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»

**Отчёт о выполнении лабораторных работ**

по дисциплине «UI/UX»

Выполнила студентка группы БВТ2001:

Бортникова Марина

Москва 2023

**Оглавление**

[Цель работы 3](#_Toc152636686)

[Задание 3](#_Toc152636687)

[Теоретическая часть 4](#_Toc152636688)

[Выполнение работы 5](#_Toc152636689)

[Получение данных с сервера. 8](#_Toc152636690)

[Реализованные компоненты 12](#_Toc152636691)

[Вывод 13](#_Toc152636692)

# **Цель работы**

Необходимо разработать SPA исходя из описания задачи.

# **Задание**

Имеется API со списком фильмов, с которым нужно наладить взаимодействие.

Необходимо визуализировать данные полученные от API в виде таблицы, содержащей только важную (на ваш взгляд) информацию о фильме.

К каждому фильму из таблицы можно оставить комментарий (неограниченное количество), а также удалить его в любой момент времени, локальное состояние.

Ссылка на документацию API: <https://yts.mx/api>

Методы, с которыми необходимо работать:

• List Movies;

• Retrieve Movie.

Требования к задаче:

• NextJS;

• Отсутствие перезагрузки страницы;

• В приложении должна присутствовать пагинация.

Правила хорошего тона:

• Адаптивная верстка;

• Приятный и удобный UI;

• Простой и лаконичный код;

• DRY.

# **Теоретическая часть**

Next.js — открытый JavaScript фреймворк, созданный поверх React.js для создания веб-приложений, созданный компанией Vercel (ранее ZEIT). Фреймворк был предназначен для решения проблемы React.js, связанной с отрисовкой приложения на стороне сервера - SSR. Работает на сервере и в браузере.

На JavaScript выполняется большая часть интерактивных элементов на сайтах и в мобильных приложениях. JavaScript отлично работает с HTML/CSS и интегрирован основные браузеры на рынке. Чистый JavaScript используется в вебе, а для общего применения JavaScript разработчики используют различные среды выполнения, например, Node.js.

Node.js — это среда выполнения кода JavaScript. Она позволяет использовать JavaScript как язык программирования общего назначения: создавать на нем серверную часть и писать полноценные десктопные приложения.

Основа Node.js — движок V8. Этот движок был разработан Google и используется в браузере Google Chrome. Он компилирует код JavaScript в машинный код, который понимает процессор. Однако, чтобы сделать из JavaScript язык общего назначения, одного движка недостаточно. Так, например, для создания серверной части нужно, чтобы язык умел работать с файлами, сетью и т.п. Для решения этой проблемы разработчики добавили к V8 дополнительные возможности, с помощью своего кода и сторонних библиотек. В итоге у них получился инструмент, который превращает JavaScript в язык общего назначения.

Node.js стала популярна среди разработчиков благодаря возможности создавать серверную и клиентскую часть на одном языке, скорости работы и NPM.

# **Выполнение работы**

Для выполнения работы был создан проект с помощью команды *npx create-next-app@latest* и был произведен выбор расширений для проекта. Проект выполнялся на языке TypeScript, в качестве стилизации использовался фреймфорк Tailwind. После установки всех пакетов, приложение было запущено на localhost:3000.

В ходе выполнения работы были реализованы следующие страницы:

1. Home страница с выводом списка фильмов в табличном представлении. Карточка фильма содержит информацию о названии, годе выхода и рейтинге фильма, а также постер фильма (рисунок 1). В конце страницы содержится пагинация (Рисунок 2)
2. Страница конкретного фильма. Переход на эту страницу реализован с помощью динамического роутинга при нажатии на карточку фильма. Эта страница содержит постер фильма, детальную информацию о фильме, описание, кнопку для добавления в избранное и комментарии к фильму (рисунки 3-4).
3. Страница любимых фильмов. Доступна по пути /favorites. Содержит карточки фильмов, помеченных как избранное.

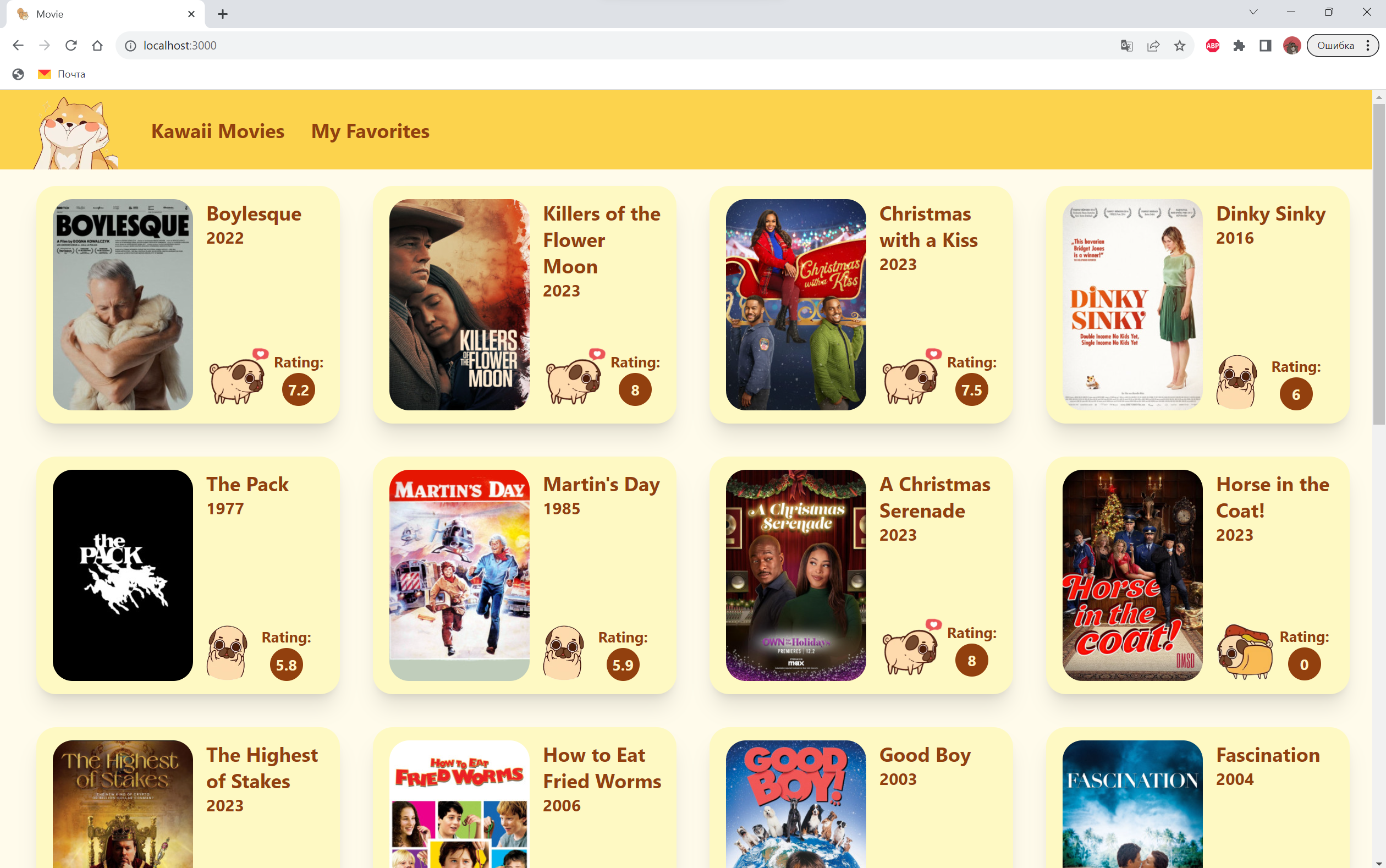


Рисунок 1 – Home page.

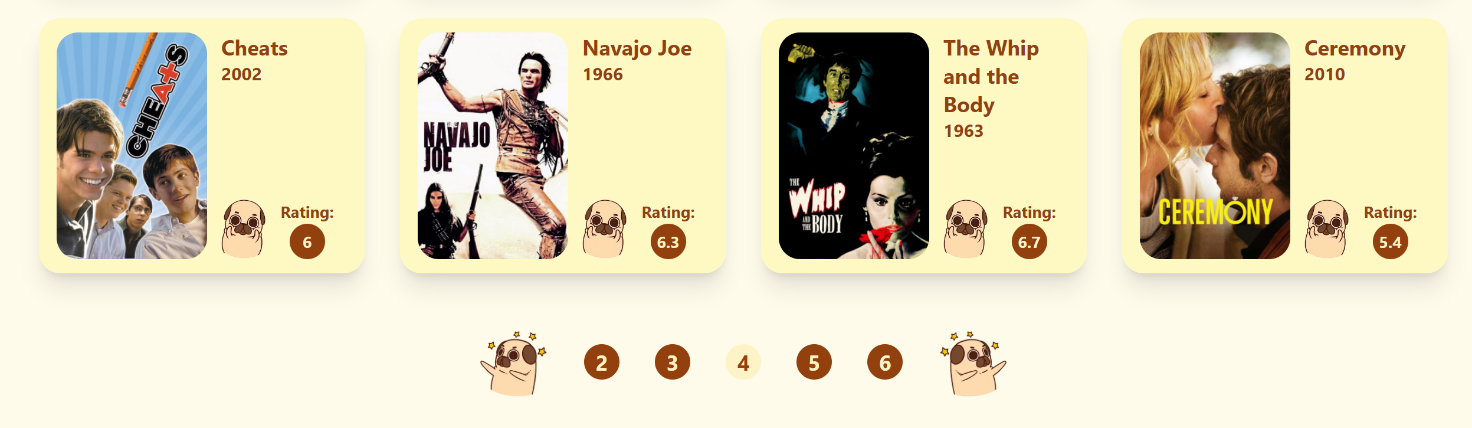


Рисунок 2 – Пагинация.

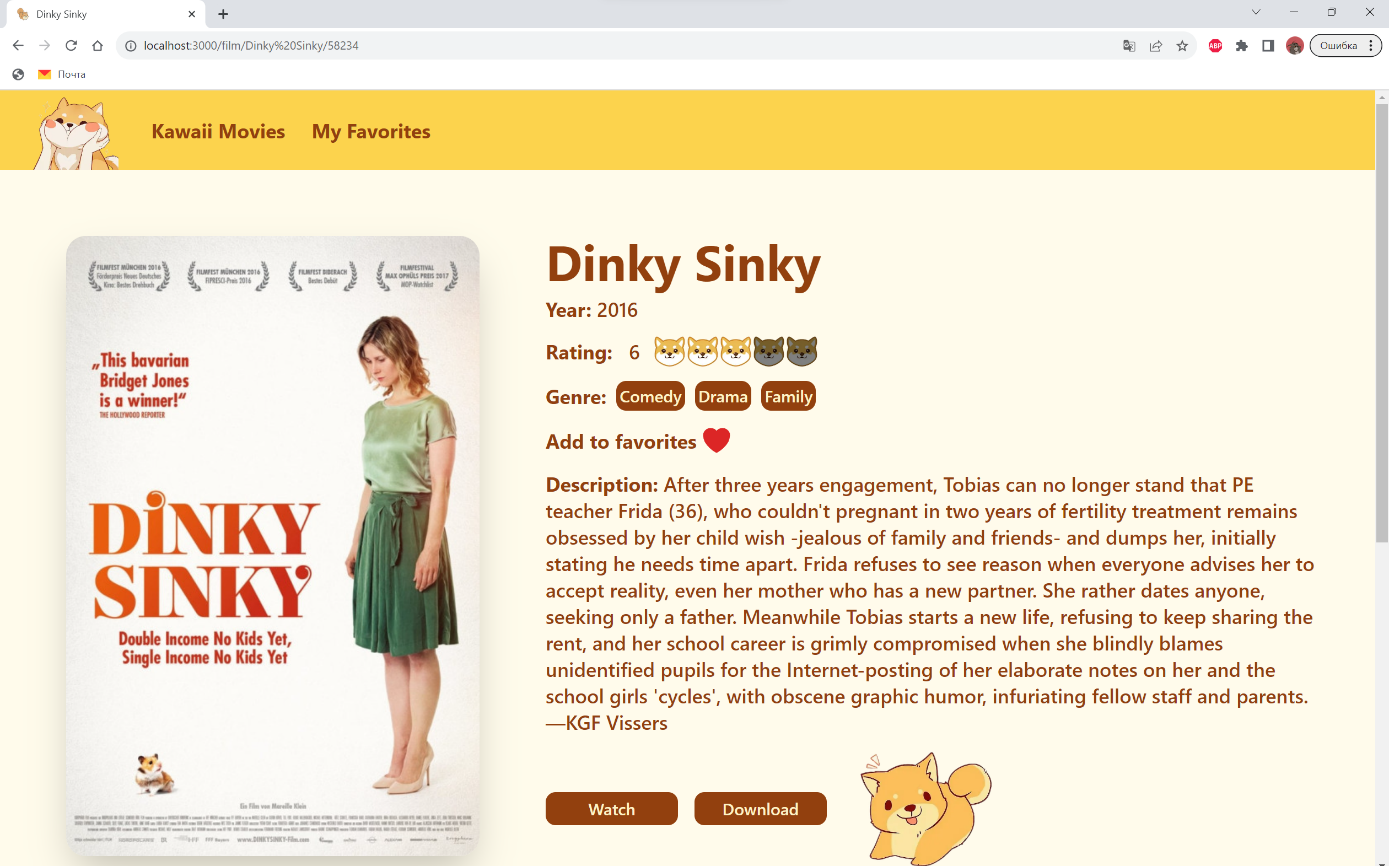


Рисунок 3 – Страница детальной информации о фильме.

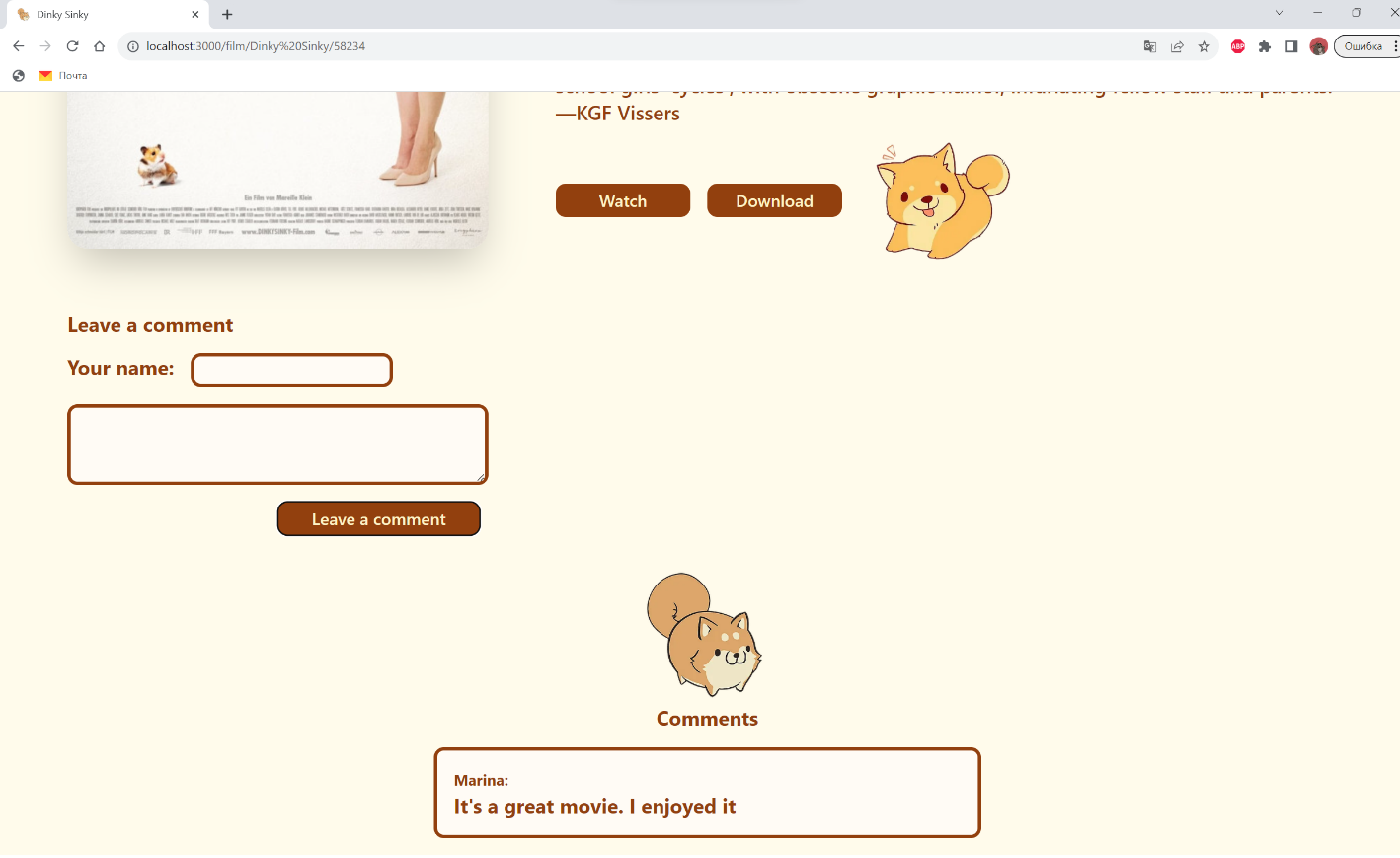


Рисунок 4 – Комментарии на странице фильма.

## **Получение данных с сервера.**

Для получения данных с сервера в папке services/movies.ts были созданы две функции getMovies и getFilm, использующие библиотеку axios. Функция getMovies получает список фильмов по URL 'https://yts.mx/api/v2//list\_movies.json?limit=24' с дополнительным параметром page, который в дальнейшем использовался для пагинации. Функция getFilm получает расширенную информацию об одном конкретном фильме и получает её по URL 'https://yts.mx/api/v2/movie\_details.json' используя дополнительный параметр movie\_id. Код функций представлен на рисунке 5.

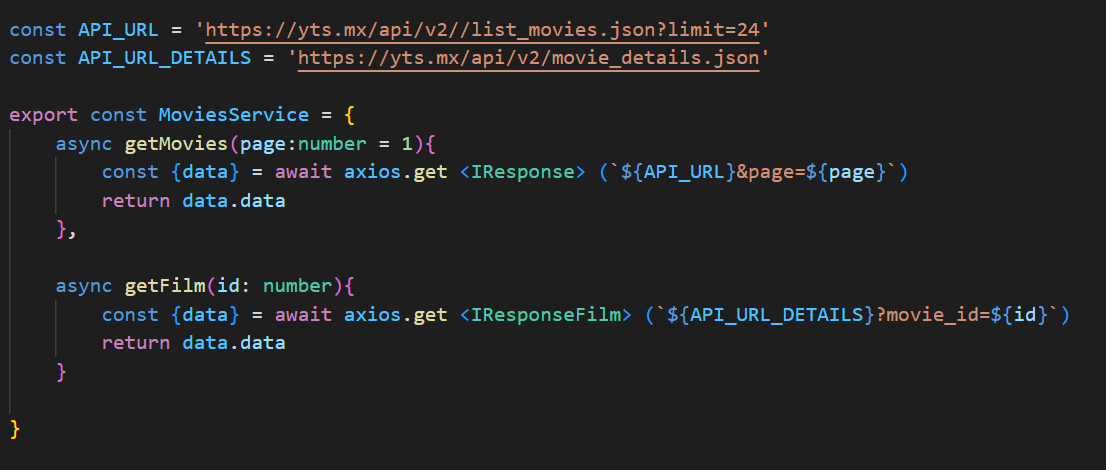


Рисунок 5 – функции для получения данных с сервера.

Далее функция getMovies использовалась в методе getStaticProps домашней страницы index.tsx для получения списка из 24 фильмов и номера страницы. Метод getStaticProps позволяет получать данные во время сборки. После получения он передает полученные данные через props в компоненту Movies.tsx. Код метода и передача в компоненту Movies представлен на рисунке 6.

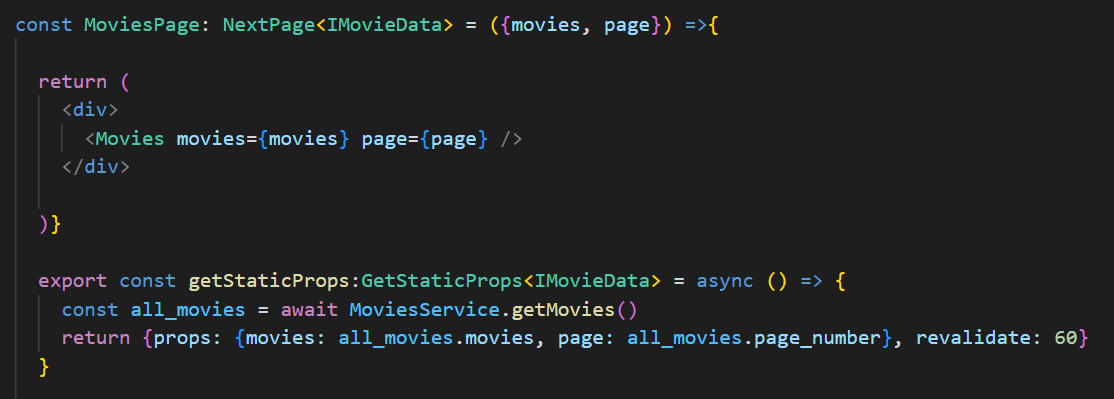


Рисунок 6 – код файла index.tsx

Для реализации пагинации используется метод getServerSideProps, в котором также используется метод getMovies. Метод getServerSideProps получает из context.params номер страницы, которая предварительно изменяется в URL по кнопкам элемента пагинации. Далее с номером полученный страницы в качестве параметров происходит вызов функции getMovies и данные передаются в качестве Props в компоненту Movies. Код запроса данных при пагинации представлен на рисунке 7.

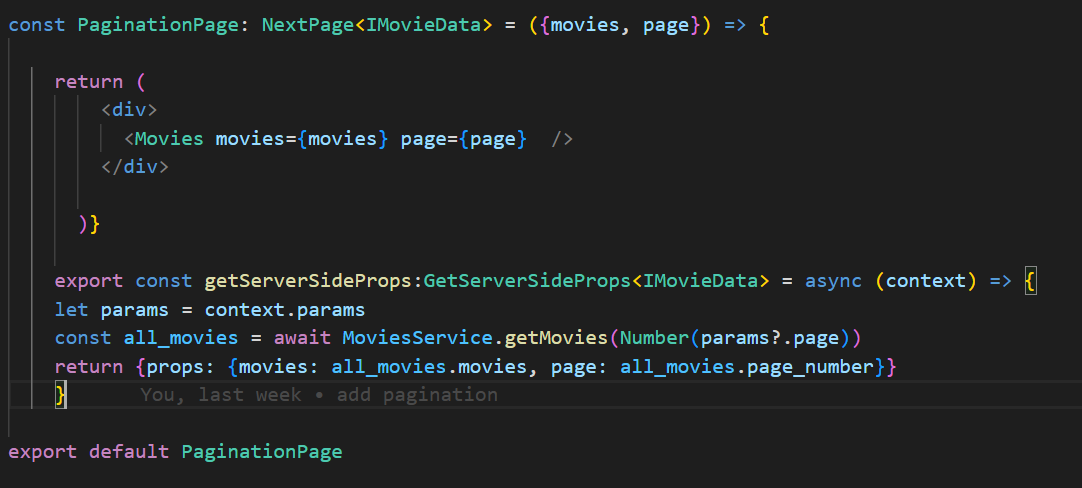


Рисунок 7 – Код динамической страницы пагинации [page].tsx

Для реализации перехода на станицу фильма используется метод getStaticProps в связке с методом getStaticPaths. Метод getStaticPaths позволяет указывать динамические маршруты для предварительной визуализации страниц. При нажатии на карточку фильма URL страницы меняется по шаблону: /film/*название-фильма/id-фильма.* МетодgetStaticPaths получает название и id из параметров, а метод getStaticProps использует функцию getFilm, описанную выше, с полученными параметрами для получения данных о конкретном фильме и передает их в компоненту DetailPage. Код динамической страницы представлен на рисунке 8.



Рисунок 8 – Код динамической страницы /film/[title]/[id].tsx

Для реализации страницы избранных фильмов была создана компонента FavoritePage, которая с помощью метода getLocalStorage, описанном в файле services/storage.services.ts получает данный из LocalStorage по ключу favorites. Создает массив из запросов на каждый id фильма и переадет полученные данные в компоненту Movies. Код компоненты FavoritesPage представлен на рисунке 9.



Рисунок 9 – Код страницы favorites.tsx

## **Реализованные компоненты**

Главные компоненты страниц:

1. Компонента Home-страницы Movies.tsx. Получает данные о фильмах и странице и для каждого фильма из списка вызывает компоненту MovieItem с передачей ей данных для отрисовки. Также вызывает компоненту Pagination с передачей ей данных о странице.
2. Компонента страницы DetailsPage. Производит отрисовку детальной информации о фильмах. Вызывает компоненту FaforiteIcon с передачей ей данных об id фильма и компоненту Comments

Вспомогательные компоненты:

1. Компонента Navbar – производит отрисовку ссылок на главную страницу и страницу избранного в верхней части окна.
2. Компонента MovieItem – производит отрисовку карточки фильма, которая отображается на старнице Movies. При нажатии на карточку меняет URL для перехода на станицу детальной информации.
3. Компонента Pagination – производит отрисовку пагинации. При нажатии на элемент пагинации меняет URL для перехода на динамическую страницу Movies.
4. Компонента Comments – производит отрисовку функционала по добавлению комментария. С помощью методов getLocalStorage, setLocalStorage формирует объект комментария и добавляет его в массив комментариев по ключу id фильма в Local Storage. Для каждого комментария из массива в Local Storage вызывает компоненту CommetItem.
5. Компонента CommentItem – производит отрисовку каждого комментария из полученных данных о массиве комментариев.
6. Компонента FavoriteIcon – производит отрисовку кнопки добавления в избранное. С помощью методов getLocalStorage, setLocalStorage добавляет или удаляет id фильма в Local Storage по ключу favorites.

# **Вывод**

В ходе данной работы было реализовано SPA- приложение, которое работает с API, содержащим список фильмов. Проект был выполнен с помощью Next.js. Приложение визуализирует данные полученные от API в виде таблицы. Производит переход на детальную информацию о фильме с возможностью оставить комментарии и добавить фильм в список избранных. Также была реализована пагинация и адаптивная верстка.