ENSA Ecole Nationale des Sciences Appliquées BERRECHID

جامعة الحسن الأول
UNIVERSITÉ HASSAN 1er

Rapport de mini projet

Système de Facturation Client

Réalisé par :

BEN AHMED AHMED
AIT ELHAJ GHIZLANE

Encadré par :

Pr. Jaouad Danane

1ere année ISIBD 24/25

Contents

1 Introduction et Contexte

Ce projet vise à simuler un système de facturation pour n'importe quel service. Il permet d'ajouter des services consommés (nom, prix, quantité) et de générer une facture détaillée avec un total final. Ce projet a été développé en collaboration avec un collègue, et il a pour but de démontrer l'intégration de fonctionnalités liées à la gestion des services et à la facturation.

2 Objectifs

Les objectifs principaux de ce projet étaient les suivants :

- Ajouter des services consommés avec des informations spécifiques (nom, prix, quantité).
- Calculer et afficher une facture détaillée.
- Ajouter une fonctionnalité de catégorisation des services pour améliorer l'expérience utilisateur.
- Diviser le programme en deux menus distincts : un pour l'administrateur et un pour le client.

3 Outils et Technologies

Le projet a été développé avec les outils et technologies suivants :

- **IDE**: CLion (utilisé par AHMED BENAHMED) et VSCode (utilisé par AITELHAJ GHIZLANE).
- **Bibliothèques C**:
  - #include ¡windows.h¿
  - #include ¡sqlext.h¿
  - #include ¡sqltypes.h¿
  - #include ¡sql.h¿
  - #include ¡stdio.h¿
  - #include ¡stdlib.h¿

- **Base de données**: MySQL Workbench pour la verification et gestion de la base de données.

4 Préparation de l'environnement (Si l'on travaille avec du code qui intègre une base de données)

Pour configurer un projet qui utilise ODBC avec CMake, suivez les étapes suivantes :

4.1 Installation d'ODBC

1. Téléchargez et installez le pilote MySQL ODBC approprié pour votre système depuis le site officiel MySQL. (Dans notre cas 8.1) `https://downloads.mysql.com/archives/c-odbc/`.
    - Utilisez le fichier MSI pour une installation facile.
2. Vérifiez que le pilote est installé correctement en ouvrant le Gestionnaire ODBC et en consultant l'onglet Drivers .
3. Configurez une Source de Données (DSN) via le Gestionnaire ODBC.
    - Assurez-vous que le test DSN passe avec succès.

4.2 Inclusion dans un Projet CMake

1. Assurez-vous que les bibliothèques et fichiers d'en-tête nécessaires à ODBC sont disponibles sur votre système.
2. Ajoutez les instructions suivantes à votre fichier `CMakeLists.txt` :

```
find_package(ODBC REQUIRED)

target_link_libraries(${PROJECT_NAME} PRIVATE ODBC::ODBC)
```

3. Ajoutez les fichiers sources contenant votre code reliant la base de données au projet dans `CMakeLists.txt`.

5 Conception du Système

Dans la phase de conception, nous avons décidé d'ajouter une fonctionnalité de catégorisation des services, bien qu'elle ne fût pas initialement demandée. Cette fonctionnalité permet de classer les services en différentes catégories afin d'améliorer l'expérience utilisateur et rendre l'interface plus intuitive.

Nous avons également divisé le programme en deux menus distincts :

- **Menu Admin** : Permet à l'administrateur d'ajouter des services, afficher les services existants et ajouter des catégories.
- **Menu Client** : Permet au client de visualiser les services disponibles et de générer une facture.

6 Méthodologie

6.1 Répartition des tâches

- Réunion initiale : Les deux partenaires ont revu les exigences du projet et créé le MCD pour la conception du schéma de base de données.
- AITELHAJ GHIZLANE : Développement du programme CLI de base et de sa logique.
- BEN AHMED AHMED : Intégration de la base de données, y compris la connexion avec MySQL et la création des fonctions nécessaires pour l'initialisation des tables, l'insertion des données et les requêtes.

6.2 Phases de développement

1. Développement du programme CLI.
2. Intégration des fonctionnalités de base de données.
3. Validation des fonctionnalités CLI indépendantes de la base de données avant l'intégration finale.

7 Implémentation

7.1 Vue d'ensemble du code

Le programme est structuré autour de deux modules principaux :

- Fonctionnalité CLI : Gère l'interaction avec l'utilisateur et la logique pour gérer les catégories, les services et les factures.
- Opérations sur la base de données : Assure le stockage persistant et la récupération des données via MySQL en utilisant ODBC.

7.2 Fonctions clés

- `connect_to_database` : Établit la connexion à la base de données.
- `init_tables` : Crée les tables si elles n'existent pas.
- `addService` : Ajoute un service sélectionné à la facture en cours.
- `genererFacture` : Génère une facture détaillée avec un coût total.
- `insert_category` : Insère une nouvelle catégorie dans la base de données.
- `insert_service` : Insère un nouveau service dans la base de données.
- `get_entity_count` : Récupère le nombre d'entités d'un type spécifique dans la base de données.
- `get_categories` : Récupère toutes les catégories de la base de données.
- `insert_default_categories` : Insère les catégories par défaut si aucune catégorie n'existe.
- `insert_default_admin` : Insère l'administrateur par défaut si aucun administrateur n'existe.
- `show_tables` : Affiche toutes les tables de la base de données.
- `get_last_inserted_id` : Récupère le dernier identifiant inséré.
- `menu_admin` : Affiche le menu d'administration pour gérer les services et les catégories.

- `menu_client` : Affiche le menu client pour consulter les services et générer des factures.
- `addCategorie` : Ajoute une nouvelle catégorie.
- `afficherServices` : Affiche tous les services disponibles.
- `genererFacture` : Génère la facture finale basée sur les services sélectionnés.

7.3 Extraits de code pour la partie CLI uniquement

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct categorie {
        int id_categorie;
        char nom[50];
} categorie;

typedef struct service {
        int id_service;
        char nom_service[50];
        double prix_service;
        categorie categorie_service;
} service;

typedef struct client {
        int id_client;
        char nom_client[50];
        double montant_total;
        char details_facture[500];
} client;

#define MAX_SERVICES 100
#define MAX_CATEGORIES 10
#define MAX_CLIENTS 2

service services[MAX_SERVICES];
categorie categories[MAX_CATEGORIES] = {
        {1, "Coiffure"},
        {2, "Electricite"},
        {3, "Etudes"},
        {4, "Restauration"},
```

```
        {5, "Informatique"}
};
client clients[MAX_CLIENTS];
int count = 0;
int categorie_count = 5;
int client_count = 0; // Nombre de clients enregistr s

void menu_admin();
void menu_client();
void addService();
void afficherServices();
void afficherCategories();
void addCategorie();
void genererFacture(int id_client);
void afficherFacture(double montant, char *details, char *
    nom_client);
int findClientByName(char *nom_client);
void addClient();
void supprimerService();
void supprimerCategorie();

int main() {
        int choix;
        printf("Bonjour !\n");

        while (1) {
                printf("\nMerci de selectionner si vous etes
                    admin ou client :\n");
                printf("1. Admin\n");
                printf("2. Client\n");
                printf("3. Quitter\n");
                printf("Votre choix : ");

                if (scanf("%d", &choix) != 1) {
                        printf("Entree invalide. Veuillez
                            reessayer.\n");
                        while (getchar() != '\n');
                        continue;
                }

                switch (choix) {
                        case 1: menu_admin(); break;
                        case 2: menu_client(); break;
                        case 3:
                        printf("Au revoir !\n");
```

```
                        return 0;
                        default:
                        printf("Choix invalide. Veuillez reessayer.\n");
                }
        }
        return 0;
}
```

Listing 1: Code pour la fonction main

```
void menu_admin() {
        int choix;
        while (1) {
                printf("\nMenu Admin :\n");
                printf("1. Ajouter un service\n");
                printf("2. Afficher tous les services\n");
                printf("3. Ajouter une categorie\n");
                printf("4. Supprimer un service\n");
                printf("5. Supprimer une categorie\n");
                printf("6. Retour au menu principal\n");
                printf("Votre choix : ");

                if (scanf("%d", &choix) != 1) {
                        printf("Entree invalide. Veuillez reessayer.\n");
                        while (getchar() != '\n');
                        continue;
                }

                switch (choix) {
                        case 1: addService(); break;
                        case 2: afficherServices(); break;
                        case 3: addCategorie(); break;
                        case 4: supprimerService(); break;
                        case 5: supprimerCategorie(); break;
                        case 6: return;
                        default:
                        printf("Choix invalide. Veuillez reessayer.\n");
                }
        }
}
```

Listing 2: Code pour le menu administrateur

```c
void menu_client() {
        int choix;
        char nom_client[50];

        // Demande du nom du client
        printf("Veuillez entrer votre nom : ");
        scanf("%s", nom_client);

        int id_client = findClientByName(nom_client);
        if (id_client == -1) {
                printf("Client introuvable. Enregistrement d'un nouveau client...\n");
                addClient(nom_client);
                id_client = client_count - 1;  // Nouvel ID du client
        }

        do {
                printf("\nMenu Client :\n");
                printf("1. Voir tous les services\n");
                printf("2. Generer une facture\n");
                printf("3. Retour au menu principal\n");
                printf("Votre choix : ");
                if (scanf("%d", &choix) != 1) {
                        printf("Entree invalide. Veuillez reessayer.\n");
                        while (getchar() != '\n');
                        continue;
                }

                switch (choix) {
                        case 1:
                        afficherServices();
                        break;
                        case 2:
                        genererFacture(id_client); // Passe l'ID du client    la fonction
                        break;
                        case 3:
                        printf("Retour au menu principal...\n");
                        break;
                        default:
                        printf("Choix invalide. Veuillez reessayer.\n");
```

```
                }
        } while (choix != 3);
}
```

Listing 3: Code pour le menu client

```
void supprimerService() {
        int id_service;
        printf("\nEntrez l'ID du service a supprimer : ");
        if (scanf("%d", &id_service) != 1) {
                printf("Entree invalide. Veuillez reessayer.\n");
                while (getchar() != '\n');
                return;
        }

        int found = 0;
        for (int i = 0; i < count; i++) {
                if (services[i].id_service == id_service) {

                        for (int j = i; j < count - 1; j++) {
                                services[j] = services[j + 1];
                        }
                        count--;
                        printf("Service avec ID %d supprim avec succ s.\n", id_service);
                        found = 1;
                        break;
                }
        }

        if (!found) {
                printf("Service introuvable avec l'ID %d.\n", id_service);
        }
}

void supprimerCategorie() {
        int id_categorie;
        printf("\nEntrez l'ID de la categorie a supprimer : ");
        if (scanf("%d", &id_categorie) != 1) {
                printf("Entree invalide. Veuillez reessayer.\n");
                while (getchar() != '\n');
```

```
35                  return;
36          }
37
38          // V rification de l'existence de la cat gorie
39          int found = 0;
40          for (int i = 0; i < categorie_count; i++) {
41                  if (categories[i].id_categorie ==
                        id_categorie) {
42                      for (int j = i; j < categorie_count -
                            1; j++) {
43                          categories[j] = categories[j
                                + 1];
44                      }
45                      categorie_count--;
46                      printf("Categorie avec ID %d
                            supprim e avec succ s.\n",
                            id_categorie);
47                      found = 1;
48                      break;
49                  }
50          }
51
52          if (!found) {
53                  printf("Categorie introuvable avec l'ID %d.\n
                        ", id_categorie);
54          }
55  }
56
57  void addCategorie() {
58          if (categorie_count >= MAX_CATEGORIES) {
59                  printf("Le nombre maximum de categories a ete
                         atteint.\n");
60                  return;
61          }
62
63          printf("\nAjouter le nom de la categorie : ");
64          while (getchar() != '\n');
65          fgets(categories[categorie_count].nom, sizeof(
                categories[categorie_count].nom), stdin);
66          categories[categorie_count].nom[strcspn(categories[
                categorie_count].nom, "\n")] = '\0';
67
68          categories[categorie_count].id_categorie =
                categorie_count + 1;
69          printf("Categorie '%s' ajoutee avec succes !\n",
```

```
            categories[categorie_count].nom);
70          categorie_count++;
71  }
72
73  void addService() {
74          if (count >= MAX_SERVICES) {
75                  printf("Le nombre maximum de services a ete
                        atteint.\n");
76                  return;
77          }
78
79          printf("\nAjouter le nom du service : ");
80          while (getchar() != '\n');
81          fgets(services[count].nom_service, sizeof(services[
                count].nom_service), stdin);
82          services[count].nom_service[strcspn(services[count].
                nom_service, "\n")] = '\0';
83
84          printf("Ajouter le prix du service : ");
85          if (scanf("%lf", &services[count].prix_service) != 1)
                 {
86                  printf("Entree invalide pour le prix. Service
                         non ajoute.\n");
87                  while (getchar() != '\n');
88                  return;
89          }
90
91          while (1) {
92                  printf("Veuillez choisir une categorie dans
                        la liste ci-dessous :\n");
93                  afficherCategories();
94
95                  int id_categorie;
96                  printf("Entrez l'ID de la categorie (ou 0
                        pour ajouter une nouvelle categorie) : ");
97                  if (scanf("%d", &id_categorie) != 1) {
98                          printf("Entree invalide. Veuillez
                                reessayer.\n");
99                          while (getchar() != '\n');
100                         continue;
101                 }
102
103                 if (id_categorie == 0) {
104                         addCategorie();
105                         services[count].categorie_service =
```

```
                        categories[categorie_count - 1];
106                     break;
107             }
108
109             int categorie_trouvee = 0;
110             for (int i = 0; i < categorie_count; i++) {
111                     if (categories[i].id_categorie ==
                            id_categorie) {
112                             services[count].
                                    categorie_service =
                                    categories[i];
113                             categorie_trouvee = 1;
114                             break;
115                     }
116             }
117
118             if (categorie_trouvee) {
119                     break;
120             } else {
121                     printf("Categorie invalide. Veuillez
                            reessayer.\n");
122             }
123     }
124
125     services[count].id_service = count + 1;
126     printf("Service ajoute avec succes !\n");
127     count++;
128 }
129
130 void afficherServices() {
131     if (count == 0) {
132             printf("Aucun service disponible.\n");
133             return;
134     }
135
136     for (int i = 0; i < count; i++) {
137             printf("ID : %d | Nom : %s | Prix : %.2f |
                    Categorie : %s\n",
138             services[i].id_service,
139             services[i].nom_service,
140             services[i].prix_service,
141             services[i].categorie_service.nom);
142     }
143 }
144
```

```
145 void afficherCategories() {
146         printf("\nListe des categories disponibles :\n");
147         for (int i = 0; i < categorie_count; i++) {
148                 printf("ID : %d | Nom : %s\n", categories[i].
                        id_categorie, categories[i].nom);
149         }
150 }
```

Listing 4: Code pour la gestion des services et categories

```
1  int findClientByName(char *nom_client) {
2          for (int i = 0; i < client_count; i++) {
3                  if (strcmp(clients[i].nom_client, nom_client)
                           == 0) {
4                          return i; //retourne l'indice du
                               client
5                  }
6          }
7          return -1; // Client non trouv
8  }
9
10 void addClient(char *nom_client) {
11         if (client_count >= MAX_CLIENTS) {
12                 printf("Le nombre maximum de clients a ete
                        atteint.\n");
13                 return;
14         }
15
16         clients[client_count].id_client = client_count + 1;
17         strcpy(clients[client_count].nom_client, nom_client);
18         clients[client_count].montant_total = 0.0;
19         clients[client_count].details_facture[0] = '\0';
20         printf("Client '%s' ajoute avec succes !\n",
                nom_client);
21         client_count++;
22 }
```

Listing 5: Code pour la gestion des clients

```
1  void genererFacture(int id_client) {
2          double montant_total = 0.0;
3          int id_service;
4          char details_facture[500] = "";
5
6          printf("\nGeneration de facture pour %s\n", clients[
                id_client].nom_client);
```

```
7           printf("\nListe des services disponibles :\n");
8           afficherServices();
9
10          do {
11                  printf("Entrez l'ID du service que vous
                        souhaitez ajouter (0 pour terminer) : ");
12                  if (scanf("%d", &id_service) != 1) {
13                          printf("Entree invalide. Veuillez
                                reessayer.\n");
14                          while (getchar() != '\n');
15                          continue;
16                  }
17
18                  if (id_service == 0) {
19                          break;
20                  }
21
22                  int service_trouve = 0;
23                  for (int i = 0; i < count; i++) {
24                          if (services[i].id_service ==
                                id_service) {
25                                  montant_total += services[i].
                                        prix_service;
26                                  sprintf(details_facture +
                                        strlen(details_facture),
27                                  "Service: %s | Prix: %.2f\n",
28                                  services[i].nom_service,
                                        services[i].prix_service);
29                                  strcat(details_facture, "
                                        +-------------------------+\
                                        n");
30                                  service_trouve = 1;
31                                  break;
32                          }
33                  }
34
35                  if (!service_trouve) {
36                          printf("Service introuvable. Essayez
                                avec un ID valide.\n");
37                  }
38
39          } while (id_service != 0);
40
41          afficherFacture(montant_total, details_facture,
                clients[id_client].nom_client);
```

```
42 }
43
44 void afficherFacture(double montant, char *details, char *
       nom_client) {
45         printf("\n+-------- Facture ----------+\n");
46         printf("+--------------------------+\n");
47         printf("+--------%s--------+\n", nom_client);
48         printf("+--------------------------+\n");
49         printf("%s\n", details);
50         printf("\n+Total:          +%.2f+\n", montant);
51         printf("+--------------------------+\n");
52 }
```

Listing 6: Code pour generer une facture

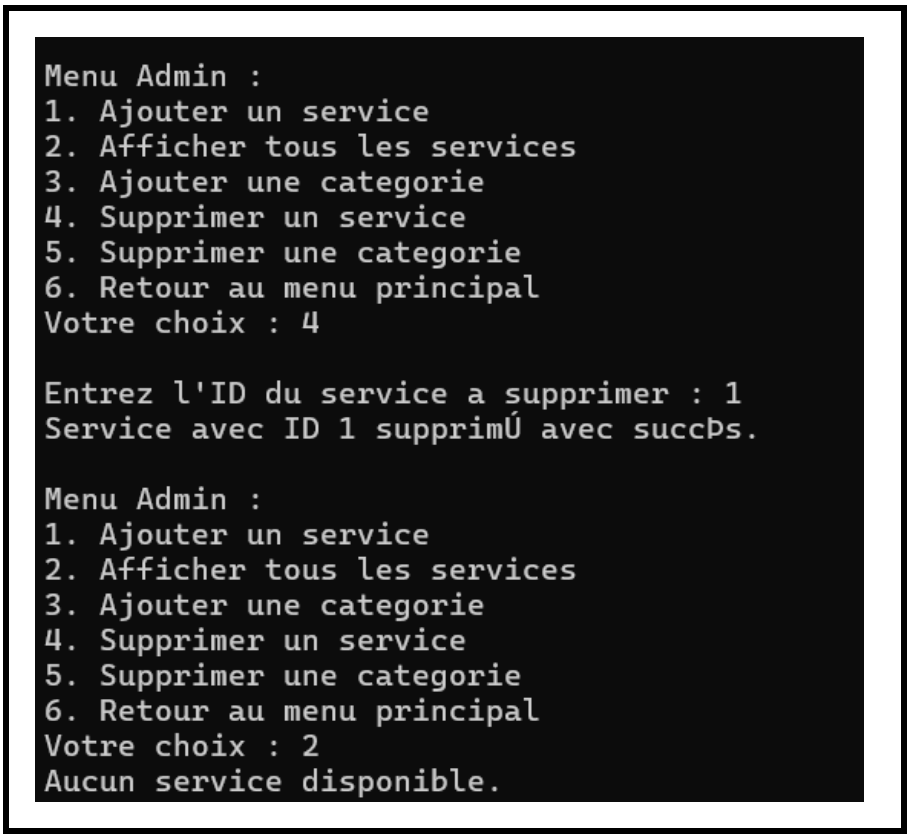
7.4 L'interface

7.5 Partie Admin

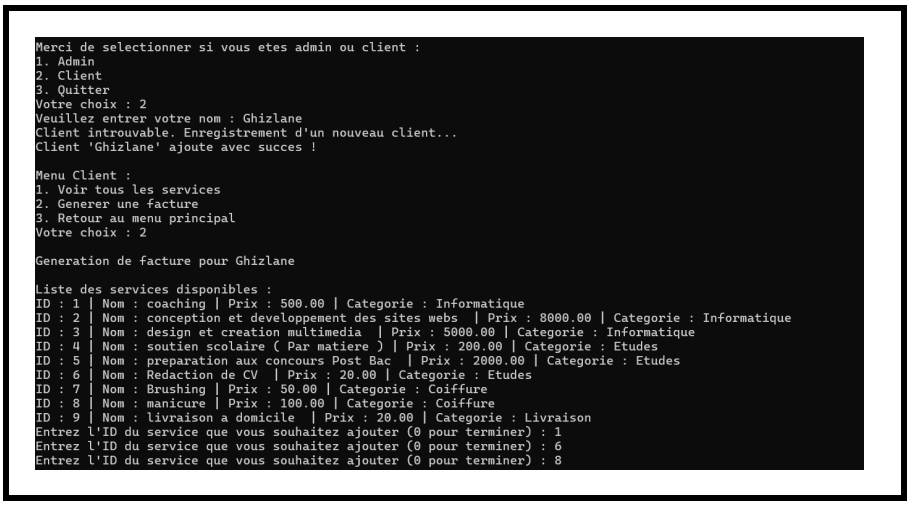
```
Bonjour !

Merci de selectionner si vous etes admin ou client :
1. Admin
2. Client
3. Quitter
Votre choix : 1

Menu Admin :
1. Ajouter un service
2. Afficher tous les services
3. Ajouter une categorie
4. Supprimer un service
5. Supprimer une categorie
6. Retour au menu principal
Votre choix :
```

on choisit d'acceder en tant que Admin

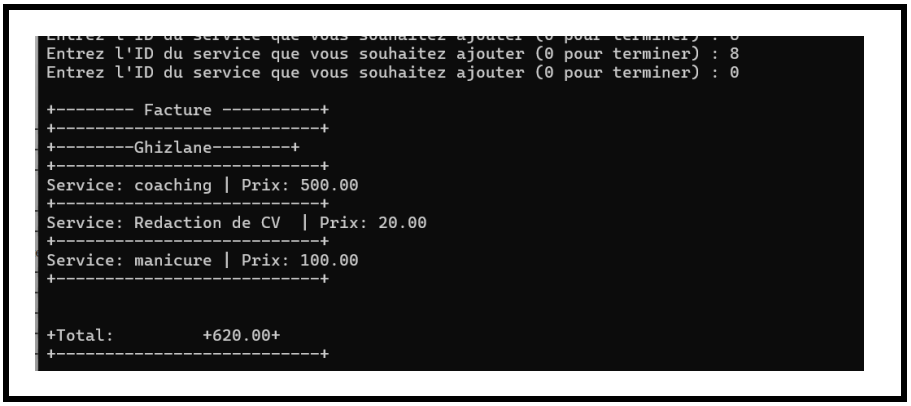
on choisit d'ajouter un service // on peut meme ajouter une nouvelle categorie si elle n'existe pas



on peut meme supprimer un service / categorie



on accede maintenant en tant que client pour voir les services et avoir une facture



on prend notre facture

```
Menu Client :
1. Voir tous les services
2. Generer une facture
3. Retour au menu principal
Votre choix : 3
Retour au menu principal...

Merci de selectionner si vous etes admin ou client :
1. Admin
2. Client
3. Quitter
Votre choix : 3
Au revoir !
```

Et on quitte ....

7.6 Extraits de code pour la partie base de donnees uniquement

Les extraits de code suivants illustrent des parties cles de l'integration de la base de donnees :

```
1  void connect_to_database(DBContext* db_context) {
2      // Allocate environment handle
3      db_context->retcode = SQLAllocHandle(SQL_HANDLE_ENV,
           SQL_NULL_HANDLE, &db_context->env);
4      check_error(db_context->retcode, db_context->env,
           SQL_HANDLE_ENV, "connect_to_database - SQLAllocHandle"
           );
5
6      // Set the ODBC version
7      db_context->retcode = SQLSetEnvAttr(db_context->env,
           SQL_ATTR_ODBC_VERSION, (SQLPOINTER)SQL_OV_ODBC3, 0);
8      check_error(db_context->retcode, db_context->env,
           SQL_HANDLE_ENV, "connect_to_database - SQLSetEnvAttr")
           ;
9
10     // Allocate connection handle
11     db_context->retcode = SQLAllocHandle(SQL_HANDLE_DBC,
           db_context->env, &db_context->dbc);
12     check_error(db_context->retcode, db_context->dbc,
           SQL_HANDLE_DBC,  "connect_to_database - SQLAllocHandle
           ");
13
```

```
    // Connect to the database
    SQLCHAR connStr[] = "DRIVER={C:\\Program Files\\MySQL\\
       Connector ODBC 8.1\\myodbc8w.dll};SERVER=localhost;
       PORT=3306;DATABASE=test;USER=root;PASSWORD=;";
    db_context->retcode = SQLDriverConnect(db_context->dbc,
       NULL, connStr, SQL_NTS, NULL, 0, NULL,
       SQL_DRIVER_COMPLETE);
    check_error(db_context->retcode, db_context->dbc,
       SQL_HANDLE_DBC, "connect_to_database -
       SQLDriverConnect");

    printf("Connected successfully to MySQL database!\n");
}
```

Listing 7: Code pour la connexion a la base de donnees

```
void init_tables(DBContext* db_context) {
    SQLAllocHandle(SQL_HANDLE_STMT, db_context->dbc, &
       db_context->hstmt);
    init_table("Admin",
        "CREATE TABLE IF NOT EXISTS admin(admin_id INT
           PRIMARY KEY AUTO_INCREMENT, nom VARCHAR(45) NOT
           NULL, prenom VARCHAR(45) NOT NULL)",
        db_context);
    init_table("Client",
        "CREATE TABLE IF NOT EXISTS client(client_id INT
           PRIMARY KEY AUTO_INCREMENT, nom VARCHAR(45) NOT
           NULL, prenom VARCHAR(45) NOT NULL)",
        db_context); // Si On va implementez la fonction de
           log in des utilisateurs.

    init_table("Category",
        "CREATE TABLE IF NOT EXISTS category(category_id INT
           PRIMARY KEY AUTO_INCREMENT, nom VARCHAR(45) NOT
           NULL)",
        db_context);
    init_table("Services",
    "CREATE TABLE IF NOT EXISTS services(service_id INT
       PRIMARY KEY AUTO_INCREMENT, nom VARCHAR(45) NOT NULL,
       prix DOUBLE NOT NULL, admin_id INT NOT NULL,
       category_id INT, FOREIGN KEY (admin_id) references
       admin(admin_id) on delete cascade on update restrict,
       FOREIGN KEY (category_id) REFERENCES category(
       category_id) on delete cascade on update restrict)",
        db_context);
    init_table("Commands",
```

```
17      "CREATE TABLE IF NOT EXISTS commands(service_id INT,
            client_id INT, FOREIGN KEY (client_id) references
            client(client_id) on delete cascade on update restrict
            ,  FOREIGN KEY (service_id) references services(
            service_id) on delete cascade on update restrict)",
18           db_context);
19      // Execute an SQL query
20      show_tables(db_context);
21      insert_default_admin(db_context);
22      insert_default_categories(db_context);
23  }
```

Listing 8: Code pour l'initialisation des tables

```
1   void init_table(char* tableName, char* tableQuery, DBContext*
        db_context) {
2       SQLAllocHandle(SQL_HANDLE_STMT, db_context->dbc, &
            db_context->hstmt);
3       printf("CREATING TABLE %s  ---> ", tableName);
4
5       db_context->retcode = SQLExecDirect(db_context->hstmt, (
            SQLCHAR*)tableQuery, SQL_NTS);
6       char error_message[100];
7
8       sprintf(error_message, "Error Creating the table %s.",
            tableName);
9       check_error(db_context->retcode, db_context->dbc,
            SQL_HANDLE_DBC, error_message);
10      if(db_context->retcode == SQL_SUCCESS || db_context->
            retcode == SQL_SUCCESS_WITH_INFO) {
11          printf("TABLE %s HAS BEEN CREATED SUCCESSFULLY \n",
                tableName);
12      }
13  }
```

Listing 9: Code pour l'initialisation de chaque table

```
1   int get_entity_count(char* entity, DBContext* db_context) {
2       SQLAllocHandle(SQL_HANDLE_STMT, db_context->dbc, &
            db_context->hstmt);
3       char query[100] = "SELECT count(*) FROM ";
4       strcat(query, entity);
5       printf("query: %s \n", query);
6       db_context->retcode = SQLExecDirect(db_context->hstmt, (
            SQLCHAR*)query, SQL_NTS);
7       char err[70] = "ERROR RETREVING COUNT ";
```

```
    strcat(err, entity);
    check_error(db_context->retcode, db_context->dbc,
        SQL_HANDLE_DBC, err);
    SQLINTEGER countRowsInCategoryTable;  // Buffer to hold
        table names
    SQLLEN indicator;         // Indicator for NULL values
    SQLFetch(db_context->hstmt);
    SQLGetData(db_context->hstmt, 1, SQL_INTEGER,  &
        countRowsInCategoryTable, sizeof(
        countRowsInCategoryTable), &indicator);
    return countRowsInCategoryTable;
}
```

Listing 10: Code pour recuperer le nombre d'entites

```
void insert_default_categories(DBContext* db_context) {
    int countCategories = get_entity_count("category",
        db_context);
    if(countCategories == 0) {
        printf("\n----------------- INSERTING DEFAULT
            CATEGORIES -----------------\n");
        insert_category("Informatique", db_context);
        insert_category("Electricite", db_context);
        insert_category("Etudes", db_context);
        insert_category("Restauration", db_context);
        insert_category("Coiffure", db_context);
    } else {
        printf("%d Default categories were found \n",
            countCategories);
        get_categories(db_context);
    }
}

void insert_default_admin(DBContext* db_context) {
    int countAdmins = get_entity_count("admin", db_context);
    if(countAdmins == 0) {
        SQLAllocHandle(SQL_HANDLE_STMT, db_context->dbc, &
            db_context->hstmt);
        printf(" ---> INSERTING ADMIN: %s \n", "Ahmed Ben
            Ahmed");
        char *query = "INSERT INTO admin(nom, prenom) values
            (?,?)"; // Supposing there is only one admin using
             the system
        SQLBindParameter(db_context->hstmt, 1,
            SQL_PARAM_INPUT, SQL_C_CHAR, SQL_VARCHAR, 0,0,(
            SQLCHAR*) "BEN_AHMED", 0, NULL);
```

```
23          SQLBindParameter(db_context->hstmt, 2,
                SQL_PARAM_INPUT, SQL_C_CHAR, SQL_VARCHAR, 0,0,(
                SQLCHAR*) "AHMED", 0, NULL);
24          db_context->retcode = SQLExecDirect(db_context->hstmt
                , (SQLCHAR*)query, SQL_NTS);
25          char *error_message;
26          sprintf(error_message, "ERROR INSERTING THE Admin %s.
                ", "Ahmed Ben Ahmed");
27          check_error(db_context->retcode, db_context->dbc,
                SQL_HANDLE_DBC, error_message);
28          printf("ADMIN %s HAS BEEN CREATED SUCCESSFULLY \n", "
                Ahmed Ben Ahmed");
29      }
30  }
```

Listing 11: Code pour ajouter les donnees par defaut

```
1   void get_services(DBContext* db_context) {
2       SQLAllocHandle(SQL_HANDLE_STMT, db_context->dbc, &
            db_context->hstmt);
3       char* query = "SELECT s.service_id, s.nom, a.nom, a.
            prenom, c.nom, s.prix FROM services as s JOIN category
             as c on c.category_id = s.category_id JOIN admin as a
             ON a.admin_id = s.admin_id";
4       db_context->retcode = SQLExecDirect(db_context->hstmt, (
            SQLCHAR*)query, SQL_NTS);
5       check_error(db_context->retcode, db_context->dbc,
            SQL_HANDLE_DBC, "RETRIEVING ALL SERVICES");
6       SQLLEN indicator;        // Indicator for NULL values
7       while (SQLFetch(db_context->hstmt) != SQL_NO_DATA) {
8           int service_id;
9           double service_price;
10          char serviceName[100]; // maybe change
11          char adminNom[100]; // maybe change
12          char adminPrenom[100]; // maybe change
13          char categorieName[100]; // maybe change
14          SQLGetData(db_context->hstmt, 1, SQL_INTEGER, &
                service_id, sizeof(service_id), &indicator);
15          SQLGetData(db_context->hstmt, 2, SQL_C_CHAR,
                serviceName, sizeof(serviceName), &indicator);
16          SQLGetData(db_context->hstmt, 3, SQL_C_CHAR, adminNom
                , sizeof(adminNom), &indicator);
17          SQLGetData(db_context->hstmt, 4, SQL_C_CHAR,
                adminPrenom, sizeof(adminPrenom), &indicator);
18          SQLGetData(db_context->hstmt, 5, SQL_C_CHAR,
                categorieName, sizeof(categorieName), &indicator);
```

```
19          SQLGetData(db_context->hstmt, 6, SQL_DOUBLE, &
                service_price, sizeof(service_price), &indicator);
20          printf("ID : %d | Nom : %s | Prix : %.2f | Categorie
                : %s | Ajoutee par: %s %s\n",service_id,
                serviceName, service_price, categorieName,
                adminPrenom, adminNom);
21      }
22  }
```

Listing 12: Code pour recuperer tous les services

7.7 Extraits de code pour les deux parties combinées

Il n'y a pas beaucoup de modifications dans les parties menus.

```
1
2   typedef struct {
3       SQLHANDLE dbc;
4       SQLRETURN retcode;
5       SQLHSTMT hstmt;
6       SQLHANDLE env;
7   } DBContext;
8
9   int main() {
10      DBContext db_context;
11      connect_to_database(&db_context);
12      init_tables(&db_context);
13      int choix;
14      printf("Bonjour !\n");
15
16      while (1) {
17          printf("\nMerci de selectionner si vous etes admin ou
                client :\n");
18          printf("1. Admin\n");
19          printf("2. Client\n");
20          printf("3. Quitter\n");
21          printf("Votre choix : ");
22
23          if (scanf("%d", &choix) != 1) {
24              printf("Entree invalide. Veuillez reessayer.\n");
25              while (getchar() != '\n');
26              continue;
27          }
28
29          switch (choix) {
```

```
30              case 1: menu_admin(&db_context); break;
31              case 2: menu_client(&db_context); break;
32              case 3:
33                  printf("Au revoir !\n");
34                  SQLDisconnect(db_context.dbc);
35                  SQLFreeHandle(SQL_HANDLE_DBC, db_context.dbc)
                        ;
36                  SQLFreeHandle(SQL_HANDLE_ENV, db_context.env)
                        ;
37              return 0;
38              default:
39                  printf("Choix invalide. Veuillez reessayer.\n
                        ");
40          }
41      }
42  }
```

Listing 13: Code pour le nouveau main

```
1   service* addService(DBContext* db_context) {
2       service *service;
3       if((service = malloc(sizeof(service))) == NULL) return
            NULL;
4       printf("\nAjouter le nom du service : ");
5       while (getchar() != '\n');
6       fgets(service->nom_service, sizeof(service->nom_service),
            stdin);
7       service->nom_service[strcspn(service->nom_service, "\n")]
            = '\0';
8
9       printf("Ajouter le prix du service : ");
10      if (scanf("%lf", &service->prix_service) != 1) {
11          printf("Entree invalide pour le prix. Service non
                ajoute.\n");
12          while (getchar() != '\n');
13          return NULL;
14      }
15      int id_categorie;
16      while (1) {
17          printf("Veuillez choisir une categorie dans la liste
                ci-dessous :\n");
18          get_categories(db_context);
19
20          printf("Entrez l'ID de la categorie (ou 0 pour
                ajouter une nouvelle categorie) : ");
21          if (scanf("%d", &id_categorie) != 1) {
```

```
            printf("Entree invalide. Veuillez reessayer.\n");
            while (getchar() != '\n');
            continue;
        }

        if (id_categorie == 0) {
            categorie categorie;
            printf("\nAjouter le nom de la nouvelle categorie
                : ");
            while (getchar() != '\n');
            fgets(categorie.nom, sizeof(categorie.nom), stdin
                );
            categorie.nom[strcspn(categorie.nom, "\n")] = '\0
                ';
            id_categorie = insert_category(categorie.nom,
                db_context);
            printf("Nouvelle categorie ajoutee avec succ s
                aved ID: %d !\n",id_categorie);
            break;
        }

        int categorie_trouvee = getCategorieById(id_categorie
            , db_context) != NULL;

        if (categorie_trouvee) {
            break;
        }
        printf("Categorie invalide. Veuillez reessayer.\n");
    }
    service->categorie_service_id = id_categorie;
    insert_service(*service, db_context);
    printf("Service ajoute avec succes !\n");
}
```

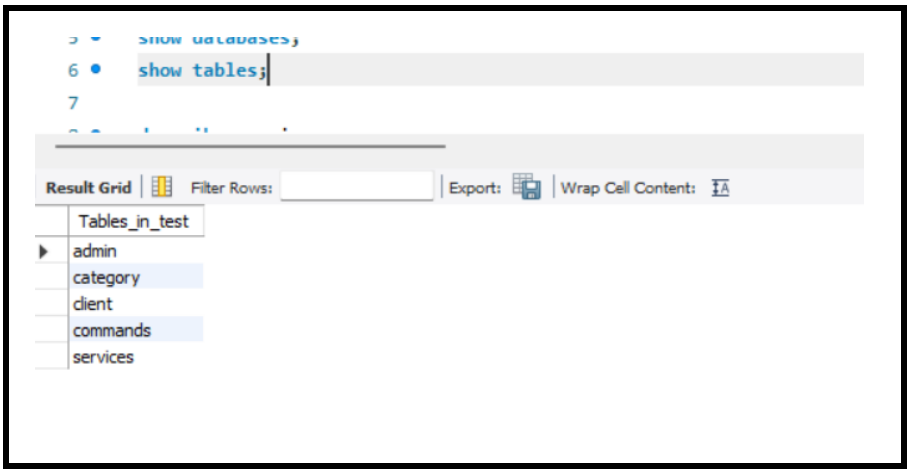
Listing 14: Code pour la nouvelle fonction d'ajout de service

```
void afficherServices(DBContext* db_context) {
    int count = get_entity_count("services", db_context);
    if (count == 0) {
        printf("Aucun service disponible.\n");
        return;
    }
    get_services(db_context);
}
```

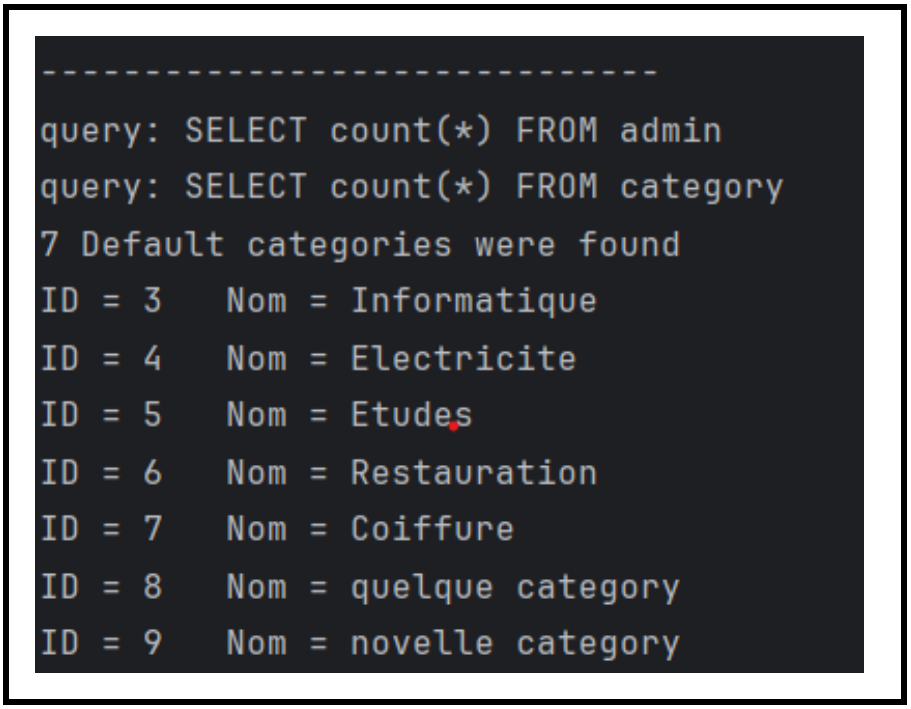
Listing 15: Code pour la nouvelle fonction d'affichage des services

7.8 Service Management Interface - CLI and MySQL Workbench

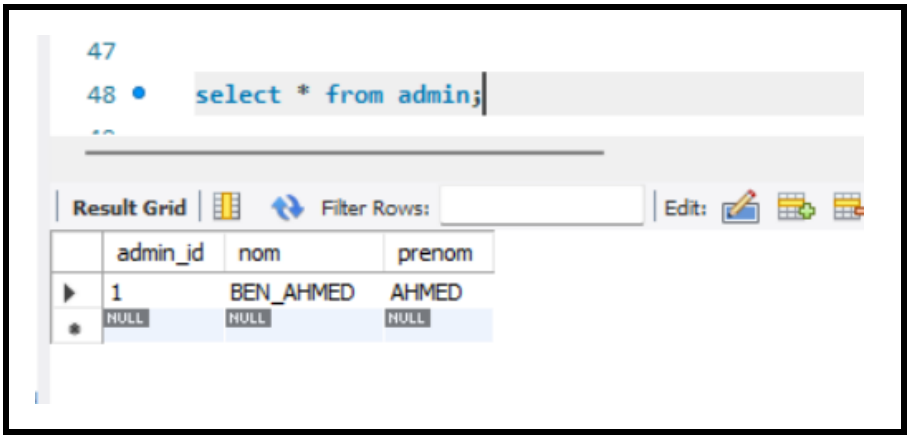
Création réussie des tables de la base de données (Admin, Client, Category, Services, Commands)



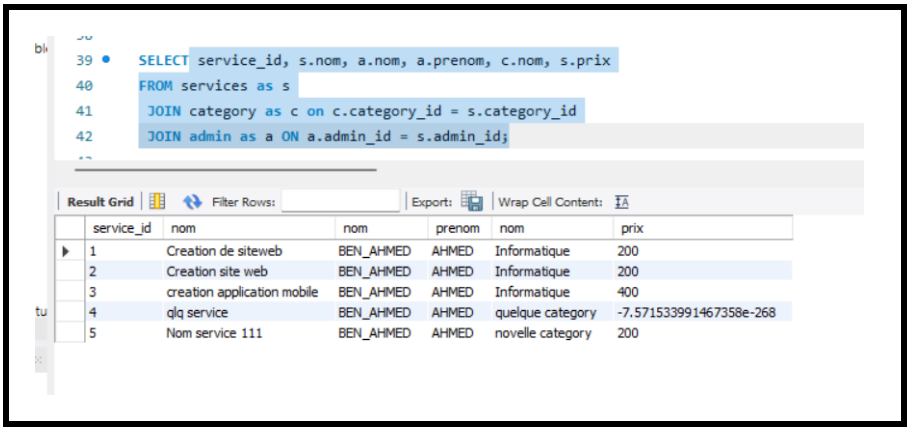
Vue des tables dans MySQL Workbench



Affichage des catégories par défaut dans le système



Requête SELECT montrant les informations de l'administrateur

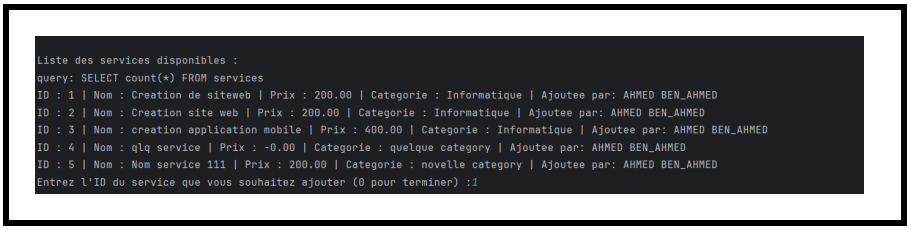


```
39 • SELECT service_id, s.nom, a.nom, a.prenom, c.nom, s.prix
40   FROM services as s
41    JOIN category as c on c.category_id = s.category_id
42    JOIN admin as a ON a.admin_id = s.admin_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

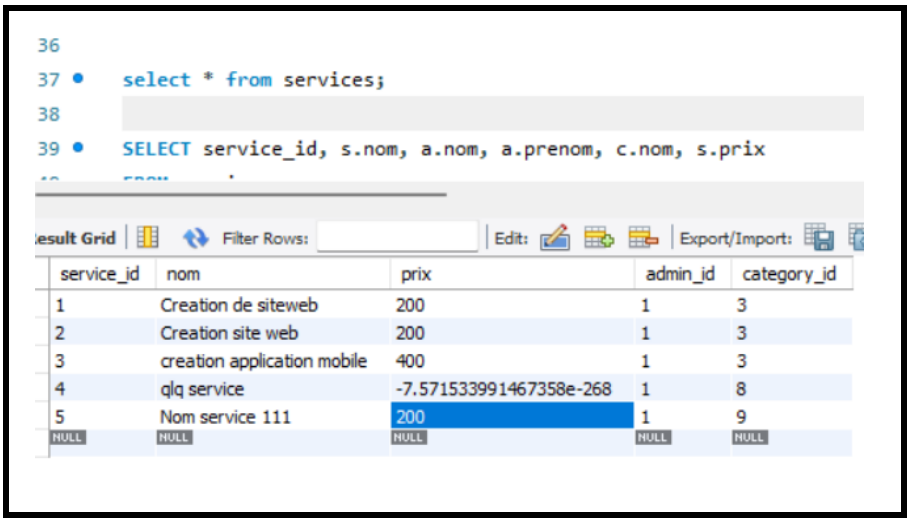
service_id	nom	nom	prenom	nom	prix
1	Creation de siteweb	BEN_AHMED	AHMED	Informatique	200
2	Creation site web	BEN_AHMED	AHMED	Informatique	200
3	creation application mobile	BEN_AHMED	AHMED	Informatique	400
4	qlq service	BEN_AHMED	AHMED	quelque category	-7.571533991467358e-268
5	Nom service 111	BEN_AHMED	AHMED	novelle category	200

Requête JOIN affichant les services avec les informations associées des tables admin et category



```
Liste des services disponibles :
query: SELECT count(*) FROM services
ID : 1 | Nom : Creation de siteweb | Prix : 200.00 | Categorie : Informatique | Ajoutee par: AHMED BEN_AHMED
ID : 2 | Nom : Creation site web | Prix : 200.00 | Categorie : Informatique | Ajoutee par: AHMED BEN_AHMED
ID : 3 | Nom : creation application mobile | Prix : 400.00 | Categorie : Informatique | Ajoutee par: AHMED BEN_AHMED
ID : 4 | Nom : qlq service | Prix : -0.00 | Categorie : quelque category | Ajoutee par: AHMED BEN_AHMED
ID : 5 | Nom : Nom service 111 | Prix : 200.00 | Categorie : novelle category | Ajoutee par: AHMED BEN_AHMED
Entrez l'ID du service que vous souhaitez ajouter (0 pour terminer) :1
```

Liste des services disponibles avec leurs détails (ID, Nom, Prix, Catégorie)



```
36
37 • select * from services;
38
39 • SELECT service_id, s.nom, a.nom, a.prenom, c.nom, s.prix
```

esult Grid | Filter Rows: | Edit: | Export/Import:

service_id	nom	prix	admin_id	category_id
1	Creation de siteweb	200	1	3
2	Creation site web	200	1	3
3	creation application mobile	400	1	3
4	qlq service	-7.571533991467358e-268	1	8
5	Nom service 111	200	1	9
NULL	NULL	NULL	NULL	NULL

Requête SELECT affichant les services avec leurs détails

8 Développement et Intégration

Le développement a été réalisé de manière collaborative. Après avoir convenu de la structure générale du projet, nous avons divisé les tâches :

- Mon collègue a développé la partie logicielle du programme en ligne de commande (CLI), qui gère les interactions avec l'utilisateur.
- De mon côté, je me suis concentré sur l'intégration de la base de données via le connecteur ODBC 8.1. J'ai créé les fonctions pour la connexion à la base de données, l'initialisation des tables, et l'insertion des données par défaut (par exemple, un utilisateur admin et des catégories par défaut).

8.1 Gestion des Erreurs

Pendant le processus d'intégration, nous avons rencontré plusieurs erreurs, notamment liées à la gestion de la connexion à la base de données. Un des principaux problèmes était de s'assurer que les tables étaient correctement initialisées avant d'effectuer toute opération. Voici un exemple de gestion des erreurs dans notre code C :

```
1  void check_error(SQLRETURN retcode, SQLHANDLE handle,
       SQLSMALLINT type, char* error_message) {
2      if (retcode != SQL_SUCCESS && retcode !=
           SQL_SUCCESS_WITH_INFO) {
3          SQLCHAR sqlState[6], message[256];
4          SQLINTEGER nativeError;
5          SQLSMALLINT messageLength;
6          SQLGetDiagRec(type, handle, 1, sqlState, &nativeError
               , message, sizeof(message), &messageLength);
7          printf("%s \n", error_message);
8          printf("ODBC Error: %s (%d): %s\n", sqlState,
               nativeError, message);
9          exit(EXIT_FAILURE);
10         }
11 }
```

Listing 16: Code pour la nouvelle fonction d'affichage des services

9 Résultats

Les résultats du projet ont été positifs. Après avoir intégré les deux parties (menu CLI et base de données), nous avons pu générer des factures détaillées et afficher les services disponibles. Nous avons également validé le bon fonctionnement de la base de données en utilisant MySQL Workbench pour vérifier les données insérées.

10 Conclusion et Perspectives

Ce projet a permis de renforcer nos compétences en programmation C et en gestion de base de données avec ODBC. Nous avons réussi à créer un système de facturation simple mais fonctionnel. À l'avenir, nous pourrions étendre ce système pour permettre une gestion multi-clients, ainsi que l'ajout d'autres fonctionnalités comme la gestion des paiements ou la génération de rapports de facturation.

11 Défis rencontrés

- **Problèmes de connexion à la base de données** : Des erreurs initiales sont survenues en raison d'une configuration incorrecte d'ODBC. Cela a été résolu en déboguant la chaîne de connexion et en testant les requêtes dans MySQL Workbench.
- **Validation des entrées** : Des entrées utilisateur non validées ont causé des plantages. Une validation robuste a été ajoutée pour les options du menu.
- **Bugs d'intégration** : L'intégration des fonctionnalités CLI avec les opérations sur la base de données a entraîné des erreurs de récupération de données, qui ont été résolues en testant les composants individuellement avant l'intégration.

Tout le code de la partie CLI

```
1
2  #include <stdio.h>
3  #include <stdlib.h>
4  #include <string.h>
5
6  typedef struct categorie {
7      int id_categorie;
8      char nom[50];
9  } categorie;
10
11 typedef struct service {
12     int id_service;
13     char nom_service[50];
14     double prix_service;
15     categorie categorie_service;
16 } service;
17
18 typedef struct client {
19     int id_client;
20     char nom_client[50];
21     double montant_total;
22     char details_facture[500];
23 } client;
24
```

```
25 #define MAX_SERVICES 100
26 #define MAX_CATEGORIES 10
27 #define MAX_CLIENTS 2
28
29 service services[MAX_SERVICES];
30 categorie categories[MAX_CATEGORIES] = {
31     {1, "Coiffure"},
32     {2, "Electricite"},
33     {3, "Etudes"},
34     {4, "Restauration"},
35     {5, "Informatique"}
36 };
37 client clients[MAX_CLIENTS];
38 int count = 0;
39 int categorie_count = 5;
40 int client_count = 0; // Nombre de clients enregistr s
41
42 void menu_admin();
43 void menu_client();
44 void addService();
45 void afficherServices();
46 void afficherCategories();
47 void addCategorie();
48 void genererFacture(int id_client);
49 void afficherFacture(double montant, char *details, char *
       nom_client);
50 int findClientByName(char *nom_client);
51 void addClient();
52 void supprimerService();
53 void supprimerCategorie();
54
55 int main() {
56     int choix;
57     printf("Bonjour !\n");
58
59     while (1) {
60         printf("\nMerci de selectionner si vous etes admin ou
               client :\n");
61         printf("1. Admin\n");
62         printf("2. Client\n");
63         printf("3. Quitter\n");
64         printf("Votre choix : ");
65
66         if (scanf("%d", &choix) != 1) {
67             printf("Entree invalide. Veuillez reessayer.\n");
```

```
68              while (getchar() != '\n');
69              continue;
70          }
71
72          switch (choix) {
73              case 1: menu_admin(); break;
74              case 2: menu_client(); break;
75              case 3:
76                  printf("Au revoir !\n");
77                  return 0;
78              default:
79                  printf("Choix invalide. Veuillez reessayer.\n
                        ");
80          }
81      }
82      return 0;
83  }
84
85  void menu_admin() {
86      int choix;
87      while (1) {
88          printf("\nMenu Admin :\n");
89          printf("1. Ajouter un service\n");
90          printf("2. Afficher tous les services\n");
91          printf("3. Ajouter une categorie\n");
92          printf("4. Supprimer un service\n");
93          printf("5. Supprimer une categorie\n");
94          printf("6. Retour au menu principal\n");
95          printf("Votre choix : ");
96
97          if (scanf("%d", &choix) != 1) {
98              printf("Entree invalide. Veuillez reessayer.\n");
99              while (getchar() != '\n');
100             continue;
101         }
102
103         switch (choix) {
104             case 1: addService(); break;
105             case 2: afficherServices(); break;
106             case 3: addCategorie(); break;
107             case 4: supprimerService(); break;
108             case 5: supprimerCategorie(); break;
109             case 6: return;
110             default:
```

```
111                 printf("Choix invalide. Veuillez reessayer.\n
                        ");
112             }
113         }
114     }
115     void supprimerService() {
116         int id_service;
117         printf("\nEntrez l'ID du service a supprimer : ");
118         if (scanf("%d", &id_service) != 1) {
119             printf("Entree invalide. Veuillez reessayer.\n");
120             while (getchar() != '\n');
121             return;
122         }
123 
124         int found = 0;
125         for (int i = 0; i < count; i++) {
126             if (services[i].id_service == id_service) {
127 
128                 for (int j = i; j < count - 1; j++) {
129                     services[j] = services[j + 1];
130                 }
131                 count--;
132                 printf("Service avec ID %d supprim  avec succ s
                        .\n", id_service);
133                 found = 1;
134                 break;
135             }
136         }
137 
138         if (!found) {
139             printf("Service introuvable avec l'ID %d.\n",
                    id_service);
140         }
141     }
142 
143     void supprimerCategorie() {
144         int id_categorie;
145         printf("\nEntrez l'ID de la categorie a supprimer : ");
146         if (scanf("%d", &id_categorie) != 1) {
147             printf("Entree invalide. Veuillez reessayer.\n");
148             while (getchar() != '\n');
149             return;
150         }
151 
152         // V rification de l'existence de la cat gorie
```

```
153     int found = 0;
154     for (int i = 0; i < categorie_count; i++) {
155         if (categories[i].id_categorie == id_categorie) {
156             for (int j = i; j < categorie_count - 1; j++) {
157                 categories[j] = categories[j + 1];
158             }
159             categorie_count--;
160             printf("Categorie avec ID %d supprim e avec
                    succ s.\n", id_categorie);
161             found = 1;
162             break;
163         }
164     }
165 
166     if (!found) {
167         printf("Categorie introuvable avec l'ID %d.\n",
                id_categorie);
168     }
169 }
170 
171 void addCategorie() {
172     if (categorie_count >= MAX_CATEGORIES) {
173         printf("Le nombre maximum de categories a ete atteint
                .\n");
174         return;
175     }
176 
177     printf("\nAjouter le nom de la categorie : ");
178     while (getchar() != '\n');
179     fgets(categories[categorie_count].nom, sizeof(categories[
            categorie_count].nom), stdin);
180     categories[categorie_count].nom[strcspn(categories[
            categorie_count].nom, "\n")] = '\0';
181 
182     categories[categorie_count].id_categorie =
            categorie_count + 1;
183     printf("Categorie '%s' ajoutee avec succes !\n",
            categories[categorie_count].nom);
184     categorie_count++;
185 }
186 
187 void addService() {
188     if (count >= MAX_SERVICES) {
189         printf("Le nombre maximum de services a ete atteint.\
                n");
```

```
190         return;
191     }
192 
193     printf("\nAjouter le nom du service : ");
194     while (getchar() != '\n');
195     fgets(services[count].nom_service, sizeof(services[count
            ].nom_service), stdin);
196     services[count].nom_service[strcspn(services[count].
            nom_service, "\n")] = '\0';
197 
198     printf("Ajouter le prix du service : ");
199     if (scanf("%lf", &services[count].prix_service) != 1) {
200         printf("Entree invalide pour le prix. Service non
                ajoute.\n");
201         while (getchar() != '\n');
202         return;
203     }
204 
205     while (1) {
206         printf("Veuillez choisir une categorie dans la liste
                ci-dessous :\n");
207         afficherCategories();
208 
209         int id_categorie;
210         printf("Entrez l'ID de la categorie (ou 0 pour
                ajouter une nouvelle categorie) : ");
211         if (scanf("%d", &id_categorie) != 1) {
212             printf("Entree invalide. Veuillez reessayer.\n");
213             while (getchar() != '\n');
214             continue;
215         }
216 
217         if (id_categorie == 0) {
218             addCategorie();
219             services[count].categorie_service = categories[
                    categorie_count - 1];
220             break;
221         }
222 
223         int categorie_trouvee = 0;
224         for (int i = 0; i < categorie_count; i++) {
225             if (categories[i].id_categorie == id_categorie) {
226                 services[count].categorie_service =
                        categories[i];
227                 categorie_trouvee = 1;
```

```
228                     break;
229                 }
230             }
231 
232             if (categorie_trouvee) {
233                 break;
234             } else {
235                 printf("Categorie invalide. Veuillez reessayer.\n
                        ");
236             }
237         }
238 
239         services[count].id_service = count + 1;
240         printf("Service ajoute avec succes !\n");
241         count++;
242     }
243 
244     void afficherServices() {
245         if (count == 0) {
246             printf("Aucun service disponible.\n");
247             return;
248         }
249 
250         for (int i = 0; i < count; i++) {
251             printf("ID : %d | Nom : %s | Prix : %.2f | Categorie
                    : %s\n",
252                    services[i].id_service,
253                    services[i].nom_service,
254                    services[i].prix_service,
255                    services[i].categorie_service.nom);
256         }
257     }
258 
259     void afficherCategories() {
260         printf("\nListe des categories disponibles :\n");
261         for (int i = 0; i < categorie_count; i++) {
262             printf("ID : %d | Nom : %s\n", categories[i].
                    id_categorie, categories[i].nom);
263         }
264     }
265 
266     void menu_client() {
267         int choix;
268         char nom_client[50];
269 
```

```
270     // Demande du nom du client
271     printf("Veuillez entrer votre nom : ");
272     scanf("%s", nom_client);
273 
274     int id_client = findClientByName(nom_client);
275     if (id_client == -1) {
276         printf("Client introuvable. Enregistrement d'un
                nouveau client...\n");
277         addClient(nom_client);
278         id_client = client_count - 1;  // Nouvel ID du client
279     }
280 
281     do {
282         printf("\nMenu Client :\n");
283         printf("1. Voir tous les services\n");
284         printf("2. Generer une facture\n");
285         printf("3. Retour au menu principal\n");
286         printf("Votre choix : ");
287         if (scanf("%d", &choix) != 1) {
288             printf("Entree invalide. Veuillez reessayer.\n");
289             while (getchar() != '\n');
290             continue;
291         }
292 
293         switch (choix) {
294             case 1:
295                 afficherServices();
296                 break;
297             case 2:
298                 genererFacture(id_client); // Passe l'ID du
                    client    la fonction
299                 break;
300             case 3:
301                 printf("Retour au menu principal...\n");
302                 break;
303             default:
304                 printf("Choix invalide. Veuillez reessayer.\n
                    ");
305         }
306     } while (choix != 3);
307 }
308 
309 void genererFacture(int id_client) {
310     double montant_total = 0.0;
311     int id_service;
```

```
312     char details_facture[500] = "";
313 
314     printf("\nGeneration de facture pour %s\n", clients[
            id_client].nom_client);
315     printf("\nListe des services disponibles :\n");
316     afficherServices();
317 
318     do {
319         printf("Entrez l'ID du service que vous souhaitez
                ajouter (0 pour terminer) : ");
320         if (scanf("%d", &id_service) != 1) {
321             printf("Entree invalide. Veuillez reessayer.\n");
322             while (getchar() != '\n');
323             continue;
324         }
325 
326         if (id_service == 0) {
327             break;
328         }
329 
330         int service_trouve = 0;
331         for (int i = 0; i < count; i++) {
332             if (services[i].id_service == id_service) {
333                 montant_total += services[i].prix_service;
334                 sprintf(details_facture + strlen(
                        details_facture),
335                         "Service: %s | Prix: %.2f\n",
336                         services[i].nom_service, services[i].
                            prix_service);
337                 strcat(details_facture, "
                        +---------------------------+\n");
338                 service_trouve = 1;
339                 break;
340             }
341         }
342 
343         if (!service_trouve) {
344             printf("Service introuvable. Essayez avec un ID
                    valide.\n");
345         }
346 
347     } while (id_service != 0);
348 
349     afficherFacture(montant_total, details_facture, clients[
            id_client].nom_client);
```

```
}

void afficherFacture(double montant, char *details, char *
    nom_client) {
    printf("\n+-------- Facture ----------+\n");
    printf("+--------------------------+\n");
    printf("+--------%s--------+\n", nom_client);
    printf("+--------------------------+\n");
    printf("%s\n", details);
    printf("\n+Total:          +%.2f+\n", montant);
    printf("+--------------------------+\n");
}

int findClientByName(char *nom_client) {
    for (int i = 0; i < client_count; i++) {
        if (strcmp(clients[i].nom_client, nom_client) == 0) {
            return i; //retourne l'indice du client
        }
    }
    return -1; // Client non trouv
}

void addClient(char *nom_client) {
    if (client_count >= MAX_CLIENTS) {
        printf("Le nombre maximum de clients a ete atteint.\n
            ");
        return;
    }

    clients[client_count].id_client = client_count + 1;
    strcpy(clients[client_count].nom_client, nom_client);
    clients[client_count].montant_total = 0.0;
    clients[client_count].details_facture[0] = '\0';
    printf("Client '%s' ajoute avec succes !\n", nom_client);
    client_count++;
}
```

Listing 17: Code pour la nouvelle fonction d'affichage des services

Tout le code de la partie CLI integrée avec MySQL

```
1
2  #include <windows.h>  // For HWND
3  #include <sqlext.h>   // ODBC headers
4  #include <sqltypes.h>
5  #include <sql.h>
6  #include <stdio.h>
7  #include <stdlib.h>
8
9  typedef struct {
10     SQLHANDLE dbc;
11     SQLRETURN retcode;
12     SQLHSTMT hstmt;
13     SQLHANDLE env;
14 } DBContext;
15
16 typedef struct categorie {
17     int id_categorie;
18     char nom[50];
19 } categorie;
20
21 typedef struct service {
22     int id_service;
23     char nom_service[50];
24     double prix_service;
25     int categorie_service_id;
26 } service;
27
28
29 void connect_to_database(DBContext* db_context);
30 void init_tables(DBContext* db_context);
31 int insert_category(char* categoryName, DBContext* db_context
       );
32 int insert_service(service service, DBContext* db_context);
33 int get_entity_count(char* entity, DBContext* db_context);
34 void get_categories(DBContext* db_context);
35 void insert_default_categories(DBContext* db_context);
36 void insert_default_admin(DBContext* db_context);
37 void show_tables(DBContext* db_context);
38 int get_last_inserted_id(DBContext* db_context);
39 void menu_admin(DBContext* db_context);
40 void menu_client(DBContext* db_context);
41 service* addService(DBContext* db_context);
42 void afficherServices(DBContext* db_context);
43 void afficherCategories(DBContext* db_context);
44 void addCategorie(DBContext* db_context);
```

```
45 void genererFacture(DBContext* db_context);
46
47 int main() {
48     DBContext db_context;
49     connect_to_database(&db_context);
50     init_tables(&db_context);
51     int choix;
52     printf("Bonjour !\n");
53
54     while (1) {
55         printf("\nMerci de selectionner si vous etes admin ou
               client :\n");
56         printf("1. Admin\n");
57         printf("2. Client\n");
58         printf("3. Quitter\n");
59         printf("Votre choix : ");
60
61         if (scanf("%d", &choix) != 1) {
62             printf("Entree invalide. Veuillez reessayer.\n");
63             while (getchar() != '\n');
64             continue;
65         }
66
67         switch (choix) {
68             case 1: menu_admin(&db_context); break;
69             case 2: menu_client(&db_context); break;
70             case 3:
71                 printf("Au revoir !\n");
72                 SQLDisconnect(db_context.dbc);
73                 SQLFreeHandle(SQL_HANDLE_DBC, db_context.dbc)
                       ;
74                 SQLFreeHandle(SQL_HANDLE_ENV, db_context.env)
                       ;
75             return 0;
76             default:
77                 printf("Choix invalide. Veuillez reessayer.\n
                       ");
78         }
79     }
80
81 }
82
83
84 void check_error(SQLRETURN retcode, SQLHANDLE handle,
       SQLSMALLINT type, char* error_message) {
```

```
85      if (retcode != SQL_SUCCESS && retcode !=
            SQL_SUCCESS_WITH_INFO) {
86          SQLCHAR sqlState[6], message[256];
87          SQLINTEGER nativeError;
88          SQLSMALLINT messageLength;
89          SQLGetDiagRec(type, handle, 1, sqlState, &nativeError
                , message, sizeof(message), &messageLength);
90          printf("%s \n", error_message);
91          printf("ODBC Error: %s (%d): %s\n", sqlState,
                nativeError, message);
92          exit(EXIT_FAILURE);
93      }
94  }
95  void init_table(char* tableName, char* tableQuery,DBContext*
        db_context) {
96      SQLAllocHandle(SQL_HANDLE_STMT, db_context->dbc, &
            db_context->hstmt);
97      printf("CREATING TABLE %s  ---> ", tableName);
98
99      db_context->retcode = SQLExecDirect(db_context->hstmt, (
            SQLCHAR*)tableQuery, SQL_NTS);
100     char error_message[100];
101
102     sprintf(error_message, "Error Creating the table %s.",
            tableName);
103     check_error(db_context->retcode, db_context->dbc,
            SQL_HANDLE_DBC, error_message);
104     if(db_context->retcode == SQL_SUCCESS || db_context->
            retcode == SQL_SUCCESS_WITH_INFO) {
105         printf("TABLE %s HAS BEEN CREATED SUCCESSFULLY \n",
                tableName);
106     }
107 }
108
109 int insert_category(char categoryName[100], DBContext*
        db_context) {
110     SQLAllocHandle(SQL_HANDLE_STMT, db_context->dbc, &
            db_context->hstmt);
111     printf(" ---> INSERTING CATEGORY: %s \n", categoryName);
112     char *query = "INSERT INTO category(nom) values (?)";
113     SQLBindParameter(db_context->hstmt, 1, SQL_PARAM_INPUT,
            SQL_C_CHAR, SQL_VARCHAR, 0,0,(SQLCHAR*) categoryName,
            0, NULL);
114     db_context->retcode = SQLExecDirect(db_context->hstmt, (
            SQLCHAR*)query, SQL_NTS);
```

```
115     char error_message[200];
116     sprintf(error_message, "ERROR INSERTING THE CATEGORY %s."
            , categoryName);
117     check_error(db_context->retcode, db_context->dbc,
            SQL_HANDLE_DBC, error_message);
118     if(db_context->retcode == SQL_SUCCESS || db_context->
            retcode == SQL_SUCCESS_WITH_INFO) {
119         printf("CATEGORY %s HAS BEEN CREATED SUCCESSFULLY \n"
                , categoryName);
120     }
121     return get_last_inserted_id(db_context);
122 }
123 int insert_service(service service ,DBContext* db_context) {
124     SQLAllocHandle(SQL_HANDLE_STMT, db_context->dbc, &
            db_context->hstmt);
125     printf(" ---> INSERTING Service: %s \n", service.
            nom_service);
126     char *query = "INSERT INTO services(nom,prix, category_id
            , admin_id) values (?,?,?,1)"; // Supposing there is
            only one admin using the system
127     SQLBindParameter(db_context->hstmt, 1, SQL_PARAM_INPUT,
            SQL_C_CHAR, SQL_VARCHAR, 0,0,(SQLCHAR*) service.
            nom_service, 0, NULL);
128     SQLBindParameter(db_context->hstmt, 2, SQL_PARAM_INPUT,
            SQL_C_DOUBLE, SQL_DOUBLE, 0,0 ,&service.prix_service,
            0, NULL);
129     SQLBindParameter(db_context->hstmt, 3, SQL_PARAM_INPUT,
            SQL_INTEGER, SQL_INTEGER, 0,0 ,&service.
            categorie_service_id, 0, NULL);
130     db_context->retcode = SQLExecDirect(db_context->hstmt, (
            SQLCHAR*)query, SQL_NTS);
131     char error_message[200];
132     sprintf(error_message, "ERROR INSERTING THE SERVICE %s.",
             service.nom_service);
133     check_error(db_context->retcode, db_context->dbc,
            SQL_HANDLE_DBC, error_message);
134     if(db_context->retcode == SQL_SUCCESS || db_context->
            retcode == SQL_SUCCESS_WITH_INFO) {
135         printf("SERVICE %s HAS BEEN CREATED SUCCESSFULLY \n",
                service.nom_service);
136     }
137     return get_last_inserted_id(db_context);
138 }
139
140
```

```
141
142
143 int get_entity_count(char* entity, DBContext* db_context) {
144     SQLAllocHandle(SQL_HANDLE_STMT, db_context->dbc, &
        db_context->hstmt);
145     char query[100] = "SELECT count(*) FROM ";
146     strcat(query, entity);
147     printf("query: %s \n", query);
148     db_context->retcode = SQLExecDirect(db_context->hstmt, (
        SQLCHAR*)query, SQL_NTS);
149     char err[70] = "ERROR RETREVING COUNT ";
150     strcat(err, entity);
151     check_error(db_context->retcode, db_context->dbc,
        SQL_HANDLE_DBC, err);
152     SQLINTEGER countRowsInCategoryTable;  // Buffer to hold
        table names
153     SQLLEN indicator;        // Indicator for NULL values
154     SQLFetch(db_context->hstmt);
155     SQLGetData(db_context->hstmt, 1, SQL_INTEGER,  &
        countRowsInCategoryTable, sizeof(
        countRowsInCategoryTable), &indicator);
156     return countRowsInCategoryTable;
157 }
158
159 void get_categories(DBContext* db_context) {
160     SQLAllocHandle(SQL_HANDLE_STMT, db_context->dbc, &
        db_context->hstmt);
161     char* query = "SELECT category_id, nom FROM category";
162     db_context->retcode = SQLExecDirect(db_context->hstmt, (
        SQLCHAR*)query, SQL_NTS);
163     check_error(db_context->retcode, db_context->dbc,
        SQL_HANDLE_DBC, "RETRIEVING ALL CATEGORIES");
164     SQLLEN indicator;        // Indicator for NULL values
165     int i = 0;
166     while (SQLFetch(db_context->hstmt) != SQL_NO_DATA) {
167         int category_id;
168         char nom[100];
169         SQLGetData(db_context->hstmt, 1, SQL_INTEGER, &
            category_id, sizeof(category_id), &indicator);
170         SQLGetData(db_context->hstmt, 2, SQL_C_CHAR, nom,
            sizeof(nom), &indicator);
171         printf("ID = %d \t Nom = %s\n",category_id, nom);
172         i++;
173     }
174 }
```

```
175 void get_services(DBContext* db_context) {
176     SQLAllocHandle(SQL_HANDLE_STMT, db_context->dbc, &
        db_context->hstmt);
177     char* query = "SELECT s.service_id, s.nom, a.nom, a.
        prenom, c.nom, s.prix FROM services as s JOIN category
         as c on c.category_id = s.category_id JOIN admin as a
         ON a.admin_id = s.admin_id";
178     db_context->retcode = SQLExecDirect(db_context->hstmt, (
        SQLCHAR*)query, SQL_NTS);
179     check_error(db_context->retcode, db_context->dbc,
        SQL_HANDLE_DBC, "RETRIEVING ALL SERVICES");
180     SQLLEN indicator;        // Indicator for NULL values
181     while (SQLFetch(db_context->hstmt) != SQL_NO_DATA) {
182         int service_id;
183         double service_price;
184         char serviceName[100]; // maybe change
185         char adminNom[100]; // maybe change
186         char adminPrenom[100]; // maybe change
187         char categorieName[100]; // maybe change
188         SQLGetData(db_context->hstmt, 1, SQL_INTEGER, &
            service_id, sizeof(service_id), &indicator);
189         SQLGetData(db_context->hstmt, 2, SQL_C_CHAR,
            serviceName, sizeof(serviceName), &indicator);
190         SQLGetData(db_context->hstmt, 3, SQL_C_CHAR, adminNom
            , sizeof(adminNom), &indicator);
191         SQLGetData(db_context->hstmt, 4, SQL_C_CHAR,
            adminPrenom, sizeof(adminPrenom), &indicator);
192         SQLGetData(db_context->hstmt, 5, SQL_C_CHAR,
            categorieName, sizeof(categorieName), &indicator);
193         SQLGetData(db_context->hstmt, 6, SQL_DOUBLE, &
            service_price, sizeof(service_price), &indicator);
194         printf("ID : %d | Nom : %s | Prix : %.2f | Categorie
            : %s | Ajoutee par: %s %s\n",service_id,
            serviceName, service_price, categorieName,
            adminPrenom, adminNom);
195     }
196 }
197 void insert_default_categories(DBContext* db_context) {
198     // SQLAllocHandle(SQL_HANDLE_STMT, db_context->dbc, &
        db_context->hstmt);
199     int countCategories = get_entity_count("category",
        db_context);
200     if(countCategories == 0) {
201         printf("\n----------------- INSERTING DEFAULT
            CATEGORIES -----------------\n");
```

```
202         insert_category("Informatique", db_context);
203         insert_category("Electricite", db_context);
204         insert_category("Etudes", db_context);
205         insert_category("Restauration", db_context);
206         insert_category("Coiffure", db_context);
207     } else {
208         printf("%d Default categories were found \n",
                countCategories);
209         get_categories(db_context);
210     }
211 }
212 void insert_default_admin( DBContext* db_context) {
213     int countAdmins = get_entity_count("admin", db_context);
214     if(countAdmins == 0) {
215         SQLAllocHandle(SQL_HANDLE_STMT, db_context->dbc, &
                db_context->hstmt);
216         printf(" ---> INSERTING ADMIN: %s \n", "Ahmed Ben
                Ahmed");
217         char *query = "INSERT INTO admin(nom, prenom) values
                (?,?)"; // Supposing there is only one admin using
                 the system
218         SQLBindParameter(db_context->hstmt, 1,
                SQL_PARAM_INPUT, SQL_C_CHAR, SQL_VARCHAR, 0,0,(
                SQLCHAR*) "BEN_AHMED", 0, NULL);
219         SQLBindParameter(db_context->hstmt, 2,
                SQL_PARAM_INPUT, SQL_C_CHAR, SQL_VARCHAR, 0,0,(
                SQLCHAR*) "AHMED", 0, NULL);
220         db_context->retcode = SQLExecDirect(db_context->hstmt
                , (SQLCHAR*)query, SQL_NTS);
221         char *error_message;
222         sprintf(error_message, "ERROR INSERTING THE Admin %s.
                ", "Ahmed Ben Ahmed");
223         check_error(db_context->retcode, db_context->dbc,
                SQL_HANDLE_DBC, error_message);
224         printf("ADMIN %s HAS BEEN CREATED SUCCESSFULLY \n", "
                Ahmed Ben Ahmed");
225     }
226
227 }
228 void show_tables(DBContext* db_context) {
229     char* query = "SHOW TABLES";
230     db_context->retcode = SQLExecDirect(db_context->hstmt, (
            SQLCHAR*)query, SQL_NTS);
231     check_error(db_context->retcode, db_context->dbc,
            SQL_HANDLE_DBC, "Error Retreiving all the tables
```

```
        existed on the database.");
232     SQLCHAR tableName[256];  // Buffer to hold table names
233     SQLLEN indicator;        // Indicator for NULL values
234
235     printf("Tables in the database:\n");
236     printf("------------------------\n");
237
238     // Fetch each row of the result
239     while (SQLFetch(db_context->hstmt) != SQL_NO_DATA) {
240         SQLGetData(db_context->hstmt, 1, SQL_C_CHAR,
            tableName, sizeof(tableName), &indicator);
241         printf("%s\n", tableName);  // Print the table name
242     }
243
244     printf("------------------------------\n");
245 }
246 void init_tables( DBContext* db_context) {
247     SQLAllocHandle(SQL_HANDLE_STMT, db_context->dbc, &
        db_context->hstmt);
248     init_table("Admin",
249         "CREATE TABLE IF NOT EXISTS admin(admin_id INT
            PRIMARY KEY AUTO_INCREMENT, nom VARCHAR(45) NOT
            NULL, prenom VARCHAR(45) NOT NULL)",
250         db_context);
251     init_table("Client",
252         "CREATE TABLE IF NOT EXISTS client(client_id INT
            PRIMARY KEY AUTO_INCREMENT, nom VARCHAR(45) NOT
            NULL, prenom VARCHAR(45) NOT NULL)",
253         db_context); // Si On va implementez la fonction de
            log in des utilisateurs.
254
255     init_table("Category",
256         "CREATE TABLE IF NOT EXISTS category(category_id INT
            PRIMARY KEY AUTO_INCREMENT, nom VARCHAR(45) NOT
            NULL)",
257         db_context);
258     init_table("Services",
259     "CREATE TABLE IF NOT EXISTS services(service_id INT
        PRIMARY KEY AUTO_INCREMENT, nom VARCHAR(45) NOT NULL,
        prix DOUBLE NOT NULL, admin_id INT NOT NULL,
        category_id INT, FOREIGN KEY (admin_id) references
        admin(admin_id) on delete cascade on update restrict,
        FOREIGN KEY (category_id) REFERENCES category(
        category_id) on delete cascade on update restrict)",
260         db_context);
```

```
261     init_table("Commands",
262     "CREATE TABLE IF NOT EXISTS commands(service_id INT,
            client_id INT, FOREIGN KEY (client_id) references
            client(client_id) on delete cascade on update restrict
            ,  FOREIGN KEY (service_id) references services(
            service_id) on delete cascade on update restrict)",
263          db_context);
264     // Execute an SQL query
265     show_tables(db_context);
266     insert_default_admin(db_context);
267     insert_default_categories(db_context);
268 }
269 int get_last_inserted_id(DBContext* db_context) {
270     SQLAllocHandle(SQL_HANDLE_STMT, db_context->dbc, &
            db_context->hstmt);
271     char *id_query = "SELECT LAST_INSERT_ID()";
272     db_context->retcode = SQLExecDirect(db_context->hstmt, (
            SQLCHAR*)id_query, SQL_NTS);
273     check_error(db_context->retcode, db_context->dbc,
            SQL_HANDLE_DBC, "ERROR RETRIEVING LAST INSERTED ID");
274
275     SQLINTEGER last_inserted_id;
276     SQLLEN indicator;
277     SQLFetch(db_context->hstmt);
278     SQLGetData(db_context->hstmt, 1, SQL_INTEGER, &
            last_inserted_id, sizeof(last_inserted_id), &indicator
            );
279     return last_inserted_id;
280 }
281 void connect_to_database(DBContext* db_context) {
282     // Allocate environment handle
283      db_context->retcode = SQLAllocHandle(SQL_HANDLE_ENV,
             SQL_NULL_HANDLE, &db_context->env);
284     check_error(db_context->retcode, db_context->env,
            SQL_HANDLE_ENV, "connect_to_database - SQLAllocHandle"
            );
285
286     // Set the ODBC version
287     db_context->retcode = SQLSetEnvAttr(db_context->env,
            SQL_ATTR_ODBC_VERSION, (SQLPOINTER)SQL_OV_ODBC3, 0);
288     check_error(db_context->retcode, db_context->env,
            SQL_HANDLE_ENV, "connect_to_database - SQLSetEnvAttr")
            ;
289
290     // Allocate connection handle
```

```
291     db_context->retcode = SQLAllocHandle(SQL_HANDLE_DBC,
            db_context->env, &db_context->dbc);
292     check_error(db_context->retcode, db_context->dbc,
            SQL_HANDLE_DBC,  "connect_to_database - SQLAllocHandle
            ");
293
294     // Connect to the database
295     SQLCHAR connStr[] = "DRIVER={C:\\Program Files\\MySQL\\
            Connector ODBC 8.1\\myodbc8w.dll};SERVER=localhost;
            PORT=3306;DATABASE=test;USER=root;PASSWORD=;";
296     db_context->retcode = SQLDriverConnect(db_context->dbc,
            NULL, connStr, SQL_NTS, NULL, 0, NULL,
            SQL_DRIVER_COMPLETE);
297     check_error(db_context->retcode, db_context->dbc,
            SQL_HANDLE_DBC, "connect_to_database -
            SQLDriverConnect");
298
299     printf("Connected successfully to MySQL database!\n");
300 }
301
302
303
304 void menu_admin(DBContext* db_context) {
305     int choix;
306     while (1) {
307         printf("\nMenu Admin :\n");
308         printf("1. Ajouter un service\n");
309         printf("2. Afficher tous les services\n");
310         printf("3. Ajouter une categorie\n");
311         printf("4. Retour au menu principal\n");
312         printf("Votre choix : ");
313
314         if (scanf("%d", &choix) != 1) {
315             printf("Entree invalide. Veuillez reessayer.\n");
316             while (getchar() != '\n');
317             continue;
318         }
319
320         switch (choix) {
321             case 1: addService(db_context); break;
322             case 2: afficherServices(db_context); break;
323             case 3: addCategorie(db_context); break;
324             case 4: return;
325             default:
```

```
326                 printf("Choix invalide. Veuillez reessayer.\n
                        ");
327         }
328     }
329 }
330
331 void addCategorie(DBContext* db_context) {
332     categorie categorie;
333     printf("\nAjouter le nom de la categorie : ");
334     while (getchar() != '\n');
335     fgets(categorie.nom, sizeof(categorie.nom), stdin);
336     categorie.nom[strcspn(categorie.nom, "\n")] = '\0';
337     insert_category(categorie.nom, db_context);
338     printf("Categorie '%s' ajoutee avec succes !\n",
            categorie.nom);
339 }
340
341 categorie* getCategorieById(int id_categorie, DBContext*
        db_context) {
342     categorie *categorie;
343     if((categorie = malloc(sizeof(categorie))) == NULL)
            return NULL;
344     SQLAllocHandle(SQL_HANDLE_STMT, db_context->dbc, &
            db_context->hstmt);
345     char* query = "SELECT category_id, nom from category
            where category_id = ?";
346     SQLBindParameter(db_context->hstmt, 1, SQL_PARAM_INPUT,
            SQL_C_LONG, SQL_INTEGER,0, 0,&id_categorie, 0 , NULL);
347     db_context->retcode = SQLExecDirect(db_context->hstmt, (
            SQLCHAR*)query, SQL_NTS);
348     check_error(db_context->retcode, db_context->dbc,
            SQL_HANDLE_DBC, "Getting category by id");
349     SQLLEN indicator;        // Indicator for NULL values
350     SQLFetch(db_context->hstmt);
351     SQLGetData(db_context->hstmt, 1, SQL_INTEGER,  &categorie
            ->id_categorie, sizeof(categorie->id_categorie), &
            indicator);
352     SQLGetData(db_context->hstmt, 2, SQL_CHAR,  &categorie->
            nom, sizeof(categorie->nom), &indicator);
353     return categorie;
354 }
355 service* getServiceById(int service_id, DBContext* db_context
        ) {
356     service *service;
```

```
357     if((service = malloc(sizeof(service))) == NULL) return
            NULL;
358     SQLAllocHandle(SQL_HANDLE_STMT, db_context->dbc, &
            db_context->hstmt);
359     char* query = "SELECT service_id, nom, prix  from
            services where service_id = ?";
360     SQLBindParameter(db_context->hstmt, 1, SQL_PARAM_INPUT,
            SQL_C_LONG, SQL_INTEGER,0, 0,&service_id, 0 , NULL);
361     db_context->retcode = SQLExecDirect(db_context->hstmt, (
            SQLCHAR*)query, SQL_NTS);
362     check_error(db_context->retcode, db_context->dbc,
            SQL_HANDLE_DBC, "Getting Service by id");
363     SQLLEN indicator;         // Indicator for NULL values
364     SQLFetch(db_context->hstmt);
365     SQLGetData(db_context->hstmt, 1, SQL_INTEGER,  &service->
            id_service, sizeof(service->id_service), &indicator);
366     SQLGetData(db_context->hstmt, 2, SQL_CHAR,  &service->
            nom_service, sizeof(service->nom_service), &indicator)
            ;
367     SQLGetData(db_context->hstmt, 3, SQL_DOUBLE,  &service->
            prix_service, sizeof(service->prix_service), &
            indicator);
368     return service;
369 }
370
371 service* addService(DBContext* db_context) {
372     service *service;
373     if((service = malloc(sizeof(service))) == NULL) return
            NULL;
374     printf("\nAjouter le nom du service : ");
375     while (getchar() != '\n');
376     fgets(service->nom_service, sizeof(service->nom_service),
             stdin);
377     service->nom_service[strcspn(service->nom_service, "\n")]
             = '\0';
378
379     printf("Ajouter le prix du service : ");
380     if (scanf("%lf", &service->prix_service) != 1) {
381         printf("Entree invalide pour le prix. Service non
                ajoute.\n");
382         while (getchar() != '\n');
383         return NULL;
384     }
385     int id_categorie;
386     while (1) {
```

```
387         printf("Veuillez choisir une categorie dans la liste
                ci-dessous :\n");
388         get_categories(db_context);
389 
390         printf("Entrez l'ID de la categorie (ou 0 pour
                ajouter une nouvelle categorie) : ");
391         if (scanf("%d", &id_categorie) != 1) {
392             printf("Entree invalide. Veuillez reessayer.\n");
393             while (getchar() != '\n');
394             continue;
395         }
396 
397         if (id_categorie == 0) {
398             categorie categorie;
399             printf("\nAjouter le nom de la nouvelle categorie
                    : ");
400             while (getchar() != '\n');
401             fgets(categorie.nom, sizeof(categorie.nom), stdin
                    );
402             categorie.nom[strcspn(categorie.nom, "\n")] = '\0
                    ';
403             id_categorie = insert_category(categorie.nom,
                    db_context);
404             printf("Nouvelle categorie ajoutee avec succ s
                    aved ID: %d !\n",id_categorie);
405             break;
406         }
407 
408         int categorie_trouvee = getCategorieById(id_categorie
                , db_context) != NULL;
409 
410         if (categorie_trouvee) {
411             break;
412         }
413         printf("Categorie invalide. Veuillez reessayer.\n");
414     }
415     service->categorie_service_id = id_categorie;
416     insert_service(*service, db_context);
417     printf("Service ajoute avec succes !\n");
418 }
419 
420 void afficherServices(DBContext* db_context) {
421     int count = get_entity_count("services", db_context);
422     if (count == 0) {
423         printf("Aucun service disponible.\n");
```

```
424         return;
425     }
426     get_services(db_context);
427 }
428 
429 
430 void menu_client(DBContext* db_context) {
431     int choix;
432     do {
433         printf("\nMenu Client :\n");
434         printf("1. Voir tous les services\n");
435         printf("2. Generer une facture\n");
436         printf("3. Retour au menu principal\n");
437         printf("Votre choix : ");
438         if (scanf("%d", &choix) != 1) {
439             printf("Entree invalide. Veuillez reessayer.\n");
440             while (getchar() != '\n');
441             continue;
442         }
443 
444         switch (choix) {
445             case 1:
446                 afficherServices(db_context);
447                 break;
448             case 2:
449                 genererFacture(db_context);
450                 break;
451             case 3:
452                 printf("Retour au menu principal...\n");
453                 break;
454             default:
455                 printf("Choix invalide. Veuillez reessayer.\n
                    ");
456         }
457     } while (choix != 3);
458 }
459 
460 void genererFacture(DBContext* db_context) {
461     // option reintilizer la facutre pour commencez au debut
462     // stockage des services selectionnez pour leur afficher
        dans la facture.
463 
464     double montant_total = 0.0;
465     int id_service;
466 
```

```
467     printf("\nGeneration de facture\n");
468     printf("\nListe des services disponibles :\n");
469     afficherServices(db_context);
470 
471     do {
472         printf("Entrez l'ID du service que vous souhaitez
                ajouter (0 pour terminer) : ");
473         if (scanf("%d", &id_service) != 1) {
474             printf("Entree invalide. Veuillez reessayer.\n");
475             while (getchar() != '\n');
476             continue;
477         }
478 
479         if (id_service == 0) {
480             break;
481         }
482 
483         service *service_trouve = getServiceById(id_service,
                db_context);
484         montant_total += service_trouve->prix_service;
485         if (service_trouve  == NULL) {
486             printf("Service introuvable. Essayez avec un ID
                    valide.\n");
487         }
488 
489     } while (id_service != 0);
490 
491     printf("\nMontant total de la facture : %.2f\nMerci pour
            votre visite !", montant_total);
492 }
```

Listing 18: Code pour la nouvelle fonction d'affichage des services

References

1. ODBC Installation (Version 8.1): https://downloads.mysql.com/archives/c-odbc/.

2. MySQL Workbench Documentation: https://dev.mysql.com/doc/workbench/en/.