Projet de Gestion des Clients avec **MySQL, C# et React**

Ce projet présente une architecture de gestion des clients dans une base de données MySQL, avec une interface en React.js, un Backend en C#. Les parties importantes du projet sont les suivantes :

1. Stockage des données clients dans une base **MySQL**.
2. Usage d'une API pour exposer ces données avec **C#/.NET**.
3. Vérifier l'API et la structure des données avec **Postman**.
4. Consommer l'API et afficher les données dans une application **React.js**.

- **Présentation détaillée de chacune des parties (BD, Front end et Back end) avec une capture du résultat final**

1. Base de Données : MySQL

- Création d'un schéma et d'une table customers pour stocker les informations clients.
- Utilisation de **requêtes SQL** pour l'insertion et manipulation des données.

```
********** step 1 : Create Schema for master bidabi **********
*/

CREATE SCHEMA master_bidabi;


/*
********** Create our first table : customers informations **********
*/

CREATE TABLE master_bidabi.customers
(
    name VARCHAR(50),
    surname VARCHAR(50),
    city VARCHAR (30),
    age INT,
    Postcal_code INT,
    job VARCHAR(45)
)
;


/*
********** Read the customers table : empty table **********
*/

SELECT * FROM master_bidabi.customers;


/*
********** fill the table with rows **********
*/

INSERT INTO master_bidabi.customers (name, surname, city, age, Postcal_code,job )
VALUES ("AAanne", "presi", "Paris", 24, 75010, "presidente"),
       ("jean martin martin", "lobo", "Rennes", 40, 50500, "patissier"),
       ("AAanne", "presi", "Paris", 24, 75010, "presidente"),
       ("JCean martin martin", "lgobo", "Rennhges", 40, 50500, "jpatissier"),
       ("VAAanne", "Cpresi", "Paris", 244, 754010, "jpresiyydente"),
       ("BJCean martin Fmartin", "lobo", "Renynes", 40, 504500, "patiossier")
;


/*
********** add additional rows to the table **********
```

2. Partie Backend en **C#**

Connexion et récupération des données dans la base de donnée, conception d'une API

- Connexion à MySQL via **MySqlConnection**.
- Requêtes SQL via **MySqlCommand** pour récupérer et afficher les données.

projet1_avec_mysqlWorkbench.cs | App.js M | index.js M | ClientList.css U | Program.cs 4

mysqlworkbench > Program.cs > Program > Main

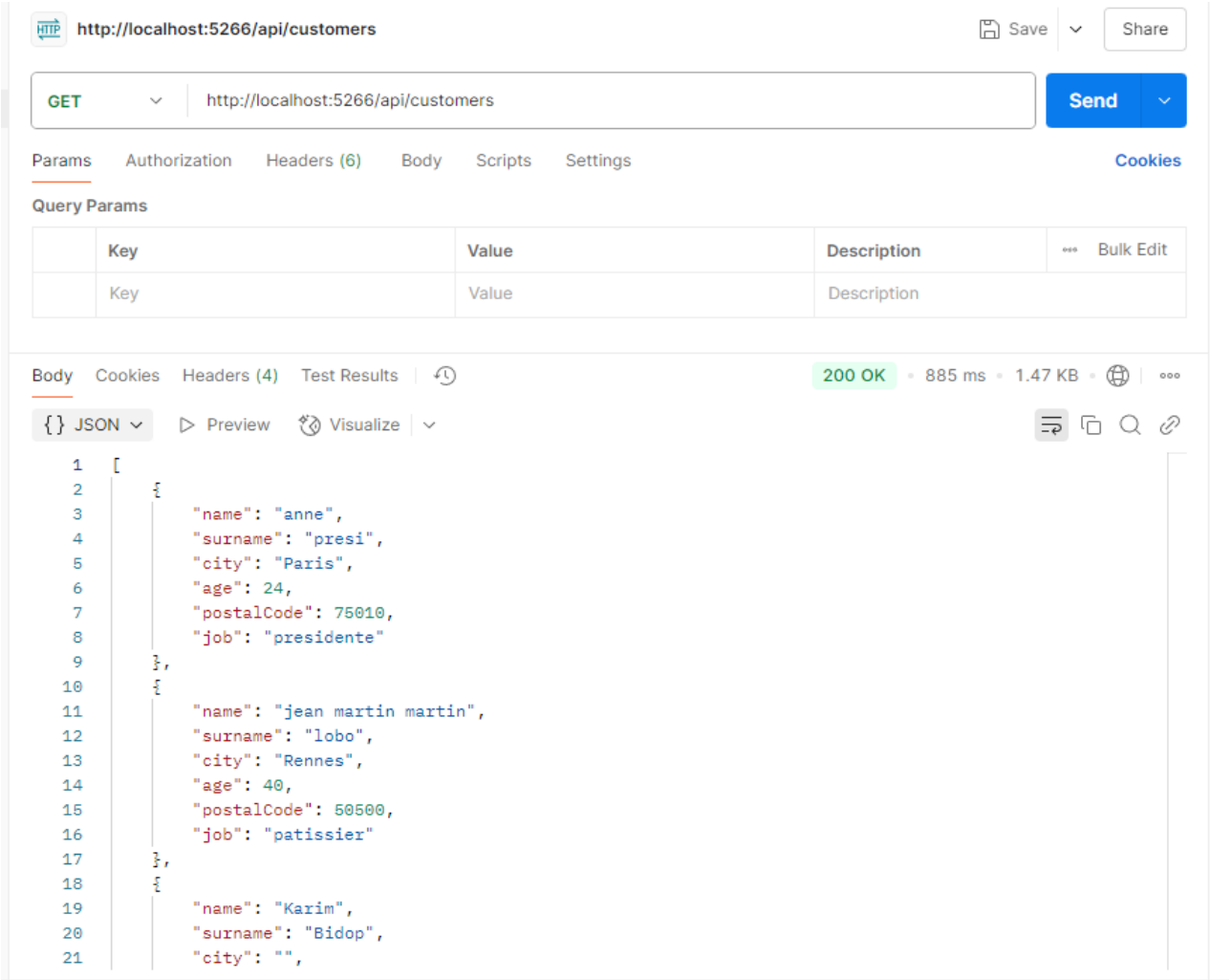
```
1   using System;
2   using MySql.Data.MySqlClient;
3
4   namespace DataExtractionTool {
        0 references
5       class Program {
            0 references
6           static void Main(string[] args) {
7               Console.WriteLine("Connecting to the database...");
8
9               string connectionString = "Server=127.0.0.1;Port=3306;uid=root;Password=azerty;Database=master_bidabi;SslMode=No
10
11              using (MySqlConnection connection = new MySqlConnection(connectionString)) {
12                  try {
13                      connection.Open();
14                      Console.WriteLine("Connection successful.");
15
16  // Query to extract data from customers table
17                      string query = "SELECT name, surname, city, age, Postcal_code, job FROM customers";
18
19                      using (MySqlCommand command = new MySqlCommand(query, connection)) {
20                          using (MySqlDataReader reader = command.ExecuteReader()) {
21                              Console.WriteLine("Data from customers table:");
22                              Console.WriteLine("-----------------------------------------------------");
23                              Console.WriteLine("Name\tSurname\tCity\tAge\tPostal Code\tJob");
24
25                              // Reading and displaying data
26                              while (reader.Read()) {
27                                  string name = reader["name"].ToString();
28                                  string surname = reader["surname"].ToString();
29                                  string city = reader["city"].ToString();
30                                  int age = Convert.ToInt32(reader["age"]);
31                                  int postalCode = Convert.ToInt32(reader["Postcal_code"]);
32                                  string job = reader["job"].ToString();
33
34                                  Console.WriteLine($"{name}\t{surname}\t{city}\t{age}\t{postalCode}\t{job}");
35                              }
36                          }
37                      }
38                  } catch (Exception ex) {
39                      Console.WriteLine("Connection failed: " + ex.Message);
40                  }
41              }
```

PROBLEMS 4 | OUTPUT | DEBUG CONSOLE | TERMINAL | PORTS | COMMENTS

```
PS C:\Users\kamel\C#\mysqlworkbench> cd .\MyApi\
PS C:\Users\kamel\C#\mysqlworkbench\MyApi> dotnet run
Building...
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: http://localhost:5266
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Development
info: Microsoft.Hosting.Lifetime[0]
      Content root path: C:\Users\kamel\C#\mysqlworkbench\MyApi
info: Microsoft.Hosting.Lifetime[0]
info: Microsoft.Hosting.Lifetime[0]
info: Microsoft.Hosting.Lifetime[0]
info: Microsoft.Hosting.Lifetime[0]
info: Microsoft.Hosting.Lifetime[0]
      Content root path: C:\Users\kamel\C#\mysqlworkbench\MyApi
```

3. Vérification de l'API avec **Postman**

L'objectif était de vérifier que les données étaient bien retournées sous forme de **JSON**



4. Frontend : **React.js**

Requête API avec **Axios**.

- Affichage dynamique des données sous forme de tableau HTML.
- Gestion des états avec useState et useEffect.

```
mysqlworkbench > my-app > src > JS App.js > ...
import React, { useEffect, useState } from 'react';
import axios from 'axios';

function App() {
    const [customers, setCustomers] = useState([]);

    useEffect(() => {
        axios.get("http://localhost:5266/api/customers")
            .then(res => {
                console.log("Données reçues:", res.data);
                setCustomers(res.data);
            })
            .catch(error => console.error("Erreur:", error));
    }, []);
```

```
return (
    <div>
        <h1>Liste des Clients</h1>
        <table border="1">
            <thead>
                <tr>
                    <th>Nom</th>
                    <th>Prénom</th>
                    <th>Ville</th>
                    <th>Âge</th>
                    <th>Code Postal</th>
                    <th>Profession</th>
                </tr>
            </thead>
            <tbody>
                {customers.length > 0 ? (
                    customers.map((customer, index) => (
                        <tr key={index}>
                            <td>{customer.name}</td>
                            <td>{customer.surname}</td>
                            <td>{customer.city}</td>
                            <td>{customer.age}</td>
                            <td>{customer.postalCode}</td>
                            <td>{customer.job}</td>
                        </tr>
                    ))
                ) : (
                    <tr>
                        <td colSpan="6">Aucune donnée disponib
                    </tr>
```

5. **Résultat Final du Projet**

Après l'implémentation complète, le projet permet d'afficher la liste des clients sous forme de tableau dont le rendu final est en dessous :

Liste des Clients

NOM	PRÉNOM	VILLE	ÂGE	CODE POSTAL	PROFESSION
anne	presi	Paris	24	75010	presidente
jean martin martin	lobo	Rennes	40	50500	patissier
Karim	Bidop		80	60604	Taxi driver
AAanne	presi	Paris	24	75010	presidente
jean martin martin	lobo	Rennes	40	50500	patissier
AAanne	presi	Paris	24	75010	presidente
JCean martin martin	lobo	Rennes	40	50500	patissier
AAanne	presi	Paris	24	75010	presidente
jean martin martin	lobo	Rennes	40	50500	patissier
AAanne	presi	Paris	24	75010	presidente
JCean martin martin	lgobo	Rennhges	40	50500	jpatissier
VAAanne	Cpresi	Paris	244	754010	jpresiyydente
BJCean martin Fmartin	lobo	Renynes	40	504500	patiossier

Ce projet montre une application concrète des concepts de Bases de données, développement d'applications. Il met en avant une approche full-Stack MVC associant SQL pour la gestion des données, C# pour la logique métier et React.js pour l'interface utilisateur.