ЗМІСТ

Скорочення та умовні познаки 5
Вступ 8
1 Аналіз та обгрунтування вибору підходів до розробки Web-застосунку 9
1.1 Архітектурний шаблон проектування MVC 9
1.2 Переваги фреймворку ASP.NET MVC 5 13
1.3 Фреймворк для створення веб-інтерфейсів Bootstrap 16
1.4 Динамічне оновлення CSS і JS 19
1.4.1 Дослідження проблеми динамічного оновлення файлів 19
1.4.2 Реалізація динамічного оновлення файлів 20
1.5 Бібліотеки, які були використані в проекті 22
2 Проектування веб-застосунку «VIEWER» 23
2.1 Вибір середовища розробки 23
2.2 Розробка бази даних з використанням EntityFramework 24
2.3 Моделювання динамічних і поведінкових аспектів системи 31
2.3.1 Діаграма прецедентів 31
2.3.2 Діаграма послідовностей 32
2.3.3 Діаграма станів 34
3 Програмна реалізація веб-застосунку «VIEWER» 35
3.1 Створення Controllers 35
3.2 Створення сервісів (Service) 36
3.3 Створення Views 37
3.4 Управління користувачами та генератором 40
3.5 Використання сторонніх API 42
3.6 Опис та тестування веб-застосунку «Viewer» 44
Висновки 47
Перелік джерел посилання 49
Додаток А Структура програми веб-застосунку 51

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

БД – база даних
ПЗ – програмне забезпечення
ПК – персональний комп’ютер
ПП – програмний продукт
СКБД – системи керування бази даних
EF – Entity Framework
HTML – HyperText Markup Language
JS – JavaScript
WCF – Windows Communication Foundation

ASP.NET MVC Framework – фреймворк для створення веб-застосувань, який реалізує шаблон Model-view-controller. Цей фреймворк доданий Microsoft в ASP.NET. У квітні 2009 року, вихідний код ASP.NET MVC був опублікований під ліцензією Microsoft Public License (27 березня 2012 ліцензія була змінена на Apache License 2.0).

AJAX (Asynchronous JavaScript And XML) – підхід до побудови інтерфейсів веб-застосувань користувача, за яких веб-сторінка, не перезавантажуючись, у фоновому режимі надсилає запити на сервер і сама звідти довантажує потрібні користувачу дані. AJAX – один з компонентів концепції DHTML.

API (програмний інтерфейс програми, інтерфейс програмування додатків, інтерфейс прикладного програмування) – набір готових класів, процедур, функцій, структур і констант, які надаються додатком (бібліотекою, сервісом) або операційною системою для зовнішнього використання в програмних продуктах. Використовується програмістами при написанні всіляких додатків.

Bootstrap – це безкоштовний набір інструментів з відкритим кодом, призначений для створення веб-сайтів та веб-застосунків, який містить шаб-

лони CSS та HTML для типографіки, форм, кнопок, навігації та інших компонентів інтерфейсу, а також додаткові розширення JavaScript.

CSS (Cascading Style Sheets – каскадні таблиці стилів) – одна з базових технологій у сучасному Інтернеті. CSS-код – це список інструкцій для браузера – як і де відображати елементи веб-сторінки, написаний особливим чином.

Entity Framework – об'єктно-орієнтована технологія доступу до даних, є об'єктно-реляційним відображенням (ORM) рішення для .NET Framework від Microsoft. Надає можливість взаємодії з об'єктами як за допомогою LINQ у вигляді LINQ для осіб, так і з використанням Entity SQL. Для полегшення побудови веб-рішень використовується як ADO.NET Data Services (Astoria), так і зв'язка з Windows Communication Foundation і Windows Presentation Foundation, що дозволяє будувати багаторівневі додатки, реалізуючи один з шаблонів проектування MVC, MVP або MVVM.

Filter – перевіряє, чи має користувач права доступу, у випадку, якщо ні, то відбувається редірект на сторінку помилки, тощо.

HTML – мова розмітки гіпертексту, не є мовою програмування. Це форма збереження даних.

JavaScript – мова програмування для створення інтерактивних Web-сторінок.

Контролер (Controller) – інтерпретує дії користувача, сповіщаючи модель про потребу у зміні.

Модель – надає дані і реагує на команди контролера, змінюючи свій стан.

Пагінація (від (латинський) pagina — сторінка), порядкова нумерація сторінок твору друку. Для позначення номерів сторінок застосовують колонцифри, що розташовуються вгорі або внизу сторінки. П. називається також загальне число сторінок в творі друку, включаючи окремі листи, карти, додатки і т.д.

Представлення (View) – відповідає за відображення даних моделі користувачеві, реагуючи на зміни моделі.

Thread – клас, який створює і контролює потік, задає пріоритет і повертає статус. Дозволяє паралельно виконувати декілька операцій без підвисань системи.

WCF – програмний фреймворк, який використовується для обміну даними між додатками, що входить до складу .NET Framework.

Windows Forms – інтерфейс програмування додатків (API), відповідальний за графічний інтерфейс користувача і є частиною Microsoft.NET Framework.

XAML – це декоративна мова розмітки. З погляду моделі програмування .NET Framework мова XAML спрощує створення інтерфейсу користувача для програми .NET Framework.

ВСТУП

Сьогодні все частіше люди звертають увагу на торенти та рейтинги програмних медіа продуктів. Адже використання подібних рішень економічно вигідно і виправдано, бо гарантує якісний вибір медіа або програмних продуктів.

Тому актуальним завдання є розробка безкоштовного програмного засобу, що являє собою торент, призначений для виконання файлообмінних протоколів, зі зручним графічним інтерфейсом і широкими функціональними можливостями.

Застосунок «Viewer» – це не лише система перегляду відеофайлів з youtube онлайн, а й повноцінний торент, який дозволяє завантажувати файли за прямим посиланням, а після завантаження чи перегляду дозволяє залишити відгук про якість продукту.

Мета кваліфікаційної роботи – розробити веб-застосунок на основі технологій ASP MVC 5 та Windows Forms з можливостями перегляду онлайн відео, завантаження файлів за прямим посиланням та можливістю написання власних відгуків.

Для досягнення поставлених цілей потрібно розв’язати наступні завдання:

- виконати аналіз предметної області для уточнення і формулювання вимог до програмного продукту;
- обґрунтувати вибір програмних засобів та мережевих протоколів, які нададуть можливість автоматизувати процеси аналізу предметної області для зв’язування функціональних вимог та проектування програмного виробу;
- розробити систему рейтингів;
- виконати проектування і розробку інтерфейсу користувача з використанням об’єктно-орієнтованих технологій;

– реалізувати веб-застосунок засобами Visual Studio.NET, SQL, Entity Fraemwork, WCF та графічного інтерфесу Windows Forms у вигляді програмного засобу;
– провести тестування розробленого програмного продукту.

Реалізація веб-застосунку «Viewer» забезпечить можливість перегляду рейтингів програм, відео-файлів з подальшим їх завантаженням.

1 АНАЛІЗ ТА ОБГРУНТУВАННЯ ВИБОРУ ПІДХОДІВ ДО РОЗРОБКИ WEB-ЗАСТОСУНКУ

1.1 Архітектурний шаблон проектування MVC

Архітектурні шаблони проектування, які надають рішення проблеми проектування, що часто виникає, добре зарекомендували себе в якості підходу до створення веб-застосунків. Шаблони треба розуміти, як приклад розв’язання задач, який можна використовувати в різних ситуаціях.

Серед переваг використання шаблонів треба перш за все відмітити, що шаблон дає готовий набір абстракцій для вирішення проблеми та іменує проблему. Також шаблони пропонують різноманіття варіантів рішення, а за рахунок того, що шаблони надають назви для популярних проблем, вони сприяють комунікації з іншими розробниками та прискорюють розуміння чужого коду.

Серед мінусів використання шаблонів слід відмітити, що сліпе слідування деяким обраним шаблоном може привести до ускладнення програми. У розробника може виникнути бажання спробувати деякий шаблон без особливих підстав.

Прикладами архітектурних шаблонів, є:

– Model-View-Controller (MVC);
– Model-View-Presenter (MVP);
– Model-View-View Model (MVVM);

– Presentation-Abstraction-Control (PAC);

– View-Interactor-Presenter-Entity-Routing (VIPER).

Розглянемо більш докладно шаблон проектування MVC, що був використаний в роботі.

MVC (Model-View-Controller: модель-вид-контролер) – це не шаблон проекту, це конструкційний шаблон, який описує спосіб побудови структури застосунку, сфери відповідальності та взаємодія кожної з частин в даній структурі [1][1)].

MVC далеко не новий (його поява датується 1978 роком і пов'язана з проектом Smalltalk в Xerox PARC), але в наші дні він завоював величезну популярність як шаблон для веб-застосунків з перелічених нижче причин.

Ідея, яка лежить в основі конструкційного шаблону MVC, дуже проста: потрібно чітко розділяти відповідальність за різне функціонування в додатках:

– Model – обробка даних і логіка застосунку;

– View – представлення даних користувачеві в будь-якому підтримуваному форматі;

– Controller – обробка запитів користувача і виклик відповідних ресурсів.

Застосунок розділяється на три основних компоненти, кожен з яких відповідає за різні завдання.

Контролер (Controller) управляє запитами користувача (одержувані у вигляді запитів HTTP GET або POST, коли користувач натискає на елементи інтерфейсу для виконання різних дій). Його основна функція – викликати і координувати дію необхідних ресурсів і об'єктів, потрібних для виконання дій, що задаються користувачем. Зазвичай контролер викликає відповідну модель для задачі і вибирає відповідний вид.

[1)] [1] Модель-вид-контролер – Вікіпедія. (загол. з екрана). URL: https://uk.wikipedia.org/wiki/Модель-вид-контролер (дата звернення 13.05.2019)

Модель (Model) – це дані і правила, які використовуються для роботи з даними, які представляють концепцію управління застосунком. У будь-якому додатку вся структура моделюється як дані, які обробляються певним чином.

Модель дає контролеру представлення даних, які запросив користувач (повідомлення, сторінку книги, фотоальбом, тощо). Модель даних буде однаковою, незалежно від того, як ми хочемо представляти їх користувачеві. Тому ми вибираємо будь-який доступний вид для відображення даних.

Модель містить найбільш важливу частину логіки програми, яка вирішує задачу, з якою має справу користувач (форум, магазин, банк, тощо). Контролер містить організаційну логіку для самого застосунка.

Вид (View) забезпечує різні способи представлення даних, які отримані з моделі. Він може бути шаблоном, який заповнюється даними. Може бути кілька різних видів, і контролер вибирає, який підходить якнайкраще для поточної ситуації.

Веб-застосування зазвичай складається з набору контролерів, моделей і видів. Контролер може бути влаштований як основний, який отримує всі запити і викликає інші контролери для виконання дій в залежності від ситуації.

Структурна схема шаблону MVC представлена на рис. 1.1.

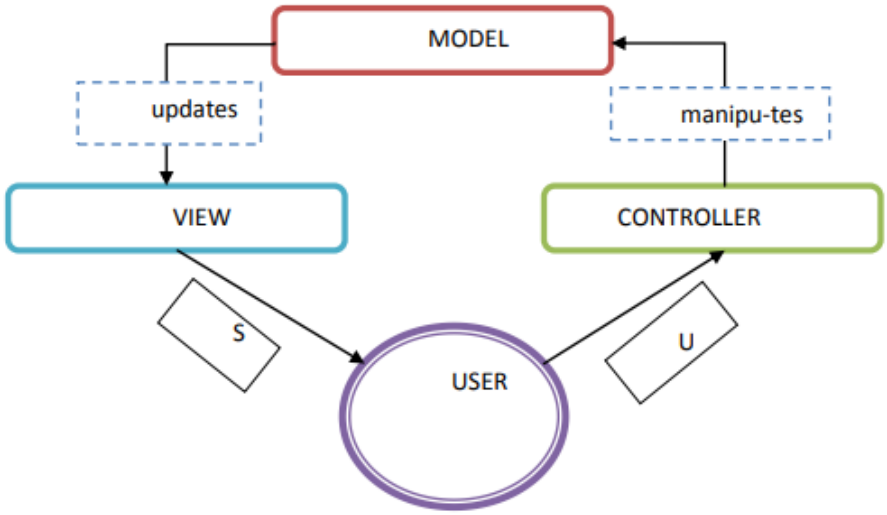

Рисунок 1.1 – Структурна схема архітектурного шаблону MVC

Перевага від використання концепції MVC – це чіткий поділ логіки представлення (інтерфейсу користувача) і логіки програми.

Підтримка різних типів користувачів, які використовують різні типи пристроїв, є спільною проблемою. Наданий інтерфейс повинен відрізнятися, якщо запит приходить з персонального комп'ютера або з мобільного телефону. Модель повертає однакові дані, єдина відмінність полягає в тому, що контролер вибирає різні види для виведення даних [2][1)].

Крім ізолювання видів від логіки програми, концепція MVC істотно зменшує складність великих застосунків. Код виходить набагато більш структурованим, і, тим самим, полегшується підтримка, тестування і повторне використання рішень.

Шаблон MVC був реалізований в декількох мовах програмування. Мовою C# була створена платформа ASP.NET MVC, яка представляє собою альтернативу схемі веб-форм ASP.NET при створенні веб-застосунків. В ASP.NET MVC впроваджений сучасний варіант MVC, який особливо добре підходить для веб-застосунків [3]–[4][2)]. Платформа ASP.NET MVC є легковагою платформою відображення з широкими можливостями тестування і, подібно до застосунків на основі веб-форм, інтегрована з існуючими функціями ASP.NET.

За рахунок прийняття та адаптації шаблону MVC ASP.NET MVC інфраструктура Framework складає сильну конкуренцію Ruby on rails та аналогічним платформам, виводячи модель MVC в авангард розвитку світу .NET. Узагальнюючи досвід і найбільш рекомендовані прийоми, виявлені розробниками, які використовують інші платформи, ASP.NET MVC у багатьох відношеннях перевершили навіть те, що може запропонувати Rails [5][3)].

Платформа ASP.NET MVC базується на взаємодії трьох компонентів: контролера, моделі та представлення. Контролер приймає запити, обробляє

[1)] [2] Фримен Эрик, Фримен Элизабет. Паттерны проектирования. СПб.: Питер, 2011. 656 с.

[2)] [3] Гонсалвес Энтони, Изучаем Java EE 7. СПб.: Питер, 2014. 640 с.

[3)] [5] Freeman A. «Pro ASP.NET MVC 5 Platform». Apress, 2014. 428 p

введення користувача, взаємодіє з моделлю і представленням і повертає користувачеві результат обробки запиту. Модель представляє шар, що описує логіку організації даних в застосунку. Представлення отримує дані з контролера і генерує елементи призначеного для користувача інтерфейсу для відображення інформації.

Для управління розміткою і вставками коду в представлені використовується движок представлень. До версії MVC 5 використовувалися два движка: Web Forms і Razor. Починаючи з MVC 5 єдиним двигуном, вбудованим за замовчуванням, є Razor [6][1)]. Движок WebForms використовує файли .aspx, а Razor – файли .cshtml і .vbhtml для зберігання коду представлень.

Основою синтаксису Razor є знак @, після якого здійснюється перехід до коду на мовах C # / VB.NET. Також можливо і використання сторонніх движків. Файли представлень не є стандартними статичними сторінками з кодом html, а в процесі генерації контролером відповіді з використанням представлень компілюються в класи, з яких потім генерується сторінка html.

При обробці запитів фреймворк ASP.NET MVC спирається на систему маршрутизації, яка зіставляє всі вхідні запити з певними в системі маршрутами, які вказують який контролер і метод повинен обробити такий запит. Вбудований маршрут за умовчанням передбачає триланкову структуру: контролер / дію / параметр.

1.2 Переваги фреймворку ASP.NET MVC 5

Інфраструктура MVC Framework побудована у вигляді набору незалежних компонентів, які задовольняють інтерфейс .NET або створені на основі абстрактного базового класу. Компоненти, подібна система маршрутизації, механізм, візуалізація, фабрика, контролери, можна легко замінювати на інші

[1)] [6] Троелсен Э. C# и платформа .NET. Библиотека программиста. СПб.: Питер, 2002. 800 с.

компоненти з власною реалізацією. У загальному випадку для кожного компонента MVC Framework пропонує три можливості [7]–[8][1)]:

– використання стандартної реалізації компонента в тому вигляді, як вона є (цього повинно бути достатньо для більшості додатків);

– створення підкласу зі стандартної реалізації з метою коригування існуючої поведінки;

– можливість повної заміни на нову реалізацію.

Різні компоненти, а також способи та причини їх можливого налаштовування або заміни буде розглянуто в наступних підрозділах.

ASP.NET MVC інфраструктура генерує зрозумілі та відповідні стандарти коду розмітки. Її вбудовані допоміжні методи HTML дозволяють створювати відповідні стандартам представлення. Замість генерації величезного обсягу HTML-розмітки, яка важко піддається управлінню, інфраструктура MVC Framework стимулює створення простих та елегантних елементів, оформлених за допомогою стилів CSS.

Якщо потрібно використовувати деякі готові віджети для таких складних елементів, інферйсу користувача (UI), як вікна вибору дати або каскадне меню, яке застосовується в ASP.NET MVC, то вони дозволяють легко використовувати найкращі бібліотеки для побудови інтерфейсів користувача, подібні JQuery або Bootstrap CSS. Наприклад, бібліотека JQuery настільки ефективно підтримується, що постачається як вбудована частина стандартного шаблону проекту ASP.NET MVC у Visual Studio поряд з іншими популярними бібліотеками, такими, як Bootstrap і Modernizr.

В ASP.NET MVC згенеровані сторінки можуть не містити даних, тому у порівнянні з веб-формами вони мають менше однотипних сторінок, що в свою чергу економить розмір. Незважаючи на сучасні швидкі з'єднання, така економія трафіку підвищує комфорт кінцевого користувача і допомагає скоротити витрати, пов'язані з запуском популярних веб-додатків.

[1)] [7] Troelsen Andrew Pro C# 5.0 and the .NET 4.5. Framework Apress, 2012. 1463 p.
[8] Benedetti R., Cranley R. O'Reilly Media, Apress Inc, 2011. 500 p.

ASP.NET MVC інфраструктура діє в тісній співпраці з HTTP. При цьому існує контроль над запитами, які передаються між браузером і сервером, що дозволяє дуже точно налаштовувати інтерфейс користувача на свій розсуд. Технологія AJAX проста, і їй не потрібні якісь автоматичні зворотні відправки запитів для взаємодії зі станом клієнтської сторони.

Існуюча платформа ASP.NET виробництва Microsoft пропонує зрілий, добре перевірений набір компонентів і засобів для розробки ефективних і високопродуктивних веб-додатків.

Перша і найбільш очевидна перевага полягає в тому, що, оскільки інфраструктура ASP.NET MVC побудована на основі платформи .NET, можна писати код на будь-якій мові .NET і при цьому мати доступ до одних і тих же функцій API-інтерфейсів, які визначені не тільки в MVC Framework, але і у великій бібліотеці класів .NET, а також у багатьох бібліотеках .NET від незалежних розробників [9][1)].

По-друге, готові засоби платформи ASP.NET, такі як аутентифікація, членство, ролі, профілі та інтернаціоналізація можуть істотно скоротити обсяг коду, який доведеться писати і підтримувати в будь-якому веб-додатку, і в проекті MVC Framework. Вони настільки ж ефективні, як і у класичному проекті Web Forms. Платформа, яка лежить в основі ASP.NET надає розвинений набір інструментів, на базі яких будуються веб-додатки за допомогою MVC Framework [10][2)].

Платформа Microsoft .NET розвивалася з кожним великим випуском, підтримуючи – і навіть визначаючи – багато передових аспектів сучасного програмування.

Версія ASP.NET MVC 5 побудована для .NET Framework 4.5.1, тому її API-інтерфейс може у повній мірі задіяти останні нововведення мови і виконавчого середовища, в тому числі ключове слово await, що розширюють ме-

---

[1)] [9] Jon Galloway, Brad Wilson K., Scott Allen, David Matson. Professional ASP.NET MVC John Wiley & Sons, 2014. 624 p

[2)] [10] Munro Jamie. ASP.NET MVC 5 with Bootstrap and Knockout.js. O'Reilly Media, 2015. 278 p.

тоди, лямбда-вирази, анонімні і динамічні типи , а також мову інтегрованих запитів (Language Integrated Query - LINQ). Багато методів і шаблонів кодування API-інтерфейсу MVC Framework дотримуються більш чіткої та виразної композиції, ніж це було можливо в ранніх платформах.

1.3 Фреймворк для створення веб-інтерфейсів Bootstrap

Bootstrap – фреймворк, набір HTML + CSS інструментів і шаблонів для верстки та більш ефективного і швидкого створення сайтів і веб-застосунків.

Bootstrap – сучасний помічник, розробників інтерфейсів, дизайнерів і веб-майстрів, доступний для використання за відкритою ліцензією. Фреймворк дуже динамічний і регулярно оновлюваний, тому не всі його функції можуть коректно підтримуватися старими браузерами [11][1)].

У Bootstrap є наступні шаблони:

- шрифти;
- кнопки;
- форми;
- мітки;
- навігація;
- сітка;
- JavaScript-розширення.

Зараз найбільш популярною версією фреймворка Bootstrap є третя. У ній подальший розвиток отримала адаптивність і принцип mobilefirst.

Сучасний світ дуже динамічний, і веб-програмісти часто стикаються з вибором – робити швидкі прототипи і запускати їх або повільно і довго переробляти верстку, доводячи все до ідеального стану. Bootstrap прекрасно реалізує перший підхід.

Переваги фреймворка Bootstrap:

[1)] [11] Bootstrap – Вікіпедія. (загол. з екрана). URL: https://uk.wikipedia.org/wiki/Bootstrap (дата звернення 13.05.2019)

- Економія часу – досягається за рахунок використання вже готових класів і дизайну. Це дозволяє направити заощаджені енергію і гроші на розробку додаткової функціональності;
- Адаптивність (висока швидкість і оптимізація, стандартизація інтерфейсів) – динамічні макети якісно відображаються на самих різних пристроях без необхідності внесення змін до розмітки;
- Дизайн – єдині шаблони і стильове оформлення елементів макета і всіх сторінок на сайті в цілому. І при цьому Bootstrap крос-браузерні і добре відображається у всіх браузерах Safari, Firefox, IE, EDGE і тих, що на основі Chromium (движок Blink на основі Webkit: яндекс.браузер, Opera, Google Chrome);
- Простота і відкритість – використовувати Bootstrap настільки просто, що з ним справляються навіть початківці веб-розробники, а відкритий вихідний код дозволяє брати участь в розробці, модифікувати під потреби або просто користуватися хорошим безкоштовним рішенням.

При цьому код HTML, JavaScript і CSS в Bootstrap продуманий сотнями розробників з усього світу – все для того, щоб пересічні веб-майстри і верстальники могли легко і просто налаштувати сітку сайту або вбудувати необхідні елементи в інтерфейс.

Також, в Bootstrap використовується динамічна мову стилів LESS, які розширює можливості CSS: розробники можуть керувати кольорами, створювати вкладені колонки і змінні [12][1)].

До недоліків Bootstrap можна віднести:

- Одноманітність – Bootstrap часто критикують за те що сайти виглядають однаково. Всі сайти, які використовують Bootstrap, схожі один на одного, тобто відсутня унікальність. Подібні сайти-близнюки просто не запам'ятовуються;

[1)] [12] Чебыкин Р.И., Самоучитель HTML и CSS. Современные технологии. СПБ.: БХВ-Петербург, 2008. 624 с.

- Негнучкість – якщо потрібно створити що-небудь відмінне від інтерфейсу Bootstrap, то доводиться з найперших кроків боротися зі стилями за замовчуванням. На практиці виходить подвійна робота;
- Надмірний код – те, що реально зробити двома вкладеними блоками, часто робиться п'ятьма. Будь-які зміни тягнуть за собою годинник мук;
- Обмеження у використанні в старих браузерах – регулярне оновлення і доповнення фреймворка найсучаснішими можливостями HTML і CSS вносить деякі обмеження у використанні з IE7 і IE8.

Розвиток інтернету і вебсайтів показало, що в 2018 році половина користувачів інтернету відвідує сайти з мобільних пристроїв і невеликих екранів. Це означає, що сучасний веб-розробник повинен думати про функціонування сайту не тільки на ПК, але і на смартфонах і планшетах з тачскріном.

Створення окремої мобільної версії для сайту звичайно вихід, але тоді потрібно робити в два рази більше роботи на розробку і підтримку коду, а це не завжди економічно ефективно.

Концепція чуйного веб-дизайну, яка втілена у фреймворку Bootstrap, вирішує саме цю проблему: сайт однаково відображає інформацію найбільш повним чином незалежно від типу екрану і розміру пристрою. Зміст і колірна гамма не змінюються, змінюється лише форма і спосіб згрупувати інформаційні та навігаційні блоки сайту найбільш зручним для користувача способом.

Bootstrap це сучасний багатофункціональний фреймворк. Їм можна користуватися регулярно або навпаки лише в разі потреби, але кожен початківець веб-майстер, верстальник або фронтенд-розробник повинен мати в своєму арсеналі мінімальний набір навичок і знань для роботи з Bootstrap.

Після перелічених переваг фреймворку Bootstrap не залишається ніяких сумнівів у необхідності його використання у проекті.

1.4 Динамічне оновлення CSS і JS

1.4.1 Дослідження проблеми динамічного оновлення файлів

Для розроблення сайтів часто виникає потреба оновлення файлів стилів (.css) або Java Script (.js), однак браузер користувача ще довгий час не відображає змін. Для вирішення цієї проблеми, починаючи з ASP.NET MVC 4, була введена концепція бандлів, яка допомагає організувати файли скриптів і стилів ефективнішим шляхом – зниження витрат пам’яті при передачі клієнту та автоматичний контроль версій файлів. Розглянемо ці бандли.

У попередніх версіях MVC можна підключати скрипти і стилі звичайним способом, наприклад:

```
<head>
    <meta name="viewport" content="width=device-width" />
    <link  type="text/css" rel="stylesheet"
     href="~/Content/Site.css" />
    <script src="~/Scripts/jquery-1.7.1.js"
     type="text/javascript"></script>
    <title>Index</title>
</head>
```

Фреймворк MVC для налаштування шляхів дає можливість використовувати URL-хелпери:

```
<head>
    <meta name="viewport" content="width=device-width" />
    <link type="text/css" rel="stylesheet"
     href="@Url.Content("~/Content/Site.css")" />
    <script src="@Url.Content("~/Scripts/jquery-1.7.1.js")"
     type="text/javascript"></script>
    <title>Index</title>
</head>
```

Оновлення Java Script або CSS файлів, які вже закешувалися в браузерах користувачів, можуть не виявлятися одразу і як результат – багато користувачів не зможуть отримати оновлені файли упродовж деякого проміжку часу. Тому важливо, щоб браузер автоматично завантажував оновлені файли. На жаль, немає ефективного способу миттєвого оновлення даних у всіх браузерах одночасно. Однак, можливо змінити ім'я файлу або ж змінити URL файлів, вводячи деякі унікальні рядки запиту. Більшість веб-розробників ви-

користовують рядок запиту і використовують суфікс-версії, щоб відправити новий файл в браузер. Наприклад:

```
<script type=”text/javascript” src=”js/jquery-1.4.1.min.js?v=1”>
</script>
<script type=”text/javascript” src=”js/myscript.js?v=1” >
</script>
<link rel=”stylesheet” type=”text/css” href=”css/style.css?v=1”/
>
```

Для того, щоб виконати цю команду, потрібно провести оновлення на всіх сторінках версії даних файлів. Якщо ж якась сторінка буде пропущена, то вона може «зламатися», бо браузер буде використовувати старий сценарій кешування. Щоб уникнути цього, потрібно автоматично змінювати CSS і JS версії щоразу, коли відбуваються зміни.

1.4.2 Реалізація динамічного оновлення файлів

Використання бандлів забезпечує інший підхід до використання скриптів і стилів. Тобто при створенні нового проекту MVC 5 за шаблоном Basic або Internet Application функціональність бандлів вже за замовчуванням включається в застосунок, який контролює версії файлів та автоматично оновлює їх. Тобто за замовчуванням проекти MVC 5 (виключаючи проекти Empty) вже містять реєстрацію бандлів – у файлі BundleConfig.cs, який знаходиться в папці App_Start:

```
public static void RegisterBundles(BundleCollection bundles)
        {
         bundles.Add(new ScriptBundle("~/bundles/jquery")
                .Include("~/Scripts/jquery-{version}.js"));
        }
```

Метод RegisterBundles додає вже бандли, які створено в колекції bundles. Оголошення бандла виглядає наступним чином:

```
new ScriptBundle("~/bundles/jquery").
             Include("~/Scripts/jquery-{version}.js").
```

У конструктор Script Bundle передається віртуальний шлях бандла. А за допомогою методу Include в бандл включаються конкретні файли скриптів. У виразі "~ / Scripts / jquery- {version} .js" параметр {version} є замінником для

будь-якого символьного позначення версії скрипта. Це дуже зручно, бо через деякий час можна замінити версію бібліотеки, але при цьому в коді нічого не доведеться міняти, бо система вже автоматично прийме нову версію.

Вираз "~ / Scripts / jquery.validate *" за допомогою знака зірочки замінює іншу частину рядка. Наприклад, цей вираз підключить в бандл відразу два файли: jquery.validate.js і jquery.validate.unobtrusive.js (і їх мінімізовані версії), бо їх назви починаються з jquery.validate * ". Проте використання бандлів не завжди є оптимальним, бо деякі файли js і css потрібні лише для окремих сторінок. У такому випадку потрібно самостійно контролювати версії файлів. Для цього створено клас VersionedHelper, який забезпечує контроль версій файлів та періодично оновлює їх. Наприклад,

```
public static class VersionedHelper
{
    public static string VersionedContent(this UrlHelper helper,
                                           string contentPath)
    {
        var context = helper.RequestContext.HttpContext;

        if (context.Cache[contentPath] == null)
        {
            var physicalPath =
                      context.Server.MapPath(contentPath);
            var version = @"v=" + new
FileInfo(physicalPath).LastWriteTime.ToString(@"yyyyMMddHHmmss")
;
     var translatedContentPath = helper.Content(contentPath);
     var versionedContentPath = contentPath.Contains(@"?")
               ? translatedContentPath + @"&" + version
             : translatedContentPath + @"?" + version;
            context.Cache.Add(physicalPath, version, null,
            DateTime.Now.AddMinutes(1), TimeSpan.Zero,
            CacheItemPriority.Normal, null);
            context.Cache[contentPath] = versionedContentPath;
            return versionedContentPath;
        }
        else
        {
            return context.Cache[contentPath] as string;
        }
    }
}
```

Після підключення даного класу замість того, щоб писати щось на зразок:

```
<link href="@Url.Content(@"~/Content/bootstrap.min.css")"
 rel="stylesheet" type="text/css" />
<script
src="@Url.Content(@"~/Scripts/bootstrap.min.js")"></script>
```

Потрібно записати:

```
<link
href="@Url.VersionedContent(@"~/Content/bootstrap.min.css")"
rel="stylesheet" type="text/css" />
<script
src="@Url.VersionedContent(@"~/Scripts/bootstrap.min.js")">
</script>
```

І як результат, отримаємо файли вже з версіями, які будуть самостійно оновлюватись.

```
<link href="/Content/bootstrap.min.css?v=20151104105858"
rel="stylesheet" type="text/css" />
<script
src="/Scripts/bootstrap.min.js?v=20151029213517">
</script>
```

Отже, результатом проведеного дослідження став VersionedHelper, який забезпечує динамічне оновлення версій css та js.

1.5 Бібліотеки, які були використані в проекті

System.Linq – простір імен, який містить класи та інтерфейси, що підтримують LINQ.

System.Threading.Tasks – простір імен, який надає типи, які спрощують задачу написання паралельного та асинхронного коду. Основні типи System.Threading.Tasks.Task представляють собою асинхронну операцію очікування і скасування. System.Threading.Tasks.Task <TResult> може повертати значення. System.Threading.Tasks.TaskFactory клас, який надає статичні методи для створення і запуску завдань. Клас, що надає інфраструктуру з планування потоку за замовчуванням – System.Threading.Tasks.TaskScheduler.

System.Windows.Controls – простір імен .NET Framework, котрий надає класи для створення елементів, відомих як елементи управління, що дозволяють користувачеві взаємодіяти з додатком. Класи елементів управління є основою для взаємодії користувача з додатком, бо вони дозволяють користувачеві переглядати, вибирати або вводити дані або іншу інформацію.

System.Web – простір імен надає класи та інтерфейси, які забезпечують взаємодію між оглядачем і сервером. Цей простір імен включає HttpRequest клас, який надає докладні відомості про поточний запит HTTP; HttpResponse клас, який управляє вихідними даними HTTP-клієнт; HttpServerUtility клас, що надає доступ до серверних службових програм і процесів. System.Web також містить класи для роботи з файлами кукі, передачі файлів, відомості про винятки та управління кешем виведення [13][1)].

Простір імен System.Web.Mvc містить класи та інтерфейси, які підтримують платформу ASP.NET MVC для створення веб-додатків. У цей простір імен входять класи, які представляють контролери, фабрики контролерів, результати дій, уявлення, часткові уявлення, зв'язуючі моделі і багато іншого.

Простір імен містить типи System.Web.Routing розширення для маршрутизації.

2 ПРОЕКТУВАННЯ ВЕБ-ЗАСТОСУНКУ «VIEWER»

2.1 Вибір середовища розробки

У цьому розділі розглянемо найбільш важливі етапи реалізації веб-застосунку за допомогою обраних технологій.

Хочеться відзначити, що веб-застосунку можна розділити на три основні частини:

– створення бази даних, і робота з нею;

– створення серверної частини, в якій реалізується вся бізнес-логіка і взаємодія з базою даних;

– створення клієнтської частини, а саме розробка користувальницького інтерфейсу.

В якості середовища розробки була обрана Visual Studio 2013 Ultimate. Вибір даної IDE обумовлюється тим, що всі великомасштабні веб-сайти ASP.NET написані на ній. До складу цього професійного засобу для розробки

[1)] [13] Faithe Wempen. HTML5 Step by Step. Microsoft Press, 2011. 416 p.

входить розвинений набір інструментів для проектування, в тому числі легендарні інструменти для налагодження і механізм IntelliSense, здатний перехоплювати помилки і пропонувати варіанти під час введення. Крім того, в Visual Studio підтримується потужна модель відокремленого коду, яка дозволяє розділяти створюваний код .NET і дескриптори розмітки веб-сторінки. І, нарешті, в Visual Studio має вбудований тестовий вебсервер, який значно спрощує процес налагодження веб-сайтів.

Для роботи з даними будемо використовувати СКБД SQL Server 2012, в склад якої входить компонент Database Engine, служби аналізу Analysis Services, служби звітів Reporting Services, інтеграційні служби Integration Services і розширення SQLXML – є найкращим вибором для широкого діапазону кінцевих користувачів і програмістів баз даних, що працюють над створенням бізнес-додатків, з двох причин:

- SQL Server — оптимальна система для операційних систем Windows, внаслідок її тісної інтеграції з ними (а також внаслідок низької вартості). Завдяки величезному і все зростаючій кількості встановлених систем Windows;
- SQL Server є широко застосовуваною системою керування базами даних.

2.2 Розробка бази даних з використанням EntityFramework

В даний час при моделюванні бази даних використовують одну з різновидів моделювання "об'єкт-відношення" (ER). Такі моделі поміщають в центр уваги деякі сутності і зв'язку між ними – об'єкти і відносини. Більшість інструментів розробки баз даних є інструментами ER-моделювання [14][1)]. Для розробки використовується інструмент моделювання баз даних ERwin версії 3.5, продукт компанії CA / Logic Works. Не дивлячись на те, що генерація скрипта по створенню бази не знадобиться (так як в роботі ми використову-

[1)] [14] Мюллер Роберт Дж. Базы данных и UML. М.: Лори, 2002. 420 с.

ється підхід Code-First, а не Model-First), використання ERwin дає ряд переваг. Можливості редагування і візуалізації в середовищі ERwin вельми широкі, так, наприклад, створення відносин можливо за допомогою перетягування атрибута з однієї сутності в іншу. Таке редагування моделі дозволяє вносити зміни і проводити нормалізацію швидше й ефективніше, ніж з використанням інших інструментів. Для того, щоб додати новий елемент на діаграму, його просто потрібно вибрати на панелі інструментів і перенести в потрібне місце діаграми. Додавши нову сутність на діаграму, в неї можна додати атрибути, не відкриваючи ніяких редакторів, а просто ввести їх назви прямо на діаграмі. Таким чином, ERwin дозволяє значно знизити час на створення самої діаграми і сконцентруватися на самих завданнях, що стоять перед розробником [15][1)].

ER-модель бази даних веб-застосунку представлена на рис. 2.1.

Після аналізу предметної області було виявлено декілька сутностей, необхідних для роботи веб-застосунку:

а) Фільм є головною сутністю в системі. Таблиця зберігає інформацію про всі завантажені фільми. Назва таблиці: films. Поля таблиці:
   1) Id – унікальний ідентифікатор фільму в системі;
   2) Title – назва фільму;
   3) Tagline – слоган фільму;
   4) Description – опис фільму;
   5) Search_query – пошуковий запит для пошуку фільму на торрент–трекерах;
   6) Search_filter – додаткові опції пошуку;
   7) Big_image – банер фільму;
   8) Mainpage_show – прапор, чи потрібно показувати фільм в рекламному блоці на головній сторінці;
   9) Trailer_id – ідентифікатор трейлера фільму на Youtube;

[1)] [15] Entity Framework Code First на практике. Хабрахабр. (загол. з екрана). URL: https://habrahabr.ru/post/236037 (дата звернення 3.039.2019)

10) Update_time – unix час останнього пошуку роздач на торрент–трекерах;
11) Search_topics – прапор, чи потрібно шукати фільм на торрент–трекерах;
12) Search_trailers – прапор, чи потрібно автоматично оновлювати трейлер для фільму;
13) Release_time – час появи першої роздачі в хорошій якості;
14) Serial – прапор, чи є фільм серіалом.

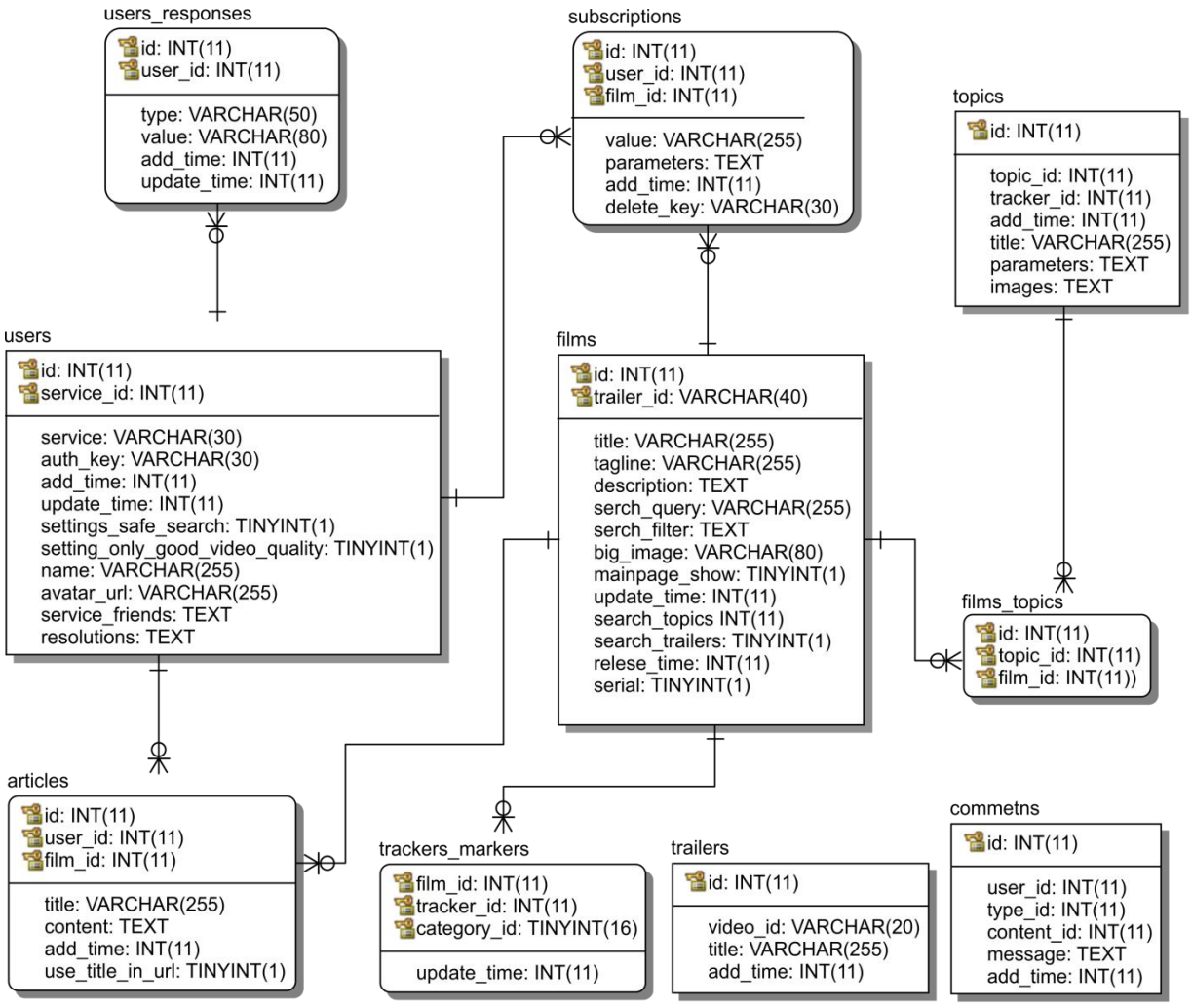

Рисунок 2.1 – ER-структура бази даних

б) Користувач. Для відстеження фільмів необхідно ідентифікувати користувача. Це робиться за допомогою реєстрації користувача. Таб-

лиця зберігає інформацію про всі зареєстровані в системі користувачів. Назва таблиці: users. Поля таблиці:

1) Id – унікальний ідентифікатор користувача в системі;
2) Service – код сервісу реєстрації, з його допомогою можна визначити яку соціальну мережу користувач використовує для авторизації;
3) Service_id – унікальний ідентифікатор користувача в соціальній мережі;
4) Auth_key – унікальний ключ для виконання віддалених запитів до API;
5) Add_time - unix – час реєстрації користувача в системі;
6) Update_time - unix – час останньої зміни профілю;
7) Setting_safe_search – прапор, чи шукати фільми тільки на якісних торрент–трекерах;
8) Setting_only_good_video_quality – прапор, повідомляти чи тільки про фільми в хорошій якості;
9) Name – повне ім'я користувача;
10) Avatar_url – посилання на аватар користувача;
11) Service_friends – масив ідентифікаторів користувачів, які є друзями у соціальній мережі;
12) Resolutions – масив роздільних здатностей відео, які цікаві користувачеві.

в) Стаття. Для ознайомлення користувачів з фільмами на сайт були додані статті, кожна з яких присвячена конкретному фільму. Таблиця зберігає інформацію про всі статті в системі. Назва таблиця: articles. Поля таблиці:

1) Id – унікальний ідентифікатор статті в системі;
2) Title – назва статті;
3) Content – зміст статті;
4) User_id – ідентифікатор користувача, що додав статтю;

5) Film_id – ідентифікатор фільму, якому присвячена дана стаття;
6) Add_time – unix час додавання статті в систему;
7) Use_title_in_url – прапор, додавати чи назва фільму в кінець назви статті. Використовується для однотипних назв, таких як «Огляд фільму »або« Рецензія на фільм».

г) Трейлер. Для вирішення завдання інформування користувачів про новинки кіноіндустрії був розроблений компонент трейлерів. Він дозволяє в автоматичному режимі поповнювати список нових трейлерів і показувати їх користувачам. Таблиця зберігає в собі інформацію про всі трейлери в системі. Процес оновлення списку трейлерів виконується щогодини. Скрипт використовує в якості джерела канал «Офіційні трейлери» на Youtube.com. Назва таблиці: trailers. Поля таблиці:

1) Id – унікальний ідентифікатор трейлера в системі;
2) Video_id – ідентифікатор трейлера фільму на Youtube;
3) Title – назва фільму, якому присвячений трейлер;
4) Add_time - unix – час додавання трейлера в систему.

д) Маркер торрент-трекера. Таблиця зберігає інформацію про останні результати пошуку на торрент-трекерах по кожному фільму. За допомогою такого підходу можна відфільтрувати нові роздачі із загального списку роздач. Назва таблиці: trakers_markers. Поля таблиці:

1) Film_id – ідентифікатор фільму;
2) Traker_id – ідентифікатор торрент-трекера;
3) Category_id – ідентифікатор категорії на торрент-трекер;
4) Update_time – unix час публікації найсвіжішої роздачі.

е) Підписка. Таблиця зберігає інформацію про всі підписки користувачів. При додаванні нової підписки, користувач починає отримувати повідомлення про нові роздачі, які задовольняють налаштуванню його профілю. Назва таблиці: subscriptions. Поля таблиці:

1) Id – унікальний ідентифікатор підписки в системі;
2) User_id – ідентифікатор користувача, якому належить підписка;
3) Film_id – ідентифікатор фільму;
4) Value – пошуковий запит, який ввів користувач при пошуку фільму;
5) Parameters – додаткові параметри підписки;
6) Add_time - unix – час додавання підписки в систему;
7) Delete_key – символьний ключ, призначений для видалення підписки без авторизації користувача. Використовується в поштовому повідомленні для скасування підписки на фільм.

ж) Роздача на торент-трекер. Таблиця зберігає в собі інформацію про всі знайдені на торент-трекерах роздачі. Назва таблиці: topics. Поля таблиці:

1) Id – унікальний ідентифікатор роздачі в системі;
2) Topic_id – унікальний ідентифікатор роздачі на торрент-трекер;
3) Tracker_id – ідентифікатор торрент-трекера;
4) Add_time – час додавання роздачі в систему;
5) Title – назва роздачі;
6) Parameters – додаткові параметри роздачі: дозвіл відео і якість відео;
7) Images – масив посилань на зображення, знайдені в роздачі.

з) Метод повідомлення користувача. Таблиця зберігає в собі інформацію про всі доступні способи повідомлення користувачів. Назва таблиці: users_responses. Поля таблиці:

1) Id – унікальний ідентифікатор в системі;
2) User_id – ідентифікатор користувача;
3) Type – спосіб повідомлення, наприклад, через електронну пошту;

4) Value – значення, може містити будь-яку інформацію в залежності від способу повідомлення, наприклад, адресу електронної скриньки;
5) Add_time - unix – час додавання в систему;
6) Update_time - unix – час останнього оновлення.

к) Коментар. У цій таблиці зберігаються коментарі до фільмів, трейлерів і статей. Назва таблиці: comments. Поля таблиці:
1) Id – унікальний ідентифікатор коментаря в системі;
2) User_id – ідентифікатор користувача, який написав коментар;
3) Type_id – ідентифікатор, що вказують на тип контенту, до якого відноситься коментар (фільм, трейлер, стаття);
4) Content_id – ідентифікатор сутності (фільму, трейлера або статті), до якої належить цей коментар;
5) Message – текст коментаря;
6) Add_time - unix – час додавання коментарів в систему.

Маючи перед очима ER-модель бази даних можна приступити до написання класів моделі. Для реалізації підходу Code-First використовується Entity Framework версія 6.1.3.

Entity Framework є продовженням технології Microsoft ActiveX Data і надає можливість роботи з базами даних через об'єктно-орієнтований код C#. Цей підхід надає ряд істотних переваг: не потрібно турбуватися про код доступу до даних, не потрібно знати деталей роботи СКБД SQL Server і синтаксису мови запитів T-SQL, замість цього можна працювати з таблицями бази даних як з класами C #, з полями цих таблиць – як з властивостями класів, а синтаксис SQL-запитів, який раніше в ADO.NET потрібно було вставляти в код C # у вигляді команд, замінений на більш зручний підхід з LINQ. Entity

Framework бере на себе обов'язки по перетворенню коду C # в SQL-інструкції [16][1].

При роботі з Entity Framework надаються величезні можливості по створенню моделі бази даних за допомогою інтегрованого середовища розробки Visual Studio. При використанні підходу Code-First спочатку визначається модель в коді, а потім, на її основі створюється (або модифікується) база даних.

2.3 Моделювання динамічних і поведінкових аспектів системи

Моделювання програмної системи було виконано з використанням уніфікованої мова моделювання (Unified Modeling Language, UML), яка є графічною мовою для специфікації, візуалізації, проектування та документування систем. За допомогою UML можна розробити детальну модель створюваної системи, яка буде показувати не тільки її концепцію, а й конкретні особливості реалізації. В рамках UML-моделі все уявлення про систему фіксуються у вигляді спеціальних графічних конструкцій – діаграм [17][2].

Розроблено наступні діаграми:

1) Діаграма прецедентів (use case diagram).
2) Діаграма послідовностей (sequence diagram).
3) Діаграма станів (statechart diagram).

2.3.1 Діаграма прецедентів

Будь-які (в тому числі і програмні) системи проектуються з урахуванням того, що в процесі своєї роботи вони будуть використовуватися людьми і/або взаємодіяти з іншими системами. Суті, з якими взаємодіє система в

[1] [16] Introduction to Entity Framework. (загол. з екрана). URL: https://msdn.microsoft.com/en-us/library/aa937723(v=vs.113).aspx (дата звернення 13.05.2019)

[2] [17] Unified Modeling Language (UML) – Вікіпедія. (загол. з екрана). URL: https://uk.wikipedia.org/wiki/Unified_Modeling_Language (дата звернення 13.05.2019)

процесі своєї роботи, називаються Актори, причому кожен Актор очікує, що система буде вести себе строго певним, передбачуваним чином. Актор (actor) – це безліч логічно пов'язаних ролей, виконуваних при взаємодії з прецедентами або сутностями. Актором може бути людина або інша система, підсистема або клас, які представляють щось поза суті. Прецедент (use-case) – опис окремого аспекту поведінки системи з точки зору користувача.

Діаграми прецедентів представляють динамічні або поведінкові аспекти системи.

Цілі створення діаграм прецедентів:

1) Визначення межи і контексту предметної області, що моделюється, на ранніх етапах проектування.

2) Формування загальних вимог до поведінки проектованої системи.

3) Розробка концептуальної моделі системи для її подальшої деталізації.

4) Підготовка документації для взаємодії з замовниками і користувачами системи.

Діаграма прецедентів, що була розроблена для проектованої програмної системи наведена на рис.2.2.

2.3.2 Діаграма послідовностей

Діаграма послідовностей відображає взаємодію об'єктів в динаміці. В UML взаємодія об'єктів розуміється як обмін інформацією між ними. При цьому інформація набуває вигляду повідомлень. Крім того, що повідомлення несе якусь інформацію, воно певним чином також впливає на одержувача. В цьому плані UML повністю відповідає основним принципам ООП, відповідно до яких інформаційну взаємодію між об'єктами зводиться до відправки і прийому повідомлень.

Діаграма послідовностей відноситься до діаграм взаємодії UML, що описує поведінкові аспекти системи, але розглядає взаємодію об'єктів в часі.

Іншими словами, діаграма послідовностей відображає тимчасові особливості передачі і прийому повідомлень об'єктами.

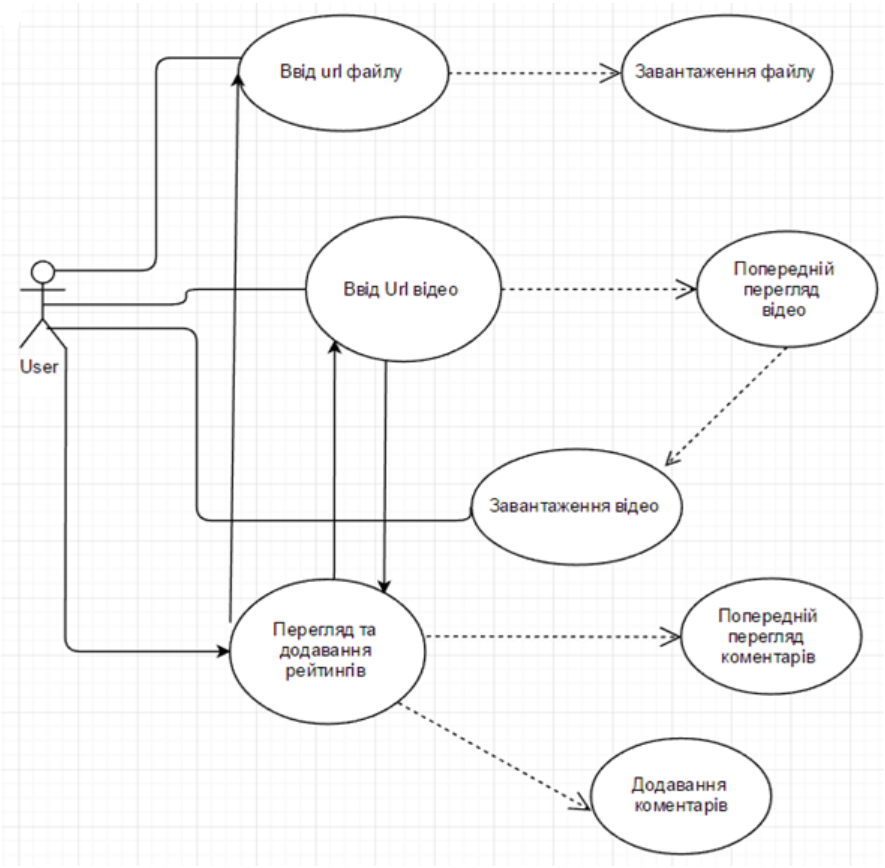

Рисунок 2.2 – Діаграма прецедентів системи

Діаграми послідовностей використовують для уточнення діаграм прецедентів і більш детального опису логіки сценаріїв використання. Це відмінний засіб документування проекту з точки зору сценаріїв використання. Діаграми послідовностей зазвичай містять об'єкти, які взаємодіють в рамках сценарію, повідомлення, якими вони обмінюються, і результати, що повертаються, пов'язані з повідомленнями. Втім, часто результати, що повертаються, позначають лише в тому випадку, якщо це не очевидно з контексту.

Діаграма послідовностей, що була розроблена для проектованої програмної системи наведена на рис.2.3.

2.3.3 Діаграма станів

Стан (state) – ситуація в життєвому циклі об'єкта, під час якої він задовольняє деякій умові, виконує певну діяльність або очікує якоїсь події. Стан об'єкта визначається значеннями деяких його атрибутів і присутністю або відсутністю зв'язків з іншими об'єктами.

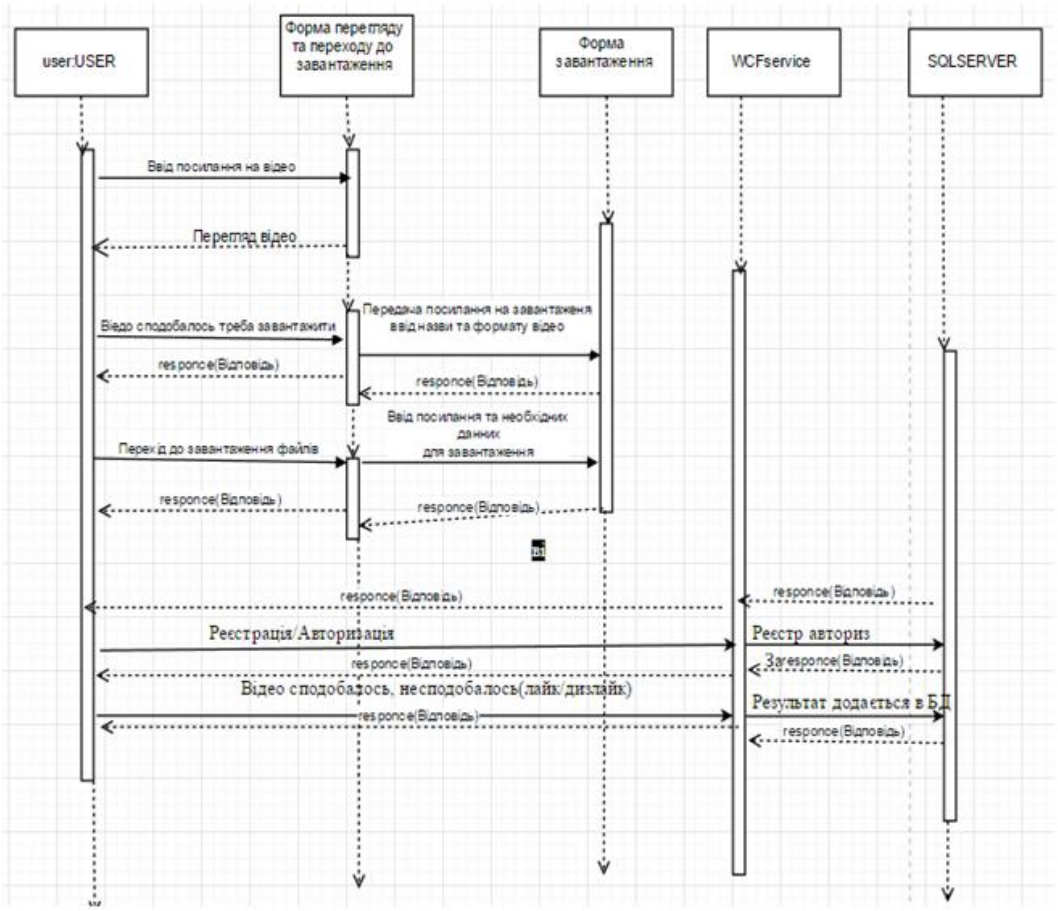

Рисунок 2.3 – Діаграма послідовностей системи

Діаграма станів показує, як об'єкт переходить з одного стану в інший. Діаграми станів служать для моделювання динамічних аспектів системи (як і діаграми послідовностей і прецедентів).

Діаграма станів корисна при моделюванні життєвого циклу об'єкта. Від інших діаграм діаграма станів відрізняється тим, що описує процес зміни

станів тільки одного примірника певного класу – одного об'єкта, причому об'єкту реактивного, тобто об'єкта, поведінка якого характеризується його реакцією на зовнішні події. Діаграма станів, що була розроблена для проектованої програмної системи наведена на рис.2.4.

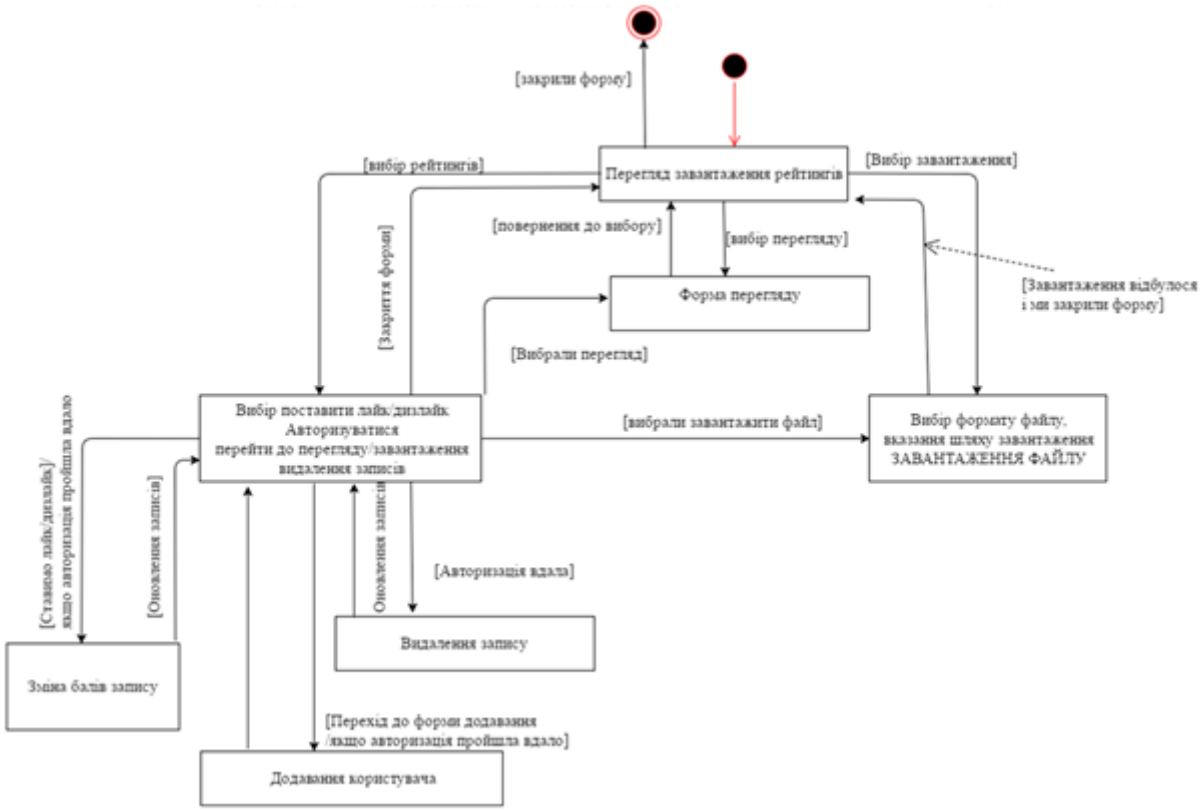

Рисунок 2.4 – Діаграма станів

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБ-ЗАСТОСУНКУ «VIEWER»

3.1 Створення Controllers

Згідно з моделлю MVC для взаємодії даних з views в програмі були створені контролери. На рис. 3.1 наведений програмний код, реалізація класів для реєстрації нового користувача. За аналогією була створена решта контролерів.

```
/// <summary>
/// View of login form
/// </summary>
/// <returns></returns>
public ActionResult Authorization()
{
    return View();
}
```

```
/// <summary>
/// Registration new user
/// </summary>
/// <param name="login">login user</param>
/// <param name="email">email user</param>
/// <param name="phone">phone user</param>
/// <param name="password">password user</param>
/// <returns></returns>
[HttpPost]
public async Task<string> Registration(string login, string email, string phone, string password)
{
    var user = new User
    {
        Login = login,
        Email = email,
        Phone = phone,
        Password = password
    };
    return await UsersRepositories.RegisterUserAsync(user);
}
```

```
/// <summary>
/// Return list user
/// </summary>
/// <returns>Return users list</returns>
public ActionResult UsersTable()
{
    var model = new UserModel
    {
        UsersList = UsersRepositories.GetUsers()
    };
    return View(model);
}
```

Рисунок 3.1 – Контролери створені в проекті

3.2 Створення сервісів (Service)

У ході роботи були створені сервіси для обробки даних з контролерів та взаємодії з БД за допомогою Entity Framework. На рис. 3.2 наведений приклад, що відповідаю дії видалення користувача.

```
/// <summary>
/// Get users list
/// </summary>
/// <returns>Returns list of users</returns>
IQueryable<User> GetUsers();

/// <summary>
/// Remove user by id
/// </summary>
/// <param name="id">Identifier of user</param>
/// <returns>Returns true or false</returns>
bool DeleteUser(int id);

/// <summary>
///  Delete user(async)
/// </summary>
/// <param name="id">Identifier of user</param>
/// <returns>Returns true or false</returns>
Task<bool> DeleteUserrAsync(int id);
```

```
private bachelorEntities userEntities = new bachelorEntities();

/// <summary>
/// Get users list
/// </summary>
/// <returns>Returns list of users</returns>
public IQueryable<User> GetUsers()
{
    return userEntities.User;
}

/// <summary>
/// Remove user by id
/// </summary>
/// <param name="id">Identifier of user</param>
/// <returns>True or false</returns>
public bool DeleteUser(int id)
{
    bool isSuccess = true;

    if (userEntities.User.Find(id) != null)
    {
        userEntities.User.Remove(userEntities.User.Find(id));
        userEntities.SaveChanges();
    }
    else { isSuccess = false; }
    return isSuccess;
}
```

Рисунок 3.2 – Сервіси для обробки даних з контролерів та взаємодії з БД

3.3 Створення Views

Для взаємодії користувачів з серверною частиною була створена клієнтська частина «**Views**». Для стилізації було використано:

– HTML;

– Bootstrap;

– CSS.

Приклад програмного коду для створення форм авторизації та реєстрації користувачів наведено на рис. 3.3. На рис.3.4 представлена форма авторизації за стосунку, а на рис. 3.5 – форма реєстрації.

```
<div class="container">
    <div class="row main">
        <div class="panel-heading">
            <div class="panel-title text-center">
                <h2 class="title">Registration</h2>
                <hr />
            </div>
        </div>
        <div class="main-login main-center">
            <form class="form-horizontal" method="post" action="#">
                <div class="form-group">
                    <label for="login" class="cols-sm-2 control-label">Your login</label>
                    <div class="cols-sm-10">
                        <div class="input-group">
                            <span class="input-group-addon"><i class="fa fa-user fa" aria-hidden="true"></i></span>
                            <input type="text" class="form-control" name="login" id="login" placeholder="Enter your l
                        </div>
                    </div>
                </div>
                <div class="form-group">
                    <label for="email" class="cols-sm-2 control-label">Your Email</label>
                    <div class="cols-sm-10">
                        <div class="input-group">
                            <span class="input-group-addon"><i class="fa fa-envelope fa" aria-hidden="true"></i></spa
                            <input type="text" onKeyUp="validationEmail();" class="form-control" name="email" id="emai
                        </div>
                    </div>
```

Рисунок 3.3 – Приклад HTML коду для форм авторизації та реєстрації користувачі системи

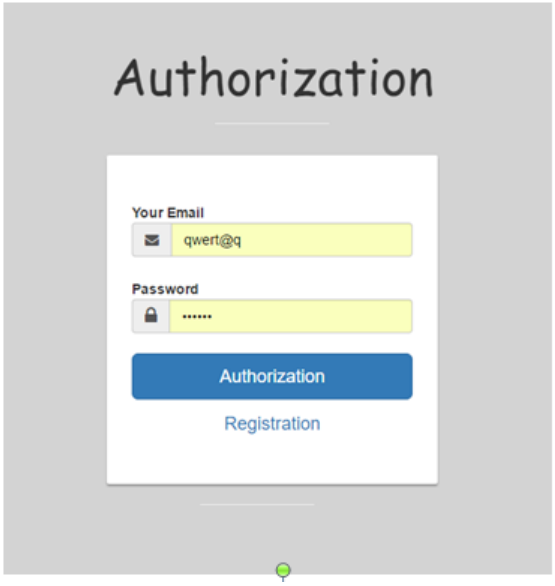

Рисунок 3.4 – Форма авторизації користувачів

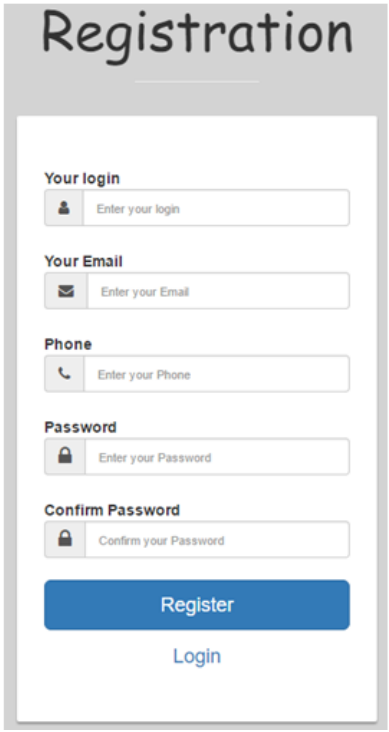

Рисунок 3.5 – Форма реєстрації користувачів

Для взаємодії користувача з серверною частиною було використано AJAX та Java Script. На рис. 3.6 наведено приклад відповідного програмного коду.

```
function GetRecords() {
    $(".main-loader").css("visibility", "visible");
    $.ajax({
        type: "Post",
        url: "@Url.Action("GetRecords", "Generator")",
        data: { type: $("#category").val(), category: $("#mood").val() },
        success: function (data) {
            if (data.records.length > 0) {
                var rand = Math.floor(Math.random() * data.records.length);
                var date = data.records[rand].Data;
                var num = parseInt(date.replace(/\D+/g, ""))
                var datetime = new Date(num);
                $("#date").text("Posted: " + datetime.toISOString());
                $('#img').attr('src', data.records[rand].IMG);
                console.log(data.records[rand]);
                $("#description").text(data.records[rand].Description);
                $("#text").text(data.records[rand].Message);
                $('#video').attr('src', data.records[rand].VIDEOURL);
            }
            $(".main-loader").css("visibility", "hidden");
        },
        error: function (data) {
            $(".main-loader").css("visibility", "hidden");
        }
});}
```

Рисунок 3.6 – Приклад JS-коду форми реєстрації

3.4 Управління користувачами та генератором

Для керування користувачами, та операціями над даними в генераторі було створено окремий контролер для адміністратора (рис. 3.7). У результаті було додано функціонал, який дозволяє редагувати/видаляти, додавати всі потрібні дані.

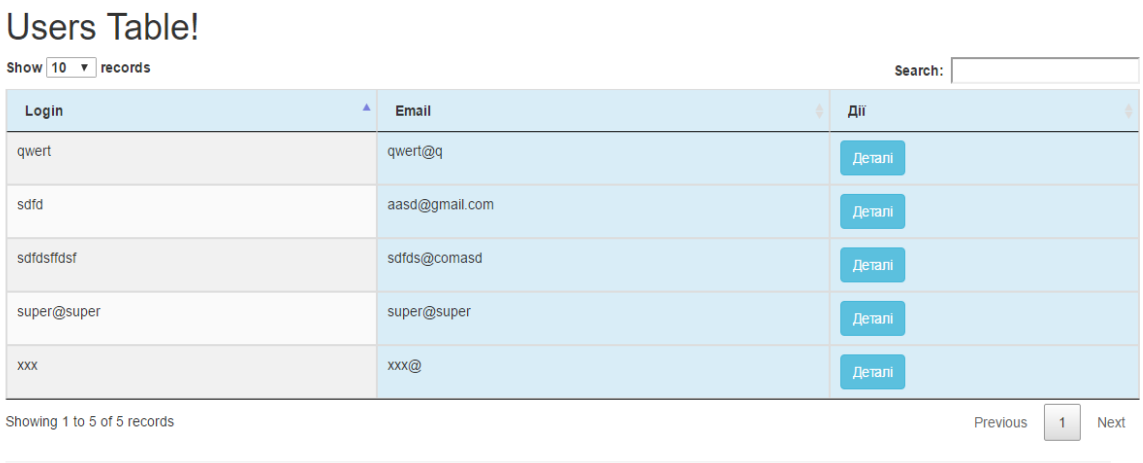

Рисунок 3.7 – Контролер для адміністратора

У ході роботи було додано перевірку на роль користувача, а також було додано фільтр, який перевіряє, чи авторизований користувач.

В якості додаткових плагинів було використано «DataTable» та «bootstrap-select». Була додана пагінація, сортування та пошук користувачів.

При натисненні на кнопку «деталі» можливо змінити дані про користувачів (рис.3.8).

Для вибору фільму, серіалу, книги, відповідна сторінка наведена на рис.3.9, треба натиснути на генератор в пункті меню, задати параметри, які підходять в селектах, після чого потрібно натиснути «генерувати».

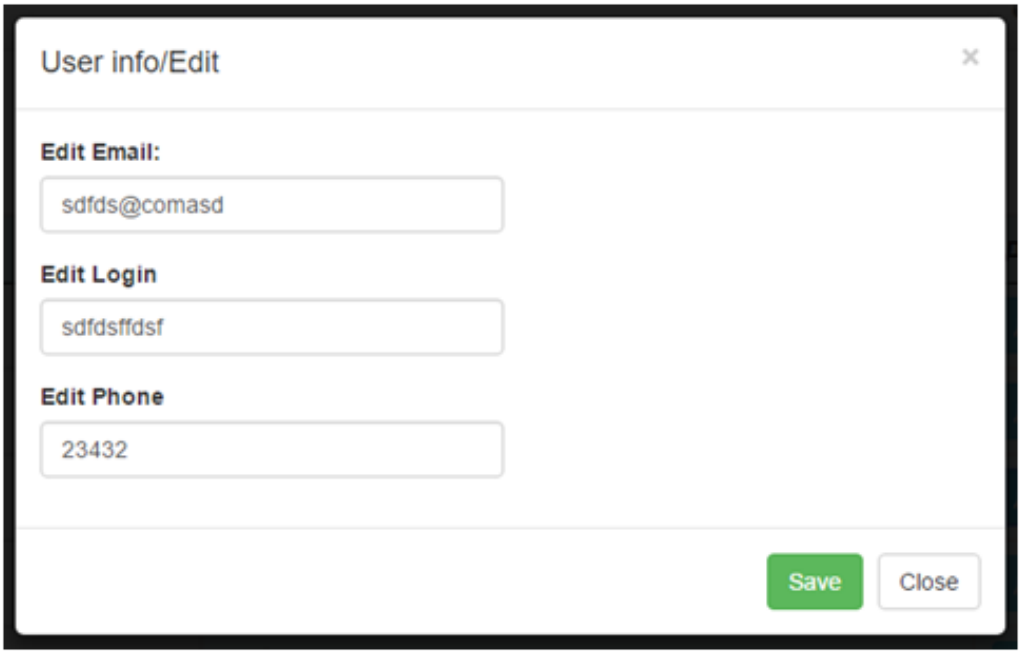

Рисунок 3.8 – Форма зміни даних про користувача

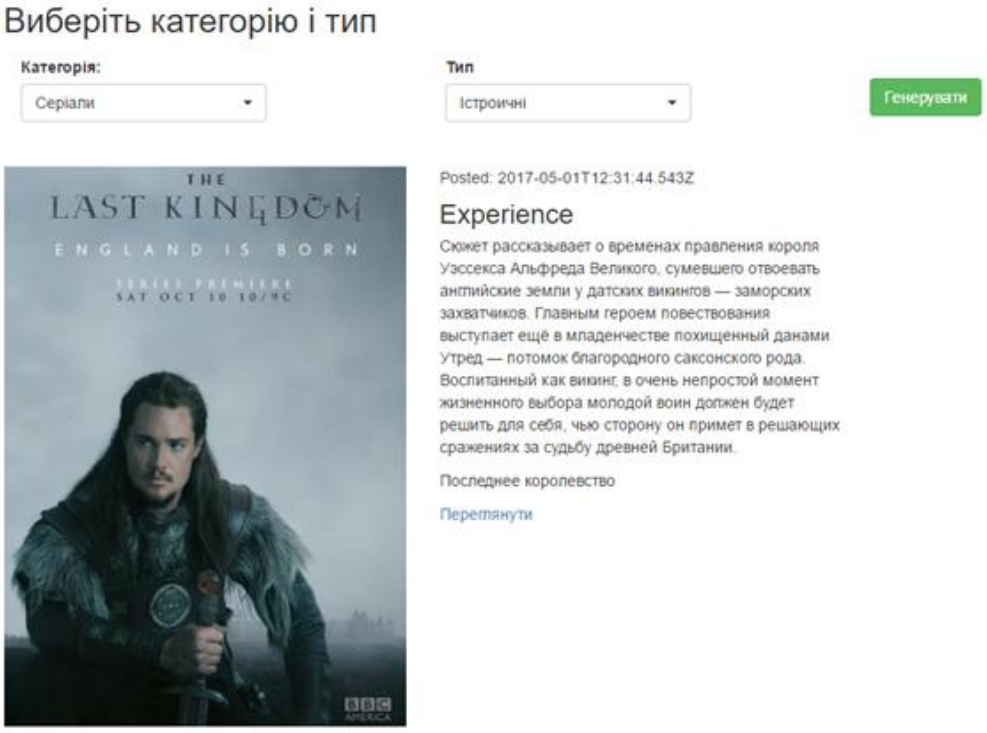

Рисунок 3.9 – Форма вибору параметрів пошуку ресурсу

3.5 Використання сторонніх API

Для пошуку книг, фільмів за ключовими тегами було використано сторонні API. Вигляд форми для пошука книг наведено на рис. 3.10.

Наприклад, для завантаження та перегляду відео файлів була використана клієнтська бібліотека API Google, що забезпечує доступ до YouTube Data API [18][1)].

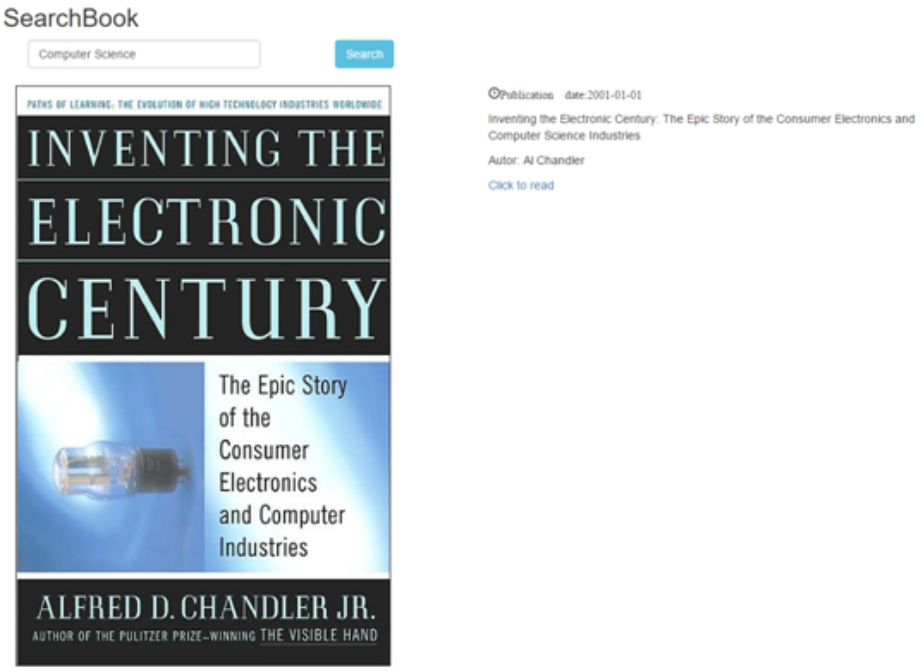

Рисунок 3.10 – Форма пошуку книг

Головне вікно програми Torrent наведено на рис. 3.11. Для користувача існує можливість введення посилання на файл, завантаження файлу, його перегляду і отримання інформації щодо рейтингу фільму.

Процес завантаження відео-файлу за заданим запитом представлено на рис.3.12.

[1)] [18] YouTube Data API. (загол. з екрана). URL:https://developers.google.com /youtube/v3/(дата звернення 13.05.2019)

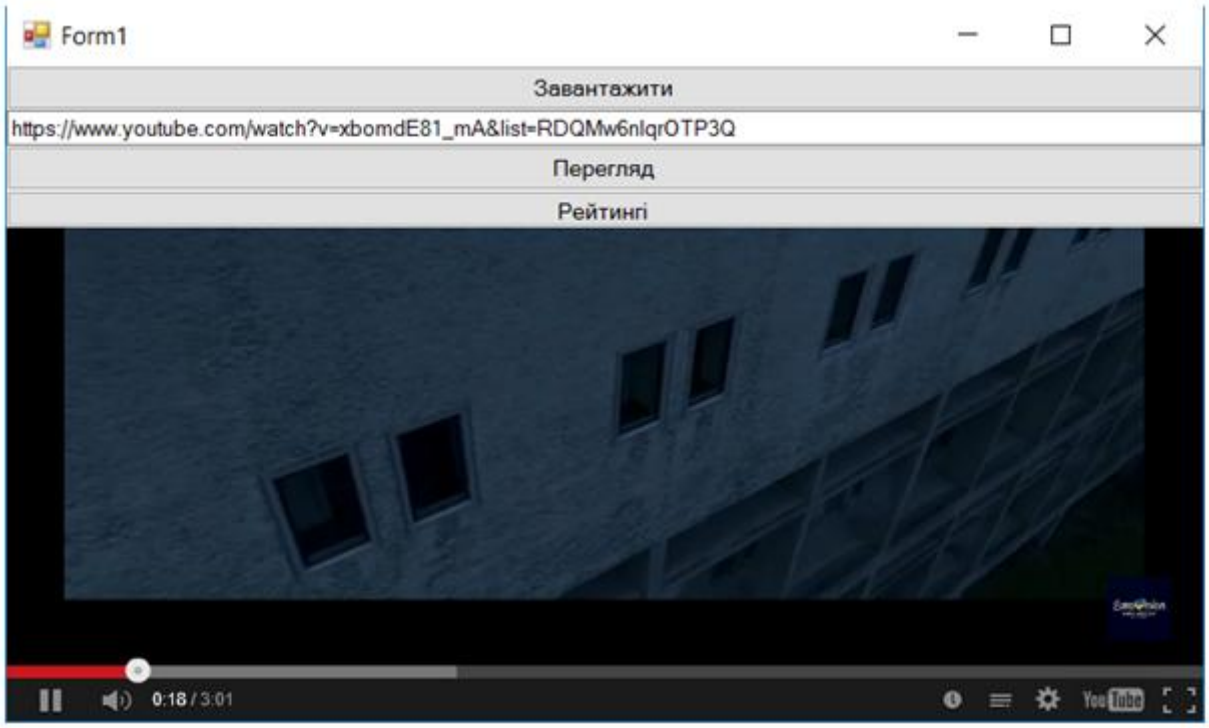

Рисунок 3.11 – Головне вікно програми Torrent

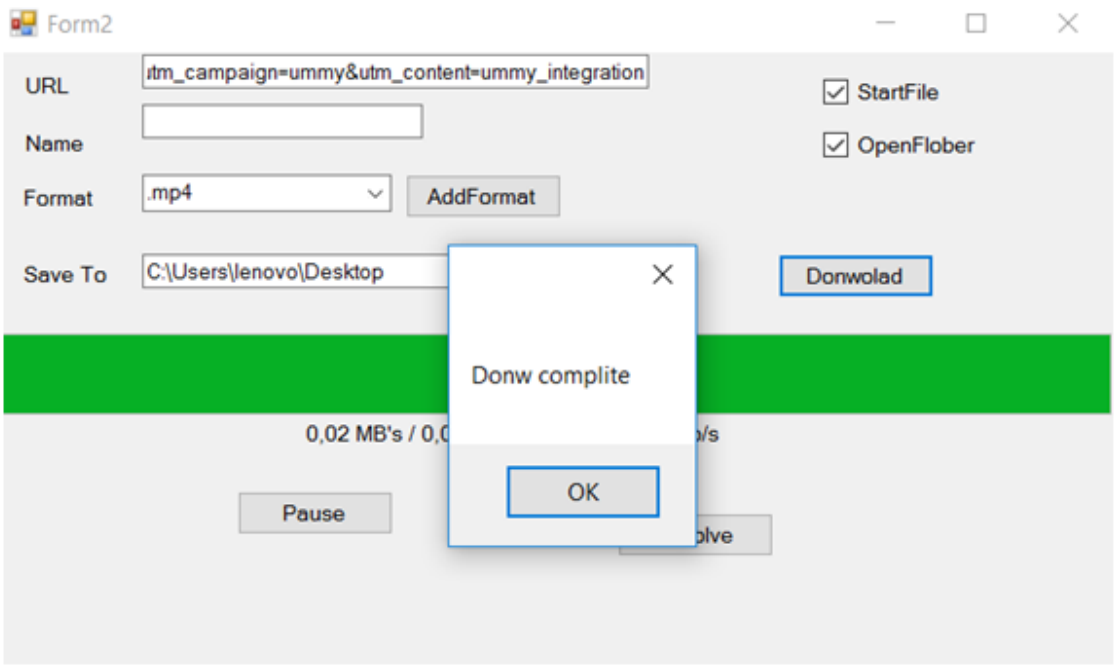

Рисунок 3.12 – Процес завантаження відео-файлу за заданим запитом

3.6 Опис та тестування веб-застосунку «Viewer»

За наведеними у попередніх розділах підходами було реалізований та протестований веб-застосунок «Viewer», який надає користувачам функції торента. Застосунок дозволяє здійснювати пошук відео-файлів за критеріями, виконувати їх завантаження, залишити та зберегти коментар. Всі можливі виключення були оброблені. Головна сторінка веб-застосування наведена на рис.3.13. Як можна побачити, в меню головної сторінки знаходяться пункти: про нас, переваги і команда. Вигляд сторінки «Про наш сайт» представлений на рис.3.14.

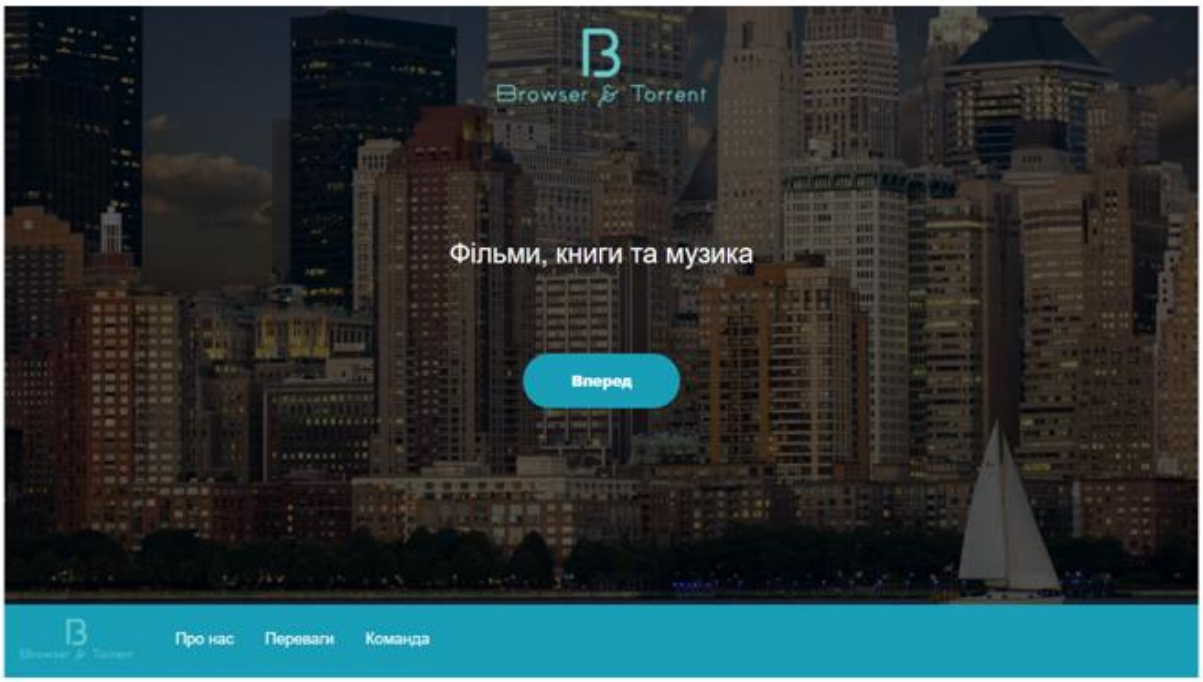

Рисунок 3.13 – Головна сторінка веб-застосунку Viewer

За результатами дослідження розроблено структуру програми веб-застосунку, яка приведена у додатку А.

Розроблений веб-застосунк «Viewer» може бути використаний:

– в освітніх та розважальних цілях;

– для завантаження файлів.

Крім того, сайт підтримує відображення новинного контексту. На рис.3.15 приведений вигляд сторінки «Новини».

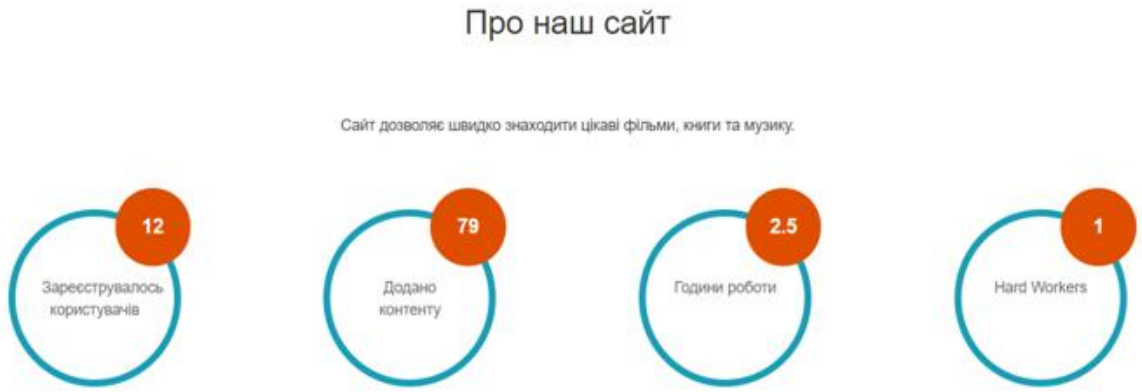

Рисунок 3.14 – Сторінка веб-застосунку Viewer “Про наш сайт”

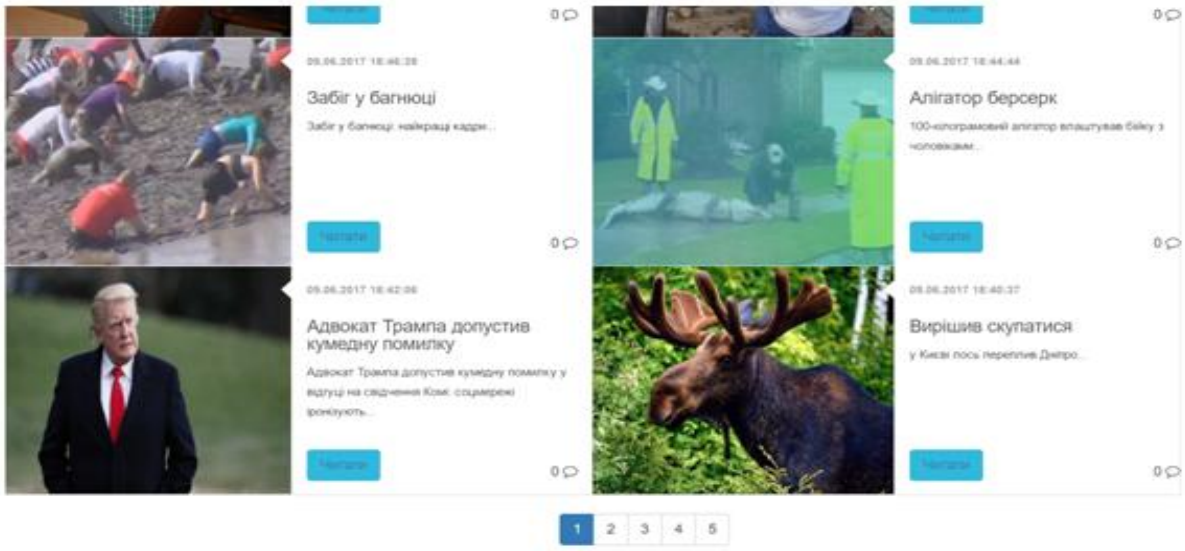

Рисунок 3.15 – Сторінка веб-застосунку Viewer “Новини”

Для роботи з веб-застосунком від користувача не потрібні якісь спеціальні навички, вміння та досвід роботи з ПК.

Переваги програми:

– можливість швидко знайти цікавий фільм, додаток, відео;

– можливість комфортно завантажити потрібний файл;

– зручний перегляд рейтингів і відгуків фільмів.

Головна сторінка сайту наведена на рис.3.16.

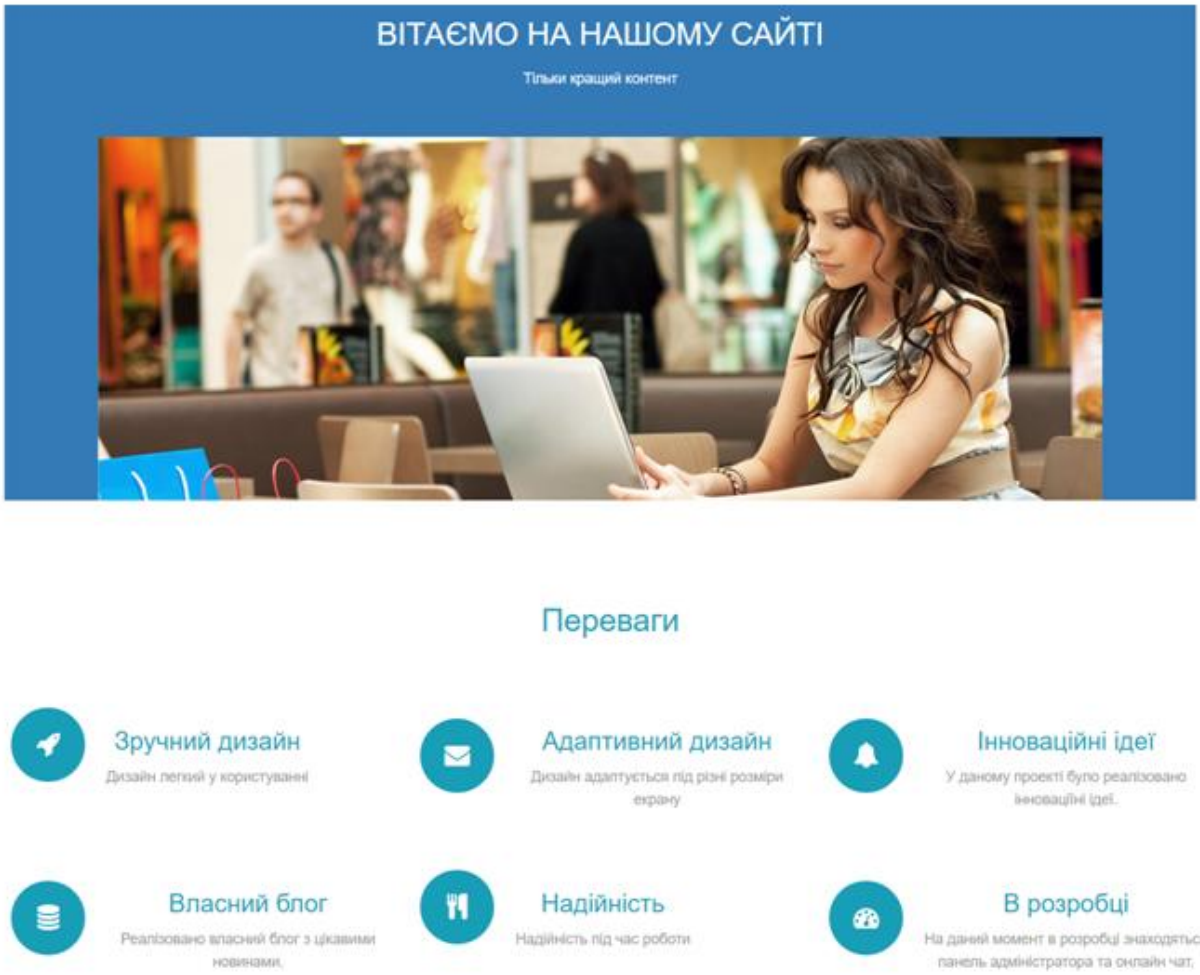

Рисунок 3.16 – Головна сторінка сайту

Таким чином веб-застосунку «Viewer» є повноцінним торентом і дозволяє завантажувати файли за прямим посиланням, а після завантаження чи он-лайн перегляду дозволяє залишити відгук (коментар) про якість продукту. Були виконані всі заплановані на початку роботи функції за стосунку.

ВИСНОВКИ

В кваліфікаційній роботі виконана розробка веб-застосунку «Viewer», який є повноцінним торентом і дозволяє завантажувати файли за прямим посиланням, а після завантаження чи он-лайн перегляду дозволяє залишити відгук (коментар) про якість продукту.

Ключові функції, які було реалізовано, є:

1) Робота з запитами користувачів (додавання, перегляд, видалення).
2) Робота з фільмами (додавання, видалення).
3) Робота зі статтями (додавання, редагування, видалення);
4) Авторизація користувачів;

В роботі на основі знань про принципи, структуру та логіку роботи торент-трекерів було визначено набір сутностей. На основі отриманого набору сутностей спроектовано і реалізовано базу даних. За допомогою уніфікованої мова моделювання UML розроблено діаграму прецедентів, діаграму послідовностей і діаграму станів для проектованої програмної системи. Для застосунку був реалізувано зручний інтерфейс для взаємодії з системою.

На основі ретельного дослідження середовищем реалізації було вибрано технологію .NET C# ASP MVC 5 з використанням ajax запитів.

Для застосунку «Viewer» технологією реалізації було вибрано C# .Net з використання WCF service Entity Faremwork та баз даних SQL.

У роботі описана загальна характеристика, об’єктну модель та принципи роботи ASP MVC 5. Для завдання розмітки та стилів було використано мову розмітки HTML та стилі CSS. Також розглянуто і використано можливості js та jquery для динамічного оновлення сторінки.

Для роботи з веб-застосунком від користувача не потрібні якісь спеціальні навички, вміння та досвід роботи з ПК.

Переваги програми:

– можливість швидко знайти цікавий фільм, додаток, відео;
– можливість комфортно завантажити потрібний файл;

– зручний перегляд рейтингів і відгуків фільмів.

Результати тестування за стосунку підтвердити його працездатність і здатність в режимі реальному часі швидко знаходити цікаві фільми, книги, завантажувати їх, а також додавати до списку рейтингів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Модель-вид-контролер – Вікіпедія. (загол. з екрана) URL: https:// uk.wikipedia.org/wiki/Модель-вид-контролер (дата звернення 13.05.2019)
2. Фримен Эрик, Фримен Элизабет. Паттерны проектирования. СПб.: Питер, 2011. 656 с.
3. Гонсалвес Энтони, Изучаем Java EE 7. СПб.: Питер, 2014. 640 с.
4. Митчелл С., Уолтер С., Севен Д. ASP.NET: советы, рекомендации, примеры. М.: Вильямс, 2002. 863 с.
5. Freeman A. «Pro ASP.NET MVC 5 Platform». Apress, 2014. 428 p
6. Троелсен Э. C# и платформа .NET. Библиотека программиста. СПб.: Питер, 2002. 800 с.
7. Troelsen Andrew. Pro C# 5.0 and the .NET 4.5. Framework Apress, 2012. 1463 p.
8. Benedetti R., Cranley R. Head First jQuery.O'Reilly Media, 2011. 544 p.
9. Jon Galloway, Brad Wilson K., Scott Allen, David Matson. Professional ASP.NET MVC 5. John Wiley & Sons, 2014. 624 p.
10. Munro Jamie. ASP.NET MVC 5 with Bootstrap and Knockout.js. O'Reilly Media, 2015. 278 p.
11. Bootstrap – Вікіпедія. (загол. з екрана). URL: https://uk.wikipedia.org/ wiki/Bootstrap (дата звернення 3.039.2019)
12. Чебыкин Р.И., Самоучитель HTML и CSS. Современные технологии. СПБ.: БХВ-Петербург, 2008. 624 с.
13. Faithe Wempen. HTML5 Step by Step. Microsoft Press, 2011. 416 p.
14. Мюллер Роберт Дж. Базы данных и UML. М.: Лори, 2002. 420 с.
15. Entity Framework Code First на практике. Хабрахабр. (загол. з екрана). URL: https://habrahabr.ru/post/236037 (дата звернення 13.05.2019)

16. Introduction to Entity Framework. (загол. з екрана). URL: https://msdn.microsoft.com/en-us/library/aa937723(v=vs.113).aspx (дата звернення 13.05.2019)
17. Unified Modeling Language (UML) – Вікіпедія. (загол. з екрана). URL: https://uk.wikipedia.org/wiki/Unified_Modeling_Language (дата звернення 13.05.2019)
18. YouTube Data API. (загол. з екрана). URL: https://developers.google.com /youtube/v3/(дата звернення 13.05.2019)

ДОДАТОК А

Структура програми веб-застосунку