Содержание

[Введение 3](#_Toc483176900)

[Анализ задачи 4](#_Toc483176901)

[Анализ предметной области, выявление необходимой пользователю функциональности 4](#_Toc483176902)

[Разработка общей архитектуры ИС 5](#_Toc483176903)

[Разработка серверной части информационной системы 7](#_Toc483176904)

[Выявление сущностей, их атрибутов и ключей. Выявление связей. Построение ER-диаграммы 7](#_Toc483176905)

[Переход от ER-диаграммы к предварительным отношениям БД. Заполнение предварительных отношений атрибутами 10](#_Toc483176906)

[Нормализация предварительных отношений до 3НФ. Построение схемы данных 12](#_Toc483176907)

[Выявление, описание и реализация внешних ограничений целостности данных 14](#_Toc483176908)

[Разработка клиентской части информационной системы. 17](#_Toc483176909)

[Организация взаимодействия клиентской программы с БД. 17](#_Toc483176910)

[Разработка форм и отчетов 23](#_Toc483176911)

[Разработка сценария инсталляции клиентской программы. 34](#_Toc483176912)

[Руководство пользователя. 35](#_Toc483176913)

[Заключение 37](#_Toc483176914)

[Список использованных источников 38](#_Toc483176915)

[Приложение 39](#_Toc483176916)

[SQL скрипт для создания базы данных 39](#_Toc483176917)

[SQL скрипт для создания базы данных с данными, хранимыми процедурами и функциями 42](#_Toc483176918)

# Введение

Реляционные СУБД являются в настоящий момент самыми распространёнными. Их реализации существуют на всех пригодных для этого платформах, для всех операционных систем и для всех применений от простейших продуктов, предназначенных для ведения картотек индивидуального пользования, до сложнейших распределённых многопользовательских систем.

СУБД в общем случае это собственно база данных, которая предполагает какой-либо метод сохранения информации на диске и возможности доступа и манипуляции с нею, и набор программных продуктов, представляющий пользователю все допустимые в базе средствах работы с данными. Набор программных средств манипуляции данными СУБД удовлетворяет свойствам полноты.

СУБД позволяет структурировать, систематизировать и организовать данные для их компьютерного хранения и обработки. Сегодня невозможно представить себе деятельность любого современного предприятия или организации без использования профессиональных СУБД. Несомненно, они составляют фундамент информационной деятельности во всех сферах – начиная с производства и заканчивая финансами и телекоммуникациями

В данной работе была разработана база данных для автоматизации процесса выдачи в проката видеофильмов.

# Анализ задачи

## Анализ предметной области, выявление необходимой пользователю функциональности

В настоящее время эффективное функционирование современного предприятия невозможно без применения информационных систем. Эта проблема актуальна как для крупных предприятий, так и для предприятий среднего и даже малого бизнеса. Информационные системы имеют ряд существенных отличий от стандартных прикладных программ. В зависимости от предметной области информационные системы могут сильно различаться по своей архитектуре и функциям.

При разработке базы данных «Видеотека» было проведено обследование предметной области. В результате в БД «Видеотека» используются следующие входные данные:

* информация о клиентах;
* информация о сотрудниках;
* информация о видео;
* информация о форматах видео;
* информация о производителях;
* информация о режиссёрах;
* информация о прокате.

Выходными данными являются выходные формы, запросы.

Основными функциями пользователя ИС «Видеотека» являются:

* Добавление информации о видео;
* Добавление информации о режиссёрах;
* Добавление информации о сотрудниках;
* Оформление проката фильмов/удаления проката (Фильмы выданные другими сотрудниками, не должны быть доступны для удаления). Удаленные фильмы должны сохраняться в историю;
* Отчет, выводит оформленные фильмы в прокат сотрудником, вошедшим в систему.

## Разработка общей архитектуры ИС

В основе разрабатываемой системы лежит архитектура «клиент-сервер», которая состоит из трёх основных компонентов: клиентская часть, серверная часть, база данных.

* Браузер через Интернет отсылает HTTP-запросы web-серверу
* web-сервер вызывает скрипт PHP
* PHP-скрипт обращается к базе данных, если это нужно
* В результате PHP-cкрипт возвращает клиенту информацию, которую и отображает браузер.

Клиентская часть – любой стандартный браузер, поддерживающий формы HTML и JavaScript. Браузер функционирует как обобщенное устройство с интерфейсом пользователя. Пользователь приложения посредством браузера запрашивает Web-страницы – либо статические в формате HTML, либо динамические. На возвращаемой странице содержится полностью отформатированный пользовательский интерфейс — текст и управляющие элементы, которые отображаются браузером на экране клиентского компьютера. Взаимодействие пользователя с системой осуществляется через браузер. Художественное оформление web – страниц описывается таблицами стилей – CSS. Для "оживления" графического интерфейса используются скрипты JavaScript.

Серверная часть web-приложения – это программа или скрипт на сервере, обрабатывающая запросы пользователя. При каждом переходе пользователя по ссылке браузер отправляет запрос к серверу. Сервер обрабатывает этот запрос, вызывая некоторый скрипт, который формирует web-страницу, описанную языком HTML, и отсылает клиенту по сети. Браузер тут же отображает полученный результат в виде очередной web-страницы. Для программирования серверной части web-приложения использовался язык PHP поскольку он является самым распространённым и самым простым.

База данных (БД, или система управления базами данных, СУБД) - программное обеспечение на сервере, занимающееся хранением данных и их выдачей в нужный момент.

Соединение HTTP (HTTP connection) – стандартный протокол взаимодействия клиентских браузеров и Web-серверов. Этот архитектурный элемент представляет взаимодействие между клиентом и сервером без поддержки соединения. Каждый раз, когда клиент и сервер обмениваются информацией, устанавливается новое независимое соединение.

На достаточно высоком уровне абстракции архитектуру системы можно отнести к шаблону Thin Web Client(на основе "тонкого" Web-клиента) который используется в большинстве приложений Internet и предоставляет ограниченные возможности по управлению конфигурацией клиента. В распоряжении клиента имеется только стандартный браузер, поддерживающий формы и JavaScript. Все операции, связанные с бизнес – логикой, выполняются на сервере. Данный шаблон используется, поскольку нет никакого смысла ставить мощные ПК, каждому работнику, плюс сотрудники могут заключать сделки, в одном из приличнын заведений, где под рукой будет только мобильный телефон. Типичное приложение электронной коммерции предназначено для выдачи в прокат фильмов в любое время, в любом месте с использованием оборудования в котором есть всего лишь браузер и интернет, а как мы с вами знаем – это сейчас уже есть в каждом телефоне.

# Разработка серверной части информационной системы

## Выявление сущностей, их атрибутов и ключей. Выявление связей. Построение ER-диаграммы

В результате анализа предметной области были выявлены следующие сущности:

* Клиент;
* Видео;
* Прокат;
* Режиссёр;
* Производитель;
* Жанр;
* Страна;
* Работник;
* Формат;

В таблице «сущностей» описаны сущности и выявленные первичные ключи.

**Таблица «сущностей»**

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Первичный ключ |
| Клиент | Кому, которому предоставляется прокат | Код клиента |
| Сотрудник | Работник, который предоставляет прокат | Код работника |
| Производитель | Видеостудия | Код производителя |
| Страна | Страна видеостудии | Код страны |
| Формат | Формат видео | Код формата |
| Прокат | Информация о прокате видео | Код проката |
| Режиссёр | Режиссёр видео | Код режиссёра |
| Жанр | Категория видео | Код жанра |
| Видео | Видео, которое предоставляется в прокат | Код видео |

В таблице «атрибутов» описаны выявленные атрибуты сущностей.

**Таблица «атрибутов»**

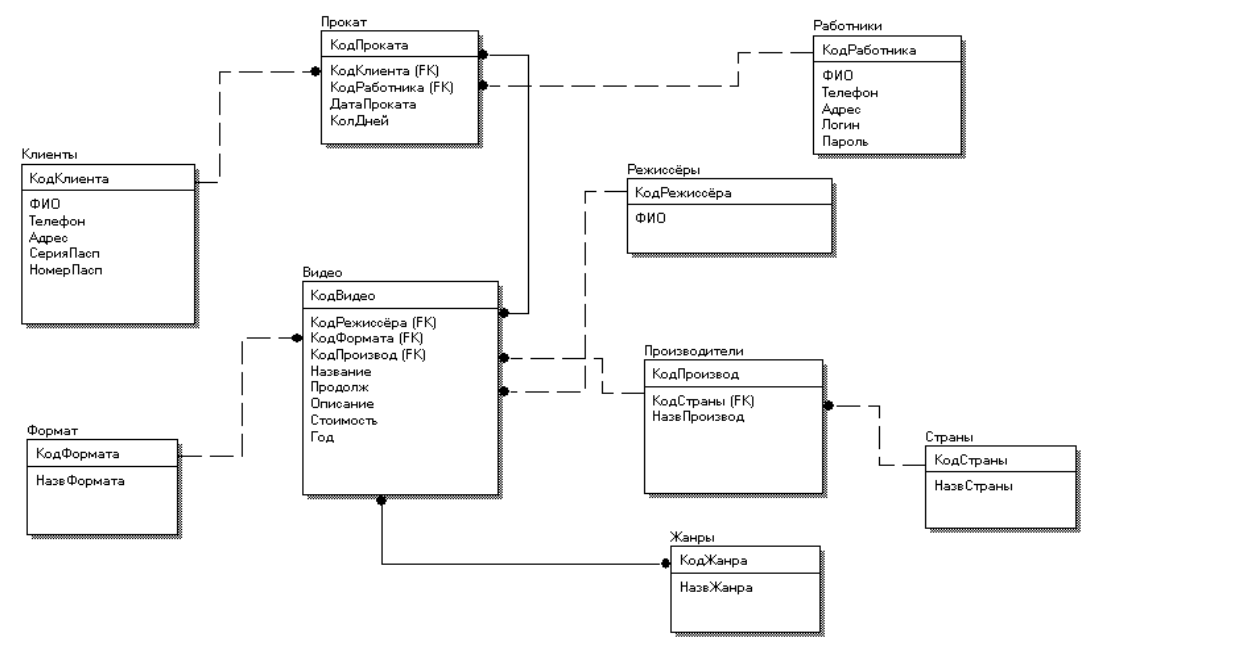
|  |  |  |  |
| --- | --- | --- | --- |
| Наименование | Признак ключа | Формат | Неопределённое значение |
| Код Клиента | PK | Числовой |  |
| ФИО | - | Текстовый(30) |  |
| Адрес | - | Текстовый(30) |  |
| Телефон | - | Текстовый(12) | да |
| Серия паспорта |  | Текстовый(4) |  |
| Номер паспорта |  | Текстовый(6) |  |
| Код сотрудника | PK | Числовой |  |
| Телефон | - | Текстовый(12) |  |
| ФИО | - | Текстовый(30) |  |
| Адрес |  | Текстовый(30) |  |
| Логин |  | Текстовый(30) |  |
| Пароль |  | Текстовый(50) |  |
| Код производителя | PK | Числовой |  |
| Код страны | FK | Числовой |  |
| Название |  | Текстовый(30) |  |
| Код формата | PK | Числовой |  |
| Название |  | Текстовый(20) |  |
| Код проката | PK | Числовой |  |
| Код сотрудника | FK | Числовой |  |
| Код Клиента | FK | Числовой |  |
| Даты |  | Дата |  |
| Кол-во дней |  | Числовой | да |
| Код жанра | PK | Числовой |  |
| Название |  | Текстовый(45) |  |
| Код видео | PK | Числовой |  |
| Название |  | Текстовый(60) |  |
| Стоимость |  | Денежный | да |
| Код режиссёра | FK | Числовой |  |
| Код производителя | FK | Числовой |  |
| Год издания |  | Текстовый(4) | да |
| Описание |  | Текстовый(255) |  |
| Код режиссёра | PK | Числовой |  |
| ФИО | - | Текстовый(30) |  |

В таблице «связей» описаны выявленные связи.

**Таблица «связей»**

|  |  |  |  |
| --- | --- | --- | --- |
| Сущность 1 | Название связи | Сущность 2 | Тип связи |
| Работник | Оформляет | Прокат | Один - ко - многим |
| Прокат | Содержит | Видео | Многие-ко-многим |
| Видео | Имеет | Жанр | Многие-ко-многим |
| Видео | Имеет | Производитель | Один - ко - многим |
| Видео | Имеет | Режиссёр | Один - ко - многим |
| Видео | Имеет | Формат | Один - ко - многим |
| Производитель | Имеет | Страна | Один - ко - многим |

На рис.1 представлена ER диаграмма базы данных.



**Рис.1 ER диаграмма**

## Переход от ER-диаграммы к предварительным отношениям БД. Заполнение предварительных отношений атрибутами

ER модель позволяет более точно представить предметную область, чем реляционные и другие более ранние модели. Но в настоящее время существует немного СУБД, поддерживающих эти модели. На практике наиболее распространены системы, реализующие реляционную модель. Поэтому необходим метод перевода ER модели в реляционную. Такой метод основывается на формировании набора предварительных таблиц из ER-диаграмм.

Для каждой сущности создается таблица. Причем каждому атрибуту сущности соответствует столбец таблицы.

Правила генерации таблиц из ER-диаграмм опираются на два основных фактора – тип связи и класс принадлежности сущности.

Если связь типа 1:1 и класс принадлежности обеих сущностей является обязательным, то необходима только одна таблица. Первичным ключом этой таблицы может быть первичный ключ любой из двух сущностей.

Если связь типа 1:1 и класс принадлежности одной сущности является обязательным, а другой – необязательным, то необходимо построить таблицу для каждой сущности. Первичный ключ сущности должен быть первичным ключом соответствующей таблицы. Первичный ключ сущности, для которой класс принадлежности является необязательным, добавляется как атрибут в таблицу для сущности с обязательным классом принадлежности.

Сущность с необязательным классом принадлежности именуется родительской, а с обязательным – дочерней. Первичный ключ родительской сущности (НФ), помещаемый в таблицу, представляющую дочернюю сущность, называется внешним ключом родительской сущности. Связь между указанными таблицами устанавливается путем связи первичного и внешнего ключа.

Если связь типа 1:1 и класс принадлежности обеих сущностей является необязательным, то необходимо построить три таблицы – по одной для каждой сущности и одну для связи. Первичный ключ сущности должен быть первичным ключом соответствующей таблицы. Таблица для связи среди своих атрибутов должна иметь ключи обеих сущностей.

Если связь типа 1:М и класс принадлежности сущности на стороне М является обязательным, то необходимо построить таблицу для каждой сущности. Первичный ключ сущности должен быть первичным ключом соответствующей таблицы. Первичный ключ сущности на стороне 1 добавляется как атрибут в таблицу для сущности на стороне М.

Если связь типа 1:М и класс принадлежности сущности на стороне М является необязательным, то необходимо построить три таблицы – по одной для каждой сущности и одну для связи. Первичный ключ сущности должен быть первичным ключом соответствующей таблицы. Таблица для связи среди своих атрибутов должна иметь ключи обеих сущностей.

Если связь типа М:N, то необходимо построить три таблицы – по одной для каждой сущности и одну для связи. Первичный ключ сущности должен быть первичным ключом соответствующей таблицы. Таблица для связи среди своих атрибутов должна иметь ключи обеих сущностей.

## Нормализация предварительных отношений до 3НФ. Построение схемы данных

Нормализация – это пошаговый, обратимый процесс замены исходной схемы другой схемой, в которой таблицы имеют более простую и логичную структуру.

Каждая нормальная форма налагает определенные ограничения на данные. Каждая нормальная форма более высокого уровня предполагает, что анализируемая таблица уже находится в нормальной форме на уровень ниже рассматриваемой. В ходе нормализации схема базы данных становится все более строгой, а ее таблицы все менее подвержены различного рода аномалиям.

Для реляционных баз данных необходимо, чтобы ее таблицы находились в 1НФ. Нормальные формы более высоких уровней могут использоваться разработчиками по своему усмотрению. Однако грамотный специалист, стремится к тому, чтобы довести уровень нормализации базы данных хотя бы до 3НФ, тем самым исключив избыточность данных и аномалии обновления.

Таблица находится в первой нормальной форме, если все ее поля имеют простые (атомарные) значения. Само понятие атомарности определить достаточно трудно. Значение, атомарное в одном случае, может быть неатомарным в другом. Общий принцип здесь такой: значение не атомарно, если оно используется по частям.

В наших таблицах все поля имеют простые значения, поэтому таблицы можно считать приведёнными к 1НФ.

Таблица находится во второй нормальной форме, если она находится в первой нормальной форме, а каждое неключевое поле функционально полно зависит от составного ключа. Эта форма применяется к таблицам с составными ключами. Таблица, у которой первичный ключ включает только одно поле, всегда находится во 2НФ. В нашей БД таблицы не имеют составных ключей, поэтому можно считать, что все таблицы находятся во 2НФ.

Таблица находится в третьей нормальной форме, если она находится во второй нормальной форме, и каждое неключевое поле нетранзитивно зависит от первичного ключа.

Транзитивная зависимость наблюдается в том случае, если одно из двух неключевых полей зависит от первичного ключа, а другое зависит от первого неключевого поля.

Все остальные таблицы нашей базы данных находятся в 3НФ.

## Выявление, описание и реализация внешних ограничений целостности данных

Обеспечение целостности данных гарантирует качество данных в таблице. Например, если видео присвоен идентификатор «123», база данных не должна позволять другому видео иметь такое же значение идентификатора. Если существует столбец «year\_estab», в котором значения должны находиться в диапазоне из 4 символов, база данных не должна сохранять в нем значения, лежащие вне этого диапазона.

При планировании таблиц имеются два важных шага: определить допустимые значения для столбца и решить, каким образом обеспечить целостность данных в этом столбце. Целостность данных подразделяется на следующие категории: сущностная целостность, доменная целостность, ссылочная целостность, пользовательская целостность.

Сущностная целостность определяет строку как уникальную сущность в конкретной таблице. Она обеспечивает целостность столбцов идентификаторов или первичного ключа таблицы с помощью индексов и ограничений UNIQUE или PRIMARY KEY

Доменная целостность — это достоверность записей в конкретном столбце. Она включает ограничения типа данных, ограничения формата при помощи ограничений CHECK и правил, а также ограничения диапазона возможных значений при помощи ограничений FOREIGN KEY, CHECK, DEFAULT, определений NOT NULL и правил.

Ссылочная целостность сохраняет определенные связи между таблицами при добавлении или удалении строк. Ссылочная целостность основана на связи первичных и внешних ключей (либо внешних и уникальных ключей) и обеспечивается с помощью ограничений FOREIGN KEY и CHECK. Ссылочная целостность гарантирует согласованность значений ключей во всех таблицах. Этот вид целостности требует отсутствия ссылок на несуществующие значения, а также обеспечивает согласованное изменение ссылок во всей базе данных при изменении значения ключа.

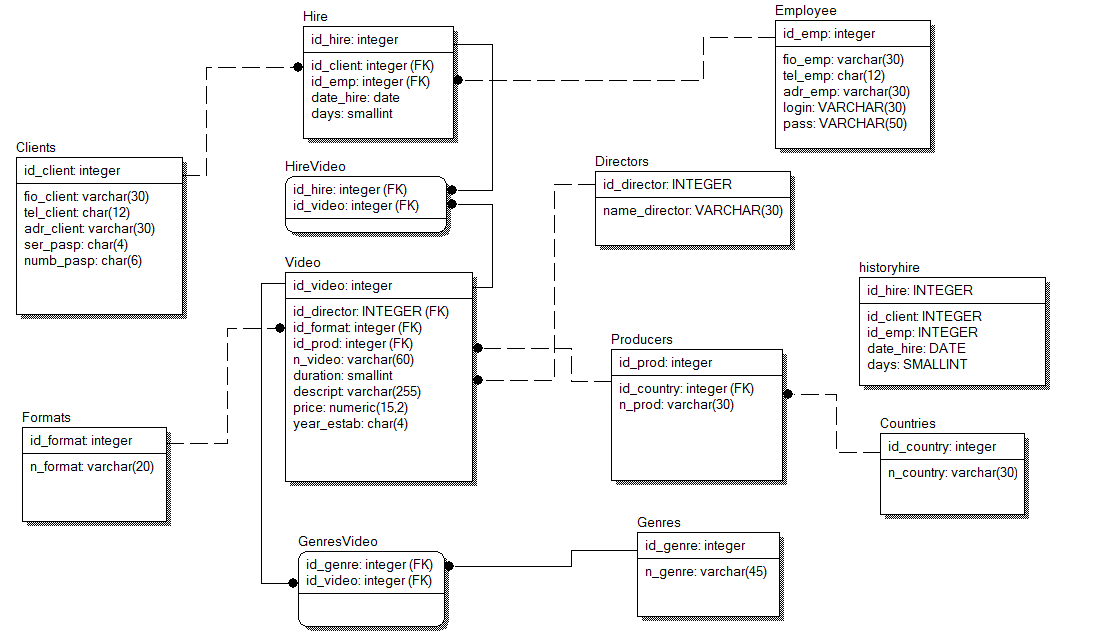
При обеспечении ссылочной целостности не допускается следующих действий пользователей.

* Добавления или изменения строк в связанной таблице, если в первичной таблице нет соответствующей строки.
* Изменения значений в первичной таблице, которое приводит к появлению потерянных строк в связанной таблице.
* Удаления строк из первичной таблицы, если имеются соответствующие ей строки в связанных таблицах.

Пользовательская целостность позволяет определять бизнес-правила, не входящие ни в одну из категорий целостности. Поддержку пользовательской целостности обеспечивают все остальные категории целостности: любые типы ограничений уровня столбца и уровня таблицы в инструкции CREATE TABLE, хранимых процедурах и триггерах.

Обычно ограничители задаются при создании таблиц. Но в дальнейшем их можно изменять, удалять или временно запрещать при помощи соответствующих команд СУБД.

На рисунке 2 представлена окончательная схема БД.



**Рисунок 2 Физическая модель**

# Разработка клиентской части информационной системы.

## Организация взаимодействия клиентской программы с БД.

На стороне клиента нет возможности выполнять запросы к базе данных, т.к. информация о подключении (логин, пароль, сервер подключения, и т.д.), хранилась бы в открытом виде. Нужно воспользоваться серверными языками программирования, такими как, PHP, Perl, Python, ASPA. Я буду использовать PHP. Что бы данные выводились без обновления страницы, прибегнем к использованию технологии AJAX.

**PHP (PHP: Hypertext Preprocessor – «PHP: препроцессор гипертекста»)** – скриптовый язык общего назначения, интенсивно применяемый для разработки веб-приложений. В настоящее время поддерживается подавляющим большинством хостинг-провайдеров и является одним из лидеров среди языков, применяющихся для создания динамических веб-сайтов.

**AJAX, Ajax (от англ. Asynchronous Javascript and XML** – **«асинхронный JavaScript и XML»)** – подход к построению интерактивных пользовательских интерфейсов веб-приложений, заключающийся в «фоновом» обмене данными браузера с веб-сервером. В результате, при обновлении данных веб-страница не перезагружается полностью, и веб-приложения становятся быстрее и удобнее. По-русски иногда произносится по существующей аналогии у имени Ajax – Аякс, но у аббревиатуры AJAX нет устоявшегося варианта на кириллице.

Пример реализация функционала с выводом и добавления фильмов.

**JS**

'use strict';

(function(){

var $form = $('form');

var $button = $('button');

var $result = $('.result-bn');

var validationFormObj = new validationForm();

$button.modalPlugin();

$button.off();

var $popupMessage = $('.popup-message-bn');

var animationIn\_ = function($this, animation, duration){

var duration = (duration !== undefined) ? duration : '';

$this.filter('.bn-hidden').removeClass('bn-hidden');

$this.not('.'+animation).addClass(animation+' '+duration);

$this.on('animationend.fadeIn', function(){

$this.filter('.'+animation).removeClass(animation+' '+duration);

$this.off('.fadeIn');

});

};

var animationOut\_ = function($this, animation, duration){

var duration = (duration !== undefined) ? duration : '';

$this.not('.'+animation).addClass(animation+' bn-animation-reverse '+duration);

$this.on('animationend.fadeOut', function(){

$this.filter('.'+animation).removeClass(animation+' bn-animation-reverse '+duration).addClass('bn-hidden');

$this.off('.fadeOut');

});

};

var outputEmployee = function(){

var ajax = $.ajax({

data: $form.serialize(),

dataType: 'json',

type: 'POST',

url: '../php/video.php'

});

ajax.done(function(result){

if (!result.status){

return false;

};

$result.html(result.data);

var $select = $('.select-bn');

$select.selectPlugin();

$select.each(function(){

var $selectListVal = $(this).children('p');

var $select = $(this).children('select');

var selectVal = $select.val();

var selectValText = $select.children('option:selected').text();

$selectListVal.text(selectValText);

});

});

};

outputEmployee();

$form.on('blur', 'input[name="n\_video"]', function(){

validationFormObj.input($(this), 'name');

});

$form.on('blur', 'input[name="duration"]', function(){

validationFormObj.input($(this), 'number');

});

$form.on('blur', 'input[name="descript"]', function(){

validationFormObj.input($(this), 'name');

});

$form.on('blur', 'input[name="price"]', function(){

validationFormObj.input($(this), 'empty');

});

$form.on('blur', 'input[name="year\_estab"]', function(){

validationFormObj.input($(this), 'number');

});

$button.on('click', function(){

var statusInput = validationFormObj.formInput($form);

if (!statusInput){

return false;

};

var ajax = $.ajax({

data: $form.serialize(),

dataType: 'json',

type: 'POST',

url: '../php/add-video.php'

});

ajax.done(function(result){

var popupData = '<p class="c-success-bn"><strong>Спасибо!</strong> Данные добавлены</p>';

if (!result.status){

popupData = '<p class="c-danger-bn"><strong>Ошибка!</strong> Данные не добавлены</p>'+

'<p>'+result.message+'</p>';

$popupMessage.html(popupData);

$button.modalPlugin('show');

return false;

};

$popupMessage.html(popupData);

$button.modalPlugin('show');

outputEmployee(popupData);

});

});

})();

**PHP скрипт для вывода видео**

<?php

include 'connect-bd.php';

$query = 'SELECT \*, group\_concat(genres.n\_genre) as n\_genres

FROM video

INNER JOIN directors using(id\_director)

INNER JOIN formats using(id\_format)

INNER JOIN producers using(id\_prod)

INNER JOIN genresvideo using(id\_video)

INNER JOIN genres using(id\_genre)

group by video.id\_video';

$res = mysql\_query($query);

$data = '<table>'.

'<tr>'.

'<th>#</th>'.

'<th>Режисер</th>'.

'<th>Формат</th>'.

'<th>Продюсер</th>'.

'<th>Жанр</th>'.

'<th>Название</th>'.

'<th>Продолжительность</th>'.

'<th>Описание</th>'.

'<th>Стоимость</th>'.

'<th>Год</th>'.

'</tr>';

while($row = mysql\_fetch\_assoc($res)){

$data.= '<tr>'.

'<td>'.$row['id\_video'].'</td>'.

'<td>'.$row['name\_director'].'</td>'.

'<td>'.$row['n\_format'].'</td>'.

'<td>'.$row['n\_prod'].'</td>'.

'<td>'.$row['n\_genres'].'</td>'.

'<td>'.$row['n\_video'].'</td>'.

'<td>'.$row['duration'].'</td>'.

'<td>'.$row['descript'].'</td>'.

'<td>'.$row['price'].'</td>'.

'<td>'.$row['year\_estab'].'</td>'.

'</tr>';

};

$data.= '<tr>'.

'<td>&nbsp;</td>'.

'<td>'.

'<div data-select-position="top" class="select-bn">'.

'<p>&nbsp;</p>'.

'<select class="hidden-bn" name="id\_director">';

$query = 'SELECT \*

FROM directors';

$res = mysql\_query($query);

while($row = mysql\_fetch\_assoc($res)){

$data.= '<option value="'.$row['id\_director'].'">'.$row['name\_director'].'</option>';

};

$data.= '</select>'.

'</div>'.

'</td>'.

'<td>'.

'<div data-select-position="top" class="select-bn">'.

'<p>&nbsp;</p>'.

'<select class="hidden-bn" name="id\_format">';

$query = 'SELECT \*

FROM formats';

$res = mysql\_query($query);

while($row = mysql\_fetch\_assoc($res)){

$data.= '<option value="'.$row['id\_format'].'">'.$row['n\_format'].'</option>';

};

$data.= '</select>'.

'</div>'.

'</td>'.

'<td>'.

'<div data-select-position="top" class="select-bn">'.

'<p>&nbsp;</p>'.

'<select class="hidden-bn" name="id\_prod">';

$query = 'SELECT \*

FROM producers';

$res = mysql\_query($query);

while($row = mysql\_fetch\_assoc($res)){

$data.= '<option value="'.$row['id\_prod'].'">'.$row['n\_prod'].'</option>';

};

$data.= '</select>'.

'</div>'.

'</td>'.

'<td>'.

'<select name="id\_genre[]" multiple>';

$query = 'SELECT \*

FROM genres';

$res = mysql\_query($query);

$selectOptionInc = 0;

while($row = mysql\_fetch\_assoc($res)){

if ($selectOptionInc === 0){

$data.= '<option selected value="'.$row['id\_genre'].'">'.$row['n\_genre'].'</option>';

$selectOptionInc++;

continue;

}

$data.= '<option value="'.$row['id\_genre'].'">'.$row['n\_genre'].'</option>';

$selectOptionInc++;

};

$data.= '</select>'.

'</td>'.

'<td><input name="n\_video" type="text"></td>'.

'<td><input name="duration" type="text"></td>'.

'<td><input name="descript" type="text"></td>'.

'<td><input name="price" type="tel"></td>'.

'<td><input name="year\_estab" type="text"></td>'.

'</tr>'.

'</table>';

$resultArr = array(

'data' => $data,

'status' => true

);

echo json\_encode($resultArr);

mysql\_close($baseConnect);

?>

**PHP скрипт для добавления видео**

<?php

include 'connect-bd.php';

$postData = $\_POST;

$postDataCount = count($postData) - 1;

$postDataInc = 0;

$columnName = '';

$columnData = '';

foreach ($postData as $key => $value){

if ($key === 'id\_genre'){

$postDataInc++;

continue;

};

$columnName.= $key;

$columnData.= '"'.$value.'"';

if ($postDataInc !== $postDataCount){

$columnName.= ', ';

$columnData.= ', ';

}

$postDataInc++;

};

$query = 'INSERT INTO video (

'.$columnName.'

)

VALUES (

'.$columnData.'

)';

$res = mysql\_query($query);

$id = mysql\_insert\_id();

if (!$res){

$resultArr = array(

'message' => mysql\_error(),

'status' => false

);

echo json\_encode($resultArr);

mysql\_close($baseConnect);

return false;

};

$ganreDataCount = count($\_POST['id\_genre']) - 1;

$columnData = '';

foreach ($\_POST['id\_genre'] as $key => $value){

$columnData.= '("'.$id.'", "'.$value.'")';

if ($key !== $ganreDataCount){

$columnData.= ', ';

}

};

$query = 'INSERT INTO genresvideo (

id\_video, id\_genre

)

VALUES

'.$columnData;

$res = mysql\_query($query);

$id = mysql\_insert\_id();

if (!$res){

$resultArr = array(

'message' => mysql\_error(),

'status' => false

);

echo json\_encode($resultArr);

mysql\_close($baseConnect);

return false;

};

$resultArr = array(

'status' => true

);

echo json\_encode($resultArr);

mysql\_close($baseConnect);

?>

## Разработка форм и отчетов

Используя «HTML», «CSS» и «JS», сверстаем шаблон. В самом низу я приведу результат, как это все будет выглядеть.

**HTML (HyperText Markup Language – «язык гипертекстовой разметки»)** – стандартизированный язык разметки документов во Всемирной паутине. Большинство веб-страниц содержат описание разметки на языке HTML (или XHTML). Язык HTML интерпретируется браузерами; полученный в результате интерпретации форматированный текст отображается на экране монитора компьютера или мобильного устройства.

**CSS (Cascading Style Sheets – каскадные таблицы стилей)** **–** формальный язык описания внешнего вида документа, написанного с использованием языка разметки.

**JS (JavaScript)** **–** мультипарадигменный язык программирования. Поддерживает объектно-ориентированный, императивный и функциональный стили. Является реализацией языка ECMAScript (стандарт ECMA-262).

Шаблон мы разделили на несколько частей:

* Шапка;
* Навигация (меню);
* Основная часть (где будет выводиться контент);
* Футер.

**Header**

'use strict';

(function(){

document.write(

'<header>'+

'<p>&nbsp;</p>'+

'<p class="hidden-bn"></p>'+

'</header>'

);

moment.updateLocale('ru', {

months : {

format: 'января\_февраля\_марта\_апреля\_мая\_июня\_июля\_августа\_сентября\_октября\_ноября\_декабря'.split('\_'),

standalone: 'январь\_февраль\_март\_апрель\_май\_июнь\_июль\_август\_сентябрь\_октябрь\_ноябрь\_декабрь'.split('\_')

},

weekdays : {

standalone: 'Воскресенье\_Понедельник\_Вторник\_Среда\_Четверг\_Пятница\_Суббота'.split('\_'),

format: 'Воскресенье\_Понедельник\_Вторник\_Среду\_Четверг\_Пятницу\_Субботу'.split('\_'),

isFormat: /\[ ?[Вв] ?(?:прошлую|следующую|эту)? ?\] ?dddd/

}

});

var $window = $(window);

var $header = $('header');

var $headerText = $header.children('p:first-child');

var $headerLogin = $header.children('p:last-child');

var headerTop = $header.offset().top;

var headerHeight = $header.outerHeight();

var headerBotoomLine = headerTop + headerHeight;

var locationPathname = window.location.pathname;

var locationSearchReg = new RegExp(/sign-on.htm/i);

var locationSearchResult = locationSearchReg.test(locationPathname);

var escapeReg = function(string){

return string.replace(/[.\*+?^${}()|[\]\\]/g, '\\$&');

};

(function curentTime(){

$headerText.text(moment().format('dddd, D MMMM YYYY года. Время: H:mm:ss'))

setTimeout(function(){

curentTime();

}, 1000)

})();

var ajax = $.ajax({

dataType: 'json',

url: '../php/header.php'

});

ajax.done(function(result){

if (!locationSearchResult && !result.status){

window.location.replace('../html/sign-on.htm');

} else if (locationSearchResult && result.status){

window.location.replace('../html/employee.htm');

};

if (!result.status){

return false;

};

$headerLogin.html('<strong>Вы вошли как:</strong> '+result.login);

$headerLogin.removeClass('hidden-bn');

});

$window.on('scroll resize load', function(){

var scrollTop = $window.scrollTop();

var windowHeight = $window.height();

if (scrollTop > headerBotoomLine){

$header.not('.active-bn').addClass('active-bn slide-top-bn');

} else if (scrollTop === headerTop){

$header.filter('.active-bn').removeClass('active-bn slide-top-bn');

}

});

})();

**Head**

'use strict';

(function(){

var titleSelector = document.querySelector('body > div.title');

var title = titleSelector.innerText;

document.write(

'<head>'+

'<title>BN| '+title+'</title>'+

'<meta charset="utf-8">'+

'<meta name="viewport" content="width=device-width, user-scalable=no">'+

'<base href="">'+

'<meta name="description" content="">'+

'<meta name="keywords" content="">'+

'<link rel="shortcut icon" href="/">'+

'<link rel="apple-touch-icon-precomposed" href="/">'+

'<link rel="stylesheet" href="../layout/css/normalize.css">'+

'<link rel="stylesheet" href="../layout/css/grid-bn.css">'+

// '<link rel="stylesheet" href="../layout/css/font-awesome.min.css">'+

'<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-awesome.min.css">'+

'<link rel="stylesheet" href="../layout/css/layout.css">'+

'<link rel="stylesheet" href="../layout/css/typography.css">'+

'<script src="../layout/js/moment.min.js"></script>'+

'<script src="../layout/js/jquery-3.1.1.min.js"></script>'+

'</head>'

);

})();

**Nav**

'use strict';

(function(){

var navArray = [{

"title": "Управления работниками",

"url": "employee.htm"

},{

"title": "Управления режиссерами",

"url": "directors.htm"

},{

"title": "Управления странами",

"url": "countries.htm"

},{

"title": "Управления жанрами",

"url": "genres.htm"

},{

"title": "Управления форматами",

"url": "formats.htm"

},{

"title": "Управления клиентами",

"url": "clients.htm"

},{

"title": "Управления продюсерами",

"url": "producers.htm"

},{

"title": "Управления видео",

"url": "video.htm"

},{

"title": "Выдать в прокат",

"url": "hire.htm"

},{

"title": "Выйти",

"class": 'sign-out-bn',

"url": "javascript:void(0);"

}];

var locationPathname = window.location.pathname;

var escapeReg = function(string){

return string.replace(/[.\*+?^${}()|[\]\\]/g, '\\$&');

};

var textWrite = '<div class="col-12-bn col-md-4-bn col-lg-3-bn">'+

'<nav>'+

'<ul>'+

'<li><h2>Меню</h2></li>';

$.each(navArray, function(){

var locationSearchReg = new RegExp(escapeReg(this.url));

var locationSearchResult = locationSearchReg.test(locationPathname);

var className = (this.class !== undefined) ? this.class : '';

textWrite+= (locationSearchResult) ? '<li class="active-bn'+className+'"><p>'+this.title+'</p></li>' :

'<li><a class="'+className+'" href="'+this.url+'">'+this.title+'</a></li>';

});

textWrite+= '</ul>'+

'</nav>'+

'</div>';

document.write(textWrite);

var $signOut = $('.sign-out-bn');

$signOut.on('click', function(){

var ajax = $.ajax({

dataType: 'json',

url: '../php/sign-out.php'

});

ajax.done(function(result){

if (!result.status){

return false;

};

window.location.replace('../html/sign-on.htm');

});

});

})();

**Main**

<!DOCTYPE html>

<html>

<meta charset="utf-8">

<body>

<div class="title hidden-bn">Управления видео</div>

<script src="../layout/js/head.js"></script>

<section>

<script src="../layout/js/header.js"></script>

<main class="row-bn no-gutters-bn">

<script src="../layout/js/nav.js"></script>

<div class="col-12-bn col-md-8-bn col-lg-9-bn">

<article>

<h2>Управления видео</h2>

<form name="employee">

<div class="row-bn justify-content-center-bn">

<div class="col-12-bn">

<div class="box-bn my-10-bn">

<div class="result-bn"></div>

</div>

</div>

</div>

<div class="row-bn justify-content-center-bn">

<div class="col-auto-bn">

<div class="box-bn my-10-bn">

<button data-modal-popup-id="popup-bn" type="button">Добавить</button>

</div>

</div>

</div>

</form>

</article>

</div>

</main>

<script src="../layout/js/footer.js"></script>

</section>

<div id="popup-bn" class="hidden-bn">

<div class="popup-message-bn"></div>

</div>

</body>

<script src="../layout/js/device.min.js"></script>

<script src="../layout/js/modal-plugin.js"></script>

<script src="../layout/js/select-plugin.js"></script>

<script src="../js/validation.js"></script>

<script src="../js/video.js"></script>

</html>

**Footer**

'use strict';

(function(){

document.write(

'<footer>'+

'<p><i class="fa fa-copyright"></i> Все права защищены!</p>'+

'<span class="top-bn"><i class="fa fa-chevron-up fa-2x"></i></span>'+

'</footer>'

);

var $footerTop = $('footer span.top-bn');

$footerTop.click(function(){

$('html, body').animate({

scrollTop: '0'

}, 'slow');

return false;

});

})();

Также сразу напишем два плагина для модальных окон и для стилизации элемента select. Скрипт для валидации форм, для удобства, вынесем в отдельный файл.

**Modal-plugin**

'use strict';

(function($) {

var $window = $(window);

var $document = $(document);

var $html = $('html');

var $body = $('body');

var modalInc = 0;

var methods = {

init: function(){

return this.each(function(){

if ($(this).data('modalId') !== undefined){

return;

};

var $self = $(this);

var modalIdData = 'modal-'+modalInc+'-bn';

$self.attr('data-modal-id', modalIdData).data('modalId', modalIdData);

var $modalDialog = pasteModalFunc\_($self, modalIdData).children('.modal-dialog-bn');

var $close = $modalDialog.children('.close-bn');

$self.on('click.modalPlugin', function(){

$self.modalPlugin('show');

});

$close.on('click.modalPlugin', function(){

$self.modalPlugin('hide');

});

$document.on('click.modalPlugin', function(e){

if ($(e.target).is('.modal-bn') && !$(e.target).closest('.modal-dialog-bn').length){

$self.modalPlugin('hide');

};

});

modalInc++;

});

},

show: function(){

return this.each(function(){

var modalIdData = $(this).data('modalId');

if ($('#'+modalIdData).is(':visible')){

return;

};

var $modal = $('#'+modalIdData);

var $modalDialog = $modal.children('.modal-dialog-bn');

var widthWindow = $window.width();

$html.addClass('modal-page-bn');

var paddingRigth = $window.width() - widthWindow;

$body.css('padding-right', paddingRigth+'px');

$modal.not('.show-bn').addClass('show-bn');

animationIn\_($modalDialog, 'slide-top-bn', 'animation-medium-bn');

});

},

hide: function(){

return this.each(function(){

var modalIdData = $(this).data('modalId');

if (!$('#'+modalIdData).is(':visible')){

return;

};

var $modal = $('#'+modalIdData);

var $modalDialog = $modal.children('.modal-dialog-bn');

animationOut\_($modalDialog, 'slide-top-bn');

animationOut\_($modal, 'fade-bn');

$modal.on('animationend.fadeOut', function(){

$modal.filter('.show-bn').removeClass('show-bn');

$body.css('padding-right', '');

$html.removeClass('modal-page-bn');

$modal.off('.fadeOut');

});

});

},

destroy: function(){

return this.each(function(){

if ($(this).data('modalId') === undefined){

return;

};

$(this).removeAttr('data-modal-id').removeData('modalId');

$(this).off('.modalPlugin');

modalInc--;

if(modalInc === 0) {

$(document).off('.modalPlugin');

}

});

}

};

var pasteModalFunc\_ = function($this, modalIdData){

var modalPaste = $('#'+$this.data('modalPopupId')).html();

var modalBlock = '<div id="'+modalIdData+'" class="modal-bn">'+

'<div class="modal-dialog-bn">'+

'<span class="close-bn"></span>'+

modalPaste+

'</div>'+

'</div>';

$body.append(modalBlock);

var $modal = $('#'+modalIdData);

return $modal;

};

var animationIn\_ = function($this, animation, duration){

var duration = (duration !== undefined) ? duration : '';

$this.filter('.hidden-bn').removeClass('hidden-bn');

$this.not('.'+animation).addClass(animation+' '+duration);

$this.on('animationend.fadeIn', function(){

$this.filter('.'+animation).removeClass(animation+' '+duration);

$this.off('.fadeIn');

});

};

var animationOut\_ = function($this, animation, duration){

var duration = (duration !== undefined) ? duration : '';

$this.not('.'+animation).addClass(animation+' animation-reverse-bn '+duration);

$this.on('animationend.fadeOut', function(){

$this.filter('.'+animation).removeClass(animation+' animation-reverse-bn '+duration).addClass('hidden-bn');

$this.off('.fadeOut');

});

};

$.fn.modalPlugin = function(method){

if (methods[method]){

return methods[method].apply(this, Array.prototype.slice.call(arguments, 1));

} else if (typeof method === 'object' || !method){

return methods.init.apply(this, arguments);

} else {

$.error('Метод "' + method + '" не найден в плагине jQuery.modalPlugin');

};

};

})(jQuery);

**Select-plugin**

'use strict';

(function($) {

var selectInc = 0;

var methods = {

init: function(){

return this.each(function(){

if ($(this).data('selectId') !== undefined){

return;

};

var $self = $(this);

var selectIdData = 'select-'+selectInc+'-bn';

$self.attr('data-select-id', selectIdData).data('selectId', selectIdData);

var $selectListOption = pasteSelectFunc\_($(this), selectIdData).children('li');

var $select = $(this).children('select');

switch (device.desktop()){

case true:

$(this).on('click.selectPlugin', function(){

if (!$self.hasClass('open-bn')){

$(this).selectPlugin('show');

} else {

$(this).selectPlugin('hide');

}

});

$selectListOption.on('click.selectPlugin', function(){

$(this).selectPlugin('change');

});

$(document).on('click.selectPlugin', function(e){

if (!$(e.target).is($self) && !$(e.target).closest('.select-list-bn').length){

$self.selectPlugin('hide');

};

});

break;

case false:

$select.filter('.hidden-bn').removeClass('hidden-bn');

$select.on('change.selectPlugin', function(){

$(this).selectPlugin('changeSelect');

});

};

selectInc++;

});

},

show: function(){

return this.each(function(){

var selectIdData = $(this).data('selectId');

if ($('#'+selectIdData).is(':visible')){

return;

};

var $selectList = $('#'+selectIdData);

var position = ($(this).data('selectPosition') !== undefined) ? $(this).data('selectPosition') : 'top';

var classAnimation;

switch(position){

case 'top':

$selectList.filter('.bottom-bn').removeClass('bottom-bn');

classAnimation = 'slide-top-medium-bn';

break;

case 'bottom':

$selectList.not('.bottom-bn').addClass('bottom-bn');

classAnimation = 'slide-bottom-medium-bn';

};

$(this).not('.open-bn').addClass('open-bn');

animationIn\_($selectList, classAnimation);

});

},

change: function(){

return this.each(function(){

var $selectListOption = $(this).closest('.select-list-bn').children('li');

var $select = $(this).closest('.select-list-bn').prev('select');

var selectListVal = $(this).data('value');

$selectListOption.filter('.active-bn').not($(this)).removeClass('active-bn');

$(this).not('.active-bn').addClass('active-bn');

$select.val(selectListVal);

$select.selectPlugin('changeSelect');

});

},

changeSelect: function(){

return this.each(function(){

var $selectListVal = $(this).prev('p');

var selectVal = $(this).val()

var selectValText = $(this).children('option[value="'+selectVal+'"]').text();

var $select = $(this).closest('.select-bn');

$selectListVal.text(selectValText);

$select.filter('.placeholder-bn').removeClass('placeholder-bn');

$select.selectPlugin('hide');

});

},

hide: function(){

return this.each(function(){

var selectIdData = $(this).data('selectId');

if (!$('#'+selectIdData).is(':visible')){

return;

};

var $selectList = $('#'+selectIdData);

var position = ($(this).data('selectPosition') !== undefined) ? $(this).data('selectPosition') : 'top';

var classAnimation;

switch(position){

case 'top':

classAnimation = 'slide-top-medium-bn';

break;

case 'bottom':

classAnimation = 'slide-bottom-medium-bn';

};

animationOut\_($selectList, classAnimation);

$(this).filter('.open-bn').removeClass('open-bn');

});

},

destroy: function(){

return this.each(function(){

if ($(this).data('selectStatus') === undefined){

return;

};

var $selectList = $(this).children('.select-list-bn');

$selectList.remove();

$(this).filter('.open-bn').removeClass('open-bn');

$(this).off('.selectPlugin');

$(this).removeAttr('data-select-status').removeData('selectStatus');

selectInc--;

if(selectInc === 0) {

$(document).off('.modalPlugin');

}

});

}

};

var pasteSelectFunc\_ = function($this, selectIdData){

var $select = $this.children('select');

var selectHtml = $select.html();

var replaceOptionReg = /option/gi;

var replaceValueReg = /value/gi;

var selectListOption = selectHtml.replace(replaceOptionReg, 'li');

var selectList = '<ul id="'+selectIdData+'" class="select-list-bn hidden-bn">'+

selectListOption.replace(replaceValueReg, 'data-value');

'</ul>';

$this.append(selectList);

var $selectList = $('#'+selectIdData);

return $selectList;

};

var animationIn\_ = function($this, animation, duration){

var duration = (duration !== undefined) ? duration : '';

$this.filter('.hidden-bn').removeClass('hidden-bn');

$this.not('.'+animation).addClass(animation+' '+duration);

$this.on('animationend.fadeIn', function(){

$this.filter('.'+animation).removeClass(animation+' '+duration);

$this.off('.fadeIn');

});

};

var animationOut\_ = function($this, animation, duration){

var duration = (duration !== undefined) ? duration : '';

$this.not('.'+animation).addClass(animation+' animation-reverse-bn '+duration);

$this.on('animationend.fadeOut', function(){

$this.filter('.'+animation).removeClass(animation+' animation-reverse-bn '+duration).addClass('hidden-bn');

$this.off('.fadeOut');

});

};

$.fn.selectPlugin = function(method){

if (methods[method]){

return methods[method].apply(this, Array.prototype.slice.call(arguments, 1));

} else if (typeof method === 'object' || !method){

return methods.init.apply(this, arguments);

} else {

$.error('Метод "' + method + '" не найден в плагине jQuery.selectPlugin');

};

};

})(jQuery);

**Validation**

'use strict';

var validationForm = function(){

this.emptyReg = /^.+$/i;

this.nameReg = /^[а-яёa-z][а-яёa-z-\s]\*$/i;

this.telReg = /^((8|\+7)[\- ]?)?(\(?\d{3}\)?[\- ]?)?[\d\- ]{7,10}$/;

this.loginReg = /^[a-z\d@-\_]+$/i;

this.numberReg = /^\d{1,10}$/i;

this.dateReg = /^\d{4}-\d{2}-\d{2}$/i;

};

validationForm.prototype.input = function($this, type){

var self = this;

var reg;

switch (type){

case 'empty':

reg = self.emptyReg;

break;

case 'name':

reg = self.nameReg;

break;

case 'tel':

reg = self.telReg;

break;

case 'login':

reg = self.loginReg;

break;

case 'number':

reg = self.numberReg;

break;

case 'date':

reg = self.dateReg;

};

if ($this.val().trim() && reg.test($this.val())){

self.success\_($this);

} else {

self.error\_($this);

};

};

validationForm.prototype.select = function($this){

var self = this;

var $select = $this.parent('.select-bn');

if (!$select.hasClass('placeholder-bn')){

self.success\_($select);

};

};

validationForm.prototype.formInput = function($form){

var $input = $form.find('input');

var status = true;

$input.each(function(){

$(this).trigger('blur');

if($(this).hasClass('b-c-danger-bn')){

status = false;

};

});

return status;

};

validationForm.prototype.formSelect = function($form){

var self = this;

var $select = $form.find('.select-bn');

var status = true;

$select.each(function(){

if($(this).hasClass('placeholder-bn')){

self.error\_($select);

status = false;

};

});

return status;

};

validationForm.prototype.success\_ = function($this){

$this.filter('.b-c-danger-bn').removeClass('b-c-danger-bn')

$this.not('.b-c-success-bn').addClass('b-c-success-bn')

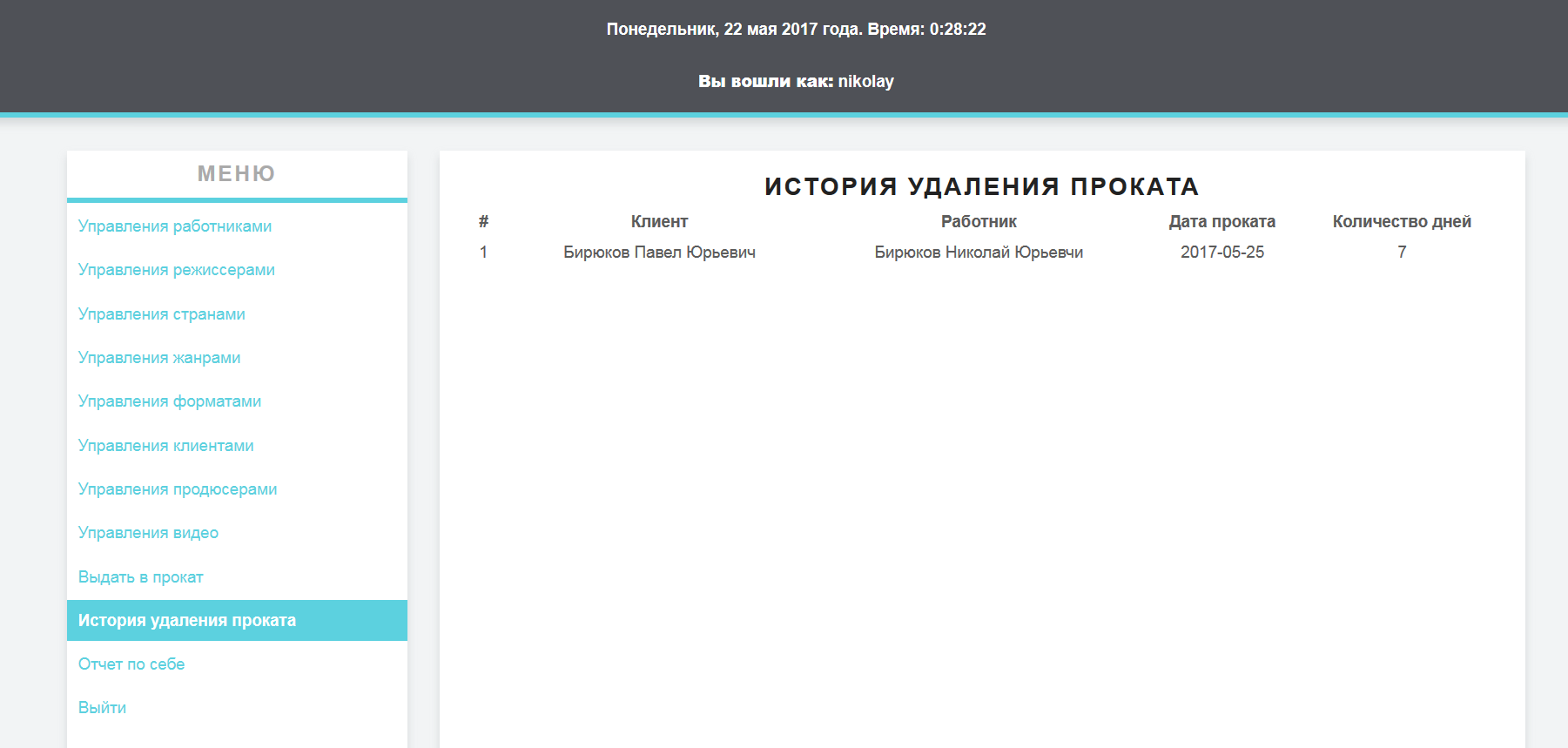
};

validationForm.prototype.error\_ = function($this){

$this.filter('.b-c-success-bn').removeClass('b-c-success-bn')

$this.not('.b-c-danger-bn').addClass('b-c-danger-bn')

};



## Разработка сценария инсталляции клиентской программы

В связи с тем, что для доступа к клиентской части нужен только браузер, мы не стали создавать инсталляционный пакет. Взамен этому, мы предлагаем воспользоваться предустановленным приложением (Internet Explorer) или же скачать по ссылкам ниже.

**Mozilla Firefox**

<https://www.mozilla.org/ru/firefox/new/>

**Google Chrome**

<https://www.google.ru/chrome/browser/desktop/index.html>

Как мы и писали выше, нужен только интернет обозреватель с поддержкой HTML5, CSS3 и JavaScript, ну и кончено доступ интернет.

Наш портал по выдачи фильмов в прокат доступен по ссылке <http://test.mtrans62.ru/>.

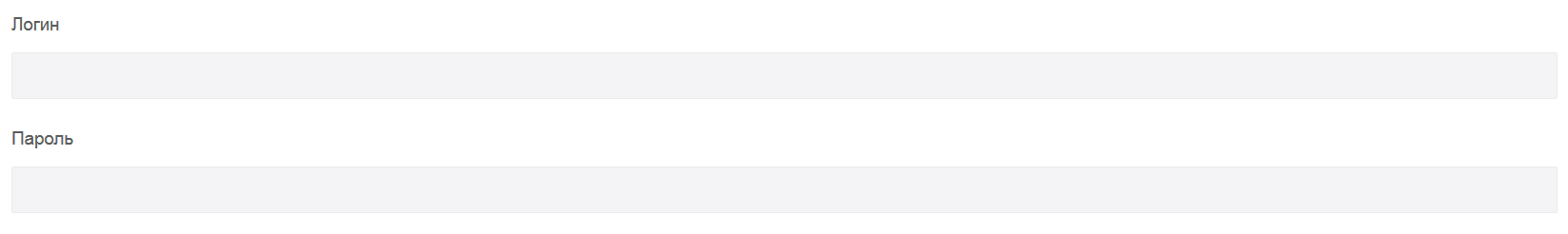
Логин и пароль для доступа к сайту **admin/test**.

Логин и пароль для доступа к информации **nikolay/4041**, альтернативные данные для входа **admin/4041**.

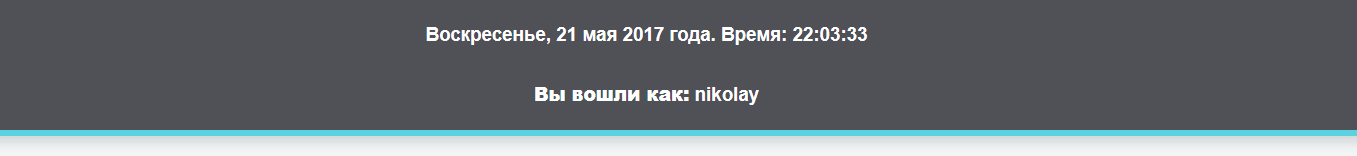
## Руководство пользователя

В целом интерес разработанного портала: клиент ориентированный, интуитивный и понятный, как наши друзья из Европы говорят «User Friendly». Ниже я приведу скрины элементов управления, полей для ввода и т.д.

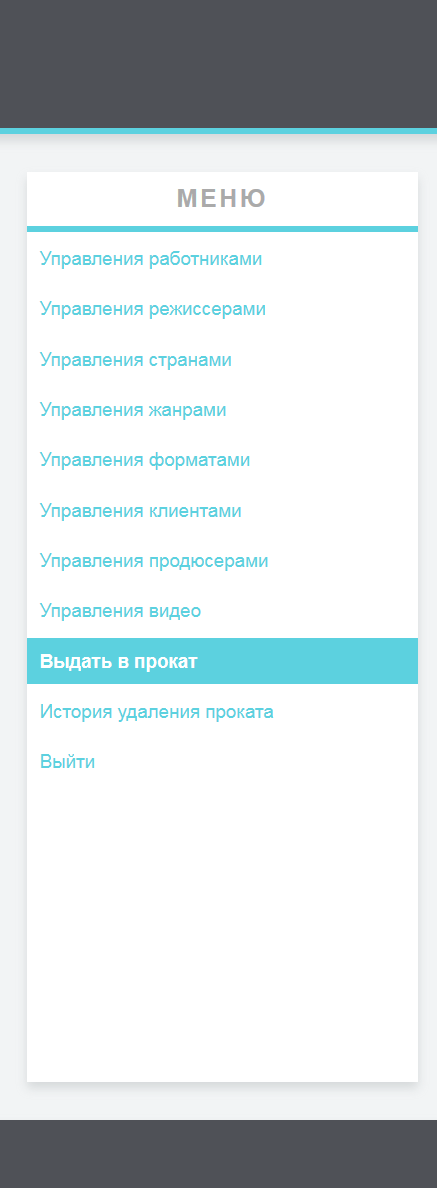
**Поля для ввода информации**



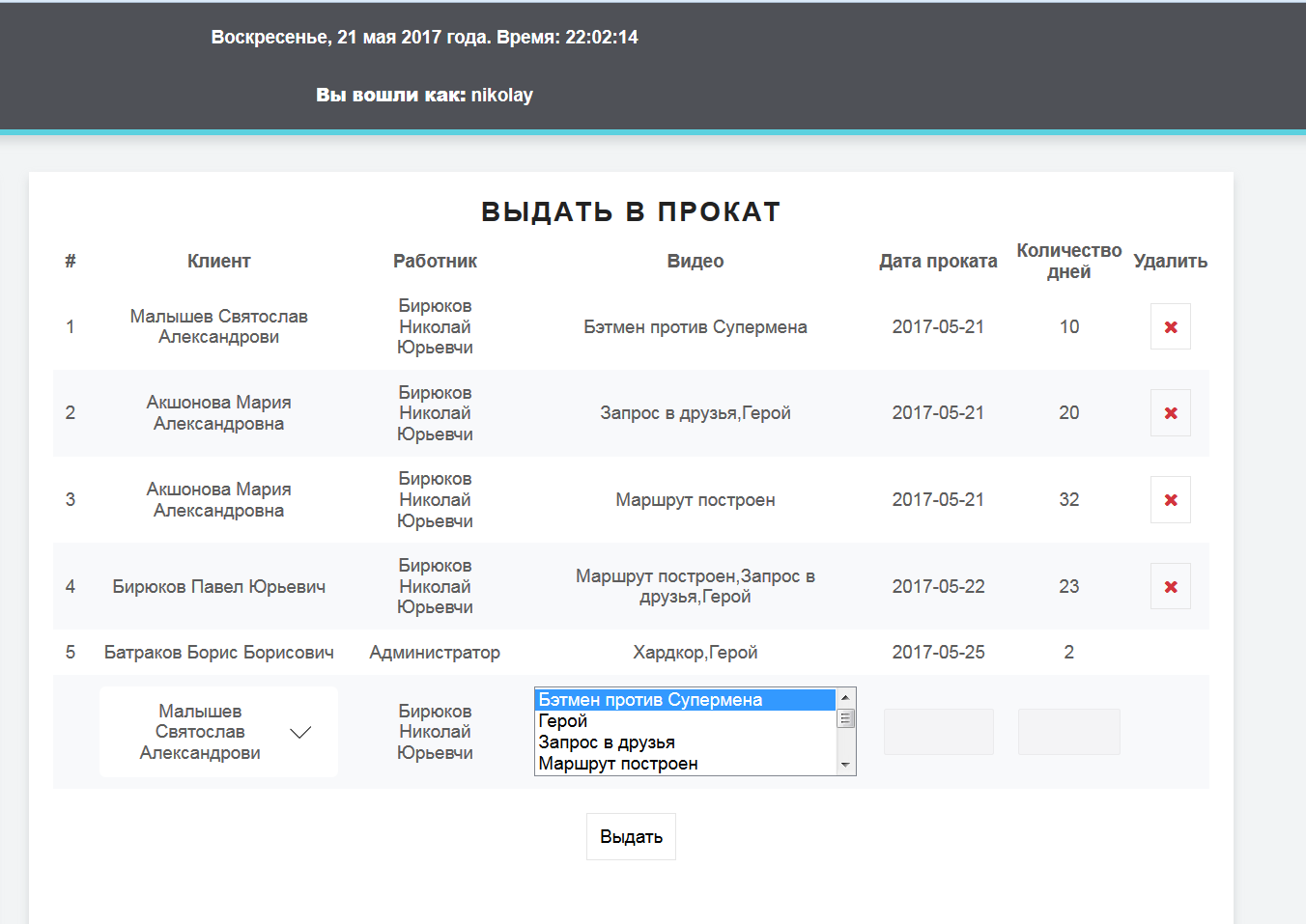
**Информационная область**



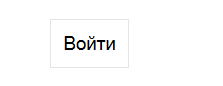
**Область навигации**



**Основная область**



**Кнопка действия, вход/добавить/выдать**



**Кнопка удаления записи**



# Заключение

В результате проделанной работы, если получилось даже не полноценная система по выдачи фильмов в прокат, то хорошая база для развития и дальнейшего вывода проекта в PROD.

Мы познакомили Вас с хранимыми процедурами, их не много не мало, целых две. Также показали, что такое триггер и для чего он используется, в нашем примере он срабатывает по событию Delete (удаления записи). Благодаря чему происходит логирвоание удаленных данных.

Разобрали, что такое HTML, CSS, JS. Написали несколько плагинов с использованием JQuery, включающие различные методы (инициализация, показа и скрытия блока, дестроя).

Показали, что такое куки сессии и зачем они нужны, без них обойтись невозможно, они используются в каждом «серьезном» проекте.

Цель данного курсового проекта, была вовсе не написать приложения «супер» защищенное и без единой уязвимости, а познакомиться c языками программирования, которые используются в WEB сфере. Попробовать, «потрогать» различные технологии, опять же не стоит забывать о том что, что такое AJAX, он на сегодняшний день используется на 99,99 процентах интернета. Без него вообще очень трудно представить современный сайт.

Безусловно, хотелось бы рассказать и показать, другие интересные технологии, такие, какие как: Node.js, Angular и т.п. Но тогда стоило бы задуматься о выпуске учебника, вероятнее всего многотомного.

# Список использованных источников

* Алан Бьюли, Энди Оппель. Изучаем SQL. – М.: Символ, 2007. – 312 с.
* Жилинский А. Самоучитель Microsoft SQL Server 2005. – СПб:BHV, 2004. – 224 с.
* Кишори Бхамидипати. SQL. Справочник программиста. – М.: Эком, 2003. – 304 с.
* Роберт Виейра. Программирование баз данных Microsoft SQL Server 2005. Базовый курс. – М.: Вильямс, 2003. – 848 с.
* Уилтон П., Колби Дж. SQL для начинающих. – М.: Вильямс, 2006. – 496 с.
* Фленов М. Transact SQL в подлиннике. – СПб: BHV, 2006. – 576 с.
* Практическое руководство по SQL. – М.: Вильямс, 2002. – 352 с.
* Моисеенко С. SQL. Задачи и решения. СПб.: Питер, 2006. – 256 с.
* Крис Джамса, Конрад Кинг, Энди Андерсон: «Эффективный самоучитель по креативному Web-дизайну. HTML, XHTML, CSS, JavaScript, PHP, ASP, ActiveX. Текст, графика, звук и анимация» – М.: «ДиаСофтЮП», 2005.
* Дэвид Флэнаган: «JavaScript. Подробное руководство» – СПб: «Символ-Плюс», 2008.
* Влад Мержевич: «HTML и CSS на примерах» – СПб: «БХВ-Петербург», 2005.
* Артемий Ломов: «HTML, CSS, скрипты: практика создания сайтов» – СПб: «БХВ-Петербург», 2007.
* >Кристофер Шмитт: «CSS. Рецепты программирования» – СПб: «БХВ-Петербург», 2007.
* Интернет ресурсы:

<https://webref.ru/>

<http://htmlbook.ru/>

<http://cssbook.ru>

<http://zvirec.com/>

<http://htmlsam.ru>

<http://JavaScriptbook.ru>

# Приложение

## SQL скрипт для создания базы данных

-- phpMyAdmin SQL Dump

-- http://www.phpmyadmin.net

SET SQL\_MODE = "NO\_AUTO\_VALUE\_ON\_ZERO";

SET time\_zone = "+00:00";

/\*!40101 SET @OLD\_CHARACTER\_SET\_CLIENT=@@CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET @OLD\_CHARACTER\_SET\_RESULTS=@@CHARACTER\_SET\_RESULTS \*/;

/\*!40101 SET @OLD\_COLLATION\_CONNECTION=@@COLLATION\_CONNECTION \*/;

/\*!40101 SET NAMES utf8 \*/;

--

-- База данных: `videoteka`

--

-- --------------------------------------------------------

--

-- Структура таблицы `Clients`

--

CREATE TABLE IF NOT EXISTS `Clients` (

`id\_client` int(11) NOT NULL AUTO\_INCREMENT,

`fio\_client` varchar(30) DEFAULT NULL,

`tel\_client` char(12) DEFAULT NULL,

`adr\_client` varchar(30) DEFAULT NULL,

`ser\_pasp` char(4) DEFAULT NULL,

`numb\_pasp` char(6) DEFAULT NULL,

PRIMARY KEY (`id\_client`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO\_INCREMENT=1 ;

-- --------------------------------------------------------

--

-- Структура таблицы `Countries`

--

CREATE TABLE IF NOT EXISTS `Countries` (

`id\_country` int(11) NOT NULL AUTO\_INCREMENT,

`n\_country` varchar(30) DEFAULT NULL,

PRIMARY KEY (`id\_country`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO\_INCREMENT=1 ;

-- --------------------------------------------------------

--

-- Структура таблицы `Directors`

--

CREATE TABLE IF NOT EXISTS `Directors` (

`id\_director` int(11) NOT NULL AUTO\_INCREMENT,

`name\_director` varchar(30) DEFAULT NULL,

PRIMARY KEY (`id\_director`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO\_INCREMENT=1 ;

-- --------------------------------------------------------

--

-- Структура таблицы `Employee`

--

CREATE TABLE IF NOT EXISTS `Employee` (

`id\_emp` int(11) NOT NULL AUTO\_INCREMENT,

`fio\_emp` varchar(30) DEFAULT NULL,

`tel\_emp` char(12) DEFAULT NULL,

`adr\_emp` varchar(30) DEFAULT NULL,

`login` varchar(30) DEFAULT NULL,

`pass` varchar(50) DEFAULT NULL,

PRIMARY KEY (`id\_emp`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO\_INCREMENT=1 ;

-- --------------------------------------------------------

--

-- Структура таблицы `Formats`

--

CREATE TABLE IF NOT EXISTS `Formats` (

`id\_format` int(11) NOT NULL AUTO\_INCREMENT,

`n\_format` varchar(20) DEFAULT NULL,

PRIMARY KEY (`id\_format`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO\_INCREMENT=1 ;

-- --------------------------------------------------------

--

-- Структура таблицы `Genres`

--

CREATE TABLE IF NOT EXISTS `Genres` (

`id\_genre` int(11) NOT NULL AUTO\_INCREMENT,

`n\_genre` varchar(45) DEFAULT NULL,

PRIMARY KEY (`id\_genre`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO\_INCREMENT=1 ;

-- --------------------------------------------------------

--

-- Структура таблицы `GenresVideo`

--

CREATE TABLE IF NOT EXISTS `GenresVideo` (

`id\_genre` int(11) NOT NULL,

`id\_video` int(11) NOT NULL,

PRIMARY KEY (`id\_genre`,`id\_video`),

KEY `R\_12` (`id\_video`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- --------------------------------------------------------

--

-- Структура таблицы `Hire`

--

CREATE TABLE IF NOT EXISTS `Hire` (

`id\_hire` int(11) NOT NULL AUTO\_INCREMENT,

`date\_hire` date DEFAULT NULL,

`id\_emp` int(11) NOT NULL,

`id\_client` int(11) NOT NULL,

`days` smallint(6) DEFAULT NULL,

PRIMARY KEY (`id\_hire`),

KEY `R\_5` (`id\_emp`),

KEY `R\_6` (`id\_client`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO\_INCREMENT=1 ;

-- --------------------------------------------------------

--

-- Структура таблицы `HireVideo`

--

CREATE TABLE IF NOT EXISTS `HireVideo` (

`id\_hire` int(11) NOT NULL,

`id\_video` int(11) NOT NULL,

PRIMARY KEY (`id\_hire`,`id\_video`),

KEY `R\_9` (`id\_video`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- --------------------------------------------------------

--

-- Структура таблицы `Producers`

--

CREATE TABLE IF NOT EXISTS `Producers` (

`id\_prod` int(11) NOT NULL AUTO\_INCREMENT,

`n\_prod` varchar(30) DEFAULT NULL,

`id\_country` int(11) NOT NULL,

PRIMARY KEY (`id\_prod`),

KEY `R\_1` (`id\_country`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO\_INCREMENT=1 ;

-- --------------------------------------------------------

--

-- Структура таблицы `Video`

--

CREATE TABLE IF NOT EXISTS `Video` (

`id\_video` int(11) NOT NULL AUTO\_INCREMENT,

`n\_video` varchar(60) DEFAULT NULL,

`duration` smallint(6) DEFAULT NULL,

`descript` varchar(255) DEFAULT NULL,

`price` decimal(15,2) DEFAULT NULL,

`id\_prod` int(11) NOT NULL,

`year\_estab` char(4) DEFAULT NULL,

`id\_format` int(11) NOT NULL,

`id\_director` int(11) NOT NULL,

PRIMARY KEY (`id\_video`),

KEY `R\_2` (`id\_prod`),

KEY `R\_4` (`id\_format`),

KEY `R\_10` (`id\_director`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO\_INCREMENT=1 ;

--

-- Ограничения внешнего ключа сохраненных таблиц

--

--

-- Ограничения внешнего ключа таблицы `GenresVideo`

--

ALTER TABLE `GenresVideo`

ADD CONSTRAINT `genresvideo\_ibfk\_1` FOREIGN KEY (`id\_genre`) REFERENCES `Genres` (`id\_genre`),

ADD CONSTRAINT `genresvideo\_ibfk\_2` FOREIGN KEY (`id\_video`) REFERENCES `Video` (`id\_video`);

--

-- Ограничения внешнего ключа таблицы `Hire`

--

ALTER TABLE `Hire`

ADD CONSTRAINT `hire\_ibfk\_1` FOREIGN KEY (`id\_emp`) REFERENCES `Employee` (`id\_emp`),

ADD CONSTRAINT `hire\_ibfk\_2` FOREIGN KEY (`id\_client`) REFERENCES `Clients` (`id\_client`);

--

-- Ограничения внешнего ключа таблицы `HireVideo`

--

ALTER TABLE `HireVideo`

ADD CONSTRAINT `hirevideo\_ibfk\_1` FOREIGN KEY (`id\_hire`) REFERENCES `Hire` (`id\_hire`),

ADD CONSTRAINT `hirevideo\_ibfk\_2` FOREIGN KEY (`id\_video`) REFERENCES `Video` (`id\_video`);

--

-- Ограничения внешнего ключа таблицы `Producers`

--

ALTER TABLE `Producers`

ADD CONSTRAINT `producers\_ibfk\_1` FOREIGN KEY (`id\_country`) REFERENCES `Countries` (`id\_country`);

--

-- Ограничения внешнего ключа таблицы `Video`

--

ALTER TABLE `Video`

ADD CONSTRAINT `video\_ibfk\_1` FOREIGN KEY (`id\_prod`) REFERENCES `Producers` (`id\_prod`),

ADD CONSTRAINT `video\_ibfk\_2` FOREIGN KEY (`id\_format`) REFERENCES `Formats` (`id\_format`),

ADD CONSTRAINT `video\_ibfk\_3` FOREIGN KEY (`id\_director`) REFERENCES `Directors` (`id\_director`);

/\*!40101 SET CHARACTER\_SET\_CLIENT=@OLD\_CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET CHARACTER\_SET\_RESULTS=@OLD\_CHARACTER\_SET\_RESULTS \*/;

/\*!40101 SET COLLATION\_CONNECTION=@OLD\_COLLATION\_CONNECTION \*/;

## SQL скрипт для создания базы данных с данными, хранимыми процедурами и функциями

-- phpMyAdmin SQL Dump

-- https://www.phpmyadmin.net/

SET SQL\_MODE = "NO\_AUTO\_VALUE\_ON\_ZERO";

SET time\_zone = "+00:00";

/\*!40101 SET @OLD\_CHARACTER\_SET\_CLIENT=@@CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET @OLD\_CHARACTER\_SET\_RESULTS=@@CHARACTER\_SET\_RESULTS \*/;

/\*!40101 SET @OLD\_COLLATION\_CONNECTION=@@COLLATION\_CONNECTION \*/;

/\*!40101 SET NAMES utf8mb4 \*/;

--

-- База данных: `vozimbdr\_test`

--

DELIMITER $$

--

-- Процедуры

--

DROP PROCEDURE IF EXISTS `inputHistoryhire`$$

CREATE DEFINER=`vozimbdr\_test`@`localhost` PROCEDURE `inputHistoryhire` () NO SQL

SELECT \*

FROM historyhire

INNER JOIN clients using(id\_client)

INNER JOIN employee using(id\_emp)$$

DROP PROCEDURE IF EXISTS `inputReportEmployee`$$

CREATE DEFINER=`vozimbdr\_test`@`localhost` PROCEDURE `inputReportEmployee` () NO SQL

SELECT \*, group\_concat(video.n\_video) as n\_videos

FROM hire

INNER JOIN clients using(id\_client)

INNER JOIN employee using(id\_emp)

INNER JOIN hirevideo using(id\_hire)

INNER JOIN video using(id\_video)

group by hire.id\_hire$$

DELIMITER ;

-- --------------------------------------------------------

--

-- Структура таблицы `clients`

--

-- Создание: Май 21 2017 г., 17:58

-- Последнее обновление: Май 21 2017 г., 18:58

--

DROP TABLE IF EXISTS `clients`;

CREATE TABLE `clients` (

`id\_client` int(11) NOT NULL,

`fio\_client` varchar(30) DEFAULT NULL,

`tel\_client` char(12) DEFAULT NULL,

`adr\_client` varchar(30) DEFAULT NULL,

`ser\_pasp` char(4) DEFAULT NULL,

`numb\_pasp` char(6) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--

-- Дамп данных таблицы `clients`

--

INSERT INTO `clients` (`id\_client`, `fio\_client`, `tel\_client`, `adr\_client`, `ser\_pasp`, `numb\_pasp`) VALUES

(1, 'Малышев Святослав Александрови', '9209773399', 'г. Рязань', '0000', '999999'),

(2, 'Кальщиков Никита Сергеевич', '9209773399', 'г. Рязань', '1111', '999999'),

(3, 'Батраков Борис Борисович', '9209773399', 'г. Рязань', '2222', '999999'),

(4, 'Акшонова Мария Александровна', '9209773399', 'г. Рязань', '3333', '999999'),

(5, 'Бирюков Павел Юрьевич', '9209773399', 'г. Рязань', '4444', '999999');

-- --------------------------------------------------------

--

-- Структура таблицы `countries`

--

-- Создание: Май 21 2017 г., 17:58

-- Последнее обновление: Май 21 2017 г., 18:06

--

DROP TABLE IF EXISTS `countries`;

CREATE TABLE `countries` (

`id\_country` int(11) NOT NULL,

`n\_country` varchar(30) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--

-- Дамп данных таблицы `countries`

--

INSERT INTO `countries` (`id\_country`, `n\_country`) VALUES

(1, 'США'),

(2, 'Россия'),

(3, 'Германия');

-- --------------------------------------------------------

--

-- Структура таблицы `directors`

--

-- Создание: Май 21 2017 г., 17:58

-- Последнее обновление: Май 21 2017 г., 18:03

--

DROP TABLE IF EXISTS `directors`;

CREATE TABLE `directors` (

`id\_director` int(11) NOT NULL,

`name\_director` varchar(30) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--

-- Дамп данных таблицы `directors`

--

INSERT INTO `directors` (`id\_director`, `name\_director`) VALUES

(1, 'Снайдер Зк'),

(2, 'Васильев Юрий'),

(3, 'Асадулин Олег'),

(4, 'Ферхоэвен Симон'),

(5, 'Эгоян Атом'),

(6, 'Найшуллер Илья'),

(7, 'Руссо Энтони'),

(8, 'Сингер Брайан');

-- --------------------------------------------------------

--

-- Структура таблицы `employee`

--

-- Создание: Май 21 2017 г., 17:58

-- Последнее обновление: Май 21 2017 г., 18:58

--

DROP TABLE IF EXISTS `employee`;

CREATE TABLE `employee` (

`id\_emp` int(11) NOT NULL,

`fio\_emp` varchar(30) DEFAULT NULL,

`tel\_emp` char(12) DEFAULT NULL,

`adr\_emp` varchar(30) DEFAULT NULL,

`login\_emp` varchar(30) NOT NULL,

`password\_emp` varchar(50) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--

-- Дамп данных таблицы `employee`

--

INSERT INTO `employee` (`id\_emp`, `fio\_emp`, `tel\_emp`, `adr\_emp`, `login\_emp`, `password\_emp`) VALUES

(1, 'Бирюков Николай Юрьевчи', '9209773399', 'г. Рязань, ул. Зубковой, д. 20', 'nikolay', '4041'),

(2, 'Администратор', '9209773399', 'г. Рязань, ул. Зубковой, д. 20', 'admin', '4041');

-- --------------------------------------------------------

--

-- Структура таблицы `formats`

--

-- Создание: Май 21 2017 г., 17:58

-- Последнее обновление: Май 21 2017 г., 18:08

--

DROP TABLE IF EXISTS `formats`;

CREATE TABLE `formats` (

`id\_format` int(11) NOT NULL,

`n\_format` varchar(20) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--

-- Дамп данных таблицы `formats`

--

INSERT INTO `formats` (`id\_format`, `n\_format`) VALUES

(1, 'Full HD '),

(2, '4K'),

(3, '5K'),

(4, '8K');

-- --------------------------------------------------------

--

-- Структура таблицы `genres`

--

-- Создание: Май 21 2017 г., 17:58

-- Последнее обновление: Май 21 2017 г., 18:06

--

DROP TABLE IF EXISTS `genres`;

CREATE TABLE `genres` (

`id\_genre` int(11) NOT NULL,

`n\_genre` varchar(45) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--

-- Дамп данных таблицы `genres`

--

INSERT INTO `genres` (`id\_genre`, `n\_genre`) VALUES

(1, 'Фантастика'),

(2, 'Драма'),

(3, 'Ужасы'),

(4, 'Триллер'),

(5, 'Боевик');

-- --------------------------------------------------------

--

-- Структура таблицы `genresvideo`

--

-- Создание: Май 21 2017 г., 00:36

-- Последнее обновление: Май 21 2017 г., 18:55

--

DROP TABLE IF EXISTS `genresvideo`;

CREATE TABLE `genresvideo` (

`id\_genre` int(11) NOT NULL,

`id\_video` int(11) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--

-- Дамп данных таблицы `genresvideo`

--

INSERT INTO `genresvideo` (`id\_genre`, `id\_video`) VALUES

(1, 1),

(2, 2),

(4, 3),

(3, 4),

(4, 4),

(2, 5),

(4, 5),

(1, 6),

(1, 7),

(2, 7),

(4, 7),

(1, 8),

(5, 8);

-- --------------------------------------------------------

--

-- Структура таблицы `hire`

--

-- Создание: Май 21 2017 г., 21:09

-- Последнее обновление: Май 21 2017 г., 21:09

--

DROP TABLE IF EXISTS `hire`;

CREATE TABLE `hire` (

`id\_hire` int(11) NOT NULL,

`date\_hire` date DEFAULT NULL,

`id\_emp` int(11) NOT NULL,

`id\_client` int(11) NOT NULL,

`days` smallint(6) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--

-- Дамп данных таблицы `hire`

--

INSERT INTO `hire` (`id\_hire`, `date\_hire`, `id\_emp`, `id\_client`, `days`) VALUES

(1, '2017-05-21', 1, 1, 10),

(2, '2017-05-21', 1, 4, 20),

(3, '2017-05-21', 1, 4, 32),

(4, '2017-05-22', 1, 5, 23),

(5, '2017-05-25', 2, 3, 2);

--

-- Триггеры `hire`

--

DROP TRIGGER IF EXISTS `historyRemoveHire`;

DELIMITER $$

CREATE TRIGGER `historyRemoveHire` BEFORE DELETE ON `hire` FOR EACH ROW INSERT INTO historyhire (

date\_hire, id\_emp, id\_client, days

) VALUES (

OLD.date\_hire, OLD.id\_emp, OLD.id\_client, OLD.days

)

$$

DELIMITER ;

-- --------------------------------------------------------

--

-- Структура таблицы `hirevideo`

--

-- Создание: Май 21 2017 г., 09:38

-- Последнее обновление: Май 21 2017 г., 21:09

--

DROP TABLE IF EXISTS `hirevideo`;

CREATE TABLE `hirevideo` (

`id\_hire` int(11) NOT NULL,

`id\_video` int(11) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--

-- Дамп данных таблицы `hirevideo`

--

INSERT INTO `hirevideo` (`id\_hire`, `id\_video`) VALUES

(1, 1),

(2, 2),

(4, 2),

(5, 2),

(2, 3),

(4, 3),

(3, 4),

(4, 4),

(5, 6);

-- --------------------------------------------------------

--

-- Структура таблицы `historyhire`

--

-- Создание: Май 21 2017 г., 21:08

-- Последнее обновление: Май 21 2017 г., 21:09

--

DROP TABLE IF EXISTS `historyhire`;

CREATE TABLE `historyhire` (

`id\_hire` int(11) NOT NULL,

`date\_hire` date DEFAULT NULL,

`id\_emp` int(11) NOT NULL,

`id\_client` int(11) NOT NULL,

`days` smallint(6) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--

-- Дамп данных таблицы `historyhire`

--

INSERT INTO `historyhire` (`id\_hire`, `date\_hire`, `id\_emp`, `id\_client`, `days`) VALUES

(1, '2017-05-25', 1, 5, 7);

-- --------------------------------------------------------

--

-- Структура таблицы `producers`

--

-- Создание: Май 21 2017 г., 17:59

-- Последнее обновление: Май 21 2017 г., 18:16

--

DROP TABLE IF EXISTS `producers`;

CREATE TABLE `producers` (

`id\_prod` int(11) NOT NULL,

`n\_prod` varchar(30) DEFAULT NULL,

`id\_country` int(11) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--

-- Дамп данных таблицы `producers`

--

INSERT INTO `producers` (`id\_prod`, `n\_prod`, `id\_country`) VALUES

(1, 'Чарльз Ровен', 1),

(2, 'Эльмира Айнулова', 2),

(3, 'Мэттью Баллен', 3),

(4, 'Георгий Малков', 2),

(5, 'Эри Лантос', 3),

(6, 'Тимур Бекмамбетов', 2),

(7, 'Кевин Файги', 1),

(8, 'Лорен Шулер Доннер', 1);

-- --------------------------------------------------------

--

-- Структура таблицы `video`

--

-- Создание: Май 21 2017 г., 17:59

-- Последнее обновление: Май 21 2017 г., 18:55

--

DROP TABLE IF EXISTS `video`;

CREATE TABLE `video` (

`id\_video` int(11) NOT NULL,

`n\_video` varchar(60) DEFAULT NULL,

`duration` smallint(6) DEFAULT NULL,

`descript` varchar(255) DEFAULT NULL,

`price` decimal(15,2) DEFAULT NULL,

`id\_prod` int(11) NOT NULL,

`year\_estab` char(4) DEFAULT NULL,

`id\_format` int(11) NOT NULL,

`id\_director` int(11) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--

-- Дамп данных таблицы `video`

--

INSERT INTO `video` (`id\_video`, `n\_video`, `duration`, `descript`, `price`, `id\_prod`, `year\_estab`, `id\_format`, `id\_director`) VALUES

(1, 'Бэтмен против Супермена', 153, 'Описание', '300.00', 1, '2016', 4, 1),

(2, 'Герой', 86, 'Описание', '150.00', 2, '2016', 3, 2),

(3, 'Запрос в друзья', 92, 'Здесь описание', '150.00', 3, '2016', 3, 4),

(4, 'Маршрут построен', 85, 'Описание', '200.00', 4, '2016', 4, 3),

(5, 'Помнить', 94, 'Описание', '200.00', 5, '2016', 2, 5),

(6, 'Хардкор', 96, 'Здесь описание', '250.00', 6, '2016', 4, 6),

(7, 'Первый мститель Противостояние', 146, 'Здесь описание', '300.00', 7, '2016', 4, 7),

(8, 'Люди Икс Апокалипсис', 136, 'Здесь описание', '300.00', 8, '2016', 4, 8);

--

-- Индексы сохранённых таблиц

--

--

-- Индексы таблицы `clients`

--

ALTER TABLE `clients`

ADD PRIMARY KEY (`id\_client`);

--

-- Индексы таблицы `countries`

--

ALTER TABLE `countries`

ADD PRIMARY KEY (`id\_country`);

--

-- Индексы таблицы `directors`

--

ALTER TABLE `directors`

ADD PRIMARY KEY (`id\_director`);

--

-- Индексы таблицы `employee`

--

ALTER TABLE `employee`

ADD PRIMARY KEY (`id\_emp`),

ADD UNIQUE KEY `login\_emp` (`login\_emp`);

--

-- Индексы таблицы `formats`

--

ALTER TABLE `formats`

ADD PRIMARY KEY (`id\_format`);

--

-- Индексы таблицы `genres`

--

ALTER TABLE `genres`

ADD PRIMARY KEY (`id\_genre`);

--

-- Индексы таблицы `genresvideo`

--

ALTER TABLE `genresvideo`

ADD PRIMARY KEY (`id\_genre`,`id\_video`),

ADD KEY `R\_12` (`id\_video`);

--

-- Индексы таблицы `hire`

--

ALTER TABLE `hire`

ADD PRIMARY KEY (`id\_hire`),

ADD KEY `R\_5` (`id\_emp`),

ADD KEY `R\_6` (`id\_client`);

--

-- Индексы таблицы `hirevideo`

--

ALTER TABLE `hirevideo`

ADD PRIMARY KEY (`id\_hire`,`id\_video`),

ADD KEY `R\_9` (`id\_video`);

--

-- Индексы таблицы `historyhire`

--

ALTER TABLE `historyhire`

ADD PRIMARY KEY (`id\_hire`),

ADD KEY `R\_7` (`id\_emp`) USING BTREE,

ADD KEY `R\_8` (`id\_client`) USING BTREE;

--

-- Индексы таблицы `producers`

--

ALTER TABLE `producers`

ADD PRIMARY KEY (`id\_prod`),

ADD KEY `R\_1` (`id\_country`);

--

-- Индексы таблицы `video`

--

ALTER TABLE `video`

ADD PRIMARY KEY (`id\_video`),

ADD KEY `R\_2` (`id\_prod`),

ADD KEY `R\_4` (`id\_format`),

ADD KEY `R\_10` (`id\_director`);

--

-- AUTO\_INCREMENT для сохранённых таблиц

--

--

-- AUTO\_INCREMENT для таблицы `clients`

--

ALTER TABLE `clients`

MODIFY `id\_client` int(11) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=6;

--

-- AUTO\_INCREMENT для таблицы `countries`

--

ALTER TABLE `countries`

MODIFY `id\_country` int(11) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=4;

--

-- AUTO\_INCREMENT для таблицы `directors`

--

ALTER TABLE `directors`

MODIFY `id\_director` int(11) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=9;

--

-- AUTO\_INCREMENT для таблицы `employee`

--

ALTER TABLE `employee`

MODIFY `id\_emp` int(11) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=3;

--

-- AUTO\_INCREMENT для таблицы `formats`

--

ALTER TABLE `formats`

MODIFY `id\_format` int(11) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=5;

--

-- AUTO\_INCREMENT для таблицы `genres`

--

ALTER TABLE `genres`

MODIFY `id\_genre` int(11) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=6;

--

-- AUTO\_INCREMENT для таблицы `hire`

--

ALTER TABLE `hire`

MODIFY `id\_hire` int(11) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=7;

--

-- AUTO\_INCREMENT для таблицы `historyhire`

--

ALTER TABLE `historyhire`

MODIFY `id\_hire` int(11) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=2;

--

-- AUTO\_INCREMENT для таблицы `producers`

--

ALTER TABLE `producers`

MODIFY `id\_prod` int(11) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=9;

--

-- AUTO\_INCREMENT для таблицы `video`

--

ALTER TABLE `video`

MODIFY `id\_video` int(11) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=9;

--

-- Ограничения внешнего ключа сохраненных таблиц

--

--

-- Ограничения внешнего ключа таблицы `genresvideo`

--

ALTER TABLE `genresvideo`

ADD CONSTRAINT `genresvideo\_ibfk\_1` FOREIGN KEY (`id\_genre`) REFERENCES `genres` (`id\_genre`),

ADD CONSTRAINT `genresvideo\_ibfk\_2` FOREIGN KEY (`id\_video`) REFERENCES `video` (`id\_video`);

--

-- Ограничения внешнего ключа таблицы `hire`

--

ALTER TABLE `hire`

ADD CONSTRAINT `hire\_ibfk\_1` FOREIGN KEY (`id\_emp`) REFERENCES `employee` (`id\_emp`),

ADD CONSTRAINT `hire\_ibfk\_2` FOREIGN KEY (`id\_client`) REFERENCES `clients` (`id\_client`);

--

-- Ограничения внешнего ключа таблицы `hirevideo`

--

ALTER TABLE `hirevideo`

ADD CONSTRAINT `hirevideo\_ibfk\_1` FOREIGN KEY (`id\_hire`) REFERENCES `hire` (`id\_hire`) ON DELETE CASCADE,

ADD CONSTRAINT `hirevideo\_ibfk\_2` FOREIGN KEY (`id\_video`) REFERENCES `video` (`id\_video`);

--

-- Ограничения внешнего ключа таблицы `historyhire`

--

ALTER TABLE `historyhire`

ADD CONSTRAINT `historyhire\_ibfk\_1` FOREIGN KEY (`id\_emp`) REFERENCES `employee` (`id\_emp`),

ADD CONSTRAINT `historyhire\_ibfk\_2` FOREIGN KEY (`id\_client`) REFERENCES `clients` (`id\_client`);

--

-- Ограничения внешнего ключа таблицы `producers`

--

ALTER TABLE `producers`

ADD CONSTRAINT `producers\_ibfk\_1` FOREIGN KEY (`id\_country`) REFERENCES `countries` (`id\_country`);

--

-- Ограничения внешнего ключа таблицы `video`

--

ALTER TABLE `video`

ADD CONSTRAINT `video\_ibfk\_1` FOREIGN KEY (`id\_prod`) REFERENCES `producers` (`id\_prod`),

ADD CONSTRAINT `video\_ibfk\_2` FOREIGN KEY (`id\_format`) REFERENCES `formats` (`id\_format`),

ADD CONSTRAINT `video\_ibfk\_3` FOREIGN KEY (`id\_director`) REFERENCES `directors` (`id\_director`);

/\*!40101 SET CHARACTER\_SET\_CLIENT=@OLD\_CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET CHARACTER\_SET\_RESULTS=@OLD\_CHARACTER\_SET\_RESULTS \*/;

/\*!40101 SET COLLATION\_CONNECTION=@OLD\_COLLATION\_CONNECTION \*/;

Код самого приложения со всеми плагинами будет занимать порядка 300 страниц, считаю не целесообразным его прикладывать сюда.

Несмотря на это к курсовому проекту, будет приложен оптический накопитель информации в виде пластикового диска, где и можно будет увидеть весь проект целиком.