МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Львівський національний університет імені Івана Франка
Факультет електроніки та комп'ютерних технологій
Кафедра системного проектування

Допустити до захисту

Завідувач кафедри
_____ доцент Шувар Р. Я.
«____»_____ 2023 р.

Курсова робота

Магістр
(освітній ступінь)

Сервіс для покупки білетів на ASP.NET

Виконав:
Студент V курсу групи ФЕІм-12
спеціальності 122 –
Комп'ютерні науки

_____ Крохмальний А. І.

Науковий керівник:
_____ професор. Юзевич В. М.
«____»_____ 2023 р.

Львів 2023

АНОТАЦІЯ

Під час виконання даної роботи було визначено актуальність програми, проведено аналіз конкурентів на ринку та поставлені чіткі задачі для реалізації. Зокрема, створено сервіс для покупки білетів на кіносеанси в конкретному кінотеатрі. У цьому прикладі враховано функціонал з широкими можливостями як для адміністратора, так і для користувача.

В результаті програма була успішно створена та протестована. Інтерфейс інтуїтивно зручний та зрозумілий для користувача. Програма містить узагальнений варіант заявленого функціоналу. Всі функції, які відповідають функціоналу, працюють справно і відповідні дані динамічно змінюються при заданих командах.

SUMMARY

During the execution of this work, the relevance of the program was determined, an analysis of competitors on the market was conducted, and clear tasks for implementation were set. In particular, a service was created for buying tickets for movie screenings in a specific cinema. This example takes into account functionality with wide possibilities for both the administrator and the user.

As a result, the program was successfully created and tested. The interface is intuitive and easy to understand for the user. The program contains a generalized version of the declared functionality. All the functions that correspond to the functionality work properly and the corresponding data changes dynamically with the given commands.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ **3**
ВСТУП **3**
1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ **5**
1.1. Актуальність задачі 5
1.2. Аналіз наявних сервісів та конкурентів 7
1.3. Постановка задачі. 9
2. ПРОЕКТУВАННЯ ВЕБ ЗАСТОСУНКУ **12**
2.1. Аналіз обраних інструментів 12
2.1.1. Мова програмування C# 12
2.1.2. Технологія ASP.NET 13
2.1.3. Microsoft SQL Server 15
2.1.4. Bootstrap 17
2.2. Архітектура додатку 18
2.3 Діаграма баз даних 20
3. ПРОГРАМНА РЕАЛІЗАЦІЯ ЗАСТОСУНКУ **24**
3.1. Структура проекту 24
3.2. Розробка баз даних 26
3.3. Тестування програми 32
ВИСНОВКИ **37**
ПЕРЕЛІК ПОСИЛАНЬ **41**

ПЕРЕЛІК СКОРОЧЕНЬ

CLR - Common Language Runtime

CLI - Common Language Infrastructure

JIT - Just-In-Time

RDBMS - Relational Database Management System

SQL - Structured query language

T-SQL - Transact-SQL

ACID - Atomicity, Consistency, Isolation, Durability

HTML - HyperText Markup Language

CSS - Cascading Style Sheets

JS - JavaScript

ERD - Entity Relationship diagram

DSD - Data Structure Diagrams

DFD - Data Flow Diagrams

ВСТУП

У сучасному світі зростає популярність онлайн-сервісів, які надають можливість придбати білети на різні події безпосередньо через Інтернет. Такі сервіси забезпечують зручність, швидкість та доступність для користувачів, дозволяючи їм легко знайти, порівняти і придбати білети на кіносеанси, культурні заходи, театральні вистави та інші події.

Метою даної курсової роботи є розробка сервісу для покупки білетів, який буде забезпечувати зручну та ефективну процедуру пошуку та придбання білетів для користувачів. Цей сервіс має на меті спростити процес купівлі квитків, забезпечити широкий вибір подій, а також надати зручні функції оплати.

Проведення даного дослідження має значення в контексті сучасних тенденцій розвитку онлайн-сервісів та електронної комерції. Зростання популярності цифрових платформ та зручність їх використання змушують багато компаній звертати увагу на розробку інтернет-сервісів для забезпечення максимального комфорту та задоволення потреб клієнтів. Розробка сервісу для покупки білетів відповідає цим тенденціям та має потенціал стати важливим інструментом для організацій, які орієнтуються на продаж квитків на події.

У рамках цієї роботи будуть розглянуті основні етапи розробки сервісу, такі як аналіз конкурентів на ринку, проектування архітектури та інтерфейсу, реалізація функціональності, тестування та впровадження. Крім того, будуть розглянуті основні проблеми та виклики, пов'язані з розробкою сервісу для покупки білетів, а також запропоновані можливі шляхи їх вирішення.

Результатом цієї курсової роботи буде готовий прототип сервісу для покупки білетів, який буде включати основні функціональні можливості для зручного та ефективного придбання білетів на різноманітні події. Після завершення розробки цей сервіс може бути використаний як основа для подальшої розробки та комерційного використання. Крім того, отримані результати можуть послужити основою для подальших досліджень та

вдосконалення сервісу з метою задоволення потреб користувачів щодо покупки білетів на події.

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Актуальність задачі

Актуальність створення сервісу для покупки білетів є надзвичайно важливою в контексті сучасних тенденцій розвитку цифрових технологій та електронної комерції. Онлайн-сервіси для придбання білетів надають користувачам зручність, швидкість і доступність у процесі замовлення квитків на різноманітні події.

Одним з головних факторів актуальності є зручність і ефективність, які надає сервіс для покупки білетів. Користувачам більше не потрібно витрачати час на фізичний візит до каси або торгової точки для придбання білетів. Вони можуть з легкістю знайти необхідну подію, переглянути розклад, порівняти ціни та придбати квитки, просто за декілька кліків мишкою або дотиком екрану.

Також важливим фактором є доступність. Онлайн-сервіси для покупки білетів дозволяють користувачам мати доступ до широкого спектру подій і виставок, які можуть відбуватися не тільки в їхньому регіоні, але й у будь-якій частині світу. Вони можуть легко придбати квитки на концерти улюблених виконавців, спортивні змагання, кіносеанси, театральні вистави та багато інших заходів, незалежно від місця проживання, якщо це дозволяє функціонал додатку.

У зв'язку зі зростанням популярності онлайн-сервісів, попит на такі платформи для покупки білетів також значно збільшується. Компанії, які організовують події, та постачальники квитків усвідомлюють важливість присутності у цьому цифровому середовищі, оскільки онлайн-сервіси для покупки білетів дозволяють їм залучити більше клієнтів і розширити свою аудиторію. Також це надає можливість покращити процес продажу квитків, оптимізувати управління складною логістикою та забезпечити ефективну обробку та облік проданих білетів.

Крім того, створення сервісу для покупки білетів відкриває нові можливості для збільшення прибутків і заробітку. Це полягає у використанні різних стратегій ціноутворення, продаж додаткових послуг та пакетів,

просуванні спеціальних пропозицій та акцій, а також співпраці з іншими організаціями для забезпечення зручності та додаткових переваг для користувачів.

Покупки онлайн наздоганяють своєю кількістю оффлайн торгівлю в усьому світі. Неважко припустити, що незабаром інтернет-транзакції і зовсім витіснять розрахунки готівкою, які набагато важче контролювати. Не вдаючись в конспірологію, потрібно визнати один факт: покупки в інтернеті дійсно стали безпечнішими, ніж покупки оффлайн [1].

Отже, створення сервісу для покупки білетів має велике значення в сучасному світі, де швидкість, зручність та доступність є важливими чинниками для користувачів. Цей сервіс допоможе забезпечити ефективну та зручну процедуру покупки квитків на різноманітні події, забезпечуючи задоволення потреб і вимог сучасних споживачів.

1.2. Аналіз наявних сервісів та конкурентів

Ринок онлайн сервісів для покупки білетів не стоїть на місці, все нові і нові додатки та сервіси для покупки білетів на різні культурні заходи, починаючи з кіносеансів закінчуючи виставками з’являються у світі.

Їх можна розділити на 3 великі групи:

1. Онлайн додатки конкретного місця, наприклад кінотеатру.
2. Сервіси для покупки білетів по всій мережі закладів, які можуть бути як в одному місті, так і в багатьох.
3. Сервіси для покупки білетів незалежно від конкретного закладу чи мережі.

До першої групи можна віднести сайт “Кінотеатр імені О.Довженка” [2], цей застосунок акутальний тільки для конкретного кінотеатру, тому має

невеликий спектр послуг для користувачів і досить скромний функціонал.

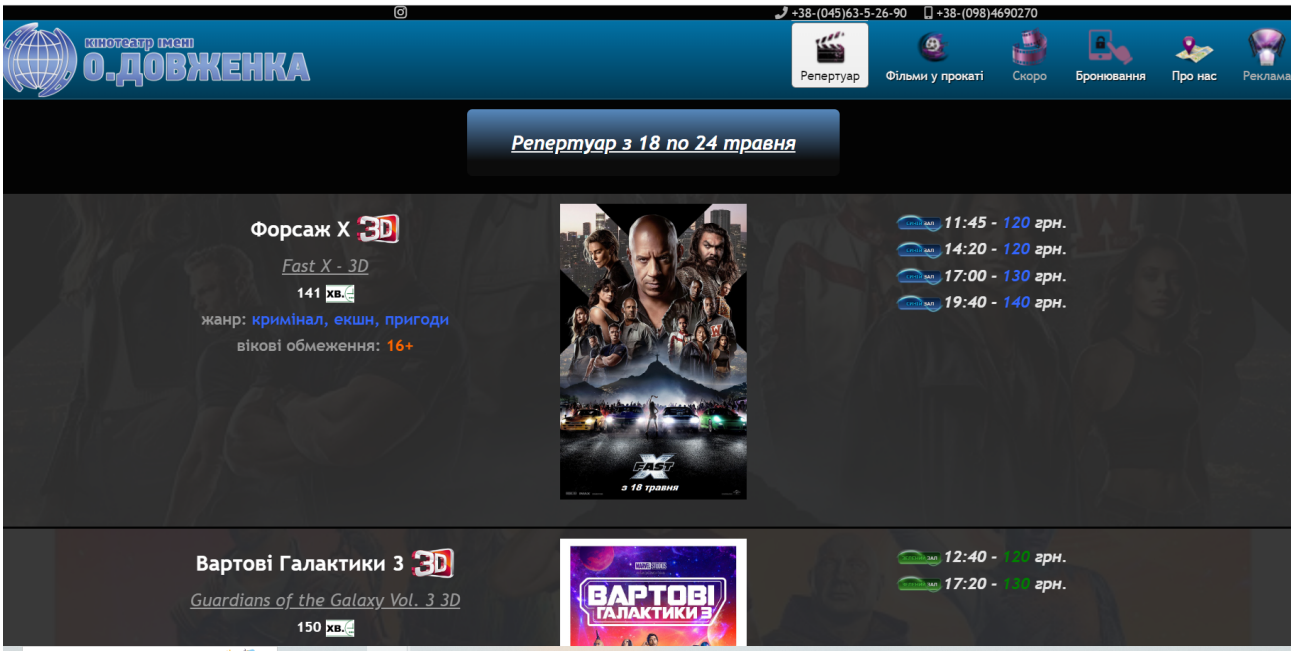

Рис. 1.2.1 - вебсайт “Кінотеатр імені О.Довженка”

Ще відносно недавно, декілька років тому не було потреби в сервісах з покупки білетів, але в останнє десятиліття почався значних ріст всієї сфери надання послуг, і звісно цифровізації, яка і є основною причиною росту.

Представників такого формату стає все менше і менше, оскільки вони не витримують конкуренцію в глобальних мереж і відстають в технічній складові. Це також представлено і функціоналом додатку, який програє своїм більшим конкурентам.

До другої групи можна віднести мережеві сайти для покупки білетів, такі як “«Стрічка» кіно та кава” [3], та “Планета Кіно” [4]. Ці програма є в версії і десктопного додатку, і веб застосунку. За допомогою цих застосунків можна шукати, та купувати квитки по всім кінотеатрам мереж.

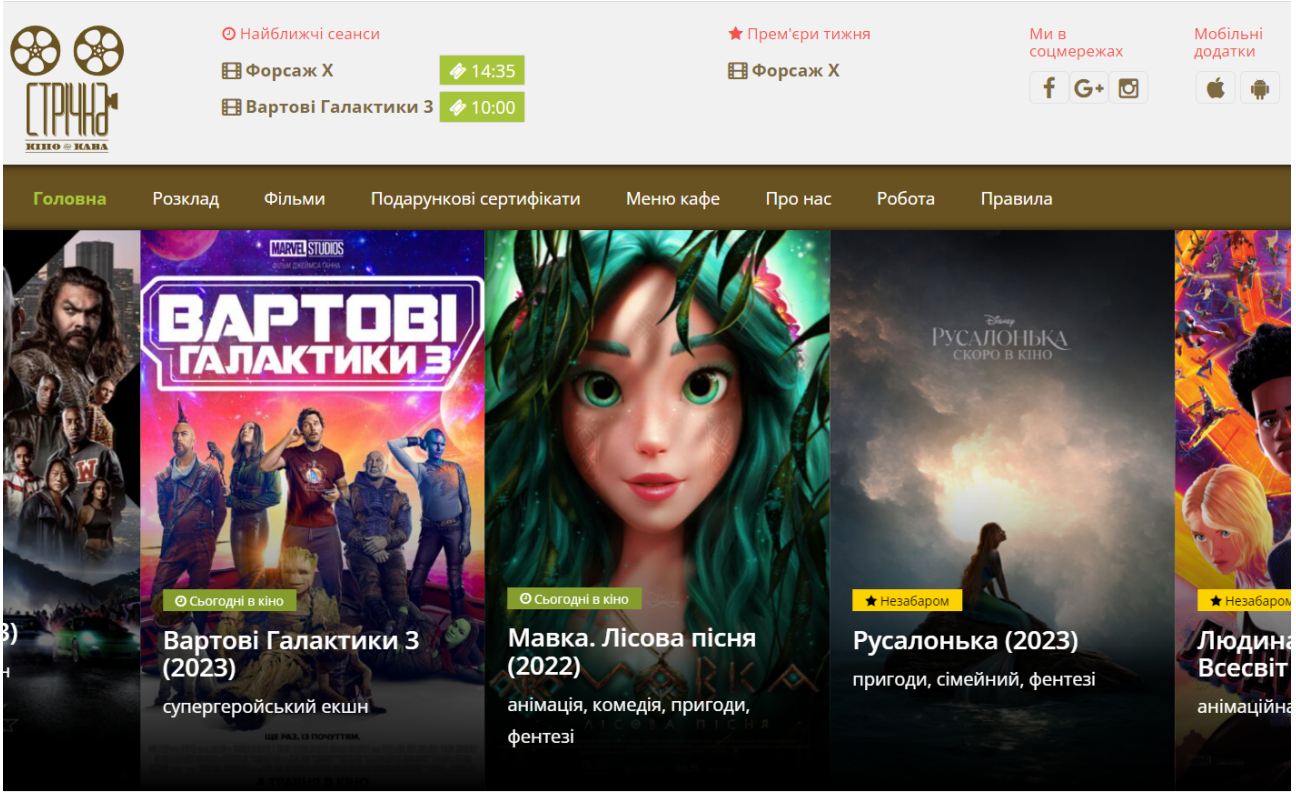

Рис. 1.2.2 - вебсайт Стрічка» кіно та кава

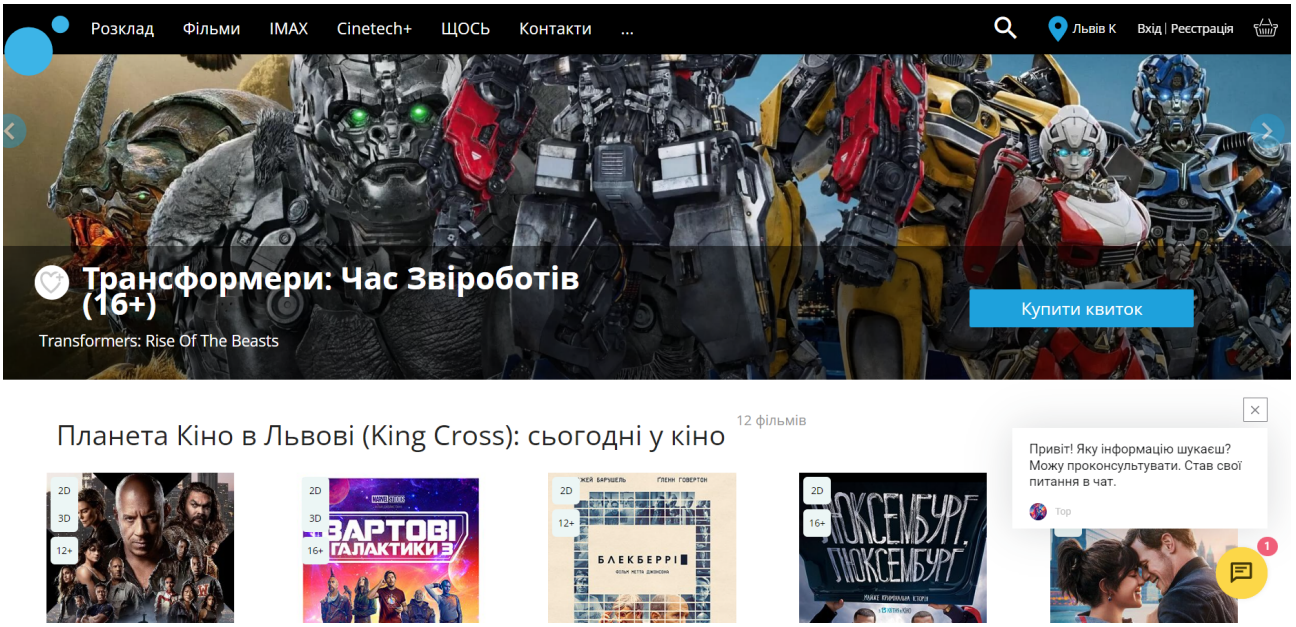

Рис. 1.2.3 - вебсайт “Планета кіно”

Мінусом цього додатку є все-таки обмеження однією мережею, проте це і є бізнесмоделлю і логічною закономірністю, функціонал таких сервісів залишається досить обширним.

Основний функціонал:

- створення баз даних
- перегляд різної статистики
- перегляд куплених білетів
- можливість безпечної покупки
- можливість створення профілю
- можливість пошуку фільмів

До 3 групи можна віднести сервіс “Вкіно” [5]. Він надає найобширніший функціонал і є прямим конкурентом додатку представленого в курсовій. Мінусом додатку є невелика популярність і обмеження лише сферою кіно.

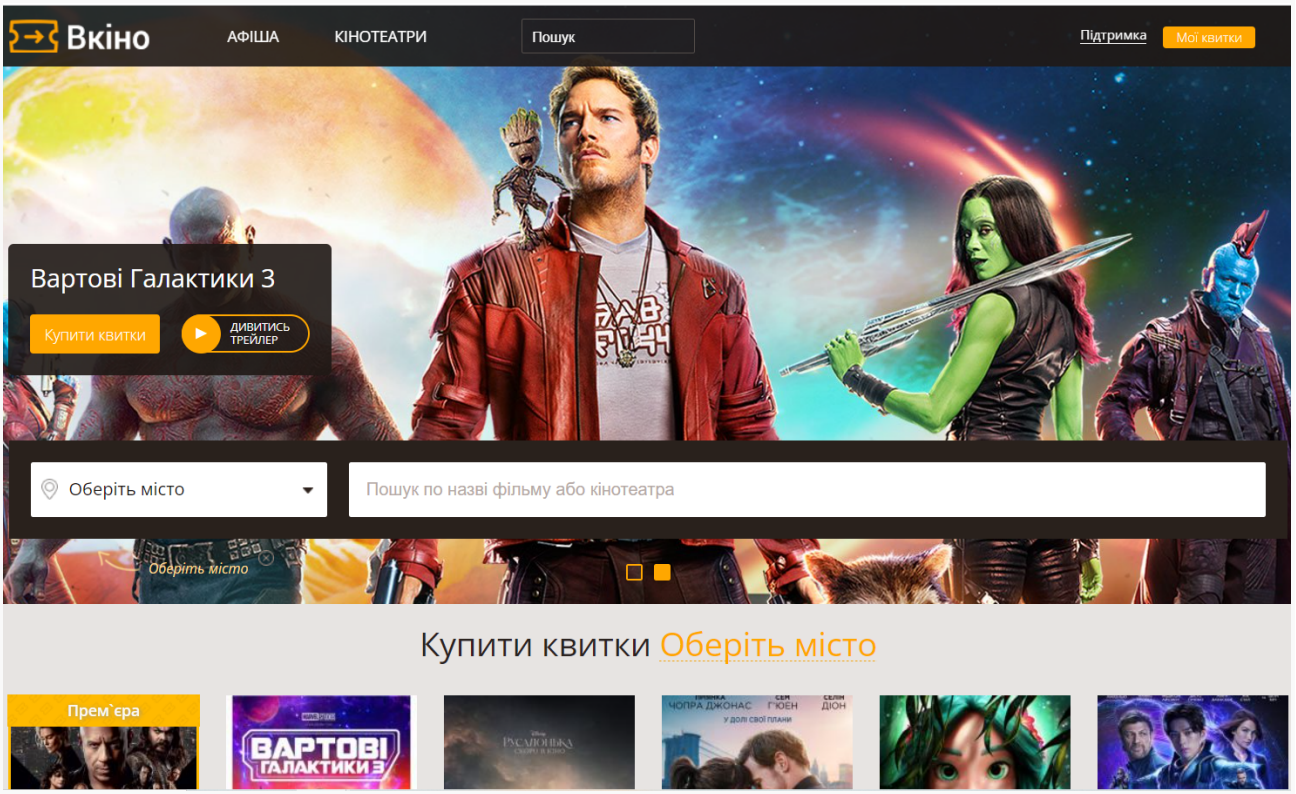

Рис. 1.2.4 - вебсайт “ВКіно”

Загалом сайт надає достатньо функціоналу для зручного користування, проте бачимо що є куди рости, адже відсутні можливості, для пошуку інших культурних подій.

1.3. Постановка задачі.

Цей сервіс, для покупки білетів повинен підвищити ефективність і якість праці робітників(адміна), а також забезпечити відвідувача(юзера) необхідними йому даними і ресурсами для максимально зручного користування сервісом.

У цьому веб-застосунку мають бути реалізовані такі функції:

1. Інтерфейс повинен бути не перевантаженим та оформленим у зрозумілому та інтуїтивному стилі, щоб користувачу було комфортно використовувати сервіс.
2. Можливість логінитись і розлогінитись.
3. Різний доступ до функціоналу застосунку в залежності від класу користувача(visiter, member, admin).
4. Можливість перегляду доступних фільмів для всіх відвідувачів сервісу.
5. Можливість замовляти квиток, або декілька.
6. Зручний інтерфейс admin, який дозволяє керувати базою фільмів, додавати нові, модифікувати наявні, або видаляти їх

Проаналізувавши наявні аналоги, можна виокремити певні переваги, які будуть в даному сервісі:

- Зручний і інтуїтивний інтерфейс;
- Розширені можливості для бібліотекаря(admin);
- Розширені можливості для користувача(user);
- Можливість покупки квитка через PayPal;
- Релевантна цінова політика;

2. ПРОЕКТУВАННЯ ВЕБ ЗАСТОСУНКУ

2.1. Аналіз обраних інструментів

2.1.1. Мова програмування C#

C# - це досить нова, об'єктно-орієнтована та строготипізована мова програмування. C# дає можливість розробникам створювати багато типів надійних і безпечних програм, які працюють у .NET. C# як мова має певні коріння в сімействі мов C.

C# — об’єктно-орієнтована, компонентно-орієнтована мова програмування. C# надає мовні конструкції для безпосередньої підтримки цих концепцій, що робить C# природною мовою для створення та використання програмних компонентів. З моменту свого створення C# додав функції для підтримки нових робочих навантажень і новітніх практик проектування програмного забезпечення. За своєю суттю C# є об’єктно-орієнтованою мовою. Ви визначаєте типи та їх поведінку [6].

Програми C# виконуються на .NET, віртуальній системі виконання, що називається середовищем загальномовного виконання (CLR), і наборі бібліотек класів. CLR — це реалізація Microsoft спільної мовної інфраструктури (CLI), міжнародного стандарту. CLI є основою для створення середовищ виконання та розробки, у яких мови та бібліотеки безперебійно працюють разом [6].

Вихідний код, написаний мовою C#, компілюється в проміжну мову (IL) , яка відповідає специфікації CLI. Код IL і ресурси, такі як растрові зображення та рядки, зберігаються в збірці, зазвичай із розширенням .dll . Збірка містить маніфест, який надає інформацію про типи, версію та культуру збірки [6].

Коли програма C# виконується, збірка завантажується в CLR. CLR виконує компіляцію Just-In-Time (JIT) для перетворення коду IL у власні машинні інструкції. CLR надає інші служби, пов’язані з автоматичним збиранням сміття, обробкою винятків і керуванням ресурсами. Код, який виконує CLR, іноді називають «керованим кодом». «Некерований код» компілюється в рідну машинну мову, націлену на певну платформу [6].

Сумісність мов є ключовою особливістю .NET. Код IL, створений компілятором C#, відповідає специфікації загального типу (CTS). Код IL, згенерований з C#, може взаємодіяти з кодом, який було згенеровано з версій .NET F#, Visual Basic, C++. Існує понад 20 інших CTS-сумісних мов. Одна збірка може містити декілька модулів, написаних різними мовами .NET. Типи можуть посилатися один на одного, як якщо б вони були написані однією мовою [6].

2.1.2. Технологія ASP.NET

ASP.NET — технологія створення вебзастосунків і вебсервісів від компанії Майкрософт. Вона є складовою частиною платформи Microsoft.NET і розвитком старішої технології Microsoft ASP. На цей час останньою версією цієї технології є ASP.NET Core 6.0 [7].

ASP.NET зовні багато в чому зберігає схожість із старшою технологією ASP, що дає змогу розробникам відносно легко перейти на ASP.NET. У той же час внутрішній устрій ASP.NET істотно відрізняється від ASP, оскільки вона заснована на платформі. NET і, отже, використовує всі нові можливості, що надаються цією платформою [8].

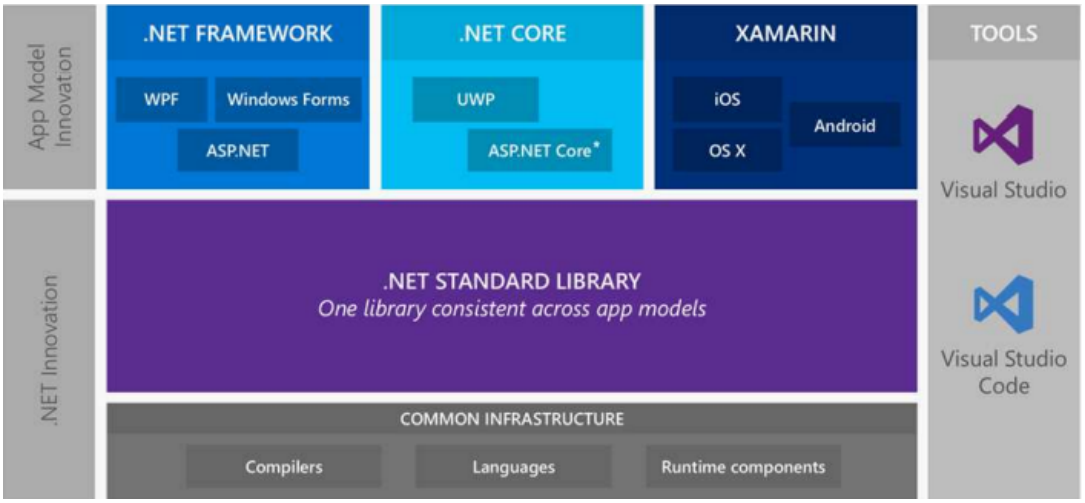

Рис. 2.1.2.1 - середовище .NET

Основні особливості ASP.NET:

1. Стабільність та надійність: ASP.NET базується на стабільній та надійній платформі .NET, що дозволяє створювати веб-додатки, які працюють стабільно та мають високий рівень надійності.
2. Масштабованість: ASP.NET надає можливість масштабування веб-додатків, що дозволяє вирішувати вимоги великих обсягів трафіку та кількості користувачів. Він підтримує розподілену архітектуру та можливість використання серверів додатків.
3. Широкий набір функціональності: ASP.NET надає багатий набір вбудованих функціональних можливостей, таких як автентифікація та авторизація, кешування, маршрутизація, керування станом та інші. Це спрощує розробку складних веб-додатків і полегшує підтримку їх функціональності.
4. Підтримка мов програмування: ASP.NET підтримує багато мов програмування, включаючи C#, Visual Basic .NET, F# та інші. Це дозволяє розробникам використовувати мову, з якою вони найбільш знайомі та зручні.
5. Шаблони веб-сторінок: ASP.NET використовує концепцію шаблонів веб-сторінок, що дозволяє відокремити логіку від представлення. Це полегшує розробку та підтримку веб-інтерфейсу та дозволяє використовувати пере використання коду та дизайну.
6. Інтеграція зі сторонніми сервісами: ASP.NET має підтримку для інтеграції зі сторонніми сервісами, такими як бази даних, системи електронної комерції, платіжні шлюзи та інші. Це дозволяє побудувати комплексні веб-додатки, які взаємодіють з різними системами та сервісами.

ASP.NET надає потужні інструменти для розробки веб-додатків, що включають в себе високу продуктивність, безпеку, легкість розгортання та підтримку великої спільноти розробників. Він є відмінним вибором для будь-яких веб-проектів, від невеликих сайтів до великих підприємственних додатків.

2.1.3. Microsoft SQL Server

Microsoft SQL Server — це система керування реляційною базою даних (RDBMS), яка в свою чергу підтримує широкий спектр додатків для бізнес-аналітики, обробки транзакцій та аналітики в корпоративних IT-середовищах. Microsoft SQL Server є однією з найбільш популярних технологій баз даних, поряд з IBM DB2 і Oracle Database [9].

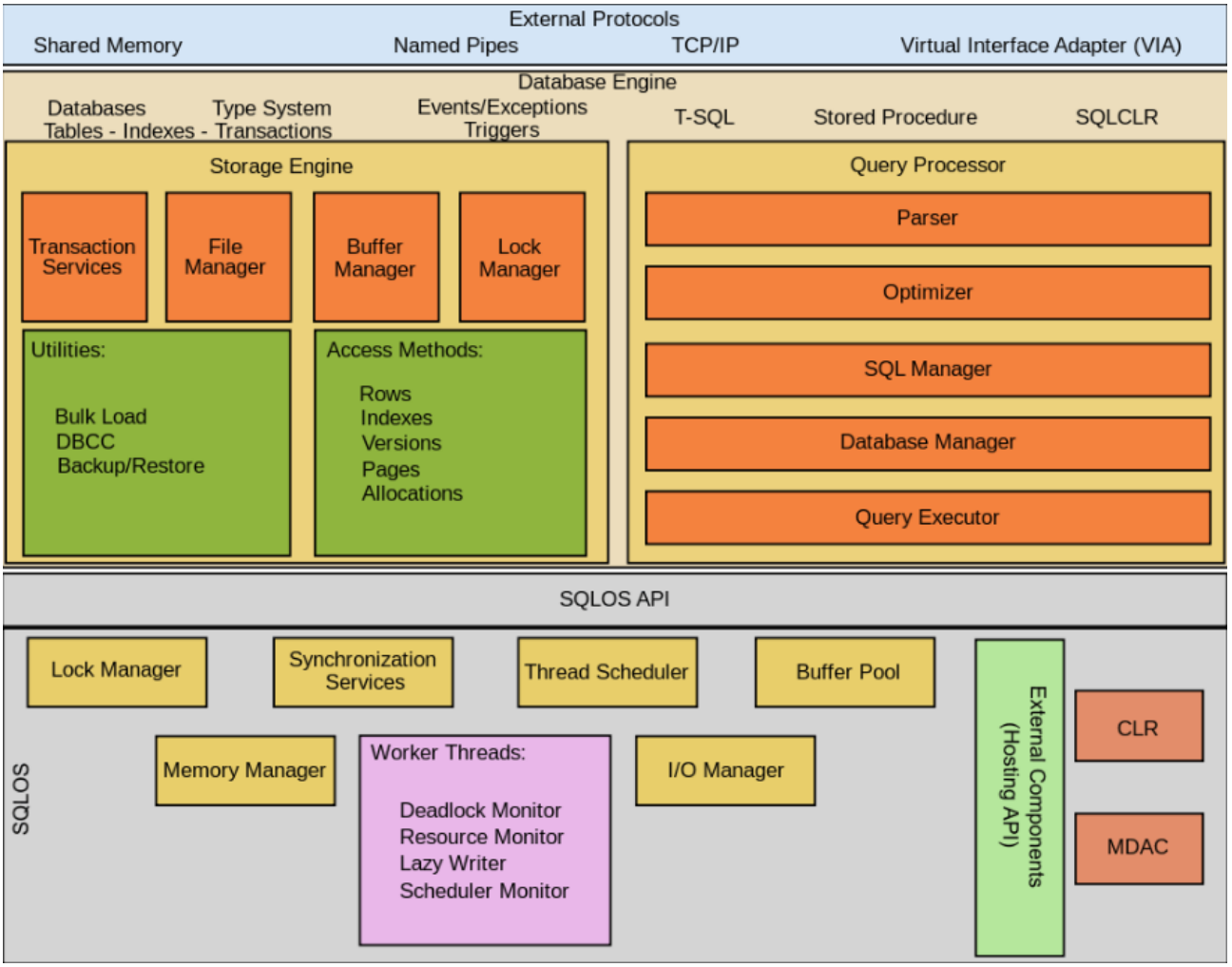

Рис. 2.1.3.1 - архітектура SQL Server

Особливості Microsoft SQL Server:

1. Реляційна модель даних: SQL Server базується на реляційній моделі даних, де дані представлені у вигляді таблиць зі зв'язками між ними. Це дозволяє ефективно організовувати та керувати структурованими даними.
2. Потужність та масштабованість: SQL Server підтримує великі обсяги даних та високі навантаження. Він забезпечує широкі можливості для масштабування баз даних, включаючи можливість розподілення даних на кілька серверів.
3. Управління транзакціями: SQL Server підтримує ACID (Atomicity, Consistency, Isolation, Durability) властивості транзакцій. Це забезпечує надійність та цілісність даних, дозволяючи виконувати групу операцій як єдину транзакцію.
4. Мова запитів SQL: SQL Server використовує стандартну мову запитів SQL (Structured Query Language) для взаємодії з базою даних. SQL дозволяє виконувати різноманітні операції, такі як вставка, оновлення, видалення та вибірка даних з бази.
5. Безпека даних: SQL Server надає широкий спектр функцій та інструментів для захисту даних. Він підтримує автентифікацію, авторизацію, шифрування даних, аудит, контроль доступу та інші механізми безпеки.
6. Інтеграція зі засобами Microsoft: SQL Server має глибоку інтеграцію з іншими продуктами та рішеннями Microsoft, такими як Visual Studio, Azure, Power BI, SharePoint та інші. Це дозволяє легко використовувати SQL Server у комплексних рішеннях та екосистемі Microsoft.
7. Аналітика та бізнес-інтелект: SQL Server надає розширені можливості для аналітики даних та роботи з бізнес-інтелектом. Він підтримує OLAP (Online Analytical Processing), інструменти звітності, аналітичні функції та інтеграцію з інструментами аналізу даних.

Microsoft SQL Server є потужним та надійним інструментом для управління базами даних. Він дозволяє організаціям ефективно керувати своїми даними, забезпечувати безпеку та масштабованість, а також використовувати дані для прийняття важливих бізнесових рішень.

2.1.4. Bootstrap

Bootstrap - це безкоштовна бібліотека веб-розробки з відкритим вихідним кодом. Вона призначений для полегшення процесу веб-розробки адаптивних веб-сайтів, орієнтованих на різного роду пристрої, надаючи набір синтаксису для дизайну шаблонів [11].

Особливості Bootstrap:

1. Адаптивний дизайн: Bootstrap підтримує адаптивний дизайн, що означає, що веб-сторінки, побудовані з використанням Bootstrap, автоматично адаптуються до різних розмірів екранів, включаючи комп'ютери, планшети та мобільні пристрої. Це дозволяє забезпечити оптимальний користувацький досвід на різних пристроях.
2. Готові компоненти: Bootstrap має велику бібліотеку готових компонентів, таких як кнопки, форми, навігаційні панелі, картки, модальні вікна та багато інших. Ці компоненти можна легко використовувати та налаштовувати, що прискорює розробку інтерфейсу та полегшує підтримку стилів.
3. Гнучкість і налаштування: Bootstrap надає багато можливостей для налаштування та розширення. Завдяки використанню Sass (розширеної версії CSS) і можливостям кастомізації, розробники можуть швидко змінювати стилі, кольори, шрифти та інші аспекти фреймворку, щоб відповідати вимогам свого проекту.
4. Хороша документація: Bootstrap має докладну та зрозумілу документацію, яка надає вичерпні пояснення та приклади використання кожного компонента і функціональності. Це дозволяє розробникам швидко засвоювати фреймворк та використовувати його ефективно.
5. Підтримка кроссбраузерності: Bootstrap гарантує сумісність з різними веб-браузерами, включаючи Chrome, Firefox, Safari, Edge та Internet

Explorer. Це дозволяє розробникам створювати веб-додатки, які працюють на різних платформах та браузерах без проблем.

Bootstrap є вельми корисним інструментом для швидкої та ефективної розробки стильних та адаптивних веб-додатків. Він спрощує процес розробки, забезпечує єдність дизайну та полегшує підтримку сторінок. Завдяки своїй популярності та великій спільноті розробників, Bootstrap є надійним вибором для будь-якого проекту веб-розробки.

2.2. Архітектура додатку

Архітектура інформаційної системи - це концепція, що визначає модель, структуру, функції які будуть виконуватись і взаємозв'язок компонентів інформаційної системи.

Багаторівнева архітектура клієнт-сервер - це різновид такої архітектури як клієнт-сервер, в якому функція обробки даних винесена на збереження, обробки і представлення даних для більш ефективного використання можливостей серверів і клієнтів [12].

1. Клієнт-серверна архітектура є однією з основних архітектурних моделей для розробки додатків, особливо в контексті мережевих систем. У цій архітектурі система розбивається на дві основні складові: клієнтську частину і серверну частину, які взаємодіють між собою.
2. Клієнт: Клієнт - це програмне забезпечення або пристрій, що надає інтерфейс для взаємодії користувача з додатком. Клієнт може бути веб-браузером, мобільним додатком, настільною програмою або будь-яким іншим засобом, який взаємодіє з сервером для отримання даних або надсилання запитів.
3. Сервер: Сервер - це комп'ютер або система, яка надає послуги та обробляє запити від клієнтів. Сервер має доступ до ресурсів (таких як база даних, файлові системи, додаткові обчислювальні потужності) і виконує необхідні дії для обробки запитів та надання відповіді клієнту.

4. Комунікація: Клієнт та сервер взаємодіють за допомогою мережевого протоколу, такого як HTTP, TCP/IP або іншого протоколу передачі даних. Клієнт надсилає запити на сервер, а сервер обробляє ці запити та надсилає відповіді клієнту.
5. Розподілена обробка: У клієнт-серверній архітектурі обробка даних та бізнес-логіка можуть бути розподілені між клієнтом та сервером. Наприклад, клієнт може відповідати за відображення інтерфейсу користувача, валідацію даних, а сервер - за обробку запитів, збереження даних та виконання складніших операцій.
6. Масштабованість: Клієнт-серверна архітектура дозволяє гнучко масштабувати систему, дозволяючи додавати нові клієнти або сервери в залежності від потреб. Це дозволяє обробляти більше запитів та розподіляти навантаження між серверами.

Клієнт-серверна архітектура є широко використовуваною у багатьох додатках, включаючи веб-додатки, мобільні додатки та корпоративні системи. Вона дозволяє розділити функціональність та відповідальність між клієнтом та сервером, що сприяє покращенню швидкодії, масштабованості та підтримці.

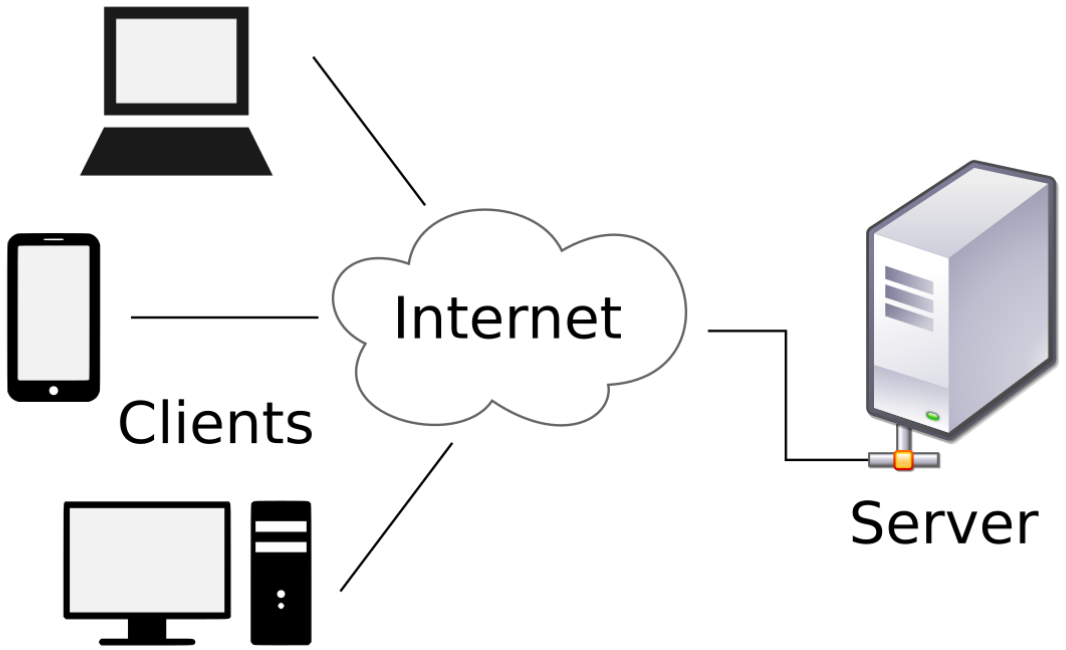

Рис. 2.2.1 - Модель клієнт-сервер

2.3 Діаграма баз даних

База даних є одним з ключових компонентів практично будь-якої інформаційної системи. Вона представляє собою організовану колекцію структурованих даних, які зберігаються у електронній формі на комп'ютерній системі. База даних виконує важливі функції зберігання та управління інформацією, що дозволяє зберігати дані користувачів, відомості про книги, авторів та видавців, а також формувати звіти і надавати рекомендації користувачам.

Діаграма зв’язків між об’єктами (ER) — це тип блок-схеми, яка ілюструє, як «суб’єкти», такі як люди, об’єкти або поняття, пов’язані один з одним у системі. Діаграми ER найчастіше використовуються для проектування або налагодження реляційних баз даних у сферах програмної інженерії, бізнес-інформаційних систем, освіти та досліджень. Також відомі як ERD або моделі ER, вони використовують певний набір символів, таких як прямокутники, ромби, овали та сполучні лінії, щоб відобразити взаємозв’язок

сутностей, відносин та їхніх атрибутів. Вони відображають граматичну структуру з сутностями як іменники, а відносини як дієслова [14].

Діаграми зв'язку сутностей (ER-діаграми) мають широкий спектр застосувань, які охоплюють різні аспекти розробки та управління базами даних. Ось кілька основних варіантів використання ER-діаграм:

- Проектування баз даних: ER-діаграми використовуються для моделювання та проектування реляційних баз даних з точки зору логіки та бізнес-правил. Вони допомагають визначити структуру бази даних, включаючи сутності, атрибути та зв'язки між ними. ER-діаграми допомагають уточнити вимоги до бази даних і створити логічну модель даних, яка буде використовуватися для подальшої реалізації фізичної моделі.
- Усунення несправностей у базі даних: ER-діаграми використовуються для аналізу існуючих баз даних з метою виявлення та вирішення проблем у логіці або розгортанні. Вони дозволяють візуалізувати структуру бази даних та взаємозв'язки між сутностями, що сприяє виявленню проблемних областей та розробці відповідних виправлень.
- Бізнес-інформаційні системи: ER-діаграми використовуються для проектування або аналізу реляційних баз даних, які використовуються в бізнес-процесах. Вони дозволяють описати структуру та взаємозв'язки даних, що використовуються у бізнес-процесах, та забезпечують підставу для оптимізації процесів, полегшення доступу до інформації та покращення результатів.
- Реінжиніринг бізнес-процесів (BPR): ER-діаграми допомагають аналізувати бази даних, що використовуються при реінжинірингу бізнес-процесів, та моделювати нові конфігурації бази даних. Вони дозволяють ідентифікувати потенційні вдосконалення та визначити оптимальну структуру даних для покращення бізнес-процесів.
- Освіта: ER-діаграми використовуються для планування структур даних у освітніх цілях. Вони можуть використовуватися для моделювання та

вивчення реляційних баз даних, а також для підготовки студентів до роботи з базами даних у практичному середовищі.

- Дослідження: ER-діаграми можуть бути використані в дослідницьких проектах, де структуровані дані є ключовим елементом. Вони дозволяють моделювати та аналізувати дані, що дозволяє здійснювати дослідження та аналізувати результати.

Загалом, ER-діаграми є потужним інструментом для моделювання та управління базами даних у різних сферах застосування, починаючи від проектування до аналізу та оптимізації бізнес-процесів. Вони допомагають створювати логічні та фізичні моделі даних, а також забезпечують зручну візуалізацію та аналіз структури бази даних.

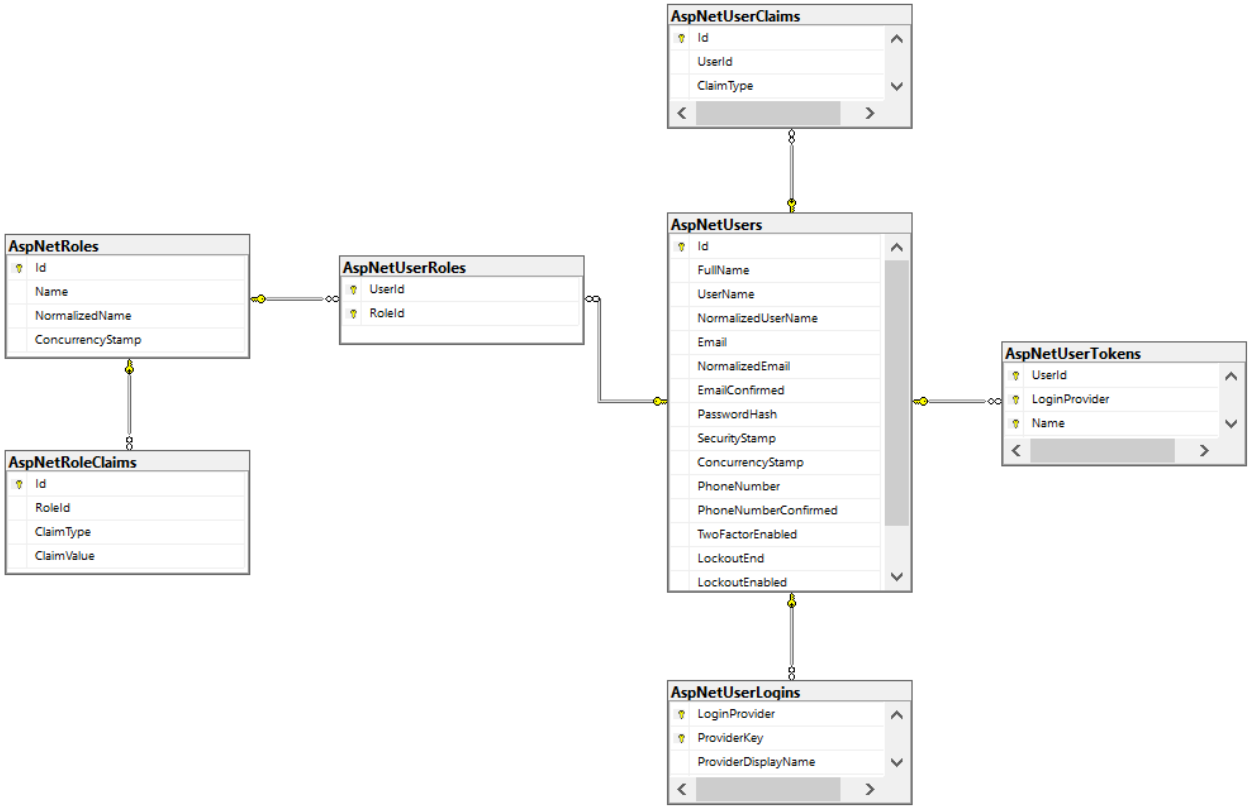

Рис. 2.4.1 - ER діаграма User-Admin

Елементи діаграми згенеровані автоматично(дані по дефолту) з допомогою бібліотеки Microsoft.AspNetCore.Identity.

Microsoft.AspNetCore.Identity є бібліотекою, яка надає розширений функціонал для керування аутентифікацією та авторизацією в додатках на платформі ASP.NET Core. Вона пропонує потужні засоби для роботи зі збереженням та керуванням інформацією про користувачів, ролях, правах доступу та інших аспектах ідентифікації та авторизації.

Основні можливості Microsoft.AspNetCore.Identity включають:

1. Управління користувачами: бібліотека надає можливість створювати, зберігати, оновлювати та видаляти облікові записи користувачів. Вона дозволяє зберігати важливі дані про користувача, такі як ім'я, електронна адреса, хешований пароль та інші корисні властивості.
2. Ролева базова авторизація: Microsoft.AspNetCore.Identity надає можливість визначати ролі користувачів та надавати їм відповідні права доступу до різних частин додатка. Це дозволяє контролювати, які ресурси та функціональні можливості доступні різним категоріям користувачів.
3. Внутрішні та зовнішні постачальники авторизації: бібліотека підтримує різні методи аутентифікації, включаючи стандартну форму аутентифікації з використанням логіна та пароля, а також зовнішні постачальники авторизації, такі як Google, Facebook, Twitter тощо. Це дозволяє користувачам вибирати зручний метод аутентифікації та авторизації.
4. Підтримка двофакторної аутентифікації: Microsoft.AspNetCore.Identity дозволяє використовувати двофакторну аутентифікацію для забезпечення додаткового рівня безпеки. Це може включати в себе використання SMS-повідомлень, електронної пошти, аутентифікаторів або інших методів підтвердження.
5. Парольні політики та хешування: бібліотека надає можливість встановлення політик щодо складності паролів, максимальної кількості спроб введення паролю, та інших параметрів безпеки. Вона також використовує безпечні алгоритми хешування для збереження паролів в базі даних.

Microsoft.AspNetCore.Identity забезпечує потужні засоби для реалізації системи ідентифікації та авторизації в додатках на платформі ASP.NET Core. Вона спрощує розробку безпечних та функціональних систем управління користувачами та доступом, дозволяючи розробникам швидко використовувати готові рішення для цих завдань.

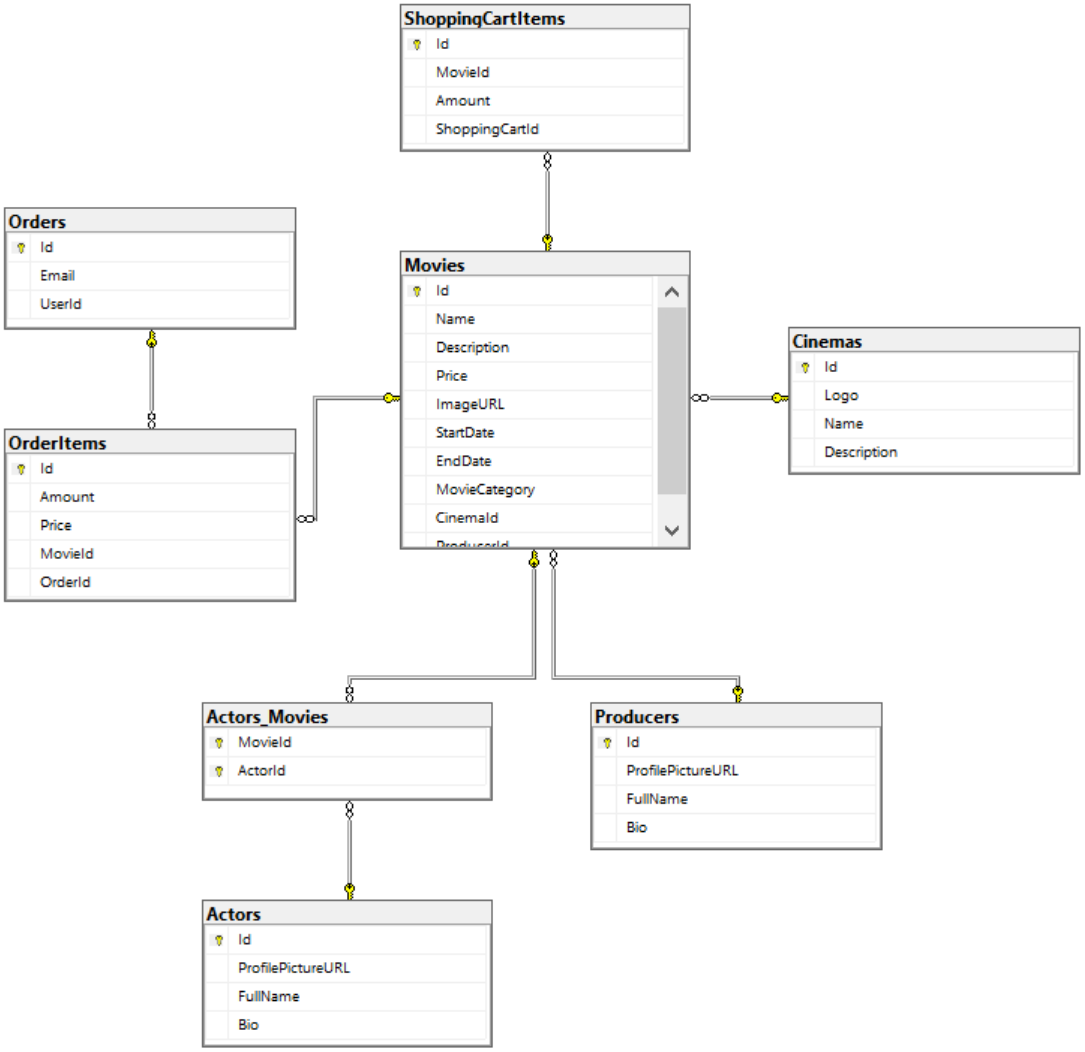

Рис. 2.4.2 - ER діаграма сервісу

На рисунку вище, зображено 8 сутностей, які використовуються в застосунку і зв'язок між ними.

- Orders - дані про user, який бере замовлення;
- OrdersItems - вся інформація про конкретне замовлення;
- ShoppingCartItems - окремий елемент корзини;

- Cinemas - дані про кінотеатри;
- Producers - дані про продюсерів;
- Movies - дані про фільми;
- Actors-дані про акторів.
- Actors_Movies - заглушка між акторами і фільмами.

3. ПРОГРАМНА РЕАЛІЗАЦІЯ ЗАСТОСУНКУ

3.1. Структура проекту

На рисунку нижче представлена структура проекту бібліотечного сервісу, який проектувався в 2 розділі.

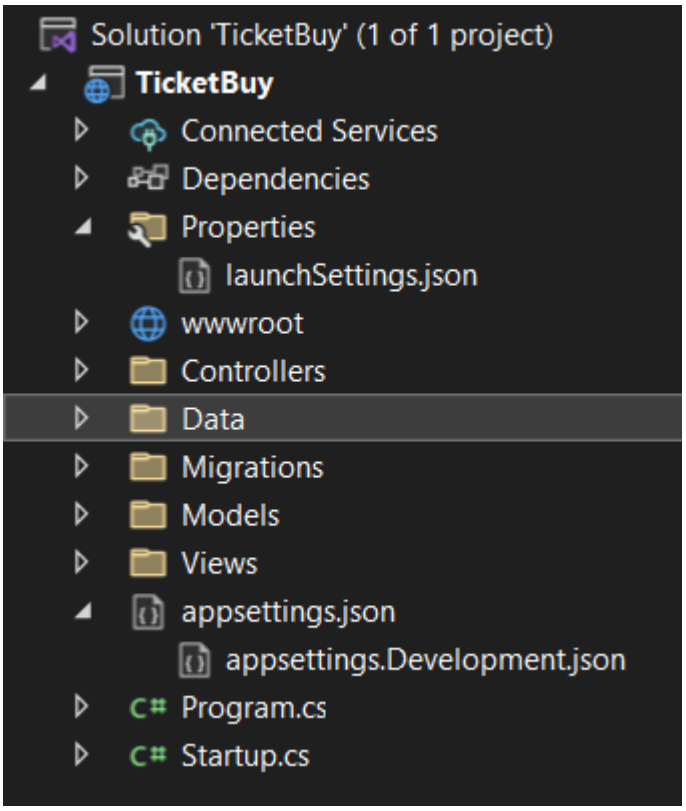

Рис. 3.1.1 - Структура проекту

Проект складається з папок:

- Controllers - папка в якій знаходяться контролери;

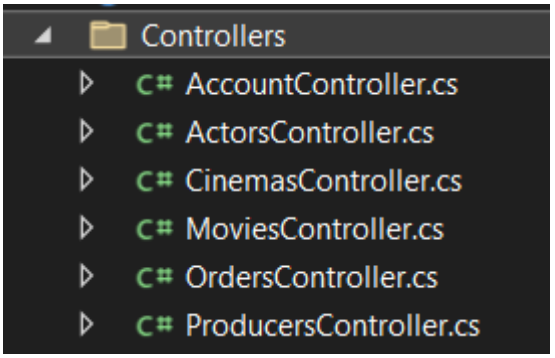

Рис. 3.1.2 - контроллери

- data - папка з вмістом, що пов'язаний з даними(enums, services, viewmodels)

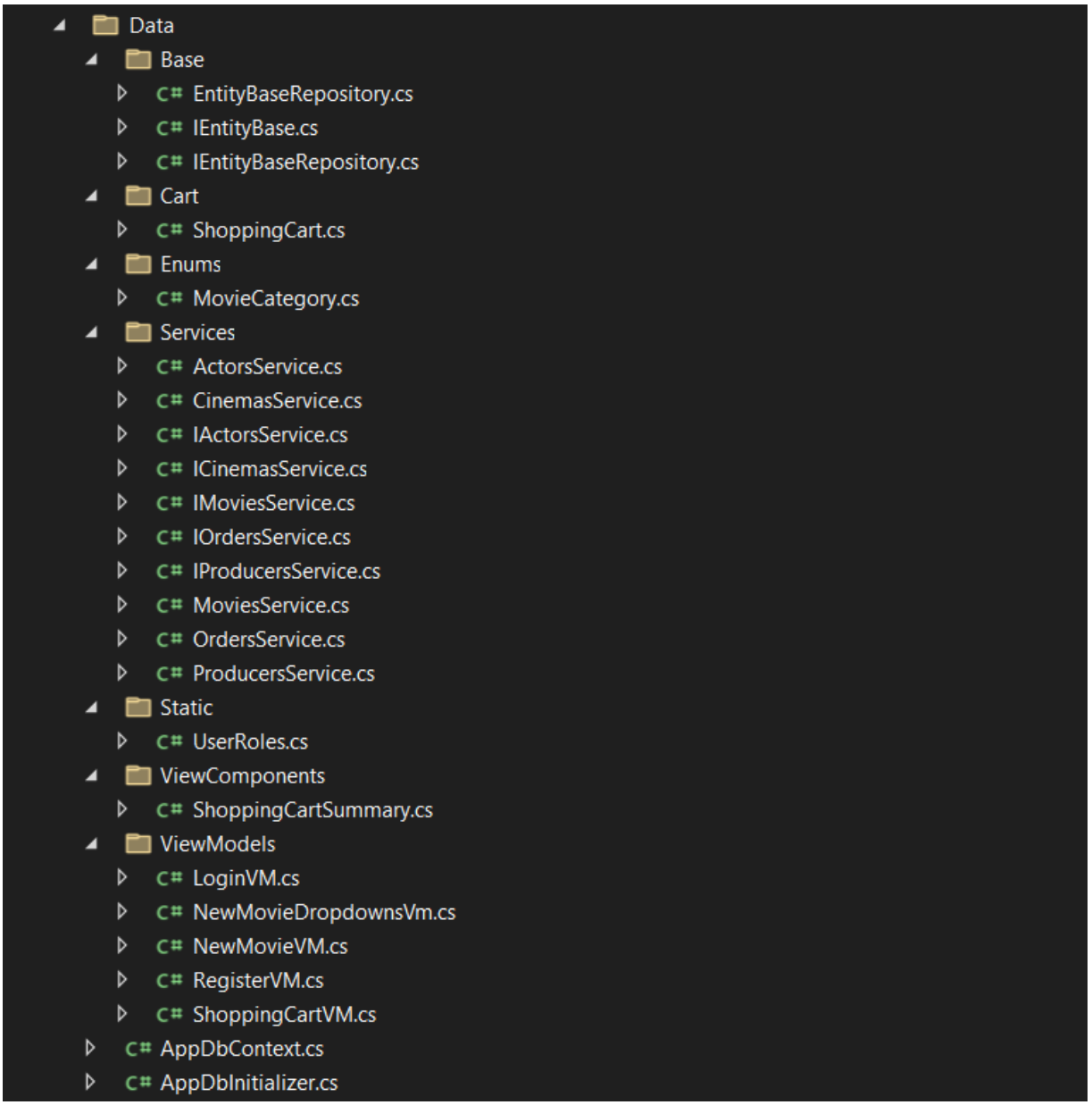

Рис. 3.1.2 - data related дані

- migrations - “міграції”(операції з бд);

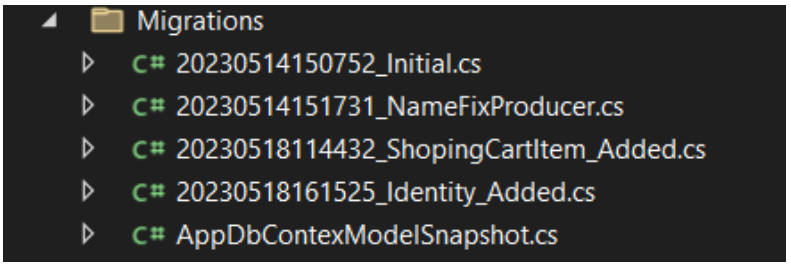

Рис. 3.1.3 - migrations

- models - моделі;

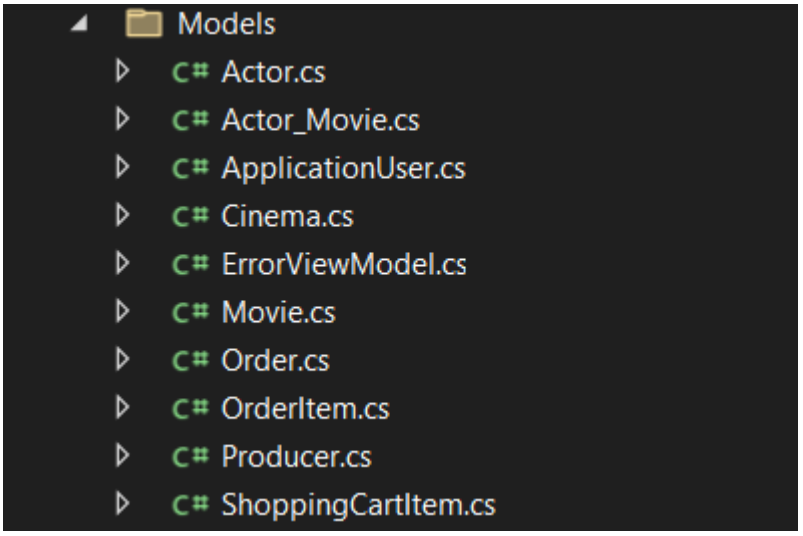

Рис. 3.1.4 - models

- views - обгортки сторінок(вкладки);

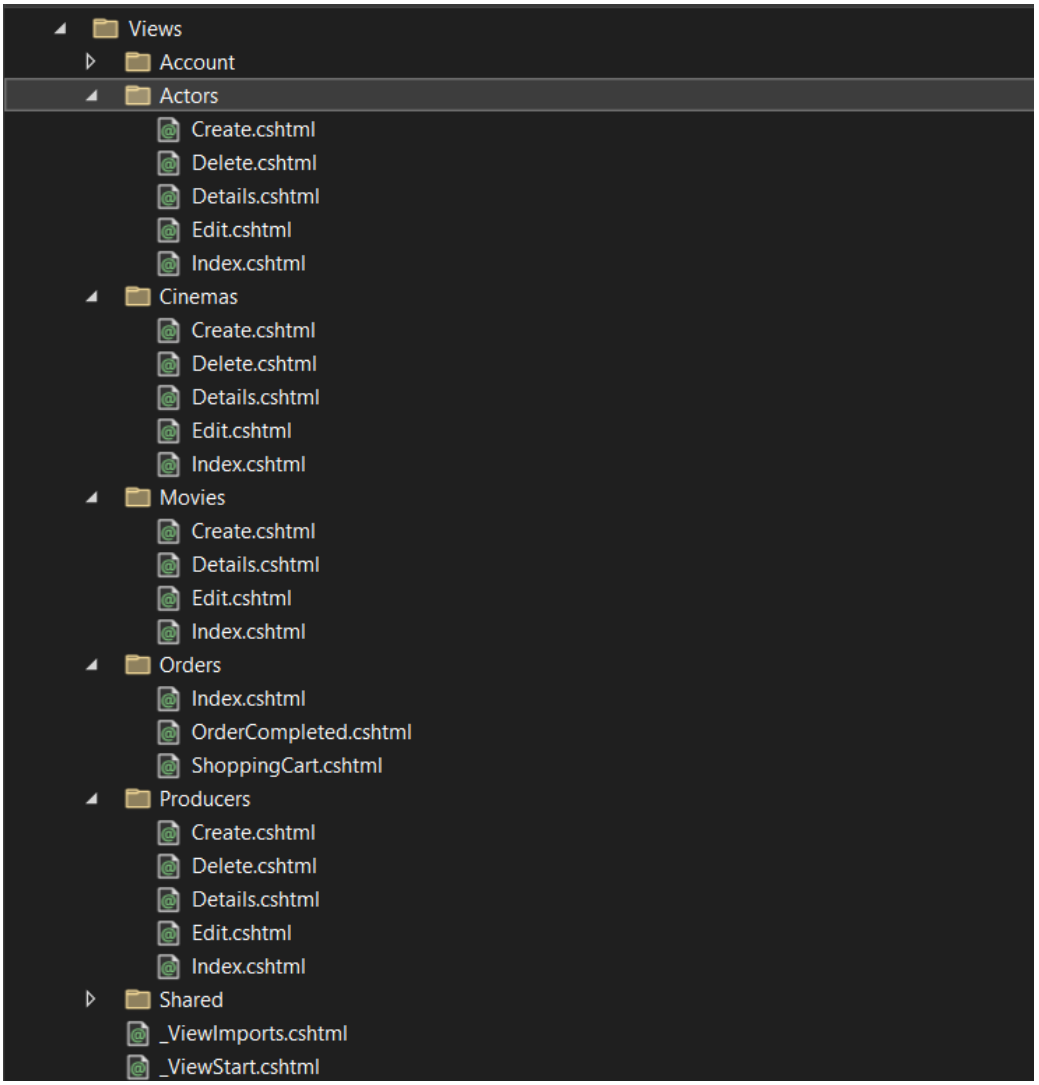

Рис. 3.1.5 - views

- Program.cs - точка входу в додатках .Net;
- Startup.cs - configuration class для додатків на .Net;

3.2. Розробка баз даних

Для розробки бази даних ticketbuy-db використовувалась Microsoft SQL Server Management Studio. І для застосунку було необхідно 8 створених таблиць, а також згенеровані бібліотекою Identity.

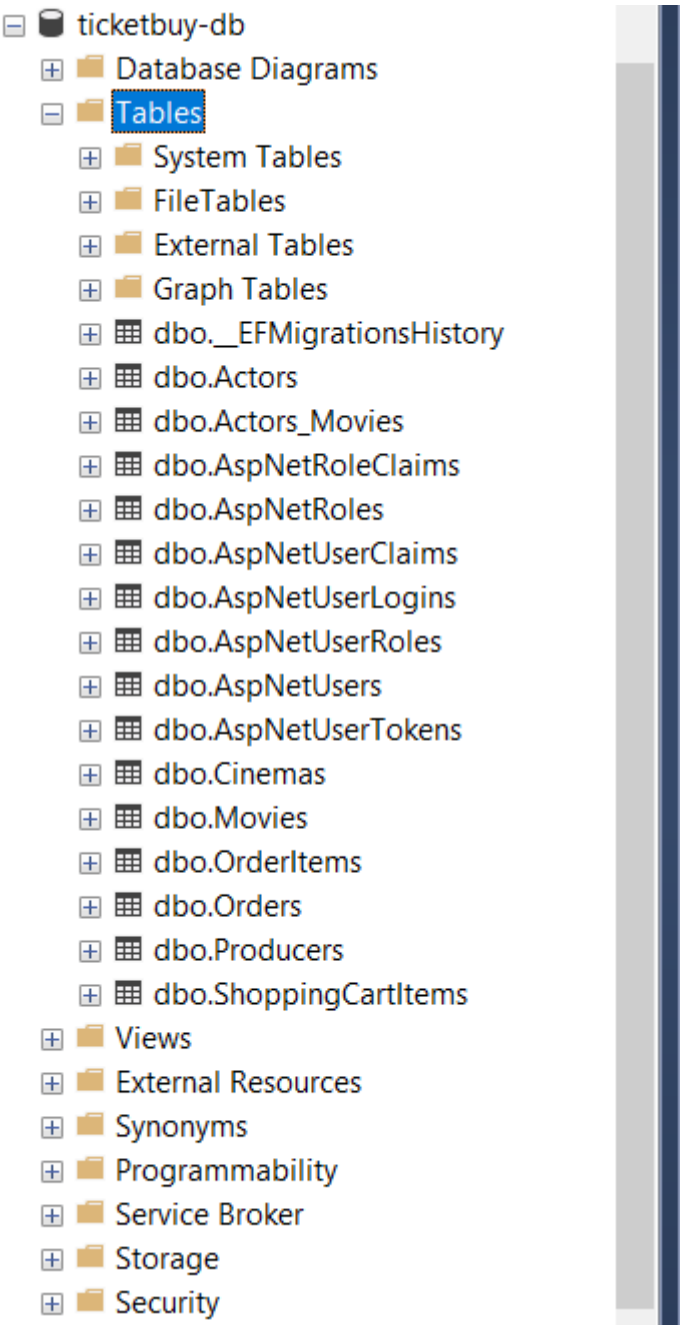

Рис. 3.2.1 - ticketbuy-db

Таблиця 1

Таблица “**Actors”** містить інформацію про актора

Поле	Тип	Опис
ProfilePictureURL	nvarchar(MAX)	силка на фото
FullName	nvarchar(50)	Ім’я
Bio	nvarchar(MAX)	Біографія

Таблиця 2

Таблиця “**Actors_Movies”** містить інформацію про зв'язок актор-фільм.

Поле	Тип	Опис
MovieId	int	Ідентифікатор фільму
ActorId	int	Ідентифікатор актора

Таблиця 3

Таблиця “**Cinemas”** містить інформацію кінотеатр.

Поле	Тип	Опис
Logo	nvarchar(MAX)	Логотип
Name	nvarchar(MAX)	Назва
Description	nvarchar(MAX)	Опис

Таблиця 4

Таблиця “**Movies”** містить всю інформацію про фільм.

Поле	Тип	Опис
Name	nvarchar(MAX)	Назва
Description	nvarchar(MAX)	Опис
Price	float	Ціна
ImageURL	nvarchar(MAX)	Постер
StartDate	datetime2(7)	Початок прокату
EndDate	datetime2(7)	Кінець прокату
MovieCategory	int	Жанр
CinemaId	int	Кінотеатр
ProducerId	int	Продюсер

Таблиця 5

Таблиця **“OrderItems”** містить дані про замовлення.

Поле	Тип	Опис
Amount	int	Кількість
Price	float	Ціна
MovieId	int	Фільм
OrderId	int	Хто замовляв

Таблиця 6

Таблиця “**Orders”** містить всю інформацію про користувача, який зробив замовлення.

Поле	Тип	Опис
Email	nvarchar(MAX)	Пошта користувача
UserId	nvarchar(MAX)	Id користувача

Таблиця 7

Таблиця “**Producers”** містить інформацію про видавництво певної книги.

Поле	Тип	Опис
ProfilePictureURL	nvarchar(MAX)	силка на фото
FullName	nvarchar(50)	Ім’я

Таблиця 8

Таблиця “**ShoppingCartItems”** містить інформацію про елемент кошику.

Поле	Тип	Опис
MovieId	int	Фільм
Amount	int	Кількість білетів
ShoppingCartId	nvarchar(MAX)	Id кошика

3.3. Тестування програми

При вході в сервіс, користувача буде зустрічати сторінка з переліком фільмів та всією інформацією про них.

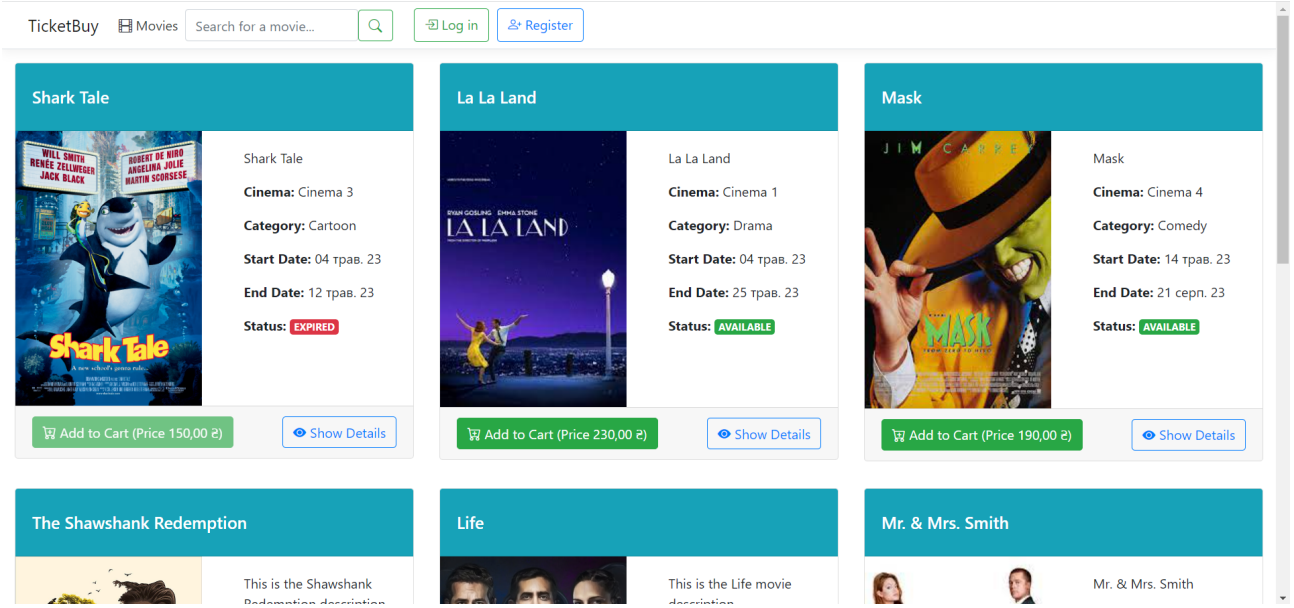

Рис. 3.4.1 - Домашня сторінка для незареєстрованого користувача

Відвідувач сайту, може навіть без акаунту користувача переглянути деталі фільму такі :

- актори
- продюсер
- кінотеатр, що веде прокат
- опис

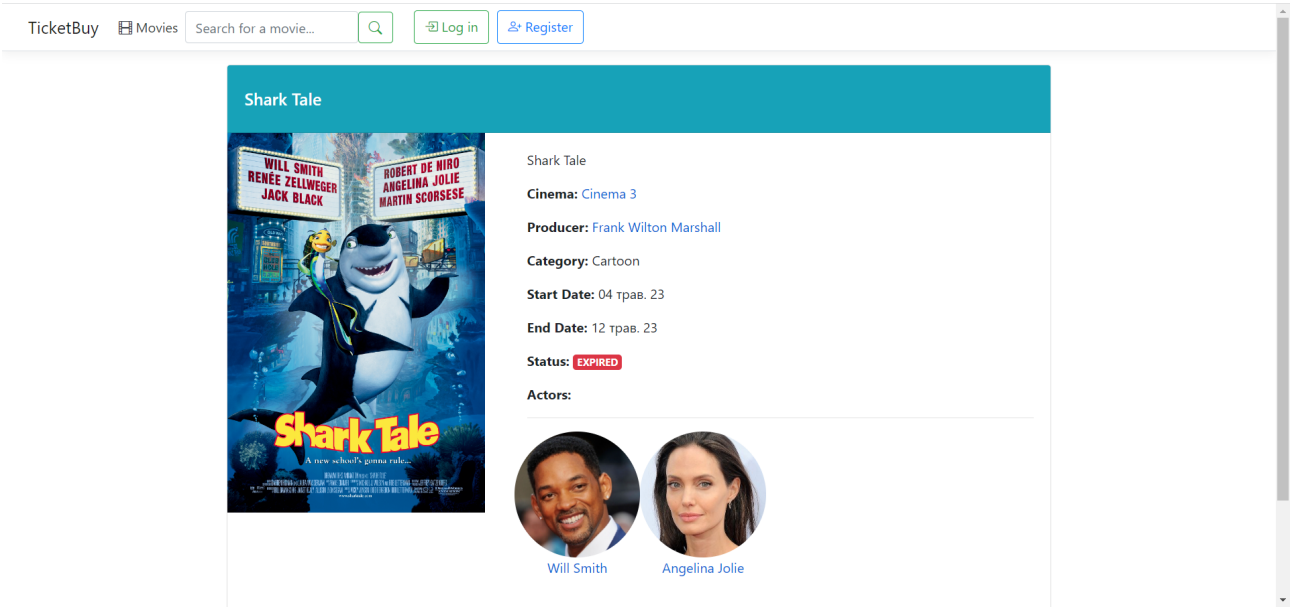

Рис. 3.4.2 - Details

Незареєстрований користувач може зареєструватись, або ж залогінитись, якщо в нього вже є створений акаунт учасника.

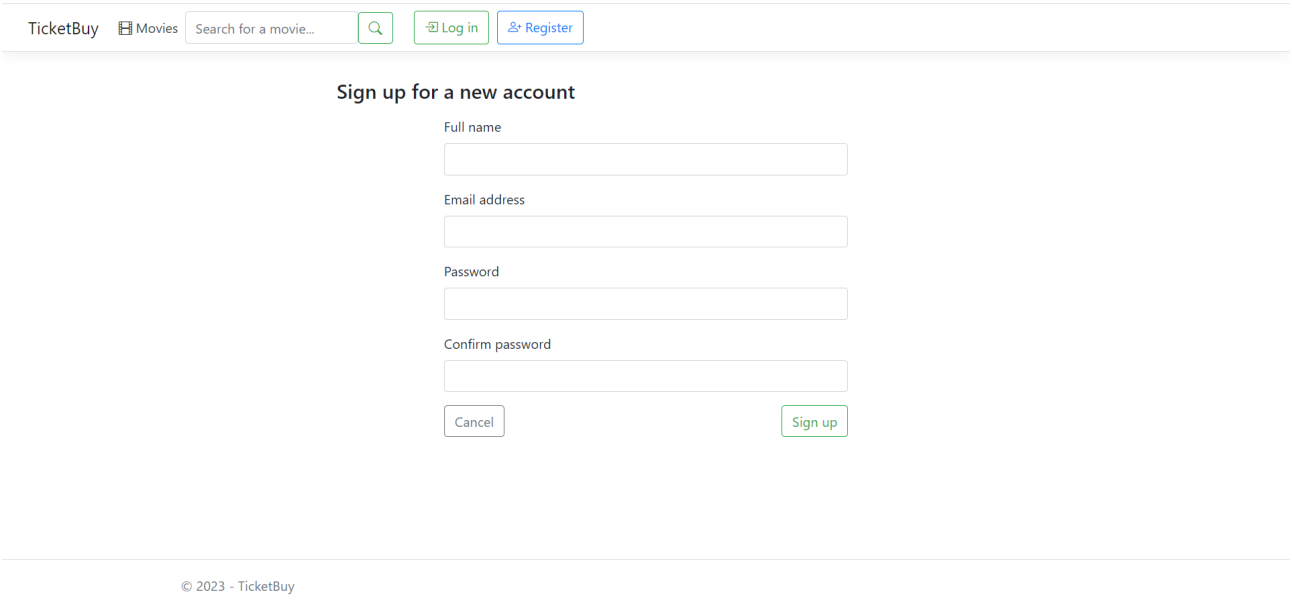

Рис. 3.4.3 - сторінка Sign Up

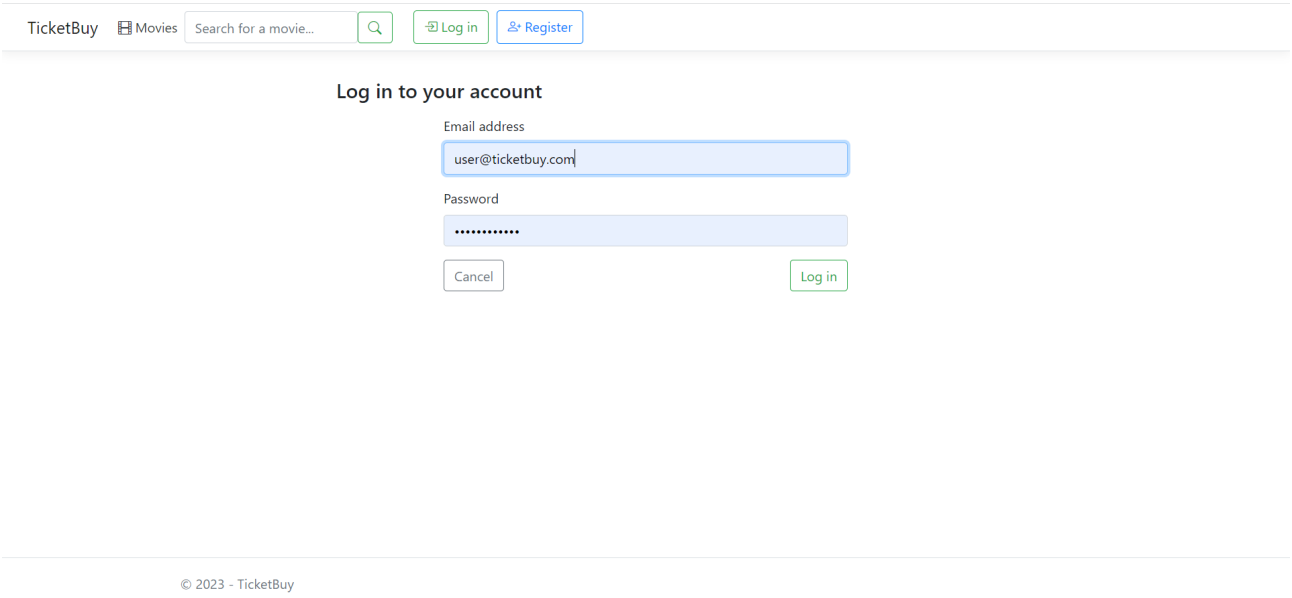

Рис. 3.4.4 - сторінка User Login

Після успішної реєстрації, або ж залогінення користувач має можливість замовляти квитки причому з різних кінотеатрів, і різну кількість, а також сплачувати це з допомогою платіжної системи PayPal.

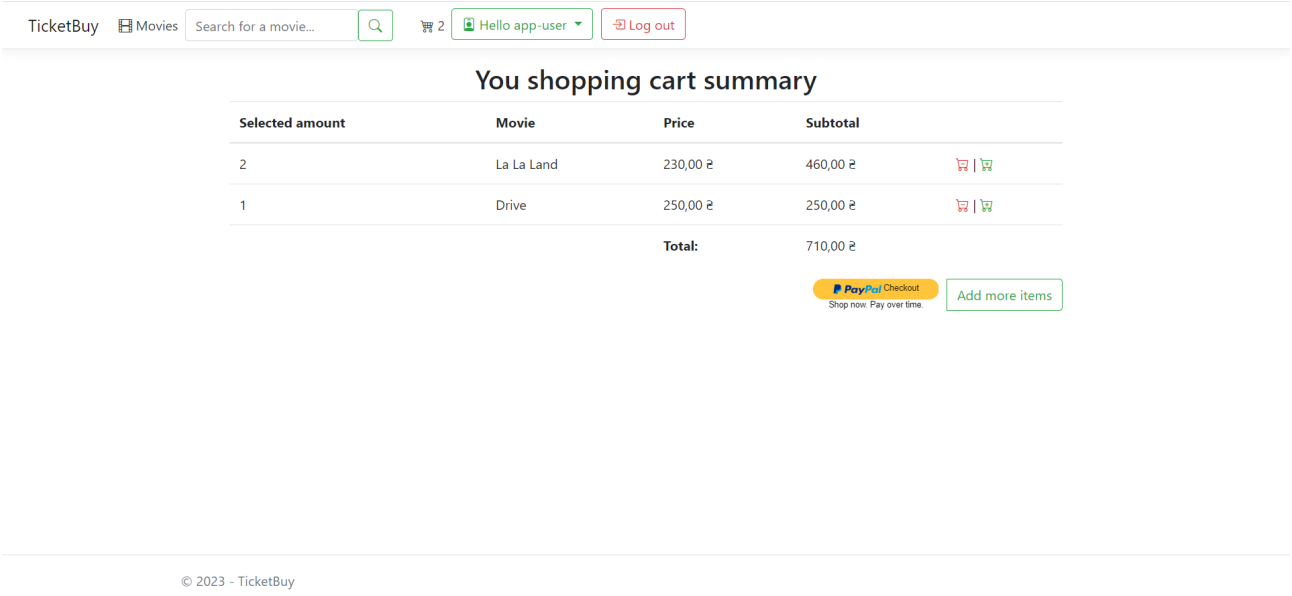

Рис. 3.4.5 - кошик

В кошик було додано 2 квитки на фільм “La La Land”, та один квиток на фільм “Drive”. Можна помітити з деталей фільмів, що вони показ фільмів буде в різних кінотеатрах, тобто функціонал це дозволяє.

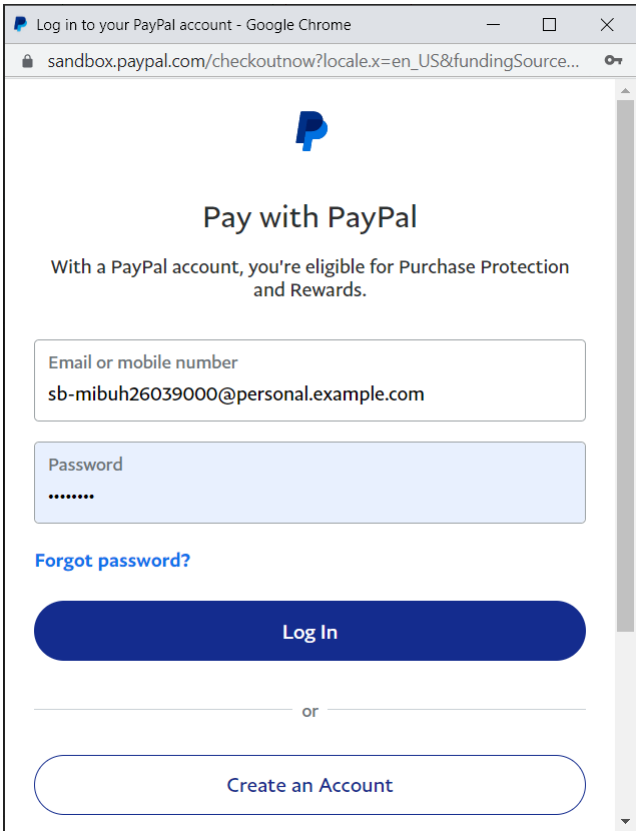

Рис. 3.4.6 - Оплата квитків через PayPal

Також варто зауважити, що у випадку, якщо термін прокату вже пройшов, то замовити квиток вже не можна.

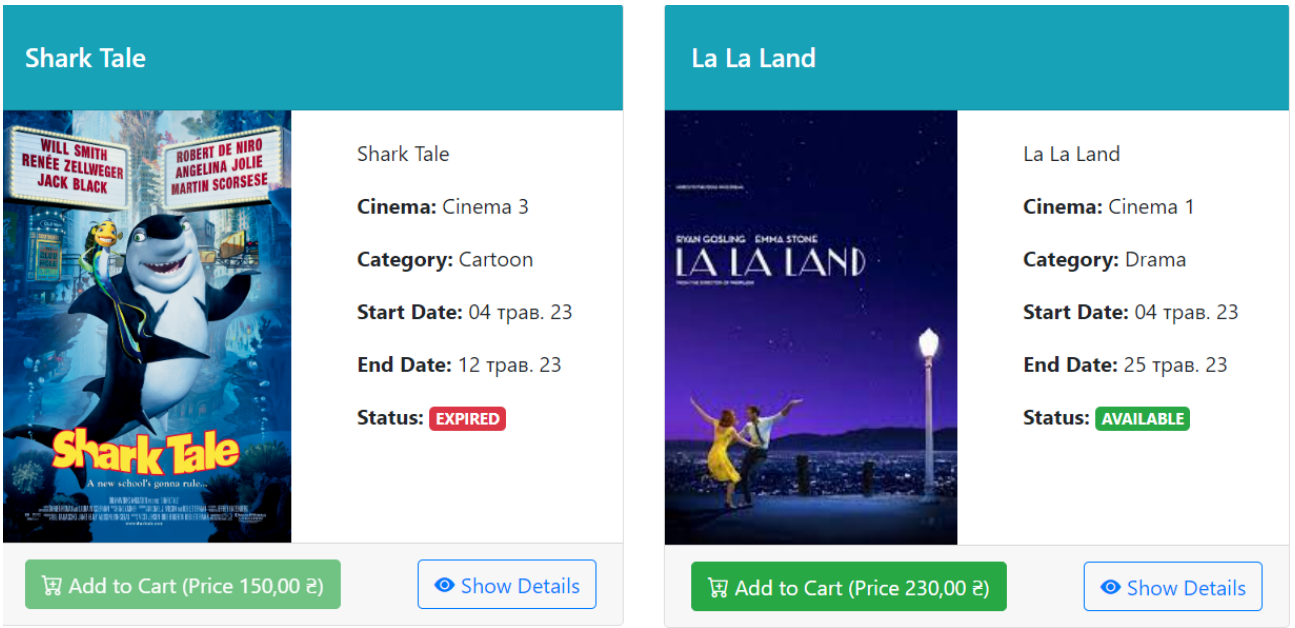

Рис. 3.4.7 - Функціонал (Expired/Available)

На рисунку вище видно, що статус Expired забирає можливість додати квиток до кошика.

Вхід в систему як адміністратор.

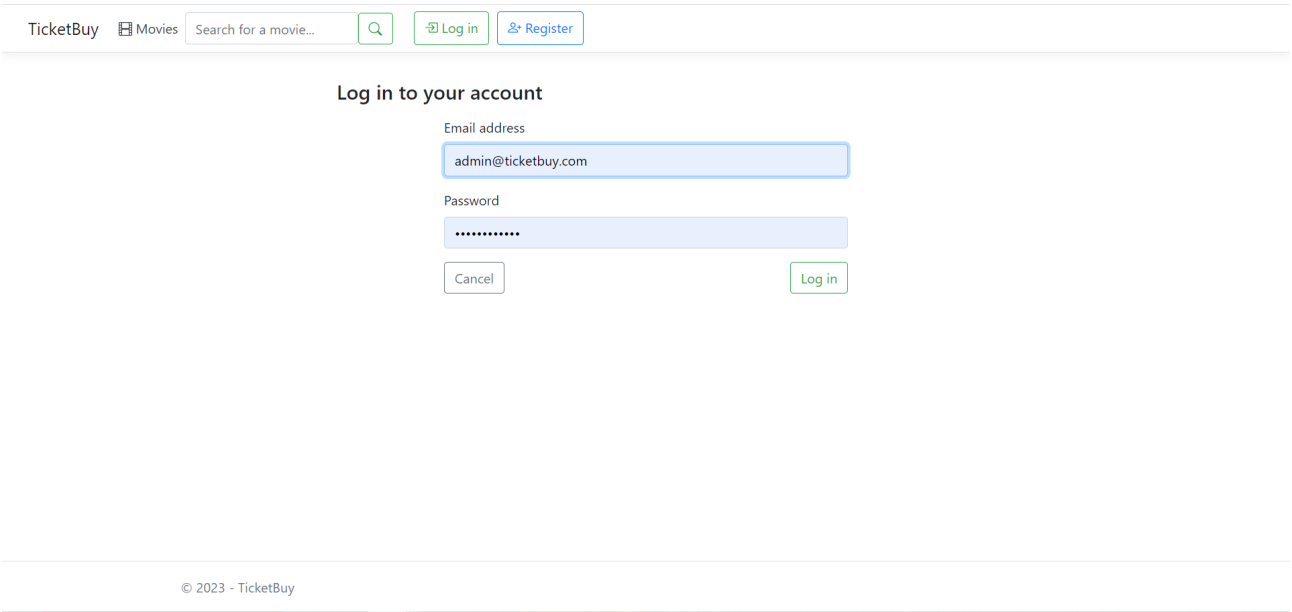

Рис. 3.4.8 - сторінка Admin Login

В адміністратора є широкий спектр можливостей для ефективного управління сервісом.

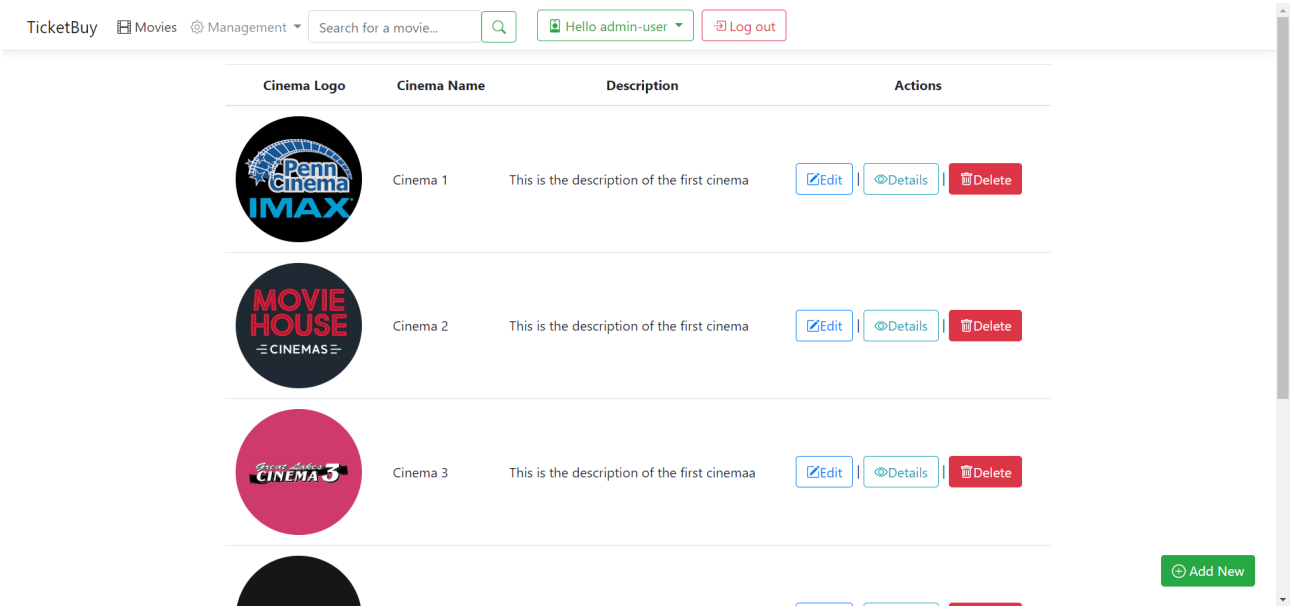

Рис. 3.4.9 - сторінка Admin/Management/Cinemas

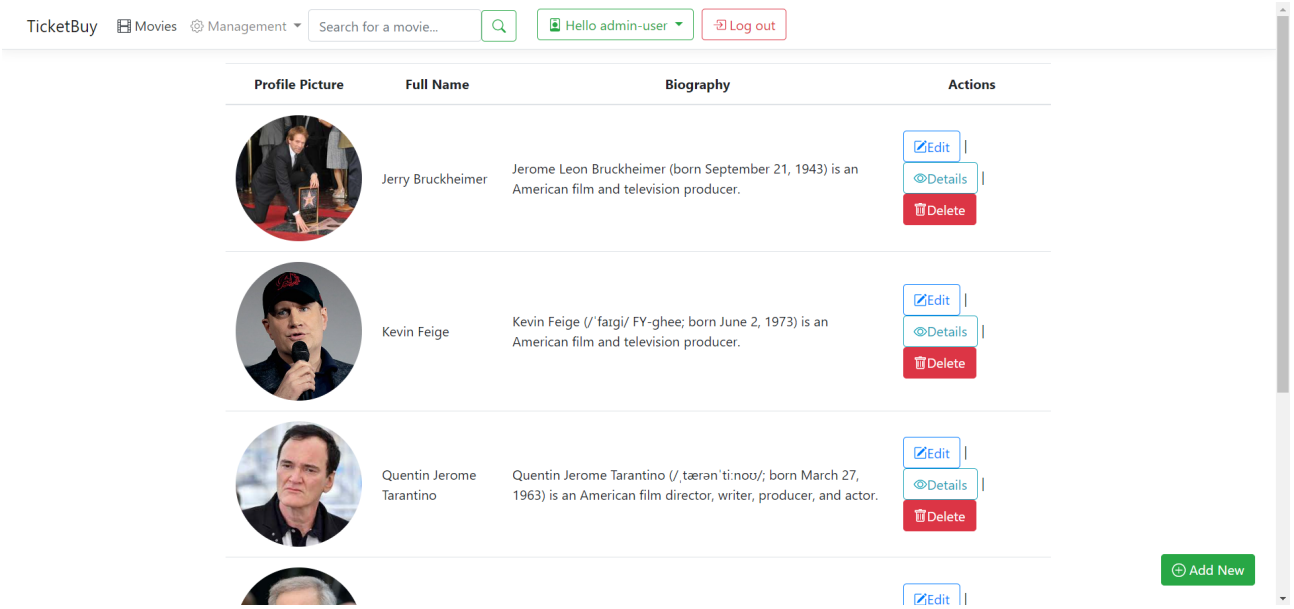

Рис. 3.4.10 - сторінка Admin/Management/Producers

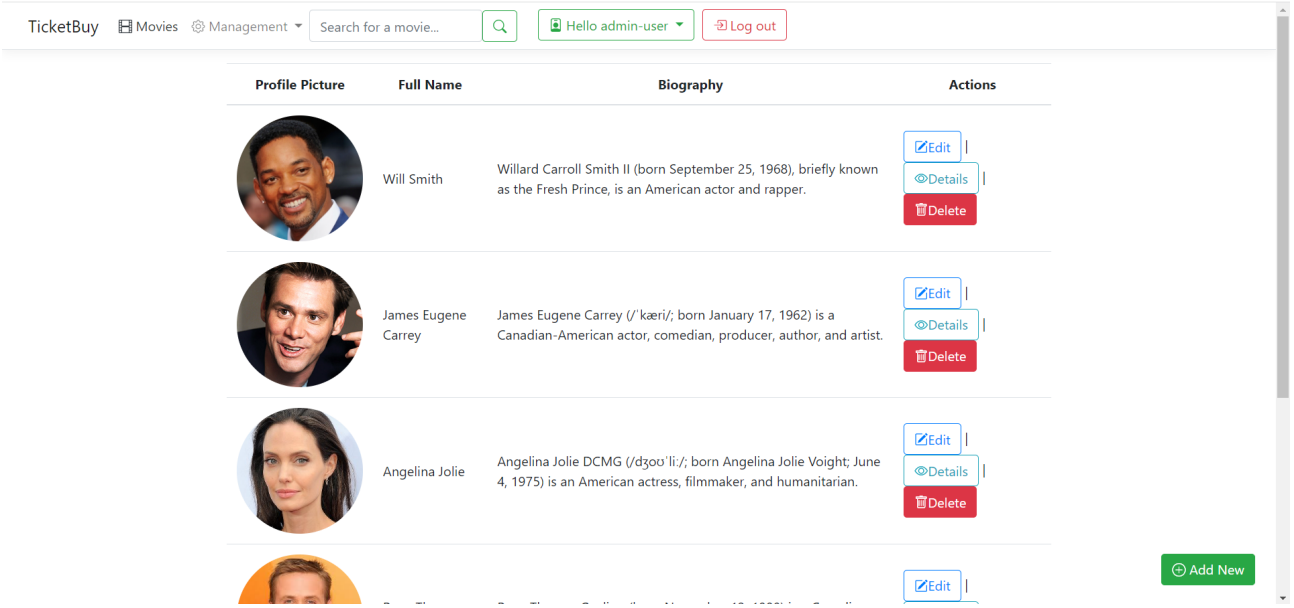

Рис. 3.4.11 - сторінка Admin/Management/Actors

З рисунків вище, можна помітити, що admin може модифікувати деталі, переглядати ці ж деталі. Тобто профовити різноманітні операції як з кінотеатром, так і з акторам та продюсерами.

Також звісно в адміна є можливість як модифікувати деталі фільму, так і додавати новий.

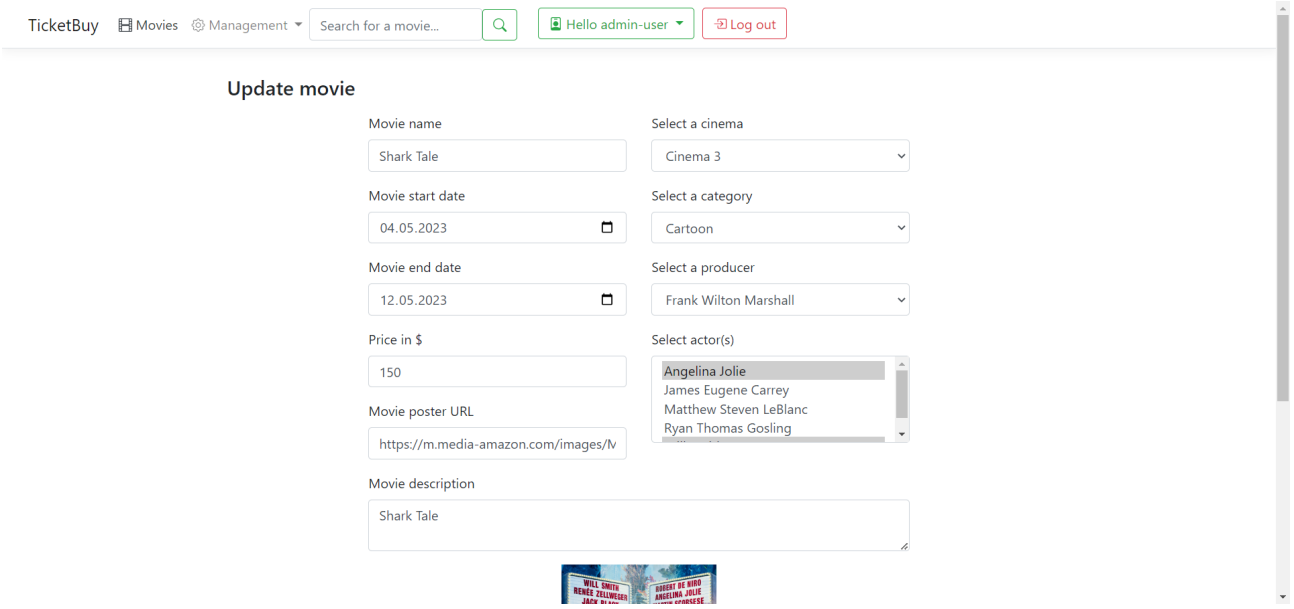

Рис. 3.4.12 - сторінка Admin/Movies/Update

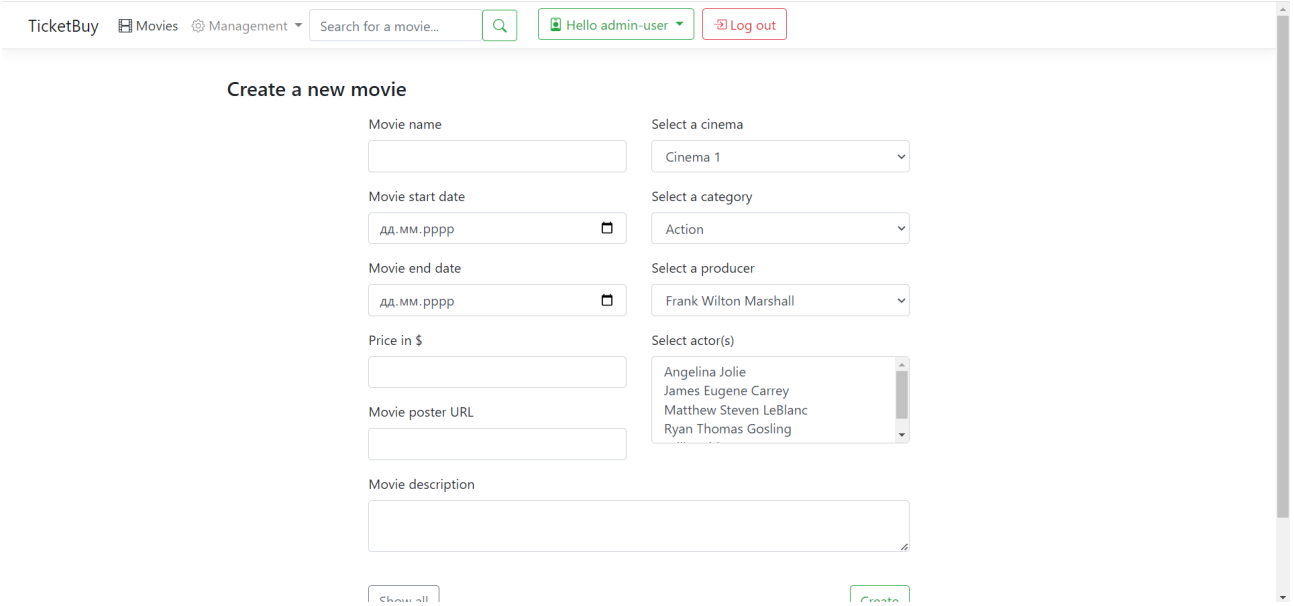

Рис. 3.4.12 - сторінка Admin/Movies/New

Можна бачити, що списки динамічно змінюються, при відповідних командах, тобто функціонал працює справно.

ВИСНОВКИ

В результаті виконання курсової робити був розроблений веб-сервіс для покупки білетів, якому відповідають зручні можливості як і для адміністратора, так і для користувача. Зокрема, можливості адміністратора: ефективне управління сервісом, модифікація таких даних як: кінотеатри, фільми, актори, продюсери.

Можливості учасника: пошук потрібних фільмів, відправлення відповідних фільмів у кошик, покупка (оплата через PayPal) білетів.

На етапі проектування був проведений загальний аналіз конкурентів, що присутні на ринку, таких як: “Вкіно” та “Планета кіно”. Були враховані основні їх недоліки, щоб зробити якісний продукт. Побудовано Use Case діаграми та діаграми баз даних.

З технічного боку були використані актуальні технології в сфері веб-застосунків, що надає можливість використання даного проекту в комерційних цілях, а не тільки в навчальних.

Підбиваючи підсумки, можна сказати, що запропонований проект є актуальним і дозволяє ефективно виконувати відповідні задачі. Він має велике поле для модернізації, наприклад додавання варіанту пошуку та покупки білетів, з допомогою яких буде можливо приймати участь у таких культурних заходах як виставки та театральні вистави.

ПЕРЕЛІК ПОСИЛАНЬ

1. 6 причин, чому купувати в інтернеті - безпечно [Електронний ресурс]. — Режим доступу: https://busfor.ua/uk/blog/safe-online-shopping
2. Кінотеатр імені О.Довженка [Електронний ресурс]. — Режим доступу: https://kinobc.com.ua
3. Стрічка, кіно, кава. [Електронний ресурс]. — Режим доступу: https://kino-strichka.com
4. Планета Кіно [Електронний ресурс]. — Режим доступу: https://planetakino.ua
5. Вкіно [Електронний ресурс]. — Режим доступу: https://vkino.com.ua/ua
6. A tour of the C# language [Електронний ресурс]. — Режим доступу: https://docs.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/
7. .NET [Електронний ресурс]. — Режим доступу: https://web.archive.org/web/20210429082913/https://dotnet.microsoft.com/download/dotnet
8. What is ASP.NET? [Електронний ресурс]. — Режим доступу: https://uk.wikipedia.org/wiki/ASP.NET#cite_note-2
9. Microsoft SQL Server [Електронний ресурс]. — Режим доступу: https://www.techtarget.com/searchdatamanagement/definition/SQL-Server
10. What is SQL Server [Електронний ресурс]. — Режим доступу: https://www.sqlservertutorial.net/getting-started/what-is-sql-server/
11. What Is Bootstrap? [Електронний ресурс]. — Режим доступу: https://www.hostinger.com/tutorials/what-is-bootstrap/
12. Клієнт-серверна архітектура [Електронний ресурс]. — Режим доступу: https://training.qatestlab.com/blog/technical-articles/client-server-architecture/
13. What is a use case diagram? [Електронний ресурс]. — Режим доступу: https://www.lucidchart.com/pages/uml-use-case-diagram
14. What are your ERD needs? [Електронний ресурс]. — Режим доступу: https://www.lucidchart.com/pages/er-diagrams

ДОДАТОК А

Файли програми:

Program.cs:

```
using Microsoft.AspNetCore.Hosting;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace TicketBuy
{
    public class Program
    {
        public static void Main(string[] args)
        {
            CreateHostBuilder(args).Build().Run();
        }

        public static IHostBuilder CreateHostBuilder(string[] args) =>
            Host.CreateDefaultBuilder(args)
                .ConfigureWebHostDefaults(webBuilder =>
                {
                    webBuilder.UseStartup<Startup>();
                });
    }
}
```

Startup.cs:

```
using Microsoft.AspNetCore.Authentication.Cookies;
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.HttpsPolicy;
```

```
using Microsoft.AspNetCore.Identity;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using TicketBuy.Data;
using TicketBuy.Data.Cart;
using TicketBuy.Data.Services;
using TicketBuy.Models;

namespace TicketBuy
{
    public class Startup
    {
        public Startup(IConfiguration configuration)
        {
            Configuration = configuration;
        }

        public IConfiguration Configuration { get; }

        // This method gets called by the runtime. Use this method to add services to the container.
        public void ConfigureServices(IServiceCollection services)
        {

            //DbContext configuration
services.AddDbContext<AppDbContext>(options => options.UseSqlServer(Configuration.GetConnectionString("DefaultConnectionString")));

            //Services configuration
            services.AddScoped<IActorsService, ActorsService>();
            services.AddScoped<IProducersService, ProducersService>();
            services.AddScoped<ICinemasService, CinemasService>();
            services.AddScoped<IMoviesService, MoviesService>();
            services.AddScoped<IOrdersService, OrdersService>();
```

```
        services.AddSingleton<IHttpContextAccessor, HttpContextAccessor>();
        services.AddScoped(sc => ShoppingCart.GetShoppingCart(sc));

        //Authentication and authorization
        services.AddIdentity<ApplicationUser, IdentityRole>().AddEntityFrameworkStores<AppDbContext>();
        services.AddMemoryCache();
        services.AddSession();
        services.AddAuthentication(options =>
        {
            options.DefaultScheme = CookieAuthenticationDefaults.AuthenticationScheme;
        });

        services.AddControllersWithViews();
    }

    // This method gets called by the runtime. Use this method to configure the HTTP request pipeline.
    public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
    {
        if (env.IsDevelopment())
        {
            app.UseDeveloperExceptionPage();
        }
        else
        {
            app.UseExceptionHandler("/Home/Error");
                    // The default HSTS value is 30 days. You may want to change this for production scenarios, see
https://aka.ms/aspnetcore-hsts.
            app.UseHsts();
        }
        app.UseHttpsRedirection();
        app.UseStaticFiles();

        app.UseRouting();
        app.UseSession();

        //Authentication & Authorization
        app.UseAuthentication();
        app.UseAuthorization();
```

```
        app.UseAuthorization();

        app.UseEndpoints(endpoints =>
        {
            endpoints.MapControllerRoute(
                name: "default",
                pattern: "{controller=Home}/{action=Index}/{id?}");
        });

        //Seed database
        AppDbInitializer.Seed(app);
        AppDbInitializer.SeedUsersAndRolesAsync(app).Wait();
    }
  }
}
```

_Layout.cshtml:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>@ViewData["Title"] - TicketBuy</title>
  <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />
  <link rel="stylesheet" href="~/css/site.css" />
  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.10.5/font/bootstrap-icons.css">
</head>
<body>
  <header>
      <nav class="navbar navbar-expand-sm navbar-toggleable-sm navbar-light bg-white border-bottom box-shadow
mb-3">
      <div class="container-fluid">
          <a class="navbar-brand" asp-controller="Movies" asp-action="Index">TicketBuy</a>
                <button class="navbar-toggler" type="button" data-toggle="collapse" data-target=".navbar-collapse"
aria-controls="navbarSupportedContent"
                aria-expanded="false" aria-label="Toggle navigation">
              <span class="navbar-toggler-icon"></span>
          </button>
```

```
<div class="navbar-collapse collapse d-sm-inline-flex justify-content-between">
  <ul class="navbar-nav flex-grow-1">
    <li class="nav-item">
        <a class="nav-link text-dark" asp-area="" asp-controller="Movies" asp-action="Index"><i class="bi
bi-film"></i> Movies</a>
    </li>

    @if(User.Identity.IsAuthenticated && User.IsInRole("Admin"))
    {
        <li class="nav-item dropdown">
                <a class="nav-link dropdown-toggle" href="#" id="navbarDropdown" role="button"
data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
            <i class="bi bi-gear"></i> Management
          </a>

          <div class="dropdown-menu" aria-labelledby="navbarDropdown">
                <a class="dropdown-item" asp-controller="Cinemas" asp-action="Index"><i class="bi
bi-camera-reels"></i> Cinemas</a>
                <a class="dropdown-item" asp-controller="Producers" asp-action="Index"><i class="bi
bi-headset"></i> Producers</a>
                <a class="dropdown-item" asp-controller="Actors" asp-action="Index"><i class="bi
bi-person-square"></i> Actors</a>
          </div>
        </li>
    }

                <form class="form-inline my-2 my-lg-0" asp-controller="Movies" asp-action="Filter"
style="padding-right: 25px;">
        <input name="searchString" type="text" class="form-control" placeholder="Search for a movie..."
aria-label="Recipient's username" aria-describedby="button-addon2">
                <button class="btn btn-outline-success my-2 my-sm-0" type="submit"><i class="bi
bi-search"></i></button>
  </form>

  <ul class="navbar-nav">
    @await Component.InvokeAsync("ShoppingCartSummary")
  </ul>

  <partial name="_Identity.cshtml" />
</div>
```

```
            </div>
        </nav>
    </header>
    <div class="container-fluid">
        <main role="main" class="pb-3">
            @RenderBody()
        </main>

    </div>



    <footer class="border-top footer text-muted">
        <div class="container">
            &copy; 2023 - TicketBuy
        </div>
    </footer>
    <script src="~/lib/jquery/dist/jquery.min.js"></script>
    <script src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script>
    <script src="~/lib/bootstrap/dist/js/bootstrap.min.js"></script>
    <script src="~/js/site.js" asp-append-version="true"></script>

    @await RenderSectionAsync("Scripts", required: false)
</body>
</html>
```