

КУРСОВИЙ ПРОЕКТ (РОБОТА)

з **Рационального уніфікованого процесу проектування програмного забезпечення**

(назва дисципліни)

на тему: **Система автоматизації роботи касира театру**

Студента (ки) 5 курсу, групи СПм-51
напряму підготовки

спеціальності 121 Інженерія
програмного забезпечення

Біланіка З. Б.

(прізвище та ініціали)

Керівник: Мудрик І. Я.

(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Оцінка за національною
шкалою

Кількість балів: Оцінка ECTS

Члени
комісії:

(підпис)

(прізвище та ініціали)

(підпис)

(прізвище та ініціали)

(підпис)

(прізвище та ініціали)

ЗМІСТ

ВСТУП.....	4
1 ПОСТАНОВКА ПРОБЛЕМИ	6
1.1. Загальний опис тематики.....	6
1.2. Огляд існуючих програмних систем	7
1.2.1 SERVIO Tickets	7
1.2.2 CashFront.	9
1.3 Аналіз вимог до програмного забезпечення	11
2 ІНСТРУМЕНТИ РЕАЛІЗАЦІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	12
2.1 Опис мови програмування C#.....	13
2.2 Опис середовища програмування Visual Studio 2019.....	14
2.3 Фреймворк для створення системних зв'язків. Entity Framework	15
2.4 Механізм взаємодії з базою даних MySQL Server.....	18
3 ПРОЕКТУВАННЯ ПРОГРАМНОГО ПРОДУКТУ	20
3.1 Вибір моделі розробки.....	20
3.2. Проектування вимог та варіантів використання програмного забезпечення	23
3.3 Організація класів	25
3.4 Метамоделі системи та опис системи класів.....	27
3.5 Проектування бази даних	36
4 ТЕСТУВАННЯ ПРОГРАМНОГО ПРОДУКТУ	40
ВИСНОВКИ.....	46
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	47
Додатки.....	48
Додаток А. Код програмного продукту	48
Додаток Б. Скрипти створення таблиць бази даних.....	49
Додаток В. Технічне завдання	51
Додаток Г. Рецензія.....	52

ВСТУП

Людина в своїй діяльності прагне оптимізувати будь-яку свою роботу, створити масиви інформації для ефективного виконання завдань та передбачити механізми контролю та використання інформації в робочому процесі. Інформаційні автоматизовані системи найкраще підходять для виконання поставлених завдань та забезпечуватимуть стабільну роботу будь-якої компанії, підтримуватимуть облік активів та контроль їх використання.

Для будь-якої компанії поширеною практикою є впровадження в робочий цикл систем та механізмів для контролю, оптимізації виконання завдань та спрощення виконання деяких дій через приведення їх реалізацій до електронного варіанту що призведе до покращення рівня якості виконання роботи. Найбільше така практика розповсюджена серед компаній та організацій що пов'язані зі сферою послуг. З плином часу кількість таких структур збільшується, вони переводять свою адміністративну діяльність в програмну сферу що спричиняє покращення організаційних рішень та дозволяє розширити асортимент інструментів для виконання організаційно-адміністративних завдань.

Метою проекту є розробка системи для автоматизації роботи касира організації зв'язаної чи відповідальної за адміністрацію та організацію робочого процесу театру. Безпосередньо передбачається виконання системою проведення обслуговування клієнтів організації та аналізу касового збору через створення системи для впровадження механізму автоматизації в ці процеси. Програма забезпечуватиме облік усіх дій касира та надаватиме можливості для адміністратора долучатися до робочих інструментів та документації що є звітом з описом виконаних робіт працівником. Адміністративна частина також додаватиме можливість зберігати та архівувати дані від клієнтів, додавати, корегувати, і видаляти розклад сеансів, репертуар театральної трупи та підтримувати грамотність графіку попередньо визначених вистав через оновлюваність даних в системі.

Завданням даного проекту є розробка системи що неситиме інформаційно-адміністративно-організаційні обов'язки для обслуговування театрів. Система повинна забезпечувати гнучкість для легкого її впровадження в театри будь-якого типу та рівня, від місцевих, аматорських до регіональних, міжнародних та інші. Як компоненти системи, можна відзначити застосунок, що забезпечує авторизацію користувача, збереження та обробку інформації, редагування даних системи. Результатом роботи є цілісна система, яку можна використовувати на підприємстві спрямованих на облік клієнтів, та забезпечення найоптимальнішого процесу обслуговування відвідувачів.

Процес розробки додатку поділений на декілька частин, серед яких можна виділити наступні: реалізація графічного інтерфейсу користувача, розробка логіки програми та проектування бази даних. Для впровадження графічного інтерфейсу було обрано систему для програмування інтерфейсів Windows Forms, який є частиною Microsoft .NET Framework, програмну частину було реалізовано з використанням мови програмування C#. Для зберігання даних було обрано реляційну БД 8.0.20 MySQL Community Server.

В якості програми для розробки вибрана система автоматизації роботи касира театру.

1 ПОСТАНОВКА ПРОБЛЕМИ

1.1 Загальний опис тематики

Сфера обслуговування в своїй суті, покликана для проведення дій чи професій для задоволення потреб клієнта його бажань та вимог. Обслуговування передбачає наявність в персоналу організації необхідного мінімального досвіду для того, щоб клієнт залишився задоволений представленими послугам та якістю їх виконання.

Від якості обслуговування залежить успішність компанії та лояльність клієнтів, які можуть в майбутньому неодноразово звертатися за послугами в організацію. Погане обслуговування може призвести до скарг, зниження репутації та втрати користувачів, що є наслідком переорієнтації клієнтів на послуги конкурентів в пошуку кращого обслуговування. В свою чергу, якісне обслуговування сприяють побудові зв'язків між споживачем та організацією, що призводить до збільшення репутації та клієнтів які зацікавлені в послугах компанії, тим самим підвищує прибуток.

Щоб забезпечити якісний та швидкий сервіс для клієнта, підприємствам необхідно реалізовувати шляхи покращення своєї компанії або мережі, що надасть поштовх для розвитку організації в цілому. До таких покращень можна віднести впровадження комп'ютиризації в роботу підприємства. Через покращення пристроїв та функцій і забезпечення автоматизації компанія може позбутися методів які призводять до помилок які спричинятимуть падіння якості роботи.

Інформатизація театру надасть можливість пришвидшити операції що спричиняють затримку обслуговування клієнтів, та ведення ручного обліку, який не є економічно та ресурсно вигідним для роботи компанії. Звідси можна судити про необхідність створення системи для театрів, що виконуватимуть функції обслуговування які б могли оптимізувати ресурси та автоматизувати більшість його компонентів.

1.2 Огляд існуючих програмних систем

Для зайнятих у цій галузі осіб існують різноманітні програмні рішення з різними недоліками та перевагами. Одні підійдуть для компаній з централізованим управлінням, інші для тих хто прагне лише частково інформатизувати робочий процес. Тому перед проектуванням і реалізацією власного програмного рішення необхідно проаналізувати аналоги та визначити основний функціонал, що стане основою для розроблюваної системи. Далі розглянемо деякі з них.

1.2.1 SERVIO Tickets

SERVIO Tickets – це програмне забезпечення, створене для кінотеатрів, театрів, музеїв, опер, концерт-холів та інших розважальних закладів, діяльність яких пов'язана з реалізацією квитків. Вона складається з комплексу програмних інструментів, які допомагають контролювати продаж квитків з повною видачею аналітичної та статистичної звітності[1].

До складу програми автоматизації належать валідатор та інформаційна панель для відвідувачів для того, щоб значно спростити та прискорити процес продажу квитків.

Система продажу квитків може використовуватися для керування як окремими закладами, так і мережею мультиплексів.

Використання квиткової системи для кінотеатрів, музеїв, театрів, опер дає змогу уникнути черг перед сеансом у пікові години та створити комфортні умови для відвідувачів, за яких вони зможуть самостійно обрати сеанс, вільне місце та скористатися іншими додатковими послугами, на які за статистикою припадає до 50-70% від доходу закладу[1].

Завдяки продуманному функціоналу система продажу квитків SERVIO Tickets може використовуватися для вирішення широкого спектру завдань:

- Створення гнучкої системи моніторингу сеансів, розкладу фільмів, вистав, концертів.

- Забезпечення швидкого доступу касира до інформації про фільм чи спектакль, тривалість сеансів, вікові обмеження.
- Зручне відображення вільних/зайнятих місць у залах, їхнє бронювання.
- Забезпечення відвідувачів інформацією про кількість доступних місць у залі.
- Синхронізація системи продажу електронних квитків із сайтом.
- Відображення інформації про сеанси на ТВ-панелях, встановлених в прикасовій зоні та зоні відпочинку. Це допомагає демонструвати інформацію про вільні місця відвідувачам, які очікують обслуговування в черзі.
- Забезпечення роботи мобільного додатка-валідатора, який зчитує штрих-коди квитків для проходу відвідувачів до зали.
- Автоматичне завантаження даних у міжнародну аналітичну систему ComScore.
- Підготовка аналітичної звітності з продажу квитків і реалізації додаткових послуг, популярності тих чи інших фільмів чи вистав, моніторинг відвідувачів у різних сегментах (віковому, гендерному тощо).

Програма SERVIO Tickets може використовуватися для автоматизації продажу квитків у кінотеатрах, театрах, музеях, стадіонах, операх, парках культури та відпочинку. Використання квиткової системи для музею чи іншого розважального закладу допомагає покращити якість обслуговування клієнтів і підвищити їхню лояльність.[1]



Рисунок 1.2.1 – Зображення вигляду програми SERVIO Tickets

Підтримка: Веб-браузер.

Сайт програми: <https://expertsolution.com.ua/uk/avtomatizacija-kinoteatrov--teatrov--operi/>.

Український інтерфейс: ні.

Умови розповсюдження: cashware.

1.2.2 CashFront.

Програмний продукт «Cashfront» це сучасна Pos-Система, призначена для автоматизації операцій продажу товарів з використанням штрих- кодування, з швидким пошуком артикула по внутрішньому коду, або найменуванню. Продукт розроблений для автоматизації процесу торгівлі в реальному часі, при цьому акцент зроблений на максимальну зручність роботи користувачів[2].

Фіскальних і не фіскальних (ESC/POS- принтери, чекові принтери й фіскальні реєстратори, сканери штрих-коду з RS232 інтерфейсом) дасть можливість підібрати оптимальний варіант додаткового обладнання, максимально використовувати пристрої, що є в наявності. Касова програма, не пред'являє ні яких, специфічних вимог до комп'ютера й не вимагає установки

додаткових яких-небудь компонентів, типу 1С або SQL, що робить процес установки й експлуатацію Cashfront зрозумілим і доступним.

Автономна робота забезпечить необхідний рівень автоматизації торговельних точок, з використанням усіх сучасних віянь у цій сфері: організацією дисконтного клубу, обліком торговельних операцій і контролем залишків товару. Cashfront дозволяє поєднувати й оперативно підключати додаткові робочі місця в одну систему, не перериваючи при цьому процес торгівлі. Фіскального реєстратора й POS – принтера дасть можливість виконувати продаж на одному робочому місці від різних підприємств, наприклад, фірма платник ПДВ(фіскальний реєстратор) і приватного підприємця (чековий принтер). Залежно від ознаки артикула продажі фіксуються в різних реєстраторах. Це дозволяє значно заощадити на обладнанні, раціонально використовувати площу торговельного залу, і зменшити витрати на утримання персоналу[2].

Це програмне забезпечення, яке працює без зависань, збоїв і перезавантажень. Відсутність складних механізмів звертання до даних і продуктивна платформа Microsoft .NET Framework забезпечують високу якість для Cashfront. Використання прямого вводу-виводу даних на жорсткий диск дозволяють зробити операції транзакцій практично миттєвими, що з однієї сторони забезпечує високу швидкість програмного забезпечення в реальному часі, і цілісність даних у випадку непередбачених ситуацій (наприклад, збій у мережі, відключення живлення і т.д.), з іншої сторони. Обробка зовнішніх OLE/Activex- об'єктів (модулі підключення торговельного обладнання) в окремому, ізольованому процесі гарантують стабільність роботи програми у випадку виникнення помилок у цих об'єктах[2].

Вбудовані механізми конфігурування дозволяють обмежити доступ до функцій ОС (операційної системи), захищаючи систему від стороннього втручання й надаючи можливість налаштування системи- адміністраторам (вхід адміністратора захищений паролем). Обмежений доступ до ОС має на увазі завантаження програми Cashfront у якості оболонки, що виконується, Windows

при включенні комп'ютера й вимикання комп'ютера при виході із програми. Тільки адміністратор може дозволити/заборонити введення знижок, видалення рядків із чеків, змінювати ціни на товар, включати/виключати інші функції програми. Для забезпечення запобігання несумлінної роботи персоналу передбачена фіксація всіх операцій у спеціальному лог-файлі.

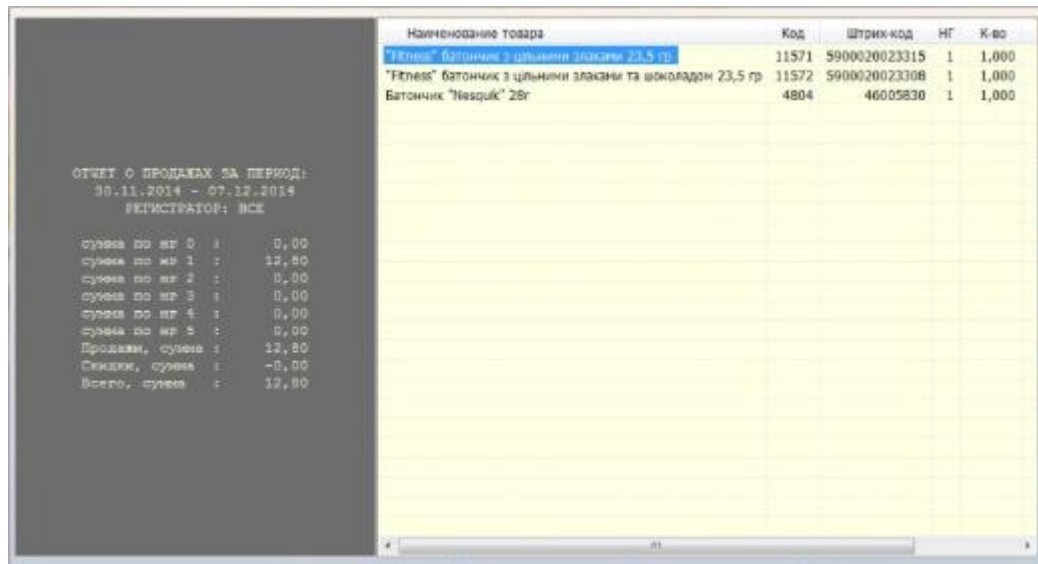


Рисунок 1.2.2 – Зображення вигляду програми CashFront

Підтримка: Windows, веб-браузер.

Сайт програми: <https://www.intech.co.ua/product/kopiiaprograma-dlia-kasira-cashfront/>

Український інтерфейс: так.

Умови розповсюдження: cashware.

1.3 Аналіз вимог до програмного забезпечення

З аналізу аналогічних систем було визначено функції що будуть реалізовувати, і які будуть основними в розроблювальній програмі. До основних задач відносяться:

- пошук та сортування за різними критеріями;
- ведення та редагування баз даних;
- продаж та бронювання квитків на сеанси;
- автоматичне формування звітів у вигляді графіків.

2 ІНСТРУМЕНТИ РЕАЛІЗАЦІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розвиток обчислювальної техніки підштовхує до розвитку всіх аспектів людського життя. Зокрема виникає необхідність в оновленні і модернізації вже існуючих програмних рішень для найефективнішого впровадження і використання наявних в системі ресурсів. Кожна програма підлягає періодичному оновленню та оптимізації і перерозподілу залучених ресурсів. Оскільки система що розроблюється передбачає одночасне її використання на декількох пристроях, а також вимагає швидкої обробки та створення інформаційних кластерів та роботи з невизначеною кількістю блоків даних через базу даних. Тому для реалізації поставленої задачі необхідно вибрати інструменти, які забезпечать можливості обробити швидко і ефективно інформацію, що перебуває в системі. Для розробки програмного продукту було обрано:

- Середовище MS Visual Studio 2019;
- Об'єктно-орієнтовану мову програмування C#;
- Фреймворк ADO.NET Entity Framework(Windows Forms);
- Систему керування базами даних MySQL Server;
- Операційну систему Windows 10;

Відповідно до вибраних інструментів можна створити зв'язок між ними та відобразити вплив один на одного як на рисунку:

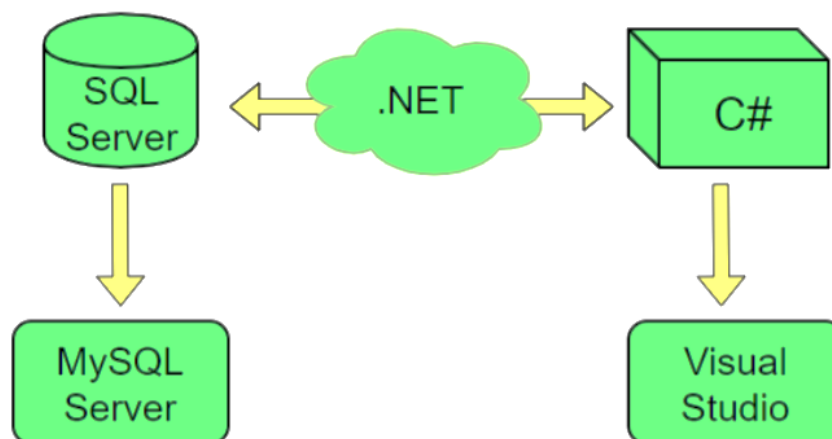


Рисунок 2.1 – Принцип функціональний роботи системи

2.1 Опис мови програмування C#

Програма розроблятиметься з допомогою мови програмування C# і з підтримкою фреймворку .NET Framework. Поєднання цих компонентів забезпечуватиме оптимальну роботу створених алгоритмів та спрощуватиме реалізацію і розгортання програми на комп'ютерних пристроях.

C# – об'єктно-орієнтована мова програмування з безпечною системою типізації для платформи .NET, являється простою і в той же час потужною мовою програмування, що дозволяє програмістам створювати багатофункціональні програми. Розроблена Андерсом Гейлсбергом, Скотом Вілтанутом та Пітером Гольде під егідою Microsoft Research [3].

Синтаксис мови програмування C# близький до C++ і Java. Звідти вона і наслідує строгу статичну типізацію, підтримує поліморфізм, перевантаження операторів, вказівники на функції-члени класів, атрибути, та ін. Також кірм плюсив своїх попередників C# виключає зі своєї структури інструменти що порушують стійкість програм, такі як множинне наслідування. Для реалізації програм застосовується компонент реалізації Система Common Language Runtime (CLR) – він включений в пакету Microsoft .NET Framework, представляє собою систему віртуальної машини, на якій виконуються всі мови платформи .NET Framework. Це відбувається через те що пакет CLR перетворює розроблений користувачем код в байт-код на мові IL, звідси і відбувається реалізації компіляції спроектованої системи з можливістю використання бібліотек класів .NET Framework, або так званою .NET Framework Class Library(FCL).

Мова C# позиціонується як мова програмування прикладного рівня для CLR і тому вона безпосередньо залежить, від можливостей та налаштувань самої CLR. Це стосується, перш за все, системи типів C#[3]. Присутність або відсутність тих або інших виразних особливостей мови диктується тим, чи може конкретна мовна особливість бути трансльована у відповідні конструкції CLR, тобто її розвиток впливає на ефективність використання мови програмування C# для розробки програмних рішень різної складності

До основних переваг C# можна віднести:

- об'єктно-орієнтованість мови сприяє широкому застосуванню мови в проектах різної спеціалізації;
- повний і добре визначений набір основних типів;
- автоматичне звільнення динамічно розподіленої пам'яті;
- повний доступ до бібліотеки базових класів .NET, а також легкий доступ до Windows API;
- просте зміна ключів компіляції. Дозволяє отримувати виконувані файли або бібліотеки компонентів .NET, які можуть бути викликані іншим кодом так само, як елементи управління ActiveX (компоненти COM)[3];



Рисунок 2.1.1 – Логотип C#

2.2 Опис середовища програмування Visual Studio 2019

Визначивши мову розробки, рекомендується вибрати середовище що буде зручне для користувача і водночас підтримуватиме використання всіх необхідних пакетів для виконання проектних завдань. Тому було обрано середовище Visual Studio 2019. Microsoft Visual Studio — складається з серії одноіменних продуктів що призначені для реалізації програмних завдань та вирішень проектних питань в розробці програмного забезпечення продукти з серії компанії Майкрософт [4], включають інтегроване середовище розробки програмного забезпечення та ряд інших інструментальних засобів. Ці продукти

дозволяють розробляти як консольні програми, так і програми з графічним інтерфейсом, в тому числі з підтримкою технології Windows Forms, веб-сайти, веб-застосунки, веб-служби на різних платформах в різних керованих середовищах із забезпеченням збереження програмних потужностей та алгоритмів, що підтримуються Microsoft Windows, Windows Mobile, Windows Phone, Windows CE, .NET Framework, .NET Compact Framework та Microsoft Silverlight [4]. Перевагами цього середовища є надійність, зручність та простота використання, можливість розгортання веб-застосунку через використання IIS сервері та можливість відладки програм, що спрощує пошук логічних помилок в коді.



Рисунок 2.2.1 – Логотип Visual Studio

2.3 Фреймворк для створення системних зв'язків. Entity Framework

ADO.NET Entity Framework представляє собою об'єктно-орієнтовану систему з набором методів для доступу для даних. Система Microsoft .NET представляє через EF власне програмне відображення реляційних зв'язків в системі та забезпечує реалізацію підключень та взаємодії з об'єктами бази даних за межами системних реалізацій.

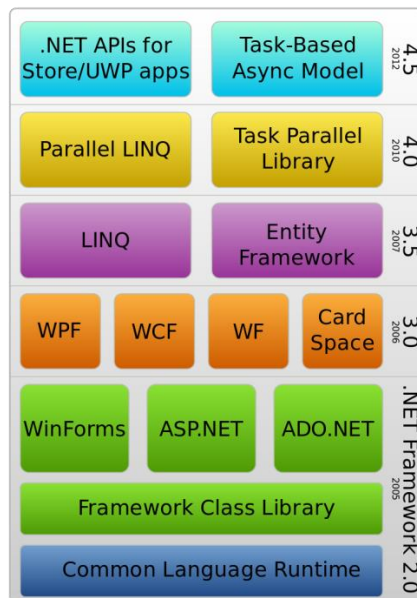


Рисунок 2.3.1 - Компоненти .NET Framework та їхня градація

.NET та окремі її реалізації включає в себе компоненти, зображені на рисунку вище:

- одне або декілька виконавчих середовищ (приклади CLR для .NET Framework, CoreCLR і CoreRT для .NET Core);
- бібліотека класів, яка реалізує .NET Standard, а також може реалізовувати додаткові API-інтерфейси (приклади: бібліотека базових класів .NET Framework, бібліотека базових класів .NET Core);
- одна платформа додатків або кілька (приклади ASP.NET, Windows Forms і Windows Presentation Foundation (WPF) входять в .NET Framework)[5];

Entity Framework передбачає роботу не з таблицями, а з об'єктами та їх множинами. Будь-яка сутність, як і будь-який об'єкт з реального світу, має ряд властивостей. Властивості не обов'язково являють собою прості дані типу int, але можуть також представляти більш складні структури даних. І кожна сутність може мати одне або кілька властивостей, які будуть відрізняти цю сутність від інших і однозначно ідентифікувати цю сутність. У той же час сутність може бути пов'язана різними асоціативними зв'язками з одніч чи багатьма елементами. Відмінною особливістю Entity Framework є використання

запитів LINQ для вибірки даних бази даних. отримувати об'єкти, пов'язані з різними асоціативними посиланнями, та працювати з ними чи модифікувати.

Іншою ключовою концепцією є модель даних суб'єкта. Ця модель порівнює класи об'єктів з реальними таблицями в базі даних. Модель даних об'єкту складається з трьох рівнів: концептуального, рівень сховища і рівень зіставлення (маппінга),

На концептуальному рівні відбувається визначення класів сутностей, додатку.

Рівень зіставлення (маппінга) служить посередником між попередніми двома, визначаючи зіставлення між властивостями класу суті і стовпцями таблиць. Таким чином, ми можемо через класи, визначені у додатку, взаємодіяти з таблицями з бази даних [5]. Схематично структура моделі даних зображена на рисунку нижче.

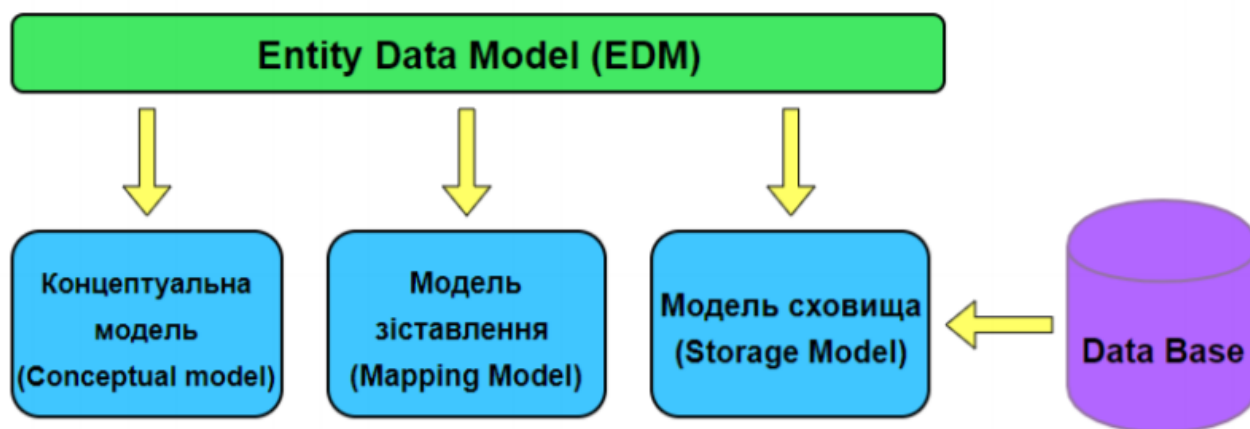


Рисунок 2.3.2 - Модель Entity Data Model і її відображення її зв'язків

Фреймворк Entity Framework передбачає створення і використання трьох способів взаємодії з базою даних :

- метод Database first: застосовується і більш поширений серед проектувальників баз даних. В своїй суті визначає попереднє створення базу даних за допомогою різних інструментів, а потім генерується її EDMX-модель. В даному випадку вам потрібно працювати з SQL Server і добре знати

синтаксис T-SQL, але при цьому не потрібно бути досвідченим у використанні мови програмуванні, зокрема в C #;

- метод Model first: поширений серед програмних архітекторів. Її використання починається через створення графічної моделі EDMX в Visual Studio і після цього через створене модельне представлення генерується базу даних. В цьому випадку знання для побудови баз даних, зокрема T-SQL чи навички роботи з C#. не відіграватимуть вирішальну роль в процесі розробки зв'язків зі сховищем даних.

- метод Code first: використовується програмістами. Процес визначається тим що розробник не застосовує EDMX і всі налаштування класів системи відбувається вручну.



Рисунок 2.3.3 – Логотип методології Entity Framework

2.4 Механізм взаємодії з базою даних MySQL Server

MySQL Server представляє собою набір інструментів та представлень для взаємодії з базами даних. База даних в своїй основі являє структуровану сукупність даних. Вони представляють різні можливості для внутрішньої взаємодії зокрема для запису, вибірки і обробки даних, що зберігаються в комп'ютері. Дані можливості потребують наявності та включення в структуру програми системи управління базами, якою і є MySQL.

Оскільки обробка великих обсягів даних є важливим для комп'ютера процесом, то управління базами даних відіграє в ньому головну роль. У реляційній базі даних MySQL дані зберігаються в окремих таблицях, це забезпечує швидкість і гнучкість[6].

Таблиці зв'язуються між собою за допомогою відносин, це дозволяє об'єднувати дані з декількох таблиць. Система керування даними MySQL Server використовується для роботи з базами даних розміром від персональних до великих баз даних. Тому для реалізації систем з невизначеним розміром ефективного сховища даних MySQL Server підходить найкраще



Рисунок 2.4.1 – Логотип системи MySQL Server

3 ПРОЕКТУВАННЯ ПРОГРАМНОГО ПРОДУКТУ

3.1 Вибір моделі розробки

Модель життєвого циклу – це структура, що складається із процесів, робіт та задач, які включають в себе розробку, експлуатацію і супровід програмного продукту; охоплює життя системи від визначення вимог до неї до припинення її використання.[7]

Найбільш широко використовуються наступні моделі життєвого циклу (моделі розробки) :

- каскадна;
- прототипного проектування;
- еволюційна.

Каскадна – модель розробки, в якій кожний етап виконується лише один раз. На кожному етапі робота виконується настільки ретельно, щоб не повертатись до попереднього. Результат виконання кожного етапу, перед передачею в наступний, піддається верифікації.

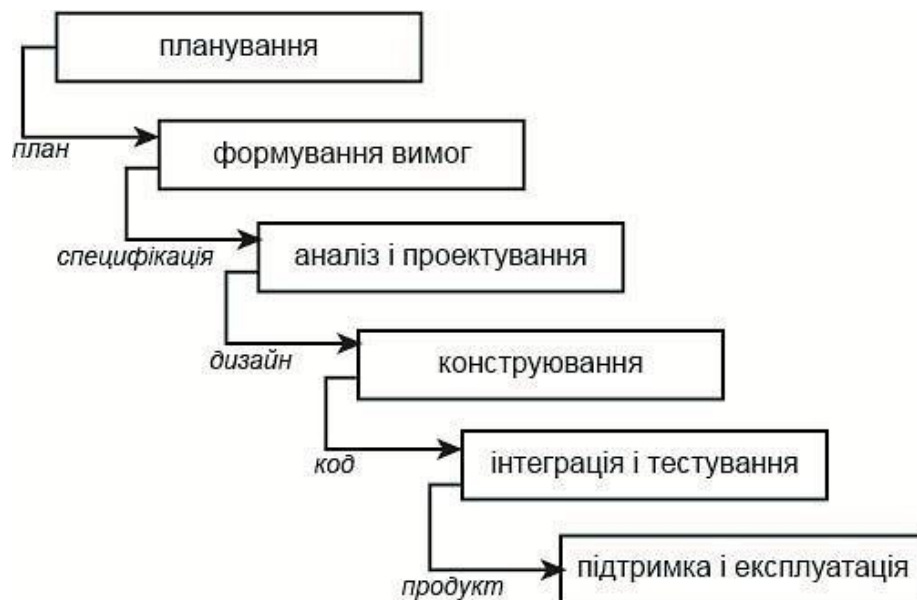


Рисунок 3.1.1 – Каскадна модель розробки

Методологія прототипного програмування в своїй основі заснована на концепції RAD, що еволюціонувала з спіральної моделі розробки ПЗ.

RAD (RapidApplicationDevelopment), – представляє концепцію методології розробки додатків, програмних продуктів та засобів, спрямованих

на пришвидшення та часової оптимізації виконання завдань, з метою збільшення зручності програмування та створенню технологічного процесу, що дозволяє максимально швидко створювати комп'ютерні програми.

Дана технологія забезпечує на ранній стадії реалізацію реально працюючого ПЗ (прототипу), що дозволяє наочно продемонструвати користувачеві майбутню систему, уточнить її інтерфейсні елементи: форми введення повідомлень, меню, вихідні документи, структуру діалогу, склад функцій, що реалізуються тощо. В процесі роботи з прототипом користувач реально усвідомлює можливості майбутньої системи і визначає найбільш зручний для нього режим обробки даних.[7] На етапі аналізу і проектування передбачається виконання наступних дій:

- визначення функцій, які повинна виконувати система;
- виділення пріоритетів функцій, що вимагають опрацювання насамперед;
- опис інформаційних потреб.

На етапі проектування уточнюються і доповнюються вимоги до системи, що не були виявлені на попередній стадії проектування, зокрема:

- детальніше розглядаються процеси системи;
- для кожного елементарного процесу створюється (при необхідності) частковий прототип: екранна форма, діалог і звіт, що знімає неясності і неоднозначності;
- встановлюються вимоги розмежування доступу до даних;
- визначається склад необхідної документації.

А на етапі реалізації відбувається розробка прототипу програмного продукту з поступовою реалізацією основних функцій системи, починаючи від найважливіших за пріоритетом.

Дана методологія є гнучкою, бо дозволяє доповнювати специфікацію і тому на відміну від каскадної є широкоживаною.[7]

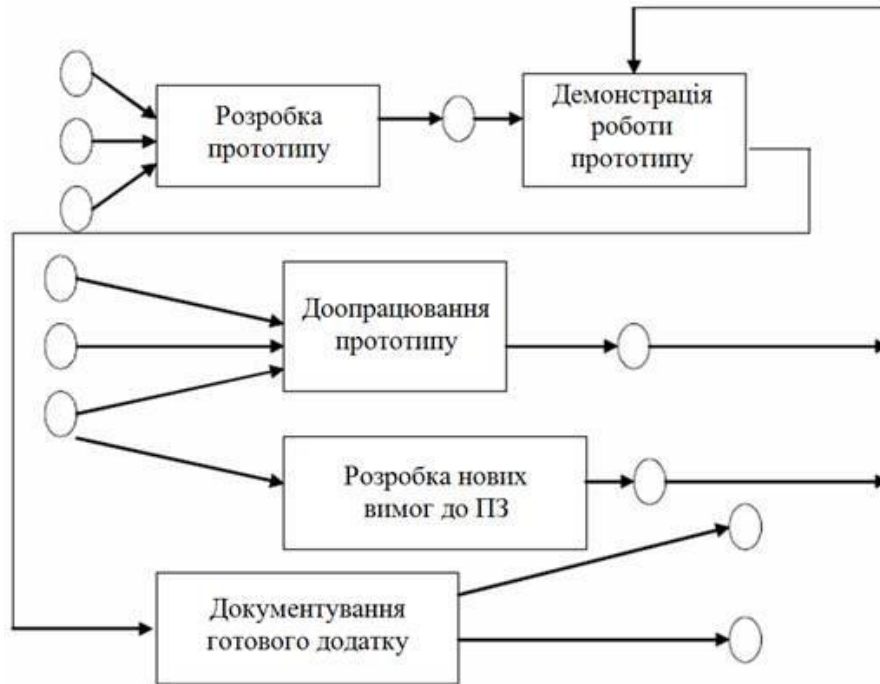


Рисунок 3.1.2 – Методологія прототипного програмування

Еволюційна модель передбачає послідовну розробку через створення блоків конструкцій. На відміну від інкрементної моделі в еволюційній моделі вимоги встановлюються частково і уточнюються в кожному наступному проміжному блоці структури системи.[7]

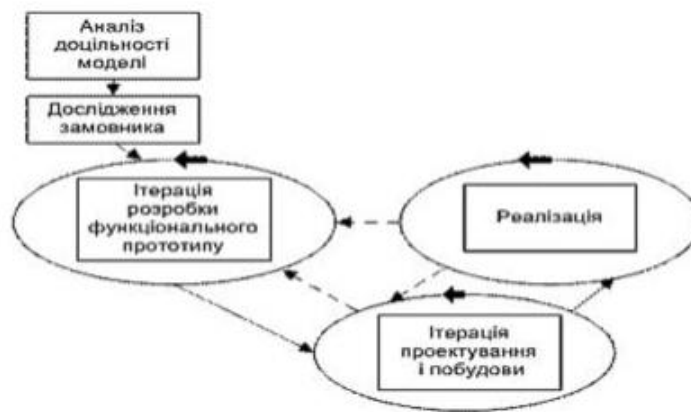


Рисунок 3.1.3 – Еволюційна модель розробки

Проаналізувавши основні переваги кожної із розглянутої методологію було вирішено вибрати модель що дає найшвидші результати розробки застосунку, що в свою чергу призведе до можливості швидкої зміни проектнів планів та в результаті зменшити час на валідацію та тестування кожного функціонального блоку системи через періодичну перевірки вже готових

реалізацій. Також необхідно враховувати часові рамки проекту та обмеженість ресурсів що спричиняють необхідність швидких результатів для подільшої розробки. Тому реалізація тестової демонстраційної програми буде розроблена з використанням методології прототипного проектування ПЗ.

3.2. Проектування вимог та варіантів використання програмного забезпечення

Оскільки система спрямована на полегшення роботи касира то повинна включати в свій функціонал дії спрямовані на оптимізацію завдань покладених на робітника. Після аналізу систем аналогічної роботи було вирішено розробляти систему згідно наступних вимог до функціоналу

- авторизація користувачів;
- перегляд списку користувачів;
- редагування даних користувачів;
- додавання, редагування та видалення виступів;
- додавання, редагування та видалення запланованих постанов;
- продаж квитків;
- бронювання квитків;
- створення звітів касового обігу;

До нефункціональних вимог відноситься потреби для коректного відображення і роботи системи для забезпечення комфортного користування користувачем програми. В систему передбачається включити:

- підтримку версії для Windows від 7 версії;
- створення інтуїтивно простого авторського інтерфейсу;
- автоматизацію контролю системного оновлення;
- мінімізацію елементів з надмірним системними налаштуваннями (для запобігання нагромадження інформації та установок що можуть зашкодити коректній роботі програми);

Відповідно до вказаних умов формуються варанти використання та актори що безпосереднь впливатимуть на процес їх виконання. Варіант

використання – у розробці програмного забезпечення та системному проектуванні це опис поведінки системи, як вона відповідає на зовнішні запити. Для проведення даного етапу необхідно насамперед визначаються можливі ролі, що будуть присвоюватись користувачам, серед яких:

- Касир;
- Адміністратор;
- Система;

Касир – це користувач який покликаний проводити операції щодо продажу для клієнтів білетів та маніпулювати даними про заплановані вистави. Тож він повинен мати право:

- Додавати, редагувати, видаляти сеанси вистави;
- Продавати квитки на вистави;
- Бронювати квитки на вистави;

Адміністратор – це актор що з допомогою програми повинен мати повний доступ до маніпуляції даними щодо інформаційного наповнення системи, також передбачається надати можливість проводити звітність та облік в системі, контролювати роботу персоналу, мати доступ до системи впровадження нових ресурсів підконтрольних компанії та впровадження прямого контролю за базою даних. Функцій що виконуватимуться адміністратором повинні виконувати наступні дії:

- Додавати, редагувати, видаляти сеанси вистави;
- Продавати квитки на вистави;
- Бронювати квитки на вистави;
- Додавати, редагувати, видаляти вистави театру;
- Додавати, редагувати, видаляти дані про зали театру;
- Маніпулювати базою даних персоналу театру;
- Маніпулювати базою даних користувачів системи;
- Перегляд звітів.

Система повинна:

- Зберігати дані сеансів, квитків, фільмів, залів, персоналу та користувачів;
- Реалізовувати продаж і бронювання квитків на сеанс;
- Формувати різні звіти;
- Виконувати пошук, сортування, фільтрацію даних.

Діаграма варіантів використання допомагає представити проектовану систему у вигляді сутностей чи акторів і їх методів взаємодії з системою, варіантів використання. Усі варіанти використання утворюватимуть робочий(користувацький) процес програми.

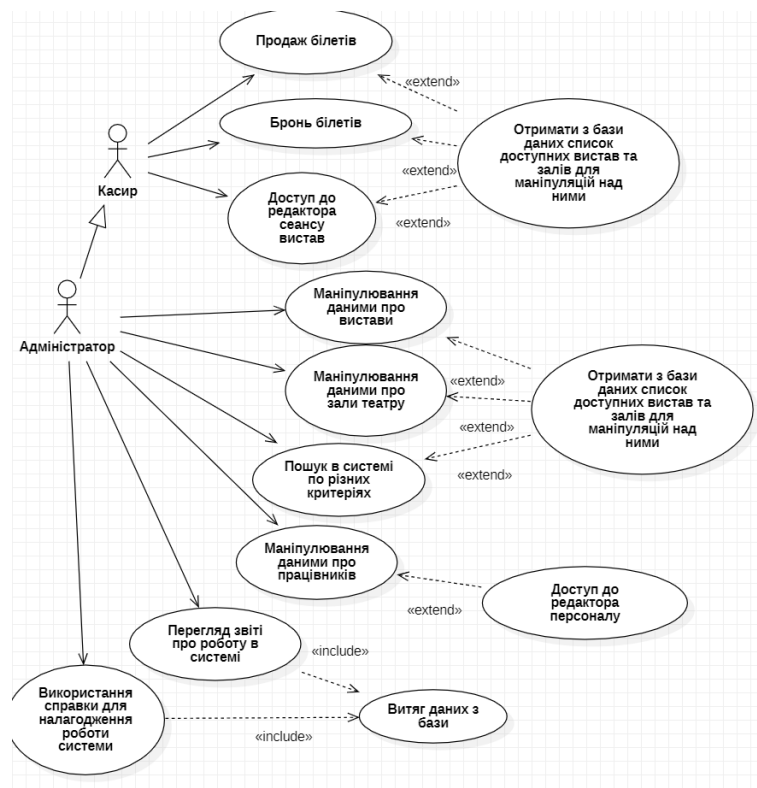


Рисунок 3.2.1 – Діаграма варіантів використання системи

3.3 Організація класів

Для реалізації запланованих сценаріїв вирішено створити класи що реалізовуватимуть запланований функціонал. Так праграма складається з таких компонентів:

- AuthorizationReal – містить спосіб переводу користувача до меню що йому доступно згідно виділеною роллю в системі;

- AuthorizationProxy – мітить спосіб перевірки перед переводом користувача до меню згідно виділеною роллю в системі;
- ServiceInterface - інтерфейс, що передбачає перехоплення вводу даних для перевірки рівня доступу користувача;
- CashierSys – реалізує вікно з методами доступу до функцій передбачених для роботи касираструктура яка відповідає за безпосередній контакт з базою даних;
- AdminSys – реалізує вікно з методами доступу до функцій передбачених для роботи касираструктура яка відповідає за безпосередній контакт з базою даних;
- StaffDetail – реалізує механізми роботи з даними про персонал театру;
- FormDocumentation – надає механізми для реалізації звітів по продажах білетів;
- SettingsHelp – клас що викликає справку з порадами для використання програми;
- ConnectToBD – клас представляє реалізацію зв'язку з базою даних системи;
- ShowManager – представляє інструменти для зв'язку та роботи з даними, що допомагає реалізовувати роботу по створення білетів та контролю інформації про вистави;
- SessionManager– реалізує можливість створювати сеанси виступів та деталі їх проведення;
- TiketSettings – реалізує можливість купівлі та реєстрації придбаних білетів в системі для ідентифікації перед виступом;
- ReservedTikets – клас що приводить можливість резервування білетів перед купівлею;
- ResourseManager – клас, що реалізує систему доступу до ресурсів театру, як

дані про зали і виступи;

- TheatreAction – клас, що реалізує роботу з виступами театру;
- HallDetail – клас, що реалізує роботу з залами театру;

Крім того передбачено використання для класів-форм для відтворення візуального супровіду роботи більшості класів системи. а також використання бібліотек для спрощення флгоритмів для маніпуляції з базою даних та відтворення їх звязків.

3.4 Метамодель системи та опис системи класів

Створюючи будь яку програму розробникам виникає необхідність розділяти розроблювану систему на пакети що нестимуть окрему відповідальність за виконання тієї чи іншої функції програми. Так можна ієрархічно розділити компонент для досягнення порядку між зв'язками системи та визначити як взаємодітимуть елементи програми один з одним та їх логічну структуру. В результаті оформлюється модель розробки для кожного компонента, які в сукупності становитимуть метамодель програмної системи.

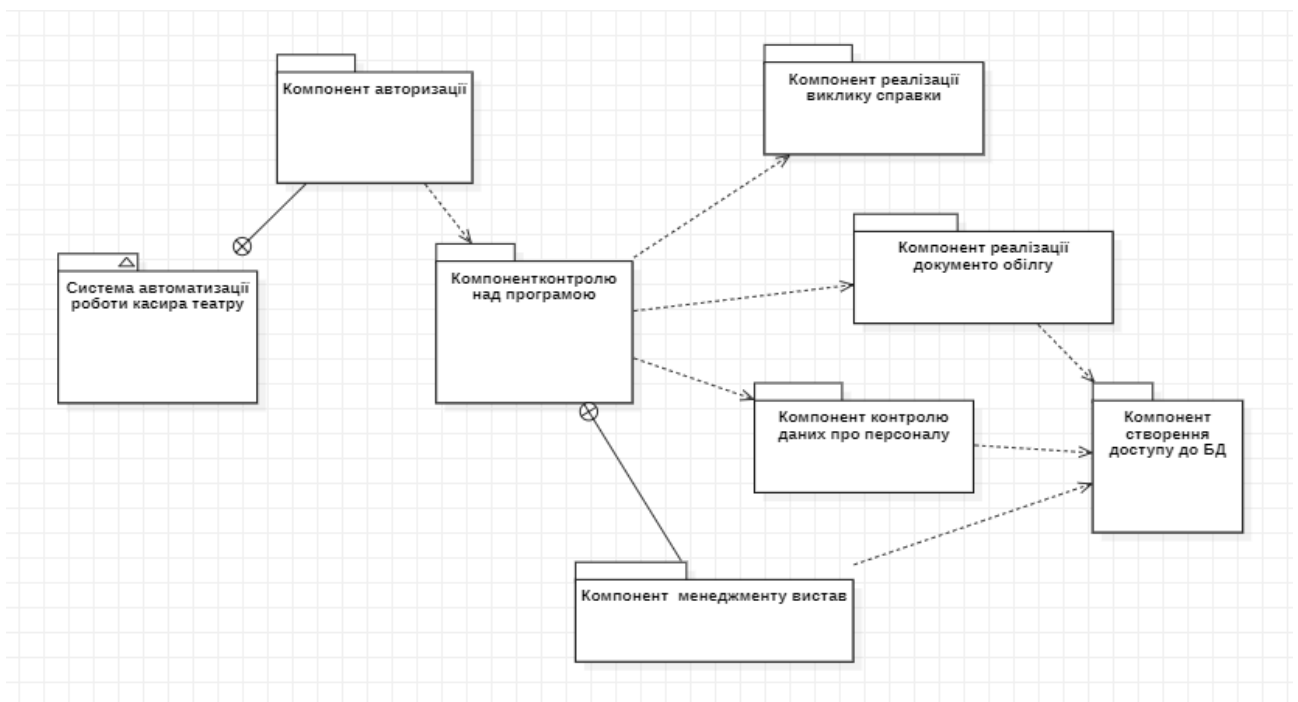


Рисунок 3.4.1 – Діаграма мета моделі системи

Кожний з компонентів реалізує певні вимоги що були поставлені до системи. Кожна частина відповідає за логічну реалізацію поставлених вимог та інструментів для взаємодії з ними.

Компонент авторизації – відповідає за авторизацію користувача та надання відповідно до актора чи відхилення доступу до роботи з системою. Компонент включає групу класів та інтерфейс що реалізуються патерном замісник для проведення авторизації користувача. Класи компонента:

AuthorizationReal – містить спосіб переводу користувача до меню що йому доступно згідно виділеною роллю в системі. Реалізує ServiceInterface агрегує клас AuthorizationProxy, має зв'язок з класом ConnectToBD, CashierSys, AdminSys.

Методи:

- redirectToActor() – проводить процес переходу системи до функціонального меню програми для конкретного актора;

AuthorizationProxy – містить спосіб перевірки перед переводом користувача до меню згідно виділеною роллю в системі. Реалізує ServiceInterface агрегується класом AuthorizationReal

Змінні:

- real – об'єкт класу AuthorizationReal;

Методи:

- AuthorizationProxy(in s:AuthorizationReal) – конструктор що ініціалізує об'єкт класу AuthorizationReal;
- redirectToActor() – проводить процес переходу до класу призначеного для надання інструментів відповідно до визначеного актора.
- checkAccess() – проводить перевірку введених даних. дозволяє перехід до класу призначеного для надання інструментів відповідно до визначеного актора

ServiceInterface - інтерфейс, що передбачає перехоплення вводу даних для перевірки рівня доступу користувача; Реалізується класами

Методи:

- `redirectToActor()` – визначає метод переходу користувача до інструментів системи;

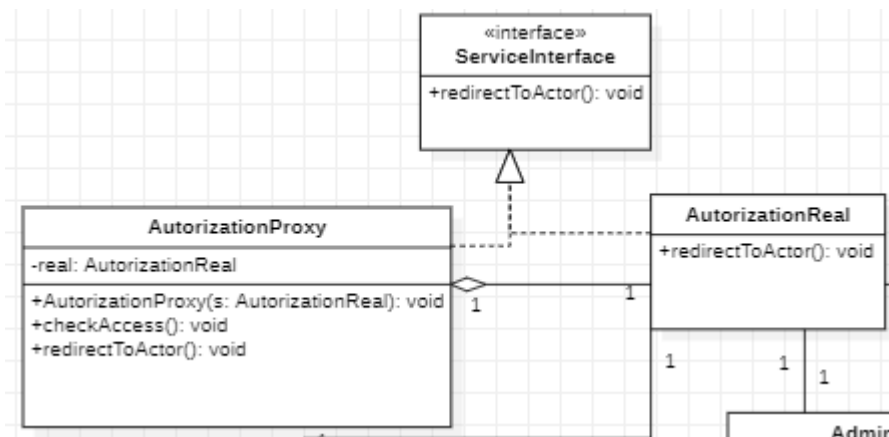


Рисунок 3.4.2 – Діаграма класів що відображає компонент авторизації

Компонент контролю над програмою – відповідає за представленням для користувача методів та інструментів роботи з інформацією відповідно до його рівня допуску. Компонент включає групу класів що реалізують меню для подальшої роботи з системою. Класи компонента:

CashierSys – реалізує вікно з методами доступу до функцій передбачених для роботи касираструктура яка відповідає за безпосередній контакт з базою даних. Має зв'язок з `AuthorizationReal`, наслідує `ShowManager`.

Методи:

- `callTheaterRessorse()` – надає інструменти для доступу та витягу даних з бази ресурсів театру;
- `callTiketsSetting()` – надає інструменти для фомування і маніпулювання даних для білетів на вистави;
- `callSessionManager()` – надає інструменти для доступу та витягу даних з бази розкладів та опису вистав;

AdminSys – реалізує вікно з методами доступу до функцій передбачених для роботи касираструктура яка відповідає за безпосередній контакт з базою даних; Має зв'язок з класами `AuthorizationReal`, `FormDocumentation`, `SettingsHelp`, `StaffDetail` наслідує `ShowManager`.

Змінні:

- `staf` – об'єкт класу `StaffDetail`;

- doc – об’єкт класу FormDocumentation;
- setting – об’єкт класу SettingsHelp;

Методи:

- callStaffDetWindow() – надає інструменти для доступу та витягу даних з бази персоналу;
- callSettingsWindow() – надає інструменти для доступу та витягу даних зпро програму;
- callFormDocWindow() – надає інструменти для доступу та формування документації про роботу театру;
- callTheaterRessorse() – надає інструменти для доступу та витягу даних з бази ресурсів театру;
- callTiketsSetting() – надає інструменти для фомування і маніпулювання даних для білетів на вистави;
- callSessionManager() – надає інструменти для доступу та витягу даних з бази розкладів та опису вистав;

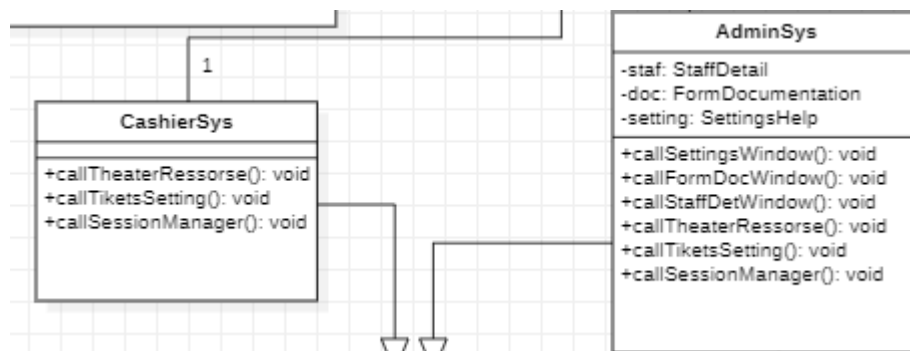


Рисунок 3.4.3 – Діаграма класів що відображає компонент контролю над програмою

Компонент реалізації виклику справки – відповідає за представлення для користувача доступу до справки програми. Компонент включає клас що реалізує доступ до виклику справки. Клас компонента:

SettingsHelp – клас що викликає справку з порадами для використання програми. Має зв'язок з класом, AdminSys.

Методи:

- settingCall():– проводить процес виклику справки та доступу до її перегляду;

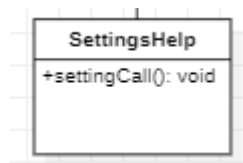


Рисунок 3.4.4 – Діаграма класів що відображає компонент реалізації виклику справки

Компонент контролю даних про персоналу – відповідає за представлення для користувача доступу до даних про персонал театру. Компонент включає клас що реалізує доступ до виклику вікна з функції для роботи з інформацією про персонал. Клас компонента:

StaffDetail – реалізує механізми роботи з даними про персонал театру. Має зв'язок з класом, AdminSys та ConnectToBD.

Методи:

- getStuff():– проводить процес виклику даних персоналу;
- setStuff() – створює новий запис;
- changeStuff() – змінює записи про персонал в базі;
- delStuff() – видаляє дані про персонал;

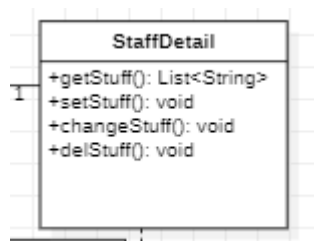


Рисунок 3.4.5 – Діаграма класів що відображає компонент контролю даних про персоналу

Компонент реалізації документообігу – відповідає за представлення для користувача можливості створення документів та аналізу проведених дій щодо продажу та популяризації вистав театру. Компонент включає клас що реалізує доступ до виклику вікна з функції для роботи з створення та аналізом інформації про виступи. Клас компонента:

FormDocumentation – надає механізми для реалізації звітів по продажах білетів. Має зв'язок з класом, AdminSys та ShowManager.

Змінні:

- show – об'єкт класу ShowManager;

Методи:

- createDocToTiket() – проводить процес створення звіту по активності продажу білетів на конкретний виступ;
- createDocToHallOccuring() – створює звіт для аналізу заповненості залу;
- createDocToSelingPeriod() – надає звіт щодо рівня продажів на виставу протягом певного періоду;
- createDocToSessionTiketSeling() – надає звіт щодо рівня продажів білетів на різні вистави;

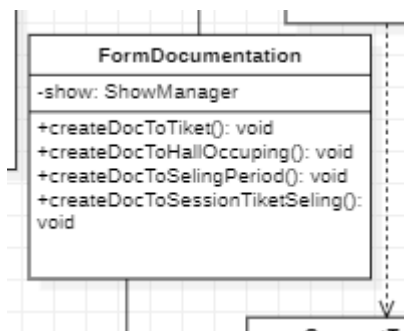


Рисунок 3.4.6 – Діаграма класів що відображає компонент реалізації документообігу

Компонент створення доступу до БД– відповідає за представлення доступу іншим компонентам до створення каналу зв'язку з масивами даних в системі Компонент включає клас що реалізує патерн синглетон для зменшення нагромадження надлишкових об'єктів доступу до бд, що дозволить використовувати обчислювані ресурси компютера на оптимальні завдання. Клас компонента:

ConnectToBD – клас представляє реалізацію зв'язку з базою даних системи. Має зв'язки з класами AutorizationReal, ShowManage, StaffDetail.

Змінні:

- inst – об’єкт класу ConnectToBD;

Методи:

- ConnectToBD () – конструктор що ініціалізує об’єкт класу;
- getItem() – метод для отримання об’єкту доступу до БД

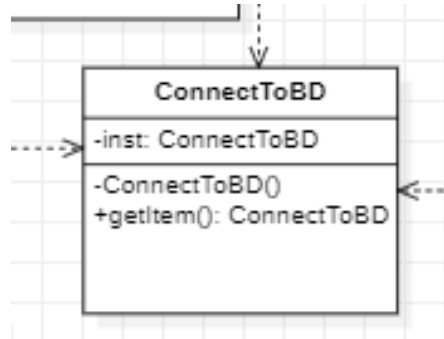


Рисунок 3.4.7 – Діаграма класів що відображає компонент створення доступу до БД

Компонент менеджменту вистав – відповідає за представленням для користувача доступу даних про ресурси театру, керування властивостями білетів та створювати сеанси виступів з описами їх проведення. Компонент включає класи що реалізує патерни посередник та фасад для впорядкування звязків між класами системи та об’єднання систем з різними властивостями але однією спрямованістю в одну компонентну систему. Класи компонента:

ShowManager – представляє інструменти для звязку та роботи з даними, що допомагає реалізовувати роботу по створення білетів та контролю інформації про вистави. Наслідується класами AdminSys, CashierSys агрегує SessionManager, TiketSettings, ResourceManager, має звязок з класами FormDocumentation, ConnectToBD.

Методи:

- callTheaterRessorse() – ініціалізує процес роботи з ресурсами театру;
- callTiketsSetting() – ініціалізує процес роботи з білетами;
- callSessionManager() – ініціалізує процес роботи з сесіями;

SessionManager– реалізує можливість створювати сеанси виступів та деталі їх проведення. Агрегується класом ShowManager.

Методи:

- getSession():– проводить процес виклику даних про сесію;
- setSession() – створює нову сесію;
- changeSession() – змінює записи в сесії;
- delSession() – видаляє дані про сесію;

TiketSettings – реалізує можливість купівлі та реєстрації придбаних білетів в системі для ідентифікації перед виступом. Агрегується класом ShowManager, включає в себе як композицію клас ReservedTikets.

Методи:

- getTiket():– проводить процес виклику даних про білети;
- setTiket() – створює новий білет;
- changeTiket() – змінює записи про білети;
- delTiket() – видаляє дані про білет;

ReservedTikets – клас що приводить можливість резервування білетів перед купівлею. Агрегується класом TiketSettings.

Методи:

- getResTiket():– проводить процес виклику даних про резервовані білети;
- setResTiket() – створює новий резервований білет;
- changeResTiket() – змінює записи про резервовані білети;
- delResTiket() – видаляє дані про резервовані білети;

ResourceManager – клас, що реалізує систему доступу до ресурсів театру, як дані про зали і виступи. Агрегується класом ShowManager, має звязки з класами TheatreAction, HallDetail.

Змінні:

- hall – об’єкт класу HallDetail
- theatre – об’єкт класу TheatreAction;

Методи:

- getAccessToHallDet() – ініціалізує процес роботи з залами театру;

- `getAccessTheatreAction()` – ініціалізує процес роботи з виступами театру;

`TheatreAction` – клас, що реалізує роботу з виступами театру. Має зв'язок з класом `ResourceManager`.

Методи:

- `getShow()`:- проводить процес виклику даних про виставу;
- `setShow()` – створює новий запис вистави;
- `changeShow()` – змінює записи про вистави;
- `delShow()` – видаляє дані про вистави;

`HallDetail` – клас, що реалізує роботу з залами театру. Має зв'язок з класом `ResourceManager`.

Методи:

- `getHallDet()`:- проводить процес виклику даних персоналу;
- `setHallDet()` – створює новий запис;
- `changeHallDet()` – змінює записи про персонал в базі;
- `delHallDet()` – видаляє дані про персонал;

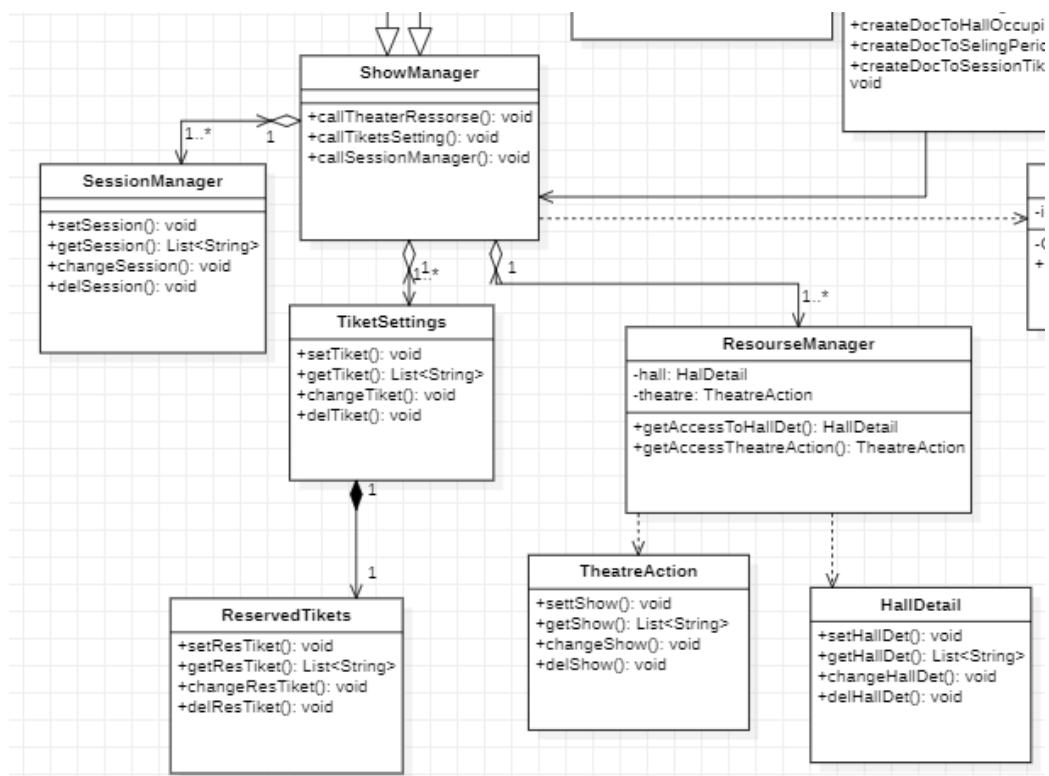


Рисунок 3.4.8 – Діаграма класів що відображає компонент менеджменту вистав

В результаті при заміні компонентів на класи що реалізують відповідний функціонал, утворюється діаграма класів, яка визначає шляхи виконання роботи системи, зв'язки між компонентами та механізми реалізації тих чи інших завдань що будуть ініціалізовані користувачем.

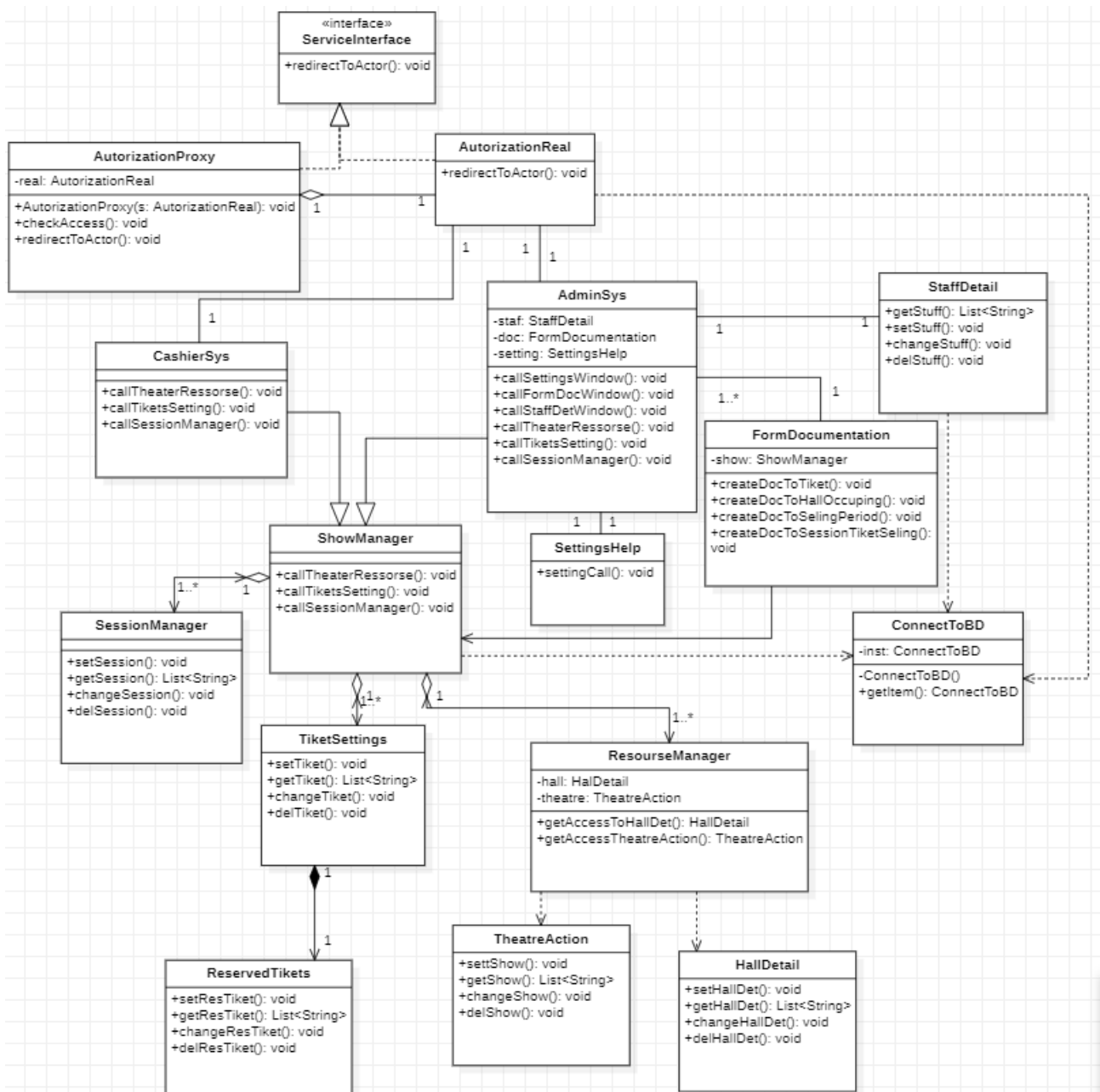


Рисунок 3.4.9 – Цілісна діаграма класів системи

3.5 Проектування бази даних

Програма для своєї роботи вимагає наявності бази даних що буде зберігати інформацію щодо роботи касира, структури вистав, дані про білети та ін. Тому для реалізації поставленої задачі необхідно спроектувати и реалізувати базу даних.

Проектування БД представляє процес створення схеми БД і визначення необхідних обмежень цілісності. До його етапів відносять:

- Концептуальне проектування.
- Логічне проектування.
- Фізичне проектування.

Проектування БД може бути виконано різними методами. Концептуальне проектування проводиться за допомогою інтерв'ювання, тобто через порівняння та аналізу роботи аналогічних систем.

В якості методу логічного проектування БД в рамках даної системи обраний метод «сутність-зв'язок» або по іншому «ER-діаграм». Основними поняттями цього методу є: сутність, атрибут сутності, ключ сутності, зв'язок між сутностями, ступінь зв'язку, клас приналежності екземплярів суті, діаграми ER-екземплярів, діаграми ER-типу[8].

Сутність визначається як деякий об'єкт розглянутої предметної області, інформація про який повинна бути відображена в БД. Атрибут сутності – це властивість сутності.

В результаті проектування БД була побудована інформаційно логічна модель БД. Дана модель являє собою структуру таблиць БД і встановлену між ними зв'язок. Отримана інформаційно-логічна модель представлена на рисунку 3.5.1

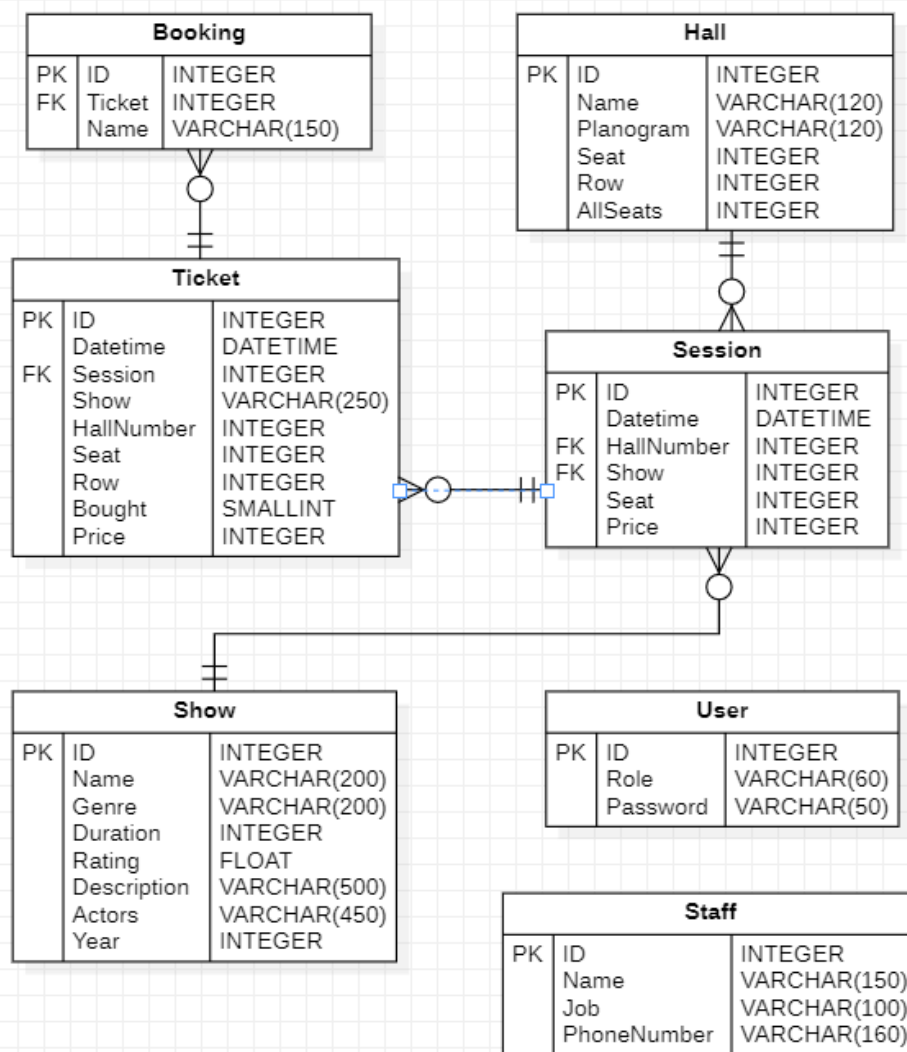


Рисунок 3.5.1 - ER-діаграма сутностей

Діаграма представляє наступні таблиці та їх атрибути:

- Таблиця «User» зберігає дані користувачів для авторизації, їхню роль в системі та пароль для авторизації;
- Таблиця «Staff» зберігає дані персоналу театру, до яких відносяться ім'я, посада та номер телефону;
- Таблиця «Session» зберігає дані про дату і час, номер залу, виставу, доступні місця і ціну за сеанс, з таблицями «Ticket», «Movie», «Hall» має зв'язок багато до багатьох, Це необхідно для того, щоб створювати детальні описи сеансу вказуючи всі можливі деталі для надання необхідної інформації про особливості вистави та форму її проведення;

– Таблиця «Ticket» зберігає дані про час, сеанс, виставу, номер залу, місце і ряд, стан купівлі та ціну квитка, Відображає зв'язок багато до багатьох з таблицями «Session», «Booking» для перегляду, додавання та видалення інформації щодо бронювання і продажу квитків для кожного доступного сеансу.

– Таблиця «Show» зберігає інформацію про вистави театру, а саме назву, жанр, тривалість, рейтинг, опис, акторів та рік випуску фільму та представляє зв'язок один до багатьох з таблицею «Session»;

– Таблиця «Hall» відповідає даним про доступні зали, зберігає назву, планограму, місце і ряд, всі доступні місця зв'язується через відношення один до багатьох з таблицею «Session»;

– Таблиця «Booking» містить інформацію про заброньовані квитки та контактує з таблицею «Ticket» через зв'язок один до багатьох;

4 ТЕСТУВАННЯ ПРОГРАМНОГО ПРОДУКТУ

Програма тестується з допомогою системи Windows 10, з об'ємом оперативної пам'яті 8ГБ

Користувач при запуску програми має доступ до вікна авторизації користувача, де він має поля для логіну та паролю, по замовчуванню поле що відповідає користувачу відображає роль «Адміністратор».

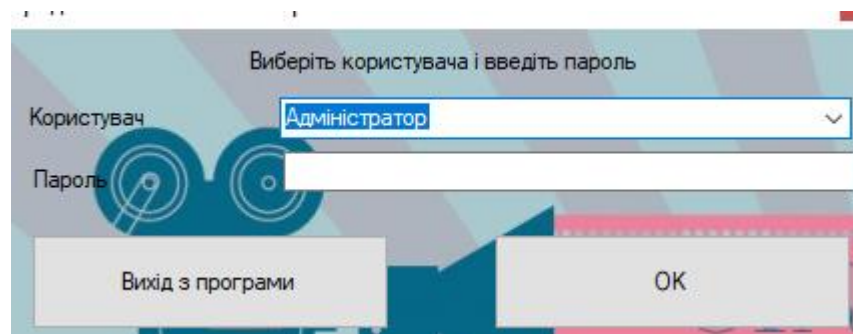


Рисунок 4.1 – Вікно авторизації користувача при вході в систему

Якщо ввести не правильний пароль чи логін система відобразить відповідне повідомлення користувачу.

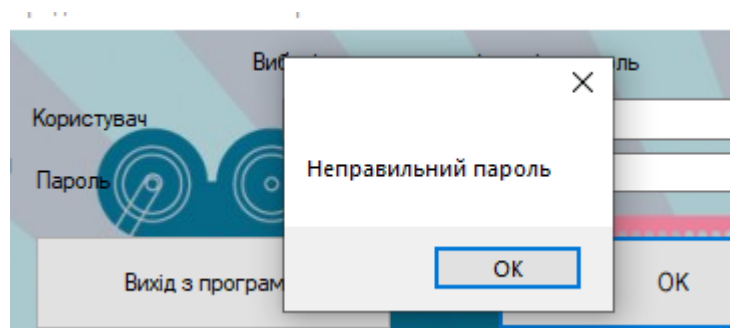


Рисунок 4.2 – Реакція на неправильно введені дані

Після успішної авторизації, форма авторизації закриється та появляється вікно головного меню де користувач зможе переглянути список користувачів, клієнтів, категорій, список продуктів, замовлень та повернутись до вікна авторизації.

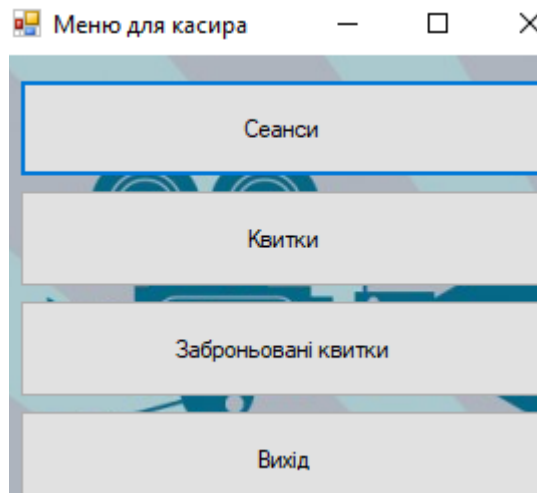


Рисунок 4.3 – Меню для системи обслуговування

Якщо користувач натисне на кнопку «Сеанси», він перейде в форму з сеансами. Форма містить таблицю зі списком сеансів, планогомою залів та списком розпроданих місць. Авторизований користувач може проводити пошук, сортування, додавання та редагування сеансів

При виборі клієнта, його дані переносяться в поля вводу, а система в свою чергу вираховує кількість замовлень, підраховує суму всіх покупок та дату останнього замовлення вибраного користувача.

Дата час початку	Зал	Вистава	Всього місць	Вільних місць	Ціна квитка
12.12.2020 21:30	Великий	Втеча з Шоушен...	56	51	120
16.12.2020 15:05	Великий	Ігри розуму	56	56	90
22.12.2020 18:45	Комфортний	Зелена миля	48	48	100

Пошук по:

- ☐ Дати та часу початку
- ☐ Залу
- ☐ Фільму

Шукати Очистити

Продати квиток:

Ряд 1 Місце 1

Продати квиток

ПІБ для бронювання

Забронювати квиток

Планова:

Сортування:

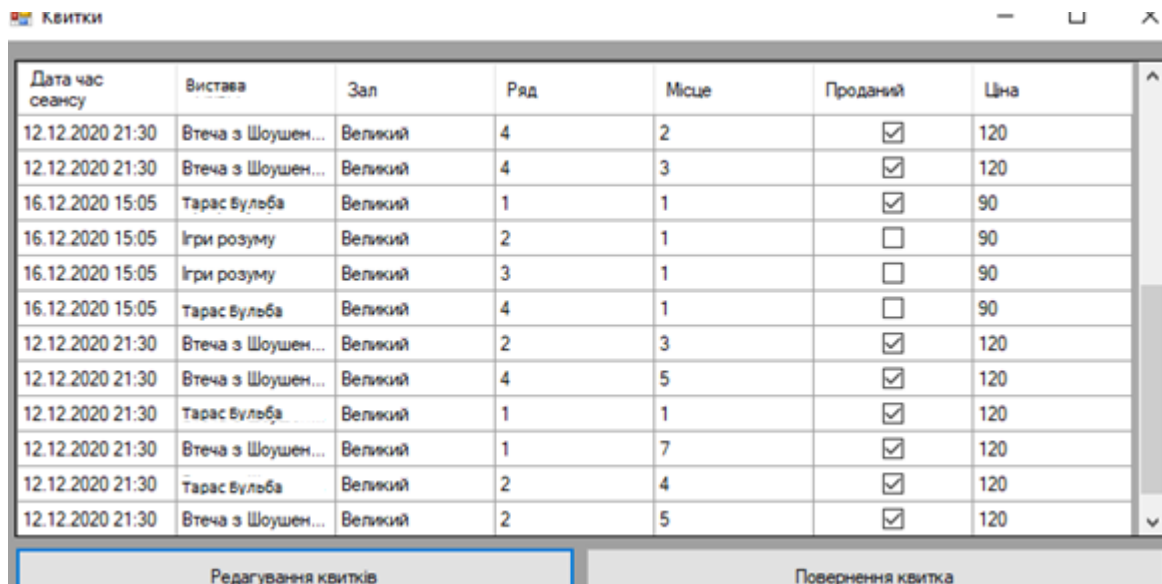
Сортувати

Редагувати

Додати новий

Рисунок 4.4 – Вікно з сеансами, опція облік клієнтів

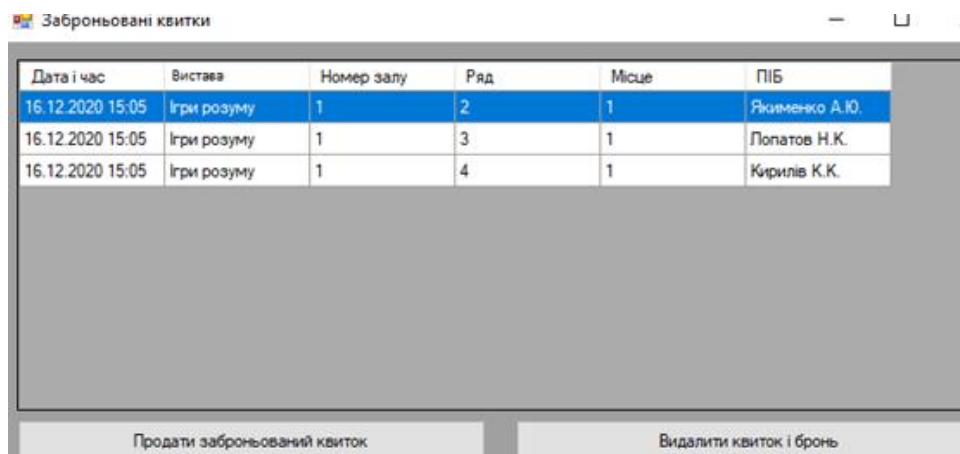
На рисунку 4.5 відображена форма обліку квитків. Форма надає можливість переглядати, редагувати та виконувати функцію повернення квитка.



Дата час сеансу	Вистава	Зал	Ряд	Місце	Проданий	Ціна
12.12.2020 21:30	Втеча з Шоушен...	Великий	4	2	<input checked="" type="checkbox"/>	120
12.12.2020 21:30	Втеча з Шоушен...	Великий	4	3	<input checked="" type="checkbox"/>	120
16.12.2020 15:05	Тарас Бульба	Великий	1	1	<input checked="" type="checkbox"/>	90
16.12.2020 15:05	Ігри розуму	Великий	2	1	<input type="checkbox"/>	90
16.12.2020 15:05	Ігри розуму	Великий	3	1	<input type="checkbox"/>	90
16.12.2020 15:05	Тарас Бульба	Великий	4	1	<input type="checkbox"/>	90
12.12.2020 21:30	Втеча з Шоушен...	Великий	2	3	<input checked="" type="checkbox"/>	120
12.12.2020 21:30	Втеча з Шоушен...	Великий	4	5	<input checked="" type="checkbox"/>	120
12.12.2020 21:30	Тарас Бульба	Великий	1	1	<input checked="" type="checkbox"/>	120
12.12.2020 21:30	Втеча з Шоушен...	Великий	1	7	<input checked="" type="checkbox"/>	120
12.12.2020 21:30	Тарас Бульба	Великий	2	4	<input checked="" type="checkbox"/>	120
12.12.2020 21:30	Втеча з Шоушен...	Великий	2	5	<input checked="" type="checkbox"/>	120

Рисунок 4.5 – Вікно обліку проданих квитків

Рисунок 4.6 зображає форму заброньованих квитків. Користувач може переглянути таблицю з квитками заброньованими на визначений сеанс клієнтом. Користувач може видалити квиток і бронь або продати заброньований квиток.



Дата і час	Вистава	Номер залу	Ряд	Місце	ПІБ
16.12.2020 15:05	Ігри розуму	1	2	1	Якименко А.Ю.
16.12.2020 15:05	Ігри розуму	1	3	1	Лопатов Н.К.
16.12.2020 15:05	Ігри розуму	1	4	1	Кирилів К.К.

Рисунок 4.6 – Вікно що надає інструменти для додавання заброньованих білетів

Для використання меню адміністративного меню, необхідно здійснити вхід як адміністратор. Користувач може користуватись попередньо згаданими функціями, а також редагувати фільми, зали, персонал кінотеатру та користувачів програми і переглядати звіти.

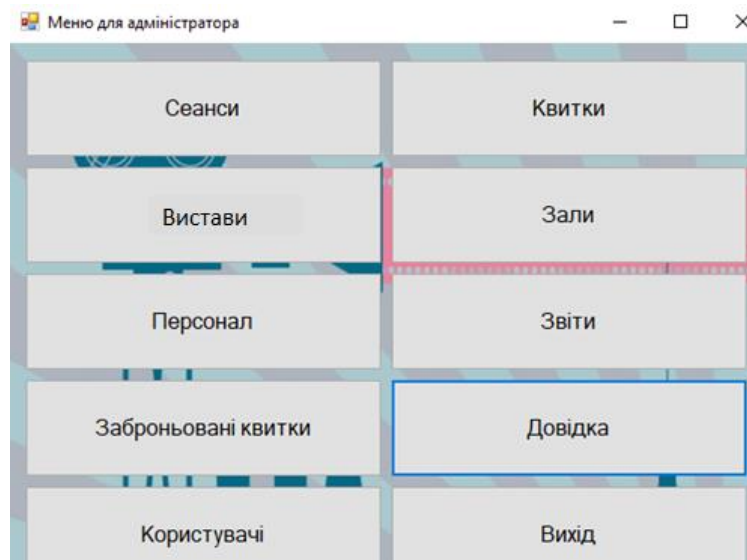


Рисунок 4.7 – Меню адміністратора

На рисунку 4.8 зображене меню «Вистави», у якому користувач має доступ до таблиці з вистав, де вказано жанр, тривалість, рейтинг, рік випуску та таблиць з описом та акторського складу обраного шоу.



Рисунок 4.8 - Меню вистав

Також надається можливість провести пошук та сортування по фільтрам або редагувати та додавати нові шоу.

Назва: Безславні кіпачки

Жанр: комедія, бойовик, драма

Тривалість, хв: 123

Рейтинг: 8.9

Опис: Правила самурая прості: битися на двох мечях, тренуватися по півдобі і не наживатися на неприємності заради почуттів. Але як виконати бодай одне, коли твоє ім'я Тарас Шевченко, твою дівчину мордує поганець, а твій наставник луцює тебе, аби натренувати залізну волю? Доведеться стати самураєм за 2 дні: стріляти «по-македонськи», швидко багаті і нарешті перестати

Акторський склад: Роман Луцький, Сергій Стрельников, Яків Ткаченко, Ген Сето, Катерина Слюсар, Андрій Борис, Андрій Малинович

Рік: 2020

Вибір Додати

Рисунок 4.9 – меню додавання фільму та приклад

Рисунок 4.10 відображає меню перегляду персоналу кінотеатру. Користувач має змогу здійснити пошук або сортування по фільтрам, також додавати та редагувати дані.

ПІБ	Посада	Телефон
Часник А.А.	Прибиральник	+380444658885
Страховус У.Е.	Охоронець	+380296598564

Пошук по:

☐ ПІБ ☐ Посаді ☐ Телефон

Пошук Скинути

Сортування

☒ По зростанню ☐ По спаданню

Сортування Редагування Додати

Рисунок 4.10 – Меню персоналу

Наступний рисунок зображає меню «Звіти», де користувач може вести облік проданих квитків та рух виручки за визначений період.

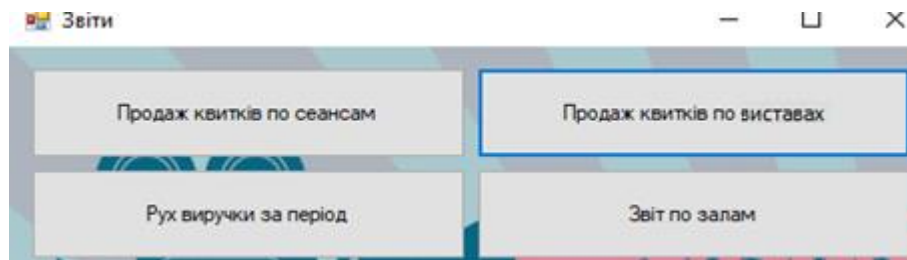


Рисунок 4.11 – Меню опції «Звіти»

Далі можна оцінити статистику проданих білетів на виставу Ромео і Джульєтта.

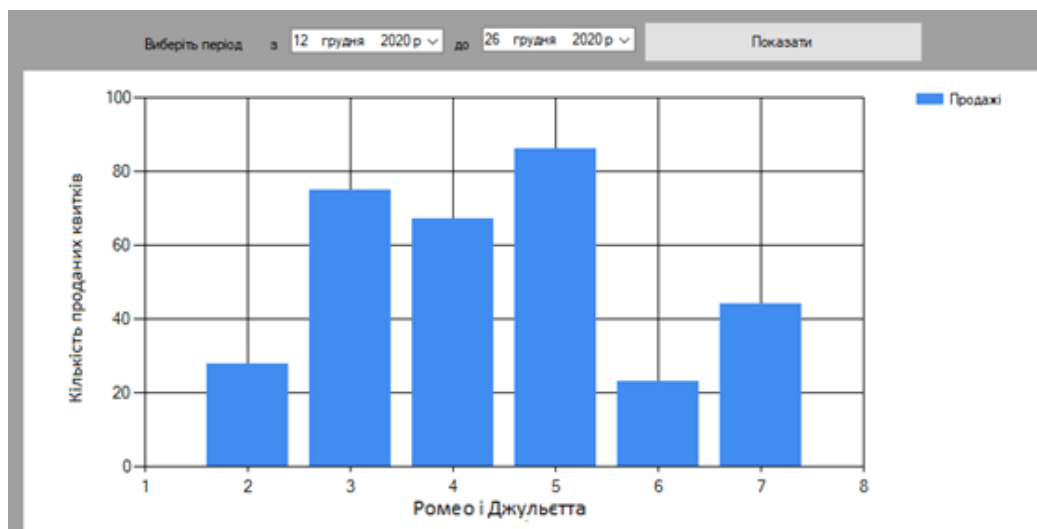


Рисунок 4.12 – Приклад звіту продажу квитків по виставах

Також Адміністратор може викликати справку про програму.

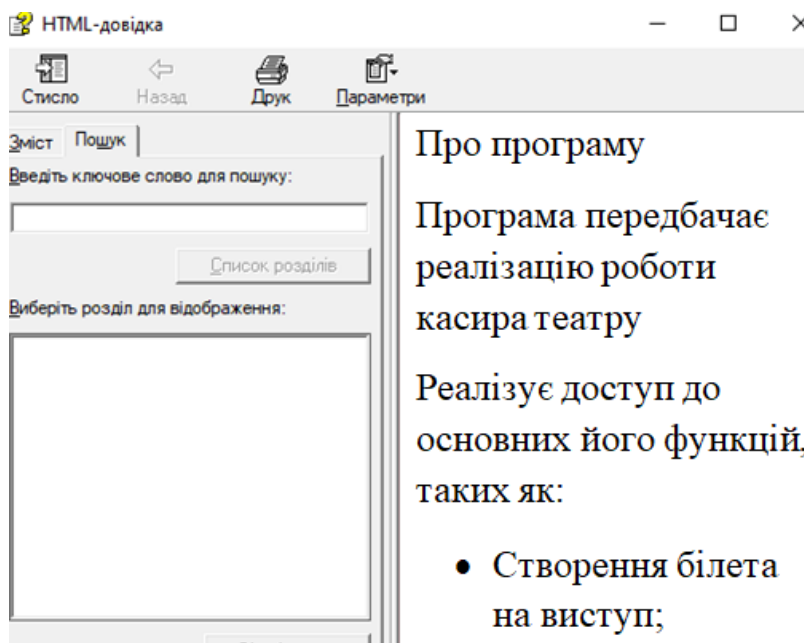


Рисунок 4.13 – Приклад виклику справки програми

ВИСНОВКИ

Під час розробки програмного забезпечення, основна увага була зосереджена на проектування архітектури системи з дотриманням вимог правильної архітектури та з використанням патернів для створення механізмів організації системи. Також система спрямована на реалізацію зрозумілого графічного інтерфейсу та побудові логічної бази даних, що найважливіша для створення інформаційних систем.

Також було розроблено технічну демонстраційну версію додатку «Системи автоматизації роботи касира театру», що дало змогу застосувати отримані знання на практиці через реалізацію проектно-архітектурних прийомів. Написана програма виконує усі передбачені функції. Під час тестування програми, неполадок не виявлено. Результатом виконання курсового проекту є програма що відповідає поставленому завданню, а саме розробити систему що виконуватиме інформаційно-адміністративними маніпуляції для організацій що зайняті в сфері послуг зокрема в театральній сфері. Також створена система відповідає вимогам архітектурної гнучкості, що забезпечує можливість її подальшої модернізації чи реструктуризації.

В процесі роботи досягну наступних результатів:

- дослідив продукти які виконують аналогічну роботу;
- поглибив навички в роботі з Visual Studio, C#, MySQL;
- розробив архітектурний проект реалізації роботи системи відповідно визначених вимог до програми;
- розробив систему автоматизації роботи касира театру.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Сайт програми Servio Tikest [Електронний ресурс] – Режим доступу до ресурсу: <https://expertsolution.com.ua/uk/avtomatizacija-kinoteatrov--teatrov--operi>.
2. Офіційний сайт програми CashFront [Електронний ресурс] – Режим доступу до ресурсу: <https://www.intech.co.ua/product/kopiiaprograma-dlia-kasira-cashfront/>.
3. Visual C# 2008: базовий курс / [К. Уотсон, К. Нейгел, Я. Педерсен та ін.]. – М.: И.Д. Вильямс, 2009
4. Рекомендації щодо проектних рішень на мові C# [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.microsoft.com/ua-ua/visualstudio/get-started/tutorial-projectssolutions?view=vs-2019>.
5. Entity Framework інструкції щодо застосування [Електронний ресурс] – Режим доступу до ресурсу: <https://metanit.com/sharp/entityframework/>
6. Сайт-інструкція з використання системи MySQL [Електронний ресурс] – Режим доступу до ресурсу: <http://www.rldp.ua/mysql/mysql80/index.htm>
7. Яровенко А. Г. Лабораторний практикум з проектування алгоритмів та програм. Навчально-методичний посібник / А. Г. Яровенко. – Вінниця, 2014. – 105 с.
8. Розбір особливостей проектування та моделювання БД [Електронний ресурс] // TextReferat. – 2018. – Режим доступу до ресурсу: <http://ua.textreferat.com/referat-23369-5.html>.

Додатки

Додаток А

Код програмного продукту

Лінк на код програми розміщений на репозиторії GitHub:

<https://github.com/Mozerini/ChashSystem>

Додаток Б

Скрипти створення таблиць бази даних

```
use theatre
```

```
create table Hall
(ID int not null primary key,
Name varchar(200) not null,
Planogram varchar(200) not null
Seat int not null,
Row int not null,
AllSeats int not null
)
```

```
create table Show
(ID int not null identity(1,1) primary key,
Name varchar(200) not null,
Genre varchar(200) not null,
Duration int not null,
Rating float not null,
Description varchar(500) not null,
Actors varchar(450) not null,
Year int not null
)
```

```
create table Session
(ID int not null identity(1,1) primary key,
Datetime datetime not null,
HallNumber int not null foreign key references Hall(ID) on update cascade on
delete cascade,
Show int not null foreign key references Show(ID) on update cascade on delete
cascade,
Seats int not null,
Price bigint not null
)
```

```
create table Ticket
(ID int not null identity(1,1) primary key,
Datetime datetime not null,
Session int not null foreign key references Session(ID) on update cascade on
delete cascade,
Show varchar(250) not null,
HallNum int not null,
```



```
Seat int not null,  
Bought tinyint not null,  
Price bigint not null  
)
```

```
create table Booking  
(ID int not null identity(1,1) primary key,  
Ticket int not null foreign key references Ticket(ID) on update cascade on  
delete cascade,  
Name varchar(150) not null  
)
```

```
create table User  
(ID int not null identity(1,1) primary key,  
Role varchar(50) not null,  
Password varchar(60) not null  
)
```

```
create table Staff  
(ID int not null identity(1,1) primary key,  
Name varchar(150) not null,  
Job varchar(100) not null,  
PhoneNumber varchar(160) not null  
)
```

Додаток В

Технічне завдання

Додаток Г

Рецензія