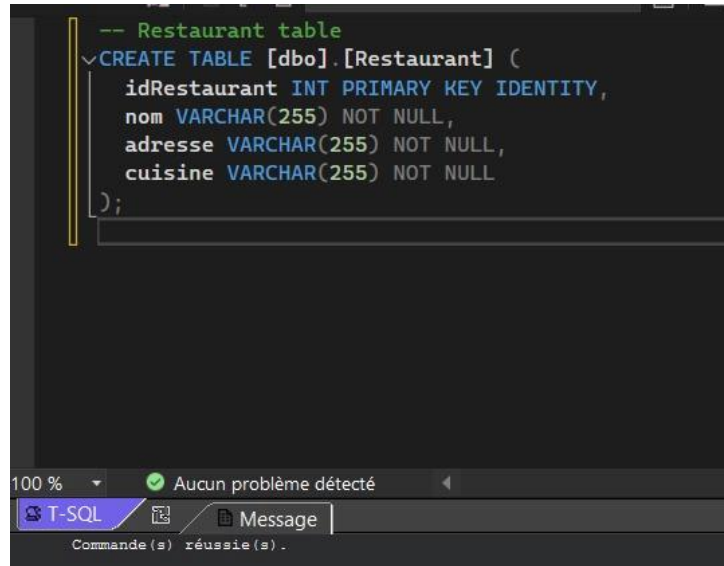
T-SQL Message

Commande(s) réussie(s).

3. Création des tables :

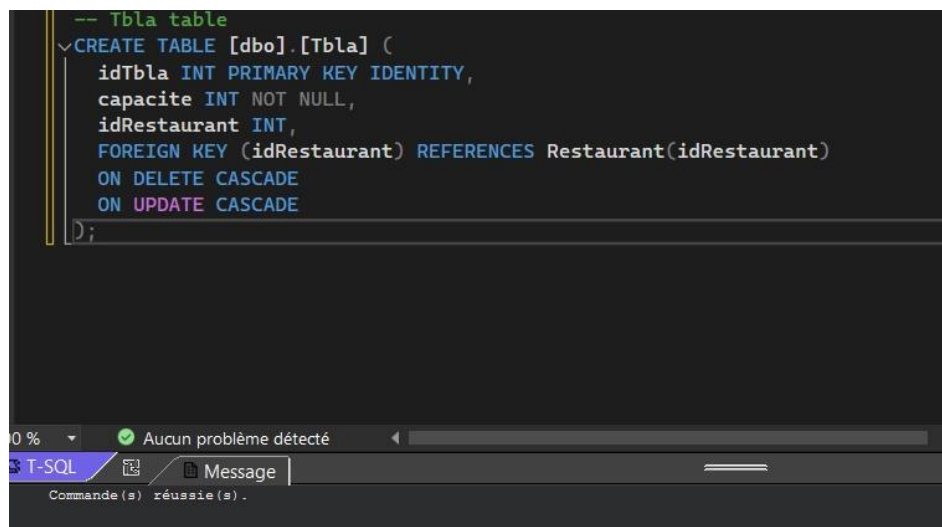
a. Table Restaurant :

```
-- Restaurant table
CREATE TABLE [dbo].[Restaurant] (
    idRestaurant INT PRIMARY KEY IDENTITY,
    nom VARCHAR(255) NOT NULL,
    adresse VARCHAR(255) NOT NULL,
    cuisine VARCHAR(255) NOT NULL
);
```



b. Table Tbla (Table) :

```
-- Tbla table
CREATE TABLE [dbo].[Tbla] (
    idTbla INT PRIMARY KEY IDENTITY,
    capacite INT NOT NULL,
    idRestaurant INT,
    FOREIGN KEY (idRestaurant) REFERENCES Restaurant(idRestaurant)
    ON DELETE CASCADE
    ON UPDATE CASCADE
);
```



c. Table Réservation :

```
-- Reservation table
CREATE TABLE [dbo].[Reservation] (
    idReservation INT PRIMARY KEY IDENTITY,
    date DATE NOT NULL,
    heureArr FLOAT NOT NULL,
    heureDep FLOAT NOT NULL,
    nbrPersonne INT NOT NULL,
    idTbla INT, -- Foreign key DE Tbla table
    FOREIGN KEY (idTbla) REFERENCES Tbla(idTbla)
    ON DELETE CASCADE
    ON UPDATE CASCADE
);
```

SQL Message

Commande(s) réussie(s).

d. Table Client :

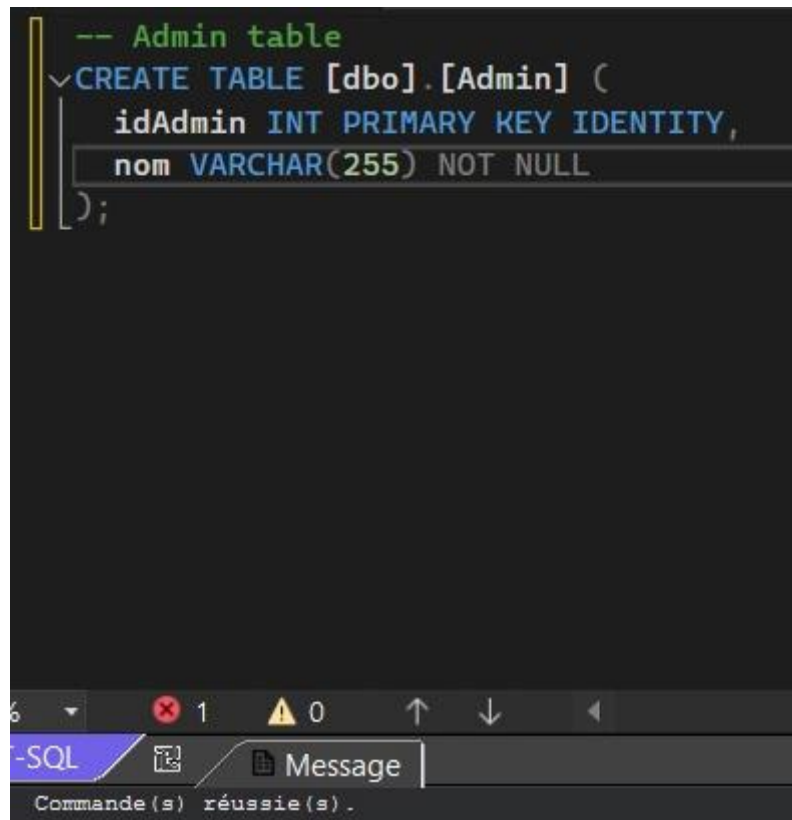
```
-- Client table
CREATE TABLE [dbo].[Client] (
    idClient INT PRIMARY KEY IDENTITY,
    nom VARCHAR(255) NOT NULL,
    email VARCHAR(255) NOT NULL
);
```

T-SQL Message

Commande(s) réussie(s).

e. Table Admin :

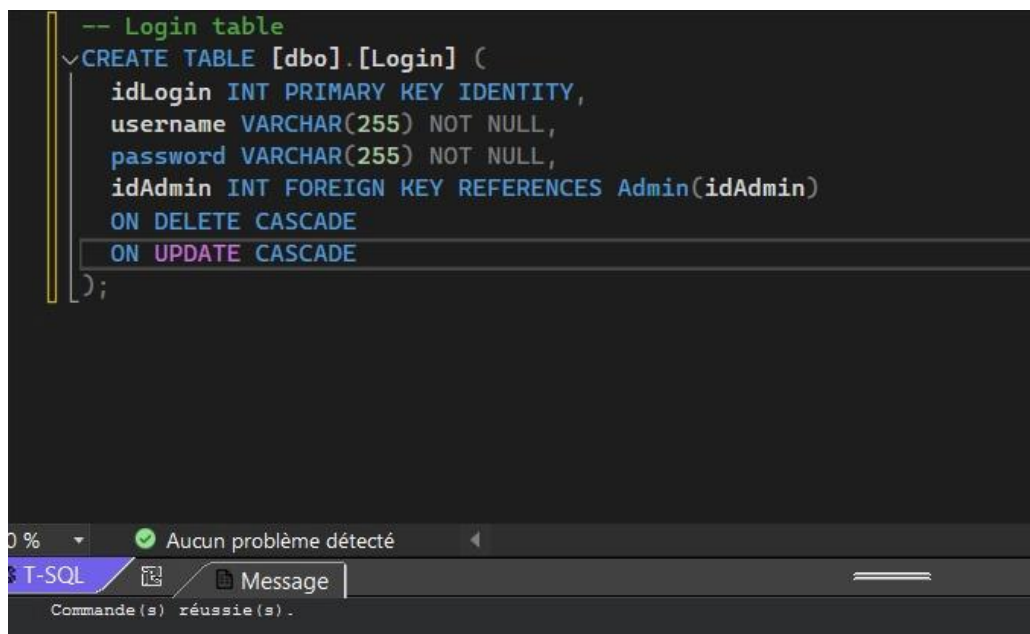
```
-- Admin table
CREATE TABLE [dbo].[Admin] (
    idAdmin INT PRIMARY KEY IDENTITY,
    nom VARCHAR(255) NOT NULL
);
```



The screenshot shows a SQL Server Enterprise Manager interface. The top pane displays the T-SQL script for creating the Admin table. The bottom pane shows the execution results, indicating that the command was successful. The status bar at the bottom of the window displays "Commande(s) réussie(s)." (Command(s) successful).

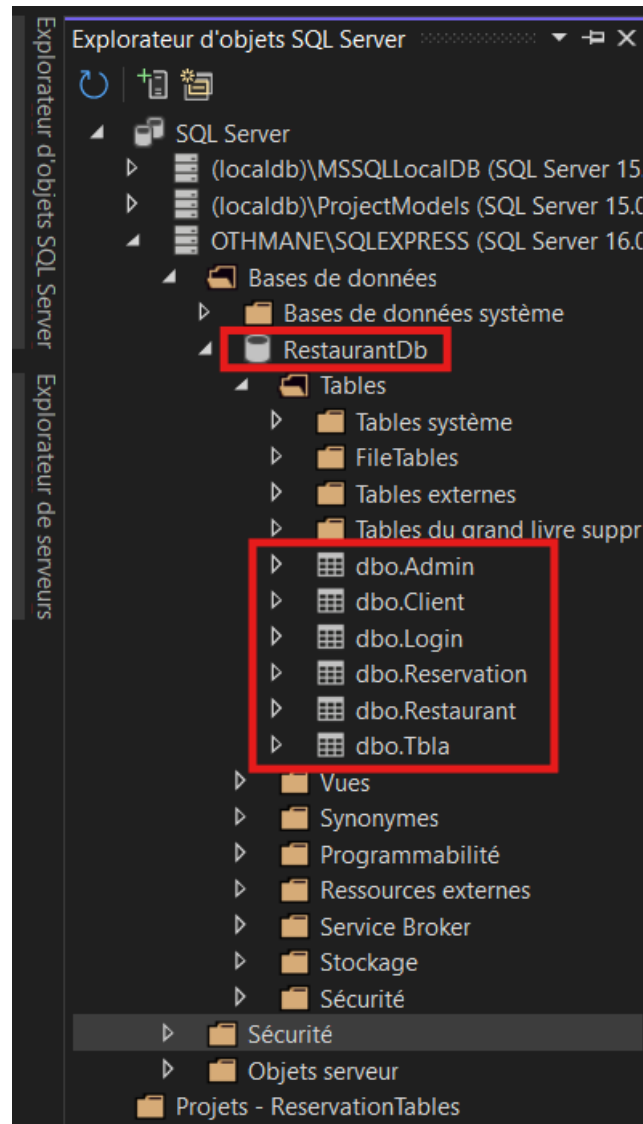
f. Table Login :

```
-- Login table
CREATE TABLE [dbo].[Login] (
    idLogin INT PRIMARY KEY IDENTITY,
    username VARCHAR(255) NOT NULL,
    password VARCHAR(255) NOT NULL,
    idAdmin INT FOREIGN KEY REFERENCES Admin(idAdmin)
    ON DELETE CASCADE
    ON UPDATE CASCADE
);
```



The screenshot shows a SQL Server Enterprise Manager interface. The top pane displays the T-SQL script for creating the Login table, which includes a foreign key relationship with the Admin table. The bottom pane shows the execution results, indicating that the command was successful. The status bar at the bottom of the window displays "Commande(s) réussie(s)." (Command(s) successful).

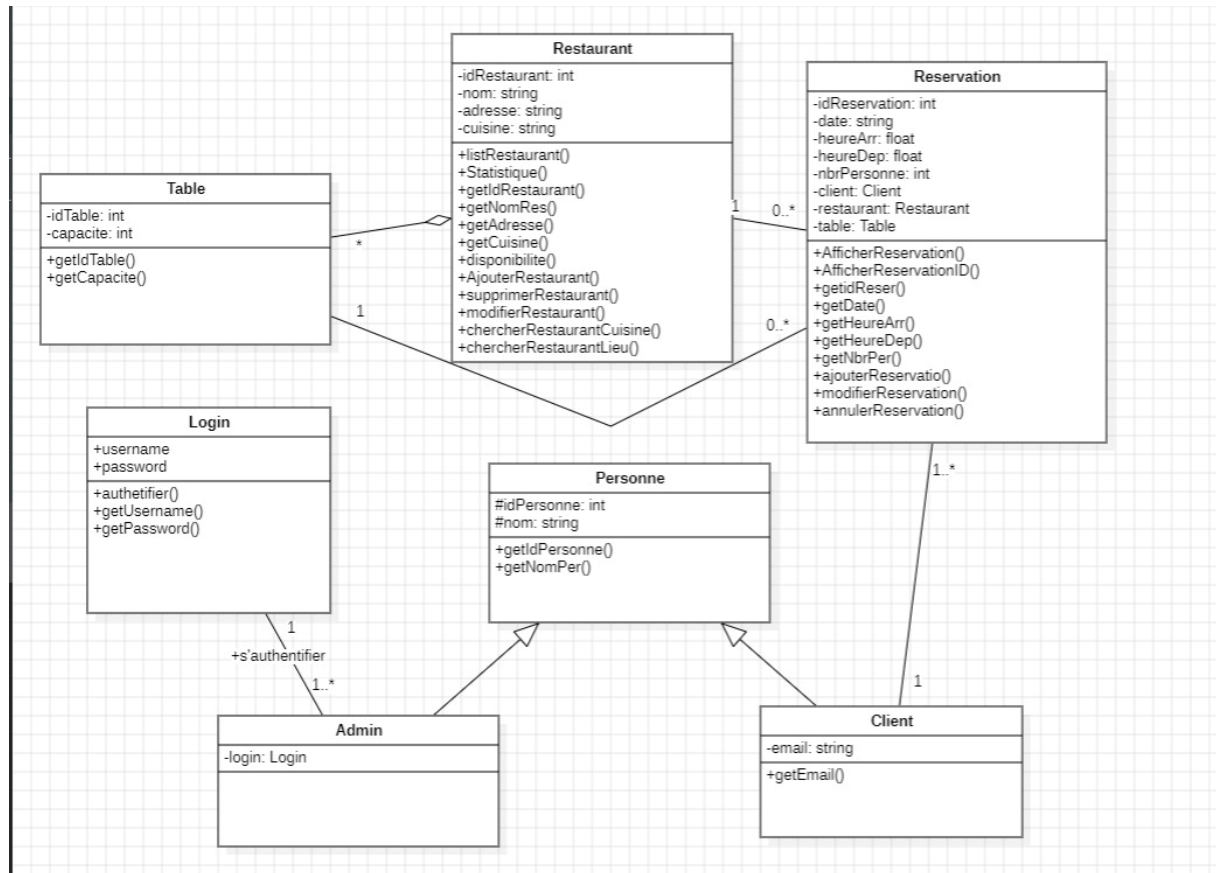
4. Tables sont créés :



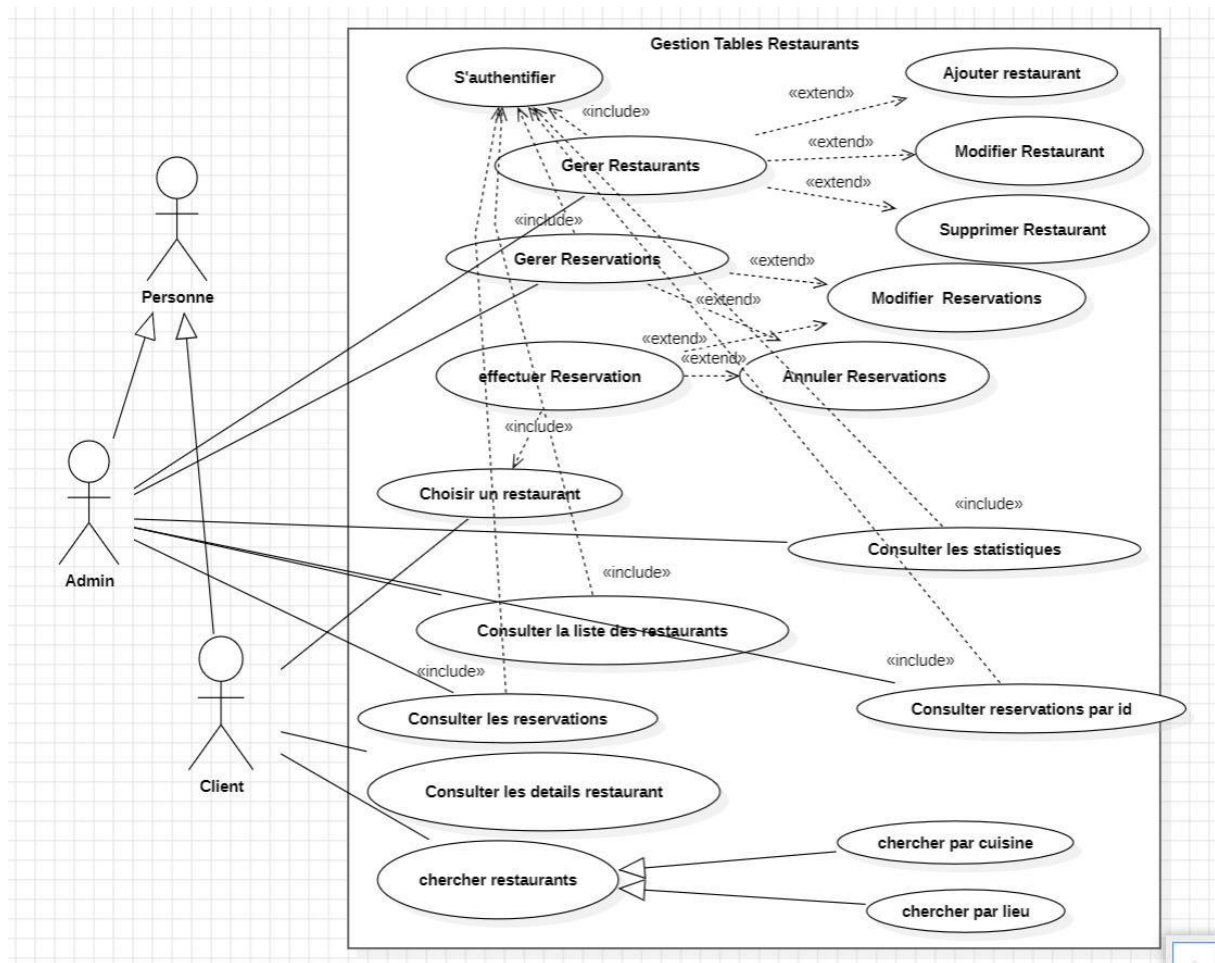
V. Annexes

1. Diagrammes UML :

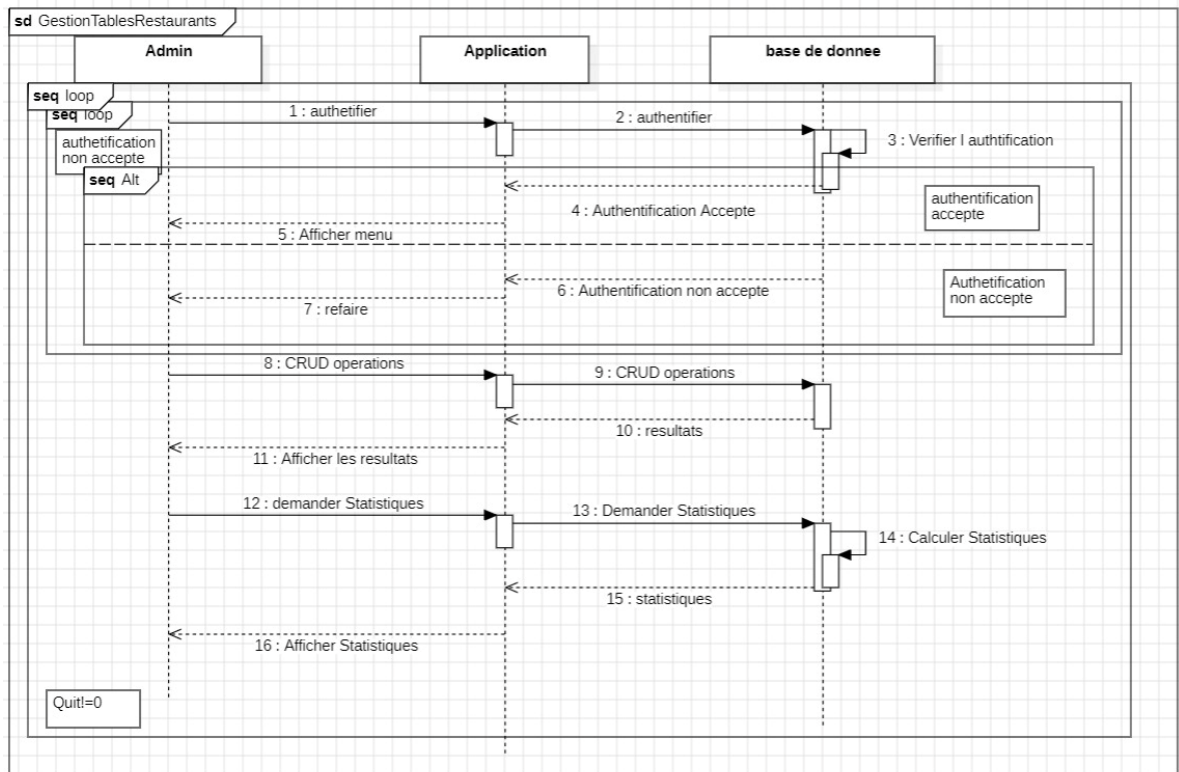
i. Diagramme de classes



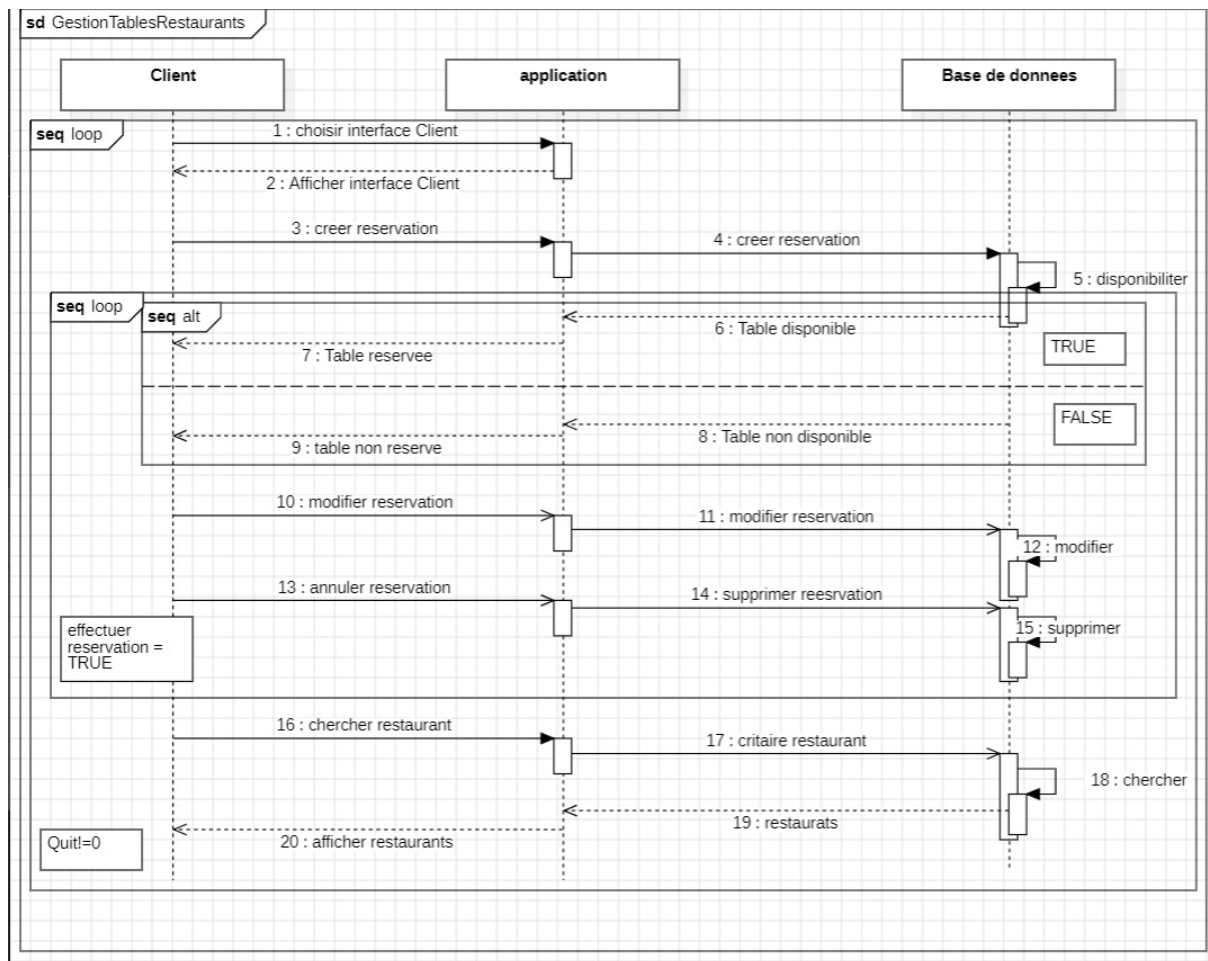
ii. Diagramme de cas d'utilisation



iii. Diagramme de séquence (Admin)



iv. Diagramme de séquence (Client)



2. Requêtes SQL dans fonctions :

```
#ifndef CONNECTION_H
#define CONNECTION_H

#include <cstdlib>
#include <sql.h>
#include <sqlext.h>

class Connection {
public:
    SQLHENV henv;
    SQLHDBC hdbc;
    SQLHSTMT hstmt;
    SQLRETURN retcode;

public:
    Connection();
    ~Connection();
    void connect();
    void disconnect();
    SQLHSTMT getStatement() const;
};

#endif // CONNECTION_H
```

```
#include <cstdlib>

#include "Connection.h"

Connection::Connection() {
    henv = SQL_NULL_HENV;
    hdbc = SQL_NULL_HDBC;
    hstmt = SQL_NULL_HSTMT;
    retcode = SQL_SUCCESS;
}

Connection::~Connection() {
    disconnect();
}

void Connection::connect() {
    SQLAllocHandle(SQL_HANDLE_ENV, SQL_NULL_HANDLE, &henv);
    SQLSetEnvAttr(henv, SQL_ATTR_ODBC_VERSION, (SQLPOINTER)SQL_OV_ODBC3, 0);
    SQLAllocHandle(SQL_HANDLE_DBC, henv, &hdbc);
    SQLWCHAR outstr[1024];
    SQLSMALLINT outstrlen;
    SQLDriverConnect(hdbc, NULL, (SQLWCHAR*)"DRIVER={SQL Server};SERVER=DESKTOP-G5SLSAV\\SQLEXPRESS;DATABASE=ResDB;UID=DESKTOP-G5SLSAV\\Douaa;PWD=", SQL_NTS, outstr, sizeof(outstr), &outstrlen, SQL_DRIVER_NOPROMPT);
    SQLAllocHandle(SQL_HANDLE_STMT, hdbc, &hstmt);
}

void Connection::disconnect() {
    SQLFreeHandle(SQL_HANDLE_STMT, hstmt);
    SQLDisconnect(hdbc);
    SQLFreeHandle(SQL_HANDLE_DBC, hdbc);
    SQLFreeHandle(SQL_HANDLE_ENV, henv);
}

SQLHSTMT Connection::getStatement() const {
    return hstmt;
}
```

```
// Requête SQL pour obtenir l'ID du client à partir de l'email
SQLWCHAR query[255];
swprintf(query, L"SELECT idClient FROM Client WHERE email = '%s'", email.c_str());
```

```
// Execute SQL query to fetch password for the provided username
SQLHSTMT stmt = connection.createStatement();
SQLWCHAR query[255];
swprintf(query, L"SELECT password FROM Login WHERE username = '%s'", inputUsername.c_str());
SQLExecDirect(stmt, query, SQL_NTS);
```

```
// Requête SQL pour récupérer toutes les réservations de la base de données
SQLWCHAR query[255];
swprintf(query, L"SELECT * FROM Reservation");
```

```
// Requête SQL pour récupérer les détails de la réservation par ID
SQLWCHAR query[255];
swprintf(query, L"SELECT * FROM Reservation WHERE idReservation = %d", idReservation);
```

```
// Requête SQL pour ajouter la réservation à la base de données
SQLWCHAR query[1024];
swprintf(query, L"INSERT INTO Reservation (date, heureArr, heureDep, nbrPersonne, idClient) VALUES ('%s', %f, %f, %d, %d)",
    dateReservation.c_str(), heureArrivee, heureDepart, nbrPersonnes, idClient);
```

```
// Requête SQL pour mettre à jour la réservation dans la base de données
SQLWCHAR query[1024];
swprintf(query, L"UPDATE Reservation SET date = '%s', heureArr = %f, heureDep = %f, nbrPersonne = %d WHERE idReservation = %d",
    dateReservation.c_str(), heureArrivee, heureDepart, nbrPersonnes, idReservation);
```

```
// Requête SQL pour supprimer la réservation de la base de données
SQLWCHAR query[255];
swprintf(query, L"DELETE FROM Reservation WHERE idReservation = %d", idReservation);
```

```
// SQL query pour afficher les restaurants
SQLWCHAR query[255];
swprintf(query, L"SELECT * FROM Restaurant");
```

```
// SQL query to add restaurant to the database
SQLWCHAR query[1024];
swprintf(query, L"INSERT INTO Restaurant (nom, adresse, cuisine) VALUES ('%s', '%s', '%s')",
    nom.c_str(), adresse.c_str(), cuisine.c_str());
```

```
// SQL query pour supprimer une restaurant
SQLWCHAR query[1024];
swprintf(query, L"DELETE FROM Restaurant WHERE nom = '%s'", nomRestaurant.c_str());
```

```
// SQL query pour afficher les details d une restaurant
SQLWCHAR query[255];
swprintf(query, L"SELECT * FROM Restaurant WHERE idRestaurant = %d", idRestaurant);
```

```
// SQL query to update restaurant details in the database
SQLWCHAR updateQuery[1024];
swprintf(updateQuery, L"UPDATE Restaurant SET nom = '%s', adresse = '%s', cuisine = '%s' WHERE idRestaurant = %d",
    nouveauNom.c_str(), nouvelleAdresse.c_str(), nouvelleCuisine.c_str(), idRestaurant);
```

```
// SQL query pour afficher les restaurants du meme cuisine
SQLWCHAR query[255];
swprintf(query, L"SELECT * FROM Restaurant WHERE cuisine = '%s'", cuisine.c_str());
```

```
// SQL query pour afficher les restaurants du meme lieu
SQLWCHAR query[255];
swprintf(query, L"SELECT * FROM Restaurant WHERE adresse = '%s'", adresse.c_str());
```

```
// SQL query pour afficher les restaurants qui ont des tables qui ne sont pas reservee
SQLWCHAR query[1024];
swprintf(query, L"SELECT DISTINCT R.* FROM Restaurant R JOIN Tbla T ON R.ID = T.idRestaurant
    LEFT JOIN Reservation RV ON T.idRestaurant = RV.idTbla WHERE RV.idTbla IS NULL
    OR (RV.heureArr IS NOT NULL AND RV.heureDep IS NOT NULL)");
```

```
// Requête SQL pour ajouter la table à la base de données
SQLWCHAR query[255];
swprintf(query, L"INSERT INTO Tbla (capacite, IdRestaurant) VALUES (%d, %d)", capacite, idRestaurant);
```

```
// Requête SQL pour supprimer la table de la base de données
SQLWCHAR query[255];
swprintf(query, L"DELETE FROM Tbla WHERE idRestaurant IN (SELECT idRestaurant FROM Restaurant WHERE nom = '%s')",
    nomRestaurant.c_str());
```

```
// Requête SQL pour modifier la capacité de la table dans la base de données
SQLWCHAR query[255];
swprintf(query, L"UPDATE Tbla SET capacite = %d WHERE idTbla = %d", nouvelleCapacite, idTable);
```

```
// Requête SQL pour récupérer les tables disponibles
SQLWCHAR query[255];
swprintf(query, L"SELECT idTbla, capacite FROM Tbla WHERE idTbla NOT IN (SELECT idTbla FROM Reservation)");
```

3. Les classes (Headers) :

```
Admin.h X Client.h Login.h Personne.h Reservation.h Restaurant.h Table.h
C: > Users > othma > OneDrive > Desktop > Restauration > Restauration > Admin.h
1  #ifndef ADMIN_H
2  #define ADMIN_H
3
4  #include "Personne.h"
5  #include "Login.h"
6
7  class Admin : public Person {
8  private:
9      Login login;
10
11 public:
12     Admin();
13     Admin(int id, const string& nom, const Login& login);
14     void afficherInformations() const override;
15     void setLogin(const Login& login);
16     Login getLogin() const;
17 };
18
19 #endif // ADMIN_H
20
```

```
Admin.h Client.h X Login.h Personne.h Reservation.h Restaurant.h Table.h
C: > Users > othma > OneDrive > Desktop > Restauration > Restauration > Client.h
1  #ifndef CLIENT_H
2  #define CLIENT_H
3
4  #include "Personne.h"
5  #include <string>
6
7  using namespace std;
8
9  class Client : public Person {
10 private:
11     string email;
12
13 public:
14     Client();
15     Client(int id, string nom, string email);
16     string getEmail() const;
17     void setEmail(string email);
18     void afficherInformations() const override;
19 };
20
21 #endif // CLIENT_H
22
```



```
Admin.h Client.h Login.h X Personne.h Reservation.h Restaurant.h Table.h
C: > Users > othma > OneDrive > Desktop > Restauration > Restauration > C Login.h
1  #ifndef LOGIN_H
2  #define LOGIN_H
3
4  #include <string>
5  using namespace std;
6
7  class Login {
8  private:
9      string username;
10     string password;
11
12 public:
13     Login(string& uname,string& pword);
14     int authentifier(string& uname,string& pword);
15     string getUsername();
16     string getPassword();
17 };
18
19
20 #endif // LOGIN_H
21
```

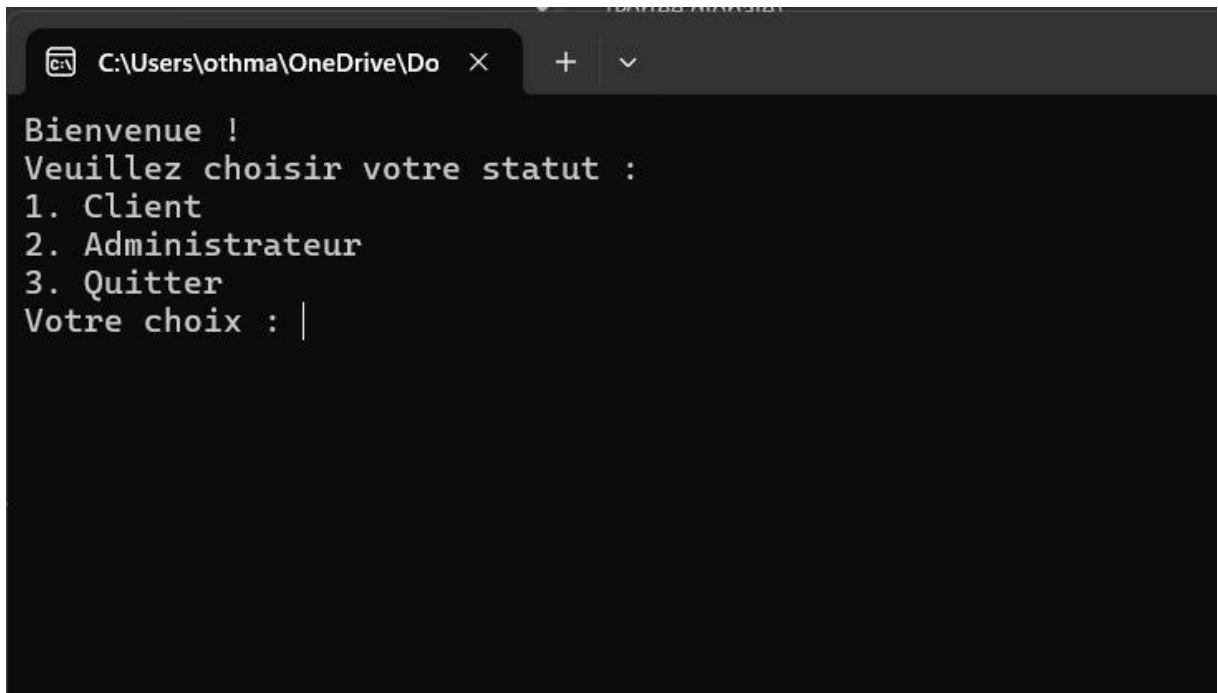
```
Admin.h Client.h Login.h Personne.h X Reservation.h Restaurant.h Table.h
C: > Users > othma > OneDrive > Desktop > Restauration > Restauration > C Personne.h
1  #ifndef PERSON_H
2  #define PERSON_H
3
4  #include <string>
5
6  using namespace std;
7
8  class Person {
9  protected:
10     int idPersonne;
11     string nom;
12
13 public:
14     Person();
15     Person(int id, string nom);
16     int getId() const;
17     string getNom() const;
18     void setId(int id);
19     void setNom(string nom);
20     virtual void afficherInformations() const;
21 };
22
23 #endif // PERSON_H
24
```

```
C Admin.h X C Client.h C Login.h C Personne.h C Reservation.h X ⚙ Settings C Restaurant.h C Table.h
C: > Users > othma > OneDrive > Desktop > Restauration > Restauration > C Reservation.h
1  #ifndef RESERVATION_H
2  #define RESERVATION_H
3
4  #include <string>
5  #include "Client.h"
6  #include "Restaurant.h"
7  #include "Table.h"
8
9  using namespace std;
10
11 class Reservation {
12 private:
13     int idReservation;
14     string date;
15     string heureArr;
16     string heureDep;
17     Client client;
18     Table table;
19     Restaurant restaurant;
20
21 public:
22     Reservation(int id, string d, string arr, string dep, const Client& c, const Table& t, const Restaurant& r);
23     int getId() const;
24     string getDate() const;
25     string getHeureArr() const;
26     string getHeureDep() const;
27     Client getClient() const;
28     Table getTable() const;
29     Restaurant getRestaurant() const;
30     void setId(int id);
31     void setDate(string d);
32     void setHeureArr(string arr);
33     void setHeureDep(string dep);
34     void setClient(const Client& c);
35     void setTable(const Table& t);
36     void setRestaurant(const Restaurant& r);
37     static void afficherReservations(const vector<Reservation>& reservations);
38     static void ajouterReservation(vector<Reservation>& reservations, const Reservation& newReservation);
39     static void annulerReservation(vector<Reservation>& reservations, int id);
40     static void modifierReservation(vector<Reservation>& reservations, int id, string newDate, string newArr, string newDep);
41 };
42
43 #endif // RESERVATION_H
```

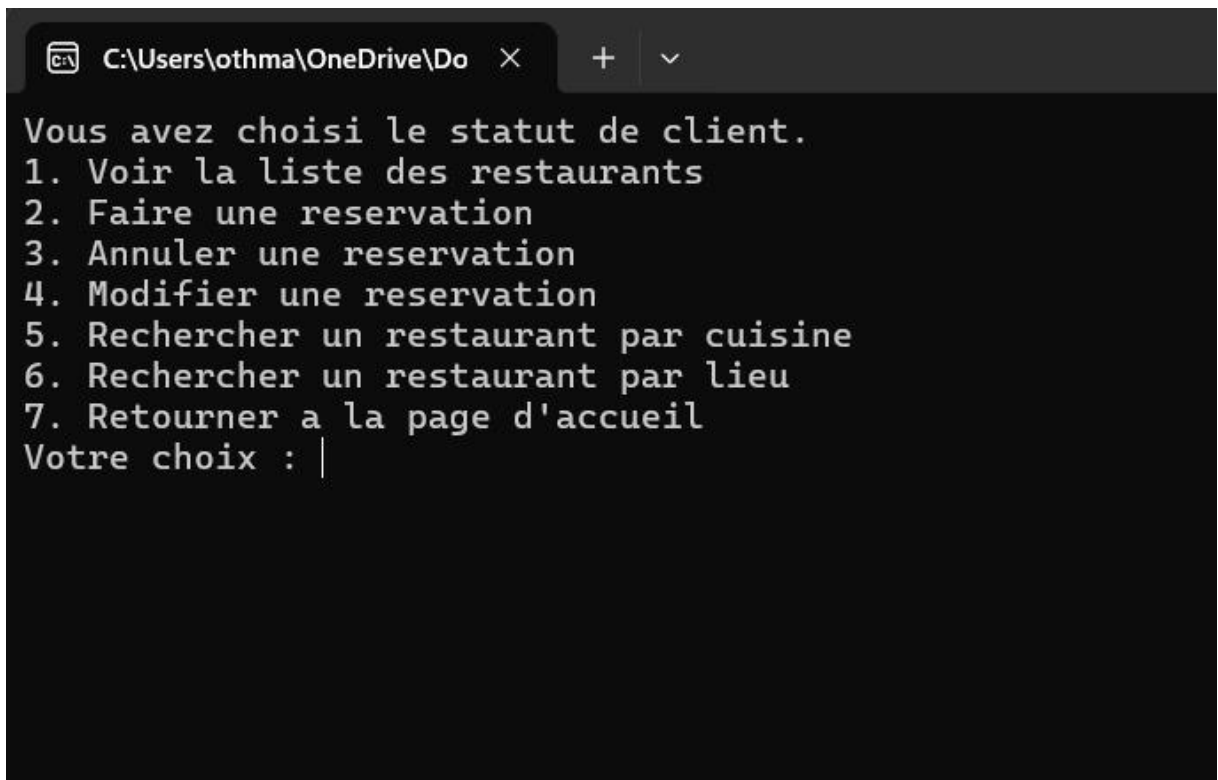
```
C Admin.h C Client.h C Login.h C Personne.h C Reservation.h C Restaurant.h X C Table.h
C: > Users > othma > OneDrive > Desktop > Restauration > Restauration > C Restaurant.h
1  #ifndef RESTAURANT_H
2  #define RESTAURANT_H
3
4  #include <string>
5  #include <vector>
6
7  using namespace std;
8
9  class Restaurant {
10 private:
11     int idRestaurant;
12     string nom;
13     string cuisine;
14     string adresse;
15     int reservations;
16
17 public:
18     Restaurant(int id, string n, string c, string a, int r);
19     int getId() const;
20     string getNom() const;
21     string getCuisine() const;
22     string getAdresse() const;
23     int getReservations() const;
24     void setId(int id);
25     void setNom(string n);
26     void setCuisine(string c);
27     void setAdresse(string a);
28     void setReservations(int r);
29     static void listRestaurants(const vector<Restaurant>& restaurants);
30     static void ajouterRestaurant(vector<Restaurant>& restaurants, const Restaurant& newRestaurant);
31     static void supprimerRestaurant(vector<Restaurant>& restaurants, int id);
32     static void modifierRestaurant(vector<Restaurant>& restaurants, int id, string newNom, string newCuisine, string newAdresse, int newReservations);
33     static void statistiques(const vector<Restaurant>& restaurants);
34     static void chercherRestaurantLieu(const vector<Restaurant>& restaurants, string adresse);
35     static void chercherRestaurantCuisine(const vector<Restaurant>& restaurants, string cuisine);
36 };
37
38 #endif // RESTAURANT_H
39
```

```
Admin.h Client.h Login.h Personne.h Reservation.h Restaurant.h Table.h X
C: > Users > othma > OneDrive > Desktop > Restauration > Restauration > C Table.h
1  #ifndef TABLE_H
2  #define TABLE_H
3
4  #include <string>
5
6  using namespace std;
7
8  class Table {
9  private:
10     int idTable;
11     int capacite;
12
13 public:
14     Table();
15     Table(int id, int capacite);
16     int getId() const;
17     int getCapacite() const;
18     void setId(int id);
19     void setCapacite(int capacite);
20     void afficherInformations() const;
21 };
22
23 #endif // TABLE_H
24
```


4. Captures d'écran de l'application :



```
C:\Users\othma\OneDrive\Do  X  +  v
Bienvenue !
Veuillez choisir votre statut :
1. Client
2. Administrateur
3. Quitter
Votre choix : |
```



```
C:\Users\othma\OneDrive\Do  X  +  v
Vous avez choisi le statut de client.
1. Voir la liste des restaurants
2. Faire une reservation
3. Annuler une reservation
4. Modifier une reservation
5. Rechercher un restaurant par cuisine
6. Rechercher un restaurant par lieu
7. Retourner a la page d'accueil
Votre choix : |
```

```
C:\Users\othma\OneDrive\Do  X + v
Liste des restaurants :
1. Dorsia RAK (Cuisine: Francaise, Localisation: Marrakech)
2. Dorsia PAR (Cuisine: Italienne, Localisation: Paris)
3. Dorsia LND (Cuisine: Chinoise, Localisation: London)
Appuyez sur Entrée pour continuer...
```

```
C:\Users\othma\OneDrive\Do  X + v
Effectuer une reservation
Liste des restaurants :
1. Dorsia RAK (Cuisine: Francaise, Localisation: Marrakech)
2. Dorsia PAR (Cuisine: Italienne, Localisation: Paris)
3. Dorsia LND (Cuisine: Chinoise, Localisation: London)
Choisissez un restaurant en entrant son nom : 1
Entrez votre nom : othmane
Entrez la date de reservation (jj/mm/aaaa) : 12/05/2024
Entrez l'heure de reservation (hh:mm) : 22:00
Reservation effectuee avec succes ! Identifiant de reservation : RES-1
Appuyez sur Entrée pour continuer...
```

```
C:\Users\othma\OneDrive\Do  X + v
Entrez l'identifiant de la reservation que vous souhaitez annuler : RES-1
```

```
C:\Users\othma\OneDrive\Do  X + v
Reservation annulee avec succes !
Appuyez sur Entrée pour continuer...
```

C:\Users\othma\OneDrive\Do X + v
Entrez le lieu que vous recherchez : Marrakech|

C:\Users\othma\OneDrive\Do X + v
Resultats de la recherche pour le lieu Marrakech:
Dorsia RAK (Localisation: Marrakech)
Appuyez sur Entrée pour continuer...|

VI. Remerciements

Nous tenons à exprimer nos sincères remerciements à nos enseignants de **l'École Marocaine des Sciences de l'Ingénieur** pour leur précieuse contribution à notre formation et à la réalisation de ce projet.

Un grand merci à Professeur **SAFSOUF Yassine** pour son enseignement approfondi du langage C++, qui a posé les bases essentielles pour le développement de notre application.

Nous exprimons également notre gratitude envers Professeur **SNIBA Farah** pour sa guidance experte dans le domaine de SQL, qui nous a permis d'intégrer efficacement les interactions avec la base de données dans notre application.

Nous remercions chaleureusement Professeur **HAZMAN Chaimaa** pour ses précieux enseignements sur les concepts de conception, qui ont enrichi notre compréhension et notre approche de la conception logicielle.

Nos remerciements vont également à tous ceux qui ont contribué de près ou de loin à la réalisation de ce projet, ainsi qu'à nos camarades de classe pour leur soutien et leur collaboration.

Enfin, un merci spécial à nos familles et proches pour leur soutien inconditionnel tout au long de ce parcours académique.

VII. Conclusion

La réalisation de ce projet de développement d'une application de gestion des tables pour un groupe de restaurants a été une expérience enrichissante et stimulante pour notre équipe. Ce projet nous a permis d'appliquer les connaissances acquises lors de notre formation à **l'École Marocaine des Sciences de l'Ingénieur** dans un contexte pratique et concret.

Nous sommes fiers du résultat final, une application fonctionnelle qui répond aux besoins des clients et des administrateurs des restaurants. L'application offre une interface conviviale pour les utilisateurs, facilitant la réservation de tables et la gestion des restaurants.

Ce projet nous a également permis d'acquérir de nouvelles compétences techniques, notamment dans le développement d'applications web, l'intégration de bases de données et la programmation en C++. Nous avons appris à travailler en équipe, à résoudre des problèmes de manière collaborative et à respecter des délais serrés.

Bien que notre application soit complète dans sa version actuelle, nous reconnaissons qu'il existe toujours des opportunités d'amélioration. Dans l'avenir, nous pourrions envisager d'ajouter de nouvelles fonctionnalités, d'améliorer la sécurité et la performance de l'application, ou d'explorer d'autres technologies pour répondre aux besoins évolutifs de l'industrie de la restauration.

En conclusion, ce projet a été une expérience précieuse qui a renforcé notre compréhension des principes de développement logiciel et nous a préparés à relever de nouveaux défis dans notre future carrière professionnelle.