

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»

Выполнила:

студентка группы БВТ2002

Леонова Виктория Олеговна

Проверил:

Дворянинов Павел Владимирович

## **Введение**

По предмету UI/UX Design была поставлена задача — доработать функции и дизайн веб-сайт, посвященный фильмам. Для этого были использованы следующие технологии: Next.js и React Query, нацеленные на создание современного и отзывчивого пользовательского интерфейса.

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

Использование React позволяет эффективно организовывать компонентный подход, что способствует чистому и лаконичному коду.

Также адаптивный дизайн, а самое главное, понятный для пользователя, без каких-либо перегрузок страницы.

## **Цель работы**

Цель проекта - визуализация данных о фильмах, полученные из API, удобной таблице, предоставляя пользователям подробную информацию о каждом фильме.

Каждая карточка фильма содержит описание, жанры, рейтинг, дату загрузки, год выпуска, продолжительность, количество лайков и ссылку на imdb. Помимо этого, предусматривается возможность оставлять комментарии и удалять их в любой момент.

## Выполнение работы и листинг кода

Структура проекта состоит из двух страниц - главной и детальной фильма, файлов компонентов (карточка на главной, детальная карточка, пагинация, сетка карточек, формы отправки комментариев, блок комментариев и каркас страницы), а также стилевых файлов стилей scss (globals).

Все стили написаны на фреймворке sass, который позволяет использовать и писать миксины, переменные наследования стилей и многое другое, вместе с ним был использован Bootstrap 5 для адаптивности верстки.

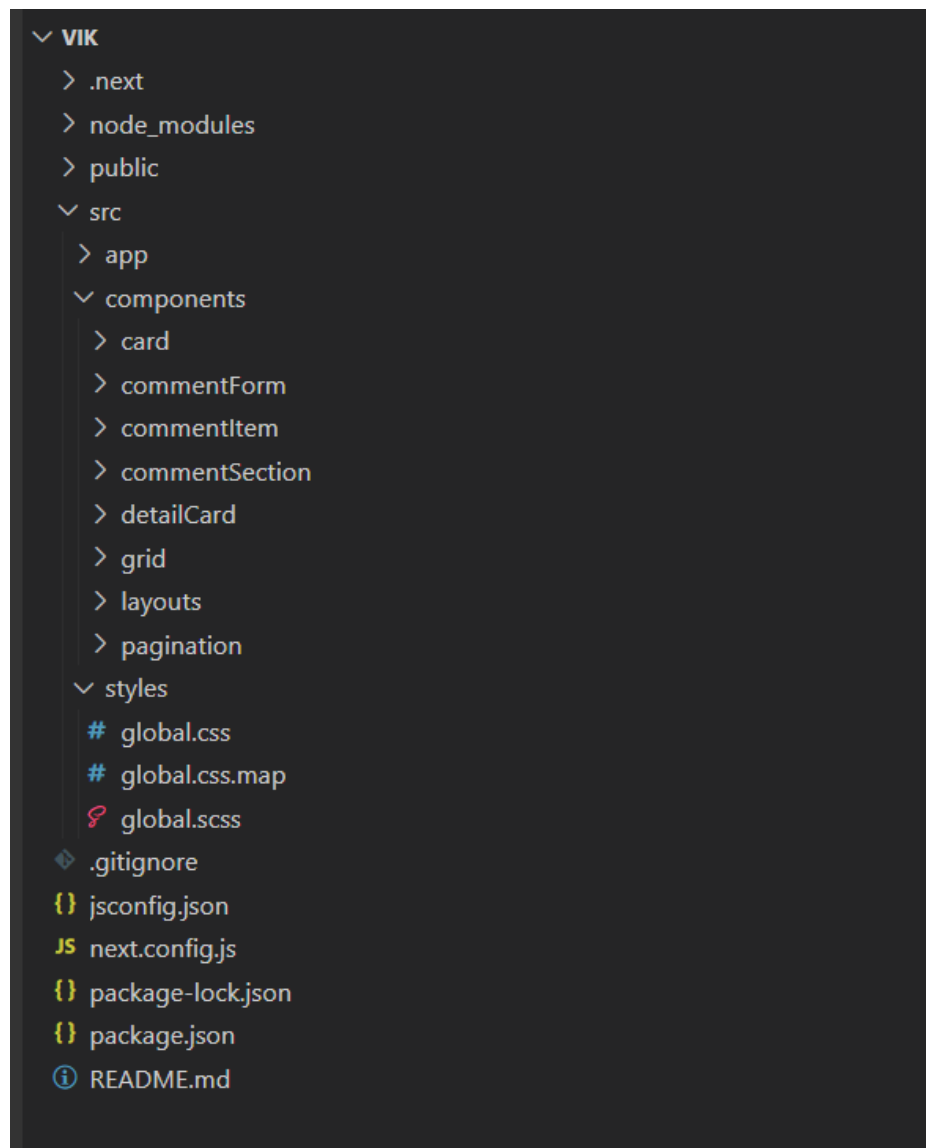


Рисунок 1 – Структура проекта.

На рисунке 2 листинг кода компонента карточек (Cards.tsx), где представлена реализация отдельной карточки фильма. Компонент принимает необходимые свойства, такие как id, title, image. При клике на карточку происходит перенаправление на детальную страницу фильма через next/link из Next.js.

```
JS card.js
src > components > card > JS card.js > ...
1   import Link from 'next/link'
2
3   export default function Card(props) {
4     const onError = (e) => {
5       e.target.onerror = null
6       e.target.src = '/fon.jpg'
7     }
8
9     return (
10      <Link className='no-decoration' href={'/movie/' + props.id}>
11        <div className='card-item text-start h-100'>
12          <img
13            className='card-item-img-top'
14            src={props.image}
15            onError={(e) => onError(e)}
16          />
17          <div className='card-item-body'>
18            <h4 className='card-item-title'>{props.title}</h4>
19          </div>
20        </div>
21      </Link>
22    )
23  }
24
```

Рисунок 2 – Код card.js.

На рисунках 3, 4 представлен листинг кода шаблона страницы, состоящей из шапки и футера страницы. Шапка состоит из лого сайта (ссылка на главную), футер - фотокарточка с информацией о создателе сайта и кнопкой скролла вверх страницы.

Каждая страница сайта встраивается внутрь этого шаблона.

```
JS main.js X
src > components > layouts > JS main.js > Layout
1 export default function Layout({ children }) {
2   return (
3     <html lang='en'>
4       <head>
5         <title>Главная</title>
6         <link
7           rel='icon'
8           type='image/x-icon'
9           href='/free-icon-font-play-alt-3914327.svg'
10        />
11       </head>
12       <body>
13         <div className='container-fluid'>
14           <header class='py-3 mb-4 border-bottom'>
15             <a
16               href='/'
17               class='center-logo mb-3 mb-md-0 me-md-auto text-dark text-decoration-none'
18             >
19               <span class='fs-1 head-text'>RMDB</span>
20             </a>
21           </header>
22           <main>{children}</main>
23         </div>
24         <div class='footer'>
25           <footer class='d-flex flex-wrap justify-content-between align-items-center py-5 px-5 border-top'>
26             <div class='col-md-4 d-flex align-items-center w-100'>
27               <div class='row w-100'>
28                 <div class='col-foot-small'>
29                   <span class='mb-3 mb-md-0 text-body-secondary'>
30                     Сайт сделан в учебных целях
31                   <br />
32                   Веб-дизайнер Виктория
33                   <br />
34                   <img src='/tg.svg'></img>
35                   &nbsp;&nbsp;&@leonovva_victorria
36                 </span>
37               </div>
38               <div class='col-foot-wide'>
39                 <img src='/Ellipse.svg' width='100px' height='100px'></img>
40               </div>

```

### Рисунок 3 – Код main.js.

```

41     <ul class='nav col-foot-arrow list-unstyled d-flex'>
42       <li class='mx-5'>
43         <img
44           src='/Group.svg'
45           width='100px'
46           onClick={() => window.scrollTo(0, 0)}
47           style={{ cursor: 'pointer' }}
48           height='100px'
49         ></img>
50       </li>
51     </ul>
52   </div>
53 </div>
54 </footer>
55 </div>
56 </body>
57 </html>
58 )
59 }
60

```

**Рисунок 4** – Код main.js.

Сетка фильмов реализована в виде отдельного компонента (рисунки 5-6), в котором подключаются компонент карточки и пагинации. Фильмы получаются через axios запрос к api, обернутый в useQuery хук, который возвращает флаги состояния запроса (ошибка или загрузка) и его результат по мере получения (data) и как любой хук вызывает перерисовку компонента.

Если запрос еще в процессе, то выводится loader, если же запрос выполнен с ошибкой - сообщение Error, если данных не было получено - по data и если данные пришли, то отрисовывается сетка фильмов. Также в хук передается параметр keepPreviousData, который сохраняет прошлый стейт между переключениями, из-за чего при пагинации сетка не исчезает).

В каждую карточку в сетке через props передаются данные, полученные из api, а именно: id, title, medium\_cover\_image каждого фильма.

Пагинация сетки подключается отдельным компонентом, а само состояние создается именно в компоненте сетки. Сам стейт хранит единственное значение - номер страницы пагинации, по умолчанию значение берется из GET параметра ?page, либо, если параметр не определен, устанавливается 1.

На странице выводится 20 фильмов по умолчанию. В компонент пагинации передается номер текущей страницы, общее количество страниц и функция для смены состояния пагинации.

```
JS grid.js  X
src > components > grid > JS grid.js > ...
1  'use client'
2
3  import { useState, useEffect } from 'react'
4  import { useQuery } from 'react-query'
5
6  import Pagination from '@components/pagination/pagination'
7  import Card from '@components/card/card'
8  import axios from 'axios'
9
10 import { useSearchParams } from 'next/navigation'
11
12 async function fetchFilms(pagen = 1) {
13   const response = await axios.get(
14     `https://yts.mx/api/v2/list_movies.json?page=${pagen}`
15   )
16
17   return response.data.data
18 }
19
20 function getLastSessionPage() {
21   const searchParams = useSearchParams()
22   const page = searchParams.get('page')
23
24   return page ?? 1
25 }
26
27 export default function Grid(props) {
28   const [page, setPage] = useState(getLastSessionPage())
29   const { data, isLoading, isError } = useQuery(
30     [page, 'movies'],
31     () => fetchFilms(page),
32     {
33       keepPreviousData: true,
34     }
35   )
36
37   if (isError) {
38     return <h1>Error</h1>
39   }
40
```

Рисунок 5 – Код grid.js.

```
JS grid.js X
src > components > grid > JS grid.js > ...
40
41   if (isLoading) {
42     return (
43       <div className='d-flex justify-content-center'>
44         <div
45           class='spinner-border'
46           style={{ color: '#f7bc46', width: '5rem', height: '5rem' }}
47           role='status'
48         >
49           <span class='visually-hidden'>Loading...</span>
50         </div>
51       </div>
52     )
53   }
54
55   if (!data) {
56     return <h1>No data</h1>
57   }
58
59   const movies = data.movies
60   const maxPages = Math.ceil(data.movie_count / data.limit)
61   return (
62     <div className='container main pb-5 pt-5'>
63       <div className='container overflow-hidden w-md-50'>
64         <div className='row row-cols-2 row-cols-md-5 g-4 gy-5'>
65           {movies.map((card) => (
66             <div className='col'>
67               <Card
68                 id={card.id}
69                 title={card.title}
70                 image={card.medium_cover_image}
71               ></Card>
72             </div>
73           ))}
74         </div>
75         <Pagination
76           currentPage={page}
77           maxPages={maxPages}
78           setPage={setPage}
79         ></Pagination>
80       </div>
81     </div>
82   )
83 }
```

Рисунок 6 – Код grid.js.



Сам компонент пагинации (рисунок 7) содержит в себе логику расчета буллетов (кнопок пагинации) и саму верстку. На каждую кнопку и буллет привязывается обработчик события `OnClick`, который вызывает функцию изменения состояния пагинации из `props` в соответствии с нажатой кнопкой (выбранной страницей).

Пагинация всегда отображает 5 буллетов пагинации (2 слева и 2 справа от выбранной или же все 4 слева или справа), а также кнопки пролистывания на следующую и предыдущую страницы).

Стиль каждой кнопки устанавливается в соответствии с текущей выбранной страницей.

```
JS pagination.js X
src > components > pagination > JS pagination.js > Pagination > currentPage
1  export default function Pagination(props) {
2    let pages = []
3    const currentPage = props.currentPage
4    const maxPages = props.maxPages
5    const setPage = props.setPage
6
7    if (currentPage <= 3) {
8      pages = [...Array(Math.min(maxPages, 5)).keys()].map((i) => i + 1)
9    } else {
10     let i = currentPage - 2
11     while (i <= parseInt(currentPage) + 2 && i <= maxPages) {
12       pages.push(i)
13       i++
14     }
15   }
16   return (
17     <nav aria-label='Page navigation example'>
18       <ul className='pagination'>
19         <li className='page-item'>
20           <a className='no-decoration page-link' + (currentPage != 1 ? '' : 'disabled')>
21             onClick={() => setPage(currentPage - 1)}
22           >
23             <b>{'<'}</b>
24           </a>
25         </li>
26         {pages.map((page) => (
27           <li className='page-item'>
28             <a className='no-decoration page-link' + (currentPage != page ? '' : 'active')>
29               onClick={() => setPage(page)}
30             >
31               <b>{page}</b>
32             </a>
33           </li>
34         ))}
35         <li className='page-item'>
36           <a className='no-decoration page-link' + (currentPage != maxPages ? '' : 'disabled')>
37             onClick={() => setPage(currentPage + 1)}
38           >
39             <b>{'>'}</b>
40           </a>
41         </li>
42       </ul>
43     </nav>
44   )
}
```

Рисунок 7 – Код `pagination.js`.

В компоненте детальной карточки (рисунки 8-9), также как и в сетке, делается запрос в апи для получения детальной информации по фильму через хук useQuery. В качестве параметра передается id текущего фильма, ответ представляет собой объект с детальной информацией, которая выводится через jsx код.

```
JS detailCard.js X
src > components > detailCard > JS detailCard.js > ...
1  import { useQuery } from 'react-query'
2  import axios from 'axios'
3  import CommentSection from '@components/commentSection/commentSection'
4
5  async function fetchFilm(id = 1) {
6    const response = await axios.get(
7      `https://yts.mx/api/v2/movie_details.json?movie_id=${id}&with_images=true`
8    )
9    return response.data.data
10 }
11
12 export default function DetailCard(props) {
13   const id = props.id
14   const { data, isLoading, isError } = useQuery([id, 'film'], () =>
15     fetchFilm(id)
16   )
17
18   if (isError) {
19     return <h1>Error</h1>
20   }
21
22   if (isLoading) {
23     return (
24       <div className='d-flex justify-content-center'>
25         <div
26           className='spinner-border'
27           style={{ color: '#f7bc46', width: '5rem', height: '5rem' }}
28           role='status'
29         >
30           <span class='visually-hidden'>Loading...</span>
31         </div>
32       </div>
33     )
34   }
35
36   if (!data) {
37     return <h1>No data</h1>
38   }
39   const movie = data.movie
```

Рисунок 8 – Код detailCard.js.

```

JS detailCard.js X
src > components > detailCard > JS detailCard.js > DetailCard
43   return (
44     <div className='container-detail ml-4 p-5 mb-5'>
45       <div className='row row-cols-1 row-cols-md-2'>
46         <div class='col'>
47           <div class='project-info-box mt-0'>
48             <h5 className='card-title'>
49               <b>{movie.title_long}</b>
50             </h5>
51             {movie.description_full && (
52               <p
53                 class='card-body-text-detail mb-0'
54                 style={{ textAlign: 'justify' }}
55               >
56                 {movie.description_full}
57             </p>
58             )}
59           </div>
60           <div class='mt-3 project-info-box'>
61             <p class='mb-0 card-title'><b>Дата загрузки:</b> {movie.date_uploaded}</p>
62             <p class='mb-0 card-title'><b>Рейтинг:</b> {movie.rating} </p>
63             <p class='mb-0 card-title'><b>Год выпуска:</b> {movie.year} </p>
64             <p class='mb-0 card-title'><b>Продолжительность:</b> {movie.runtime} минут</p>
65             <p class='mb-0 card-title'><b>Жанры:</b> {movie.genres.join(', ')} </p>
66             <p class='mb-0 card-title'><b>Понравилось:</b> {movie.like_count ? movie.like_count : 0} </p>
67             <p class='mb-0 card-title'>
68               <a
69                 target='_blank'
70                 href={'https://www.imdb.com/title/' + movie.imdb_code}
71                 className='no-decoration'
72               >
73                 <b>Ссылка imdb</b>
74               </a>
75             </p>
76           </div>
77         </div>
78         <div class='col col-img'>
79           <img src={movie.large_cover_image} class='img-fluid' />
80         </div>
81         <div className='col w-100'>
82           <CommentSection movieId={movie.id}></CommentSection>
83         </div>
84       </div>
85     </div>
86   )

```

Рисунок 9 – Код detailCard.js.

Комментарии выводятся через компонент CommentSection (рисунок 10). Он содержит логику получения комментариев из localStorage для текущего фильма, хранение в формате ключ = id фильма, значение = json массив комментариев. Также для добавления комментариев используется состояние (хук useState), оно хранит в себе массив с комментариями к текущему фильму, значение по умолчанию берется из localStorage, либо пустой массив.

Добавление комментария в localStorage реализовано через хук useEffect, который вызывается после каждой отрисовки компонента.

Функция обработки отправки комментария вызывает смену состояния и последующую очистку формы, саму функцию мы передаем в компонент с формой отправки, который подключаем ниже в jsx.

Для каждого комментария также подключаем отдельный компонент, в который передаем id комментария, его содержание и обработчик удаления комментария, внутри которого модифицируем наше состояние комментариев.

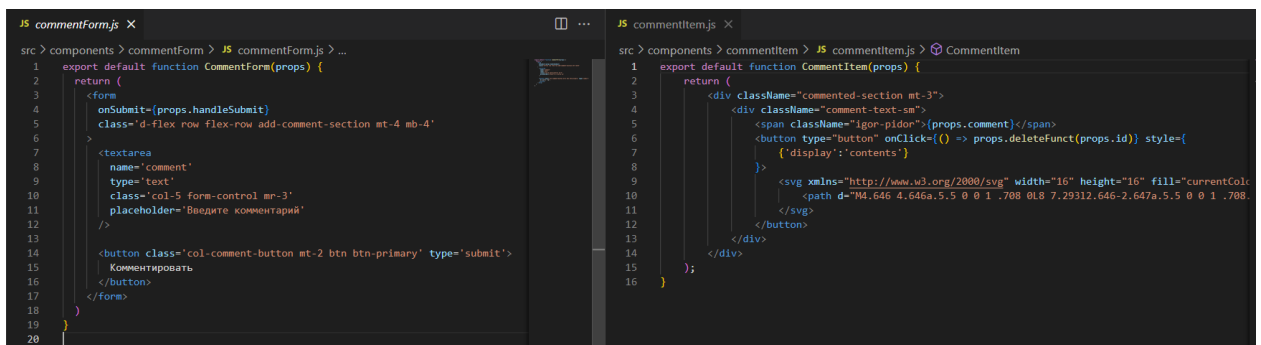
В случае, если комментарии отсутствуют - выводим сообщение, что комментариев пока нет.

```
JS commentSection.js X
src > components > commentSection > JS commentSection.js > CommentSection > comments.ma
1  import { useEffect, useState, useReducer } from 'react'
2  import CommentForm from '../commentForm/commentForm'
3  import CommentItem from '../commentItem/commentItem'
4
5  function getComments(movieId) {
6    const comments = localStorage.getItem(movieId)
7    return comments ? JSON.parse(comments) : []
8  }
9
10 export default function CommentSection(props) {
11   function handleSubmit(e) {
12     e.preventDefault()
13     const form = e.target
14     const comment = new FormData(form).get('comment')
15     form.reset()
16
17     if (comment == '') return
18     setComments([comment, ...comments])
19   }
20   function commentDelete(id) {
21     setComments(comments.filter((com, ind) => ind !== id))
22   }
23
24   let [comments, setComments] = useState(getComments(props.movieId))
25
26   useEffect(() => {
27     localStorage.setItem(props.movieId, JSON.stringify(comments))
28   })
29
30   return (
31     <div className='col-lg-12 col-md-12 col-sm-12'>
32       <div className='coment-bottom p-2 px-4'>
33         <CommentForm handleSubmit={handleSubmit}></CommentForm>
34         {comments.length > 0 ? (
35           comments.map((comment, index) => (
36             <CommentItem id={index} comment={comment} deleteFunc={commentDelete}
37             />
38           ))
39         ) : ( <p>Комментарии отсутствуют.</p> )}
40       </div>
41     </div>
42   )
43 }
```

Рисунок 10 – Код commentSection.js.

Сам код компонентов формы и комментария содержит только саму разметку (рисунок 11). На форму отправки комментария подписывается обработчик события onSubmit (происходит при нажатии на кнопку типа “submit” внутри формы), который мы получаем через объект props.

В самом комментарии отображается его содержимое, полученное через props и добавляется обработчик события удаления комментария для кнопки “удалить”.



**Рисунок 11** – Код commentForm.js и commentItem.js

Весь код стилей наследуется от фреймворка Bootstrap 5 с использованием его стандартных миксинов, переменных и точек останова адаптива(на рисунках 12-18). Для реализации всего выше описанного необходим фреймворк sass, однако для подключения стилей файлы расширения .scss не могут быть использованы, поэтому все стили компилируются в .css файлы с помощью плагина Live SaSS для VSCode.

```
global.scss X
src > styles > global.scss > $pagination-color
1 | $pagination-color: #f7bc46;
2 | $pagination-bg: #054673;
3 | $pagination-active-color: rgb(0, 0, 0);
4 | $pagination-active-bg: #f7bc46;
5 | $pagination-active-border-color: #f7bc46;
6 | $pagination-disabled-color: #054673;
7 | $pagination-disabled-bg: #e09b12;
8 | $pagination-disabled-border-color: #f7bc46;
9 |
10 | $table-color: rgb(136, 122, 122);
11 | $table-bg: transparent;
12 |
13 | $primary-text: #054673;
14 | $secondary-text: black;
15 | $primary-background: #094039;
16 | $secondary-background: #f7bc46;
17 | $border-color: $secondary-background;
18 |
19 | $font-size-lg: 3;
20 | $h1-font-size: 80px;
21 |
22 | @import 'C:/Users/igork/OneDrive/Рабочий стол/vik/node_modules/bootstrap/scss/bootstrap.scss';
23 |
24 | @font-face {
25 |   font-family: Gilroy;
26 |   src: url(/Gilroy-Light.otf);
27 | }
28 |
29 | .col-img {
30 |   width: 30%;
31 | }
32 |
33 | @include media-breakpoint-down(sm) {
34 |   * {
35 |     font-size: 0.9rem;
36 |   }
37 | }
38 |
39 | .col-foot-wide {
40 |   @extend .col-8;
41 | }
42 |
```

Рисунок 12 – global.scss

```
global.scss X
src > styles > global.scss > $pagination-color
39 .col-foot-wide {
40 |   @extend .col-8;
41 | }
42
43 .col-foot-small {
44 |   @extend .col-2;
45 | }
46
47 .col-foot-arrow {
48 |   @extend .col;
49 | }
50
51 * {
52 |   line-height: 1.3333333333;
53 |   font-family: Gilroy, helvetica neue, Helvetica, Arial, sans-serif;
54 | }
55
56 body {
57 |   background-color: $primary-background;
58 | }
59
60 .head-text {
61 |   color: #f7bc46;
62 | }
63
64 .button {
65 |   color: #054673;
66 | }
67
68 .center-logo {
69 |   margin: auto;
70 |   display: block;
71 |   max-width: 190px;
72 | }
73
74 .btn-primary {
75 |   background-color: $primary-background;
76 | }
77
78 .footer {
79 |   background-color: $primary-text;
80 | }
81
```

Рисунок 13 – global.scss

```
global.scss X
src > styles > global.scss > [e] $pagination-color
81
82 .header {
83   background-color: $primary-text;
84   display: block;
85 }
86
87 .card-title {
88   color: $primary-text;
89 }
90
91 .card-subtitle {
92   color: $primary-text;
93 }
94
95 .box-title {
96   color: $primary-text;
97 }
98
99 .card-body-text-detail {
100   color: $secondary-text;
101 }
102
103 .card-item {
104   @extend .card;
105   box-sizing: border-box;
106   border: 0px solid transparent;
107   z-index: 15;
108   overflow: hidden;
109   border-radius: 15px;
110   max-height: 40rem;
111 }
112
113 .text-body-secondary {
114   color: #f7bc46 !important;
115 }
116
```

Рисунок 14 – global.scss

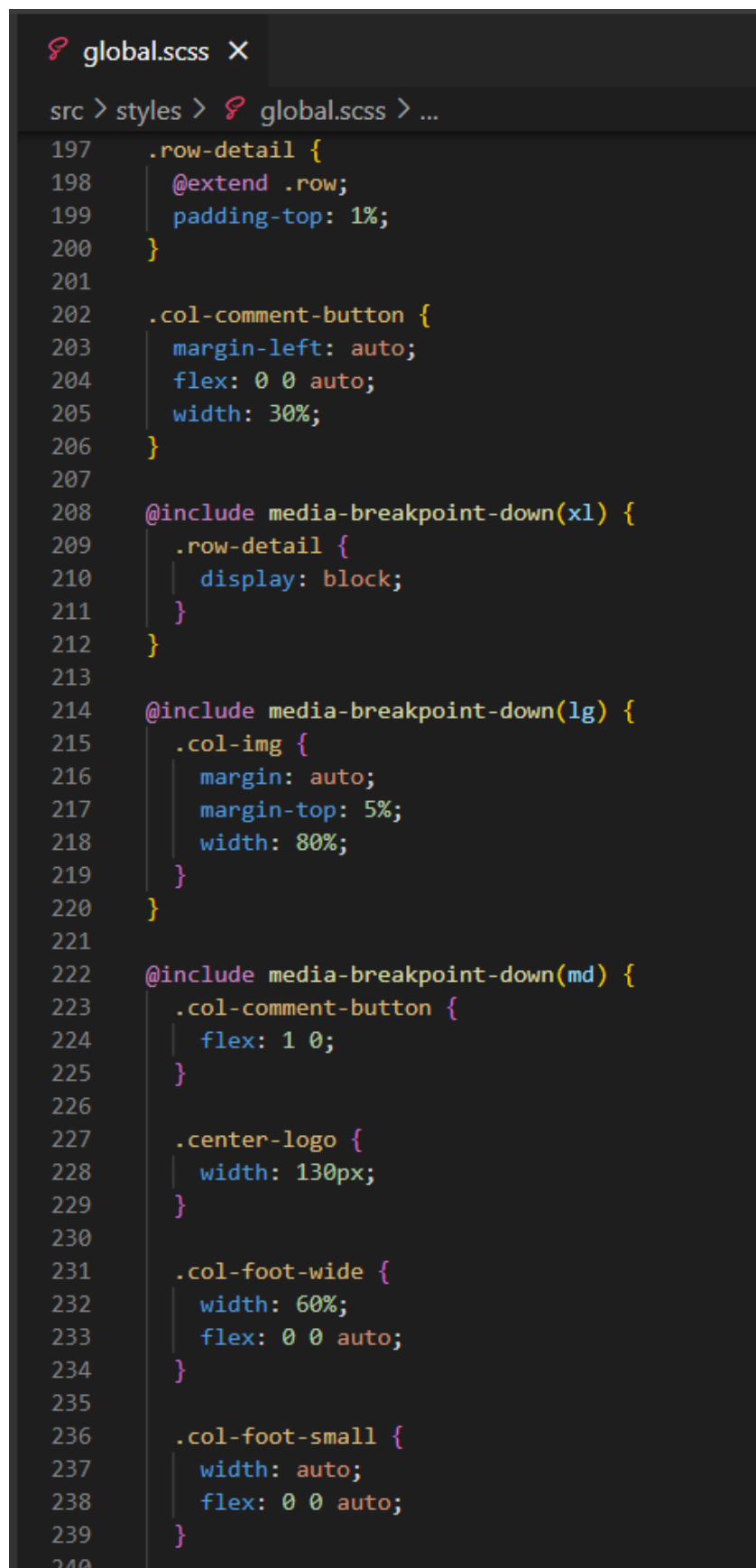


```
global.scss X
src > styles > global.scss > [🔍] $pagination-color
116
117 .card-item-body {
118   @extend .card-body;
119   border: 1px solid black;
120   border-top: none;
121   border-bottom-left-radius: inherit;
122   border-bottom-right-radius: inherit;
123   background-color: $secondary-background;
124   min-height: 0;
125 }
126
127 .card-item-title {
128   @extend .card-title;
129 }
130
131 .card-item-text {
132   @extend .card-text;
133   color: $secondary-text;
134   text-overflow: ellipsis;
135   overflow: hidden;
136   display: -webkit-box;
137   -webkit-line-clamp: 4;
138   -webkit-box-orient: vertical;
139 }
140
141 .card-item-img,
142 .card-item-img-top {
143   @extend .card-img, .card-img-top;
144   border-radius: 0px;
145 }
146
147 .card-detail {
148   @extend .card;
149   background-color: transparent;
150   border: none;
151 }
152
```

Рисунок 15 – global.scss

```
global.scss X
src > styles > global.scss > ...
153 @include media-breakpoint-down(lg) {
154   .card-item-text {
155     font-size: 1rem;
156   }
157
158   .card-item-title {
159     font-size: 1.5rem;
160   }
161 }
162
163 .nav-tabs .nav-link.active {
164   border-color: transparent;
165 }
166
167 .nav-tabs {
168   border-width: 0px;
169 }
170
171 .no-decoration {
172   text-decoration: none;
173   cursor: pointer;
174 }
175
176 .pagination {
177   width: fit-content;
178   margin: auto;
179   padding-top: 100px;
180 }
181
182 .card-detail {
183   @extend .card;
184 }
185
186 .container-detail {
187   @extend .container;
188   display: flex;
189   background-color: $secondary-background;
190 }
191
192 .card-body-detail {
193   @extend .card-body;
194   padding: 1%;
195 }
```

Рисунок 16 – global.scss



```
global.scss X
src > styles > global.scss > ...
197 .row-detail {
198   @extend .row;
199   padding-top: 1%;
200 }
201
202 .col-comment-button {
203   margin-left: auto;
204   flex: 0 0 auto;
205   width: 30%;
206 }
207
208 @include media-breakpoint-down(xl) {
209   .row-detail {
210     display: block;
211   }
212 }
213
214 @include media-breakpoint-down(lg) {
215   .col-img {
216     margin: auto;
217     margin-top: 5%;
218     width: 80%;
219   }
220 }
221
222 @include media-breakpoint-down(md) {
223   .col-comment-button {
224     flex: 1 0;
225   }
226
227   .center-logo {
228     width: 130px;
229   }
230
231   .col-foot-wide {
232     width: 60%;
233     flex: 0 0 auto;
234   }
235
236   .col-foot-small {
237     width: auto;
238     flex: 0 0 auto;
239   }
240 }
```

Рисунок 17 – global.scss

```

✓ .col-foot-arrow {
  width: 33.33333%;
  flex: 0 0 auto;
}

✓ .col-img {
  margin: auto;
  margin-top: 5%;
  width: 80%;
}
}

```

Рисунок 18 – global.scss

## Демонстрация работы веб-сайта

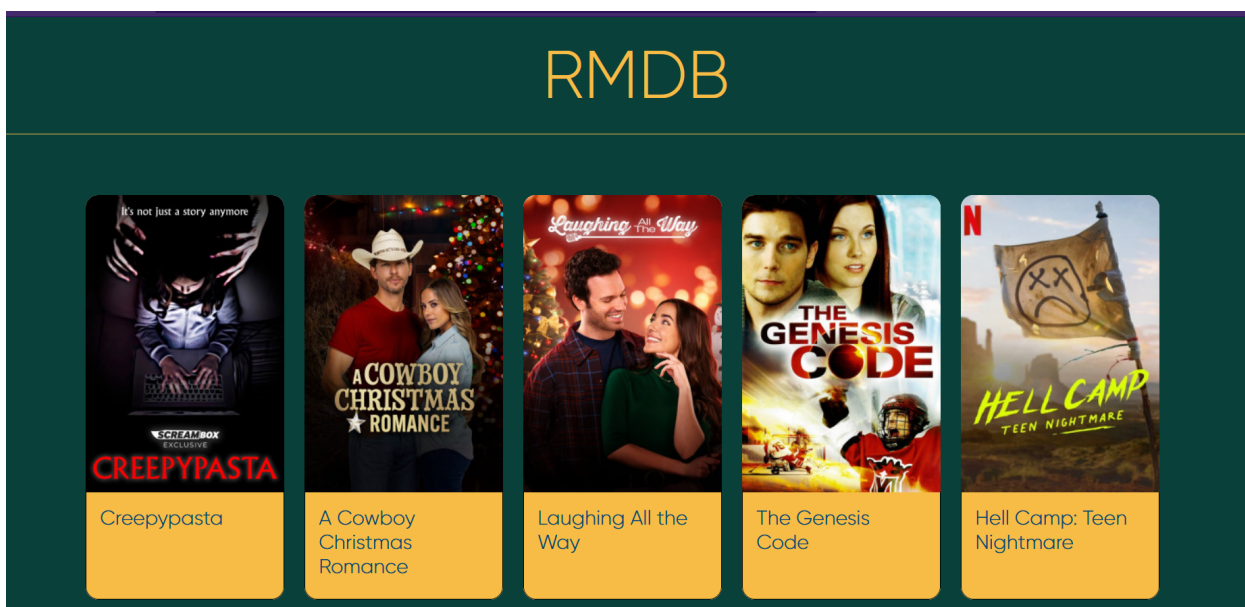


Рисунок 19 – Главная страница.

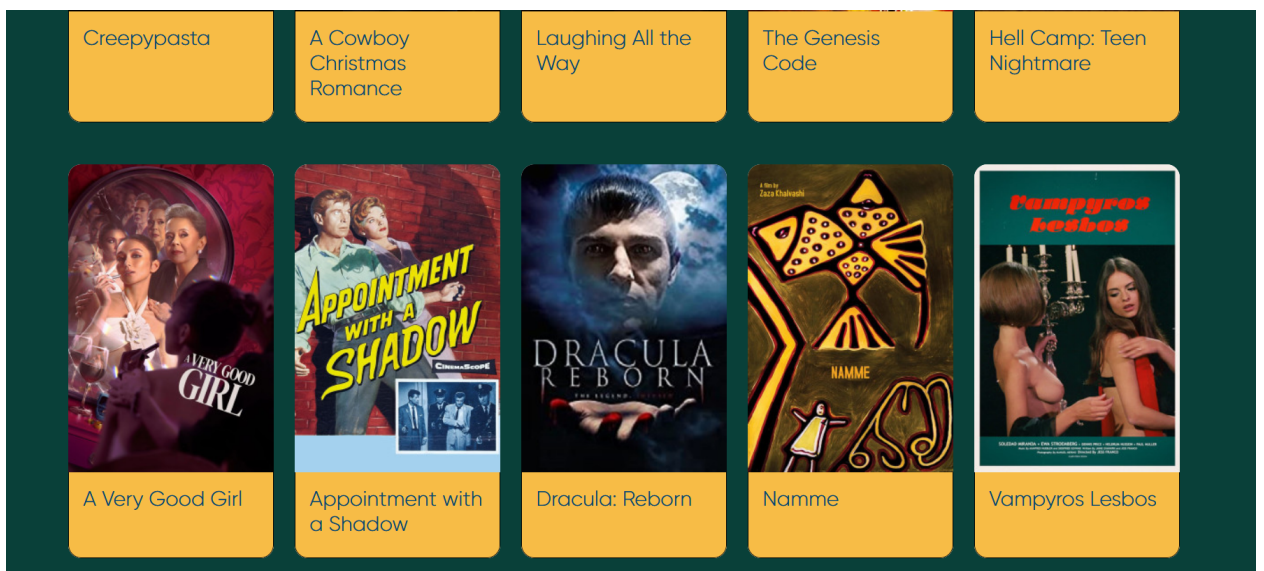


Рисунок 20 – Карточка фильма.

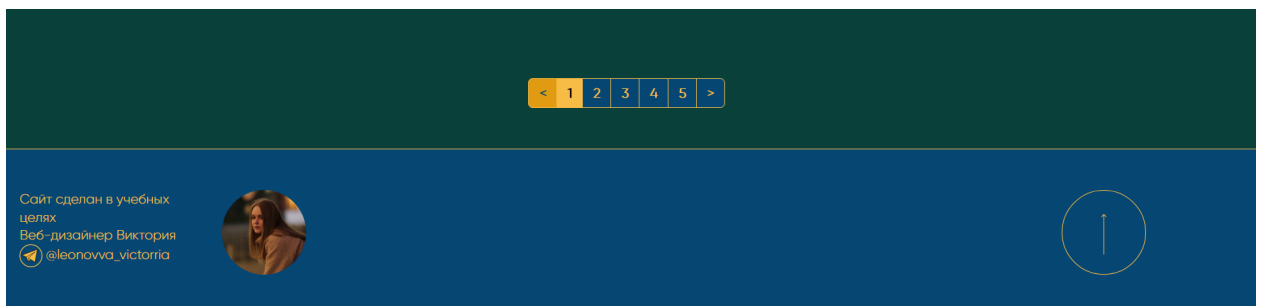


Рисунок 21 – Пагинация и футер.

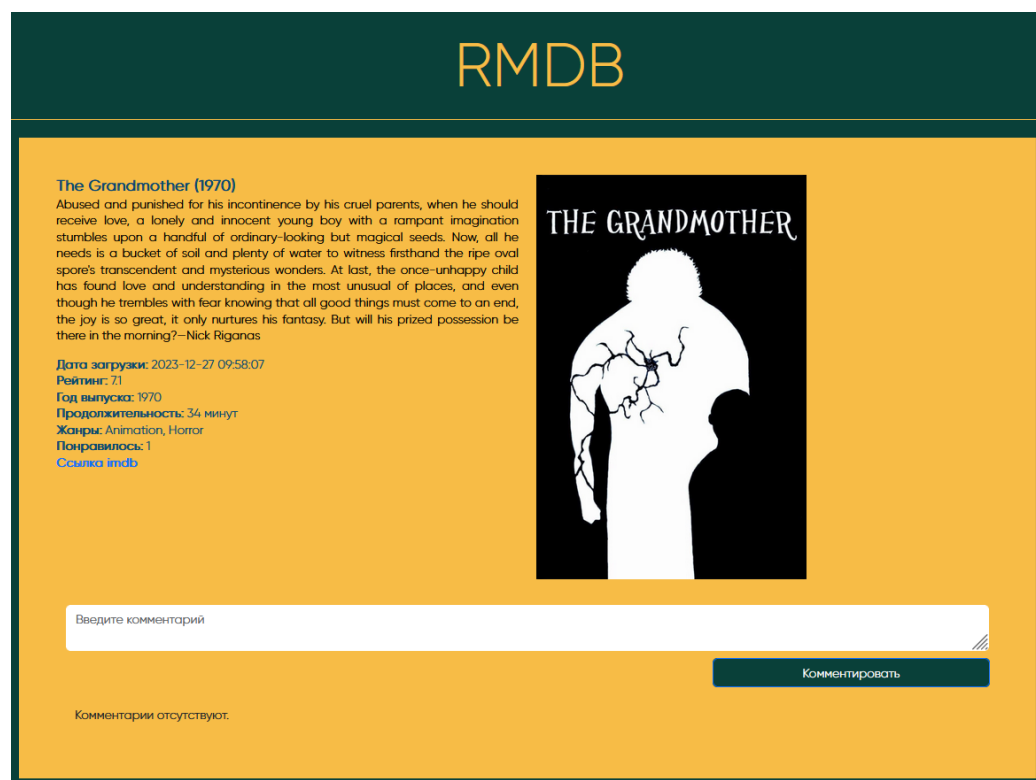


Рисунок 22 – Страница с фильмом.

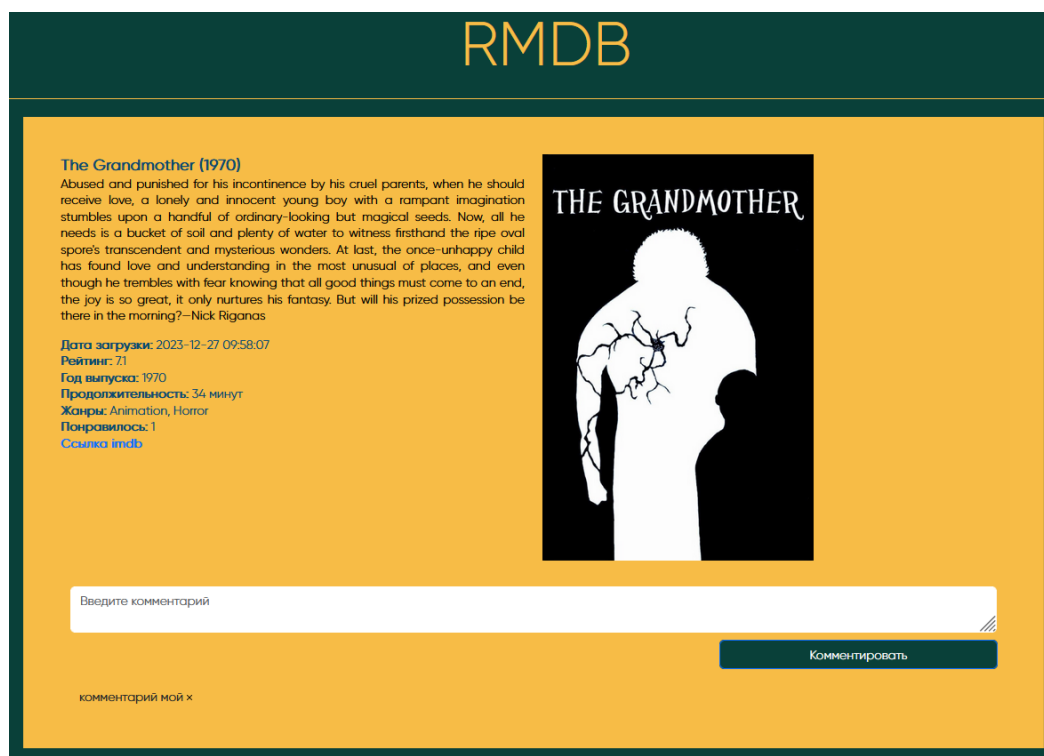


Рисунок 23 – Страница с фильмом и комментарием.

## **Вывод**

В данной работе была выполнена визуализация данных о фильмах, полученные из API, удобной таблице, предоставляя пользователям подробную информацию о каждом фильме.

Каждая карточка содержит описание, жанры, рейтинг, дату загрузки, год выпуска, продолжительность, количество лайков и ссылку на imdb.

Помимо этого есть возможность оставления комментариев и удаление их в любой момент.