ДЕРЖАВНИЙ ТОРГОВЕЛЬНО-ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ

Кафедра інженерії програмного забезпечення та кібербезпеки

Захищено на кафедрі інженерії

програмного забезпечення та кібербезпеки

«\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2025р.

з оцінкою\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Підпис членів комісії:

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

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

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

КУРСОВА РОБОТА

з дисципліни

“ОБ’ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ”

НА ТЕМУ:

Програмний додаток для управління ресурсами багатофункціонального житлового комплексу

Студента факультету Інформаційних технологій

групи\_курсу ФІТ 2-6\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Пушенко Є.А\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Прізвище, ім’я, по-батькові (підпис)

Науковий керівник  доктор технічних наук

професор Цюцюра С. В.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Науковий ступень, вчене звання

Прізвище, ім’я, по-батькові (підпис)

Київ 2025

**Анотація**

Курсова робота присвячена розробці програмного додатку для управління ресурсами багатофункціонального житлового комплексу. У роботі розглянуто актуальність проблеми автоматизації обліку оренди та бронювання апартаментів, конференц-залів, паркомісць, додаткових послуг (спортзал, обладнання тощо) та місць у ресторані, розташованому в межах житлового комплексу.

Загальний обсяг пояснювальної записки становить 47 сторінок. У роботі представлено 1 таблицю та 24 рисунки, що ілюструють архітектуру системи, інтерфейс користувача, структурні модулі та фрагменти коду.

Система реалізована з використанням C# та технології WPF у середовищі Visual Studio. База даних створена на основі SQLite із застосуванням DB Browser for SQLite, що забезпечує локальне зберігання інформації про ресурси та користувачів. Інтерфейс підтримує шаблон MVVM, а основна функціональність охоплює CRUD-операції, авторизацію, генерацію PDF-звітів, фільтрацію та інтерактивне бронювання.

Ключові слова: *управління ресурсами, житловий комплекс, бронювання, WPF, SQLite, автоматизація, інформаційна система.*

Сутність розробки полягає в створенні безкоштовного адміністративного додатку, орієнтованого на менеджерів та директорів житлового комплексу, що дозволяє централізовано керувати усіма ресурсами без необхідності залучення стороннього комерційного ПЗ. Система забезпечує простий, гнучкий та ефективний інструмент для щоденного управління, оптимізації ресурсів і підвищення якості обслуговування мешканців.

**Зміст**

**Вступ4**

1. **Теоретична частина6**
   1. Аналіз існуючих рішень у сфері управління ресурсами житлових комплексів6
   2. Огляд літератури та досліджень у сфері управління ресурсами ЖК11
   3. Методи та моделі управління ресурсами в житлових комплексах14
2. **Основна частина16**
   1. Постановка задачі та техніко-економічне обґрунтування16
   2. Розробка алгоритму функціонування програми17
   3. Вибір середовища розробки та технологій19
   4. Практична реалізація програмного додатку21
      1. Архітектура програмного забезпечення21
      2. Розробка інтерфейсу користувача24
      3. Реалізація основних функціональних модулів30
3. **Заключна частина36**
   1. Опис системи та взаємодії модулів36
   2. Інструкція для розробника та користувача38
   3. Результати тестування програмного додатку41

**Висновки47**

**Список використаної літератури48**

**Додатки49**

**Вступ**

У сучасному світі ефективне управління ресурсами багатофункціональних житлових комплексів є важливою передумовою забезпечення комфортного проживання мешканців та раціонального використання інфраструктури. З розвитком інформаційних технологій та цифрової трансформації управлінських процесів зростає потреба у створенні програмних рішень, здатних інтегрувати облік, бронювання та контроль за всіма доступними ресурсами комплексу — від апартаментів до ресторанних зон. В умовах збільшення вимог до якості сервісу та прозорості внутрішніх операцій автоматизація таких процесів набуває особливого значення.

*Актуальність теми* зумовлена недостатньою кількістю безкоштовних, зручних і гнучких програмних інструментів, які б дозволяли менеджерам та адміністраторам житлових комплексів здійснювати централізоване управління апартаментами, залами, паркомісцями, спортивними послугами та ресторанною зоною. Більшість комерційних рішень на ринку мають обмежений функціонал, високу вартість або складність налаштування, що не завжди є прийнятним для невеликих компаній або новостворених об’єктів.

*Метою* даної курсової роботи є розробка програмного додатку для внутрішнього використання адміністрацією житлового комплексу, який дозволить ефективно управляти ресурсами без потреби в ліцензійних платежах чи складній інфраструктурі.

*Об’єктом дослідження* є процес управління ресурсами в багатофункціональному житловому комплексі.

*Предметом дослідження* є методи та інструменти програмної реалізації функціоналу управління, обліку та звітності щодо використання ресурсів у житловому комплексі.

Для досягнення поставленої мети передбачено виконання таких завдань:

* провести аналіз існуючих рішень у сфері управління ресурсами житлової нерухомості;
* здійснити огляд теоретичних джерел і сучасних підходів до управління ресурсами;
* сформувати архітектуру програмного додатку;
* реалізувати основні функціональні модулі;
* протестувати роботу системи та підготувати рекомендації щодо використання.

Результатом дослідження стане створення повноцінного десктопного програмного продукту, що дозволяє управляти ресурсами житлового комплексу в одному зручному інтерфейсі.

1. **Теоретична частина**
   1. **Аналіз існуючих рішень у сфері управління ресурсами житлових комплексів**

Управління ресурсами житлових комплексів є складним багатокомпонентним процесом, що охоплює облік, планування, розподіл та оптимізацію використання матеріальних, енергетичних та людських ресурсів. З розвитком інформаційних технологій значна частина цього процесу автоматизується за допомогою спеціалізованих програмних рішень.

На ринку програмного забезпечення вже існує ряд готових продуктів, спрямованих на вирішення завдань обліку та управління ресурсами у сфері житлово-комунального господарства. Серед найбільш поширених можна виділити:

* BAS "Оренда та управління нерухомістю";
* Rental Property Management System;
* Камала Нерухомість.

1. **BAS "Оренда та управління нерухомістю"**

BAS (Business Automation Software) ― програмний продукт українського виробництва, призначений для автоматизації процесів оренди та управління об'єктами нерухомості. Система реалізована на платформі 1С:Підприємство та активно використовується керуючими компаніями, девелоперами та орендодавцями.

**Основні можливості:**

* ведення повного обліку об’єктів нерухомості;
* реєстрація договорів оренди з урахуванням умов, термінів та ставок;
* автоматичне нарахування орендних платежів та комунальних послуг;
* контроль оплат і заборгованостей;
* формування звітності та аналітичних даних по кожному об'єкту;
* інтерфейс українською мовою, локалізована підтримка.

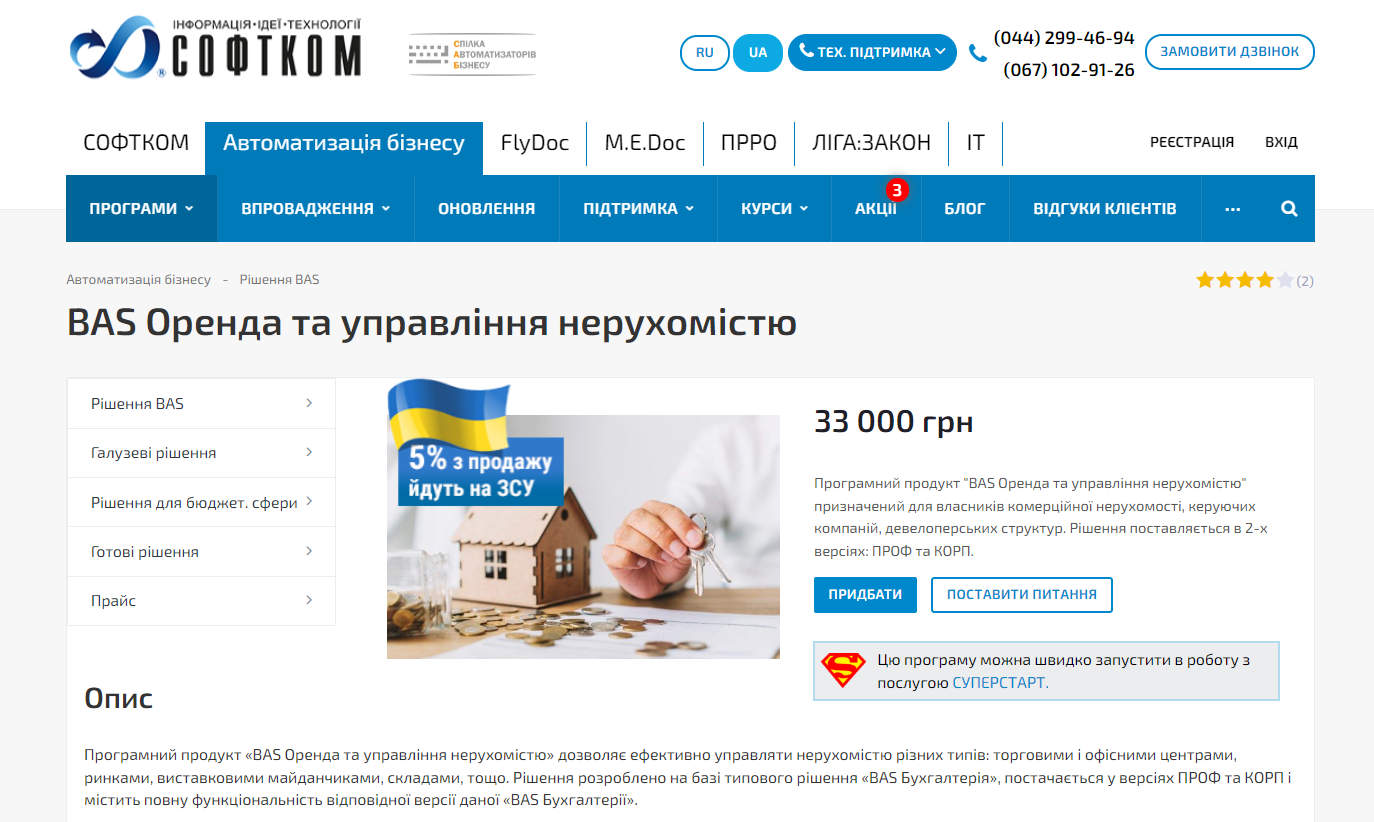


Рисунок 1.1 – Сторінка з системою BAS

*Джерело: [1]*

**Переваги:**

* висока гнучкість у налаштуванні під конкретну організацію;
* інтеграція з бухгалтерськими модулями (BAS Бухгалтерія);
* підтримка вітчизняного законодавства та норм.

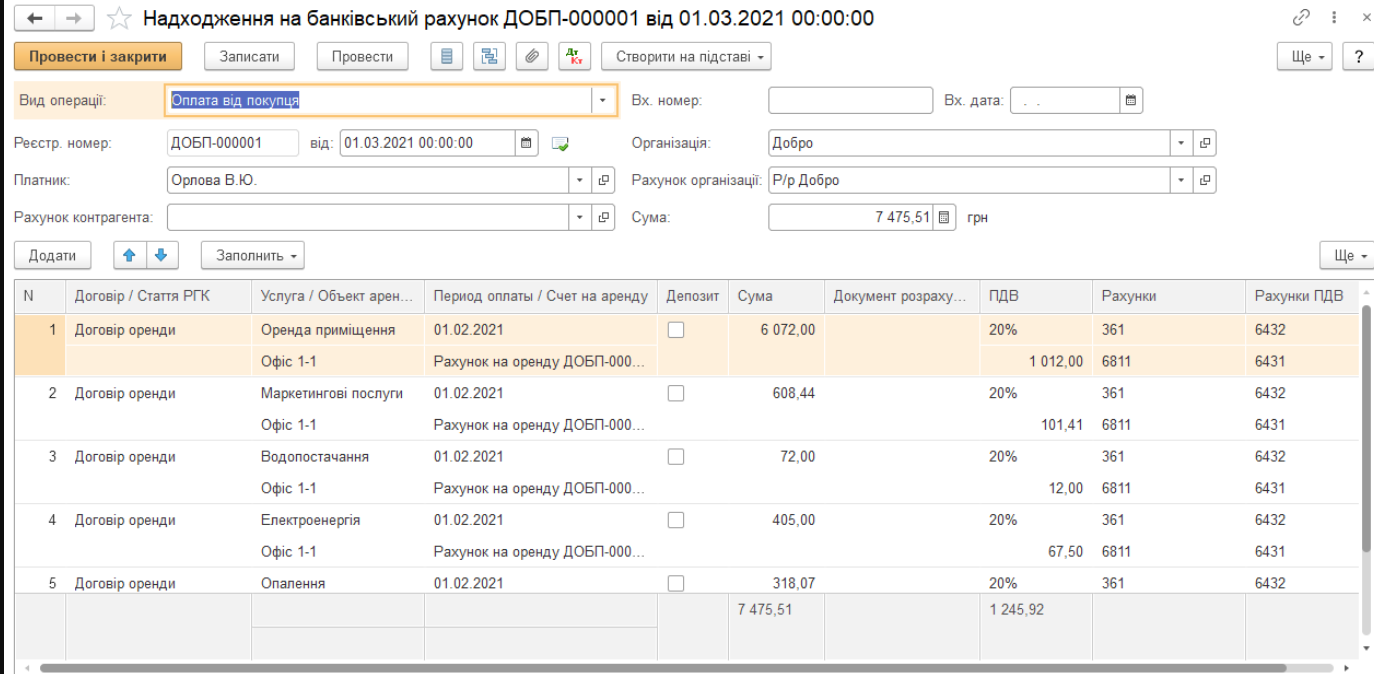


Рисунок 1.2 – Облік платежів по оренді BAS

*Джерело: [2]*

**Недоліки:**

* складний процес впровадження для невеликих компаній;
* необхідність навчання персоналу;
* програмне забезпечення єплатним, передбачає ліцензування і технічну підтримку за додаткову плату.

1. **Rental Property Management System**

Rental Property Management System — це англомовна хмарна система (SaaS), орієнтована на управління житловою та комерційною орендною нерухомістю. Вона використовується як інструмент централізованого управління для менеджерів, власників об’єктів та керуючих компаній.

**Основні можливості:**

* створення та супровід договорів оренди;
* автоматичні нагадування орендарям про платежі;
* інтегрована система обліку технічного обслуговування;
* контроль доходів/витрат і формування фінансових звітів;
* підтримка декількох об'єктів одночасно, включаючи житлові комплекси.

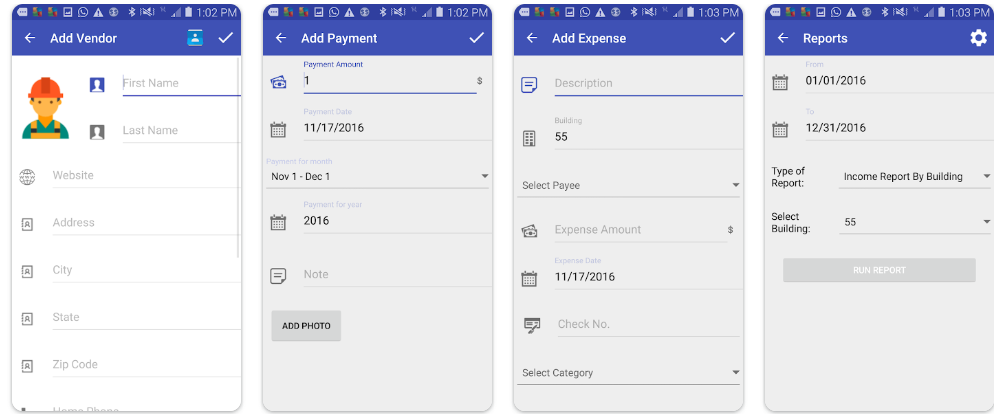


Рисунок 1.3 – Інтерфейс Rental Property Management System

*Джерело: [3]*

**Переваги:**

* простий інтерфейс, орієнтований на бізнес-користувача;
* можливість роботи з мобільних пристроїв;
* автоматизація ключових бізнес-процесів.

**Недоліки:**

* відсутність української локалізації;
* відсутність адаптації під вітчизняне законодавство;
* платна модель (щомісячна/щорічна підписка);
* залежність від стабільного інтернет-з’єднання.

1. **Камала Нерухомість**

«Камала Нерухомість» — українське програмне забезпечення, створене для автоматизації процесів обліку та управління житловою і комерційною нерухомістю. Система орієнтована на потреби девелоперів, рієлторських агентств та керуючих компаній.

**Основні можливості:**

* створення електронної бази об'єктів із детальними характеристиками;
* ведення історії продажів, оренди та супровідних операцій;
* підтримка CRM-функцій для взаємодії з клієнтами;
* модулі фінансового аналізу, документообігу та нагадувань;
* можливість налаштування під бізнес-процеси компанії.

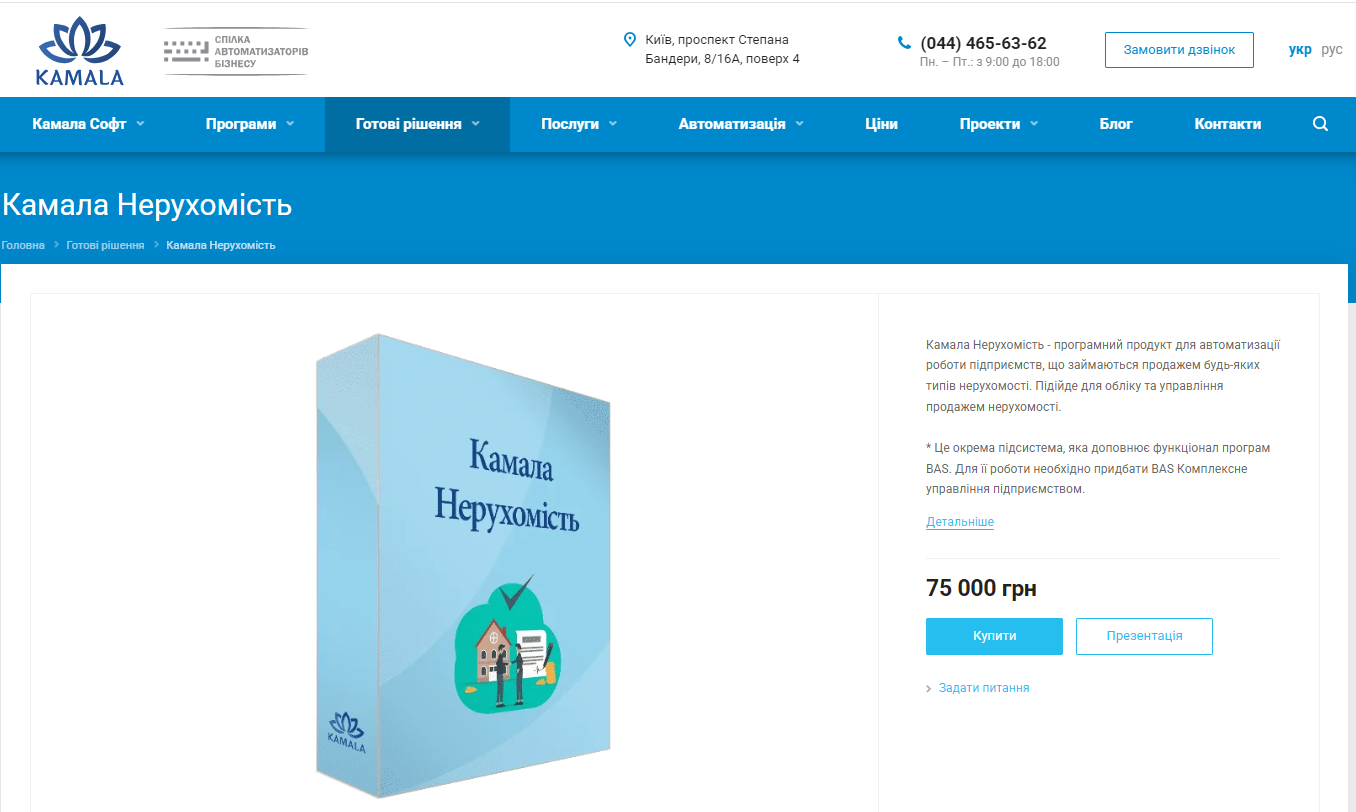


Рисунок 1.4 – Сторінка з системою «Камала Нерухомість»

*Джерело: [4]*

**Переваги:**

* український розробник із локальною підтримкою;
* гнучкість і масштабованість рішення;
* зручний візуальний інтерфейс.

**Недоліки:**

* не всі функції доступні «з коробки» — потрібно налаштування;
* обмежена інтеграція з сторонніми системами;
* програма є платною, вимагає купівлі ліцензії та технічного супроводу.

Таблиця 1. Порівняльна таблиця існуючих рішень

| **Характеристика** | **BAS "Оренда та управління нерухомістю"** | **Rental Property Management System** | **Камала Нерухомість** |
| --- | --- | --- | --- |
| **Країна розробника** | Україна | США / міжнародна | Україна |
| **Мова інтерфейсу** | Українська | Англійська | Українська |
| **Платформа** | 1С:Підприємство | Хмарна (SaaS) /мобільна | Десктоп / хмара |
| **Основне призначення** | Облік, оренда, звітність | Управління орендою, звітність, нагадування | Управління об’єктами, CRM, документообіг |
| **Орієнтація** | Менеджери, бухгалтери, керівники | Власники об’єктів, менеджери | Девелопери, рієлтори, менеджери |
| **Можливість налаштування** | Висока | Низька | Висока |
| **Вартість** | Платна (ліцензія + підтримка) | Платна (підписка) | Платна (ліцензія) |
| **Переваги** | Локалізація, підтримка норм, гнучкість | Простота, доступ онлайн | Інтуїтивний інтерфейс, локальна підтримка |
| **Недоліки** | Складність впровадження, платна | Немає укр. мови, залежність від інтернету, платна | Потрібне налаштування, платна |

*Джерело:* *побудовано автором*

Розглянуті програмні рішення мають широкий функціонал, проте більшість з них орієнтовані на середній та великий бізнес. Вони вимагають фінансових витрат на впровадження, що може бути недоцільним для невеликих житлових комплексів або нових керуючих компаній. Також жодне з програмних забезпечень не надає повної відкритості коду або безкоштовного доступу до базових функцій, що спонукає до розробки власного програмного додатку з урахуванням індивідуальних вимог.

**1.2 Огляд літератури та досліджень у сфері управління ресурсами ЖК**

У сучасних умовах цифровізації та зростаючої складності функціонування житлових комплексів управління їхніми ресурсами набуває особливого значення. У науковій літературі висвітлено широкий спектр підходів до розуміння ресурсного потенціалу, методів управління ресурсами, а також їхньої ролі у забезпеченні ефективності функціонування систем. У цьому контексті доцільно звернути увагу на монографію *«Управління ресурсами підприємства»* під редакцією проф. Г. О. Швиданенко, де окреслено сучасні теоретичні та методичні засади ресурсного підходу, який може бути адаптовано до потреб управління ресурсами житлових комплексів [5].

У дослідженні розглянуто еволюцію ресурсної теорії, наголошено на її гуманітарному вимірі та важливості інтелектуальних і соціальних чинників у процесі формування ресурсного потенціалу. Зазначається, що у контексті інформаційної економіки зростає значення нематеріальних ресурсів — знань, компетенцій, репутації, що формує нові виклики й можливості для менеджменту. Окрему увагу приділено системному підходу до управління ресурсами, де ресурси розглядаються як динамічні елементи організаційної системи, що мають бути інтегровані в загальну стратегію розвитку.

У роботі акцентовано на необхідності ефективного інструментарію управління, який включає методи оцінювання стратегічних ресурсів, інтелектуальне забезпечення, енергоефективність та управління ризиками. Такий підхід є релевантним для сфери управління житловими комплексами, де ресурси мають різну природу — від матеріальних (комунікації, обладнання, енергоносії) до нематеріальних (організаційна структура, інформаційні потоки, рівень підготовки персоналу). Дослідження також вказує на значення знань як ресурсу, що формує конкурентоспроможність системи, що особливо актуально для цифрових платформ, орієнтованих на автоматизацію управління ЖК.

У теоретичному дискурсі також обґрунтовано важливість соціальної відповідальності та клієнт-орієнтованості в системах управління. Хоча житлові комплекси орієнтовані насамперед на ефективне використання матеріальних ресурсів, управління має враховувати інтереси мешканців як кінцевих бенефіціарів. Це відкриває новий вимір у плануванні й реалізації стратегій розвитку таких систем.

У результаті аналізу літературних джерел можна зробити висновок, що питання управління ресурсами житлових комплексів потребує системного підходу, який враховує як економічні, так і соціальні, організаційні та інноваційні аспекти. Розглянуті дослідження свідчать про необхідність розробки адаптивних моделей управління, що поєднують класичні інструменти з новітніми технологічними рішеннями.

У науковій статті Наталії Мороз та Тетяни Малетич, присвяченій інноваційним способам управління багатоквартирними житловими будинками, розкрито сучасні виклики та напрями вдосконалення управлінських практик у сфері житлової нерухомості. Авторки акцентують увагу на зростанні вимог до якості житлового середовища, а також на необхідності адаптації до нових умов за допомогою впровадження інновацій. Управління багатоквартирними житловими будинками позиціонується як складне завдання, що потребує переосмислення традиційних підходів та використання сучасних технологічних рішень [6].

У дослідженні представлено класифікацію управлінських моделей, серед яких розглянуто централізовані, децентралізовані схеми та моделі на основі самоорганізації мешканців, зокрема через створення об’єднань співвласників багатоквартирних будинків. Зазначено, що ефективність управління безпосередньо залежить від залучення мешканців до прийняття рішень, що підвищує рівень довіри та відповідальності.

Особливу увагу авторки приділяють цифровізації управління, розглядаючи її як важливий чинник підвищення ефективності. Застосування автоматизованих систем моніторингу ресурсів, онлайн-комунікацій з мешканцями та цифрових платформ для контролю послуг дозволяє оптимізувати управлінські процеси та оперативно реагувати на запити користувачів. Такі рішення дозволяють не лише підвищити прозорість, а й зменшити витрати, пов’язані з обслуговуванням будинку.

У контексті сучасних викликів акцент зроблено також на питаннях енергетичної ефективності. Авторки підкреслюють важливість впровадження технологій енергозбереження, які дозволяють суттєво знизити витрати мешканців, одночасно сприяючи зменшенню негативного впливу на довкілля. Це може включати встановлення енергоощадного обладнання, автоматизованих систем освітлення, теплоізоляційних матеріалів та інших технологічних рішень.

Соціальна відповідальність розглядається як невід’ємний елемент ефективного управління. Управлінські рішення мають враховувати потреби мешканців, забезпечуючи прозорість, відкритість і можливість громадського контролю. Залучення мешканців до управління створює умови для формування відповідальної спільноти, що здатна ефективно співпрацювати з керуючими органами.

Стаття завершується міркуваннями щодо перспектив розвитку інновацій у сфері управління житловою нерухомістю. Підкреслюється необхідність постійного вдосконалення інструментів і механізмів управління, з урахуванням динамічних змін у технологічному середовищі та зростаючих очікувань мешканців. Таким чином, запропоновані підходи спрямовані на формування якісно нового рівня організації житлового простору, що відповідає сучасним стандартам комфорту, прозорості та економічної доцільності.

* 1. **Методи та** **моделі управління ресурсами в житлових комплексах**

Управління ресурсами в житлових комплексах ґрунтується на системному підході до планування, контролю та оптимізації використання матеріальних, енергетичних, інформаційних і людських ресурсів. З огляду на багатофункціональність сучасних житлових комплексів, які поєднують у собі житлові, комерційні та соціальні зони, виникає потреба у використанні спеціалізованих методів і моделей управління, що забезпечують ефективну взаємодію між усіма підсистемами.

Серед методів управління ресурсами найбільш поширеними є адміністративно-організаційні, економічні, інформаційно-аналітичні та соціальні. Адміністративні методи передбачають централізоване планування і контроль з боку керуючих органів або обслуговуючих організацій. Вони включають регламентацію дій, встановлення нормативів споживання, розробку правил експлуатації інженерних систем. Економічні методи, у свою чергу, базуються на використанні фінансових стимулів, таких як система тарифікації, обґрунтування витрат, складання бюджетів обслуговування та формування резервів на ремонтно-експлуатаційні роботи.

Інформаційно-аналітичні методи набувають особливої актуальності в умовах цифровізації. Системи диспетчеризації, автоматизованого обліку енергоресурсів, цифрові платформи для комунікації з мешканцями та CRM-системи забезпечують високий рівень оперативного контролю, збору статистичних даних та аналізу ефективності управлінських рішень. Соціальні методи спрямовані на формування культури відповідального користування ресурсами, залучення мешканців до прийняття рішень, а також на стимулювання екологічно відповідальної поведінки.

Моделі управління ресурсами в житлових комплексах можуть бути класифіковані за рівнем централізації, формами власності на об’єкти інфраструктури, рівнем цифровізації та ступенем інтеграції з іншими міськими системами. У централізованих моделях управління відповідальність за управління лежить на єдиній керуючій компанії, яка координує всі процеси та забезпечує повний контроль за використанням ресурсів. У децентралізованих моделях, зокрема в рамках ОСББ, функції управління розподіляються між мешканцями, що дає змогу оперативно реагувати на локальні потреби, але водночас вимагає більшої активності з боку співвласників.

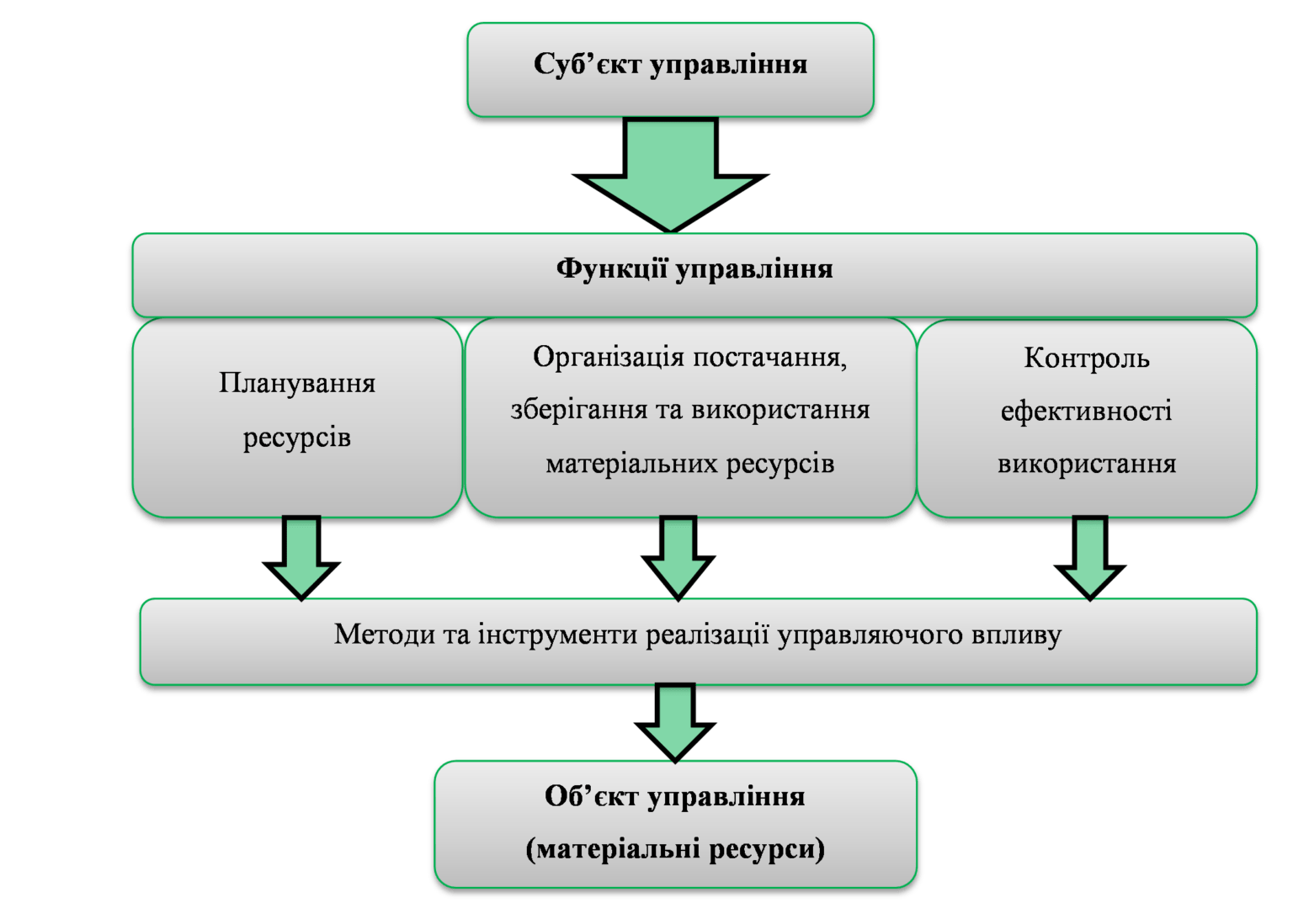


Рисунок 1.5 – Приклад моделі управління ресурсами в житлових комплексах

*Джерело: [7]*

Інноваційні моделі передбачають впровадження концепцій "розумного будинку" (smart house) та "розумного міста" (smart city), де управління ресурсами здійснюється за допомогою IoT-технологій, сенсорних мереж, хмарних сервісів та мобільних додатків. Такі моделі дозволяють автоматизувати процеси контролю споживання, передбачати технічні збої, оптимізувати витрати на утримання та забезпечити високий рівень комфорту для мешканців.

Загалом, вибір методів і моделей управління залежить від специфіки житлового комплексу, його масштабу, технічної оснащеності, рівня організаційної структури та фінансових можливостей. В умовах трансформації ринку нерухомості та підвищення вимог до якості житла ефективне управління ресурсами стає ключовим чинником забезпечення сталого функціонування та розвитку житлових об'єктів.

1. **Основна частина**
   1. **Постановка задачі та техніко-економічне обґрунтування**

У сучасних умовах ефективне управління багатофункціональними житловими комплексами вимагає централізованого підходу до обліку та бронювання доступних ресурсів. До таких ресурсів можуть належати апартаменти, конференц-зали, паркомісця, спортивні та оздоровчі послуги, а також ресторанні зони, що розташовані на території комплексу. Розрізненість обліку або його відсутність призводить до нераціонального використання простору, втрат прибутку, складнощів у плануванні та зниження якості обслуговування клієнтів.

У зв’язку з цим постає потреба у створенні програмного додатку, який забезпечуватиме централізоване управління всіма ресурсами комплексу. Такий інструмент має бути орієнтований на адміністративний персонал: менеджерів, адміністраторів або директора житлового комплексу. Кінцева мета — спростити процес прийняття рішень, підвищити прозорість обліку, уникнути конфліктів у графіку бронювання та поліпшити загальну ефективність управління.

Функціональність додатку охоплює:

* управління апартаментами (статус, тип, доступність, періоди оренди);
* бронювання конференц-залів з урахуванням розкладу та місткості;
* облік паркомісць із фіксацією їхнього статусу та резервуванням;
* обслуговування додаткових послуг комплексу (спортзал, обладнання тощо);
* бронювання місць у ресторані (дата, час, кількість гостей, вибір столу);
* перегляд звітів щодо завантаженості та доходів від оренди й сервісів;
* інтерактивне відображення поточного стану всіх ресурсів.

Особливістю цього рішення є його безкоштовність: додаток створюється як універсальний інструмент для внутрішнього використання, без ліцензійних чи абонентських платежів. Це дозволяє забезпечити цифрову трансформацію управління житловим комплексом без додаткових витрат на стороннє програмне забезпечення.

Реалізація такого додатку дозволить підвищити якість управління ресурсами, забезпечити прозорий облік та зменшити витрати на адміністративне обслуговування. У перспективі це також сприятиме підвищенню рівня задоволеності мешканців та гостей комплексу завдяки оперативному та точному наданню послуг.

* 1. **Розробка** **алгоритму функціонування програми**

Програма для управління ресурсами житлового комплексу реалізована за модульним принципом, що дозволяє окремо керувати апартаментами, конференц-залами, рестораном, паркомісцями, послугами та користувачами. Основні етапи функціонування наведено нижче:

**1. Вхід у систему**

Користувач вводить логін і пароль. Залежно від ролі (наприклад, директор або менеджер) програма відкриває відповідні функціональні модулі.

**2. Головна панель**

Після авторизації відкривається головне меню з іконками для переходу до керування апартаментами, бронювання залів, паркінгу, ресторану, додаткових послуг і користувачів.

**3. Робота з модулями**

Кожен модуль має єдину логіку:

* **Завантаження даних:** з бази даних завантажується список відповідних ресурсів.
* **Фільтрація:** можна швидко знайти потрібний запис за статусом, датою чи типом.
* **Редагування:** доступне оновлення параметрів об'єкта (ціна, час, опис).
* **Бронювання:** інтерфейс дозволяє легко обрати дату й час.
* **додавання/видалення:** зміни відразу зберігаються, а при видаленні підтверджуються користувачем.

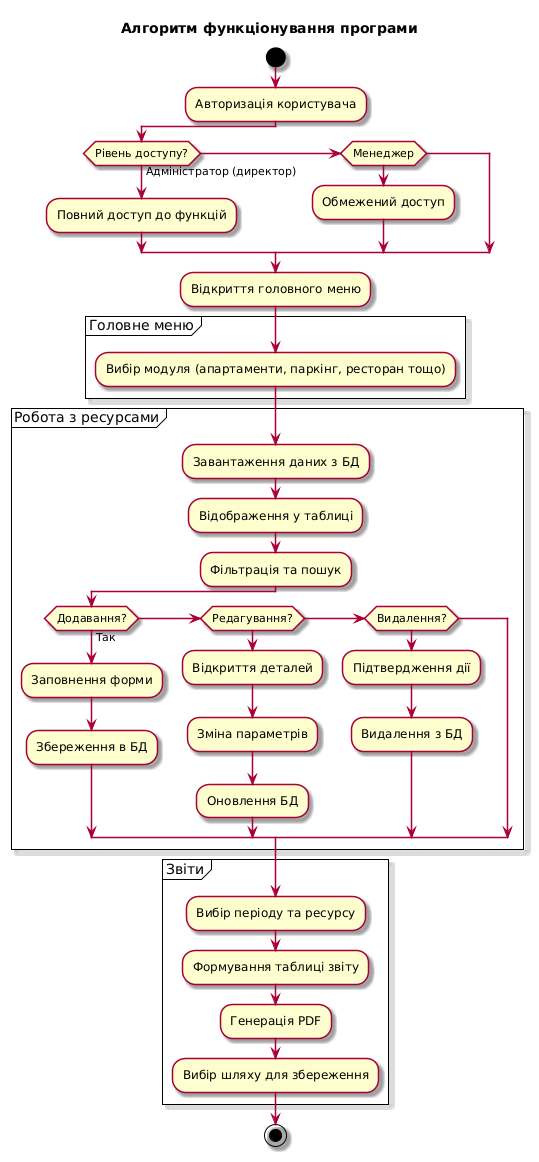


Рисунок 2.1 – Алгоритм функціонування програми

*Джерело: побудовано автором*

**4. Звіти**

Користувач обирає період і тип ресурсу. Програма генерує звіт (зайнятість, дохід) і експортує його у PDF.

**5. Збереження даних**

Усі зміни одразу зберігаються в базі даних.

Цей алгоритм забезпечує прозорість та зручність роботи програми для адміністраторів (директорів) та менеджерів.

* 1. **Вибір середовища розробки та технологій**

Для створення програмного додатку з управління ресурсами житлового комплексу було обрано інструменти та технології, які поєднують зручність розробки, сучасний інтерфейс і кросплатформені можливості.

У якості основного середовища розробки використано **Visual Studio 2022** — потужну IDE з підтримкою C# та WPF, вбудованим редактором XAML, інструментами налагодження й інтеграцією з системами контролю версій. Це дозволяє легко керувати проєктом на всіх етапах — від створення до тестування та публікації [8].

Основна мова програмування — **C#**, що відзначається гнучкістю, сучасним синтаксисом і потужною стандартною бібліотекою. Завдяки підтримці LINQ, асинхронного програмування та інтеграції з фреймворком WPF, C# дозволяє ефективно реалізовувати логіку і зручний інтерфейс [9].

Графічний інтерфейс реалізовано за допомогою **Windows Presentation Foundation (WPF)**, що базується на використанні мови розмітки XAML. Це дає можливість створювати адаптивні інтерфейси з прив’язкою даних (Data Binding), підтримкою стилів, шаблонів і анімацій.

Зберігання інформації реалізовано на базі **SQLite** — вбудованої бази даних, яка не потребує окремого сервера. Вона забезпечує стабільність, транзакційність (ACID), і проста у використанні завдяки інтеграції з Entity Framework Core. Уся база даних міститься в одному файлі, що спрощує розгортання та обслуговування програми [10].

Для попереднього проєктування та тестування структури бази даних було використано інструмент **DB Browser for SQLite** — безкоштовний графічний інтерфейс для роботи з файлами SQLite [11].

За допомогою цього інструменту створено базу даних HotelDB.sqlite, яка містить основні таблиці для зберігання інформації про користувачів, квартири, послуги та платежі. У DB Browser зручно створювати таблиці через вкладку *Database Structure*, додавати поля, типи даних, ключі та зв’язки.

Крім створення структури, інструмент дозволяє наповнити таблиці тестовими даними, використовуючи вкладку *Browse Data*. Це дає змогу перевірити правильність запитів і логіку взаємозв’язків між таблицями ще до підключення бази до самого додатку.



Рисунок 2.2 – Обраний стек технологій

*Джерело: побудовано автором*

Для експорту даних у формат PDF використовується бібліотека **iTextSharp**, яка дозволяє створювати структуровані документи з таблицями та стилізованим форматуванням. Також у проекті використовуються допоміжні пакети System.Data.Sqlite та System.Text.Encoding.CodePages.

Обраний стек технологій дозволяє зосередитися на розробці функціональності без зайвих складнощів. Для розробника це означає зручний UI-дизайнер, сучасні інструменти відладки та висока продуктивність. Для користувача — швидкодія, зрозумілий інтерфейс і можливість формувати звіти у PDF. А для проєкту загалом — низькі вимоги до інфраструктури та просте оновлення функціоналу.

* 1. **Практична реалізація програмного додатку**

**2.4.1.** **Архітектура програмного забезпечення**

Програмний додаток **HotelServices** реалізований за принципами багаторівневої архітектури з чітким розділенням відповідальностей (Separation of Concerns). Загалом архітектура охоплює три основні рівні: представлення, бізнес-логіку та доступ до даних. Система складається з трьох основних рівнів: представлення (UI), бізнес-логіки та доступу до даних.

**1. Рівень представлення (Presentation Layer)**

Відповідає за інтерфейс користувача та взаємодію з ним. Реалізований на WPF/XAML з використанням шаблону MVVM (Model-View-ViewModel):

* + MainWindow.xaml – головне вікно програми з фреймом для навігації.
  + DashboardPage.xaml – головна сторінка з кнопками доступу до функцій.
  + ResourcePage.xaml – сторінка управління ресурсами (апартаменти, парковки тощо).
  + UserManagementPage.xaml – сторінка керування користувачами.
  + Діалогові вікна (Dialogs/):
    - ResourceEditDialog.xaml – форма додавання/редагування ресурсу.
    - ReportDialog.xaml – вікно генерації звітів.
    - UserEditDialog.xaml – вікно додавання користувачів (для директора).

Ключові особливості:

* + Використання DataBinding для зв'язку UI з даними.
  + Стилізація через App.xaml (глобальні стилі кнопок, тексту).
  + Навігація між сторінками через Frame.Navigate().

**2. Рівень бізнес-логіки (Business Logic Layer)**

Містить основну логіку програми та обробку даних:

* + AuthService.cs – аутентифікація користувачів (перевірка логінів/паролів).
  + DataService.cs – центральний сервіс для роботи з даними:
    - CRUD-операції для ресурсів (GetAllResources, AddResource).
    - Фільтрація за типами ресурсів (GetResourcesByType).
    - Генерація звітів (GenerateReport).
  + Класи моделей (Models/):
    - Resource.cs – сутність ресурсу (властивості: Id, Name, Price, Status, Dates).
    - User.cs – сутність користувача (логін, пароль, роль).

Принципи роботи:

* + Інкапсуляція логіки в окремі сервіси.
  + Валідація даних перед збереженням (наприклад, перевірка конфліктів бронювань).
  + Події (Events) для оновлення UI при зміні даних.

**3. Рівень доступу до даних (Data Access Layer)**

Відповідає за зберігання та відновлення даних:

* + SQLite – основна база даних. Використовується Entity Framework Core (або ручні SQL-запити) для:
    - Створення таблиць (Resources, Users).

Завдяки цій структурі додаток залишається модульним, гнучким до розширення та зручним у підтримці. Бізнес-логіка відокремлена від інтерфейсу, що дозволяє легко масштабувати функціональність, вводити нові типи ресурсів або змінювати поведінку окремих компонентів без втручання у візуальну частину. Безпека реалізована на рівні авторизації користувачів згідно з ролями.

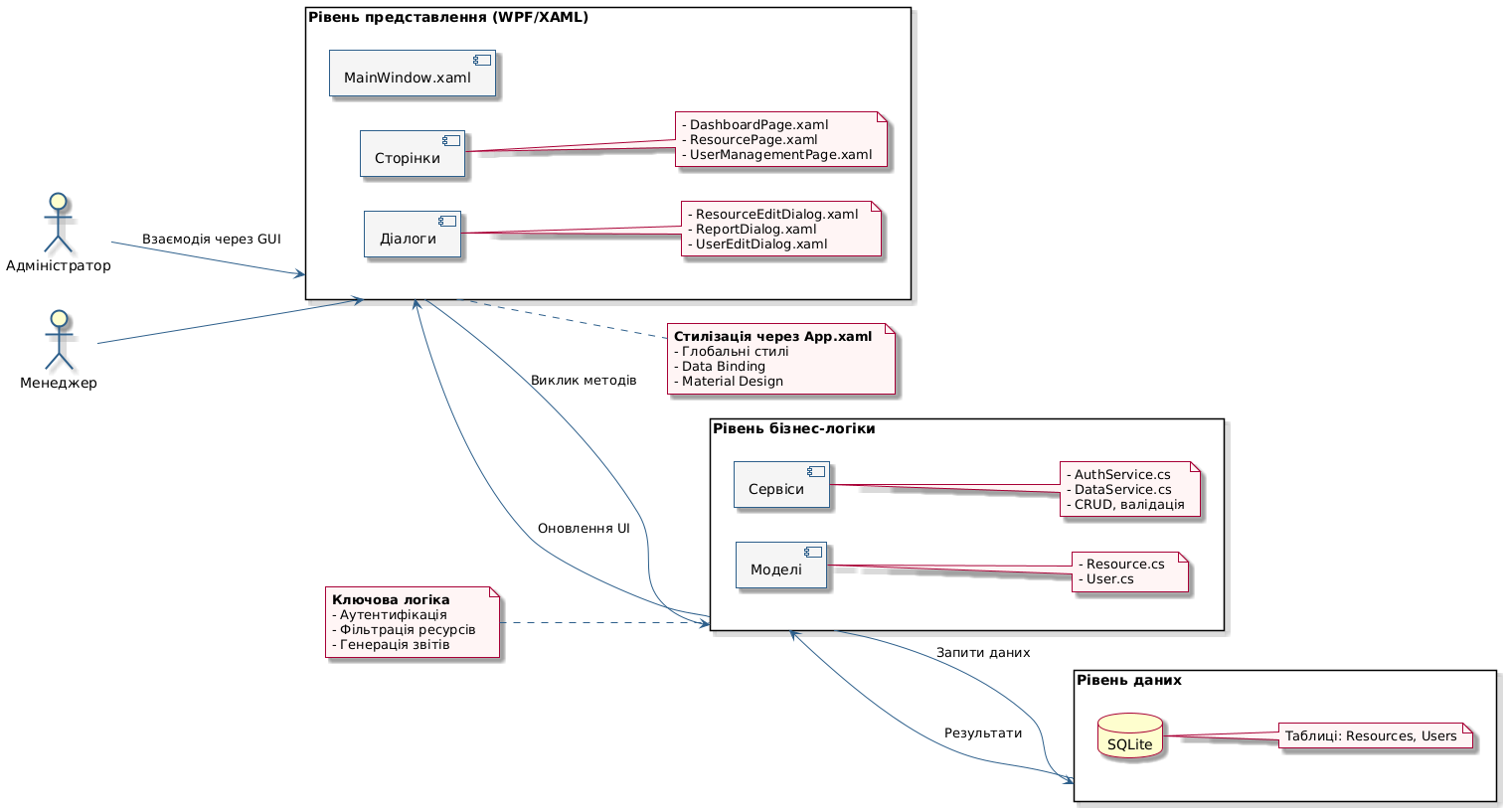


Рисунок 2.3 – Архітектура програмного забезпечення

*Джерело: побудовано автором*

Підсумовуючи, архітектура програми розділена на три основні рівні:

Представлення (UI) реалізоване з використанням WPF і шаблону MVVM. Основні компоненти включають MainWindow.xaml, сторінки (Dashboard, ResourcePage, UserManagementPage) та діалогові вікна (наприклад, ResourceEditDialog.xaml, ReportDialog.xaml). Завдяки Frame.Navigate() забезпечено зручну навігацію, а механізм Data Binding дозволяє динамічно оновлювати вміст інтерфейсу. Всі стилі та шаблони оформлення зберігаються в App.xaml.

Бізнес-логіка зосереджена в окремих сервісах, зокрема DataService, який відповідає за CRUD-операції, фільтрацію та валідацію даних. Аутентифікацію виконує AuthService. Моделі User та Resource інкапсулюють дані, які передаються між рівнями, забезпечуючи зручність супроводу та тестування.

Доступ до даних забезпечує SQLite. Інформація зберігається у файлі HotelDB.sqlite, доступ до якого здійснюється через Entity Framework Core або SQL-запити.

**2.4.2 Розробка інтерфейсу користувача**

Розробка інтерфейсу користувача програмного додатку розпочалась із етапу візуального проектування. На цьому етапі були створені первинні макети всіх ключових сторінок за допомогою графічного редактора. Основною метою було визначення загальної структури інтерфейсу, місця розташування елементів управління, а також навігаційних потоків між сторінками. У цей момент увага не зосереджувалась на деталізації стилів або конкретному функціональному наповненні — акцент робився саме на логічній організації та зручності користувача.

В процесі візуального прототипування були створені спрощені схеми для головного вікна, інтерфейсу адміністратора, сторінки управління ресурсами та сторінки звітів. У кожному з макетів чітко виділялася зона навігації, де розташовувалися кнопки з піктограмами, взятими з папки Assets. Центральна частина макету була призначена для відображення змінного вмісту таблиць із ресурсами та іншої динамічної інформації. Також у макетах було закладено місце для панелей з фільтрами, полями пошуку та іншими допоміжними елементами керування.

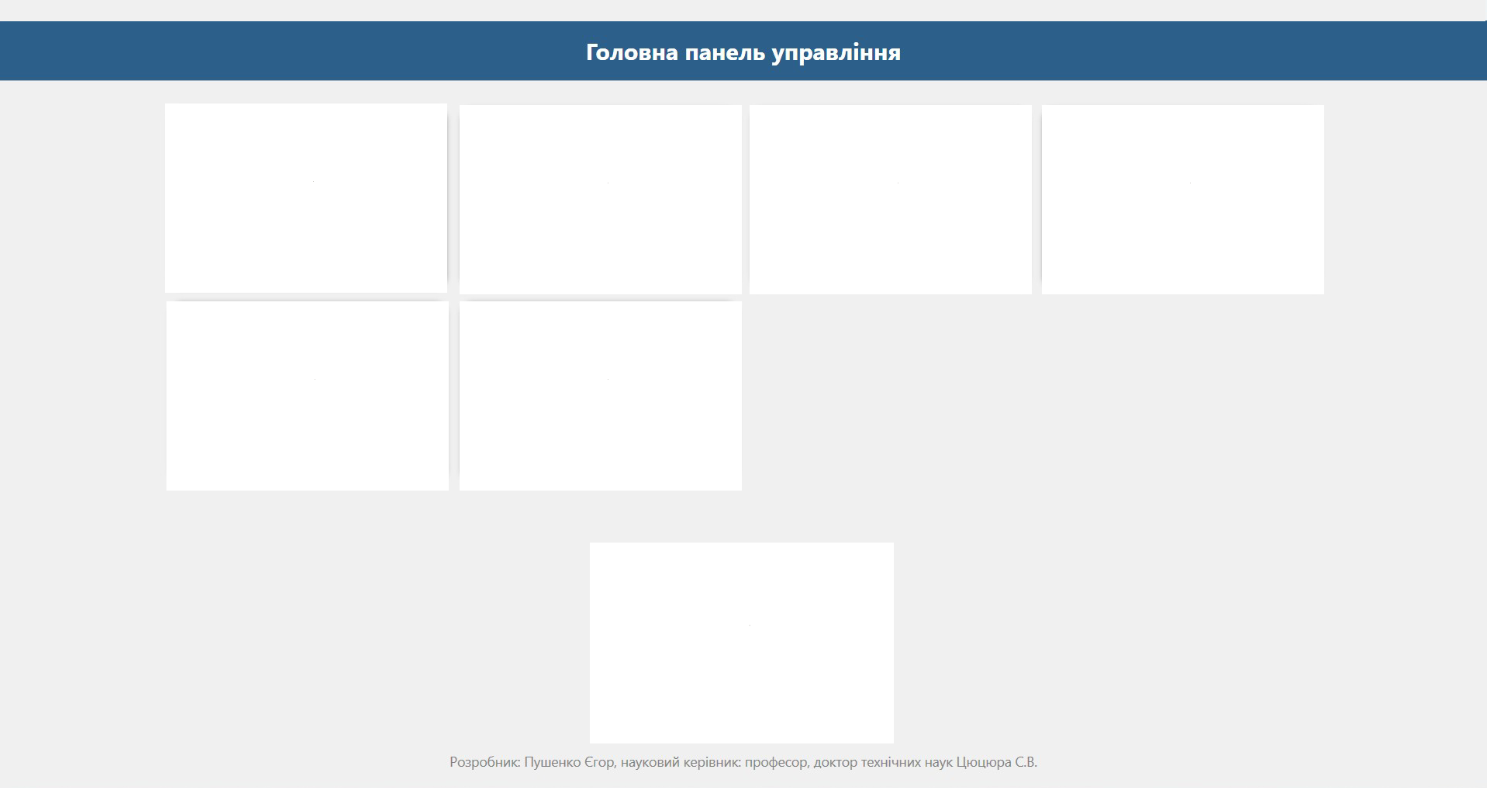


Рисунок 2.4 – Низькополігональний макет головного вікна

*Джерело: побудовано автором*

Після погодження логічної структури макетів почалась їх адаптація у середовищі WPF із використанням мови розмітки XAML. Кожне вікно реалізовувалось за допомогою ієрархічних контейнерів, таких як Grid і StackPanel, що дозволяло точно відтворити структуру, розроблену на етапі прототипування. Елементи інтерфейсу, такі як кнопки, текстові поля та таблиці, були реалізовані через стандартні WPF-контроли — Button, TextBox, DataGrid тощо. Паралельно проводилась прив’язка елементів до відповідних джерел даних, що дозволяло автоматично заповнювати таблиці або відображати інформацію без додаткових ручних операцій.

Після побудови базового каркасу інтерфейсу розпочато етап деталізації. На цьому кроці до макетів додавались анімаційні ефекти, які покращували сприйняття переходів між сторінками.

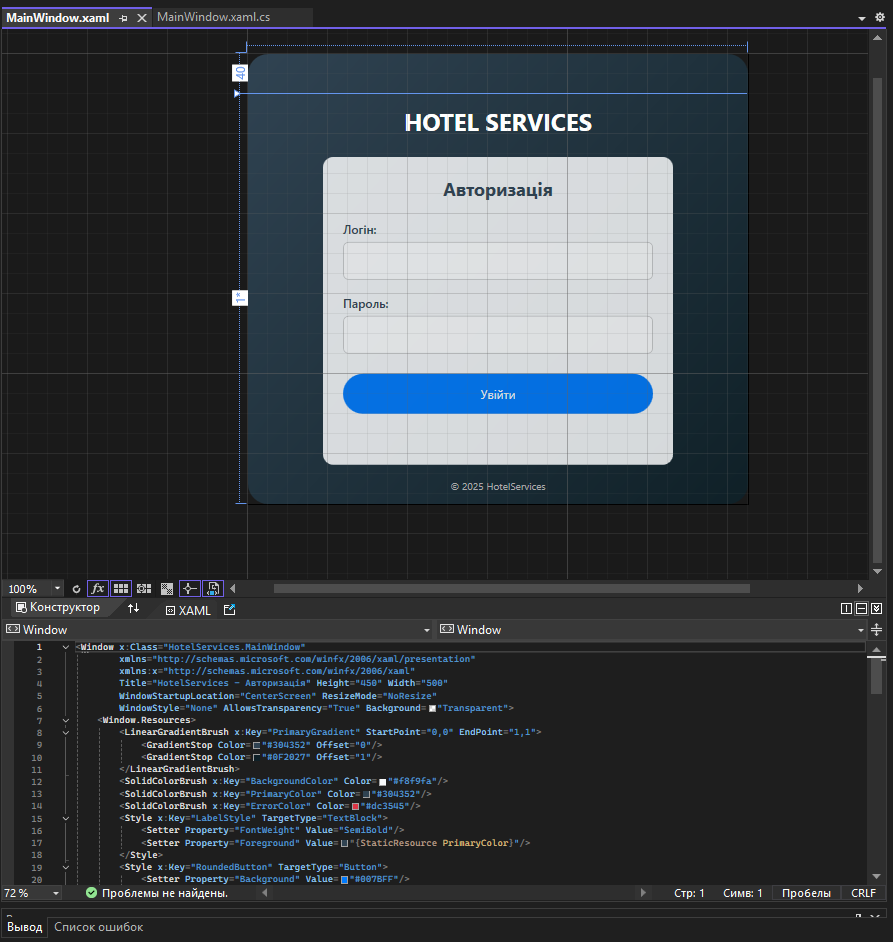


Рисунок 2.5 –XAML-розмітка з використанням контейнерів Grid і StackPanel

*Джерело: побудовано автором*

Особливу увагу було приділено стилізації — у файлі App.xaml було створено набір шаблонів для елементів управління, зокрема кнопок. Наприклад, для кнопок з піктограмами використовувався спеціальний стиль ImageButtonStyle, що забезпечував однаковий зовнішній вигляд на всіх сторінках. Крім того, інтерфейс підтримує адаптивність: при зміні розміру вікна всі основні контейнери автоматично перебудовуються, зберігаючи зручне розміщення елементів.

Одним із прикладів реалізації є сторінка керування ресурсами (ResourcePage.xaml). Її структура побудована з використанням Grid з чотирма рядками. У верхньому блоці розміщено заголовок з TextBlock та відображенням поточної дати. Другий рядок містить панель дій з набором кнопок управління StackPanel. Третій рядок представляє панель фільтрів з полем пошуку та випадаючим списком. Нижня частина сторінки відведена під DataGrid, який відображає дані ресурсів з різними колонками. Усі кнопки на сторінці стилізовані відповідно до шаблонів, визначених у App.xaml, що забезпечує узгоджений дизайн на всіх рівнях взаємодії з додатком.

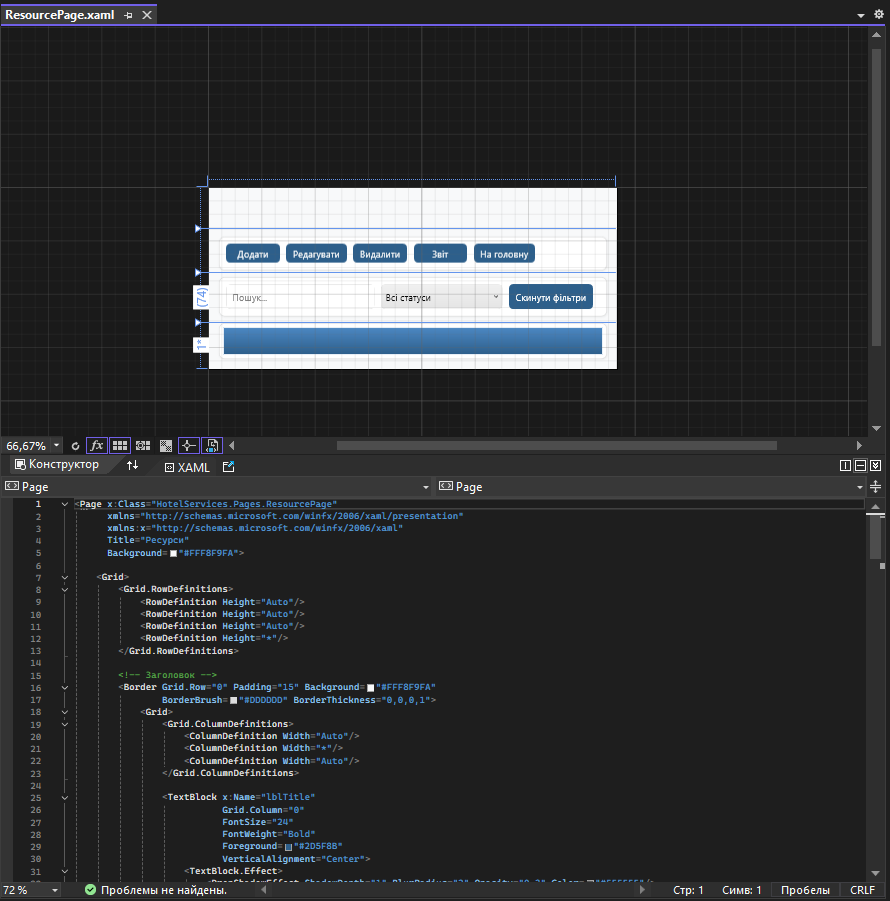


Рисунок 2.6 – XAML-структура сторінки ResourcePage.xaml

*Джерело: побудовано автором*

Для діалогових вікон, наприклад для діалогу редагування ресурсів (ResourceEditDialog.xaml) обрано сіткову структуру (Grid) з дев'ятьма рядками, що дозволило гармонійно розподілити елементи:

* Верхній блок містив заголовок із динамічним текстом ("Додати" або "Редагувати").
* Центральна частина включала поля введення (TextBox) для назви, опису та ціни, а також ComboBox для статусу.
* Для вибору дати та часу додано DatePicker і два ComboBox (години та хвилини), об'єднані в горизонтальні StackPanel.
* Додаткові параметри (наприклад, кількість кімнат для апартаментів) динамічно додавалися у нижній StackPanel через код (additionalFieldsPanel).
* Кнопки "Зберегти" та "Скасувати" розміщувалися праворуч у окремому StackPanel.

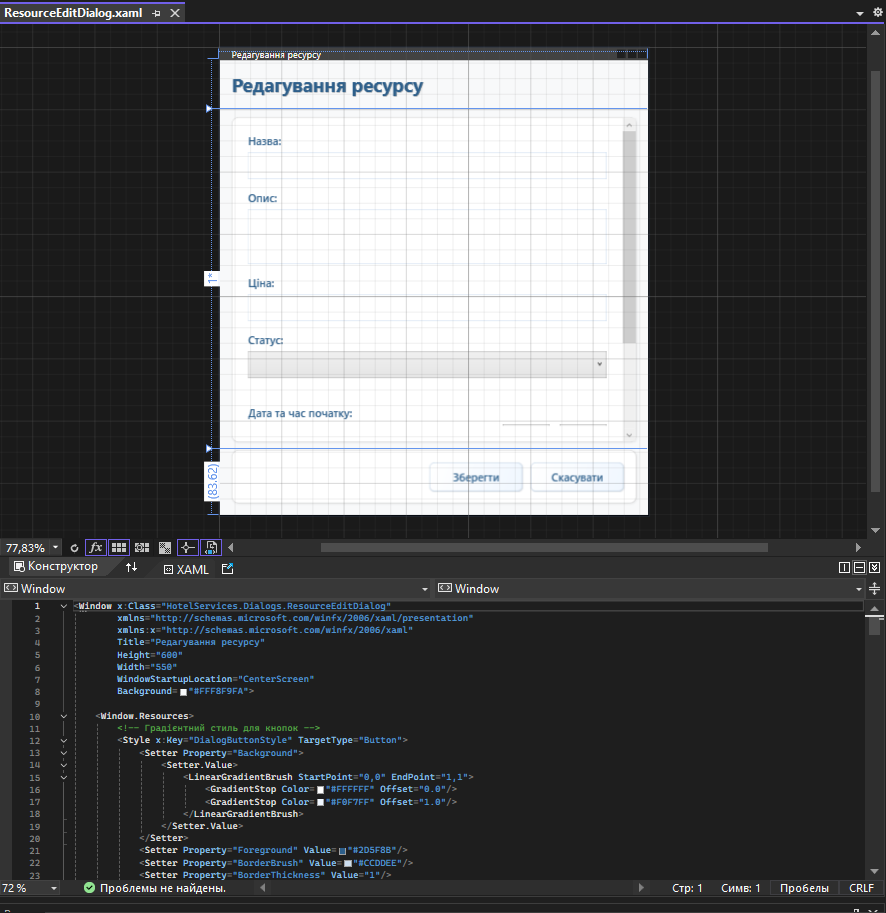


Рисунок 2.7 – XAML-структура діалогової сторінки ResourceEditDialog.xaml

*Джерело: побудовано автором*

Отже, розробка інтерфейсу користувача в межах проєкту базувалася на поетапному підході: від візуального прототипування до реалізації інтерактивних елементів у середовищі WPF. Старт роботи передбачав створення макетів усіх основних сторінок у графічному редакторі, де визначалась загальна структура, логіка переходів між розділами та розташування ключових елементів. Ці макети надалі лягли в основу XAML-розмітки, де використовувалися гнучкі компоненти, як-от Grid і StackPanel, для формування чіткої й масштабованої структури сторінок.

Завдяки інкрементальному підходу інтерфейс поступово наповнювався логікою: спочатку створювався каркас сторінки, потім підключалися джерела даних і стилі, винесені у файл App.xaml, що дозволило досягти єдиного візуального вигляду елементів у всьому додатку. Особлива увага приділялася деталізації — реалізовано плавні анімації переходів між вікнами, адаптивну поведінку елементів при зміні розмірів вікна, а також інтерактивну логіку, як-от умовне відображення полів залежно від вибраних параметрів.

Важливою деталлю стало оформлення головного вікна програми — для нього було використано зображення з папки Assets, що надало інтерфейсу додаткової естетичної привабливості.

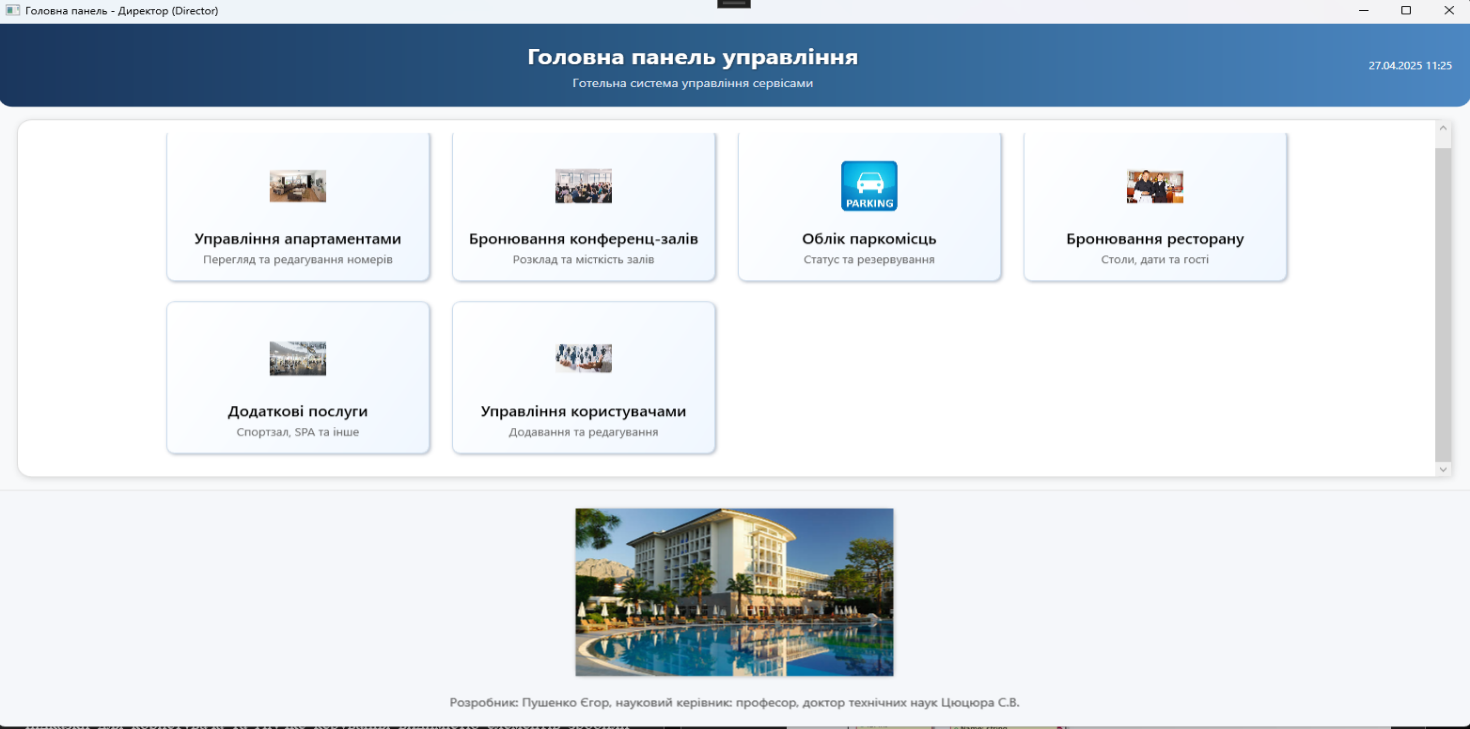


Рисунок 2.8 – Інтерфейс головного вікна програми

*Джерело: побудовано автором*

Діалогові вікна, які відігравали роль точок введення та редагування даних, створювалися з урахуванням контексту їх використання. Кожне таке вікно було оптимізоване під конкретну задачу, мало компактну й логічно організовану структуру з можливістю динамічного наповнення полями. Валідація даних, підказки для користувача та гнучке керування видимістю елементів зробили взаємодію з діалогами простою й інтуїтивною. Результатом стало зручне, чисте й функціональне середовище, в якому користувач легко орієнтується, швидко знаходить потрібні інструменти і без зайвих дій виконує поставлені задачі.

Розробка інтерфейсу користувача завершилася після повної реалізації всіх ключових елементів, включаючи діалогові вікна, сторінки адміністрування та управління ресурсами. Завдяки поєднанню функціональності, інтуїтивності та візуальної чіткості, готовий інтерфейс відповідає вимогам зручності для кінцевого користувача.

**2.4.3 Реалізація основних функціональних модулів**

Метою реалізації цих модулів є створення зручної, функціональної та безпечної інформаційної системи для управління готельними ресурсами (апартаментами, залами, парковками тощо). Кожен модуль виконує конкретне завдання: від забезпечення доступу користувачів до системи, управління даними про ресурси та користувачів, до генерації аналітичних звітів. Завдяки модульній структурі забезпечується гнучкість, масштабованість, ефективне обслуговування системи та персоналізований інтерфейс для різних ролей користувачів.

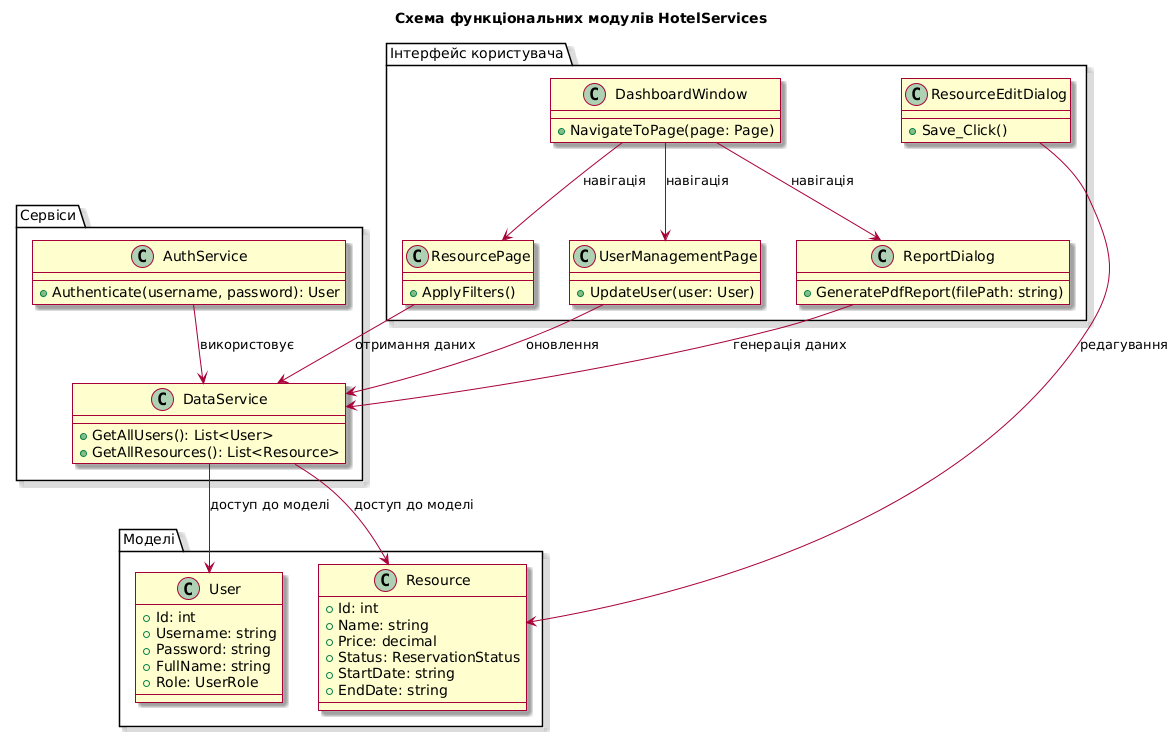


Рисунок 2.9 – Схема основних функціональних модулів програми

*Джерело: побудовано автором*

**1. Система автентифікації (AuthService.cs)**

Реалізує перевірку облікових даних користувачів. Використовує DataService для отримання даних про користувачів. Основні методи:

        public AuthService()

        {

            \_dataService = new DataService();

        }

        public User Authenticate(string username, string password)

        {

            var user = \_dataService.GetAllUsers().FirstOrDefault(u => u.Username == username && u.Password == password);

            return user;

        }

Після успішної автентифікації створюється головне вікно програми (DashboardWindow) з урахуванням ролі користувача.

**2. Головний інтерфейс (DashboardWindow.xaml.cs)**

Центральний інтерфейс програми містить:

* Навігаційне меню з кнопками доступу до різних модулів
* Фрейм для відображення контенту (ResourcePage, UserManagementPage тощо)
* Відображення поточної інформації про користувача

Приклад навігації:

public void NavigateToPage(Page page)

{

    mainFrame.Navigate(page);

}

**3. Управління ресурсами (ResourcePage.xaml.cs)**

Ключовий модуль для роботи з апартаментами, залами, парковками тощо. Основні функції:

* Відображення списку ресурсів у DataGrid
* Фільтрація за статусом та пошук за назвою
* CRUD операції через діалогові вікна

Приклад фільтрації:

private void ApplyFilters()

{

    var filtered = \_resources.Where(r =>

        r.Name.Contains(SearchTextBox.Text, StringComparison.OrdinalIgnoreCase) &&

        (StatusFilterComboBox.SelectedIndex == 0 || r.Status == (ReservationStatus)StatusFilterComboBox.SelectedItem));

    resourcesGrid.ItemsSource = filtered.ToList();

}

**4. Діалогові вікна редагування (ResourceEditDialog.xaml.cs)**

Спеціалізовані форми для роботи з конкретними типами ресурсів. Містять:

* Універсальні поля (назва, опис, ціна)
* Специфічні поля для кожного типу ресурсів
* Календарі для вибору дат бронювання
* Валідацію введених даних

Приклад обробки збереження:

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

{

    if (!ValidateInput()) return;

    Resource.Name = txtName.Text;

    Resource.Price = decimal.Parse(txtPrice.Text);

    // ... інші поля

    DialogResult = true;

    Close();

}

Повний код ResourceEditDialog.xaml.cs наведено в додатку А.

**5. Звітність (ReportDialog.xaml)**

Генерація звітів у форматі PDF з використанням iTextSharp. Особливості:

* Вибір періоду для звіту
* Фільтрація за типом ресурсу
* Експорт до PDF з професійним оформленням
* Розрахунок доходів та тривалості бронювань

Приклад генерації PDF:

private void GeneratePdfReport(string filePath)

{

    using (var document = new Document(PageSize.A4.Rotate()))

    {

        PdfWriter.GetInstance(document, new FileStream(filePath, FileMode.Create));

        document.Open();

        // Додавання заголовків

        document.Add(new Paragraph($"Звіт за період: {StartDate} - {EndDate}"));

        // Додавання таблиці з даними

        PdfPTable table = new PdfPTable(5);

        table.AddCell("Назва");

        table.AddCell("Статус");

        // ... інші стовпці

        foreach (var resource in reportData)

        {

            table.AddCell(resource.Name);

            table.AddCell(resource.Status.ToString());

            // ... інші дані

        }

        document.Add(table);

    }

}

**6. Управління користувачами (UserManagementPage.xaml)**

Модуль для адміністраторів системи. Дозволяє:

* Переглядати список користувачів
* Додавати/редагувати/видаляти облікові записи
* Призначати ролі (менеджер/адміністратор)

Приклад оновлення користувача:

private void UpdateUser(User user)

{

    using (var connection = new SQLiteConnection(\_connectionString))

    {

        connection.Open();

        var query = @"UPDATE Users SET Username=@Username, Password=@Password,

                     FullName=@FullName, Role=@Role WHERE Id=@Id";

        using (var cmd = new SQLiteCommand(query, connection))

        {

            cmd.Parameters.AddWithValue("@Id", user.Id);

            // ... інші параметри

            cmd.ExecuteNonQuery();

        }

    }

}

**7. База даних (DataService.cs)**

SQLite-база даних з двома основними таблицями:

CREATE TABLE Users (

    Id INTEGER PRIMARY KEY AUTOINCREMENT,

    Username TEXT UNIQUE NOT NULL,

    Password TEXT NOT NULL,

    FullName TEXT NOT NULL,

    Role INTEGER NOT NULL

);

CREATE TABLE Resources (

    Id INTEGER PRIMARY KEY AUTOINCREMENT,

    Type INTEGER NOT NULL,

    Name TEXT NOT NULL,

    Description TEXT,

    Price REAL NOT NULL,

    Status INTEGER NOT NULL,

    -- Специфічні поля для різних типів ресурсів

    StartDate TEXT,

    EndDate TEXT

);

**8. Моделі даних (Models/)**

Визначають структуру основних сутностей:

**User.cs**:

public class User

{

    public int Id { get; set; }

    public string Username { get; set; }

    public string Password { get; set; }

    public string FullName { get; set; }

    public UserRole Role { get; set; }

}

**Resource.cs**:

    public class Resource

    {

        public int Id { get; set; }

        public ResourceType Type { get; set; }

        public string Name { get; set; }

        public string Description { get; set; }

        public decimal Price { get; set; }

        public ReservationStatus Status { get; set; }

        // Для апартаментів

        public int? Rooms { get; set; }

        public bool? IsLuxury { get; set; }

        // Для конференц-залів

        public int? Capacity { get; set; }

        // Для паркомісць

        public string? ParkingNumber { get; set; }

        // Для ресторану

        public int? TableNumber { get; set; }

        public int? Guests { get; set; }

        // Для додаткових послуг

        public string? ServiceType { get; set; }

        // Бронювання

        public DateTime? StartDate { get; set; }

        public DateTime? EndDate { get; set; }

        public int? ReservedByUserId { get; set; }

        public decimal TotalIncome { get; set; } // Загальний дохід

        public double OccupancyDuration { get; set; } // Тривалість у годинах

    }

У результаті розробки реалізовано ключові функціональні модулі, які формують повноцінну систему керування готельними послугами. Кожен модуль має чітке призначення, взаємодіє з іншими через єдину архітектуру, забезпечує збереження даних, гнучке адміністрування та зручний інтерфейс. Такий підхід дозволяє ефективно автоматизувати процеси в межах готельного комплексу та забезпечити високий рівень обслуговування користувачів.

1. **Заключна частина**
   1. **Опис системи та взаємодії модулів**

Взаємодія модулів у системі **HotelServices** відбувається за чітко визначеним сценарієм, що відображений у діаграмі, і залежить від ролі користувача, який увійшов у систему. Нижче подано пояснення логіки згідно зі схемою (рис. 3.1), а також уточнення щодо доступу до функцій залежно від ролі.

Користувач починає роботу, вводячи свої облікові дані в **MainWindow**. Ці дані передаються до модуля **AuthService**, який через **DataService** перевіряє наявність користувача з відповідним логіном і паролем. Якщо такі дані знайдені, система повертає авторизованого користувача, і **MainWindow** відкриває **DashboardWindow** — головне вікно системи.

У **DashboardWindow** користувач може переходити до модуля управління ресурсами — **ResourcePage**. Цей модуль отримує дані про ресурси відповідного типу через **DataService**. Усі ресурси відображаються для перегляду, фільтрації чи пошуку. При редагуванні певного ресурсу відкривається **ResourceEditDialog**, який після внесення змін передає оновлені дані назад у **DataService** для збереження. Успішне збереження спричиняє оновлення відображення ресурсу в **ResourcePage**.

Для аналітики користувач може ініціювати відкриття **ReportDialog**, який звертається до **DataService** для отримання потрібних звітних даних. Отримані дані потім передаються у **PDFGenerator**, що формує звіт у PDF-форматі.

Ключовий момент стосується ролі користувача. Якщо авторизований користувач має роль **Адміністратор** (що в контексті системи відповідає директору), він отримує додатковий доступ до модуля **UserManagementPage**. Цей модуль дозволяє адміністратору переглядати всіх користувачів, відкривати **UserEditDialog** для редагування їхніх профілів, змінювати логіни, паролі, ПІБ і найважливіше — **призначати ролі**.

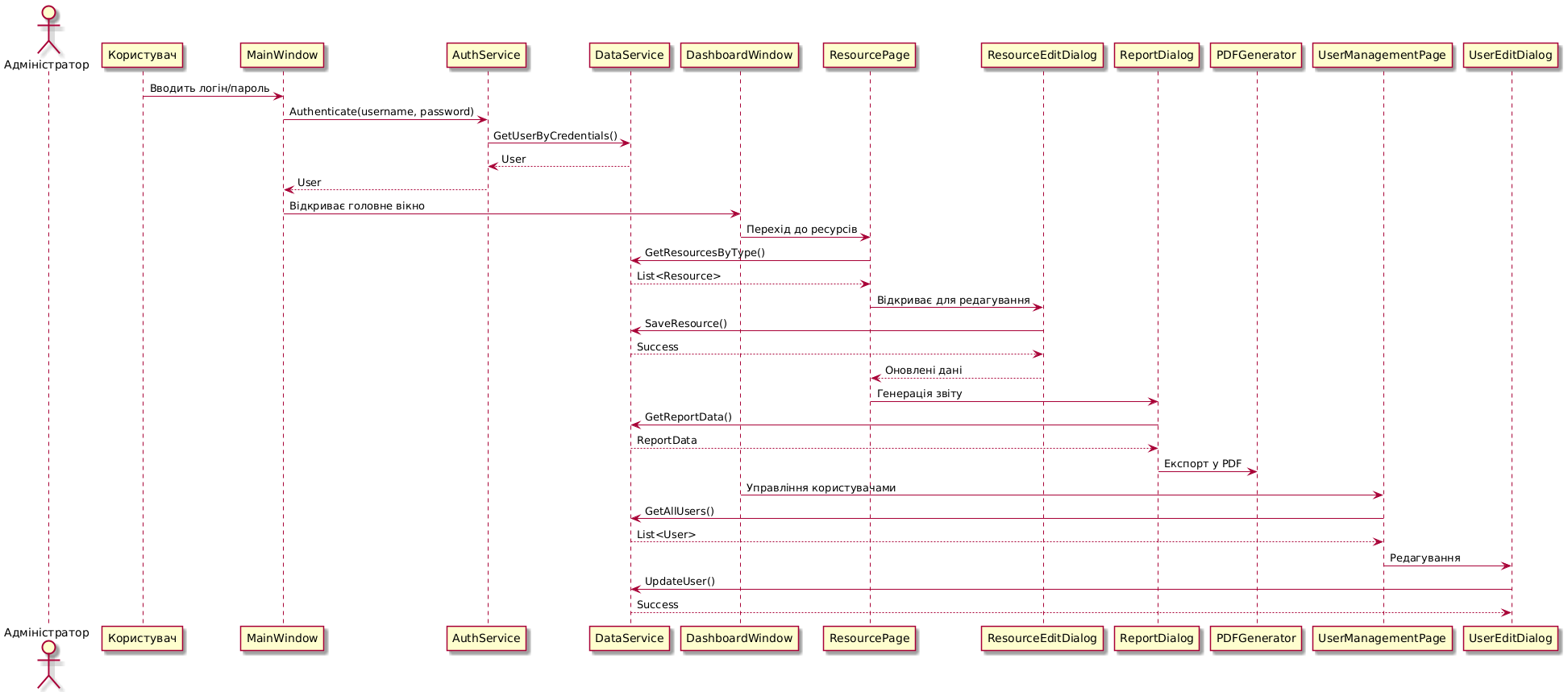


Рисунок 3.1 – Схема взаємодії модулів системи

*Джерело: побудовано автором*

Адміністратор може додавати нових користувачів, видаляти або оновлювати вже наявних. Доступні ролі — **Менеджер** або **Адміністратор**, і лише останній має змогу керувати персоналом.

Якщо ж авторизований користувач має роль **Менеджер**, він працює тільки з ресурсами та звітами, не маючи доступу до модуля управління користувачами. Таким чином, система реалізує чіткий розподіл прав доступу відповідно до ролі, що забезпечує як безпеку, так і структурованість роботи.

Цей механізм є важливою частиною загальної архітектури HotelServices і реалізований через перевірку ролі користувача після автентифікації перед відкриттям відповідних розділів у **DashboardWindow**.

**3.2 Інструкція для розробника та користувача**

**Інструкція для розробника**

1. Встановлення та налаштування

1. Відкрийте проект у Visual Studio (або іншій IDE з підтримкою .NET).
2. Встановіть необхідні пакети:
   * System.Data.SQLite (для роботи з базою даних)
   * iTextSharp (для генерації PDF-звітів)
3. Запустіть проект:
   * Переконайтеся, що збірка пройшла успішно.
   * При першому запуску автоматично створиться база даних HotelDB.sqlite з тестовими даними.

2. Архітектура проекту

* Models/ – класи User та Resource (сутності бази даних).
* Services/ – бізнес-логіка (AuthService, DataService).
* Pages/ – основні сторінки додатку (Dashboard, ResourcePage, UserManagement).
* Dialogs/ – діалогові вікна для редагування даних.
* Assets/ – зображення та іконки.

3. Розширення функціоналу

1. Додати новий тип ресурсу:
   * Додайте нове значення в enum ResourceType.
   * Розширте метод LoadAdditionalFields() у ResourceEditDialog.xaml.cs.
   * Додайте відповідні поля в SQLite-таблицю Resources.
2. Додати новий звіт:
   * Створіть новий діалог у папці Dialogs/.
   * Реалізуйте запит до БД у DataService.cs.
   * Додайте кнопку для виклику звіту на відповідній сторінці.
3. Змінити логіку авторизації:
   * Редагуйте AuthService.cs (наприклад, додати шифрування паролів).

4. Тестування

1. Юніт-тести:
   * Перевірка методів DataService (наприклад, AddUser, GetResourcesByType).
   * Тестування валідації даних у діалогових вікнах.
2. UI-тести:
   * Переконайтеся, що всі кнопки та фільтри працюють коректно.
   * Протестуйте генерацію PDF-звітів.

**Інструкція для користувача**

1. Запуск програми

1. Відкрийте файл HotelServices.exe.
2. Увійдіть за допомогою облікових даних:

2. Основні функції

1. Головне меню (Dashboard) – перехід до різних розділів:
   * Апартаменти, конференц-зали, парковки тощо.
2. Управління ресурсами:
   * Додати: Натисніть кнопку "Додати", заповніть форму та збережіть.
   * Редагувати: Виділіть ресурс у таблиці, натисніть "Редагувати".
   * Видалити: Виділіть ресурс, натисніть "Видалити" та підтвердіть.
   * Фільтрація: Використовуйте поле пошуку та випадаючий список статусів.
3. Бронювання:
   * Виберіть дати в календарі у вікні редагування.
   * Для апартаментів/залів вкажіть тривалість бронювання.
4. Звіти:
   * Виберіть період та натисніть "Сформувати звіт".
   * Для експорту в PDF натисніть "Зберегти як PDF".
5. Управління користувачами (тільки для адміністратора):
   * Додавання, редагування та видалення облікових записів.

3. Поширені питання користувачів

❓ Як змінити пароль?  
→ Адміністратор може змінити пароль у розділі "Управління користувачами".

❓ Чому не зберігаються зміни?  
→ Перевірте, чи всі обов’язкові поля заповнені (назва, ціна, дати).

❓ Як надрукувати звіт?  
→ Згенеруйте PDF та відправте на принтер із стандартного переглядача.

Даний додаток є зручним інструментом для управління готельними ресурсами, з підтримкою:  
✅ Бронювання (апартаменти, зали, парковки).  
✅ Фільтрації та пошуку.  
✅ Генерації звітів у PDF.  
✅ Ролей користувачів (адмін/менеджер).

Розробники можуть зручно розширювати функціонал, а користувачі – інтуїтивно керувати системою.

**3.3 Результати тестування програмного додатку**

Тестування програмного додатку проводилося на всіх етапах розробки для забезпечення стабільності та відповідності вимогам. Використовувалися різні методи перевірки, включаючи модульне, інтеграційне та системне тестування.

**Функціональне тестування** підтвердило коректність роботи основних модулів. Система автентифікації успішно ідентифікує користувачів та надає доступ відповідно до їхніх ролей.

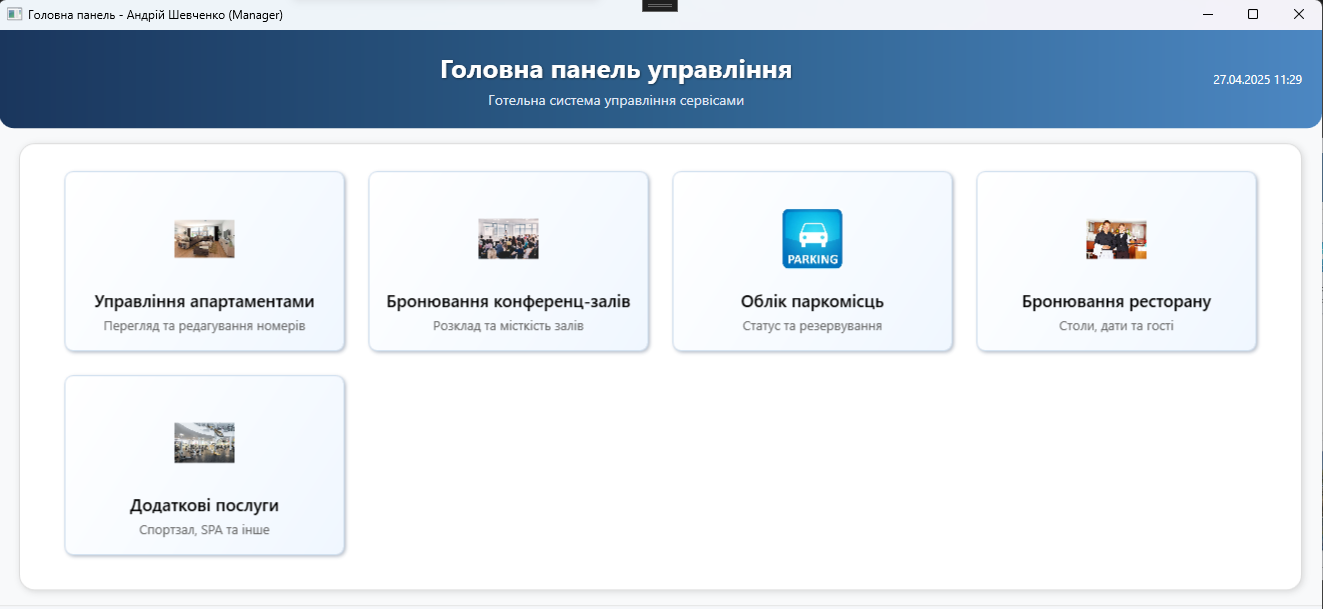
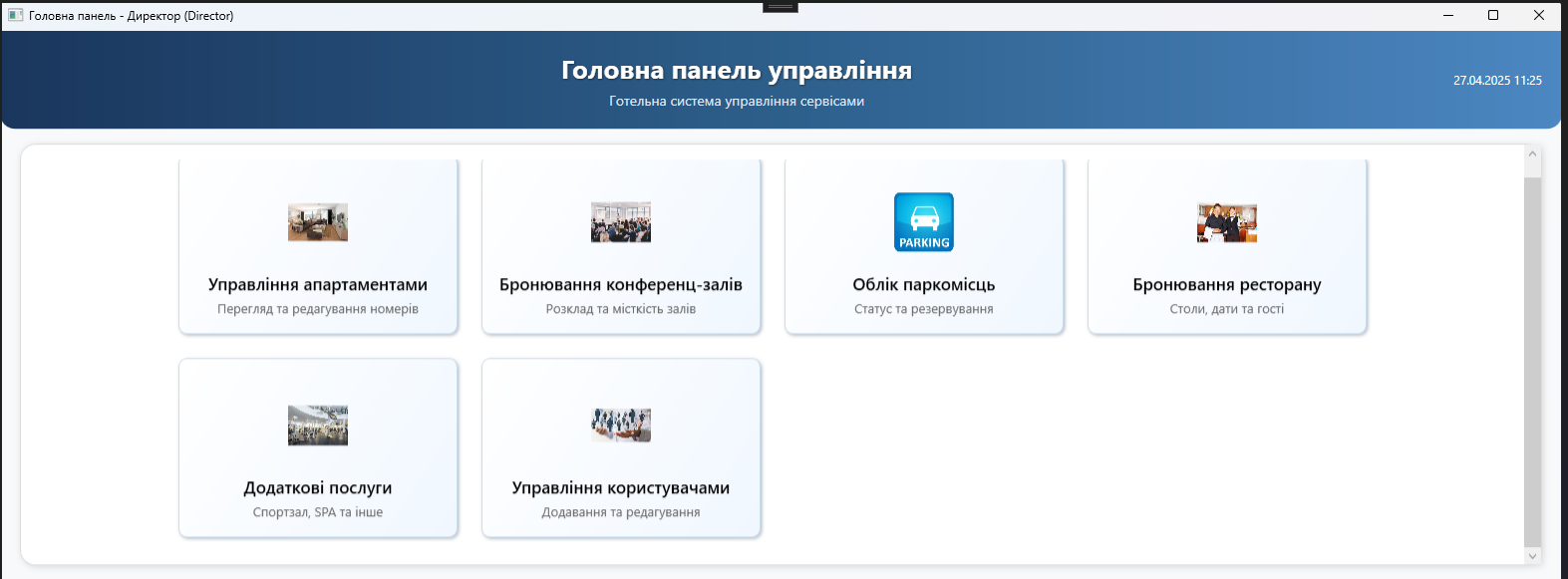


Рисунок 3.2 – Головне вікно програми залежно від ролі

*Джерело: побудовано автором*

Модуль управління ресурсами стабільно виконує операції додавання, редагування та видалення даних, зберігаючи цілісність інформації в базі SQLite. Генерація звітів у PDF форматі працює без збоїв, навіть при великих обсягах даних.

**Тестування інтерфейсу** показало, що всі елементи UI реагують на дії користувача. Валідація полів у формах редагування запобігає введенню некоректних даних, а динамічне оновлення інтерфейсу забезпечує актуальність інформації.

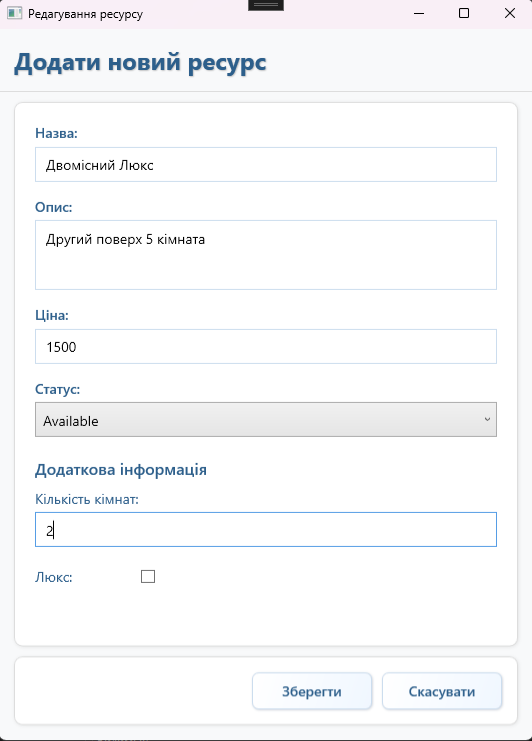
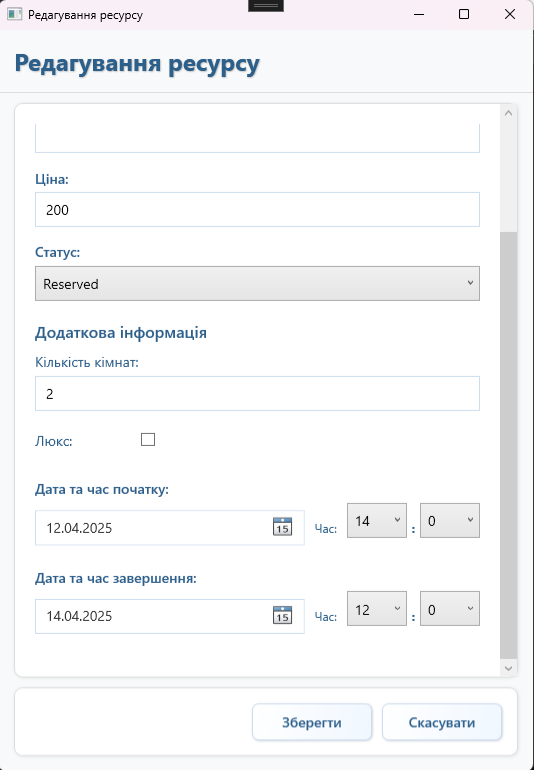
 

Рисунок 3.3 –Вікна додавання та редагування ресурсу

*Джерело: побудовано автором*

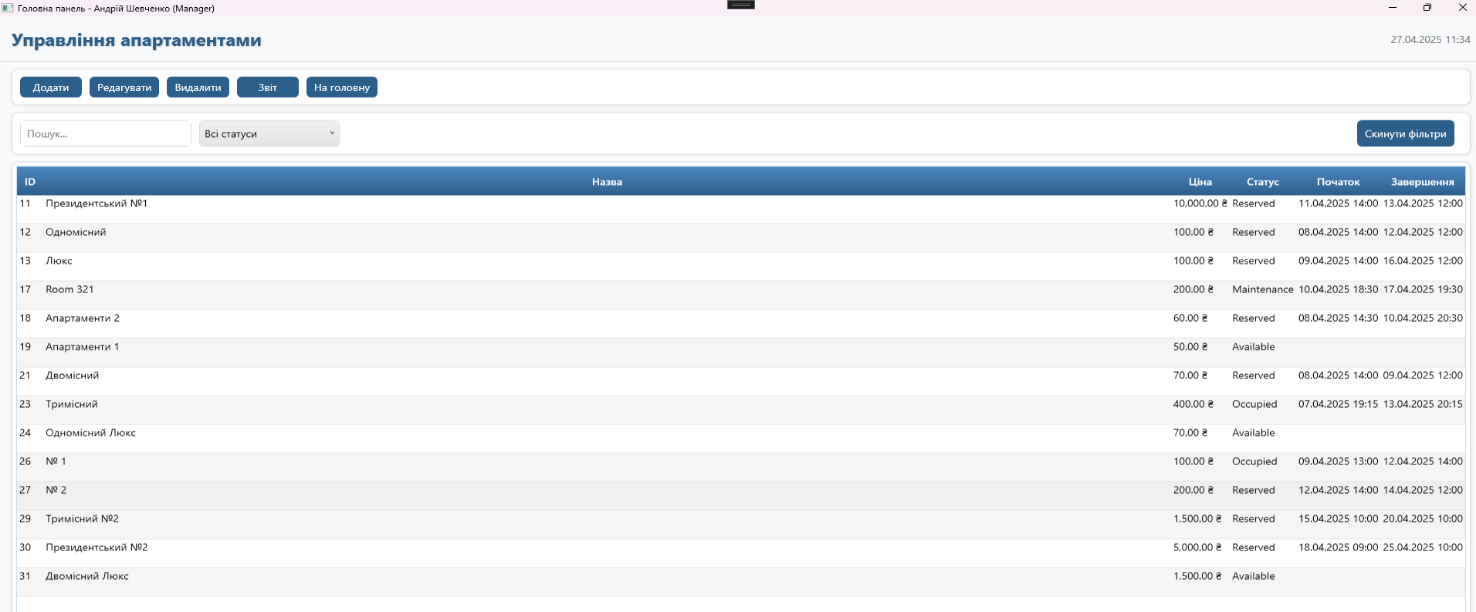
Відображення даних у таблицях та діалогових вікнах відбувається без затримок.

Рисунок 3.4 – Відображення даних у таблицях

*Джерело: побудовано автором*

**Перевірка продуктивності** виявила, що додаток ефективно працює на різних конфігураціях ПК. Оптимізація запитів до бази даних значно зменшила навантаження на систему.

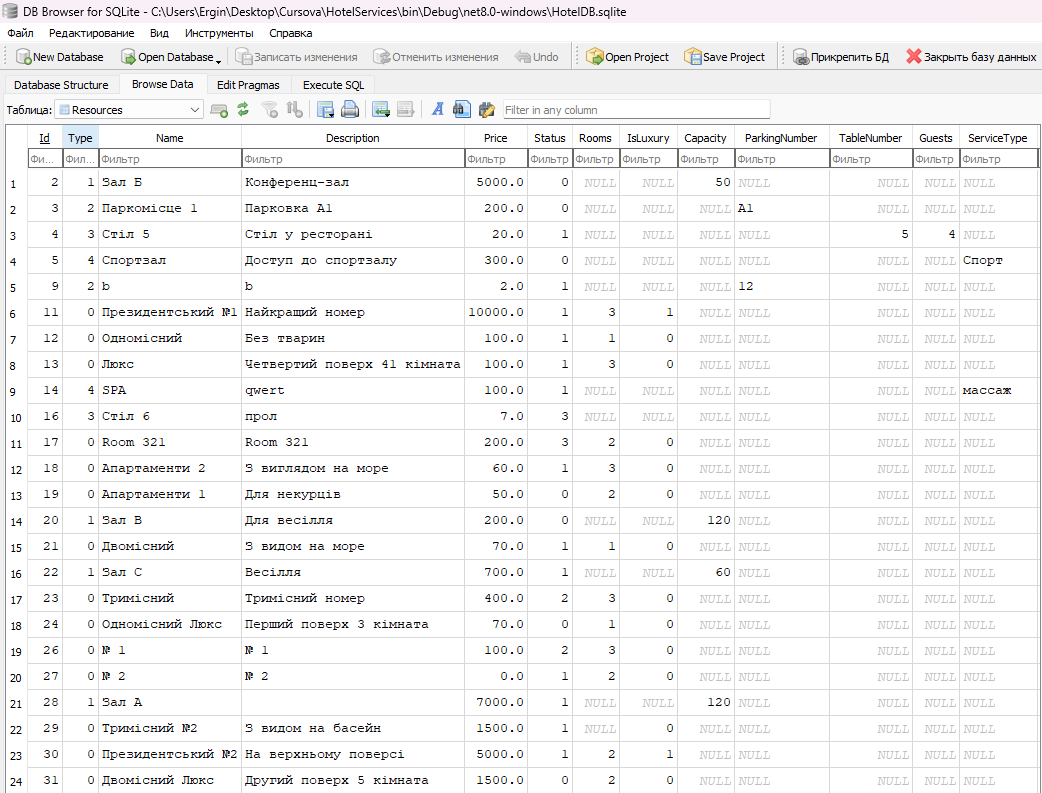


Рисунок 3.5 – Уніфікована таблиця Resources в базі даних

*Джерело: побудовано автором*

Система коректно обробляє спроби введення невірних даних або несанкціонованого доступу.

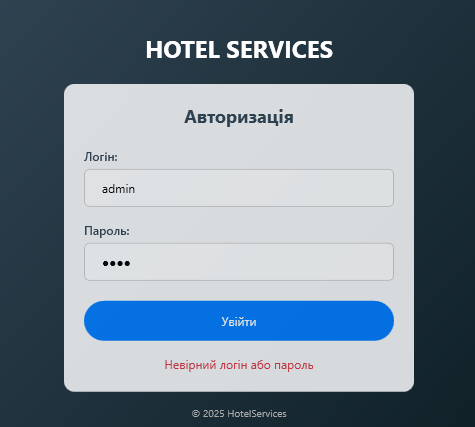


Рисунок 3.6 – Валідація даних в програмі

*Джерело: побудовано автором*

**Юзабіліті-тестування** показало, що інтерфейс програми інтуїтивно зрозумілий для користувачів. Всі функції легко знаходяться, а підказки спрощують роботу з системою.

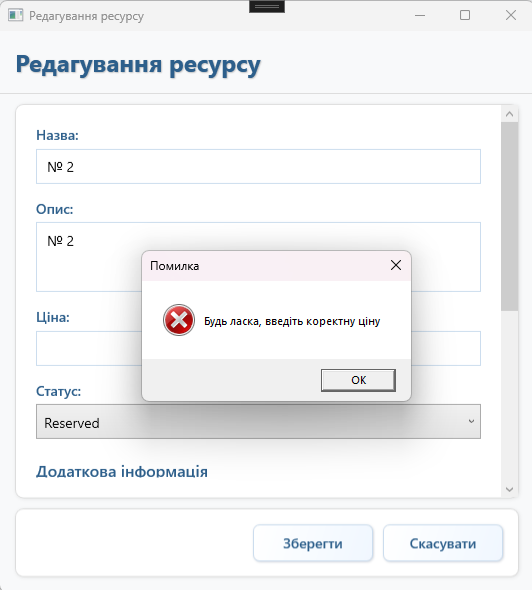


Рисунок 3.7 – Підказки в роботі програми

*Джерело: побудовано автором*

Тестування функцій пошуку та фільтрації також показало стабільність системи та правильність виконання команд.

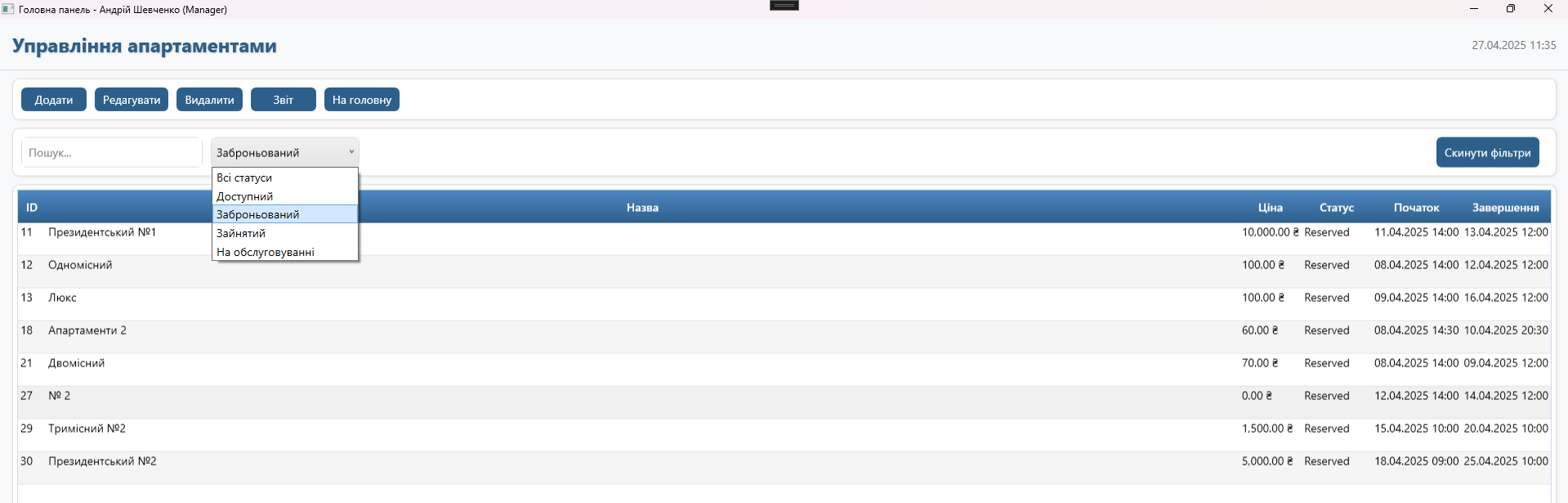


Рисунок 3.8 – Фільтрація в роботі програми

*Джерело: побудовано автором*

Під час тестування особливу увагу було приділено функціоналу генерації звітів, оскільки він є важливим для менеджменту житлового комплексу. Звіти успішно формуються в PDF-форматі з урахуванням української локалізації, форматування заголовків та збереженням таблиць у зручному для друку вигляді.

Експорт відбувається без помилок, незалежно від обсягу даних, що свідчить про стабільність механізму та готовність до використання в реальних умовах.

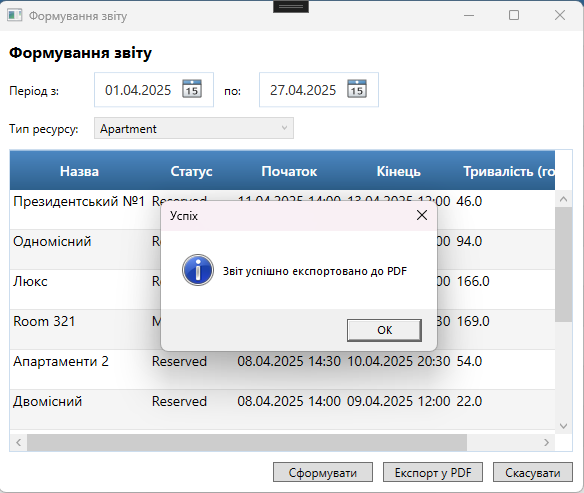
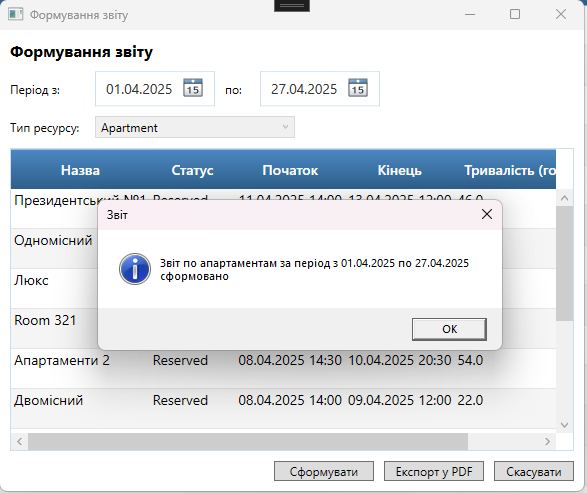


Рисунок 3.9 – Формування звіту та експорт його в pdf

*Джерело: побудовано автором*

Формування звітів здійснюється з використанням бібліотеки iTextSharp, яка дозволяє включати як загальну статистику (доходи, кількість бронювань), так і деталізацію по кожному ресурсу. Під час тестування жодного разу не було зафіксовано збоїв чи втрати даних при генерації, що підтверджує надійність реалізованого модуля.

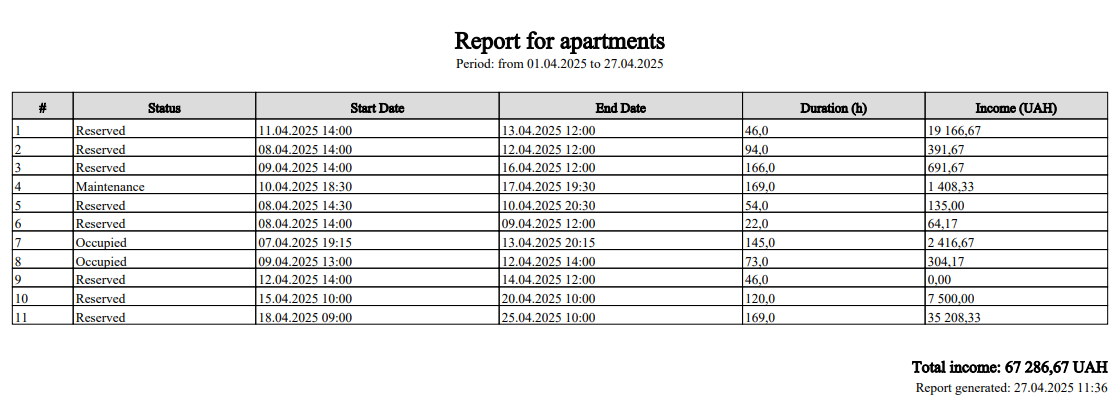


Рисунок 3.10 – Вміст експортованого pdf звіту

*Джерело: побудовано автором*

Під час тестування було виявлено та виправлено незначні помилки, пов’язані з обробкою крайніх випадків, таких як введення некоректних дат або спроба видалення використовуваного ресурсу. Фінальна версія додатка демонструє високу стабільність та готовність до експлуатації в реальних умовах.

**Висновки**

Система HotelServices реалізує комплексний підхід до управління готельними ресурсами, поєднуючи архітектурну чіткість, функціональну повноту, сучасні технології та зручний інтерфейс. В основі архітектури лежить чітке розділення рівнів: інтерфейс користувача, бізнес-логіка і робота з даними, що взаємодіють через сервіси.

Проведене дослідження та розробка програмного додатку засвідчили доцільність впровадження сучасних інформаційних рішень у сфері управління ресурсами багатофункціонального житлового комплексу. Результатом роботи стала створена система, що дозволяє централізовано керувати бронюванням та обліком апартаментів, паркомісць, конференц-залів, місць у ресторані та додаткових сервісів. Реалізований інструмент орієнтований на внутрішнє використання керівним персоналом комплексу — адміністраторами або менеджерами, що забезпечує зручність управління та високу швидкість прийняття рішень.

Архітектура програми побудована з урахуванням принципів модульності та розділення відповідальностей, що дозволяє легко масштабувати систему, адаптуючи її до змін у структурі комплексу або до появи нових типів ресурсів. Інтерфейс, розроблений на базі WPF із використанням MVVM-шаблону, забезпечує інтуїтивну взаємодію користувача з системою та дозволяє швидко отримувати доступ до потрібних даних. Застосування локальної бази SQLite дозволило мінімізувати вимоги до інфраструктури, а також забезпечити автономність роботи додатку.

Протестована система продемонструвала стабільність роботи, зручність у використанні та відповідність поставленим вимогам. Вона може бути ефективно використана в реальних умовах, значно підвищуючи рівень автоматизації та оптимізуючи процеси управління ресурсами в житловому комплексі. Таким чином, запропоноване рішення має потенціал для впровадження у подібних інфраструктурних об’єктах, забезпечуючи прозорість, контроль та аналітику на якісно новому рівні.

**Список використаної літератури**

* 1. Система BAS [Електронний ресурс] – URL: <https://www.softcom.ua/ua/bas/prorams/orenda-i-upravlinnya-nerukhomistyu/>

(дата звернення 07.04.2021)

* 1. Облік платежів по оренді BAS [Електронний ресурс] – URL: <https://its.bas-soft.eu/db/content/basrmre/src/_img/image145.png?_=0000F1D729469578-v2>

(дата звернення 07.04.2021)

* 1. Rental Property Management System [Електронний ресурс] – URL: <https://play.google.com/store/apps/details?id=com.apps.PropertyManagerRentTracker&hl=uk> (дата звернення 08.04.2021)
  2. Система «Камала Нерухомість» [Електронний ресурс] – URL: <https://kamala-soft.com/uk/solutions/kamala-nedvizhimost/> (дата звернення 08.04.2021)
  3. Управління ресурсами підприємства [Електронний ресурс] : монографія / за заг. ред. к.е.н., проф. Г. О. Швиданенко. — К. : КНЕУ, 2014. — 418, [6] с. ISBN 978-966-483-948-5 Онлайн-доступ: <https://ir.kneu.edu.ua/server/api/core/bitstreams/3dd0ff7a-9465-484c-bff2-ea89b4211555/content>
  4. Мороз, Наталія, і Тетяна Малетич. 2013. «ІННОВАЦІЙНІ СПОСОБИ УПРАВЛІННЯ БАГАТОКВАРТИРНИМИ ЖИТЛОВИМИ БУДИНКАМИ». *Mechanism of an Economic Regulation*, вип. 4(62) (Жовтень), 164-73.  Онлайн-доступ: <http://mer-journal.sumy.ua/index.php/journal/article/view/556/511>
  5. Модель управління ресурсами в житлових комплексах [Електронний ресурс] – URL: <https://studmir.com/wp-content/uploads/2020/08/sistema-upravlinnya-materialnimi-resursami-1-1536x1098.png> (дата звернення 09.04.2021)
  6. Visual Studio [Електронний ресурс] – URL: <https://visualstudio.microsoft.com> (дата звернення 10.04.2021)
  7. Об’єктно-орієнтоване програмування. Частина 1. Основи об’єктноорієнтованого програмування на мові C#.: Навчальний посібник. / Д.В. Настенко, А. Б. Нестерко. – К.: НТУУ «КПІ», 2016. - 76с. стр. 7
  8. SQLite [Електронний ресурс] – URL: <https://sqlite.org/> (дата звернення 10.04.2021)
  9. DB Browser for SQLite [Електронний ресурс] – URL: <https://sqlitebrowser.org/> (дата звернення 10.04.2021)

**Додатки**

**ДОДАТОК А**

**ResourceEditDialog.xaml.cs**

using HotelServices.Models;

using System;

using System.Collections.Generic;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Media.Animation;

namespace HotelServices.Dialogs

{

public partial class ResourceEditDialog : Window

{

public Resource Resource { get; private set; }

private readonly ResourceType \_resourceType;

public ResourceEditDialog(ResourceType resourceType, Resource resource)

{

InitializeComponent();

\_resourceType = resourceType;

// Ініціалізація вибору часу

InitializeTimeComboBoxes();

if (resource == null)

{

Resource = new Resource

{

Type = resourceType,

Status = ReservationStatus.Available,

StartDate = null,

EndDate = null

};

lblTitle.Text = "Додати новий ресурс";

}

else

{

Resource = resource;

lblTitle.Text = "Редагування ресурсу";

txtName.Text = resource.Name;

txtDescription.Text = resource.Description;

txtPrice.Text = resource.Price.ToString();

if (resource.StartDate.HasValue)

{

dpStartDate.SelectedDate = resource.StartDate.Value.Date;

cmbStartHour.SelectedItem = resource.StartDate.Value.Hour;

cmbStartMinute.SelectedItem = resource.StartDate.Value.Minute;

}

if (resource.EndDate.HasValue)

{

dpEndDate.SelectedDate = resource.EndDate.Value.Date;

cmbEndHour.SelectedItem = resource.EndDate.Value.Hour;

cmbEndMinute.SelectedItem = resource.EndDate.Value.Minute;

}

}

cmbStatus.ItemsSource = Enum.GetValues(typeof(ReservationStatus));

cmbStatus.SelectedItem = resource?.Status ?? ReservationStatus.Available;

cmbStatus.SelectionChanged += CmbStatus\_SelectionChanged;

UpdateDatePickersVisibility();

LoadAdditionalFields();

// Анімація появи вікна

AnimateDialogLoad();

}

private void AnimateDialogLoad()

{

// Анімація заголовка

var titleAnimation = new DoubleAnimation

{

From = 0,

To = 1,

Duration = TimeSpan.FromMilliseconds(300)

};

lblTitle.BeginAnimation(UIElement.OpacityProperty, titleAnimation);

// Анімація появи основного вмісту

var contentAnimation = new DoubleAnimation

{

From = 0.7,

To = 1,

Duration = TimeSpan.FromMilliseconds(400)

};

this.BeginAnimation(UIElement.OpacityProperty, contentAnimation);

}

private void InitializeTimeComboBoxes()

{

// Заповнення годин (0-23)

for (int i = 0; i < 24; i++)

{

cmbStartHour.Items.Add(i);

cmbEndHour.Items.Add(i);

}

// Заповнення хвилин (00, 15, 30, 45)

cmbStartMinute.Items.Add(0);

cmbStartMinute.Items.Add(15);

cmbStartMinute.Items.Add(30);

cmbStartMinute.Items.Add(45);

cmbEndMinute.Items.Add(0);

cmbEndMinute.Items.Add(15);

cmbEndMinute.Items.Add(30);

cmbEndMinute.Items.Add(45);

// Встановлення поточного часу за замовчуванням

var now = DateTime.Now;

cmbStartHour.SelectedItem = now.Hour;

cmbStartMinute.SelectedItem = (now.Minute / 15) \* 15; // Округлюємо до 15 хв

cmbEndHour.SelectedItem = now.AddHours(1).Hour;

cmbEndMinute.SelectedItem = (now.AddHours(1).Minute / 15) \* 15;

}

private DateTime? GetDateTimeFromPickers(DatePicker datePicker, ComboBox hourCombo, ComboBox minuteCombo)

{

if (datePicker.SelectedDate == null || hourCombo.SelectedItem == null || minuteCombo.SelectedItem == null)

return null;

return new DateTime(

datePicker.SelectedDate.Value.Year,

datePicker.SelectedDate.Value.Month,

datePicker.SelectedDate.Value.Day,

(int)hourCombo.SelectedItem,

(int)minuteCombo.SelectedItem,

0);

}

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

{

UpdateDatePickersVisibility();

}

private void UpdateDatePickersVisibility()

{

var currentStatus = (ReservationStatus)cmbStatus.SelectedItem;

bool datesRequired = currentStatus != ReservationStatus.Available;

Visibility visibility = datesRequired ? Visibility.Visible : Visibility.Collapsed;

// Використовуємо StackPanel для групового керування видимістю

startDatePanel.Visibility = visibility;

endDatePanel.Visibility = visibility;

}

private void LoadAdditionalFields()

{

additionalFieldsPanel.Children.Clear();

// Заголовок для розділу додаткових полів

TextBlock header = new TextBlock

{

Text = "Додаткова інформація",

FontSize = 16,

FontWeight = FontWeights.SemiBold,

Foreground = new System.Windows.Media.SolidColorBrush((System.Windows.Media.Color)System.Windows.Media.ColorConverter.ConvertFromString("#2D5F8B")),

Margin = new Thickness(0, 0, 0, 10)

};

additionalFieldsPanel.Children.Add(header);

switch (\_resourceType)

{

case ResourceType.Apartment:

AddField("Кількість кімнат:", "txtRooms", Resource?.Rooms?.ToString() ?? "");

AddCheckBox("Люкс:", "chkIsLuxury", Resource?.IsLuxury ?? false);

break;

case ResourceType.ConferenceRoom:

AddField("Місткість:", "txtCapacity", Resource?.Capacity?.ToString() ?? "");

break;

case ResourceType.ParkingSpace:

AddField("Номер паркомісця:", "txtParkingNumber", Resource?.ParkingNumber ?? "");

break;

case ResourceType.RestaurantTable:

AddField("Номер столу:", "txtTableNumber", Resource?.TableNumber?.ToString() ?? "");

AddField("Кількість гостей:", "txtGuests", Resource?.Guests?.ToString() ?? "");

break;

case ResourceType.AdditionalService:

AddField("Тип послуги:", "txtServiceType", Resource?.ServiceType ?? "");

break;

}

}

private void AddField(string label, string fieldName, string value)

{

var labelBlock = new TextBlock

{

Text = label,

Foreground = new System.Windows.Media.SolidColorBrush((System.Windows.Media.Color)System.Windows.Media.ColorConverter.ConvertFromString("#2D5F8B")),

FontSize = 14,

Margin = new Thickness(0, 0, 0, 5)

};

var textBox = new TextBox

{

Name = fieldName,

Text = value,

Height = 35,

BorderBrush = new System.Windows.Media.SolidColorBrush((System.Windows.Media.Color)System.Windows.Media.ColorConverter.ConvertFromString("#CCDDEE")),

BorderThickness = new Thickness(1),

Padding = new Thickness(8, 5, 8, 5),

VerticalContentAlignment = VerticalAlignment.Center,

FontSize = 14,

Background = System.Windows.Media.Brushes.White,

Margin = new Thickness(0, 0, 0, 15)

};

additionalFieldsPanel.Children.Add(labelBlock);

additionalFieldsPanel.Children.Add(textBox);

}

private void AddCheckBox(string label, string fieldName, bool isChecked)

{

var stackPanel = new StackPanel

{

Orientation = Orientation.Horizontal,

Margin = new Thickness(0, 5, 0, 15)

};

var labelBlock = new TextBlock

{

Text = label,

Width = 100,

VerticalAlignment = VerticalAlignment.Center,

Foreground = new System.Windows.Media.SolidColorBrush((System.Windows.Media.Color)System.Windows.Media.ColorConverter.ConvertFromString("#2D5F8B")),

FontSize = 14

};

var checkBox = new CheckBox

{

Name = fieldName,

IsChecked = isChecked,

VerticalAlignment = VerticalAlignment.Center,

Margin = new Thickness(5, 0, 0, 0)

};

stackPanel.Children.Add(labelBlock);

stackPanel.Children.Add(checkBox);

additionalFieldsPanel.Children.Add(stackPanel);

}

private TextBox FindTextBox(string name)

{

foreach (var child in additionalFieldsPanel.Children)

{

if (child is TextBox textBox && textBox.Name == name)

{

return textBox;

}

}

return null;

}

private CheckBox FindCheckBox(string name)

{

foreach (var child in additionalFieldsPanel.Children)

{

if (child is StackPanel panel)

{

foreach (var panelChild in panel.Children)

{

if (panelChild is CheckBox checkBox && checkBox.Name == name)

{

return checkBox;

}

}

}

}

return null;

}

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

{

if (string.IsNullOrWhiteSpace(txtName.Text))

{

ShowErrorMessage("Будь ласка, введіть назву ресурсу");

return;

}

if (!decimal.TryParse(txtPrice.Text, out decimal price))

{

ShowErrorMessage("Будь ласка, введіть коректну ціну");

return;

}

var currentStatus = (ReservationStatus)cmbStatus.SelectedItem;

DateTime? startDate = null;

DateTime? endDate = null;

if (currentStatus != ReservationStatus.Available)

{

startDate = GetDateTimeFromPickers(dpStartDate, cmbStartHour, cmbStartMinute);

endDate = GetDateTimeFromPickers(dpEndDate, cmbEndHour, cmbEndMinute);

if (startDate == null || endDate == null)

{

ShowErrorMessage("Будь ласка, вкажіть дату та час початку і завершення");

return;

}

if (startDate >= endDate)

{

ShowErrorMessage("Час початку повинен бути раніше часу завершення");

return;

}

}

Resource.Name = txtName.Text;

Resource.Description = txtDescription.Text;

Resource.Price = price;

Resource.Status = currentStatus;

Resource.StartDate = startDate;

Resource.EndDate = endDate;

switch (\_resourceType)

{

case ResourceType.Apartment:

if (int.TryParse(FindTextBox("txtRooms")?.Text, out int rooms))

Resource.Rooms = rooms;

Resource.IsLuxury = FindCheckBox("chkIsLuxury")?.IsChecked ?? false;

break;

case ResourceType.ConferenceRoom:

if (int.TryParse(FindTextBox("txtCapacity")?.Text, out int capacity))

Resource.Capacity = capacity;

break;

case ResourceType.ParkingSpace:

Resource.ParkingNumber = FindTextBox("txtParkingNumber")?.Text;

break;

case ResourceType.RestaurantTable:

if (int.TryParse(FindTextBox("txtTableNumber")?.Text, out int tableNumber))

Resource.TableNumber = tableNumber;

if (int.TryParse(FindTextBox("txtGuests")?.Text, out int guests))

Resource.Guests = guests;

break;

case ResourceType.AdditionalService:

Resource.ServiceType = FindTextBox("txtServiceType")?.Text;

break;

}

// Анімація при збереженні

AnimateButtonClick(sender as Button);

DialogResult = true;

Close();

}

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

{

// Анімація при скасуванні

AnimateButtonClick(sender as Button);

DialogResult = false;

Close();

}

private void AnimateButtonClick(Button button)

{

if (button == null) return;

// Створюємо стислу анімацію для кнопки при натисканні

var scaleDownAnimation = new DoubleAnimation

{

From = 1.0,

To = 0.95,

Duration = TimeSpan.FromMilliseconds(100)

};

var scaleUpAnimation = new DoubleAnimation

{

From = 0.95,

To = 1.0,

Duration = TimeSpan.FromMilliseconds(100),

BeginTime = TimeSpan.FromMilliseconds(100)

};

var scaleTransform = new System.Windows.Media.ScaleTransform(1, 1);

button.RenderTransform = scaleTransform;

scaleTransform.BeginAnimation(System.Windows.Media.ScaleTransform.ScaleXProperty, scaleDownAnimation);

scaleTransform.BeginAnimation(System.Windows.Media.ScaleTransform.ScaleYProperty, scaleDownAnimation);

scaleTransform.BeginAnimation(System.Windows.Media.ScaleTransform.ScaleXProperty, scaleUpAnimation);

scaleTransform.BeginAnimation(System.Windows.Media.ScaleTransform.ScaleYProperty, scaleUpAnimation);

}

private void ShowErrorMessage(string message)

{

MessageBox.Show(message, "Помилка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

}