Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Програмної інженерії
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

магістр
(назва освітнього ступеня)

на тему: Розробка автоматизованої системи обліку внутрішньопереміщених осіб з використанням MySQL, мова програмування C#

Виконав(ла): студент(ка) 6 курсу, групи СПм-61
спеціальності 121 Програмна інженерія
(шифр і назва спеціальності)

(підпис) Петрик О.В. (прізвище та ініціали)

Керівник (підпис) Стадник І.Я. (прізвище та ініціали)

Нормоконтроль (підпис) Стоянов Ю.М. (прізвище та ініціали)

Завідувач кафедри (підпис) Петрик М.Р. (прізвище та ініціали)

Рецензент (підпис) Луцик Н.С (прізвище та ініціали)

Тернопіль
2022

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет Комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра Програмної інженерії
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Петрик М.Р.
(підпис) (прізвище та ініціали)
« » 2022 р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня магістр
(назва освітнього ступеня)

за спеціальністю 121 Програмна інженерія
(шифр і назва спеціальності)

студенту Петрику Олегу Володимировичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка автоматизованої системи обліку внутрішньопереміщених осіб з використанням MySQL, мова програмування C#

Керівник роботи Стадник Ігор Ярославович, д.т.н., проф.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «12» грудня 2022 року № *4/7-1004*

2. Термін подання студентом завершеної роботи

3. Вихідні дані до роботи Вимоги замовника, технічні умови, технічне завдання

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ, 1 Аналітична частина, 1.1 Аналіз вимог до програмної системи, 1.2 Проектування програмної системи, 1.3 Конструювання програмної системи, 2. Тестування програмної системи, 3. Охорона праці та оцінка стійкості роботи об'єкту економіки до впливу поражаючих факторів ядерної зброї

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
Рисунки.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	Осухівська Галина Михайлівна	30.11.2022	
Безпека в надзвичайних ситуаціях	Клепчик Василь Михайлович		

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Отримання завдання	19.09.2022	
2	Аналіз завдання	27.09.2022	
3	Виконання розділу 1	15.11.2022	
4	Виконання розділу 2	20.11.2022	
5	Виконання розділу 3	25.11.2022	
6	Оформлення пояснювальної записки	28.11.2022	
7	Оформлення графічного та презентаційного матеріалу	29.11.2022	
8	Попередній захист	30.11.2022	
9	Нормоконтроль, рецензування		
10	Занесення диплома в електронний архів		
11	Допуск до захисту у зав. кафедри		

Студент _____ Петрик О.В.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Стадник І.Я.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Кваліфікаційна робота магістра містить: с. – , рис. – , табл. – , джер. – .

АВТОМАТИЗОВАНА СИСТЕМА ОБЛІКУ, ВНУТРІШНЬО ПЕРЕМІЩЕНІ ОСОБИ, SQL ТЕХНОЛОГІЇ, АРХІТЕКТУРА «КЛІЄНТ-СЕРВЕР»

Метою роботи є створення закінченого програмного продукту який може суттєво полегшити процес обліку внутрішньо переміщених осіб (мешканців готелю), реєстрації нових ВПО і ведення фінансового обліку.

Під час виконання роботи розроблено автоматизовану систему управління підприємством при застосуванні SQL технології та архітектури «клієнт-сервер» на прикладі готельного підприємства. Створено закінчений програмний продукт, який може суттєво полегшити процес обліку внутрішньо переміщених осіб (мешканців готелю) і ведення фінансового обліку.

В процесі розробки прийнято рішення використовувати реляційну СУБД MySQL, мову програмування C#, та засоби створення програмного інтерфейсу Windows Presentation Foundation.

Результатом цієї роботи є завершена програма яка володіє всіма функціями описаними в технічному завданні, а також (додатково) естетичним та зручним інтерфейсом. Додатковою функцією є відображення часу роботи у системі.

ABSTRACT

AUTOMATED ACCOUNTING SYSTEM, INTERNALLY DISPLACED PERSONS, SQL TECHNOLOGIES, CLIENT-SERVER ARCHITECTURE

The goal of the work is to create a finished software product that can significantly facilitate the process of accounting for internally displaced persons (residents of the hotel), registration of new IDPs, and financial accounting.

During the execution of the work, an automated enterprise management system was developed using SQL technology and "client-server" architecture on the example of a hotel enterprise. A finished software product has been created that can significantly facilitate the process of accounting for internally displaced persons (residents of the hotel) and financial accounting.

In the development process, it was decided to use the MySQL relational DBMS, the C# programming language, and the tools for creating the Windows Presentation Foundation software interface.

The result of this work is a completed program that has all the functions described in the technical task, as well as (in addition) an aesthetic and convenient interface. An additional function is the display of working time in the system.

ЗМІСТ

ВСТУП

Через війну понад сім мільйонів українців отримали статус внутрішньопереміщеної особи (ВПО), з них близько 1 млн. діти. З них близько 1.7 млн. для тимчасового проживання обрали західну частину України. Постає проблема оперативного створення прихистків для обліку та поселення ВПО. Прихистком можуть слугувати готелі, які знаходяться на мирній території.

Створення ефективної інформаційної готельної системи вимагає використання як нових, так і традиційних технологій. Це можна пояснити тим, що інформаційним готелям необхідно керувати як внутрішньою інформацією, зібраною співробітниками, так і будь-якими зовнішніми джерелами даних. Створення високоякісної системи також вимагає організаційних навичок обробки інформації, а не лише автоматизованих процедур.

КІС– це об’єкт якиї є багатоплановий , вінвключає складову апаратно-програмну , яка здійснює збір, передачу та опрацювання інформації, відповідну базу даних.Сучасні готелі потребують постійного обслуговування та розвитку. Їх потрібно автоматизувати, щоб належним чином функціонувати в економічному кліматі, що постраждав від війни. Готелі потребують досконалого управління, що потребує зусиль від розробника системи для створення ефективної автоматизованої системи управління всіма виробничо-технологічними процесами. Крім того, це потрібно зробити для належного управління ресурсами.

Сучасні готелі мають складну систему, яка постійно оновлюється. Кожна частина готелю працює разом з іншими частинами для досягнення максимальної продуктивності та функціональності. Щоб продовжувати працювати в умовах війни, керівникам готелів необхідно доопрацювати системи автоматизації всіх виробничих і технологічних процесів. Крім того, вони повинні керувати ресурсами.

Щоб належним чином вести зруйновану війною ринкову економіку, сучасні готелі потребують досконалої практики управління, заснованої на передовій технологічній автоматизації. Це пояснюється тим, що їх окремі складові — такі як сантехніка та технологічні процеси — постійно змінюються та взаємодіють один з

одним.

Інформаційна система управління, або МІС, — це набір програмного забезпечення, моделей і даних разом із спеціалістами та програмним забезпеченням, призначених для прийняття керуючих рішень та опрацювання інформації .

ISU має забезпечити буфер між фінансовими, матеріальними та кадровими ресурсами фірми. Він також повинен займатися плануванням стратегічних, тактичних і оперативних процесів. Крім того, він повинен виконувати завдання, пов'язані з оперативним обліком. Керівник використовує дані, зібрані в ході операцій, для розробки стратегії та балансування ресурсів. Після цього він вивчає результати минулих рішень і будує плани на майбутнє. Він також управляє витратами, керуючи використанням ресурсів і матеріалів, а також оцінюючи результати попередніх управлінських рішень.

Основними цілями ISU є підвищення продуктивності праці та зниження витрат шляхом надання точної фінансової інформації. Надаючи доступ до фінансової інформації, ISU сприяє тому, що менеджери мають найновішу інформацію про свій господарський об’єкт. Це призводить до більш обґрунтованих рішень і підвищення ефективності управління. Крім того, ISU покращує управління, надаючи менеджерам дані про всі частини компанії, а не лише про поточні тенденції. Це призводить до більш узгоджених рішень і швидшої реалізації нових планів. Нарешті, ISU надає менеджерам єдине джерело інформації, щоб вони могли приймати рішення швидше та точніше.Сучасні готельні комплекси практично в кожній країні є окремою галуззю економіки. Це зумовило підвищення вимог до готелів, а також високий розвиток матеріально-технічної інфраструктури, необхідної для їх функціонування. Разом із цією підвищеною увагою до управління та організації приходить увага до сучасних прогресивних методів ведення бізнесу. Використання комп’ютерів як у традиційних, так і в нових комп’ютерних технологіях полегшує управління. Це пов'язано зі створенням інформаційно-управлінських систем на базі комп'ютерних

технологій. Зараз багато готелів використовують цей підхід завдяки зростанню IT-індустрії.

На даний момент у світі існують різні способи ведення обліку і реєстрації мешканців готелю, це може бути від звичайного журналу, в якому записується інформація про мешканців, до використання сучасних комп'ютерних систем в яких інформація про мешканців міститься в базах даних на серверах підприємства а реєстрація можлива через мережу Інтернет. В першому випадку використання журналу є неефективним оскільки це забирає трохи часу і є вірогідність втрати даних при якихось пошкодженнях. В наступному випадку, використання такого способу, є набагато ефективнішим, надійнішим і має ряд інших суттєвих переваг. Проте недоліком такого способу є його вартість і потребує високо кваліфікованих працівників. Такі системи використовують переважно на великих готельних підприємствах.Даний програмний продукт може посісти середнє місце між цими двома способами. З однієї сторони дані зберігаються в базі даних, що є набагато надійнішим ніж паперовий носій, з іншої вона не потребує додаткових витрат зі сторони підприємства і висококваліфікованих (як другий варіант), достатньо встановити програму на комп'ютер.

Отже, метою роботи є створення закінченого програмного продукту який може суттєво полегшити процес обліку ВПО – мешканців готелю, реєстрації нових ВПО і ведення фінансового обліку.

Розробка програмного продукту проводилась у середовищі розробки Microsoft Visual Studio 2010 на мові програмування C#. У якості СУБД було обрано MySQL через її переваги:

- легкість у встановлені та користуванні;
- велика кількість активних процесів у один час
- кількість колонок у таблицях може сягати 50 млн.;
- велика швидкість;
- ефективна та проста система захисту даних
- найкраща швидкість обробки даних на обсязі до 500000 записів;

безкоштовні відкриті ліцензії.

РОЗДІЛ 1
АНАЛІТИЧНА ЧАСТИНА

1.1 Аналіз вимог до програмної системи

Аналіз вимог сервісу забезпечує загальний огляд послуг, які майбутня система повинна надавати своїм клієнтам (користувачам). Така презентація є основою для аналізу метою якого є подальше з’ясування проблеми, яку буде вирішувати запропонована система. Оскільки обрана мною архітектура є об’єктно-орієнтованою, результатом є набір об’єктів, які взаємодіють, щоб визначити функціональність системи. Визначений набір створюється шляхом поступового розбиття кожного сценарію на об’єкти, які представляють дії сценарію.

1.1.1 Аналіз предметної області

Готельний бізнес є грунтовною частиною індустрії, щоб забезпечити тимчасове житло для переміщених осіб. Це неможливо без добре спланованих систем, які підтримують підприємства з високою економічною ефективністю. Для різних готельних комплексів можуть бути створені різноманітні системи готельної автоматизації. Ці системи містять програми, які автоматизують різноманітні послуги в готелі, такі як інженерні, кейтерингові та комерційні, а також керування номерами. Програми централізованого управління можна додати до будь-якої готельної системи, щоб спростити координацію між усіма службами готелю, не витрачаючи час і ресурси [2].

При виборі системи автоматизованого управління готелями важливо враховувати кілька професійних розробок.
У найбільших готелях України використовуються комп’ютерні системи, які

вважаються світовими. Ці системи використовуються багатьма готельними мережами.

Найбільш часто створюваним продуктом є комплекс Micros-Fidelio, спеціальність якого є створенні систем управління для ресторанів і готелів. Понад 20 років це універсальний продукт, який використовують такі великі готельні мережі, як Hilton, Marriott, Sheraton і Kempinski. [3].

Fidelo Front Office — це автоматизована система управління основними процесами готелю: бронювання номерів, відстеження бронювання, облік перебування, виселення гостей. Крім того, Fidelo Front Office може керувати інвентаризацією номерів, фінансами та управлінням пансіонатом. Перевагами є те, що ця система проста у використанні, має зручний інтерфейс користувача, забезпечує високий рівень безпеки та містить чітке розмежування доступу користувачів. Його також можна використовувати в готелях, які належать до мереж або не пов'язані з мережами, іноді навіть у пансіонатах, які використовують інші методи роботи.[2].

У 1960-х роках дві авіакомпанії почали використовувати комп'ютерні системи для планування рейсів. Ці системи були створені самими авіакомпаніями і призначені лише для використання їх співробітниками. Коли більше пасажирів стало літати, комп'ютери авіакомпаній перевантажилися. Це спонукало їх до створення автоматизованих CSR для обробки запитів на планування. З часом ця потреба спонукала інші організації до створення подібних систем. Авіакомпанії працювали разом, щоб просувати KSB у майбутньому. Для цього вони розробили маркетинговий план, запустивши Глобальну систему розподілу, або GDS. Ця система об'єднує всі бронювання авіакомпаній по всьому світу. GDS допомагають підприємствам туристичної індустрії, надаючи їм дистриб'юторські мережі. [3] GDS призначені для задоволення потреб цих підприємств, які включають збір інформації та виконання завдань, а саме:

- призначений для забезпечення постачальників туристичних послуг (авіаперевезень, оренди автомобілів, готелів, туроператорів, залізниць тощо) надійною та ефективною глобальною мережею продажів, яка забезпечує їм

доступ до найбільшої клієнтської бази, яка складається з туристичних агентств. і комерційні офіси авіакомпаній.

- пропонувати абонентам послуги які надають можливість керувати своїми підписками та провайдерами, мати доступ до вітчизняних і міжнародних провайдерів, керувати їхньою діяльністю та вести фінансовий облік своєї роботи.
- надавати авіакомпаніям можливість використовувати систему у своїх комерційних офісах таким чином, щоб і авіакомпанії, і туристичні агенції могли використовувати систему одночасно.
- надавати глобальні послуги через технічні зв’язки та альянси з іншими системами.

Сьогодні існують численні програми, які сприяють розповсюдженню туристичних послуг через KSB [4].

KSB надає туристичним агентствам спеціальні пакети, що включають ці програми і, звичайно, спеціальне обладнання для проведення заходів. Очевидно, що всю діяльність з бронювання сприяють туристичні агентства за допомогою зв'язку з певними організаціями, які займаються цією сферою. Обладнання та методи зв'язку залежать від обраного туристичною агенцією способу підключення:

- підключення виділеним каналом зв’язку (прямий дріт або не комутований канал зв’язку) – пряме підключення;
- підключення з набором телефонного номера (комутований канал зв’язку) – телефонне підключення;
- підключення через мережу Internet.

Функції агентства при самостійному бронюванні клієнтами через Internet полягають в перевірці коректності виконаних бронювань, оптимізації вибраних маршрутів, виписці і доставці авіаквитків.

Отже, кожна КСБ створює свої власні комп’ютерні програми, які виконують наступні функції:

- дозволяють проводити пошук оптимальних цінових варіантів туристичних послуг;

• дозволяють працювати за різними тарифами, у тому числі, за конфіденційними;

• дозволяють туристичному агентству автоматично роздруковувати на принтері квитки, призначені для ручної виписки, використовуючи дані про бронювання;

• дозволяють туристичним агентствам використовувати програму автоматичного складання звітів;

• дозволяють вести різні види статистики;

• надають доступ до великої довідкової системи оперативної інформації;

• дозволяють контролювати усі операції за взаєморозрахунками з клієнтами і постачальниками;

• дозволяють створювати на Web-сайті туристичного агентства систему бронювання туристичних послуг для корпоративних і приватних клієнтів. Таким чином, відвідувач Web-сайту отримує можливість самостійного вибору авіарейсів і готелів і їх подальшого бронювання.

Найпопулярнішими системами бронювання в індустрії міжнародного туризму є Amadeus, Galileo, Sabre і Worldspan.
Ці системи мають близькр 500тис терміналів у різних готелях навколо світу, що становить 90% ринку, отож не дивнр, що вони мають назву «золотою четвіркою». Відсоток громадської землі, яка зарезервована для корінних народів, становить 10% і складається з систем резервації, які все ще знаходяться в процесі злиття з одною з вищезазначених систем [4].

Кожна GDS, хоча й глобальна, має власну сферу впливу. Для Amadeus основним фокусом є Європа. Система Galileo використовується в 116 країнах, понад 45 000 організацій, які надають можливий доступ до інформаційних ресурсів. Він поширений в США і таких країнах, як Великобританія, Італія, Греція, країни Бенілюксу, Швейцарія, Португалія. Загалом частка цієї системи на європейському ринку становить 29,8% (друге місце). Системи Sabre і Worldspan мають меншу площу [5].

Для готелів, які є афілійованими мережами готелів, такими як Intercontinental, Radisson та іншими, процес бронювання через GDS виконується централізовано та є частиною основної стратегії маркетингу готельної мережі. Представлення готелю в будь-якій GDS повинно містити наступні пункти: опис готелю, опис фонду номерів, опис тарифів, цін, інформацію про вільні місця.

На українському ринку послуг активно працює лише система Amadeus в цілому але кількість українських компаній, яка її використовує є невеликою. Це програмне забезпечення, призначене для використання в авіакомпаніях, автомобілях, готелях, сфері послуг.

Основними перевагами системи - це економія часу , адже є можливість отримати повну інформацію про туристів, клієнтів і агентства; в систему постійно завантажується нова інформація; підвищення продуктивності підвищує доступність інформації.

Незважаючи на численні переваги системи Amadeus, подальше використання інших систем в Україні дасть наступні позитивні результати:

• український туристичний сектор; система Worldspan не вимагає чітко визначеної кількості бронювань, оскільки є досить демократичною системою;

• Використання системи Sabre в українських туристичних компаніях дозволить реалізувати систему яка може отримати доступ до широкого спектру туристичних послуг, продавати туристичний продукт, та розповсюджувати технологічні рішення

• Система Galileo забезпечить туристичні компанії версією системи бронювання для Windows, а програма Premier дозволить агентствам повністю автоматизувати роботу з обслуговування клієнтів. [7].

Водночас широкому впровадженню комп’ютеризованих систем бронювання в Україні перешкоджають такі фактори:

• недостатністю фінансових ресурсів для плати за підключення до GDS і установку необхідного програмного забезпечення;

• низький рівень технічної підготовки менеджерів;

• відсутністю впровадження комп’ютерних систем у туристичних фірм;

• інформаційною небезпекою при широкому використанні Internet-технологій: комерційною (шахрайство і несанкціонований доступ); технічною (комп'ютерні віруси і атаки «хакерів»). Незважаючи на спеціальні апаратні та програмні системи захисту, обмеження інформації, що передається з публічних каналів зв'язку, Із застосуванням різноманітних заходів адміністративного контролю проблеми інформаційної безпеки все ще існують і мають вирішуватися в масштабах країни. Потрібні закони, органи контролю і пряма державна підтримка вітчиз-няних виробників засобами забезпечення безпеки;

• відсутністю єдиних стандартів із застосування Internet і інших інформаційних технологій.

Люди, які бронюють готелі онлайн через GDS, зазвичай отримують доступ до інформації та бронювання через власний Інтернет-сервер готелю. Таким чином, люди з різних країн світу — у тому числі міжнародні ділові мандрівники, відпочиваючі та будь-які інші гості з доступом до Інтернету — можуть використовувати свої комп'ютери для планування свого перебування заздалегідь. Готелі мають власні Інтернет-сервери, щоб вони могли забезпечити бронювання та доступ до інформації через GDS. Таким чином, люди, які не знаходяться в країні походження готелю, можуть використовувати свої персональні комп'ютери, щоб забронювати номер перед бронюванням поїздки чи відпустки. Крім того, це дозволяє людям з різних країн світу використовувати свій доступ до Інтернету для бронювання та пошуку перед поїздкою.

Оскільки це стає все більш і більш популярним, готелі розробляють свої Web-застосунки все більш дружніми та інтуїтивно-зрозумілими для користувачів інтерфейсами. Іноді такі готелі припускаються серйозної помилки, забуваючи про те, що Web-сайт – візитна картка готелю в Internet – і не приділять належної уваги його дизайну та підтримці Web-сайту в робочому стані цілодобово, що впливає на імідж готелю.

При бронюванні номерів у готелях відвідувачеві Web-сайту – потенційному клієнтові надається загальна інформація про готель, Загальна інформація про готель, фотографії різних типів номерів, опис додаткових послуг (харчування,

конференц-зали, сауни, спортивно-розважальні центри), а також перелік культурно-розважальних закладів у безпосередній близькості від готелю, та які готель пропонує пропозиції.

Для цього використовуються спеціальні програми бронювання. Так, одна із найбільш успішних програм – програма Horse – 21 (абревіа-тура назви Hotel Reservation Service, що належить голландській компанії Hors BV) має базу даних, яка містить інформацію про більш ніж 240 тисяч готелів. У порівнянні з аналогічними програмами має такі переваги: реальний онлайн (підтвердження броні за лічені секунди); централізована система отримання комісій; унікальна база цих готелів; може працювати без абонентської плати, норм поділу, депозитів; легке використання - Без навчання, необхідногос спеціального програмного забезпечення, обладнання.

Окрім систем комп'ютерного бронювання послуг в готельному бізнесі, широко використовуються програми для забезпечення управління різними службами готелю.

Так, система Lodging Touch американської компанії MAI Hospita-lity, один з лідерів у розробці програм для готельного господарства, це інтегрований пакет, який автоматизує різні підрозділи готелю: консьєрж, ресторанне обслуговування, продажі, організацію банкетів. Його можна інтегрувати з периферійними системами, такими як бухгалтерські програми, системи оптимізації прибутку готелів, платне телебачення, телефонні системи, контроль доступу в номери тощо.Програма Lodging Touch – Портьє виконує функції бронювання номерів , заселення і виписку гостей, нарахування платежів і процедуру нічного аудиту. Тут можна проглянути звіт за станом готелю, отримати різну довідкову інформацію (гостьові картки, паспортні дані).

До інших програмних модулів системи належить блок управління тарифами, блок управління номерним фондом (для складання графіку прибирання номерів і роботи покоївок). Варто зазначити, що в системі Lodging Touch є програмний блок для співпраці з турагентствами, який дозволяє обробляти заявки

від турфірм, розраховувати комісійні, визначати знижки та спеціальні пропозиції, переглядати статистику. [10].

Система Nimeta - відносно новий продукт, який розроблявся для готелів малих та середніх розмірів. Робота системи Nimeta передбачає можливість оренди та використання системи управління за допомогою Internet-технологій. Ця технологія включає дані готелю і, власне, саму програму системи управління готелем, розташовану на спеціально відведених серверах за межами готелю. Такі сервери знаходяться в сервісних центрах, обладнаних інтернет-провайдерами. Тому все, що потрібно для підключення та користування системою «Німета» – мати комп'ютер та доступ до мережі Інтернет.

Комплекс програмних продуктів Epitome Enterprise розроблено компанією HIS Corporation (США), одним зі світових лідерів у сфері інформаційних технологій для індустрії гостинності.

Використання вказаних систем дозволяє обмінюватися управлінською і фінансовою інформацією, автоматизувати процес виконання щоденним и завданнями обслуговуючого персоналу та адміністрації готелю. При цьому можна отримати хороший взаємозв'язок між різноманітними службами готелю, а це, в свою чергу, підвищує ефективність та дає змогу позбутися помилок. Разом з цим, керівництво готелю отримує потужний інструмент контролю за станом готелю і фінансовими потоками.

Розглянуті можливості Internet для готелів не обмежуються лише роботою з GDS. Основними перевагами для готелю є широке охоплення аудиторії (більше 100 млн. чоловік у світі) і невисокі накладні витрати, в порівнянні з GDS. Отже, Internet – реальний конкурент КСБ, оскільки робота через Internet може виконуватися на будь-яких серверах, будь-яку кількістю разів, будь-якою приватною особою, що прагне заощадити на готелі, тоді як з GDS працюють тільки професіонали . А перевага GDS полягає в тому, що будь-який готель, завантаживши інформацію про себе, може бути упевнений, що цю інформацію побачить кожен туристичний агент, а для розміщення даних про готель в Internet

потрібна клопітка робота з занесення їх в різні пошукові системи, довідники і каталоги.

У нашій країні впровадження комп'ютеризованих систем управління готелями (Property Management System/PMS) за західними технологіями почалося з появою висококласних готелів, багато з яких працюють за корпоративними правилами, що передбачають «лояльність» до тієї чи іншої системи.

Найбільш поширена система обслуговування в нашій країні – система Fidelio, вона повністю русифікована і адаптована для російського та українського ринків. Як правило, вона встановлюється готелями спільно з популярним розрахунково-касовим комплексом Micros. Крім того, можуть бути встановлені програми для автоматизації фінансово-господарської діяльності, для організації відділу продажів і маркетингу, для головного інженера. Усі програми працюють як єдина інтегрована система в операційному середовищі DOS.

В готельному бізнесі України, окрім переваг використання інформаційних технологій, виникає безліч проблем, обумовлених вимогами, що висуваються до умов експлуатації і грамотного їх застосування, а також слабким розвитком українського ринку.

Основна проблема при встановлені програмного забезпечення, окрім технічних питань, є комп'ютерна неграмотність і недостатня кваліфікованість персоналу. Переважна кількість працівників готелів має гуманітарну освіту, що викликає певні труднощі при роботі з комп'ютером і Internet. Перенавчання співробітників або найманих працівників вимагають додаткових фінансових коштів. Далеко не усі готельні підприємства можуть дозволити собі утримувати в штаті IT – спеціалістів. Часто буває так, що програмне забезпечення зарубіжних виробників не адаптоване для Українського ринку.

Перешкодою до розвитку систем on-line продаж є незначна кількість користувачів Internet, низький рівень життя населення, низький ступінь розвитку платіжних систем, механізмів оплати за банківськими чеками і кредитними картками. І, попри активний розвиток інструментів безготівкової оплати приватних осіб, туристичні Internet -компанії не поспішають впроваджувати

платіжні технології тому, що доки ще є можливість понести занадто високі витрати: зокрема, кредитні картки має в розпорядженні досі невеликий відсоток українських туристів. Крім того, українські туристи вважають за краще заплатити “живому” менеджерові , а не серверу, і при цьому бути переконаним в тому, що їх не обмануть.

Проведений аналіз сучасного стану використання інформаційних систем і технологій в індустрії туризму та готельного господарства показав, що:

• в Україні відсутні вітчизняні розробки інформаційних систем управління готелями та комп’ютерних систем бронювання, що обумовлюється слабким розвитком українського ринку інформаційних технологій;

• найбільш поширеною системою обслуговування в нашій країні є система Fidelio, яка повністю русифікована і адаптована для російського та українського ринків і використовується у висококласних готелях, кількість яких є незначною;

• вихід на український ринок туристичних послуг глобальної системи Amadeus, послугами якої сьогодні користується невелика кількістю туристичних фірм України, не вирішує проблеми розвитку індустрії туризму і вимагає використання також послуг таких відомих систем, як Sabre та Worldspan;

• перспективним для готельних мереж, а також незалежних готелів і пансіонатів, які задіяні в сфері зеленого туризму, є створення власних Internetсерверів, через які здійснюється доступ до інформації і бронювання в GDS, і своїх web-сайтів та використання однієї із розглянутих вище програми для бронювання місць і забезпечення управління різними службами готелю;

• широке впровадження відмічених вище систем на базі західних технологій та програмних продуктів вимагає комп’ютерної підготовки і відповідної кваліфікації обслуговуючого їх персоналу, а також адаптації окремих програмних продуктів до українського ринку послуг.

Враховуючи вищесказане визначено предметну область дипломного проекту – готельне підприємство, в якому ведеться облік мешканців, реєстрація нових мешканців, а також ведеться фінансовий облік.

В готельних підприємствах існують наступні проблеми:

- проблема обліку мешканців готелю;
- проблема ведення фінансового обліку.

Підприємству потрібна інформаційна система яка надасть наступні переваги:

- полегший облік мешканців готелю;
- дозволить бронювання номерів в готелі;
- дозволить виводити інформацію про номери готелю;
- дозволить вводити і виводити інформації про доходи підприємства;
- дозволить вводити і виводити інформацію про витрати підприємства;
- дозволить вивести фінансовий звіт на основі витрат і доходів підприємства.

В предметній області існують наступні об’єкти: клієнт; кімната; бронь.

Клієнт – об’єкт, який містить інформації про мешканців готельного підприємства, його використовують при бронювання номера.

Кімната – об’єкт, який мітить інформацію про номери готелю і використовується для бронювання номерів.

Бронь – об’єкт, який містить інформацію про заброньовані номери готелю.

Атрибути якими володіють ці об’єкти.

Клієнт:

- ім’я;
- прізвище;
- номер паспорта;
- номер телефону.

Кімната:

- номер кімнати;
- кількість ліжок;
- кількість кімнат;
- орендна плата;

- статус.

Бронь:

- номер кімнати;
- ім’я мешканця;
- дата поселення;
- кількість ночей;
- статус оплати.

В результаті аналізу предметної області було з’ясовано що всі функції, які потрібні в цій інформаційній системи можуть бути реалізовані в кінцевому програмному продукті. А атрибути об’єктів будуть базовою інформацією яка буде занесена у базу даних.

1.1.2 Постановка задачі

Отже, потрібно розробити інформаційну систему для готельного підприємства, яка буде володіти всіма необхідними властивостями, поселення ВПО, введення доходів і витрат.

Перелік завдань вирішення яких потрібне для реалізації системи:

- обрати СУБД;
- обрати мову розробки;
- обрати середовище розробки;
- обрати додаткові засоби розробки, якщо потрібно;
- спроектувати та реалізувати базу даних;
- спроектувати та реалізувати програмний продукт;

В базі даних повинна міститися наступна інформація:

Про ВПО:

- ім’я;
- прізвище;
- номер паспорта;
- номер телефону.

Про номер:

- номер кімнати;
- кількість ліжок;
- кількість кімнат;
- орендна плата;
- стан номера.

Про доходи:

- номер кімнати;
- кількість ночей;
- сума оплати;
- дата оплати.

Про витрати:

- призначення;
- сума витрат;
- дата оплати;
- отримувач.

Про бронь:

- номер кімнати;
- номер ВПО;
- дата поселення;
- кількість ночей;
- оплачено.

Система має підтримувати наступні функції:

- виведення інформації про номери;
- виведення інформації про ВПО;
- виселення ВПО;
- поселення нового ВПО;
- бронювання номерів готелю;
- оплата броні;
- введення інформації про доходи підприємства;

- виведення інформації про доходи підприємства;
- введення інформації про витрати підприємства;
- виведення інформації про витрати підприємства;
- оформлення фінансового звіту на основі доходів і витрат;
- візуальне зображення стану підлеглого (його зайнятість на даний момент часу);
- можливість додавання нових користувачів (для адміністратора).

1.1.3 Пошук актантів та варіантів використання

Проаналізувавши постановку задачі для розроблювальної системи, можна виділити три основні типи користувачів (актантів) які будуть нею користуватись:

– адміністратор – користувач, який володіє найбільшими правами в системі, має можливість створювати, редагувати, видаляти інформацію про працівника. Крім того він може переглядати створені документи, та змінювати їх статус;

– неавторизований користувач – звичайний користувач системи, може переглядати дані про працініків та інформацію про них. Також може переглядати інформацію про підприємство;

– авторизований користувач – має такі самі можливості як і неавторизований користувач.

Для кращого розуміння того, якою функціональністю наділений кожен актант, було створено діаграму варіантів використання, з описом ключових варіантів.

Суть діаграми полягає в наступному: система проектується у вигляді багатьох сутностей або акторів, які взаємодіють із системою за допомогою так званих варіантів використання. Варіанти використання використовуються для опису послуг, які система надає учасникам. Іншими словами, кожен варіант використання визначає набір дій, які система виконує під час діалогу з актором. При цьому нічого не сказано про те, як буде реалізована взаємодія актора з системою.

У мові UML існує декілька стандартних типів зв’язків між акторами та варіантами використання:

– асоціації;

– включення;

– розширення;

– узагальнення.

У той же час, загальні властивості варіантів використання можна виразити трьома різними способами, а саме за допомогою відносин утримання, відносин розширення та відносин узагальнення

1.1.4 Опис ключових варіантів використання

Метою побудови діаграми варіантів використання є виявлення дійових осіб - для кого призначене ПЗ, і які дії воно повинне виконувати. Розробка діаграми робиться в три етапи - виявлення дійових осіб – акторів, виявлення варіантів використання системи і побудова діаграми.

Акторами вважаються зовнішні чинники системи, поведінка яких має невизначений характер. Таким чином, підкреслюється відмінність між користувачем як конкретною особою та актором (роллю, яку будь-хто може виконувати в системі).

Якщо інша програмна система ініціює виконання якоїсь роботи цієї системи, вона може виступати в ролі учасника, тобто учасник є абстракцією зовнішнього об'єкта, екземпляром якого може бути людина або зовнішня система.

Актори в моделі представлені класами, а користувачі представлені екземплярами класів. Людина може бути екземпляром кількох дійових осіб (наприклад, водія та касира), але ми враховуємо лише ролі та сценарії, у яких вона бере участь у вимогах.

Екземпляр рольового гравця або актора - ініціює послідовність дій (транзакцій) у системі, ми називаємо це сценарієм. Екземпляр цього сценарію починається, коли користувач, який є екземпляром актора, вводить стимул, що призводить до виконання послідовності дій, що відповідають транзакції, і

завершується, коли екземпляр сценарію знову очікує вхідного стимулу від актора екземпляр.

Екземпляр сценарію існує під час виконання. Його можна розглядати як екземпляр класу, опис якого є описом транзакції.

Для акторів визначені такі правила:

1) кожен сценарій може запускати тільки один актор;
2) кожен актор може запускати кілька сценаріїв;
3) Взаємодія акторів на користь системи (тобто як невід'ємна частина її функціональності) дозволена лише за сценаріями, передбаченими для цього;
4) актор не визначає сценарію, він лише ініціює ланцюжок подій, який визначить сценарій;
5) Для кожного актора визначається його інтерфейс до сценарію, який він виконує (неформально).

Сценарій — це повний потік подій у системі, очевидно, із станом і поведінкою. Кожна взаємодія між актором і системою вважається новою сценою. Сценарій можна розглядати як об'єкт. Якщо багато системних сценаріїв виконуються з подібною поведінкою, ми можемо розглядати їх як клас сценаріїв. Виклики сценаріїв є підкласами екземплярів класу. Отже, сценарії — це транзакції з внутрішнім станом. Вони описані детально - вони необхідні для ідентифікації дійсних системних об'єктів.

Модель системи керується сценаріями, і її потрібно змінити шляхом заміни необхідних акторів і сценаріїв, які вони виконують. Така модель відображає побажання користувачів і легко змінюється відповідно до їх бажання. Користувач добре розуміє це і може вирішити свою проблему до початку проектування.

Зокрема, розробляється скриптовий інтерфейс – можна моделювати виконання скриптів за допомогою прототипів.

Впровадження Actor дозволяє відповісти на питання: для чого потрібна система? Хто його основні користувачі? Актор визначає зовнішнє середовище системи, а її внутрішню сутність визначає сцена.

Сценарій — це повний ланцюжок подій, ініційованих актором, і специфікація реакції на ці події. Набір сценаріїв визначає всі можливі способи використання системи. У кожного актора свій набір сцен.

Це може підкреслити основну мету події, яка має вирішальне значення для сцени та її розуміння, а також інші цілі, зокрема у випадках помилки користувача, наприклад.

Розглядаючи окремі сценарії, ми розділяємо функціональність системи на незалежні компоненти, які можуть оброблятися паралельно.

Для моделей сцен пропонується спеціальна графічна нотація, основні правила якої такі:

- Актори позначені значком персонажа з його ім'ям під ним;

- Сценарії представлені еліпсами з їх назвою (як правило, відображають компоненти мети, досягнутої сценарієм);

- Значки акторів з’єднані стрілками з кожним сценарієм, запущеним відповідним актором.

1.1.5 Виявлення основних сутностей предметної області

На основі аналізу предметної області були отримані наступні основні сутності та відносини між ними які можна побачити на Рисунку 1.1:

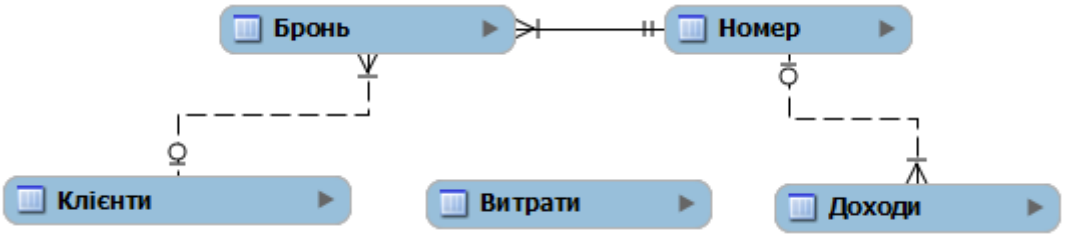

Рисунок 1.1 – Відносини між сутностями

Клієнти містить наступні атрибути:

- ім’я
- прізвище
- номер паспорта
- номер телефону

Бронь містить наступні атрибути:

- номер кімнати
- номер мешканця
- дата поселення
- кількість ночей
- оплата

Номер містить наступні атрибути:

- кількість ліжок
- кількість кімнат
- орендна плата
- статус

Доходи містить наступні атрибути:

- номер кімнати
- кількість ночей
- сума оплати
- дата оплати

Витрати містить наступні атрибути:

- призначення
- сума
- дата оплати
- отримувач

Бронь містить наступні відносини:

- атрибут номер кімнати є посиланням на сутність Номер
- атрибут номер мешканця є посиланням на сутність Клієнти

Доходи містить наступні відносини:

- атрибут номер кімнати є посиланням на сутність Номер

Номер містить наступні відносини:

- атрибут номер кімнати є посиланням на сутність Доходи
- атрибут номер кімнати є посиланням на сутність Бронь

Клієнт містить наступні відносини:

- атрибут id є посиланням на сутність Бронь

1.1.6 Побудова схеми реляційної бази даних

На підставі виявлених сутностей, їх атрибутів та взаємозв'язків була побудована наступна схема реляційної бази даних.

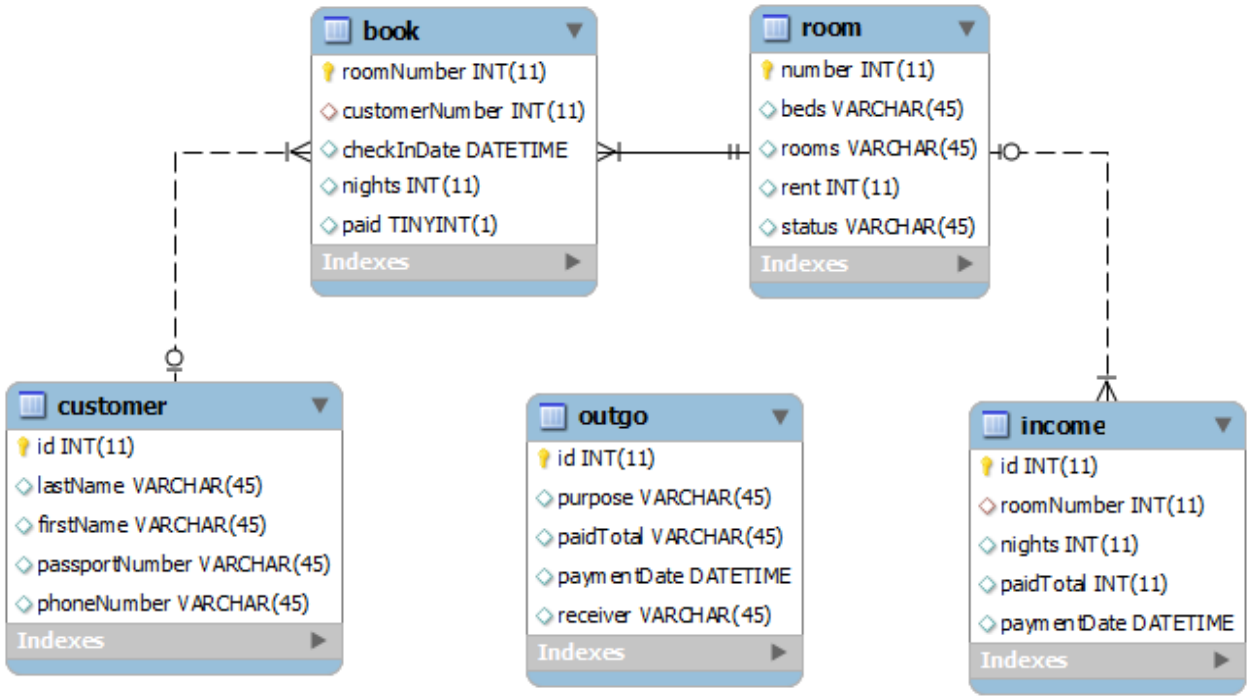

Рисунок 1.2 - Схема реляційної бази даних

Нормалізація бази даних:

- Перша нормальна форма

Перша нормальна форма утворює ґрунт для структурованої схеми баз даних.

Вимоги до 1НФ:

- Кожна таблиця повинна мати первинний ключ: найменший набір стовпців, який ідентифікує запис.
- Уникайте повторюваних груп (категорій даних, які різночасно з'являються в різних записах), правильно визначаючи неключові атрибути.
- Атомарність: кожен атрибут повинен мати лише одне значення, а не множину значень.

Отже можна зробити висновок що наша база даних є в першій нормальній формі.

- Друга нормальна форма

Щоб перебувати в другій нормальній формі, таблиця в першій нормальній формі повинна відповідати іншим умовам.

Тобто: таблиця 1NF знаходиться в 2NF тоді і тільки тоді, коли для будь-якого потенційного ключа K і будь-якого атрибута A, який не є потенційним ключем, A залежить від усього потенційного ключа, а не від його частини.

Трошки більш формально: 1NF таблиця знаходиться в 2НФ тоді і тільки тоді, коли всі її неключові атрибути функціонально залежні від цілих потенційних ключів.

Зауважимо, що коли 1НФ таблиця не має складних потенційних ключів (таких, що складаються більш ніж з одного атрибута), тоді таблиця автоматично знаходиться в 2НФ.

Таким чином, оскільки БД знаходиться в 1 НФ і в ній немає складних потенційних ключів, то БД автоматично знаходиться в 2 НФ.

- Третя нормальна форма

Таблиця знаходиться в 3НФ тоді і тільки тоді, коли виконуються наступні умови:

- відношення r (таблиця) знаходиться в 2нф
- кожен неключовий атрибут відношення r нетранзитивно залежить (тобто залежить безпосередньо) від кожного потенційного ключа в r.
- неключовий атрибут — атрибут, що не є частиною будь-якого потенційного ключа.

Транзитивною називають таку функціональну залежність, в якій $X \to Z$ (X визначає Z) непрямо, а через $X \to Y$ і $Y \to Z$ (і невірно, що $Y \to X$).

3НФ було дане Біллом Кентом: кожен неключовий атрибут «має надавати факт про ключ, цілий ключ, і ні про що окрім ключа.»

Таким чином, оскільки БД знаходиться в 2 НФ і в ній кожен неключовий атрибут нетранзитивно залежить від кожного потенційного ключа, то БД знаходиться в 3 НФ

1.2. Проектування програмної системи

1.2.1 Побудова UML-діаграми класів

Опишіть поведінку системи, що розробляється, використовуючи функціональну модель, яка відображає системні прецеденти, системне середовище (активні люди або актори) і зв'язки між прецедентами та акторами (прецедентні діаграми). Основне завдання прецедентної моделі - бути загальним інструментом, який дозволяє експертам предметної області та розробникам - експертам з інформаційних технологій спільно обговорювати функціональні можливості та поведінку системи.

Актори – це користувачі, або системи, чия взаємодія з системою є унікальною.

В інформаційній системі, яка розробляється, такі сутності є кандидатами на роль актора:

Адміністратори - це користувачі, які конфігурують і налаштовують систему;

Оператор - користувач, який коригує та налаштовує курс;

Читачі (або клієнти) - це користувачі, які отримують інформацію про книги та авторів, доступну в системі.

Змоделюйте діалог між актором і системою за допомогою прецеденту. Прецеденти визначають можливості, які система надає учасникам. Набір усіх прецедентів для системи визначає всі способи її використання. Ми можемо сказати, що прецедент – це серія транзакцій, які виконує система, які призводять до певних результатів для конкретних учасників.У системі, що розробляється, повинні забезпечуватися наступні потреби:

– актор Адміністратор реєструється в системі, як користувач, що володіє відповідними правами доступу до даних;

– актор Адміністратор конфігурує систему;

– актор Оператор реєструється в системі, як користувач, що володіє відповідними правами доступу до даних;

– актор Оператор вводить і редагує дані в базі даних системи;

– актор Клієнт реєструється в системі, як користувач, що володіє відповідними правами доступу до даних;

– актор Клієнт виконує пошук необхідної йому інформації.

На підставі перерахованих потреб можна виділити наступні прецеденти:

реєстрація в системі;

конфігурація системи;

введення і редагування даних;

пошук даних.

Діаграма прецедентів — це графічне представлення всіх або деяких акторів, прецедентів та їх взаємодії в системі. Кожна система зазвичай має головну діаграму варіантів використання, яка представляє межі системи (актори) і основну функціональну поведінку системи (варіанти використання).

На цьому етапі життєвого циклу також можна побудувати графіки дій. Вони відображають динаміку проекту і є графіками потоку управління від однієї дії до іншої в системі, а також паралельних дій і чергування потоків управління. Оператор повинен ввести пароль. Якщо пароль неправильний, редагування буде неможливим. Якщо пароль правильний, оператор вводить усі необхідні дані.

Діаграми класів дозволяють створити логічне представлення вашої системи. Піктограми діаграм дозволяють відображати складні ієрархії об’єктів, зв’язки між класами та інтерфейсами.

Нею легко керувати, якщо у вашій системі мало класів. Однак для системи, що складається з великої кількості класів, необхідний механізм для їх групування та полегшення керування та повторного використання. Тут стане в нагоді концепція пакетів. Пакет у логічному представленні моделі – це набір класів та інших пакетів.

1.2.2 Загальний опис програмного продукту

Даний програмний продукт створювався в середовищі розробки Microsoft Visual Studio 2010 на мові програмування C#.

У якості СУБД було обрано MySQL через її переваги:

- простота у встановленні та використанні;
- підтримується необмежена кількість користувачів, що одночасно працюють із БД;
- кількість рядків у таблицях може досягати 50 млн.;
- висока швидкість виконання команд;
- наявність простої і ефективної системи безпеки;
- найкраща швидкість обробки даних на обсязі до 500000 записів;
- безкоштовні відкриті ліцензії.

Для нормального функціонування програми потрібен компютер за наступними характеристиками: процесор з тактовою частотою 1 GHz; оперативна пам‘ять об’ємом 256 MB; 5 MB вільного простору на жорсткому диску для самої програми, а також необхідний мінімум для нормального функціонування операційної системи; відеоадаптер з 64 MB пам‘яті.

1.2.3 Моделювання архітектури системи

Архітектура програмної системи є абсолютно прямолінійною, що видно з діаграми:

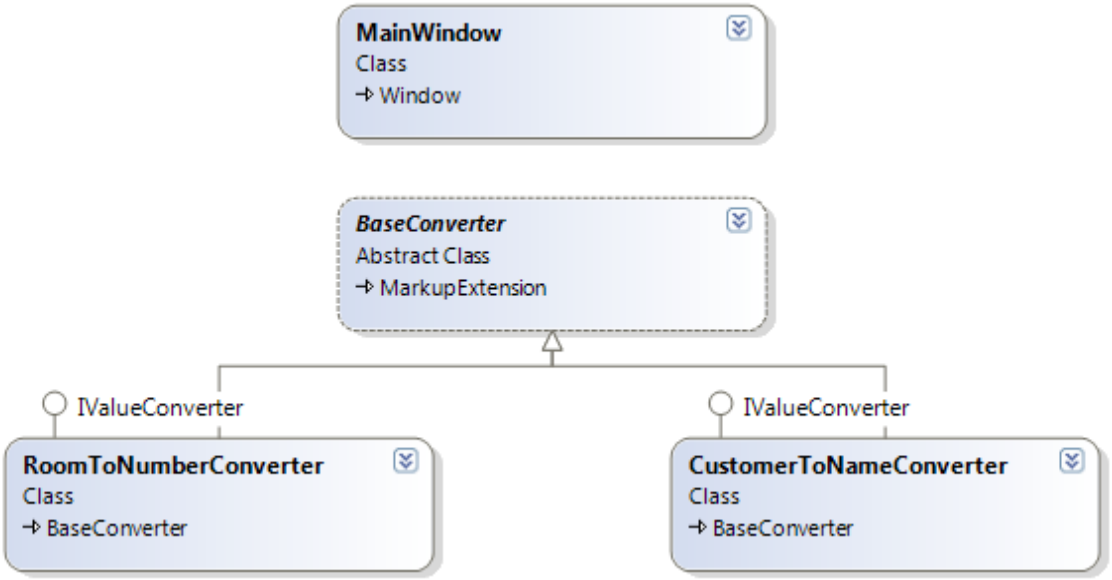

Рисунок 1.3 – Діаграма класів

Клас-вікно MainWindow - головне вікно програми.

Класи-конвертери:

- RoomToNumberConverter – перетворює номер в номер кімнати.
- CustomerToNameConverter – перетворює клієнт в ім’я і фамілія мешканця.

Ці класи призначені для коректного відображення інформації з бази даних на користувацький інтерфейс.

MainWindow:

Поля:

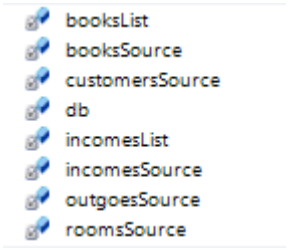

Рисунок 1.4 – Поля у MainWindow

Методи:

private void saveButton_Click(object sender, RoutedEventArgs e) – зберігає всі зміни до бази даних.

private void addBook_Click(object sender, RoutedEventArgs e) – додати в базу даних нове бронювання номера.

private void nightsTextBox_TextChanged(object sender, TextChangedEventArgs e) – виводить загальну ціну броні.

private void Button_Click(object sender, RoutedEventArgs e) – видалення вибраної броні.

private void paidButton_Click(object sender, RoutedEventArgs e) – оплата вибраної броні.

private void Button_Click_1(object sender, RoutedEventArgs e) – видалити вибраний дохід.

private void tabItem6_Loaded(object sender, RoutedEventArgs e) – оформлення звіту.

Опис концептуальної моделі бази даних:

book:

Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
roomNumber	INT(11)	☑	☑	☐	☐	☐	☐	☐	
customerNumber	INT(11)	☐	☐	☐	☐	☐	☐	☐	NULL
checkInDate	DATETIME	☐	☐	☐	☐	☐	☐	☐	NULL
nights	INT(11)	☐	☐	☐	☐	☐	☐	☐	NULL
paid	TINYINT(1)	☐	☐	☐	☐	☐	☐	☐	NULL

Рисунок 1.5 – Таблиця book

customer:

Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
id	INT(11)	☑	☑	☐	☐	☐	☐	☑	
lastName	VARCHAR(45)	☐	☐	☐	☐	☐	☐	☐	NULL
firstName	VARCHAR(45)	☐	☐	☐	☐	☐	☐	☐	NULL
passportNumber	VARCHAR(45)	☐	☐	☐	☐	☐	☐	☐	NULL
phoneNumber	VARCHAR(45)	☐	☐	☐	☐	☐	☐	☐	NULL

Рисунок 1.6 – Таблиця customer

income:

Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
id	INT(11)	☑	☑	☐	☐	☐	☐	☐	
roomNumber	INT(11)	☐	☐	☐	☐	☐	☐	☐	NULL
nights	INT(11)	☐	☐	☐	☐	☐	☐	☐	NULL
paidTotal	INT(11)	☐	☐	☐	☐	☐	☐	☐	NULL
paymentDate	DATETIME	☐	☐	☐	☐	☐	☐	☐	NULL

Рисунок 1.6 – Таблиця income

outgo:

Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
id	INT(11)	☑	☑	☐	☐	☐	☐	☐	
purpose	VARCHAR(45)	☐	☐	☐	☐	☐	☐	☐	NULL
paidTotal	VARCHAR(45)	☐	☐	☐	☐	☐	☐	☐	NULL
paymentDate	DATETIME	☐	☐	☐	☐	☐	☐	☐	NULL
receiver	VARCHAR(45)	☐	☐	☐	☐	☐	☐	☐	NULL

Рисунок 1.7 – Таблиця outgo

room:

Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
number	INT(11)	☑	☑	☐	☐	☐	☐	☐	
beds	VARCHAR(45)	☐	☐	☐	☐	☐	☐	☐	NULL
rooms	VARCHAR(45)	☐	☐	☐	☐	☐	☐	☐	NULL
rent	INT(11)	☐	☐	☐	☐	☐	☐	☐	NULL
status	VARCHAR(45)	☐	☐	☐	☐	☐	☐	☐	NULL

Рисунок 1.8 – Таблиця room

Опис програмної реалізації:

Створення нової броні. Вибираємо з списку номер кімнати і ім’я клієнта також обираємо дату поселення і вводимо кількість днів проживання. Ці дані додаються в базу в табличку book.

```
private void addBook_Click(object sender, RoutedEventArgs
e)
        {
            book newBook = new book
            {
                room = (room) roomComboBox.SelectedItem,
                checkInDate = checkInDatePicker.Value,
                customer = (customer)
customerComboBox.SelectedItem,
                nights = Int32.Parse(nightsTextBox.Text),
                paid = false
            };
            booksList.Add(newBook);
            booksDataGrid.Items.Refresh();
            db.AddTobooks(newBook);
            db.SaveChanges();

        }
```

Оплата вибраної броні. В програмі помічаємо що оплата здійснена. Вносимо дані в таблицю income(доходи) про доходи від оплаченої оренди і зберігаємо зміни в цій таблиці.

```
    private void paidButton_Click(object sender,
RoutedEventArgs e)
        {
            int lastId = db.incomes.Max(inc => inc.id);
            book book = booksDataGrid.SelectedItem as book;
            book.paid = true;
            db.SaveChanges();
            income income = db.CreateObject<income>();
            income.id = lastId + 1;
            income.room = book.room;
            income.nights = book.nights;
            income.paidTotal = book.nights*book.room.rent;
            income.paymentDate = DateTime.Now;
            db.incomes.AddObject(income);
            db.SaveChanges();
            incomesList.Add(income);
            incomesDataGrid.Items.Refresh();
        }
```

Оформлення звіту. Додаємо всі доходи і витрати підприємства і виводимо їх а також виводимо баланс(різницю між доходами і витратами).

```
        private void tabItem6_Loaded(object sender,
RoutedEventArgs e)
        {
            int incomeSum = (int) db.incomes.Sum(inc =>
inc.paidTotal);
            Income.Content = incomeSum;
            int outgoSum = 0;
            foreach (var outgo in db.outgoes)
            {
                outgoSum += int.Parse(outgo.paidTotal);
            }
            Outgo.Content = outgoSum;

            Balance.Content = incomeSum - outgoSum;
        }
```

Конвертер Клієнт в ім’я і прізвище мешканця.

Якщо є дані про клієнта то виводимо його ім’я і фамілію.

```
  public class CustomerToNameConverter : BaseConverter,
IValueConverter
```

```
    {
        public object Convert(object value, Type
targetType, object parameter,
                              System.Globalization.CultureInfo
culture)


        {
            customer cust = value as customer;
            if (cust == null)
                return null;
            return cust.lastName + " " +cust.firstName;

        }
        public object ConvertBack(object value, Type
targetType, object parameter,
                             System.Globalization.CultureInfo
culture)
        {
            return null;
        }
    }
```

Конвертер Номер в номер кімнати.

Якщо є дані про номер готелю то повертаємо номер кімнати.

```
    public class RoomToNumberConverter : BaseConverter,
IValueConverter
    {
        public object Convert(object value, Type
targetType, object parameter,
                              System.Globalization.CultureInfo
culture)
        {
            room room = value as room;
            if (room == null)
                return null;
            return room.number;

        }
        public object ConvertBack(object value, Type
targetType, object parameter,
                             System.Globalization.CultureInfo
culture)
        {
```

```
            return null;
        }
    }
```

1.3 Конструювання програмної системи

1.3.1 Основні відомості про Microsoft SQL Server

SQL Server - система управління базами даних, розроблена компанією Microsoft. Розроблений для заміни системи управління базами даних Microsoft Access, SQL Server пропонує простий спосіб доступу, обслуговування та управління базами даних для організацій та приватних осіб. Організовуючи дані, SQL Server допомагає бізнесу працювати більш ефективно. Крім того, Microsoft періодично випускає версії SQL Server з новими функціями та підвищеним рівнем безпеки, щоб зробити управління даними ще більш ефективним.

Як випливає з назви, SQL Server зберігає і управляє даними в реляційній базі даних. Дані, що зберігаються в базі даних, організовані в таблицях, які можуть мати багато різних типів даних. Доступ до даних, що зберігаються в базі даних, здійснюється за допомогою мови запитів SQL, яка дозволяє користувачам виконувати конкретні завдання, такі як ведення записів або їх оновлення. Користувач з необхідними правами доступу до бази даних може виконати будь-яке завдання в базі даних. SQL Server працює на операційних системах Microsoft, таких як Windows NT та Windows 2000. Крім того, він працює в хмарі на хмарній платформі Microsoft Azure.

Щоб підтримувати SQL Server в актуальному стані, корпорація Майкрософт створює нові функції та модернізує SQL Server для роботи з новітніми технологіями. Наприклад, коли корпорація Майкрософт випустила Windows Vista, вона включила оновлення для Microsoft SQL Server 2005. Це дозволило організаціям використовувати SQL Server для зберігання критично важливої для них інформації, наприклад, баз даних для бухгалтерії або записів про співробітників компанії. Крім того, щоразу, коли Microsoft випускає нову версію своїх операційних систем, вони часто включають оновлення для SQL Server. Це

гарантує, що організації можуть продовжувати користуватися найпотужнішою у світі системою управління даними.

На додаток до зберігання та управління даними, SQL Server також допомагає у вирішенні повсякденних завдань адміністрування баз даних. Адміністратор бази даних (DBA) контролює роботу SQL Server і забезпечує його безперебійну роботу. На додаток до звичайних оновлень програмного забезпечення, DBA також відстежує та виправляє будь-які проблеми з безпекою в кодовій базі програмного забезпечення. Ці проблеми можуть виникнути після того, як розробник входить на сервер, використовуючи обліковий запис комп'ютера розробника замість облікового запису сервера. Це дозволяє розробникам отримати несанкціонований доступ до сервера під час роботи над проектами для організації - що може призвести до катастрофічних результатів, якщо не буде негайно виправлено кваліфікованим адміністратором сервера.

Будучи однією з найпопулярніших систем управління базами даних, що використовуються сьогодні, Microsoft SQL Server неодноразово доводив свої переваги протягом багатьох років. Вперше випущений для громадськості в 1992 році, він з тих пір здійснив революцію в адмініструванні серверів у всьому світі, значно розширивши можливості для успішного управління бізнесом підприємства. Вони використовуються в бізнесі та уряді для безпечного та ефективного зберігання інформації. По суті, база даних - це система, яка збирає, організовує, аналізує та ефективно управляє даними для практичного застосування, наприклад, для прогнозування показників продажів або створення звітів для зручності управління. Одним з основних напрямків використання баз даних є зберігання даних з різних джерел, таких як фінансові системи або логіни в Інтернеті.

1.3.3 Розробка структури бази даних

За допомогою програмного продукту SQL SMS зручно створювати окремі таблиці (файли даних) і реляційні схеми баз даних. Фізична структура таблиці повинна відповідати визначеній раніше моделі. Фізична модель визначає заданий порядок, у якому дані розміщуються на

комп’ютері, формат, у якому вони зберігаються, і конкретні методи доступу до даних.Отже, за допомогою фізичної моделі ми будемо мати вміст таблиці, які користувач фактично зберігає та використовує. Об’єктами баз даних є таблиці, які необхідно створити, а їхні властивості — це поля таблиць. Задля цього при створенні однієї таблиці вкажіть наступне:

- Назви стовпців (назви властивостей об'єктів, заголовки полів таблиці).
- Тип даних (вказати тип даних, що визначає властивості об'єкта, тип поля таблиці).
- Допускаються значення NULL (вміст властивостей заданого об'єкта, вміст полів таблиці).

Створювання заголовків колонок дає можливість задати структуру таблиці.

Вибір відповідного типу даних дозволить: уникнути помилок при роботі з таблицями, оскільки неможливо вводити тести в поля з форматом дати і навпаки, зменшити загальну ємність бази даних у пам'яті програми. Якщо відповідне поле таблиці допускає NULL, поле може не містити даних, і навпаки, що визначає цілісність даних об’єкта.

Ці типи даних визначають набір прийнятних значень, формат їх зберігання, розмір виділеної пам’яті та набір операцій, які можна виконувати над даними поля.

Структура таблиці бази даних задається за допомогою опису (конструкції) таблиці в середовищі SQL, яка точно відповідає зазначеному інформаційному об’єкту. Після того як структура таблиці сформована, база даних заповнюється даними.

Вони об’єднуються на основі формування схеми даних з метою встановлення зв’язків між таблицями згідно з логічною моделлю ER-діаграми.

Існує 3 типи зв’язку, кожен з яких визначається зв’язками «один до одного», «один до багатьох» і «багато до багатьох». «Один до одного» використовується для того, щоб один запис в одній таблиці відповідав тільки одному Запишіть інший. У наступних двох випадках твердження походить від назви відносини.

У цьому розділі розбираються поставлені завдання для дипломного проекту. На основі попереднього аналізу визначаються засоби і методи вирішення відповідних завдань. Розглянуто середовище та мовні засоби розробки програм, визначення окремих понять і загальних властивостей.

РОЗДІЛ 2
ТЕСТУВАННЯ ПРОГРАМНОЇ СИСТЕМИ

2.1 Програма задачі та її опис

Програма компонентів інформаційної системи складається з окремих складових кожного з модулів, до складу яких входять процедури та функції, які виконують обробку інформації або забезпечують обробку визначених подій. Програми наведені у додатках.

2.2 Налагодження та тестування програми

Тестування програми – це перевірка правильності виконання програми при різних початкових даних та при виконані функціональних режимів.

На етапі тестування програми перевіряється логіка роботи програми в ході її виконання з конкретними вхідними даними.

Відладка – процес розробки комп’ютерної програми, що виявляє, локалізує і усуває помилки. На цьому етапі програма перевіряється на синтаксичні та семантичні помилки.

Налагодження - це процес, який починається з моменту виявлення помилки та її локалізації в програмі, тобто визначення її характеру та місця розташування. Тому налагодження програми передбачає, що помилки є обов'язковими.

Налагодження програми є досить складним процесом, тому що для виправлення помилок необхідно повністю визначити їх причини, а ці причини часто неочевидні. Крім того, налагодження — це процес, який лише тимчасово зупиняється, доки тест не знайде іншу помилку.

Синтаксична помилка – помилка, яка пов’язана з неправильністю написання послідовності символів або токенів певною мовою програмування, призводить до

краху програми при виконанні. У середовищі Visual Studio в редакторі коду більшість синтаксичних помилок виділяються і їх легко помітити і усунути.

Семантична помилка (баг) – призводить до некоректної роботи програми, але не призводить до її краху. Засобами Visual Studio можна покроково виконувати код і переглядати значення змінних в кожен момент часу, щоб побачити де саме в програмі виникає некоректна робота і виправити її. Також для цього можна встановлювати точки «зупину» на певних рядках коду, де буде зупинятися робота програми для аналізу значень змінних.

Коли програма виявила помилку вона видає повідомлення про знаходження даної помилки, та намагається як найточніше вказати місце її знаходження.

Під час розробки програмного додатку було виявлено такі помилки:

- [Error] Unit3.pas(65): Missing operator or semicolon
- [Fatal Error] Project1.dpr(17): Could not compile used unit 'Unit1.pas'
- [Error] Unit3.pas(86): Undeclared identifier: 'ataModule2'
- [Error] Unit3.pas(278): '.' expected but end of file found

Всі виявлені помилки було проаналізовано і виправлено. Після відлагодження проект запрацював нормально.

2.3 Дослідна експлуатація програми

Дослідна експлуатація – одна з останніх стадій створення обладнання, пристроїв, систем, програм та методик, що полягає в використанні вище перерахованого за прямим призначенням в реальних робочих чи технологічних процесах та під час якої остаточно визнається їх функціональна придатність.

Дослідна експлуатація проекту відбулась відразу після закінчення етапу тестування та налагодження, в результаті чого було отримано виконавчий модуль програми. Перед початком дослідної експлуатації були створені макети таблиць БД. Склад даних цих таблиць передбачав можливість випробовувань

проекту для різних значень даних включаючи виникнення виключних ситуацій. Під час експлуатації програми перевіряємо усі режими роботи програми, щоб переконатися у відповідності функціонування додатку поставленим вимогам.

У процесі дослідної експлуатації проекту відбувались його багаторазові запуски та виправлення всих недоліків, які виникали, після виправлення яких програма почала працювати коректно. Також було виконано корегування кольорів вікон діалогу та окремих елементів керування.

Розроблена АІС надає доступ до редагування всіх необхідних довідників розробленої бази даних промислового підприємства, а саме довідника гатунка та матеріалу, довідника міста и вулиці.

Отже, розроблений додаток повністю відповідає вимогам постановки задачі.

2.4 Результати роботи з програмою

Вкладка Клієнти

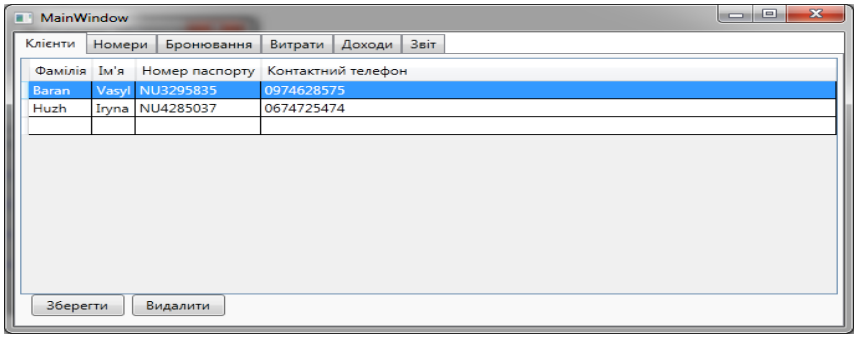

Рисунок 2 – Вкладка Клієнти

Вкладка Номери

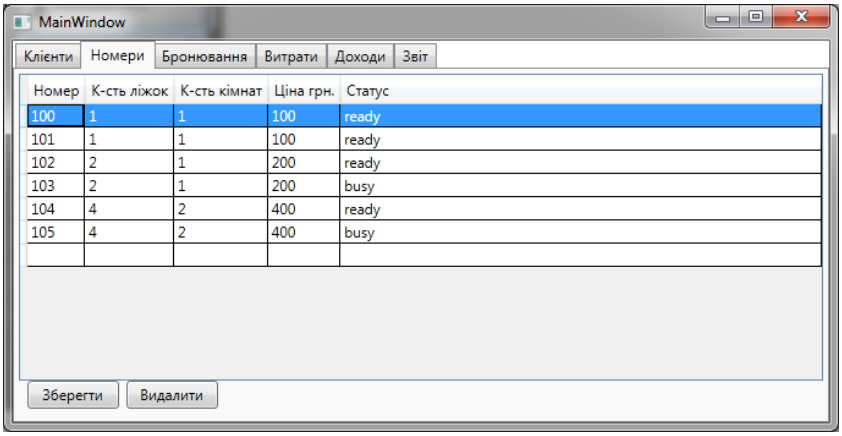

Рисунок 2.1 – Вкладка Номери

Вкладка Бронювання

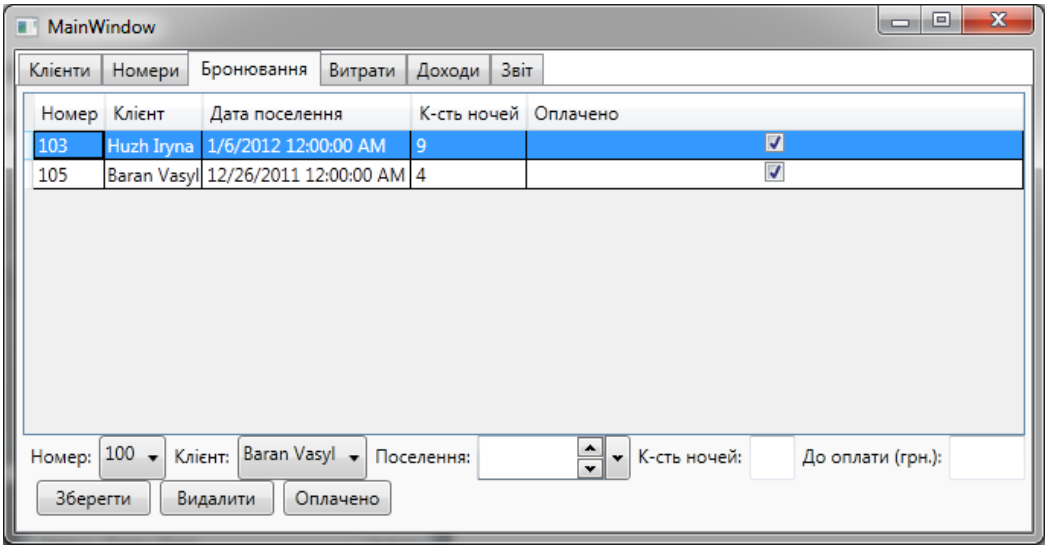

Рисунок 2.2 - Вкладка Бронювання

Вкладка Витрати

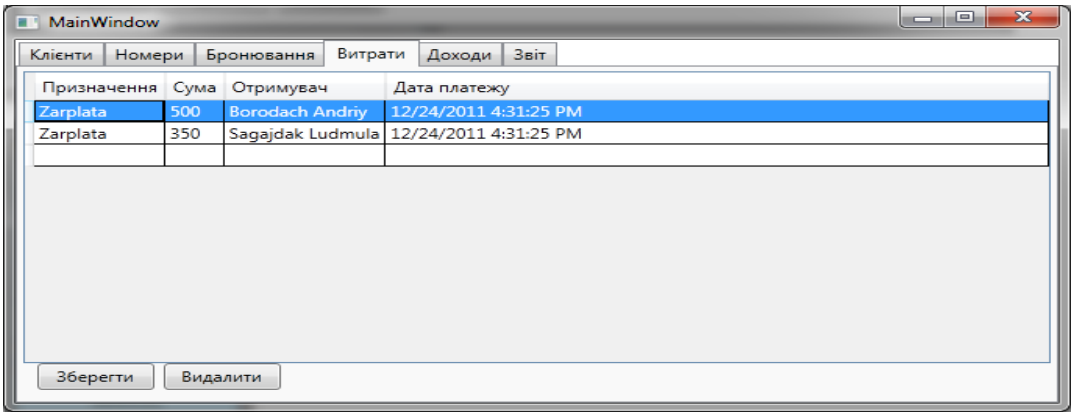

Рисунок 2.3 – Вкладка Витрати

Вкладка Доходи

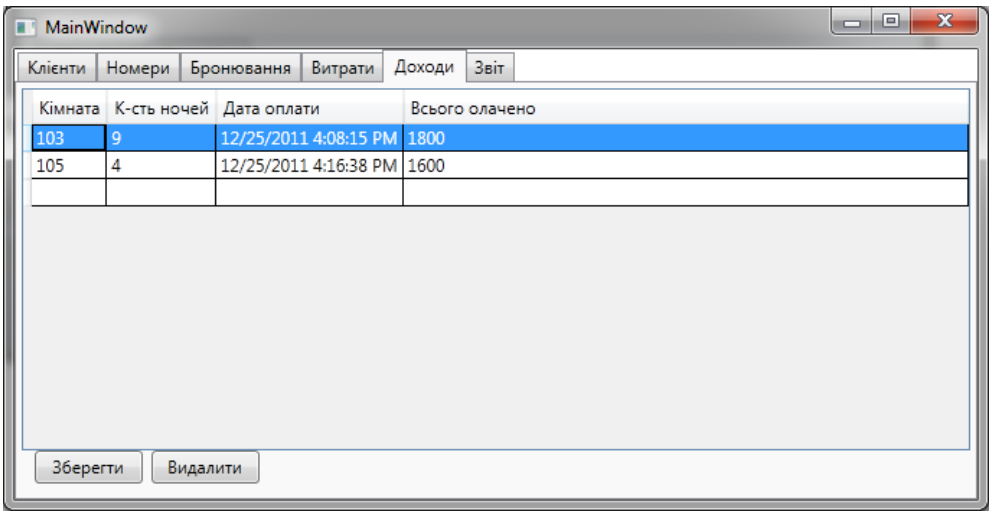

Рисунок 2.4 – Вкладка Доходи

Введення інформації про нового мешканця

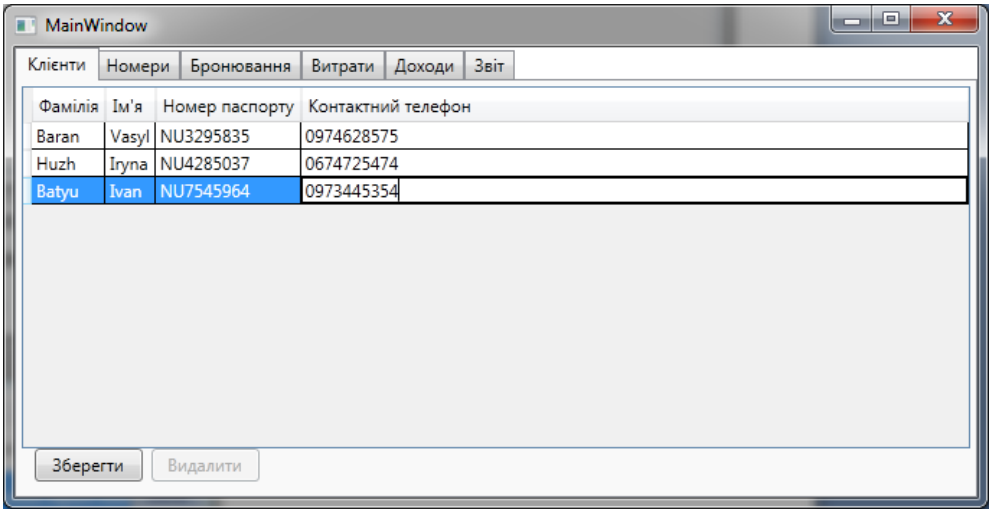

Рисунок 2.5 - Введення інформації про нового мешканця

Бронювання номера.

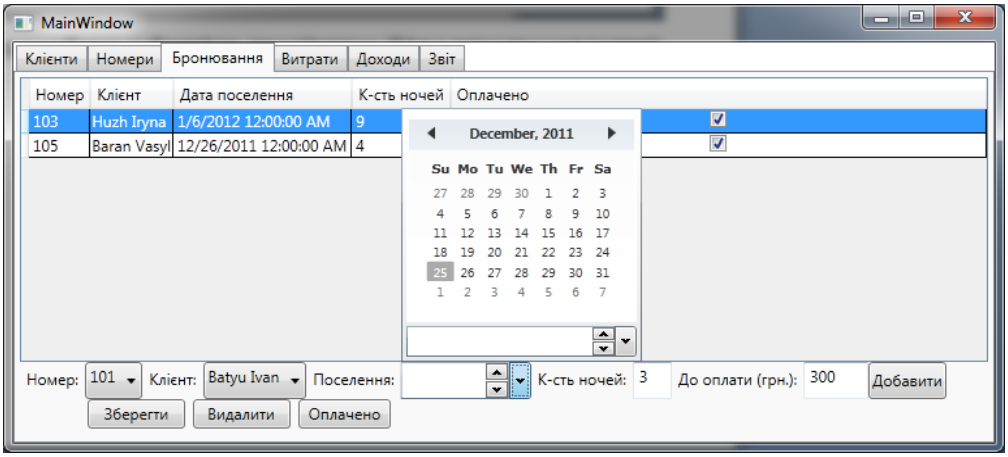

Рисунок 2.6 - Бронювання номера

Оформлення звіту

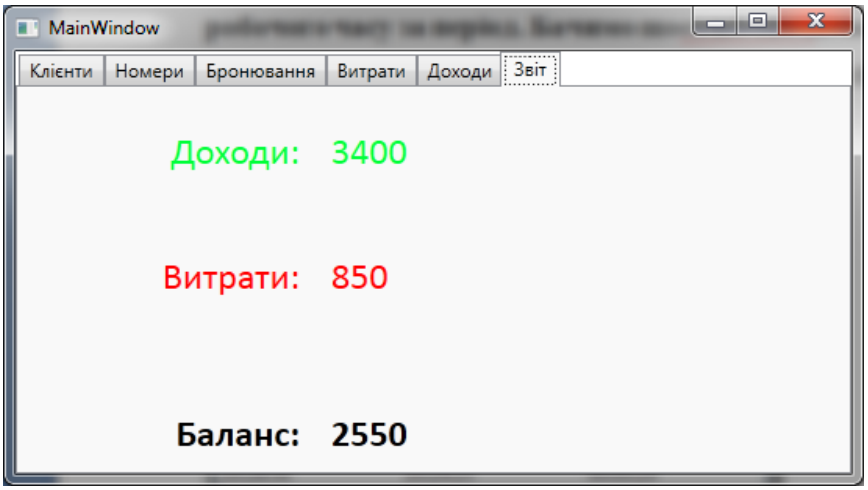

Рисунок 2.7 - Оформлення звіту

У цьому розділі вивчалася робота з розробки програми. Показано робочий приклад тестування. Проведено верифікації, які працюють, у наданих ситуаціях. Розглянено, чи існує обробка деяких винятків. Результати проведених операцій представлені графічно.

РОЗДІЛ 3
ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

3.1 Охорона праці

Незалежно від професійної діяльності питання охорони повинні вирішуватися на всіх етапах трудового процесу.

Безпечні і здорові умови праці значною мірою залежать від правильного оцінювання небезпечних та шкідливих факторів виробництва. Не менш складні зміни в організмі викликаюится різними причинами. Фактори можуть бути такі: виробничої обстановки, надмірне фізичне та психічне напруження, нервово-емоційне напруження, різні поєднання цих причин.

При розробці програмного забезпечення інформаційної системи управління підприємством використано приміщення загальною площею 20 м^2, висотою стелі 3 м, де розташовано 6 місць з ПК. Кожне місце має бути обладнане робочим столом площею 1,2 м^2, стільцем та комп'ютером з такими складовими: монітор, системний блок, клавіатура та миша. Потрібно відзначити, що площа робочого місця повинна бути більшою або дорівнювати 6 м^2, а об'єм повинен бути в межах 20 м^3 або більше, отже площа та об'єм приміщення підходить для розташування шести робочих місць.

Аналіз умов праці показує, що на програмістів негативно впливають такі фізичні та психофізіологічні фактори:

• Підвищення або зниження температури повітря в робочій зоні;

• підвищення або зниження вологості повітря;

• Недостатнє освітлення на робочому місці;

• Рівень шуму який є підвищиний на робочому місці;

• Іонізація повітря;

• Рівень електромагнітного випромінювання;

• Нервово-психічні перевантаження (психічне перенапруження, перенапруження аналізатора);

• Фізичні перевантаження (статична втома через одноманітну позу).

Робота програміста за енерговитратами відноситься до легких робіт Ia, Ib і тому повинна відповідати наступним вимогам згідно з ДСН 3.3.6.042-99:

• Оптимальна температура повітря – 22°С (допустимою є – 20-24°С);

• Вологість яка є оптимальною - 40-60% (допускається - не більше 75%);

• Швидкість повітря не перевищує 0,1 м/с.

Джерелом тепловиділення є 6 ПК, які знаходяться в приміщенні, крім того, нагріті поверхні системи опалення використовуються для підтримки оптимальних параметрів мікроклімату в приміщенні в холодну пору року. Нормованим показником ІЧ є гранично допустима щільність потоку енергії $J_{г.д}$, Вт/м2, яка встановлюється відповідно до площі опромінюваної поверхні тіла людини ($S_{опр}$).

Стандартизовані рівні:

- $J_{г.д}$=35 Вт/м2 при $S_{опр}$ > 50%;
- $J_{г.д}$=70 Вт/м2 при $S_{опр}$ ~ 25-50%;
- $J_{г.д}$=100 Вт/м2 при $S_{опр}$ < 25%.

Параметром нормування денного освітлення для ДБН В.2.5-28:2018 є коефіцієнт денного освітлення (КПО). КПО встановлюються залежно від виду виконуваних зорових робіт. Робота програмістів належить до середньої точності робіт (робота IV класу зору, мінімальний розмір об'єкта роздільної здатності 0,5-1,0 мм), при використанні бічного світла КПО = 1,5%.Для штучного освітлення нормованими параметрами є Emin — мінімальна освітленість, Кр — коефіцієнт флуктуації світлового потоку, який не повинен перевищувати 20 %. Мінімальні рівні освітлення встановлюються відповідно до виду виконуваної зорової роботи. IV категорія візуальних робіт 300-500 лк. Для забезпечення штучного освітлення використовують лампи з типом ЛПО.Кожний світильник укомплектований двома

лампами. Тобто необхідно використовувати 6 світильників із 12 працюючими лампами в них.

Оскільки при розробці програмного забезпечення інформаційної системи управління підприємством використовуються ПК, кожен з яких оснащений монітором, жорстким диском в системному блоці, системою охолодження, а саме трьома вентиляторами та клавіатурою. Крім того, поруч працює периферія, тому шум від механічних і аеродинамічних джерел час від часу посилюється широкосмуговим (під час роботи принтера). Стандартним параметром невикористаного рентгенівського випромінювання є потужність експозиційної дози. Його рівень не повинен перевищувати 100 мкР/год на відстані 5 см від поверхні екрана монітора. 20 мкР/год є максимальним рівнем рентгенівського випромінювання на робочому місці програміста та зазвичай не перевищує цього значення. Відстані від екрану є 5-10 см та корпусу монітора рівень, напруги в електричних компонентах може досягати 140 В/м, що значно перевищує допустиме значення.

Таким чином, встановлено необхідність забезпечення вимог охорони праці та техніки безпеки на робочому місці програміста при розробці програмного забезпечення інформаційної системи управління підприємством

3.2 Оцінка стійкості роботи об'єкту економіки до впливу поражаючих факторів ядерної зброї

Приділяючи велику увагу зміцненню обороноздатності нашої країни, урядом неодноразово підкреслювалося, що оборонна міць держави складається не тільки з високої готовності й оснащення Збройних сил, а й нерозривно позв’язана з високим рівнем економічного розвитку країни, підготовкою населення й об’єктів народного господарства до захисту від зброї масового ураження. Велику

роль у цьому важливому питанні відіграє цивільний захист країни, що будучи системою загальнодержавних оборонних заходів, покликаний не тільки забезпечити захист населення в надзвичайних ситуаціях, але і здійснювати заходи, спрямовані на забезпечення стабільної роботи підприємств господарювання країни в таких умовах.

3.2.1 Стійкість роботи об’єктів господарювання в умовах надзвичайних ситуацій воєнного часу.

Під стійкістю роботи промислового об’єкта (об’єкта господарювання любої форми власності) розуміють здатність його в умовах надзвичайних ситуацій мирного і воєнного часу випускати продукцію в запланованому обсязі й номенклатурі, а при одержанні слабких і середніх руйнувань, порушенні зв’язків по кооперації і постачанням відновлювати виробництво в мінімальний термін. Здатність об’єкта народного господарства випускати продукцію залежить від захисту і нормального функціонування чотирьох основних елементів сучасного виробництва, якими є:

– виробничий персонал (робітники та службовці);

– будинки і споруди з технологічним устаткуванням;

– система постачання енергією, водою, паливом, устаткуванням і ремонтною базою;

– система виробничих і кооперативних зв’язків з іншими об’єктами. Тому стійкість роботи об’єктів і галузі народного господарства в цілому в умовах надзвичайних ситуацій визначається наступними факторами:

– надійністю захисту робітників та службовців від усіх вражаючих факторів зброї масового ураження;

– здатністю інженерно-технічного комплексу (ІТК) об’єкта протистояти вражаючим факторам ядерного вибуху; – надійністю системи постачання об’єкта

всім необхідним для виробництва продукції (сировиною, паливом, що комплектують виробами, електроенергією, водою, газом тощо.);

– захищеності об’єкта від вторинних вражаючих факторів (пожеж, вибухів, затоплень, зараження місцевості отруйними і сильнодіючими отруйними речовинами);

– стійкістю і безперервністю керування виробництвом і цивільною обороною;

– підготовленість об’єкта до проведення рятувальних та інших невідкладних робіт і робіт з відновленням порушеного виробництва.

Перераховані фактори визначають собою й основні, загальні для всіх об’єктів господарювання, шляхи підвищення стійкості роботи в надзвичайних ситуаціях, а саме:

– забезпечення надійного захисту робітників та службовців від вражаючих факторів зброї масового ураження;

– захист основних виробничих фондів від вражаючих факторів, у тому числі й від вторинних;

– підвищення надійності й оперативності керування виробництвом;

– забезпечення стійкості постачання всім необхідним для випуску запланованої на час надзвичайних ситуацій продукцією;

– підготовка до відновлення порушеного виробництва.

3.2.2 Захист засобів виробництва

Такий захист полягає в підвищенні фізичної опірності будинків, споруджень і конструкцій об’єкта до впливу вражаючих факторів ядерного вибуху, захисту технологічного і верстатного устаткування, засобів зв’язку й інших засобів, що складають матеріальну основу виробничого процесу. Методика оцінки стійкості будинків, технологічного устаткування об’єкта народного

господарства до вражаючих факторів ядерного вибуху виконується по трьох основних вражаючих факторах:

– від впливу ударної хвилі ядерного вибуху;

– від світлового випромінювання на предмет виникнення пожеж;

– від радіації на предмет захисту виробничого персоналу від опромінення.

3.2.3 Оцінка стійкості об’єкта господарювання до впливу ударної хвилі.

Як кількісний показник приймається надлишковий тиск △Рф, при якому будинок, спорудження, й устаткування зберігається або одержує слабкі й середні руйнування. За межу стійкості до ударної хвилі, береться нижня границя діапазону середніх руйнувань основних елементів об’єкта. Висновок про стійкість об’єкта до ударної хвилі здійснюється шляхом порівняння знайденої межі стійкості об’єкта **(△Рфlim)** з очікуваним за прогнозом максимальним значенням надлишкового тиску **(△Рфmax)** і дотримується умова:

△Рфlim ≥ △Рфmax, – об’єкт стійкий.

Оцінка стійкості об’єкта до впливу світлового випромінювання. Як показник стійкості об’єкта народного господарства до світлового випромінювання приймається мінімальне значення світлового імпульсу, при якому може відбутися запалення матеріалів конструкції, у результаті яких виникнуть пожежі. Це значення світлового імпульсу прийнято вважати межею стійкості об’єкта народного господарства до впливу світлового випромінювання **Uсвlim**. Об’єкт вважається стійким до світлового випромінювання, якщо при очікуваному імпульсі не загоряються будь-які елементи, матеріали, тобто дотримується умова:

Uсвlim ≥ Uсвmax, – об’єкт стійкий

де **Uсвmax** – очікуваний за прогнозом світловий імпульс на об’єкті народного господарства.

3.2.4 Оцінка стійкості об’єкта до впливу проникаючої радіації і радіоактивного зараження.

За критерій стійкості роботи об’єкта від радіації приймається припустима доза радіації, яку можуть одержати люди за час роботи в конкретних умовах.

Таблиця визначення дози проникаючої радіації дана в П.9 (довідник Демиденко Г.І., с. 246). Доза радіації від впливу радіоактивного зараження визначається:

Дрзвм = 5·Р1·(tп -0,2 – tк -0,2),

де Р1 – рівень радіації на 1 годину після вибуху (Р/год.);

tп – час утворення радіоактивного зараження: tп = Rx / Vсв + tвип.

де Rx – відстань до об’єкта;

Vсв – середня швидкість вітру;

tвип – час випадіння радіоактивних опадів

tк – кінець опромінення: tк = tп + tр

де tр – час робочої зміни, або час перебування у захисних спорудах.

За значеннями дози визначаються втрати людей (табл. 11.4., довідник Демиденко Г.П., с. 124).

Межа стійкості цеху в умовах радіоактивного зараження (граничне значення рівня радіації **△P1lim** Р/год. на об’єкті, при якому можлива робота) визначається за формулою:

$$\Delta P_{1lim} = \frac{Д_{уст} \cdot К_{посл\ ЗС}}{5 \cdot (t_п^{-0,2} - t_к^{-0,2})},$$ – цех стійкий,

де $Д_{уст}$ – установлена доза радіації;

$\triangle P_{1lim}$ – рівень радіації на 1 годину після вибуху;

$К_{посл\ ЗС}$ – коефіцієнт послаблення радіації цеху, сховища (тобто для умов, у яких буде знаходиться персонал).

Методику оцінки стійкості об’єкта господарювання ми будемо розглядати під час практичних занять

3.2.5 Організація проведення дослідження з оцінки стійкості роботи об’єкта господарювання.

Оцінка стійкості роботи об’єкта – це всебічне вивчення підприємства з погляду здатності його протистояти впливу вражаючих факторів ядерного вибуху, відновлення виробництва при одержанні середніх і слабких руйнувань. Мета дослідження складається в тому, щоб виявити уразливі місця в роботі об’єкта у воєнний час і виробити найбільш ефективні пропозиції і рекомендації, спрямовані на підвищення його стійкості. Надалі ці рекомендації включаються в план заходів щодо підвищення стійкості роботи об’єкта, що і реалізується. Дослідження стійкості підприємства проводиться силами інженернотехнічного персоналу із залученням фахівців науково-дослідних і проектних організацій, пов’язаних із даним підприємством. Організатором і керівником дослідження є голова підприємства – начальник ЦЗ об’єкта. Весь процес планування і проведення дослідження можна розділити на три етапи: перший – підготовчий, другий – оцінка стійкості роботи об’єкта в умовах воєнного часу, третій – розробка заходів,що підвищують стійкість роботи об’єкта.

ВИСНОВКИ

У кваліфікаційній роботі магістра розкрито питання розробки автоматизованої системи управління підприємством при застосуванні SQL технології та архітектури «клієнт-сервер» на прикладі готельного підприємства.

У першому розділі після аналізу поставленого завдання розглянуто основні принципи побудови інформаційних систем. Вказано на необхідність застосуванні технології клієнт-сервер і мови SQL.

Проведено вибір засібу реалізації поставленої мети, а саме створення програмного забезпечення.

У другому розділі розроблені структура бази даних і інтерфейс програми. До створеного програмного продукту була складена документація, включаючи інструкцію по введенню в експлуатацію і роботі з програмою.

У розділі «Охорона праці та безпека в надзвичайних ситуаціях» розглянуті питання, які стосуються тематики роботи.

Враховуючи вище сказане можна зробити висновок, що всі поставленні завдання виконані в повному обсязі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Виноградська А.М. Технологія комерційного підприємництва: навч. посібник / А.М.Виноградська. – До.: Центр навчальної літератури, 2006. – 780 с.

2. Сайт глобальної системи бронювання Amadeus [Електронний ресурс]. – Режим доступу: http://www.amadeus.ru

3. Виноградська А.М. Технологія комерційного підприємництва: навч. посібник / А.М.Виноградська. – До.: Центр навчальної літератури, 2006. – 780 с..

4. Сайт, що містить основні відомості про проектування інформаційних систем в Microsoft SQL Server 2008 и Visual Studio 2008 – www.intuit.ru

5. Сайт, що містить уроки по програмуванню в Visual Studio – easyprog.ru

6. Сайт, що містить основні принципи роботи в Microsoft SQL Server 2008 – c.inf.ua

7. РАБОТА С БАЗАМИ ДАННЫХ НА ЯЗЫКЕ C#. ТЕХНОЛОГИЯ ADO .NET: учебное пособие / сост. О. Н. Евсеева, А. Б. Шамшев. - Ульяновск: УлГТУ, 2009. - 170 с.

8. Дейт К. Введение в системы баз данных / К. Дейт; 7-е издание; пер. с англ. – М. : Издательский дом "Вильямс", 2001. –1072 с.

9. Мартин Дж. Организация баз данных в вычислительных системах / Дж. Мартин. – М.: Мир, 1980. – 662 с.

10. Диго С.М. Проектирование и использования баз данных / С.М.Диго. – М.: Финансы и статистика, 1995. – 208 с.

11. Дейт К. Руководство по реляционной СУБД DB2 / К. Дейт. – М.: Финансы и статистика, 1988. – 320 с.

12. Кириллов В.В. Основы проектирования реляционных баз данных. Учебное пособие / В.В. Кириллов . – СПб.: ИТМО, 1994. – 90 с.

13. Мейер М. Теория реляционных баз данных / М. Мейер. – М.: Мир, 1987. – 608 с.

14. Голованов Б.Г. Введение в программирование в сетях Novell NetWare / Б.Г. Голованов. – С-П.: Питер, 2000. – 384 с.

15. Губський А.І. Цивільна оборона / А.І. Губський. – К: Міністерство освіти, 1996. – 216 с.

16. Методичні вказівки до виконання атестаційної роботи магістра за спеціальністю 121 – Інженерія програмного забезпечення (Освітньо-професійна програма - «Програмне забезпечення систем», Освітньо-наукова програма - «Інженерія програмного забезпечення») для студентів усіх форм навчання / Упор.: М.Р. Петрик, Д.М. Михалик, О.Ю. Петрик, Г.Б. Цуприк - Тернопіль: ТНТУ, 2020-51с.

ДОДАТКИ

ДОДАТОК А

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ПУЛЮЯ

ТЕХНІЧНЕ ЗАВДАННЯ

на розробку кваліфікаційної роботи

Розробка автоматизованої системи обліку внутрішньопереміщених осіб з використанням MySQL, мова програмування C#

Узгоджено:

Керівник курсової роботи к.т.н., д.т.н проф. Стадник І.Я. (підпис)

“____” вересня 2022р.

Виконавець:

Студент групи СПм- 61 Петрик Олег Володимирович (підпис)

“____” вересня 2022 р

Тернопіль 2022

ЗВІТ

1. ПІДСТАВИ ДО РОЗРОБКИ

Розробка проводиться у відповідності до графіку навчального плану підготовки бакалаврів за спеціальністю 121 «Інженерія програмного забезпечення».

Тема кваліфікаційної роботи: «Розробка автоматизованої системи обліку внутрішньопереміщених осіб з використанням MySQL, мова програмування C#».

Термін виконання: до «___» _____ 2022р.

2. ПРИЗНАЧЕННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

Інформаційна система призначена для управління та автоматизації роботи із обліку внутрішньопереміщених осіб

3. ВИМОГИ ДО ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1 Функціональні вимоги

Система повинна передбачати дві ролі: адміністратор та працівник.

Адміністратор має доступ до всіх таблиць бази даних, а також може редагувати чи видаляти дані із таблиць. Працівник буде мати можливість переглядати інформацію про осіб.

Розроблювана система буде являти собою закінчений програмний продукт, що реалізує автоматизацію процесу управління потоком послуг відповідно до поставлених завдань. Власне проблема вирішення цих проблем і вирішується безпосередньо у дипломній роботі

Щоб система коректно працювала, вона матиме дружній та інтуєтивно зрозумілий інтерфейс користувача, зрозуміле призначення функцій та очікуваний результат людино машинної взаємодії, що дозволить їй зайняти відповідне місце у існуючому програмному просторі сьогодення.

Набір даних функцій дозволяє користувачу використовувати програму для автоматизації процесу обліку запчастин для підприємств агропромислового сектору.

3.2 Технічні вимоги

Вимоги до клієнтської частини: OS Windows, дружній та їнтуєтивно зрозумілий інтерфейс користувача, зрозуміле призначення функцій та очікуваний результат людино машинної взаємодії.

3.3 Програмні вимоги

Розробка серверної частини: SQL.

Розробка клієнтської частини: C#, .NET .

Додаткові вимоги: СКБД - Microsoft SQL Server.

4. ЕТАПИ РОЗРОБКИ

Розробка інформаційної системи проводиться в наступному порядку:

– аналіз предметної області, аналіз конкурентів та основних алгоритмів програмної системи;

– вибір засобів розробки;

– розробка моделі та складових програмного комплексу;

– оформлення супровідної документації;

– здача проекту.

Результати виконання кожного етапу проекту погоджуються з керівником проекту.

5. СУПРОВІДНА ДОКУМЕНТАЦІЯ

Для інформаційної системи повинні бути розроблені наступні документи:

- завдання
- пояснювальна записка до кваліфікаційної роботи;
- презентація проекту;
- рецензія на проект;
- диск з проектом.

Пояснювальна записка до проекту оформляється згідно діючих вимог до нормоконтролю проектів.

6. ПОРЯДОК ЗДАЧІ ПРОЕКТУ

Розроблена інформаційна системи повинна відповідати вимогами, що складаються з перерахованих у п.3.1 цього документу характеристик.

Для здачі проекту необхідно підготувати весь перелік документів зазначений у п.5 цього документу.

Приймання проекту проводиться спеціально створеною комісією в термін зазначені в п.1 цього документу.

ДОДАТОК Б

Тези

УДК 004.41
О. Петрик, І. Стадник
(Тернопільський національний технічний університет імені Івана Пулюя, Україна)

РОЗРОБКА АВТОМАТИЗОВАНОЇ СИСТЕМИ ОБЛІКУ ВНУТРІШНЬО ПЕРЕМІЩЕНИХ ОСІБ

UDC 004.41
O. Petryk, I. Stadnyk

DEVELOPMENT OF AN AUTOMATED ACCOUNTING SYSTEM FOR INTERNALLY DISPLACED PERSONS

У наш важкий час надзвичайно велика кількість українців, нажаль, отримала статус ВПО та були змушені перебратися до західної України в пошуках прихистку. Готелі та різні бази, які знаходяться на західній частині нашої держави можуть стати чудовим варіантом для цих людей. Тому у сьогоденні нагальною проблемою стало встановлення порядку у ведені документацій різних готельних систем. На основі цього було вирішено створити умови для закладів, які зможуть з легкістю автоматизувати та вести облік таких людей у своїх системах. Умови полягають у створенні систем інформаційного управління.

Аналізувавши вимоги майбутньої системи дозволив получити уявлення про те, як система повинна надавати послуги своїм користувачам. У процесі було переглянуто кілька продуктів, які зараз представлені на ринку, одна з них Fidelio Front Office. Вона дозволяє проводити автоматизацію бронювання, реєстрації і виселення відвідувачів із веденням документації. Її основні переваги це: легкість, зручність, безпека та гнучкість. Також зроблено аналіз браузерних версій деяких програм, які дозволяють бронювати номери, надають інформацію про готель. У результаті аналізу предметної області було визначено основні проблеми:

- Більшість систем на ринку нашої держави є зарубіжними, що також піднімає планку для початкового користувача, адже потрібно мати навички іноземної мови при роботі з нею.
- Однією з найбільш поширених систем на ринку є Fidelio.
- Впровадження нових систем розробленими закордонними фірмами вимагає від роботодавця робити додаткові видатки на підвищення кваліфікації.

Враховуючи усе вищесказане, було вирішено задля правильного функціонування місць розміщення ВПО, потрібно надати відповідне комп'ютерне забезпечення. Дане забезпечення локалізовано під наш ринок та дозволить бізнесу витрачати менше коштів на підтримку програмного забезпечення.

Під час проектування програмного забезпечення було описано ключові сценарії використання системи, визначено кінцевого користувача та виконуванні дії. Початок планування складався з виявлення основи програми та її сутностей, після цього було побудовано схему бази даних, а саме приведення кожної таблиці до певної нормальної форми. Проведено проектування програми та проведено комплексне тестування усієї програми.

Отже, після реалізації проекту було проведено комплексне тестування усієї програми. Тестування – це дії, які націлені на перевірку правильності програми, тому цей процес є одним з ключових у життєвому циклі створення програмного забезпечення. На кінцевій стадії відбулася дослідна експлуатація системи, усі недоліки було виправлено. У результаті ми отримали конкурентноспроможну систему, яка вирішує нагальні питання, що виникли на ринку за останній час.

Література

1. Диго С. М. Проектирование и использования баз данных. М.: Финансы и статистика, 1995. 208 с.
2. Кириллов В. В. Основы проектирования реляционных баз данных: учебное пособие. СПб.: ИТМО, 1994. 90 с.

ДОДАТОК В

Тест план та результати тестування

На лістингах представлено реалізацію модульних тестів методів створення та редагування даних відповідно.

```
using System;
using System.Collections.Generic;
using System.Data.Objects;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Markup;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using Hotel.DAL;

namespace Hotel
{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window
    {
        private hotelEntities db;
        private CollectionViewSource customersSource;
        private CollectionViewSource roomsSource;
        private CollectionViewSource booksSource;
        private CollectionViewSource outgoesSource;
        private CollectionViewSource incomesSource;
        private List<book> booksList;
        private List<income> incomesList;
        public MainWindow()
        {
            InitializeComponent();
            db = new hotelEntities();
            customersSource =
(CollectionViewSource)this.Resources["customersSource"];
            roomsSource =
(CollectionViewSource)this.Resources["roomsSource"];
            booksSource =
(CollectionViewSource)this.Resources["booksSource"];
            outgoesSource =
(CollectionViewSource)this.Resources["outgoesSource"];
```

```
            incomesSource =
(CollectionViewSource)this.Resources["incomesSource"];

            customersSource.Source = db.customers;
            roomsSource.Source = db.rooms;
            outgoesSource.Source = db.outgoes;


            booksList = db.books.ToList();
            incomesList = db.incomes.ToList();

            booksSource.Source = booksList;
            incomesSource.Source = incomesList;
        }
        private void saveButton_Click(object sender, RoutedEventArgs
e)
        {
            db.SaveChanges();
        }

        private void addBook_Click(object sender, RoutedEventArgs e)
        {
            book newBook = new book
            {
                room = (room) roomComboBox.SelectedItem,
                checkInDate = checkInDatePicker.Value,
                customer = (customer) customerComboBox.SelectedItem,
                nights = Int32.Parse(nightsTextBox.Text),
                paid = false
            };
            booksList.Add(newBook);
            booksDataGrid.Items.Refresh();
            db.AddTobooks(newBook);
            db.SaveChanges();

        }

        private void nightsTextBox_TextChanged(object sender,
TextChangedEventArgs e)
        {
            totalTextBox.Text = ((roomComboBox.SelectedItem as
room).rent*Int32.Parse(nightsTextBox.Text)).ToString();
        }

        private void Button_Click(object sender, RoutedEventArgs e)
        {
            db.books.DeleteObject((book)
booksDataGrid.SelectedItem);
        }

        private void paidButton_Click(object sender, RoutedEventArgs
e)
        {
```

```
            int lastId = db.incomes.Max(inc => inc.id);
            book book = booksDataGrid.SelectedItem as book;
            book.paid = true;
            db.SaveChanges();
            income income = db.CreateObject<income>();
            income.id = lastId + 1;
            income.room = book.room;
            income.nights = book.nights;
            income.paidTotal = book.nights*book.room.rent;
            income.paymentDate = DateTime.Now;
            db.incomes.AddObject(income);
            db.SaveChanges();
            incomesList.Add(income);
            incomesDataGrid.Items.Refresh();

        }

        private void Button_Click_1(object sender, RoutedEventArgs
e)
        {

db.incomes.DeleteObject((income)incomesDataGrid.SelectedItem);
        }

        private void tabItem6_Loaded(object sender, RoutedEventArgs
e)
        {
            int incomeSum = (int) db.incomes.Sum(inc =>
inc.paidTotal);
            Income.Content = incomeSum;
            int outgoSum = 0;
            foreach (var outgo in db.outgoes)
            {
                outgoSum += int.Parse(outgo.paidTotal);
            }
            Outgo.Content = outgoSum;

            Balance.Content = incomeSum - outgoSum;
        }

        private void Button_Click_2(object sender, RoutedEventArgs
e)
        {

        }
    }
    public abstract class BaseConverter : MarkupExtension
    {
        public override object ProvideValue(IServiceProvider
serviceProvider)
        {
            return this;
        }
```

```
    }
    [ValueConversion(typeof(object), typeof(string))]
    public class CustomerToNameConverter : BaseConverter,
IValueConverter
    {
        public object Convert(object value, Type targetType, object
parameter,
                              System.Globalization.CultureInfo culture)
        {
            customer cust = value as customer;
            if (cust == null)
                return null;
            return cust.lastName + " " +cust.firstName;

        }
        public object ConvertBack(object value, Type targetType,
object parameter,
                          System.Globalization.CultureInfo culture)
        {
            return null;
        }
    }
    [ValueConversion(typeof(object), typeof(string))]
    public class RoomToNumberConverter : BaseConverter,
IValueConverter
    {
        public object Convert(object value, Type targetType, object
parameter,
                              System.Globalization.CultureInfo culture)
        {
            room room = value as room;
            if (room == null)
                return null;
            return room.number;

        }
        public object ConvertBack(object value, Type targetType,
object parameter,
                          System.Globalization.CultureInfo culture)
        {
            return null;
        }
    }
}
```

```
using System;
using System.Data.Objects;
using System.Data.Objects.DataClasses;
using System.Data.EntityClient;
using System.ComponentModel;
```

```
using System.Xml.Serialization;
using System.Runtime.Serialization;

[assembly: EdmSchemaAttribute()]
#region EDM Relationship Metadata

[assembly: EdmRelationshipAttribute("hotelModel", "customer1", "customer",
System.Data.Metadata.Edm.RelationshipMultiplicity.ZeroOrOne, typeof(Hotel.DAL.customer),
"book", System.Data.Metadata.Edm.RelationshipMultiplicity.Many, typeof(Hotel.DAL.book), true)]
[assembly: EdmRelationshipAttribute("hotelModel", "room1", "room",
System.Data.Metadata.Edm.RelationshipMultiplicity.One, typeof(Hotel.DAL.room), "book",
System.Data.Metadata.Edm.RelationshipMultiplicity.ZeroOrOne, typeof(Hotel.DAL.book), true)]
[assembly: EdmRelationshipAttribute("hotelModel", "rooom", "room",
System.Data.Metadata.Edm.RelationshipMultiplicity.ZeroOrOne, typeof(Hotel.DAL.room), "income",
System.Data.Metadata.Edm.RelationshipMultiplicity.Many, typeof(Hotel.DAL.income), true)]

#endregion

namespace Hotel.DAL
{
   #region Contexts

   /// <summary>
   /// No Metadata Documentation available.
   /// </summary>
   public partial class hotelEntities : ObjectContext
   {
      #region Constructors

      /// <summary>
      /// Initializes a new hotelEntities object using the connection string found in the 'hotelEntities'
section of the application configuration file.
      /// </summary>
      public hotelEntities() : base("name=hotelEntities", "hotelEntities")
      {
         this.ContextOptions.LazyLoadingEnabled = true;
         OnContextCreated();
      }

      /// <summary>
      /// Initialize a new hotelEntities object.
      /// </summary>
      public hotelEntities(string connectionString) : base(connectionString, "hotelEntities")
      {
         this.ContextOptions.LazyLoadingEnabled = true;
         OnContextCreated();
      }

      /// <summary>
      /// Initialize a new hotelEntities object.
      /// </summary>
      public hotelEntities(EntityConnection connection) : base(connection, "hotelEntities")
```

```
{
    this.ContextOptions.LazyLoadingEnabled = true;
    OnContextCreated();
}

#endregion

#region Partial Methods

partial void OnContextCreated();

#endregion

#region ObjectSet Properties

/// <summary>
/// No Metadata Documentation available.
/// </summary>
public ObjectSet<book> books
{
    get
    {
        if ((_books == null))
        {
            _books = base.CreateObjectSet<book>("books");
        }
        return _books;
    }
}
private ObjectSet<book> _books;

/// <summary>
/// No Metadata Documentation available.
/// </summary>
public ObjectSet<customer> customers
{
    get
    {
        if ((_customers == null))
        {
            _customers = base.CreateObjectSet<customer>("customers");
        }
        return _customers;
    }
}
private ObjectSet<customer> _customers;

/// <summary>
/// No Metadata Documentation available.
/// </summary>
public ObjectSet<income> incomes
{
```

```
            get
            {
                if ((_incomes == null))
                {
                    _incomes = base.CreateObjectSet<income>("incomes");
                }
                return _incomes;
            }
        }
        private ObjectSet<income> _incomes;

        /// <summary>
        /// No Metadata Documentation available.
        /// </summary>
        public ObjectSet<outgo> outgoes
        {
            get
            {
                if ((_outgoes == null))
                {
                    _outgoes = base.CreateObjectSet<outgo>("outgoes");
                }
                return _outgoes;
            }
        }
        private ObjectSet<outgo> _outgoes;

        /// <summary>
        /// No Metadata Documentation available.
        /// </summary>
        public ObjectSet<room> rooms
        {
            get
            {
                if ((_rooms == null))
                {
                    _rooms = base.CreateObjectSet<room>("rooms");
                }
                return _rooms;
            }
        }
        private ObjectSet<room> _rooms;

        #endregion
        #region AddTo Methods

        /// <summary>
        /// Deprecated Method for adding a new object to the books EntitySet. Consider using the .Add
method of the associated ObjectSet&lt;T&gt; property instead.
        /// </summary>
        public void AddTobooks(book book)
        {
```

```
            base.AddObject("books", book);
        }

        /// <summary>
        /// Deprecated Method for adding a new object to the customers EntitySet. Consider using the
.Add method of the associated ObjectSet&lt;T&gt; property instead.
        /// </summary>
        public void AddTocustomers(customer customer)
        {
            base.AddObject("customers", customer);
        }

        /// <summary>
        /// Deprecated Method for adding a new object to the incomes EntitySet. Consider using the .Add
method of the associated ObjectSet&lt;T&gt; property instead.
        /// </summary>
        public void AddToincomes(income income)
        {
            base.AddObject("incomes", income);
        }

        /// <summary>
        /// Deprecated Method for adding a new object to the outgoes EntitySet. Consider using the .Add
method of the associated ObjectSet&lt;T&gt; property instead.
        /// </summary>
        public void AddTooutgoes(outgo outgo)
        {
            base.AddObject("outgoes", outgo);
        }

        /// <summary>
        /// Deprecated Method for adding a new object to the rooms EntitySet. Consider using the .Add
method of the associated ObjectSet&lt;T&gt; property instead.
        /// </summary>
        public void AddTorooms(room room)
        {
            base.AddObject("rooms", room);
        }

        #endregion
    }


    #endregion

    #region Entities

    /// <summary>
    /// No Metadata Documentation available.
    /// </summary>
    [EdmEntityTypeAttribute(NamespaceName="hotelModel", Name="book")]
    [Serializable()]
```

```
[DataContractAttribute(IsReference=true)]
public partial class book : EntityObject
{
    #region Factory Method

    /// <summary>
    /// Create a new book object.
    /// </summary>
    /// <param name="roomNumber">Initial value of the roomNumber property.</param>
    public static book Createbook(global::System.Int32 roomNumber)
    {
        book book = new book();
        book.roomNumber = roomNumber;
        return book;
    }

    #endregion
    #region Primitive Properties

    /// <summary>
    /// No Metadata Documentation available.
    /// </summary>
    [EdmScalarPropertyAttribute(EntityKeyProperty=true, IsNullable=false)]
    [DataMemberAttribute()]
    public global::System.Int32 roomNumber
    {
        get
        {
            return _roomNumber;
        }
        set
        {
            if (_roomNumber != value)
            {
                OnroomNumberChanging(value);
                ReportPropertyChanging("roomNumber");
                _roomNumber = StructuralObject.SetValidValue(value);
                ReportPropertyChanged("roomNumber");
                OnroomNumberChanged();
            }
        }
    }
    private global::System.Int32 _roomNumber;
    partial void OnroomNumberChanging(global::System.Int32 value);
    partial void OnroomNumberChanged();

    /// <summary>
    /// No Metadata Documentation available.
    /// </summary>
    [EdmScalarPropertyAttribute(EntityKeyProperty=false, IsNullable=true)]
    [DataMemberAttribute()]
    public Nullable<global::System.Int32> customerNumber
```

```
{
    get
    {
        return _customerNumber;
    }
    set
    {
        OncustomerNumberChanging(value);
        ReportPropertyChanging("customerNumber");
        _customerNumber = StructuralObject.SetValidValue(value);
        ReportPropertyChanged("customerNumber");
        OncustomerNumberChanged();
    }
}
private Nullable<global::System.Int32> _customerNumber;
partial void OncustomerNumberChanging(Nullable<global::System.Int32> value);
partial void OncustomerNumberChanged();

/// <summary>
/// No Metadata Documentation available.
/// </summary>
[EdmScalarPropertyAttribute(EntityKeyProperty=false, IsNullable=true)]
[DataMemberAttribute()]
public Nullable<global::System.DateTime> checkInDate
{
    get
    {
        return _checkInDate;
    }
    set
    {
        OncheckInDateChanging(value);
        ReportPropertyChanging("checkInDate");
        _checkInDate = StructuralObject.SetValidValue(value);
        ReportPropertyChanged("checkInDate");
        OncheckInDateChanged();
    }
}
private Nullable<global::System.DateTime> _checkInDate;
partial void OncheckInDateChanging(Nullable<global::System.DateTime> value);
partial void OncheckInDateChanged();

/// <summary>
/// No Metadata Documentation available.
/// </summary>
[EdmScalarPropertyAttribute(EntityKeyProperty=false, IsNullable=true)]
[DataMemberAttribute()]
public Nullable<global::System.Int32> nights
{
    get
    {
        return _nights;
```

```
    }
    set
    {
        OnnightsChanging(value);
        ReportPropertyChanging("nights");
        _nights = StructuralObject.SetValidValue(value);
        ReportPropertyChanged("nights");
        OnnightsChanged();
    }
}
private Nullable<global::System.Int32> _nights;
partial void OnnightsChanging(Nullable<global::System.Int32> value);
partial void OnnightsChanged();

/// <summary>
/// No Metadata Documentation available.
/// </summary>
[EdmScalarPropertyAttribute(EntityKeyProperty=false, IsNullable=true)]
[DataMemberAttribute()]
public Nullable<global::System.Boolean> paid
{
    get
    {
        return _paid;
    }
    set
    {
        OnpaidChanging(value);
        ReportPropertyChanging("paid");
        _paid = StructuralObject.SetValidValue(value);
        ReportPropertyChanged("paid");
        OnpaidChanged();
    }
}
private Nullable<global::System.Boolean> _paid;
partial void OnpaidChanging(Nullable<global::System.Boolean> value);
partial void OnpaidChanged();

#endregion

#region Navigation Properties

/// <summary>
/// No Metadata Documentation available.
/// </summary>
[XmlIgnoreAttribute()]
[SoapIgnoreAttribute()]
[DataMemberAttribute()]
[EdmRelationshipNavigationPropertyAttribute("hotelModel", "customer1", "customer")]
public customer customer
{
    get
```

```
        {
            return
((IEntityWithRelationships)this).RelationshipManager.GetRelatedReference<customer>("hotelModel.
customer1", "customer").Value;
        }
        set
        {

((IEntityWithRelationships)this).RelationshipManager.GetRelatedReference<customer>("hotelModel.
customer1", "customer").Value = value;
        }
    }
    /// <summary>
    /// No Metadata Documentation available.
    /// </summary>
    [BrowsableAttribute(false)]
    [DataMemberAttribute()]
    public EntityReference<customer> customerReference
    {
        get
        {
            return
((IEntityWithRelationships)this).RelationshipManager.GetRelatedReference<customer>("hotelModel.
customer1", "customer");
        }
        set
        {
            if ((value != null))
            {

((IEntityWithRelationships)this).RelationshipManager.InitializeRelatedReference<customer>("hotelM
odel.customer1", "customer", value);
            }
        }
    }

    /// <summary>
    /// No Metadata Documentation available.
    /// </summary>
    [XmlIgnoreAttribute()]
    [SoapIgnoreAttribute()]
    [DataMemberAttribute()]
    [EdmRelationshipNavigationPropertyAttribute("hotelModel", "room1", "room")]
    public room room
    {
        get
        {
            return
((IEntityWithRelationships)this).RelationshipManager.GetRelatedReference<room>("hotelModel.roo
m1", "room").Value;
        }
        set
```

```
            {
((IEntityWithRelationships)this).RelationshipManager.GetRelatedReference<room>("hotelModel.roo
m1", "room").Value = value;
            }
        }
        /// <summary>
        /// No Metadata Documentation available.
        /// </summary>
        [BrowsableAttribute(false)]
        [DataMemberAttribute()]
        public EntityReference<room> roomReference
        {
            get
            {
                return
((IEntityWithRelationships)this).RelationshipManager.GetRelatedReference<room>("hotelModel.roo
m1", "room");
            }
            set
            {
                if ((value != null))
                {
((IEntityWithRelationships)this).RelationshipManager.InitializeRelatedReference<room>("hotelMode
l.room1", "room", value);
                }
            }
        }

        #endregion
    }

    /// <summary>
    /// No Metadata Documentation available.
    /// </summary>
    [EdmEntityTypeAttribute(NamespaceName="hotelModel", Name="customer")]
    [Serializable()]
    [DataContractAttribute(IsReference=true)]
    public partial class customer : EntityObject
    {
        #region Factory Method

        /// <summary>
        /// Create a new customer object.
        /// </summary>
        /// <param name="id">Initial value of the id property.</param>
        public static customer Createcustomer(global::System.Int32 id)
        {
            customer customer = new customer();
            customer.id = id;
            return customer;
```

```
}

#endregion
#region Primitive Properties

/// <summary>
/// No Metadata Documentation available.
/// </summary>
[EdmScalarPropertyAttribute(EntityKeyProperty=true, IsNullable=false)]
[DataMemberAttribute()]
public global::System.Int32 id
{
   get
   {
      return _id;
   }
   set
   {
      if (_id != value)
      {
         OnidChanging(value);
         ReportPropertyChanging("id");
         _id = StructuralObject.SetValidValue(value);
         ReportPropertyChanged("id");
         OnidChanged();
      }
   }
}
private global::System.Int32 _id;
partial void OnidChanging(global::System.Int32 value);
partial void OnidChanged();

/// <summary>
/// No Metadata Documentation available.
/// </summary>
[EdmScalarPropertyAttribute(EntityKeyProperty=false, IsNullable=true)]
[DataMemberAttribute()]
public global::System.String lastName
{
   get
   {
      return _lastName;
   }
   set
   {
      OnlastNameChanging(value);
      ReportPropertyChanging("lastName");
      _lastName = StructuralObject.SetValidValue(value, true);
      ReportPropertyChanged("lastName");
      OnlastNameChanged();
   }
}
```

```
private global::System.String _lastName;
partial void OnlastNameChanging(global::System.String value);
partial void OnlastNameChanged();

/// <summary>
/// No Metadata Documentation available.
/// </summary>
[EdmScalarPropertyAttribute(EntityKeyProperty=false, IsNullable=true)]
[DataMemberAttribute()]
public global::System.String firstName
{
   get
   {
      return _firstName;
   }
   set
   {
      OnfirstNameChanging(value);
      ReportPropertyChanging("firstName");
      _firstName = StructuralObject.SetValidValue(value, true);
      ReportPropertyChanged("firstName");
      OnfirstNameChanged();
   }
}
private global::System.String _firstName;
partial void OnfirstNameChanging(global::System.String value);
partial void OnfirstNameChanged();

/// <summary>
/// No Metadata Documentation available.
/// </summary>
[EdmScalarPropertyAttribute(EntityKeyProperty=false, IsNullable=true)]
[DataMemberAttribute()]
public global::System.String passportNumber
{
   get
   {
      return _passportNumber;
   }
   set
   {
      OnpassportNumberChanging(value);
      ReportPropertyChanging("passportNumber");
      _passportNumber = StructuralObject.SetValidValue(value, true);
      ReportPropertyChanged("passportNumber");
      OnpassportNumberChanged();
   }
}
private global::System.String _passportNumber;
partial void OnpassportNumberChanging(global::System.String value);
partial void OnpassportNumberChanged();
```

```
        /// <summary>
        /// No Metadata Documentation available.
        /// </summary>
        [EdmScalarPropertyAttribute(EntityKeyProperty=false, IsNullable=true)]
        [DataMemberAttribute()]
        public global::System.String phoneNumber
        {
            get
            {
                return _phoneNumber;
            }
            set
            {
                OnphoneNumberChanging(value);
                ReportPropertyChanging("phoneNumber");
                _phoneNumber = StructuralObject.SetValidValue(value, true);
                ReportPropertyChanged("phoneNumber");
                OnphoneNumberChanged();
            }
        }
        private global::System.String _phoneNumber;
        partial void OnphoneNumberChanging(global::System.String value);
        partial void OnphoneNumberChanged();

        #endregion

        #region Navigation Properties

        /// <summary>
        /// No Metadata Documentation available.
        /// </summary>
        [XmlIgnoreAttribute()]
        [SoapIgnoreAttribute()]
        [DataMemberAttribute()]
        [EdmRelationshipNavigationPropertyAttribute("hotelModel", "customer1", "book")]
        public EntityCollection<book> books
        {
            get
            {
                return
((IEntityWithRelationships)this).RelationshipManager.GetRelatedCollection<book>("hotelModel.cust
omer1", "book");
            }
            set
            {
                if ((value != null))
                {

((IEntityWithRelationships)this).RelationshipManager.InitializeRelatedCollection<book>("hotelMode
l.customer1", "book", value);
                }
            }
```

```
    }

    #endregion
}

/// <summary>
/// No Metadata Documentation available.
/// </summary>
[EdmEntityTypeAttribute(NamespaceName="hotelModel", Name="income")]
[Serializable()]
[DataContractAttribute(IsReference=true)]
public partial class income : EntityObject
{
    #region Factory Method

    /// <summary>
    /// Create a new income object.
    /// </summary>
    /// <param name="id">Initial value of the id property.</param>
    public static income Createincome(global::System.Int32 id)
    {
        income income = new income();
        income.id = id;
        return income;
    }

    #endregion
    #region Primitive Properties

    /// <summary>
    /// No Metadata Documentation available.
    /// </summary>
    [EdmScalarPropertyAttribute(EntityKeyProperty=true, IsNullable=false)]
    [DataMemberAttribute()]
    public global::System.Int32 id
    {
        get
        {
            return _id;
        }
        set
        {
            if (_id != value)
            {
                OnidChanging(value);
                ReportPropertyChanging("id");
                _id = StructuralObject.SetValidValue(value);
                ReportPropertyChanged("id");
                OnidChanged();
            }
        }
    }
```

```
private global::System.Int32 _id;
partial void OnidChanging(global::System.Int32 value);
partial void OnidChanged();

/// <summary>
/// No Metadata Documentation available.
/// </summary>
[EdmScalarPropertyAttribute(EntityKeyProperty=false, IsNullable=true)]
[DataMemberAttribute()]
public Nullable<global::System.Int32> roomNumber
{
    get
    {
        return _roomNumber;
    }
    set
    {
        OnroomNumberChanging(value);
        ReportPropertyChanging("roomNumber");
        _roomNumber = StructuralObject.SetValidValue(value);
        ReportPropertyChanged("roomNumber");
        OnroomNumberChanged();
    }
}
private Nullable<global::System.Int32> _roomNumber;
partial void OnroomNumberChanging(Nullable<global::System.Int32> value);
partial void OnroomNumberChanged();

/// <summary>
/// No Metadata Documentation available.
/// </summary>
[EdmScalarPropertyAttribute(EntityKeyProperty=false, IsNullable=true)]
[DataMemberAttribute()]
public Nullable<global::System.Int32> nights
{
    get
    {
        return _nights;
    }
    set
    {
        OnnightsChanging(value);
        ReportPropertyChanging("nights");
        _nights = StructuralObject.SetValidValue(value);
        ReportPropertyChanged("nights");
        OnnightsChanged();
    }
}
private Nullable<global::System.Int32> _nights;
partial void OnnightsChanging(Nullable<global::System.Int32> value);
partial void OnnightsChanged();
```

```
/// <summary>
/// No Metadata Documentation available.
/// </summary>
[EdmScalarPropertyAttribute(EntityKeyProperty=false, IsNullable=true)]
[DataMemberAttribute()]
public Nullable<global::System.Int32> paidTotal
{
   get
   {
      return _paidTotal;
   }
   set
   {
      OnpaidTotalChanging(value);
      ReportPropertyChanging("paidTotal");
      _paidTotal = StructuralObject.SetValidValue(value);
      ReportPropertyChanged("paidTotal");
      OnpaidTotalChanged();
   }
}
private Nullable<global::System.Int32> _paidTotal;
partial void OnpaidTotalChanging(Nullable<global::System.Int32> value);
partial void OnpaidTotalChanged();

/// <summary>
/// No Metadata Documentation available.
/// </summary>
[EdmScalarPropertyAttribute(EntityKeyProperty=false, IsNullable=true)]
[DataMemberAttribute()]
public Nullable<global::System.DateTime> paymentDate
{
   get
   {
      return _paymentDate;
   }
   set
   {
      OnpaymentDateChanging(value);
      ReportPropertyChanging("paymentDate");
      _paymentDate = StructuralObject.SetValidValue(value);
      ReportPropertyChanged("paymentDate");
      OnpaymentDateChanged();
   }
}
private Nullable<global::System.DateTime> _paymentDate;
partial void OnpaymentDateChanging(Nullable<global::System.DateTime> value);
partial void OnpaymentDateChanged();

#endregion

#region Navigation Properties
```

```
        /// <summary>
        /// No Metadata Documentation available.
        /// </summary>
        [XmlIgnoreAttribute()]
        [SoapIgnoreAttribute()]
        [DataMemberAttribute()]
        [EdmRelationshipNavigationPropertyAttribute("hotelModel", "rooom", "room")]
        public room room
        {
            get
            {
                return
((IEntityWithRelationships)this).RelationshipManager.GetRelatedReference<room>("hotelModel.rooo
m", "room").Value;
            }
            set
            {

((IEntityWithRelationships)this).RelationshipManager.GetRelatedReference<room>("hotelModel.rooo
m", "room").Value = value;
            }
        }
        /// <summary>
        /// No Metadata Documentation available.
        /// </summary>
        [BrowsableAttribute(false)]
        [DataMemberAttribute()]
        public EntityReference<room> roomReference
        {
            get
            {
                return
((IEntityWithRelationships)this).RelationshipManager.GetRelatedReference<room>("hotelModel.rooo
m", "room");
            }
            set
            {
                if ((value != null))
                {

((IEntityWithRelationships)this).RelationshipManager.InitializeRelatedReference<room>("hotelMode
l.rooom", "room", value);
                }
            }
        }

        #endregion
    }

    /// <summary>
    /// No Metadata Documentation available.
    /// </summary>
```

```
[EdmEntityTypeAttribute(NamespaceName="hotelModel", Name="outgo")]
[Serializable()]
[DataContractAttribute(IsReference=true)]
public partial class outgo : EntityObject
{
    #region Factory Method

    /// <summary>
    /// Create a new outgo object.
    /// </summary>
    /// <param name="id">Initial value of the id property.</param>
    public static outgo Createoutgo(global::System.Int32 id)
    {
        outgo outgo = new outgo();
        outgo.id = id;
        return outgo;
    }

    #endregion
    #region Primitive Properties

    /// <summary>
    /// No Metadata Documentation available.
    /// </summary>
    [EdmScalarPropertyAttribute(EntityKeyProperty=true, IsNullable=false)]
    [DataMemberAttribute()]
    public global::System.Int32 id
    {
        get
        {
            return _id;
        }
        set
        {
            if (_id != value)
            {
                OnidChanging(value);
                ReportPropertyChanging("id");
                _id = StructuralObject.SetValidValue(value);
                ReportPropertyChanged("id");
                OnidChanged();
            }
        }
    }
    private global::System.Int32 _id;
    partial void OnidChanging(global::System.Int32 value);
    partial void OnidChanged();

    /// <summary>
    /// No Metadata Documentation available.
    /// </summary>
    [EdmScalarPropertyAttribute(EntityKeyProperty=false, IsNullable=true)]
```

```
[DataMemberAttribute()]
public global::System.String purpose
{
   get
   {
      return _purpose;
   }
   set
   {
      OnpurposeChanging(value);
      ReportPropertyChanging("purpose");
      _purpose = StructuralObject.SetValidValue(value, true);
      ReportPropertyChanged("purpose");
      OnpurposeChanged();
   }
}
private global::System.String _purpose;
partial void OnpurposeChanging(global::System.String value);
partial void OnpurposeChanged();

/// <summary>
/// No Metadata Documentation available.
/// </summary>
[EdmScalarPropertyAttribute(EntityKeyProperty=false, IsNullable=true)]
[DataMemberAttribute()]
public global::System.String paidTotal
{
   get
   {
      return _paidTotal;
   }
   set
   {
      OnpaidTotalChanging(value);
      ReportPropertyChanging("paidTotal");
      _paidTotal = StructuralObject.SetValidValue(value, true);
      ReportPropertyChanged("paidTotal");
      OnpaidTotalChanged();
   }
}
private global::System.String _paidTotal;
partial void OnpaidTotalChanging(global::System.String value);
partial void OnpaidTotalChanged();

/// <summary>
/// No Metadata Documentation available.
/// </summary>
[EdmScalarPropertyAttribute(EntityKeyProperty=false, IsNullable=true)]
[DataMemberAttribute()]
public Nullable<global::System.DateTime> paymentDate
{
   get
```

```
        {
            return _paymentDate;
        }
        set
        {
            OnpaymentDateChanging(value);
            ReportPropertyChanging("paymentDate");
            _paymentDate = StructuralObject.SetValidValue(value);
            ReportPropertyChanged("paymentDate");
            OnpaymentDateChanged();
        }
    }
    private Nullable<global::System.DateTime> _paymentDate;
    partial void OnpaymentDateChanging(Nullable<global::System.DateTime> value);
    partial void OnpaymentDateChanged();

    /// <summary>
    /// No Metadata Documentation available.
    /// </summary>
    [EdmScalarPropertyAttribute(EntityKeyProperty=false, IsNullable=true)]
    [DataMemberAttribute()]
    public global::System.String receiver
    {
        get
        {
            return _receiver;
        }
        set
        {
            OnreceiverChanging(value);
            ReportPropertyChanging("receiver");
            _receiver = StructuralObject.SetValidValue(value, true);
            ReportPropertyChanged("receiver");
            OnreceiverChanged();
        }
    }
    private global::System.String _receiver;
    partial void OnreceiverChanging(global::System.String value);
    partial void OnreceiverChanged();

    #endregion

}

/// <summary>
/// No Metadata Documentation available.
/// </summary>
[EdmEntityTypeAttribute(NamespaceName="hotelModel", Name="room")]
[Serializable()]
[DataContractAttribute(IsReference=true)]
public partial class room : EntityObject
{
```

```
#region Factory Method

/// <summary>
/// Create a new room object.
/// </summary>
/// <param name="number">Initial value of the number property.</param>
public static room Createroom(global::System.Int32 number)
{
    room room = new room();
    room.number = number;
    return room;
}

#endregion
#region Primitive Properties

/// <summary>
/// No Metadata Documentation available.
/// </summary>
[EdmScalarPropertyAttribute(EntityKeyProperty=true, IsNullable=false)]
[DataMemberAttribute()]
public global::System.Int32 number
{
    get
    {
        return _number;
    }
    set
    {
        if (_number != value)
        {
            OnnumberChanging(value);
            ReportPropertyChanging("number");
            _number = StructuralObject.SetValidValue(value);
            ReportPropertyChanged("number");
            OnnumberChanged();
        }
    }
}
private global::System.Int32 _number;
partial void OnnumberChanging(global::System.Int32 value);
partial void OnnumberChanged();

/// <summary>
/// No Metadata Documentation available.
/// </summary>
[EdmScalarPropertyAttribute(EntityKeyProperty=false, IsNullable=true)]
[DataMemberAttribute()]
public global::System.String beds
{
    get
    {
```

```
        return _beds;
    }
    set
    {
        OnbedsChanging(value);
        ReportPropertyChanging("beds");
        _beds = StructuralObject.SetValidValue(value, true);
        ReportPropertyChanged("beds");
        OnbedsChanged();
    }
}
private global::System.String _beds;
partial void OnbedsChanging(global::System.String value);
partial void OnbedsChanged();

/// <summary>
/// No Metadata Documentation available.
/// </summary>
[EdmScalarPropertyAttribute(EntityKeyProperty=false, IsNullable=true)]
[DataMemberAttribute()]
public global::System.String rooms
{
    get
    {
        return _rooms;
    }
    set
    {
        OnroomsChanging(value);
        ReportPropertyChanging("rooms");
        _rooms = StructuralObject.SetValidValue(value, true);
        ReportPropertyChanged("rooms");
        OnroomsChanged();
    }
}
private global::System.String _rooms;
partial void OnroomsChanging(global::System.String value);
partial void OnroomsChanged();

/// <summary>
/// No Metadata Documentation available.
/// </summary>
[EdmScalarPropertyAttribute(EntityKeyProperty=false, IsNullable=true)]
[DataMemberAttribute()]
public Nullable<global::System.Int32> rent
{
    get
    {
        return _rent;
    }
    set
    {
```

```
            OnrentChanging(value);
            ReportPropertyChanging("rent");
            _rent = StructuralObject.SetValidValue(value);
            ReportPropertyChanged("rent");
            OnrentChanged();
        }
    }
    private Nullable<global::System.Int32> _rent;
    partial void OnrentChanging(Nullable<global::System.Int32> value);
    partial void OnrentChanged();

    /// <summary>
    /// No Metadata Documentation available.
    /// </summary>
    [EdmScalarPropertyAttribute(EntityKeyProperty=false, IsNullable=true)]
    [DataMemberAttribute()]
    public global::System.String status
    {
        get
        {
            return _status;
        }
        set
        {
            OnstatusChanging(value);
            ReportPropertyChanging("status");
            _status = StructuralObject.SetValidValue(value, true);
            ReportPropertyChanged("status");
            OnstatusChanged();
        }
    }
    private global::System.String _status;
    partial void OnstatusChanging(global::System.String value);
    partial void OnstatusChanged();

    #endregion

    #region Navigation Properties

    /// <summary>
    /// No Metadata Documentation available.
    /// </summary>
    [XmlIgnoreAttribute()]
    [SoapIgnoreAttribute()]
    [DataMemberAttribute()]
    [EdmRelationshipNavigationPropertyAttribute("hotelModel", "room1", "book")]
    public book book
    {
        get
        {
```

```
            return
((IEntityWithRelationships)this).RelationshipManager.GetRelatedReference<book>("hotelModel.roo
m1", "book").Value;
        }
        set
        {

((IEntityWithRelationships)this).RelationshipManager.GetRelatedReference<book>("hotelModel.roo
m1", "book").Value = value;
        }
    }
    /// <summary>
    /// No Metadata Documentation available.
    /// </summary>
    [BrowsableAttribute(false)]
    [DataMemberAttribute()]
    public EntityReference<book> bookReference
    {
        get
        {
            return
((IEntityWithRelationships)this).RelationshipManager.GetRelatedReference<book>("hotelModel.roo
m1", "book");
        }
        set
        {
            if ((value != null))
            {

((IEntityWithRelationships)this).RelationshipManager.InitializeRelatedReference<book>("hotelModel
.room1", "book", value);
            }
        }
    }

    /// <summary>
    /// No Metadata Documentation available.
    /// </summary>
    [XmlIgnoreAttribute()]
    [SoapIgnoreAttribute()]
    [DataMemberAttribute()]
    [EdmRelationshipNavigationPropertyAttribute("hotelModel", "rooom", "income")]
    public EntityCollection<income> incomes
    {
        get
        {
            return
((IEntityWithRelationships)this).RelationshipManager.GetRelatedCollection<income>("hotelModel.ro
oom", "income");
        }
        set
        {
```

```
            if ((value != null))
            {

((IEntityWithRelationships)this).RelationshipManager.InitializeRelatedCollection<income>("hotelMo
del.rooom", "income", value);
            }
        }
    }

    #endregion
  }

  #endregion

}
```