ROYAUME DU MAROC
Université Abdelmalek Essaâdi
École Nationale des Sciences Appliquées d'Al Hoceima

Rapport de Projet

Système de Gestion des Réservations d'Hôtel

Structure de Données et Algorithmique Avancées

Réalisé par :
Yazid TAHRI ALAOUI
Dounia LAMDAKKI
Kaoutar BENOUAHI

Filière : 1ère année – Ingénierie Données (ID1)
Encadré par : Dr. BAHRI Abdelkhalek

Année universitaire : 2024 – 2025

Remerciements

La réalisation de ce projet ne saurait être réduite à une simple accumulation de lignes de code ; elle est le fruit d'une synergie, d'un apprentissage continu et d'un accompagnement précieux. Avant d'entamer la présentation technique de notre application de gestion hôtelière, il nous est agréable de témoigner notre reconnaissance envers ceux qui ont contribué, de près ou de loin, à l'aboutissement de ce travail.

Nous tenons en premier lieu à exprimer notre profonde gratitude à notre professeur, Monsieur **Dr. BAHRI Abdelkhalek**. Au-delà des connaissances techniques en langage C qu'il a su nous transmettre avec clarté, nous le remercions pour sa pédagogie exigeante mais bienveillante. Ses directives précises et son insistance sur la rigueur architecturale nous ont poussés à ne pas nous contenter de solutions fonctionnelles, mais à rechercher l'optimisation et la propreté du code. Merci de nous avoir guidés dans le labyrinthe de la gestion de la mémoire et des pointeurs, transformant nos difficultés en opportunités d'apprentissage.

Nos remerciements s'adressent également au corps professoral et administratif de l'École Nationale des Sciences Appliquées d'Al Hoceima, pour la qualité de l'environnement d'étude et les ressources mises à notre disposition, propices à la recherche et à l'innovation.

Nous n'oublions pas nos familles et nos proches, dont le soutien moral inconditionnel et les encouragements constants ont été notre moteur, particulièrement lors des phases de débogage les plus ardues et des nuits blanches consacrées à la finalisation du projet.

Enfin, une pensée particulière pour nos collègues de promotion, avec qui les échanges fructueux et l'entraide ont permis de surmonter les obstacles techniques et de maintenir une motivation intacte tout au long de ce semestre.

Que tous trouvent ici l'expression de notre estime et de notre respect.

Résumé

L'industrie hôtelière moderne nécessite des solutions informatiques robustes pour gérer efficacement les opérations quotidiennes. Ce rapport présente la conception et l'implémentation d'un système complet de gestion hôtelière développé en langage C, utilisant la bibliothèque ncurses pour offrir une interface utilisateur textuelle riche et intuitive.

Notre application propose une solution modulaire intégrant la gestion des clients, l'inventaire des chambres, le système de réservations et la facturation automatisée. Le système implémente un contrôle d'accès basé sur les rôles (RBAC) avec trois niveaux de privilèges : administrateur, réceptionniste et client, garantissant la sécurité et la séparation des responsabilités.

L'architecture du projet repose sur des structures de données optimisées et des algorithmes efficaces pour la recherche, le tri et la validation des données. La persistance des informations est assurée par un système de fichiers binaires, permettant la sauvegarde et la récupération des données entre les sessions.

Ce travail démontre notre maîtrise des concepts avancés de programmation en C, incluant la gestion dynamique de la mémoire, la manipulation de pointeurs, la programmation modulaire et l'utilisation de bibliothèques système. L'application résultante offre une expérience utilisateur professionnelle comparable aux solutions commerciales, tout en restant légère et performante.

Abstract

The modern hotel industry requires robust IT solutions to efficiently manage daily operations. This report presents the design and implementation of a comprehensive hotel management system developed in C language, using the ncurses library to provide a rich and intuitive text-based user interface.

Our application offers a modular solution integrating client management, room inventory, reservation system, and automated billing. The system implements role-based access control (RBAC) with three privilege levels : administrator, receptionist, and client, ensuring security and separation of responsibilities.

The project architecture is based on optimized data structures and efficient algorithms for searching, sorting, and data validation. Data persistence is ensured by a binary file system, allowing data storage and retrieval between sessions.

This work demonstrates our mastery of advanced C programming concepts, including dynamic memory management, pointer manipulation, modular programming, and system library usage. The resulting application provides a professional user experience comparable to commercial solutions while remaining lightweight and performant.

Table des matières

Liste des figures

Chapitre 1

Introduction Générale

1.1 Contexte du Projet

L'industrie hôtelière est un secteur où la qualité de service et la réactivité sont des vecteurs essentiels de compétitivité. Dans ce contexte dynamique, la gestion de l'information — qu'il s'agisse des disponibilités des chambres, des préférences des clients ou des transactions financières — ne peut plus se permettre d'être approximative. La transition numérique s'impose donc non plus comme une option, mais comme une nécessité opérationnelle pour toute structure souhaitant optimiser son rendement et garantir la satisfaction de sa clientèle.

C'est dans ce cadre que s'inscrit notre projet académique : la conception d'un système d'information complet, développé en langage C, capable de piloter les activités fondamentales d'un établissement hôtelier.

1.2 Problématique

Malgré l'avènement des technologies numériques, de nombreuses structures modestes continuent de s'appuyer sur des méthodes de gestion artisanales. L'utilisation de registres manuscrits ou de tableurs non centralisés engendre des dysfonctionnements critiques :

- **Perte d'intégrité des données** : Risque élevé d'erreurs humaines lors de la saisie ou de la recopie d'informations (nom mal orthographié, dates erronées).
- **Manque de visibilité temps réel** : Impossibilité pour le gérant de connaître instantanément le taux d'occupation ou les chambres nécessitant un nettoyage.
- **Insécurité des informations** : Absence de contrôle d'accès aux données sensibles des clients et volatilité des supports physiques.
- **Conflits de réservation** : Le phénomène de surréservation (overbooking) est fréquent lorsque la mise à jour des plannings n'est pas automatisée et synchronisée.

Dès lors, la question centrale de ce projet est la suivante : *Comment concevoir une architecture logicielle modulaire et robuste en langage C, capable d'automatiser le flux de travail hôtelier tout en garantissant la persistance et la sécurité des données ?*

1.3 Objectifs du Projet

Ce projet poursuit un double objectif, à la fois fonctionnel et pédagogique :

1.3.1 Objectifs Fonctionnels

L'application vise à fournir une interface console ergonomique permettant :

- Une gestion centralisée et sécurisée du parc immobilier (chambres) et du fichier clientèle.
- L'automatisation du cycle de vie d'une réservation, de la vérification des disponibilités à la facturation.
- La pérennité des informations entre les sessions d'utilisation grâce à un système de sauvegarde sur fichiers binaires.
- Un contrôle d'accès basé sur les rôles pour garantir la sécurité et la séparation des privilèges.

1.3.2 Objectifs Techniques

Sur le plan technique, ce travail est l'occasion de démontrer notre maîtrise des concepts avancés du langage C, notamment :

- La programmation modulaire pour structurer un code complexe.
- La manipulation de fichiers binaires pour la persistance des données.
- L'utilisation de structures de données et de pointeurs pour optimiser la gestion de la mémoire.
- La mise en œuvre d'algorithmes de recherche et de validation pour l'exploitation des données.
- L'intégration de la bibliothèque ncurses pour créer une interface utilisateur professionnelle.

1.4 Annonce du Plan

Pour rendre compte de notre démarche, ce rapport s'articulera autour de quatre axes majeurs :

1. **Analyse et Conception** : Nous détaillerons l'architecture des données, les choix techniques et la modélisation du système.
2. **Réalisation Technique** : Nous expliquerons l'implémentation des modules clés, les algorithmes développés et les défis techniques rencontrés.
3. **Validation et Tests** : Nous présenterons le guide d'utilisation à travers des scénarios de test et des captures d'écran.
4. **Conclusion** : Nous conclurons sur les acquis et les perspectives d'évolution de l'application.

Chapitre 2

Analyse et Conception du Système

2.1 Architecture Générale

Notre système de gestion hôtelière repose sur une architecture modulaire en trois couches distinctes, garantissant la séparation des responsabilités et la maintenabilité du code :

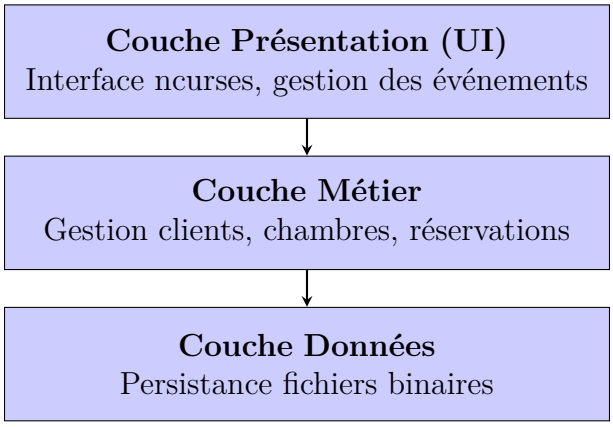

Figure 2.1 – *Architecture en trois couches du système*

2.1.1 Couche Présentation

La couche présentation utilise la bibliothèque `ncurses` pour offrir une interface utilisateur textuelle riche. Elle comprend :

- **Système de thèmes** : Gestion des couleurs et du style visuel
- **Composants réutilisables** : Formulaires, listes, boîtes de dialogue
- **Gestion des événements** : Navigation clavier, validation des saisies
- **Routage des vues** : Machine à états pour la navigation entre écrans

2.1.2 Couche Métier

La couche métier implémente la logique applicative :

- **Module Clients** : CRUD des informations client
- **Module Chambres** : Gestion de l'inventaire et disponibilités
- **Module Réservations** : Validation des dates, calcul des nuitées

- **Module Facturation** : Génération automatique des factures
- **Module Authentification** : Gestion des utilisateurs et RBAC

2.1.3 Couche Données

La couche données assure la persistance via des fichiers binaires :

- `clients.dat` : Enregistrements des clients
- `chambres.dat` : Inventaire des chambres
- `reservations.dat` : Historique des réservations
- `factures.dat` : Archive des factures
- `users.dat` : Base de données des utilisateurs

2.2 Modélisation des Données

2.2.1 Structures de Données Principales

Notre système utilise quatre structures de données fondamentales, définies dans `structures.h` :

```
typedef struct {
    int id;
    char nom[50];
    char prenom[50];
    char email[100];
    char telephone[20];
} Client;
```

Listing 2.1 – Structure Client

```
typedef struct {
    int numero;
    char type[20];
    float prix;
    int disponible;
} Chambre;
```

Listing 2.2 – Structure Chambre

```
typedef struct {
    int id;
    int client_id;
    int chambre_numero;
    char date_debut[11];
    char date_fin[11];
    float montant;
    char statut[20];
} Reservation;
```

Listing 2.3 – Structure Reservation

```c
typedef struct {
    int idFacture;
    int idClient;
    int nbNuits;
    float prixNuit;
    float total;
} Facture;
```

Listing 2.4 – Structure Facture

2.2.2 Diagramme de Relations

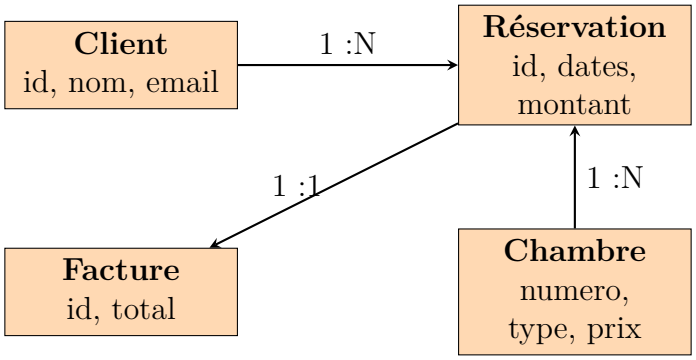

Figure 2.2 – *Diagramme des relations entre entités*

2.3 Diagrammes UML du Système

Cette section présente les diagrammes UML créés lors de la phase de conception. Ces modèles formalisent l'architecture, les workflows opérationnels et les interactions entre acteurs selon les standards UML 2.0, et correspondent directement aux fonctionnalités implémentées dans le code C.

2.3.1 Diagramme de Cas d'Utilisation

Le diagramme de cas d'utilisation définit la portée du système et identifie les acteurs principaux ainsi que leurs interactions avec les fonctionnalités.

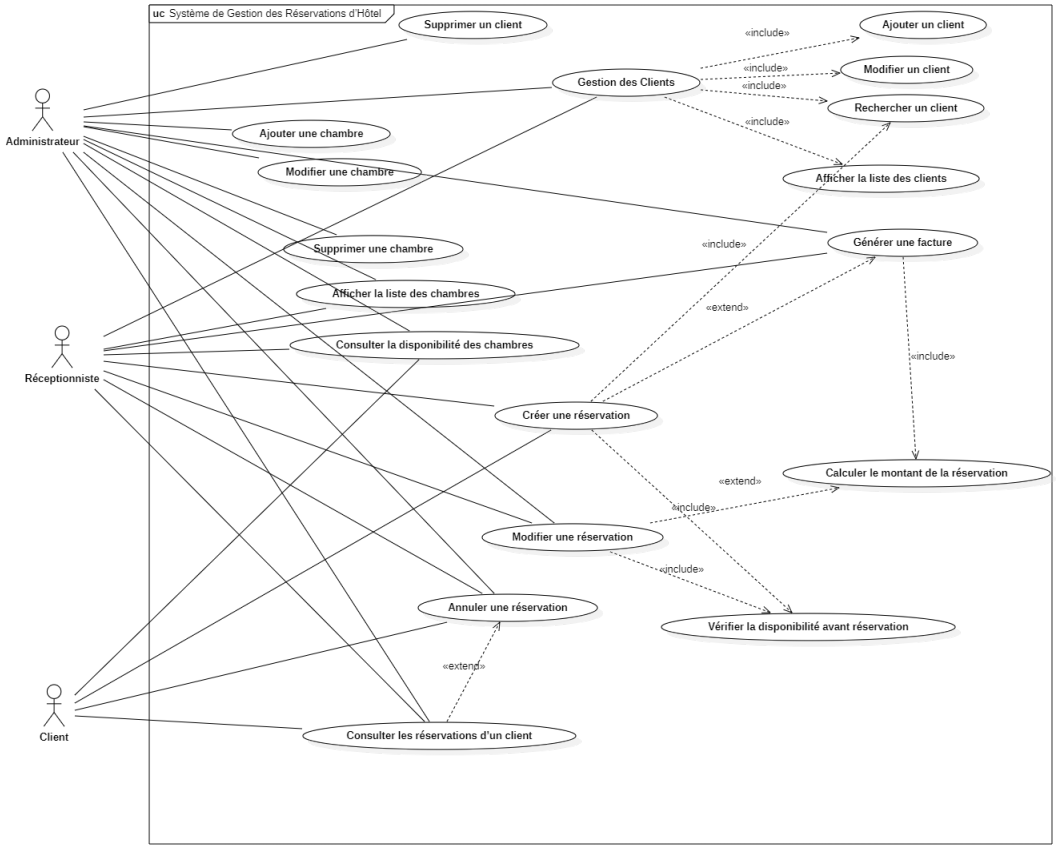

Figure 2.3 – *Diagramme de cas d'utilisation - Vue globale du système*

Acteurs identifiés :

- **Administrateur** : Dispose d'un accès complet au système
  - Gestion des chambres (CRUD)
  - Gestion des clients (CRUD)
  - Gestion de la facturation
  - Configuration du système
- **Réceptionniste** : Gère les opérations quotidiennes
  - Créer, Modifier, Annuler des réservations
  - Vérifier les disponibilités
  - Générer des factures
- **Client** : Accès en libre-service
  - Consulter ses propres réservations
  - Vérifier les disponibilités

Logique clé : Le diagramme montre les relations d'inclusion et d'extension. Par exemple, le cas d'utilisation "Créer Réservation" inclut nécessairement "Vérifier Disponibilité" et "Calculer Montant", garantissant l'intégrité des données.

2.3.2 Diagramme de Classes

Le diagramme de classes représente la structure statique du système, définissant les entités métier, leurs attributs, méthodes et relations.

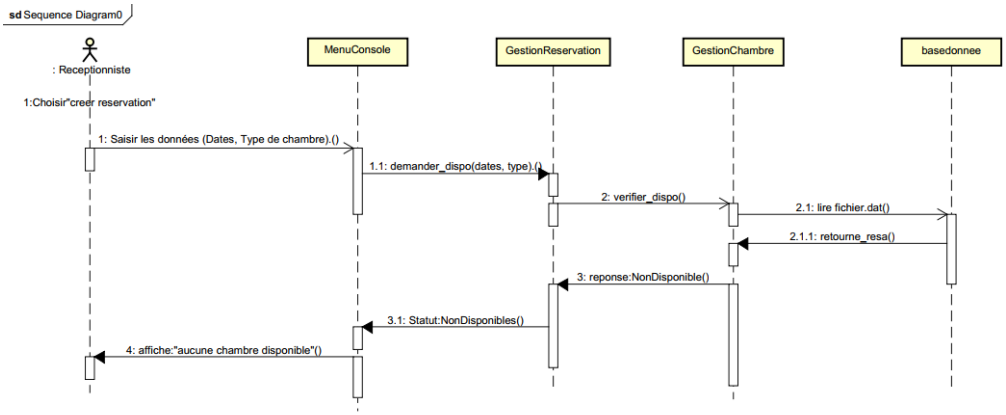

Figure 2.4 – *Diagramme de classes - Structure des entités métier*

Classes principales :

- **Client**
  - Attributs : id, nom, prénom, email, téléphone
  - Cardinalité : 1 Client peut avoir N Réservations
- **Chambre**
  - Attributs : numéro, étage, type (Single/Double/Suite), prix, disponibilité
  - Cardinalité : 1 Chambre peut être liée à N Réservations
- **Réservation**
  - Attributs : id, date_début, date_fin, montant, statut
  - Classe associative reliant Client et Chambre
  - Cardinalité : 1 Réservation génère 1 Facture
- **Facture**
  - Attributs : id, nombre_nuits, total, date_émission
  - Cardinalité : 1 :1 avec Réservation
- **Utilisateur**
  - Attributs : username, mot_de_passe (hashé), rôle (Admin/Réceptionniste/Client)
  - Gère l'authentification et les autorisations RBAC

Les associations sont implémentées en C via des clés étrangères dans les structures de données, garantissant l'intégrité référentielle.

2.3.3 Diagrammes de Séquence - Workflows de Réservation

Créer une Réservation (Workflow Complet)

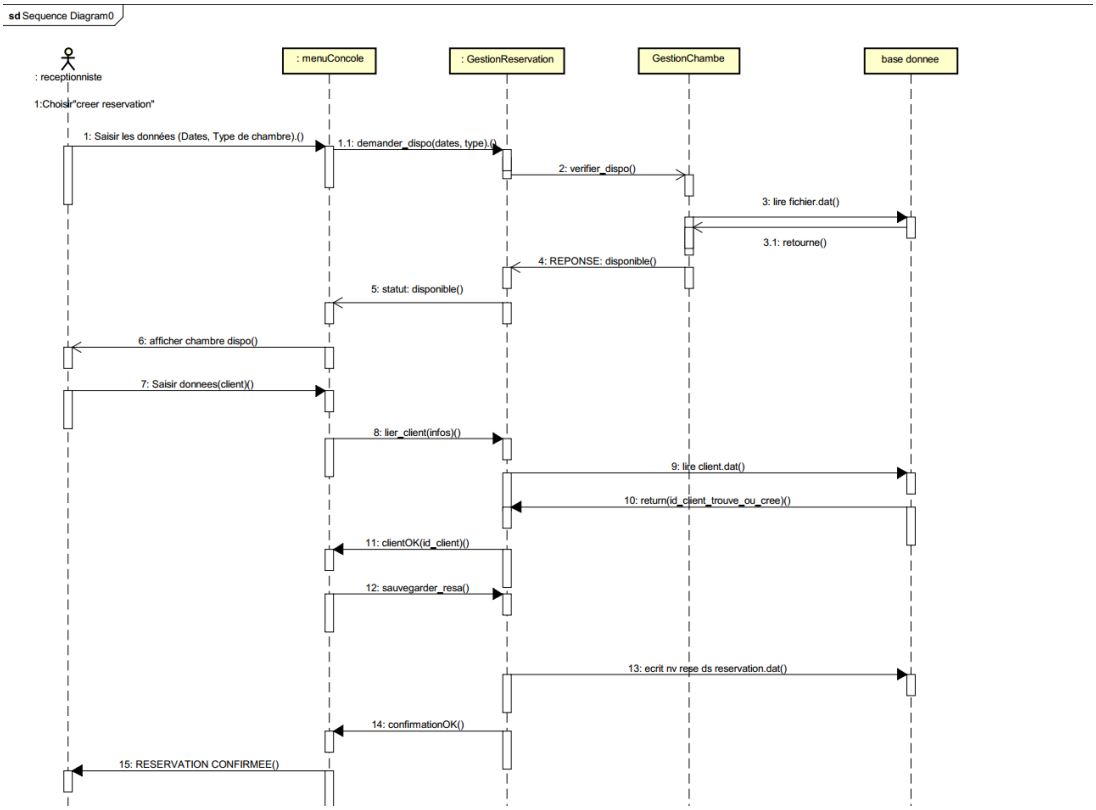

Figure 2.5 – *Séquence détaillée - Création de réservation par le réceptionniste*

Ce diagramme représente le workflow complet "Créer réservation" exécuté par le réceptionniste. Le flux opérationnel comprend :

1. **Saisie des données** : Dates de séjour et type de chambre souhaité
2. **Vérification disponibilité** : Appel de la fonction `verifier_dispo()`
3. **Saisie données client** : Informations client (nouveau ou existant)
4. **Liaison client** : Appel de `lier_client()` pour associer la réservation
5. **Sauvegarde** : Persistance dans `reservations.dat`
6. **Confirmation** : Message de succès affiché

Correspondance UI : Cette logique s'exécute derrière les écrans 4-8.png (Step 1 - saisie dates) et 3-4.png (Step 3 - liaison client).

Vérifier Disponibilité (Boucle de Contrôle)

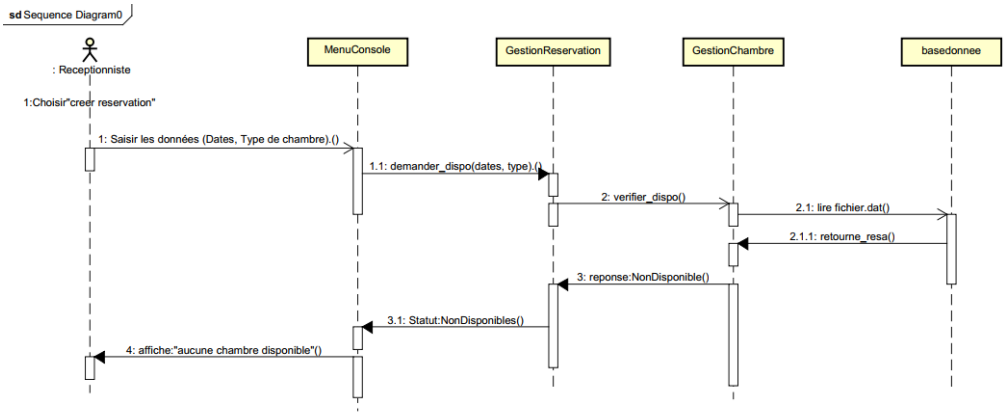

Figure 2.6 – *Séquence détaillée - Vérification de disponibilité des chambres*

Séquence détaillée pour la vérification des disponibilités :

1. Le réceptionniste entre les dates et le type de chambre
2. Le système interroge `reservations.dat` pour trouver les conflits
3. Pour chaque chambre, vérification du chevauchement de dates
4. Retour du statut : "Disponible" ou "NonDisponible"
5. Si aucune chambre libre : affichage "Aucune chambre disponible"

Cette fonction critique implémente l'algorithme de détection de conflits et prévient les surréservations.

Modifier une Réservation

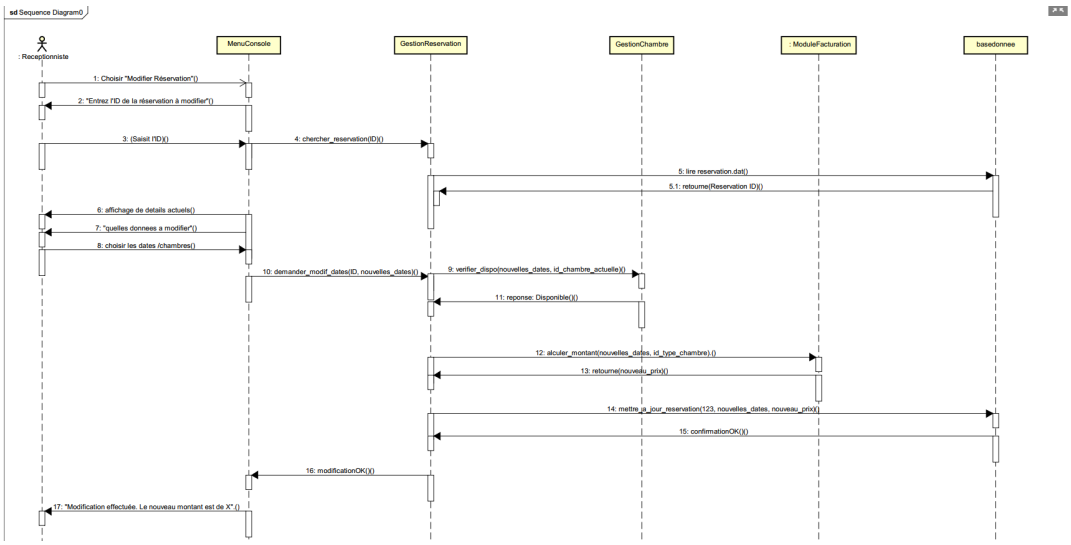

Figure 2.7 – *Séquence - Modification d'une réservation existante*

Le workflow de modification comprend :

1. Recherche par ID de réservation

2. Affichage des détails actuels
3. Saisie des nouvelles données (dates, chambre)
4. Vérification de disponibilité pour les nouvelles dates
5. Recalcul automatique du prix
6. Mise à jour de l'enregistrement

Correspondance UI : Cette logique correspond à la fonction [E]dit visible dans le footer de l'écran 4-7.png (liste des réservations).

Annuler une Réservation

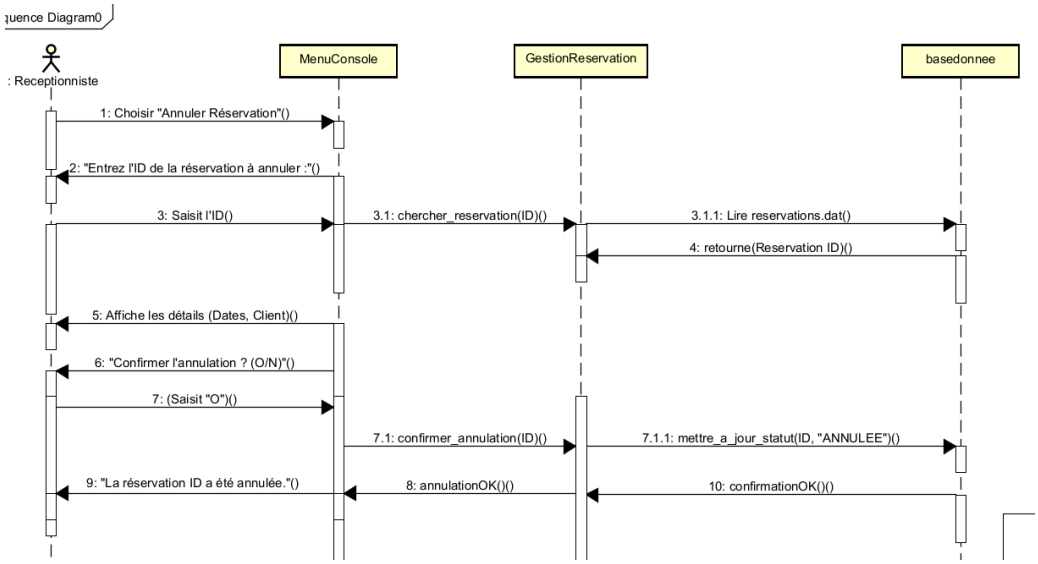

Figure 2.8 – *Séquence - Annulation de réservation*

Le processus d'annulation suit les étapes :

1. Sélection de la réservation par ID
2. Affichage des détails pour confirmation
3. Demande de confirmation utilisateur (O/N)
4. Si confirmé : mise à jour du statut vers "ANNULEE"
5. Libération de la chambre associée

L'annulation est implémentée comme un "soft delete" pour préserver l'historique.

Consulter Disponibilité

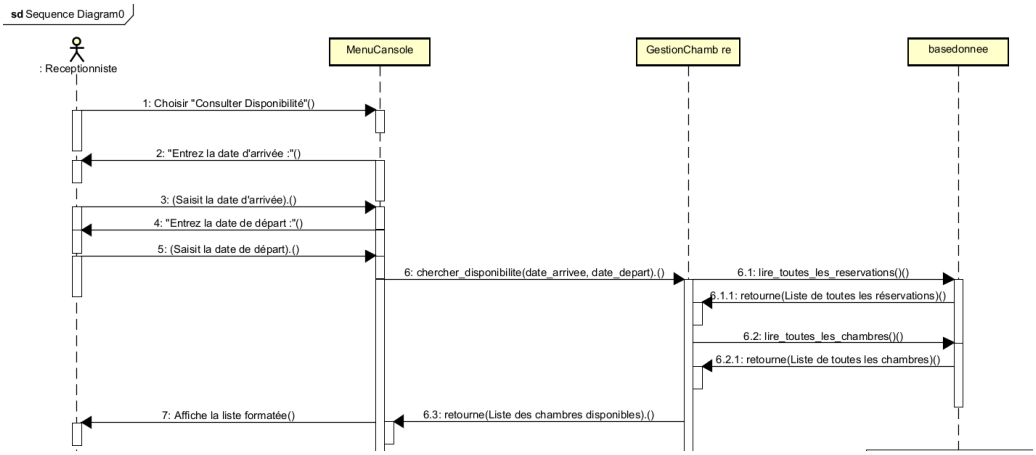

Figure 2.9 – *Séquence - Consultation des disponibilités*

Workflow de consultation pour les clients et réceptionnistes :

1. Saisie des dates de début et fin
2. Le système compare contre toutes les réservations actives
3. Pour chaque chambre : calcul du statut (libre/occupée)
4. Retour d'une liste des chambres libres avec détails (type, prix)

Cette fonctionnalité offre une vue en temps réel de l'inventaire disponible.

Portail Client

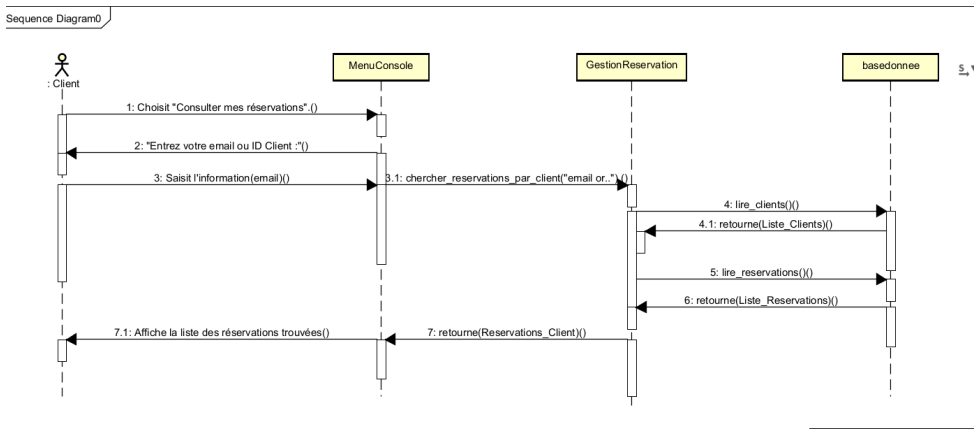

Figure 2.10 – *Séquence - Workflow du portail client*

Le workflow spécifique à l'acteur "Client" permet :

1. Client sélectionne "Consulter mes réservations"
2. Saisie Email/ID pour authentification
3. Recherche dans la base de données
4. Affichage de la liste personnalisée des réservations

Ce portail en libre-service réduit la charge des réceptionnistes.

2.3.4 Diagrammes de Séquence - Gestion des Entités

Ajouter un Client

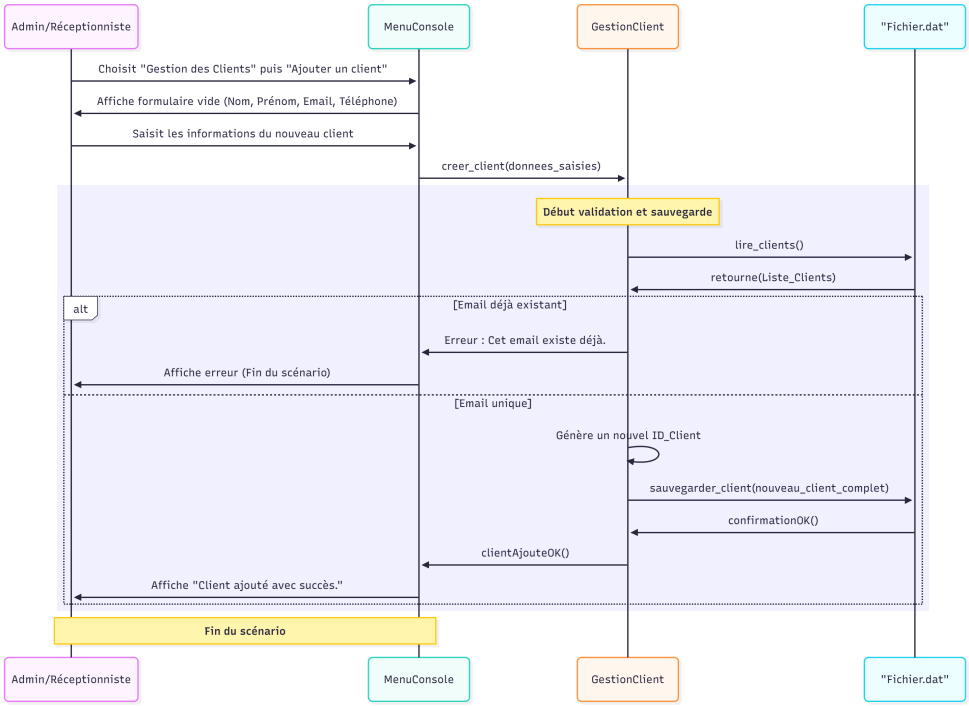

Figure 2.11 – *Séquence - Ajout d'un nouveau client*

La logique d'ajout de client assure l'unicité :

1. Utilisateur sélectionne "Ajouter un client"
2. Système affiche le formulaire vierge
3. Saisie des données (nom, prénom, email, téléphone)
4. Vérification dans `Fichier.dat` : l'email doit être unique
5. Si email existe : retour erreur "Client déjà enregistré"
6. Si unique : génération d'un nouvel ID client et sauvegarde

Correspondance UI : Le formulaire vide correspond à l'écran 4-3.png (AJOUTER NOUVEAU CLIENT).

Rechercher un Client

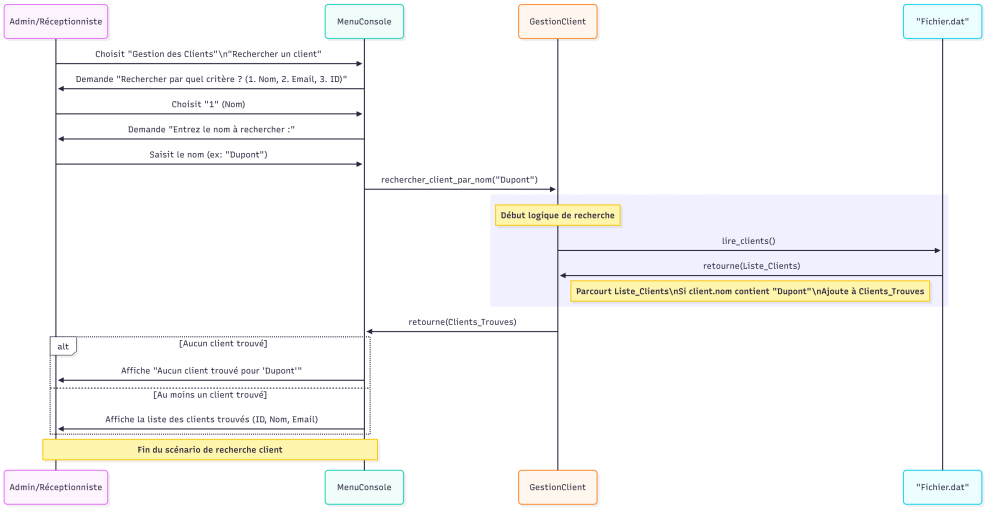

Figure 2.12 – *Séquence - Recherche de client dans la base*

Le mécanisme de recherche flexible supporte plusieurs critères :

1. Utilisateur choisit le critère (Nom, Email ou ID)
2. Saisie de la requête (ex : "Dupont")
3. Système parcourt la liste des clients
4. Retour des correspondances exactes ou partielles
5. Affichage des résultats en tableau

Correspondance UI : La saisie correspond à 3-5.png, les résultats à 3-6.png (tableau de résultats).

Ajouter une Chambre

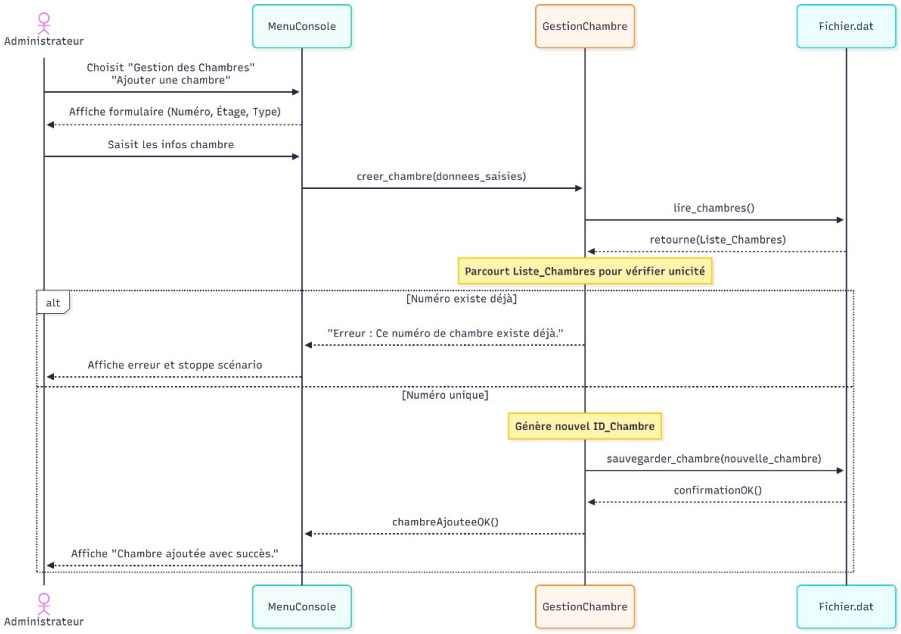

Figure 2.13 – *Séquence - Expansion de l'inventaire des chambres*

Le processus administratif d'ajout de chambre :

1. Administrateur saisit Numéro de chambre, Étage et Type
2. Vérification de l'unicité du numéro de chambre
3. Si numéro existe déjà : erreur "Chambre déjà enregistrée"
4. Si unique : ajout à l'inventaire avec génération automatique d'ID

Correspondance UI : Cette logique s'exécute dans l'interface 4-5.png (Add New Room - Debug Mode).

2.3.5 Diagramme de Séquence - Facturation

Générer une Facture

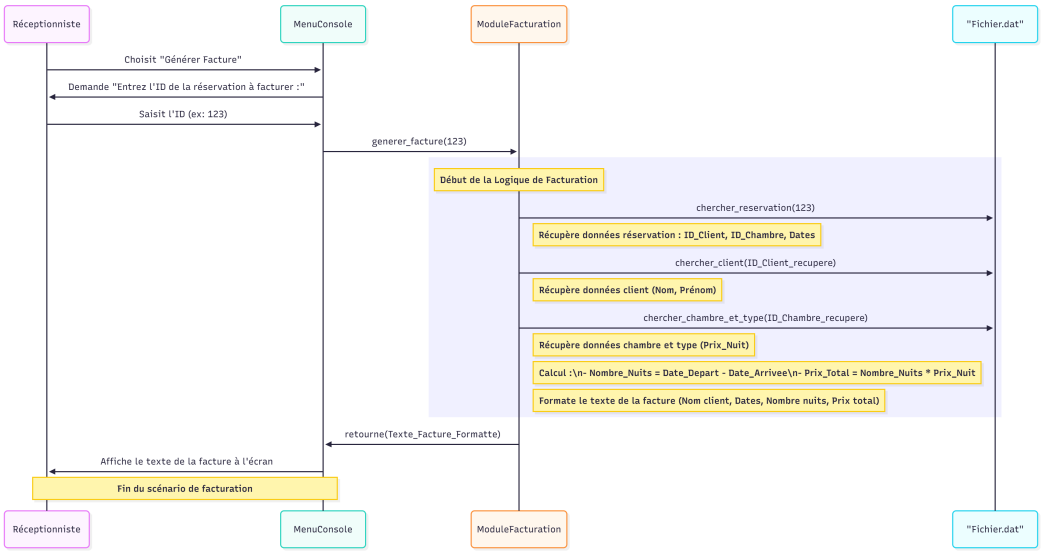

Figure 2.14 – *Séquence - Moteur de calcul et génération de factures*

Le module de facturation automatise le processus complet :

1. Réceptionniste saisit un ID de réservation (ex : 123)
2. Le `ModuleFacturation` récupère les données de trois sources :
   - `reservations.dat` : Dates et ID chambre
   - `clients.dat` : Nom et coordonnées du client
   - `chambres.dat` : Prix par nuit
3. Calcul explicite : $Total = (Date\_Départ - Date\_Arrivée) \times Prix\_Nuit$
4. Génération d'un ID unique de facture
5. Création de l'objet Facture complet
6. Sauvegarde dans `factures.dat`
7. Option "Print to File" : export vers `facture_N.txt`

Correspondance UI :

- L'affichage final correspond à 4-10.png (Facture #4 avec calcul : 731 nuits × 500.00 MAD = 365,500.00 MAD)
- L'export fichier correspond à 4-11.png (création de `facture_4.txt`)

2.4 Système de Contrôle d'Accès

2.4.1 Modèle RBAC

Notre application implémente un contrôle d'accès basé sur les rôles (Role-Based Access Control) avec trois profils utilisateur :

Rôle	
Administrateur	Accès complet : gestion utilisateurs, clients, chambres, réserv
Réceptionniste	Gestion clients, chambres, réservations, facturation (
Client	Portail libre-service : consultation disponibilités, créat

Table 2.1 – *Matrice des privilèges par rôle (RBAC)*

2.4.2 Flux d'Authentification

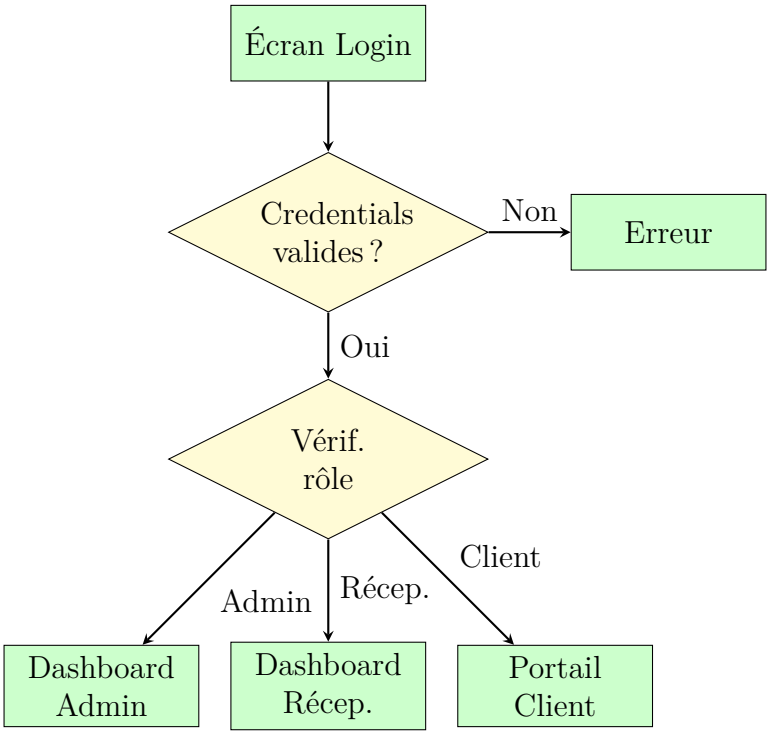

Figure 2.15 – *Flux d'authentification et routage par rôle*

2.5 Algorithmes Clés

2.5.1 Validation des Disponibilités

L'algorithme de vérification des disponibilités est crucial pour éviter les conflits de réservation :

```
1 fonction verifier_disponibilite(chambre, date_debut,
    date_fin):
2     pour chaque reservation dans reservations:
```

```
        si reservation.chambre == chambre ET
           reservation.statut == "ACTIVE" ET
           dates_se_chevauchent(reservation, date_debut,
              date_fin):
            retourner INDISPONIBLE
    retourner DISPONIBLE
```

Listing 2.5 – Pseudo-code de vérification des disponibilités

2.5.2 Calcul Précis des Nuitées

Nous utilisons les fonctions `mktime()` et `difftime()` de la bibliothèque standard C pour un calcul précis :

```c
int calculer_nuitees(const char* date_debut, const char*
    date_fin) {
    struct tm tm_debut = {0}, tm_fin = {0};

    // Parsing des dates
    sscanf(date_debut, "%d-%d-%d",
           &tm_debut.tm_year, &tm_debut.tm_mon, &tm_debut.
              tm_mday);
    sscanf(date_fin, "%d-%d-%d",
           &tm_fin.tm_year, &tm_fin.tm_mon, &tm_fin.tm_mday)
              ;

    // Ajustement format struct tm
    tm_debut.tm_year -= 1900;
    tm_debut.tm_mon -= 1;
    tm_fin.tm_year -= 1900;
    tm_fin.tm_mon -= 1;

    // Conversion et calcul
    time_t time_debut = mktime(&tm_debut);
    time_t time_fin = mktime(&tm_fin);
    double diff_secondes = difftime(time_fin, time_debut);

    return (int)(diff_secondes / 86400); // 86400 sec par
       jour
}
```

Listing 2.6 – Calcul des nuitées avec mktime

Chapitre 3

Réalisation Technique

3.1 Organisation du Code Source

3.1.1 Structure des Répertoires

```
hotel_reservation_system/
|-- src/                    # Logique metier
|   |-- clients.c
|   |-- chambres.c
|   |-- reservations.c
|   |-- fichiers.c
|   |-- auth.c
|   +-- data_init.c
|-- ui/                     # Interface utilisateur
|   |-- ui.c
|   |-- ui_draw.c
|   |-- ui_theme.c
|   |-- ui_login.c
|   |-- ui_client_portal.c
|   +-- ...
|-- include/                # En-tetes
|   |-- structures.h
|   |-- clients.h
|   |-- chambres.h
|   +-- ...
|-- data/                   # Fichiers de donnees
+-- Makefile
```

3.1.2 Compilation et Dépendances

Le projet utilise `make` pour automatiser la compilation :

```
1 CC = gcc
2 CFLAGS = -Wall -Wextra -Iinclude
3 LDFLAGS = -lncursesw
4
5 SOURCES = $(wildcard src/*.c ui/*.c)
```

```
6  OBJECTS = $(SOURCES:.c=.o)
7
8  hotel_app.exe: $(OBJECTS) main.o
9      $(CC) $(CFLAGS) -o $@ $^ $(LDFLAGS)
10
11 clean:
12     rm -f $(OBJECTS) main.o hotel_app.exe
```

Listing 3.1 – Extrait du Makefile

3.2 Modules Fonctionnels

3.2.1 Module de Gestion des Clients

Le module `clients.c` implémente les opérations CRUD :

- `ajouter_client()` : Ajout avec génération automatique d'ID
- `modifier_client()` : Mise à jour des informations
- `supprimer_client()` : Suppression avec vérification des contraintes
- `rechercher_client()` : Recherche par nom, ID ou email

3.2.2 Module de Gestion des Réservations

Le module `reservations.c` gère le cycle de vie complet :

- Validation des dates (pas de dates passées, début < fin)
- Vérification des disponibilités en temps réel
- Calcul automatique du montant basé sur les nuitées
- Gestion des statuts (ACTIVE, ANNULEE)
- Soft delete pour conservation de l'historique

3.2.3 Module d'Interface Utilisateur

L'interface ncurses offre une expérience professionnelle :

- **Système de couleurs** : 8 paires de couleurs définies dans `ui_theme.c`
- **Formulaires wizards** : Navigation Tab/Shift-Tab entre champs
- **Validation en temps réel** : Retour visuel immédiat
- **Composants réutilisables** : Boîtes de dialogue, listes scrollables

3.3 Gestion de la Mémoire

3.3.1 Stratégie d'Allocation

Nous utilisons une approche hybride :

- **Tableaux statiques** : Pour les limites connues (MAX_CLIENTS = 100)
- **Allocation dynamique** : Pour les buffers temporaires
- **Libération systématique** : Chaque `malloc()` a son `free()`

3.3.2 Prévention des Fuites Mémoire

```c
char* buffer = malloc(256 * sizeof(char));
if (buffer == NULL) {
    // Gestion d'erreur
    return ERROR_MALLOC;
}

// Utilisation du buffer
strncpy(buffer, source, 255);
buffer[255] = '\0';

// Lib ration syst matique
free(buffer);
buffer = NULL;
```

Listing 3.2 – Exemple de gestion propre de la mémoire

3.4 Persistance des Données

3.4.1 Format Binaire

Les données sont stockées en format binaire pour :

- Rapidité de lecture/écriture (pas de parsing)
- Compacité (pas de métadonnées textuelles)
- Intégrité (typage fort conservé)

3.4.2 Opérations de Fichier

```c
void sauvegarder_clients(Client clients[], int nb_clients) {
    FILE* f = fopen("data/clients.dat", "wb");
    if (f == NULL) {
        perror("Erreur ouverture fichier");
        return;
    }

    fwrite(&nb_clients, sizeof(int), 1, f);
    fwrite(clients, sizeof(Client), nb_clients, f);

    fclose(f);
}
```

Listing 3.3 – Sauvegarde des clients

Chapitre 4

Validation et Interface Utilisateur

4.1 Système d'Authentification et Enregistrement

4.1.1 Écran de Connexion

Le système démarre sur un écran de connexion professionnel offrant une expérience utilisateur sécurisée et intuitive.

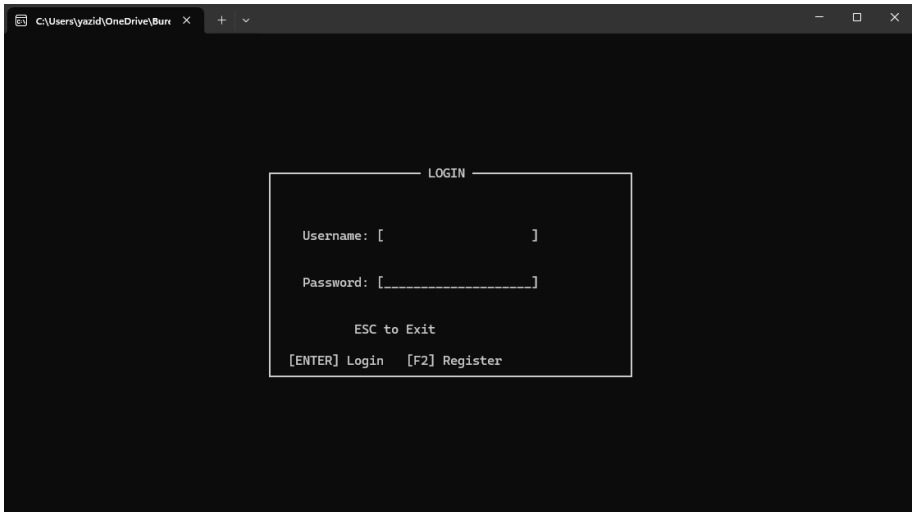

Figure 4.1 – *Écran principal de connexion - Saisie des identifiants*

L'écran de login présente une boîte de dialogue "LOGIN" sollicitant le nom d'utilisateur et le mot de passe. Les options de navigation sont clairement indiquées : ESC pour quitter l'application, ENTER pour se connecter, et F2 pour accéder à l'écran d'enregistrement. Le mot de passe est automatiquement masqué pour garantir la confidentialité.

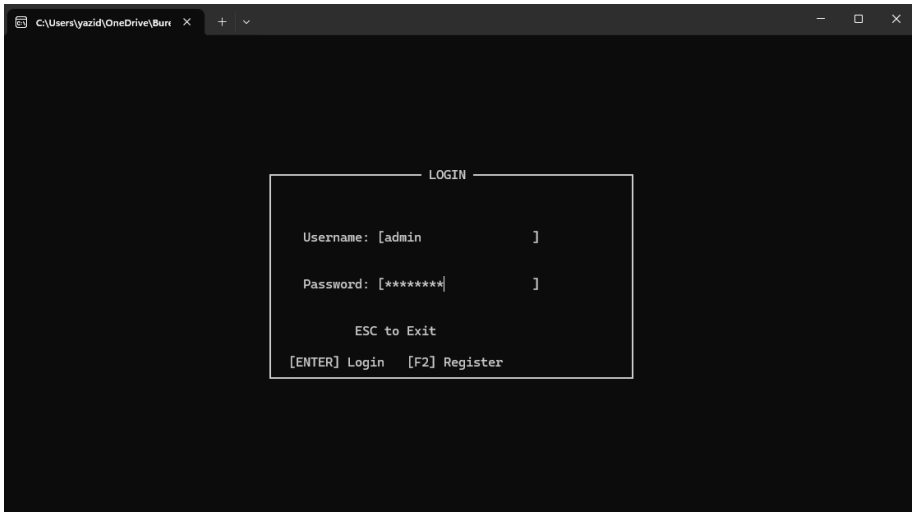

Figure 4.2 – *Tentative de connexion administrative - Pré-remplissage du champ utilisateur*

Cette capture montre une tentative de connexion avec un compte administrateur. Le champ "Username" est pré-rempli avec "admin", démontrant la capacité du système à mémoriser les dernières saisies pour faciliter l'expérience utilisateur.

4.1.2 Processus d'Enregistrement

Le système offre un processus d'inscription guidé permettant la création de nouveaux comptes utilisateurs avec attribution de rôles.

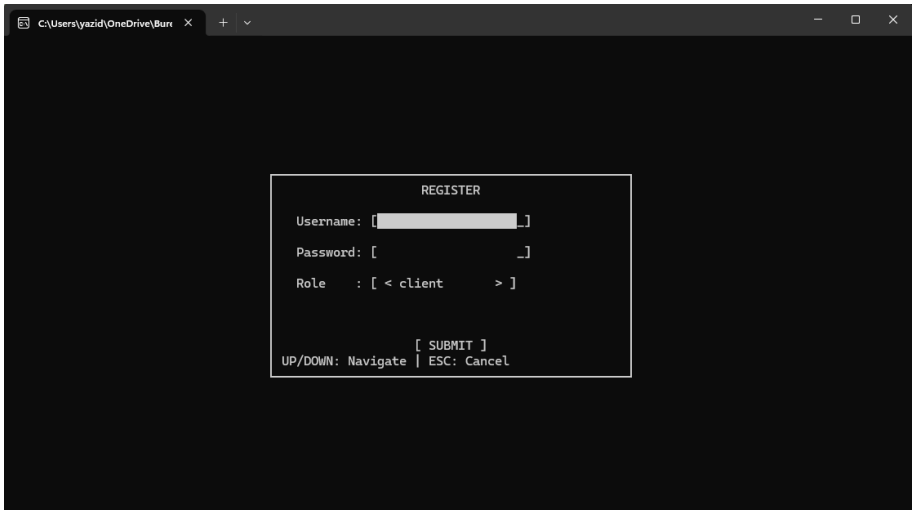

Figure 4.3 – *Écran d'enregistrement - État initial avec focus sur le champ Username*

L'écran d'enregistrement dans son état initial montre le focus (surligné en gris) sur le champ "Username", prêt à recevoir la saisie utilisateur. Le rôle par défaut est configuré sur `< client >`, mais peut être modifié selon les besoins.

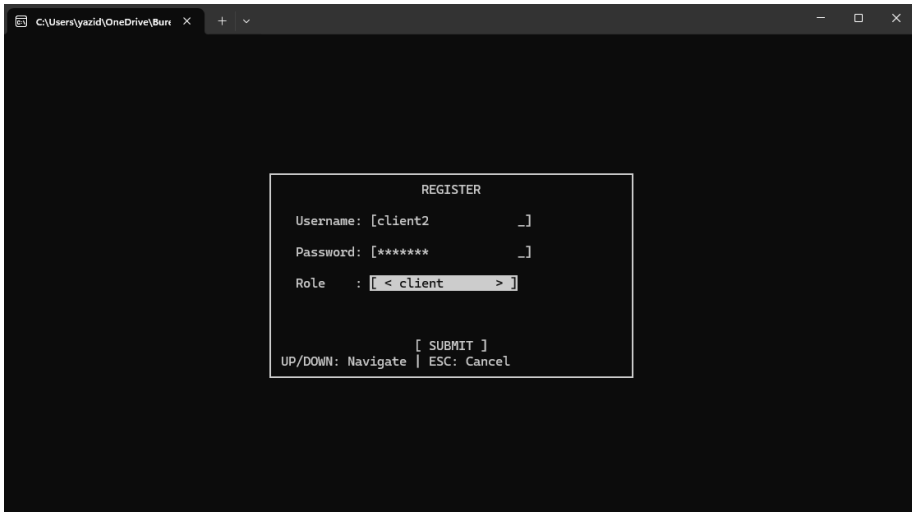

Figure 4.4 – *Formulaire d'enregistrement complété - Compte client*

Cette capture présente le formulaire d'enregistrement entièrement rempli. L'utilisateur a saisi "client2" comme nom d'utilisateur, un mot de passe masqué, et le rôle sélectionné est "client". Le bouton [SUBMIT] est visible et prêt à être activé.

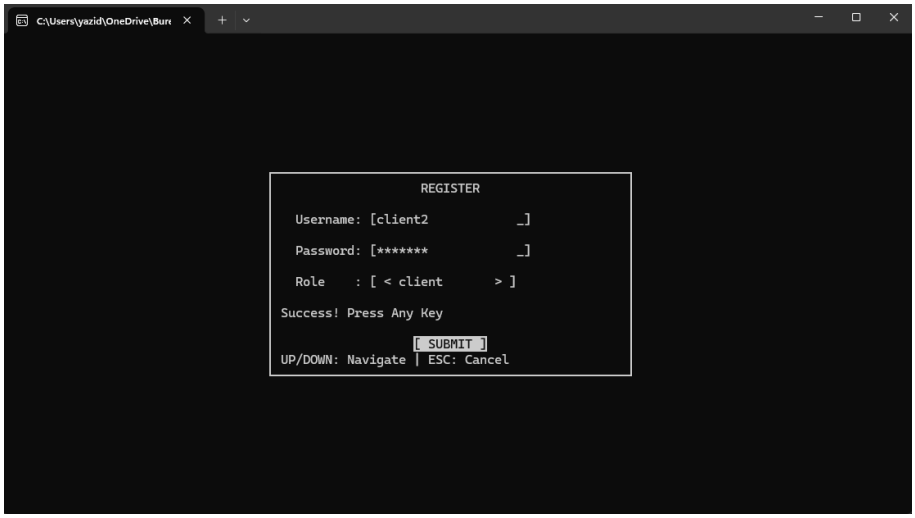

Figure 4.5 – *Confirmation d'enregistrement réussi pour l'utilisateur client2*

L'écran de confirmation après un enregistrement réussi affiche les détails du compte "client2" avec le message "Success ! Press Any Key", validant la création du compte dans le système.

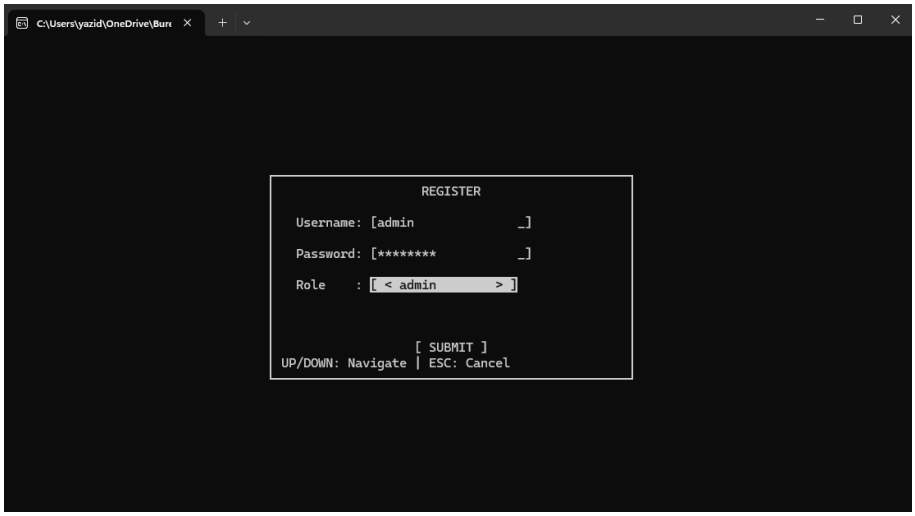

Figure 4.6 – *Enregistrement d'un utilisateur administrateur - Sélection du rôle admin*

Cette capture montre le formulaire d'enregistrement configuré pour un utilisateur administratif. Le nom d'utilisateur est défini sur "admin" et le sélecteur de rôle a été basculé sur `< admin >`, accordant les privilèges d'administration complets.

4.2 Tableaux de Bord et Interfaces Utilisateur

4.2.1 Dashboard Administrateur

Le tableau de bord administrateur fournit une vue d'ensemble complète des métriques hôtelières et de l'état du système.

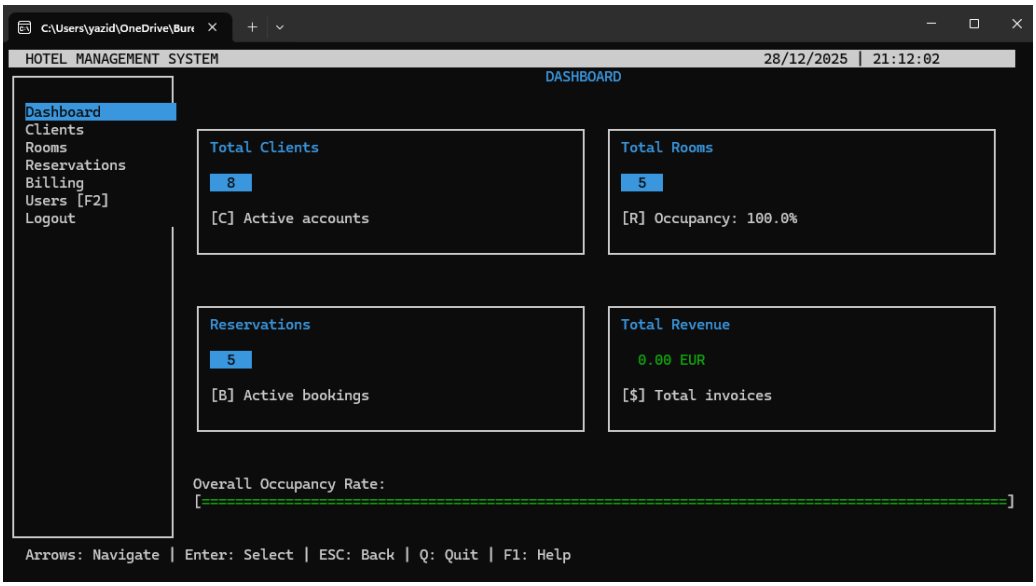

Figure 4.7 – *Tableau de bord administrateur - Vue d'ensemble des métriques hôtelières*

Le dashboard principal de l'administrateur présente quatre cartes de synthèse essentielles :

- **Total Clients** : Affiche "8" comptes actifs
- **Total Rooms** : Montre "5" chambres avec un taux d'occupation calculé de "100.0%"
- **Reservations** : Indique "5" réservations actives

- **Total Revenue** : Actuellement à "0.00 EUR"

Une barre de progression visuelle en bas de l'écran illustre le "Overall Occupancy Rate" (taux d'occupation global), offrant une visualisation immédiate de la performance de l'hôtel.

4.2.2 Dashboard Client

Les clients disposent d'un portail dédié avec fonctionnalités en libre-service.

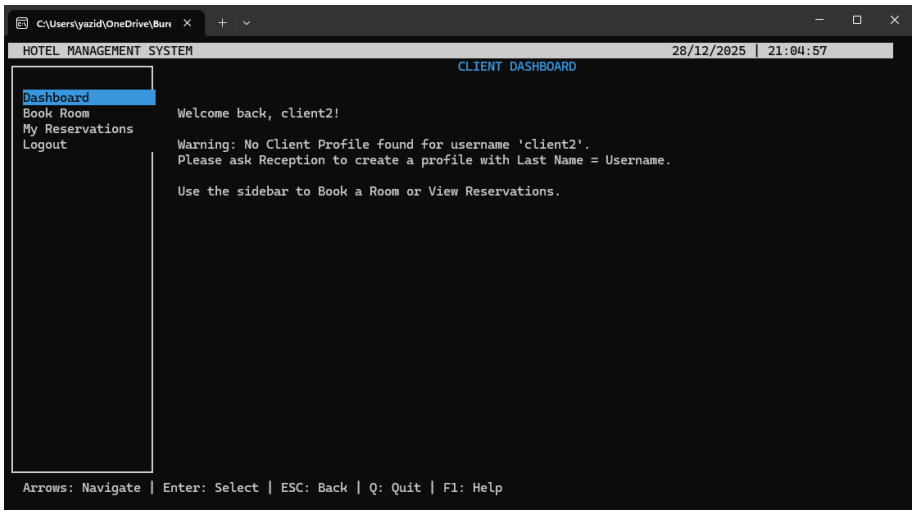

Figure 4.8 – *Tableau de bord client - Avertissement de profil manquant*

Cette capture présente le dashboard client immédiatement après connexion en tant que "client2". Un message d'avertissement s'affiche : "Warning : No Client Profile found for username 'client2'. Please ask Reception to create a profile...". Cela illustre la distinction entre l'authentification (compte existant) et le profil client (détails personnels), garantissant l'intégrité relationnelle du système.

4.3 Gestion des Clients

4.3.1 Liste et Recherche de Clients

Le module de gestion des clients offre des capacités complètes de visualisation et de recherche.

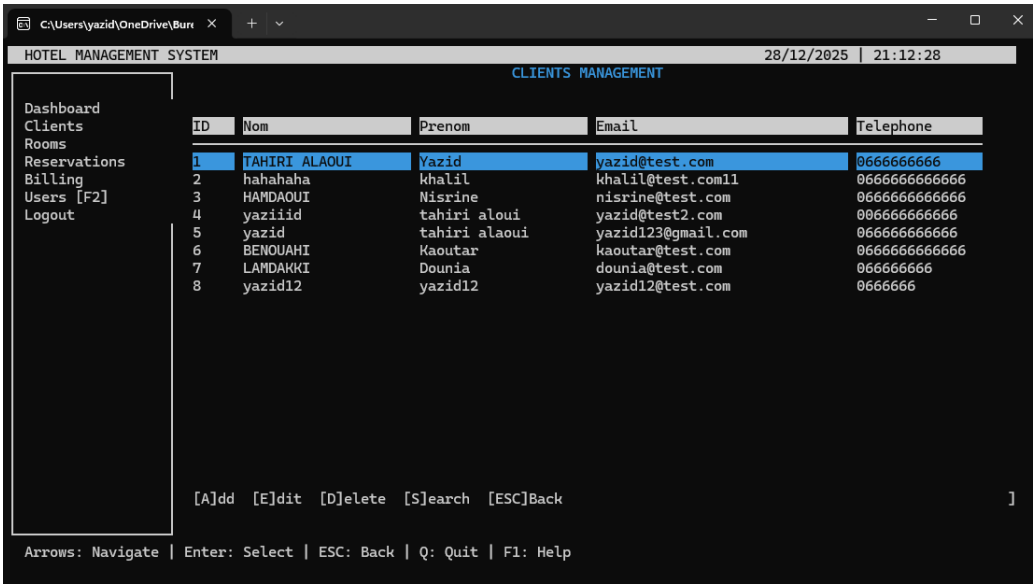

Figure 4.9 – *Écran de gestion des clients - Liste complète des enregistrements*

L'écran de gestion des clients présente tous les utilisateurs enregistrés dans un format tabulaire avec les colonnes : ID, Nom, Prénom, Email et Téléphone. On peut observer des entrées notables comme "TAHIRI ALAOUI Yazid" (ID 1) et "HAMDAOUI Nisrine" (ID 3). Le pied de page offre des options pour [A]jouter, [E]diter, [D]upprimer et [S]earcher des clients.

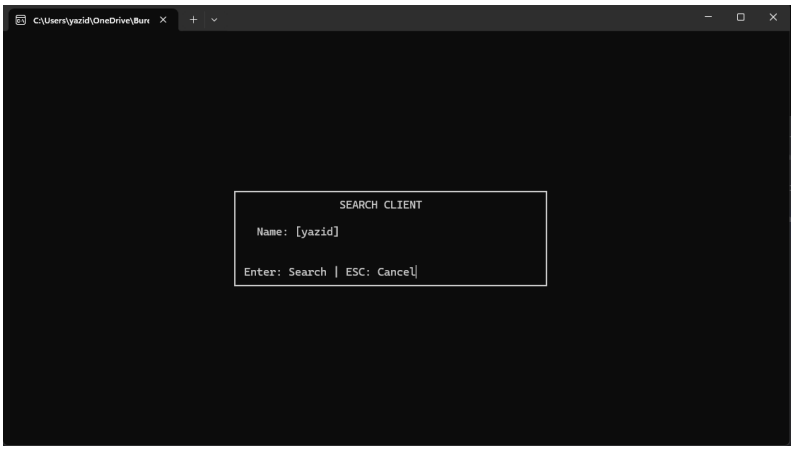

Figure 4.10 – *Boîte de dialogue de recherche client - Saisie du critère*

La boîte de recherche client invite l'utilisateur à entrer un nom pour trouver des enregistrements spécifiques. Dans cet exemple, "[yazid]" a été saisi comme terme de recherche.

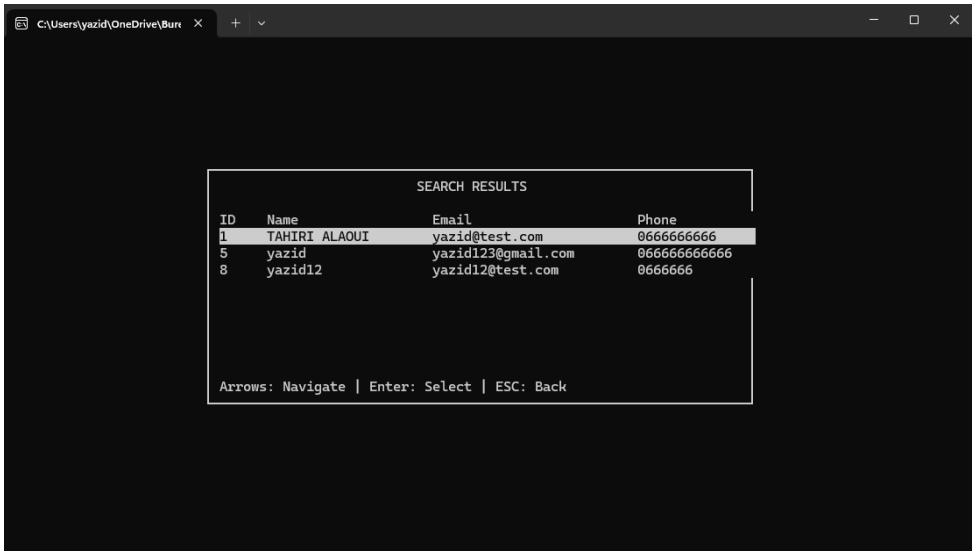

Figure 4.11 – *Résultats de recherche - Correspondances pour la requête "yazid"*

La fenêtre des résultats de recherche affiche les correspondances pour la requête "yazid", listant trois résultats avec leurs IDs, Noms, Emails et Numéros de téléphone. Cette fonctionnalité permet une localisation rapide des clients dans une base de données volumineuse.

4.3.2 Ajout de Nouveau Client

L'interface d'ajout de client permet la saisie manuelle de nouvelles fiches.

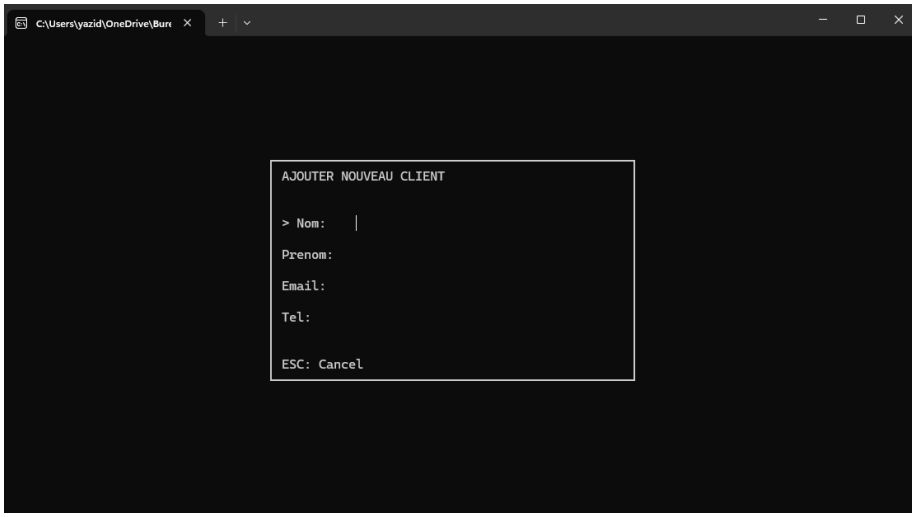

Figure 4.12 – *Formulaire d'ajout de nouveau client - Champs vierges*

Le formulaire "AJOUTER NOUVEAU CLIENT" fournit des champs vides permettant à l'administrateur de saisir manuellement les informations d'un nouveau client : Nom, Prénom, Email et Téléphone. L'ID est généré automatiquement par le système.

4.4 Gestion de l'Inventaire des Chambres

4.4.1 Vue d'Ensemble et Modification

Le module de gestion des chambres permet le suivi complet de l'inventaire hôtelier.

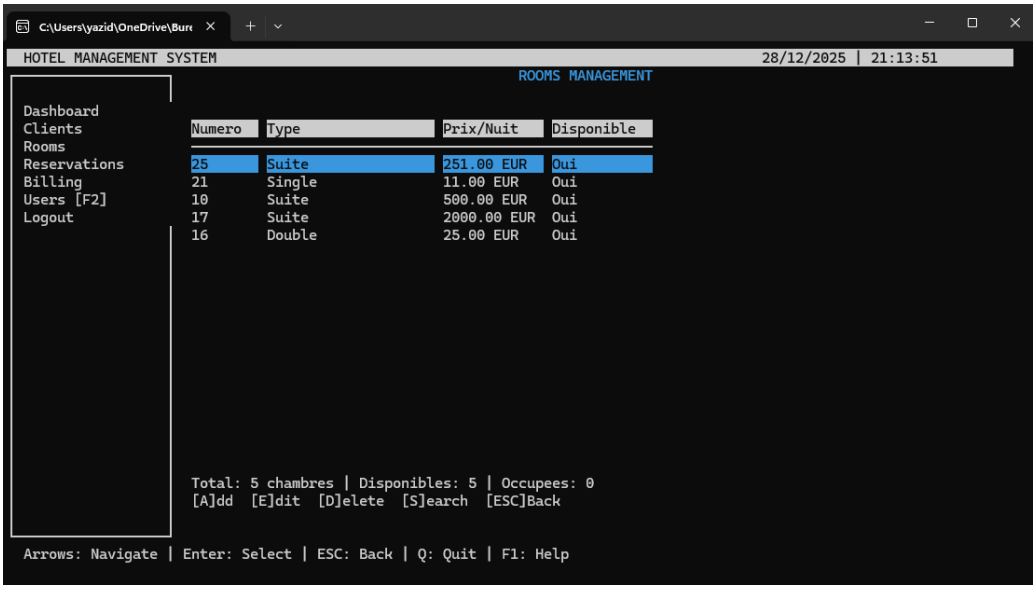

Figure 4.13 – *Gestion des chambres - Inventaire complet avec statuts*

L'écran de gestion des chambres liste l'inventaire de l'hôtel avec les colonnes : Numéro, Type, Prix/Nuit et Disponibilité. Le pied de page résume l'inventaire : "Total : 5 chambres | Disponibles : 5 | Occupees : 0", offrant une vue instantanée de la capacité hôtelière.

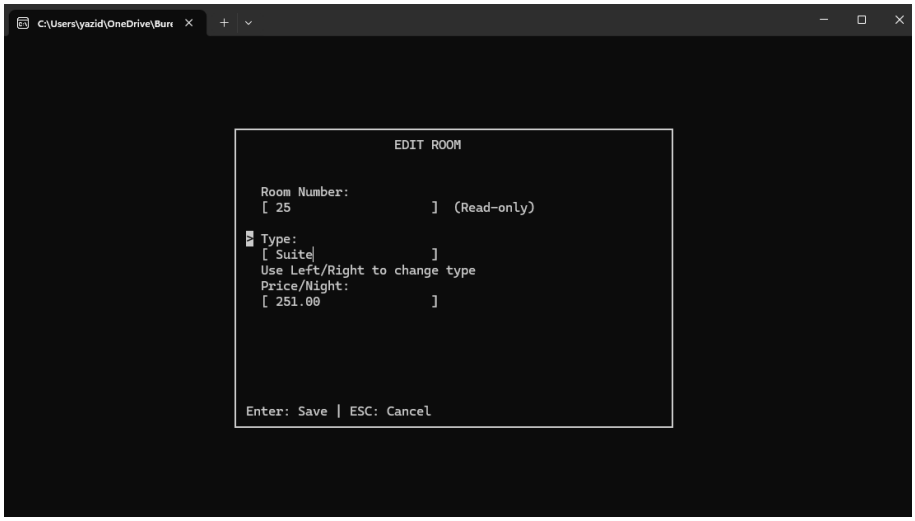

Figure 4.14 – *Interface de modification de chambre - Édition des paramètres*

L'interface "Edit Room" permet à l'administrateur de modifier les détails d'une chambre existante. Le "Room Number" (25) est en lecture seule pour préserver l'intégrité référentielle. L'utilisateur peut basculer le Type (actuellement "Suite") avec les touches fléchées Gauche/Droite et modifier le Prix/Nuit (actuellement "251.00").

4.4.2 Ajout de Chambres

Le système permet l'expansion de l'inventaire via un formulaire dédié.

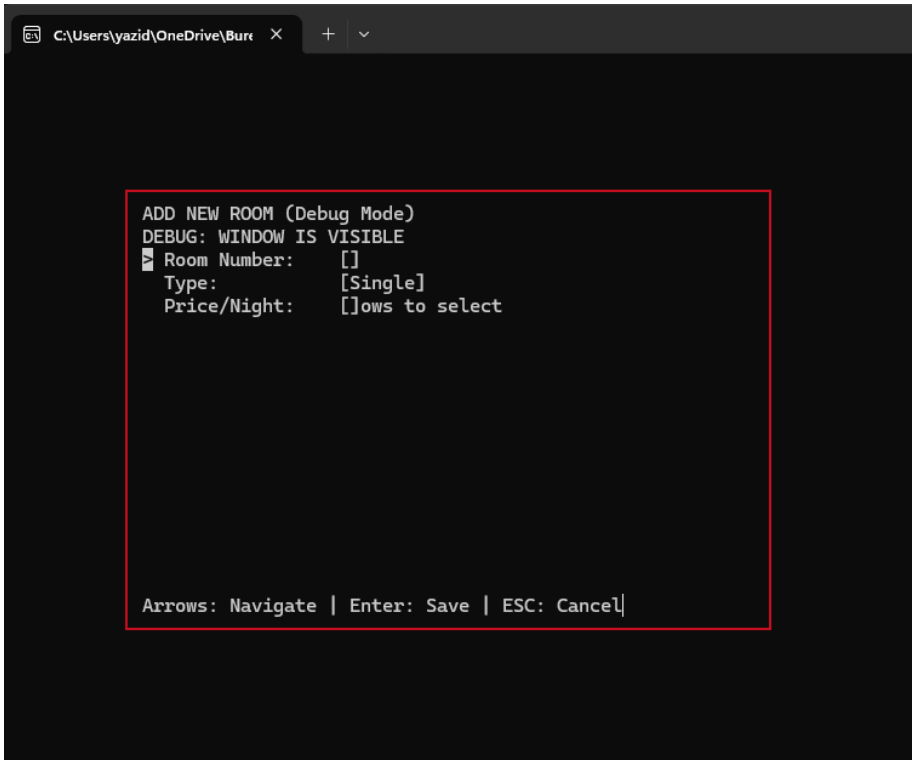

Figure 4.15 – *Ajout de nouvelle chambre en mode Debug - Saisie des paramètres*

L'écran "Add New Room" en mode Debug affiche un flag "DEBUG : WINDOW IS VISIBLE" et permet la saisie des paramètres pour un nouveau bien immobilier : Numéro de chambre, Type (par défaut Simple) et Prix/Nuit. Ce mode debug aide au développement et au dépannage.

4.5 Système de Réservation

4.5.1 Navigation et Réservation Côté Client

Les clients peuvent parcourir l'inventaire et effectuer des réservations en autonomie.

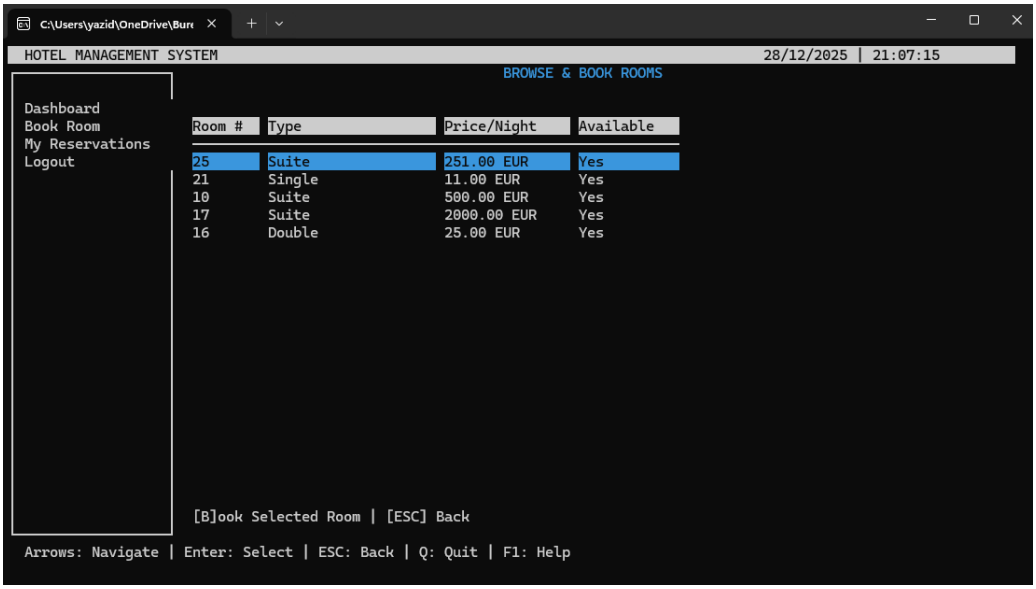

Figure 4.16 – *Interface Browse & Book Rooms - Sélection de chambre disponible*

L'interface "Browse & Book Rooms" présente un tableau listant les Numéros de chambre, Types, Prix et Disponibilité. La chambre 25 (une Suite) est actuellement surlignée en bleu, indiquant la sélection en cours pour une potentielle réservation.

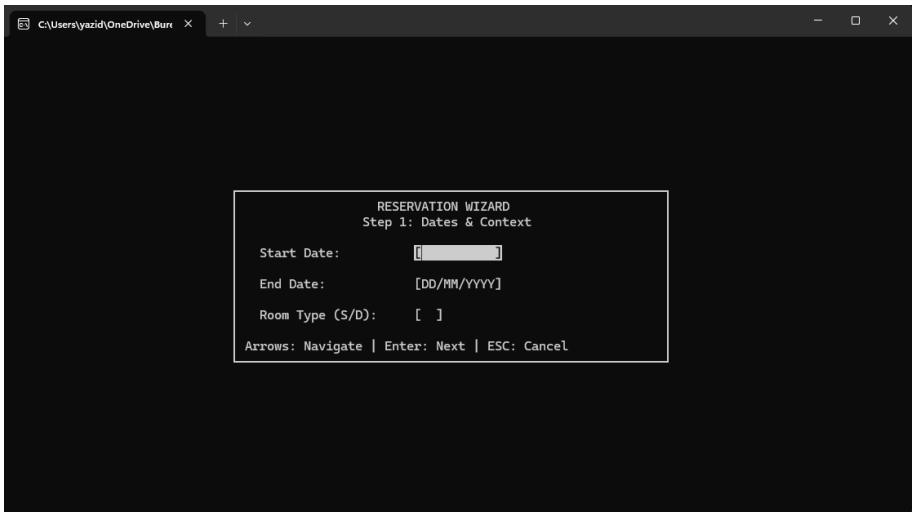

Figure 4.17 – *Assistant de réservation - Étape 1 : Saisie des dates et contexte*

Le Reservation Wizard - Step 1 demande à l'utilisateur de saisir les "Dates & Context" pour une réservation, spécifiquement la Date de début, Date de fin et le Type de chambre désiré (Simple/Double). Cette approche wizard guide l'utilisateur pas à pas.

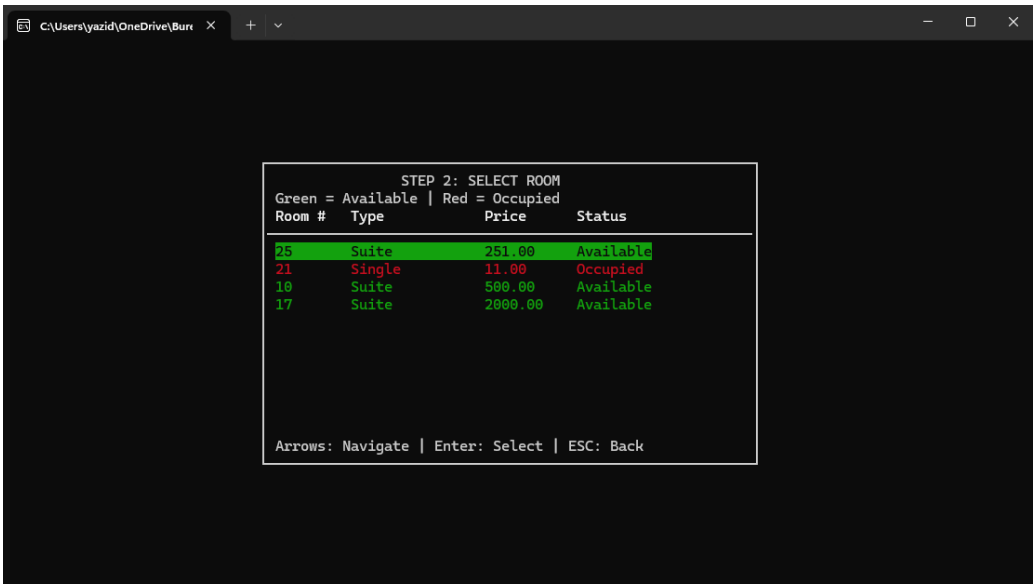

Figure 4.18 – *Assistant de réservation - Étape 2 : Sélection avec codage couleur*

Le Reservation Wizard - Step 2 permet à l'utilisateur de sélectionner une chambre spécifique. Un codage couleur intuitif est utilisé : le texte Vert indique les chambres "Available" (disponibles), tandis que le texte Rouge indique les chambres "Occupied" (occupées), par exemple la chambre 21 est occupée.

4.5.2 Workflow Administratif de Réservation

Les administrateurs et réceptionnistes disposent d'outils avancés de gestion des réservations.

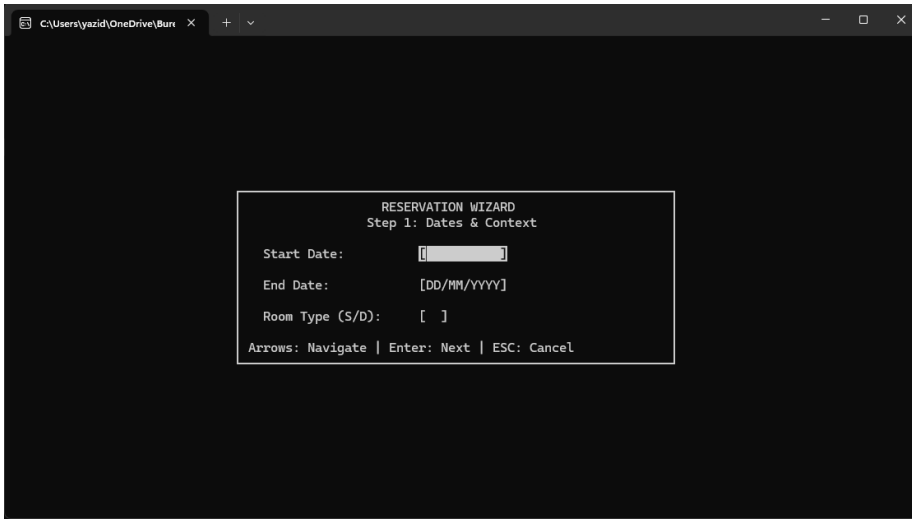

Figure 4.19 – *Assistant de réservation (vue admin) - Étape 1 : Configuration initiale*

Le Reservation Wizard - Step 1 côté administrateur présente l'écran initial pour créer une nouvelle réservation. Il invite l'utilisateur à entrer la Date de début, Date de fin et le Type de chambre préféré (Single/Double), avec une interface similaire au portail client mais dans un contexte administratif.

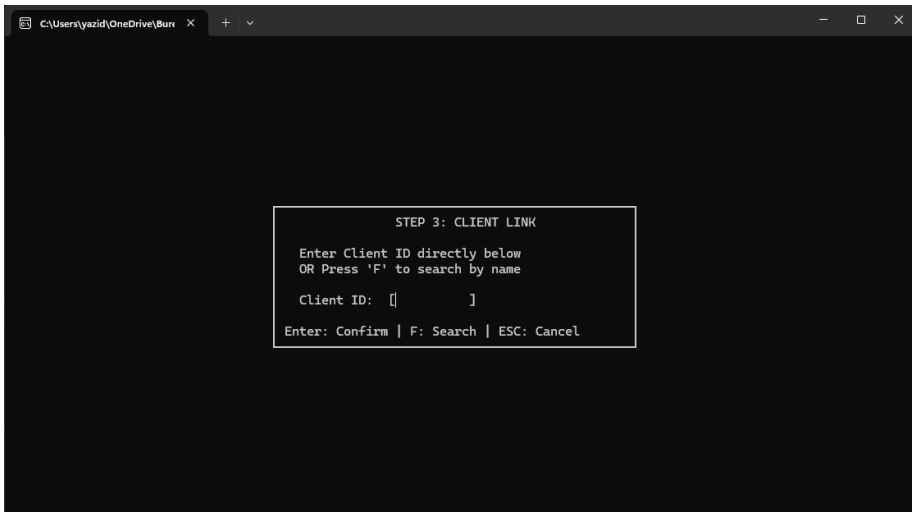

Figure 4.20 – *Assistant de réservation - Étape 3 : Association avec le profil client*

L'étape 3 "Client Link" connecte une réservation à un profil utilisateur. Cette étape offre deux options : saisir directement le Client ID ou appuyer sur 'F' pour rechercher un client par son nom dans la base de données, facilitant l'association correcte.

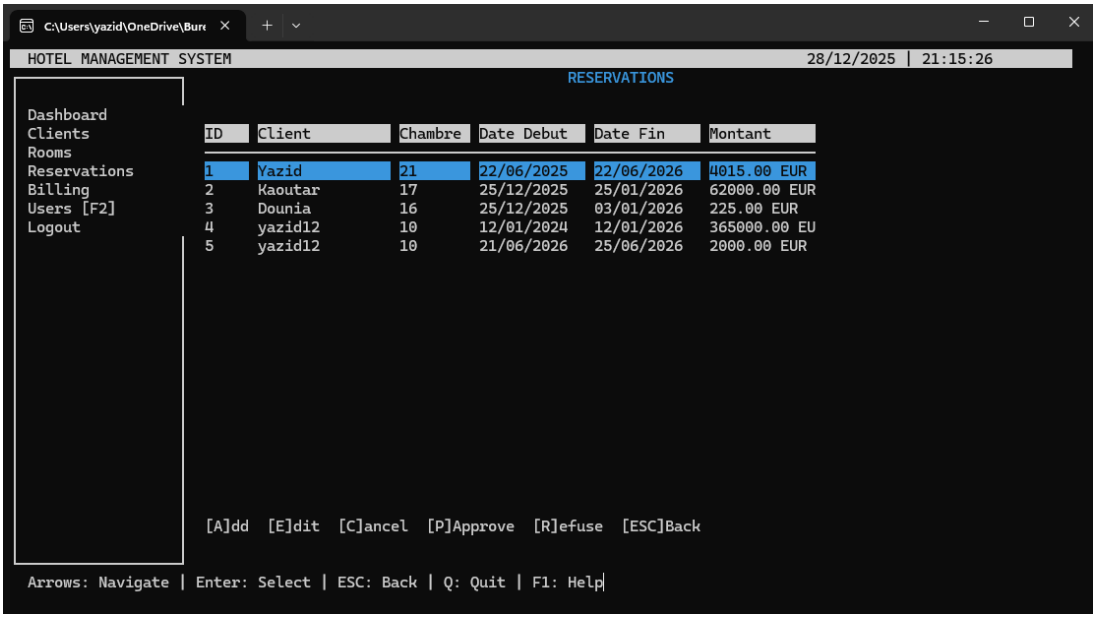

Figure 4.21 – *Gestion des réservations - Liste complète avec détails et montants*

L'écran "Reservations Management" affiche une liste exhaustive de toutes les réservations avec les colonnes : ID, Nom du client, Numéro de chambre, Date de début, Date de fin et Montant total. Une réservation notable apparaît pour le client "yazid12" (ID 4) pour la chambre 10, couvrant 2 ans (2024-2026) pour un total de "365000.00 EU". Le pied de page offre des options pour [A]pprouver ou [R]efuser des réservations, ainsi que les fonctions standard d'édition et d'annulation.

4.6 Système de Facturation

4.6.1 Génération et Consultation de Factures

Le module de facturation automatise le calcul et la génération de documents.

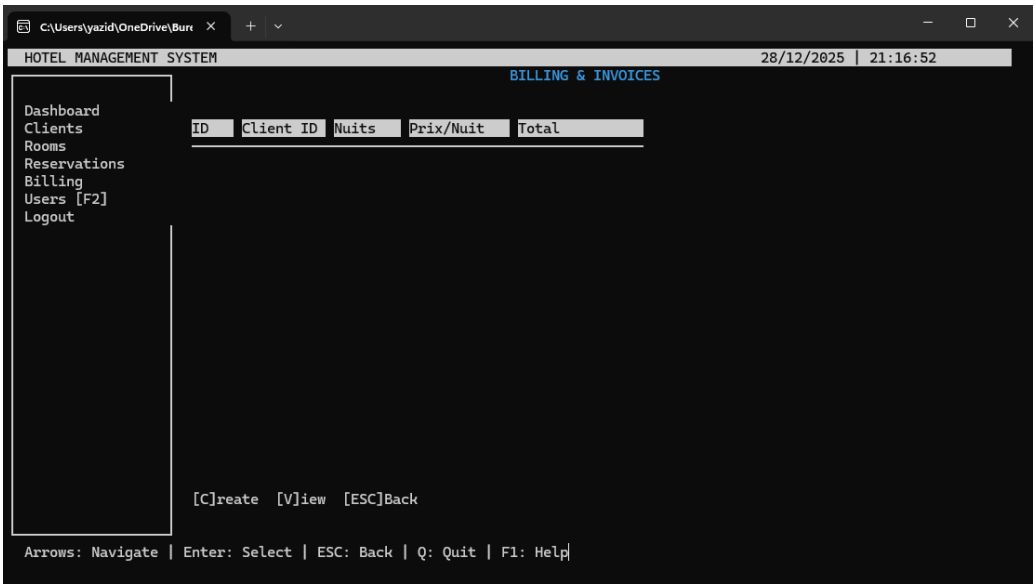

Figure 4.22 – *Menu principal Billing & Invoices - Structure de table vide*

Le menu principal "Billing & Invoices" présente une structure de table prête à afficher

les enregistrements de factures avec les colonnes : ID, Client ID, Nuits, Prix/Nuit et Total. Les options au bas permettent de [C]réer une nouvelle facture ou [V]isualiser une facture existante.

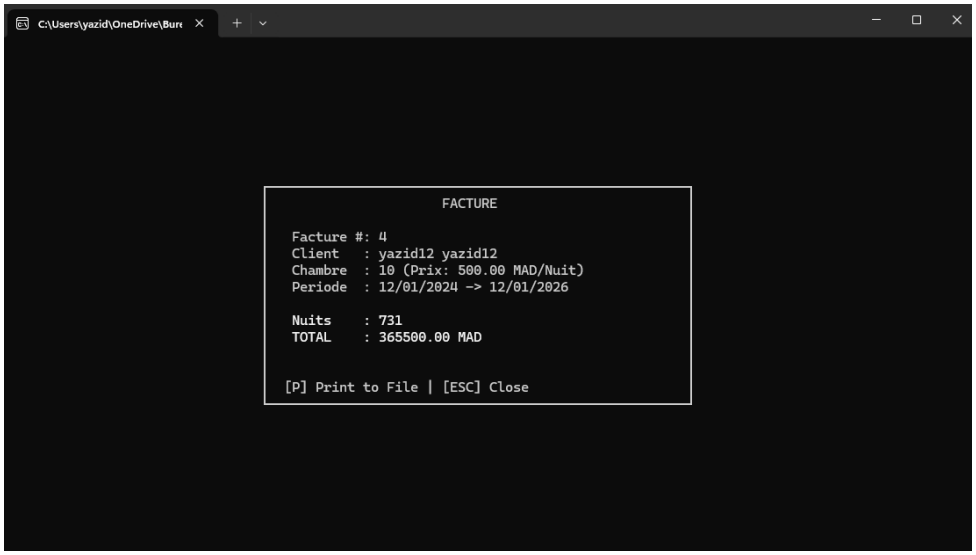

Figure 4.23 – *Vue détaillée d'une facture générée - Facture #4*

La vue "FACTURE" affiche les détails pour la "Facture # : 4", émise au client "yazid12". La facture couvre la chambre 10 pour une période de 731 nuits (du 12/01/2024 au 12/01/2026), résultant en un total de 365,500.00 MAD. Une option spécifique [P] Print to File est disponible pour exporter ces données vers un fichier texte.

4.6.2 Export et Archivage

Le système permet l'exportation des factures pour archivage et impression.

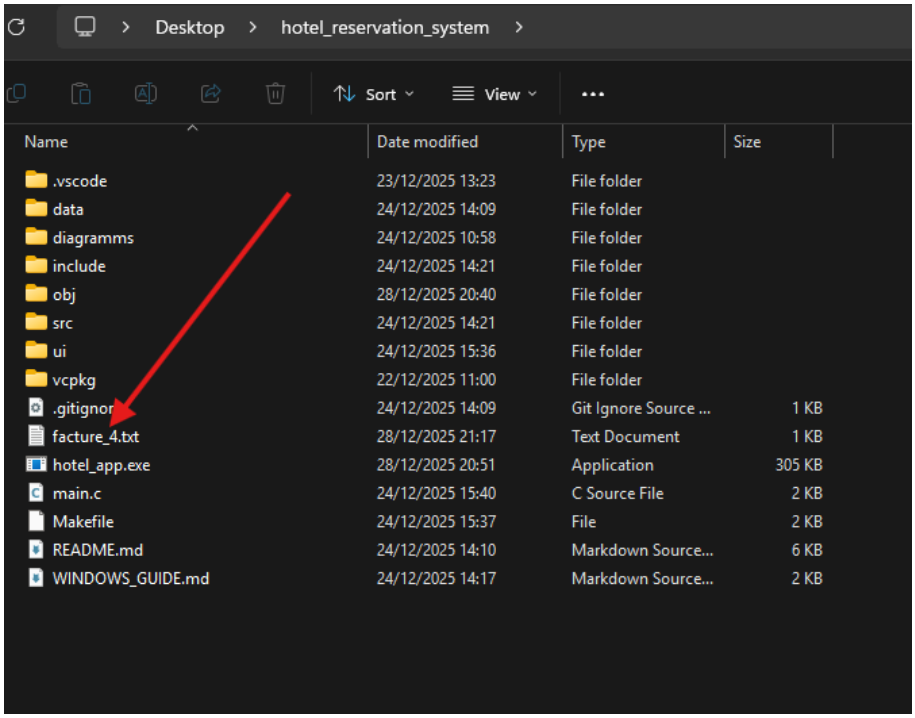

Figure 4.24 – *Explorateur Windows - Confirmation d'export de facture*

Cette capture d'écran de l'Explorateur de fichiers Windows montre le répertoire du projet. Une flèche rouge pointe vers un fichier nouvellement généré nommé `facture_4.txt`, confirmant que la fonction "Print to File" de l'écran précédent a bien créé une facture au format texte, garantissant la traçabilité et l'archivage des transactions.

4.7 Tests de Validation

4.7.1 Scénarios de Test Fonctionnels

Le système a été validé à travers des scénarios de test couvrant tous les cas d'usage critiques.

Scénario	Description
Authentification Admin	Login avec identifiants valides
Enregistrement Client	Remplir formulaire d'inscription
Ajout Client	Saisie formulaire par admin
Réservation Valide	Dates valides et chambre libre
Conflit de Réservation	Dates se chevauchant avec existant
Recherche Client	Recherche par nom ou ID
Facturation	Sélectionner une réservation finie
Export Facture	Génération du fichier .txt
Portail Client	Login client et consultation

Table 4.1 – *Scénarios de tests fonctionnels validés*

4.7.2 Validation des Contraintes Métier

Toutes les contraintes métier et règles de gestion ont été rigoureusement testées et validées :

Unicité des identifiants : IDs uniques pour clients, réservations et factures

Validation des formats : Dates au format YYYY-MM-DD, emails valides

Contraintes temporelles : Interdiction des dates passées pour nouvelles réservations

Cohérence des dates : Vérification date_début < date_fin

Détection des conflits : Algorithme de chevauchement empêchant double réservation

Calculs précis : Nuitées calculées avec mktime/difftime pour précision

Persistance : Données correctement sauvegardées et récupérées entre sessions

Intégrité référentielle : Liens cohérents entre clients et réservations

Contrôle d'accès : RBAC respecté selon les rôles définis

Conclusion

Bilan du Projet

La réalisation de ce système de gestion hôtelière a constitué une expérience formatrice à plusieurs niveaux. Sur le plan technique, nous avons consolidé notre maîtrise du langage C, en particulier dans ses aspects avancés : manipulation de pointeurs, gestion dynamique de la mémoire, programmation modulaire et utilisation de bibliothèques système.

L'intégration de la bibliothèque ncurses nous a permis de découvrir les subtilités de la programmation d'interfaces textuelles professionnelles, un domaine souvent négligé au profit des interfaces graphiques modernes, mais qui reste pertinent pour les systèmes embarqués et les environnements serveur.

Défis Rencontrés

Plusieurs obstacles techniques ont jalonné le développement :

- **Gestion des dates** : L'implémentation d'un calcul précis des nuitées a nécessité l'utilisation de `mktime()` et `difftime()`, après avoir constaté les limitations des approches arithmétiques naïves.
- **Contrôle ncurses** : La maîtrise des modes de saisie (notamment le mode `cbreak` et la désactivation de l'écho) a demandé une compréhension approfondie de la documentation.
- **Persistance binaire** : La gestion robuste des erreurs lors des opérations de fichiers a été essentielle pour garantir l'intégrité des données.
- **Architecture RBAC** : L'implémentation du contrôle d'accès basé sur les rôles a nécessité une refonte partielle de la structure de navigation.

Acquis Techniques

Ce projet nous a permis de développer des compétences solides en :

- Conception d'architecture logicielle multicouche
- Modélisation de données et gestion des relations entre entités
- Implémentation d'algorithmes de validation et de recherche
- Gestion rigoureuse de la mémoire en C
- Développement d'interfaces utilisateur avec ncurses
- Utilisation d'outils de build (Make)
- Versionnement de code (Git)

Perspectives d'Évolution

Plusieurs axes d'amélioration sont envisageables :

Fonctionnalités

- **Statistiques avancées** : Taux d'occupation, revenus mensuels, clients fidèles
- **Export multi-formats** : PDF, CSV pour les rapports
- **Notifications** : Alertes pour les réservations imminentes
- **Gestion du personnel** : Attribution des chambres au personnel de ménage
- **Tarification dynamique** : Modulation des prix selon la saison

Techniques

- **Base de données** : Migration vers SQLite pour une meilleure scalabilité
- **Cryptage** : Hashage des mots de passe (bcrypt)
- **Logs d'audit** : Traçabilité complète des opérations
- **Tests unitaires** : Couverture avec framework de test C
- **Documentation** : Génération automatique avec Doxygen

Conclusion Finale

Ce projet concrétise notre progression en programmation système et démontre qu'une application C bien architecturée peut rivaliser en ergonomie et en fonctionnalités avec des solutions développées dans des langages de plus haut niveau. Il confirme également que la rigueur imposée par le langage C, loin d'être une contrainte, constitue une école d'excellence pour tout développeur souhaitant comprendre en profondeur le fonctionnement des systèmes informatiques.

Nous sommes fiers du résultat obtenu et convaincus que les compétences acquises lors de ce projet constitueront un socle solide pour nos futurs développements, qu'ils soient en C ou dans d'autres langages.

Yazid TAHRI ALAOUI, Dounia LAMDAKKI, Kaoutar BENOUAHI
ENSAH - ID1 - 2024/2025