НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
Факультет інформаційних технологій

ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ
Завідувач кафедри
Комп'ютерних наук
_____ Голуб Б.Л.
(підпис)
"_25_" _травня_ 2025 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА
на тему:

Програмне забезпечення автоматизованої системи продажі квитків у кінотеатрі

Спеціальність 121 – «Інженерія програмного забезпечення»
ОПП «Інженерія програмного забезпечення»

Гарант освітньої програми
К.т.н., доцент _____ / Вайганг Г.О./
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Керівник бакалаврської кваліфікаційної роботи
К.т.н., доцент _____ / Голуб Б.Л./
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Виконав: _____ / Беттяр Н.Д./
(підпис) (ПІБ студента)

КИЇВ – 2025

НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
Факультет інформаційних технологій

ЗАТВЕРДЖУЮ
Завідувач кафедри
Комп'ютерних наук
_____ Голуб Б.Л.
(підпис)
"_16_" __грудня___2024 р.

З А В Д А Н Н Я
на виконання бакалаврської кваліфікаційної роботи студенту
Беттяру Нікіті Денисовичу
(прізвище, ім'я, по батькові)

Спеціальність 121 – «Інженерія програмного забезпечення»

ОПП «Інженерія програмного забезпечення»

Тема бакалаврської кваліфікаційної роботи: Програмне забезпечення автоматизованої системи продажі квитків у кінотеатрі затверджена наказом ректора НУБіП України від "_16_" __грудня__ 2024 р. № 2249 " С "

Термін подання завершеної роботи на кафедру: 25 травня 2025 р.
(число, місяць, рік)

Вихідні дані до бакалаврської кваліфікаційної роботи взято з джерел:

- https://promo.ticketcrm.com/uk
- https://mticket.com.ua

Перелік питань, які потрібно розробити:

1. Процес продажу квитків у кінотеатрі як об'єкт дослідження
2. Інформаційне забезпечення
3. Проєктування і розробка програмного забезпечення
4. Рекомендації щодо впровадження та експлуатації системи

Дата видачі завдання "_16_" __грудня__ 2024 р.
Керівник бакалаврської кваліфікаційної роботи
К.т.н., доцент _____ _____ / Голуб Б.Л./
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Завдання прийняв до виконання _____ / Беттяр Н.Д. /
(підпис) (ПІБ студента)

ЗМІСТ

ВСТУП

Актуальність. У сучасних умовах цифровізації важливим фактором успішного розвитку підприємств сфери обслуговування є впровадження автоматизованих інформаційних систем, що дозволяють оптимізувати бізнес-процеси, підвищити якість обслуговування клієнтів та зменшити витрати. Кінотеатри, як частина індустрії розваг, не є винятком - ефективна організація процесу продажу квитків відіграє ключову роль у забезпечення зручності для глядачів, зменшенні черг і підвищенні загального рівня задоволеності відвідувачів.

Традиційні методи продажу квитків у касах, мають деякі обмеження: обмежений час роботи, людський фактор, складність ведення статистики. У зв’язку з цим виникає необхідність у створенні програмного забезпечення, яке дозволяє автоматизувати процес продажу квитків, забезпечити швидкий та зручний доступ до інформації про фільми та сеанси, а також надає можливість користувачам самостійно обирати місця та здійснювати оплату.

Мета розробки. Основною метою проєкту є створення програмного забезпечення автоматизованої системи продажу квитків у кінотеатрі, яке дозволяє ефективно організовувати процес взаємодії з клієнтами, автоматизувати продаж квитків, полегшити роботу персоналу та забезпечити прозорість усіх операцій. Дане програмне забезпечення має забезпечити зручність, надійність, масштабованість та адаптивність до потреб як працівників кінотеатру, так і для його відвідувачів.

Методи та технології. Під час розробки автоматизованої системи продажу квитків використовувався сучасний стек технологій, до якого входять: C# з Windows Forms для створення зручного користувацького інтерфейсу, Microsoft SQL Server для управління базою даних, а також архітектурні принципи модульності та розмежування рівнів доступу. Особливу увагу приділено реалізації авторизації з поділом ролей, інтеграції з базою даних для

збереження інформації про фільми, сеанси, квитки та користувачів, а також забезпеченню надійності та безпеки обробки транзакцій.

Апробація програмного продукту. Результати розробки продукту були представлені на VII Всеукраїнській науково-практичній конференції студентів і аспірантів «Теоретичні та прикладні аспекти розробки комп'ютерних систем» 2025 у вигляді тез доповіді з назвою «Програмне забезпечення автоматизованої системи продажу квитків у кінотеатрі».

Структура записки. Бакалаврська кваліфікаційна робота складається з вступу, чотирьох розділів, висновків, списку використаних джерел та додатків. У вступі коротко описано актуальність теми, сформульовано мету та завдання проєкту, визначено об'єкт і предмет дослідження, а також наведено методи та технології, що використовувались під час розробки програмної системи.

У першому розділі під назвою «Процес продажу квитків у кінотеатрі як об'єкт дослідження» проведено детальний аналіз процесу продажу квитків у кінотеатрі, здійснено огляд існуючих програмних рішень, визначено основні функціональні та нефункціональні вимоги до інформаційної системи.

Другий розділ «Інформаційне забезпечення» присвячений аналізу та опису інформаційного забезпечення системи продажу квитків у кінотеатрі. У ньому визначено склад, структуру, джерела і способи зберігання вхідної, проміжної та вихідної інформації, що використовується в системі. Розглянуто інформаційні потоки між користувачами та підсистемами, а також побудовано інформаційну модель, яка включає структури таблиць бази даних, їх зв'язки та атрибути. Особливу увагу приділено формалізації інформаційних об'єктів і визначенню вимог до бази даних, яка забезпечує підтримку основних функцій системи.

У третьому розділі «Реалізація інформаційної системи» детально описано процес розробки Windows Forms застосунку, представлено його структуру, охарактеризовані основні форми, модулі, класи, методи та фрагменти коду, що реалізують ключові функціональні можливості системи, зокрема взаємодію з

базою даних, відображення інформації про фільми та реалізацію ролей користувачів.

Четвертий розділ «Рекомендації щодо впровадження та експлуатації системи» містить опис методів та інструментів тестування, що були використані для перевірки працездатності, надійності та якості програмного продукту. Проведено модульне тестування з використанням NUnit для перевірки функціональності окремих компонентів, а також інтеграційне тестування для оцінки взаємодії між основними модулями системи. Наведено результати тестування та ілюстрації інтерфейсу програми, що підтверджують коректну роботу системи.

У висновках підбито підсумки виконаної роботи, сформульовано основні результати, а також надано рекомендації щодо подальшого використання, масштабування і супроводу розробленої системи.

Кваліфікаційна робота викладена на 62 сторінках основного тексту, містить 39 рисунків, 8 додатків. До роботи включено список використаних джерел, який налічує 21 найменувань.

1 ПРОЦЕС ПРОДАЖУ КВИТКІВ У КІНОТЕАТРІ ЯК ОБ'ЄКТ ДОСЛІДЖЕННЯ

1.1 Опис предметної області

У сучасному світі індустрія розваг розвивається дуже швидко, і однією з найпопулярніших її складових є кіноіндустрія. Кінотеатри залишаються важливими культурними та соціальними просторами, що забезпечують дозвілля для широкого кола глядачів. У зв'язку з цим ефективна організація діяльності кінотеатрів набуває особливої актуальності. Швидке зростання кількості клієнтів, розширення репертуару фільмів і потреба в якісному обслуговуванні вимагають чіткої координації внутрішніх процесів.

Одним із ключових аспектів функціонування кінотеатру є забезпечення зручного та швидкого обслуговування клієнтів. Продаж квитків охоплює не лише сам факт передачі глядачеві права на перегляд фільму, а й низку супутніх процесів: взаємодію з клієнтами, управління розкладом сеансів, бронювання місць, контроль за заповненістю залів і ведення фінансової звітності. Усі ці компоненти мають бути злагодженими та керованими, щоб уникнути непорозумінь та помилок.

На практиці процес продажу квитків у кінотеатрі стикається з низкою проблем. Зокрема, виникають труднощі з точним контролем кількості вільних місць, що може призводити до дублювання бронювань або помилок у розсадженні глядачів. Особливо помітними ці недоліки стають під час пікового навантаження у вихідні дні або під час прем'єрних показів. У цей період також ускладнюється керування чергами та якісне обслуговування великого потоку клієнтів.

Працівники кінотеатру в такій ситуації зазнають додаткового навантаження. Їм доводиться вручну перевіряти наявність місць, вести облік продажів, формувати звіти для адміністрації, що збільшує ризик помилок і

затримок у роботі. Відсутність централізованого способу контролю за розкладом ускладнює внесення змін до графіка показів, що може призводити до плутанини як серед працівників, так і серед глядачів.

Окрему проблему становить доступ до актуальної інформації для різних категорій користувачів. Глядачі не завжди можуть оперативно отримати відомості про наявність квитків чи розклад сеансів. Персонал, своєю чергою, не завжди має змогу швидко дістати дані про стан продажів, залишок місць або підготувати звітність.

Сучасні відвідувачі очікують зручності та швидкості у процесі вибору й придбання квитків. Невідповідність цим очікуванням часто стає причиною невдоволення, втрати клієнтів і погіршення репутації закладу. Тому важливо вчасно виявляти та усувати проблеми, пов’язані з організацією роботи кінотеатру, зокрема в частині обслуговування глядачів, планування сеансів та ведення обліку.

1.2 Огляд існуючих програм-аналогів

Під час огляду аналогів системи продажу квитків в кінотеатрі, виявлено три основні системи: SERVIO POS InfoDisplay, TicketCRM та Mticket. Далі проведено детальний аналіз кожної з них, звертаючи увагу на їхні недоліки.

1.2.1 SERVIO POS InfoDisplay. Система SERVIO POS InfoDisplay використовується для клієнтських моніторів, найчастіше встановлюється на касі. Основне призначення - демонстрація клієнту інформації з екрана касира. У перервах цей монітор може у фоновому режимі виводити будь-яку рекламну інформацію. Розробник цієї системи - українська компанія Expert Solution[1]. На рис. 1.1 можна побачити вигляд цієї системи при вибору місця.

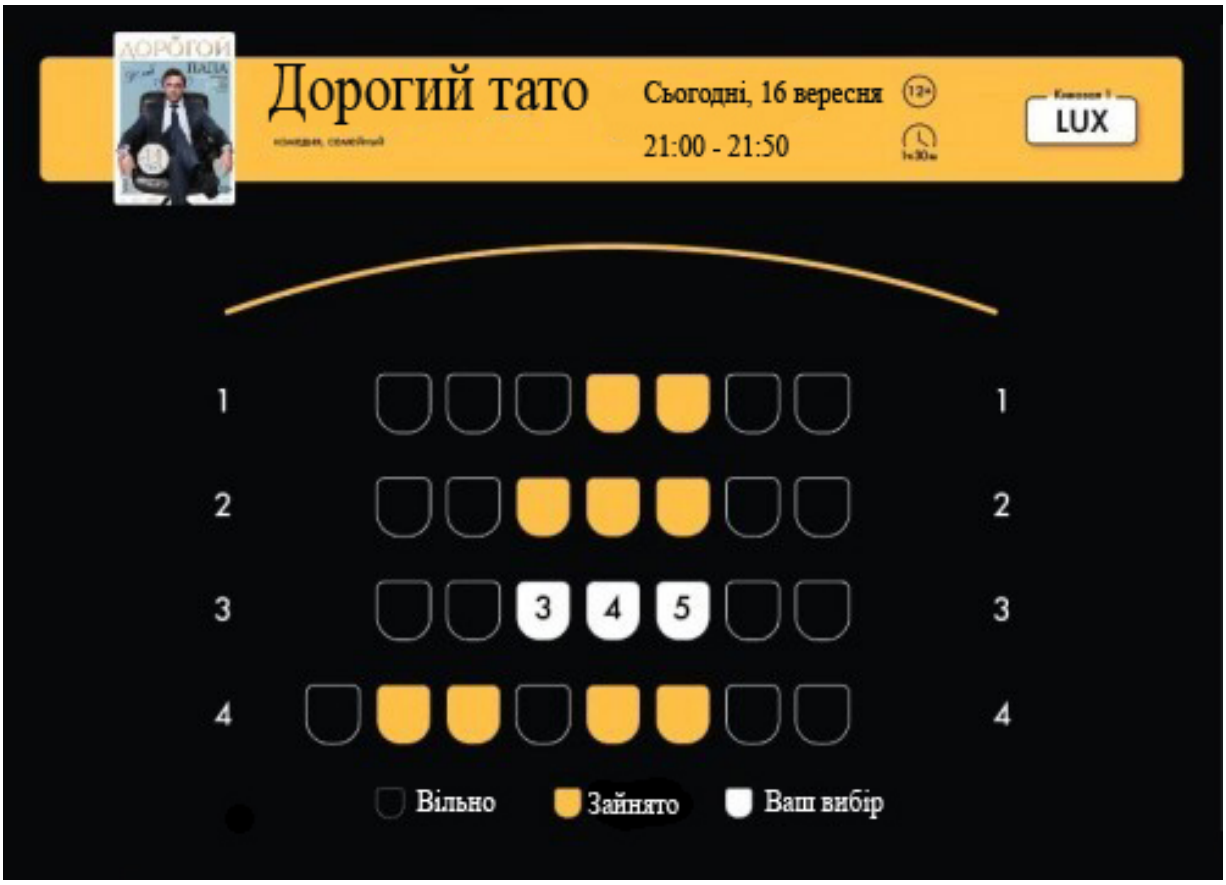


Рис. 1.1 Вигляд системи SERVIO POS InfoDisplay

Також на рис. 1.2 представлено вигляд фонового меню з фільмами у прокаті від SERVIO POS InfoDisplay.

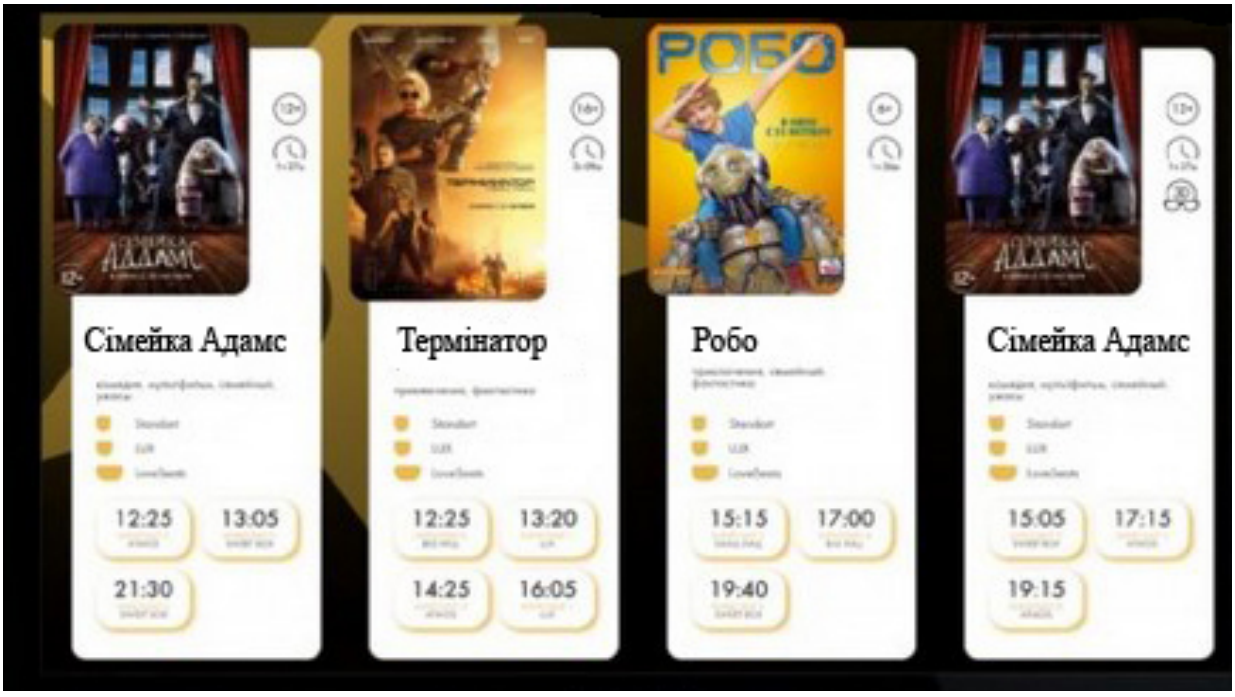


Рис. 1.2 Вигляд фонового меню з фільмами у прокаті від SERVIO POS InfoDisplay

Недоліки SERVIO POS InfoDisplay:

- так як SERVIO InfoDisplay це POS система, її не можливо викупити, потрібно буде постійно платити за підписку;
- систему потрібно буде вручну налаштувати під ваш кінотеатр, SERVIO POS InfoDisplay не надає такої можливості навіть за додаткову плату;
- система має складну інтеграцію з іншими системами через обмеженість API та несумісність з частиною стандартів;
- так як це POS система, при технічних проблемах в SERVIO POS InfoDisplay може зупинитись робота всього кінотеатру;
- система немає функціоналу створення клієнтської бази.

1.2.2 TicketCRM. Система TicketCRM має усі інструменти та модулі, щоб створювати, редагувати та продавати квитки на будь-які кіносеанси. Також дозволяє провести контроль входу навіть у багатозальному кінотеатрі. Принцип роботи real-time дозволяє продавати усю масу квитків одночасно як онлайн, так і в касі кінотеатру[2]. На рис. 1.3 показно приклад як виглядає система квитків для касиру.

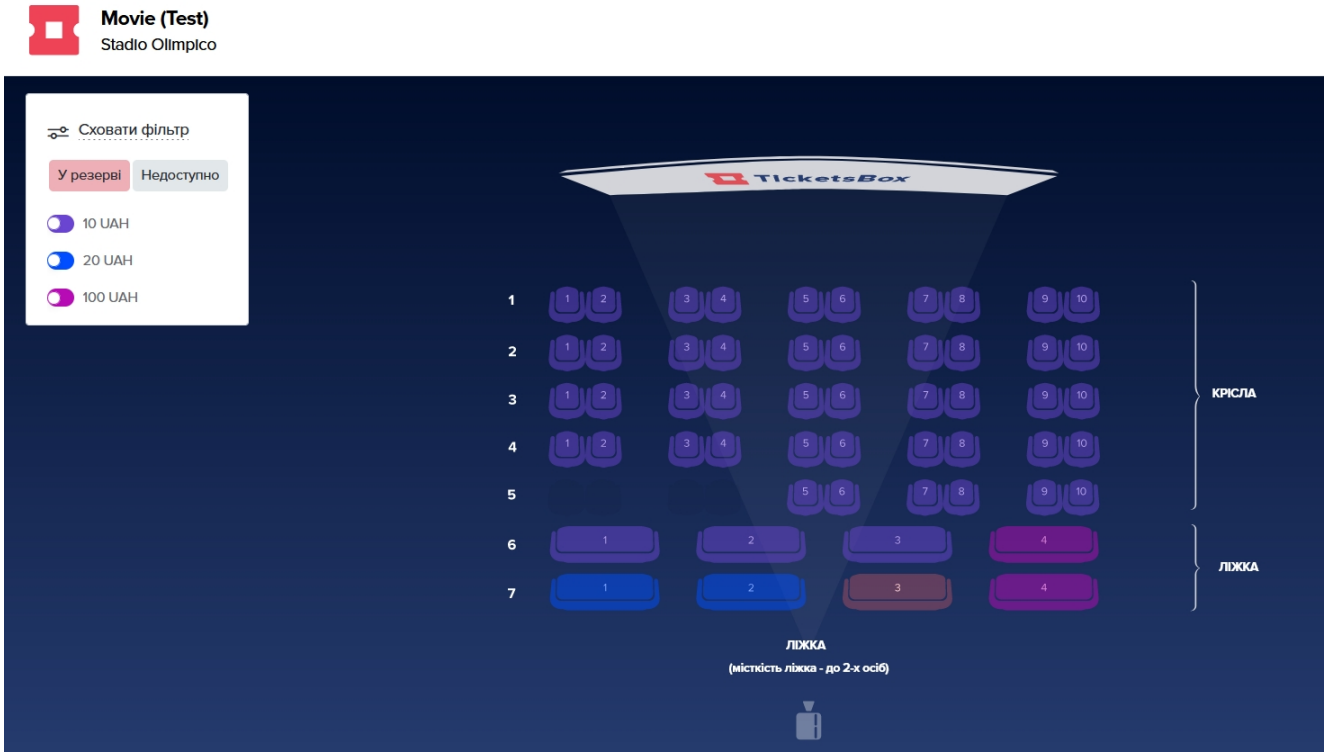


Рис. 1.3 Приклад системи від TicketCRM

Недоліки TicketCRM:

- відсутня можливість редагувати всі елементи системи, що унеможливлює адаптацію під конкретний кінотеатр без залучення технічної підтримки;
- використовується хмарна архітектура, що при нестабільному інтернеті унеможливить роботу каси чи терміналу;
- частина функцій, вимагаються додаткової оплати;
- не адаптована під великі навантаження;
- має складнощі з адаптуванням інтерфейсу;
- складний інтерфейс для користувачів без досвіду роботи з CRM-системами;
- відсутність аналітики продажів.

1.2.3 Maestro Ticket System. Maestro Ticket System - системний інтегратор з автоматизації продажу квитків і системами платіжно-пропускного доступу для спортивних і культурно-розважальних установ [3]. На рис. 1.4 показно головна сторінка при відкритті фільму.

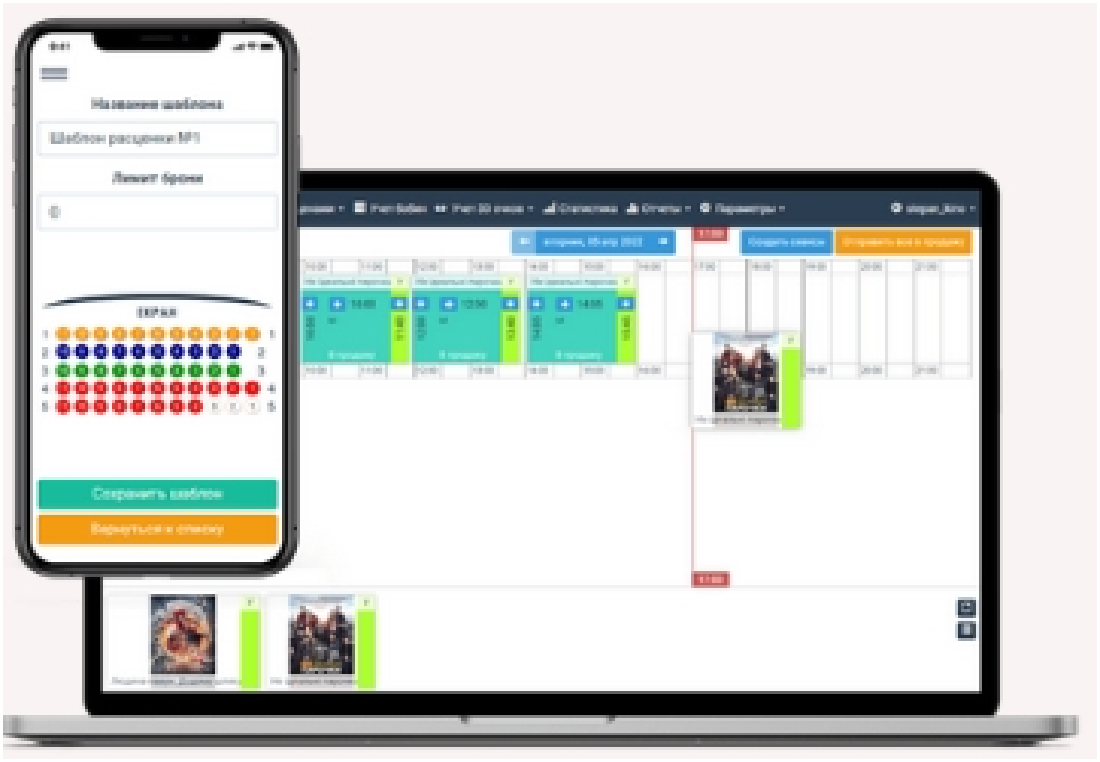


Рис. 1.4 Сторінка при відкритті фільму

Недоліки Mticket:

- комісійні збори за кожен продаж можуть суттєво знизити загальний прибуток;
- немає можливості адаптувати сторінки програми під свій кінотеатр або налаштувати схеми залу;
- частину налаштувань потрібно узгоджувати через службу підтримки;
- обмежений доступ до даних про користувачів, що ускладнює побудову власної клієнтської бази.

1.3 Постановка завдання

Для відвідувача кінотеатру важливим є зручність у використанні та надійність сервісу. Клієнти повинні мати змогу зручного перегляду репертуару фільмів, швидко знаходити потрібний їм фільм, мати можливість ознайомитись з розкладом сеансів, вибрати місце серед вільних та здійснювати оплату зручним для них методом. Наявність можливості отримувати електронний квиток також спрощує вхід до кінозали, зменшує витрату паперу та мінімізує випадки втрати квитка.

Програмне забезпечення для автоматизованого продажу квитків у кінотеатрі не тільки оптимізує внутрішні процеси підприємства, а й покращує досвід взаємодії з клієнтом. Це сприяє підвищенню ефективності роботи кінотеатру, зменшенню витрат, підвищенню рівня задоволеності клієнтів та збільшенню конкурентоспроможності закладу на ринку розважальних послуг.

Метою створення інформаційної системи для кінотеатру є розробка зручного, ефективного та доступного програмного забезпечення для ведення обліку репертуару фільмів, організації розкладу сеансів, управління процесом бронювання та продажу квитків, а також забезпечення швидкого доступу до актуальної інформації як для працівників кінотеатру, так і для його відвідувачів.

Система повинна підтримувати просту і зрозумілу взаємодію з користувачем, забезпечувати надійність у роботі з даними, можливість гнучкого редагування розкладу та формування статистичних звітів для прийняття управлінських рішень.

Для відвідувачів.

- **Афіша фільмів та розклад сеансів:** відвідувачі мають можливість переглядати репертуар фільмів, ознайомлюватись з розкладом сеансів, тривалістю фільмів, жанром, віковими обмеженнями та іншою інформацією про фільм.
- **Вибір місця та оформлення замовлення:** відвідувачі обирають зручне місце у залі за допомогою візуального плану розташування сидінь та оформлюють замовлення квитка на обраний сеанс.
- **Самостійна покупка квитків через термінал:** відвідувачі мають можливість купувати квитків через термінали самообслуговування, що дає змогу зменшити черги та підвищити комфорт для клієнтів.
- **Отримання електронного або паперового квитка:** відвідувачі після успішного оформлення замовлення вибирає отримати електронний квиток або друкований чек для входу до зали.

Для касирів.

- **Продаж квитків у касі:** касири мають можливість здійснювати продаж квитків у ручному режимі з вибором місць, сеансів та кількості квитків.
- **Проведення оплати через касовий термінал:** інтеграція з платіжними пристроями дозволяє швидко та зручно здійснювати оплату відвідувачами на касі.
- **Друк квитків:** касири використовують функцію друку паперових квитків або надсилання клієнтам електронних квитків.
- **Надання інформації клієнтам:** касир надає консультації клієнтам щодо розкладу, фільмів, наявності місць.

Для адміністраторів.

- **Управління репертуаром:** додавання, редагування або видалення інформації про фільми, включаючи опис, жанр, тривалість, постери.
- **Формування та редагування розкладу сеансів:** можливість створення графіку показів фільмів із прив'язкою до залів, часу показу та доступних місць.
- **Управління ціноутворенням та знижками:** встановлення вартості квитків.
- **Формування звітності та аналітики:** перегляд статистики продажів, рівня завантаженості залів, популярності фільмів.

Нефункціональні вимоги.

Масштабованість:

- система повинна підтримувати одночасний доступ великого потоку користувачів без зниження продуктивності.

Безпека:

- система повинна забезпечувати захист даних користувачів від несанкціонованого доступу, використовуючи шифрування та аутентифікацію.

Зручність:

- система повинна мати інтуїтивний та привабливий інтерфейс, який адаптується до різних розмірів екранів та пристроїв.

Сумісність:

- система повинна підтримувати операційні системи та протоколи зв'язку.

1.4 Моделювання предметної області

Для візуалізації та моделювання процесів та взаємодії в системі оцінки компетенцій шукачів роботи та пошуку роботи, можна використовувати різні типи діаграм з нотації UML (Unified Modeling Language) - уніфіконава мова моделювання.

UML-діаграми корисні для моделювання архітектури проєктів, оскільки дозволяють охопити як загальну структуру, так і окремі деталі. Вони допомагають створити так званий каркас майбутньої програми, який згодом слугуватиме основою для написання коду [4].

1.4.1 Діаграма прецедентів. Діаграма прецедентів відображає функціональні вимоги системи з точки зору користувача, показуючи типи ролей та їх взаємодію із системою. Вона демонструє, що система може зробити, але не фіксує порядок виконання дій. Така діаграма може бути представлена як у вигляді графічної схеми, так і у текстовій формі [5].

На діаграмі прецедентів, зображеній на рис. 1.5, представлено чотири основні дійові особи, двоє з яких виконують однакові дії.

Відвідувач може виконувати наступні дії:

- купувати квитки;
- перегляд фільмів, сеансів та місць;
- вибір фільмів, сеансів та місць.

Касир та термінал самообслуговування може виконувати наступні дії:

- продаж квитків;
- оплата;
- оформлення квитка.

Адміністратор може виконувати наступні дії:

- управління розкладом кіносеансів;
- налаштування цін на квитки.

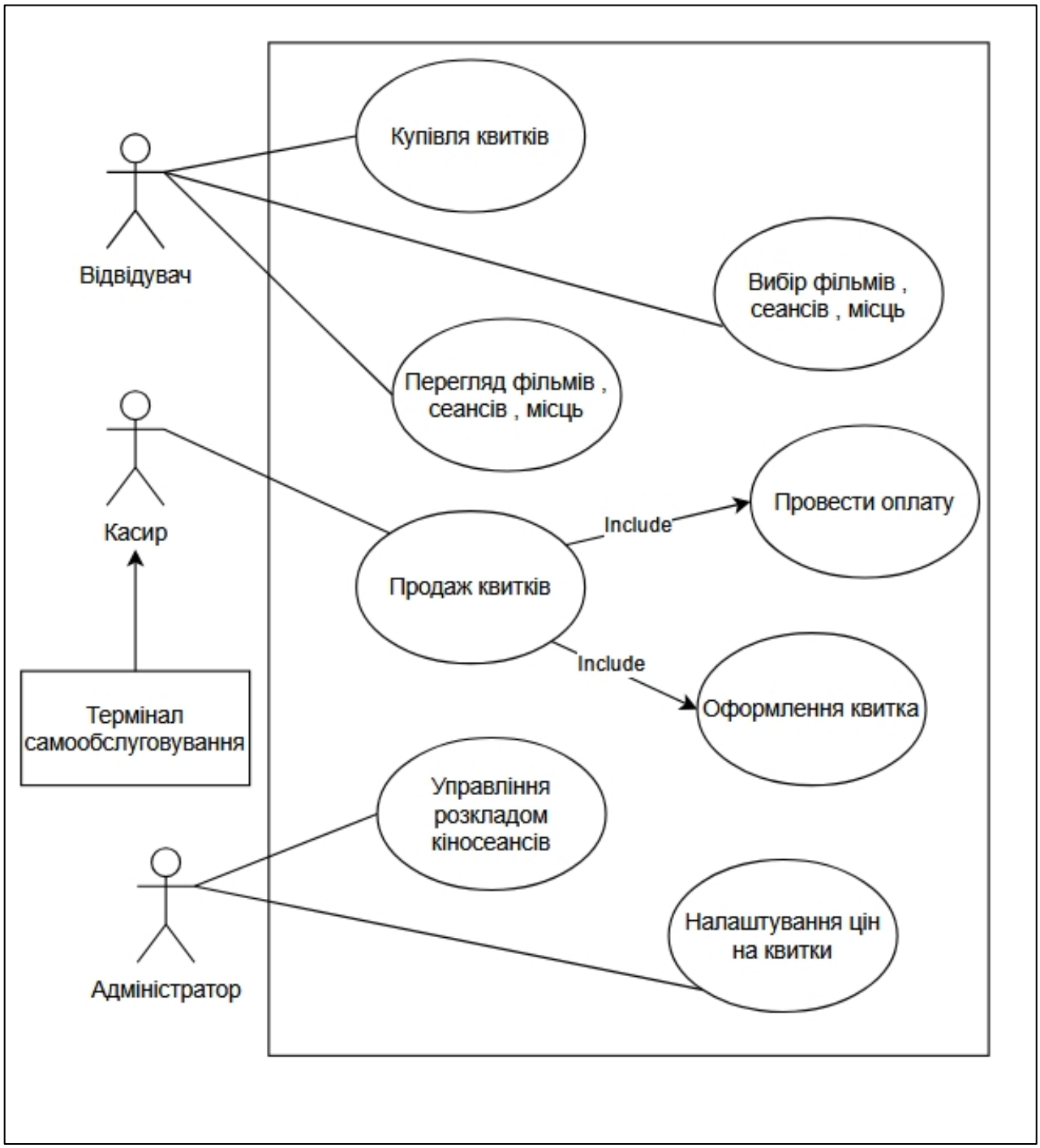


Рис. 1.5 Діаграма прецедентів

1.4.2 Діаграма послідовності. Діаграма послідовності відображає взаємодію між об’єктами або процесами системи з плином часу, показуючи порядок повідомлень, які передаються між ними під час виконання певного сценарію. Вона використовується для моделювання логіки сценаріїв використання, демонструючи, які класи задіяні у процесі та яку інформацію вони обмінюються відповідно до кроків у варіанті використання [6].

На рис. 1.6 показана діаграма послідовності сценарію роботи програми. На рисунку зображено взаємодію основних учасників автоматизованої системи продажу квитків у кінотеатрі: відвідувача, касира, адміністратора та кінотеатру (інформаційної системи).

Адміністратор додає нові фільми та сеанси до системи кінотеатру, після чого база даних оновлюється. Касир отримує актуальну інформацію про фільми та сеанси для обслуговування відвідувачів. Відвідувач, у свою чергу, запитує інформацію про доступні фільми, розклад сеансів та вільні місця.

Після перегляду варіантів відвідувач обирає потрібний фільм, сеанс і місце, отримує дані про вартість квитків, здійснює оплату та отримує квиток. Уся взаємодія організована через централізовану базу даних кінотеатру, що забезпечує актуальність і синхронізацію інформації.

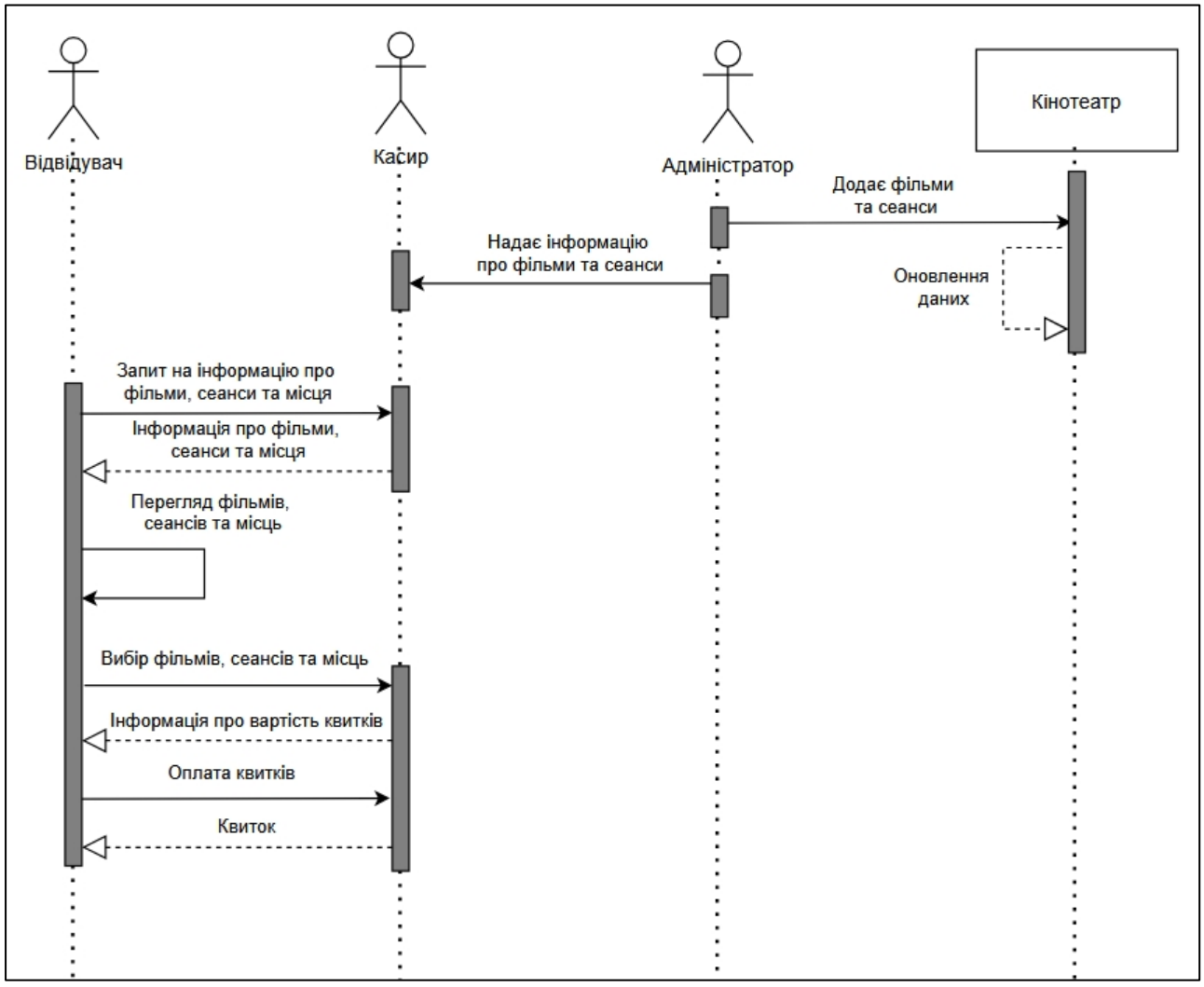


Рис. 1.6 Діаграма послідовності

1.4.3 Діаграма діяльності. Діаграма активності - це як об’єктно-орієнтована блок-схема, яка представляє потік виконання від однієї діяльності до іншої та відображає динамічну поведінку системи. Вона описує, як система виконує певну роботу або процес [7].

На діаграмі, зображеній на рис. 1.7, представлено процеси взаємодії відвідувача та касира в автоматизованій системі продажу квитків кінотеатру.
Відвідувач розпочинає з перегляду інформації про фільми. Далі він приймає рішення: завершити перегляд або перейти до купівлі квитка. У випадку купівлі відвідувач обирає фільм, сеанс і місце, після чого здійснює оплату квитка.
Касир паралельно надає інформацію про доступні фільми, сеанси та місця, а також виконує продаж квитків через систему. Процеси обох учасників координуються з єдиною інформаційною базою даних кінотеатру, що забезпечує актуальність вибору місць та фільмів.

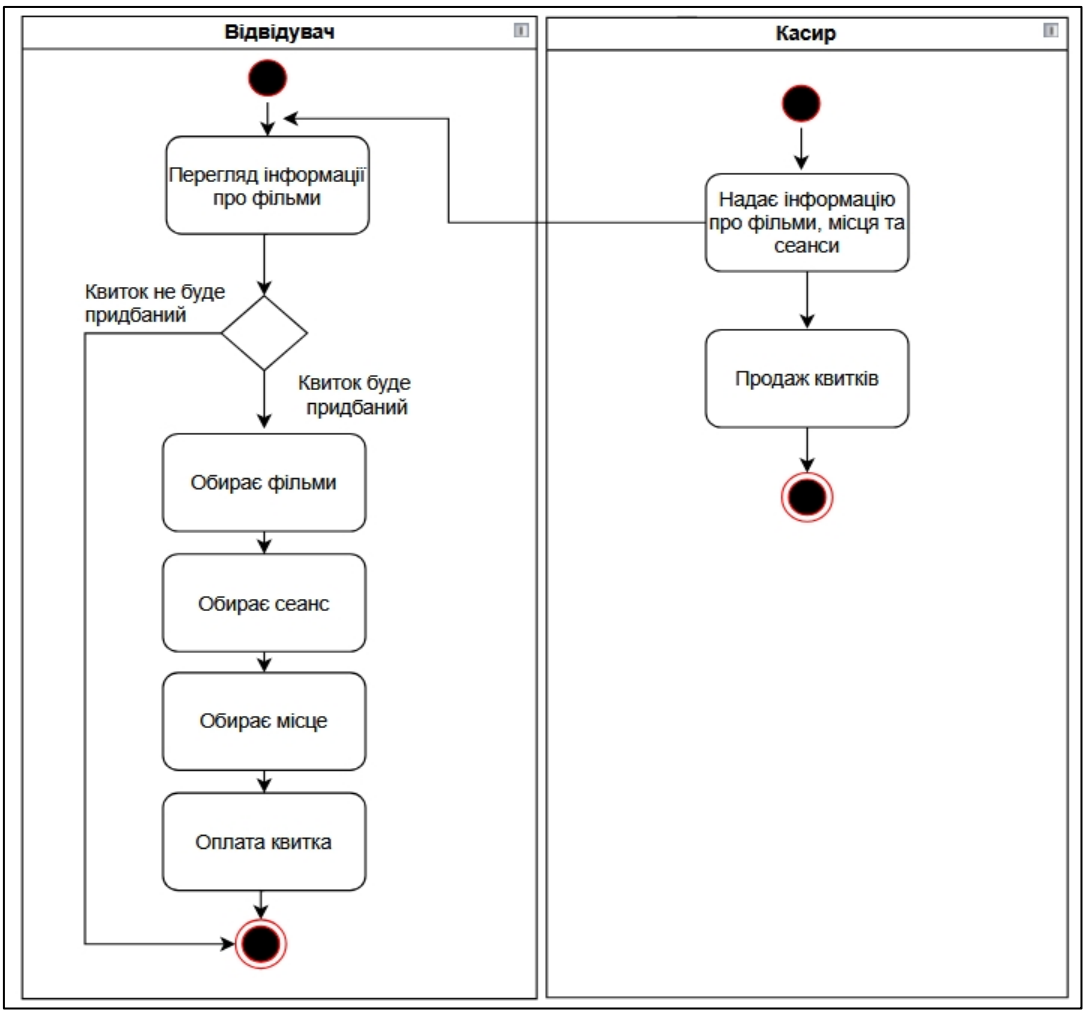


Рис. 1.7 Діаграма активності

2 ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1 Логічна модель даних у вигляді ER-діаграми

Для проєктування схеми бази даних було використано концептуальну модель «сутність-зв’язок» (ER-модель), яка відображає сутності, їх атрибути та зв’язки між ними, забезпечуючи логічну структуру даних. ER-діаграма демонструє взаємозв’язок сутностей, що зберігаються в базі даних і широко використовується під час фази проєктування інформаційної системи. Вона дозволяє описати інформаційні вимоги та типи даних, які зберігатимуться в базі, що робить її важливим інструментом для побудови концептуальної структури системи[8].

У системі автоматизованого продажу квитків у кінотеатрі були визначені основні сутності: квитки, сеанси, кінотеатри, фільми, країна та категорії. Кожна з цих сутностей містить набір атрибутів, які дозволяють ідентифікувати її та описати відповідні характеристики. Для встановлення зв’язків між сутностями використовуються зовнішні ключі, що гарантують цілісність даних та забезпечать ефективну роботу системи. Логічну модель системи продемонстровано в ДОДАТКУ А.

Для коректної реалізації моделі «сутність-зв’язок», забезпечення цілісності даних та ефективного управління залежностями між сутностями використовувались наступні зв’язки:

- **ідентифікуючі зв’язки:** використовуються, коли дочірня сутність не може існувати без батьківської. У таких випадках первинний ключ дочірньої сутності включає зовнішній ключ батьківської сутності;
- **неідентифікуючі зв’язки:** використовуються, коли дочірня сутність може існувати незалежно від батьківської. У цьому випадку зовнішній ключ батьківської сутності присутній у дочірній сутності, але не є частиною її первинного ключа;

- **батьківські та дочірні сутності:** у контексті зв'язків, батьківська сутність -це та, яка надає свій ключ для зв'язку, а дочірня - яка отримує цей ключ.

2.1.1 Основні сутності та їх зв’язки.

1. Квиток (Ticket). Містить інформацію про конкретний квиток: ідентифікатор, кінотеатр, фільм, дата, ряд, місце та ціну.

Зв’язки:

- неідентифікуючий зв’язок із сутністю розклад, оскільки зовнішні ключі «id_cinema», «id_movie», «dataOfSeason» не є частиною первинного ключа таблиці «Квиток», а існують як окремі поля;
- дочірня сутність по відношенню до «Розкладу», оскільки залежить від нього;
- має зв’язок із «Фільмом» через «id_movie», який також є неідентифікуючим.

2. Розклад (Schedule). Описує коли і де відбувається показ фільму. Первинним ключем є комбінація «id_cinema», «id_movie», «dataOfSeason».

Зв’язки:

- ідентифікуючий зв’язок з «Кінотеатром» та «Фільмом», оскільки їхні ключі входять до складу первинного ключа «Розклад»;
- дочірня сутність по відношенню до «Кінотеатру» та «Фільму»;
- батьківська сутність до «Квитку»;
- є посередником між 3 сутностями «Квиток», «Фільм» та «Кінотеатр».

3. Кінотеатр (Cinema). Містить інформацію про кінотеатр ідентифікатор, назву та кількість залів.

Зв’язки:

- батьківська сутність у зв’язку з «Розкладом», один «Кінотеатр» може мати багато «Сеансів»;
- має ідентифікуючий зв’язок один-до-багатьох з «Розкладом».

4. **Фільм (Movie).** Містить детальну інформацію про фільм, такі як назва, тривалість, режисер, опис, зображення, вікове обмеження.

Зв’язки:

- батьківська сутність у зв’язках з «Квитком» та «Розкладом»;
- має неідентифікуючий зв’язок з країною, оскільки «id_country» не є частиною первинного ключа;
- має зв’язок багато-до-багатьох із «Категорією».

5. **Країна (Country).** Вказує країну походження фільму.

Зв’язки:

- батьківська сутність до «Фільму»;
- має неідентифікуючий зв’язок з «Фільмом».

6. **Категорія (Category).** Містить атрибути фільму, такі як жанр, країна, рік, режисер.

Зв’язки:

- батьківська сутність по відношенню до зв’язувальної таблиці «Категорія-Фільм»;
- має зв’язок багато-до-багатьох, реалізується через ідентифікуючий зв’язок із допоміжною таблицею де «id_category» та «id_movie» формують первинний ключ.

Ця схема бази даних дозволяє ефективно організувати зберігання, обробку та доступ до інформації, пов’язаної з продажем квитків у кінотеатрі. Завдяки використанню зв’язків між сутностями, забезпечується узгодженість даних, зниження дублювання інформації та можливості швидкого формування запитів.

2.1.2 Нормалізація бази даних. Відповідно [9], “нормальна форма - властивість відношення в реляційній моделі даних, що характеризує його з погляду надмірності, що потенційно призводить до логічно помилкових результатів вибірки або зміни даних. Нормальна форма окреслюється сукупністю вимог, яким має задовольняти відношення таблиць у базі даних ”.

Нормалізація бази даних є важливим етапом проєктування, що дозволяє зменшити надлишковість, уникнути логічних аномалій та забезпечити цілісність даних. У процесі нормалізації таблиці бази даних організовуються таким чином, щоб кожна з них відображала лише одну сутність або логічну одиницю інформації. Це досягається за допомогою визначення ключових атрибутів, встановлення залежностей між полями та поділу таблиць при необхідності. Застосування нормалізації покращує узгодженість даних та полегшує підтримку бази даних у довгостроковій перспективі.

2.1.3 Аналіз відповідності діаграми нормальним формам. Дана діаграма відповідає першій нормальній формі, оскільки всі атрибути в таблицях мають атомарні значення, а кожна сутність має визначений первинний ключ, який однозначно ідентифікує запис.

Діаграма відповідає другій нормальній формі, оскільки вона перебуває у першій нормальній формі і кожен неключовий атрибут залежить від усього первинного ключа, а не лише від його частини. Це забезпечує усунення часткових залежностей у таблицях.

Діаграма також відповідає третій нормальній формі, оскільки вона задовольняє вимоги другої нормальної форми і всі неключові атрибути в таблицях безпосередньо залежать від первинного ключа та немає зайвих проміжних залежностей. Це дозволяє краще організувати дані та уникнути дублювання.

Таким чином реляційна модель вважається завершеною, тому що, вона досягла такої форми нормалізації, за якої усунені функціональні залежності, що спричиняють аномалії, і структура даних повністю відповідає вимогам предметної області.

2.2 Вибір системи керування базами даних

Для реалізації програмного забезпечення системи автоматизованого продажу квитків у кінотеатрі необхідне використання надійної та ефективної системи керування базами даних, яка здатна обробляти великі обсяги даних, забезпечувати рівні доступу в систему, гарантувати цілісність інформації та швидке виконання запитів.

Перед процесом розробки системи розглядались кілька популярних рішень для управління базою даних з реляційною структурою, таких як: MySQL, PostgreSQL, SQLite, та Microsoft SQL Server. Кожна з цих система має свої переваги та недоліки, однак враховуючи специфіку проєкту та потребу в високому рівні безпеки, можливості зручної роботи та інтеграції з Visual Studio, а також необхідність ефективного адміністрування, було прийнято рішення використовувати Microsoft SQL Server.

Обраний Microsoft SQL Server забезпечує стабільну та безпечну роботу з даними, що охоплюють фільми, сеанси, квитки та користувачів. Його функціональність відповідає вимогам системи з точки зору адміністрування, масштабованості та захисту інформації, що робить його доцільним вибором для реалізації даного програмного продукту.

2.2.1 Переваги Microsoft SQL Server. Microsoft SQL Server - це сучасна реляційна система управління базами даних, що поєднує високу продуктивність, інтеграцію з екосистемою Microsoft і розширені засоби безпеки. Серед ключових переваг, що стали вирішальними при виборі цієї платформи, слід зазначити [10].

- **Інтеграція з платформою Microsoft.** SQL Server тісно інтегрується з іншими продуктами Microsoft, зокрема Visual Studio, .NET Framework, що значно спрощує розробку, налаштування та підтримку програми.
- **Висока продуктивність.** SQL Server має вбудовані механізми оптимізації запитів, що забезпечують швидке виконання складних операцій з

великими обсягами даних, що важливо для системи продажу квитків, яка має бути розрахована на великий обсяг користувачів.

- **Засоби безпеки.** SQL Server підтримує автентифікацію Windows, шифрування даних, контроль доступу на основі ролей, що дозволяє надійно захищати персональні дані користувачів та фінансову інформацію.
- **Надійність і відновлення.** Завдяки функціям резервного копіювання та відновлення, збереження журналів транзакцій та спеціальних механізмів, які дозволяють системі продовжувати роботу навіть у разі збою одного з її компонентів, забезпечується безперервна робота система навіть під час збоїв.
- **Масштабованість і продуктивність.** SQL Server легко адаптується до зростання обсягів даних і кількості користувачів, що дає змогу розвивати систему без необхідності її повної перебудови.
- **Інструменти для адміністрування.** Наявність SQL Server Management Studio дозволяє ефективно управляти базами даних, виконувати SQL-запити, налаштовувати безпеку, створювати резервні копії, моніторити навантаження та виконання запитів.
- **Звітування та аналітика.** Вбудована підтримка сервісів звітування SQL Server Reporting Services дозволяє генерувати звіти за продажами, статистикою відвідуваності.

2.3 Реалізація бази даних з використанням Microsoft SQL Server

На цьому етапі буде здійснено фізичну реалізацію розробленої концептуальної моделі бази даних у середовищі Microsoft SQL Server. Для створення структури бази даних використовувались команди мови SQL (Structured Query Language), для визначення типів даних та оператори.

- CREATE DATABASE - було застосовано оператор для створення нової бази даних [11].
- CREATE TABLE - було застосовано оператор для створення таблиць [12].
- PRIMARY KEY - первинні ключі було застосовано для унікальної ідентифікації рядків [13].
- FOREGIN KEY - зовнішні ключі було застосовано для зв'язування таблиць між собою [14].

2.3.1 Створення бази даних (CinemaBD). Код реалізації бази даних показано на рис. 2.1.

```
CREATE DATABASE CinemaBD;
```

Рис. 2.1 Код реалізації бази даних «CinemaBD»

2.3.2 Таблиця «Кінотеатр» (Cinema). Код реалізації таблиці показано на рис. 2.2.

```
CREATE TABLE Cinema (
    id_cinema INT IDENTITY(1,1) PRIMARY KEY,
    nameOfCinema NVARCHAR(255) NOT NULL,
    hall_number INT NOT NULL
);
```

Рис. 2.2 SQL-код реалізації таблиці «Cinema»

2.3.3 Таблиця «Країна» (Country). Код реалізації таблиці показано на рис. 2.3.

```
CREATE TABLE Country (
    id_country INT IDENTITY(1,1) PRIMARY KEY,
    nameOfCountry NVARCHAR(255) NOT NULL
);
```

Рис. 2.3 SQL-код реалізації таблиці «Country»

2.3.4 Таблиця «Фільм» (Movie). Код реалізації таблиці показано на рис. 2.4.

```
CREATE TABLE Movie (
    id_movie INT IDENTITY(1,1) PRIMARY KEY,
    id_country INT,
    nameOfMovie NVARCHAR(255) NOT NULL,
    duration INT NOT NULL,
    producer NVARCHAR(255),
    description NVARCHAR(MAX),
    movie_image NVARCHAR(255),
    age_limit INT,
    FOREIGN KEY (id_country) REFERENCES Country(id_country) ON DELETE SET NULL
);
```

Рис. 2.4 SQL-код реалізації таблиці «Movie»

2.3.5 Таблиця «Категорія» (Category). Код реалізації таблиці показано на рис. 2.5.

```
CREATE TABLE Category (
    id_category INT IDENTITY(1,1) PRIMARY KEY,
    by_genre NVARCHAR(100),
    by_country NVARCHAR(100),
    by_producer NVARCHAR(100),
    by_year INT
);
```

Рис. 2.5 SQL-код реалізації таблиці «Category»

2.3.6 Таблиця «Фільм_Категорія» (MovieCategory). Код реалізації таблиці показано на рис. 2.6.

```
CREATE TABLE MovieCategory (
    id_category INT,
    id_movie INT,
    PRIMARY KEY (id_category, id_movie),
    FOREIGN KEY (id_category) REFERENCES Category(id_category) ON DELETE CASCADE,
    FOREIGN KEY (id_movie) REFERENCES Movie(id_movie) ON DELETE CASCADE
);
```

Рис. 2.6 SQL-код реалізації таблиці «MovieCategory»

2.3.7 Таблиця «Розклад» (Schedule). Код реалізації таблиці показано на рис. 2.7.

```
CREATE TABLE Schedule (
    id_cinema INT,
    id_movie INT,
    DataOfSeasion DATE,
    movie_time TIME,
    PRIMARY KEY (id_cinema, id_movie, DataOfSeasion),
    FOREIGN KEY (id_cinema) REFERENCES Cinnema(id_cinema) ON DELETE CASCADE,
    FOREIGN KEY (id_movie) REFERENCES Movie(id_movie) ON DELETE CASCADE
);
```

Рис. 2.7 SQL-код реалізації таблиці «Schedule»

2.3.8 Таблиця «Квиток» (Ticket). Код реалізації таблиці показано на рис. 2.8.

```
CREATE TABLE Ticket (
    id_ticket INT IDENTITY(1,1) PRIMARY KEY,
    id_cinema INT,
    id_movie INT,
    dateOfSeasion DATE,
    row_number INT NOT NULL,
    place_number INT NOT NULL,
    price DECIMAL(10,2) NOT NULL,
    FOREIGN KEY (id_cinema, id_movie, dateOfSeasion) REFERENCES Schedule(id_cinema,
id_movie, DataOfSeasion) ON DELETE CASCADE
);
```

Рис. 2.8 SQL-код реалізації таблиці «Ticket»

2.3.9 Створення користувачів системи. Створення користувачів системи передбачає визначення трьох основних ролей: адміністратор, касир та звичайний відвідувач. Для кожного з них були створені відповідні облікові записи з індивідуальними рівнями доступу до функціоналу системи. На рис. 2.9 наведено код створення таблиці користувачів, в ДОДАТКУ Б надання прав доступу відповідно до призначених ролей.

```
--створення ролі адміністратор
CREATE login admincinema
WITH password = 'admin'
CREATE user admincinema for login admincinema

--створення ролі касир
CREATE login cashiercinema
WITH password = 'cashier'
CREATE user cashiercinema for login cashiercinema

--створення ролі відвідувач
CREATE login usercinema
WITH password = 'user'
CREATE user usercinema for login usercinema
```

Рис. 2.9 SQL-код створення користувачів

2.3.10 Процедура додавання нового фільму. Збережені процедури в SQL - це набір команд, які виконують певні дії та можуть викликатися багаторазово. Вони схожі на функції в мовах програмування, оскільки дозволяють передавати параметри та централізовано виконувати логіку [15].

Для автоматизації процесу внесення нових фільмів до бази даних було створено збережену процедуру. Процедура додавання фільму приймає параметри, такі як назва фільму, опис, рік випуску, країна-виробник та ідентифікатор кінотеатру, в якому фільм демонструється. Це дозволяє зменшити дублювання коду, спростити адміністрування та забезпечити зручне додавання фільмів у систему. SQL-код створення процедури продемонстровано в ДОДАТКУ В.

2.3.11 Тригер заборони дублювання місць на один сеанс. Тригер - це особливий тип збереженої процедури, яка автоматично виконується при настанні певної події у реляційній базі даних без участі користувача[16].

Для забезпечення цілісності даних та уникнення ситуацій, коли одне й те саме місце продається кілька разів на той самий сеанс, було створено тригер. У даному випадку тригер перевіряє, чи не зайняте вже місце для вказаного сеансу, і в разі дублювання автоматично блокує додавання нового квитка, тим самим забезпечуючи унікальність бронювання. SQL-код створення тригера заборони дублювання місць на один сеанс продемонстровано в ДОДАТКУ Д.

2.3.12 Створення уявлення виводу сеансів на вибраний фільм. Уявлення - це віртуальна таблиця в SQL, яка містить рядки і стовпці, подібні до звичайної таблиці, але не зберігає фактичних даних. Вона створюється на основі однієї або кількох таблиць з метою спрощення доступу та роботи з даними [17].

Для зручного виведення інформації про доступні сеанси певного фільму було створено уявлення. Це уявлення об’єднує таблиці «Movie», «Schedule», та «Cinema», щоб у результаті користувач міг побачити назву фільму, назву кінотеатру, дату й час показу. SQL-код створення представлення наведено на рис. 2.10.

```sql
--представлення виводу сеаснів на вибраний фільм
CREATE VIEW vw_SessionsByMovie AS
SELECT
    m.nameOfMovie,
    c.nameOfCinema,
    s.DataOfSeasion AS session_date,
    s.movie_time AS session_time
FROM Schedule s
JOIN Movie m ON s.id_movie = m.id_movie
JOIN Cinnema c ON s.id_cinema = c.id_cinema;
```

Рис. 2.10 SQL-код створення уявлення виведення сеансів на вибраний фільм

2.4 Загальна структура бази даних

У результаті реалізації створена база даних, як забезпечує зберігання, зв’язки та обробку інформації про кінотеатри, фільми, розклад сеансів, квитки та категорії фільмів. Структуру бази даних у вигляді діаграми наведено в ДОДАТКУ Е.

3 ПРОЄКТУВАННЯ І РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Організаційна структура програмного забезпечення

Організаційна структура програмного забезпечення відображає внутрішню логіку системи, що реалізує автоматизований продаж квитків у кінотеатрі. Вона охоплює основні компоненти системи, їхні функціональні ролі, взаємодію між об'єктами та логічну побудову програмної архітектури. Для реалізації проєкту було застосовано об'єктно-орієнтований підхід, що дозволяє створити гнучку, масштабовану та легко підтримувану систему. У цьому підрозділі наведено основні абстракції, які моделюють поведінку користувачів і сутностей системи, а також описано логічну структуру у вигляді діаграми класів та діаграми пакетів.

3.1.1 Абстракції. Абстракція - це концепція в UML та програмуванні, яка полягає у виділенні суттєвих характеристик об’єкта або процесу, відкидаючи несуттєві деталі. У контексті UML абстракція означає подання моделі на вищому рівні узагальнення елементи описуються через їхню загальну поведінку або властивості, що дозволяє створювати простіші та зрозуміліші моделі складних систем [18]. На основі функціональних вимог до програмного забезпечення були виділені такі ключові абстракції.

- **Відвідувач -** користувач, який переглядає репертуар фільмів, обирає сеанси, купує квитки. Має атрибут вік та здійснює дії, пов’язані з вибором і купівлею. Продемонстровано на рис. 3.1.

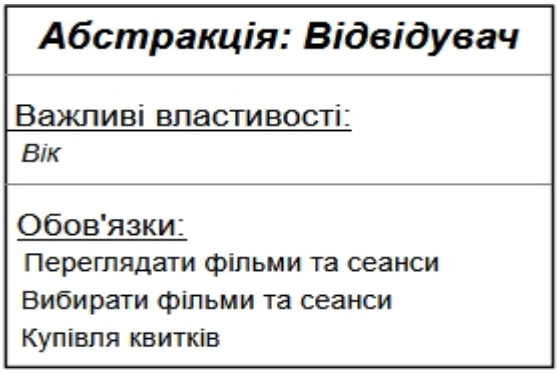


Рис.3.1 Абстракція «Відвідувач»

● **Квиток -** електронний документ, що містить інформацію про оплату, фільм, сеанс, зал та місце. Включає дату покупки, тип та суму оплати. Продемонстровано на рис. 3.2.

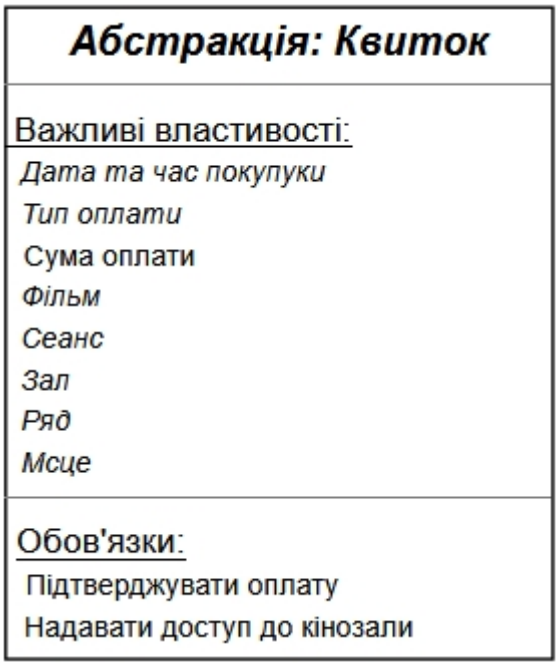


Рис. 3.3 Абстракція «Квиток»

● **Сеанс -** представлення показу фільму в певний час, у конкретному форматі та залі. Включає атрибути: фільм, час, формат, зал. Продемонстровано на рис. 3.4.

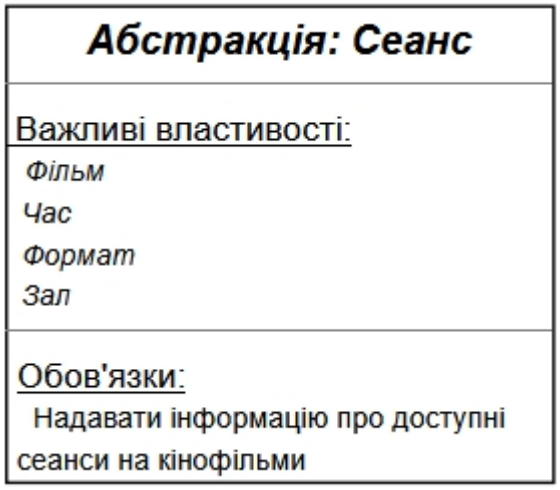


Рис. 3.4 Абстракція «Сеанс»

● **Фільм -** об’єкт, що зберігає основну інформацію про кінофільм: назву, автора, жанр, рік, опис та вікові обмеження. Продемонстровано на рис. 3.5.

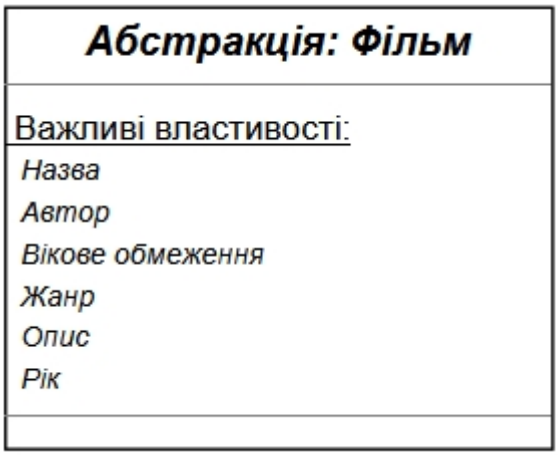


Рис. 3.5 Абстракція «Фільм»

● **Місце -** об’єкт, що характеризує конкретне місце в залі: ряд, номер місця, статус (вільне/зайняте). Продемонстровано на рис. 3.6.

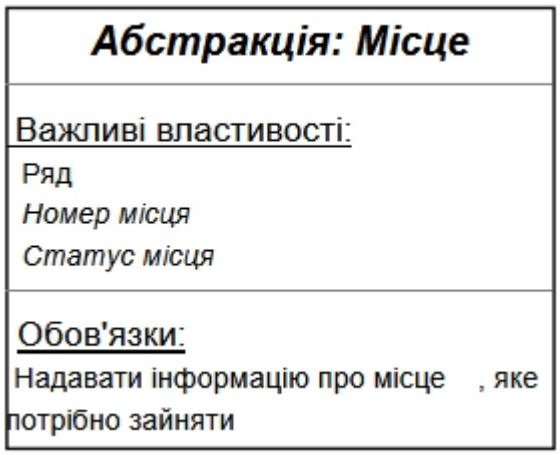


Рис. 3.6 Абстракція «Місце»

● **Касир -** співробітник, що займається продажем квитків. Має атрибут ПІБ та відповідний функціонал. Продемонстровано на рис. 3.7.

Рис. 3.7 Абстракція «Касир»

● **Адміністратор -** відповідальний за управління розкладом сеансів і цінами на квитки. Має доступ до редагування даних і підтвердження оплати. Продемонстровано на рис. 3.8.

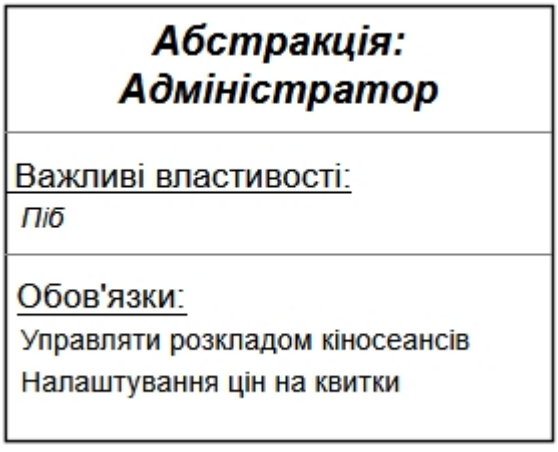


Рис. 3.8 Абстракція «Адміністратор кінотеатру»

3.1.2 Діаграма класів. Діаграма класів у мові моделювання UML є статичною структурною діаграмою, яка відображає складові частини системи: класи, їхні атрибути, методи, а також взаємозв'язки між ними. Вона слугує засобом для опису внутрішньої будови програмного забезпечення [19].

Для відображення логічної структури системи та взаємозв'язків між основними компонентами було створена діаграма класів з асоціаціями. Ця діаграма демонструє основні сутності, які беруть участь у функціонуванні програмного забезпечення, їхні атрибути, метода та зв'язки. Вона є частиною об'єктно-орієнтованого аналізу й дозволяє краще зрозуміти архітектуру системи, спростити реалізацію та підтримку коду, її можна побачити на рис. 3.9.

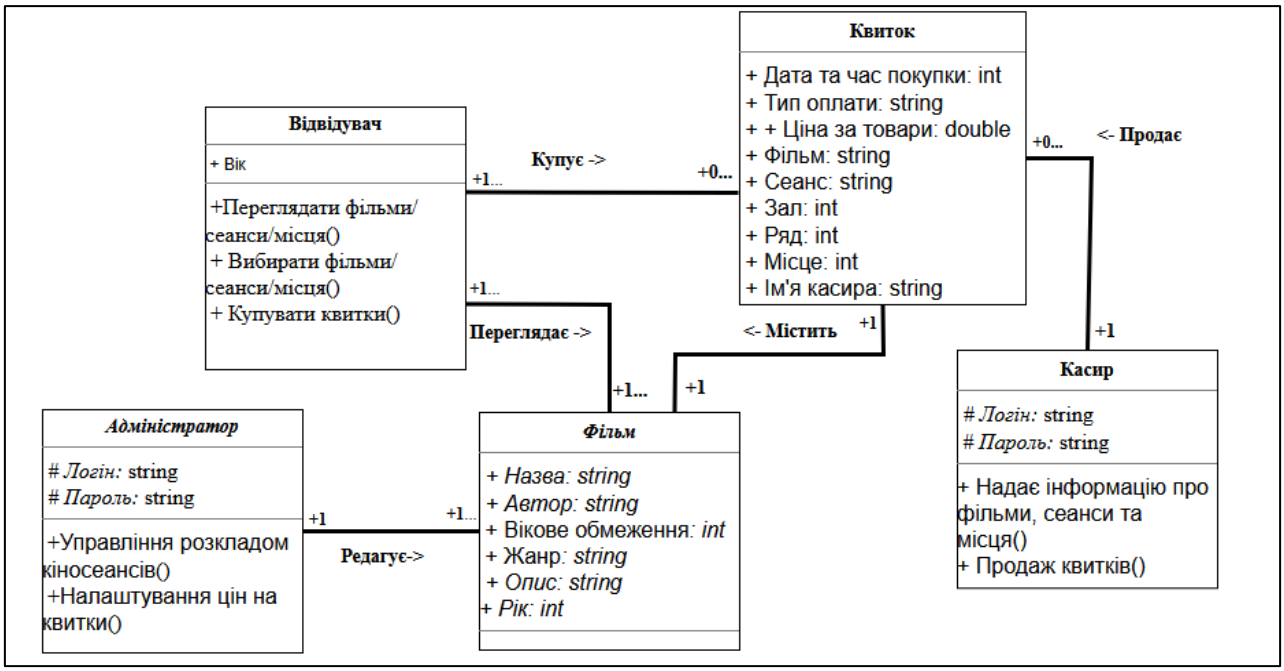


Рис. 3.9 Діаграма класів

На діаграмі класів зображено 5 основних класів.

- **Відвідувач** - користувач системи, який може переглядати наявні фільми, сеанси та місця, вибирати потрібний варіант і купувати квитки. Має атрибут вік і методи для виконання відповідних дій.

- **Квиток** - містить інформацію про дату та час покупки, тип оплати, ціну, назву фільму, час сеансу, зал, ряд місце та ім'я касира, що продав квиток. Клас пов'язаний з фільмом, залом та касиром.

- **Фільм -** включає атрибути, які описують фільм: назву, автора, жанр, вікове обмеження, опис та рік випуску. Клас пов’язаний із відвідувачем, який його переглядає , адміністратором, який його редагує та квитком, який містить інформацію про нього.

- **Касир -** відповідає за продаж квитків та надання інформації про фільми, сеанси і місця. Має атрибути логін і пароль, а також методи для реалізації цих функцій.

- **Адміністратор -** керує розкладом сеансів та цінами на квитки. Також має атрибути логін і пароль та відповідні методи.

Зв’язки між класами демонструють взаємодію між об’єктами:

- відвідувач переглядає фільм та купує квиток;
- касир продає квиток;
- адміністратор редагує фільм;
- квиток містить інформацію про фільм.

3.1.3 Прості кооперації. Кооперація - це сценарій взаємодії між об'єктами, що працюють разом для досягнення певної функціональності. Вона показує, як об'єкти обмінюються повідомленнями з метою реалізації певного випадку використання або дії.

Далі наведені прості кооперації в інформаційній системі продажу квитків у кінотеатрі.

1. Проста кооперація «Взаємодія адміністратора з фільмом». Ця кооперація описує сценарій, коли адміністратор працює взаємодіє з фільмом зокрема:

- додає новий фільм до репертуару;
- налаштовує або редагує інформацію про фільм;
- управляє розкладом сеансів та цінами квитків.

Об’єкти простої кооперації:

- адміністратор;
- фільм.

Основна взаємодія:

- адміністратор передає інформацію фільму, викликаючи метод включення фільму до репертуару;
- також адміністратор керує цінами та розкладом через відповідні функції.

Це ілюструє контроль адміністраторського рівня над контентом кінотеатру, цю кооперацію продемонстровано на рис. 3.10.

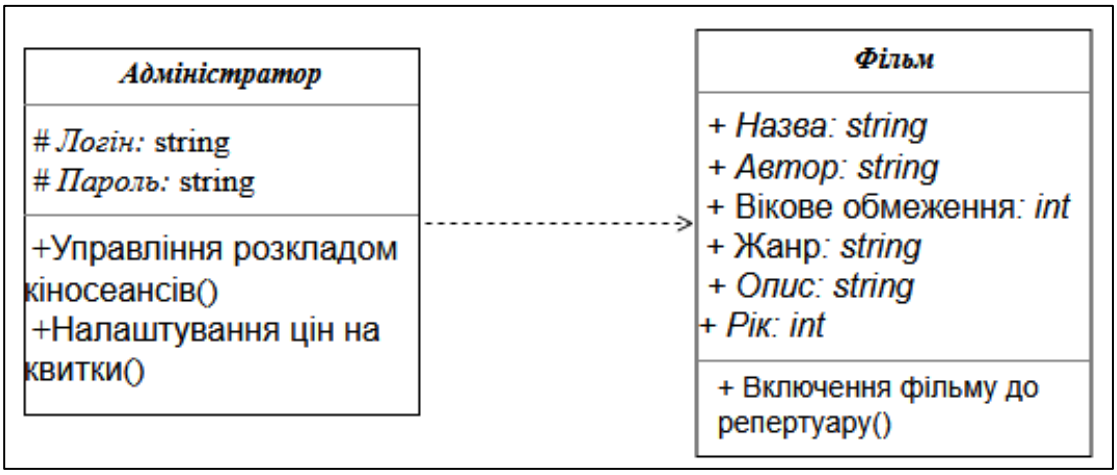


Рис. 3.10 Проста кооперація «Взаємодія адміністратора з фільмом»

2. Проста кооперація "Купівля квитка". Ця кооперація описує взаємодію між відвідувачем, касирами та системою квитків під час здійснення покупки:

Об’єкти простої кооперації:

- відвідувач;
- касир;
- квиток.

Основна взаємодія:

- відвідувач вибирає фільм, сеанс, місце;
- касир обробляє покупку, надає інформацію та оформлює продаж;
- створюється екземпляр квиток з відповідними даними: дата, час, зал, ряд, місце.

Ця кооперація демонструє, як система обслуговує процес покупки квитка від вибору до видачи квитки, , цю кооперацію продемонстровано на рис. 3.11.

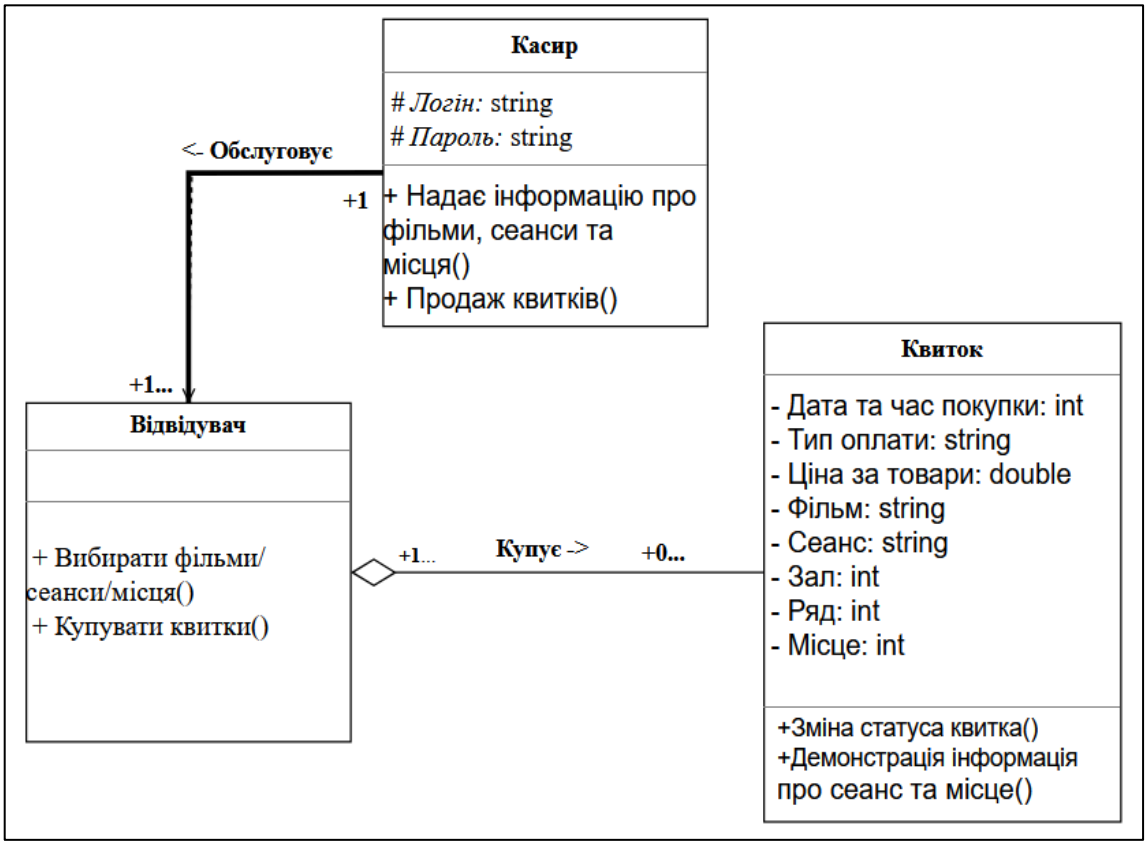


Рис. 3.11 Проста кооперація "Купівля квитка"

3.2 Архітектура системи

Архітектура системи визначає загальну структуру програмного забезпечення, його основні компоненти та взаємозв’язки між ними. Вона є ключовим етапом проєктування, що дозволяє забезпечити модульність, масштабованість, зрозумілість і підтримуваність системи.

Інформаційна система продажу квитків у кіно побудована за принципами багаторівневої архітектури, яка включає наступні основні рівні:

- **рівень користувача** - відповідає за взаємодію з кінцевими користувачами, такими як відвідувачі, касири та адміністратори. Інтерфейс дозволяє користувачу переглядати доступні фільми та сеанси, обирати місця, купувати квитки;

• **рівень бізнес-логіки** - реалізує основну функціональність системи. Тут обробляються запити користувачів, виконується перевірка ролей, продаж квитків, управління репертуаром, налаштування розкладу та цін;

• **рівень доступу до даних** - відповідає за збереження та отримання інформації з бази даних. Запити до бази даних забезпечують доступ до інформації про фільми, сеанси, користувачів та куплені квитки.

Загальна архітектура системи забезпечує розділення обов'язків між компонентами, що сприяє гнучкості при оновленнях або розширеннях функціональності.

Усі компоненти системи взаємодіють через чітко визначені інтерфейси та відповідальні лише за свої функціональні обов'язки, що робить систему більш захищеною, надійною та масштабованою.

У даному проєкті реалізована тришарова клієнт-серверна архітектура, яка чітко розділяє обов'язки між клієнтською частиною, серверною логікою та рівнем роботи з базою даних.

3.2.1 Клієнтській рівень. Клієнтська частина надає інтерфейс користувача для взаємодії з системою. Саме через неї відвідувачі обирають фільми, переглядають розклад сеансів, обирають місця та купують квитки. Адміністратори та касири також мають відповідні інтерфейси для виконання функцій керування репертуаром, розкладом і продажем квитків.

Для реалізації клієнтської частини використано технологію.

• Windows Forms - для терміналу самообслуговуваня, а також застосунку касира та адміністратора.

3.2.2 Серверний рівень. Серверна частина виконує роль посередника між клієнтом і базою даних, обробляє бізнес-логіку та забезпечує безпеку та контроль доступу. Саме тут реалізуються основні процеси: перевірка прав доступу, обробка запитів на покупку квитків, керування фільмами та розкладом, формування звітів.

Для реалізації серверної частини використано такі технології:

- C# - як основна мова програмування для логіки застосунку;
- ASP.NET - для обробки запитів від десктопного клієнта;
- RESTful API - для обміну даними у форматі JSON через HTTP-протокол.

3.2.3 Рівень доступу до даних. Цей рівень відповідає за взаємодію з базою даних, забезпечуючи зчитування, збереження та оновлення інформації про фільми, сеанси та квитки.

Для реалізації доступу до даних використано таку технологію.

- **ADO.NET** - для підключення до бази даних Microsoft SQL Server, виконання SQL-запитів та обробки результатів.

3.2.4 Переваги обраної архітектури.

- **Модульність**. Легко підтримувати та оновлювати окремі частини системи без ризику впливу на інші компоненти.
- **Масштабованість**. Можливість розширення системи, додавання нових функцій, ролей, підтримка мобільного застосунку.
- **Гнучкість**. Реалізація інтерфейсів для різних типів користувачів з урахуванням ролей.
- **Безпека**. Розмежування доступу до різних частин системи, збереження конфіденційних даних.
- **Повторне використання коду**. Використання єдиного API для різних клієнтів.

Таким чином, обрана архітектура реалізує принципи шаблону Model-View-Controller (MVC), що дозволяє розділити відповідальність між інтерфейсом користувача (View), логікою керування (Controller) та доступом до даних (Model). Такий підхід підвищує керованість, спрощує тестування і супровід системи, а також сприяє повторному використанню компонентів.

3.2.5 Діаграма пакетів. На основі багаторівневої архітектури, описаної у попередніх підрозділах, було побудовано діаграму пакетів, що візуалізує модульну структуру інформаційної системи продажу квитків у кінотеатрі.

Діаграма пакетів у мові моделювання UML є структурною діаграмою, що відображає залежності між логічними групами елементів моделі, які об’єднані у пакети. Пакет виступає засобом для організації структури великої системи, спрощуючи її сприйняття та підтримку. Кожен пакет може містити класи, інтерфейси або інші пакети, що дозволяє впорядкувати компоненти за функціональністю чи роллю в системі [20].

Ця діаграма дозволяє побачити модульну структуру програмного забезпечення, визначити межі відповідальності окремих компонентів і забезпечити краще планування розробки та тестування системи.

На діаграмі пакетів, представленій в ДОДАТКУ Ж, система програмного забезпечення розподілена на такі логічні пакети.

- **Клієнт -** включає користувачів системи, які взаємодіють через клієнт з системою з метою перегляду інформації про фільми, сеанси, місця та купівлю квитків.
- **Купівля квитка -** відповідає за процес придбання квитків та включає класи квиток і фільм, які пов’язані із відповідними сутностями в системі.
- **Управління фільмом -** містить класи, пов’язані з адмініструванням кінопоказів, включаючи адміністратора та фільм.
- **База даних -** окремий пакет, який містить класи, відповідальні за взаємодію з базою даних, зокрема реалізацію запитів.

3.3 Алгоритмізація та програмування різних модулів

Інформаційна система продажу квитків у кіно реалізує ключові функціональні можливості, що забезпечують ефективну взаємодію користувачів із системою та підтримку адміністративного управління.

Основні функції:

- авторизація користувачів із розмежуванням доступу за ролями (відвідувач, касир, адміністратор);
- додавання фільмів до системи (для адміністратора);
- перегляд репертуару фільмів;
- перегляд та вибір сеансів;
- вибір місць і оформлення покупки квитків;
- продаж квитків через інтерфейс касира.

3.3.1 Авторизація користувачів із розмежуванням доступу за ролями. Функція авторизації забезпечує вхід користувача в систему із розпізнаванням його ролі: відвідувач, касир або адміністратор. Від ролі залежить інтерфейс та можливості. Такий підхід дозволяє реалізувати гнучке управління доступом і розмежувати права користувачів відповідно до їхніх функцій у системі.

Основні етапи авторизації.

1. Введення логіна та пароля користувачем.
2. Перевірка введених даних.
3. Визначення ролі користувача.
4. Перенаправлення на відповідний інтерфейс:
   - **відвідувач** - інтерфейс для перегляду фільмів та купівлі квитків;
   - **касир** - інтерфейс для продажу квитків;
   - **адміністратор** - інтерфейс для управління фільмами та сеансами.

Блок-схема реалізації авторизація користувачів із розмежуванням доступу за ролями продемонстровано на рис. 3.12, а код цього алгоритму наведений в ДОДАТКУ К.

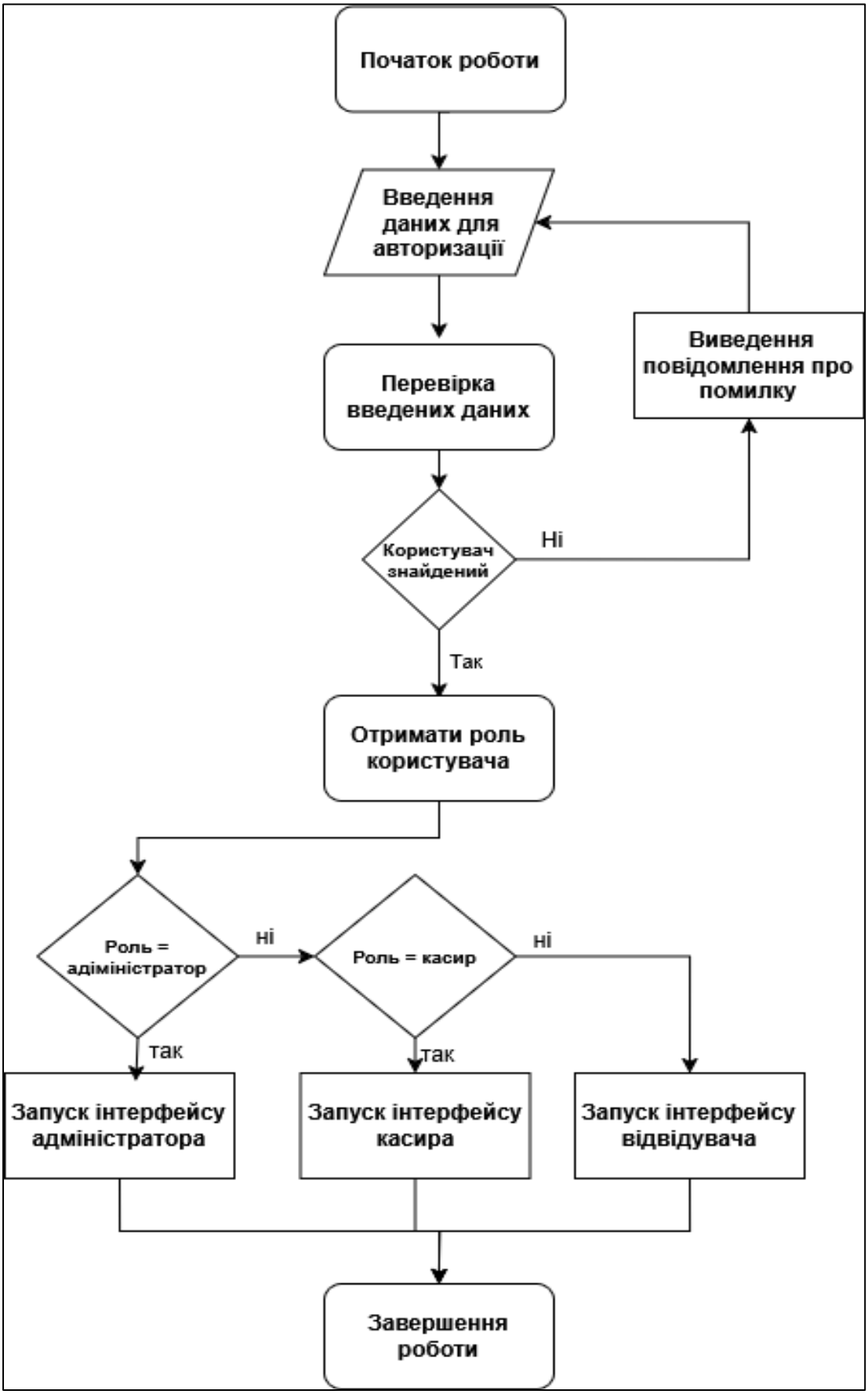


Рис. 3.12 Блок-схема реалізації авторизація користувачів із розмежуванням доступу за ролями

3.3.2 Додавання фільмів до системи. Дозволяє додавати нові фільми до репертуару. Функція доступна лише адміністратору:

- введення даних про фільм;
- додавання постера;
- збереження в базі даних.

Блок-схема реалізації додавання фільму продемонстровано на рис. 3.13, а код цього алгоритму наведений в ДОДАТКУ Л.

Рис. 3.13 Блок-схема реалізації додавання фільму в систему

3.3.3 Перегляд репертуару фільмів. Ця функція дозволяє користувачу переглядати список фільмів, доступних для перегляду, з детальною інформацією: назва, жанр, тривалість, опис, афіша.

- Отримання списку фільмів із бази даних.
- Відображення фільмів (продемонстровано в ДОДАТКУ М).
- Можливість сортування або пошуку.

3.3.4 Перегляд та вибір сеансів. Користувач обирає фільм, після чого система відображає доступні сеанси з часом та залом.

- Вибір фільму.
- Завантаження відповідних сеансів.
- Відображення сеансів (продемонстровано в ДОДАТКУ Н).

3.3.4 Вибір місць і оформлення покупки квитків. Дає змогу вибрати вільне місце на сеансі та здійснити покупку.

- Завантаження мапи залу.
- Вибір доступного місця.
- Підтвердження покупки.
- Видача квитка.

3.3.5 Продаж квитків через інтерфейс касира. Касир виконує продаж квитків через спеціальний інтерфейс.

- Вибір фільму.
- Вибір сеансу та місця.
- Підтвердження та видача квитка.

4 РЕКОМЕНДАЦІЇ ЩОДО ВПРОВАДЖЕННЯ ТА ЕКСПЛУАТАЦІЇ СИСТЕМИ

4.1 Тестування системи

У розробці програмного забезпечення тестування є критично важливим етапом, що дозволяє виявити помилки на ранніх етапах, перевірити коректність реалізованих функцій і гарантувати якість кінцевого продукту. Для системи продажу квитків у кінотеатрі тестування має особливе значення, оскільки забезпечує стабільність, безпеку та зручність використання для кінцевого користувача.

Основною метою тестування є перевірка того, що система:

- працює згідно з функціональними та нефункціональними вимогами;
- коректно обробляє дані в різних сценаріях використання;

є стійкою до помилок користувача;

- демонструє стабільну поведінку за різних навантажень.

У процесі розробки системи було застосовано кілька видів тестування.

Модульне тестування. Цей тип тестування проводиться на рівні окремих модулів або класів. Модульні тести були написані для перевірки логіки основних функцій системи:

- додавання, редагування та видалення фільмів з бази даних;
- робота із замовленнями та покупкою квитків;
- обчислення вартості квитка з урахуванням знижок.

Тестування проводилось за допомогою фреймворку NUnit. Кожен модуль був протестований із використанням як коректних, так і граничних вхідних даних.

Інтеграційне тестування. Було проведено для перевірки взаємодії між різними компонентами системи, зокрема:

- зв'язок між інтерфейсом користувача та базою даних;
- перевірка цілісності даних при виконанні запитів;
- взаємодія між модулями авторизації, замовлень і касовим інтерфейсом.

Функціональне тестування. Проводилось вручну за підготовленими сценаріями, які охоплювали всі основні функціональні можливості системи:

- авторизація користувачів;
- вибір фільму, сеансу, місця в залі;
- оформлення та оплата замовлення;
- друк квитка та надсилання квитка;
- керування фільмами.

Тестування інтерфейсу. Було перевірено:

- зручність розташування елементів управління;
- правильність відображення даних;
- поведінку при неочікуваних діях користувача.

Тестування продуктивності. Проводилось емпірично на робочій машині середнього класу з типовим обсягом даних. Було виміряно час відкриття основних вікон програми, виконання запитів до бази даних, обробки запиту на купівлю квитка тощо. Результати показали задовільну продуктивність для заданого навантаження.

Інструменти, що використовувалися:

- NUnit - для створення модульних тестів;
- SQL Server Management Studio - для перевірки коректності даних у базі;
- Visual Studio Debugger - для виявлення помилок під час виконання.

Висновки за результатами тестування

Після проходження всіх типів тестування було виявлено кілька незначних помилок, які були оперативно виправлені. У результаті система демонструє високу стабільність, відповідність поставленій меті та зручність використання. Програмне забезпечення готове до подальшого впровадження й експлуатації в реальному середовищі.

4.2 Вимоги до апаратного та програмного забезпечення

4.2.1 Діаграма розміщення. Діаграма розгортання - це різновид UML-діаграм, що описує фізичне середовище, в якому функціонує програмна система [21]. Вона показує, як компоненти програмного забезпечення встановлюються та працюють на відповідному апаратному забезпеченні, зокрема на серверах, клієнтських пристроях, базах даних або терміналах самообслуговування.

У контексті розподілених систем така діаграма моделює розміщення програмних компонентів між фізичними вузлами та ілюструє взаємодію між ними в реальному середовищі.

Діаграма розгортання є важливим інструментом для планування, розробки та впровадження системи, оскільки дозволяє візуалізувати фізичну структуру програмного забезпечення, визначити вимоги до апаратного й програмного забезпечення кожного вузла, а також забезпечити ефективне управління ресурсами, масштабованість і стабільність роботи системи.

На наведеній діаграмі розміщення, рис. 4.1, зображено фізичну архітектуру системи продажу квитків у кінотеатрі. Система складається з двох основних компонентів: клієнтського пристрою та віртуального сервера, які взаємодіють між собою за допомогою HTTP-протоколу (REST API). На клієнтському пристрої виконується файл cinema.exe, який містить основну логіку користувацького інтерфейсу. Для відображення інтерфейсу використовується окрема бібліотека ui.dll, яка підключається до основного виконуваного файлу.

На віртуальному сервері розміщується серверна частина системи, побудована на основі ASP.NET Core Web API. Вона включає в себе сервіси бізнес-логіки, які обробляють запити клієнтів, та рівень доступу до даних, який реалізується через ADO.NET. Всі запити до бази даних виконуються через відповідний інтерфейс ADO, що взаємодіє з базою даних Cinema, яка зберігається в системі управління базами даних Microsoft SQL Server.

Ця архітектура забезпечує чітке розділення відповідальностей між клієнтом і сервером, дозволяє масштабувати систему та спрощує супровід завдяки використанню модульних компонентів.

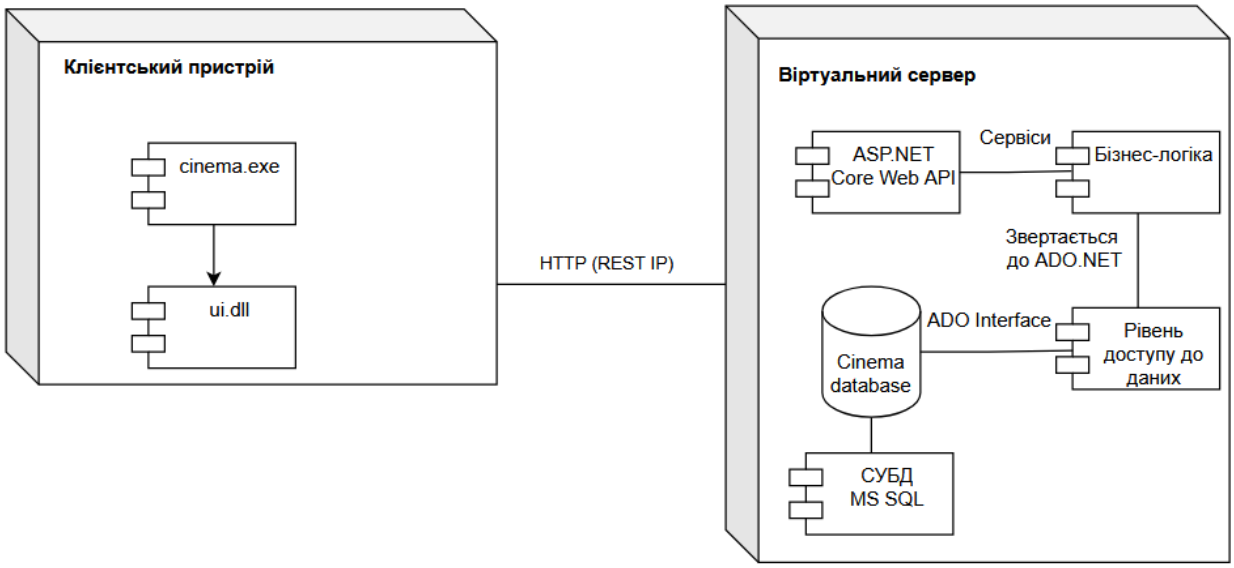


Рис. 4.1 Діграма розгортання

4.2.2 Вимоги до апаратного забезпеченн. Для розгортання та стабільної роботи системи продажу квитків у кінотеатрі необхідно забезпечити відповідні вимоги до апаратного забезпечення. Ці вимоги можуть змінюватися залежно від масштабу системи, кількості користувачів і навантаження на сервер, проте загалом рекомендованими є наступні мінімальні характеристики:

✓ **Клієнтський прістрій:**

- процесор. Не нижче 4-ядерного процесору Intel Core i3 або аналогічного;

- оперативна пам’ять. Від 4 ГБ RAM, рекомендовано 8 ГБ RAM або більше;
- дисковий простір. Не менше 2 ГБ вільного місця на SSD або HDD;
- операційна система. Windows 10 або новіше;
- мережа. Стабільне підключення до інтернету з пропускною здатністю 100 Мбіт/с.

✓ Сервер:

- процесор: не нижче 4-ядерного процесору Intel Xeon або AMD Ryzen;
- оперативна пам’ять: від 8 ГБ RAM, рекомендовано 16 ГБ RAM або більше;
- дисковий простір: від 50 ГБ вільного місця на SSD або HDD;
- операційна система: Windows Server 2019 або Linux;
- мережа: стабільне підключення до інтернету з пропускною здатністю 100 Мбіт/с.

4.3 Склад інсталяційного пакету

Інсталяційний пакет системи продажу квитків у кінотеатрі включає всі необхідні компоненти для коректного встановлення та запуска програмного забезпечення на клієнтському пристрої й сервері. Нижче наведено перелік основник складових інсталяційного пакету.

✓ Клієнтська частина:

- cinema.exe - основний виконуваний файл програми для роботи користувача;
- ui.dll - бібліотека динамічного компонування, яка забезпечує інтерфейс користувача;

- конфігураційні файли (appsettings.json, config.xml) - для налаштування підключення до сервера;
- документація , яка пояснює як розгорнути та запустити додаток. Файл readme.txt з детальними інструкціями.

✓ **Серверна частина:**

- сервіси ASP.NET Core Web API. Cinema.WebAPI.dll - серверна логіка обробки запитів та бібліотека доступу до даних;
- база даних. SQL-скрипти для створення та ініціалізації таблиць;
- конфігураційні файли сервера;
- інструкція для адміністраторів щодо розгортання серверної частини.

4.4 Демонстрація роботи програми

У цьому розділі представлено приклади роботи з інтерфейсом програмного забезпечення системи продажу квитків у кінотеатрі. Наведені скріншоти демонструють основні вікна програми та її функціональні можливості, а також зручність та логіку взаємодії користувача з інтерфейсом. Інтерфейс розроблено з урахуванням вимог до зручного використання та забезпечення комфортної роботи для касирів, адміністраторів і кінцевих користувачів.

4.4.1 Авторизація. Механізм авторизації є ключовим компонентом програмного забезпечення, оскільки визначає рівень доступу користувача до функціональності системи. Від успішної авторизації залежить, який інтерфейс буде завантажено та які дії будуть доступні користувачу. Залежно від ролі користувача (адміністратор, касир або термінал самообслуговування) система динамічно змінює інтерфейс та доступні функції:

- **Адміністратор** після авторизації отримує доступ до розширених можливостей, зокрема управління фільмами;

• **Інтерфейс касира та терміналу самообслуговування** він має спільну функціональну основу, орієнтовану на обробку продажу квитків. Він забезпечує швидкий доступ до основних дій, пов’язаних із вибором фільму, сеансу, місця у залі та здійсненням оплати.

Сторінка авторизації зображена на рис. 4.2 .

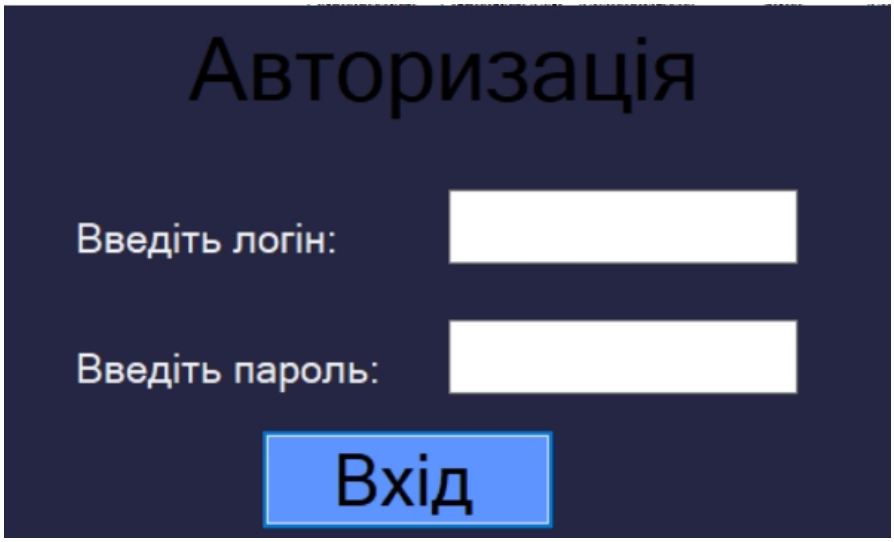


Рис. 4.2 Сторінка авторизації

4.4.2 Головна сторінка. Головна сторінка відкривається одразу після успішної авторизації в програмному забезпеченні і слугує центральним елементом взаємодії користувача з системою. Саме на ній наглядно демонструється різниця між інтерфейсами для різних категорій користувачів, що дозволяє адаптувати функціонал і відображення відповідно до їхніх прав та ролей. Це забезпечує інтуїтивне та зручне використання програми, підвищує ефективність роботи та покращує користувацький досвід.

1. Головна сторінка адміністратора

На головній сторінці адміністратора знаходиться меню для додавання та видалення фільмів, а також кнопку для переходу до каталогу фільмів. Головна сторінка адміністратора де зображений інтерфейс додавання фільмів зображена на рис. 4.3, а на рисунку 4.4 зображено інтерфейс видалення фільмів.

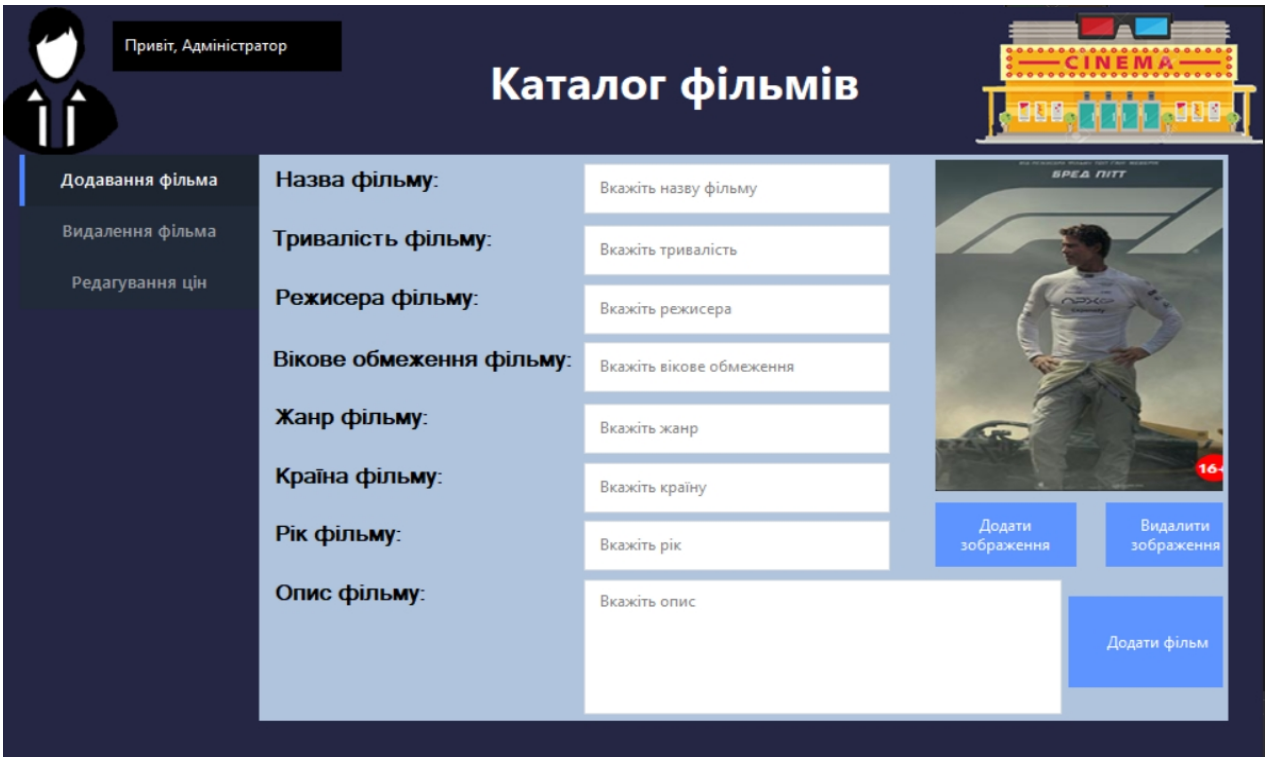


Рис. 4.3 Головна сторінка адміністратора, з інтерфейсом додавання фільмів

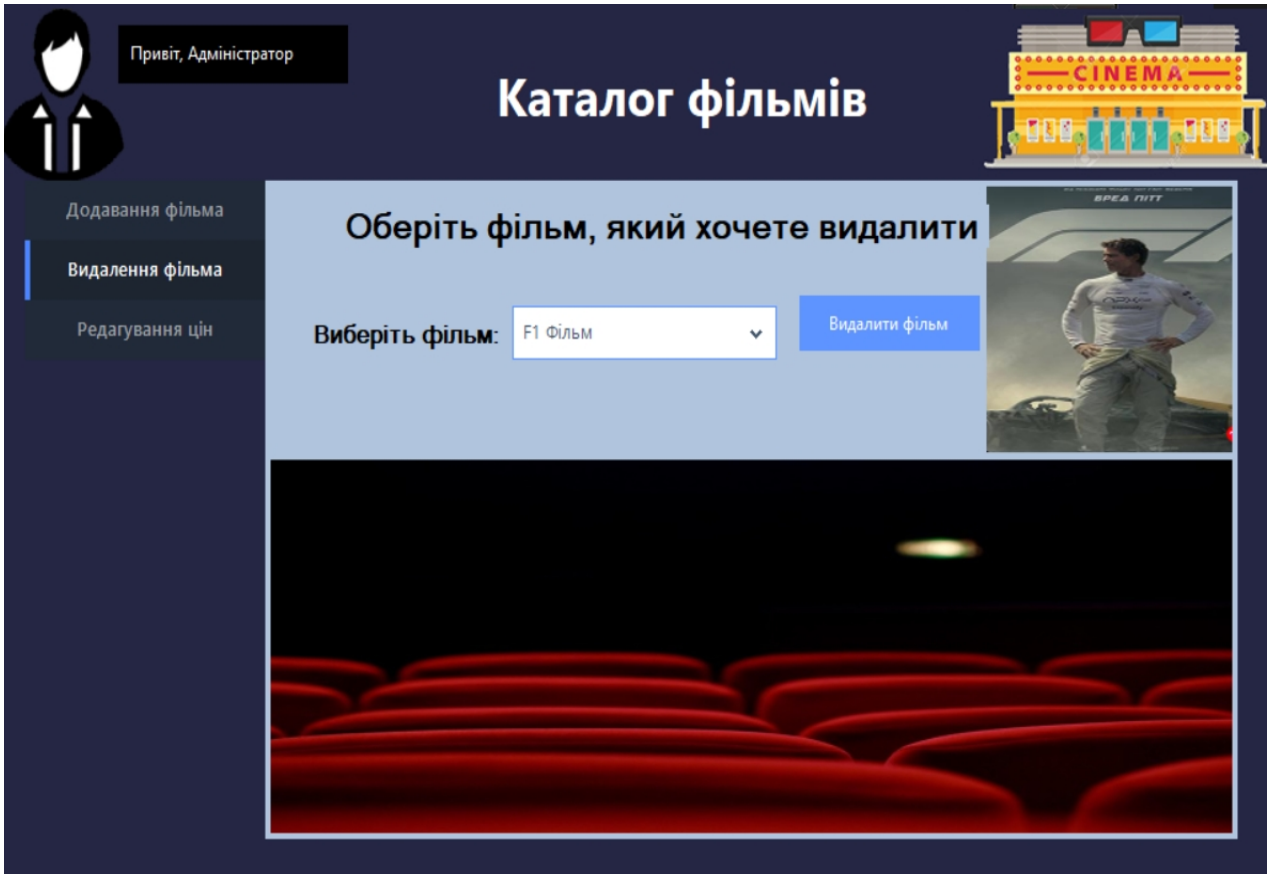


Рис. 4.4 Головна сторінка адміністратора, з інтерфейсом видалення фільмів

2. Головна сторінка касира та термінала самообслуговування

На головній сторінці касира та термінала самообслуговування знаходиться репертуар фільмів, які зараз знаходяться у прокаті, та елементи керування для пошуку фільму за назвою та можливість відсортувати фільми за жанром. Головна сторінка касира та термінала самообслуговування зображена на рис. 4.5.

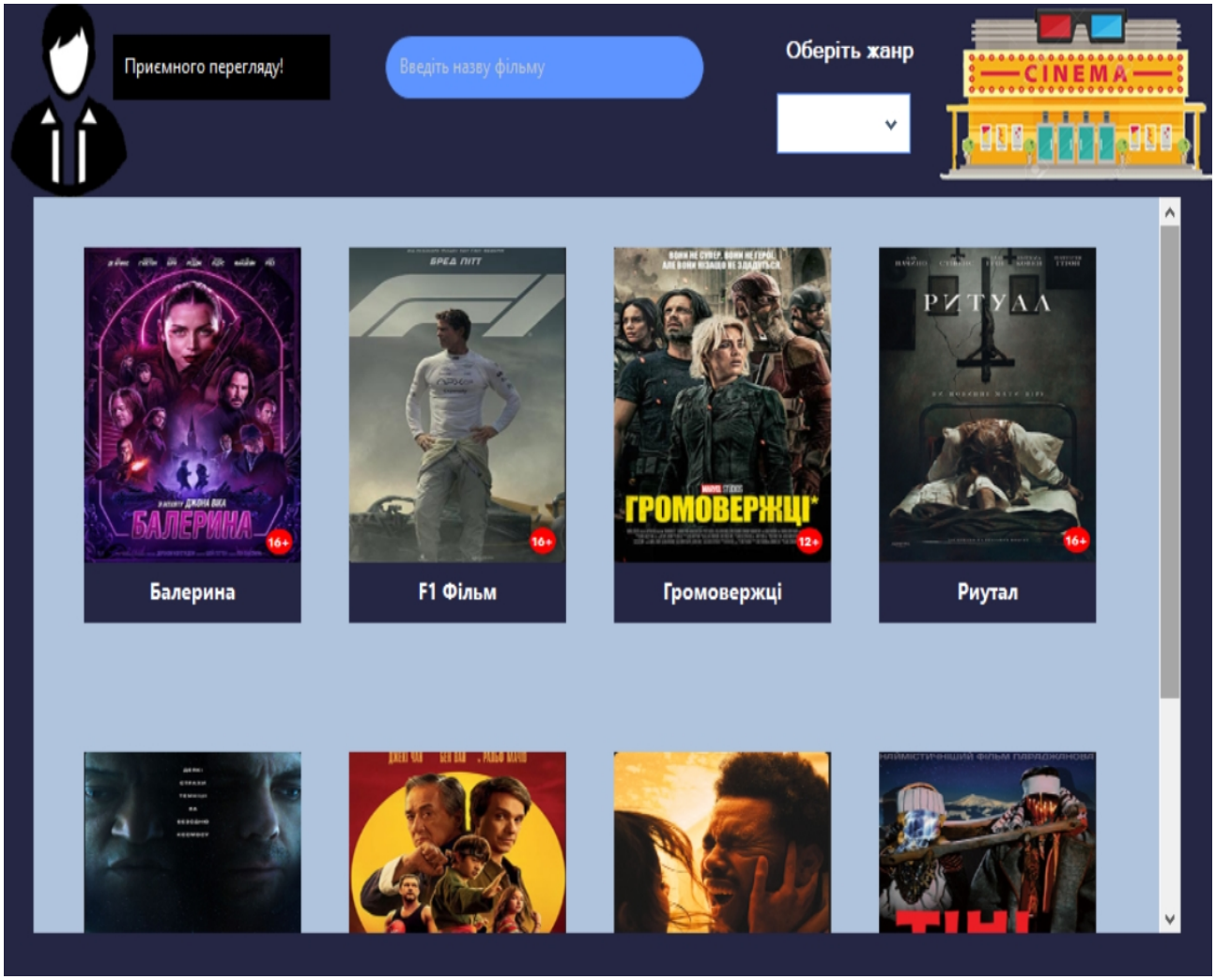


Рис. 4.5 Головна сторінка касира та термінала самообслуговування

4.4.3 Сторінка інформації про фільм. Сторінка інформації про фільм відкривається у користувача після вибору конкретного фільму з головного меню. На цій сторінці користувач може ознайомитися із загальною інформацією про фільм, включаючи назву, жанр, тривалість та інші характеристики. Крім того, представлено детальний опис сюжету, що допомагає краще зрозуміти зміст фільму. Також на цій сторінці доступний перелік сеансів, користувач може

переглянути розклад показів та обрати зручний час для перегляду. Цю сторінку продемонстровано на рис. 4.6.

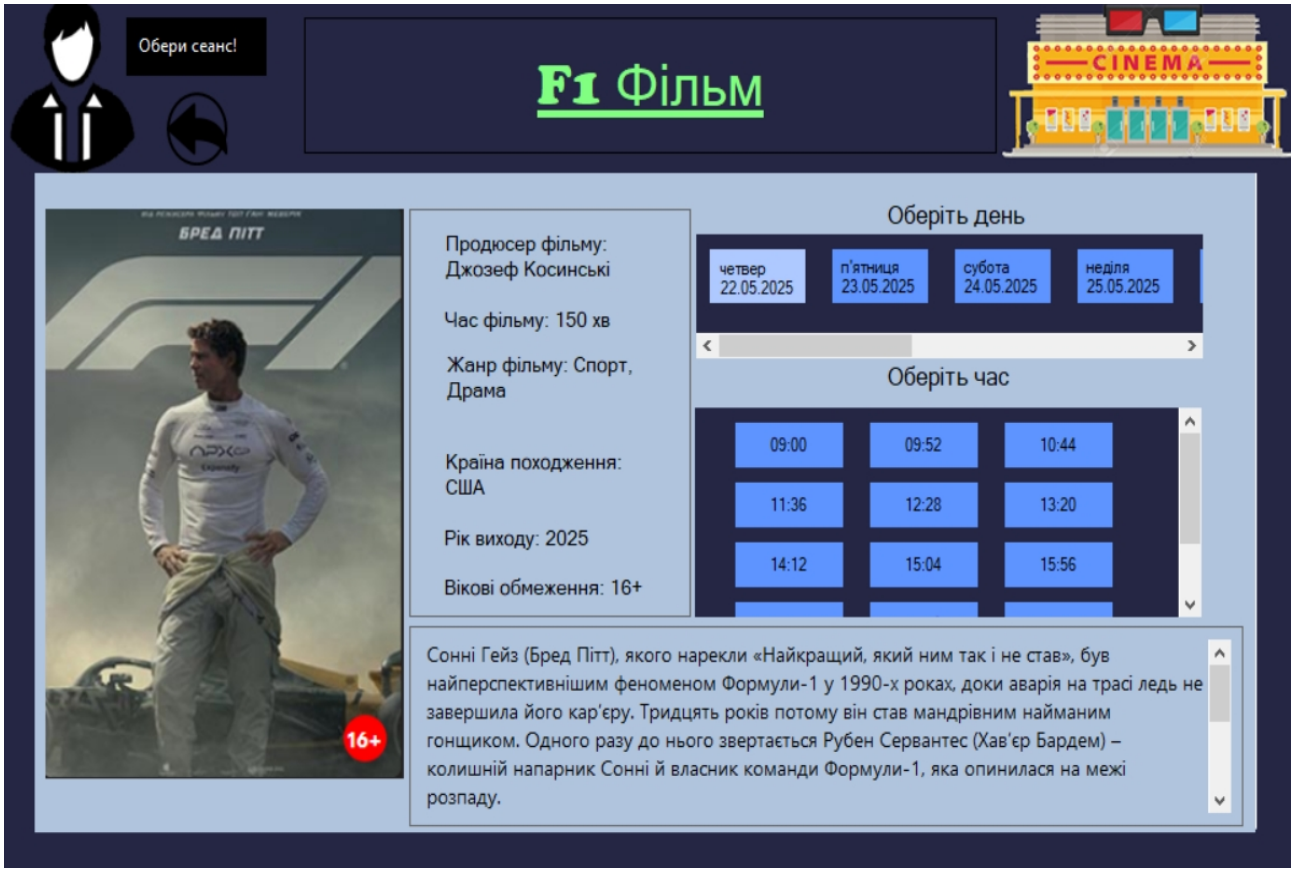


Рис. 4.6 Сторінка інформації про фільм

4.4.4 Сторінка вибору місця. Сторінка вибору місця відкривається після того, як користувач обрав фільм та відповідний сеанс. На цій сторінці відображається інформація про вибраний фільм, включаючи його назву, дату та час сеансу. Також представлена схема залу кінотеатру, де кольорами позначені різні типи місць:

- білі - звичайні місця;
- жовті - vip-місця з комфортнішими кріслами;
- червоні - зайняті місця, які недоступні для вибору;
- голубі - місця, які обрав користувач.

Збоку розташований список обраних місць із загальною сумою до оплати. Під списком розміщена кнопка, яка перенаправляє користувача до меню оплати для завершення бронювання. Такий інтерфейс забезпечує зручний вибір місць,

дозволяючи користувачу легко контролювати свій вибір перед оформленням замовлення.

Сторінка вибору місця продемонстрована на рис. 4.7.

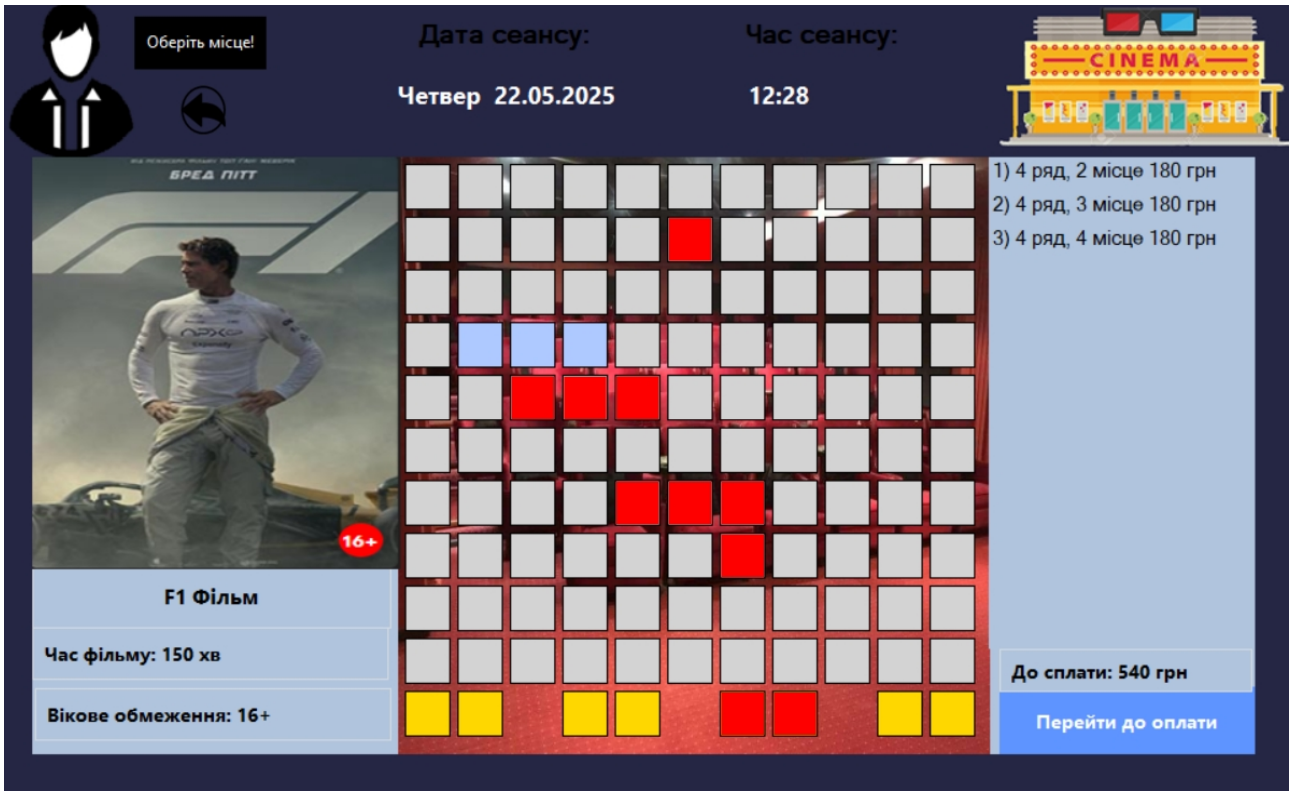


Рис. 4.7 Сторінка вибору місця

4.4.5 Сторінка вибору оплати. Після того як відвідувач обирає місце відкривається сторінка вибору оплати. На цій сторінці він ще раз може переглянути ключову інформацію: назву обраного фільму, дату та час сеансу, список вибраних місць і загальну суму до сплати. Далі користувач обирає зручний спосіб оплати:

- готівкова оплата;
- безготівкова оплата.

Після вибору способу оплати користувач має можливість обрати тип квитка:

- паперовий квиток;
- електронний квиток.

У разі вибору електронного квитка з’являється форма для введення контактної інформації. Користувач може вказати своє ім’я (за бажанням), а

також один із способів доставки квитка, номер телефону або електронну пошту. Можна ввести як один із варіантів, так і обидва, квиток буде надіслано за вказаними контактами відповідно до вибору користувача. Сторінку вибору оплати продемонстровано на рис. 4.8.

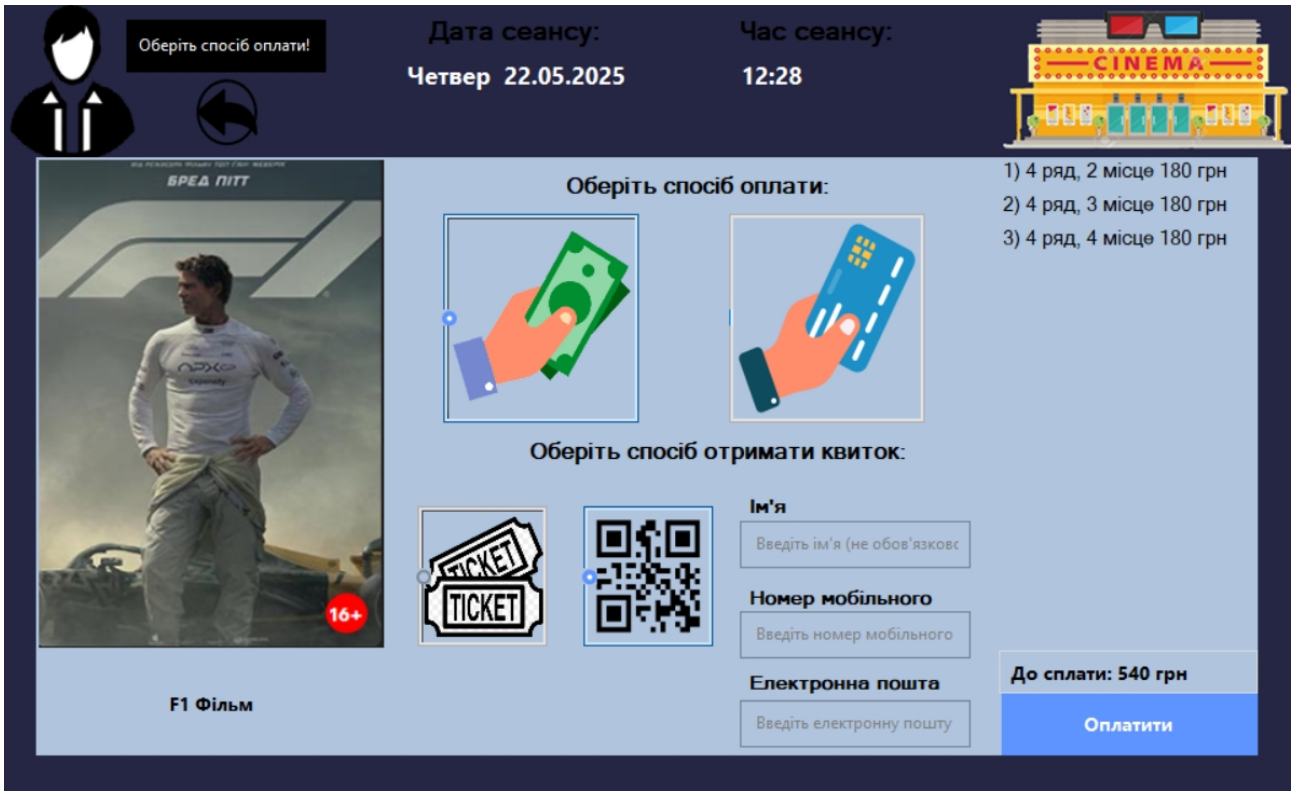


Рис. 4.8 Сторінка вибору оплати

4.4.6 Сторінка друку або надсилання квитка. Ця сторінка відкривається після завершення процесу оплати. На цій сторінці відвідувач бачить сформовані квитки з зазначенням усієї важливої інформації: назва фільму, дата та час сеансу, номер місця, тип оплати. Внизу сторінки розміщено кнопку, яка відповідає попередньому вибору користувача щодо типу квитка:

- **для паперового квитка** - кнопка «Друк», що дозволяє роздрукувати квиток безпосередньо зі сторінки;
- **для електронного квитка** - кнопка «Надіслати», яка забезпечує відправлення квитка на вказану електронну пошту або номер телефону користувача.

Ця сторінка є завершальним етапом у процесі оформлення квитка й забезпечує зручний доступ до нього в обраному форматі. Сторінку друку або надсилання квитка зображено на рис. 4.9, а приклад самого квитка на рис. 4.10.

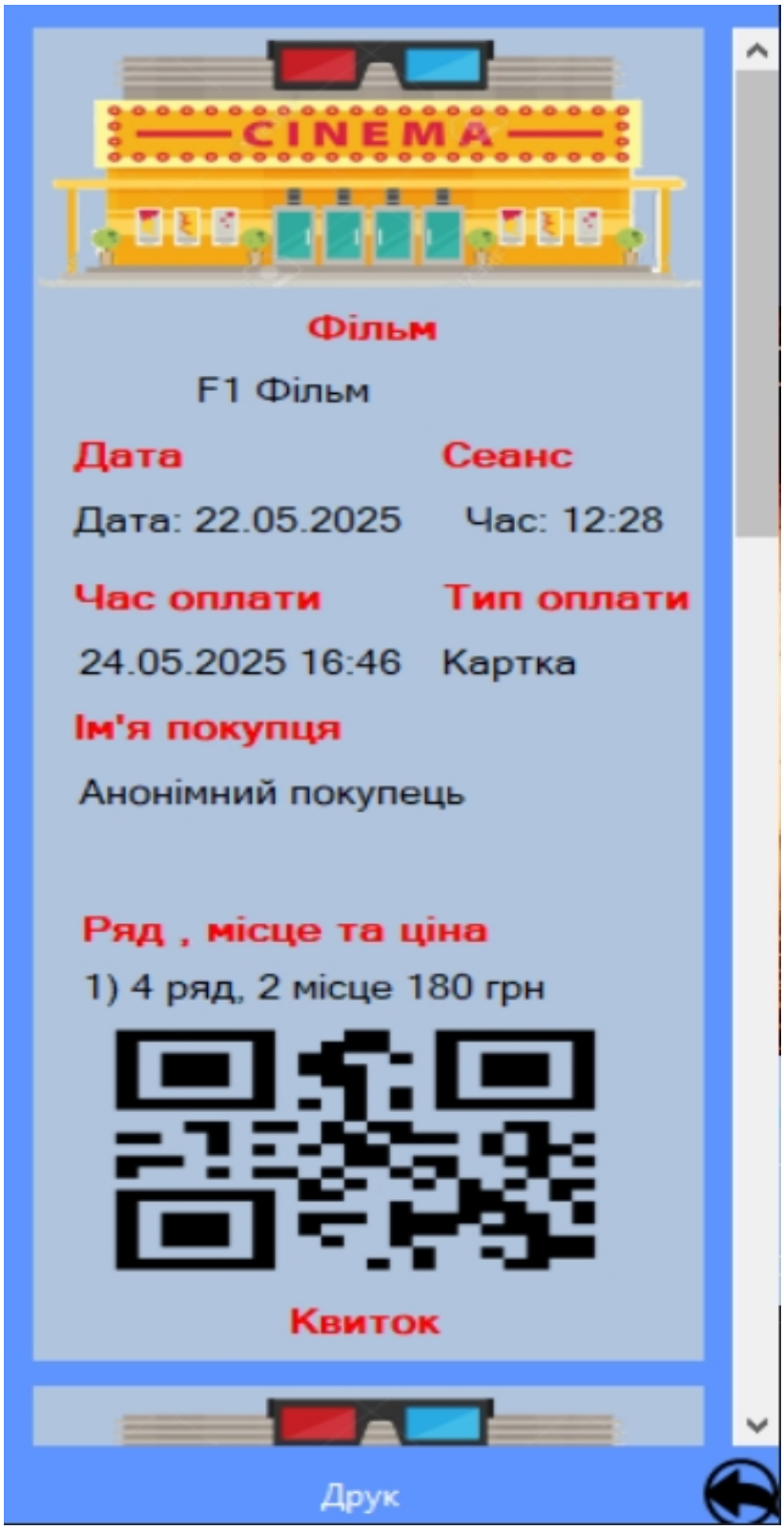


Рис. 4.9 Зовнішній вигляд сторінки друку або надсилання квитка

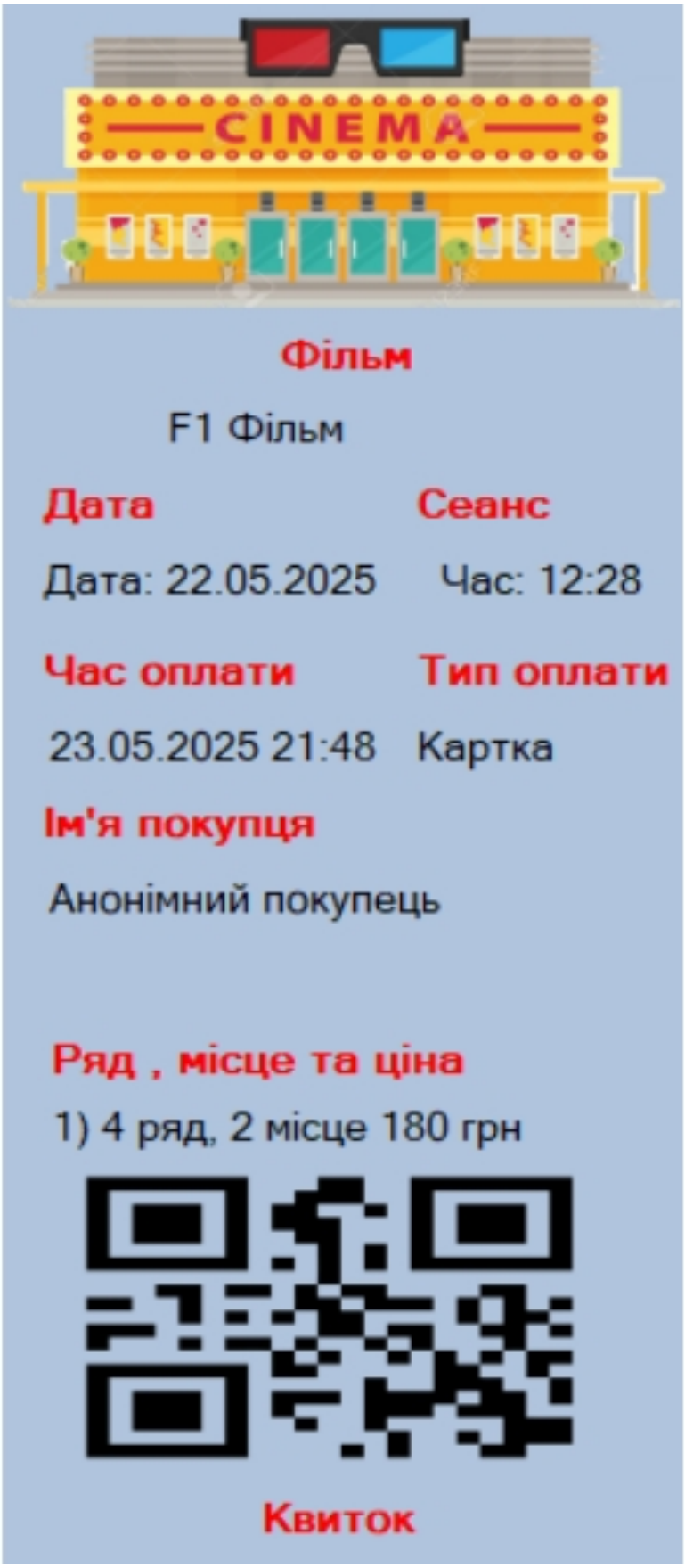


Рис. 4.10 Квиток

ВИСНОВКИ

У даній кваліфікаційній роботі було розроблено інформаційну систему для автоматизації процесу продажу квитків у кінотеатрі. Створена система забезпечує зручний інтерфейс для користувачів, дозволяючи відвідувачам переглядати на терміналі самообслуговування репертуар фільмів, обирати сеанси та місця, оформлювати квитки та здійснювати оплату, касирам проводити продаж фільмів. Для адміністратора реалізовано окремі модулі, що дозволяють керувати фільмами та цінової політикою.

Система була реалізована з використанням сучасних технологій, таких як C#, Windows Forms, Microsoft SQL Server, що забезпечило стабільну роботу додатку, зручність у користуванні, інтеграцію з базою даних та можливість масштабування. Windows Forms дозволив створити інтуїтивно зрозумілий графічний інтерфейс, а використання SQL Server дало змогу реалізувати ефективне збереження, обробку та фільтрацію даних. Особливу увагу приділено багаторівневій структурі доступу, де передбачені ролі відвідувача, касира та адміністратора.

У процесі розробки було застосовано модульний підхід, що підвищив гнучкість системи та полегшив її тестування. Усі основні модулі, включаючи модуль автентифікації, управління квитками, обробки платежів та взаємодії з платіжним терміналом, були ретельно протестовані. Для забезпечення високої якості продукту використовувались практики тестування, включаючи юніт-тести.

Розроблена система продажу квитків у кінотеатрі може бути ефективно впроваджена в реальне середовище, забезпечуючи автоматизацію бізнес-процесів, зменшення людського фактора, підвищення швидкості обслуговування клієнтів та покращення загального досвіду відвідувачів кінотеатру. Таким чином, розроблений програмний продукт має високу практичну цінність і може бути адаптований під конкретні потреби кінотеатрів різного масштабу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Система для клієнтського монітора SERVIO POS InfoDisplay. [Електронний ресурс] - Режим доступу: https://expertsolution.com.ua/produkty/pz/servio-pos-infodisplay-dlya-kliyentskogo-monitora (дата звернення 04.05.2025).

2. Бізнес інструмент для організації та продажу квитків TicketCRM. [Електронний ресурс] - Режим доступу: https://promo.ticketcrm.com/uk (дата звернення 05.05.2025).

3. Автоматизація квиткових процесів mticket. [Електронний ресурс] - Режим доступу: https://mticket.com.ua (дата звернення 06.05.2025).

4. UML для бізнес-моделювання. [Електронний ресурс] - Режим доступу: https://evergreens.com.ua/ua/articles/uml-diagrams.html (дата звернення 07.05.2025).

5. Як будувати UML-діаграми. Діаграма варіантів використання. [Електронний ресурс] - Режим доступу: https://dou.ua/forums/topic/40575/ (дата звернення 08.05.2025).

6. Діаграма послідовності. Опис діаграми послідовності. [Електронний ресурс] - Режим доступу: https://www.maxzosim.com/sequence-diagrams/ (дата звернення 09.05.2025).

7. Діаграма діяльності в UML. [Електронний ресурс] - Режим доступу: https://www.guru99.com/uk/uml-activity-diagram.html (дата звернення 10.05.2025).

8. Модель діаграми зв’язків сутностей (ER-діаграма). [Електронний ресурс] - Режим доступу: https://www.guru99.com/uk/er-diagram-tutorial-dbms.html (дата звернення 11.05.2025).

9. Нормальні форми бази даних. [Електронний ресурс] - Режим доступу: https://javarush.com/ua/quests/lectures/ua.questhibernate.level17.lecture02 (дата звернення 12.05.2025).

10. Що таке SQL Server?. [Електронний ресурс] - Режим доступу: https://www.guru99.com/uk/sql-server-introduction.html (дата звернення 13.05.2025).

11. Оператор CREATE DATABASE в SQL. [Електронний ресурс] - Режим доступу: https://acode.com.ua/create-database-sql/ (дата звернення 14.05.2025).

12. Оператор CREATE TABLE в SQL. [Електронний ресурс] - Режим доступу: https://acode.com.ua/create-table-sql/ (дата звернення 15.05.2025).

13. Первинний ключ (PRIMARY KEY) в SQL. [Електронний ресурс] - Режим доступу: https://acode.com.ua/primary-key-sql/ (дата звернення 16.05.2025).

14. Зовнішній ключ (FOREIGN KEY) в SQL. [Електронний ресурс] - Режим доступу: https://acode.com.ua/foreign-key-sql/ (дата звернення 17.05.2025).

15. Збережені процедури в SQL. [Електронний ресурс] - Режим доступу: https://acode.com.ua/stored-procedures-sql/ (дата звернення 18.05.2025).

16. Що таке тригер бази даних. [Електронний ресурс] - Режим доступу: https://kovelpost.com/blogs/11698 (дата звернення 19.05.2025).

17. Представлення в SQL. [Електронний ресурс] - Режим доступу: https://acode.com.ua/view-sql/ (дата звернення 20.05.2025).

18. Абстракція в UML. [Електронний ресурс] - Режим доступу: https://pitchfork.kultura.cx.ua/maysternist/yak-abstrakciya-viznachaietsya-v-uml.html (дата звернення 21.05.2025).

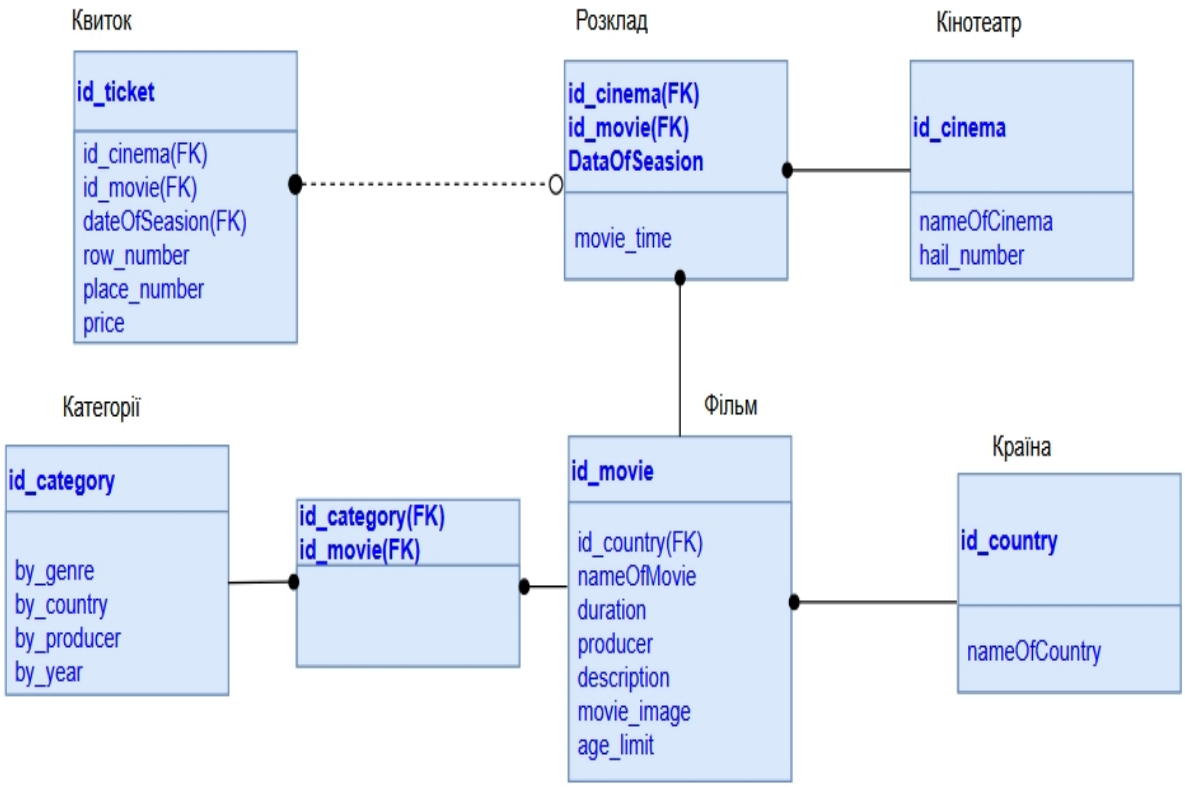
19. Діаграма класів UML. [Електронний ресурс] - Режим доступу: https://www.mindonmap.com/uk/blog/what-is-uml-class-diagram/ (дата звернення 22.05.2025).

20. Що таке діаграма UML. Діаграма упаковки. [Електронний ресурс] - Режим доступу: https://www.mindonmap.com/uk/blog/what-is-uml-diagram/ (дата звернення 23.05.2025).

21. Діаграма розгортання. [Електронний ресурс] - Режим доступу:https://www.guru99.com/uk/deployment-diagram-uml-example.html (дата звернення 24.05.2025)

ДОДАТОК А

ЛОГІЧНА МОДЕЛЬ СИСТЕМИ



ДОДАТОК Б

КОД НАДАННЯ ПРАВ ДОСТУПУ ВІДПОВІДНО ДО РОЛЕЙ

```
--права доступу для адміністратора
grant select, insert, delete, update on Cinema to admincinema
grant select, insert, delete, update on Country to admincinema
grant select, insert, delete, update on Movie to admincinema
grant select, insert, delete, update on Schedule to admincinema
grant select, insert, delete, update on Ticket to admincinema
grant select, insert, delete, update on Category to admincinema
grant select, insert, delete, update on MovieCategory to admincinema

--права доступу для касира
grant select, insert, update on Cinema to cashiercinema
grant select, insert, update on Country to cashiercinema
grant select, insert, update on Movie to cashiercinema
grant select, insert, update on Schedule to cashiercinema
grant select, insert, update on Ticket to cashiercinema
grant select, insert, update on Category to cashiercinema
grant select, insert, update on MovieCategory to cashiercinema

--права доступу для відвідувача
grant select on Cinema to usercinema
grant select on Country to usercinema
grant select on Movie to usercinema
grant select on Schedule to usercinema
grant select on Ticket to usercinema
grant select on Category to usercinema
grant select on MovieCategory to usercinema
```

ДОДАТОК В

КОД СТВОРЕННЯ ПРОЦЕДУРИ

```
--процедура додавання нового фільму
CREATE PROCEDURE AddNewMovie
    @id_country INT = NULL,
    @nameOfMovie NVARCHAR(255),
    @duration INT,
    @producer NVARCHAR(255) = NULL,
    @description NVARCHAR(MAX) = NULL,
    @movie_image NVARCHAR(255) = NULL,
    @age_limit INT = NULL
AS
BEGIN
    INSERT INTO Movie (
        id_country,
        nameOfMovie,
        duration,
        producer,
        description,
        movie_image,
        age_limit
    )
    VALUES (
        @id_country,
        @nameOfMovie,
        @duration,
        @producer,
        @description,
        @movie_image,
        @age_limit
    );
END;
```

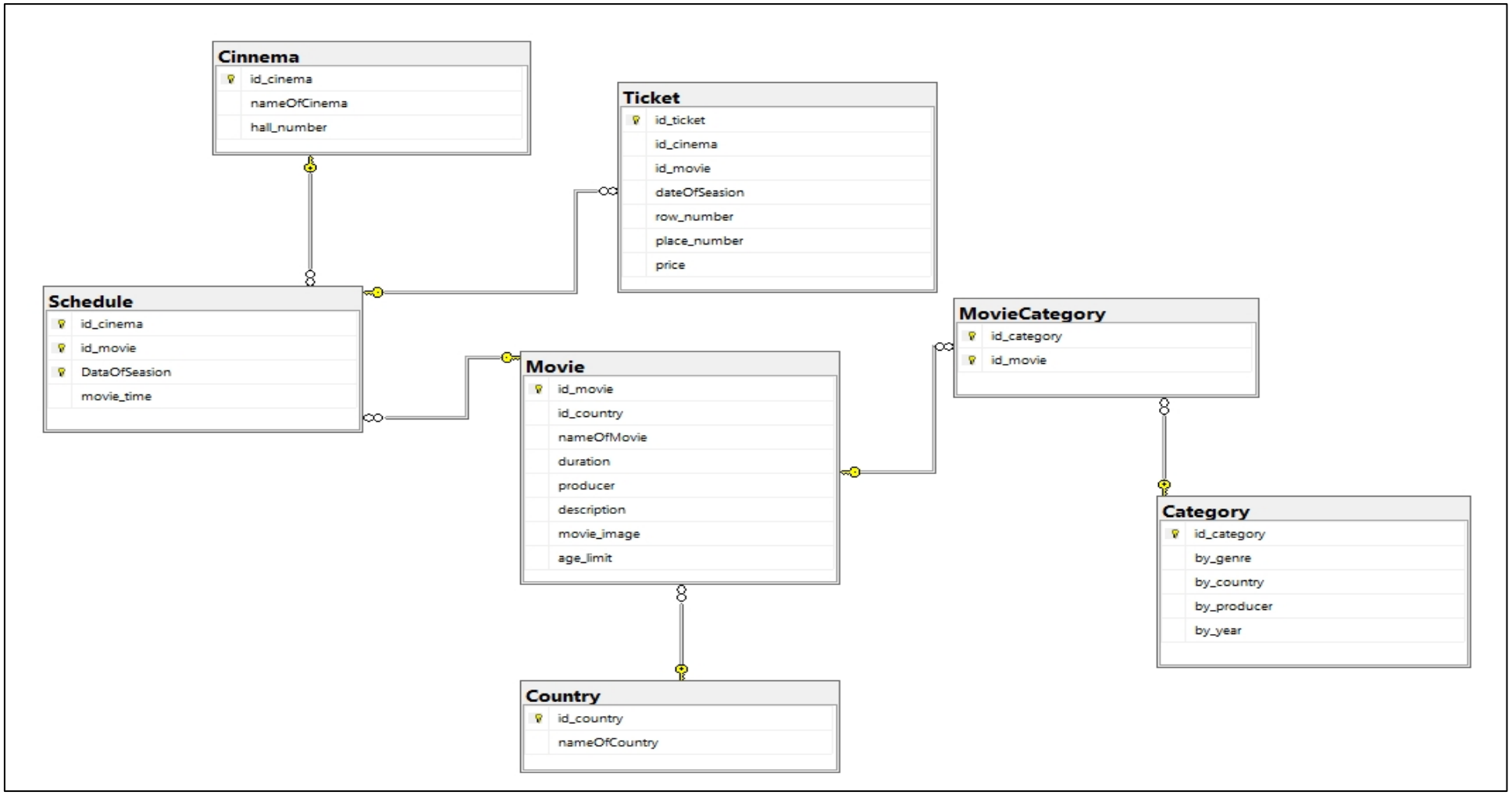
ДОДАТОК Д

КОД СТВОРЕННЯ ТРИГЕРА

```
--тригер заборона дублювання місць на один сеанс
CREATE TRIGGER trg_PreventDuplicateTicket
ON Ticket
INSTEAD OF INSERT
AS
BEGIN
    IF EXISTS (
        SELECT 1
        FROM Ticket t
        JOIN inserted i ON
            t.id_cinema = i.id_cinema AND
            t.id_movie = i.id_movie AND
            t.dateOfSeasion = i.dateOfSeasion AND
            t.row_number = i.row_number AND
            t.place_number = i.place_number
    )
    BEGIN
        RAISERROR('Таке місце вже зайняте на вказаний сеанс.', 16, 1);
    END
    ELSE
    BEGIN
        INSERT INTO Ticket (id_cinema, id_movie, dateOfSeasion, row_number, place_number, price)
        SELECT id_cinema, id_movie, dateOfSeasion, row_number, place_number, price
        FROM inserted;
    END
END
```

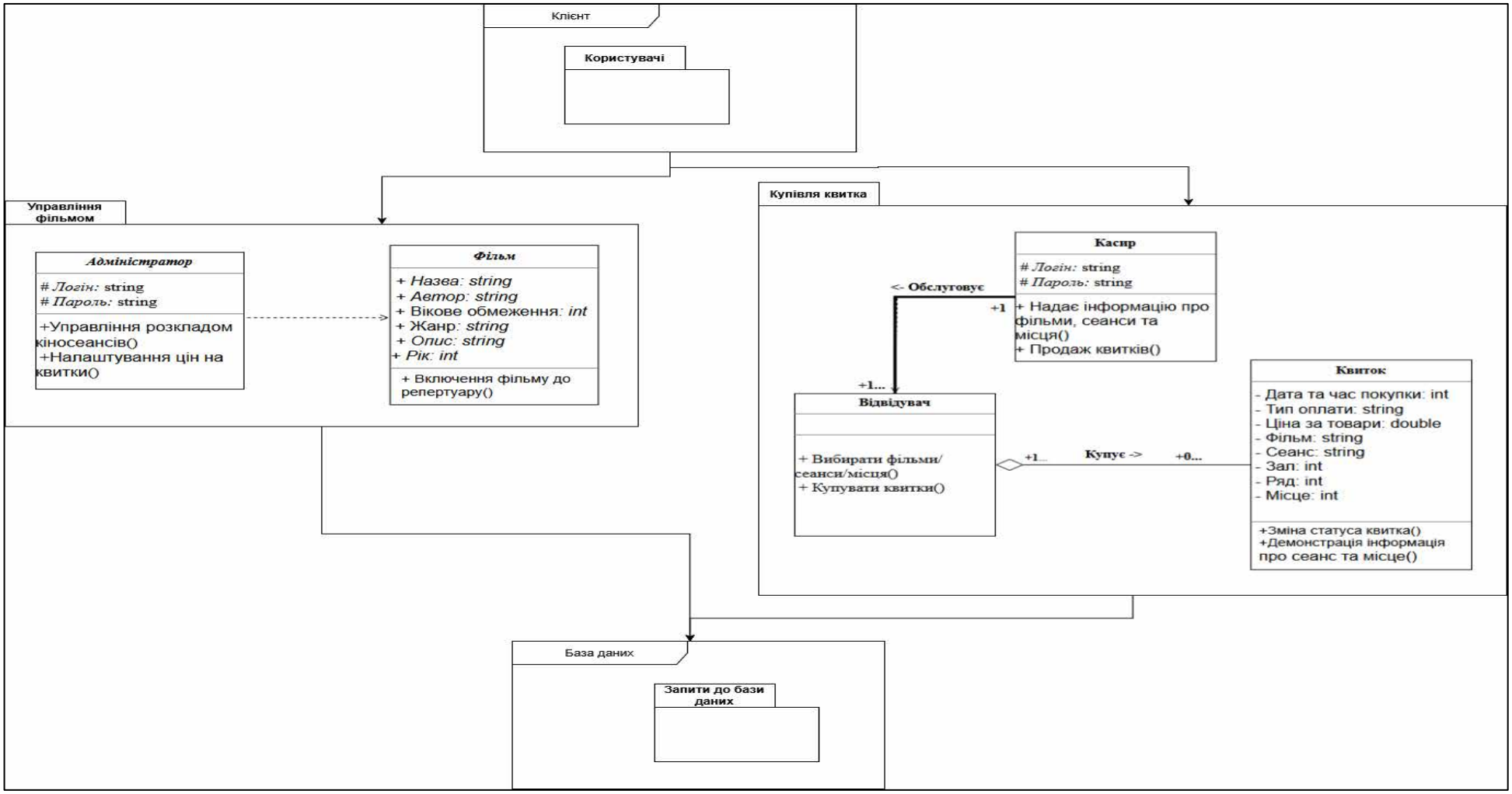
ДОДАТОК Е

БАЗА ДАНИХ У ВИГЛЯДІ ДІАГРАМИ



ДОДАТОК Ж

ДІАГРАМА ПАКЕТІВ



ДОДАТОК К

КОД РЕАЛІЗАЦІЇ АВТОРИЗАЦІЯ КОРИСТУВАЧІВ ІЗ РОЗМЕЖУВАННЯМ ДОСТУПУ ЗА РОЛЯМИ

```
private void authentication()
{
    string login = textBox_login.Text;
    string pass = textBox_pass.Text;

    CurrentUser currentUser = new CurrentUser(login, pass);

    if (currentUser.DataBase.IsConnected())
    {
        string role = currentUser.DataBase.GetUserRole(login);

        if (role == null)
        {
            MessageBox.Show("Не вдалося визначити роль користувача.");
            return;
        }

        currentUser.Role = role;

        this.Hide();

        if (role.ToLower() == "admin")
        {
            new AdminForm(currentUser).ShowDialog();
        }
        else
        {
            new MainForm(currentUser).ShowDialog();
        }
    }
    else
    {
        MessageBox.Show("Невірно введені дані!");
    }
}
```

ДОДАТОК Л

КОД РЕАЛІЗАЦІЇ ДОДАВАННЯ ФІЛЬМУ

```
private void MovieAdd()
{
    try
    {
        currentUser.DataBase.OpenConnection();
        SqlConnection conn = currentUser.DataBase.GetConnection();
        string countryName = CountryBox.Text.Trim();
        int countryId = -1;
        string checkCountryQuery = "SELECT id_country FROM Country WHERE nameOfCountry = @Name";
        using (SqlCommand checkCmd = new SqlCommand(checkCountryQuery, conn))
        {
            checkCmd.Parameters.AddWithValue("@Name", countryName);
            object result = checkCmd.ExecuteScalar();

            if (result != null)
            {
                countryId = Convert.ToInt32(result);
            }
            else
            {
                string insertCountryQuery = "INSERT INTO Country (nameOfCountry) OUTPUT INSERTED.id_country VALUES (@Name)";
                using (SqlCommand insertCmd = new SqlCommand(insertCountryQuery, conn))
                {
                    insertCmd.Parameters.AddWithValue("@Name", countryName);
                    countryId = (int)insertCmd.ExecuteScalar();
                }
            }
        }
        string movieQuery = @"INSERT INTO Movie
        (id_country, nameOfMovie, duration, producer, description, movie_image, year_movie, age_limit, genre)
        VALUES (@CountryId, @Name, @Duration, @Producer, @Description, @ImagePath, @Year, @AgeLimit, @Genre)";
        using (SqlCommand cmd = new SqlCommand(movieQuery, conn))
        {
            cmd.Parameters.AddWithValue("@CountryId", countryId);
            cmd.Parameters.AddWithValue("@Name", NameBox.Text.Trim());
            cmd.Parameters.AddWithValue("@Duration", int.Parse(MovieTimeBox.Text.Trim()));
            cmd.Parameters.AddWithValue("@Producer", ProducerBox.Text.Trim());
            cmd.Parameters.AddWithValue("@Description", DescriptionBox.Text.Trim());
            cmd.Parameters.AddWithValue("@ImagePath", MoviePictureBox.ImageLocation ?? "");
            cmd.Parameters.AddWithValue("@Year", int.Parse(YearBox.Text.Trim()));
            cmd.Parameters.AddWithValue("@AgeLimit", int.Parse(AgeLimitBox.Text.Trim()));
            cmd.Parameters.AddWithValue("@Genre", GenreBox.Text.Trim());

            cmd.ExecuteNonQuery();

    finally
    {
        currentUser.DataBase.CloseConnection();
    }
}
```

ДОДАТОК М

КОД ВІДОБРАЖЕННЯ ФІЛЬМІВ

```
}
private void LoadMovies()
{
    allMovies = MovieModule.LoadMoviesFromDatabase(currentUser.DataBase);
    DisplayMovies(allMovies);

}

private void DisplayMovies(List<Movie> movies)
{
    MoviePanel.Controls.Clear();

    int columnCount = 4;
    int cardWidth = 200;
    int cardHeight = 300;
    int padding = 10;

    int marginLeft = 40;
    int marginTop = 30;

    for (int i = 0; i < movies.Count; i++)
    {
        var movie = movies[i];
        MovieCard card = new MovieCard();
        card.SetMovie(movie);
        card.SetDataBase(currentUser.DataBase);
        card.SetMainForm(this);
        card.Width = cardWidth;
        card.Height = cardHeight;


        int row = i / columnCount;
        int col = i % columnCount;
        int verticalSpacing = 1;
        card.Location = new Point(
            marginLeft + col * (cardWidth + padding),
            marginTop + row * (cardHeight + verticalSpacing)
        );
        MoviePanel.Controls.Add(card);
    }
}
```

ДОДАТОК Н

КОД ВІДОБРАЖЕННЯ СЕАНСІВ

```
private void LoadAndGenerateDayButtons()
{
    movieSchedules = ScheduleModule.LoadSchedulesForMovie(db, movie.Id);

    var uniqueDates = movieSchedules
        .Select(s => s.Date.Date)
        .Distinct()
        .OrderBy(d => d)
        .ToList();
    daypanel.Controls.Clear();

    int left = 10;
    int top = 10;

    foreach (var date in uniqueDates)
    {
        RadioButton rb = new RadioButton
        {
            Text = date.ToString("dddd", new System.Globalization.CultureInfo("uk-UA")) + "\n" + date.ToString("dd.MM.yyyy"),
            Tag = date,
            AutoSize = true,
            Top = top,
            Left = left,
             Appearance = Appearance.Button,
             BackColor = Color.FromArgb(94, 148, 255),
            ForeColor = Color.Black,
            FlatStyle = FlatStyle.Flat,

    };
        rb.CheckedChanged += DayRadioButton_CheckedChanged;
        rb.FlatAppearance.BorderSize = 0;
        daypanel.Controls.Add(rb);
        left += rb.PreferredSize.Width + 20;
    }
    if (daypanel.Controls.Count > 0 && daypanel.Controls[0] is RadioButton firstButton)
    {
        firstButton.Checked = true;
    }
}

private void DayRadioButton_CheckedChanged(object sender, EventArgs e)
{
    if (sender is RadioButton rb && rb.Checked)
    {
        DateTime selectedDate = (DateTime)rb.Tag;
        GenerateTimeButtons(selectedDate);
    }
}
```