Министерство цифрового развития, связи и массовых коммуникаций Российской Федерации

Ордена Трудового Красного Знамени

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Московский технический университет связи и информатики»

Проектная работа «Разработка веб-сайта на тему онлайн кинотеатра»

по предмету «UI/UX»

Выполнил:

студент группы БВТ2002

Зинов Т.В.

Проверил:

Дворянинов Павел Владимирович

**Оглавление**

[Введение 2](#_Toc154219532)

[Цель работы 2](#_Toc154219533)

[Выполнение работы и листинг кода 3](#_Toc154219534)

[Демонстрация работы веб-сайта 12](#_Toc154219535)

[Вывод 16](#_Toc154219536)

# Введение

В рамках задачи по UI/UX Design передо мной стояла задача — доработать и сверстать свой собственный веб-сайт, посвященный фильмам. Для этого я выбрала технологии Next.js и React Query, нацеленные на создание современного и отзывчивого пользовательского интерфейса.

Для эффективного взаимодействия с данными я использую Node.js — среду выполнения кода JavaScript, которая позволяет мне создавать серверную часть приложения и обеспечивает единый язык программирования для клиентской и серверной частей.

Проект разрабатывается в соответствии с современными стандартами веб-разработки, где React отвечает за динамическое обновление контента и взаимодействие с пользователем. Использование React позволяет мне эффективно организовывать компонентный подход, что способствует чистому и лаконичному коду. Также мною был проработан адаптивный дизайн, и самое главное красивый, понятный и нативный для пользователя, без каких-либо перегрузок страницы.

Цель работы

Основной целью проекта была визуализация данных о фильмах, полученные из API, в удобной таблице, предоставляя пользователям подробную информацию о каждом фильме. Каждая карточка фильма содержит описание, жанры, оценки, а также предоставляет возможность скачать фильм через торрент. Помимо этого, предусматривается возможность оставлять комментарии и удалять их в любой момент.

# Выполнение работы и листинг кода

Структура моего проекта состояла из модуля api, компонентов (card, header), хуков (useComments), страниц (api, movie(id)), файлов \_app, \_document, index, хранилища store (theme), а также стилевых файлов css (globals).

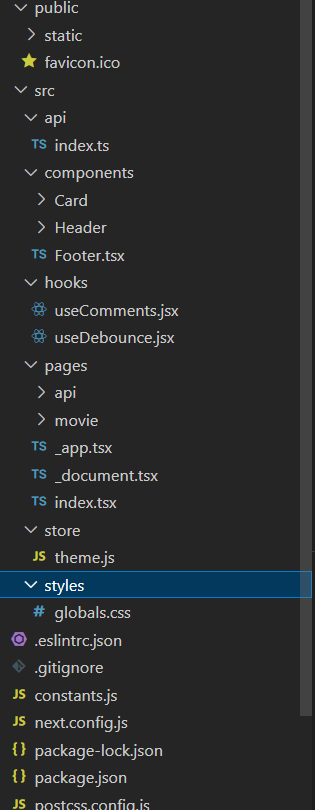


Рисунок 1 – Структура проекта

Далее на рисунках я представлю код всех компонентов, файлов, стилей, апишек и хуков моего сайта.

На рисунке 2 листинг кода **компонента карточек (Cards.tsx)**, где представлена реализация отдельной карточки фильма. Компонент принимает необходимые свойства, такие как id, title, year, medium\_cover\_image, description, rating, genre, и filter. Используется локальное состояние (useState) для отслеживания состояния наведения мыши на карточку. При наведении происходит отображение дополнительной информации о фильме. При клике на карточку происходит перенаправление на страницу с деталями фильма через useRouter из Next.js. Дополнительно, компонент содержит условие для скрытия карточки, если отсутствует описание и установлен фильтр. Структура компонента включает в себя изображение, информацию о фильме (заголовок, описание, жанр), а также интерактивные элементы, активируемые при наведении мыши.

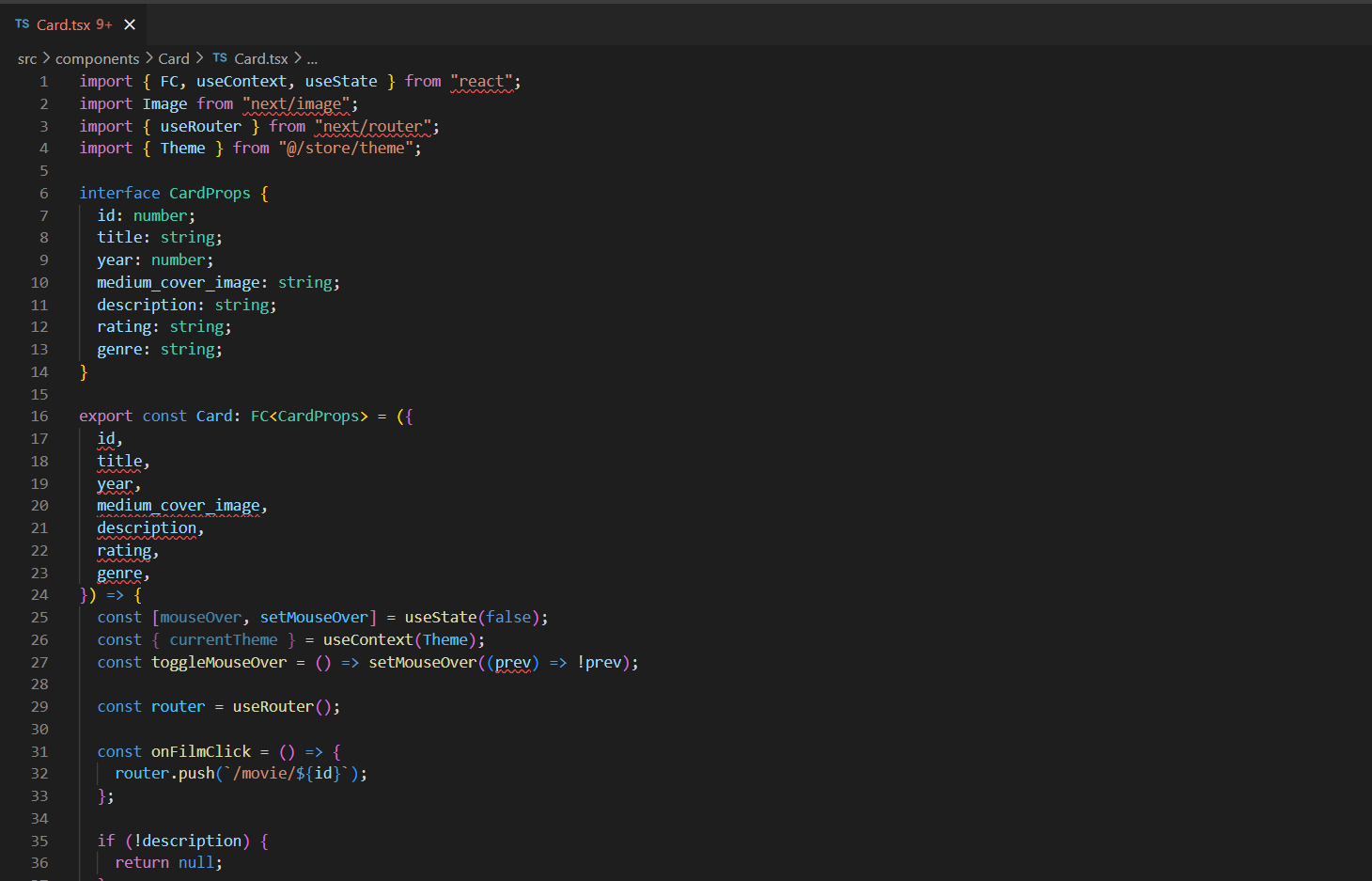


Рисунок 2 – код Cards.tsx

На рисунке 3 представлен листинг кода **шапки (header.tsx),** где реализована создание верхней части веб-страницы, иными простыми словами шапка/хэдер. Компонент принимает свойство arrowBack, которое определяет наличие кнопки "назад" (стрелка) (отображается только на странице с фильмом, дабы была возможность вернутся в главное меню). Используется локальный хук useRouter из Next.js для управления маршрутами приложения. Компонент включает в себя заголовок, который отображает название "YSMLreska", и контрольные элементы — кнопку "назад" (если установлено свойство arrowBack) и переключатель темы. Переключатель темы представлен изображением, которое меняет свой цвет в зависимости от текущей темы. При клике на переключатель вызывается функция toggleTheme для смены темы. Компонент стилизован с использованием Tailwind CSS, а его цвет и фон адаптируются в зависимости от текущей темы. Компонент Header предоставляет пользователю навигационные и тематические возможности, обеспечивая при этом удобство использования.

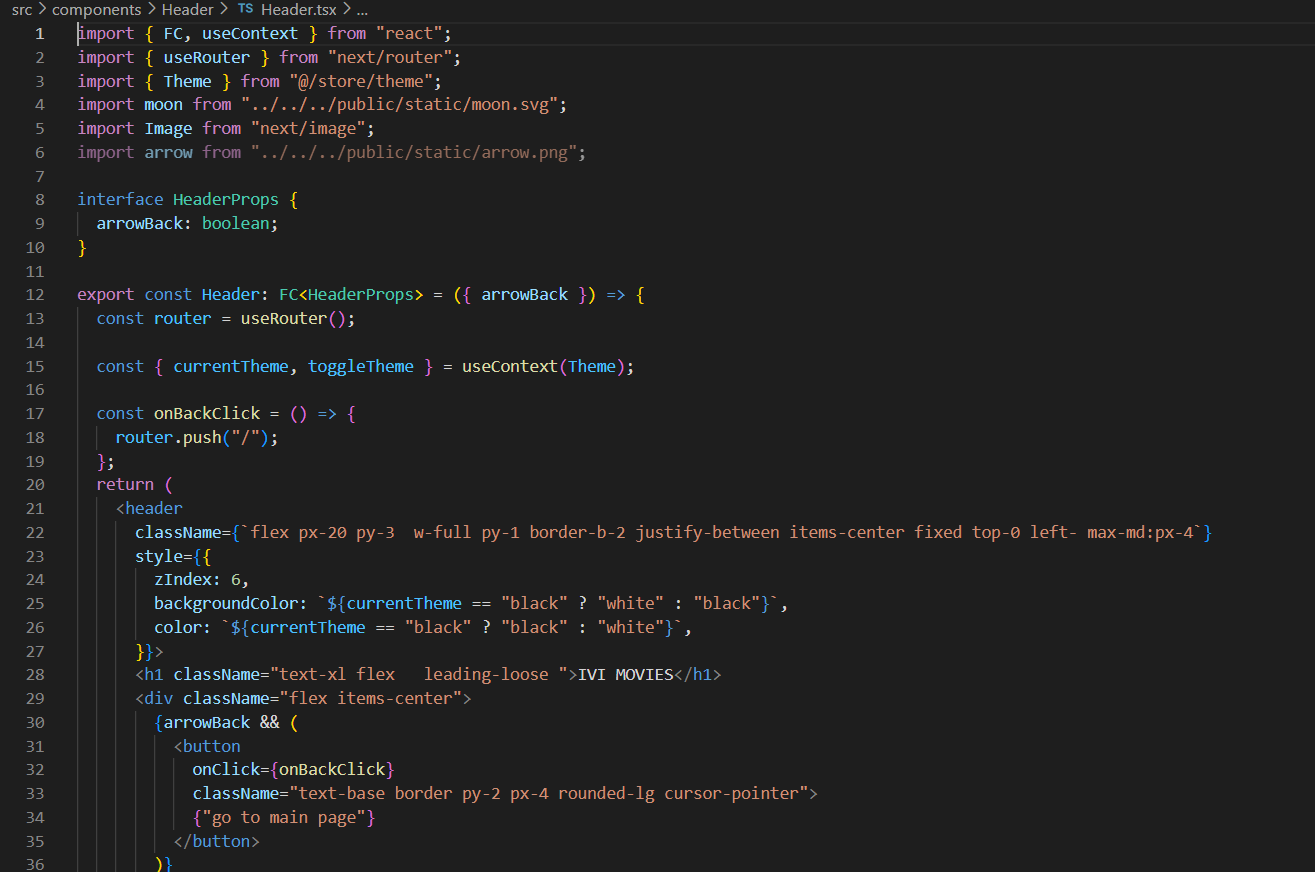


Рисунок 3 – код header.tsx

На рисунке 4 представлен **хук useComments.jsx**, в котором реализован функционал для управления комментариями. Хук использует библиотеку SWR с помощью "use client" для работы с данными.С помощью useState и useEffect хук отслеживает изменения в хранилище браузера для конкретного идентификатора (id) и обновляет состояние comments при изменениях в хранилище. Хук предоставляет две основные функции: **updateComments** - обновляет список комментариев, добавляя новый комментарий, при условии наличия имени и текста комментария. Обновленный список сохраняется в локальном хранилище и в состоянии comments; **deleteComment** - удаляет выбранный комментарий из списка. Обновленный список сохраняется в локальном хранилище и в состоянии comments.

Хук useComments предоставляет удобные методы для работы с комментариями, а также автоматически синхронизирует данные с локальным хранилищем браузера.

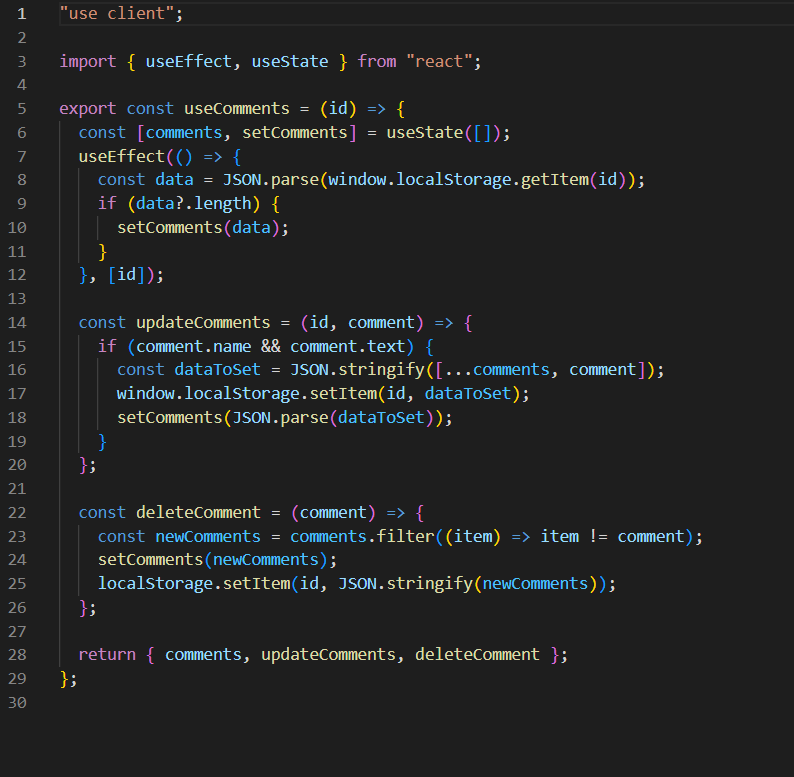


Рисунок 4 – код хука useComments.jsx

На рисунках 5-7 представлен код [id].tsx, где реализована страница деталей фильма. В данном компоненте используется SWR для работы с данными и динамической загрузки контента. Ниже я приведу описание и суть основных компонентов:

1. **Стейт и эффекты:**

* movieDetails: Состояние, хранящее информацию о выбранном фильме.
* loading: Состояние, указывающее на процесс загрузки данных.
* comment: Состояние, представляющее форму для ввода комментария.

1. **Хук useParams и SWR:**

useComments: Собственный хук, управляющий комментариями для данного фильма.

1. **Интерфейс и кнопка "назад":**

* Компонент <Header> используется для создания верхней части страницы с возможностью вернуться назад.

1. **Отображение контента:**

* Используется блок <Image> для отображения фонового изображения фильма.
* Информация о фильме, такая как заголовок, описание, жанры, рейтинг, и другие детали, отображается в структурированном формате.

1. **Кнопки "Watch now" и "Download":**

* При клике на кнопку "Watch now" осуществляется переход на страницу фильма.
* Кнопка "Download" позволяет отобразить список доступных для скачивания файлов.

1. **Форма комментариев:**

* Пользователь может вводить свое имя и комментарий.
* При отправке комментария вызывается функция onSendComment, которая обновляет состояние комментариев и сохраняет данные в локальном хранилище.

1. **Отображение комментариев:**

* Комментарии отображаются в блоке, предоставляя информацию о имени автора и тексте комментария.
* Предусмотрена возможность удаления комментария.

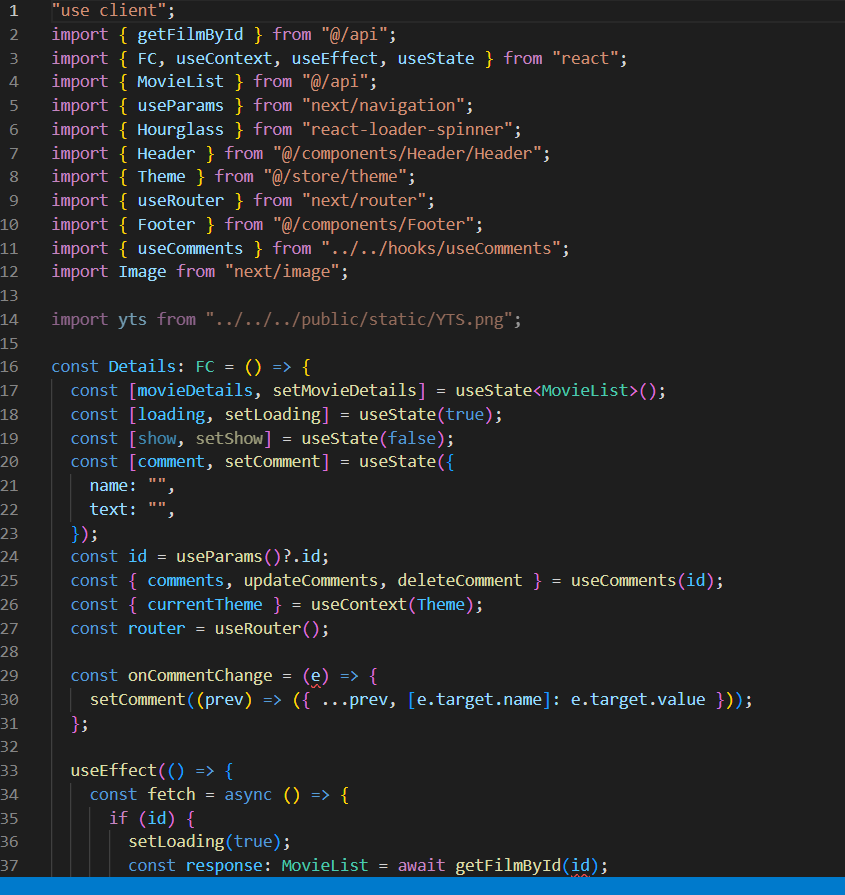


Рисунок 5 – код [id].tsx



Рисунок 6 – код [id].tsx

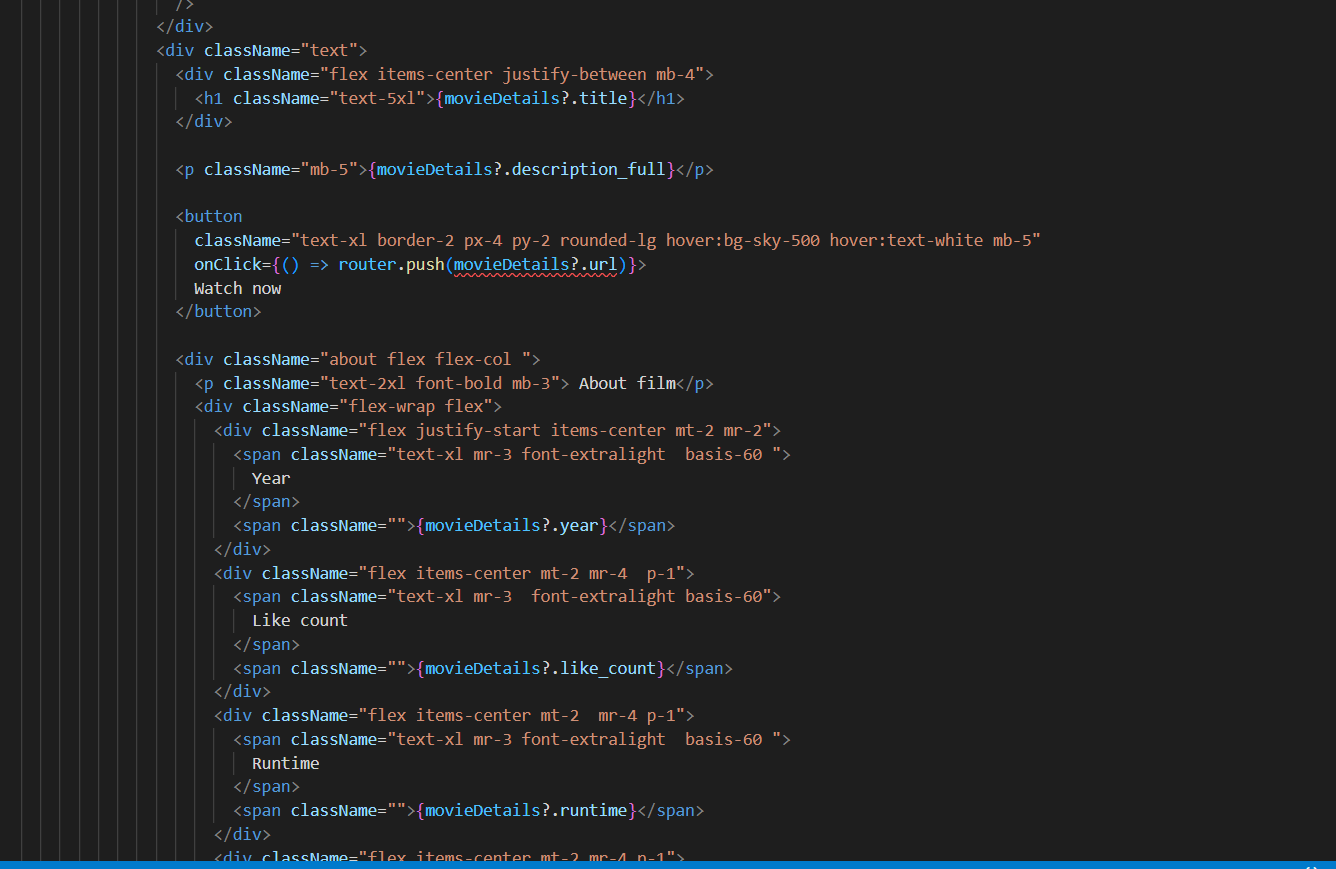


Рисунок 7 – код [id].tsx

На рисунке 8 листинг кода index.tsx — это файл, отвечающий за отображение домашней страницы приложения. Он показывает список фильмов с возможностью пагинации и фильтрации по наличию описания.

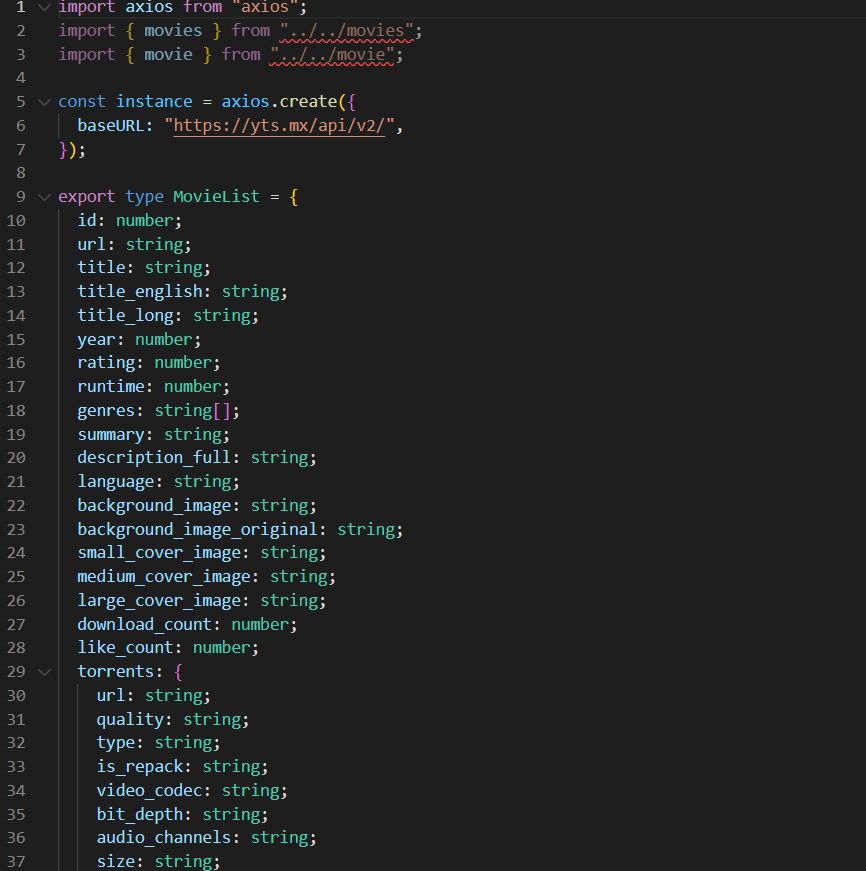




Рисунок 8 – код index.tsx

На рисунке 9 листинг кода theme.js, который управляет темой приложения. Он определяет контекст Theme, предоставляющий информацию о текущей теме и методе её изменения через компонент-провайдер ThemeProvider. Внутри ThemeProvider устанавливается состояние currentTheme для отслеживания текущей темы, и есть функция toggleTheme(), которая позволяет переключаться между темами - "темной" и "светлой". При инициализации приложения происходит проверка наличия сохраненной темы в локальном хранилище, и, если такой нет, устанавливается тема по умолчанию - "черная". Весь этот функционал позволяет применять выбранную тему в интерфейсе приложения, обеспечивая гибкость в выборе внешнего вида приложения пользователем



Рисунок 9 - код theme.js

На рисунке 10 представлен листинг кода global.css, который содержит стили, определяющие общий внешний вид элементов веб-приложения. Он включает базовые, компонентные и утилитарные стили из Tailwind CSS. В дополнение к этому, в файле определены стили для активных и наведенных элементов, настройки фона, значения z-index для слоев, цвета фона для различных тем, а также стили для обрезки и отображения описания фильма. Эти стили обеспечивают единый и структурированный внешний вид интерфейса, делая его более читаемым, удобным и лакончиным для пользователя.

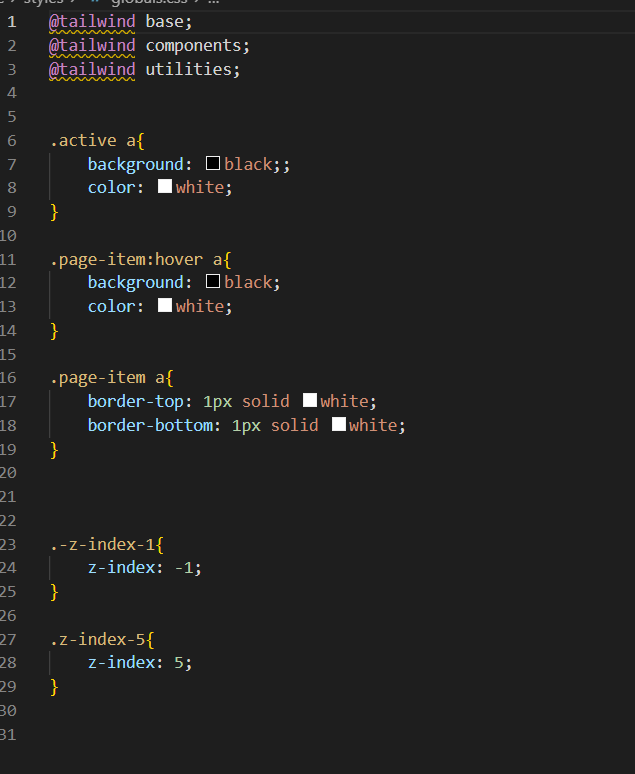


Рисунок 10 – код global.css

# Демонстрация работы веб-сайта

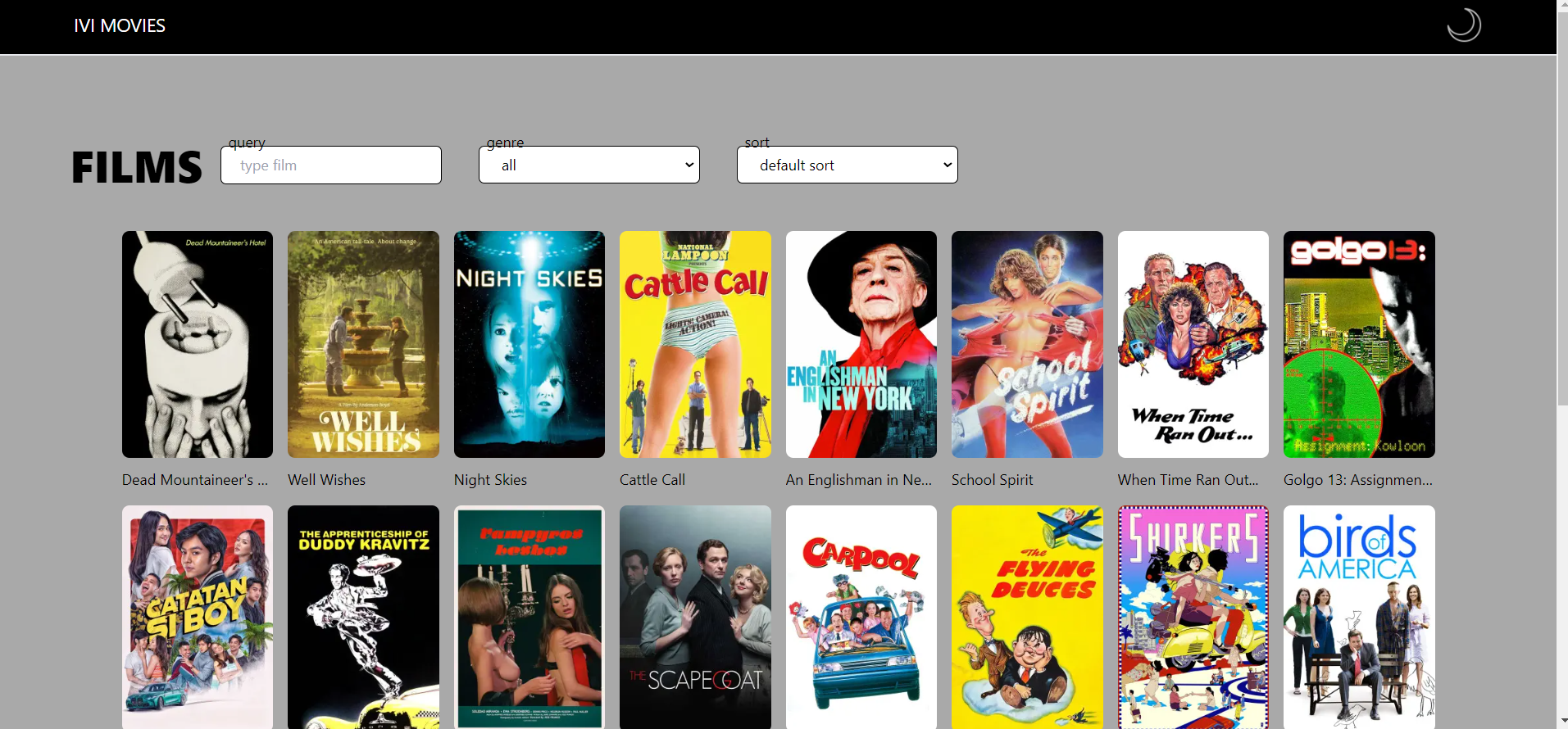


Рисунок 11 – главная страница в темной по умолчанию теме с активированным чек-боксом фильтрации фильмов без описания

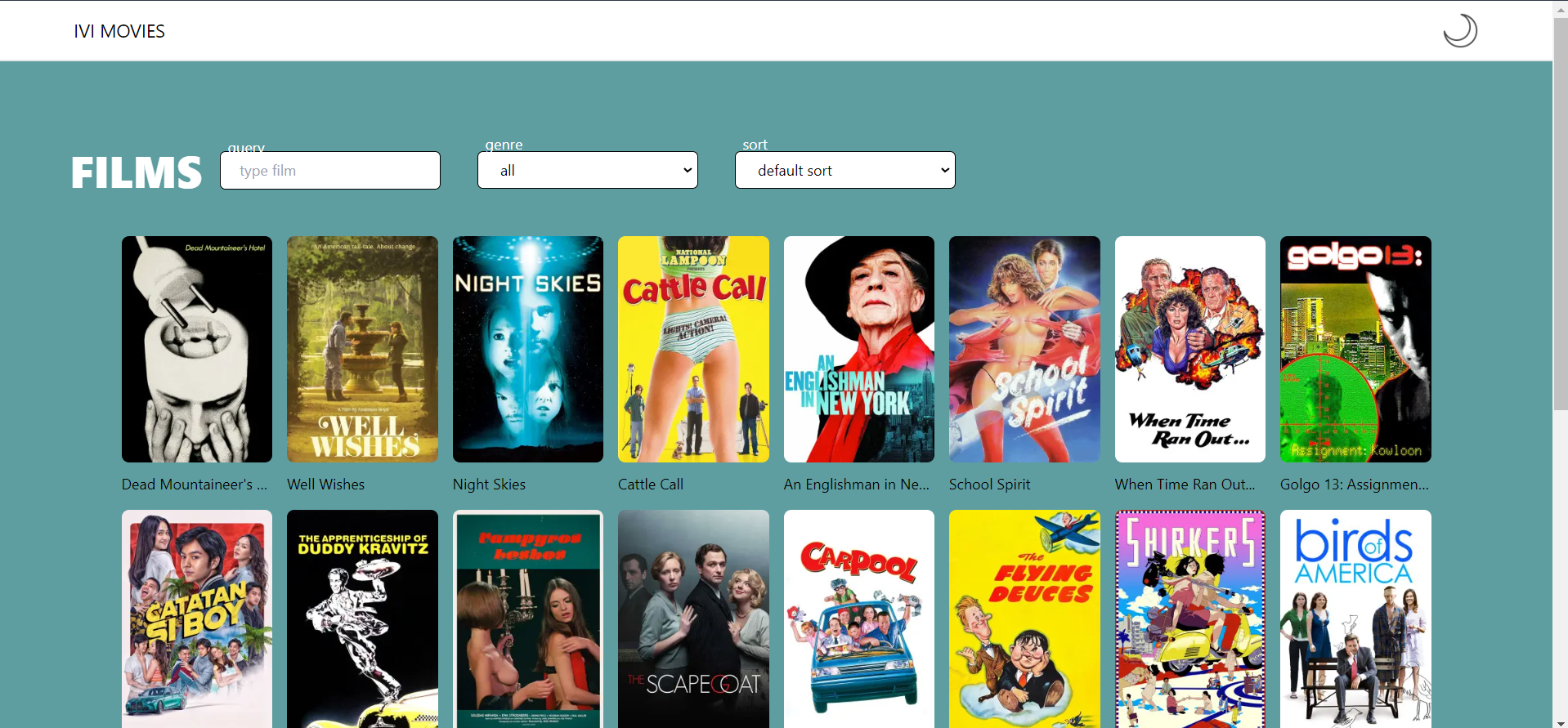


Рисунок 12 – аналогично главная страница, но с переключенной на светлую, темой



Рисунок 13 – карточка фильма при наведении на нее курсором мыши

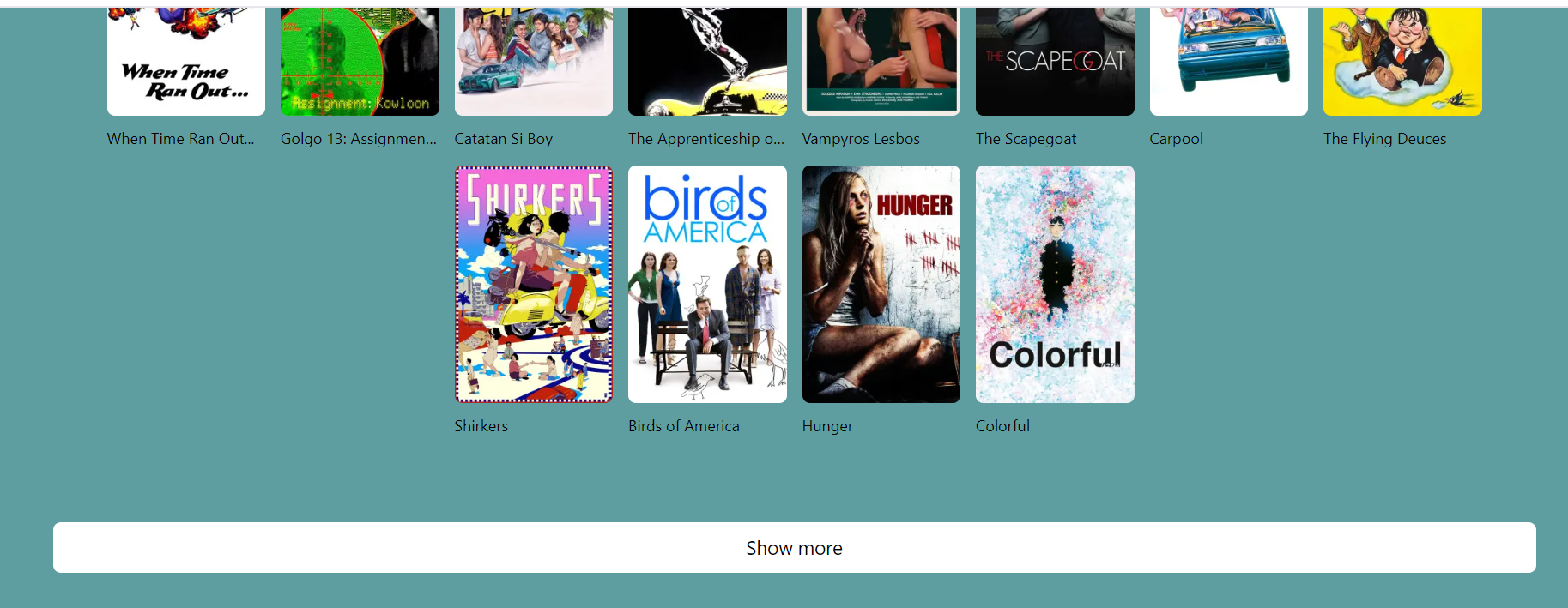


Рисунок 14 – демонстрация работающей пагинации

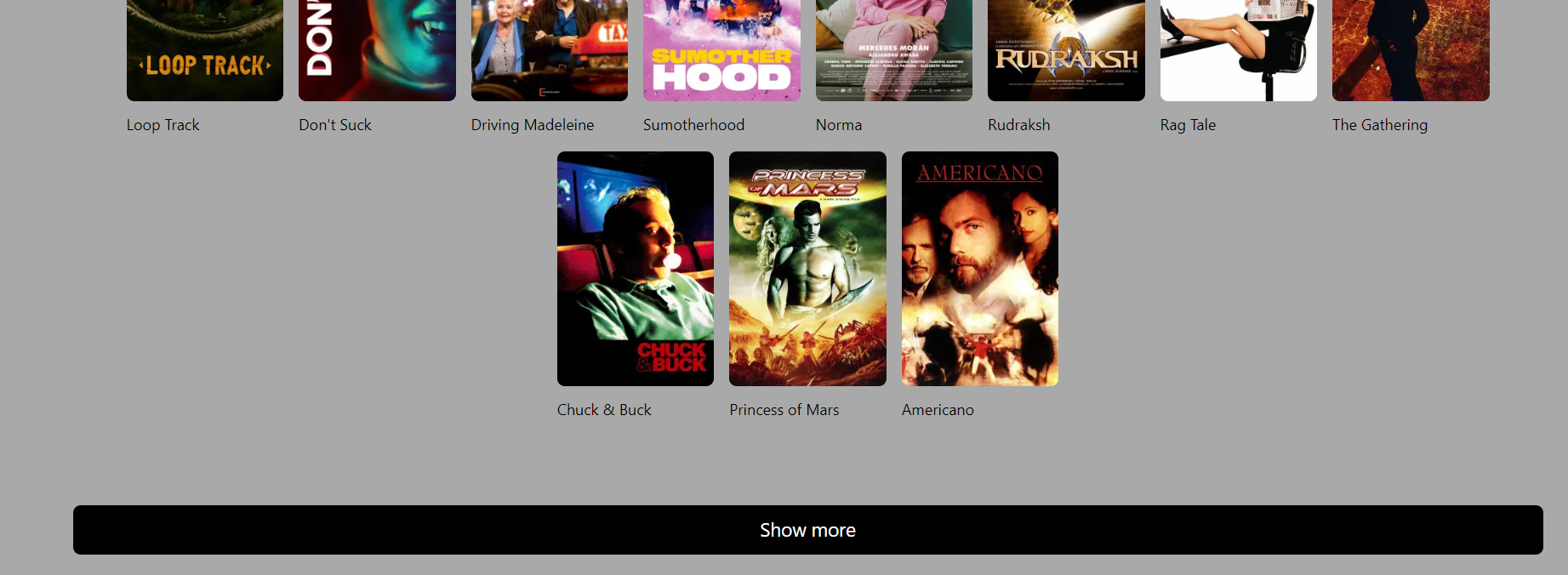


Рисунок 15 – демонстрация работающей пагинации

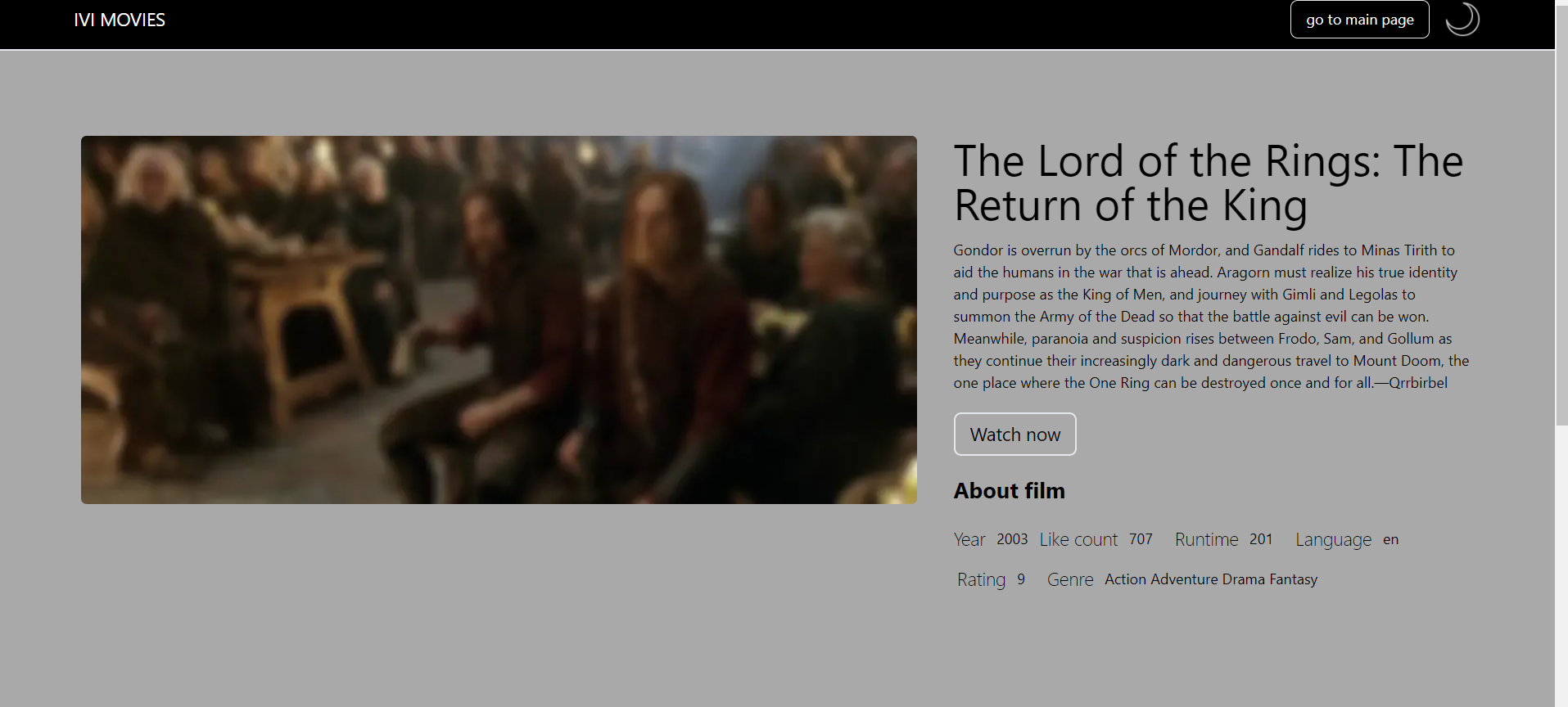


Рисунок 16 – страница с фильмом с переработанным дизайном

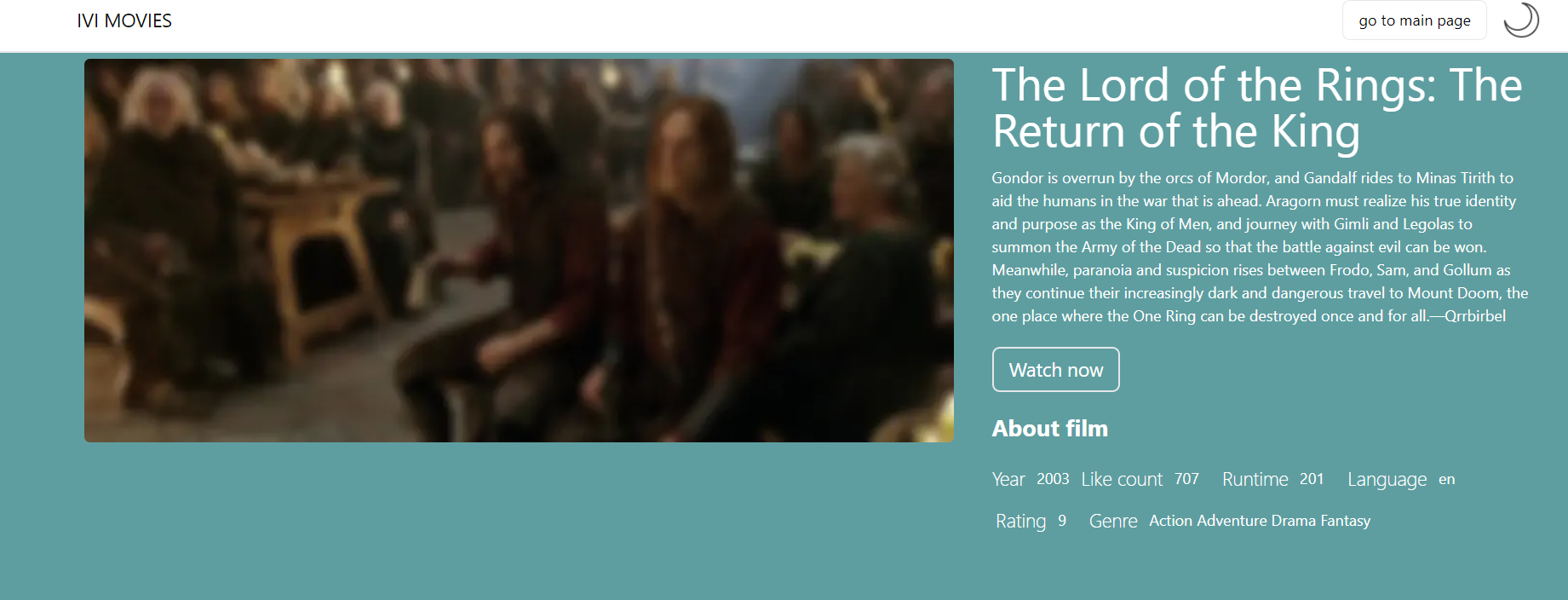


Рисунок 17 – аналогично, но со светлой темой



Рисунок 18 – блок комментариев с обработкой сценария, когда таковых нет

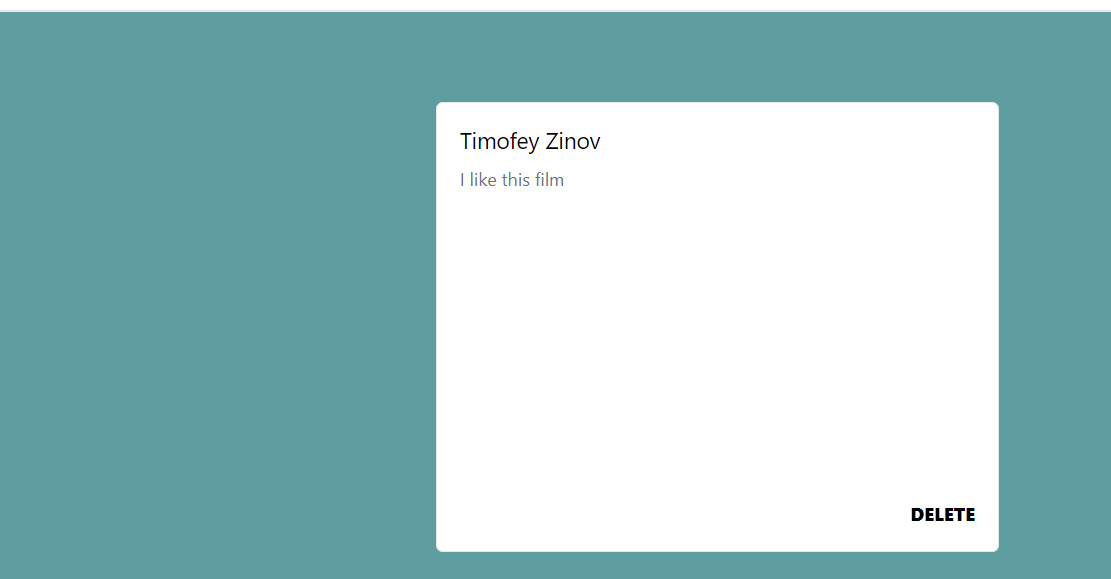


Рисунок 19 – ситуация, когда комментарий присутствует, комментарий также можно удалять

# Вывод

Мною был разработан собственный веб-сайт, посвященный фильмам, некий онлайн кинотеатр, который сочетает в себе лаконичность и удобство для пользователя. Также я подтянула свои знания в области верстки, в области React Querry, Node.js!