МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»

Кафедра інженерії програмного забезпечення

**КУРСОВА РОБОТА**

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

з дисципліни: «Web-технології Ч.2»

на тему:

**«ВЕБ-ДОДАТОК ЛЮБИТЕЛІВ КІНО»**

студента І курсу групи ІПЗк-21-1

спеціальності 121 «Інженерія

програмного забезпечення»

Остапчука Богдана Володимировича

(прізвище, ім’я та по-батькові)

Керівник:, ктн, доц. А.В. Морозов

Дата захисту: " \_\_\_ " \_\_\_\_\_\_\_\_\_\_\_\_ 2022 р.

Національна шкала \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Оцінка: ECTS \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_

Члени комісії: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ М.С. Граф

(підпис) (прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ О.Г. Чижмотря.

(підпис) (прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ О.В. Кузьменко

(підпис) (прізвище та ініціали)

Житомир – 2022

ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»

Факультет інформаційно-комп’ютерних технологій

Кафедра інженерії програмного забеспечення

Освітній рівень: бакалавр

Спеціальність 121 «Інженерія програмного забезпечення»

«ЗАТВЕРДЖУЮ»

В.о. зав. кафедри ІПЗ

\_\_\_\_\_\_\_\_\_\_\_\_ А.В.Морозов

“\_\_\_\_” \_\_\_\_\_\_\_\_\_\_\_\_\_2021р.

ЗАВДАННЯ

НА КУРСОВИЙ ПРОЕКТ СТУДЕНТУ

Остапчуку Богдану Володимировичу

1. Тема роботи: Розробка сайту «Веб-додаток любителів кіно»

керівник роботи: к.т.н., доцент Морозов Андрій Васильович.

1. Строк подання студентом: “\_\_”\_\_\_\_\_\_\_\_\_2021р.
2. Вихідні дані до роботи: Розробити сайт на тему «Веб-додаток любителів кіно».
3. Зміст розрахунково-пояснювальної записки (Перелік питань, які підлягають розробці)
   * + 1. Аналіз аналогічних розробок;
       2. Алгоритми роботи програми;
       3. Опис роботи програми;
4. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
5. Презентація
6. https://github.com/BogdanOstapchuk/ip-kurswork
7. Консультанти розділів проекту (роботи)

|  |  |  |  |
| --- | --- | --- | --- |
| Розділ | Прізвище, ініціали та посади консультанта | Підпис, дата | |
| завдання  видав | завдання прийняв |
| 1,2,3 | Чижмотря О.Г., ст. викладач каф. ІПЗ |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

1. Дата видачі завдання “15”\_вересня\_ 2021 р.

|  |  |  |  |
| --- | --- | --- | --- |
| № з/п | Назва етапів курсової роботи | Строк виконання етапів роботи | Примітки |
| 1 | Постановка задачі | 16 вересня 2021 | виконано |
| 2 | Пошук, огляд та аналіз аналогічних розробок | 1 жовтня 2021 | виконано |
| 3 | Формулювання технічного завдання | 15 жовтня 2021 | виконано |
| 4 | Опрацювання літературних джерел | 1 листопада 2021 | виконано |
| 5 | Проектування структури | 3 грудня 2021 | виконано |
| 6 | Написання програмного коду | 13 грудня 2021 | виконано |
| 7 | Відлагодження | 22 грудня 2021 | виконано |
| 8 | Написання пояснювальної записки | 28 грудня 2021 | виконано |
| 9 | Захист |  |  |

**КАЛЕНДАРНИЙ ПЛАН**

**Студент** \_ \_\_\_\_\_\_\_\_\_\_\_\_\_ Остапчук Б. В.

(підпис) (прізвище та ініціали)

**Керівник роботи** \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Чижмотря О. Г.

(підпис) (прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до курсової роботи на тему сайт «Веб-додаток любителів кіно» складається з переліку умовних скорочень, вступу, трьох розділів, висновків, списку використаної літератури та додатку.

Текстова частина викладена на 41 сторінках друкованого тексту.

Пояснювальна записка має 7 сторінок додатків. Список використаних джерел містить 12 найменувань і займає 1 сторінку. В роботі наведено 32 рисунків. Загальний обсяг роботи – 48 сторінок.

У першому розділі було обґрунтувано створення сайту на тему «Веб-додаток любителів кіно».

У другому розділі проведено проектування і розробка сайту.

У третьому розділі проведено тестування програмного продукту.

Висновок містить в собі результати виконаної роботи створення сайту на тему «Веб-додаток любителів кіно».

У додатку представлений лістинг розробленого програмного продукту.

Ключові слова: PHP, MYSQL, JAVASCRIPT, WEB-САЙТ, MVC, MODEL, CONTROLLER, ЗАПИТИ, ЗАПИСИ, КЛІЄНТ, СЕРВЕР, АДМІН.

**ЗМІСТ**

[ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ 6](#_Toc516611570)

[ВСТУП 7](#_Toc516611571)

[РОЗДІЛ 1 АНАЛІЗ ПРОБЛЕМАТИКИ, МЕТОДІВ ТА ЗАСОБІВ ВИРІШЕННЯ ЗАДАЧІ 8](#_Toc516611572)

[1.1 Аналіз задачі, засобів та методів її вирішення 8](#_Toc516611573)

[1.2 Аналіз існуючого програмного забезпечення за тематикою курсової роботи 10](#_Toc516611574)

[1.3 Технічне завдання на курсову роботу 14](#_Toc516611575)

[Висновки з першого розділу 17](#_Toc516611576)

[РОЗДІЛ 2 ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 18](#_Toc516611577)

[2.1 Загальний алгоритм роботи веб-додатку 18](#_Toc516611578)

[2.2 Розробка функціональних алгоритмів роботи програми 19](#_Toc516611579)

[2.3 Розробка програмного забезпечення 22](#_Toc516611580)

[Висновки з другого розділу 27](#_Toc516611581)

[РОЗДІЛ 3 ОПИС РОБОТИ З ПРОГРАМНИМ ДОДАТКОМ ТА ЙОГО ТЕСТУВАННЯ 28](#_Toc516611582)

[3.1 Опис роботи з додатком 28](#_Toc516611583)

[3.2 Тестування веб-додатку 35](#_Toc516611584)

[Висновоки до третього розділу 38](#_Toc516611585)

[ВИСНОВКИ 39](#_Toc516611586)

[ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ 40](#_Toc516611587)

[ДОДАТКИ 41](#_Toc516611588)

# ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

БД – База даних

MVC – Модель–вигляд–контролер (або Модель–представлення–контролер, Model-view-controller, MVC) — архітектурний шаблон, який використовується під час проектування та розробки програмного забезпечення.

ПЗ – Програмне забезпеченя

AJAX — підхід до побудови користувацьких інтерфейсів веб-застосунків, 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.

# ВСТУП

В даному курсовому проекті наведено процес створення програмного забезпечення (ПЗ), сайту, що стане корисним для людей, які хочуть вибрати кіно на будь-який смак та переглянути додаткову інформацію про вибране кіно.

Актуальність роботи полягає в застосуванні сучасних технологій об'єктно-орієнтованого підходу в програмуванні. Даний курсовий проект показує на прикладі роботу ООП. Проект зберігає дані в БД та може надавати аналіз даних, що зберігаються.

Мета роботи полягає в розробці програмного продукту - розробка програмного комплексу з використанням об'єктно-орієнтованого підходу і користувальницьких класів. Для досягнення мети необхідно провести об'єктно-орієнтований аналіз, об'єктно-орієнтоване проектування, об'єктно-орієнтоване програмування і зібрати статистичні результати.

Об'єкт дослідження курсової роботи - процес обробки інформації наданої користувачами для здійснення статистичного зрізу чи надання інформації відповідно за запитом користувача.

Предмет дослідження – сайт, що надає користувачу інформацію за запитом, накопичує дані з побажаннями користувачів для подальшої обробки та аналізує дані опитування.

Для вирішення поставлених завдань були використані наступні методи дослідження: аналіз, синтез, систематизація, зіставлення, що дозволило більш глибоко вивчити тему, яка вивчається.

## РОЗДІЛ 1 АНАЛІЗ ПРОБЛЕМАТИКИ, МЕТОДІВ ТА ЗАСОБІВ ВИРІШЕННЯ ЗАДАЧІ

### 1.1 Аналіз задачі, засобів та методів її вирішення

Основною задачею програми є обробка, подання та зберігання даних, але перед тим як починати роботу над ПЗ потрібно ознайомитись з методом програмування, основними поняттями та способами зберігання та обробки інформації.

При розробці даного сайту ми будем використовувати об’єктно-орієнтований підхід, адже він має безліч переваг над іншими методами програмування.

В основі об'єктно-орієнтованої мови програмування лежать два основних поняття: клас та об'єкт. Об'єкт – це базове поняття в ООП, це конкретна реалізація, екземпляр класу. Об'єкт складається з трьох частин: стан (змінні стану), методи (операції), ім'я об'єкта. Клас - це група даних і методів або функцій для роботи з цими даними, це шаблон. Об'єкти однаковими наборами змінних стану і методів, утворюють клас. Якщо об'єкти мають реалізацію з конкретного світу, то класи є абстракціями. Трохи більш складні об'єкти можуть взагалі не містити даних, а представляти процес і містити тільки функції, які реалізують цей процес. Для формування реального об'єкта необхідно мати шаблон, по прикладу якого і будується даний об'єкт.

Важлива часина даного проекту є зберігання та обробка інформації.

Зберігання інформації - комплекс заходів, спрямований на забезпечення збереження повноти і цілісності сформованих даних про певну інформацію, створення і підтримання належних умов для їх використання, а також запобігання несанкціонованому доступу, поширенню і використанню.

Найзручніший спосіб збереження інформації є база даних.

База даних – це інструмент, використовуючи який, можна збирати й упорядковувати інформацію. У базах даних можна зберігати відомості про людей, продукти, замовлення тощо.

Комп’ютеризована БД – це контейнер об’єктів. Одна БД може містити кілька таблиць. Наприклад, система відстеження запасів, у якій використовуються три таблиці, – це не три БД, а одна база даних, що містить три таблиці.

Тому, в даній програмі буде використано базу даних MуSQL Server.

Для роботи з даним програмним продуктом потрібно знати та використовувати мову SQL.

Мова SQL - мова програмування, призначена для управління даними в системах керування базами даних. Дана мова містить ключові слова, які слугають для визначення дій над даними. Наприклад, слово «select» використовується для вибірки даних з таблиць, «insert» - для додавання даних до таблиць, «update» - оновлення, «delete» видалення.

Для роботи з БД реалізуємо в рішенні проекту окремий клас, який буде посилати запити в базу даних.

Модель даних (Data model) — абстрактне представлення реального світу, що відображає тільки ті об'єкти, що безпосередньо стосуються програми. Це, як правило, визначає специфічну групу об'єктів, їх атрибутивне значення і відношення між ними. Використовується механізм представлення і організації просторової моделі даних, або растрової моделі даних. Вона не залежить від комп'ютерної системи і пов'язана тільки зі структурою даних.

Для зручного, гнучкого та інтерактивного написання модулів до сайту використовується MVC паттерн.

Модель–вигляд–контролер (або Модель–представлення–контролер,  Model-view-controller, MVC) — архітектурний шаблон, який використовується під час проектування та розробки програмного забезпечення.

Цей шаблон передбачає поділ системи на три взаємопов'язані частини: модель даних, вигляд (інтерфейс користувача) та модуль керування. Застосовується для відокремлення даних (моделі) від інтерфейсу користувача (вигляду) так, щоб зміни інтерфейсу користувача мінімально впливали на роботу з даними, а зміни в моделі даних могли здійснюватися без змін інтерфейсу користувача.

Мета шаблону — гнучкий дизайн програмного забезпечення, який повинен полегшувати подальші зміни чи розширення програм, а також надавати можливість повторного використання окремих компонентів програми. Крім того використання цього шаблону у великих системах сприяє впорядкованості їхньої структури і робить їх більш зрозумілими за рахунок зменшення складності.

### 1.2 Аналіз існуючого програмного забезпечення за тематикою курсової роботи

Зайшовши в пошукову систему , ви знайдете багато веб-додатків любителів кіно.

Першим ми розглянемо сайт https://www.posmotrionline.ru/. Він є дуже популярним, в пошуковій системи його посилання випадає одним з перших. Інтерфейс простий, він поділений за категоріями (рис. 1.1).

Сам сайт не є адаптивним (рис. 1.2), навігація також немає. Це є не дуже зручним, особливо для мобільних додатків. Сайт потребує читабельного формату та меню бургер. Кольорова гамма сайту проста в темних тонах.

На даному сайті замоло інформації про фільми(рис. 1.3).

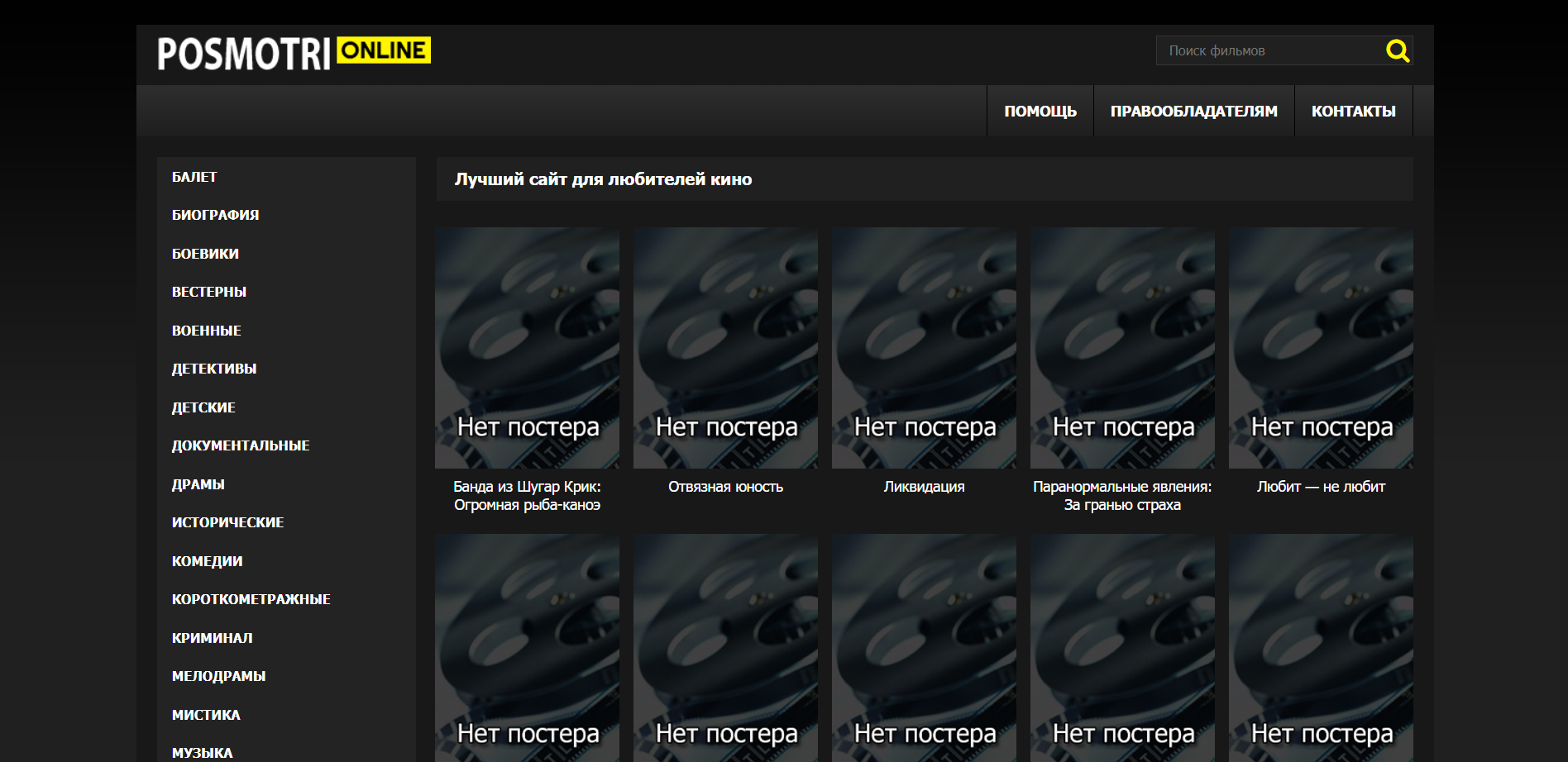


Рисунок 1.1 – Інтерфейс сайту

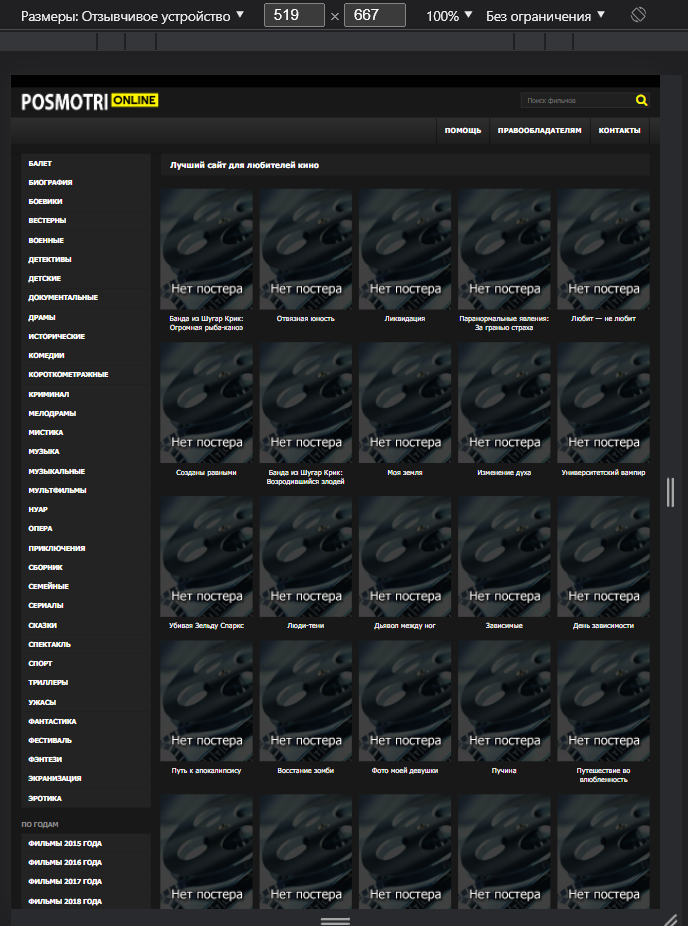


Рисунок 1.2 – Адаптивність

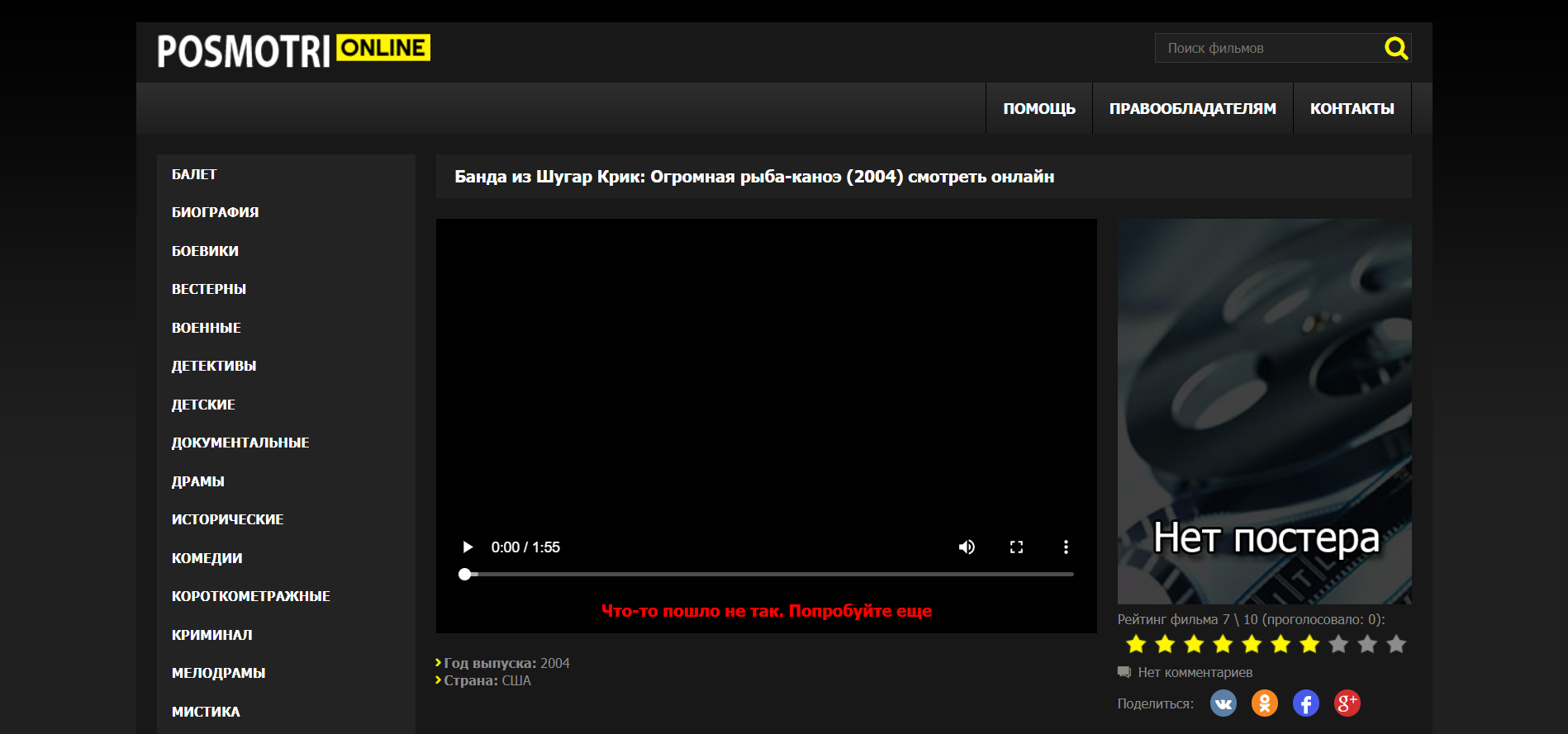


Рисунок 1.3 – Сторінка з фільмом

Другим, ми розглянемо сайт http://cinema.womancomment.com/(рис. 1.4), він також не є адаптивним, з схожим дизайном. (рис. 1.5).

На даному сайті вже більше інформації про вибране кіно



Рисунок 1.4 – Інтерфейс другого сайту



Рисунок 1.5 – Адаптивність другого сайту



Рисунок 1.6 – Сторінка з фільмом 2 сайту

Таблиця 1.1 – Підсумкова таблиця результатів

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Назва сайту/Критерій | Адаптивність | Читабельність при адаптуванні | Пошук | Сортування кіно | Вдалість кольорової гамми | Зрозумілість інтерфейсу | Сучасність дизайну |
| posmotrionline.ru | - | - | + | + | + | + | + |
| cinema.womancomment.com | - | - | + | + | - | + | - |

В результаті можемо зробити певні підсумки, та не допустити таких самих помилок у себе на сайті.

### 1.3 Технічне завдання на курсову роботу

1. Загальне положення

1.1. Найменування програмного засобу

Повне найменування програмної системи: " Веб-додаток любителів кіно" Коротка назва програмної системи - " Веб-додаток любителів кіно ".

1.2. Призначення розробки та область застосування

"Веб-додаток любителів кіно" призначений для пошуку фільмів, мультфільмів та серіалів, їх сортуванню, перегляду додаткової інформації та збереження собі в профіль.

Також існує система перегляду рекомендованих, нових та фільмів за 2021 рік.

"Веб-додаток любителів кіно" дозволить швидко переглядати інформацію про фільми, та забезпечить керівництву управління через адмін панель фільмами, жанрами та користувачами.

1.4. Найменування розробника та замовника.

Розробник даного продукту - студент групи ІПЗк-21-1 Остапчук Богдан Володимирович (надалі "розробник").

Замовник програмного продукту – кафедра інженерії програмного забезпечення Житомирського державного технологічного університету в межах виконання курсової з дисципліни «Web-технології Ч.2» Чижмотря Олена Геннадіївна, Морозов Андрій Василійович.

2. Підстава для розробки

2.1. Документ на підставі якого ведеться розробка

Робота ведеться на підставі навчального плану за напрямом 121 «Інженерія програмного забезпечення».

3. Вимоги до програми

3.1. Вимоги до функціональних характеристик.

3.1.1. Загальні вимоги

Веб-додаток має забезпечувати:

* можливість дистанційної роботи з робочих станцій локальної та глобальної мережі підприємства;
* постійний доступ користувачів веб-додатку;
* зручний пошук товарів;
* оформлення замовлення;
* організацію управління сайтом;
* можливість доступ до бази даних;

3.1.1.Складвиконуванихфункцій

Розробити Веб-додатоку любителів кіно що підтримує виконання наступних операцій:

1. Пошук фільмів по назві.
2. Зміна інформації про вже існуючий фільми, жанри та користувачі.
3. Додавання фільм

* Назва російською
* Назва англійською
* Рейтинг
* Рік
* Тип
* Жанр
* Країна
* Зображення прев’ю
* Основне зображення
* Посилання на трейлер
* Сюжет
* Мінімальний вік для перегляду
* Новинка (да/ні)
* Рекомендований (да/ні)
* Статус(відображати/не відображати)

1. Додавання жанрів
   * Назва жанру
   * Порядок сортування
   * Статус(відображати/не відображати)
2. Додавання користувачів

* Ім’я
* Email
* Пароль
* Роль

1. Фільтрація фільмів за параметрами

* Тип
* Жанр
* Рейтинг
* Рік
* Країна

1. Перегляд збережених фільмів
2. Перегляд інформації про фільми

3.1.2. Організація вхідних і вихідних даних

Вхідними даними є інформація про фільми (Назва російською, Назва англійською, Рейтинг, Рік, Тип, Id-жанру, Країна, Зображення прев’ю, Основне зображення, Посилання на трейлер, Сюжет, Мінімальний вік для перегляду, Новинка (да/ні), Рекомендований (да/ні), Статус(відображати/не відображати))

Організація вхідних і вихідних даних повинна відповідати інформаційній структурі виконуваних з операцій, вхідним та вихідним паперовим документами.

Введення оперативних даних повинно виконуватися з використанням діалогових екранних форм, побудованих на основі візуальних компонентів. Введення даних виконується на основі затверджених форм документів: анкета, заява, інформаційна довідка та в режимі online оператором зі слів користувача.

3.1.3. Часові характеристики і розмір пам'яті, необхідної для роботи програми.

Час реакції програми на дії користувача (маніпуляції з пристроями введення даних) не повинен перевищувати 0,25 с.

Час виконання команд меню не більше 1 с.

Відображення масивів даних за запитами не більше 3 хвилин.

Доступність БД – 90% цілодобово.

Операції з’єднання з БД не більше 1 хвилини.

Обсяг оперативної пам'яті, необхідний для роботи програми не менше 1Гб.

Дисковій простір, необхідний для збереження програми і файлів даних не більше 300 Мбат для робочої станції та 20 ГБайт..

Інсталяційний пакет програми, що містить у складі БД не повинні перевищувати 100 Мбайт.

3.2. Вимоги до методів рішення і мов програмування

Вибір методів рішення здійснюється розробникам без узгодження з замовником.

3.2.1. Вимоги до системи програмних засобів.

OpenServer, PHP 5, MySQL, HTML 5, CSS 3, JavaScript:

Вимоги до програмного забезпечення робочої станції:

PHP Storm

### Висновки з першого розділу

В ході виконання першого розділу було праналізовано ПЗ схожого функціоналу та повністю сформовано уявлення про сайт, що розробляється. Отримано теоретичний фундамент для подальшого написання програмного коду, використовуючи ООП, базу даних та MVC паттерн.

Далі було визначено необхідні інструменти для реалізації сайту, а також інтерфейсу та функціоналу сайту, що розробляється.

## РОЗДІЛ 2 ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

### 2.1 Загальний алгоритм роботи веб-додатку

На даному етапі КП чітко сформовано задачі, які повинний виконувати розроблений веб-сайт. Тобто визначено інтерфейс та задачі які має виконувати додаток. Розроблено методи вирішення поставлених задач та інтерфейсна частина. Основою для цього буде слугувати попередній розділ. На підставі отриманих даних можна сформувати алгоритм роботи програми, що допоможе реалізувати функціонал та методи виконня поставленої задачі. На рисунку 2.1 показано загальний алгоритм роботи програми.

Так як використовується pattern MVC, загальний алгоритм веб-додатку простий. Він буде використовуватися при кожній дії користувача. Схема алгоритму представлена нижче.

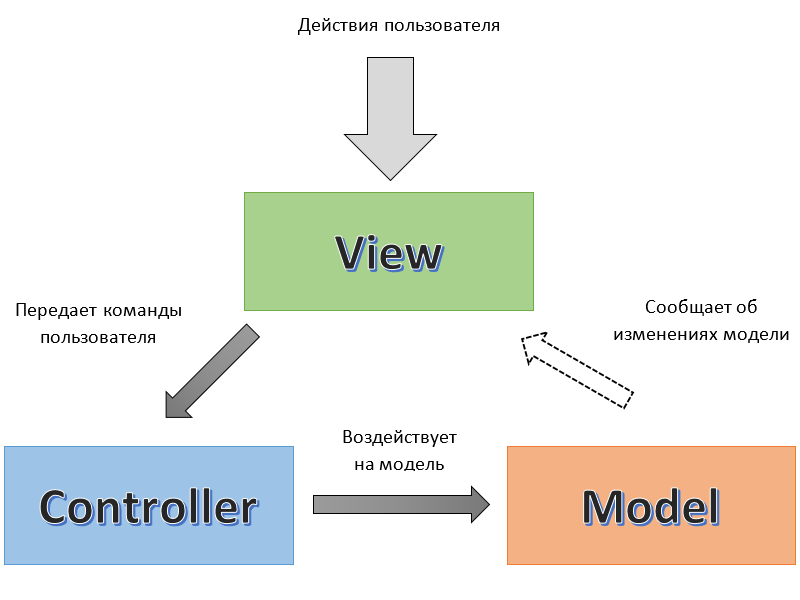


Рисунок 2.1 – Загальний алгоритм роботи веб додатку

Загальний алгоритм роботи додатку:

1. користувач вводить адресу або натискає на посилання
2. браузер реагує на ці дії і звертається до відповідного контролера
3. далі викликається відповідний метод контролера
4. метод контролера звертається до відповідної моделі
5. потім відбувається доступ до бази даних для отримання необхідної інформації
6. далі данні потрапляють назад в модель
7. через модель вони потрапляють в метод контролера
8. метод контролера передає дані в представлення
9. представлення з відповідними даними присилається користувачеві як відповідь на його запит

### 2.2 Розробка функціональних алгоритмів роботи програми

Маючи загальний алгоритм веб-додатку, що ми отримали з попередніх підрозділів, можемо почати безпосередню роботу над проектом.

Найцікавіше в даному КП – це авторизація користувачів. Для авторизації та роботи з акаунтами було створено клас User в якому написані основні функції роботи з користувачами: реєстрація, авторизація, редагування профіля та перевірка коректності введених даних і тд.

На рисунку 2.2 показано регестрацію з усіма можливими помилками.

Приклад реалізації реєстрації:

public static function register($name, $email, $password)  
{  
 // З'єднаняя з БД  
 $db = Db::*getConnection*();  
 // SQL запит  
 $sql = 'INSERT INTO user (name, email, password) '  
 . 'VALUES (:name, :email, :password)';  
 // Хешування паролю  
 $password = md5($password);  
 // Підготовляє запит до виконання та повертає пов'язаний із цим запитом об'єкт  
 $result = $db->prepare($sql);  
 //Прив'язує параметри запиту до змінних  
 $result->bindParam(':name', $name, PDO::*PARAM\_STR*);  
 $result->bindParam(':email', $email, PDO::*PARAM\_STR*);  
 $result->bindParam(':password', $password, PDO::*PARAM\_STR*);  
 return $result->execute();  
}

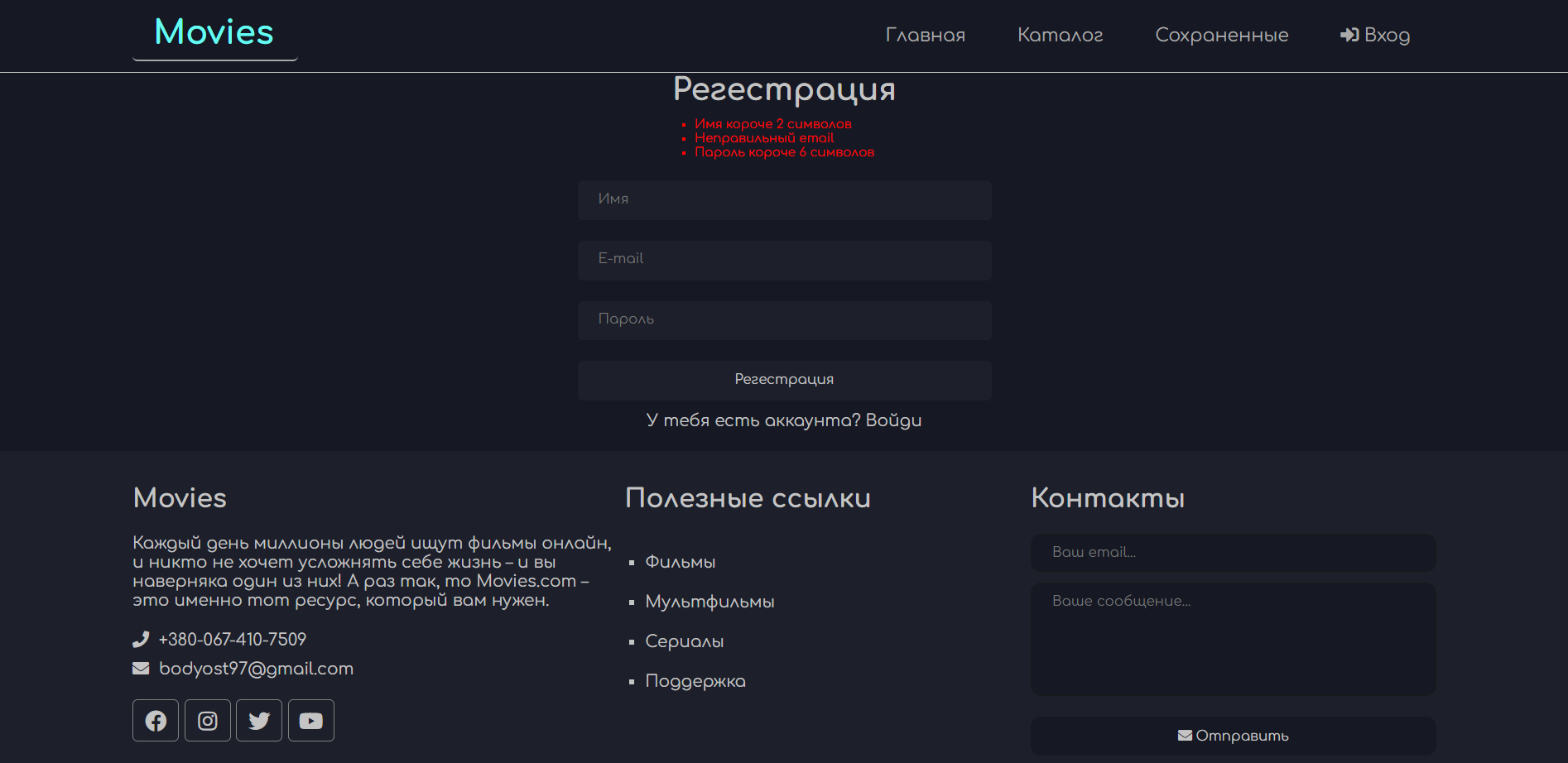


Рисунок 2.2 – Реєстрація користувача

Надалі заходячи у власний кабінет (рис. 2.3) він має можливіть змінювати власні дані та переглядати збереженні фільми. Користувач може бути адміном та просто користувачем. В кабінеті адміністратора (рис. 2.4) є адмін панель. Адмін має змогу додавати, редагувати та видаляти фільми, жанри та користувачів , звичайні користувачі можуть тільки переглядати фільми та збереженні фільми а також редагувати свою інформацію.

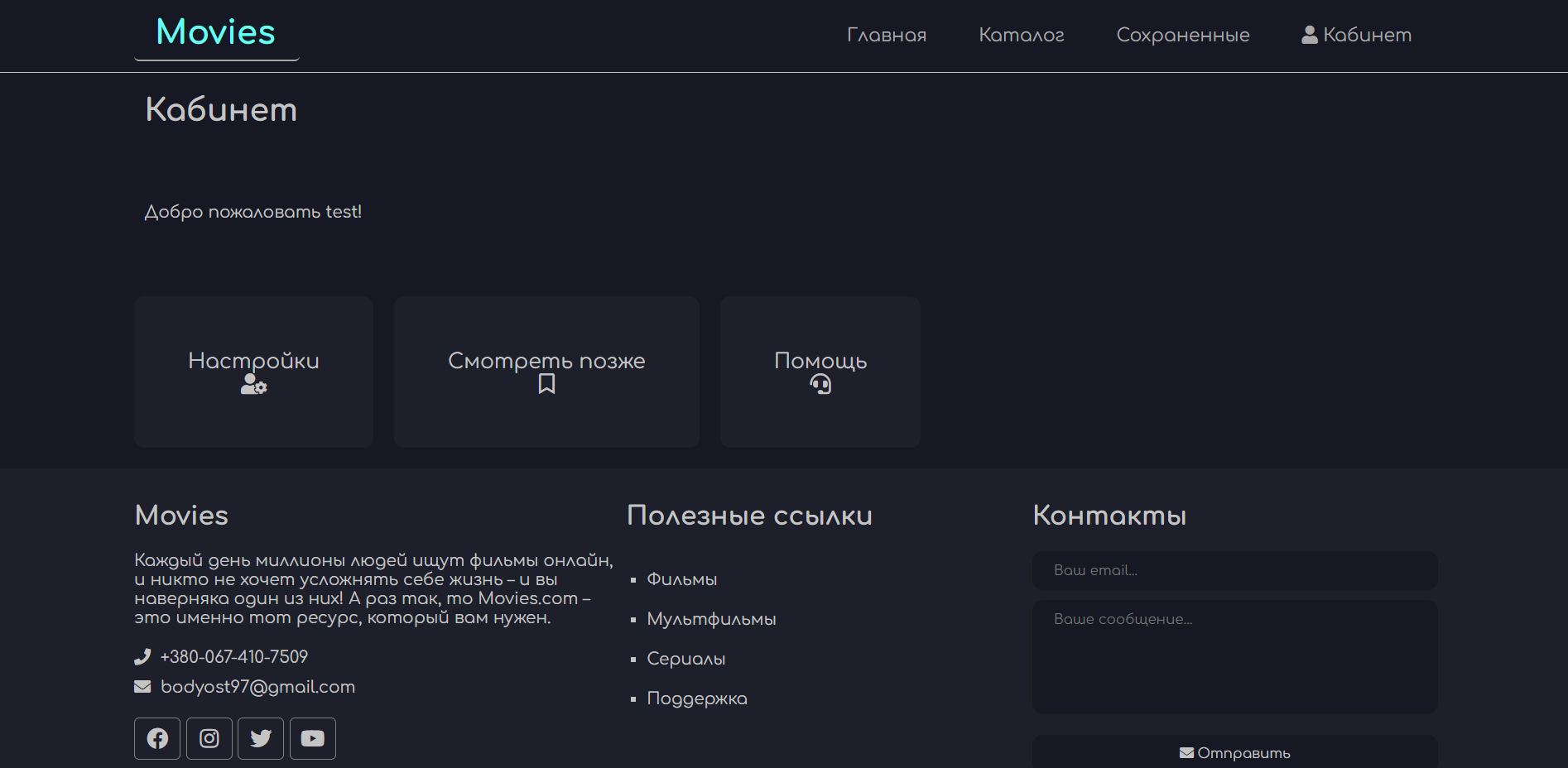


Рисунок 2.3 – Кабінет користувача

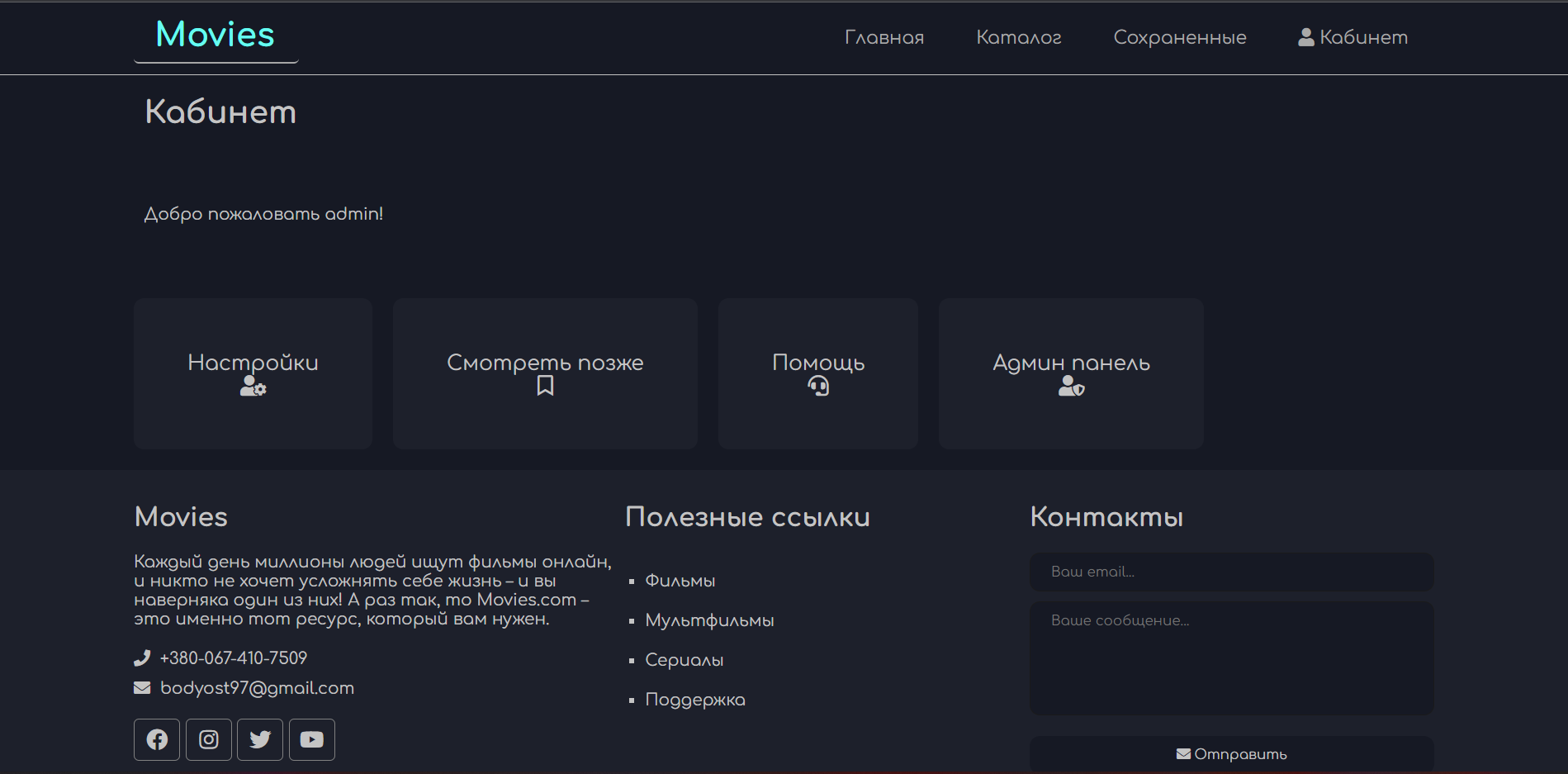


Рисунок 2.4 – Кабінет адміністратора

Всі данні про користувачів зберігаються у спеціально створеній таблиці.

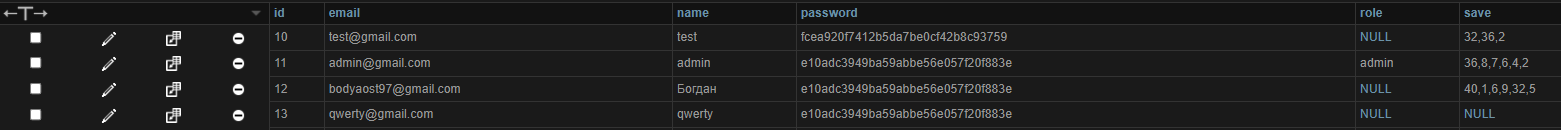


Рисунок 2.5 – Таблиця з користувачами

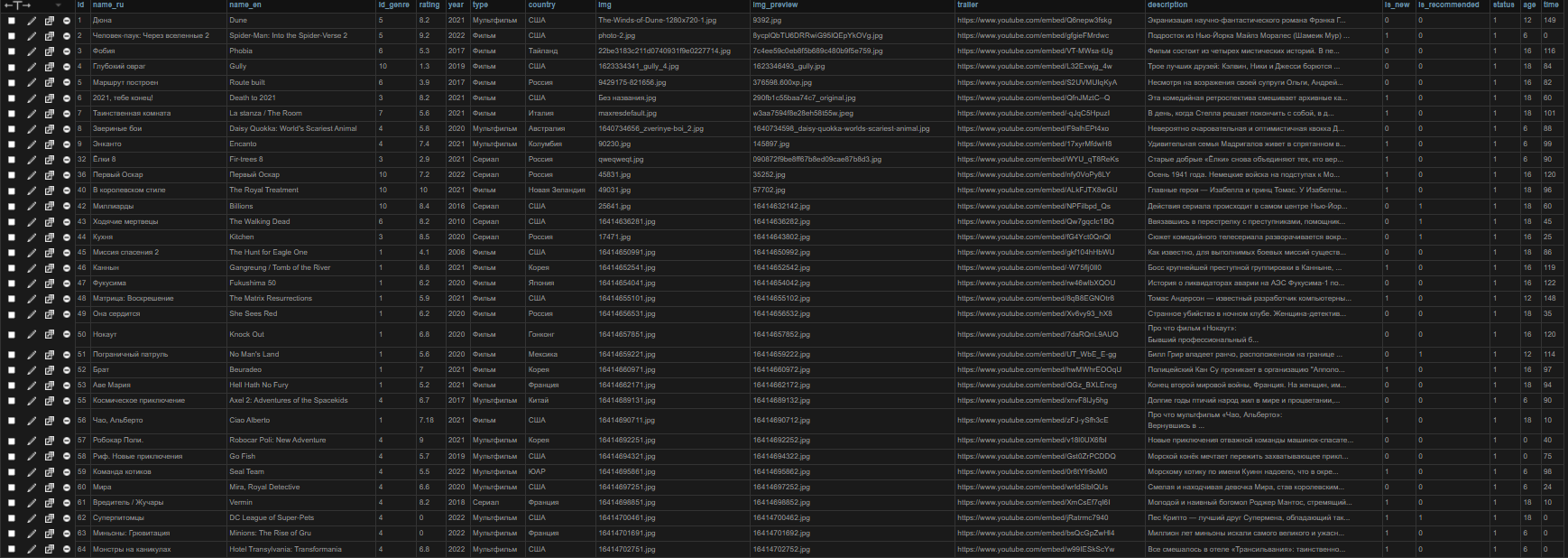


Рисунок 2.6 – Таблиця фільмів

Також створена таблиця в яку записуються дані про жанри.

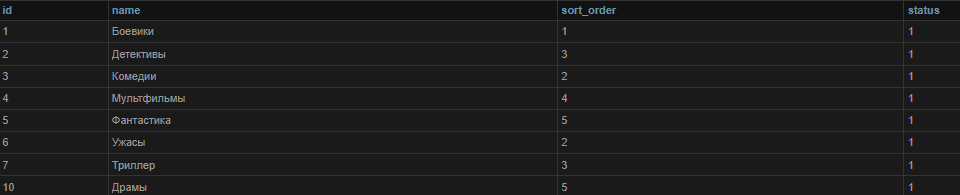


Рисунок 2.7 – Таблиця з жанами

### 2.3 Розробка програмного забезпечення

На даному етапі розробки курсової роботи відбуваєтсья перехід до безпосередньої розробки проектованого ПЗ. На даний момент вирішені всі організаційні та технічні питання з приводу розроблюваного додатку. Чітко сфомовано задачі, які повинний виконувати веб-додаток, вирішені питання по їх реалізації. Тому, притримуючись технічного завдання та вище оговорених положень можна переходити до розробки інтерфейсу додатку.

На даному етапі безпосередньо переходимо до розробки веб-сайту. Для забезпечення безпечної роботи веб-ресурсу налаштовуємо переадресацію сайту до файлу index.php. Переадресація дозволить заблокувати непотрібні файли на сервері і чітко контролювати дані, що будуть відображатися користувачам. Так як ми використовуємо Apache-cервер, то переадресацію реалізуємо за допомогою конфігураційного файлу «.htaccess». Розглянемо конфігурації файлу «.htaccess»:

AddDefaultCharset utf-8  
  
RewriteEngine on  
RewriteBase /  
  
RewriteCond %{REQUEST\_FILENAME} !-f  
RewriteCond %{REQUEST\_FILENAME} !-d  
  
RewriteRule ^(.\*)$ index.php

Надалі ми притримуємось MVC – паттерну для моделювання схеми нашого проекту.

Цей шаблон передбачає поділ системи на три взаємопов'язані частини: модель даних, вигляд (інтерфейс користувача) та модуль керування. Застосовується для відокремлення даних (моделі) від інтерфейсу користувача (вигляду) так, щоб зміни інтерфейсу користувача мінімально впливали на роботу з даними, а зміни в моделі даних могли здійснюватися без змін інтерфейсу користувача.

У рамках архітектурного шаблону модель–вигляд–контролер (MVC) програма поділяється на три окремі, але взаємопов'язані частини з розподілом функцій між компонентами. Модель (Model) відповідає за зберігання даних і забезпечення інтерфейсу до них. Вигляд (View) відповідальний за представлення цих даних користувачеві. Контролер (Controller) керує компонентами, отримує сигнали у вигляді реакції на дії користувача (зміна положення курсора миші, натискання кнопки, ввід даних в текстове поле) і передає дані у модель.

Модель є центральним компонентом шаблону MVC і відображає поведінку застосунку, незалежну від інтерфейсу користувача. Модель стосується прямого керування даними, логікою та правилами застосунку.

Вигляд може являти собою будь-яке представлення інформації, одержуване на виході, наприклад графік чи діаграму. Одночасно можуть співіснувати кілька виглядів (представлень) однієї і тієї ж інформації.

Контролер одержує вхідні дані й перетворює їх на команди для моделі чи вигляду.

Модель інкапсулює ядро даних і основний функціонал їхньої обробки і не залежить від процесу вводу чи виводу даних.

Вигляд може мати декілька взаємопов'язаних областей, наприклад різні таблиці і поля форм, в яких відображаються дані.

У функції контролера входить відстеження визначених подій, що виникають в результаті дій користувача. Контролер дозволяє структурувати код шляхом групування пов'язаних дій в окремий клас.

Таким чином ми створюємо шаблон для зовнішнього вигляду сайту, підставляючи з середину частину, що стосується кожної сторінки окремо.

Наприклад наведемо код основного коду для головної сторінки:

<!-- HEADER -->  
<?php include *ROOT* . '/views/layouts/header.php';?>  
<!-- END HEADER -->  
  
<!-- CAROUSEL -->  
<div class="container carousel-container">  
 <h1>Топ фильмы по мненинию автора</h1>  
 <div class="carousel">  
 <div class="slides">  
 <?php foreach ($isRecommended as $recommended):?>  
 <div class="slider-item">  
 <img src="/template/img/films\_img/<?php echo $recommended['img'];?>" alt="photo-1" />  
 <h5><a href="/film/<?php echo $recommended['id'];?>">«<?php echo $recommended['name\_ru'];?>» <?php echo $recommended['year'];?> г.</a></h5>  
 </div>  
 <?php endforeach;?>  
 </div>  
 <div class="btn-block">  
 <div id="left-button">&#10094;</div>  
 <div id="right-button">&#10095;</div>  
 </div>  
 <div class="dots-block">  
 <?php foreach ($isRecommended as $recommended):?>  
 <div class="dots-item "></div>  
 <?php endforeach;?>  
  
 </div>  
 </div>  
</div>  
<!-- END CAROUSEL -->  
  
<!-- MAIN -->  
<div class="container main">  
 <div class="main-container">  
 <h1>Последние поступления</h1>  
 <div class="films">  
 <?php foreach ($latestFilms as $film):?>  
 <div class="film" id="1">  
 <div class="logo-film">  
 <?php if($film['is\_new'] == 1):?>  
 <p>NEW</p>  
 <?php if($film['rating'] > 8):?>  
 <img class="green" src="<?php echo Film::*getPreviewImg*($film['img\_preview']);?>" alt="photo-1" />  
 <?php elseif($film['rating'] > 6):?>  
 <img class="orange" src="<?php echo Film::*getPreviewImg*($film['img\_preview']);?>" alt="photo-1" />  
 <?php else:?>  
 <img class="red" src="<?php echo Film::*getPreviewImg*($film['img\_preview']);?>" alt="photo-1" />  
 <?php endif;?>  
 <?php else:?>  
 <?php if($film['rating'] > 8):?>  
 <img class="green" src="<?php echo Film::*getPreviewImg*($film['img\_preview']);?>" alt="photo-1" />  
 <?php elseif($film['rating'] > 6):?>  
 <img class="orange" src="<?php echo Film::*getPreviewImg*($film['img\_preview']);?>" alt="photo-1" />  
 <?php else:?>  
 <img class="red" src="<?php echo Film::*getPreviewImg*($film['img\_preview']);?>" alt="photo-1" />  
 <?php endif;?>  
 <?php endif;?>  
  
 </div>  
 <div class="names-film">  
 <a href="/film/<?php echo $film['id'];?>">  
 <p class="names-ru"><?php echo $film['name\_ru']?></p>  
 <p class="type-film"><?php echo $film['type']?> (<?php echo $film['year']?> год)</p>  
 </a>  
 </div>  
 <a href="#" class="add-to-save" data-id="<?php echo $film['id']?>"><i class="fas fa-bookmark"></i> Сохранить</a>  
 </div>  
 <?php endforeach; ?>  
 </div>  
 </div>  
 <div class="sidebar">  
 <div class="search">  
 <h3>Поиск</h3>  
 <form action="/search/" method="post">  
 <input  
 type="text"  
 name="search"  
 class="test-input"  
 placeholder="Поиск..."  
 required  
 />  
 </form>  
 </div>  
 <div class="genres">  
 <h3>Жанры</h3>  
 <ul>  
 <?php foreach ($genres as $genreItem):?>  
 <li>  
 <a href="/genre/<?php echo $genreItem['id'];?>"  
 >  
 <?php echo $genreItem['name'];?>  
 </a>  
 </li>  
 <?php endforeach;?>  
 </ul>  
 </div>  
 </div>  
</div>  
<!-- END MAIN -->  
<!-- MAIN 2 -->  
<div class="container searchFilmsYear">  
 <h1>Фильмы <?php echo $searchFilmsYear[0]['year'];?> года</h1>  
 <div class="rev\_slider">  
 <?php if(!$searchFilmsYear):?>  
 <div class="tear">  
 <p>Фильмы <?php echo $searchFilmsYear[0]['year'];?> еще не загружено.</p>  
 <i class="fas fa-sad-tear"></i>  
 </div>  
 <?php else: ?>  
 <?php foreach ($searchFilmsYear as $film):?>  
 <div class="film swiper-slide" id="1">  
 <div class="logo-film">  
 <?php if($film['is\_new'] == 1):?>  
 <p>NEW</p>  
 <?php if($film['rating'] > 8):?>  
 <img class="green" src="<?php echo Film::*getPreviewImg*($film['img\_preview']);?>" alt="photo-1" />  
 <?php elseif($film['rating'] > 6):?>  
 <img class="orange" src="<?php echo Film::*getPreviewImg*($film['img\_preview']);?>" alt="photo-1" />  
 <?php else:?>  
 <img class="red" src="<?php echo Film::*getPreviewImg*($film['img\_preview']);?>" alt="photo-1" />  
 <?php endif;?>  
 <?php else:?>  
 <?php if($film['rating'] > 8):?>  
 <img class="green" src="<?php echo Film::*getPreviewImg*($film['img\_preview']);?>" alt="photo-1" />  
 <?php elseif($film['rating'] > 6):?>  
 <img class="orange" src="<?php echo Film::*getPreviewImg*($film['img\_preview']);?>" alt="photo-1" />  
 <?php else:?>  
 <img class="red" src="<?php echo Film::*getPreviewImg*($film['img\_preview']);?>" alt="photo-1" />  
 <?php endif;?>  
 <?php endif;?>  
 </div>  
 <div class="names-film">  
 <a href="/film/<?php echo $film['id'];?>">  
 <p class="names-ru"><?php echo $film['name\_ru']?></p>  
 <p class="type-film"><?php echo $film['type']?> (<?php echo $film['year']?> год)</p>  
 </a>  
 </div>  
 <a href="#" class="add-to-save" data-id="<?php echo $film['id']?>"><i class="fas fa-bookmark"></i> Сохранить</a>  
 </div>  
 <?php endforeach; ?>  
 <?php endif;?>  
 </div>  
</div>  
<!-- END MAIN 2 -->  
  
<!-- MAIN 3 -->  
<div class="container searchFilmsYear">  
 <h1>Новинки</h1>  
 <div class="rev\_slider">  
 <?php if(!$isNew):?>  
 <div class="tear">  
 <p>Новинки еще не загружено.</p>  
 <i class="fas fa-sad-tear"></i>  
 </div>  
 <?php else: ?>  
 <?php foreach ($isNew as $film):?>  
 <div class="film swiper-slide" id="1">  
 <div class="logo-film">  
 <?php if($film['is\_new'] == 1):?>  
 <p>NEW</p>  
 <?php if($film['rating'] > 8):?>  
 <img class="green" src="<?php echo Film::*getPreviewImg*($film['img\_preview']);?>" alt="photo-1" />  
 <?php elseif($film['rating'] > 6):?>  
 <img class="orange" src="<?php echo Film::*getPreviewImg*($film['img\_preview']);?>" alt="photo-1" />  
 <?php else:?>  
 <img class="red" src="<?php echo Film::*getPreviewImg*($film['img\_preview']);?>" alt="photo-1" />  
 <?php endif;?>  
 <?php else:?>  
 <?php if($film['rating'] > 8):?>  
 <img class="green" src="<?php echo Film::*getPreviewImg*($film['img\_preview']);?>" alt="photo-1" />  
 <?php elseif($film['rating'] > 6):?>  
 <img class="orange" src="<?php echo Film::*getPreviewImg*($film['img\_preview']);?>" alt="photo-1" />  
 <?php else:?>  
 <img class="red" src="<?php echo Film::*getPreviewImg*($film['img\_preview']);?>" alt="photo-1" />  
 <?php endif;?>  
 <?php endif;?>  
 </div>  
 <div class="names-film">  
 <a href="/film/<?php echo $film['id'];?>">  
 <p class="names-ru"><?php echo $film['name\_ru']?></p>  
 <p class="type-film"><?php echo $film['type']?> (<?php echo $film['year']?> год)</p>  
 </a>  
 </div>  
 <a href="#" class="add-to-save" data-id="<?php echo $film['id']?>"><i class="fas fa-bookmark"></i> Сохранить</a>  
 </div>  
 <?php endforeach; ?>  
 <?php endif;?>  
 </div>  
</div>  
<!-- END MAIN 3 -->  
  
<!-- FOOTER -->  
<?php include *ROOT* . '/views/layouts/footer.php';?>  
<!-- END FOOTER -->

Таким чином ми можемо кожен раз лише заповнювати необхідний контент, збираючи як пазл кінцевий вигляд кожної сторінки.

Висновки з другого розділу

В даному розділі курсової роботи було проведено розробку загального і функціонального алгоритмів сайту, та описано принцип виконання основних методів роботи. Також представлено деталізовану розробку посторінково.

Розроблено загальний алгоритм функціонування роботи програми. Була розроблена інтерфейсна частина, змодельовані вкладки сайту, та спроектовано програмний код сайту.

В результаті було отримано готовий програмний продукт з функціональністю, який виконує всі поставлені задачі та вимоги.

## РОЗДІЛ 3 ОПИС РОБОТИ З ПРОГРАМНИМ ДОДАТКОМ ТА ЙОГО ТЕСТУВАННЯ

### 3.1 Опис роботи з додатком

На даному етапі курсової роботи вже реалізовано сайт, тому необхідно пояснити його роботу. Розглянемо головну сторінку сайту, куди користувач потрапляє, щойно перейшовши за посиланням (рис. 3.1).

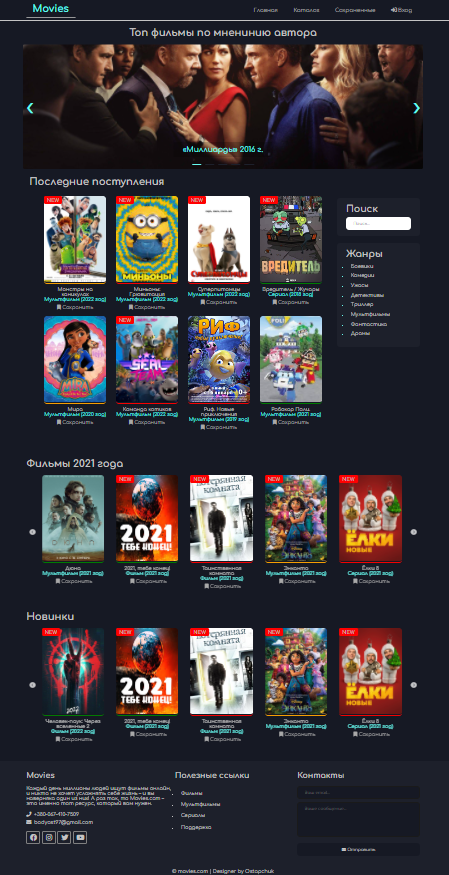


Рисунок 3.1 – Головна сторінка

Користувач бачить вкладки в шапці сайту :

* «Главная» (безпосередньо стартова сторінка)
* «Каталог»
* «Сохраненные»
* «Вход/Кабинет»
* «Поиск»
* «Поиск по жанрам»

Користувач бачить вкладки в подвалі сайту:

* «Фильмы»
* «Мультфильмы»
* «Сериалы»
* «Поддержка»

Далі відвідувач може як зареєструватись так і увійти в власний акаунт та змінити інформацію за потребою.

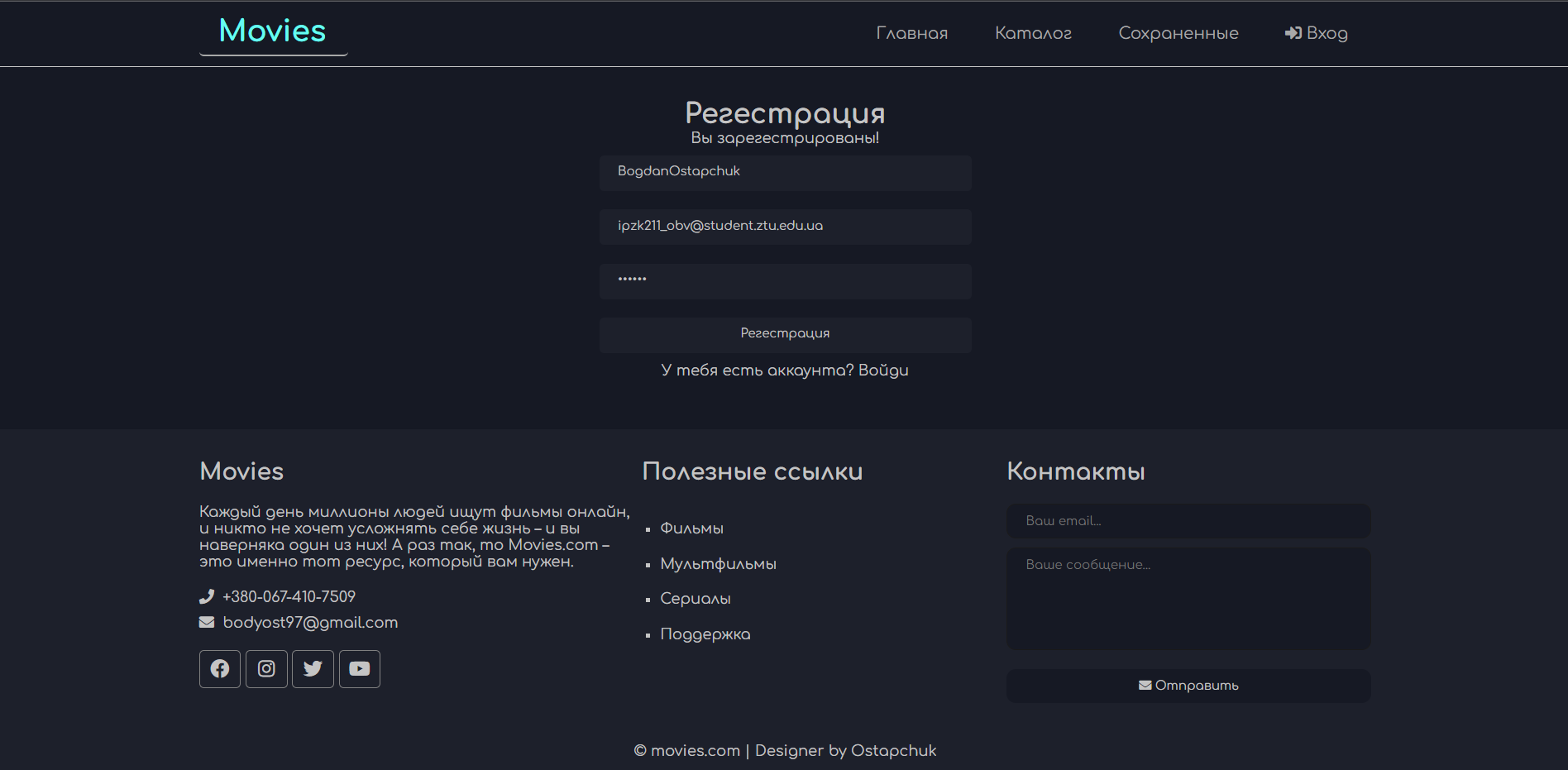


Рисунок 3.2 – Реєстрація

Одразу після реєстрації користувач заходить у власний кабінет, де з ним вітаються по імені вказаному в реєстрації.

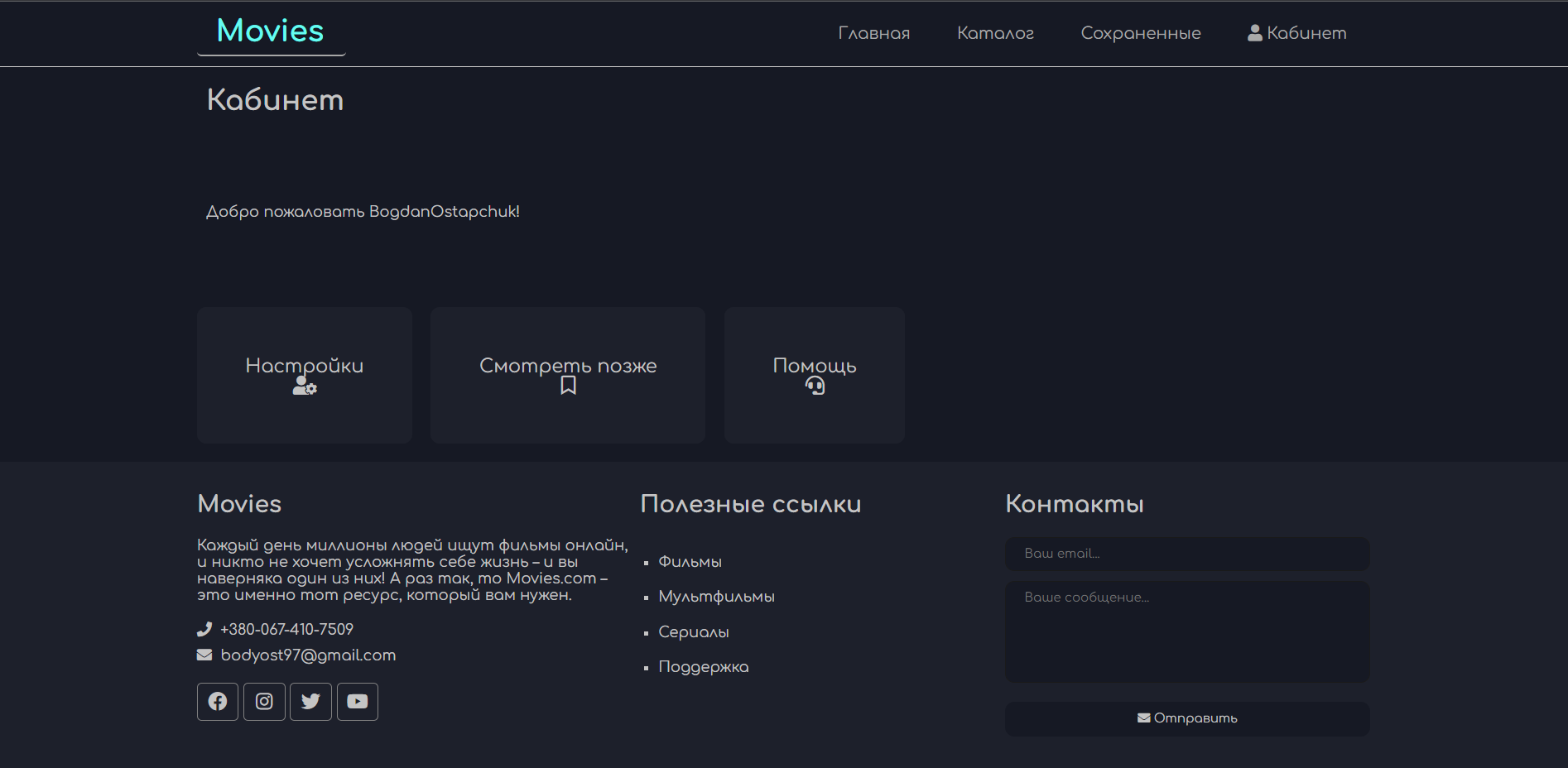


Рисунок 3.3 – Власний кабінет

Якщо до акаунту заходить адміністратор його можливості збільшуються:

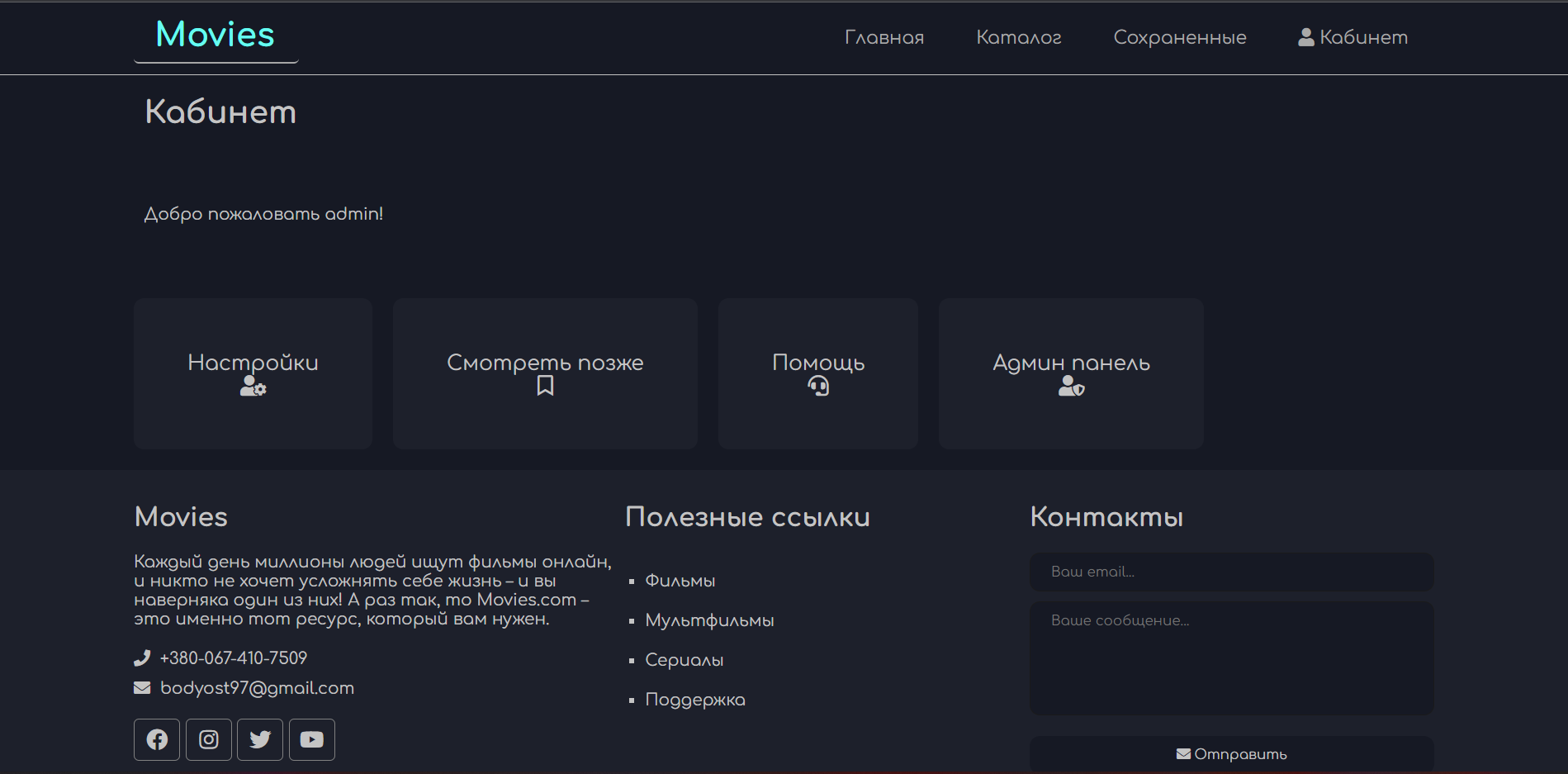


Рисунок 3.4 – Адмін-панель

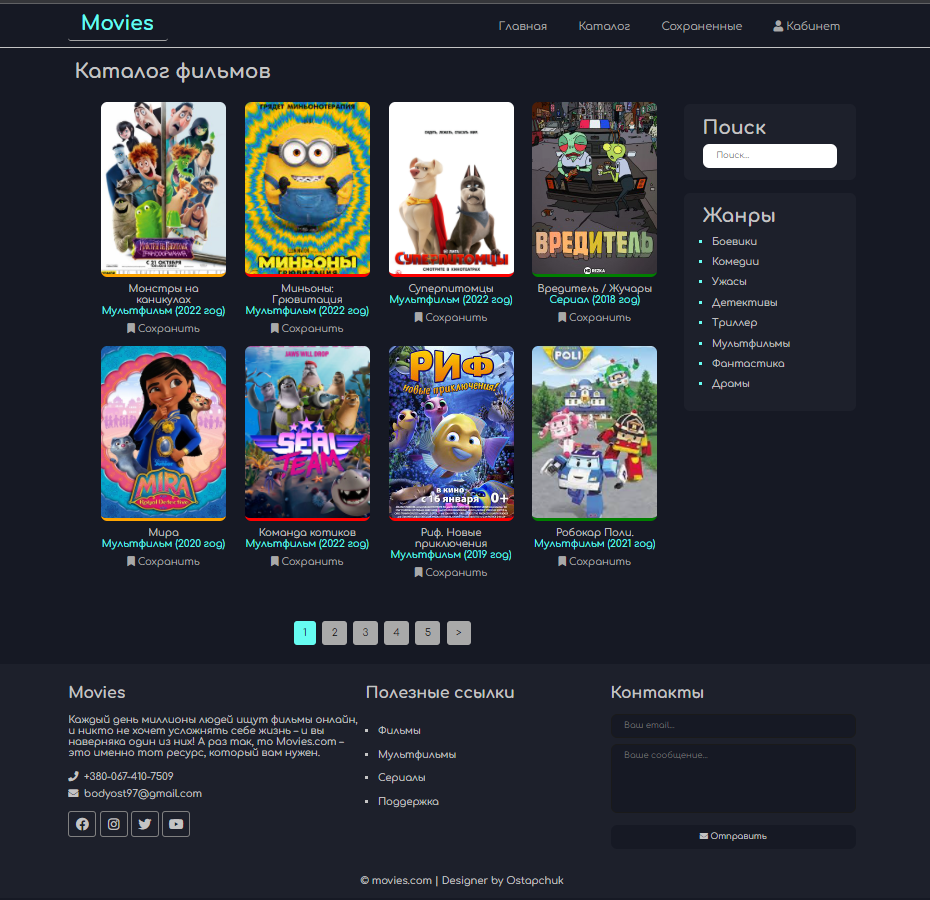


Рисунок 3.5 – Вкладка «Каталог»

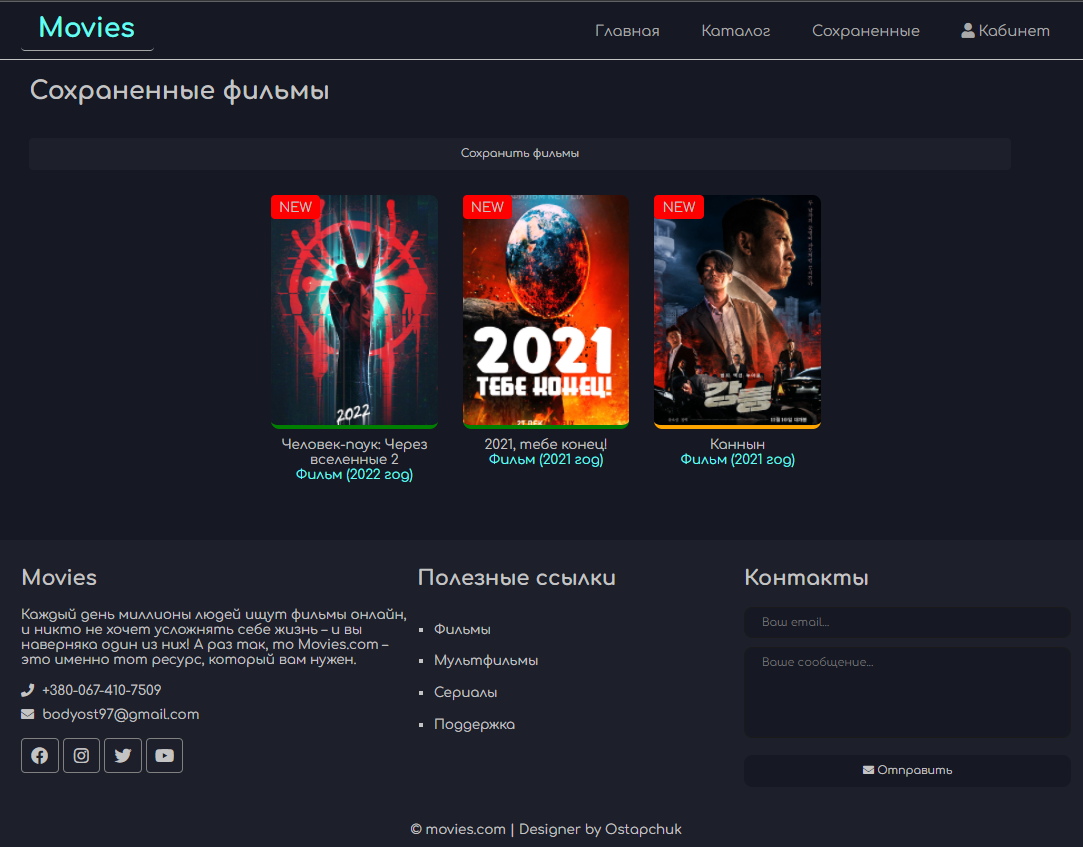


Рисунок 3.6 – Вкладка «Сохраненные»

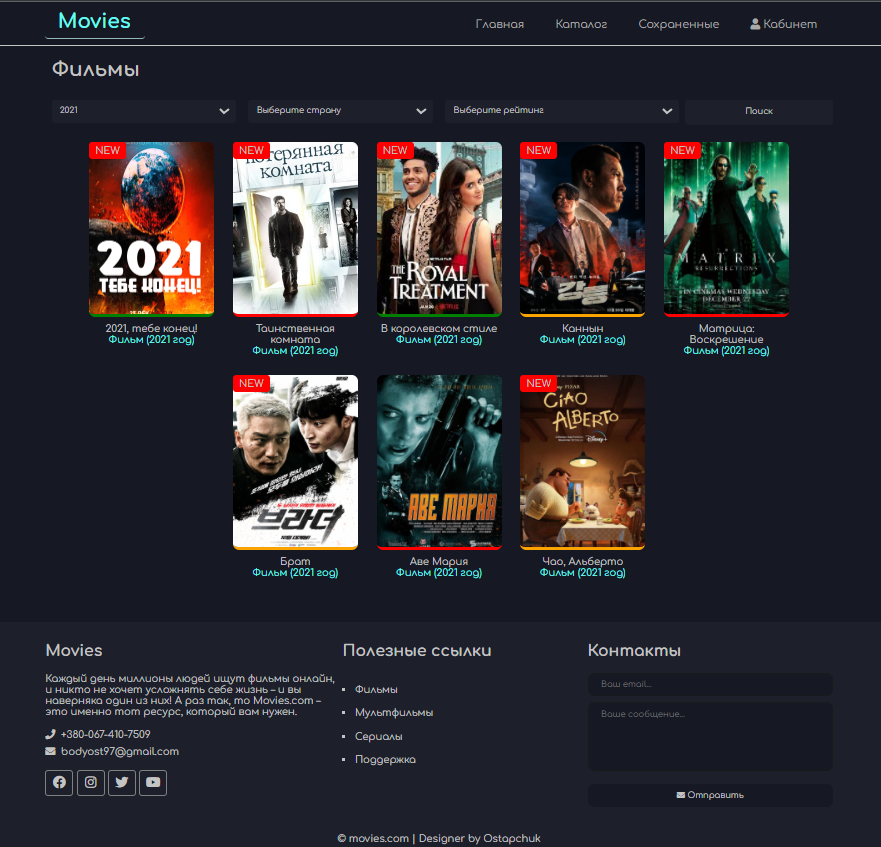


Рисунок 3.7 – Вкладка «Фильмы»

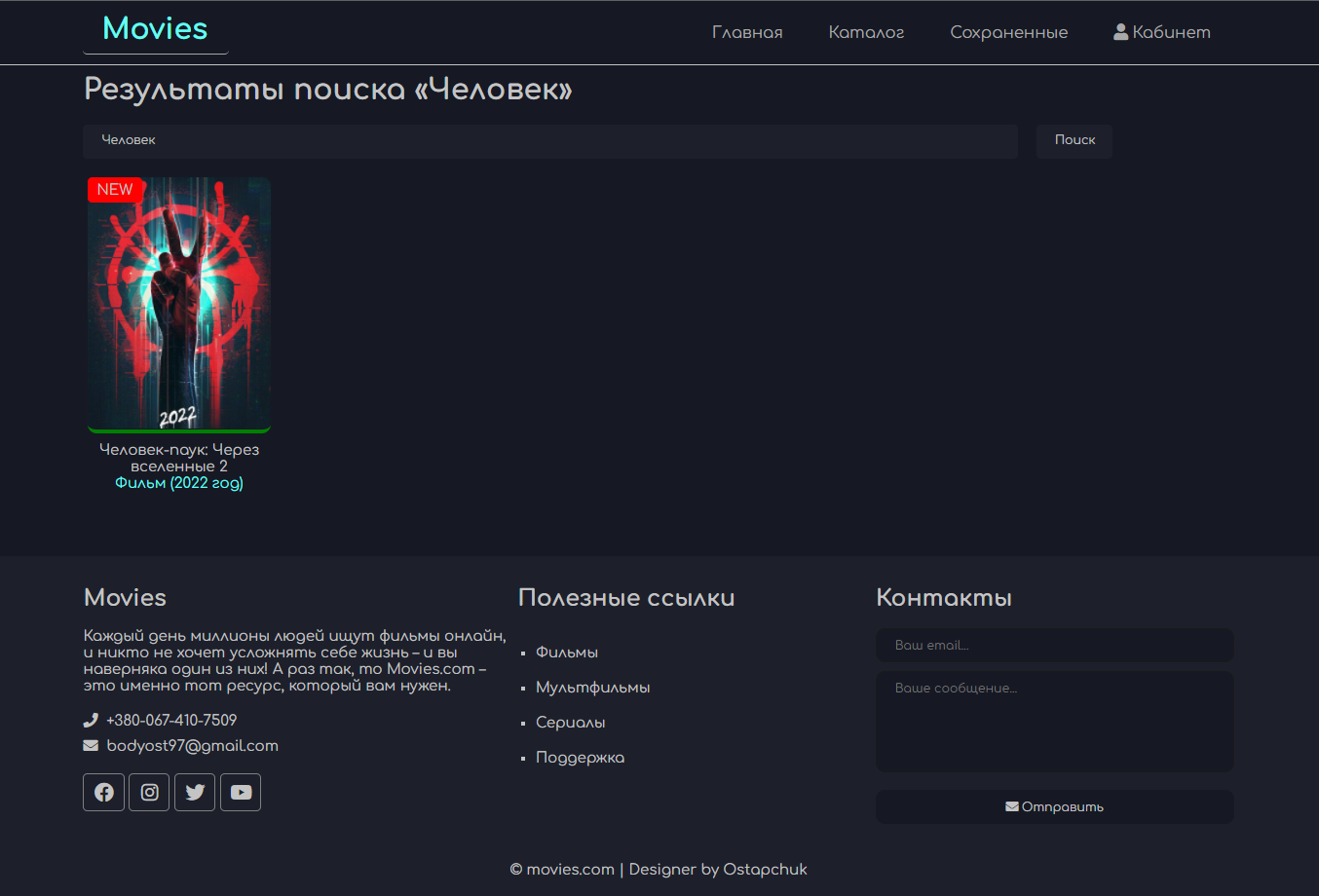


Рисунок 3.8 – Вкладка «Поиск»

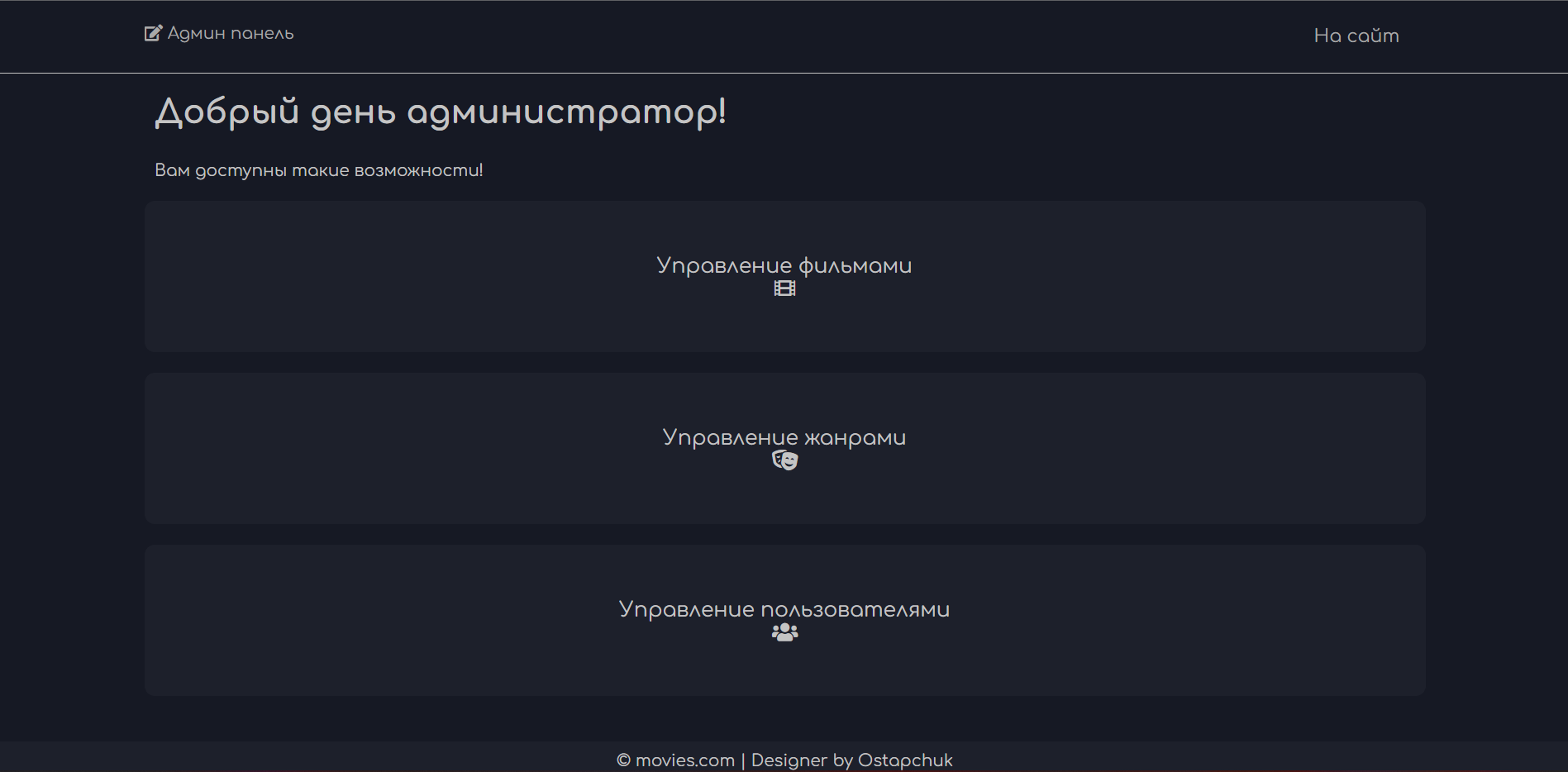


Рисунок 3.9 – Вкладка «Админ панель»

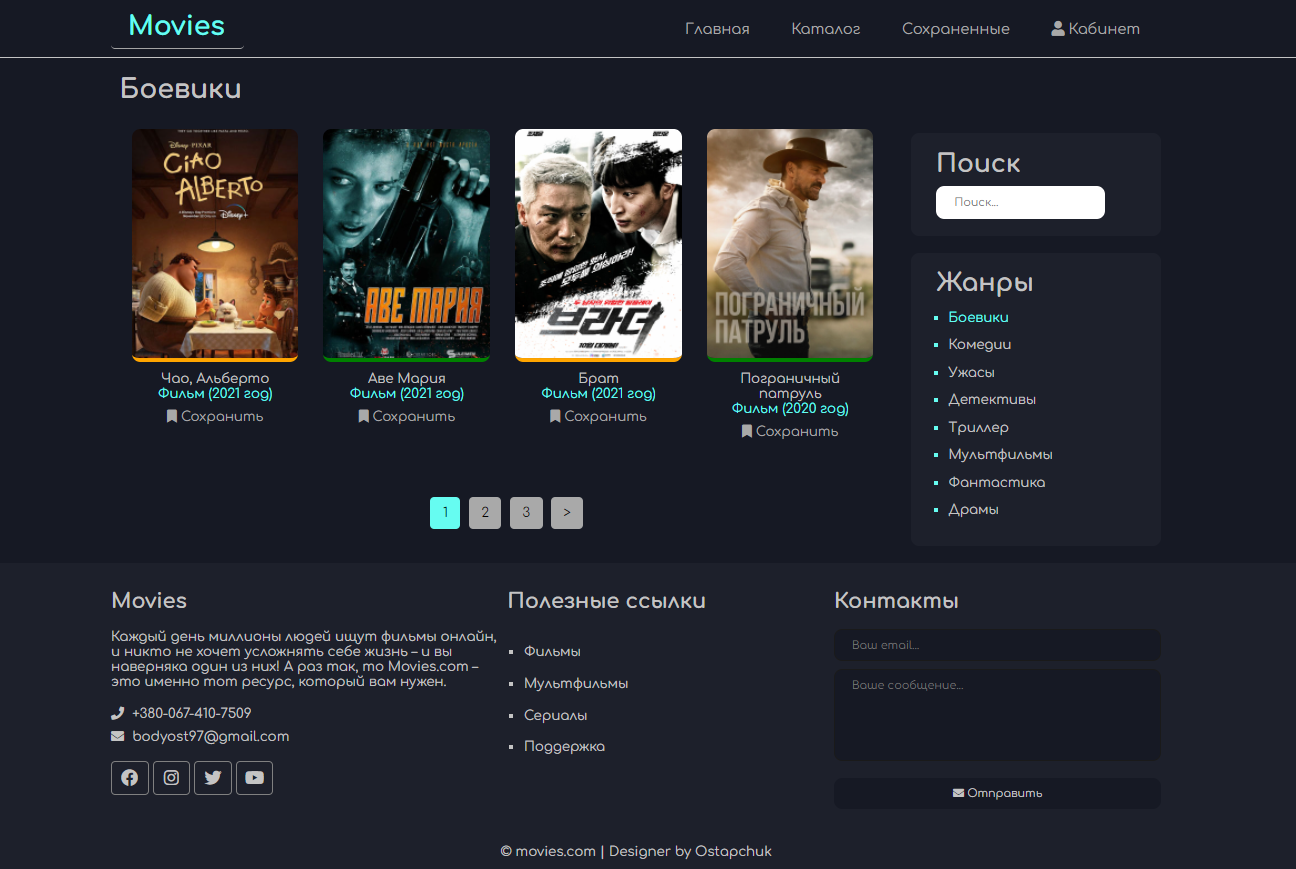


Рисунок 3.10 – Вкладка «Жанры»

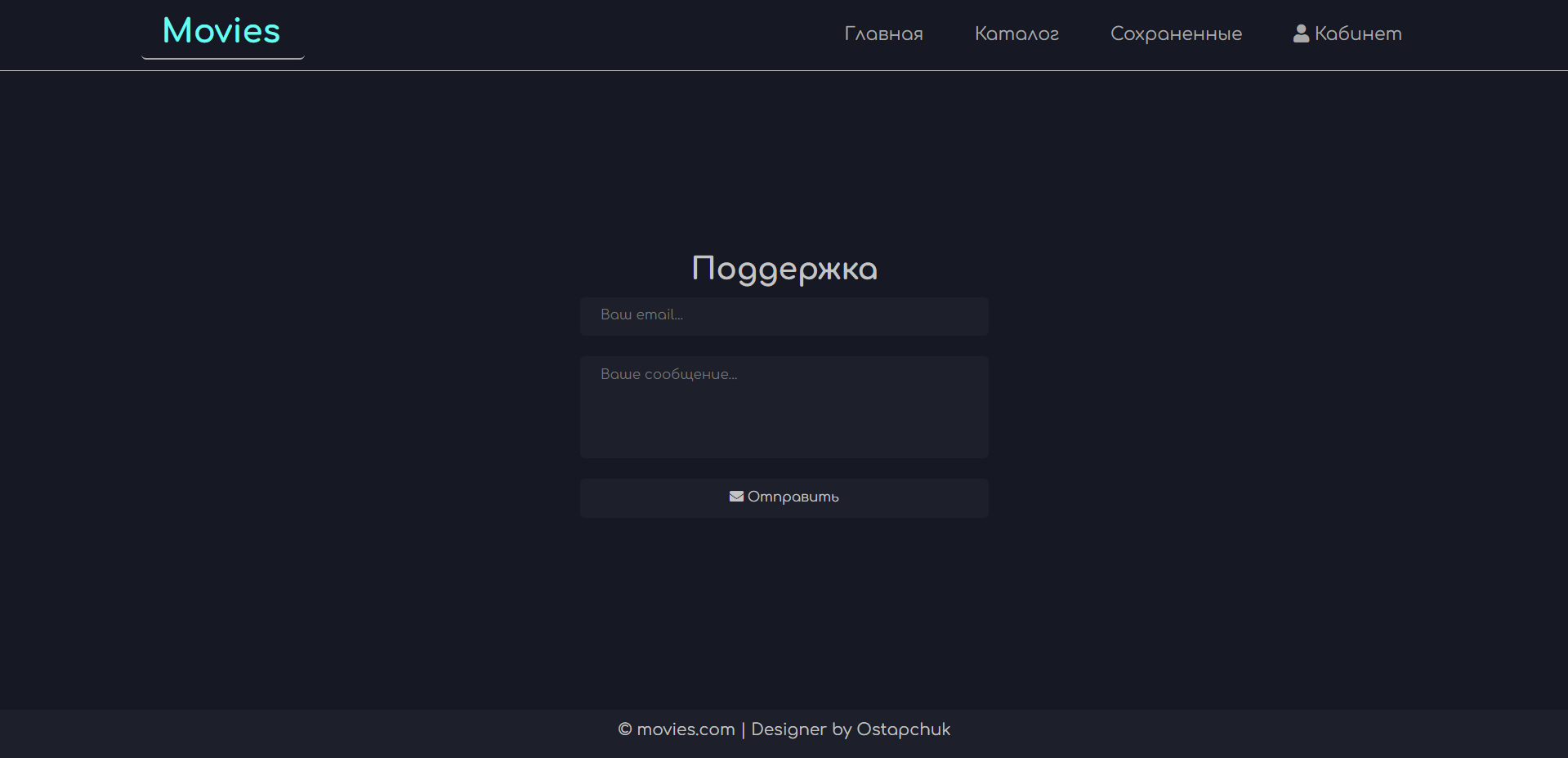


Рисунок 3.11 – Вкладка «Поддержка»

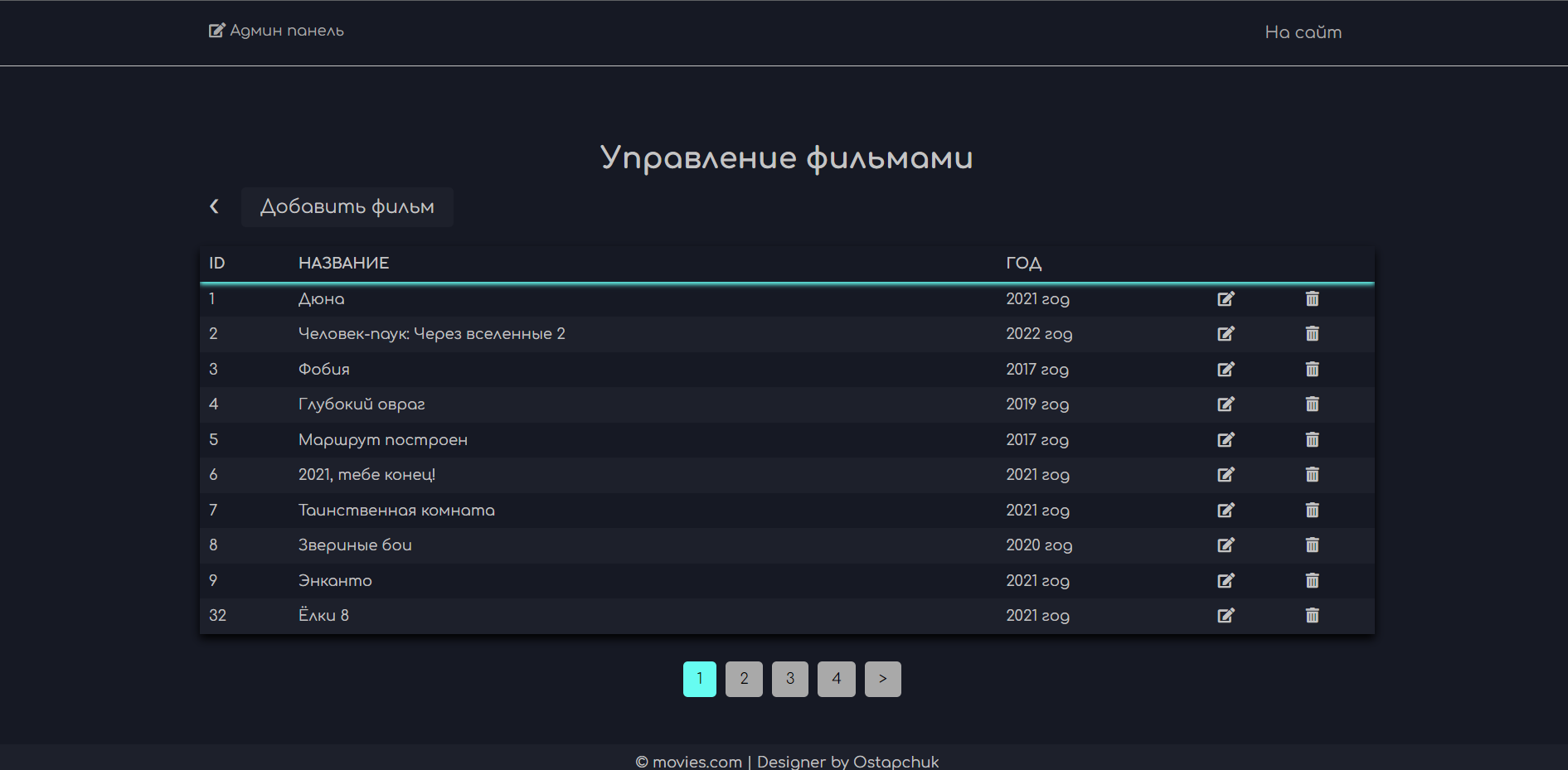


Рисунок 3.12 – Вкладка «Управления»

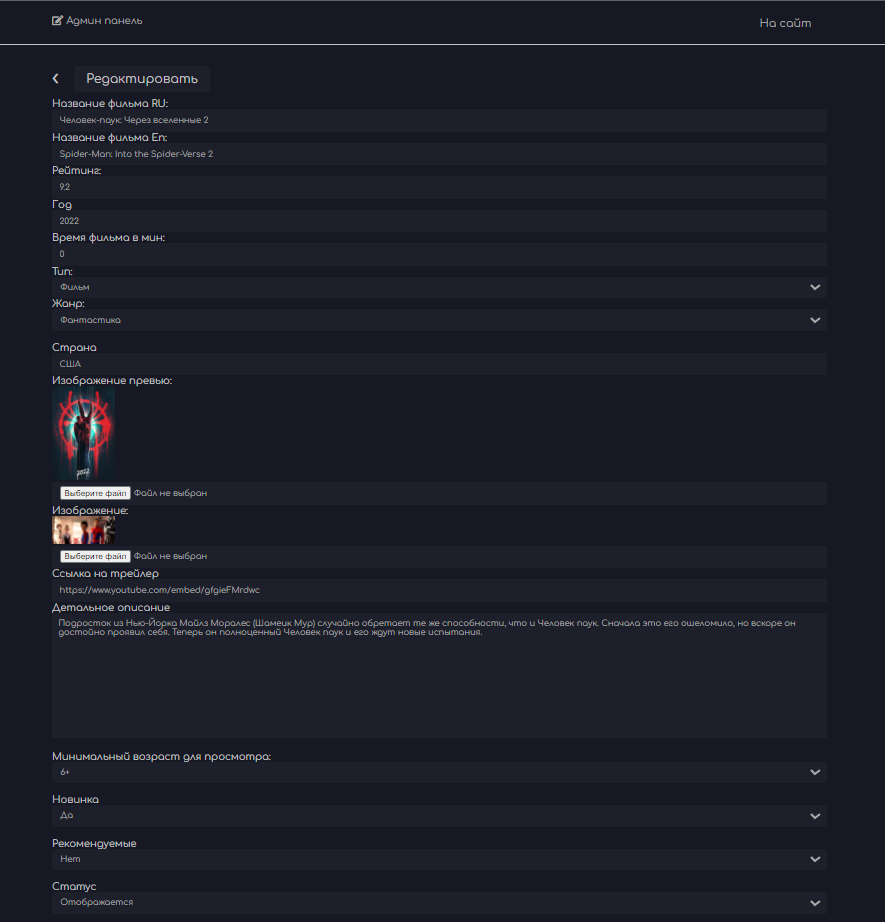


Рисунок 3.13 – Вкладка «Редактирования»

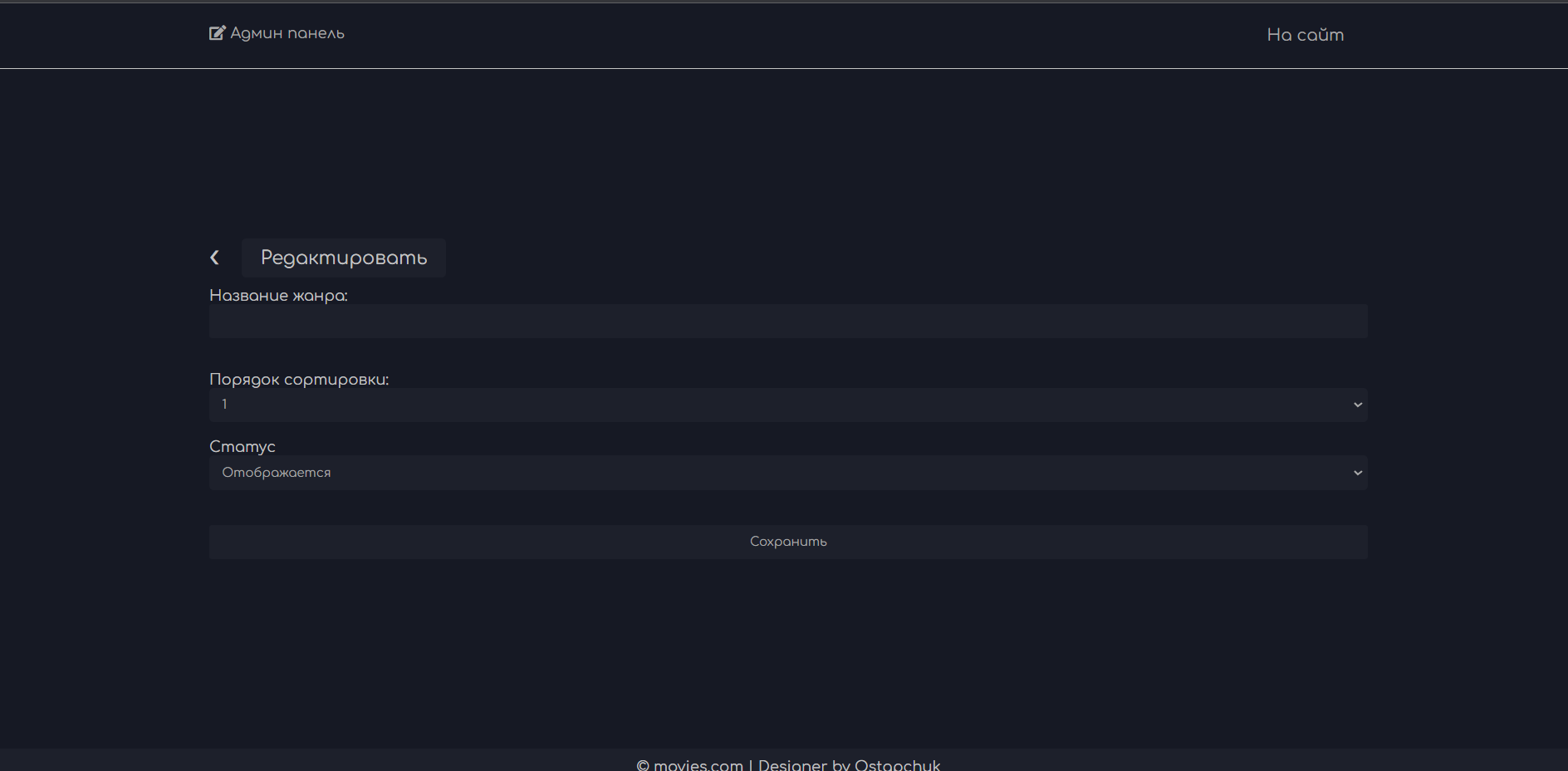


Рисунок 3.14 – Вкладка «Добавление»

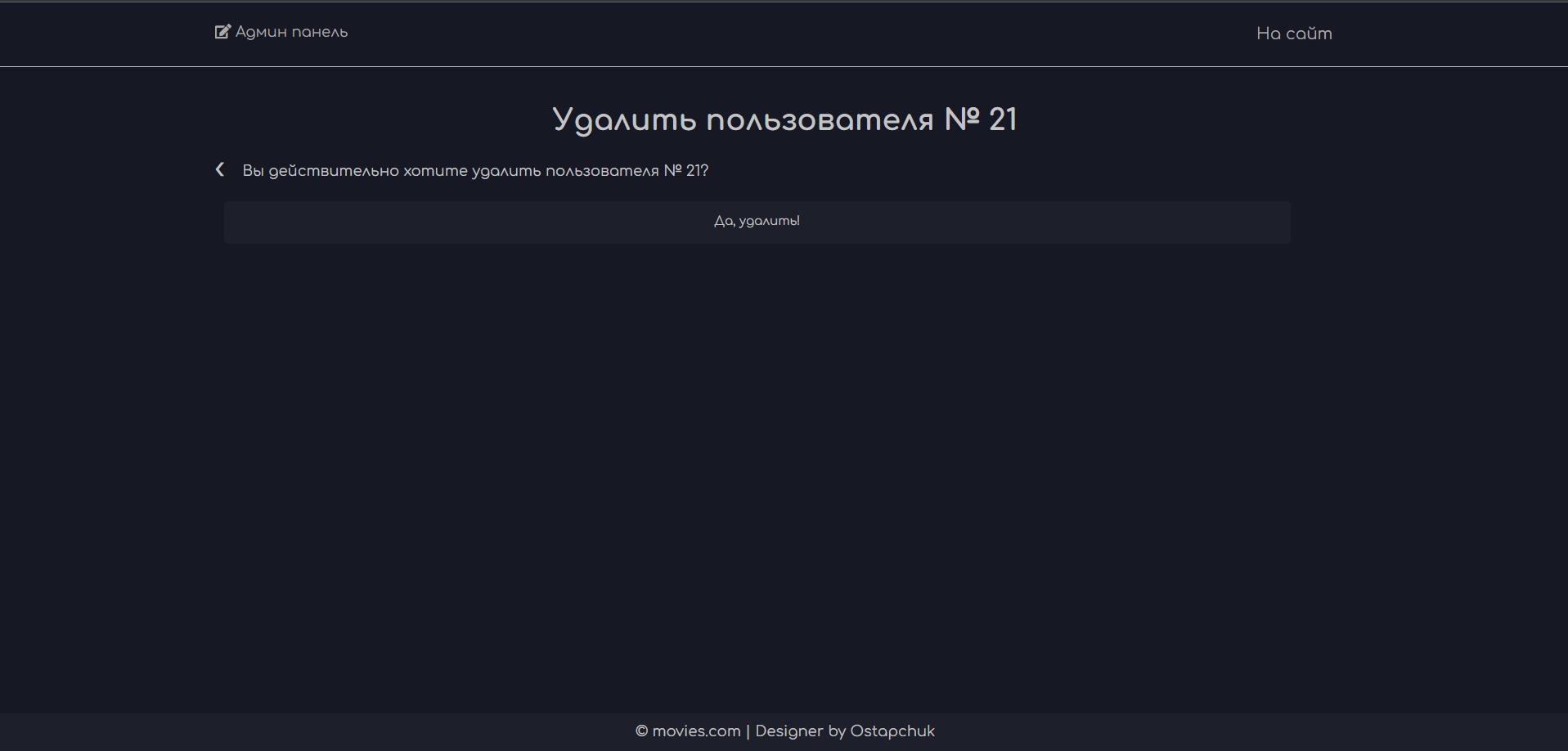


Рисунок 3.15 – Вкладка «Удаление»

### 3.2 Тестування веб-додатку

Переходимо безпосередньо до тестування розробленого сайту. На меті проведення дослідження є використання створеного програмного продукту в реальних умовах та перевірка отриманих результатів роботи.

Кожна сторінка є адаптивною та на кожній є свої перевірки на помилки.

Під час тестування розглянемо тільки сторінку регестрації.

Розглянемо естетичний, але не менш важливий параметр – адаптивність для різних пристроїв (рис. 3.16).

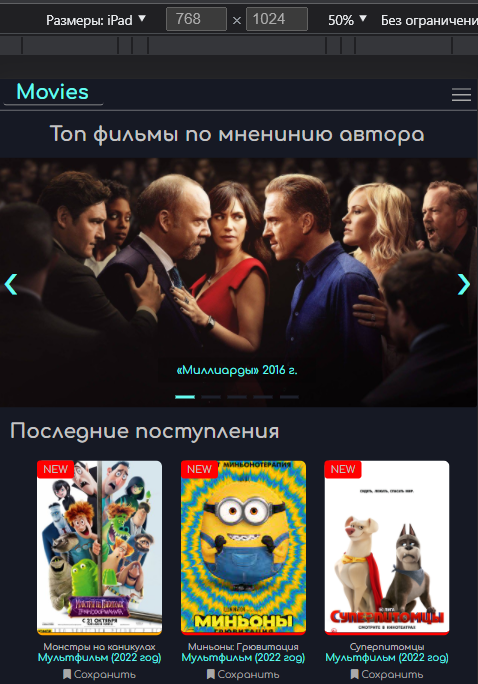


Рисунок 3.16 – Адаптивність сайту

Далі перейдемо до зони користувача при реєстрації, перевіряємо створення в базі даних нового запису (рис. 3.17).

Як бачимо на рисунку 3.18 нового користувача з такими даними успішно створено.

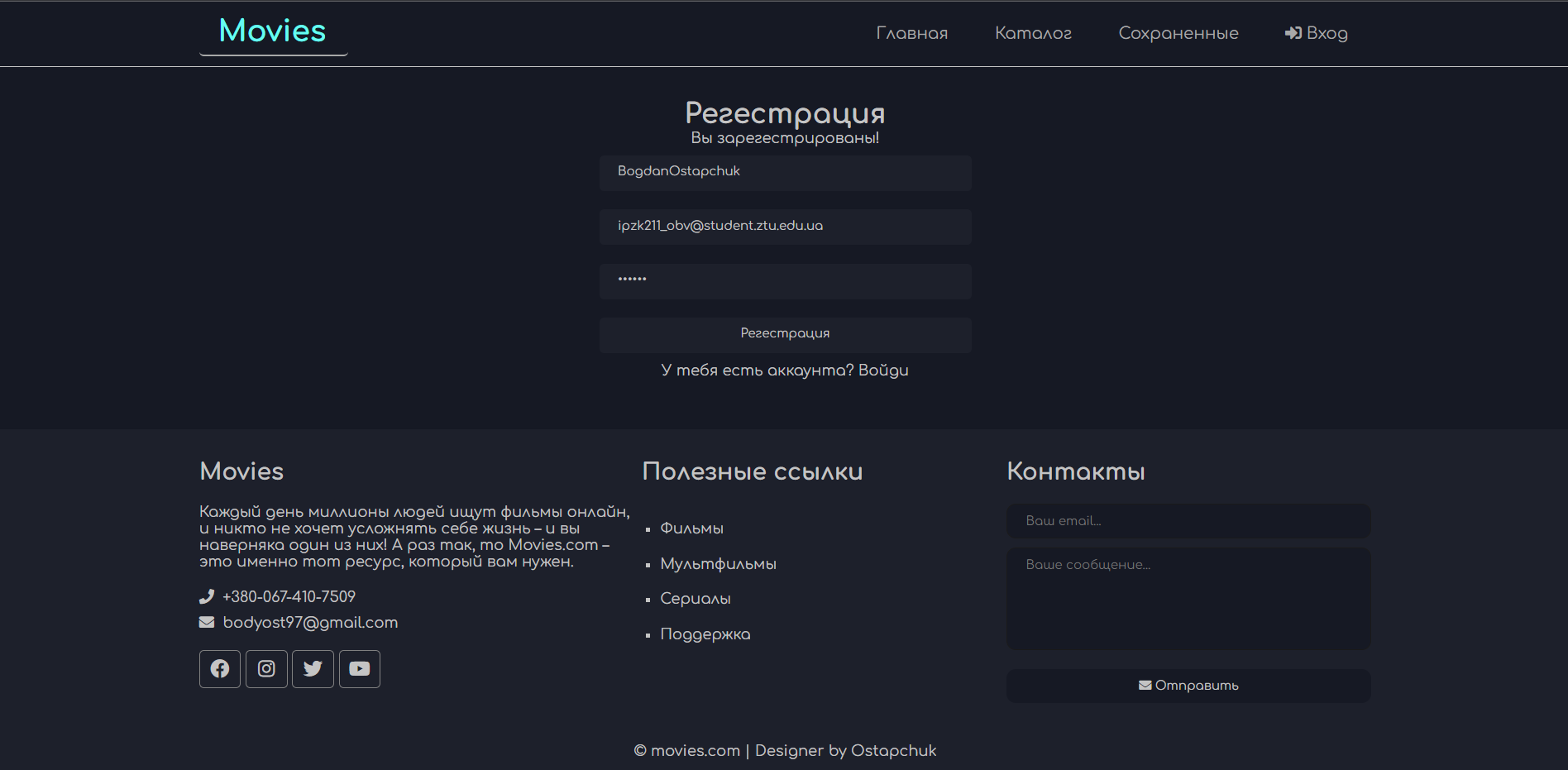


Рисунок 3.17 – Реєстрація



Рисунок 3.18 – Поява відповідного запису в БД

У випадку некоректного вводу інформації для реєстрації є 4 види помилок 2 із яких представлено нижче:

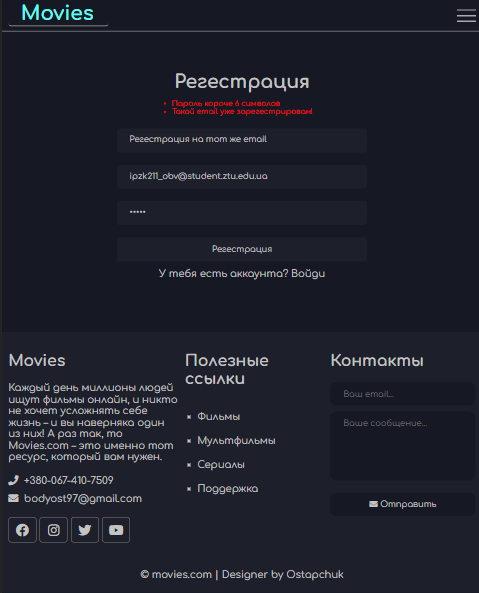


Рисунок 3.19 – Некоректний ввід даних

### Висновки до третього розділу

В даному розділі курсвої роботи було описано методологію користування стовреним сайтом, та проведено тестування створеного додатку.

Опис роботи з сайтом включає в себе текстову та ілюстраційну чатину, які пояснюють принципи роботи з сайтом та опис елементів інтерфейсу сайту. При описі було роз’яснено призначення кожного елемента інтерфейсу у всіх вікнах.

При проведені тестування додаток був запущений на виконная у реальних умовах, при яких показав стабільну і коректну роботу.

# ВИСНОВКИ

У першому розділі було праналізовано сайти схожого функціоналу та повністю сформовано уявлення про сайт, що розробляється. Отримано теоретичний фундамент для подальшого написання програмного коду, використовуючи ООП та базу даних. Далі було визначено необхідні інструменти для реалізації програмного забезпечення, а також інтерфейсу та функціоналу програми, що розробляється.

У другому розділі було проведено розробку загального і функціонального алгоритмів програм, розроблених на основі сформованого уявлення про задачі, які повинен вирішувати даний сайт та функціонал, який він повинен містити.

Розроблено загальний алгоритм функціонування роботи сайту. Була розроблена інтерфейсна частина, змодельовані вкладки сайту, та спроектовано програмний код сайту.

В результаті було отримано готовий програмний продукт з повною функціональністю, який виконує всі поставлені задачі та вимоги.

У третьому розділі курсвої роботи було описано методологію користування створеним сайтом, та проведено тестування створеного сайту.

При проведені тестування сайт був запущений на виконная у реальних умовах, при яких показав стабільну і коректну роботу.

В результаті виконання курсової роботи було розроблено сайт, що повністю відповідає поставленим вимогам. Сайт є простим у користуванні, зі зрозумілим інтерфейсом.

# ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Реализация MVC паттерна на примере создания сайта-визитки на PHP [електронний ресурс] : https://habr.com/ru/post/150267/
2. MVC в PHP [електронний ресурс] : https://anton-pribora.ru/articles/php/mvc/
3. Классы и объекты [електронний ресурс] : https://www.php.net/manual/ru/language.oop5.php
4. Документація бібліотеки JQuery [Електронний ресурс] // JQuery incorp. – 2016. – Режим доступ [електронний ресурс] : <https://jquery-docs.ru>
5. Выразительный JavaScript [електронний ресурс] - 2017. Режим доступу: https://legacy.gitbook.com/book/karmazzin/eloquentjavascript\_ru/details
6. Основы наследования [електронний ресурс] - 2015. Режим доступу: [https://professorweb.ru/my/ level7/7\_1.php](https://professorweb.ru/my/csharp/charp_theory/level7/7_1.php)
7. Плюси і мінуси обєктно-орієнтованого програмування [електронний ресурс] - 2015. Режим доступу: <http://damp.biz/plyusi-i-minusi-obyektno-oriyentovanogo-programuvannya/>
8. SQL [електронний ресурс] - 2016. Режим доступу: <https://uk.wikibooks.org/wiki/SQL>
9. Создание SQL запросов [електронний ресурс] - 2016. Режим доступу: <https://www.politerm.com/zuludoc/spatial_query_sql.htm#_sql_overview>
10. Flex [електронний ресурс]: https://developer.mozilla.org/ru/docs/Web/CSS/flex
11. Объектно-ориентированный PHP с классами и объектами [електронний ресурс]: <https://code.tutsplus.com/ru/tutorials/basics-of-object-oriented-programming-in-php--cms-31910>
12. PHP Шпаргалка по ООП [електронний ресурс]: https://prowebmastering.ru/php-shpargalka-po-oop-osnovy.html

# ДОДАТКИ

Додаток А

Програмний код

.htaccess:

AddDefaultCharset utf-8  
  
RewriteEngine on  
RewriteBase /  
  
RewriteCond %{REQUEST\_FILENAME} !-f  
RewriteCond %{REQUEST\_FILENAME} !-d  
  
RewriteRule ^(.\*)$ index.php

Головний index.php:

<?php  
  
// FRONT CONTROLLER  
  
// Общие настройки  
ini\_set('display\_errors',1);  
error\_reporting(*E\_ALL*);  
  
session\_start();  
  
// Подключение файлов системы  
define('ROOT', dirname(*\_\_FILE\_\_*));  
require\_once(*ROOT*.'/components/Autoload.php');  
  
// Вызов Router  
$router = new Router();  
$router->run();

Декілька функцій з классу User:

*/\*\*  
 \* Отримання фільмів на вибраній строрінці пагінації  
 \*/*public static function getUserListWithPage($page)  
{  
 $page = intval($page);  
 $offset = ($page - 1) \* self::*SHOW\_BY\_DEFAULT\_USER*;  
  
 $db = Db::*getConnection*();  
  
 $sql ="SELECT id, email, role FROM user LIMIT "  
 . self::*SHOW\_BY\_DEFAULT\_USER* . " OFFSET ". $offset;  
 $result = $db->query($sql);  
 $i = 0;  
 while ($row = $result->fetch()){  
 $filmsList[$i]['id'] = $row['id'];  
 $filmsList[$i]['email'] = $row['email'];  
 $filmsList[$i]['role'] = $row['role'];  
 $i++;  
 }  
 return $filmsList;  
}

*/\*\*  
 \* Перевірка на валідацію Email  
 \*/*public static function checkEmail($email)  
{  
 if(filter\_var($email, *FILTER\_VALIDATE\_EMAIL*)){  
 return true;  
 }  
 return false;  
}  
*/\*\*  
 \* Перевірка на існування Email в БД  
 \*/*public static function checkEmailExists($email){  
 $db = Db::*getConnection*();  
 $sql = 'SELECT COUNT(\*) FROM user WHERE email = :email';  
  
 $result = $db->prepare($sql);  
 $result->bindParam(':email', $email, PDO::*PARAM\_STR*);  
 $result->execute();  
  
 if ($result->fetchColumn()){  
 return true;  
 }  
 return false;  
  
}

*/\*\*  
 \* Створення нового користувача  
 \*/*public static function createUser($user)  
{  
 $db = Db::*getConnection*();  
  
 $sql = "INSERT INTO `user`(`email`, `name`, `password`, `role`) VALUES (:email,:name,:password,:role)";  
 $result = $db->prepare($sql);  
 $result->bindParam(':email', $user['email'], PDO::*PARAM\_STR*);  
 $result->bindParam(':name', $user['name'], PDO::*PARAM\_STR*);  
 $result->bindParam(':password', $user['password'], PDO::*PARAM\_STR*);  
 $result->bindParam(':role', $user['role'], PDO::*PARAM\_STR*);  
  
 return $result->execute();  
}  
*/\*\*  
 \* Зміна даних користувача  
 \*/*public static function updateUserById($id, $user)  
{  
 $db = Db::*getConnection*();  
 $sql = 'UPDATE `user` SET `email`=:email,`name`=:name,`password`=:password, `role`=:role WHERE id = :id';  
 $result = $db->prepare($sql);  
 $result->bindParam(':email', $user['email'], PDO::*PARAM\_STR*);  
 $result->bindParam(':name', $user['name'], PDO::*PARAM\_STR*);  
 $result->bindParam(':password', $user['password'], PDO::*PARAM\_STR*);  
 $result->bindParam(':role', $user['role'], PDO::*PARAM\_STR*);  
 $result->bindParam(':id', $id, PDO::PARAM\_INT);  
  
 return $result->execute();  
}  
*/\*\*  
 \* Видалення користувача  
 \*/*public static function deleteUserById($id)  
{  
 $db = Db::*getConnection*();  
  
 $sql = 'DELETE FROM user WHERE id= :id';  
 $result = $db->prepare($sql);  
 $result->bindParam(":id", $id, PDO::*PARAM\_STR*);  
 return $result->execute();  
}

Декілька функцій з классу Film:

*/\*\*  
 \* Отримання фільмів з БД  
 \*/*public static function getFilmList()  
{  
 $db = Db::*getConnection*();  
  
 $filmsList = array();  
  
 $result = $db->query("SELECT id, name\_ru, rating, type, year, is\_new, img\_preview FROM film");  
 $i = 0;  
 while ($row = $result->fetch()){  
 $filmsList[$i]['id'] = $row['id'];  
 $filmsList[$i]['name\_ru'] = $row['name\_ru'];  
 $filmsList[$i]['type'] = $row['type'];  
 $filmsList[$i]['rating'] = $row['rating'];  
 $filmsList[$i]['year'] = $row['year'];  
 $filmsList[$i]['is\_new'] = $row['is\_new'];  
 $filmsList[$i]['img\_preview'] = $row['img\_preview'];  
  
 $i++;  
 }  
 return $filmsList;  
}  
*/\*\*  
 \* Отримання фільмів на вибраной строінці пагінації  
 \*/*public static function getFilmListWithPage($page)  
{  
 $page = intval($page);  
 $offset = ($page - 1) \* self::*SHOW\_BY\_DEFAULT\_CATALOG*;  
 $db = Db::*getConnection*();  
  
 $sql ="SELECT id, name\_ru, year FROM film "  
 . "WHERE status = 1 LIMIT "  
 . self::*SHOW\_BY\_DEFAULT\_FILM* . " OFFSET ". $offset;  
 $result = $db->query($sql);  
 $i = 0;  
 while ($row = $result->fetch()){  
 $filmsList[$i]['id'] = $row['id'];  
 $filmsList[$i]['name\_ru'] = $row['name\_ru'];  
 $filmsList[$i]['year'] = $row['year'];  
 $i++;  
 }  
 return $filmsList;  
}

*/\*\*  
 \* Отримання фільма за параметром id  
 \*/*public static function getFilmById($id)  
{  
 $id = intval($id);  
 if ($id){  
 $db = Db::*getConnection*();  
  
 $result = $db->query("SELECT \* FROM film WHERE id=".$id);  
 $result->setFetchMode(PDO::*FETCH\_ASSOC*);  
 return $result->fetch();  
 }  
}

*/\*\*  
 \* Отримання фільма за id категорії  
 \*/*public static function getTotalFilmsInCategory($genreId)  
{  
 $db = Db::*getConnection*();  
  
 $result = $db->query('SELECT count(id) AS count FROM film '  
 . 'WHERE status="1" AND id\_genre ="'.$genreId.'"');  
 $result->setFetchMode(PDO::*FETCH\_ASSOC*);  
 $row = $result->fetch();  
 return $row['count'];  
}  
*/\*\*  
 \* Отримання рекомендованих фільмів   
 \*/*public static function getFilmsIsRecommended()  
{  
 $db = Db::*getConnection*();  
  
 $filmsList = array();  
  
 $result = $db->query("SELECT id, name\_ru, year, img FROM film WHERE `is\_recommended` = '1'");  
 $i = 0;  
 while ($row = $result->fetch()){  
 $filmsList[$i]['id'] = $row['id'];  
 $filmsList[$i]['name\_ru'] = $row['name\_ru'];  
 $filmsList[$i]['year'] = $row['year'];  
 $filmsList[$i]['img'] = $row['img'];  
  
 $i++;  
 }  
 return $filmsList;  
}

*/\*\*  
 \* Створення фільму  
 \*/*public static function createFilm($film, $times1, $times2)  
{  
 $db = Db::*getConnection*();  
  
 $sql = 'INSERT INTO `film`(`name\_ru`, `name\_en`, `id\_genre`, `rating`, `year`, `type`, `country`, `img`, `img\_preview`, `trailer`, `description`, `is\_new`, `is\_recommended`, `status`, `age`, `time`) VALUES (:name\_ru, :name\_en, :id\_genre, :rating , :year, :type, :country, :img, :img\_preview, :trailer, :description, :is\_new, :is\_recommended, :status, :age, :time)';  
 $result = $db->prepare($sql);  
 $result->bindParam(':name\_ru', $film['name\_ru'], PDO::*PARAM\_STR*);  
 $result->bindParam(':name\_en', $film['name\_en'], PDO::*PARAM\_STR*);  
 $result->bindParam(':id\_genre', $film['id\_genre'], PDO::PARAM\_INT);  
 $result->bindParam(':rating', $film['rating'], PDO::*PARAM\_STR*);  
 $result->bindParam(':year', $film['year'], PDO::PARAM\_INT);  
 $result->bindParam(':type', $film['type'], PDO::*PARAM\_STR*);  
 $result->bindParam(':country', $film['country'], PDO::*PARAM\_STR*);  
 $result->bindParam(':img', $times1, PDO::*PARAM\_STR*);  
 $result->bindParam(':img\_preview', $times2, PDO::*PARAM\_STR*);  
 $result->bindParam(':trailer', $film['trailer'], PDO::*PARAM\_STR*);  
 $result->bindParam(':description', $film['description'], PDO::*PARAM\_STR*);  
 $result->bindParam(':is\_new', $film['is\_new'], PDO::PARAM\_INT);  
 $result->bindParam(':is\_recommended', $film['is\_recommended'], PDO::PARAM\_INT);  
 $result->bindParam(':status', $film['status'], PDO::PARAM\_INT);  
 $result->bindParam(':age', $film['age'], PDO::PARAM\_INT);  
 $result->bindParam(':time', $film['time'], PDO::PARAM\_INT);  
  
 return $result->execute();  
}

*/\*\*  
 \* Зміна даних фільму  
 \*/*public static function updateFilmById($id, $film)  
{  
 $db = Db::*getConnection*();  
  
 $sql = 'UPDATE `film` SET `name\_ru`=:name\_ru,`name\_en`=:name\_en,`id\_genre`=:id\_genre,`rating`=:rating,`year`=:year,`type`=:type,`country`=:country,`img`=:img,`img\_preview`=:img\_preview,`trailer`=:trailer,`description`=:description,`is\_new`=:is\_new,`is\_recommended`=:is\_recommended,`status`=:status,`age`=:age,`time`=:time WHERE id = :id';  
 $result = $db->prepare($sql);  
 $result->bindParam(':name\_ru', $film['name\_ru'], PDO::*PARAM\_STR*);  
 $result->bindParam(':name\_en', $film['name\_en'], PDO::*PARAM\_STR*);  
 $result->bindParam(':id\_genre', $film['id\_genre'], PDO::PARAM\_INT);  
 $result->bindParam(':rating', $film['rating'], PDO::*PARAM\_STR*);  
 $result->bindParam(':year', $film['year'], PDO::PARAM\_INT);  
 $result->bindParam(':type', $film['type'], PDO::*PARAM\_STR*);  
 $result->bindParam(':country', $film['country'], PDO::*PARAM\_STR*);  
 $result->bindParam(':img', $film['img'], PDO::*PARAM\_STR*);  
 $result->bindParam(':img\_preview', $film['img\_preview'], PDO::*PARAM\_STR*);  
 $result->bindParam(':trailer', $film['trailer'], PDO::*PARAM\_STR*);  
 $result->bindParam(':description', $film['description'], PDO::*PARAM\_STR*);  
 $result->bindParam(':is\_new', $film['is\_new'], PDO::PARAM\_INT);  
 $result->bindParam(':is\_recommended', $film['is\_recommended'], PDO::PARAM\_INT);  
 $result->bindParam(':status', $film['status'], PDO::PARAM\_INT);  
 $result->bindParam(':age', $film['age'], PDO::PARAM\_INT);  
 $result->bindParam(':time', $film['time'], PDO::PARAM\_INT);  
 $result->bindParam(':id', $id, PDO::PARAM\_INT);  
  
 return $result->execute();  
}  
*/\*\*  
 \* Завантаження фото  
 \*/*public static function getImg($img)  
{  
 $noImg ='no-img.jpg';  
 $path = '/template/img/films\_img/';  
  
 $pathToFilmImg = $path . $img;  
  
 if (file\_exists($\_SERVER['DOCUMENT\_ROOT'].$pathToFilmImg)){  
 return $pathToFilmImg;  
 }  
 return $path . $noImg;  
}

Декілька функцій з классу Genres:

*/\*\*  
 \* Отримання списку жанрів  
 \*/*public static function getGenresList()  
{  
 $db = Db::*getConnection*();  
 $genresList = array();  
 $result = $db->query("SELECT id, name, status FROM genres ORDER BY sort\_order ASC");  
 $i= 0;  
 while ($row = $result->fetch()){  
 $genresList[$i]['id']= $row['id'];  
 $genresList[$i]['name']= $row['name'];  
 $genresList[$i]['status']= $row['status'];  
 $i++;  
 }  
 return $genresList;  
}  
*/\*\*  
 \* Отримання списку жанрів на конкретній сторінці пагінації  
 \*/*public static function getGenreListWithPage($page)  
{  
 $page = intval($page);  
 $offset = ($page - 1) \* self::*SHOW\_BY\_DEFAULT\_GENRES*;  
 $db = Db::*getConnection*();  
  
 $films = array();  
 $sql ="SELECT id, name, status FROM genres "  
 . "WHERE status = 1 LIMIT "  
 . self::*SHOW\_BY\_DEFAULT\_GENRES* . " OFFSET ". $offset;  
 $result = $db->query($sql);  
 $i = 0;  
 while ($row = $result->fetch()){  
 $genresList[$i]['id'] = $row['id'];  
 $genresList[$i]['name'] = $row['name'];  
 $genresList[$i]['status'] = $row['status'];  
 $i++;  
 }  
 return $genresList;  
}  
*/\*\*  
 \* Отпривання жанру за параметром id  
 \*/*public static function getGenreById($id)  
{  
 $id = intval($id);  
 if ($id) {  
 $db = Db::*getConnection*();  
  
 $result = $db->query("SELECT \* FROM genres WHERE id=" . $id);  
 $result->setFetchMode(PDO::*FETCH\_ASSOC*);  
 return $result->fetch();  
 }  
}  
*/\*\*  
 \* Перевірка на довжину назви жанру  
 \*/*public static function checkName($name)  
{  
 if(strlen($name)>= 2){  
 return true;  
 }  
 return false;  
}  
*/\*\*  
 \* Переырка на однакову назву жанру  
 \*/*public static function checkRepit($genres, $genreName)  
{  
 foreach ($genres as $genre){  
 if ($genre['name'] == $genreName){  
 return false;  
 }  
 }  
 return true;  
}  
*/\*\*  
 \* Створення жанру  
 \*/*public static function createGenre($genre)  
{  
 $db = Db::*getConnection*();  
  
 $sql = 'INSERT INTO genres (name, status, sort\_order) '  
 . 'VALUES (:name, :status, :sort\_order)';  
 $result = $db->prepare($sql);  
 $result->bindParam(':name', $genre['name'], PDO::*PARAM\_STR*);  
 $result->bindParam(':status', $genre['status'], PDO::*PARAM\_STR*);  
 $result->bindParam(':sort\_order', $genre['sort\_order'], PDO::*PARAM\_STR*);  
 return $result->execute();  
}  
*/\*\*  
 \* Зміна даних жанру  
 \*/*public static function updateGenreById($id, $genre)  
{  
 $db = Db::*getConnection*();  
 $sql = 'UPDATE `genres` SET `name`=:name,`sort\_order`=:sort\_order,`status`=:status WHERE id = :id';  
 $result = $db->prepare($sql);  
 $result->bindParam(':name', $genre['name'], PDO::*PARAM\_STR*);  
 $result->bindParam(':sort\_order', $genre['sort\_order'], PDO::*PARAM\_STR*);  
 $result->bindParam(':status', $genre['status'], PDO::*PARAM\_STR*);  
 $result->bindParam(':id', $id, PDO::PARAM\_INT);  
  
 return $result->execute();  
}  
*/\*\*  
 \* Видалення жанру  
 \*/*public static function deleteGenreById($id)  
{  
 $db = Db::*getConnection*();  
  
 $sql = 'DELETE FROM genres WHERE id= :id';  
 $result = $db->prepare($sql);  
 $result->bindParam(":id", $id, PDO::*PARAM\_STR*);  
 return $result->execute();  
}