НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Кафедра автоматики та управління в технічних системах

Автоматизована система управління готелем

«Flying Dutchman Service»

Курсова робота

З дисципліни «Об’єктно-орієнтоване програмування»

Керівник Виконавець

Хмелюк В.С. ст. Рубанович А.П.

«Допущена до захисту» зал. Книжка № ІТ-4120

гр. ІТ-41

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Особистий підпис керівника) (Особистий підпис виконавця)

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2016 р.

Захищений з оцінкою

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(оцінка)

Члени комісії:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Особистий підпис) (Розшифровка підпису)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Особистий підпис) (Розшифровка підпису)

Київ - 2016

#### Анотація

В курсовій роботі розглядається питання створення автоматизованої системи для управління готелем – «Flying Dutchman Service». Програма має на меті управління технічними й економічними об'єктами для розв’язання інформаційних завдань управління. Був створено приємний та зручний для користувача інтерфейс програми.

Програма реалізована мовою програмування високого рівня C# з використанням принципів об’єктно орієнтованого програмування в середовищі Microsoft Visual Studio 2015 із застосуванням Windows Forms.

Пояснювальна записка виконана на 54-ох сторінках, містить 3 додатки, 14 рисунків та 1 таблицю.

ЗМІСТ

[ВСТУП 6](#_Toc420395112)

[1 ПОСТАНОВКА ЗАДАЧІ 7](#_Toc420395113)

[2 СТРУКТУРА ПРОГРАМИ 8](#_Toc420395114)

[3 ОГЛЯД ІСНУЮЮЧИХ РІШЕНЬ 10](#_Toc420395115)

[3.1 Аналіз аналогічних програм 10](#_Toc420395116)

[3.1.1 Комплекс автоматизації готельного бізнесу «KEI-Hotel» ……...10](#_Toc420395117)

[3.1.2 Система «Fidelio FO» (Front Office) ……………………………..11](#_Toc420395118)

[3.2 Використання методів ООП ……………………………………………..14](#_Toc420395120)

[4 ПЕРЕЛІК ТА ПРИЗНАЧЕННЯ РЕЖИМІВ І ОПИС ДІАЛОГІВ 17](#_Toc420395121)

[5 СТРУКТУРА ДАНИХ І РЕСУРСІВ ПРОГРАМИ 21](#_Toc420395122)

[6 ОПИС ПРОГРАМИ 24](#_Toc420395123)

[7 КЕРІВНИЦТВО ПРОГРАМІСТА 26](#_Toc420395124)

[8 КЕРІВНИЦТВО КОРИСТУВАЧА 27](#_Toc420395125)

[ВИСНОВКИ 30](#_Toc420395126)

[ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ 31](#_Toc420395127)

[ДОДАТКИ 32](#_Toc420395128)

ДОДАТОК А - ТЕКСТ ПРОГРАМИ……………………………………………..32

ДОДАТОК Б - ДІАГРАМА ЗВ'ЯЗКІВ…..……………………………………….53

ДОДАТОК В - ДІАГРАМА КЛАСІВ……………………………………………54

# Вступ

Автоматизована система управління (АСУ) – це система, що ґрунтується на комплексному використанні технічних, математичних, інформаційних та організаційних засобів для управління складними технічними й економічними об'єктами. Саме такі засоби і використовує створена програма.

Сьогодні існує чимало реалізацій автоматизованої системи управління готелем, однак більшість з них створені для одного конкретного готелю, реалізуючи саме його послуги, таким чином унеможливлюючи використання програми в іншому закладі.

Метою курсової роботи є розробка власної системи для управління готелем, яка буде зручною та інтуїтивно зрозумілою для користувача. На відміну від заплутаного функціоналу, який не дає можливості працювати з програмою непідготовленому користувачу, дана система з точною та чітко сформульованою навігацією є легкою для використання абсолютно кожним користувачем.

Програмі дано назву “Flying Dutchman Service” що з англійської означає «Система Летючий Голландець». Розробником виступає Рубанович Анастасія.

Основою для написання програми можна вважати будь-яку автоматизовану систему управління готелем.

Програму реалізовано з використанням можливостей мови C#. Також використано головні принципи ООП [1].

# Постановка задачі

Головною метою програми є створення автоматизованої системи управління готелем.

Також під час виконання курсової роботи планується виконати такі задачі:

* реалізувати можливість бронювання та зняття броні заданого номеру;
* надати можливість оплати проживання в заданому номері за певний проміжок часу;
* організувати зручний інтерфейс для доступу до вищезгаданих елементів;
* запропонувати користувачам ознайомитись з інструкцією, щоб не допустити можливих помилок при використанні програми;
* створити механізм, який дозволить користувачу залишати відгук після перебування в готелі;
* надати можливість користувачу ознайомитися з основними можливостями програми;
* організувати пошук клієнта за відповідними параметрами, наприклад, ім’я, прізвище та по батькові;
* створити механізм, який дозволить користувачу змінювати або додавати бажану інформацію;
* реалізувати можливість в’їзду та виїзду з номеру певним відвідувачем на визначений проміжок часу;
* запропонувати користувачу зручний пошук кімнати за різними параметрами – комфорт, тип та ціна кімнати;
* реалізувати можливість збереження введених користувачем даних після його виходу з програми.

# структура програми

Програма повинна бути орієнтована на непідготовленого, тобто пересічного користувача, також мають бути реалізовані основні функції: додання клієнта, який буде проживати у вибраному номері певний проміжок часу, пошук відвідувача або номеру за вказаними параметрами, доступ до опису програми.

Щоб розпочати роботу необхідно:

* переглянути пункти навігаційного меню;
* вибрати необхідний пункт, в залежності від поставленого завдання;
* виконати бажану дію у відповідності з елементами форми;
* натиснути кнопку «Назад» для повернення до головного меню;

Щоб переглянути довідку програми необхідно:

* вибрати з меню «About»;
* ознайомитись з правилами та інформацією про програму;
* закрити вікно, щоб повернутись до головного меню;

Щоб ознайомитись з основними можливостями програми потрібно:

* вибрати в навігаційному меню пункт «About»;
* перечитати вкладений текст;
* закрити вікно.

Одразу важливо виділити головні класи для реалізації функціоналу програми – Person, Review, Room; а також додаткові – RoomPerson, Base, Comfort – для зберігання необхідної інформації та реалізації зв’язків між класами. Всі вони містять необхідні властивості та методи для роботи з екземплярами вищевказаних класів. Всі виключення та помилки, які можуть виникнути при роботі з програмою також були завбачені.

Робота всіх методів у вищенаведених класах проводиться з інформацією, яка доступна користувачеві по замовчуванню, а також з тою, яку він додасть сам.

Для реалізації довідки та опису програми була створена окрема Windows Form, функціонал якої надає користувачу можливість ознайомитись з правилами до використання, переглянути всі можливості програми та основну інформацію про неї.

Доступ до вищеописаних елементів організований через зручний та приємний для використання інтерфейс, який реалізований за допомогою Windows Form «MainWindow». Тут реалізоване головне меню для роботи з програмою та описаний функціонал всіх його пунктів. Саме «MainWindow» надає користувачу можливість зручної навігації, всі інші вікна мають кнопку «Назад», яка повертає користувача до головного вікна.

Вся інформація, змінена в одному вікні автоматично передається в інше, при цьому перезавантаження програми не є необхідним.

# Огляд існуючих рішень

## Аналіз аналогічних програм

На даний момент створено велику кількість автоматизованих систем управління готелем[2]. У цьому розділі розглянуті найбільш доречні з них. Порівняння цих програм оцінюється за наступними параметрами:

* зручність інтерфейсу;
* можливість використання програми будь-яким користувачем;
* сумісність із новими операційними системами та можливостями ПК;
* простота у користуванні;
* затрати комп’ютерних ресурсів на виконання програми;
* швидкодія та продуктивність;
* безкоштовність та легкість в отриманні програми.

### Комплекс автоматизації готельного бізнесу «KEI-Hotel»[3]

Найкращим прикладом, що яскраво може продемонструвати автоматизовану систему управління готелем є програма «KEI-Hotel» (рис. 3.1).

Головними перевагами даної системи є:

* зручний для користувача інтерфейс програми, що дозволяє за лічені секунди знайти потрібні йому функції чи розділи;
* широкий вибір функцій та їхніх опцій;
* використання програми по ролях;

Натомість дана програма має наступні недоліки:

* програма призначена лише для одного конкретного готелю;
* великі затрати системних ресурсів;
* неможливість зміни мови.

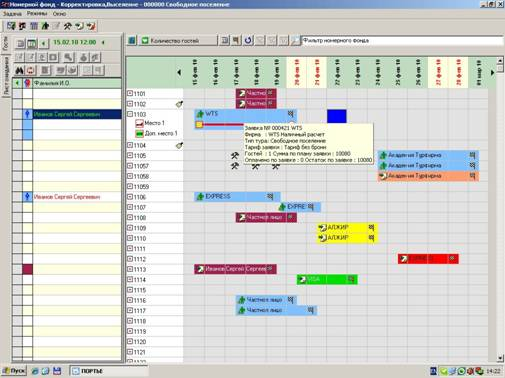


Рисунок 3.1 – Головне вікно «KEI-Hotel»

### Аавтоматизована система «Fidelia FO» (Front Office)[4]

Ця програма досить схожа на розглянуту вище, проте має свої особливості.

Позитивні якості «Fidelia FO» (рис. 3.2) в наступному:

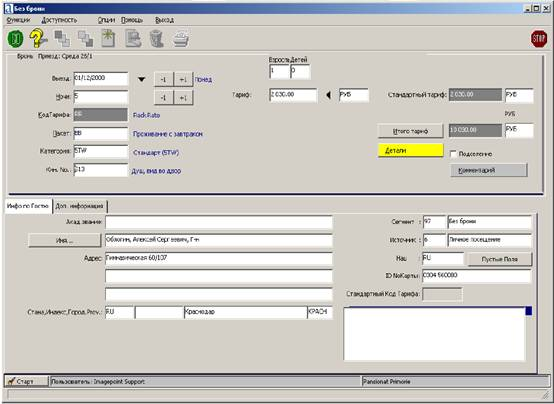
- можливість зберігання доданої інформації у файлі;

- можливість пошуку за різними параметрами;

- сумісність зі всіма операційними системами;

Проте наявні і недоліки:

* складний для користування та розуміння інтерфейс;
* неможливість зміни мови;
* незручне навігаційне меню;
* повна версія є платною.



### Рисунок 3.2 - Загальний вигляд системи «Fidelia FO»

Підсумковий аналіз розглянутих програм

Аналіз вище переглянутих програм показав позитивні та негативні якості кожної з них. На основі цих факторів складена наведена нижче таблиця порівняння цих програм та «Flying Dutchman Service» (таблиця 3.1).

Таблиця 3.1 – Порівняльний аналіз АСУ та проектної роботи

|  |  |  |  |
| --- | --- | --- | --- |
| *Критерій оцінки* | «KEI-Hotel» | **«**Fidelia FO» | «Flying Dutchman  Service» |
| Зручність інтерфейсу | + | - | + |
| Можливість використання будь-яким користувачем | + | - | + |
| Сумісність із новими операційними системами та можливостями ПК | + | + | + |
| Простота у користуванні | + | - | + |
| Малі затрати комп’ютерних ресурсів | - | - | + |
| Швидкодія та продуктивність | - | + | + |
| Безкоштовність та легкість отримання програми | + | - | + |

## Вищенаведена таблиця показує, що програма «Flying Dutchman Service» має низку переваг над програмами аналогами, а саме:

* зручність інтерфейсу;
* можливість використання будь-яким корисутвачем;
* малі затрати комп’ютених ресурсів;
* безкоштовність;
* простота у користуванні.

## Використання методів об’єктно-орієнтованого програмування

Програмування — це процес проектування, написання, тестування і підтримки комп'ютерних програм. Програмування поєднує в собі елементи інженерії (існує навіть відповідна спеціальна галузь інженерії — програмна інженерія (англ. software engineering), фундаментальних наук (перш за все математики) і мистецтва [5].

У вужчому значенні програмування розглядається як кодування — реалізація у вигляді програми одного чи кількох взаємопов'язаних алгоритмів (у сучасних умовах це здійснюється з застосуванням мов програмування). У ширшому сенсі процес програмування охоплює і створення, тобто розробку, алгоритмів, і аналіз потреб майбутніх користувачів програмного забезпечення.

У широкому значенні програмування використовується у значенні створення програми дій або алгоритмів та навчання людей або пристроїв діяти за алгоритмами.

Об'єктно-орієнтоване програмування (ООП) — одна з парадигм програмування, яка розглядає програму як множину «об'єктів», що взаємодіють між собою. Одною з переваг ООП є краща модульність програмного забезпечення (тисячу функцій процедурної мови, в ООП можна замінити кількома десятками класів із своїми методами). Попри те, що ця парадигма з'явилась в 1960-тих роках, вона не мала широкого застосування до 1990-тих, коли розвиток комп'ютерів та комп'ютерних мереж дозволив писати надзвичайно об'ємне і складне програмне забезпечення, що змусило переглянути підходи до написання програм. Сьогодні багато мов програмування або підтримують ООП або ж є цілком об'єктно-орієнтованими (зокрема, Java, C#, C++, Python, PHP, Ruby та Objective-C, ActionScript 3) [6].

Об'єктно-орієнтоване програмування сягає своїм корінням до створення мови програмування Симула в 1960-тих роках, одночасно з посиленням дискусій про кризу програмного забезпечення. Разом із тим, як ускладнювалось апаратне та програмне забезпечення, було дуже важко зберегти якість програм. Об'єктно-орієнтоване програмування частково розв'язує цю проблему шляхом наголошення на модульності програми.

На відміну від традиційних поглядів, коли програму розглядали як набір підпрограм, або як перелік інструкцій комп'ютеру, ООП програми можна вважати сукупністю об'єктів. Відповідно до парадигми об'єктно-орієнтованого програмування, кожний об'єкт здатний отримувати повідомлення, обробляти дані, та надсилати повідомлення іншим об'єктам. Кожен об'єкт — своєрідний незалежний автомат з окремим призначенням та відповідальністю.

Інкапсуляція — один з трьох основних механізмів об'єктно-орієнтованого програмування. Йдеться про те, що об’єкт вміщує не тільки дані, але і правила їх обробки, оформлені в вигляді виконуваних фрагментів(методів). А також про те, що доступ до стану об'єкта напряму заборонено, і ззовні з ним можна взаємодіяти виключно через заданий інтерфейс (відкриті поля та методи), що дозволяє знизити зв'язність. Таким чином контролюються звернення до полів класів та їхня правильна ініціалізація, усуваються можливі помилки пов'язані з неправильним викликом методу . Оскільки користувачі працюють лише через відкриті елементи класів, то розробники класу можуть як завгодно змінювати всі закриті елементи і навіть перейменовувати та видаляти їх, не турбуючись, що хтось використовує їх у своїх програмах [7].

В [об'єктно-орієнтованому програмуванні](https://uk.wikipedia.org/wiki/%D0%9E%D0%B1%27%D1%94%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D1%96%D1%94%D0%BD%D1%82%D0%BE%D0%B2%D0%B0%D0%BD%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F), наслідування — механізм утворення нових [класів](https://uk.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B0%D1%81_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)) на основі використання вже існуючих. При цьому властивості і функціональність батьківського класу переходять до класу нащадку (дочірнього класу).

Поліморфізм — концепція в [програмуванні](https://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F) та [теорії типів](https://uk.wikipedia.org/w/index.php?title=%D0%A2%D0%B5%D0%BE%D1%80%D1%96%D1%8F_%D1%82%D0%B8%D0%BF%D1%96%D0%B2&action=edit&redlink=1), відповідно до якої використовується спільний інтерфейс для обробки різних спеціалізованих [типів](https://uk.wikipedia.org/wiki/%D0%A2%D0%B8%D0%BF_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85), тобто здатність функції обробляти дані різних типів. Визначення – “один інтерфейс – багато реалізацій”.

# Перелік і призначення режимів та опис діалогу

Інтерфейс програми створений у зручному для звичайного користувача стилі. Створена практична навігація між формами, що дозволяє краще розділити всі необхідні функції.

Простий інтерфейс, меню та інструкція дозволяють навіть недосвідченим користувачам освоїти програму за лічені хвилини. Система «Flying Dutchman Service» має один режим - режим користувача.

Одразу після запуску програми користувачу одразу відкривається головне меню (рис. 4.1). Сюди можна потрапити з будь якого місця додатку – закриття будь-якого діалогового, або тестового вікна повертає доступ до головного вікна – за допомогою кнопки «Назад», звідки зручно розпочинати роботу з програмою.

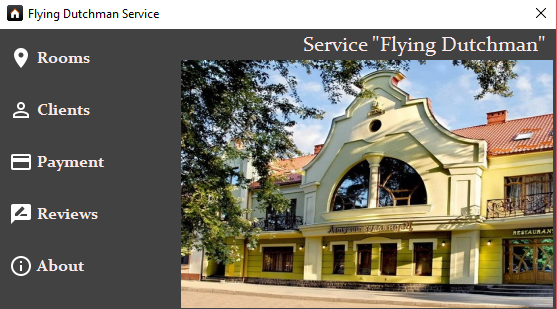


Рисунок 4.1 – Головне меню

Вибравши в навігаційному меню пункт «Rooms», користувачу відкривається нова форма (рис 4.2), де він за допомогою відповідних кнопок може забронювати або зайняти кімнату, зняти бронь або звільнити кімнату, а також здійснити пошук за необхідними параметрами.

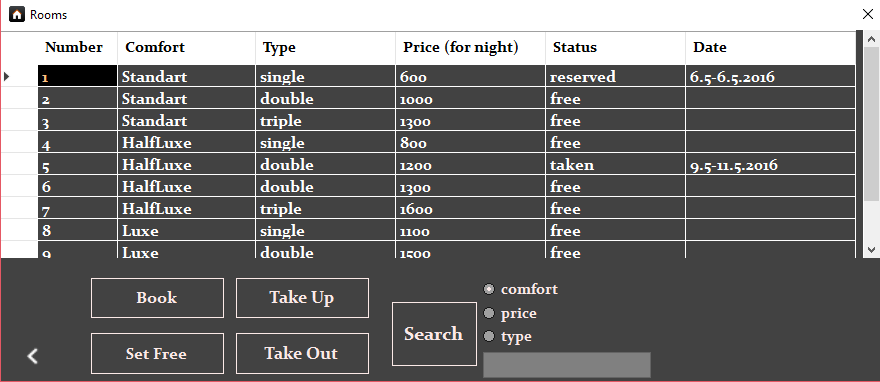


Рисунок 4.2 – Вікно «Rooms»

Пункт меню «Clients» відкриває відповідне вікно (рис. 4.3), де користувач може здійснити пошук по імені відвідувача, а також додати нового.

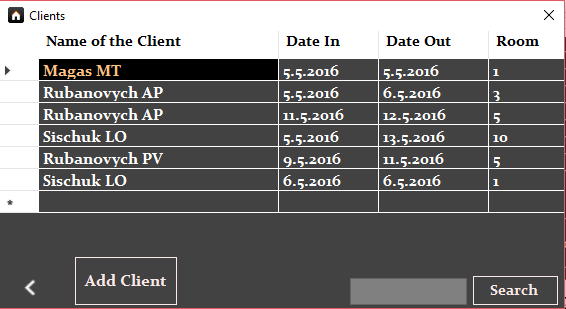


Рисунок 4.3 – Вікно «Clients»

З головного меню також можна перейти до вікна «Payment» (рис. 4.4), де користувач може оплатити проживання в кімнаті за певний проміжок часу та отримати чек.

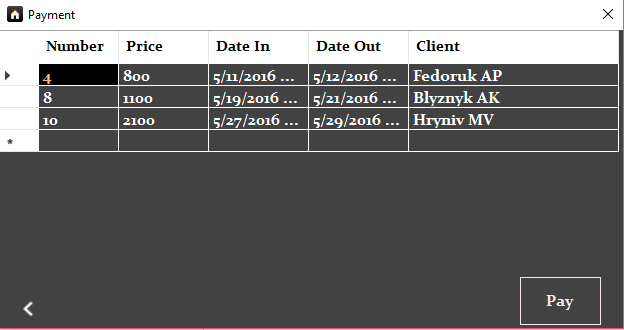


Рисунок 4.4 – Вікно «Payment»

У вікні «Reviews» (рис. 4.5) користувач може отримати доступ до відгуків відвідувачів, а також додати свою власну рецензію.

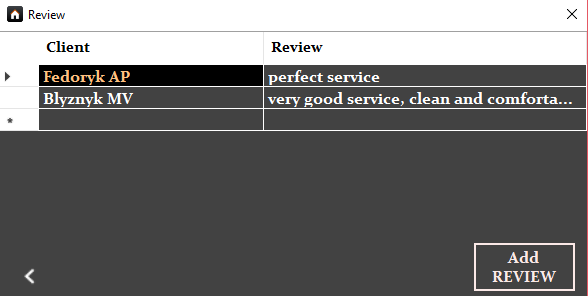


Рисунок 4.5 – Вікно «Reviews»

Важливим вікном програми є вікно «About» (рис. 4.6), де користувач має можливість ознайомитись з основною інформацією про програму та її функціями, а також дізнатись поточну версію продукту та звернутись по технічну допомогу по вказаному номеру.



Рисунок 4.6 – Вікно «About»

# Структура даних та ресурсів програми

Властивості об’єктів, а також всі алгоритми та методи містяться в трьох головних класах – Room, Person, Review.

Клас Room містить наступні поля та властивості, які їм відповідають:

* властивість Number – для позначення номеру кімнати;
* властивість Comfort – для визначення типу кімнати за комфортом (стандарт, люкс, напівлюкс);
* властивість Quantity – для позначення максимальної кількості людей, які можуть проживати в кімнати;
* властивість Price – для визначення ціни кімнати;
* властивість Accessibility – для позначення того, чи є кімната доступною на даний момент чи ні.

В класі також реалізовані методи:

* SerchComfort – для пошуку кімнати по заданому параметру, який характеризуй комфортність кімнати;
* SerachPrice – для пошуку кімнати по параметру ціни;
* SearchType – для пошуку кімнати по параметру, який характеризує тип кімнати.

Клас Person містить властивості:

* властивість Name – для зазначення імені та прізвища відвідувача;
* властивість Room – для позначення того в якій кімнаті, проживав клієнт протягом певного проміжку часу;
* властивість DateIn – для визначення дати заїзду відвідувача;
* властивість DateOut – для визначення дати виїзду відвідувача;
* властивість Payed – для позначення того, чи заплатив клієнт за проживання в кімнаті.

Клас також реалізує методи:

* Search – для пошуку людини по параметру імені;
* Pay – для можливості оплати відвідувачем проживання в кімнаті за певний проміжок часу.

Клас Review містить наступні поля та властивості, які їм відповідають:

* властивість Person – для позначення відвідувача, який залишив відгук про готель;
* властивість ReviewText – для запису тексту відгуку, який був залишений клієнтом.

Весь набір даних про об’єкти знаходиться в класі Base. Тут також є Dictionary, куди за допомогою параметра користувач може додавати об’єкти різних класів. Саме за допомогою класу Base користувач може додавати та зберігати нові об‘єкти.

Клас RoomPerson створений для реалізації зв’язку «багато до багатьох» між класами Room та Person. Клас містить дві властивості: Room – для позначення кімнати та Person – для позначення людини, яка проживала в цій кімнаті.

Також було створене додаткове перерахування (енумератор) Comfort, за допомогою якого система визначає комфортність кімнати – стандарт, напівлюкс, люкс.

В цих класах реалізовані всі алгоритми для зручної роботи з програмою, а також завбачені всі виключення та помилки, які можуть виникнути при роботі з програмою.

Можливість зберігати введені користувачем дані після закриття програми реалізована за допомогою двох класів: Message i Serializer. Клас Message містить дві властивості: Name та Data. В класі Serializer реалізовані два методи: Serialize та Deserialize для серіалізації (збереження нової інформації) та десеріалізації (для відображення збереженої інформації) відповідно.

Робота всіх методів у вищенаведених класах проводиться з інформацією, яка доступна користувачеві по замовчуванню, а також з тою, яку він додасть сам.

Для реалізації довідки та опису програми була створена окрема Windows Form, функціонал якої надає користувачу можливість ознайомитись з правилами до використання, переглянути всі можливості програми, основну інформацію про неї, а також поточну версію додатку, телефон технічної підтримки та контакти розробника додатку.

Доступ до вищеописаних елементів організований через зручний та приємний для використання інтерфейс, який реалізований за допомогою Windows Forms. Вікно «MainWindow» було створене для реалізації навігаційного меню для зручної роботи з програмою. Саме з його допомогою здійснюється вся навігація по програмі, головні вікна мають кнопку «Назад», яка повертає користувача до головного вікна. Тут описаний функціонал всіх кнопок та інших елементів управління.

Вся інформація, змінена в одному вікні автоматично передається в інше, при цьому перезавантаження програми не є необхідним.

# Опис програми

Автоматизована система управління готелем «Flying Dutchman Service» має ряд особливостей. Однією з таких особливостей є те, що додаток використовує програмний Framework WCF, відправляючи повідомлення сервісу в асинхронному режимі. WCF робить можливою побудову безпечної, надійної і транзакційної системи через спрощену уніфіковану програмну модель.

Клас служби WCF, який не може існувати самостійно, знаходиться під управлінням хостового процесу Windows, який використовує IIS (Internet Information Service).

Додаток використовує бібліотеку System.ServiceModel, яка надає класи, які зв’язані з моделлю служби.

Бібліотека System.Runtime.Serialization містить класи, які були використані для реалізації серіалізації та десеріалізації об’єктів. Серіалізація – це процес переробки об’єкта в лінійну послідовність байтів для збереження або передачі. Десеріалізація – це процес отримання даних і відновлення із них об‘єктів.

System.Xml.Linq – бібліотека, яка також була застосована при написанні програми, містить класи для Linq to Xml. Linq to Xml є інтерфейсом програмування Xml в пам’яті, який дозволяє легко та ефективно змінювати документи Xml.

Програма створена на основі автоматизованих системи управління готельним бізнесом[8] з урахування особливих потреб користувача і виконує такі функції:

* дозволяє користувачу додати відвідувача до вже існуючого списку;
* реалізовано можливість забронювати або зайняти номер, а також зняти бронь та звільнити кімнату;
* створено можливість оплатити проживання в номері за певний проміжок часу та отримати чек;
* користувач може ознайомитись з відгуками відвідувачів, а також написати власну рецензію;
* реалізовано можливість пошуку необхідних елементів по заданих параметрах;
* створене зручне та інтуїтивно зрозуміле навігаційне меню для використання програми;
* реалізований приємний та зручний для використання інтерфейс.

Крім того в майбутньому планується внести такі зміни та додати наступні функції:

* поділ фунціоналу системи по ролях;
* додання параметрів для пошуку необхідної інформації;
* можливість вибору мови;
* портація програми на мобільні пристрої (Android, Windows Phone, iPhone).

# Керівництво програміста

Програма створювалась на мові C# у середовищі Visual Studio 2015.

Сторонніх зовнішніх бібліотек або драйверів у програмі не використовувалось. До додаткових ресурсів належать файли загального типу, що розташовані у кореневій теці, та файли бінарного формату, що створюються при першому запуску програми.

Мінімальні вимоги до апаратного забезпечення:

* процесор Pentium II 850 МГц;
* пам'ять 256 МБ ОЗП;
* мінімум 25 МБ пам’яті на жорсткому диску;
* VGA-монітор з роздільною здатністю 800x600 пікселів або вищим.

Мінімальні вимоги до програмного забезпечення:

* операційна система Windows 7;
* встановлений Microsoft .NET Framework 4.5[9].

Рекомендовані вимоги до апаратного забезпечення:

* процесор Intel Core i3 1.0 GHz;
* пам’ять 512МБ ОЗП;
* 2 ГБ вільної пам’яті на жорсткому диску;
* VGA-монітор з роздільною здатністю 1024х768 пікселів.

Рекомендовані вимоги до програмного забезпечення:

* операційна система Windows 8 або Windows 10;
* встановлений Microsoft .NET Framework 4.6.

Інструкція щодо інсталяції програми: потрібно скопіювати файл з розширенням .exe з диску та запустити .exe-файл.

Інструкція до деінсталяції: видалити файл з розширенням .exe.

# Керівництво користувача

Після запуску програми відкривається головне меню системи (рис. 8.1), яке забезпечує користувача зручною та зрозумілою навігацією по системі, а також необхідною взаємодією між різними формами програми.

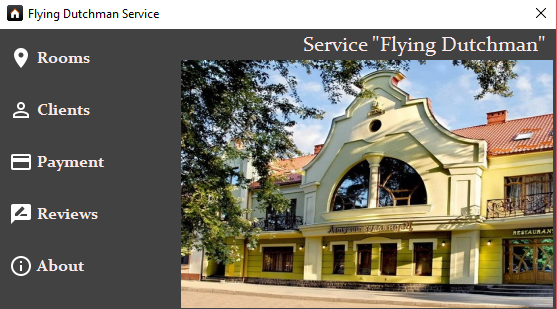
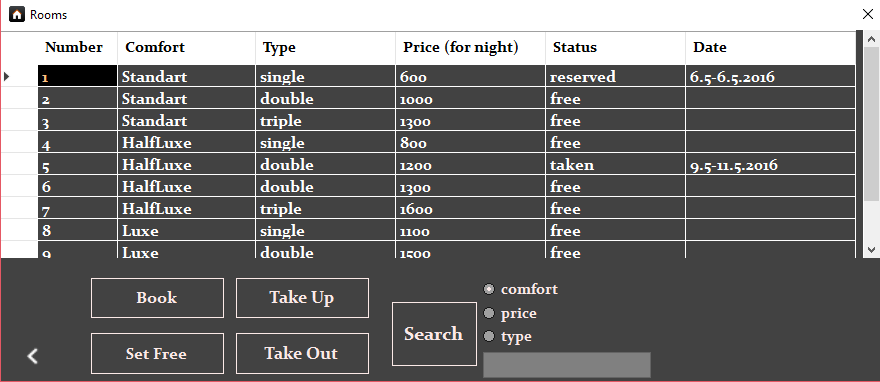


Рисунок 8.1 – Головне меню програми «Flying Dutchman Service»

Вікно «Rooms» (рис. 8.2) надає можливість ознайомитись зі списком номерів та їхніми характеристиками, а також забронювати або зайняти номер, зняти бронь або звільнити кімнату та виконати пошук, використовуючи необхідні параметри.

  
Рисунок 8.2 – Вікно «Rooms»

Форма «Clients» (рис. 8.3) створена для роботи з гостями готелю, тут відображається список клієнтів, а також користувач може додати нового або відшукати потрібного відвідувача за іменем.

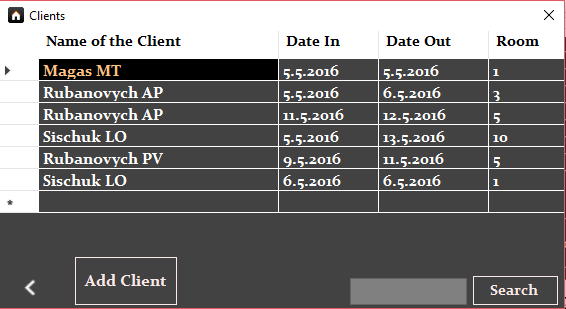


Рисунок 8.3 – Вікно «Clients»

У вікні «Payment» реалізована функція оплати кімнати за певний проміжок часу, а також отримання чеку.

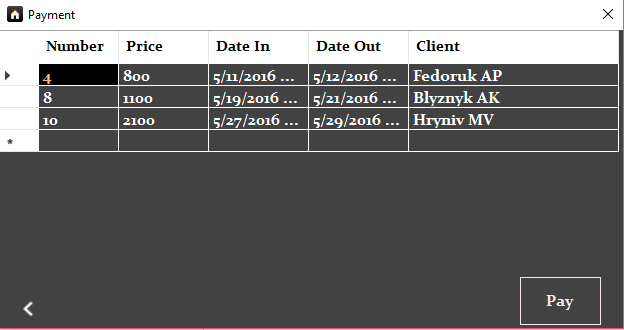


Рисунок 8.4 – Форма «Payment»

Натиснувши на пункт «Reviews», користувач відкриє відповідне вікно (рис. 8.5), яке дозволить ознайомитись з відгуками відвідувачів, а також додати власну рецензію.

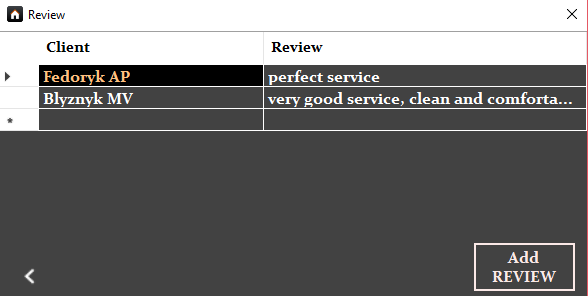


Рисунок 8.5 – Вікно «Review»

Форма «About» (рис. 8.6) надасть користувачу всю необхідну інформацію про програму – її функції та можливості, поточну версію продукту, контакти розробника та телефон технічної підтримки.



Рисунок 8.6 – Вікно «About»

# ВИсновки

В результаті виконання курсової роботи була створена автоматизована система управління готелем “Flying Dutchman Service”. Були застосовані основні принципи ООП, завдяки яким значно спростився процес написання програми. В результаті досягнуті такі цілі:

* розглянуто подібні програми, проаналізовано їх недоліки та враховано їх при написанні власної проектної роботи;
* реалізована додання клієнтів до програми;
* створена функція збереження введеної користувачем інформації;
* надана користувачу можливість виконувати пошук елементів по заданих параметрах;
* реалізовано додання відгуків до програми;
* розроблене навігаційне меню, яке дозволяє користувачу безперешкодно переходити з одного вікна в інше;
* організований зручний інтерфейс для роботи з програмою;
* надані мінімальні та рекомендовані вимоги до апаратного та програмного забезпечення для використання програми;
* розглянуті головні класи додатку, а також їхні властивості та функціонал;
* забезпечена можливість ознайомлення з правилами використання програми та основною інформацією про неї.

# Перелік використаних джерел

1. C # 2010: повний курс - Уотсон Карлі., Нейгел Крістіан, Якоб Хаммер Педерсен, Джон Рид, Морган Скиннер 2010. [Електронний курс]. – Режим

доступу: URL :

<http://padabum.com/d.php?id=16898>.

1. Автоматизовані системи управління [Електронний ресурс]. – Режим доступу: URL:

<http://tourfaq.net/hotel-business/avtomatizirovannye-sistemy-upravleniyа>

1. KEI-Company – розробник системи «KEI-Hotel» [Електронний ресурс]. – Режим доступу: URL:

<http://www.kei.ru/software/kei-hotel/default.aspx?id=page1.htm>

1. MICROS-Fidelio – розробник системи «Fidelia FO» [Електронний ресурс]. – Режим доступу: URL:

http://www.micros-fidelio.eu/en/Solutions/Products-N-Z/Suite8-System.aspx

1. Основні принципи об’єктно-орієнтованого програмування [Електронний ресурс] . – Режим доступу: URL:

<http://professorweb.ru/my/csharp/charp_theory/level3/3_1.php>

1. Об'єктно-орієнтоване програмування C # [Електронний курс]. – Режим

доступу: URL:

<http://msdn.microsoft.com/ru-ru/library/dd460654.aspx>.

1. Головні концепції програмування [Електронний курс]. – Режим

доступу: URL:

http://www.codenet.ru/progr/cpp/ipn.php

1. Сучасні АСУ готельним бізнесом та їхні функціональні можливості [Електронний ресурс]. – Режим доступу: URL:

<http://www.moluch.ua/archive/67/11284/>

1. Загальні відомості про платформу .Net Framework [Електронний ресурс]. – Режим доступу: URL:

https://msdn.microsoft.com/ru-ru/library/zw4w595w(v=vs.110).aspx

# Додаток А

Текст програми

Клас Room (файл Room.cs)

using System;

using System.Collections.Generic;

using System.Linq;

using System.Runtime.Serialization;

using System.Text;

using System.Threading.Tasks;

namespace Server.Data

{

[DataContract]

public class Room : Base<Room>

{

public Room(int number, Comfort comfort, string quantity, int price):base()

{

this.number = number;

this.comfort = comfort;

quantityP = quantity;

this.price = price;

}

[DataMember]

private int number;

public int Number

{

get { return number; }

set

{

number = value;

}

}

[DataMember]

private Comfort comfort;

public Comfort Com

{

get

{

return comfort;

}

}

[DataMember]

private string quantityP;

public string QuantityP

{

get { return quantityP; }

}

[DataMember]

private int price;

public int Price

{

get

{

return price;

}

set

{

price = value;

}

}

[DataMember]

protected bool accessibility;

public bool Accessibility

{

get;

set;

}

public List<Person> Clients

{

get

{

List<Person> client = new List<Person>();

foreach (RoomPerson rp in RoomPerson.Items.Values)

{

if (rp.Room == this)

{

client.Add(rp.Person);

}

}

return client;

}

}

public static void RoomsFill()

{

Room r1 = new Room(1, Comfort.Standart, "single", 600);

Room r2 = new Room(2, Comfort.Standart, "double", 1000);

Room r3 = new Room(3, Comfort.Standart, "triple", 1300);

Room r4 = new Room(4, Comfort.HalfLuxe, "single", 800);

Room r5 = new Room(5, Comfort.HalfLuxe, "double", 1200);

Room r6 = new Room(6, Comfort.HalfLuxe, "double", 1300);

Room r7 = new Room(7, Comfort.HalfLuxe, "triple", 1600);

Room r8 = new Room(8, Comfort.Luxe, "single", 1100);

Room r9 = new Room(9, Comfort.Luxe, "double", 1500);

Room r10 = new Room(10, Comfort.Luxe, "triple", 2100);

}

public static List<Room> SearchR(Comfort comfort)

{

List<Room> rooms = new List<Room>();

foreach (var item in Base<Room>.Items.Values)

{

if (item.Com == comfort)

{

rooms.Add(item);

}

}

if (rooms.Count == 0)

{

System.Windows.Forms.MessageBox.Show("Nothing found");

}

return rooms;

}

public static List<Room> SearchR(int price)

{

List<Room> rooms = new List<Room>();

foreach (var item in Base<Room>.Items.Values)

{

if (item.Price == price)

{

rooms.Add(item);

}

}

if (rooms.Count == 0)

{

System.Windows.Forms.MessageBox.Show("Nothing found");

}

return rooms;

}

public static List<Room> SearchR(string type)

{

List<Room> rooms = new List<Room>();

foreach (var item in Base<Room>.Items.Values)

{

if (item.QuantityP == type)

{

rooms.Add(item);

}

}

if (rooms.Count == 0)

{

System.Windows.Forms.MessageBox.Show("Nothing found");

}

return rooms;

}

}

}

Клас Person (файл Person.cs)

using System;

using System.Collections.Generic;

using System.Linq;

using System.Runtime.Serialization;

using System.Text;

using System.Threading.Tasks;

namespace Server.Data

{

[DataContract]

public class Person:Base<Person>

{

public Person(string name, DateTime dIn, DateTime dOut, Room room):base(name)

{

dateIn = dIn;

dateOut = dOut;

Room = room

}

[DataMember]

public Room Room { get; set; }

[DataMember]

private string firtName;

public string FirstName

{

get { return firtName; }

}

[DataMember]

private DateTime dateIn;

public DateTime DateIn

{

get { return dateIn; }

}

[DataMember]

private DateTime dateOut;

public DateTime DateOut {

get

{

return dateOut;

}

set

{

dateOut = value;

}

}

[DataMember]

private bool payed;

public bool Payed

{

get

{

return payed;

}

set

{

payed = value;

}

}

[DataMember]

private bool book;

public bool Book

{

get { return book; }

set { book = value; }

}

[DataMember]

private bool taken;

public bool Taken

{

get { return taken; }

set { taken = value; }

}

public List<Room> Rooms

{

get

{

List<Room> room = new List<Room>();

foreach (RoomPerson rp in RoomPerson.Items.Values)

{

if (rp.Person == this)

{

room.Add(rp.Room);

}

}

return room;

}

}

public List<Review> Reviews

{

get

{

List<Review> review = new List<Review>();

foreach (ReviewPerson rw in ReviewPerson.Items.Values)

{

if (rw.Person == this)

{

review.Add(rw.Review);

}

}

return review;

}

}

public static List<Person> Search(string name)

{

List<Person> clients = new List<Person>();

foreach (var item in Base<Person>.Items.Values)

{

if (item.Name == name)

{

clients.Add(item);

}

}

if (clients.Count == 0)

{

System.Windows.Forms.MessageBox.Show("Nothing found");

}

return clients;

}

}

}

Клас Review (файл Review.cs)

using System;

using System.Collections.Generic;

using System.Linq;

using System.Runtime.Serialization;

using System.Text;

using System.Threading.Tasks;

namespace Server.Data

{

[DataContract]

public class Review:Base<Review>

{

public Review(Person p, string text)

this.reviewText = text;

this.person = p;

}

[DataMember]

private Person person;

public Person Person

{

get

{

return person;

}

}

[DataMember]

private string reviewText;

public string ReviewText

{

get

{

return reviewText;

}

set

{

reviewText = value;

}

}

}

}

Клас RoomPerson (файл RoomPerson.cs)

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Runtime.Serialization;

namespace Server.Data

{

[DataContract]

public class RoomPerson :Base<RoomPerson>

{

public Room Room

{

get { return Room.Items[\_roomId]; }

set { \_roomId = value.Id; }

}

public Person Person

{

get { return Person.Items[\_personId]; }

set { \_personId = value.Id; }

}

[DataMember]

private Guid \_roomId;

[DataMember]

private Guid \_personId;

public RoomPerson(Room room,Person person):base()

{

Room = room;

Person = person;

}

}

}

Клас Base (файл Base.cs)

using System;

using System.Collections.Generic;

using System.Linq;

using System.Runtime.Serialization;

using System.Text;

using System.Threading.Tasks;

namespace Server.Data

{

[DataContract]

public class Base<T> where T:Base<T>

{

[DataMember]

static public Dictionary<Guid, T> Items = new Dictionary<Guid, T>();

[DataMember]

public Guid Id { get; protected set; }

[DataMember]

public string Name { get; set; }

public Base()

: this("")

{

}

public Base(string name)

{

Id = Guid.NewGuid();

if(Items == null)

{

Items = new Dictionary<Guid, T>();

}

Items.Add(Id, (T)this);

Name = name;

}

}

}

Енумератор Comfort (файл Comfort.cs)

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Server.Data

{

public enum Comfort

{

Standart,

HalfLuxe,

Luxe

}

}

Клас Message (файл Message.cs)

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Server.Data

{

public class Message

{

public string Name { get; set; }

public string Data { get; set; }

}

}

Клас Serializer (файл Serializer.cs)

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Runtime.Serialization.Json;

using System.Text;

using System.Threading.Tasks;

namespace Server.Data

{

public static class Serializer

{

public static Message Serialize<T>(Dictionary<Guid, T> d)

{

DataContractJsonSerializer jsonFormatter = new DataContractJsonSerializer(typeof(Dictionary<Guid, T>));

using (MemoryStream ms = new MemoryStream())

{

jsonFormatter.WriteObject(ms, d);

return new Message { Data = Encoding.Default.GetString(ms.ToArray()), Name = typeof(T).ToString() };

}

}

public static Dictionary<Guid, T> Deserialize<T>(Message m)

{

DataContractJsonSerializer json = new DataContractJsonSerializer(typeof(Dictionary<Guid, T>));

return (Dictionary<Guid, T>)json.ReadObject(new MemoryStream(Encoding.UTF8.GetBytes(m.Data)));

}

}

}

Інтерфейс IService1 (файл IService1.cs)

using System;

using System.Collections.Generic;

using System.Linq;

using System.Runtime.Serialization;

using System.ServiceModel;

using System.ServiceModel.Web;

using System.Text;

using Server.Data;

namespace WcfService

{

[ServiceContract]

public interface IService1

{

[OperationContract]

void SaveData(Message m);

[OperationContract]

Message LoadData(string s);

[OperationContract]

int Pay(int price, DateTime dIn, DateTime dOut);

[OperationContract]

List<Person> Search(string name);

[OperationContract]

List<Room> SearchCom(Comfort comfort);

[OperationContract]

List<Room> SearchPrice(int price);

[OperationContract]

List<Room> SearchType(string type);

}

Клас Service1 (файл Service1.cvs)

using System;

using System.Collections.Generic;

using System.Linq;

using System.Runtime.Serialization;

using System.ServiceModel;

using System.ServiceModel.Web;

using System.Text;

using System.IO;

using Server.Data;

namespace WcfService

{

public class Service1 : IService1

{

public void SaveData(Server.Data.Message m)

{

var path = String.Format("C:\\HotelDB\\{0}.json",m.Name);

File.WriteAllText(path, m.Data);

}

public Server.Data.Message LoadData(string s)

{

string fillContent = File.ReadAllText(String.Format("C:\\HotelDB\\{0}.json", s));

return new Message { Name = s, Data = fillContent };

}

public int Pay(int price, DateTime dIn, DateTime dOut)

{

if (dOut.Month == dIn.Month)

{

int quan = dOut.Day - dIn.Day;

return quan \* price;

}

else

{

int quan = dOut.Day + 30 - dIn.Day;

return quan \* price;

}

}

public List<Person> Search(string name)

{

List<Person> clients = new List<Person>();

foreach (var item in Base<Person>.Items.Values)

{

if (item.Name == name)

{

clients.Add(item);

}

}

if (clients.Count == 0)

{

System.Windows.Forms.MessageBox.Show("Nothing found");

}

return clients;

}

public List<Room> SearchCom(Comfort comfort)

{

List<Room> rooms = new List<Room>();

foreach (var item in Base<Room>.Items.Values)

{

if (item.Com == comfort)

{

rooms.Add(item);

}

}

if (rooms.Count == 0)

{

System.Windows.Forms.MessageBox.Show("Nothing found");

}

return rooms;

}

public List<Room> SearchPrice(int price)

{

List<Room> rooms = new List<Room>();

foreach (var item in Base<Room>.Items.Values)

{

if (item.Price == price)

{

rooms.Add(item);

}

}

if (rooms.Count == 0)

{

System.Windows.Forms.MessageBox.Show("Nothing found");

}

return rooms;

}

public List<Room> SearchType(string type)

{

List<Room> rooms = new List<Room>();

foreach (var item in Base<Room>.Items.Values)

{

if (item.QuantityP == type)

{

rooms.Add(item);

}

}

if (rooms.Count == 0)

{

System.Windows.Forms.MessageBox.Show("Nothing found");

}

return rooms;

}

}

}

Вікно MainWindow (файл MainWindow.cs)

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using Server.Data;

using Hotel.ServiceReference2;

namespace Hotel

{

public partial class MainWindow : Form

{

public MainWindow()

{

InitializeComponent();

Room.RoomsFill();

}

private void roomsToolStripMenuItem\_Click(object sender, EventArgs e)

{

Rooms rooms = new Rooms();

rooms.Show();

}

private void clientsToolStripMenuItem\_Click(object sender, EventArgs e)

{

Clients clientsW = new Clients();

clientsW.Show();

}

private void paymentToolStripMenuItem\_Click(object sender, EventArgs e)

{

Payment payW = new Payment();

payW.Show();

}

private void reviewToolStripMenuItem\_Click(object sender, EventArgs e)

{

Reviews r = new Reviews();

r.Show();

}

private void aboutToolStripMenuItem\_Click(object sender, EventArgs e)

{

About a = new About();

a.Show();

}

private void MainWindow\_FormClosing(object sender, FormClosingEventArgs e)

{

Service1Client sc = new Service1Client();

sc.SaveData(Serializer.Serialize(Base<Person>.Items));

Service1Client sc1 = new Service1Client();

sc1.SaveData(Serializer.Serialize(Base<Room>.Items));

Service1Client sc2 = new Service1Client();

sc2.SaveData(Serializer.Serialize(Base<Review>.Items));

}

private void MainWindow\_Load(object sender, EventArgs e)

{

Service1Client sc = new Service1Client();

var g = sc.LoadData(typeof(Person).ToString());

Base<Person>.Items = Serializer.Deserialize<Person>(g);

Service1Client sc1 = new Service1Client();

var g1 = sc1.LoadData(typeof(Room).ToString());

Base<Room>.Items = Serializer.Deserialize<Room>(g1);

Service1Client sc2 = new Service1Client();

var g2 = sc2.LoadData(typeof(Review).ToString());

Base<Review>.Items = Serializer.Deserialize<Review>(g2);

}

}

}

Вікно Rooms (файл Rooms.cs)

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using Server.Data;

namespace Hotel

{

public partial class Rooms : Form

{

public Rooms()

{

InitializeComponent();

}

private void backBtn\_Click(object sender, EventArgs e)

{

this.Close();

}

private void Rooms\_Load(object sender, EventArgs e)

{

int j = 0;

foreach (var item in Base<Room>.Items.Values)

{

dataGrid.Rows.Add();

for (int i = j; i < dataGrid.Rows.Count - 1; i++)

{

dataGrid[0, i].Value = item.Number;

dataGrid[1, i].Value = item.Com;

dataGrid[2, i].Value = item.QuantityP;

dataGrid[3, i].Value = item.Price;

dataGrid[4, i].Value = "free";

}

j++;

}

int m = 0;

for (int i = m; i < dataGrid.Rows.Count - 1; i++)

{

foreach (var it in Base<Person>.Items.Values)

{

if (it.Room.Number == Convert.ToUInt16(dataGrid[0, i].Value))

{

if ((it.Book == false)&&(it.Taken == true))

{

dataGrid[4, i].Value = "taken";

dataGrid[5, i].Value = it.DateIn.Day + "." + it.DateIn.Month + "-" + it.DateOut.Day + "." + it.DateOut.Month + "." + it.DateOut.Year;

}

else if ((it.Book == true)&&(it.Taken==false))

{

dataGrid[4, i].Value = "reserved";

dataGrid[5, i].Value = it.DateIn.Day + "." + it.DateIn.Month + "-" + it.DateOut.Day + "." + it.DateOut.Month + "." + it.DateOut.Year;

}

}

}

}

m++;

}

private void takeUpBtn\_Click(object sender, EventArgs e)

{

addClient add = new addClient();

add.Show();

this.Hide();

}

private void setFreeBtn\_Click(object sender, EventArgs e)

{

if (dataGrid.CurrentRow.Cells[4].Value == "reserved")

{

MessageBox.Show("Room isn't taken up");

}

else

{

dataGrid.CurrentRow.Cells[4].Value = "free";

dataGrid.CurrentRow.Cells[5].Value = "";

foreach (var item in Base<Person>.Items.Values)

{

if (item.Room.Number == Convert.ToUInt16(dataGrid.CurrentRow.Cells[0].Value))

{

item.Taken = false;

}

}

}

}

private void removeResBtn\_Click(object sender, EventArgs e)

{

if (dataGrid.CurrentRow.Cells[4].Value == "taken")

{

MessageBox.Show("Room is already taken up");

}

else

{

dataGrid.CurrentRow.Cells[4].Value = "free";

dataGrid.CurrentRow.Cells[5].Value = "";

foreach (var item in Base<Person>.Items.Values)

{

if (item.Room.Number == Convert.ToUInt16(dataGrid.CurrentRow.Cells[0].Value))

{

item.Book = false;

}

}

}

}

private void reserveBtn\_Click(object sender, EventArgs e)

{

addClient add = new addClient();

add.checkBook.Checked = true;

add.Show();

this.Hide();

}

private void searchBtn\_Click(object sender, EventArgs e)

{

dataGrid.Rows.Clear();

if (radioBtnComfort.Checked == true)

{

int j = 0;

string room = roomBox.Text;

try

{

Comfort comfort = (Comfort)Enum.Parse(typeof(Comfort), room);

List<Room> rooms = Room.SearchR(comfort);

foreach (var item in rooms)

{

dataGrid.Rows.Add();

for (int i = j; i < dataGrid.Rows.Count - 1; i++)

{

dataGrid[0, i].Value = item.Number;

dataGrid[1, i].Value = item.Com;

dataGrid[2, i].Value = item.QuantityP;

dataGrid[3, i].Value = item.Price;

dataGrid[4, i].Value = "free";

}

j++;

}

}

catch (Exception)

{

MessageBox.Show("Nothing found");

}

int m = 0;

for (int i = m; i < dataGrid.Rows.Count - 1; i++)

{

foreach (var it in Base<Person>.Items.Values)

{

if (it.Room.Number == Convert.ToUInt16(dataGrid[0, i].Value))

{

if ((it.Book == false) && (it.Taken == true))

{

dataGrid[4, i].Value = "taken";

dataGrid[5, i].Value = it.DateIn.Day + "." + it.DateIn.Month + "-" + it.DateOut.Day + "." + it.DateOut.Month + "." + it.DateOut.Year;

}

else if ((it.Book == true) && (it.Taken == false))

{

dataGrid[4, i].Value = "reserved";

dataGrid[5, i].Value = it.DateIn.Day + "." + it.DateIn.Month + "-" + it.DateOut.Day + "." + it.DateOut.Month + "." + it.DateOut.Year;

}

}

}

}

m++;

}

else if (radioBtnPrice.Checked == true)

{

try

{

int j = 0;

int price = Convert.ToInt16(roomBox.Text);

List<Room> rooms = Room.SearchR(price);

foreach (var item in rooms)

{

dataGrid.Rows.Add();

for (int i = j; i < dataGrid.Rows.Count - 1; i++)

{

dataGrid[0, i].Value = item.Number;

dataGrid[1, i].Value = item.Com;

dataGrid[2, i].Value = item.QuantityP;

dataGrid[3, i].Value = item.Price;

dataGrid[4, i].Value = "free";

}

j++;

}

}

catch (Exception)

{

MessageBox.Show("Nothing found");

}

int m = 0;

for (int i = m; i < dataGrid.Rows.Count - 1; i++)

{

foreach (var it in Base<Person>.Items.Values)

{

if (it.Room.Number == Convert.ToUInt16(dataGrid[0, i].Value))

{

if ((it.Book == false) && (it.Taken == true))

{

dataGrid[4, i].Value = "taken";

dataGrid[5, i].Value = it.DateIn.Day + "." + it.DateIn.Month + "-" + it.DateOut.Day + "." + it.DateOut.Month + "." + it.DateOut.Year;

}

else if ((it.Book == true) && (it.Taken == false))

{

dataGrid[4, i].Value = "reserved";

dataGrid[5, i].Value = it.DateIn.Day + "." + it.DateIn.Month + "-" + it.DateOut.Day + "." + it.DateOut.Month + "." + it.DateOut.Year;

}

}

}

}

m++;

}

else if (radioBtnType.Checked == true)

{

int j = 0;

string type = roomBox.Text;

List<Room> rooms = Room.SearchR(type);

foreach (var item in rooms)

{

dataGrid.Rows.Add();

for (int i = j; i < dataGrid.Rows.Count - 1; i++)

{

dataGrid[0, i].Value = item.Number;

dataGrid[1, i].Value = item.Com;

dataGrid[2, i].Value = item.QuantityP;

dataGrid[3, i].Value = item.Price;

dataGrid[4, i].Value = "free";

}

j++;

}

int m = 0;

for (int i = m; i < dataGrid.Rows.Count - 1; i++)

{

foreach (var it in Base<Person>.Items.Values)

{

if (it.Room.Number == Convert.ToUInt16(dataGrid[0, i].Value))

{

if ((it.Book == false) && (it.Taken == true))

{

dataGrid[4, i].Value = "taken";

dataGrid[5, i].Value = it.DateIn.Day + "." + it.DateIn.Month + "-" + it.DateOut.Day + "." + it.DateOut.Month + "." + it.DateOut.Year;

}

else if ((it.Book == true) && (it.Taken == false))

{

dataGrid[4, i].Value = "reserved";

dataGrid[5, i].Value = it.DateIn.Day + "." + it.DateIn.Month + "-" + it.DateOut.Day + "." + it.DateOut.Month + "." + it.DateOut.Year;

}

}

}

}

m++;

}

}

}

}

Вікно Clients (файл Clients.cs)

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using Server.Data;

using Hotel.ServiceReference2;

namespace Hotel

{

public partial class Clients : Form

{

public Clients()

{

InitializeComponent();

}

private void backBtn\_Click(object sender, EventArgs e)

{

this.Close();

}

private void addClientBtn\_Click(object sender, EventArgs e)

{

addClient add = new addClient();

add.Show();

this.Close();

}

private void Clients\_Load(object sender, EventArgs e)

{

int j = 0;

foreach (var item in Base<Person>.Items.Values)

{

dataGrid.Rows.Add();

for (int i = j; i < dataGrid.Rows.Count - 1; i++)

{

dataGrid[0, i].Value = item.Name;

dataGrid[1, i].Value = item.DateIn.Day + "." + item.DateIn.Month + "." + item.DateIn.Year;

dataGrid[2, i].Value = item.DateOut.Day + "." + item.DateOut.Month + "." + item.DateOut.Year;

dataGrid[3, i].Value = item.Room.Number;

}

j++;

}

}

private void searchBtn\_Click(object sender, EventArgs e)

{

dataGrid.Rows.Clear();

int j = 0;

string client = clientBox.Text;

List<Person> clients = Person.Search(client);

foreach (var item in clients)

{

dataGrid.Rows.Add();

for (int i = j; i < dataGrid.Rows.Count - 1; i++)

{

dataGrid[0, i].Value = item.Name;

dataGrid[1, i].Value = item.DateIn.Day + "." + item.DateIn.Month + "." + item.DateIn.Year;

dataGrid[2, i].Value = item.DateOut.Day + "." + item.DateOut.Month + "." + item.DateOut.Year;

dataGrid[3, i].Value = item.Room.Number;

}

j++;

}

}

}

}

Вікно addClient (файл addClient.cs)

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using Server.Data;

namespace Hotel

{

public partial class addClient : Form

{

public addClient()

{

InitializeComponent();

roomBox.DropDownStyle = ComboBoxStyle.DropDownList;

foreach (var item in Base<Room>.Items.Values)

{

roomBox.Items.Add(item.Number);

}

}

private void addBtn\_Click(object sender, EventArgs e)

{

string name = nameBox.Text;

string dIn = dInBox.Text;

string dOut = dOutBox.Text;

string room = roomBox.Text;

foreach (var item in Base<Room>.Items.Values)

{

if(Convert.ToInt16(roomBox.Text) == item.Number)

{

Person client = new Person(name, DateTime.Parse(dIn), DateTime.Parse(dOut), item);

client.Payed = false;

if(checkBook.Checked == true)

{

client.Book = true;

client.Taken = false;

}

else

{

client.Book = false;

client.Taken = true;

}

RoomPerson rp = new RoomPerson(item, client);

}

}

Clients cl = new Clients();

cl.Show();

}

private void backBtn\_Click(object sender, EventArgs e)

{

this.Hide();

}

}

}

Вікно Payment (файл Payment.cs)

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using Server.Data;

using Hotel.ServiceReference2;

namespace Hotel

{

public partial class Payment : Form

{

public Payment()

{

InitializeComponent();

}

private void backBtn\_Click(object sender, EventArgs e)

{

this.Close();

}

private void Payment\_Load(object sender, EventArgs e)

{

int j = 0;

foreach (var item in Base<Person>.Items.Values)

{

if ((item.Payed == false)&&(item.Taken == true))

{

dataGrid.Rows.Add();

for (int i = j; i < dataGrid.Rows.Count - 1; i++)

{

dataGrid[0, i].Value = item.Room.Number;

dataGrid[1, i].Value = item.Room.Price;

dataGrid[2, i].Value = item.DateIn;

dataGrid[3, i].Value = item.DateOut;

dataGrid[4, i].Value = item.Name;

j++;

}

}

}

}

private void payBtn\_Click(object sender, EventArgs e)

{

try

{

int price = Convert.ToInt16(dataGrid.CurrentRow.Cells[1].Value);

DateTime dIn = Convert.ToDateTime(dataGrid.CurrentRow.Cells[2].Value);

DateTime dOut = Convert.ToDateTime(dataGrid.CurrentRow.Cells[3].Value);

Service1Client sc = new Service1Client();

int sum = sc.Pay(price, dIn, dOut);

MessageBox.Show("Your bill: " + sum +" cu");

foreach (var item in Base<Person>.Items.Values)

{

for (int i = 0; i < dataGrid.Rows.Count - 1; i++)

{

if ((item.Room.Number == Convert.ToInt16(dataGrid.CurrentRow.Cells[0].Value)) && (item.Name == dataGrid.CurrentRow.Cells[4].Value))

{

item.Payed = true;

}

}

}

this.Hide();

}

catch(Exception)

{

MessageBox.Show("Pick the room to count");

}

}

}

}

Вікно Reviews (файл Reviews.cs)

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using Server.Data;

namespace Hotel

{

public partial class Reviews : Form

{

public Reviews()

{

InitializeComponent();

}

private void backBtn\_Click(object sender, EventArgs e)

{

this.Close();

}

private void addReviewBtn\_Click(object sender, EventArgs e)

{

addReview review = new addReview();

review.Show();

this.Hide();

}

private void Reviews\_Load(object sender, EventArgs e)

{

int j = 0;

foreach (var item in Base<Review>.Items.Values)

{

dataGrid.Rows.Add();

for (int i = j; i < dataGrid.Rows.Count - 1; i++)

{

dataGrid[0, i].Value = item.Person.Name;

dataGrid[1, i].Value = item.ReviewText;

}

j++;

}

}

}

}

Вікно addReview (файл addReview.cs)

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using Server.Data;

namespace Hotel

{

public partial class addReview : Form

{

public addReview()

{

InitializeComponent();

roomBox.DropDownStyle = ComboBoxStyle.DropDownList;

foreach (var item in Base<Room>.Items.Values)

{

roomBox.Items.Add(item.Number);

}

}

private void addBtn\_Click(object sender, EventArgs e)

{

string name = nameBox.Text;

string rev = reviewBox.Text;

string dIn = dInBox.Text;

string dOut = dOutBox.Text;

string room = roomBox.Text;

foreach (var item in Base<Room>.Items.Values)

{

if (Convert.ToInt16(roomBox.Text) == item.Number)

{

Person per = new Person(name, DateTime.Parse(dIn), DateTime.Parse(dOut), item);

per.Payed = false;

per.Book = false;

per.Taken = true;

Review review = new Review(per, rev);

ReviewPerson rp = new ReviewPerson(review, per);

RoomPerson roomp = new RoomPerson(item, per);

}

}

}

private void backBtn\_Click(object sender, EventArgs e)

{

this.Hide();

Reviews r = new Reviews();

r.Show();

}

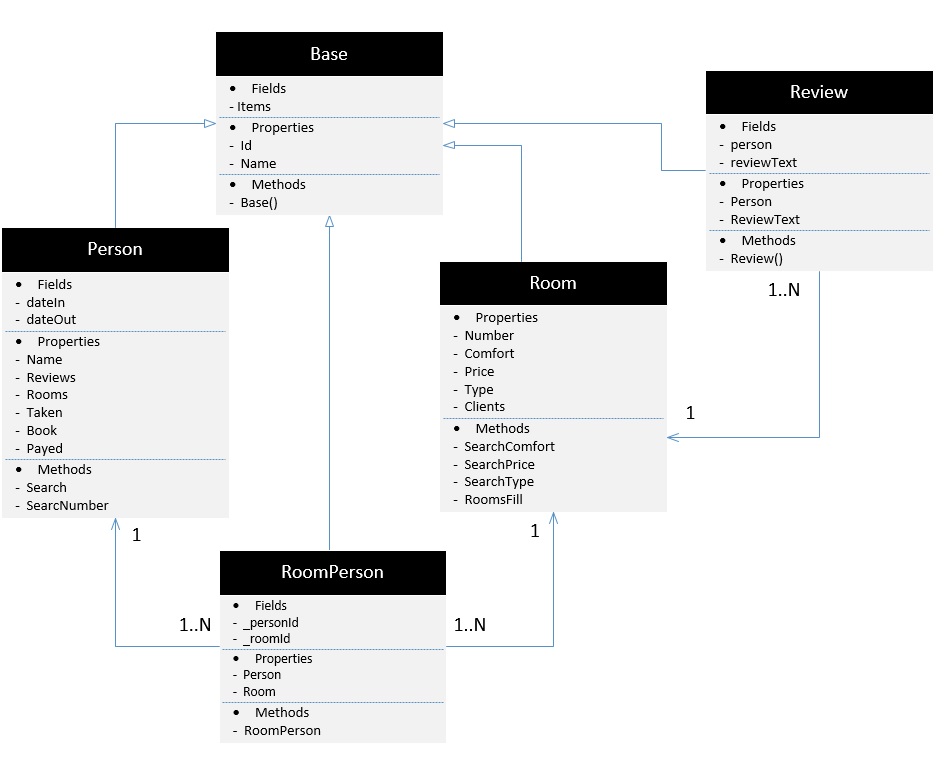
}

}

#### Дотаток Б

Діаграма зв’язків класів

Діаграма, яка описує зв’язки між класами та їхніми об’єктами.



#### Дотаток В

Діаграма класів

