**ДЕПАРТАМЕНТ ОБРАЗОВАНИЯ И НАУКИ КОСТРОМСКОЙ ОБЛАСТИ**

**ОБЛАСТНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ПРОФЕССИОНАЛЬНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ «КОСТРОМСКОЙ ЭНЕРГЕТИЧЕСКИЙ ТЕХНИКУМ ИМ. Ф.В. ЧИЖОВА»**

Химико-технологическое отделение

Специальность 09.02.07 «Информационные системы и программирование»

МДК 05.02. Разработка кода информационных систем

**КУРСОВАЯ РАБОТА**

**Тема: «Разработка кода информационной системы для работы Видеосалона»**

**Выполнил**:

Студент 3 курса, группы 3-1ИС

Савельев Константин Олегович

**Проверил**: Ищук Денис Николаевич

Дата \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Оценка\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Подпись преподавателя\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Подпись студента \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2024

Оглавление

[Глоссарий 4](#_Toc170244432)

[Введение 6](#_Toc170244433)

[Анализ предметной области 6](#_Toc170244434)

[Разработка системы 9](#_Toc170244435)

[Список функционала 9](#_Toc170244436)

[Activity системы 9](#_Toc170244437)

[Wireframe системы 11](#_Toc170244438)

[Метод реализации системы 11](#_Toc170244439)

[Архитектура веб приложения 11](#_Toc170244440)

[Протокол HTTP 11](#_Toc170244441)

[HTTP и RESTful веб-сервисы 12](#_Toc170244442)

[Ресурс 12](#_Toc170244443)

[URI ресурса 12](#_Toc170244444)

[REST и Ресурсы 12](#_Toc170244445)

[Компоненты HTTP 13](#_Toc170244446)

[Методы HTTP-запроса 13](#_Toc170244447)

[Код статуса ответа HTTP 13](#_Toc170244448)

[База данных системы 13](#_Toc170244449)

[Процесс разработки 14](#_Toc170244450)

[Файловая структура проекта 14](#_Toc170244451)

[Подключение к БД 15](#_Toc170244452)

[Класс пользователь 15](#_Toc170244453)

[Класс фильм 16](#_Toc170244454)

[Класс аренда 19](#_Toc170244455)

[Класс покупка 20](#_Toc170244456)

[Сессия пользователя 22](#_Toc170244457)

[Авторизация 22](#_Toc170244458)

[Выход из аккаунта 23](#_Toc170244459)

[Регистрация пользователя 24](#_Toc170244460)

[Действия с фильмами 25](#_Toc170244461)

[Содержимое профиля пользователя 29](#_Toc170244462)

[Документация API 31](#_Toc170244463)

[Проверка текущей сессии 31](#_Toc170244464)

[Авторизация на сайте 31](#_Toc170244465)

[Регистрация пользователя 32](#_Toc170244466)

[Действия с фильмами 33](#_Toc170244467)

[Тесты системы 36](#_Toc170244468)

[Внедрение 36](#_Toc170244469)

[Руководство администратора 36](#_Toc170244470)

[Руководство пользователя 36](#_Toc170244471)

[Заключение 36](#_Toc170244472)

[Ссылки 37](#_Toc170244473)

# Глоссарий

HTML — стандартизированный язык гипертекстовой разметки документов для просмотра веб-страниц в браузере. Веб-браузеры получают HTML документ от сервера по протоколам HTTP/HTTPS или открывают с локального диска, далее интерпретируют код в интерфейс, который будет отображаться на экране монитора.

CSS — это формальный язык, который используется для описания и оформления внешнего вида веб-страниц.

JS — Мультипарадигменный язык программирования. Поддерживает объектно-ориентированный, императивный и функциональный стили.

JQuery — Бесплатная JavaScript-библиотека с открытым исходным кодом. Она заметно упрощает веб-разработку, позволяя писать меньше кода, чем на ванильном JavaScript.

AJAX — «асинхронный JavaScript и XML» — подход к построению интерактивных пользовательских интерфейсов веб-приложений, заключающийся в «фоновом» обмене данными браузера с веб-сервером.

PHP — Интерпретируемый скриптовый язык программирования общего назначения.

API — это набор способов и правил, по которым различные программы общаются между собой и обмениваются данными.

REST расшифровывается как REpresentational State Transfer. Это был термин, первоначально введен Роем Филдингом (Roy Fielding), который также был одним из создателей протокола HTTP. Отличительной особенностью сервисов REST является то, что они позволяют наилучшим образом использовать протокол HTTP.

Вайрфрейм — это схема высшего уровня, иллюстрирующая структуру сайта, приложения или проекта.

Figma — онлайн-сервис для разработки интерфейсов и прототипирования с возможностью организации совместной работы в режиме реального времени.

UML (от англ. Unified Modeling Language) переводится как «унифицированный язык моделирования». Это графический язык, в котором каждой фигуре, символу, стрелке или их сочетаниям присвоены конкретные значения. Он позволяет визуализировать явление или процесс так, чтобы схема была понятна всем, кто знаком с UML.

PlantUML - это универсальный инструмент, позволяющий быстро и просто создавать широкий спектр диаграмм. Используя простой и интуитивно понятный язык, пользователи могут без труда создавать различные типы диаграмм.

База данных — это упорядоченный набор структурированной информации или данных, которые обычно хранятся в электронном виде в компьютерной системе.

Open Server Panel — это портативная программная среда, созданная специально для веб-разработчиков с учётом их рекомендаций и пожеланий.

phpMyAdmin — веб-приложение с открытым кодом, написанное на языке PHP и представляющее собой веб-интерфейс для администрирования СУБД MySQL.

Руководство администратора – это составная часть эксплуатационной документации, которая разрабатывается на любую программу или автоматизированную систему. При помощи руководства администратора ответственные пользователи системы получают возможность управлять ее функционированием – выполнять определенные операции по обеспечению порядка работы АСУ, распределять права доступа к ней, редактировать данные и исправлять ошибки.

Руководство пользователя – это основной документ в составе эксплуатационной документации на автоматизированную систему

# Введение

Изначально видеосалоны были созданы для просмотра фильмов на большом экране, но с появлением видеомагнитофонов они превратились в домашние кинозалы. В этих салонах люди могли арендовать или купить видеокассеты с фильмами, а затем смотреть их у себя дома.

Видеосалоны стали популярными среди молодых людей, которые хотели посмотреть новые фильмы без необходимости идти в кинотеатр. Они также были популярны среди тех, кто не мог позволить себе купить домашний видеомагнитофон или кассеты с фильмами.

Однако к середине 1990–х годов видеосалоны начали исчезать из–за появления кабельных и спутниковых телевизионных услуг, которые предлагали широкий выбор фильмов и программ. Кроме того, интернет и онлайн–кинотеатры также стали популярными, что еще больше уменьшило популярность видеосалонов.

Сегодня видеосалоны все еще существуют, но они стали редкими и часто находятся в небольших городах или сельских районах. Многие из них превратились в интернет–кафе или клубы, где люди могут смотреть фильмы

Разработка информационной системы позволит оптимизировать работу видеосалона, сократить временные и финансовые затраты, повысить удовлетворенность клиентов и улучшить качество предоставляемых услуг.

Целью данного проекта является разработка информационной системы для видеосалона, которая будет автоматизировать процесс покупки и аренды фильмов а так же управление фильмами находящимися в системе.

Архитектура системы реализована по Rest API. Для реализации приложения использовались: PHP для серверной части, JavaScript и библиотека JQuery для клиентской части приложения.

## Анализ предметной области

Для анализа своей предметной области я буду использовать метод анализ данных, так как нам нужно идентификация и описание данных, необходимых для функционирования системы.

Для своей предметной области были исследованы следующие сайты:

1. Бургерная–видеосалон [Гамбургеры | chef.ru](https://chef.ru/place/gamburgery/?ysclid=ls1gb64nck297366140)

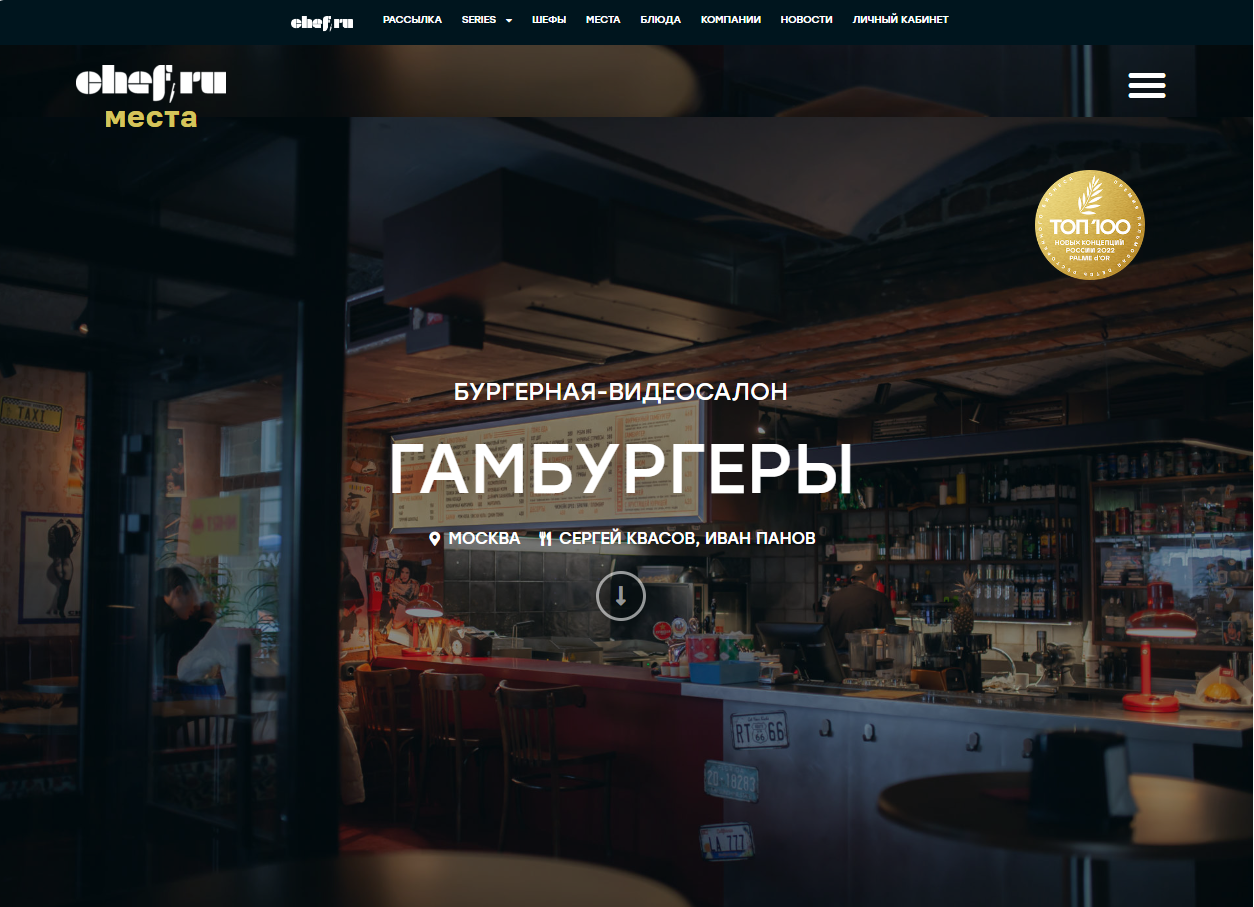


Рисунок 1 сайт для анализа

Достоинства этого видеосалона:

* Сайт встречает приятным и удобным пользовательским интерфейсом.  С первого раза пользователь поймет, что и где расположено.
* Качественные фотографии товаров.
* Красивые карточки товаров.
* Есть регистрация и личный кабинет.
* Сайт адаптирован для телефонов.

Недостатки этого видеосалона:

* Некоторые кнопки не подсвечиваются и не понятно можно ли нажать на неё или нет.

1. PSN coll game [PSNCOOLGAME | Цифровой прокат игр для PS5–PS4–VR | Каталог товара](https://psncoolgame.com/?ysclid=ls1gvj4w8g407144618)

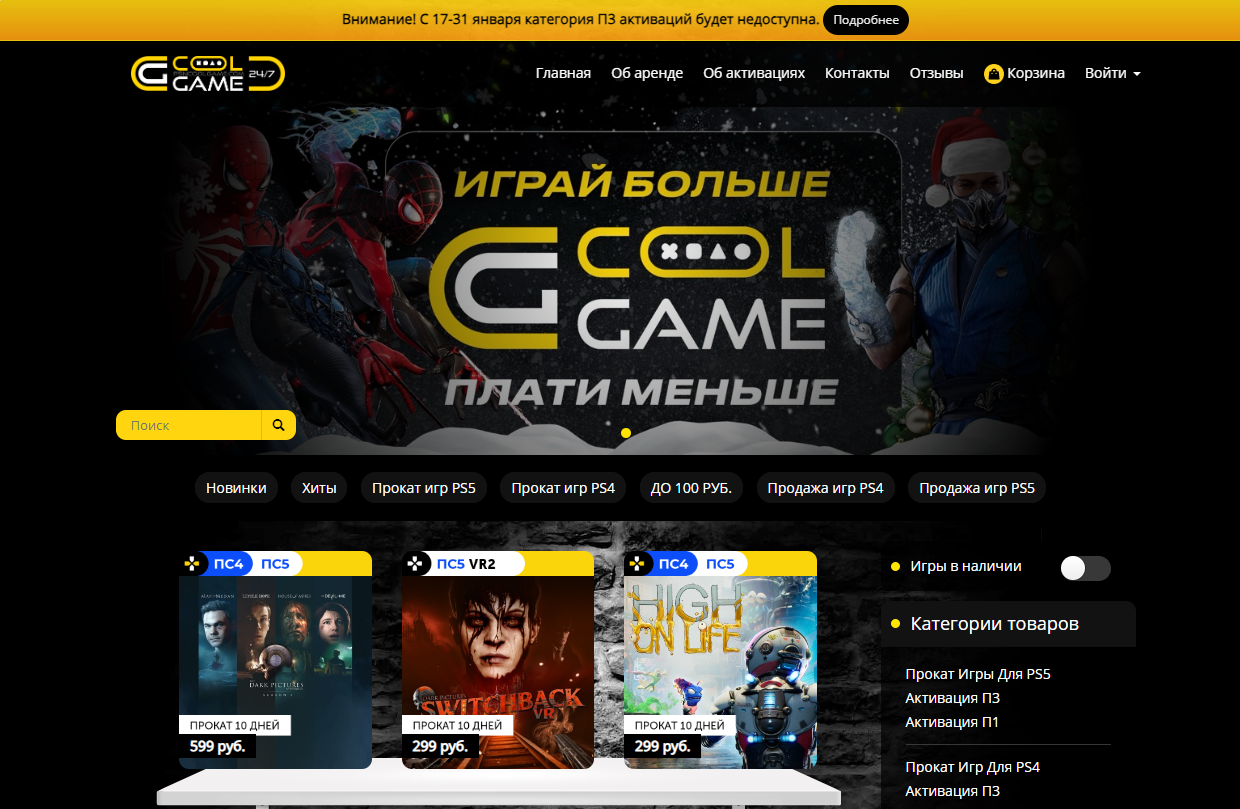


Рисунок 2 сайт для анализа

Достоинства этого видеосалона:

* Сайт встречает приятным и удобным пользовательским интерфейсом.  С первого раза пользователь поймет, что и где расположено.
* Качественные фотографии товаров.
* Красивые карточки товаров.
* Есть регистрация и личный кабинет.
* Сайт адаптирован для телефонов.

Недостатки этого видеосалона:

* много нивидимых гиперссылок

Предметная область видеосалона охватывает услуги по прокату видеофильмов и компьютерных игр. Пользователи могут арендовать фильмы на определенный срок, после чего возвращать их в видеосалон

Анализ предметной области видеосалона может включать следующие аспекты:

* Клиенты: видеосалоны обычно обслуживают широкую аудиторию клиентов, включая людей разных возрастных групп и интересов. Важно понять предпочтения и потребности клиентов, чтобы предложить им наиболее интересные и актуальные фильмы.
* Каталог: видеосалон должен иметь обширный каталог фильмов, включая различные жанры и форматы. Важно постоянно обновлять каталог новыми релизами и предлагать классические фильмы.
* Условия проката: видеосалон может предлагать различные условия проката, такие как аренда на определенное количество дней, возможность продления аренды или возможность получить скидку при аренде нескольких фильмов. Важно установить разумные и привлекательные цены и условия проката.
* Техническое оборудование: видеосалон должен обладать соответствующим техническим оборудованием, чтобы клиенты могли просматривать фильмы. Это включает DVD–плееры.
* Конкурентная среда: важно изучить конкурентов в данной сфере, чтобы определить свои преимущества и выделиться на рынке. Это может включать анализ цен, условий проката, качества обслуживания и доступности определенных фильмов.
* Маркетинг и продвижение: видеосалон может использовать различные маркетинговые стратегии для привлечения клиентов, такие как реклама в местных СМИ, социальные сети, программы лояльности или сотрудничество с другими компаниями. Важно определить наиболее эффективные способы привлечения и удержания клиентов.

# Разработка системы

## Список функционала

Функционал доступный неавторизированному пользователю:

* Авторизация на сайте
* Регистрация на сайте

Функционал доступный авторизированному пользователю:

* Просмотр каталога фильмов доступных на сайте
* Просмотр акций проходящих на данный момент
* Покупка фильма
* Аренда фильма
* Просмотр списка купленных или взятых в аренду фильмов
* Выход из аккаунта

Функционал доступный администратору:

* Добавление нового фильма
* Редактирование уже существующего фильма
* Удаление фильма

## Activity системы

Для для построения диаграммы Activity использовался плагин PlantUML для VScode

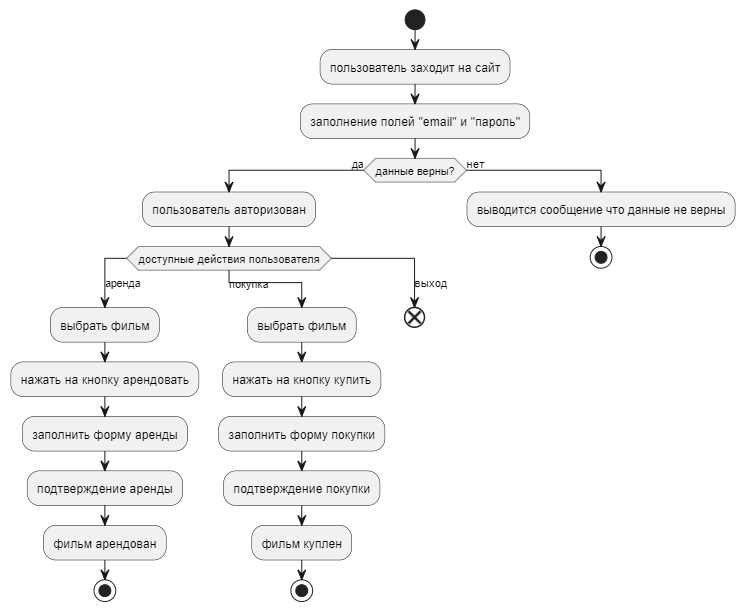


Рисунок 3 Activity диаграмма

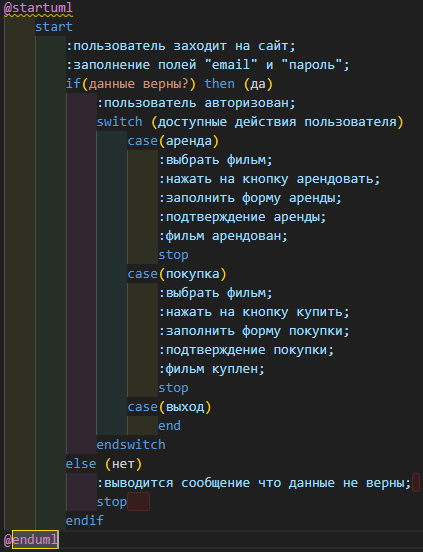


Рисунок 4 код Activity диаграммы

### Wireframe системы

Для создания wireframe использовался редактор figma

Wireframe ИС для видеосалона. (б.д.). Получено из <https://www.figma.com/design/STPWWitKppil6SpPsTYeQx/Untitled?node-id=0-1&t=9mnBJACKNrd0GidW-1>

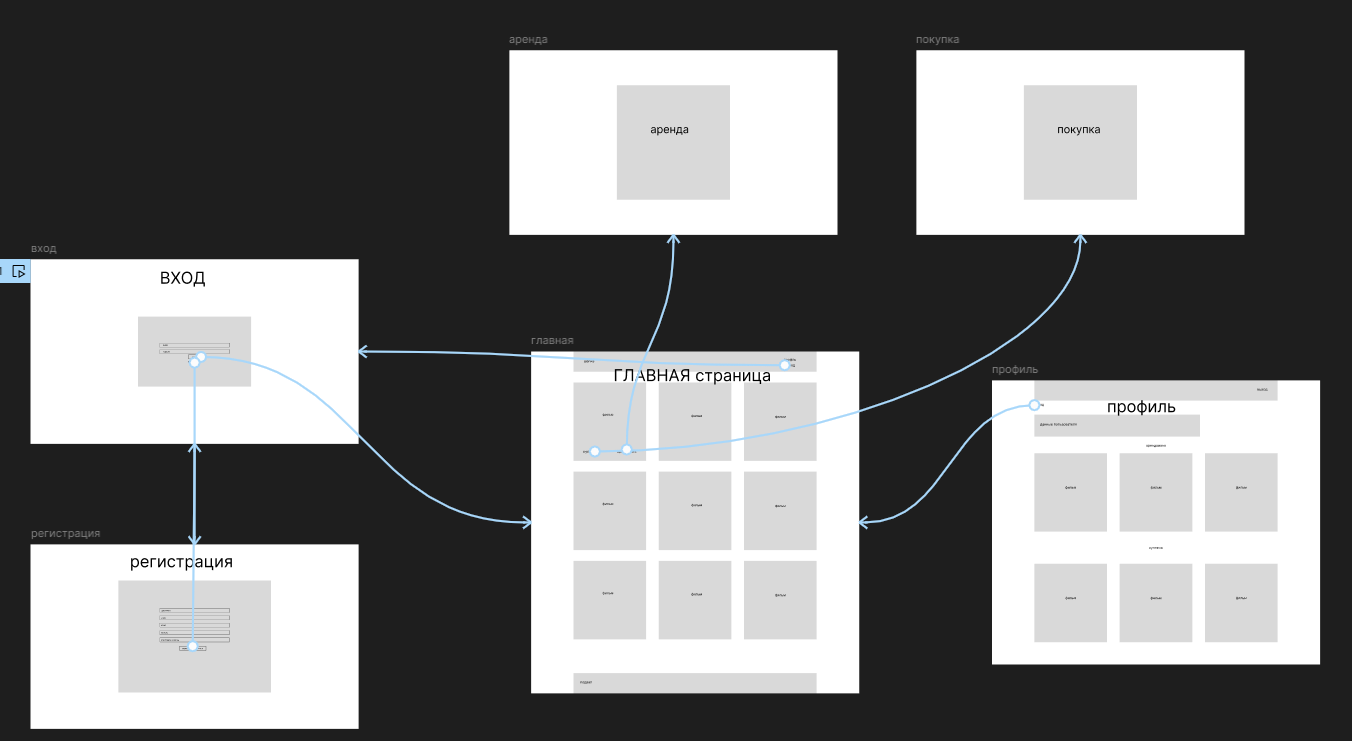


Рисунок 5 Wireframe

## Метод реализации системы

### Архитектура веб приложения

REST расшифровывается как REpresentational State Transfer. Это был термин, первоначально введен Роем Филдингом (Roy Fielding), который также был одним из создателей протокола HTTP. Отличительной особенностью сервисов REST является то, что они позволяют наилучшим образом использовать протокол HTTP.

*REST API — RESTful веб-сервисы*. (б.д.). Получено из <https://habr.com/en/articles/483202/>

Когда мы просматриваем страницы в Интернете, за кулисами происходит много вещей. Ниже приведено упрощенное представление о том, что происходит между браузером и серверами, работающими на посещаемых веб-сайтах:



Рисунок 6 принцип RestAPI

### Протокол HTTP

Когда вы вводите в браузере URL-адрес, например [www.google.com](https://www.google.com/), на сервер отправляется запрос на веб-сайт, идентифицированный URL-адресом. Затем этот сервер формирует и выдает ответ. Важным является формат этих запросов и ответов. Эти форматы определяются протоколом HTTP — Hyper Text Transfer Protocol.

Когда вы набираете URL в браузере, он отправляет запрос GET на указанный сервер. Затем сервер отвечает HTTP-ответом, который содержит данные в формате HTML — Hyper Text Markup Language. Затем браузер получает этот HTML-код и отображает его на экране.

Допустим, вы заполняете форму, присутствующую на веб-странице, со списком элементов. В таком случае, когда вы нажимаете кнопку «Submit» (Отправить), HTTP-запрос POST отправляется на сервер.

### HTTP и RESTful веб-сервисы

HTTP обеспечивает базовый уровень для создания веб-сервисов. Поэтому важно понимать HTTP. Вот несколько ключевых абстракций.

### Ресурс

Ресурс — это ключевая абстракция, на которой концентрируется протокол HTTP. Ресурс — это все, что вы хотите показать внешнему миру через ваше приложение. Например, если мы пишем приложение для управления задачами, экземпляры ресурсов будут следующие:

* Конкретный пользователь
* Конкретная задача
* Список задач

### URI ресурса

Когда вы разрабатываете RESTful сервисы, вы должны сосредоточить свое внимание на ресурсах приложения. Способ, которым мы идентифицируем ресурс для предоставления, состоит в том, чтобы назначить ему URI — универсальный идентификатор ресурса.

Например:

* Создать пользователя: POST /users
* Удалить пользователя: DELETE /users/1
* Получить всех пользователей: GET /users
* Получить одного пользователя: GET /users/1

### REST и Ресурсы

Важно отметить, что с REST вам нужно думать о приложении с точки зрения ресурсов:

* Определите, какие ресурсы вы хотите открыть для внешнего мира
* Используйте глаголы, уже определенные протоколом HTTP, для выполнения операций с этими ресурсами.

Вот как обычно реализуется служба REST:

* Формат обмена данными: здесь нет никаких ограничений. JSON — очень популярный формат, хотя можно использовать и другие, такие как XML
* Транспорт: всегда HTTP. REST полностью построен на основе HTTP.
* Определение сервиса: не существует стандарта для этого, а REST является гибким. Это может быть недостатком в некоторых сценариях, поскольку потребляющему приложению может быть необходимо понимать форматы запросов и ответов. Однако широко используются такие языки определения веб-приложений, как WADL (Web Application Definition Language) и Swagger.

REST фокусируется на ресурсах и на том, насколько эффективно вы выполняете операции с ними, используя HTTP.

### Компоненты HTTP

HTTP определяет следующую структуру запроса:

* строка запроса (request line) — определяет тип сообщения
* заголовки запроса (header fields) — характеризуют тело сообщения, параметры передачи и прочие сведения
* тело сообщения (body) — необязательное

HTTP определяет следующую структуру ответного сообщения (response):

* строка состояния (status line), включающая код состояния и сообщение о причине
* поля заголовка ответа (header fields)
* дополнительное тело сообщения (body)

### Методы HTTP-запроса

Метод, используемый в HTTP-запросе, указывает, какое действие вы хотите выполнить с этим запросом. Важные примеры:

* GET: получить подробную информацию о ресурсе
* POST: создать новый ресурс
* PUT: обновить существующий ресурс
* DELETE: Удалить ресурс

### Код статуса ответа HTTP

Код состояния всегда присутствует в ответе HTTP. Типичные примеры:

* 200 — успех
* 404 — cтраница не найдена

Для хостинга приложения мною использовался программный пакет Open Server Panel, включающий в себя модули Apache, PHP, MySQL.

open server panel. (б.д.). Получено из https://ospanel.io/

## База данных системы

Для создания базы данных использовалась встроенная в OpenServer СУБД phpMyAdmin.

phpmyadmin. (б.д.). Получено из https://www.phpmyadmin.net/

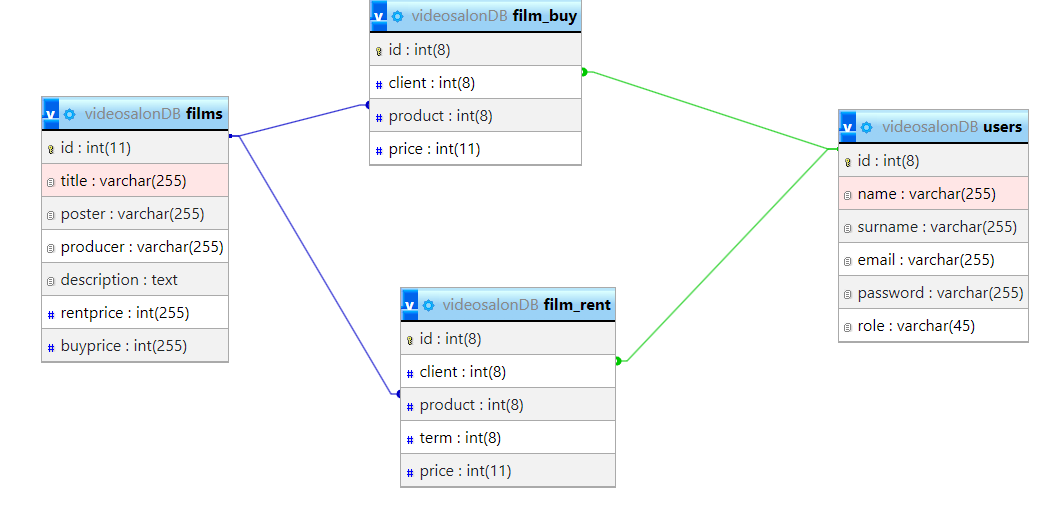


Рисунок 7 База данных

## Процесс разработки

Полный код доступен в репозитории

*репозиторий в github*. (б.д.). Получено из <https://github.com/H3LLG0/Cursovoi>

### Файловая структура проекта

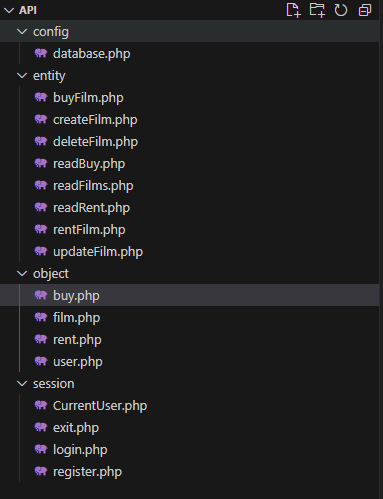


Рисунок 8 Файловая структура проекта

### Подключение к БД

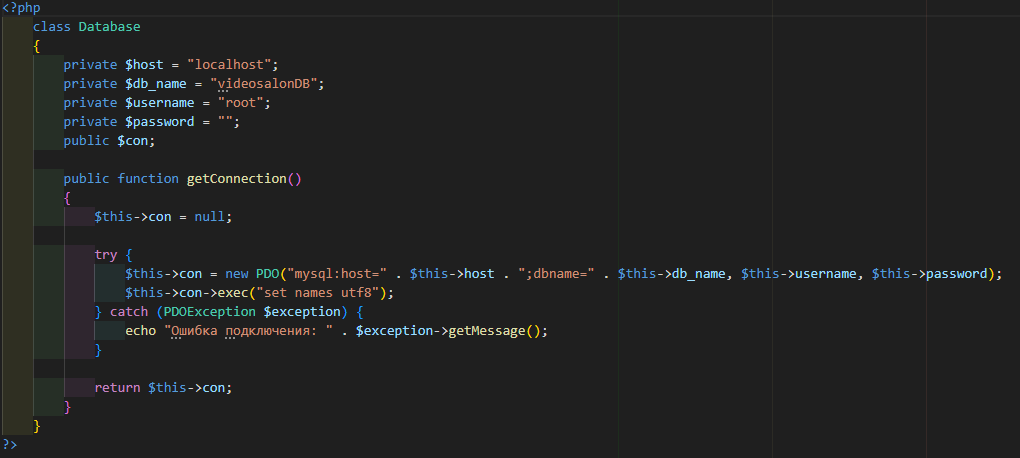


Рисунок 9 Подключение к БД

### Класс пользователь

Поля класса

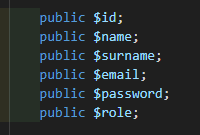


Рисунок 10 Поля класса

Конструктор подключения к базе данных

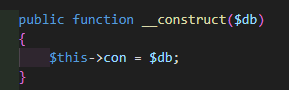


Рисунок 11 Конструктор подключения к базе данных

Получить список всех пользователей

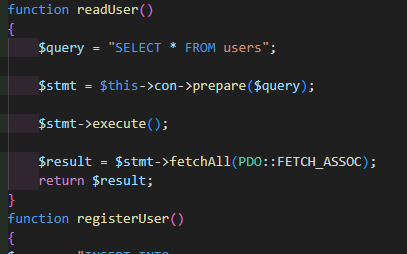


Рисунок 12 Получить список всех пользователей

Регистрация нового пользователя

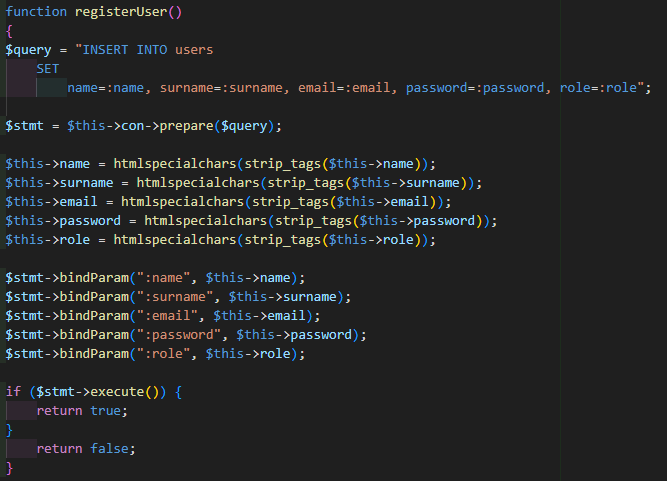


Рисунок 13 Регистрация нового пользователя

### Класс фильм

Поля класса

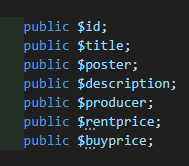


Рисунок 14 Поля класса

Получить список всех фильмов

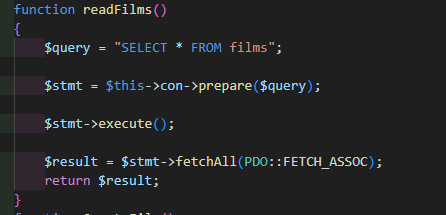


Рисунок 15 Получить список всех фильмов

Добавить фильм

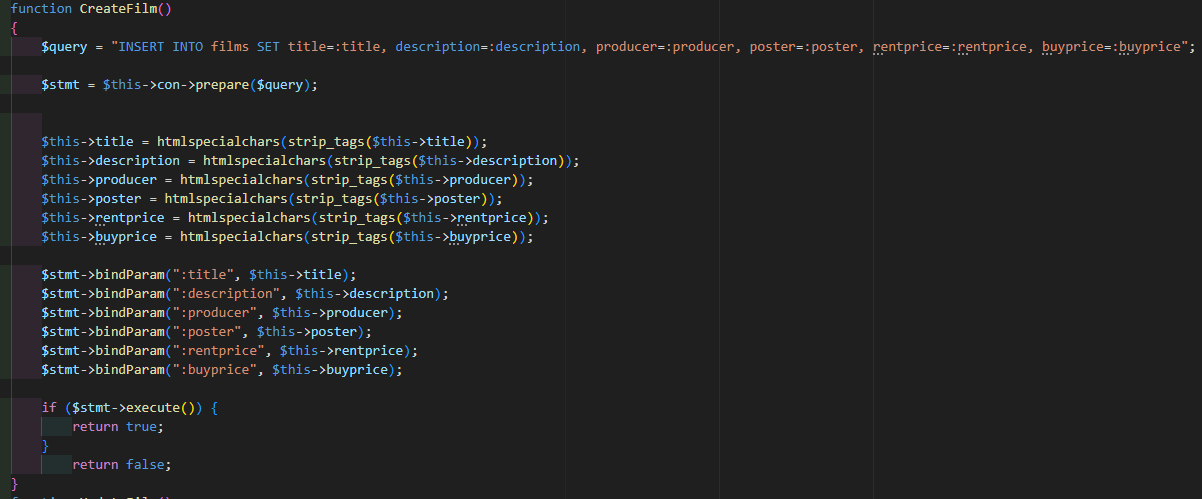


Рисунок 16 Добавить фильм

Редактировать фильм без изменения постера



Рисунок 17 Редактировать фильм без изменения постера

Редактировать фильм с изменением постера



Рисунок 18 Редактировать фильм с изменением постера

Удалить фильм

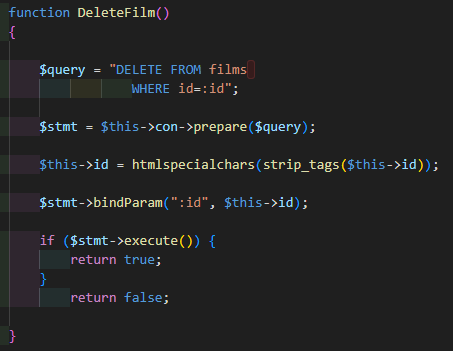


Рисунок 19 Удалить фильм

### Класс аренда

Поля класса фильм

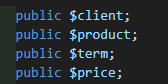


Рисунок 20 Поля класса фильм

Арендовать фильм

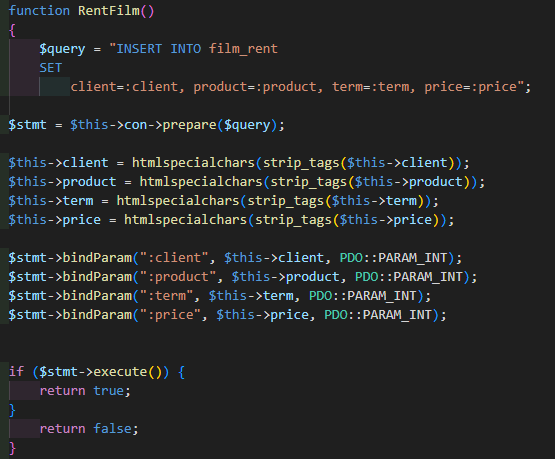


Рисунок 21 Арендовать фильм

Получить список всех арендованных пользователем фильмов

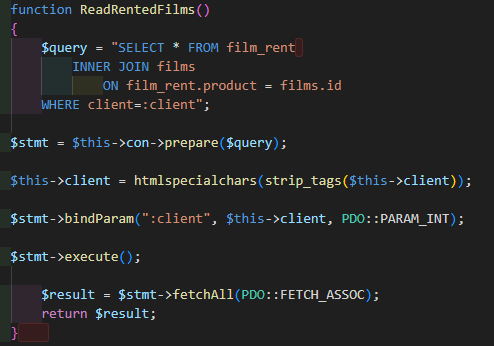


Рисунок 22 Получить список всех арендованных пользователем фильмов

### Класс покупка

Поля класса

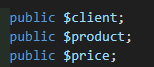


Рисунок 23 Поля класса

Купить фильм

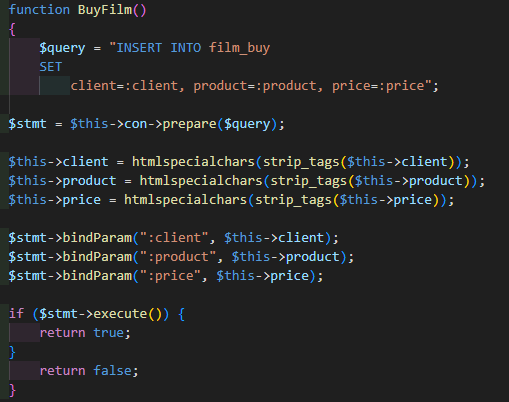


Рисунок 24 Купить фильм

Получить список всех купленных пользователем фильмов

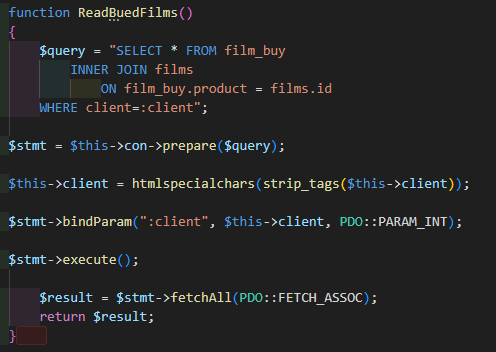


Рисунок 25 Получить список всех купленных пользователем фильмов

### Сессия пользователя

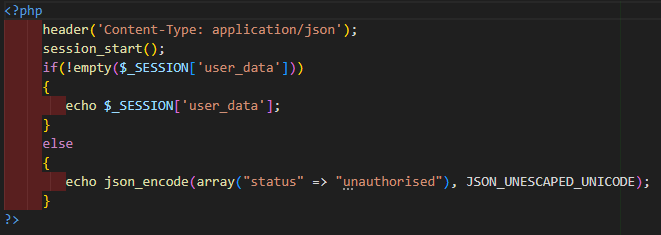


Рисунок 26 Сессия пользователя

### Авторизация

Заголовки

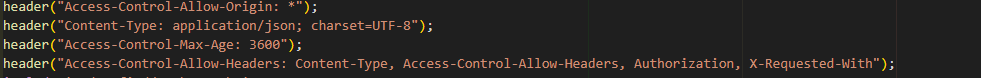


Рисунок 27 Заголовки

Подключение базы данных и класса пользователь



Рисунок 28 Подключение базы данных и класса пользователь

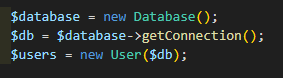


Рисунок 29 конструкторы классов

Выполнение SQL запроса и вывод результата в формате JSON

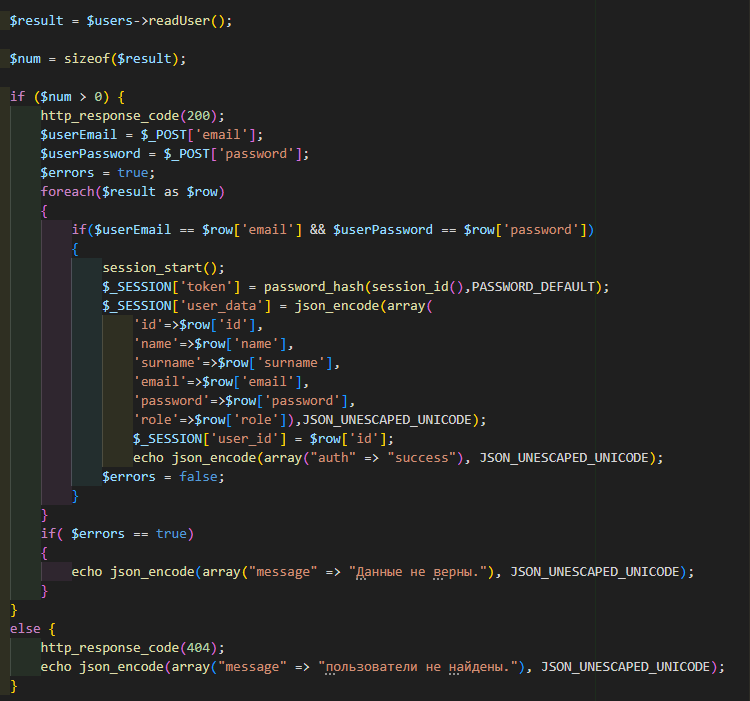


Рисунок 30 Выполнение SQL запроса и вывод результата в формате JSON

### Выход из аккаунта

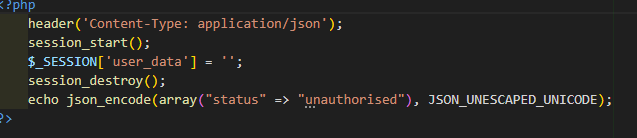


Рисунок 31 Выход из аккаунта

### Регистрация пользователя

Заголовки

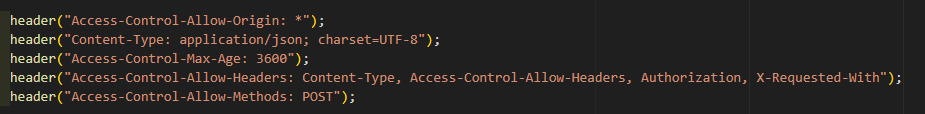


Рисунок 32 Заголовки

Подключение базы данных и класса пользователь



Рисунок 33 Подключение базы данных и класса пользователь

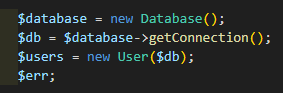


Рисунок 34 конструкторы классов

Получение данных из пользовательской формы

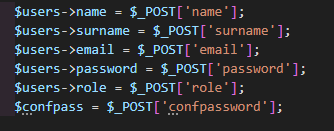


Рисунок 35 Получение данных из пользовательской формы

Получение списка всех пользователей



Рисунок 36 Получение списка всех пользователей

Проверка существования электронной почты в системе

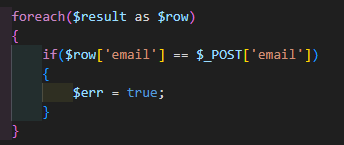


Рисунок 37 Проверка существования электронной почты в системе

Если пользователь не существует регистрирует пользователя

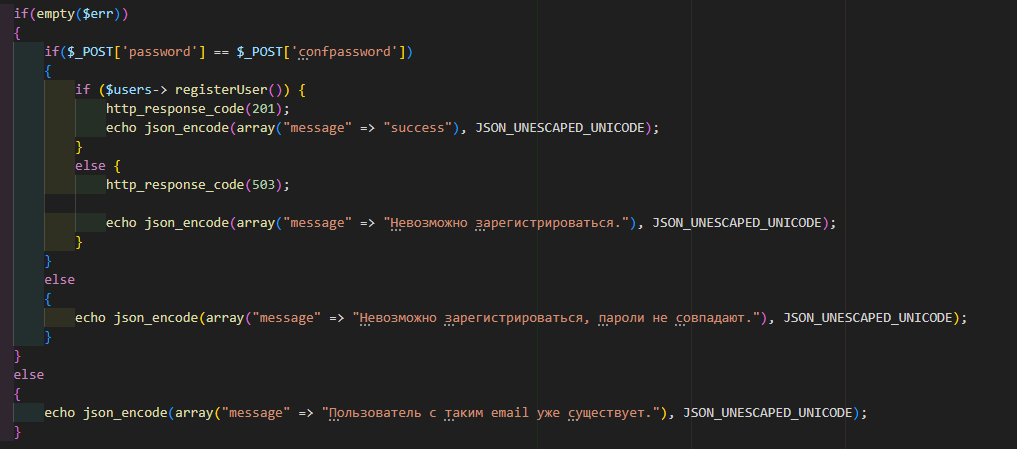


Рисунок 38 Если пользователь не существует регистрирует пользователя

### Действия с фильмами

Вывод списка всех фильмов

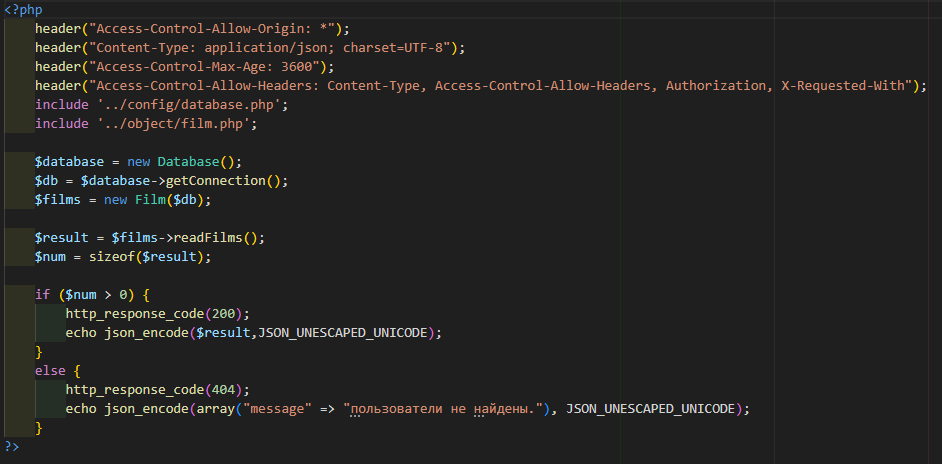


Рисунок 39 Вывод списка всех фильмов

Создание фильма



Рисунок 40 Создание фильма

Редактирование фильма

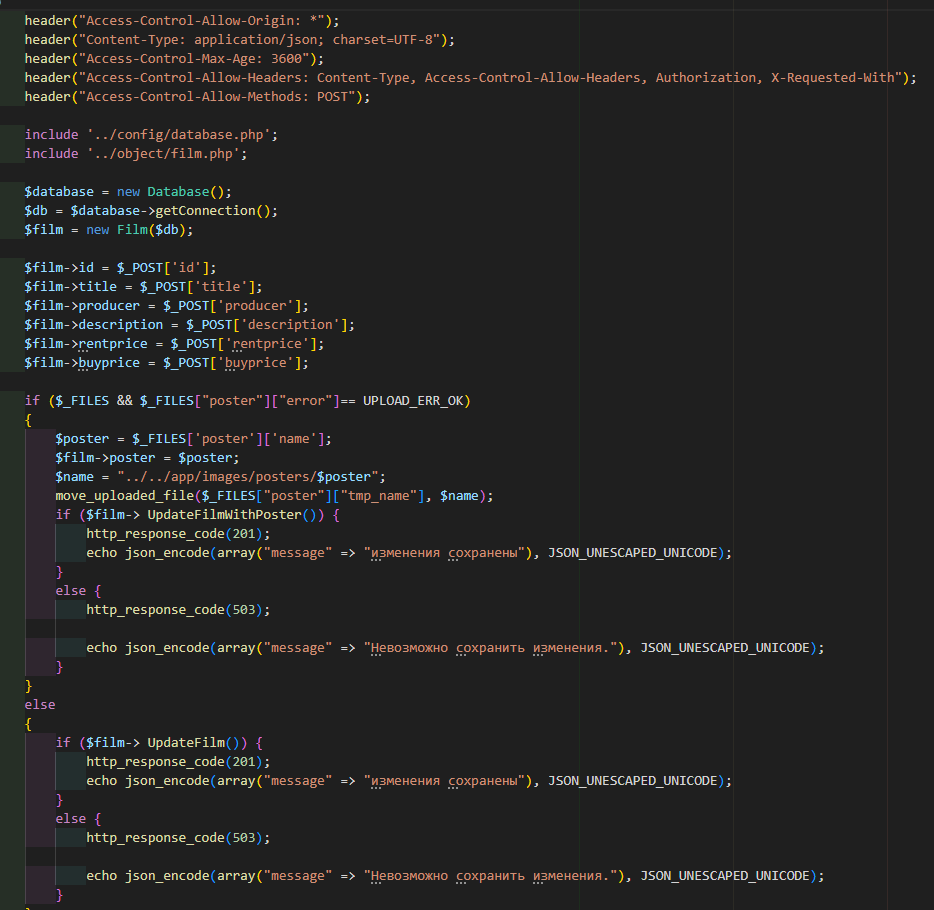


Рисунок 41 Редактирование фильма

Удаление фильма

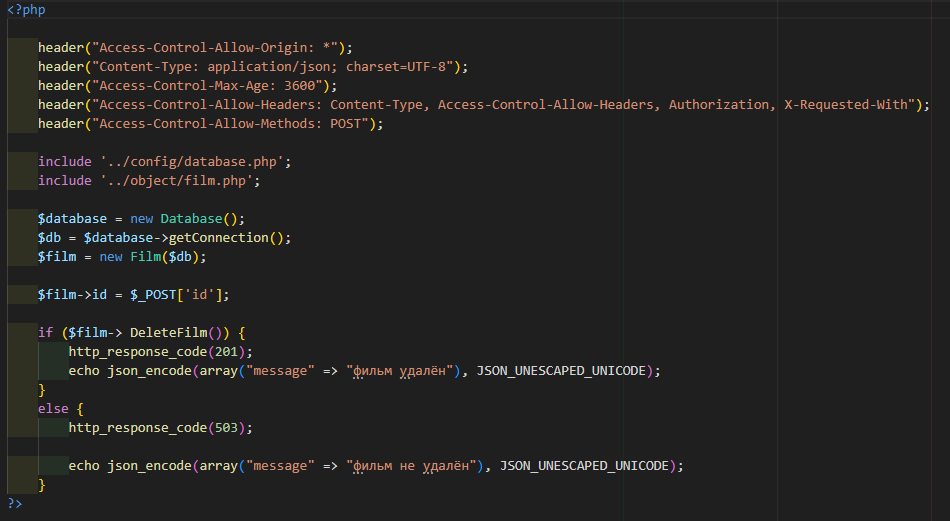


Рисунок 42 Удаление фильма

Аренда фильма

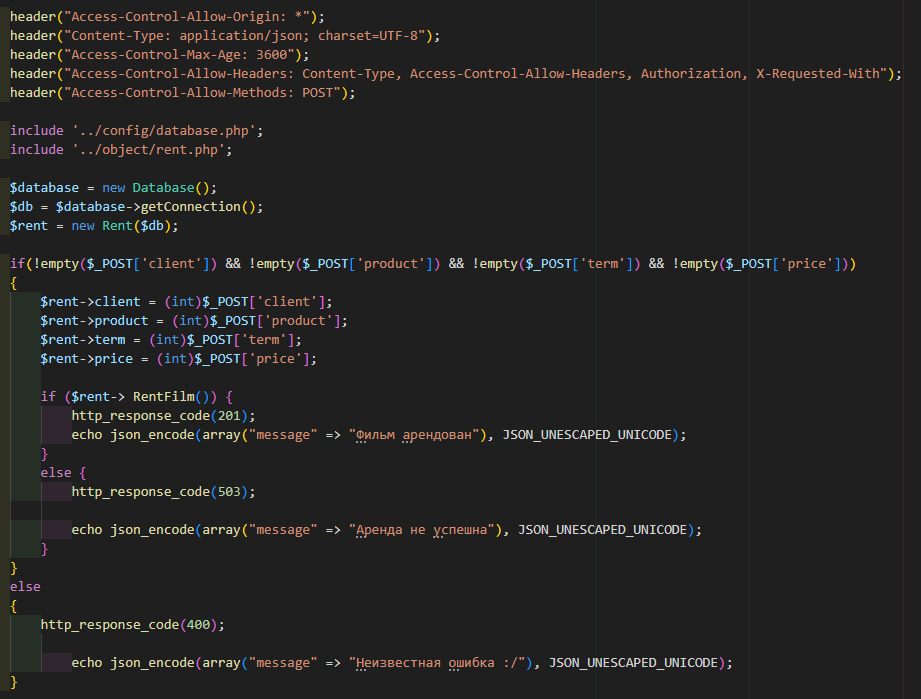


Рисунок 43 Аренда фильма

Покупка фильма

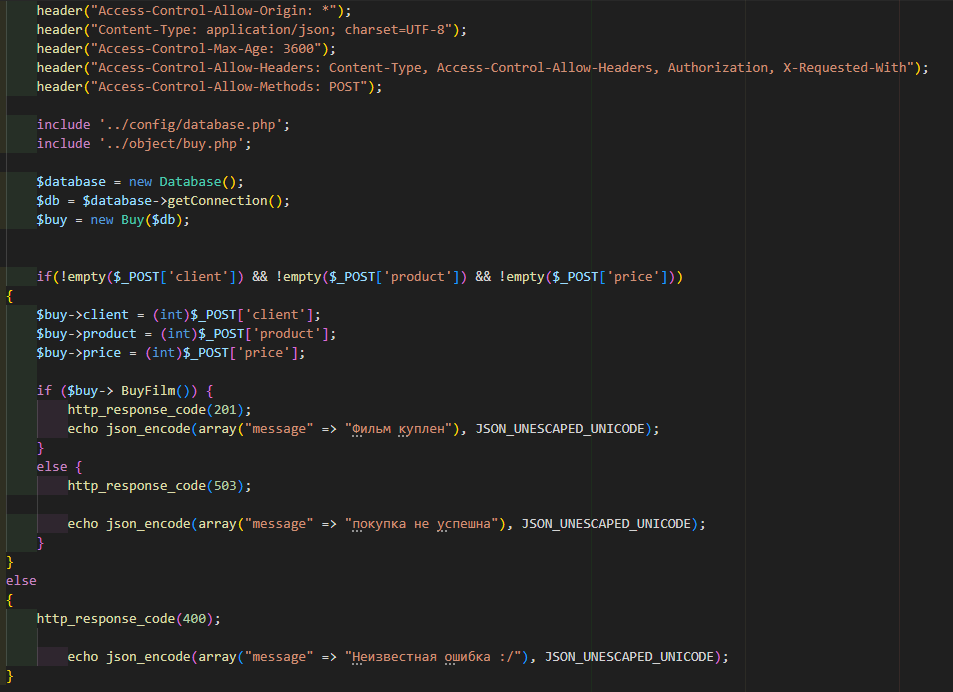


Рисунок 44 Покупка фильма

### Содержимое профиля пользователя

Список арендованных фильмов



Рисунок 45 Список арендованных фильмов

Список купленных фильмов

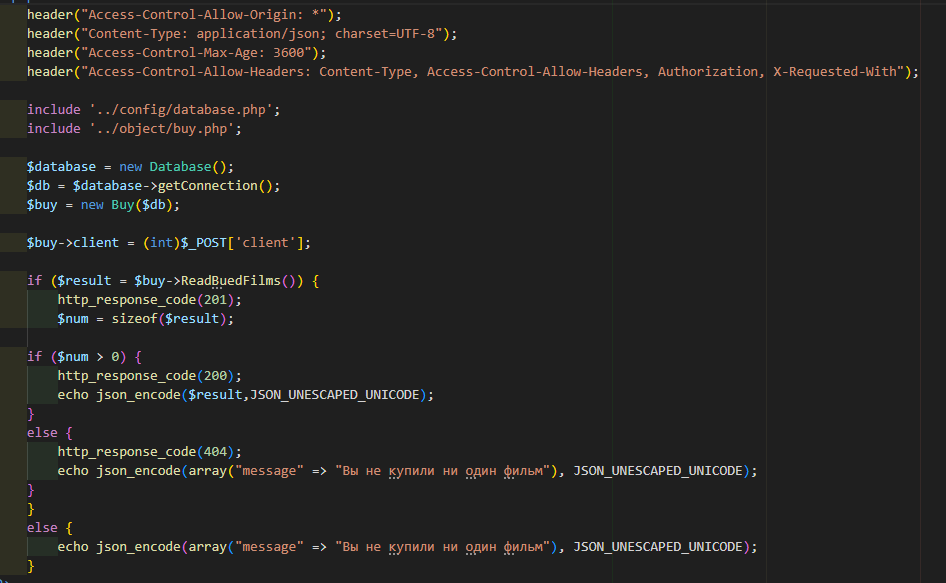


Рисунок 46 Список купленных фильмов

## Документация API

### Проверка текущей сессии

Текущей сессии нет

|  |  |
| --- | --- |
| request | response |
| url:’ http://videosalon/api/session/CurrentUser.php’  dataType: ‘json’ | Header:’ Access-Control-Allow-Origin: \*’  Header: ‘Content-Type: application/json’  Header: "Content-Type: application/json; charset=UTF-8"  Header: "Access-Control-Max-Age: 3600"  Header: "Access-Control-Allow-Headers: Content-Type, Access-Control-Allow-Headers, Authorization, X-Requested-With"  [  {  status: ’unauthorised’  }  ] |
| GET | 200 OK |

Есть активная сессия

|  |  |
| --- | --- |
| request | response |
| url:’ http://videosalon/api/session/CurrentUser.php’  dataType: ‘json’ | Header: "Content-Type: application/json; charset=UTF-8"  Header: "Access-Control-Max-Age: 3600"  Header: "Access-Control-Allow-Headers: Content-Type, Access-Control-Allow-Headers, Authorization, X-Requested-With"  [  {  Id:’2’  Name:’Константин’  Surname: ‘Савельев’  Email: ‘hgtcpvshc@gmail.com’  Password: 1234  Role: ‘admin’  }  ] |
| GET | 200 OK |

### Авторизация на сайте

Успешная авторизация

|  |  |
| --- | --- |
| request | response |
| url:’ http://videosalon/api/session/login.php’  dataType: ‘json’  [  {  email: ’hgtcpvshc@gmail.com’,  password: ’1234’  }  ] | Header: "Content-Type: application/json; charset=UTF-8"  Header: "Access-Control-Max-Age: 3600"  Header: "Access-Control-Allow-Headers: Content-Type, Access-Control-Allow-Headers, Authorization, X-Requested-With"  [  {  Massage: ’success’  }  ] |
| POST | Status: 201 Accepted |

Неверно введён логин или пароль

|  |  |
| --- | --- |
| request | response |
| url: ‘http://videosalon/api/session/login.php  ’  dataType: ‘json’  [  {  email: ’hgtcpvshc@gmail.com’,  password: ’123’  }  ] | Header: "Content-Type: application/json; charset=UTF-8"  Header: "Access-Control-Max-Age: 3600"  Header: "Access-Control-Allow-Headers: Content-Type, Access-Control-Allow-Headers, Authorization, X-Requested-With"  [  {  Massage: ’Ошибка неверный логин или пароль’  }  ] |
| POST | Status: 503 |

### Регистрация пользователя

Успешная регистрация

|  |  |
| --- | --- |
| request | response |
| url: ‘http://videosalon/api/session/register.php  ’  dataType: ‘json’  [  {  name: ’Константин’,  surname:’Савельев’,  email: ’hgtcpvshc@gmail.com’,  password: ’1234’,  confirmpassword: ‘1234’  }  ] | Header: "Content-Type: application/json; charset=UTF-8"  Header: "Access-Control-Max-Age: 3600"  Header: "Access-Control-Allow-Headers: Content-Type, Access-Control-Allow-Headers, Authorization, X-Requested-With"  [  {  Massage: ’Пользователь добавлен’  }  ] |
| POST | Status: 201 Accepted |

Не заполнены поля

|  |  |
| --- | --- |
| request | response |
| url: ‘http://videosalon/api/session/register.php  ’  dataType: ‘json’  [  {  name: ’’,  surname:’Савельев’,  email: ’hgtcpvshc@gmail.com’,  password: ’1234’,  confirmpassword: ‘1234’  }  ] | Header: "Content-Type: application/json; charset=UTF-8"  Header: "Access-Control-Max-Age: 3600"  Header: "Access-Control-Allow-Headers: Content-Type, Access-Control-Allow-Headers, Authorization, X-Requested-With"  [  {  Massage:’не все поля заполены’  }  ] |
| POST | Status: 400 Bad request |

Пользователь уже существует

|  |  |
| --- | --- |
| request | response |
| url: ‘http://videosalon/api/session/register.php  ’  dataType: ‘json’  [  {  name: ’Константин’,  surname:’Савельев’,  email: ’hgtcpvshc@gmail.com’,  password: ’1234’,  confirmpassword: ‘1234’  }  ] | Header: "Content-Type: application/json; charset=UTF-8"  Header: "Access-Control-Max-Age: 3600"  Header: "Access-Control-Allow-Headers: Content-Type, Access-Control-Allow-Headers, Authorization, X-Requested-With"  [  {  Massage:пользователь уже существует’  }  ] |
| POST | Status: 400 Bad request |

Пароли не совпадают

|  |  |
| --- | --- |
| request | response |
| url: ‘http://videosalon/api/session/register.php  ’  dataType: ‘json’  [  {  name: ’Константин’,  surname:’Савельев’,  email: ’hgtcpvshc@gmail.com’,  password: ’1234’,  confirmpassword: ‘12343’  }  ] | Header: "Content-Type: application/json; charset=UTF-8"  Header: "Access-Control-Max-Age: 3600"  Header: "Access-Control-Allow-Headers: Content-Type, Access-Control-Allow-Headers, Authorization, X-Requested-With"  [  {  Massage:пароли не совпадают’  }  ] |
| POST | Status: 400 Bad request |

### Действия с фильмами

Список всех фильмов

|  |  |
| --- | --- |
| request | response |
| url: ‘http://videosalon/api/entity/readFilms.php  ’  dataType: ‘json’ | [      {          "id": "1",          "title": "Криминальное чтиво",          "poster": "Pulp\_Fiction.jpg",          "producer": "Квентин тарантино",          "description": "тут описание",          "rentprice": "100",          "buyprice": "600"      }  ] |
| GET | 200 OK |

Добавление фильма

|  |  |
| --- | --- |
| request | response |
| url: ‘http://videosalon/api/entity/CreateFilms.php  ’  dataType: ‘json’  [      {          "title": "Криминальное чтиво",          "poster": "Pulp\_Fiction.jpg",          "producer": "Квентин тарантино",          "description": "тут описание",          "rentprice": "100",          "buyprice": "600"      }  ] | Header: "Content-Type: application/json; charset=UTF-8"  Header: "Access-Control-Max-Age: 3600"  Header: "Access-Control-Allow-Headers: Content-Type, Access-Control-Allow-Headers, Authorization, X-Requested-With"  [  {  Messege: фильм добавлен  }  ] |
| POST | 201 Accepted |

Добавление фильма неудачно

|  |  |
| --- | --- |
| request | response |
| url: ‘http://videosalon/api/entity/CreateFilms.php  ’  dataType: ‘json’  [      {          "title": "Криминальное чтиво",          "poster": "Pulp\_Fiction.jpg",          "producer": "Квентин тарантино",          "description": "тут описание",          "rentprice": "100",          "buyprice": "600"      }  ] | Header: "Content-Type: application/json; charset=UTF-8"  Header: "Access-Control-Max-Age: 3600"  Header: "Access-Control-Allow-Headers: Content-Type, Access-Control-Allow-Headers, Authorization, X-Requested-With"  [  {  Messege: Невозможно добавить фильм  }  ] |
| POST | 400 Bad request |

Удаление фильма

|  |  |
| --- | --- |
| request | response |
| url: ‘http://videosalon/api/entity/DeleteFilm.php  ’  dataType: ‘json’  [      {          ‘id‘:1      }  ] | Header: "Content-Type: application/json; charset=UTF-8"  Header: "Access-Control-Max-Age: 3600"  Header: "Access-Control-Allow-Headers: Content-Type, Access-Control-Allow-Headers, Authorization, X-Requested-With"  [  {  Messege: фильм удалён  }  ] |
| DELETE | 201 Accepted |

Редактирование фильма

|  |  |
| --- | --- |
| request | response |
| url: ‘http://videosalon/api/entity/UpdateFilm.php  ’  dataType: ‘json’  [      {          "title": "Криминальное чтиво",          "poster": "Pulp\_Fiction.jpg",          "producer": "Квентин тарантино",          "description": "описание изменено",          "rentprice": "100",          "buyprice": "600"      }  ] | Header: "Content-Type: application/json; charset=UTF-8"  Header: "Access-Control-Max-Age: 3600"  Header: "Access-Control-Allow-Headers: Content-Type, Access-Control-Allow-Headers, Authorization, X-Requested-With"  [  {  Messege: фильм обновлён  }  ] |
| PUT | 201 Accepted |

Аренда фильма

|  |  |
| --- | --- |
| request | response |
| url: ‘http://videosalon/api/entity/RentFilm.php  ’  dataType: ‘json’  [      {          'product':1,  ‘user’:1,  ‘term’:6,  ‘price’:600      }  ] | Header: "Content-Type: application/json; charset=UTF-8"  Header: "Access-Control-Max-Age: 3600"  Header: "Access-Control-Allow-Headers: Content-Type, Access-Control-Allow-Headers, Authorization, X-Requested-With"  [  {  Messege: фильм арендован  }  ] |
| POST | 201 Accepted |

Покупка фильма

|  |  |
| --- | --- |
| request | response |
| url: ‘http://videosalon/api/entity/RentFilm.php  ’  dataType: ‘json’  [      {          'product':1,  ‘user’:1,  ‘price’:600      }  ] | Header: "Content-Type: application/json; charset=UTF-8"  Header: "Access-Control-Max-Age: 3600"  Header: "Access-Control-Allow-Headers: Content-Type, Access-Control-Allow-Headers, Authorization, X-Requested-With"  [  {  Messege: фильм куплен  }  ] |
| POST | 201 Accepted |

# Тесты системы

Тест-кейс — это форма записи проверки, которую проводит тестировщик. По сути, это алгоритм действий, по которому предполагается тестировать уже написанную программу. В нём подробно прописаны шаги, которые нужно сделать для подготовки к тесту, сама проверка и ожидаемый результат.

*Тест кейсы*. (б.д.). Получено из <https://practicum.yandex.ru/blog/chto-takoe-test-keys-i-kak-ego-sostavit/>

[Подборка тест кейсов](Тесты%20системы.xlsx)

# Внедрение

## Руководство администратора

[Руководство администара](руководство%20администратора.docx) выполнено по ГОСТ 19.505-79.

*ГОСТ 19.505-79.* (б.д.). Получено из <https://npo-its.ru/misc/pdf/usage_d_monitoring.pdf>

## Руководство пользователя

[Руководство пользователя](руководство%20пользователя.docx) выполнено по ГОСТ 34.201-89

*ГОСТ на руководство пользователя*. (б.д.). Получено из <https://techwrconsult.com/library/34.201>

# Заключение

Код информационной системы для видеосалона разработан, ИС позволяет автоматизировать процесс аренды и покупки фильмов а так же управление фильмами находящимися в системе.

# Ссылки

1. *AJAX: Шпаргалка по AJAX на jQuery*. (б.д.). Получено из https://habr.com/ru/articles/658139/
2. *Frontend*. (б.д.). Получено из https://only-to-top.ru/blog/programming/2019-11-11-jquery-ajax-json-php.html
3. *open server panel*. (б.д.). Получено из https://ospanel.io/
4. *phpmyadmin*. (б.д.). Получено из https://www.phpmyadmin.net/
5. *PlantUML*. (б.д.). Получено из https://plantuml.com/ru/
6. *REST API — RESTful веб-сервисы*. (б.д.). Получено из https://habr.com/en/articles/483202/
7. *Rest API PHP пошаговое руководство*. (б.д.). Получено из https://only-to-top.ru/blog/programming/2019-11-06-rest-api-php.html
8. *UML*. (б.д.). Получено из https://practicum.yandex.ru/blog/uml-diagrammy/
9. *Wireframe ИС для видеосалона*. (б.д.). Получено из https://www.figma.com/design/STPWWitKppil6SpPsTYeQx/Untitled?node-id=0-1&t=9mnBJACKNrd0GidW-1
10. *ГОСТ 19.505-79.* (б.д.). Получено из https://npo-its.ru/misc/pdf/usage\_d\_monitoring.pdf
11. *ГОСТ на руководство пользователя*. (б.д.). Получено из https://techwrconsult.com/library/34.201
12. *документация JQuery*. (б.д.). Получено из https://api.jquery.com/
13. *Документация PHP*. (б.д.). Получено из https://www.php.net/manual/en/
14. *репозиторий в github*. (б.д.). Получено из https://github.com/H3LLG0/Cursovoi
15. *руководство пользователя ГОСТ 34.201*. (б.д.). Получено из https://techwrconsult.com/rukovodstvo-polzovatelya
16. *сайт для анализа 1*. (б.д.). Получено из https://chef.ru/place/gamburgery/?ysclid=ls1gb64nck297366140
17. *сайт для анализа 2*. (б.д.). Получено из https://psncoolgame.com/?ysclid=ls1gvj4w8g407144618
18. *Тест кейсы*. (б.д.). Получено из https://practicum.yandex.ru/blog/chto-takoe-test-keys-i-kak-ego-sostavit/

