# ЗМІСТ

[ЗМІСТ 2](#_Toc40966546)

[ВСТУП 4](#_Toc40966547)

[1 ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ 5](#_Toc40966548)

[1.1 Аналіз предметної області 5](#_Toc40966549)

[1.2 Огляд існуючих програмних продуктів 5](#_Toc40966550)

[1.2.1 Додаток Q-Parser. 6](#_Toc40966551)

[1.2.2 Додаток TargetHunter 7](#_Toc40966552)

[1.2.3 Додаток CleverTarget 8](#_Toc40966553)

[1.2.4 Додаток Діггернаут 10](#_Toc40966554)

[1.3 Постановка задачі 12](#_Toc40966555)

[2 СПЕЦИФІКАЦІЯ ВИМОГ ДО ВЕБ-ДОДАТКУ 14](#_Toc40966556)

[2.1 Загальна інформація щодо веб-застосунку 14](#_Toc40966557)

[2.1.1 Огляд продукту 14](#_Toc40966558)

[2.1.2 Мета 15](#_Toc40966559)

[2.1.3 Межі 15](#_Toc40966560)

[2.1.4 Масштаб проекту 15](#_Toc40966561)

[2.1.5 Обмеження реалізації 15](#_Toc40966562)

[2.1.6 Діаграми потоків даних 15](#_Toc40966563)

[2.2 Вимоги до веб-додатку 16](#_Toc40966564)

[2.2.1 Функціональні вимоги 16](#_Toc40966565)

[2.2.2 Нефункціональні вимоги 18](#_Toc40966566)

[2.3 Опис обраних технологій для розробки веб-додатку 19](#_Toc40966567)

[2.3.1 HTML5 19](#_Toc40966568)

[2.3.2 CSS3 20](#_Toc40966569)

[2.3.3 ASP.NET Core 22](#_Toc40966570)

[2.3.4 Microsoft Visual Studio 23](#_Toc40966571)

[2.3.5 JavaScript 25](#_Toc40966572)

[2.3.6 MySQL 26](#_Toc40966573)

[2.4 Опис архітектури веб-додатку 27](#_Toc40966574)

[2.4.1 Діаграма розміщення 27](#_Toc40966575)

[2.4.2 Діаграма компонентів 28](#_Toc40966576)

[2.5 Модель бази даних 29](#_Toc40966577)

[3 ОПИС ВЕБ-ДОДАТКУ 34](#_Toc40966578)

[3.1 Тестування веб-додатку 34](#_Toc40966579)

[3.1.1 Верифікація та валідація веб-додатку 34](#_Toc40966580)

[3.1.2 Ручне тестування 36](#_Toc40966581)

[3.2 Керівництво користувача 38](#_Toc40966582)

[3.2.1 Загальний опис можливостей адміністратора 38](#_Toc40966583)

[3.3 Використання веб-застосунку 39](#_Toc40966584)

[ВИСНОВКИ 43](#_Toc40966585)

[СПИСОК ДЖЕРЕЛ ІНФОРМАЦІЇ 44](#_Toc40966586)

# ВСТУП

Інформаційні технології увійшли досить міцно в наше життя, вони застосовуються у всіх сферах життя людства, виконуючи особливо значущу двоїсту роль. Інформаційні технології представляють весь накопичений досвід людства в форматизованому вигляді, придатному для прикладного використання. І в ньому сконцентровані наукові знання і матеріалістичний досвід для здійснення суспільних процесів, при цьому економляться витрати праці, часу, енергії, речових засобів. І з кожним днем з непомірною силою роль ця збільшується. Це явище пояснюється властивостями, якими володіють інформаційні технології.

Об’єкт дослідження – онлайн інформувач нових сеансів у кінотеатрах. Предметом дослідження є створення веб-додатку «TicketsHere» на основі технологій ASP.NET Core і JavaScript. Предметна область – парсер веб ресурсів.

Застосування парсерів дозволяє відфільтровувати непотрібну інформацію та мати можливість отримувати тільки релевантні дані.

Метою курсової роботи є розробка веб-додатку парсера вебсайту, який допоможе людині швидко та з будь якого пристрою отримувати інформацію про нові сеанси кіно, або нові вільні місця. Під час її виконання планується дослідити дану предметну область і вивчити теоретичні питання розробки даного веб-додатку і на підставі вивченого матеріалу розробити веб-застосунок з використанням сучасних технологій розробки.

# 1 ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ

## Аналіз предметної області

Основна проблема сучасного Інтернету - надлишок інформації, яку людина не в змозі систематизувати вручну.

Переваги парсинга чисельні. У порівнянні з людиною парсери можуть збирати дані швидше і в будь-якому режимі, хоч цілодобово, слідувати всім заданим параметрам, навіть дуже тонким, уникати помилок від неуважності або втоми, виконувати регулярну перевірку по заданому інтервалу (щотижня і т. п.) і т.д.

Спарсити можна все, що є на сайті у відкритому доступі. Найчастіше потрібні: найменування та категорії товарів, основні характеристики, ціна, інформація про акції та новинки, тексти опису товарів для їх подальшого перероблення «під себе» і т. п. Тож в наш час коли інформація переповнює мережі ви користування парсерів є дуже доречним.

Парсинг сайтів – це ефективний спосіб отримувати тільки необхідно інформацію з веб ресурсів при умові, якщо цей веб-ресурс не розповсюджує API для розробників.

Додаток буде корисний кіноманам – людям, що чекають виходу певного фільму, або звичайним користувачам, які не встигли купити квиток на сеанс.

Актуальність роботи полягає в тому, що фільми стають дедалі й дедалі цікавішими й багато бюджетними. Деякі люди з нетерпінням чекають виходу нового блокбастеру та готові йти до кінотеатру у перший же день.

## 1.2 Огляд існуючих програмних продуктів

Огляд існуючих веб-додатків, спрямованих на парсинг сайтів. Існуючі додатки орієнтовані на людей, які не встигли придбати щось заздалегіть.

Критерії оцінки кожного веб-додатку: дизайн, функціонал, простота і зрозумілість додатку, платний тариф.

### 1.2.1 Додаток Q-Parser.

Додаток розроблений для користувачів комп’ютерів та мобільних пристроїв, що надає можливість його використання у зручний час [1].

Інтерфейс додатку зображений на рисунках 1.1 та 1.2.

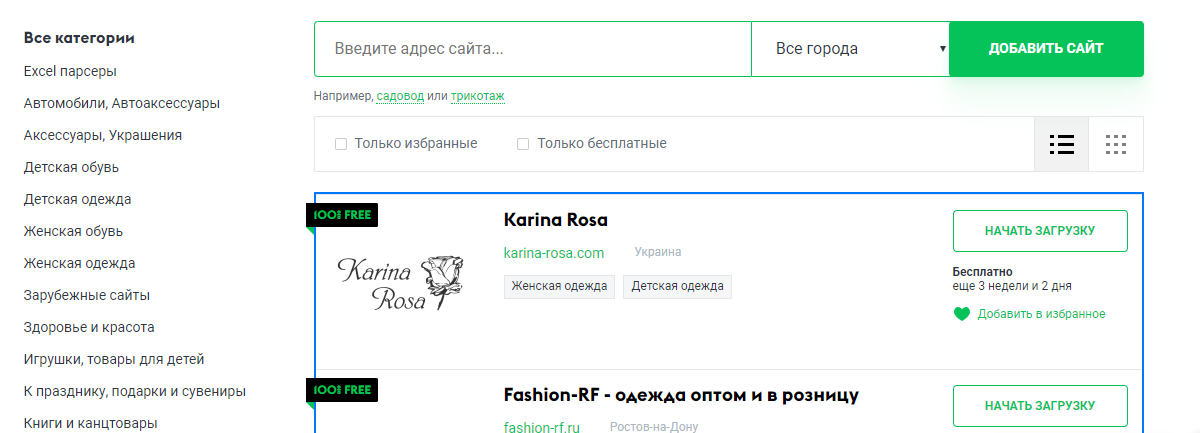


Рисунок 1.1 – Інтерфейс додатку Q-Parser

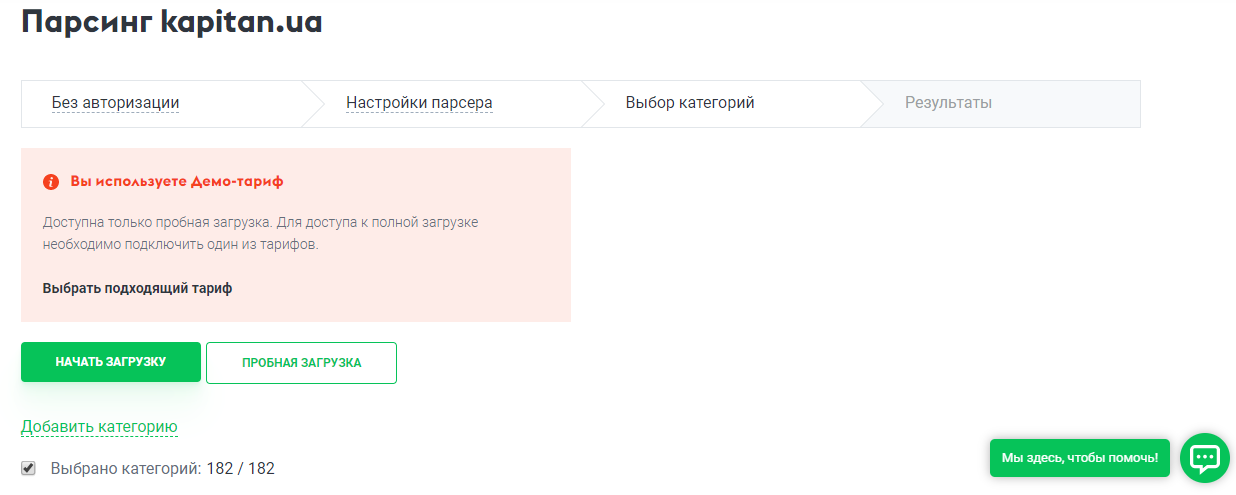


Рисунок 1.2 – Інтерфейс додатку Q-Parser

Даний додаток спрямований на вивантаження товарів у магазинах.

Додаток має наступні функції:

* реєстрація та авторизація користувача;
* система повідомлень;
* накладення тексту на фото товарів;
* вибір категорій для парсингу;
* продовження роботи в умові виходу з сайту;
* запуск парсингу за розкладом;
* фільтрація товарів перед вивантаженням;
* вивантаження у різні формати;
* вивантаження у соцмережі.

До недоліків даного сайту можна віднести платне користування.

### 1.2.2 Додаток TargetHunter

Додаток розроблений для користувачів комп’ютерів та мобільних пристроїв, що надає можливість його використання у зручний час [2].

Інтерфейс додатку зображений на рисунках 1.3 та 1.4.

Сервіс не тільки збирає групи аудиторій Вконтакте, але і завантажує їх в бази ретаргетінга. При цьому всі зібрані списки можна зберегти в хмару і повернутися до них у будь-який зручний час, навіть будучи офлайн.

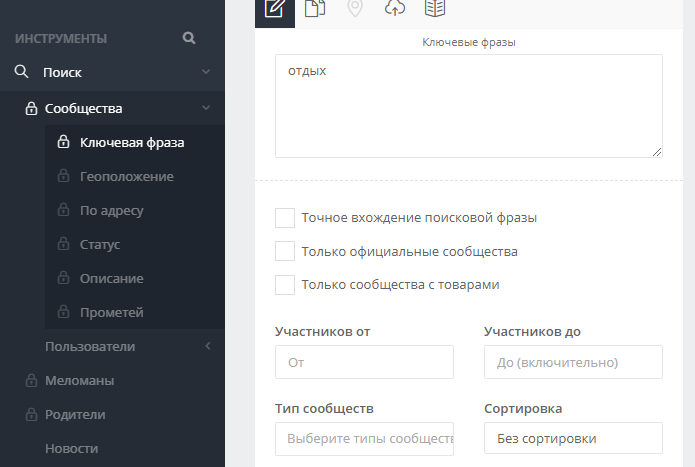


Рисунок 1.3 – Інтерфейс додатку TargetHunter

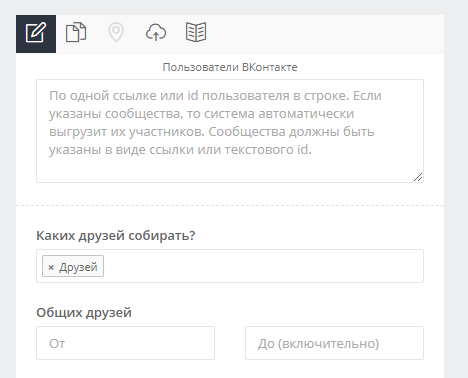


Рисунок 1.4 – Інтерфейс додатку TargetHunter

Додаток має наступні функції:

* реєстрація та авторизація користувача;
* автозбереження записів;
* автоматизація;
* аналітика;
* цілодобова підтримка;
* безкоштовний тариф.

Одним з найважливіших недоліків цього сайту є не дуже зрозумілий для недосвідченого користувача інтерфейс та інтуїтивно не зрозумілий функціонал. Також цей сервіс є здебільшого платним.

### 1.2.3 Додаток CleverTarget

Парсер аудиторії соцмережі Вконтакте

Інтерфейс додатку зображений на рисунках 1.5 та 1.6.

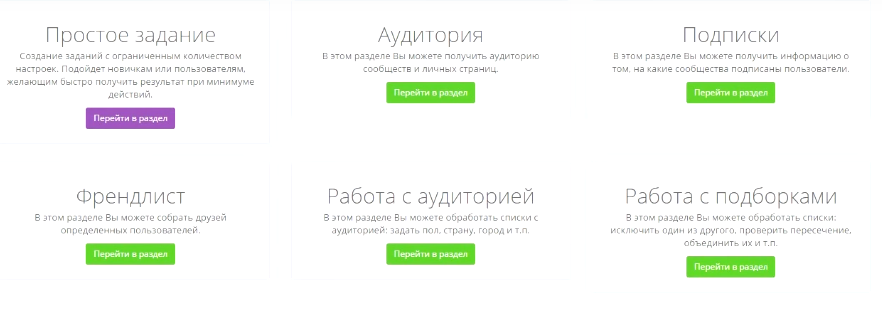


Рисунок 1.5 – Інтерфейс додатку CleverTarget

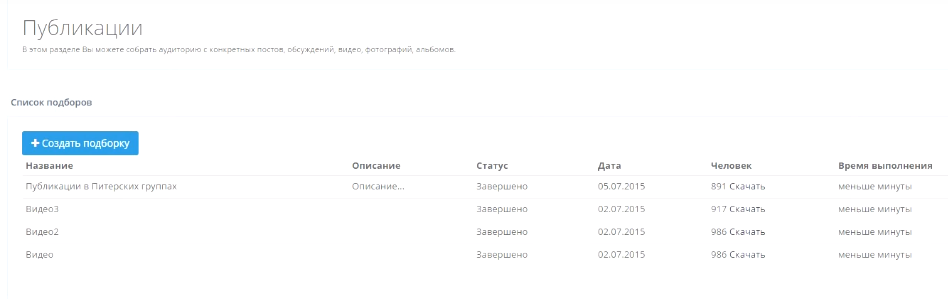


Рисунок 1.6 – Інтерфейс додатку CleverTarget

Додаток має наступні функції:

* пошук аудиторії;
* пошук здійснюється за різними критеріями.
* відстежування нових коментарів та повідомленнь в групах ВКонтакте;
* щоб збирати аудиторію, необов'язково бути онлайн;
* всі дані і звіти зберігаються в хмарі, що дозволяє повертатися до них в будь-який зручний час.
* сервіс дозволяє в тому числі створювати оголошення;
* прогнозування ставок для розміщення оголошення за допомогою штучного інтелекту;
* парсер представляє інструмент для аналізу конкурентів.

### 1.2.4 Додаток Діггернаут

Діггернаут - це хмарний сервіс для парсинга сайтів, збору інформації та інших ETL (Extract, Transform, Load) завдань. [4].

Все, що потрібно зробити - створити парсер (дігер), крихітного робота, який буде аналізувати довільні сайти по запиту, витягувати дані, нормалізувати і обробляти їх, зберігати масиви даних в хмарі, звідки їх можна завантажити в будь-якому з доступних форматів (наприклад , CSV, XML, XLSX, JSON) або забрати в автоматичному режимі через API.

Інтерфейс додатку зображений на рисунках 1.7 та 1.8.

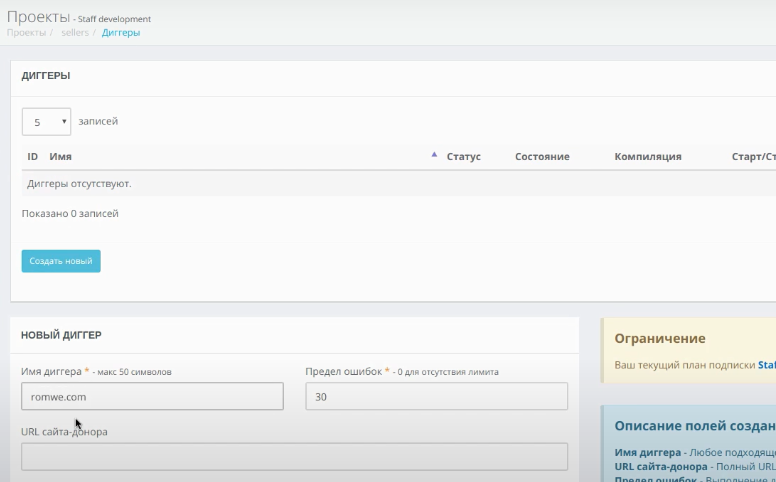


Рисунок 1.7 – Інтерфейс додатку Діггернаут

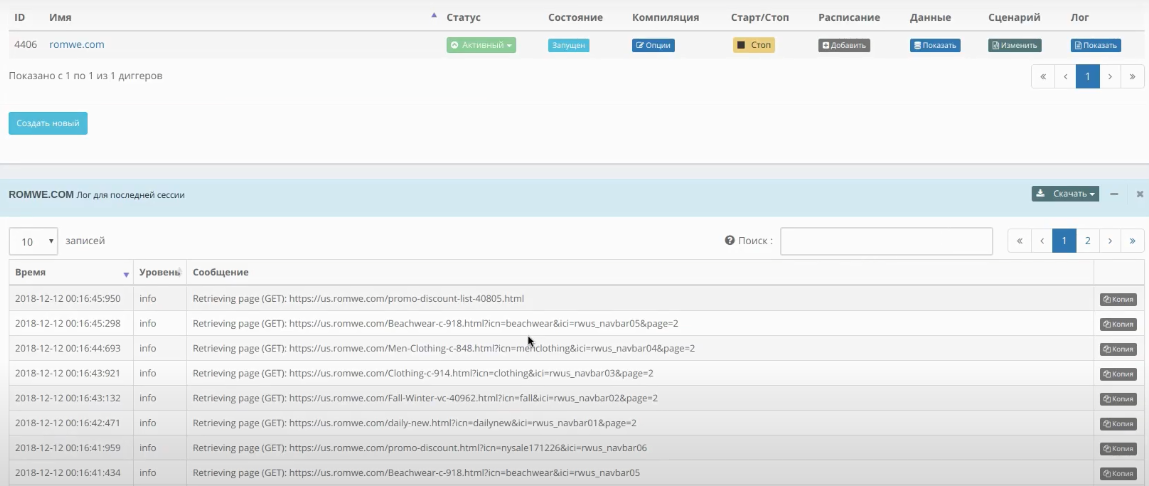


Рисунок 1.8 – Інтерфейс додатку Діггернаут

Додаток має наступні функції:

* безкоштовний план;
* швидкий збір даних;
* збір даних з декількох сайтів;
* своя мета-мова для розробки власних парсерів.

Таблиця 1.1 – Порівняння додатків

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Критерій порівняння | Q-Parser | TargetHunter | CleverTarget | Діггернаут |
| Дизайн | Сучасний, яскравий | Хороший, простий, приємний для ока | Стильний, сучасний | Задовільний |
| Рівень функціональності | Високий | Високий | Хороший | Високий |
| Простота і зрозумілість | + | + | + | – |
| Платна версія | + | + | + | + |

Всі представлені додатки задовольняють мінімальним критеріям.

Розроблюваний веб-додаток має простий, мінімалістичний та інтуїтивно зрозумілий інтерфейс. Проста авторизація для користувача дає змогу швидко приступити до роботи з додатком. У порівнянні з іншими онлайн парсерами, додаток не має реклами. Великою перевагою є можливість працювати у режимі оффлайн.

Аналізуючи статистику відвідування сайтів виробників додатків та скачування додатків, зв’язаних з парсингом, можна прийти до висновку, що парсинг сайтів в нинішній час є досить актуальним і популярним. Дивлячись на розвиток розробки парсерів, ми бачимо, що розробка додатків даного типу втратити актуальність ближнім часом не зможе.

## Постановка задачі

Веб додаток з сканування сеансів повинен мати наступний функціонал:

* можливість входу до системи;
* збереження введеної інформації і неможливість її втрати;
* можливість оповіщення користувачів про зміни у кінотеатрах.

Метою роботи є створення веб-додатку для декількох користувачів. Програмний продукт повинен бути доступним для користування людей різних вікових категорій, а саме бути простим і зрозумілим.

Відповідно до поставленої мети і вимогами були сформульовані наступні завдання:

* розробка моделі даних;
* вибір СУБД і засобів розробки веб-додатку;
* розробка веб-застосунку;
* розробка простого і зручного інтерфейсу;
* створення можливості редагування, додавання, перегляду і видалення записів;

Для досягнення поставлених завдань виділимо наступні задачі:

* вивчення принципів побудови веб-додатків;
* дослідження структури додатку;
* аналіз програмних засобів побудови веб-додатків;
* тестування веб-додатку;
* обґрунтування ефективності даного проекту;

Предмет дослідження – технології .NET для створення системи контролю процесу розробки веб-додатку «TicketsHere».

# 2 СПЕЦИФІКАЦІЯ ВИМОГ ДО ВЕБ-ДОДАТКУ

## 2.1 Загальна інформація щодо веб-застосунку

Веб-додаток “TicketsHere” призначений для надання можливості моніторингу наявності вільних місць у кіно, а також для моніторингу нових сеансів у кінотеатрах. Введені дані зберігаються в інформаційній базі даних, що дозволяє отримувати звіти з динаміки змін всіх показників за вибраний інтервал часу.

### 2.1.1 Огляд продукту

«TicketsHere» – це веб-додаток, який допоможе користувачу зручно відстежувати повернуті квитки. Контекстна діаграма (рис 2.1) показує зовнішній об'єкт і системні інтерфейси.

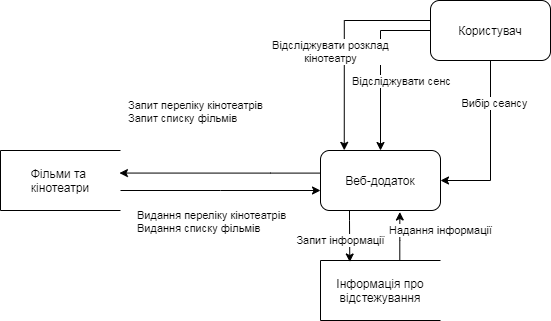


Рисунок 2.1 – Контекстна діаграма додатку

Передбачаються випуски кількох версій системи для отримання універсального веб-додатку парсера, який дозволяв би користувачеві налаштовувати інформацію, що надається у повідомленні, а також обмежить можливість парсингу багатьох сеансів для одного користувача.

### 2.1.2 Мета

Мета проекту – вдосконалити навички розробки веб-додаткі використовуючи фреймворк ASP .NET Core, розробити зручний додаток для подальшого застосування.

### 2.1.3 Межі

Використання веб-додатку поширюється на всі країни.

Існують, встановлені межі відповідно яким, парсинг сайту відбувається не частіше ніж один раз за 30 секунд. Це дозволяє не спричиняти велику загрузку на сервер-постачальник.

### 2.1.4 Масштаб проекту

Веб-додаток дозволить користувачам додавати дані через інтернет, які представляють із себе звичайні записи з налаштуваннями для моніторингу того чи іншого сеансу. Повідомлення надходять до Telegram месенджера, що виносить досвід користування на новий рівень.

Функції, реалізація яких запланована у першій версії:

* вхід за допомогою аккаунту Google, а також локальна авторизація;
* забезпечення доступу до веб-додатку через мережу інтернет;
* додавання нових користувачів до бази даних;
* редагування інформації про користувача;
* мінімалістичній набір функціоналу для реєстрації нової задачі парсингу та можливості повідомлень про зміни.

### 2.1.5 Обмеження реалізації

Веб-додаток створений за допомогою HTML 5.0, CSS3, JavaScript, стеку технологій .NET.

### 2.1.6 Діаграми потоків даних

Для опису присутньої функціональності була побудована діаграма потоків даних (DFD – data flow diagram) (рис 2.2).

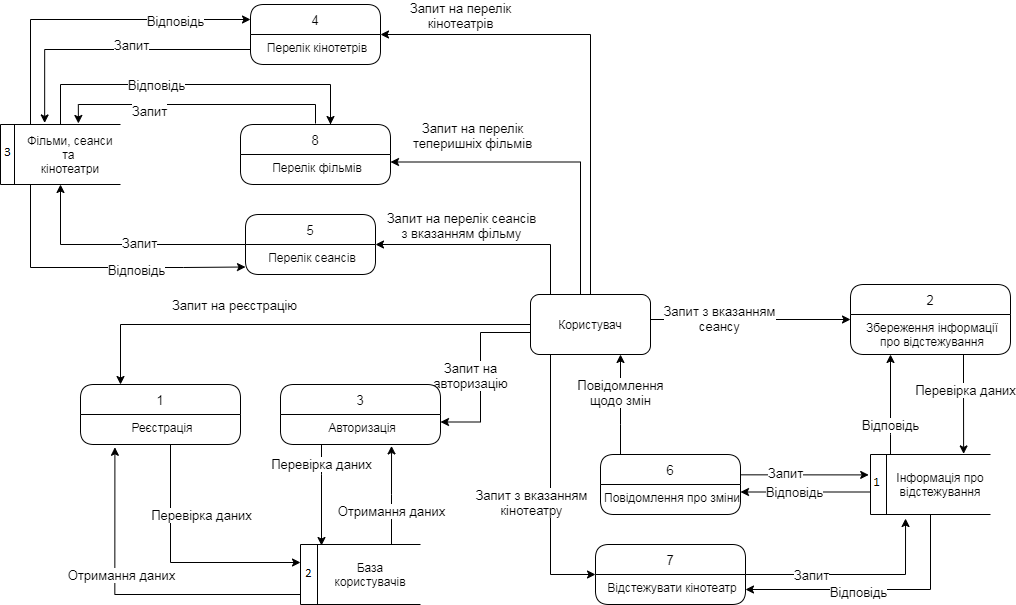


Рисунок 2.2 – Діаграма потоків даних (користувач)

Діаграми потоків даних описують зовнішні по відношенню до системи джерела і адресати даних, логічні функції, потоки даних та сховища даних, до яких здійснюється доступ.

Інформаційна система приймає ззовні потоки даних. Для позначення елементів середовища функціонування системи використовується поняття зовнішньої сутності. Всередині системи існують процеси перетворення інформації, які породжують нові потоки даних. Потоки даних можуть надходити на вхід до інших процесів, розміщуватися (і вилучатись) в накопичувачі даних, передаватися до зовнішніх сутностей.

## 2.2 Вимоги до веб-додатку

### 2.2.1 Функціональні вимоги

Функціональні вимоги наведені в таблиці 2.1, де під об’єктом мається на увазі «користувач». Так як додаток розрахований на декількох користувачів, тому є опція вибору «об’єкта».

Таблиця 2.1 – Опис функціональних вимог до веб-додатку

|  |  |
| --- | --- |
| **Вимога** | **Описання** |
| Об’єкт. ДоданняСеансу | Для додання нового запису моніторингу сеансу, користувач має використати форму додання нового запису, заповнити поля та натиснути на кнопку «Отслеживать». Після додання, новий запис буде внесений до списку. |
| Об’єкт. ДоданняКінотеатру | Для додання нового запису моніторингу кінотеатру, користувач має відкрити список кінотеатрів та натиснути кнопку «Подписаться». Після натискання, новий запис буде внесений до бази. |
| Об’єкт. ВидаленняКінотеатру | Для видалення запису моніторингу кінотеатра, користувач має відкрити список кінотеатрів та натиснути кнопку «Отписаться». Після натискання, запис буде видалений з бази. |
| Реєстрація.Клієнт.Відмова | При виявлені даного клієнта в базі даних, завершити транзакцію зі статусом «Даний клієнт уже зареєстрований». |
| Вхід.Система | Для входу до системи, користувач має ввести логін і пароль в форму входу. |
| Вхід.Система.Підтвердження | При збігу даних завершити транзакцію зі статусом «Вхід виконано успішно». |
| Вхід.Система.Відмова | При розбіжності даних завершити транзакцію зі статусом «Невірні дані» |
| ОбліковийЗапис.Користувач | Клієнт має можливість вийти з власного облікового запису. |

Діаграму варіантів використання можна зображено на рисунку 2.3.

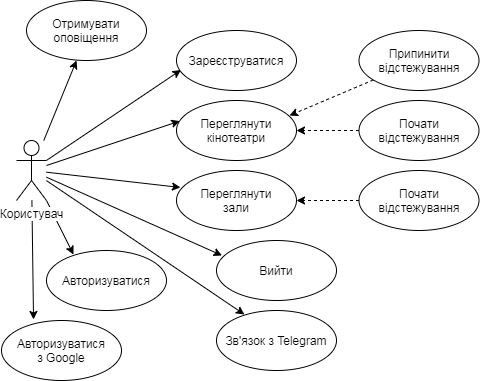


Рисунок 2.3 – Діаграма варіантів використання

### 2.2.2 Нефункціональні вимоги

Якщо з'єднання між користувачем і системою розривається до того, як дані були додані, змінені або видалені, система веб-додатку повинна дозволити користувачеві відновити дані для перегляду.

Веб-додаток «TicketsHere» повинен бути доступним для всіх користувачів через мережу інтернет. Веб-додаток буде працювати 24 години на добу, тобто не має часових обмежень.

Читабельність коду можна розглядати як важливу складову супроводу. Код повинен бути зручним для аналізу, змін та тестування.

Модифікація, яка здійснюється для усунення дефектів, удосконалення або адаптації веб-додатку до змін в умовах функціонування, a також в складі і особливостях необхідних функцій є також одним із важливих факторів.

Додавання, видалення, зміна даних про об’єкти та їх показники повинні бути зручними для користувача веб-додатку.

Веб-додаток повинен мати властивість кросбраузерності.

Веб-додаток інсталяції не потребує, так як він буде доступний користувачеві за допомогою мережі інтернет через браузер.

Веб-додаток повинен мати відмінну швидкість доступу до бази даних та завантаження сторінок, не навантажувати процесор та оперативну пам'ять.

## 2.3 Опис обраних технологій для розробки веб-додатку

### 2.3.1 HTML5

HTML5 є найновішим стандартом HTML. Цей термін представляє дві різні концепції:

* це нова версія мови HTML, з новими елементами, атрибутами і поведінкою;
* це великий стек технологій, котрі надають більшого різноманіття та потужності Веб-сайтам та додаткам. Цей набір інколи називають HTML5 & friends, але часто скорочують до HTML5.

Спроектована, для використання всіма розробниками Відкритого Вебу (Open Web), ця сторінка надає посилання до численних ресурсів про HTML5 технологї, розсортованні в декілька груп, в залежності від їх функцій.

* семантика: дозволяє описати якомога точніше з чого складається ваш контент;
* зв'язок: дозволяє вам взаємодіяти з сервером новим та інноваційним шляхом;
* оффлайн та сховище: надає можливість веб-сторінкам зберігати дані на стороні клієнта локально та оперувати оффлайном ефективніше;
* мультимедіа: робить відео та аудіо першокласними мешканцями Відкритого Вебу;
* 2D/3D графіка та ефекти: дозволяє безліч презентаційних варіантів;
* ефективність та інтеграція: надає більшу оптимізацію швидкості та кращого використання заліза (hardware) комп'ютера;
* доступ до пристрою: дозвляє використання різноманітних пристроїв вводу/виводу;
* стилі: Дозволяє авторам створювати більш складні і витончені теми.

Переваги HTML5:

* легкість створювання програмного код;
* використання елементів article, footer, nav, header та ін.;
* у вікні браузера можна відтворювати Flash (анімації, відеоролики, музику) і для цього не потрібні драйвера і утиліти;
* процес обміну даними з сервером став поліпшеним;
* отримання оповіщень про помилки при роботі сайту;
* Недоліки HTML:
* щоб створювати якісні проекти, доводиться серйозно попрацювати;
* при замовленні сайтів на HTML5, доведеться платити більше;
* деякі фахівці стверджують, що безпека після застосування нової технології стала нижче;
* розробники браузерних платформ намагаються не відставати від сучасних технологій, тим не менш, є деякі програми, в яких не вірно відображаються сайти на HTML5.

В цілому дана технологія є відмінною заміною застарілого HTML.

### 2.3.2 CSS3

CSS (англ. Cascading Style Sheets – каскадні таблиці стилів) – формальна мова опису зовнішнього вигляду документа, написаного з використанням мови розмітки.

Специфікація CSS3 - це незаперечне майбутнє в області декоративного оформлення веб-сторінок, і її розробка ще далека від завершення. Більшість модулів все ще продовжує удосконалюватися і модифікуватися, і жоден браузер не підтримує всі модулі. Це означає, що CSS3 зазнає таких самих складностей, як і HTML5. Веб-розробникам потрібно вирішувати, які можливості використовувати, а які ігнорувати, а також яким чином заповнити прогалини в браузерній підтримці.

Специфікація CSS3 не є частиною специфікації HTML5. Ці два стандарти були розроблені окремо один від одного, різними людьми, які працювали в різний час в різних місцях. Але навіть організація W3C закликає веб-розробників використовувати HTML5 та CSS3 разом, як частина однієї нової хвилі сучасного веб-дизайну.

Переваги CSS:

1. Простота самої мови CSS разом з принципом відділення оформлення від змісту дає можливість скоротити час на розробку і підтримку сайту.
2. Є кілька варіантів дизайну сторінки для перегляду на різних пристроях. Наприклад, дизайн на екрані комп'ютера розрахований на одну ширину, і буде повністю виводитися на екран, а на мобільних пристроях він буде підлаштовуватися до розмірів екрану і деякі елементи будуть виключені від показу, також і при друці, буде друкуватися потрібний текст, без зайвого (наприклад, без шапки меню).
3. Зменшується час завантаження сторінок web-сайту за рахунок перенесення правил представлення даних в окремий CSS-файл. Завдяки цьому браузер завантажує тільки структуру документа, а також дані, що зберігаються на сторінці, а представлення цих даних завантажується браузером тільки один раз і може бути закешований, - завдяки цьому зменшується трафік, час завантаження, а також навантаження на сервер.
4. Простота зміни дизайну. Один CSS управляє відображенням безлічі HTML-сторінок. Коли виникає необхідність змінити дизайн сайту, то нема чого правити кожну сторінку. Для подальшої зміни дизайну всього лише потрібно змінити CSS-файл, і як результат, зміна дизайну робиться швидше.
5. CSS надає додаткові можливості форматування, про яких при використанні тільки самих атрибутів навіть і не мріяли.
6. Підвищення сумісності з різними платформами за рахунок використання web-стандартів.

Недоліки CSS:

1. Різне відображення верстки в різних браузерах. Якщо браузери застарілі, то можливо, що одні й ті ж дані CSS по-різному ними інтерпретуються.
2. Необхідність виправляти не тільки один CSS-файл, але і теги HTML. Часто зустрічається на практиці необхідність виправляти не тільки один CSS-файл, але і теги HTML, які пов'язані з селекторами CSS. Іноді це значно збільшує час редагування, а також і тестування.

Отже, CSS – це сильний інструмент і один з основних складових практично будь-якого web-сайту. Без CSS просто неможливо уявити сучасну діяльність web-розробників. Сьогодні CSS – це загальноприйнятий стандарт розробки, який приймається усіма без винятку компаніями-розробниками, що явно показує його значимість і необхідність.

### 2.3.3 ASP.NET Core

ASP.NET Core — вільне та відкрите програмне забезпечення каркасу веб застосунків, з продуктивністю вищою ніж у ASP.NET, розроблена корпорацією Microsoft і співтовариством. Це модульна структура, яка працює як на повній платформі .NET Framework, так і на платформі .NET Core.

Фреймворк являє собою повний перепис, який об'єднує раніше окремі ASP.NET MVC та ASP.NET Web API у єдину програмувальну модель.

Не зважаючи на те, що це є новим фреймворком, побудованим на новому веб-стеку, ASP.NET Core має високий ступінь сумісності концепцій з ASP.NET MVC, який об'єднує функціональність MVC, Web API та Web Pages. В попередніх версіях платформи дані технології реалізовані окремо і тому містять багато дублювальної функціональності. Тепер це об'єднано в одну програмну модель ASP.NET Core MVC. Програми ASP.NET Core підтримують програмні версії, в якій різні програми, що працюють на одному комп'ютері, можуть орієнтуватися на різні версії ASP.NET Core. Це не можливо з попередніми версіями ASP.NET Core.

До особливостей цього фреймворку відносять:

* модульна структура розподіляється як NuGet пакунки;
* сloud-optimized runtime (оптимізована для Інтернету);
* хост-агностик за допомогою Відкритого Веб-Інтерфейсу для .Net (OWIN) підтримки - працює в IIS або в автономному режимі;
* єдина історія для створення веб UI і веб APIs (тобто обидва ті самі);
* система створення конфігурації середовища на основі хмар;
* легкий і модульний HTTP запит;
* створення та запуск крос-платформних додатків ASP.NET Core у Windows, Mac та Linux;
* відкрите джерело та орієнтоване на спільноту;
* пряме прикріплення версії додатка при націлюванні на .NET Core.

Також ASP.NET Core включає EF Core (Entity Framework (EF) Core ) – ORM для роботи з даними.

EF Core - це проста, кроссплатформена і розширюєма версія популярної технології доступу до даних Entity Framework з відкритим вихідним кодом.

EF Core може використовуватися як об'єктно реляційний модуль маппінгу (ORM), дозволяючи розробникам .NET працювати з базою даних за допомогою об'єктів .NET і усуваючи необхідність в написанні здебільшого коду, необхідного для доступу до даних.

### 2.3.4 Microsoft Visual Studio

Microsoft Visual Studio — серія продуктів фірми Майкрософт, які 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, а також веб-сайти, веб-застосунки, веб-служби як в рідному, так і в керованому кодах для всіх платформ [8].

Переваги Microsoft Visual Studio:

1. Вбудований Web-сервер. Для обслуговування веб-додатків ASP.NET Core необхідний Web-сервер, який очікує веб-запити і обробляє відповідні сторінки. Наявність в Visual Studio інтегрованого Web-сервера дозволяє запускати веб-сайт прямо з середовища проектування, а також підвищує безпеку, виключаючи можливість отримання доступу до тестового Web-сайту з будь-якого зовнішнього комп'ютера, оскільки тестовий сервер може приймати з'єднання тільки з локального комп'ютера.
2. Підтримка безлічі мов при розробці. Visual Studio дозволяє писати код у своїй мові або будь-яких інших переважних мов, використовуючи весь час один і той же інтерфейс (IDE). Більше того, Visual Studio також дозволяє створювати веб-сторінки на різних мовах. Єдиним обмеженням є те, що в кожній Web-сторінці можна використовувати лише одну мову.
3. Менше коду для написання. Для створення більшості додатків потрібна достатня кількість стандартного стереотипного коду, а також Web-сторінки ASP. NET тому не виключення. Наприклад, додавання Web-елемента управління, Приєднання обробників подій і корегування форматування вимагає встановлення в розмітці сторінки ряду деталей. У Visual Studio такі деталі встановлюються автоматично.
4. Інтуїтивний стиль кодування. За замовчуванням Visual Studio форматує код за його введенням, автоматично вставляє необхідні відступи і застосовує колірне кодування для виділення елементів типу коментарів. Такі незначні відмінності роблять код більш зручним для читання та менш піддається помилці. Приймаються Visual Studio автоматично параметри форматування, які можна навіть налаштовувати, що дуже зручно.
5. Більш висока швидкість розробки. Багато з функціональних можливостей Visual Studio направлені на те, щоб допомогти розробнику зробити свою роботу швидше. Зручні функції, такі як функції IntelliSense (яка здатна відловлювати помилки та пропонувати правильні варіанти), функції пошуку та заміни (яка дозволяє шукати ключові слова як в одному файлі, так і в цілому проекті) та функції автоматичного додавання та видалення коментарів (які можуть тимчасово закрити блоки коду), дозволяють розробнику працювати швидко і ефективно.
6. Можливості налагодження. Параметри інструментів, що пропонуються в Visual Studio, є найкращим засобом для відстеження загадкових помилок та діагностики дивної поведінки.

Visual Studio також має й інші функції: можливість управління проектом; вбудована функція управління вихідним кодом; можливість рефакторизації коду; потужна модель розширюваності. Більше того, у випадку використання Visual Studio 2008 Team System розробник отримує розширені можливості для модульного тестування, спільної роботи та управління версіями коду (що значно більше того, що пропонується в простих інструментах, як Visual SourceSafe).

До недоліків Microsoft Visual Studio можна відзначити неможливість відладчика (Microsoft Visual Studio Debugger) відстеження в коді режиму ядра. Налагодження в Windows в режимі ядра в загальному випадку виконується при використанні WinDbg, KD або SoftICE.

### 2.3.5 JavaScript

JavaScript (JS) — динамічна, об'єктно-орієнтована прототипна мова програмування. Реалізація стандарту ECMAScript. Найчастіше використовується для створення сценаріїв веб-сторінок, що надає можливість на стороні клієнта (пристрої кінцевого користувача) взаємодіяти з користувачем, керувати браузером, асинхронно обмінюватися даними з сервером, змінювати структуру та зовнішній вигляд веб-сторінки.

JavaScript класифікують як прототипну (підмножина об'єктно-орієнтованої), скриптову мову програмування з динамічною типізацією. Окрім прототипної, JavaScript також частково підтримує інші парадигми програмування (імперативну та частково функціональну) і деякі відповідні архітектурні властивості, зокрема: динамічна та слабка типізація, автоматичне керування пам'яттю, прототипне наслідування, функції як об'єкти першого класу.

## 2.3.6 MySQL

MySQL був розроблений компанією «ТсХ» для підвищення швидкодії обробки великих баз даних. Ця система керування базами даних (СКБД) з відкритим кодом була створена як альтернатива комерційним системам. MySQL з самого початку була дуже схожою на mSQL, проте з часом вона все розширювалася і зараз MySQL — одна з найпоширеніших систем керування базами даних. Вона використовується, в першу чергу, для створення динамічних веб-сторінок, оскільки має чудову підтримку з боку різноманітних мов програмування.

MySQL — компактний багатопотоковий сервер баз даних. Характеризується високою швидкістю, стійкістю і простотою використання.

MySQL вважається гарним рішенням для малих і середніх застосувань. Сирцеві коди сервера компілюються на багатьох платформах. Найповніше можливості сервера виявляються в UNIX-системах, де є підтримка багатопоточності, що підвищує продуктивність системи в цілому.

Можливості сервера MySQL:

* простота у встановленні та використанні;
* підтримується необмежена кількість користувачів, що одночасно працюють із БД;
* кількість рядків у таблицях може досягати 50 млн;
* висока швидкість виконання команд;
* наявність простої і ефективної системи безпеки.

## 2.4 Опис архітектури веб-додатку

### 2.4.1 Діаграма розміщення

Діаграми розгортання (рис. 2.4) представляють фізичне розташування системи, показуючи, на якому фізичному обладнанні запускається та чи інша складова програмного забезпечення.

Вузол (node) – це те, що може містити програмне забезпечення. Вузли бувають двох типів. Пристрій (device) – це фізичне обладнання: комп'ютер або пристрій, пов'язаний з системою. Середовище виконання (execution environment) – це програмне забезпечення, яке саме може включати інше програмне забезпечення, наприклад операційну систему або процес-контейнер.

Вузли можуть містити артефакти (artifacts), які є фізичним уособленням програмного забезпечення; зазвичай це файли. Такими файлами можуть бути виконувані файли (такі як файли .eхе, виконавчі файли, файли DLL, файли JAR, складання або сценарії) або файли даних, конфігураційні файли, HTML-документи і т. д. Перелік артефактів всередині вузла вказує на те, що на даному вузлі артефакт розгортається в систему, яка запускається.

Артефакти можна зображувати у вигляді прямокутників класів або перераховувати їх імена всередині вузла.

Артефакти часто є реалізацією компонентів. Це можна показати, задавши значення-мітки всередині прямокутників артефактів.

Інформаційні шляхи між вузлами представляють обмін інформацією в системі. Можна супроводжувати ці шляхи інформацією про використання інформаційних протоколів.

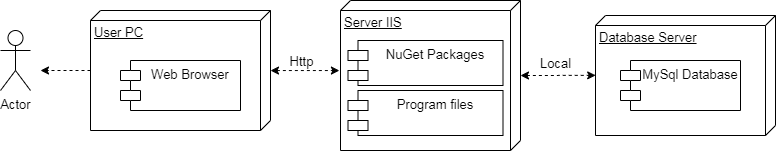


Рисунок 2.4 – Діаграма розміщення

### 2.4.2 Діаграма компонентів

На діаграмі компонентів (рис. 2.5) показаний вид «білого ящика» внутрішньої структури двох пов'язаних підсистем – «Controllers» і «DatabaseContext». В UML «підсистема» представляє собою стандартний компонентний стереотип для більших компонентів, які в свою чергу містять дрібніші компоненти.

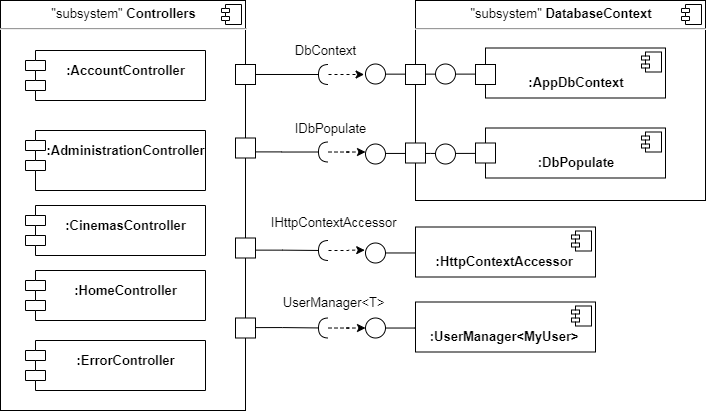


Рисунок 2.5 – Діаграма компонентів

Підсистема «Controllers» містить декілька модулів-контроллерів. Компонент «Controllers» дозволяє управляти елементами, використовуючи необхідні інтерфейси DbContext, IDbPopulate, IHttpContextAccessor, UserManager<T>, що надаються компонентами DatabaseContext, HttpContextAccessor, UserManager<MyUser>.

### 2.5 Модель бази даних

База даних, складається з дев’ятнадцяти сутностей серед яких основні одинадцять це: CinemaHalls, Cinemas, Movies, Seats, ShowTimes, Users, Trackings, UserRoles, Technologies, SeatTypes, MovieTypes. Модель бази даних представлена на рисунку 2.6.

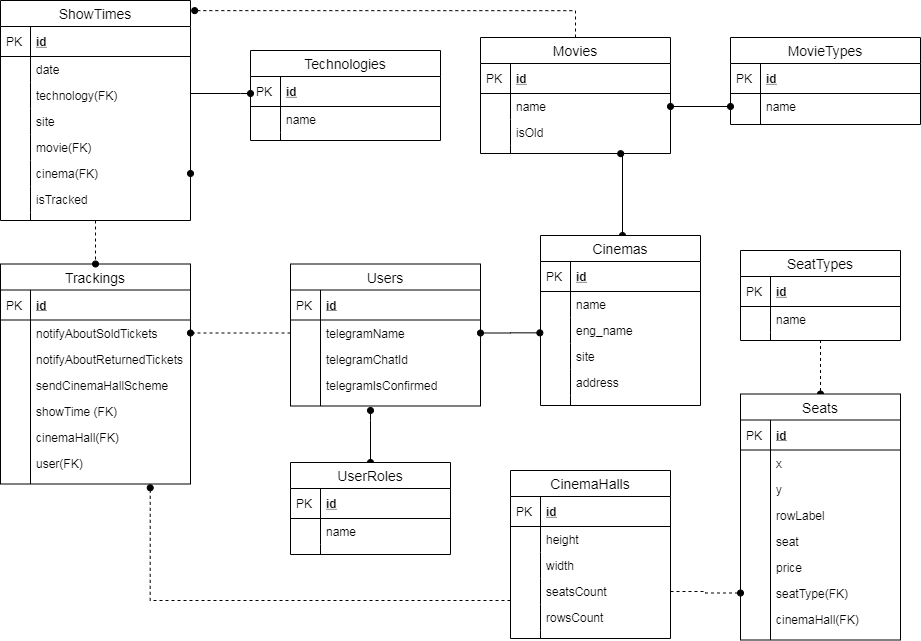


Рисунок 2.6 – Модель бази даних

Таблиця «User» призначена для зберігання інформації про користувачів веб-додатку. Кожний запис складається з наступних полів, опис яких наведений в таблиці 2.2.

Таблиця 2.2 ­ – Опис структури таблиці «User»

|  |  |  |
| --- | --- | --- |
| **Ім’я поля** | **Тип даних** | **Опис** |
| telegramName | Строковий | Ім’я у мережі Telegram |
| telegramChatId | Числовий | Id чату у мережі Telegram |
| telegramConfirmed | Логічний | Чи підтверджений Telegram акаунт |

Таблиця «UserRoles» призначена для зберігання інформації про ролі користувачів веб-додатку (адміністратор чи звичайний користувач). Кожний запис складається з наступних полів, опис яких наведений в таблиці 2.3.

Таблиця 2.3 ­ – Опис структури таблиці «UserRoles»

|  |  |  |
| --- | --- | --- |
| **Ім’я поля** | **Тип даних** | **Опис** |
| name | Строковий | Ім’я ролі |

Таблиця «Movies» призначена для зберігання інформації про фільми. Кожний запис складається з наступних полів, опис яких наведений в таблиці 2.4.

Таблиця 2.4 ­ – Опис структури таблиці «Movies»

|  |  |  |
| --- | --- | --- |
| **Ім’я поля** | **Тип даних** | **Опис** |
| name | Строковий | Назва фільму |
| isOld | Логічний | Чи вийшов фільм з прокату |

Таблиця «MovieTypes» призначена для зберігання інформації про типи фільму (боєвик, екшн). Кожний запис складається з наступних полів, опис яких наведений в таблиці 2.5.

Таблиця 2.5 ­ – Опис структури таблиці «MovieTypes»

|  |  |  |
| --- | --- | --- |
| **Ім’я поля** | **Тип даних** | **Опис** |
| name | Строковий | Ім’я типу фільму |

Таблиця «Cinemas» призначена для зберігання інформації про кінотеатри. Кожний запис складається з наступних полів, опис яких наведений в таблиці 2.6.

Таблиця 2.6 ­ – Опис структури таблиці «Cinemas»

|  |  |  |
| --- | --- | --- |
| **Ім’я поля** | **Тип даних** | **Опис** |
| name | Строковий | Назва кінотеатру |
| eng\_name | Строковий | Англійська назва кінотеатру |
| site | Строковий | Сторінка інформації про кінотеатр на сайті, з якого прозводиться парсинг |
| address | Строковий | Адреса кінотеатру |

Таблиця «Seats» призначена для зберігання інформації про міста в залі. Кожний запис складається з наступних полів, опис яких наведений в таблиці 2.7.

Таблиця 2.7 ­ – Опис структури таблиці «Seats»

|  |  |  |
| --- | --- | --- |
| **Ім’я поля** | **Тип даних** | **Опис** |
| x | Числовий | Позиція по горизонталі |
| y | Числовий | Позиція по вертикалі |
| rowLabel | Числовий | Номер ряду |
| seat | Числовий | Номер сидіння у ряді |
| price | Числовий | Ціна міста |
| seatType(FK) | Посилальний | Посилання на тип (продане, вільне) |
| cinemaHall(FK) | Посилальний | Посилання на залу |

Таблиця «SeatTypes» призначена для зберігання інформації про типи міст у залі. Кожний запис складається з наступних полів, опис яких наведений в таблиці 2.8.

Таблиця 2.8 ­ – Опис структури таблиці «SeatTypes»

|  |  |  |
| --- | --- | --- |
| **Ім’я поля** | **Тип даних** | **Опис** |
| name | Строковий | Тип міста |

Таблиця «ShowTimes» призначена для зберігання інформації про сесії кіно. Кожний запис складається з наступних полів, опис яких наведений в таблиці 2.9.

Таблиця 2.9 ­ – Опис структури таблиці «ShowTimes»

|  |  |  |
| --- | --- | --- |
| **Ім’я поля** | **Тип даних** | **Опис** |
| date | Час | Час показу |
| technology(FK) | Посилальний | Посилання на технологію (3D, IMAX …) |
| site | Строковий | Сайт для відстежування |
| movie(FK) | Посилальний | Посилання на фільм |
| cinema(FK) | Посилальний | Посилання на кінотеатр |
| isTracked | Логічний | Чи потрібно відстежувати сесію |

Таблиця «Technologies» призначена для зберігання інформації про технології, що використовуються при показі. Кожний запис складається з наступних полів, опис яких наведений в таблиці 2.10.

Таблиця 2.10 ­ – Опис структури таблиці «Technologies»

|  |  |  |
| --- | --- | --- |
| **Ім’я поля** | **Тип даних** | **Опис** |
| name | Строковий | Назва технології |

Таблиця «Trackings» призначена для зберігання інформації про відстежування сеансів. Кожний запис складається з наступних полів, опис яких наведений в таблиці 2.11.

Таблиця 2.11 ­ – Опис структури таблиці «Trackings»

|  |  |  |
| --- | --- | --- |
| **Ім’я поля** | **Тип даних** | **Опис** |
| notifyAboutSoldTickets | Логічний | Чи потрібно повідомляти про продані квитки |
| notifyAboutReturnedTickets | Логічний | Чи потрібно повідомляти про повернуті квитки |
| sendCinemaHallScheme | Логічний | Чи потрібно відправляти схему зали |
| showTime(FK) | Посилальний | Посилання на сесію кіно |
| cinemaHall(FK) | Посилальний | Посилання на залу |
| user(FK) | Посилальний | Посилання на користувача |

Таблиця «CinemaHalls» призначена для зберігання інформації про зали. Кожний запис складається з наступних полів, опис яких наведений в таблиці 2.12.

Таблиця 2.12 ­ – Опис структури таблиці «CinemaHalls»

|  |  |  |
| --- | --- | --- |
| **Ім’я поля** | **Тип даних** | **Опис** |
| height | Числовий | Висота |
| width | Числовий | Довжина |
| seatsCount | Числовий | Кількість міст |
| rowsCount | Числовий | Кількість рядів |

В результаті було сформовано закінчений опис поведінки та структури веб-додатку який потрібно розробити для кінцевого користувача. Виконуючи специфікацію вимог, були визначені і описані функції системи, а також основні вимоги до зовнішнього інтерфейсу.

# 3 ОПИС ВЕБ-ДОДАТКУ

## 3.1 Тестування веб-додатку

### 3.1.1 Верифікація та валідація веб-додатку

При створенні нового користувача нерідко можуть виникати помилки. Але механізм валідації дозволяє інформувати користувача про помилки.

Розглянемо RazorPage-сторінку з формою додання користувачів до системи – лістинг 3.1, та бекенд реєстрації – лістинг 3.2.

Лістинг 3.1 – Сторінка Register.cshtml

|  |
| --- |
| <form method="post">  <div **asp-validation-summary**="All" class="alert-danger"></div>  <div class="form-group">  <label **asp-for**="Email"></label>  <input **asp-for**="Email" class="form-control" />  <span **asp-validation-for**="Email" class="text-danger"></span>  </div>  <div class="form-group">  <label **asp-for**="Password"></label>  <input **asp-for**="Password" class="form-control" />  <span **asp-validation-for**="Password" class="text-danger"></span>  </div>  <div class="form-group">  <label **asp-for**="ConfirmPassword"></label>  <input **asp-for**="ConfirmPassword" class="form-control" />  <span **asp-validation-for**="ConfirmPassword" class="text-danger"></span>  </div>  <button type="submit" class="btn btn-primary">Зарегистрироваться</button>  </form>  <script>  <script src="~/lib/jquery/jquery.js"></script>  <script src="~/lib/jquery-validate/jquery.validate.js"></script>  <script src="~/lib/jquery-validation-unobtrusive/jquery.validate.unobtrusive.js"></script>  </script> |

Лістинг 3.2 – Метод Register

|  |
| --- |
| [HttpPost]  public async Task<IActionResult> Register(RegisterViewModel model)  {  if (ModelState.IsValid)  {  var user = new MyUser {  UserName = model.Email,  Email = model.Email,  };  var result = await userManager.CreateAsync(user, model.Password);  if (result.Succeeded)  {  var token = await userManager.GenerateEmailConfirmationTokenAsync(user);  var confirmationLink = Url.Action("ConfirmEmail", "Account",  new  {  userId = user.Id,  token = token  }, Request.Scheme);  if (signInManager.IsSignedIn(User) && User.IsInRole("Admin"))  {  return RedirectToAction("ListUsers", "Administration");  }  ViewBag.ErrorTitle = "Регистрация прошла успешно";  ViewBag.ErrorMessage = "Подтвердите ваш Email. Ссылка для подтверждения была отправлена на указанный Email";  return View("Error");  }  foreach(var error in result.Errors)  {  ModelState.AddModelError("", error.Description);  }  }  return View(model);  } |

JQuery скрипти, що завантужуються на стороні клієнта допомагають швидко вказати на неточності даних без необхідності відсилати форму на сервер – рисунок 3.1.

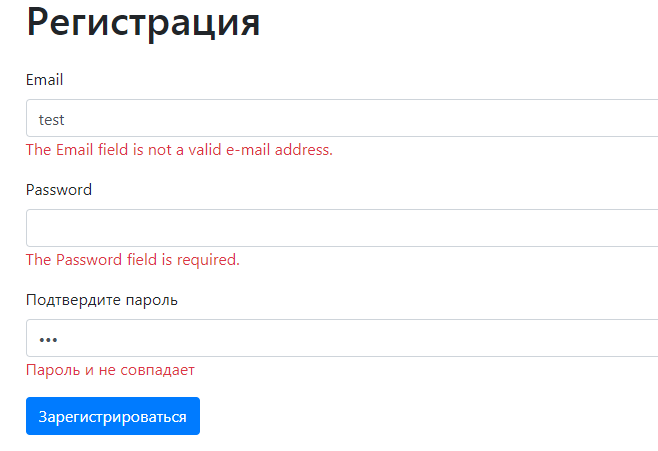


Рисунок 3.1 – Валідація на стороні клієнта

Якщо користувач відсилає форму, то метод IsValid перевіряє її та у разі неправильності вводу відсилає лист помилок – рисунок 3.2.

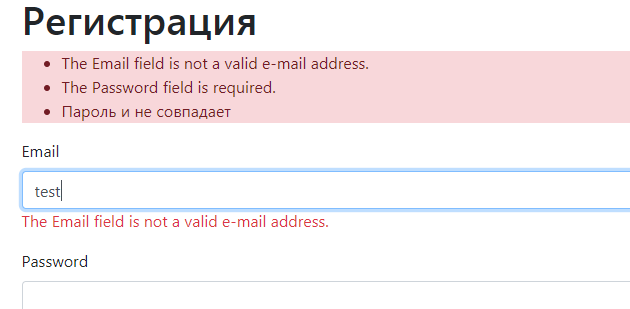


Рисунок 3.2 – Форма, що пройшла валідацію на сервері

У разі наявності користувача з вказаним ім’ям в базі, asp.net core подбає про це – рисунок 3.3.



Рисунок 3.3 – Користувача [test@gmail.com](mailto:test@gmail.com) вже зареєстровано

### 3.1.2 Ручне тестування

Тестові випадки наведені в таблицях 3.1 – 3.3.

Таблиця 3.1 – Тестовий випадок №1

|  |  |
| --- | --- |
| **Назва тесту** | **Тест на редагування користувача** |
| Описання | Тестування редагування користувача в системі. |
| Тестові кроки | Введення інших даних до старого запису . |
| Очікуваний результат | Вдале редагування імені. |
| Фактичний результат | Вдале редагування імені. |
| Статус | Пройдений. |

Таблиця 3.2– Тестовий випадок №2

|  |  |
| --- | --- |
| **Назва тесту** | **Тест на редагування користувача без змін.** |
| Описання | Тестування редагування користувача в системі без дійсних змін. |
| Тестові кроки | Спроба редагування без змін |
| Очікуваний результат | Запис залишився без змін |
| Фактичний результат | Запис залишився без змін |
| Статус | Пройдений. |

Таблиця 3.3 – Тестовий випадок №3

|  |  |
| --- | --- |
| **Назва тесту** | **Тест на видалення користувача з системи** |
| Описання | Тест на видалення користувача з системи |
| Тестові кроки | Видалення обраного запису. |
| Очікуваний результат | Вдале видалення з системи. |
| Фактичний результат | Вдале видалення з системи. |
| Статус | Пройдений. |

Таблиця 3.4 – Тестовий випадок №4

|  |  |
| --- | --- |
| **Назва тесту** | **Тест на додання користувача** |
| Описання | Тестування додання користувача |
| Тестові кроки | Введення нових записів в щоденник. |
| Очікуваний результат | Вдале додання записів до системи. |
| Фактичний результат | Вдале додання записів до системи. |
| Статус | Пройдений. |

Таблиця 3.5– Тестовий випадок №5

|  |  |
| --- | --- |
| **Назва тесту** | **Тест на додавання кінотеатру до списку відстежування** |
| Описання | Тестування додавання кінотеатру до списку відстежування |
| Тестові кроки | Введення нового медіафайлу до системи. |
| Очікуваний результат | Вдале додавання. |
| Фактичний результат | Вдале додавання. |
| Статус | Пройдений. |

Перевіряючи працездатність всіх компонентів програми, ніяких дефектів в роботі програми не було виявлено.

## 3.2 Керівництво користувача

### 3.2.1 Загальний опис можливостей адміністратора

Адміністратор додатку має доступ до перегляду, редагування та додавання інших користувачів та тільки до перегляду ролей. Інтерфейс меню адміністратора зображено на рисунках 3.4-3.5.

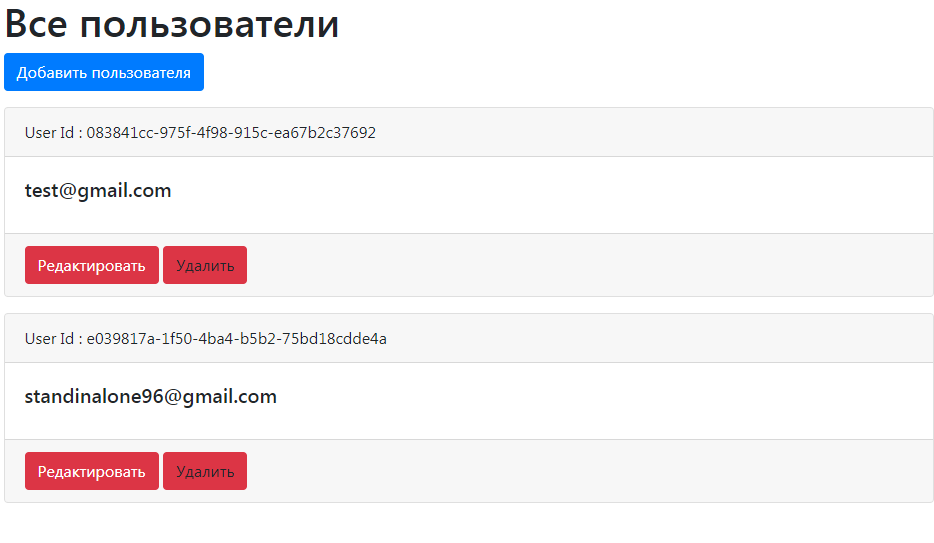


Рисунок 3.4 – Меню адміністратора – список користувачів



Рисунок 3.5 – Меню адміністратора – список ролей

## 3.3 Використання веб-застосунку

Вхід до сайту виконується за допомогою локального акаунта або акаунта Google – рисунок 3.6.

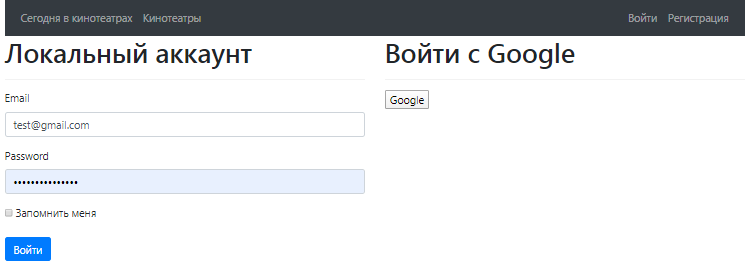


Рисунок 3.6 – Вхід на сайт

Після входу користувач потрапляє на домашню сторінку зі списком фільмів, що на цей час у прокаті – рисунок 3.7.

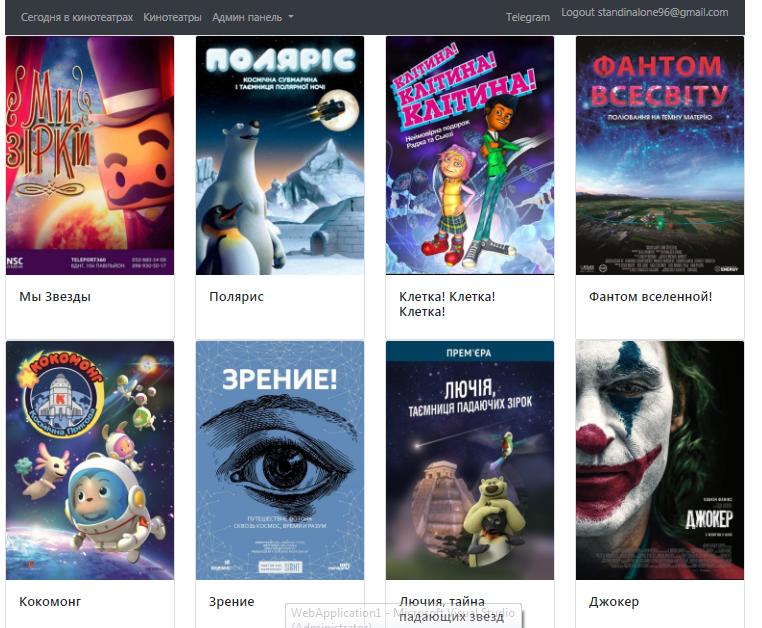


Рисунок 3.8 – Домашня сторінка

При виборі фільму починається парсинг сеансів для цього фільму для усіх кінотеатрів. Далі інформація подається у читабельному виді зі списком кінотеатрів та їх сеансами відповідно до кожного кінотеатру – рисунок 3.9.



Рисунок 3.9 – Список поточних сеансів для обраного фільму

Коли користувач обирає сеанс він отримує інформацію щодо зайнятих та вільних місць на даний час. Відображувана схема відповідає реальній схемі зали, у якій проходить сеанс. Слід вважати, що екран розміщується зверху, тобто дальні місця відображені нижче. Куплені місця позначаються зіркою (\*). Приклад схеми залу наведено на рисунку 3.10.

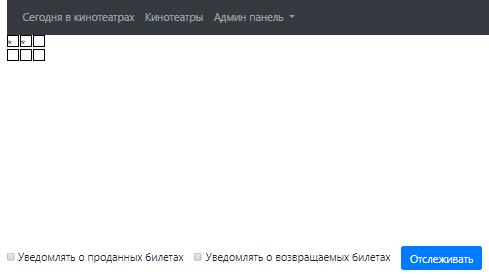


Рисунок 3.10 – Форма додавання нового відстежування

На цій сторінці користувач може почати відстежувати квитки, але спочатку йому треба створити чат з ботом @TicketsHere\_bot у телеграмі, де він повинен буде підтвердити свій акаунт. Надалі оповіщення про всі зміни будуть надходити на його Telegram. Приклад створення чату та підтвердження аканту Telegram наведено на рисунках 3.11 – 3.14.

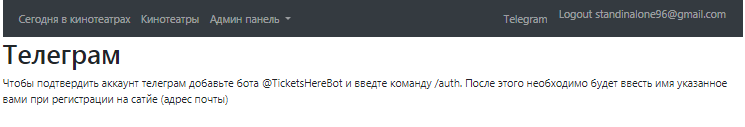


Рисунок 3.11 – Сторінка з інструкцією щодо використання бота

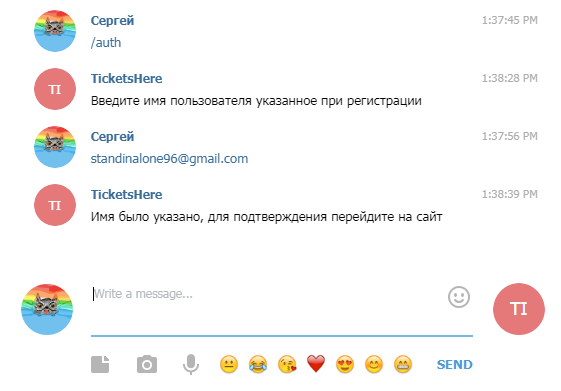


Рисунок 3.12 – Зв’язок з ботом

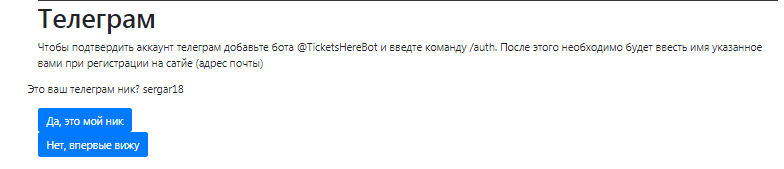


Рисунок 3.13 – З’явилися опції підтвердження та відхилення



Рисунок 3.14 – Після підтвердження

Коли користувач зв’язав свій телеграм акаунт з сайтом він може почати відстежувати вільні місця для сеансів (рисунок 3.10). У разі змін у залі (квитки були куплені або повернуті) користувач отримає повідомлення – рисунок 3.15.

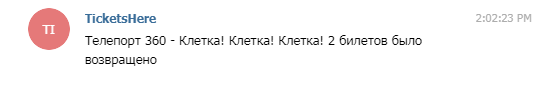


Рисунок 3.15 – Повідомлення про зміни у сеансі

Також є можливість відстежувати появу нових фільмів в улюблених кінотеатрах. Для цього необхідно перейти на сторінку з кінотеатрами та підписатися на будь-який – рисунок 3.16.

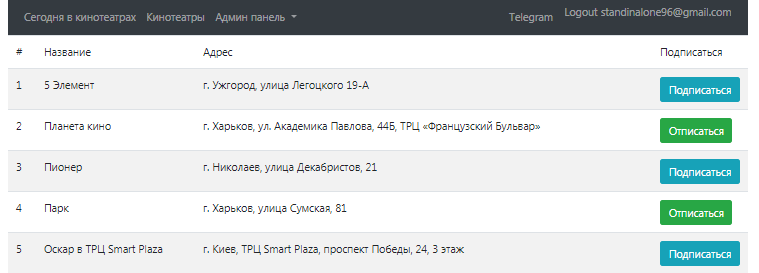


Рисунок 3.16 – Список кінотеатрів

У разі появи в цих кінотеатрах нових фільмів користувач отримає повідомлення – рисунок 3.17.

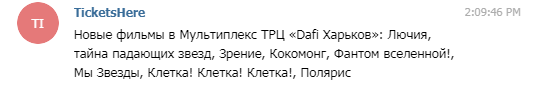


Рисунок 3.17 – Повідомлення про нові фільми

# ВИСНОВКИ

В ході виконання курсової роботи було виконано наступні етапи:

1. Було проаналізовано предметну область індивідуального завдання курсової роботи.
2. Проведено первинну розробку серверної архітектури додатку.
3. Проведено проектування архітектури інформаційної системи з використанням технології «клієнт-сервер».
4. Проведено програмну реалізацію та налагодження розробленої інформаційної системи.

Веб-додаток щоденник призначений для внесення інформації користувачем. Областю застосування даного додатку виступає допомога для зручного зберігання, перегляду або редагування даних.

Отже, в ході виконання курсової роботи, послідовно застосовуючи принципи об'єктно-орієнтованого аналізу, документо-орієнтованого моделювання даних, і середу розробки MS Visual Studio для програмування веб-додатку, було розроблено і реалізовано онлайн сервіс оповіщення нових сеансів та вільних квитків у кіно.

# СПИСОК ДЖЕРЕЛ ІНФОРМАЦІЇ

1. Q-Parser [Електронний ресурс] – Режим доступу: URL – <https://q-parser.ru/>, перегляд: 29.04.2020
2. TargetHunter [Електронний ресурс] – Режим доступу: URL – <https://targethunter.ru/>, перегляд: 29.04.2020
3. CleverTarget [Електронний ресурс] – Режим доступу: URL – <https://clevertarget.ru/>, перегляд: 29.04.2020
4. Диггернаут [Електронний ресурс] – Режим доступу: URL – https://www.diggernaut.ru/, перегляд: 29.04.2020
5. ASP.NET Core Tutorial for Beginners [Електронний ресурс] – Режим доступу:URL– <https://www.youtube.com/playlist?list=PL6n9fhu94yhVkdrusLaQsfERmL_Jh4XmU>, перегляд: 30.04.2020
6. ASP.NET Core Tutorial for Beginners [Електронний ресурс] – Режим доступу:URL – <https://csharp-video-tutorials.blogspot.com/2019/01/aspnet-core-tutorial-for-beginners.html>, перегляд: 30.04.2020
7. ASP.NET Core Razor Pages Tutorial [Електронний ресурс] – Режим доступу:URL – <https://csharp-video-tutorials.blogspot.com/2019/11/aspnet-core-razor-pages-tutorial.html>, перегляд: 30.04.2020
8. The Unified Modeling Language [Електронний ресурс] – Режим доступу:URL – <https://www.uml-diagrams.org/>, перегляд: 30.04.2020