

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «САМАРСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С.П. КОРОЛЕВА
(САМАРСКИЙ УНИВЕРСИТЕТ)»

Факультет информатики

Кафедра информационных систем и технологий

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту по дисциплине

«Разработка Web-приложений»

Выполнил: студент гр. 6402Б

Лыткин Д.П.

Руководитель проекта:

Лёзин И.А.

Дата сдачи:

Оценка:

Самара 2017

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

1 СТРУКТУРА БАЗЫ ДАННЫХ

1.1 Логическая схема БД

1.2 Физическая схема БД

2 АРХИТЕКТУРА ПРИЛОЖЕНИЯ

2.1 Технологии

2.2 Обоснования использования

2.3 Взаимодействие элементов системы

3 ИНТЕРФЕЙС ПОЛЬЗОВАТЕЛЯ

3.1 Разработка и описание интерфейса

3.2 Диаграмма вариантов использования

ЗАКЛЮЧЕНИЕ

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

ПРИЛОЖЕНИЕ А Исходный код программы

ПРИЛОЖЕНИЕ Б HTML, CSS, XML

ВВЕДЕНИЕ

Информационная система (ИС) – это система обработки данных какой-либо предметной области со средствами накопления, хранения, обработки, преобразования, передачи, обновления информации с использованием компьютерной и другой техники[1]. В ИС выполняются следующие процессы:

- – ввод информации из внешних и внутренних источников;
- – обработка входящей информации;
- – хранение информации для последующего ее использования;
- – вывод информации в удобном для пользователя виде [2].

Основа ИС, объект ее обработки - база данных (БД). База данных – это совокупность сведений о конкретных объектах реального мира в какой-либо предметной области или разделе предметной области[3]. Таким образом, БД выполняет функцию хранения информации в ИС.

В данной информационной системе «Фильмы» осуществляется работа с информацией о фильмах, их жанрах, странах и людях, принявших участие в создании фильма. Поэтому в БД должна содержаться следующая информация:

- – информация о фильмах;
- – информация о жанрах;
- – информация о странах;
- – информация о людях;

Для работы необходим удобный пользовательский интерфейс, который обеспечивает представление, добавление и редактирование данных.

1. СТРУКТУРА БАЗЫ ДАННЫХ

1.1 Логическая схема БД

Логическая модель БД описывает понятия предметной области, их взаимосвязь, а также ограничения на данные, налагаемые предметной областью без учета её реализации в конкретной СУБД [4].

Основным средством разработки логической модели данных в настоящий момент являются различные варианты ER-диаграмм (Entity-Relationship, диаграммы сущность-связь).

На логической схеме изображены следующие сущности: Film, Genre, Country, People. Описание сущностей содержится в таблице 1.

Таблица 1 – Описание сущностей логической модели данных

Название	Назначение
Film	Описывает фильмы. Содержит атрибуты: идентификатор, название, рейтинг, описание, ссылку на трейлер, дату выхода. Связана связью «многие ко многим» с сущностями «People», «Country», «Genre».
People	Описывает людей участников. Содержит атрибуты: идентификатор имя, фамилия, должность. Связана связью «многие ко многим» с сущностью «Film».
Country	Описывает страны. Содержит атрибуты: идентификатор, название страны. Связана связью «многие ко многим» с сущностью «Film».
Genre	Описывает жанры. Содержит атрибуты: идентификатор, название жанра. Связана связью «многие ко многим» с сущностью «Film».

1.2 Физическая схема БД

Физическая модель данных оперирует категориями, касающимися организации внешней памяти и структур хранения, используемых в данной операционной среде [5]. В настоящий момент в качестве физических моделей используются различные методы размещения данных, основанные на файловых структурах: это организация файлов прямого и последовательного доступа, индексных файлов и инвертированных списков.

На рисунке 1 представлена физическая схема БД. На схеме изображены сущности, а также указаны типы данных.

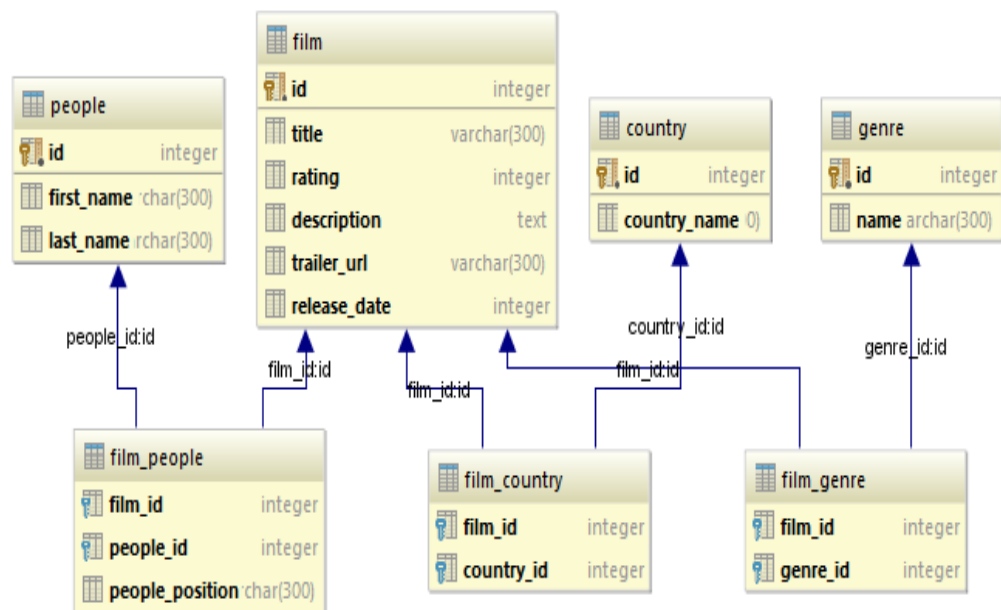


Рисунок 1– Физическая схема БД

2. АРХИТЕКТУРА ПРИЛОЖЕНИЯ

2.1 Технологии

При разработке информационной системы использовались следующие технологии Java EE: JSP (Java Server Pages) и Servlet, а также JDBC для связи с БД и Javascript, jQuery, Twitter Bootstrap 3 для динамического отображения данных и элементов страниц. В качестве локального сервера приложений использовался Tomcat 8.0.21. Для создания информационной системы была использована IDE (Integrated development environment) IntelliJ IDEA 16.0.2.

Технология JavaServer Pages (JSP) позволяет легко создавать web-содержимое, которое имеет как статические, так и динамические компоненты[6]. Она воплощает все динамические возможности технологии Java Servlet, но обеспечивает более естественный способ создания статического содержимого. Страница JSP является текстовым документом, которая содержит текст двух типов: статические исходные данные, которые могут быть оформлены в одном из текстовых форматов HTML, SVG, WML, или XML, и JSP-элементы, которые конструируют динамическое содержимое.

Сервлеты (Servlets) – это java-программы, которые выполняются на серверной стороне Web-приложения и динамически расширяют функциональные возможности Web-сервера [7].

Работа сервлета заключается в том, что при приходе запроса от клиента Web-сервер с помощью специального конфигурационного файла определяет, какой сервлет необходимо выполнить. JAVA-машина, в свою очередь, выполняет сервлет, который создает HTML-страницу и передает содержимое Web-серверу. Web-сервер отправляет клиенту сформированную сервлетом HTML-страницу.

Для отображения объектно-ориентированной модели данных в

традиционные реляционные базы данных использовалась технология JDBC[8]. Для создания статической составляющей веб-страниц применялся язык гипертекстовой разметки HTML [9], для стилизации страниц были использованы каскадные таблицы стилей CSS (Cascading Style Sheets) [10] и библиотека Twitter Bootstrap 3. Для создания динамической составляющей страниц использовался язык JavaScript и библиотека jQuery.

2.2 Обоснования использования

Технология JSP представляет собой простой способ добавлять динамические данные к фиксированным шаблонным данным путем Java вставок, заключенных в специальные теги, что значительно упрощает разработку web-приложения. Технология является независимой от платформы на динамических web-страницах и web-серверах, включает многократное использование компонентов (JavaBeans) и библиотек тегов. Использование заранее определенных переменных session, response, и request позволяет разработчику эффективно обрабатывать запросы клиента и создавать динамические ответы на них.

Сервлеты обеспечивают компонентный, платформенно независимый способ создания веб-приложений, без ограничений производительности. Они не привязаны ни к конкретному серверу, ни к платформе. Сервлеты не предназначены для какого-то конкретного протокола, хотя чаще всего их используют совместно с HTTP (HyperText Transfer Protocol). Сервлет дает возможность передавать пользовательские данные заданному ресурсу и сохранять полученные с ресурса данные в БД.

Технология JDBC позволяет выполнить отображение объектов объектно-ориентированного языка в структуры реляционных баз данных со всеми полями, значениями, отношениями и так далее.

HTML, CSS позволяют создать интерактивные страницы с приятным дизайном.

2.3 Взаимодействие элементов системы

Реализованная информационная система имеет трехуровневую архитектуру, т.е. разделена на три слоя: слой представления (клиентское приложение), слой бизнес-логики (сервер приложений), слой доступа к данным (сервер БД). Слой представления – интерфейсный компонент системы, предоставляемый конечному пользователю. Данный слой выполняет функции отображения и ввода данных для последующей обработки, и не нагружается существенной бизнес-логикой. Она в основном сосредоточена на сервере приложений, который располагается на втором уровне архитектуры. Связующее программное обеспечение позволяет принимать запросы от клиентского приложения, и после их анализа выполнять запросы к серверу БД. Слой доступа к данным обеспечивает хранение данных и выносятся на отдельный уровень, реализуется, как правило, средствами систем управления базами данных, подключение к этому компоненту обеспечивается только с уровня сервера приложений.

3. ИНТЕРФЕЙС ПОЛЬЗОВАТЕЛЯ

3.1 Разработка и описание интерфейса

При разработке интерфейса необходимо следовать следующим принципам: интерфейс должен помогать выполнять задачу, а не становиться ею. Он должен быть дружелюбным, удобным, интуитивно-понятным, чтобы минимизировать время, которое пользователь тратит на осознание того, как работать с системой.

На начальной странице пользователю представлен список фильмов, существующих в системе (рисунок 2).

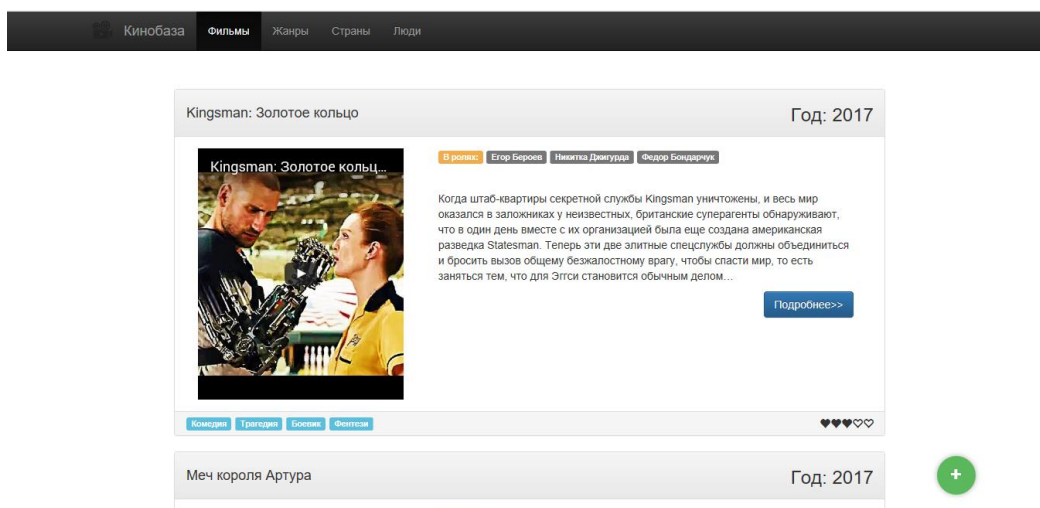


Рисунок 2 – Главная страница

При нажатии на кнопке «Подробнее» осуществляется переход на страницу представления информации о выбранном фильме (рисунок 3).

При нажатии на кнопку «+» осуществляется переход на страницу добавления фильма (рисунок 4).

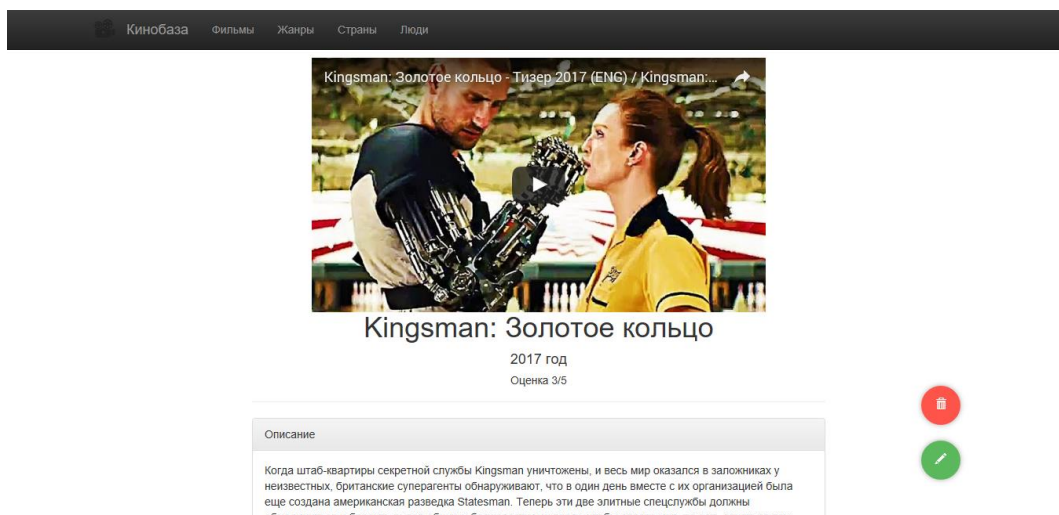


Рисунок 3 – Страница информации о выбранном фильме

На странице фильма (рисунок 3) отображается информация о выбранном фильме. При нажатии на кнопку «Удалить» осуществляется удаление выбранного фильма из БД, затем происходит переход на главную страницу. При нажатии на кнопке «Изменить» осуществляется переход к странице редактирования фильма (рисунок 5).

Рисунок 4 – страница добавления фильма.

На странице добавления фильма (рисунок 4) осуществляется ввод информации о фильме, выбор стран, жанров, а так же участников (рисунок 4.1). При нажатии на кнопку «Добавить» осуществляется добавление фильма в БД и переход на страницу добавленного фильма. При нажатии на кнопку отмена осуществляется переход на главную страницу.

Актёры

#	Имя	Фамилия
<input type="checkbox"/>	Сергей	Безруков
<input type="checkbox"/>	Егор	Бероев
<input checked="" type="checkbox"/>	Екатерина	Гусева
<input type="checkbox"/>	Федор	Бондарчук
<input type="checkbox"/>	Скарлет	Йоханссон
<input checked="" type="checkbox"/>	Джордж	Клуни
<input checked="" type="checkbox"/>	Никитка	Джигурда
<input type="checkbox"/>	Здравствуйте	Досвидания

Заккрыть

Добавить

Рисунок 4.1 – Добавление участников

На рисунке 4.1 изображено диалоговое окно добавления актеров, открывающееся при нажатии на кнопку «Добавить актеров» она странице добавления фильма (рисунок 4). Для добавления необходимо выбрать актеров и нажать на кнопку «Добавить». При нажатии на кнопку «Заккрыть» осуществляется закрытие диалогового окна и отмена введенных данных. Добавление режиссеров, стран и жанров выполняется аналогично.

КиноБаза

Фильмы

Актеры

Страны

Жанры

Название фильма

Кингсман: Золотые часы

Трейлер

https://www.youtube.com/embed/YJCuBkddm

Год выпуска

2017

Рейтинг

3

Описание

Когда штаб-квартиры секретной службы Кингсман уничтожены, и весь мир оказывается в заложниках у неизвестных. Британские спецслужбы обнаруживают, что в один день вместе с их организацией были еще созданы американская разведка Shadowland. Теперь эти две элитные спецслужбы должны объединиться и бросить вызов общему беззастенчивому врагу, чтобы спасти мир, то есть заняться тем, что для этих служб является обычным делом.

Добавить актеров

Добавить режиссера

Добавить страну

Добавить жанры

Изменить режиссера

Изменить страну

Изменить жанры

Изменить страны

Изменить жанры

Сохранить

Отмена

Рисунок 5 – страница редактирования фильма

На экране информации о жанрах (рисунок 6) отображается таблица, содержащая информацию о доступных в системе жанрах и функциональные кнопки «Удалить» и «Редактировать» для каждого жанра. При нажатии на кнопку «Удалить» осуществляется удаление выбранного

жанра из БД и обновление списка жанров на странице. При нажатии на кнопку «Редактировать» открывается диалоговое окно редактирования жанра (рисунок 5.1). При нажатии на кнопку добавить, открывается диалоговое окно добавления жанра (рисунок 5.2)

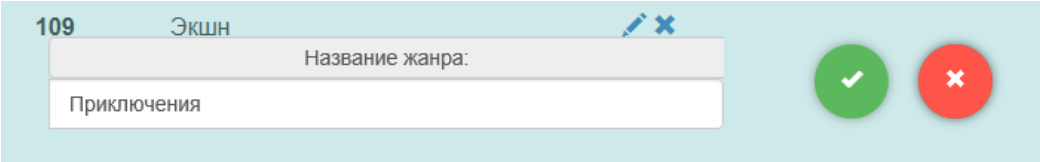


Рисунок 5.1 – Диалоговое окно редактирования жанра

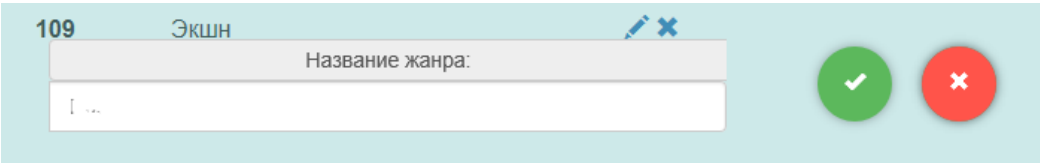


















Рисунок 5.2 – Диалоговое окно добавления жанра

Просмотр данных о странах (рисунок 6) и людях (рисунок 7) производится аналогично.

Кинобаза			
Фильмы			
Жанры			
Страны			
Люди			
#	Страна	Опции	
34	Индия	✎ ✕	
35	Пакистан	✎ ✕	
52	Россия	✎ ✕	
53	Англия	✎ ✕	
54	Франция	✎ ✕	
55	Казахстан	✎ ✕	
56	Ливия	✎ ✕	
60	Швеция	✎ ✕	
62	Латвия	✎ ✕	
63	Литва	✎ ✕	
65	Австралия	✎ ✕	

Рисунок 6 – страница информации о странах

<div> <div>Кинобаза</div> <div>Фильмы</div> <div>Жанры</div> <div>Страны</div> <div>Люди</div> </div>			
#	Имя	Фамилия	Опции
66	Сергей	Безруков	 
67	Егор	Берев	 
68	Екатерина	Гусева	 
70	Федор	Бондарчук	 
71	Скарлет	Йоханссон	 
72	Джордж	Клуни	 
69	Никитка	Джигурда	 
104	Здравствуйте	Досвидания	 




Рисунок 7 – Страница информации о людях

Панель навигации (рисунок 8) присутствует на всех экранах системы.

При нажатии на кнопку на панели навигации осуществляется переход к соответствующей информационной странице.



Рисунок 8 – Панель навигации

На рисунке 9 представлена диаграмма вариантов использования [9]. На этой диаграмме определяются возможности пользователя в ИС.

В соответствии с ней пользователь может выбрать сущность для просмотра данных. При просмотре данных пользователь имеет возможность добавлять, редактировать и удалять записи в БД, выполнять дополнительные выборки. Добавление данных подразумевает непременно заполнение обязательных полей, и заполнение необязательных полей по желанию, после чего пользователь может сохранить данные. Для редактирования пользователь должен выбрать запись из таблицы. После чего он имеет возможность редактировать некоторые параметры, однако, как и в случае добавления, пользователь

обязан оставить заполненными обязательные поля. Для удаления пользователь должен выбрать запись из таблицы.

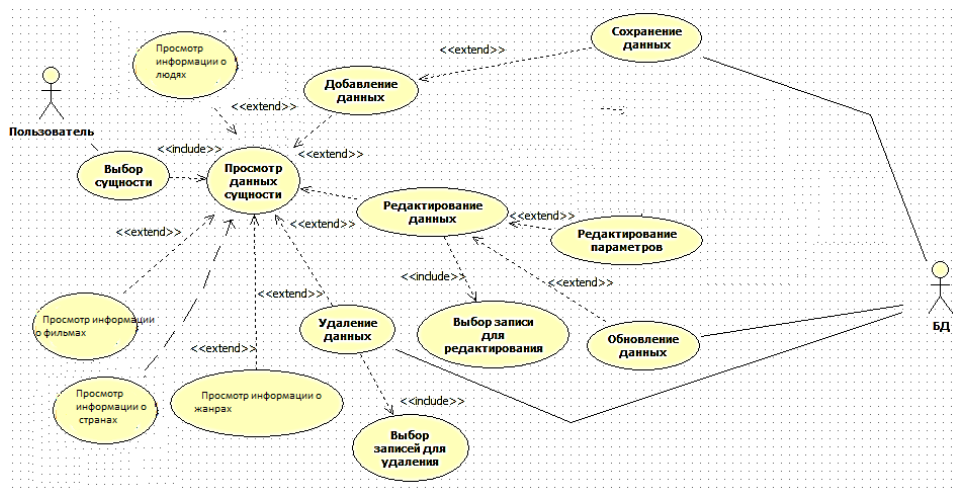


Рисунок 9 –Диаграмма вариантов использования

ЗАКЛЮЧЕНИЕ

В соответствии с заданием для курсового проекта был проведен анализ предметной области, логическое проектирование и затем выполнена физическая реализация проекта в выбранной разработчиком среде. При разработке использовались следующие технологии: Servlet, JSP, JDBC, jQuery каскадные таблицы стилей – CSS. Итогом проделанной работы является информационная система «Фильмы».

В разработанной системе реализован веб-интерфейс, который обеспечивает работу с сервер-приложением, в свою очередь, взаимодействующим с БД. Пользователю предоставлена возможность выполнения основных операций: чтения, добавления, редактирования, удаления данных.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- 1 Вишнякова, С.М. Профессиональное образование. Словарь. Ключевые понятия, термины, актуальная лексика [Электронный ресурс]/ С.М.Вишнякова – М.: НМЦ СПО, 1999.
- 2 Понятие информационной системы (ИС) [Электронный ресурс]. – URL: <http://cde.osu.ru/demoversion/course157/text/1.5.html> (дата обращения: 07 04 2015).
- 3 Информационные системы. Базы данных[Электронный ресурс]. – URL: <http://www.kolomna-school7-ict.narod.ru/st40401.htm> (дата обращения: 07 04 2015).
- 4 Кривошеин, М. ER-диаграммы сущность-связь[Электронный ресурс] /М.Кривошеин – URL: <http://kitzip.ru/upload/iblock/770/770e94b98016f3f4917cb4e38633d0f1.pdf> (Дата обращения: 27 10 2014).
- 5 Классификация моделей данных[Электронный ресурс]. – URL:http://www.sqlshop.ru/publ/klassifikacija_modelej_dannykh/1-1-0-5 (дата обращения: 18 12 2014).
- 6 Технология JavaServer Pages[Электронный ресурс]. – URL:<http://www.javable.com/tutorials/j2ee/JSPIntro/> (дата обращения: 11 04 2015).
- 7 Сервлеты. Введение[Электронный ресурс]. – URL: <http://www.java2ee.ru/servlets/> (дата обращения: 04 11 2015).
- 8 Java Hibernate. Часть 1 – Введение[Электронный ресурс]. – URL:<http://javaxblog.ru/article/java-hibernate-1/>(дата обращения: 11 04 2015).
- 9 Самоучитель HTML4 [Электронный ресурс]. – URL: <http://htmlbook.ru/samhtml> (дата обращения: 11 04 2011).
- 10 Самоучитель CSS[Электронный ресурс]. – URL: <http://htmlbook.ru/samcss>(дата обращения: 11 04 2015).

ПРИЛОЖЕНИЕ А

Исходный код программы

```
package dao.generic;

/**
 * Created by Lytki on 26.03.2017.
 */
public class DAOConstants {
    //db connection
    public static final String
    DB_URL="jdbc:postgresql://localhost:5432/films";
    public static final String DB_USER="postgres";
    public static final String DB_PASS="";
    //db common
    public static final String ID="id";
    public static final String FILM_ID = "film_id";
    //film
    public static final String TABLE_FILM="film";
    public static final String FILM_TITLE="title";
    public static final String FILM_RELEASE_DATE="release_date";
    public static final String FILM_RAITING="rating";
    public static final String FILM_POSTER="poster";
    public static final String FILM_DESCRIPTION="description";
    public static final String FILM_TRAILER_URL="trailer_url";
    //country
    public static final String TABLE_COUNTRY="country";
    public static final String COUNTRY_NAME="country_name";
    //genre
    public static final String TABLE_GENRE="genre";
    public static final String GENRE_NAME="name";
    //pepole
    public static final String TABLE_PEOPLE="people";
    public static final String PEOPLE_FIRSTNAME="first_name";
    public static final String PEOPLE_LASTNAME="last_name";
    //film country
    public static final String TABLE_FILM_COUNTRY="film_country";
    public static final String COUNTRY_ID = "country_id";
    //film_genre
    public static final String TABLE_FILM_GENRE ="film_genre";
    public static final String GENRE_ID="genre_id";
    //film_people
    public static final String TABLE_FILM_PEOLE="film_people";
    public static final String PEOPLE_ID="people_id";
    public static final String PEOPLE_POSITION="people_position";

}

package entity;

/**
 * Created by Lytki on 26.03.2017.
 */
public class People extends Entity{
    private String firstName;
    private String lastName;

    public String getFirstName() {
        return firstName;
    }
}
```

```

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    public String getLastName() {
        return lastName;
    }

    public void setLastName(String lastName) {
        this.lastName = lastName;
    }

    @Override
    public String toString() {
        return "People{" +
            "id='" + getId() + '\'' +
            "firstName='" + firstName + '\'' +
            ", lastName='" + lastName + '\'' +
            '}';
    }
}

```

```
package dao;
```

```

import dao.generic.DAOConstants;
import dao.generic.IGenericDAO;
import entity.Country;
import entity.Entity;
import entity.People;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;

```

```

/**
 * Created by Lytki on 26.03.2017.
 */

```

```

public class PeopleDAO implements IGenericDAO {
    @Override
    public Entity create(Entity entity) {
        Connection connection = null;
        PreparedStatement preparedStatement = null;
        try {
            People people = (People) entity;
            connection = this.getConnection();
            StringBuilder statementBuilder = new StringBuilder();
            statementBuilder.append("INSERT INTO ")
                .append(DAOConstants.TABLE_PEOPLE)
                .append(" (")
                .append(DAOConstants.PEOPLE_FIRSTNAME)
                .append(", ")
                .append(DAOConstants.PEOPLE_LASTNAME)
                .append(") ")
                .append("VALUES")
                .append("(?, ?)");
            preparedStatement =
connection.prepareStatement(statementBuilder.toString());
            preparedStatement.setString(1, people.getFirstName());
            preparedStatement.setString(2, people.getLastName());

```

```

        preparedStatement.execute();
    } catch (SQLException e) {
        e.printStackTrace();
    }
    finally {
        try {
            preparedStatement.close();
            connection.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
    return null;
}

@Override
public Entity read(int entityId) {
    Connection connection = null;
    PreparedStatement preparedStatement = null;
    People people = null;
    try {
        connection = this.getConnection();
        StringBuilder statementBuilder = new StringBuilder();
        statementBuilder.append("SELECT * FROM ")
            .append(DAOConstants.TABLE_PEOPLE)
            .append(" WHERE ")
            .append(DAOConstants.ID)
            .append("=")
            .append("?");
        preparedStatement =
connection.prepareStatement(statementBuilder.toString());
        preparedStatement.setInt(1, entityId);
        ResultSet resultSet = preparedStatement.executeQuery();
        if(resultSet.next()){
            people = new People();
            people.setId(resultSet.getInt(1));
            people.setFirstName(resultSet.getString(2));
            people.setLastName(resultSet.getString(3));
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
    finally {
        try {
            preparedStatement.close();
            connection.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
    return people;
}

@Override
public void update(Entity entity) {
    Connection connection = null;
    PreparedStatement preparedStatement = null;
    People people = (People) entity;
    try {
        connection = this.getConnection();
        StringBuilder statementBuilder = new StringBuilder();
        statementBuilder.append("UPDATE ")

```

```

        .append(DAOConstants.TABLE_PEOPLE)
        .append(" SET ")
        .append(DAOConstants.PEOPLE_FIRSTNAME)
        .append("=?, ")
        .append(DAOConstants.PEOPLE_LASTNAME)
        .append("=? ")
        .append(" WHERE ")
        .append(DAOConstants.ID)
        .append("=?");
    preparedStatement =
connection.prepareStatement(StatementBuilder.toString());
    preparedStatement.setString(1,people.getFirstName());
    preparedStatement.setString(2,people.getLastName());
    preparedStatement.setInt(3, people.getId());
    preparedStatement.executeUpdate();
} catch (SQLException e) {
    e.printStackTrace();
}
}
finally {
    try {
        preparedStatement.close();
        connection.close();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

}

@Override
public void delete(int entityId) {
    Connection connection = null;
    PreparedStatement preparedStatement =null;
    try {
        connection = this.getConnection();
        StringBuilder StatementBuilder = new StringBuilder();
        StatementBuilder.append("DELETE FROM ")
            .append(DAOConstants.TABLE_PEOPLE)
            .append(" WHERE ")
            .append(DAOConstants.ID)
            .append("=?");
        preparedStatement =
connection.prepareStatement(StatementBuilder.toString());
        preparedStatement.setInt(1,entityId);
        preparedStatement.execute();
    } catch (SQLException e) {
        e.printStackTrace();
    }
    finally {
        try {
            preparedStatement.close();
            connection.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}

}

public List<People> getAllPeoples(){
    Connection connection = null;
    PreparedStatement preparedStatement =null;
    People people = null;
    List<People> peoples=new ArrayList<>();

```

```

        try {
            connection = this.getConnection();
            StringBuilder StatementBuilder = new StringBuilder();
            StatementBuilder.append("SELECT * FROM ")
                .append(DAOConstants.TABLE_PEOPLE);
            PreparedStatement =
connection.prepareStatement(StatementBuilder.toString());
            ResultSet resultSet=preparedStatement.executeQuery();
            while(resultSet.next()){
                people = new People();
                people.setId(resultSet.getInt(1));
                people.setFirstName(resultSet.getString(2));
                people.setLastName(resultSet.getString(3));
                peoples.add(people);
            }
            return peoples;
        } catch (SQLException e) {
            e.printStackTrace();
        }
        finally {
            try {
                preparedStatement.close();
                connection.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
        return null;
    }
}

package servlets;

import com.google.gson.Gson;
import dao.PeopleDAO;
import entity.Country;
import entity.People;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

/**
 * Created by Lytki on 14.04.2017.
 */
@WebServlet(name="/peoples", urlPatterns="/peoples")
public class PeoplesServlet extends HttpServlet {
    private static PeopleDAO peopleDAO = new PeopleDAO();
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp)
        throws ServletException, IOException {
        //      super.doGet(req, resp);
        List<People> peoples = peopleDAO.getAllPeoples();
        Map<String,List<People>> c = new HashMap<>();
        c.put("peoples", peoples);
        Gson gson = new Gson();
        resp.setCharacterEncoding("UTF-8");

```

```

        resp.setContentType("application/json");
        PrintWriter out = resp.getWriter();
        String json = gson.toJson(c);
        out.print(json);
        out.flush();
        out.close();
    }

    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp)
    throws ServletException, IOException {
        String action=req.getParameter("action");
        if(action.equals("ADD")){
            String name = req.getParameter("firstName");
            String lastName = req.getParameter("lastName");
            People people = new People();
            people.setFirstName(name);
            people.setLastName(lastName);
            peopleDAO.create(people);

        }else if(action.equals("UPDATE")){
            Gson gson = new Gson();
            String id = req.getParameter("people[id]");
            String name=req.getParameter("people[peopleName]");
            String lastName=req.getParameter("people[peopleLastName]");
            People people = new People();
            people.setId(Integer.parseInt(id));
            people.setFirstName(name);
            people.setLastName(lastName);

            peopleDAO.update(people);
        }else if(action.equals("DELETE")){
            String id=req.getParameter("id");
            peopleDAO.delete(Integer.parseInt(id));
        }
    }
}

```

ПРИЛОЖЕНИЕ Б

HTML, CSS, XML, JavaScript

//CSS для веб страниц

```
.camera-logo{
    width: 35px;
    float: left;
    margin-top: 6px;
    margin-right: 10px;
}

.table-countries>thead>tr>th, .table-countries>tbody>tr>th, .table-
countries>tbody>tr>td,
.table-peoples>thead>tr>th, .table-peoples>tbody>tr>th, .table-
peoples>tbody>tr>td{
    text-align: center;
}

.action-button{
    width: 50px;
    height: 50px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.33);
    color: white;
    border-radius: 50%;
    position: absolute;
    text-align: center;
}

#add-dialog-button {
    bottom: 50px;
    right: 300px;
    background-color: #5cb85c;
}

#add-button {
    bottom: 0px;
    position: absolute;
    right: 350px;
    background-color: #5cb85c ;
    z-index: 100;
    margin-right: 20px;
}

#cancel-add-button {
    position: absolute;
    bottom: 0px;
    right: 300px;
    background-color: #ff544a;
    z-index: 100;
}

#cancel-add-button:hover{
    background-color: #ff7359;
}

#add-dialog-button:hover, #add-button:hover{
    background-color: #6ecc6e;
}

#country-name-input{
    z-index: 100;
    float: left;
    width: 455px;
    right: 481px;
    position: absolute;
```

```

        bottom: -22px;
    }
    .action-button>i{
        margin-top: 15px;
    }

    .opacifier{
        z-index: 95;
        opacity: 0.8;
        position: absolute;
        top:0;
        bottom: 0;
        left: 0;
        right: 0;
        background-color:rgba(92, 184, 184, 0.38);
        filter: blur(5px);
        display: none;
    }

    .add-dialog {
        position: absolute;
        bottom: 50px;
        z-index: 100;
        right: 0px;
        display: none;
    }

    #films-add-button {
        bottom:25px;
        position: fixed;
        right: 80px;
        background-color: #5cb85c ;
        z-index: 100;
        margin-right: 20px;
    }
    #films-add-button:hover{
        background-color: #6ecc6e;
    }

    /*create country*/
    .add-prop-panel-outer{
        height: 80px;
        margin-top: 20px;
    }
    .add-prop-panel{
        height: 100%;
    }
    .add-prop-btn{
        width: 165px;
    }

    .description-form{
        margin-top: 40px;
    }

    #add-film-button, #cancel-film-button{
        width:200px;
        margin-right:20px;
    }

```

// film.jsp – показывает информацию о фильмах


```

<%--
    Created by IntelliJ IDEA.
    User: Lytki
    Date: 22.03.2017
    Time: 23:18
    To change this template use File | Settings | File Templates.
--%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ page contentType="text/html; charset=UTF-8" language="java" %>
<html>
<head>
    <title>Simple JSP</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="libs/bootstrap/css/bootstrap.min.css">
    <link rel="stylesheet" href="css/style.css">
    <link rel="stylesheet" href="libs/bootstrap/css/bootstrap-theme.min.css">
    <script src="libs/jquery.min.js"></script>
    <script src="js/film.js"></script>
    <script src="libs/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>
<div class="row">
    <nav class="navbar navbar-inverse navbar-fixed-top" role="navigation">
        <div class="container">
            <div class="navbar-header">
                
                <a class="navbar-brand" href="#">Кинобаза</a>
            </div>
            <ul class="nav navbar-nav">
                <li><a href="index.jsp">Фильмы</a></li>
                <li><a href="genres.jsp">Жанры</a></li>
                <li><a href="countries.jsp">Страны</a></li>
                <li><a href="peoples.jsp">Люди</a></li>
            </ul>
        </div>
    </nav>
</div>
<div class="container" style="margin-top: 100px">
    <div class="col-md-8 col-md-offset-2">
        <div class="col-md-10 col-md-offset-1" id="video">

        </div>

        <div class="page-header" id="title" style="text-align: center; margin-top: 30px;">

        </div>

        <div class="panel panel-default" id="description">
            <div class="panel-heading">Описание</div>
            <div class="panel-body">
            </div>
        </div>

        <div class="panel panel-default" id="actors">
            <div class="panel-heading">Актеры</div>
            <div class="panel-body">
            </div>
        </div>
    </div>
</div>

```

```

</div>

<div class="panel panel-default" id="producers">
  <div class="panel-heading">Режиссеры</div>
  <div class="panel-body">
  </div>
</div>

<div class="panel panel-default" id="genres">
  <div class="panel-heading">Жанр</div>
  <div class="panel-body">
  </div>
</div>

<div class="panel panel-default" id="countries">
  <div class="panel-heading">Страна</div>
  <div class="panel-body">
  </div>
</div>

</div>

<div id="add-dialog-button" class="action-button" style="right:140px;
position: fixed;">
  <i class=" glyphicon glyphicon-pencil" title="Редактировать"></i>
</div>

<div id="cancel-add-button" class="action-button" style="right:140px;
bottom: 120px; position: fixed;">
  <i class="glyphicon glyphicon-trash" title="Удалить"></i>
</div>

</div>

</body>
</html>

//film.js
$(document).ready( ()=>{
  ID=getUrlParameter("id");
  $.get('/films?id='+ID, (data)=>{
    var res = data;
    tit=res.title;
    $('#title').html('<h1 style="margin-top: 268px;">${res.title}</h1>
    <h4>${res.releaseDate} год</h4>
    <h5>Оценка ${res.raiting}/5</h5>
    `);
    $('#description>.panel-body').html(res.description);
    $('#video').append('<iframe width="100%" height="50%"
src="${res.trailerUrl}" frameborder="0" allowfullscreen></iframe>`);

    var actors = res.actors;

    var $actors = $('#actors>.panel-body');
    for(var i=0;i<actors.length;i++){
      var template=`<span class="label label-default" style="margin-
bottom:5px; margin-left: 5px;">
        ${actors[i].firstName+" "+actors[i].lastName}
      </span>`;

      $actors.append(template);
    }
  })
})

```

```

        var $producers = $('#producers>.panel-body');
        for(var i=0;i<res.producers.length;i++){
            var template=`<span class="label label-default" style="margin-
bottom:5px; margin-left: 5px;">
                ${res.producers[i].firstName+" "+res.producers[i].lastName}
            </span>`;

            $producers.append(template);
        }

        var $genres=$('#>#genres>.panel-body');
        for(var i=0;i<res.genres.length;i++){
            var template=`<span class="label label-primary" style="margin-
bottom:5px; margin-left: 5px;">
                ${res.genres[i].name}
            </span>`;

            $genres.append(template);
        }

        var $countries = $('#countries>.panel-body');
        for(var i=0;i<res.countries.length;i++){
            var template=`<span class="label label-success" style="margin-
bottom:5px; margin-left: 5px;">
                ${res.countries[i].countryName}
            </span>`;

            $countries.append(template);
        }

    });
    $('#cancel-add-button').click(()=>{

        var isDelete = confirm("Вы действительно хотите удалить фильм
        '"+tit+"'?");

        if(isDelete){
            $.post('/films',{id:ID, action:"DELETE"},()=>{
                location.href="index.jsp";
            });
        }

    });

    $('#add-dialog-button').click(()=>{
        location.href="update_film.jsp?id="+ID;
    });
});

var ID;
var tit;

function getUrlParameter(sParam) {
    var sPageURL = decodeURIComponent(window.location.search.substring(1)),
        sURLVariables = sPageURL.split('&'),
        sParameterName,
        i;

    for (i = 0; i < sURLVariables.length; i++) {
        sParameterName = sURLVariables[i].split('=');

        if (sParameterName[0] === sParam) {
            return sParameterName[1] === undefined ? true :

```

```

sParameterName[1];
    }
}
};
function initDropdowns() {
    for (var j = 1970; j < 2018; j++) {
        $('#years').append('<option>${j}</option>');
    }

    for (var k = 1; k < 6; k++) {
        $('#raiting').append('<option>${k}</option>');
    }
}
$(document).ready(()=>{

    initTables();
    initDropdowns();
    initDialogAdders();

    $('#add-film-button').click(()=>{
        title=$('#title').val();
        video=$('#trailer').val();
        release=$('#years').val();
        raiting=$('#raiting').val();
        description=$('#description').val();

        var result ={
            action:"ADD",
            title: title,
            release:release,
            raiting: raiting,
            description: description,
            url:video,
            genres: JSON.stringify(genres),
            actors: JSON.stringify(actors),
            producers: JSON.stringify(producers),
            countries: JSON.stringify(countries)
        };

        $.post('/films',result,function(data){
            location.href="film.jsp?id="+data;
        });

    });

});

var actors=[];
var producers=[];
var countries=[];
var genres=[];

var title;
var release;
var raiting;
var description;
var video;

function initTables() {
    getAllGenres();
    getAllCountries();
}

```

```

        getAllPeoples();
    }

    function getAllGenres() {
        $.get("/genres", (data) => {
            var genres = data.genres;
            var $table = $('.table-genres>tbody');
            $table.empty();

            genres.forEach((genre) => {
                var tempalte = `<tr>

                    <th scope="row">
                        <input type="checkbox" attr-id="${genre.id}" name="checkbox-
genres"
                        style="margin-left:auto; margin-right:auto;">
                    </th>
                    <td class="genre-name-${genre.id}">${genre.name}</td>
                </tr>`;
                $table.append(tempalte);
            });
        })
    }

    function getAllCountries() {
        $.get("/countries", (data) => {
            var countries = data.countries;
            var $table = $('.table-countries>tbody');
            $table.empty();

            countries.forEach((country) => {
                var tempalte = `<tr>
                    <th scope="row">
                        <input attr-id="${country.id}" type="checkbox"
name="checkbox-countries" value="checked"
                        style="margin-left:auto; margin-right:auto;">
                    </th>
                    <td class="country-name-
${country.id}">${country.countryName}</td>
                </tr>`;
                $table.append(tempalte);
            });
        })
    }

    function getAllPeoples() {
        $.get("/peoples", (data) => {
            var chbActor = "checkbox-actor";
            var chbProd = "checkbox-producer";
            var countries = data.peoples;
            var $tableActors = $('.table-actors>tbody');
            $tableActors.empty();
            var $tableProducers = $('.table-producers>tbody');
            $tableProducers.empty();

            countries.forEach((people) => {
                var tempalte1 = `<tr>
                    <th scope="row">
                        <input attr-id="${people.id}" type="checkbox"

```

```

name="${chbActor}"
                                style="margin-left:auto; margin-
right:auto;">
    </th>
    <td class="actor-name-${people.id}">${people.firstName}</td>
    <td class="actor-lastname-
${people.id}">${people.lastName}</td>
    </tr>`;
    $tableActors.append(tempalte1);

    var tempalte2 = `<tr>
    <th scope="row">
    <input attr-id="${people.id}" type="checkbox"
name="${chbProd}"
                                style="margin-left:auto; margin-
right:auto;">
    </th>
    <td class="producer-name-
${people.id}">${people.firstName}</td>
    <td class="producer-lastname-
${people.id}">${people.lastName}</td>
    </tr>`;
    $tableProducers.append(tempalte2);
    });
    })
}

function initDialogAdders() {
    $('#add-actors-dialog-btn').click(() => {
        var checked = $("input[name=checkbox-actor]:checked");
        var $actorList = $("#actors-list");
        $actorList.empty();
        for (var i = 0; i < checked.length; i++) {

            var ID = $(checked[i]).attr("attr-id");
            var firstName = $(".actor-name-" + ID).text();
            var lastName = $(".actor-lastname-" + ID).text();
            var template = `<span class="label label-default"
style="margin-bottom:5px;">${firstName + " " + lastName}</span>`;
            actors.push(ID);
            $actorList.append(template);
            $("#modal-actors").modal('hide');
        }
    });

    $('#add-producers-dialog-btn').click(() => {
        var checked = $("input[name=checkbox-producer]:checked");
        var $producerList = $("#producers-list");
        $producerList.empty();
        for (var i = 0; i < checked.length; i++) {

            var ID = $(checked[i]).attr("attr-id");
            var firstName = $(".producer-name-" + ID).text();
            var lastName = $(".producer-lastname-" + ID).text();
            var template = `<span class="label label-default"
style="margin-bottom:5px;">${firstName + " " + lastName}</span>`;
            producers.push(ID);
            $producerList.append(template);
            $("#modal-producers").modal('hide');
        }
    });
}

```

```

$('#add-genres-dialog-btn').click(=> {
  var checked = $("input[name=checkbox-genres]:checked");
  var $genresList = $("#genres-list");
  $genresList.empty();
  for (var i = 0; i < checked.length; i++) {

    var ID = $(checked[i]).attr("attr-id");
    var genre = $(".genre-name-" + ID).text();
    var template = `<span class="label label-default"
style="margin-bottom:5px;">${genre}</span>`;
    genres.push(ID);
    $genresList.append(template);
    $("#modal-genres").modal('hide');
  }
});

$('#add-countries-dialog-btn').click(=> {
  var checked = $("input[name=checkbox-countries]:checked");
  var $countriesList = $("#countries-list");
  $countriesList.empty();
  for (var i = 0; i < checked.length; i++) {

    var ID = $(checked[i]).attr("attr-id");
    var country = $(".country-name-" + ID).text();
    var template = `<span class="label label-default"
style="margin-bottom:5px;">${country}</span>`;
    countries.push(ID);
    $countriesList.append(template);
    $("#modal-countries").modal('hide');
  }
});
}

```