Національний університет «Полтавська політехніка імені Юрія Кондратюка»

(повне найменування вищого навчального закладу)

Навчально-науковий інститут інформаційних технологій та робототехніки

(повна назва інституту)

Кафедра комп‘ютерних та інформаційних технологій і систем

(повна назва кафедри)

Пояснювальна записка до дипломного проекту (роботи)

бакалавра

(ступеню вищої освіти)

на тему

Розроблення інтерактивної інформаційної системи

готельного бізнесу

Виконав: студент 4 курсу, групи 401-ТН

спеціальності

\_\_122 Комп’ютерні науки \_

(шифр і назва напряму)

Порох Д.С.

(прізвище та ініціали)

Керівник Гафіяк А. М.

(прізвище та ініціали)

Рецензент

(прізвище та ініціали)

Полтава – 2021 року

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ**

**«ПОЛТАВСЬКА ПОЛІТЕХНІКА ІМЕНІ ЮРІЯ КОНДРАТЮКА»**

**НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА РОБОТОТЕХНІКИ**

**КАФЕДРА КОМП’ЮТЕРНИХ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ І СИСТЕМ**

**КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА**

**спеціальність 122 «Комп’ютерні науки»**

**на тему**

**«Розроблення інтерактивної інформаційної системи готельного бізнесу»**

**Студента групи 401-ТН Пороха Дмитра Сергійовича**

Керівник роботи

доктор педагогічних наук,

доцент Гафіяк А.М.

Завідувач кафедри

кандидат технічних наук,

доцент Головко Г.В.

Полтава – 2021

**РЕФЕРАТ**

Пояснювальна записка містить: 73 с., 25 рис., 20 джерел, 1 додаток.

**Об’єкт дослідження** – інтерактивна інформаційна система готельного бізнесу.

**Предмет дослідження –** інструментальні засоби розробки інформаційних систем.

**Мета кваліфікаційної роботи –** розроблення та впровадження інтерактивної інформаційної системи для управління готельним комплексом.

**Ключові слова:** інформаційна система, готель, бронь номерів, база даних, персональний комп’ютер, обліковий запис, пошук, клієнти, співробітники, авторизація.

В результаті роботи була розроблена інформаційна система забезпечення роботи готелю, яка відповідає специфіці роботи готельного бізнесу для подальшого її впровадження під мережу таких готелів.

**ЗМІСТ**

[ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ 6](#_Toc74325427)

[ВСТУП 7](#_Toc74325428)

[РОЗДІЛ 1 СПЕЦИФІКА ГОТЕЛЬНОГО БІЗНЕСУ, АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАДАЧІ 8](#_Toc74325429)

[1.1 Опис предметної області 8](#_Toc74325430)

[1.2 Аналіз існуючих інформаційних систем для управління готельним бізнесом 9](#_Toc74325431)

[1.3 Постановка задачі 10](#_Toc74325432)

[РОЗДІЛ 2 ПРОЕКТУВАННЯ ІНТЕРАКТИВНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ УПРАВЛІННЯ ГОТЕЛЬНИМ БІЗНЕСОМ 11](#_Toc74325433)

[2.1 Моделювання функціоналу та структури за допомогою UML-діаграм 11](#_Toc74325434)

[2.2 Створення діаграм діяльності 14](#_Toc74325435)

[РОЗДІЛ 3 РОЗРОБКА ІНТЕРАКТИВНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ ГОТЕЛЬНОГО БІЗНЕСУ 16](#_Toc74325436)

[3.1 Вибір мови програмування 16](#_Toc74325437)

[3.2 Швидкість розробки 16](#_Toc74325438)

[3.3 Продуктивність коду і вимогливість до ресурсів 17](#_Toc74325439)

[3.4 Зручність налаштування 17](#_Toc74325440)

[3.5 Мова та синтаксис 17](#_Toc74325441)

[3.6 Зручність збірки 18](#_Toc74325442)

[3.7 Перспективи 18](#_Toc74325443)

[3.8 Підсумок 19](#_Toc74325444)

[3.9 Вибір середовища розробки для бази даних 21](#_Toc74325445)

[3.10 Розробка Бази даних 23](#_Toc74325446)

[3.11 Програмна реалізація ІС 29](#_Toc74325447)

[РОЗДІЛ 4 ТЕСТУВАННЯ 40](#_Toc74325448)

[4.1 Вибір виду тестування 40](#_Toc74325449)

[4.2 Тест план 40](#_Toc74325450)

[ВИСНОВКИ 50](#_Toc74325451)

[СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 51](#_Toc74325452)

[ДОДАТОК А 53](#_Toc74325453)

# ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

**БД** – база даних.

**ПК** – персональний комп’ютер.

**СКБД** – система керування базами даних.

**ІС** – інформаційна система.

**C #** – це гібрид C та C ++, це мова програмування Microsoft, розроблена, щоб конкурувати з мовою Java Sun. C # – це об'єктно-орієнтована мова програмування, що використовується з веб-службами на основі XML на платформі .NET і призначена для підвищення продуктивності при розробці веб-додатків.

**WPF –** це платформа для користувача інтерфейсу для створення клієнтських додатків. Простими словами - це платформа, що дозволить вам створити будь-яку програму для вашого комп'ютера. До програми ви зможете додати інтерфейс, прописати функції, додати роботу з базою даних і багато іншого.

**XAML** – це нова описова мова програмування, розроблена Microsoft для написання користувацьких інтерфейсів для керованих додатків наступного покоління. XAML – це мова для побудови користувальницьких інтерфейсів для Windows та мобільних додатків, які використовують Windows Presentation Foundation (WPF), UWP та Xamarin Forms.

**ВСТУП**

У наш час автоматизоване управління організацією є обов’язковою частиною сучасних принципів керівницької діяльності. В сучасному світі інформаційних технологій важко знайти організацію, яка хоча б частково не була б автоматизована. Зберігання документів , клієнтської бази, а також інформації необхідної для безперебійної роботи компанії вимагає суттєвих витрат, як тимчасових так і грошових.

Мета дипломної роботи – реалізація інформаційної системи, що дозволить керувати готельним бізнесом з мінімальними витратами коштів.

Для досягнення мети сформульовані наступні завдання:

* аналіз проблеми готельного бізнесу та постановка задачі;
* моделювання функціоналу за допомогою UML діаграм;
* вибір мови програмування;
* вибір середовища розробки бази даних;
* розробка БД;
* розробка програмного забезпечення;
* тестування і налагодження програмного забезпечення;
* аналіз підсумків впровадження інформаційної системи.

**РОЗДІЛ 1 СПЕЦИФІКА ГОТЕЛЬНОГО БІЗНЕСУ, АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАДАЧІ**

* 1. **Опис предметної області**

Предметною областю даної кваліфікаційної роботи є розробка інформаційної системи для управління готельним бізнесом. Інформаційна система, яка використовується в готельній справі має забезпечувати:

* достовірність і повноту інформації, яка міститься в ній;
* швидкість роботи, яка відповідає критеріям обслуговування;
* зв’язки між різними категоріями працівників організації;
* автоматизацію інформації, що передається дочірнім службам.

Для підтримки достовірності і повноти даних в інформаційній системі, треба контролювати правильність і коректність інформації, що вводяться користувачем, а також своєчасне їх корегування.

Швидкість роботи залежить як від швидкості виконання запитів до бази даних, так і від навичок роботи користувача ІС. Для прискорення навчання користувача необхідно щоб інформаційна система мала зрозумілий інтерфейс, а так само коректно складений посібник користувача. Залежно від займаної посади програмне забезпечення буде видавати доступ до відповідних функцій, які необхідні для виконання своїх зобов'язань, це так само допоможе в швидкості роботи користувача інформаційної системи.

Організація зв’язків між різними категоріями працівників потрібна для доручення їм певних завдань і контролем їх виконання, для даної ІС пропонується розробити модуль, який автоматизує передачу завдань безпосередньо з місця користувача інформаційної системи.

1. **Аналіз існуючих інформаційних систем для управління готельним бізнесом**

В якості ІС, які працюють в області готельного бізнесу, були розглянуті наступні рішення.

Amadeus Property Management System – це потужне програмне рішення автоматизації всіх підрозділів готелю, яка поєднує в собі кілька модулів – управління готелем, маркетинг і продажі, конференції та банкети [4].

Amadeus PMS – гнучке рішення щодо розподілу, інформаційних технологій та торгових точок, щоб допомогти своїм клієнтам адаптуватися, рости та досягти успіху в туристичній галузі, що швидко змінюється. Рішення згруповані у чотири категорії рішень – розподіл та вміст, продажі та електронна комерція, управління бізнесом та послуги та консалтинг [5].

ServioHotel – функціональний інструмент для автоматизації підприємств у сфері гостинності. Багатофункціональний інструмент розроблений для підвищення ефективності обслуговування і управління, як в стандартних готелях, так і в великих мережах. Гнучкість налаштувань дозволяє використовувати продукт в закладах гостинності будь-якого масштабу і складності [6].

«Сінімекс Готель» – це програмний комплекс для готелів, головна функція якого – автоматизація оперативного управління номерним фондом і облік взаєморозрахунків з гостями і контрагентами. Залежно від індивідуальних вимог готелів система «Сінімекс: Готель» має різні варіанти настройки. В цілому система «Сінімекс: Готель» має просте і зрозуміле керівництво користувача, яке дозволяє встановити, налаштувати і експлуатувати конфігурацію самостійно [7].

* 1. **Постановка задачі**

Завданням даної кваліфікаційної роботи є розробка інформаційної системи для управління готельним бізнесом, а саме, його реалізація на практиці за допомогою C#, а також тестування цієї ІС і впровадження її в експлуатацію.

Інформаційна система має виконувати наступні функції:

1. аутентифікація користувачів та надання їм прав згідно їх посад;
2. створення броні та її видалення;
3. створення архіву;
4. додавання нової інформації про клієнтів, співробітників та номери;
5. додавання облікових записів;
6. пошук по параметрам (id, прізвище, архів);
7. редагування та видалення записів про клієнтів, співробітників та номери.

**РОЗДІЛ 2 ПРОЕКТУВАННЯ ІНТЕРАКТИВНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ УПРАВЛІННЯ ГОТЕЛЬНИМ БІЗНЕСОМ**

* 1. **Моделювання функціоналу та структури за допомогою UML-діаграм**

Use Case – це техніка, яка використовується при розробці програмного забезпечення чи інформаційної системи для охоплення функціональних вимог даної системи, Use Case описує взаємодії, що відбуваються між "акторами" –ініціатором взаємодії самої системи з існуючими системами, варіант використання представлений простою послідовністю кроків.

Сам USE CASE описує систему, системне середовище та взаємозв’язок між системою та її середовищем. Опис послідовного набору дій, виконуваних системою, що призводить до появи значення для конкретного актора.

Use Case використовується для впорядкування поведінкових речей у моделі. Варіанти використання реалізуються у співпраці. Графічний варіант діаграми прецедентів, а також таблиці акторів представлені нижче [8].

Таблиця 2.1 – опис акторів інформаційної системи

|  |  |
| --- | --- |
| **Актор** | **Короткий опис** |
| Агент з бронювання номерів | Користувач, відповідає за всі аспекти бронювання номерів у готелі, та занесення інформації про нові номери |
| Менеджер | Користувач, який відповідає за реєстрацію співробітників в системі, створення облікових записів |
| Системний Адміністратор | Користувач, який має права на редагування та видалення інформації про клієнтів, співробітників та номери |

Функції акторів системи:

* актор «Системний Адміністратор» використовує систему для редагування записів, а також оновлення бази даних інформаційної системи;
* актор «Менеджер» використовує систему для додавання нових працівників та реєстрацію облікових записів;
* актор «Агент з бронювання номерів» використовує систему для реєстрацію нових клієнтів, додавання і видалення бронювання клієнта, занесення інформації про нові номери.

Сценарії та хід їх виконання представленні в таблиці 2.2.

Таблиця 2.2 – опис сценаріїв та їх використання

|  |  |
| --- | --- |
| **Сценарій** | **Короткий опис** |
| 1 | 2 |
| Авторизація | Виконується системним адміністратором, менеджером та агентом з бронювання номерів, дозволяє війти в систему та надає права відповідно до посади. |
| Бронювання номерів та її видалення | Виконується агентом з бронювання номерів дозволяє забронювати номер для клієнта, або видалити бронь якщо вона вже створена. |
| Додати клієнта | Виконується агентом з бронювання номерів, дозволяє додати клієнта в базу даних |
| Додати співробітника | Виконується менеджером та системним адміністратором, дозволяє додати співробітника в базу даних |
| Додати номер | Виконується агентом з бронювання номерів, дозволяє додати запис про новий номер в базу даних |

Продовження таблиці 2.2

|  |  |
| --- | --- |
| 1 | 2 |
| Додати новий обліковий запис | Виконується менеджером та системним адміністратором, дозволяє додавати обліковий запис співробітникам |
| Редагування даних | Виконується системним адміністратором, дозволяє редагувати дані клієнтів, співробітників та видаляти номери |
| Пошук | Виконується системним адміністратором, дозволяє здійснити пошук по параметрам, переглянути архів |

Діаграма прецедентів для інформаційної системи управління готельним бізнесом зображено на рисунку 2.1.

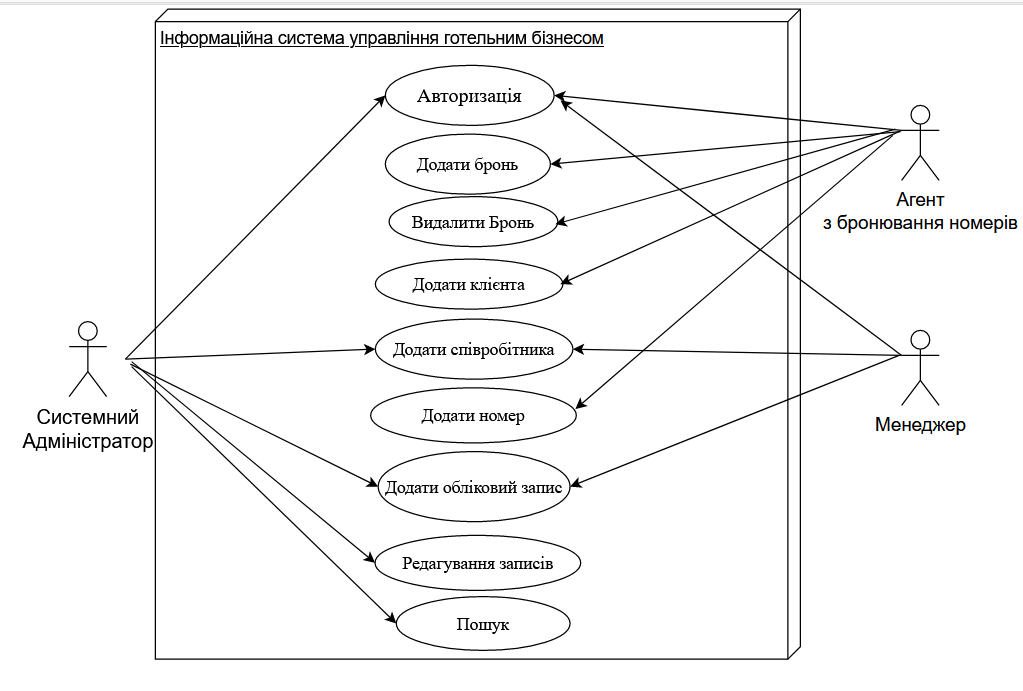


Рисунок 2.1 – Діаграма прецедентів ІС «управління готельним бізнесом»

* 1. **Створення діаграм діяльності**

Діаграми діяльності в основному описують різні потоки діяльності, які будуть розроблені в системі. Де кожен потік має початок, рішення, яке може відбутися в системі, і кінець у системі. Діаграми діяльності в основному мають структуру, яка майже схожа на блок-схему або блок-схему в структурованій системі. Ця діаграма діяльності складається на основі варіанту використання або декількох випадків використання у схемі випадків використання [8].



Рисунок 2.3 – Діаграма діяльності для форми авторизації користувачів інформаційної системи

Діаграма діяльності зображена вище призначена для зіставлення ідентифікаційних даних і ключа аутентифікації з подальшою авторизацією.

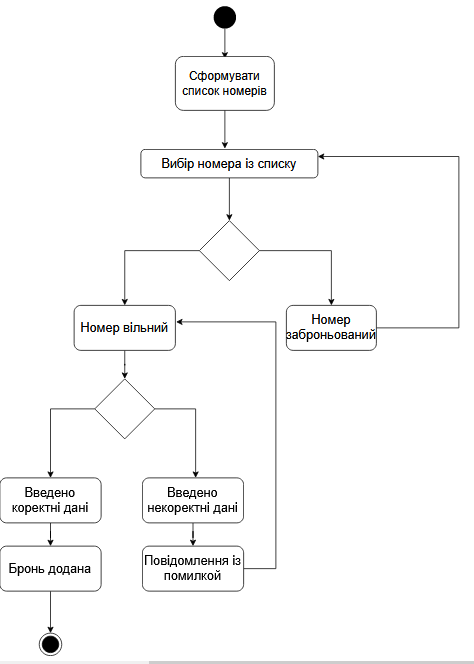


Рисунок 2.2 – Діаграма діяльності для форми бронювання клієнтів інформаційної системи

Дана діаграма призначена для створення броні для зареєстрованих клієнтів.

**РОЗДІЛ 3 РОЗРОБКА ІНТЕРАКТИВНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ ГОТЕЛЬНОГО БІЗНЕСУ**

1. **Вибір мови програмування**

На початковому етапі розробки програми постало питання, вибору мови програмування: C ++ або C #?

Щоб дати відповіді на питання, які виникають в ході розробки, необхідно порівняти їх за наступними параметрами:

* швидкість розробки;
* бібліотеки;
* продуктивність коді і вимогливість;
* зручність налаштування;
* мова та синтаксис;
* вартість підтримки;
* ризики;
* самодостатність додатків;
* зручність збірки;
* висновок.

1. **Швидкість розробки**

C# є швидшим на початковому етапі розробки і дає змогу швидше отримати вихідне рішення. Можна сказати що C# на початку розробки значною мірою перевищує швидкість розробки С++.

Але, C # містить багато накладних витрат і додає бібліотеки перед компіляцією, тому швидкість розробки C++ та C# стають майже однаковими.

Отже, в нетривалих проектах швидкість розробки C# є перевагою, але в тривалих проектах перевага є незначною [9].

1. **Продуктивність коду і вимогливість до ресурсів**

Відповідь, звичайно, залежить від того, що потрібно зробити. Але загалом, C ++ буде швидшим за C # через його нижчий рівень. Однак для отримання такої високої продуктивності потрібно скористатися її функціями низького рівня та оптимізувати її на мікрорівні.

Хоча це змусило б програму працювати швидше, фактичне кодування займе набагато більше часу, ніж для мови високого рівня. Мови високого рівня, такі як C #, призначені для скорочення часу кодування, абстрагуючись від багатьох процесів. Отже, якщо потрібно швидше обробляти числа в режимі реального часу, C ++ – це шлях. Якщо потрібно швидко щось зібрати, краще C # [10].

1. **Зручність налаштування**

Під Windows, зручніше налагоджувати C #, на відміну від C ++. У C #, як і в інших які з'явилися до .NET мовами, головна методика з налагодження полягає в додаванні точок зупину і вивченні того, що відбувається в коді в конкретні моменти під час його виконання [12].

1. **Мова та синтаксис**

C # – це мова, заснована на С, тому вона робить два синтаксиси схожими. Розробник використовує дужки для сегментування структур кодування, а об'єктно-орієнтований код у стилі С, що включає залежності та бібліотеки, дуже схожі. Якщо розробник знайомий з Java або C ++, дуже легко перейти до C #. Однак перехід від C # до C ++, мабуть, складніший для розробника C #, оскільки це значно нижча мова. C # обробляє більшу частину накладних витрат, які необхідно враховувати в програмі на C ++. Це лише одна з причин, через яку C ++ вважається більш складною для вивчення мовою у світі розвитку.

**Основну різницю в синтаксичних правилах C # та C ++:**

* C # не має глобальних функцій, рішенням цього є створення статичних класів;
* C ++ має файли заголовків, тоді як у C # їх немає;
* замість #include headline на початку коду C ++, C # застосовується за допомогою операторів (наприклад, за допомогою System);
* C # підтримує єдине успадкування, тоді як C ++ підтримує декілька;
* покажчики не застосовуються в C #, натомість C # використовує посилання в небезпечному коді;
* C ++ не підтримує кожен цикл; C # підтримує [11].

1. **Зручність збірки**

Збірка С ++ проектів помітно складніше збірки проектів С #. Однак варто розуміти, що велика складність надає і додаткову гнучкість, яка рано чи пізно може стати корисною. Однак до цього моменту буде лише збільшувати витрати часу [12].

1. **Підсумок**

C # .NET – одна з мов програмування Microsoft. Це найпотужніша мова програмування серед усіх мов програмування, доступних у .NET Framework. C # .NET містить усі можливості C ++, VB.NET, JAVA, а також деякі додаткові функції.

Мова програмування C # .NET розроблена як проста, сучасна мова загального призначення та повністю об'єктно-орієнтована мова програмування.

Зважаючи на те, що C # був розроблений, щоб конкурувати з Java, він переглядає більше схожих на мову Java, але все ще має подібність із C ++, Створені для аналогічних цілей, але серед конкурентів на ринку C # та Java мають багато спільного між собою. Обидві мови мають свої функціональні можливості, оточені спільною сутністю – об'єктом. Отже, обидві вони є об'єктно-орієнтованими мовами і широко використовують подібні функції, такі як класи, поліморфізм та успадкування. C# та Java підтримують модульний підхід, який дозволяє повторно використовувати модулі коду одного проекту в іншому. Повторне використання коду було привабливою особливістю цього модульного підходу, і це головна причина, чому програмістам було зручно працювати з ними. C #, і Java підтримують єдине успадкування, де існує лише один шлях від будь-якого базового класу до будь-якого його похідного класу. Однак, Java відсутні властивості та індексатори (а також делегати та події, але вони відсутні ще багато де). Також є можливість перерахунку контейнерів [13].

Розробляти інформаційну систему планується під платформою Windows, відповідно кроссплатформенність не важлива. Необхідно створити програму, яка буде мати зручний для використання інтерфейс, засобами C # цього набагато легше та швидше досягти. Синтаксис C # на багато зрозуміліший людині, що не мала відношення до розробки програми, отже, легше знайти людину на доопрацювання і підтримку програмного забезпечення. Так само бібліотеки на C #, зважаючи на більш адаптовані для роботи з базами даних, відповідно, легше та швидше буде зв’язати базу даних та розроблений додаток.

Підсумовуючи вище сказане вважаю вибір мови програмування C# та середовища розробки Visual studio 2019.

**WPF – W**indows **P**resentation **F**oundation. Це система, яка піклується про графічний інтерфейс користувача. Він використовує DirectX для рендерингу замість GDI (графічний інтерфейс пристрою). WPF має архітектуру, яка складається з трьох основних компонентів: фреймворк презентацій, ядро презентації та mallcore. WPF не повністю покладається на стандартні елементи керування Windows, і тому є самостійним підходом. У 2007 році Microsoft представила Windows Presentation Foundation (WPF), щоб перемогти програму WinForms for .Net Framework для настільних ПК. Ця послідовність призвела до багатьох змін у розробці настільних додатків. Для початку WPF відокремлює дизайнера та програміста, де інтерфейс користувача може бути розроблений окремо за допомогою Visual Studio або Blend, тоді як розробник може використовувати файл із кодом для обробки подій управління.

WPF використовує XAML для створення елементів керування, а його файлова структура більше нагадує ASP.NET, де у вас є свобода використовувати конструктор або написати код XAML для створення елементів керування. Використовуючи Canvas Panel, дизайнери все ще мають можливість перетягувати елементи керування на сторінці Windows, як у WinForms. Основна різниця, яку приносить WPF – це файл XAML та доступ до видимого конструктора, який поставляється поряд із файлом XAML [14].

1. **Вибір середовища розробки для бази даних**

Для спільної взаємодії користувачів та бази даних використовують систему керування базами даних (СКБД).

СКБД розшифровується як системи управління базами даних, і вона спеціально розроблена для зберігання, управління, визначення та отримання даних у базі даних. Він в першу чергу виступає як інтерфейс між базою даних та кінцевим користувачем. У той же час програмне забезпечення здатне керувати даними, механізмом бази даних та схемою бази даних, що полегшує організацію та маніпулювання даними, що зберігаються в базі даних.

Типова функція СКБД включатиме:

* система управління бібліотекою СКБД;
* доступний для користувача каталог із метаданими;
* абстракція даних та незалежність;
* підтримка відновлення даних;
* протоколювання та аудит діяльності;
* підтримка доступу до авторизації;
* впровадження обмежень.

Функції СКБД можуть сильно відрізнятися. СКБД використовує системні команди для виконання цих функцій. Спочатку він отримує вказівки від адміністратора бази даних, а потім вказівки надсилає системі для отримання, модифікації або завантаження даних. Для підвищення ясності в організації даних, реалізована техніка проектування схеми даних, яка називається нормалізацією. Це дозволяє модифікувати існуючу схему, щоб максимально зменшити надмірність та залежність даних. Це досягається шляхом розділення таблиці на менші та встановлення взаємозв'язку між ними [15].

Додатки, що створюються за допомогою засобів СКБД мусять мати досить високий рівень мобільності та легко переноситися на різні комп'ютерні і мережеві платформи. Вимоги до СКБД перераховані вище, дають змогу поглянути на існуючі нині на ринку продукти від різних виробників під відповідним кутом . Засобами СКБД повинна забезпечувати здатність гнучкої модифікації архітектури різних ІС для додержання компромісу при розділенні функціональних можливостей системи між робочою станцією клієнта і сервером [16].

Протилежністю настільної системи є розподілена база даних – система управління складається з єдиної бази даних, яка розділена на декілька фрагментів. Кожен фрагмент зберігається на одному або декількох комп'ютерах під контролем окремої СКБД, а комп'ютери з'єднані мережею зв'язку. В цілому БД MS Access представляє собою багатофункціональний продукт, який може містити в собі крім таблиць, звіти, форми, макроси і модулі. Всі ці об'єкти, як правило, зберігаються в одному файлі, що має розширення \* .mdb або \* .mde. На практиці це значно полегшує резервне копіювання даних. Така технологія називається «клієнт-сервер» [17].

Так як на початковому етапі планується впровадження ІС на поодинокий об'єкт, логічно використовувати локальну базу даних. Можливо, доведеться переносити програму на інші ПК, отже, необхідно використовувати СКБД, яка має в своєму розпорядженні всі об'єкти, якими вона оперує в одному файлі. Потрібен зручний візуальний редактор, який дасть можливість швидко виконувати SQL запити, а також створення даних SQL запитів по засобах конструктора, для занесення їх на клієнтської частини БД.

MS Access – це інструмент швидкої розробки додатків, окрім того, що він є інструментом баз даних. Також MS Access представляє себе як настільна СКБД. Поняття «настільна» СКБД указує на те, що усі операції з БД виконуються на локальному ПК користувача [17].

У MS Access передбачено всі потрібні засоби для визначення і оброблення даних, а також для управління ними при роботі з великою кількістю інформації. СКБД також значно збільшує можливість і спрощує впорядковування та ведення великої кількості інформації, що зберігається в таблицях. Саме тут зберігається інформація, а також працюють засоби управління і організації запитів. СКБД включає в собі три основні типи функцій: визначення даних, обробка даних і управління даними. Весь цей функціонал повною мірою реалізований в Microsoft Access. На практиці, зазвичай, треба вирішувати і задачі з застосуванням електронних таблиць і текстових процесорів. Наприклад, після підрахунку або аналізу даних потрібно їх представити у певному виді або шаблоні. В підсумку користувачеві доводиться використовувати декілька програмних продуктів для досягнення необхідного результату. У цьому змісті все значно полегшить можливості, надавані Microsoft Access [18].

Опираючись на вище зазначене, СКБД Microsoft Access відповідає цим вимогам. В даній кваліфікаційній роботі було вирішено використовувати Microsoft Access 2016.

1. **Розробка Бази даних**

Розробка бази даних була розпочата з загальної концепції. Для інтерактивної інформаційної системи готельного бізнесу необхідно мати в базі даних наступну інформацію:

* інформацію про клієнтів;
* інформацію про співробітників;
* інформацію про кімнати;
* інформацію про облікові записи користувачів ІС;
* інформацію про цінові категорії на види послуг;
* інформацію про коли небуть заброньовані номери (архів).

Таблиця «Клієнти» містить інформацію про клієнтів готелю та має наступні атрибути:

* код – ідентифікатор клієнта, має тип лічильник, є первинним ключем, не може бути NULL;
* ім’я – має тип varchar(255), не може бути NULL;
* прізвище – має тип varchar(255), не може бути NULL;
* по батькові – має тип varchar(255), не може бути NULL;
* дата народження – має тип date, не може бути NULL;
* стать – має тип varchar(255), не може бути NULL;
* вид документу – має тип varchar(255), не може бути NULL;
* серія – має тип integer, не може бути NULL;
* номер – має тип integer, не може бути NULL;
* коментар – має тип varchar(255), не може бути NULL.

Таблиця «Контакти\_клієнтів» містить інформацію про контакти клієнтів готелю та має наступні атрибути:

* id– ідентифікатор клієнта, має тип лічильник, не може бути NULL;
* номер телефону – має тип integer, не може бути NULL;
* адреса– має тип varchar(255), не може бути NULL.

Вміст таблиць «Клієнти» і «Контакти\_Кліента», представлено на рисунках 3.1 і 3.2.

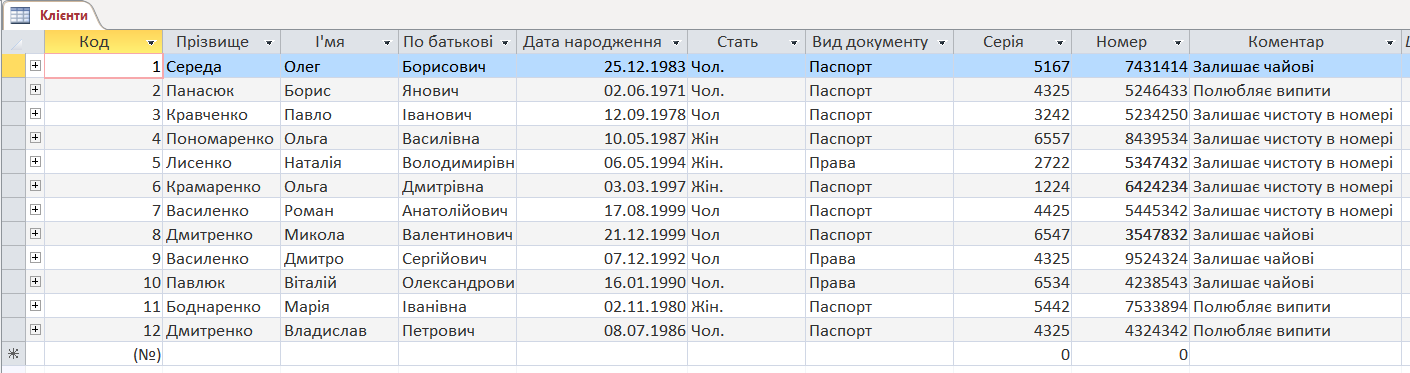


Рисунок 3.1 –Таблиця «Клієнти»

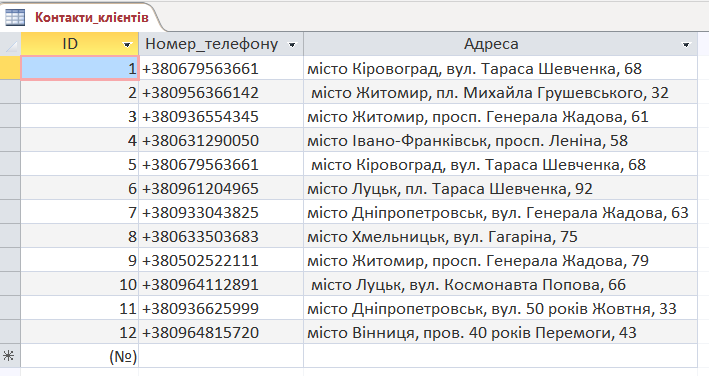


Рисунок 3.2 –Таблиця «Контакти\_клієнта»

У стовпцях даної таблиці міститься інформація, яка необхідна для ідентифікації кожного клієнта. Таблиця «Клієнти» і «Контакти\_кліента» зв'язані по ідентифікаційному коду.

Таблиця «Співробітники» містить інформацію про співробітників готелю та має наступні атрибути:

* код – ідентифікатор клієнта, має тип лічильник, є первинним ключем, не може бути NULL;
* ім’я – має тип varchar(255), не може бути NULL;
* прізвище – має тип varchar(255), не може бути NULL;
* по батькові – має тип varchar(255), не може бути NULL;
* дата народження – має тип date, не може бути NULL;
* освіта – має тип varchar(255), не може бути NULL;
* посада – має тип varchar(255), не може бути NULL;
* адреса – має тип varchar(255), не може бути NULL;
* номер – має тип integer, не може бути NULL;
* заробітна плата $ – має тип integer, не може бути NULL;
* додатковий коментар– має тип varchar(255).

Вміст таблиці «Співробітники» , представлено на рисунку 3.3.

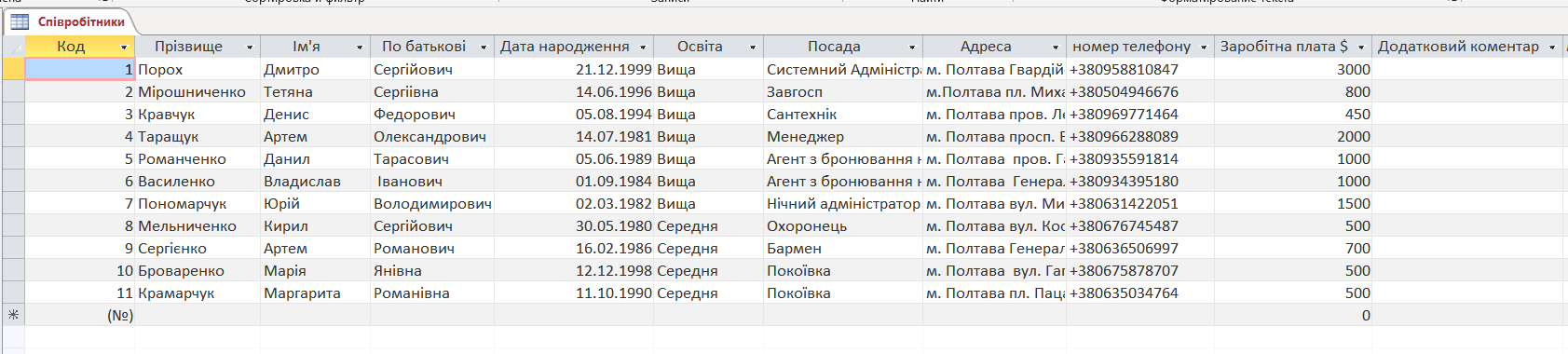


Рисунок 3.3 –Таблиця «Співробітники»

У стовпцях вище наведеної таблиці, зберігається інформація про співробітників організації. Кожен співробітник ідентифікується по персональному коду.

Таблиця «Номери» містить інформацію про кімнати готелю та має наступні атрибути:

* код – ідентифікатор кімнати, має тип лічильник, є первинним ключем, не може бути NULL;
* категорія – має тип varchar(255), не може бути NULL;
* дата заселення– має тип date, не може бути NULL;
* дата виселення– має тип date, не може бути NULL;
* статус– має тип Boolean, не може бути NULL.

Таблиця «Номери» містить в собі інформацію з номером кімнати, датою заїзду та виїзду, а також маркером, який вказує, чи зайнятий в даний момент номер. Вміст таблиці «Номери» , представлено на рисунку 3.4.

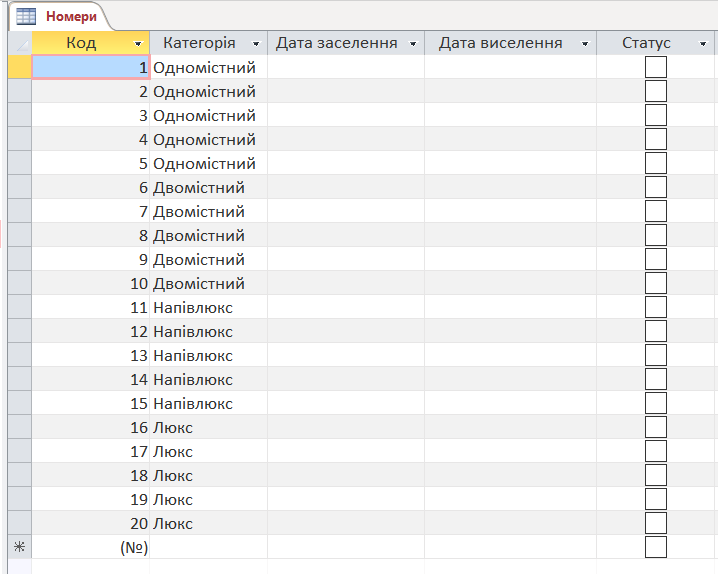


Рисунок 3.4 –Таблиця «Номери»

Інформація про облікові записи користувачів інформаційної системи, яка в подальшому буде використовуватися для аутентифікації і авторизації, міститься в таблиці «Таблиця Аутентифікації», вона, в свою чергу, містить такі атрибути як:

* код – ідентифікатор користувача, має тип лічильник, є первинним ключем, не може бути NULL;
* логін – має тип varchar(255), не може бути NULL;
* пароль – має тип varchar(255), не може бути NULL;
* ім’я – має тип varchar(255), не може бути NULL;
* посада – має тип varchar(255), не може бути NULL;
* права, що видаються при авторизації– має тип Boolean, не може бути NULL.

Таблиця «Категорія\_номерів» містить інформацію про кімнати готелю та має наступні атрибути:

* код – ідентифікатор кімнати, має тип лічильник, є первинним ключем, не може бути NULL;
* категорія – має тип varchar(255), не може бути NULL;
* кількість місць – має тип integer, не може бути NULL;
* кількість місць – має тип integer, не може бути NULL;
* ціна/доба – має тип integer, не може бути NULL.

Дані про цінові категорії на види послуг містяться в таблиць «Категорії номерів», які представлені на рисунку 3.5.

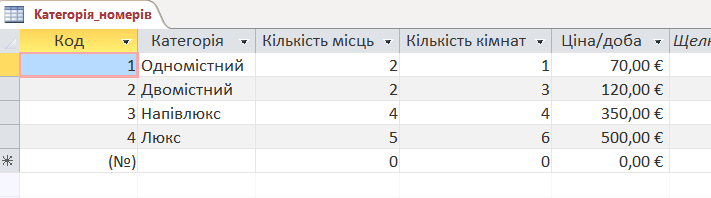


Рисунок 3.5 – Таблиця «Категорія\_номерів»

Так само в базі даних є таблиця, яка відображає архів замовлених номерів. Дана таблиця називається «Архів». У ній відображені такі атрибути:

* код – ідентифікатор кімнати, має тип лічильник, є первинним ключем, не може бути NULL;
* номер заброньованої кімнати – має тип integer, не може бути NULL;
* прізвище заброньованої людини – має тип varchar(255), не може бути NULL;
* номер телефону – має тип integer, не може бути NULL;
* дата заселення– має тип date, не може бути NULL;
* дата виселення– має тип date, не може бути NULL;
* співробітник, прийнявший завяку – має тип varchar(255), не може бути NULL.

1. **Програмна реалізація ІС**

Розробка програми виконана згідно UML діаграм. Запустивши програму користувач потрапляє на вікно авторизації, де потрібно ввести логін і пароль відповідно.

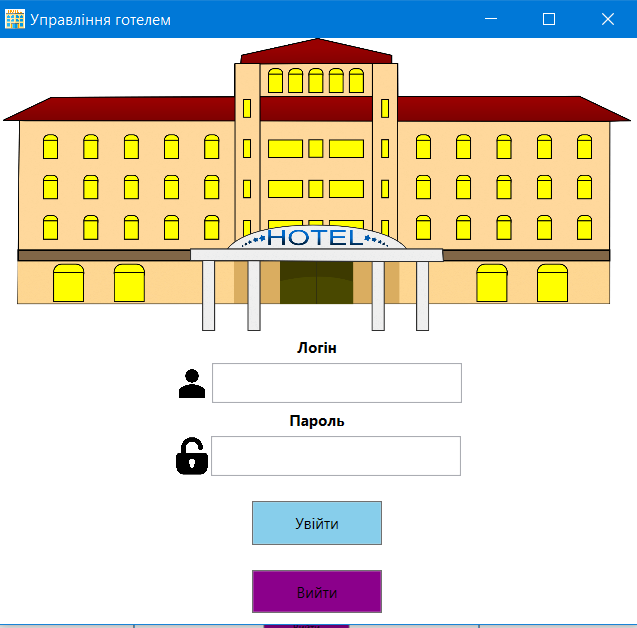


Рисунок 3.6 – Вікно авторизації

Після успішної авторизації користувач потрапляє на головне вікно програми.

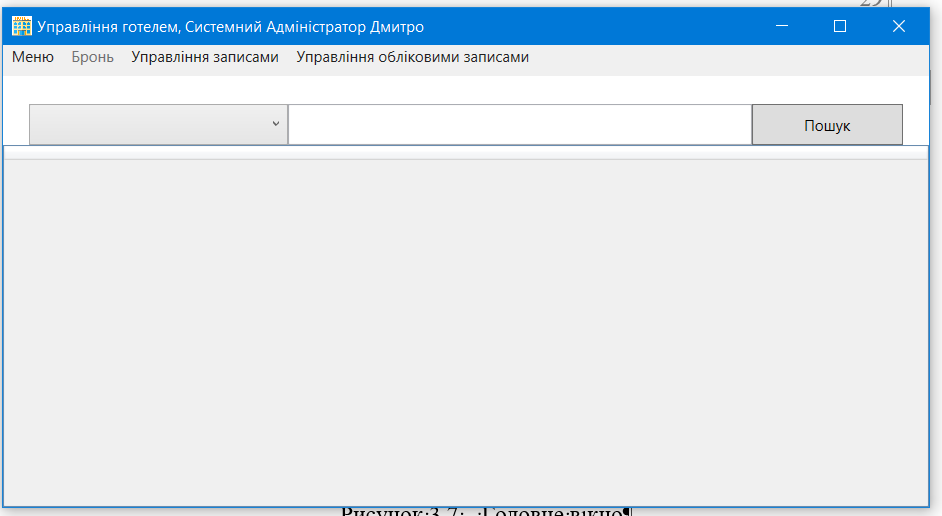


Рисунок 3.7 – Головне вікно

Залежно від прав, виданих для облікового запису користувача, у вікні, зображеному на рисунку 3.7 є наступні пункти меню:

* меню;
* бронь;
* управління записами;
* управління обліковими записами.

Підменю, пункту меню містить наступні функції:

* перепідключитися;
* вийти.

Обравши пункт «Перепідключитися» користувач повертається до вікна авторизації зображеному на рисунку 3.6. Пункт «Вийти» закриває програму.

Елемент меню «Бронь» містить в собі наступні елементи підменю:

* створення броні;
* скасування броні.

Пункт «Створення броні» дозволяє перейти на діалогове вікно яке дає змогу забронювати номер. Приклад такого вікна зображено на рисунку 3.8.

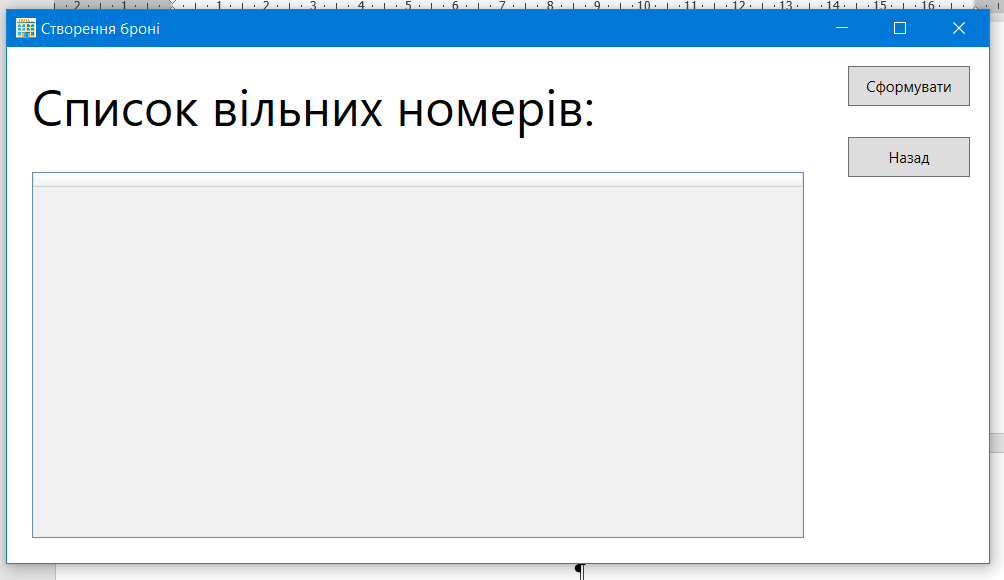


Рисунок 3.8 – Діалогове вікно створення броні

Натиснувши кнопку «Назад» користувач повертається на головне вікно на рисунку 3.7.

При натисканні кнопки «Сформувати» відправляється запит до бази даних, результатом якого є таблиця вільних на даний момент номерів. Результат представлений на рисунку 3.9.

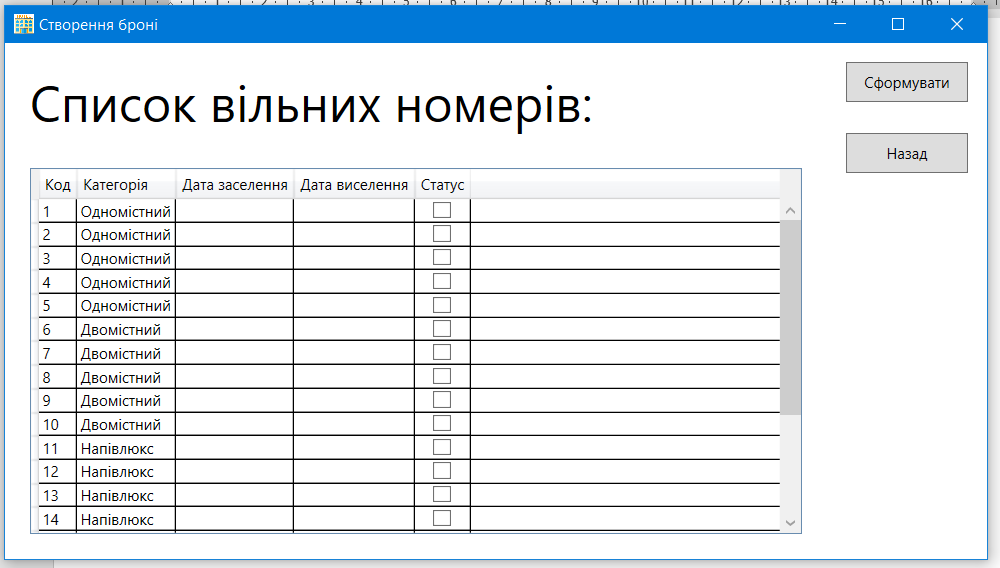


Рисунок 3.9 – Діалогове вікно створення броні

Для того, щоб забронювати номер, необхідно зробити подвійне натискання лівою кнопкою миші на рядок з потрібним номером, після цього перед користувачем випаде діалогове вікно «Активація броні», зображене на рисунку 3.10.

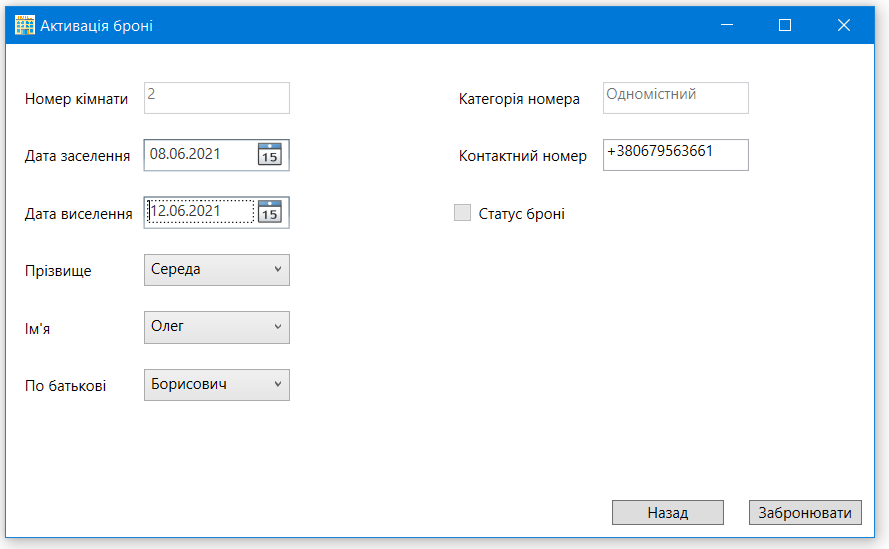


Рисунок 3.10 – Діалогове вікно активація броні

Натиснувши кнопку «Назад» користувач повертається на діалогове вікно створення броні, на рисунку 3.9.

У процесі створення броні, необхідно вказати дані, представлених на рисунку 3.10. Створити бронь можливо тільки для клієнта, зареєстрованого в базі даних. За збігом прізвища клієнта складається список імен, відповідних прізвищу, далі по прізвищу та імені, складається список можливих по батькові. Таким чином, введені дані в будь-якому випадку відповідають одному з відвідувачів. Автоматично заповнюється поле «Контактний телефон», відповідно до П.І.Б. клієнта. Після натискання кнопки «Забронювати», в базі даних, таблиці «Номери», змінюється статус броні номера, з'являється так дата заселення і виселення, а так же заноситься запис в таблицю архів.

В архів заносяться дані про броні і про користувача , який зробив бронь. При виборі пункту «Видалення броні», користувачеві видається наступне діалогове вікно, з яким можна ознайомитися на рисунку 3.11.

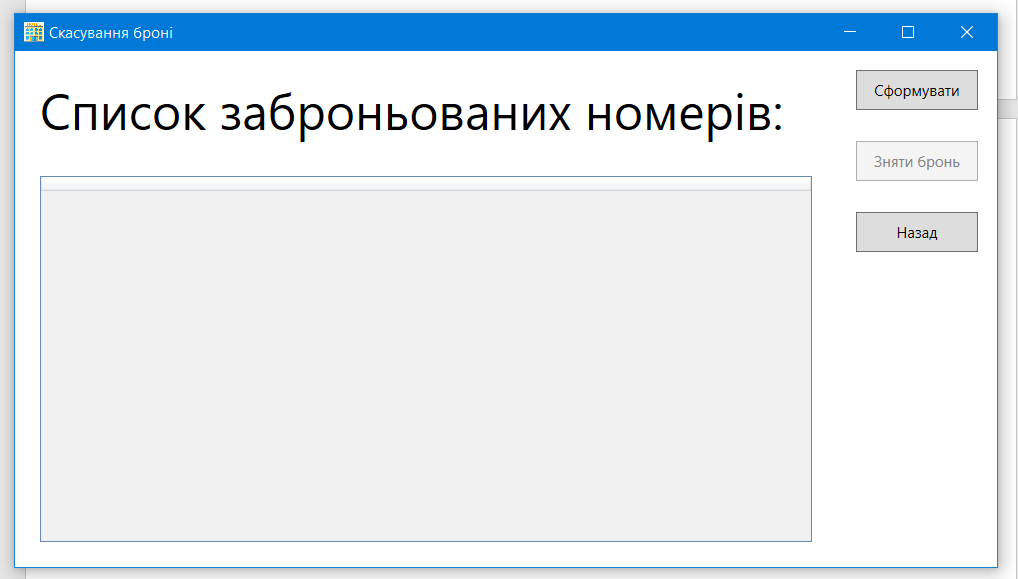


Рисунок 3.11 – Діалогове вікно скасування броні

При натисканні кнопки «сформувати», виводиться таблиця заброньованих в даний час номерів. Після виділення конкретної зайнятої кімнати і подальшого натискання кнопки «Зняти бронь», номер звільняється, змінюючи свій статус у базі даних.

Пункт «Управління записами» дозволяє:

* додати клієнта;
* додати cпівробітника;
* додати кімнату.

При виборі кожного з пунктів відкривається відповідне діалогове вікно, яке надає поля для введення необхідної інформації з метою створення обраного об'єкта.

Діалогове вікно для додавання клієнта представлено на рисунку 3.12, для додавання співробітника – на рисунку 3.13 і для додавання кімнати – на рисунку 3.14.

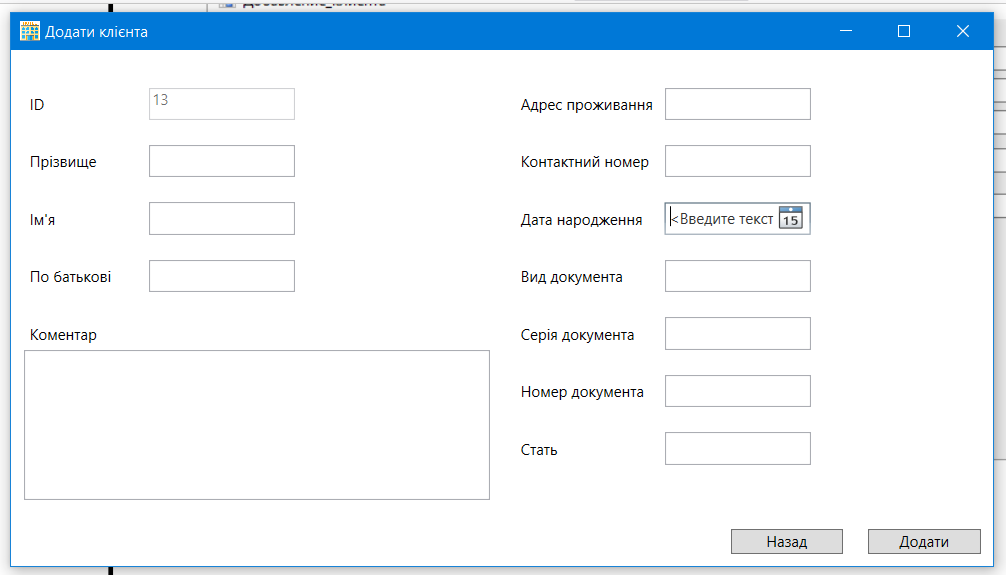


Рисунок 3.12 – Діалогове вікно додавання клієнта

Дані, які вводяться в дану форму, додаються в базу даних в таблиці «Клієнти» і таблицю «Контакти\_кліента».

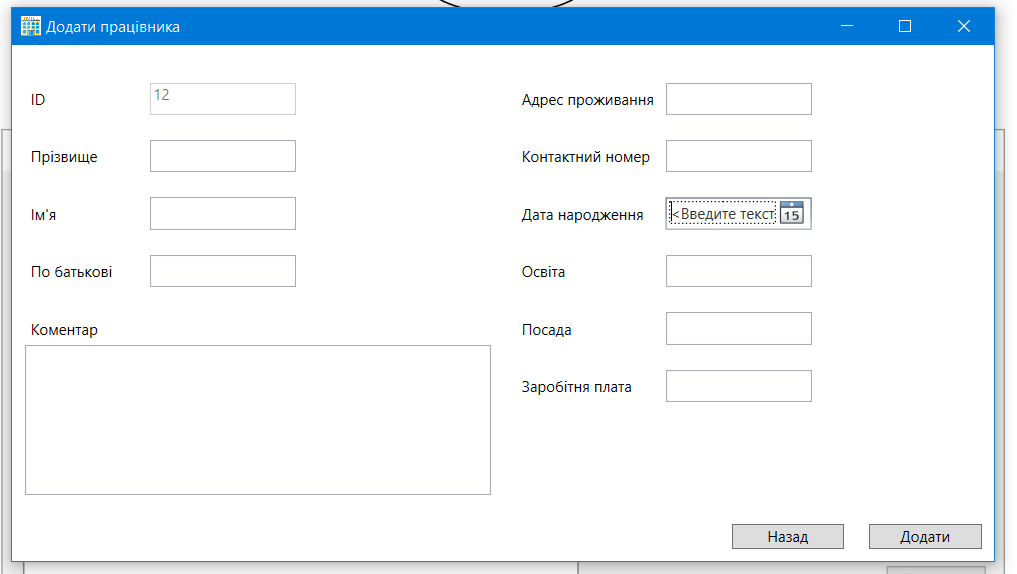


Рисунок 3.13 – Діалогове вікно додавання співробітника

Інформація з даної форми заноситься в таблицю «Співробітники» бази даних після натискання кнопки «Додати».

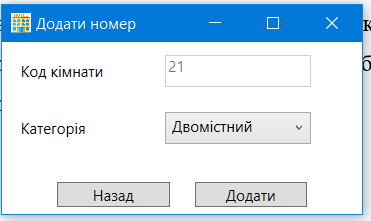


Рисунок 3.14 – Діалогове вікно додавання номера

Відповідно до вибору пункту «Додати номер» можна створити записи додаткових приміщень, які з'явилися в готелі. При натисканні кнопки додати, формується запис в БД.

Якщо користувач має права на додавання облікових записів і вибирає пункт «Управління облікових записів», йому надається діалогове вікно, яке дозволяє додати користувача інформаційної системи. Дане діалогове вікно представлено на рисунку 3.15.

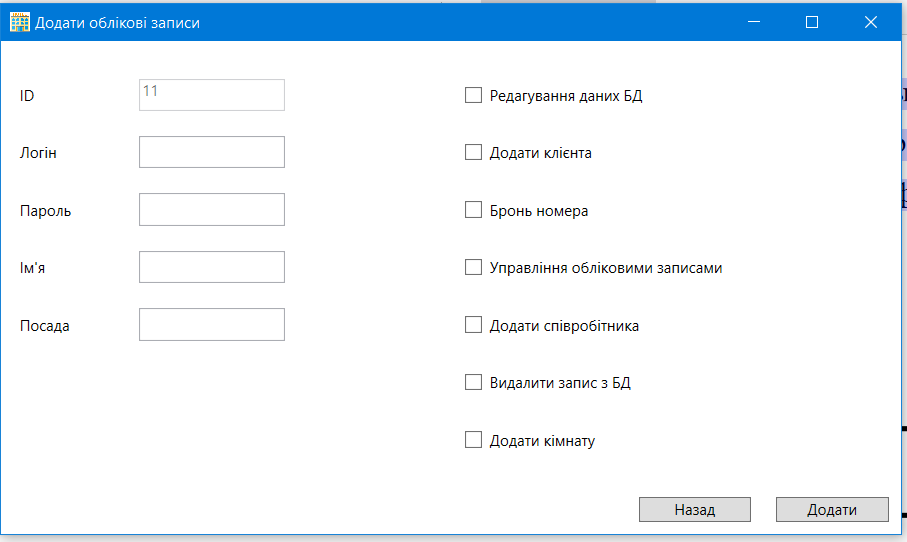


Рисунок 3.15 – Діалогове вікно додавання облікових записів

Після заповнення форми, видачі прав і натискання кнопки «Додати», створюється новий користувач інформаційної системи.

Також на головному вікні програми є пошукова система по базі даних з урахуванням певних параметрів, дана система представлена на рисунку 3.16.

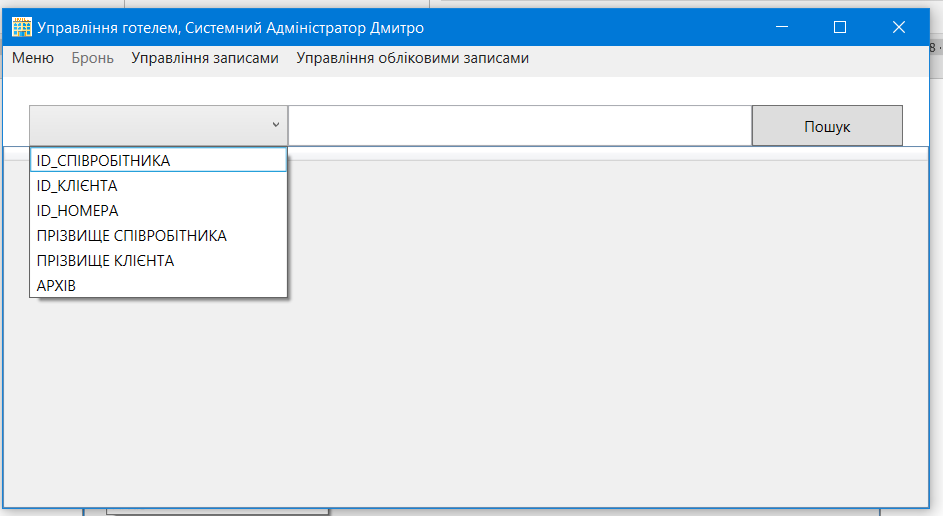


Рисунок 3.16 – Пошукова система по параметрам

Після натискання кнопки пошук вікна, показаного на рисунку 3.17 відбувається запит до бази даних.

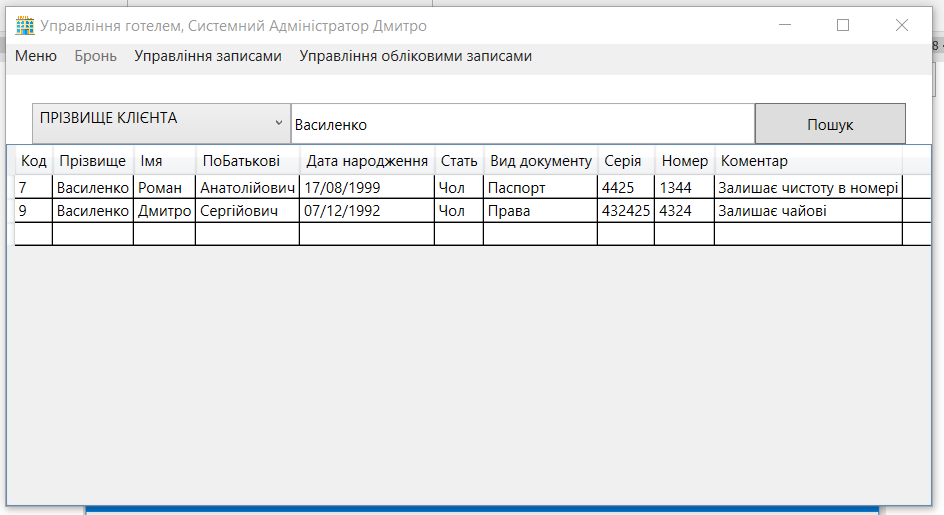


Рисунок 3.17 – Приклад роботи пошукової системи

При натисканні будь-якого рядка таблиці, відкривається вікно з докладною інформацією про той чи інший клієнта. У ньому ж є можливість її зміни або видалення запису зовсім. Приклад такого вікна можна бачити на рисунку 3.18.

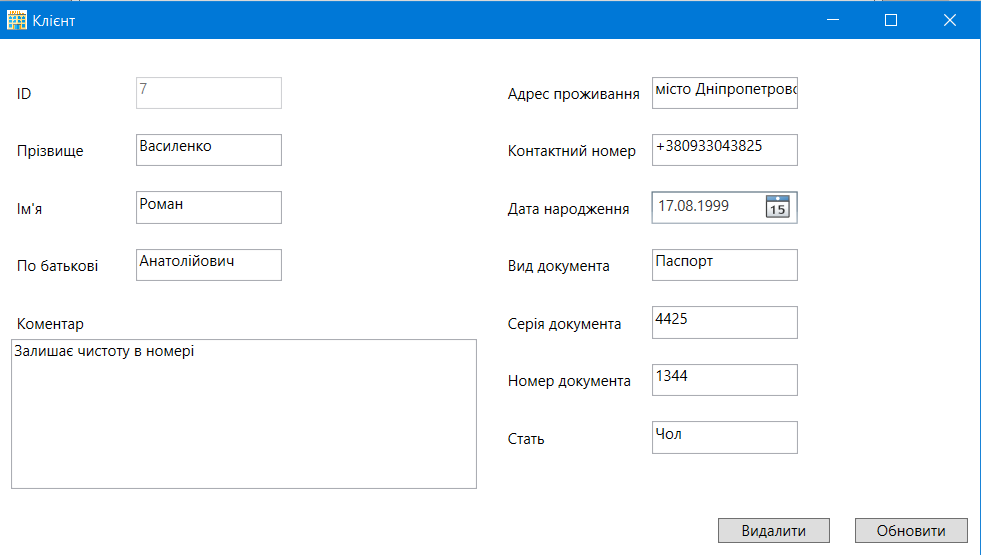


Рисунок 3.18 – Приклад роботи пошукової системи

Також у вікні перегляду детальної інформації по конкретному об'єкту, представленому на рисунку 3.18, є можливість його зміни або видалення запису про нього з бази даних. За це відповідають кнопки «Застосувати» і «Видалити».

Щоб вивести інформацію про коли небуть заброньовані номери, користувач повинен обрати параметр «Архів» в головному вікні та натиснути кнопку «Пошук». Результат на рисунку 3.19.

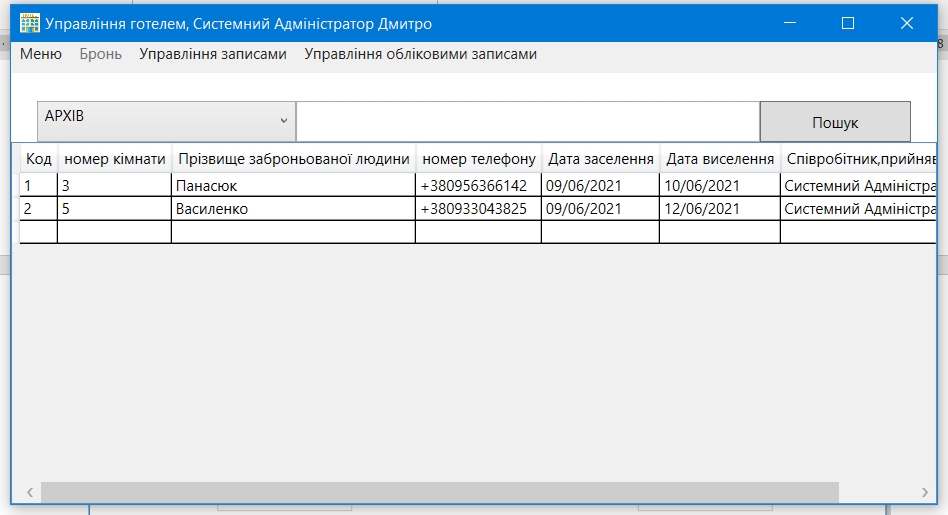


Рисунок 3.19 – Пошук по архіву

Після виведення архіву можна обрати будь який запис і подвійним натисканням перейти до його детальнішого перегляду. Приклад на рисунку 3.20.

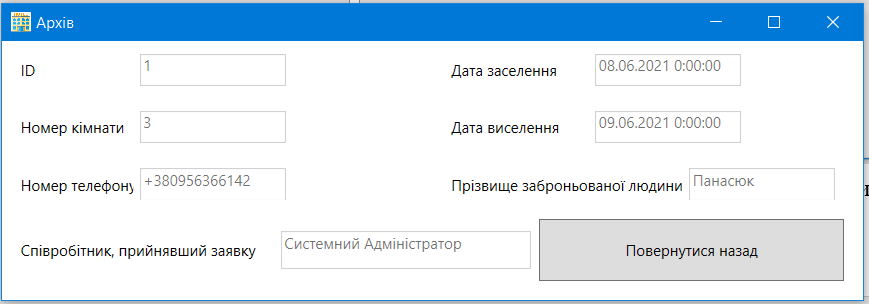


Рисунок 3.20 – Приклад роботи з архівом

З даного запису користувач може переглянути інформацію про: номер кімнати яку знімав клієнт, його прізвище, номер телефону, дату його перебування в готелі, а також посаду співробітника який забронював для нього цей номер.

Для видалення кімнати з БД готелю, треба обрати відповідний параметр «id номера» ввести номер кімнати в поле та натиснути кнопку «Пошук». Отриманий результат на рисунку 3.21.

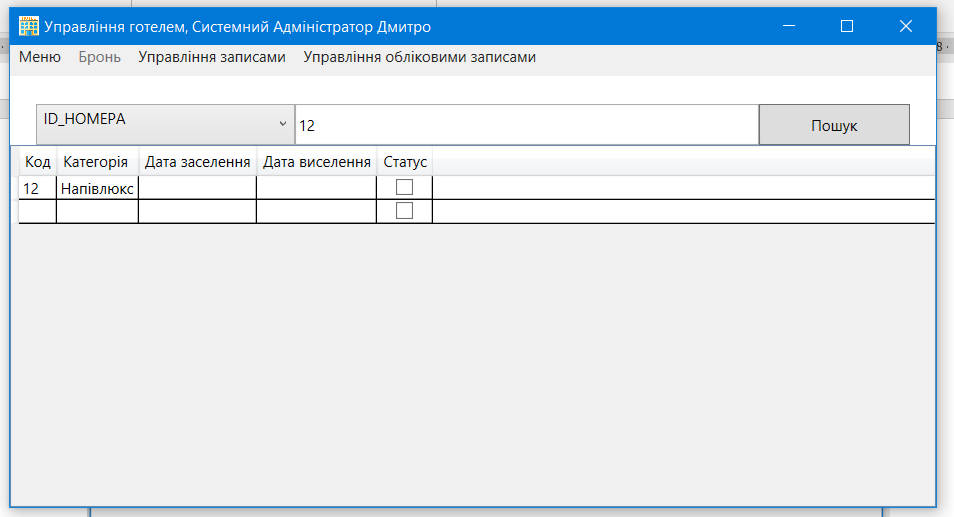


Рисунок 3.21 – Пошук кімнати

Знайшовши потрібний запис і перейшовши на нього подвійним натисканням, відкривається діалогове вікно з інформацією про кімнату. Рисунок 3.22.

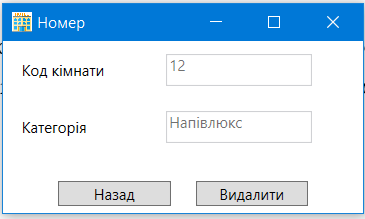


Рисунок 3.22 – Діалогове вікно видалення кімнати

Кнопка «Видалити» видаляє запис про кімнату з бази даних, а кнопка «Назад» повертає користувача на головне вікно.

.

**РОЗДІЛ 4 ТЕСТУВАННЯ**

1. **Вибір виду тестування**

Тестування чого-небудь означає підтвердження того, чи відповідають фактичні результати очікуваним. Ті самі принципи застосовуються у випадку програмного забезпечення, щоб забезпечити відсутність програмної системи дефектів . Тестування є тривалим процесом і має різні процедури виконання. Ці кроки гарантують, що кожен із програмних компонентів або системних компонентів відповідає очікуваним стандартам продуктивності, встановленим галузями або споживачами, перш ніж поставляти товар на ринок.

Ці кроки та процеси також допомагають виявити помилки, прогалини чи відсутні вимоги, що суперечать фактичним вимогам. Тестування проводиться або вручну, або за допомогою автоматизованих інструментів [19].

Для тестування розроблених модулів скористаємося ручним тестуванням, застосовуючи як негативний, так й позитивний підхід.

1. **Тест план**

Визначення функцій та формування тест плану є головними складовими тестування. Отже спочатку треба визначити загальні функції всіх користувачів ІС, а також для кожного окремо згідно його посади.

Головною функцією яка об’єднує всіх користувачів є модуль авторизації. До модуля авторизації належать наступні функції:

* аутентифікація за логіном та паролем.

Агент з бронювання номерів має доступ до таких функції:

* створення броні;
* скасування броні;
* додавання кімнат;
* додавання клієнтів.

Менеджер має доступ до наступних функції:

* додавання співробітників;
* додавання облікових записів.

Системний адміністратор має доступ до таких функції:

* додавання співробітників;
* додавання облікових записів;
* пошук по параметрам;
* оновлення інформації про клієнтів;
* видалення інформації про клієнтів;
* оновлення інформації про співробітників;
* видалення інформації про співробітників;
* видалення номерів.

Розробимо тест-кейси, які необхідно перевірити. Для початку перевіримо функції модуля авторизації.

Таблиця 4.1 – Тест-кейс модуля авторизації

|  |  |  |  |
| --- | --- | --- | --- |
| Опис | | Перевірка можливості успішної авторизації користувача | |
| Передумови | | Запуск програми | |
| № | Дія | | Очікуваний результат |
| 1 | Ввести коректні дані | | Введено коректні дані |
| 2 | Натиснути кнопку «Увійти» | | З’являється повідомлення про успішну авторизацію та перехід в головне вікно програми |

Продовження таблиці 4.1

|  |  |
| --- | --- |
| Опис | Перевірка можливості невдалої авторизації користувача |
| Передумови | Запуск програми |

|  |  |  |
| --- | --- | --- |
| № | Дія | Очікуваний результат |
| 1 | Залишити всі поля не заповненими | Поля не заповнені |
| 2 | Натиснути кнопку «Увійти» | З’являється повідомлення про помилку і необхідність ввести логін та пароль |
| 3 | Заповнити всі поля не коректною інформацією | Поля заповнені не коректною інформацією |
| 4 | Натиснути кнопку «Увійти» | З’являється повідомлення про помилку і необхідність ввести коректні дані |

Розробимо тест-кейси для функцій агента з бронювання номерів.

Таблиця 4.2 – Тест-кейс для агент з бронювання номерів

|  |  |  |  |
| --- | --- | --- | --- |
| Опис | | Перевірка можливості створення броні | |
| Передумови | | Успішна авторизація з правами агента з бронювання номерів | |
| № | Дія | | Очікуваний результат |
| 1 | Обрати в меню пункт «Бронь»  Підменю «Створити бронь» | | Відкривається діалогове вікно «Створення броні» |
| 2 | Натиснути кнопку «Сформувати» | | В DataGrid виводиться список вільних номерів |
| 3 | Подвійним натисканням вибрати вільний номер | | Відкриття вікна «Активація броні» |
| 4 | Ввести коректні дані | | Введено коректні дані |

Продовження таблиці 4.2

|  |  |  |  |
| --- | --- | --- | --- |
| Опис | | Перевірка можливості створення броні | |
| Передумови | | Успішна авторизація з правами агента з бронювання номерів | |
| № | Дія | | Очікуваний результат |
| 5 | Натиснути кнопку «Забронювати» | | З’являється повідомлення про успішне створення броні, кнопка «Забронювати» стає недоступною |
| Опис | | Перевірка можливості скасування броні | |
| Передумови | | Успішна авторизація з правами агента з бронювання номерів | |
| № | Дія | | Очікуваний результат |
| 1 | Обрати в меню пункт «Бронь»  Підменю «Скасування бронь» | | Відкривається діалогове вікно «Скасування броні» |
| 2 | Натиснути кнопку «Сформувати» | | В DataGrid виводиться список заброньованих номерів |
| 3 | вибрати заброньований номер із списку | | Вибрано заброньований номер |
| 4 | Натиснути кнопку «Зняти бронь» | | З’являється повідомлення «Бронь скасована» |
| Опис | | Перевірка можливості додавання клієнта | |
| Передумови | | Успішна авторизація з правами агента з бронювання номерів | |
| № | Дія | | Очікуваний результат |
| 1 | Обрати в меню пункт «Управління записами»  Підменю «Додати номер» | | Відкривається діалогове вікно «Додати номер» |

Продовження таблиці 4.2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Опис | | Перевірка можливості додавання клієнта | | |
| Передумови | | Успішна авторизація з правами агента з бронювання номерів | | |
| № | Дія | | | Очікуваний результат |
| 2 | Обрати категорію номера з combobox`a | | | Категорія обрана |
| 3 | Натиснути кнопку «Додати» | | | З’являється повідомлення «Номер додано» |
| Опис | | | Перевірка можливості успішного додавання клієнта | |
| Передумови | | | Успішна авторизація з правами агента з бронювання номерів | |
| № | Дія | | | Очікуваний результат |
| 1 | Обрати в меню пункт «Управління записами» Підменю «Додати клієнта»» | | | Відкривається діалогове вікно «Додати клієнта» |
| 2 | Ввести коректні дані | | | Введено коректні дані |
| 3 | Натиснути кнопку «Додати» | | | З’являється повідомлення «Клієнта додано» |

Продовження таблиці 4.2

|  |  |  |  |
| --- | --- | --- | --- |
| Опис | | Перевірка можливості невдалого додавання клієнта | |
| Передумови | | Успішна авторизація з правами агента з бронювання номерів | |
| № | Дія | | Очікуваний результат |
| 1 | Обрати в меню пункт «Управління записами» Підменю «Додати клієнта»» | | Відкривається діалогове вікно «Додати клієнта» |
| 2 | Ввести некоректні дані | | Введено некоректні дані |
| 3 | Натиснути кнопку «Додати» | | З’являється повідомлення «Введіть коректні дані» |

Розробимо тест-кейси для менеджера.

Таблиця 4.3 – Тест-кейс для менеджера

|  |  |  |  |
| --- | --- | --- | --- |
| Опис | | Перевірка можливості успішного додавання співробітника | |
| Передумови | | Успішна авторизація з правами менеджера | |
| № | Дія | | Очікуваний результат |
| 1 | Обрати в меню пункт «Управління записами» Підменю «Додати співробітника»» | | Відкривається діалогове вікно «Додати співробітника» |
| 2 | Ввести коректні дані | | Введено коректні дані |
| 3 | Натиснути кнопку «Додати» | | З’являється повідомлення «Співробітник доданий» |

Продовження таблиці 4.3

|  |  |  |  |
| --- | --- | --- | --- |
| Опис | | Перевірка можливості невдалого додавання співробітника | |
| Передумови | | Успішна авторизація з правами менеджера | |
| № | Дія | | Очікуваний результат |
| 1 | Обрати в меню пункт «Управління записами» Підменю «Додати співробітника»» | | Відкривається діалогове вікно «Додати співробітника» |
| 2 | Ввести некоректні дані | | Введено некоректні дані |
| 3 | Натиснути кнопку «Додати» | | З’являється повідомлення «Введіть коректні дані» |
| Опис | | Перевірка можливості додавання облікового запису | |
| Передумови | | Успішна авторизація з правами менеджера | |
| № | Дія | | Очікуваний результат |
| 1 | Обрати в меню пункт «Управління обліковими записами» Підменю «Додати обліковий запис»» | | Відкривається діалогове вікно «Додати обліковий запис» |
| 2 | Ввести дані та надати права | | Введено дані та надано права |
| 3 | Натиснути кнопку «Додати» | | З’являється повідомлення «Обліковий запис доданий» |

Розробимо тест-кейси для системного адміністратора.

Таблиця 4.4 – Тест-кейс для системного адміністратора

|  |  |  |  |
| --- | --- | --- | --- |
| Опис | | Перевірка можливості пошуку | |
| Передумови | | Успішна авторизація з правами системного адміністратора | |
| № | Дія | | Очікуваний результат |
| 1 | На головному вікні обрати параметр для combox’a | | Параметр обраний |
| 2 | Ввести коректні дані | | Введено коректні дані |
| 3 | Натиснути кнопку «Пошук» | | В DataGrid з’являється запис |
| Опис | | Перевірка можливості невдалого пошуку | |
| Передумови | | Успішна авторизація з правами сис. адміністратора | |
| № | Дія | | Очікуваний результат |
| 1 | На головному вікні обрати параметр для combox’a | | Параметр обраний |
| 2 | Ввести некоректні дані | | Введено некоректні дані |
| 3 | Натиснути кнопку «Пошук» | | З’являється повідомлення «Введіть коректні дані» |
| Опис | | Перевірка можливості редагування запису про клієнта | |
| Передумови | | Успішний пошук по параметрах | |
| № | Дія | | Очікуваний результат |
| 1 | Подвійним натиском обрати запис з DataGrid | | Відкривається діалогове вікно «Клієнт» |
| 2 | Ввести нові дані | | Введено нові дані |
| 3 | Натиснути кнопку «Оновити» | | З’являється повідомлення «Дані оновлено» |

Продовження таблиці 4.4

|  |  |  |  |
| --- | --- | --- | --- |
| Опис | | Перевірка можливості видалення запису про клієнта | |
| Передумови | | Успішний пошук по параметрах | |
| № | Дія | | Очікуваний результат |
| 1 | Подвійним натиском обрати запис з DataGrid | | Відкривається діалогове вікно «Клієнт» |
| 2 | Натиснути кнопку «Видалити» | | З’являється повідомлення «Дані Видалено» |
| Опис | | Перевірка можливості видалення запису про номер | |
| Передумови | | Успішний пошук по параметрах | |
| № | Дія | | Очікуваний результат |
| 1 | Подвійним натиском обрати запис з DataGrid | | Відкривається діалогове вікно «Номер» |
| 2 | Натиснути кнопку «Видалити» | | З’являється повідомлення «Номер Видалено» |
| Опис | | Перевірка можливості редагування запису про співробітника | |
| Передумови | | Успішний пошук по параметрах | |
| № | Дія | | Очікуваний результат |
| 1 | Подвійним натиском обрати запис з DataGrid | | Відкривається діалогове вікно «Співробітник» |
| 2 | Ввести нові дані | | Введено нові дані |
| 3 | Натиснути кнопку «Оновити» | | З’являється повідомлення «Дані оновлено» |

Продовження таблиці 4.4

|  |  |  |  |
| --- | --- | --- | --- |
| Опис | | Перевірка можливості видалення запису про Співробітник | |
| Передумови | | Успішний пошук по параметрах | |
| № | Дія | | Очікуваний результат |
| 1 | Подвійним натиском обрати запис з DataGrid | | Відкривається діалогове вікно «Співробітник» |
| 2 | Натиснути кнопки «Видалити» | | З’являється кнопки «Дані Видалено» |

**ВИСНОВКИ**

В процесі кваліфікаційної роботи була розроблена і реалізована інтерактивна інформаційна система готельного бізнесу, яка відповідає мінімальному функціоналу, який необхідний для роботи готельного комплекса.

Реалізовані наступні функції:

* авторизація – слугує для розподілення обов’язків працівників, а також для захисту інформації про зареєстрованих клієнтів;
* бронь / зняття броні номерів – є однією з основних функцій ІС, яка безпосередньо пов'язана з роботою готельного бізнесу;
* додавання / видалення / редагування інформації про клієнтів, співробітників, номери та облікові записи – дана функція підтримує актуальність і коректність даних.

Отже. поставлені завдання в роботі вважаю виконаними, а мету досягнутою. В подальшому планується розроблення більш складних модулів, таких як: ведення рахунків гостей, нічний аудит, облік додаткових послуг, організація тарифів, формування звітів. Також планується привести дану ІС до розподіленої бази даних на виділеному сервері, тим самим розділивши клієнтську і серверну частини.

**СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ**

1. Мэтью Мак-ДональдProWPF 4.5in *C# 5.0:* Windows Presentation Foundationin *.*NET 4.5, 4th edition. Matthew MacDonald*.* / Мэтью Мак-Дональд. – М.: «Вильямс», 2012. – 1024 с.
2. Самоучитель Microsoft Access 2016 Бекаревич Ю.Б., Пушкина Н.В. / Бакаревич Ю.Б. Пушкіна Н.В. СПб.: БХВ-Петербург, 2017. — 480 с
3. Г.В. Головко, А.М. Гафіяк Управління базами даних. Частина 1. Проектування баз даних. Бази даних та інформаційні системи. Конспект лекцій – Полтава, Полтавський національний технічний університет імені Юрія Кондратюка, 2008
4. Аналіз ринку, конкурентний аналіз [Електронний ресурс]. – Режим доступу: http://um.co.ua/7/7-13/7-130329.html
5. Hotel news resource Amadeus [Електронний ресурс]. – Режим доступу: https://www.hotelnewsresource.com/article33754.html
6. Автоматизація готелів Amadeus [Електронний ресурс]. – Режим доступу: http://servio.ru/automation-of-hotels-guest-houses
7. Системи автоматизації готелів [Електронний ресурс]. – Режим доступу: https://bstudy.net/921754/ekonomika/sistemy\_avtomatizatsii
8. Використання діаграм використання та діяльності [Електронний ресурс]. – Режим доступу: https://medium.com/@andrerahardjo/use-case-diagram-dan-activity-diagram-2b30f4471613
9. C++ vs C# порівняння [Електронний ресурс]. – Режим доступу: https://www.bitdegree.org/tutorials/c-sharp-vs-c-plus-plus/#c-vs-c-performance-which-works-faster
10. С++ vs C# порівняння і протиставлення [Електронний ресурс]. – Режим доступу: https://careerkarma.com/blog/c-plus-plus-vs-c-sharp/
11. Правила синтаксису C++ vs C# протиставлення [Електронний ресурс]. – Режим доступу: https://www.bitdegree.org/tutorials/c-sharp-vs-c-plus-plus/#syntax-rules-c-vs-c
12. Вибір між C++ і C# [Електронний ресурс]. – Режим доступу: https://habr.com/ru/post/262461/
13. C# проти C++ основні подібності [Електронний ресурс]. – Режим доступу: https://www.upwork.com/resources/c-sharp-vs-c-plus-plus
14. WPF vs Windows Forms [Електронний ресурс]. – Режим доступу: https://www.c-sharpcorner.com/article/wpf-vs-winforms/
15. Система керування базами даних [Електронний ресурс]. – Режим доступу: https://intellipaat.com/blog/dbms-vs-rdbms-difference/#no2
16. Система керування базами даних та їх вимоги [Електронний ресурс]. – Режим доступу: https://works.doklad.ru/view/G76l5wvGedU.html
17. Система керування базами даних та їх вимоги [Електронний ресурс]. – Режим доступу: http://www.gaps.tstu.ru/win-1251/lab/sreda/saoi/lect\_g4.html
18. Система керування базами даних та їх вимоги [Електронний ресурс]. – Режим доступу: https://smekni.com/a/312036/sistema-upravleniya-bazoy-dannykh-2/
19. Що таке тестування програмного забезпечення? [Електронний ресурс]. – Режим доступу: https://www.technotification.com/2018/08/software-testing-and-importance.html
20. Perfect Software and other illusions about testing. Gerald M. Weinberg Gerald M. Weinsberg / Джеральд Вайнсберг – М.: « Weinberg & Weinberg », 2011. – 200 с.

**ДОДАТОК А**

**ВИХІДНИЙ КОД ОСНОВНИХ КОМПОНЕНТІВ**

Клас Login(LoginWindow.xaml.cs)

public partial class LoginWindow : Window

{

public LoginWindow()

{

InitializeComponent();

}

private void LoginBtn\_Click(object sender, RoutedEventArgs e)

{

if (loginField.Text == "" && passField.Password == "")

{

MessageBox.Show("Заповніть всі поля", "Увага!");

}

else

{

string cmdtxt = "SELECT \* FROM [Аунтефікація] WHERE [Логін] =" + "\"" + loginField.Text + "\"" + " AND [Пароль] =" + "\"" + passField.Password + "\"";

string connString = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=hotel.mdb";

OleDbDataAdapter Login = new OleDbDataAdapter(cmdtxt, connString);

DataSet dt = new DataSet();

Login.Fill(dt);

if (dt.Tables[0].Rows.Count == 1)

{

MessageBox.Show("Ласкаво просимо, " + dt.Tables[0].Rows[0][3].ToString() + "!");

var f2 = new MainWindow();

f2.додатиКлієнтаToolStripMenuItem.IsEnabled = Convert.ToBoolean(dt.Tables[0].Rows[0][6].ToString());

f2.броньToolStripMenuItem.IsEnabled = Convert.ToBoolean(dt.Tables[0].Rows[0][7].ToString());

f2.управлінняОбліковимиЗаписамиToolStripMenuItem.IsEnabled = Convert.ToBoolean(dt.Tables[0].Rows[0][8].ToString());

f2.додатиToolStripMenuItem.IsEnabled = Convert.ToBoolean(dt.Tables[0].Rows[0][9].ToString());

f2.додатиНомерToolStripMenuItem.IsEnabled = Convert.ToBoolean(dt.Tables[0].Rows[0][11].ToString());

f2.button1.IsEnabled = Convert.ToBoolean(dt.Tables[0].Rows[0][5].ToString());

Class1.adb = dt.Tables[0].Rows[0][4].ToString();

var str = f2.Title = "Управління готелем, " + dt.Tables[0].Rows[0][4].ToString() + " " + dt.Tables[0].Rows[0][3].ToString();

Hide();

f2.Show();

}

else

{

MessageBox.Show("Невірний логін або пароль ", "Увага!");

}

}

}

private void ExitBtn\_Click(object sender, RoutedEventArgs e)

{

Application.Current.Shutdown();

}

}

Клас Main (MainWindow.xaml.cs)

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

if (comboBox1.Items.Count == 0)

{

comboBox1.Items.Add("ID\_СПІВРОБІТНИКА");

comboBox1.Items.Add("ID\_КЛІЄНТА");

comboBox1.Items.Add("ID\_НОМЕРА");

comboBox1.Items.Add("ПРІЗВИЩЕ СПІВРОБІТНИКА");

comboBox1.Items.Add("ПРІЗВИЩЕ КЛІЄНТА");

comboBox1.Items.Add("АРХІВ");

}

}

string connString = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=hotel.mdb";

private void MenuItem\_Click(object sender, RoutedEventArgs e)

{

this.Close();

var loginWindow = new LoginWindow();

loginWindow.Show();

}

private void створенняБроніToolStripMenuItem\_Click(object sender, RoutedEventArgs e)

{

var reservationWindow = new ReservationWindow();

Hide();

reservationWindow.Show();

}

private void скасуванняБроніToolStripMenuItem\_Click(object sender, RoutedEventArgs e)

{

var cancelReservationWindow = new CancelReservationWindow();

cancelReservationWindow.button2.IsEnabled = false;

Hide();

cancelReservationWindow.Show();

}

private void OnAutoGeneratingColumn(object sender, DataGridAutoGeneratingColumnEventArgs e)

{

if (e.PropertyType == typeof(System.DateTime))

(e.Column as DataGridTextColumn).Binding.StringFormat = "dd/MM/yyyy";

}

private void додатиКлієнтаToolStripMenuItem\_Click(object sender, RoutedEventArgs e)

{

var addClientWindow = new AddClientWindow();

string cmdtxt = "SELECT MAX(Код) FROM [Клієнти]";

OleDbDataAdapter dta = new OleDbDataAdapter(cmdtxt, connString);

DataSet dt = new DataSet();

dta.Fill(dt);

string txt = dt.Tables[0].Rows[0][0].ToString();

int Toin = int.Parse(txt) + 1;

addClientWindow.ID.Text = Toin.ToString();

addClientWindow.ID.IsEnabled = false;

Hide();

addClientWindow.Show();

}

private void додатиToolStripMenuItem\_Click(object sender, RoutedEventArgs e)

{

var addEmployeeWindow = new AddEmployeeWindow();

string cmdtxt = "SELECT MAX(Код) FROM [Співробітники]";

OleDbDataAdapter dta = new OleDbDataAdapter(cmdtxt, connString);

DataSet dt = new DataSet();

dta.Fill(dt);

string txt = dt.Tables[0].Rows[0][0].ToString();

int Toin = int.Parse(txt) + 1;

addEmployeeWindow.IDwork.Text = Toin.ToString();

addEmployeeWindow.IDwork.IsEnabled = false;

Hide();

addEmployeeWindow.Show();

}

private void додатиНомерToolStripMenuItem\_Click(object sender, RoutedEventArgs e)

{

var addNumberWindow = new AddNumberWindow();

string cmdtxt = "SELECT MAX(Код) FROM [Номери]";

OleDbDataAdapter dta = new OleDbDataAdapter(cmdtxt, connString);

DataSet dt = new DataSet();

dta.Fill(dt);

string txt = dt.Tables[0].Rows[0][0].ToString();

int Toin = int.Parse(txt) + 1;

addNumberWindow.IDRoom.Text = Toin.ToString();

addNumberWindow.IDRoom.IsEnabled = false;

Hide();

addNumberWindow.Show();

}

private void додатиОбліковийЗаписToolStripMenuItem\_Click(object sender, RoutedEventArgs e)

{

var addAccountsWindow = new AddAccountsWindow();

string cmdtxt = "SELECT MAX(Код) FROM [Аунтефікація]";

OleDbDataAdapter dta = new OleDbDataAdapter(cmdtxt, connString);

DataSet dt = new DataSet();

dta.Fill(dt);

string txt = dt.Tables[0].Rows[0][0].ToString();

int Toin = int.Parse(txt) + 1;

addAccountsWindow.IDw.Text = Toin.ToString();

addAccountsWindow.IDw.IsEnabled = false;

Hide();

addAccountsWindow.Show();

}

private void textBox1\_TextChanged(object sender, TextChangedEventArgs e)

{

}

private void button1\_Click(object sender, RoutedEventArgs e)

{

try

{

if (comboBox1.Text == "ПРІЗВИЩЕ СПІВРОБІТНИКА")

{

string cmd = "SELECT \* FROM [Співробітники] WHERE [Прізвище]=" + "\"" + textBox1.Text + "\"";

OleDbDataAdapter dta = new OleDbDataAdapter(cmd, connString);

DataSet ds = new DataSet();

dta.Fill(ds);

dataGridView1.ItemsSource = ds.Tables[0].DefaultView;

}

else if (comboBox1.Text == "ID\_СПІВРОБІТНИКА")

{

string cmd = "SELECT \* FROM [Співробітники] WHERE [Код]=" + textBox1.Text;

OleDbDataAdapter dta = new OleDbDataAdapter(cmd, connString);

DataSet ds = new DataSet();

dta.Fill(ds);

dataGridView1.ItemsSource = ds.Tables[0].DefaultView;

}

else if (comboBox1.Text == "ПРІЗВИЩЕ КЛІЄНТА")

{

string cmd = "SELECT \* FROM [Клієнти] WHERE [Прізвище]=" + "\"" + textBox1.Text + "\"";

OleDbDataAdapter dta = new OleDbDataAdapter(cmd, connString);

DataSet ds = new DataSet();

dta.Fill(ds);

dataGridView1.ItemsSource = ds.Tables[0].DefaultView;

}

else if (comboBox1.Text == "ID\_КЛІЄНТА")

{

string cmd = "SELECT \* FROM [Клієнти] WHERE [Код]=" + textBox1.Text;

OleDbDataAdapter dta = new OleDbDataAdapter(cmd, connString);

DataSet ds = new DataSet();

dta.Fill(ds);

dataGridView1.ItemsSource = ds.Tables[0].DefaultView;

}

else if (comboBox1.Text == "ID\_НОМЕРА")

{

string cmd = "SELECT \* FROM [Номери] WHERE [Код]=" + textBox1.Text;

OleDbDataAdapter dta = new OleDbDataAdapter(cmd, connString);

DataSet ds = new DataSet();

dta.Fill(ds);

dataGridView1.ItemsSource = ds.Tables[0].DefaultView;

}

else if (comboBox1.Text == "АРХІВ")

{

if (textBox1.Text == "")

{

string cmd = "SELECT \* FROM [Архів]";

OleDbDataAdapter dta = new OleDbDataAdapter(cmd, connString);

DataSet ds = new DataSet();

dta.Fill(ds);

dataGridView1.ItemsSource = ds.Tables[0].DefaultView;

}

else

{

string cmd = "SELECT \* FROM [Архів] WHERE [Прізвище заброньованої людини]=" + "\"" + textBox1.Text + "\"";

OleDbDataAdapter dta = new OleDbDataAdapter(cmd, connString);

DataSet ds = new DataSet();

dta.Fill(ds);

dataGridView1.ItemsSource = ds.Tables[0].DefaultView;

}

}

}

catch

{

MessageBox.Show("введіть коректні дані");

}

}

private void DataGridCell\_MouseDoubleClick(object sender, MouseButtonEventArgs e)

{

if (comboBox1.Text == "ID\_КЛІЄНТА")

{

var f10 = new ClientInfoWindow();

OleDbDataAdapter dta = new OleDbDataAdapter();

var row = (dataGridView1.SelectedItem as DataRowView).Row;

f10.ID.Text = Convert.ToString(row.ItemArray[0]);

f10.Lname.Text = Convert.ToString(row.ItemArray[1]);

f10.FName.Text = Convert.ToString(row.ItemArray[2]);

f10.Middle.Text = Convert.ToString(row.ItemArray[3]);

f10.BirthDay.Text = Convert.ToString(row.ItemArray[4]);

f10.Male.Text = Convert.ToString(row.ItemArray[5]);

f10.DocType.Text = Convert.ToString(row.ItemArray[6]);

f10.Passport\_series.Text = Convert.ToString(row.ItemArray[7]);

f10.numberd.Text = Convert.ToString(row.ItemArray[8]);

f10.Comentary.Text = Convert.ToString(row.ItemArray[9]);

string cmdtxt = "SELECT \* FROM [Контакти\_клієнтів] WHERE [ID]= " + textBox1.Text;

OleDbDataAdapter dt2 = new OleDbDataAdapter(cmdtxt, connString);

DataSet ds1 = new DataSet();

dt2.Fill(ds1);

f10.PhNumber.Text = ds1.Tables[0].Rows[0][1].ToString();

f10.address.Text = ds1.Tables[0].Rows[0][2].ToString();

f10.ID.IsEnabled = false;

Hide();

f10.Show();

}

else if (comboBox1.Text == "ПРІЗВИЩЕ КЛІЄНТА")

{

var f10 = new ClientInfoWindow();

OleDbDataAdapter dta = new OleDbDataAdapter();

var row = (dataGridView1.SelectedItem as DataRowView).Row;

f10.ID.Text = Convert.ToString(row.ItemArray[0]);

f10.Lname.Text = Convert.ToString(row.ItemArray[1]);

f10.FName.Text = Convert.ToString(row.ItemArray[2]);

f10.Middle.Text = Convert.ToString(row.ItemArray[3]);

f10.BirthDay.Text = Convert.ToString(row.ItemArray[4]);

f10.Male.Text = Convert.ToString(row.ItemArray[5]);

f10.DocType.Text = Convert.ToString(row.ItemArray[6]);

f10.Passport\_series.Text = Convert.ToString(row.ItemArray[7]);

f10.numberd.Text = Convert.ToString(row.ItemArray[8]);

f10.Comentary.Text = Convert.ToString(row.ItemArray[9]);

string cmdtxt = " SELECT \* FROM [Контакти\_клієнтів] WHERE ID IN (SELECT [Код] FROM [Клієнти] WHERE ([Прізвище] = " + "\"" + textBox1.Text + "\"" + "))";

OleDbDataAdapter dt2 = new OleDbDataAdapter(cmdtxt, connString);

DataSet ds1 = new DataSet();

dt2.Fill(ds1);

f10.PhNumber.Text = ds1.Tables[0].Rows[0][1].ToString();

f10.address.Text = ds1.Tables[0].Rows[0][2].ToString();

f10.ID.IsEnabled = false;

Hide();

f10.Show();

}

else if (comboBox1.Text == "ID\_СПІВРОБІТНИКА" || comboBox1.Text == "ПРІЗВИЩЕ СПІВРОБІТНИКА")

{

try

{

var f11 = new EmployeeInfoWindow();

OleDbDataAdapter dta = new OleDbDataAdapter();

var row2 = (dataGridView1.SelectedItem as DataRowView).Row;

f11.IDwork.Text = Convert.ToString(row2.ItemArray[0]);

f11.LName\_work.Text = Convert.ToString(row2.ItemArray[1]);

f11.FName\_work.Text = Convert.ToString(row2.ItemArray[2]);

f11.middle\_work.Text = Convert.ToString(row2.ItemArray[3]);

f11.birthday.Text = Convert.ToString(row2.ItemArray[4]);

f11.degree.Text = Convert.ToString(row2.ItemArray[5]);

f11.Posada.Text = Convert.ToString(row2.ItemArray[6]);

f11.address\_work.Text = Convert.ToString(row2.ItemArray[7]);

f11.Phone\_work.Text = Convert.ToString(row2.ItemArray[8]);

f11.Pay.Text = Convert.ToString(row2.ItemArray[9]);

f11.coment.Text = Convert.ToString(row2.ItemArray[10]);

f11.IDwork.IsEnabled = false;

Hide();

f11.Show();

}

catch { }

}

else if (comboBox1.Text == "ID\_НОМЕРА")

{

var f12 = new NumberInfoWindow();

OleDbDataAdapter dta = new OleDbDataAdapter();

var row3 = (dataGridView1.SelectedItem as DataRowView).Row;

f12.IDRoom.Text = Convert.ToString(row3.ItemArray[0]);

f12.Category.Text = Convert.ToString(row3.ItemArray[1]);

f12.IDRoom.IsEnabled = false;

f12.Category.IsEnabled = false;

Hide();

f12.Show();

}

else if (comboBox1.Text == "АРХІВ")

{

var f13 = new ArchiveWindow();

OleDbDataAdapter dta = new OleDbDataAdapter();

var row4 = (dataGridView1.SelectedItem as DataRowView).Row;

f13.IDa.Text = Convert.ToString(row4.ItemArray[0]);

f13.roomn.Text = Convert.ToString(row4.ItemArray[1]);

f13.reserv.Text = Convert.ToString(row4.ItemArray[2]);

f13.phnum.Text = Convert.ToString(row4.ItemArray[3]);

f13.Indate.Text = Convert.ToString(row4.ItemArray[4]);

f13.outdate.Text = Convert.ToString(row4.ItemArray[5]);

f13.workerr.Text = Convert.ToString(row4.ItemArray[6]);

f13.IDa.IsEnabled = false;

f13.roomn.IsEnabled = false;

f13.reserv.IsEnabled = false;

f13.phnum.IsEnabled = false;

f13.Indate.IsEnabled = false;

f13.outdate.IsEnabled = false;

f13.workerr.IsEnabled = false;

Hide();

f13.Show();

}

}

private void MenuItem\_Click\_1(object sender, RoutedEventArgs e)

{

System.Windows.Application.Current.Shutdown();

}

}

}

Клас Activation(Reservationwindow.xaml.cs)

public partial class ReservationWindow : Window

{

string connString = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=hotel.mdb";

public ReservationWindow()

{

InitializeComponent();

}

private void button2\_Click(object sender, RoutedEventArgs e)

{

this.Close();

var form2 = new MainWindow();

form2.Show();

}

private void button1\_Click(object sender, RoutedEventArgs e)

{

string cmd = "SELECT \* FROM [Номери] ";

OleDbDataAdapter dta = new OleDbDataAdapter(cmd, connString);

DataSet ds = new DataSet();

dta.Fill(ds);

dataGridView1.ItemsSource = ds.Tables[0].DefaultView;

}

private void DataGridCell\_MouseDoubleClick(object sender, MouseButtonEventArgs e)

{

var f5 = new ActivationBookingWindow();

OleDbDataAdapter dta = new OleDbDataAdapter();

var row = (dataGridView1.SelectedItem as DataRowView).Row;

f5.textBox5.Text = Convert.ToString(row.ItemArray[0]);

f5.textBox5.Text = Convert.ToString(row.ItemArray[1]);

f5.RoomBox.Text = Convert.ToString(row.ItemArray[0]);

f5.checkBox1.IsChecked = Convert.ToBoolean(row.ItemArray[4]);

f5.InDate.Text = Convert.ToString(row.ItemArray[2]);

f5.OutDate.Text = Convert.ToString(row.ItemArray[3]);

f5.textBox5.IsEnabled = false;

f5.checkBox1.IsEnabled = false;

f5.RoomBox.IsEnabled = false;

if (f5.checkBox1.IsChecked == true)

{ f5.button1.IsEnabled = false; }

Hide();

f5.Show();

}

}

}

Клас Activation(activationbookingwindow.xaml.cs)

public partial class ActivationBookingWindow : Window

{

public ActivationBookingWindow()

{

InitializeComponent();

InDate.SelectedDate = DateTime.Today;

OutDate.SelectedDate = DateTime.Today;

if (checkBox1.IsChecked == true)

{ button1.IsEnabled = false; }

string qry = " SELECT [Прізвище] FROM [Клієнти]";

var dta = new OleDbDataAdapter(qry, connString);

var ds = new DataSet();

dta.Fill(ds);

foreach (var item in ds.Tables[0].DefaultView)

{

var row = (item as DataRowView).Row;

comboBox1.Items.Add(row.ItemArray[0]);

}

}

string connString = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=hotel.mdb";

public string creator = Class1.adb;

private void TextBox\_TextChanged(object sender, TextChangedEventArgs e)

{

}

private void button1\_Click(object sender, RoutedEventArgs e)

{

string qry;

OleDbDataAdapter dta = new OleDbDataAdapter();

try

{

qry = "INSERT INTO [Архів] ([номер кімнати], [Прізвище заброньованої людини], [номер телефону],[Дата заселення],[Дата виселення],[Співробітник,прийнявший заявку]) Values(" + "\"" + RoomBox.Text + "\"" + "," + "\"" + comboBox1.Text + "\"" + "," + "\"" + PhNumber.Text + "\"" + "," + "\"" + InDate.Text + "\"" + "," + "\"" + OutDate.Text + "\"" + "," + "\"" + creator + "\"" + ")";

dta = new OleDbDataAdapter(qry, connString);

DataSet ds = new DataSet();

dta.Fill(ds);

qry = "UPDATE [Номери] SET Статус=true, [Дата заселення]=" + "\"" + InDate.Text + "\"" + ",[Дата виселення] =" + "\"" + OutDate.Text + "\"" + " WHERE Код =" + RoomBox.Text;

dta = new OleDbDataAdapter(qry, connString);

DataSet ds1 = new DataSet();

dta.Fill(ds1);

checkBox1.IsChecked = true;

if (checkBox1.IsChecked == true)

{ button1.IsEnabled = false; }

}

catch

{

MessageBox.Show("Перевірте правильність даних");

}

}

private void button2\_Click(object sender, RoutedEventArgs e)

{

this.Close();

var form3 = new ReservationWindow();

form3.Show();

}

private void comboBox1\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

string qry = " SELECT [Імя] FROM [Клієнти] WHERE Прізвище =" + "\"" + e.AddedItems[0].ToString() + "\"";

var dta = new OleDbDataAdapter(qry, connString);

var ds = new DataSet();

dta.Fill(ds);

comboBox2.SelectedItem = null;

comboBox2.Items.Clear();

foreach (var item in ds.Tables[0].DefaultView)

{

var row = (item as DataRowView).Row;

comboBox2.Items.Add(row.ItemArray[0]);

}

comboBox2.SelectedIndex = 0;

}

private void comboBox2\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

if (e.AddedItems.Count == 0)

{

return;

}

string qry = "SELECT [ПоБатькові] FROM [Клієнти] WHERE Імя =" + "\"" + e.AddedItems[0].ToString() + "\"" + "AND Прізвище =" + "\"" + comboBox1.SelectedItem + "\"";

var dta = new OleDbDataAdapter(qry, connString);

var ds = new DataSet();

dta.Fill(ds);

comboBox3.SelectedItem = null;

comboBox3.Items.Clear();

foreach (var item in ds.Tables[0].DefaultView)

{

var row = (item as DataRowView).Row;

comboBox3.Items.Add(row.ItemArray[0]);

}

comboBox3.SelectedIndex = 0;

}

private void comboBox3\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

if (e.AddedItems.Count == 0)

{

return;

}

if (comboBox1.Text != " " && comboBox2.Text != " ")

{

string qry = " SELECT [Номер\_телефону] FROM [Контакти\_клієнтів] WHERE ID IN (SELECT [Код] FROM [Клієнти] WHERE ((Прізвище = " + "\"" + comboBox1.SelectedItem + "\"" + "AND Імя =" + "\"" + comboBox2.SelectedItem + "\"" + ")AND ПоБатькові =" + "\"" + e.AddedItems[0].ToString() + "\"" + "))";

OleDbDataAdapter dta3 = new OleDbDataAdapter(qry, connString);

DataSet ds = new DataSet();

dta3.Fill(ds);

PhNumber.Text = ds.Tables[0].Rows[0][0].ToString();

}

}

}

}

Клас Cancel (cancelReservation.xaml.cs)

public partial class CancelReservationWindow : Window

{

string connString = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=hotel.mdb";

int changedindex = 0;

public CancelReservationWindow()

{

InitializeComponent();

}

private void button1\_Click(object sender, RoutedEventArgs e)

{

string cmd = "SELECT\* FROM [Номери] WHERE Статус=true";

OleDbDataAdapter dta = new OleDbDataAdapter(cmd, connString);

DataSet ds = new DataSet();

dta.Fill(ds);

dataGridView1.ItemsSource = ds.Tables[0].DefaultView;

button2.IsEnabled = true;

}

private void button2\_Click(object sender, RoutedEventArgs e)

{

if (!(dataGridView1.SelectedItem is DataRowView))

{

return;

}

var row = (dataGridView1.SelectedItem as DataRowView).Row;

string qry = "UPDATE [Номери] SET Статус=false, [Дата заселення]=\"\"" + ", [Дата виселення]=\"\"" + " WHERE Код=" + row.ItemArray[0].ToString();

OleDbDataAdapter dta = new OleDbDataAdapter(qry, connString);

DataSet ds = new DataSet();

dta.Fill(ds);

button1\_Click(sender, e);

}

private void button3\_Click(object sender, RoutedEventArgs e)

{

this.Close();

var f2 = new MainWindow();

f2.Show();

}

}

}

Клас AddCleint (AddCleintwindow.xaml.cs)

public partial class AddClientWindow : Window

{

public AddClientWindow()

{

InitializeComponent();

BirthDay.SelectedDate = DateTime.Today;

}

string connString = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=hotel.mdb";

private void ID\_TextChanged(object sender, TextChangedEventArgs e)

{

}

private void button2\_Click(object sender, RoutedEventArgs e)

{

this.Close();

var form2 = new MainWindow();

form2.Show();

}

private void button1\_Click(object sender, RoutedEventArgs e)

{

string qry = " INSERT INTO [Клієнти] ([Прізвище],[Імя],[ПоБатькові],[Дата народження],[Стать],[Вид документу],[Серія],[Номер],[Коментар]) Values(" + "\"" + Lname.Text + "\"" + "," + "\"" + FName.Text + "\"" + "," + "\"" + Middle.Text + "\"" + "," + "\"" + BirthDay.Text + "\"" + "," + "\"" + Male.Text + "\"" + "," + "\"" + DocType.Text + "\"" + "," + "\"" + Passport\_series.Text + "\"" + "," + "\"" + numberd.Text + "\"" + "," + "\"" + Comentary.Text + "\"" + ")";

string qry2 = " INSERT INTO [Контакти\_клієнтів] ([Номер\_телефону], [Адреса]) Values (" + "\"" + PhNumber.Text + "\"" + "," + "\"" + address.Text + "\"" + ")";

OleDbDataAdapter dta = new OleDbDataAdapter(qry, connString);

OleDbDataAdapter dta2 = new OleDbDataAdapter(qry2, connString);

DataSet ds = new DataSet();

DataSet ds2 = new DataSet();

dta.Fill(ds);

dta2.Fill(ds2);

MessageBox.Show("Клієнт успішно доданий");

this.Close();

var f2 = new MainWindow();

f2.Show();

}

private void PhNumber\_PreviewTextInput(object sender, TextCompositionEventArgs e)

{

Regex regex = new Regex("[^0-9]+");

e.Handled = regex.IsMatch(e.Text);

}

private void Passport\_series\_PreviewTextInput(object sender, TextCompositionEventArgs e)

{

Regex regex = new Regex("[^0-9]+");

e.Handled = regex.IsMatch(e.Text);

}

private void numberd\_PreviewTextInput(object sender, TextCompositionEventArgs e)

{

Regex regex = new Regex("[^0-9]+");

e.Handled = regex.IsMatch(e.Text);

}

}

}

Клас AddEmployeeWindow (AddEmployeeWindow.xaml.cs)

public partial class AddEmployeeWindow : Window

{

public AddEmployeeWindow()

{

InitializeComponent();

birthday.SelectedDate = DateTime.Today;

}

string connString = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=hotel.mdb";

private void ID\_TextChanged(object sender, TextChangedEventArgs e)

{

}

private void Pay\_PreviewTextInput(object sender, TextCompositionEventArgs e)

{

Regex regex = new Regex("[^0-9]+");

e.Handled = regex.IsMatch(e.Text);

}

private void button2\_Click(object sender, RoutedEventArgs e)

{

this.Close();

var form2 = new MainWindow();

form2.Show();

}

private void Add\_worker\_Click(object sender, RoutedEventArgs e)

{

string qry = "INSERT INTO [Співробітники] ([Прізвище],[Ім'я],[По батькові],[Дата народження],[Освіта],[Посада],[Адреса], [номер телефону], [Заробітна плата $], [Додатковий коментар] ) Values (" + "\"" + LName\_work.Text + "\"" + "," + "\"" + FName\_work.Text + "\"" + "," + "\"" + middle\_work.Text + "\"" + "," + "\"" + birthday.Text + "\"" + "," + "\"" + degree.Text + "\"" + "," + "\"" + Posada.Text + "\"" + "," + "\"" + address\_work.Text + "\"" + "," + "\"" + Phone\_work.Text + "\"" + "," + "\"" + Pay.Text + "\"" + "," + "\"" + comment.Text + "\"" + ")";

OleDbDataAdapter dta = new OleDbDataAdapter(qry, connString);

DataSet ds = new DataSet();

dta.Fill(ds);

MessageBox.Show("Співробітник успішно доданий");

this.Close();

var f2 = new MainWindow();

f2.Show();

}

}

}

Клас AddNumber (AddNumberWindow.xaml.cs)

public partial class AddNumberWindow : Window

{

public AddNumberWindow()

{

InitializeComponent();

SetCategories();

}

string connString = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=hotel.mdb";

private void SetCategories()

{

var qry = "select \* from [Категорія\_номерів]";

var dta = new OleDbDataAdapter(qry, connString);

var ds = new DataSet();

dta.Fill(ds);

foreach (var item in ds.Tables[0].DefaultView)

{

var row = (item as DataRowView).Row;

Category.Items.Add(row.ItemArray[1]);

}

}

private void button1\_Click(object sender, RoutedEventArgs e)

{

string qry = "INSERT INTO [Номери] (Категорія) Values (" + "\"" + Category.Text + "\"" + ")";

OleDbDataAdapter dta = new OleDbDataAdapter(qry, connString);

DataSet ds = new DataSet();

dta.Fill(ds);

this.Close();

MessageBox.Show("Номер успішно доданий");

this.Close();

var form2 = new MainWindow();

form2.Show();

}

private void button2\_Click(object sender, RoutedEventArgs e)

{

var f2 = new MainWindow();

this.Close();

f2.Show();

}

}

}

Клас AddAccount (AddAccountsWindow.xaml.cs)

public partial class AddAccountsWindow : Window

{

public AddAccountsWindow()

{

InitializeComponent();

}

private void TextBox\_TextChanged(object sender, TextChangedEventArgs e)

{

}

private void button1\_Click(object sender, RoutedEventArgs e)

{

string connString = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=hotel.mdb";

string qry = "INSERT INTO [Аунтефікація] ([Логін],[Пароль],[Ім'я],[Посада],[Редагування таблиць БД],[Додавання клієнтів],[Бронь номерів], [Управління обліковками],[Додавання Співробітників], [Видалення записів БД], [Додавання кімнат] ) Values (" + "\"" + Log.Text + "\"" + "," + "\"" + Pass.Text + "\"" + "," + "\"" + Name.Text + "\"" + "," + "\"" + Posada.Text + "\"" + "," + checkBox1.IsChecked + "," + checkBox2.IsChecked + "," + checkBox3.IsChecked + "," + checkBox4.IsChecked + "," + checkBox5.IsChecked + "," + checkBox6.IsChecked + "," + checkBox7.IsChecked + ")";

OleDbDataAdapter dta = new OleDbDataAdapter(qry, connString);

DataSet ds = new DataSet();

dta.Fill(ds);

MessageBox.Show("Співробітник успішно доданий");

this.Close();

var form2 = new MainWindow();

form2.Show();

}

private void button2\_Click(object sender, RoutedEventArgs e)

{

this.Close();

var form2 = new MainWindow();

form2.Show();

}

}

}

Клас ClientInfo (ClientInfoWindow.xaml.cs)

public partial class ClientInfoWindow : Window

{

public ClientInfoWindow()

{

InitializeComponent();

}

string connString = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=hotel.mdb";

private void ID\_TextChanged(object sender, TextChangedEventArgs e)

{

}

private void button1\_Click(object sender, RoutedEventArgs e)

{

string qry = "UPDATE [Клієнти] SET [Прізвище]=" + "\"" + Lname.Text + "\"" + ",[Імя]=" + "\"" + FName.Text + "\"" + ",[ПоБатькові]=" + "\"" + Middle.Text + "\"" + ",[Дата народження]=" + "\"" + BirthDay.Text + "\"" + ",[Стать]=" + "\"" + Male.Text + "\"" + ",[Вид документу]=" + "\"" + DocType.Text + "\"" + ",[Серія]=" + "\"" + Passport\_series.Text + "\"" + ",[Номер]=" + "\"" + numberd.Text + "\"" + ",[Коментар]=" + "\"" + Comentary.Text + "\"" + "WHERE [Код]=" + ID.Text;

string qry2 = "UPDATE [Контакти\_клієнтів] SET [Номер\_телефону]=" + "\"" + PhNumber.Text + "\"" + ",[Адреса]=" + "\"" + address.Text + "\"" + "WHERE [ID]=" + ID.Text;

OleDbDataAdapter dta = new OleDbDataAdapter(qry, connString);

OleDbDataAdapter dta2 = new OleDbDataAdapter(qry2, connString);

DataSet ds = new DataSet();

DataSet ds2 = new DataSet();

dta.Fill(ds);

dta2.Fill(ds2);

MessageBox.Show("Дані оновлено!");

var f2 = new MainWindow();

this.Close();

f2.Show();

}

private void button2\_Click(object sender, RoutedEventArgs e)

{

string qry2 = "DELETE [Номер\_телефону]=" + "\"" + PhNumber.Text + "\"" + ",[Адреса]=" + "\"" + address.Text + "\"" + " FROM [Контакти\_клієнтів] WHERE [ID]=" + ID.Text;

string qry = "DELETE [Прізвище]=" + "\"" + Lname.Text + "\"" + ",[Імя]=" + "\"" + FName.Text + "\"" + ",[ПоБатькові]=" + "\"" + Middle.Text + "\"" + ",[Дата народження]=" + "\"" + BirthDay.Text + "\"" + ",[Стать]=" + "\"" + Male.Text + "\"" + ",[Вид документу]=" + "\"" + DocType.Text + "\"" + ",[Серія]=" + "\"" + Passport\_series.Text + "\"" + ",[Номер]=" + "\"" + numberd.Text + "\"" + ",[Коментар]=" + "\"" + Comentary.Text + "\"" + "FROM [Клієнти] WHERE [Код]=" + ID.Text;

OleDbDataAdapter dta2 = new OleDbDataAdapter(qry2, connString);

OleDbDataAdapter dta = new OleDbDataAdapter(qry, connString);

DataSet ds2 = new DataSet();

DataSet ds = new DataSet();

dta2.Fill(ds2);

dta.Fill(ds);

MessageBox.Show("Дані Видалено!");

var f2 = new MainWindow();

this.Close();

f2.Show();

}

}

}

Клас EmployeetInfo (EmployeeInfoWindow.xaml.cs)

public partial class EmployeeInfoWindow : Window

{

public EmployeeInfoWindow()

{

InitializeComponent();

}

string connString = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=hotel.mdb";

private void ID\_TextChanged(object sender, TextChangedEventArgs e)

{

}

private void Add\_worker\_Click(object sender, RoutedEventArgs e)

{

string qry = "UPDATE [Співробітники] SET [Прізвище]=" + "\"" + LName\_work.Text + "\"" + ",[Ім'я]=" + "\"" + FName\_work.Text + "\"" + ",[По батькові]=" + "\"" + middle\_work.Text + "\"" + ",[Дата народження]=" + "\"" + birthday.Text + "\"" + ",[Освіта]=" + "\"" + degree.Text + "\"" + ",[Посада]=" + "\"" + Posada.Text + "\"" + ",[Адреса]=" + "\"" + address\_work.Text + "\"" + ",[номер телефону]=" + "\"" + Phone\_work.Text + "\"" + ",[Заробітна плата $]=" + "\"" + Pay.Text + "\"" + ",[Додатковий коментар]=" + "\"" + coment.Text + "\"" + "WHERE [Код]=" + IDwork.Text;

OleDbDataAdapter dta = new OleDbDataAdapter(qry, connString);

DataSet ds = new DataSet();

dta.Fill(ds);

MessageBox.Show("Дані оновлено!");

var f2 = new MainWindow();

this.Close();

f2.Show();

}

private void delete\_worker\_Click(object sender, RoutedEventArgs e)

{

string qry = "DELETE [Прізвище]=" + "\"" + LName\_work.Text + "\"" + ",[Ім'я]=" + "\"" + FName\_work.Text + "\"" + ",[По батькові]=" + "\"" + middle\_work.Text + "\"" + ",[Дата народження]=" + "\"" + birthday.Text + "\"" + ",[Освіта]=" + "\"" + degree.Text + "\"" + ",[Посада]=" + "\"" + Posada.Text + "\"" + ",[Адреса]=" + "\"" + address\_work.Text + "\"" + ",[номер телефону]=" + "\"" + Phone\_work.Text + "\"" + ",[Заробітна плата $]=" + "\"" + Pay.Text + "\"" + ",[Додатковий коментар]=" + "\"" + coment.Text + "\"" + "FROM [Співробітники] WHERE [Код]=" + IDwork.Text;

OleDbDataAdapter dta = new OleDbDataAdapter(qry, connString);

DataSet ds = new DataSet();

dta.Fill(ds);

MessageBox.Show("Дані Видалено!");

var f2 = new MainWindow();

this.Close();

f2.Show();

}

}

}

Клас EmployeetInfo (EmployeeInfoWindow.xaml.cs)

public partial class EmployeeInfoWindow : Window

{

public EmployeeInfoWindow()

{

InitializeComponent();

}

string connString = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=hotel.mdb";

private void ID\_TextChanged(object sender, TextChangedEventArgs e)

{

}

private void Add\_worker\_Click(object sender, RoutedEventArgs e)

{

string qry = "UPDATE [Співробітники] SET [Прізвище]=" + "\"" + LName\_work.Text + "\"" + ",[Ім'я]=" + "\"" + FName\_work.Text + "\"" + ",[По батькові]=" + "\"" + middle\_work.Text + "\"" + ",[Дата народження]=" + "\"" + birthday.Text + "\"" + ",[Освіта]=" + "\"" + degree.Text + "\"" + ",[Посада]=" + "\"" + Posada.Text + "\"" + ",[Адреса]=" + "\"" + address\_work.Text + "\"" + ",[номер телефону]=" + "\"" + Phone\_work.Text + "\"" + ",[Заробітна плата $]=" + "\"" + Pay.Text + "\"" + ",[Додатковий коментар]=" + "\"" + coment.Text + "\"" + "WHERE [Код]=" + IDwork.Text;

OleDbDataAdapter dta = new OleDbDataAdapter(qry, connString);

DataSet ds = new DataSet();

dta.Fill(ds);

MessageBox.Show("Дані оновлено!");

var f2 = new MainWindow();

this.Close();

f2.Show();

}

private void delete\_worker\_Click(object sender, RoutedEventArgs e)

{

string qry = "DELETE [Прізвище]=" + "\"" + LName\_work.Text + "\"" + ",[Ім'я]=" + "\"" + FName\_work.Text + "\"" + ",[По батькові]=" + "\"" + middle\_work.Text + "\"" + ",[Дата народження]=" + "\"" + birthday.Text + "\"" + ",[Освіта]=" + "\"" + degree.Text + "\"" + ",[Посада]=" + "\"" + Posada.Text + "\"" + ",[Адреса]=" + "\"" + address\_work.Text + "\"" + ",[номер телефону]=" + "\"" + Phone\_work.Text + "\"" + ",[Заробітна плата $]=" + "\"" + Pay.Text + "\"" + ",[Додатковий коментар]=" + "\"" + coment.Text + "\"" + "FROM [Співробітники] WHERE [Код]=" + IDwork.Text;

OleDbDataAdapter dta = new OleDbDataAdapter(qry, connString);

DataSet ds = new DataSet();

dta.Fill(ds);

MessageBox.Show("Дані Видалено!");

var f2 = new MainWindow();

this.Close();

f2.Show();

}

}

}

Клас NumberInfo (NumberInfoWindow.xaml.cs)

public partial class NumberInfoWindow : Window

{

public NumberInfoWindow()

{

InitializeComponent();

}

string connString = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=hotel.mdb";

private void button1\_Click(object sender, RoutedEventArgs e)

{

string qry = "DELETE [Категорія]=" + "\"" + Category.Text + "\"" + "FROM [Номери] WHERE [Код]=" + IDRoom.Text;

OleDbDataAdapter dta = new OleDbDataAdapter(qry, connString);

DataSet ds = new DataSet();

dta.Fill(ds);

MessageBox.Show("Номер Видалено!");

var f2 = new MainWindow();

this.Close();

f2.Show();

}

private void button2\_Click(object sender, RoutedEventArgs e)

{

var f2 = new MainWindow();

this.Close();

f2.Show();

}

}

}

Клас archive (ArchiveWindow.xaml.cs)

public partial class ArchiveWindow : Window

{

public ArchiveWindow()

{

InitializeComponent();

}

private void IDa\_TextChanged(object sender, TextChangedEventArgs e)

{

}

private void button2\_Click(object sender, RoutedEventArgs e)

{

var f2 = new MainWindow();

this.Close();

f2.Show();

}

}

Клас Class1 (Class1.cs)

namespace HotelManagement

{

static class Class1

{

public static string adb = "";

}

}