федеральное государственное автономное образовательное учреждение высшего образования

«Самарский национальный исследовательский университет имени академика С.П. Королева»

(Самарский университет)

Факультет информатики

Кафедра технической кибернетики

Отчёт по курсовому проекту

по дисциплине

«Технологии сетевого программирования»

**Информационная система учёта характеристик персонажей комиксов**

Выполнили: Китаев Д.Н., Булатов А.В.

Группа: 6406

Руководитель работы: Гаврилов А.В.

САМАРА 2016

ЗАДАНИЕ

Разработать приложение на основе JAVA-EE технологий. Система должна предоставлять возможность просмотра и редактирования данных из базы данных, а также выполнять и показывать результаты запросов (виды запросов определяются на этапе создания системы). Вся работа с системой производится через Web-интерфейс.

Система должна обеспечивать работу с базой данных героев комиксов. В процессе работы использовать следующие технологии: PostgreSQL, Hibernate, Java Server Pages, CSS.

РЕФЕРАТ

Отчет 27 с, 9 рисунков, 1 таблица, 4 источника, 5 приложений.

PostgreSQL, JSP, Hibernate, Tomcat, CSS.

Объектом исследования является информационная система, обеспечивающая взаимосвязанную работу базы данных и Java-EE приложения.

Цель работы - разработать информационную систему на основе JAVA-EE технологий.

В процессе работы были использованы следующие технологии: PostgreSQL, Hibernate, Java Server Pages, CSS.

В результате работы была получена информационная система, функционирующая в соответствии с поставленным заданием.

СОДЕРЖАНИЕ

[Введение 5](#_Toc350892062)

[1 Структура базы данных](#_Toc350892063) 6

[1.1 Логическая и физическая схемы](#_Toc350892064) 6

[1.2 Описание таблиц](#_Toc350892065) 8

[2 Архитектура приложения](#_Toc350892066) 10

[3 Интерфейс пользователя](#_Toc350892067) 13

[Заключение](#_Toc350892068) 17

Список использованных источников 18

[Приложение А. SQL-скрипты создания и заполнения таблиц](#_Toc350892073) 19

[Приложение Б. Примеры ORM-компонента](#_Toc350892074) 21

[Приложение В.](#_Toc350892075)  Пример JSP-страницы...…………………………..……22

[Приложение Г. Дескриптор развертывания web-приложения …..…….](#_Toc350892075)26

[Приложение Д. Файл конфигурации Hibernate](#_Toc350892075) 27

ВВЕДЕНИЕ

DC Comics и Marvel самые крупные и популярные издательства комиксов, основанные в 1934 и в 1939 году соответственно, расположенные в Нью-Йорке, которые также занимаются в отрасли кино, сериалов и видеоигр. Наиболее известные работы данных компаний: Marvel - «Человек-паук», «Железный человек», «Тор», «Капитан Америка», «Люди Икс» и «Дэдпул»;  
DC - «Супермен», «Бэтмен», «Зелёный Фонарь», «Флэш», «Зелёная стрела».

Данные работы имеют миллионы фанатов, которые следят за историей персонажей, появившихся и созданных во вселенных.

Все вышесказанное приводит к необходимости создания информационной системы учета персонажей, их возможностей.

При построении такой системы можно использовать возможности технологии Java, а также компоненты JDBC в качестве работы с базой данных. Программы, созданные с помощью Java и JDBC, не зависят от программного обеспечения и используемой платформы. При этом возможно перемещение данных из одной базы данных в другую, например из PostgreSQL в MySQL и т.п. В результате работать с данными могут различные программы.

1 СТРУКТУРА БАЗЫ ДАННЫХ

1.1 Логическая и физическая схемы

Логическая схема определяет структуру данных, обычно с указанием [первичных ключей](http://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B2%D0%B8%D1%87%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BB%D1%8E%D1%87), а также «связей» между отношениями, представляющих собой [внешние ключи](http://ru.wikipedia.org/wiki/%D0%92%D0%BD%D0%B5%D1%88%D0%BD%D0%B8%D0%B9_%D0%BA%D0%BB%D1%8E%D1%87).

Физическая схема определяет размещение данных и пути доступа к данным.

На рисунке 1 приведена логическая схема используемой базы данных, на рисунке 2 – физическая.

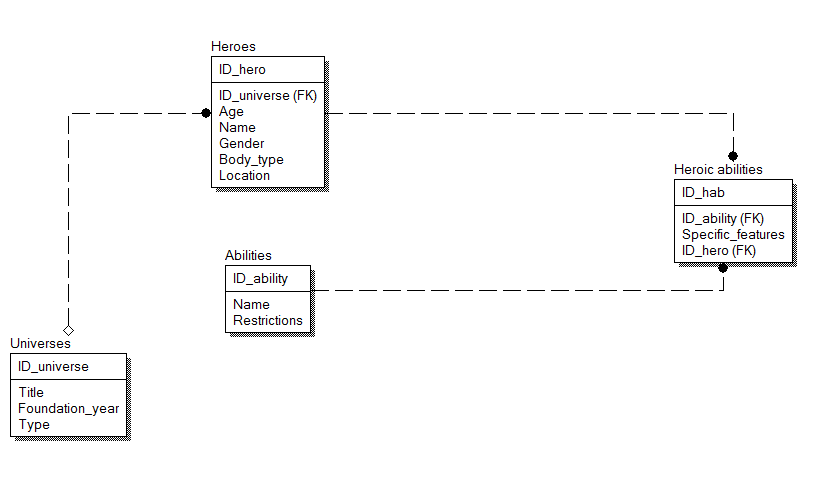


Рисунок 1 –Логическая схема базы данных

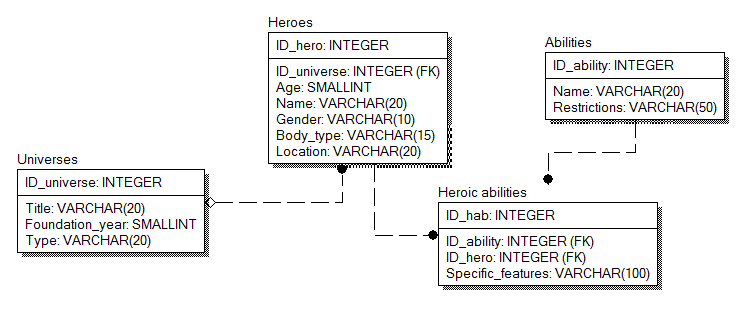


Рисунок 2 – Физическая схема базы данных

SQL-скрипты создания и заполнения таблиц приведены в приложении А.

1.2 Описание таблиц

Таблица universes имеет 4 атрибута: id\_universe (первичный ключ), foundation\_year, title, type. Таблица abilities содержит 3 атрибута: id\_ability (первичный ключ), name и restrictions. Таблица heroes состоит из 7 атрибутов: id\_hero (первичный ключ), name, age, body\_type, gender, location, id\_universe (внешний ключ для связи с таблицей universes). Таблица heroic\_abilities содержит 4 атрибута: id\_hab(первичный ключ), specific\_features, id\_ability (внешний ключ для связи с таблицей abilities), id\_hero (внешний ключ для связи с таблицей heroes).

Краткое описание таблиц используемой схемы базы данных приведено в таблице 1.

Таблица 1 - Описание таблиц базы данных

|  |  |  |
| --- | --- | --- |
| Таблица | Атрибут | Описание |
| universes | id\_universe | Уникальный идентификатор вселенной |
| foundation\_year | Год создания компании |
| Title | Название вселенной |
| Type | Отрасль компании |
| abilities | id\_ability | Уникальный идентификатор способности |
| Name | Название способности |
| Restrictions | Краткое описание способности |
| heroes | id\_hero | Уникальный идентификатор героя |
| Name | Имя героя |
| Age | Возраст персонажа |
| body\_type | Телосложение персонажа |
| Gender | Пол персонажа |
| Location | Место нахождения героя |
| id\_universe | Идентификатор вселенной в которой создан герой |

Продолжение таблицы 1

|  |  |  |
| --- | --- | --- |
| Таблица | Атрибут | Описание |
| heroic\_abilities | id\_hab | Уникальный идентификатор таблицы связи |
| specific\_features | Дополнительное описание способности героя |
| id\_ability | Идентификатор способности приписанная герою |
| id\_hero | Идентификатор героя, которому приписана способность |

2 АРХИТЕКТУРА ПРИЛОЖЕНИЯ

При создании информационной системы использовались следующие средства: PostgreSQL, JSP, Hibernate, Tomcat, CSS.

PostgreSQL – система управления базами данных, которая поддерживает большой набор типов данных. Является некоммерческой и существует в реализациях для множества платформ. К достоинствам данной СУБД можно отнести возможность расширения функционала, за счёт сохранения своих процедур и то, что она является объектно-ориентированной СУБД, с поддержкой наследования. К минусам можно отнести низкую производительность при простых операциях.

Hibernate – ORM (Object-relation mapping) технология, которая связывает базу данных с объектно-ориентированным языком. Является библиотекой для Java предназначенной для решения такого типа задач. Используется так называемый “mapping” который сопоставляет java классы с таблицами из базы данных. Одна из реализаций “маппинга” это аннотации. В качестве используемых аннотаций могут быть:

@Entity Указывает, что данный класс является сущностью БД;

@Table Указывает на имя таблицы в базе данных;

@Id Определяет первичный ключ;

@Colum Указывает имя столбца, с которым связано поле класса.

Java Server Pages – технология, созданная для упрощения работы с Web-содержимым. JSP – это страница, которая представляет из себя шаблон с уже заготовленными HTML-тэгами, между которыми надо вставить нужные данные. Позволяет отделить динамическую часть страницы от статической, с помощью специальных тегов.

Пример класса ORM приведён в приложении Б, JSP-страницы – в приложении В, дескриптор развертывания web-приложения – в приложении Г, файл конфигурации – в приложении Д.

Архитектурные решения приложения представлены в виде UML‑диаграмм. На рисунке 3 представлена структуру и взаимоотношения классов пакета Entity. Рисунок 4 описывает структуру классов пакета DAO.

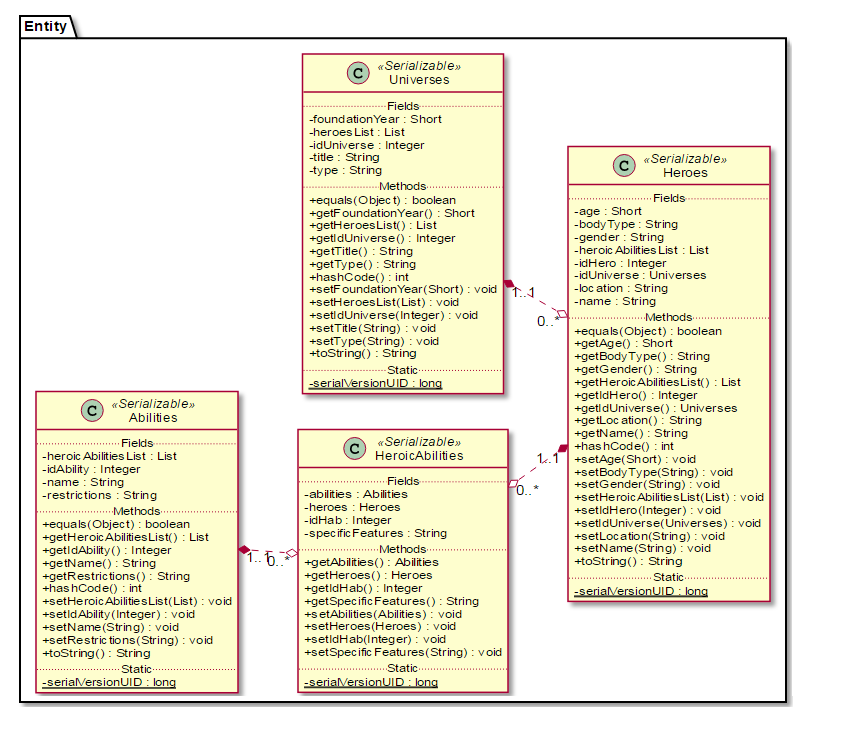


Рисунок 3 – UML‑диаграмма классов пакета Entity

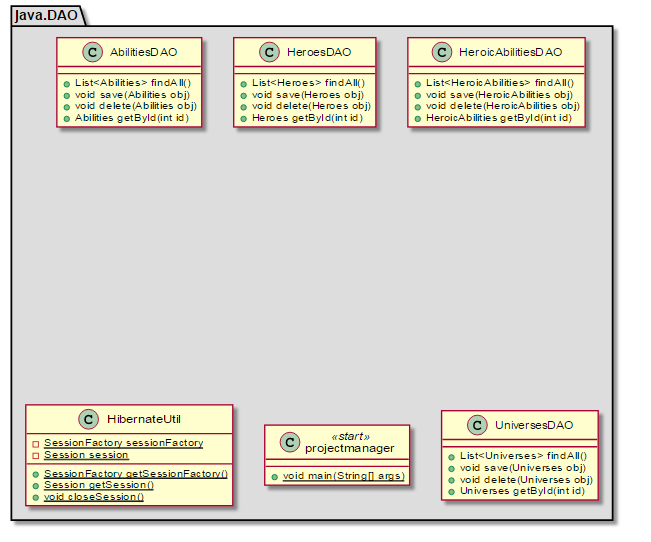


Рисунок 4 – UML‑диаграмма структура классов пакета DAO

3 ИНТЕРФЕЙС ПОЛЬЗОВАТЕЛЯ

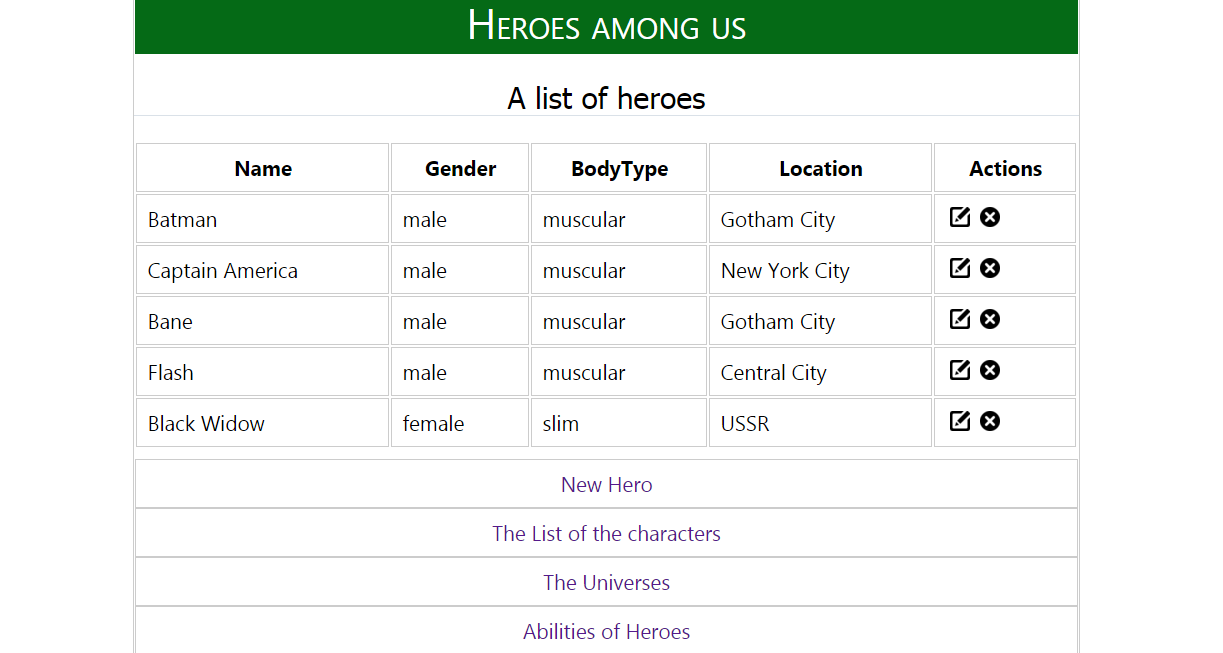
Сначала пользователь попадает на страницу, где ему предоставляется возможность просмотреть список героев (рисунок 5). Так же на этой странице есть вкладки перехода на список способностей, соответствующие различным героям, вкладка «The list of the characters», вкладка перехода на список вселенных «The Universes», вкладка перехода на список героев и их способностей «Abilities of Heroes» и вкладка создания нового героя «New Hero». Справа, на всех страницах, находится столбец Action, с помощью которого можно изменить или удалить существующую запись.

Рисунок 5 – Начальная страница

При переходе по вкладке «The Universes»пользователю предоставляется список вселенных (рисунок 6). Если пользователь решит создать новую вселенную, необходимо перейти по вкладке «New Universe». Для того чтобы вернуться к списку героев, необходимо перейти по вкладке «A list of Hero».

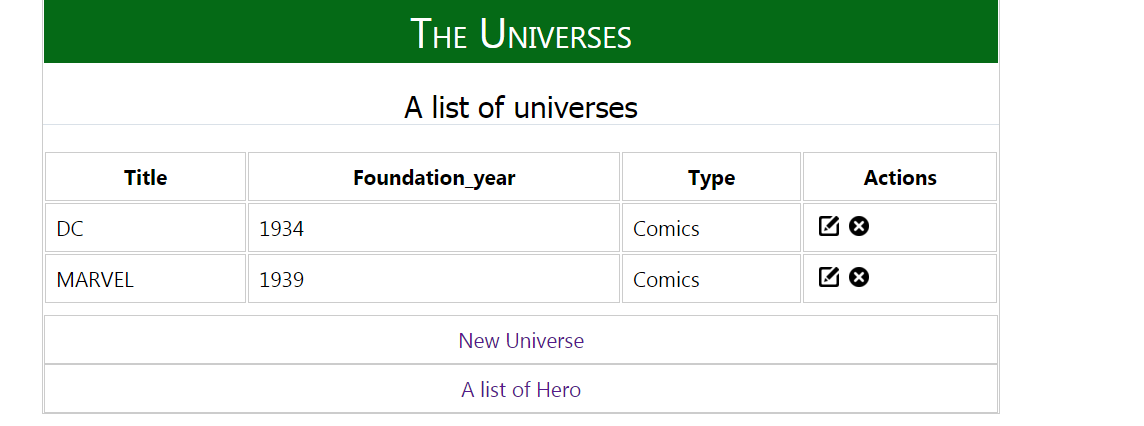


Рисунок 6 – Страница со списком вселенных

При переходе из начальной страницы по вкладке «The list of the characters» пользователь переходит к списку способностей героев (рисунок 7). Имеются вкладки «New Ability», для добавления новой способности, вкладка перехода к списку героев «A list of Heroes» и вкладка перехода на страницу списка героев и соответствующих им способностей «A list of Heroic Abilities».

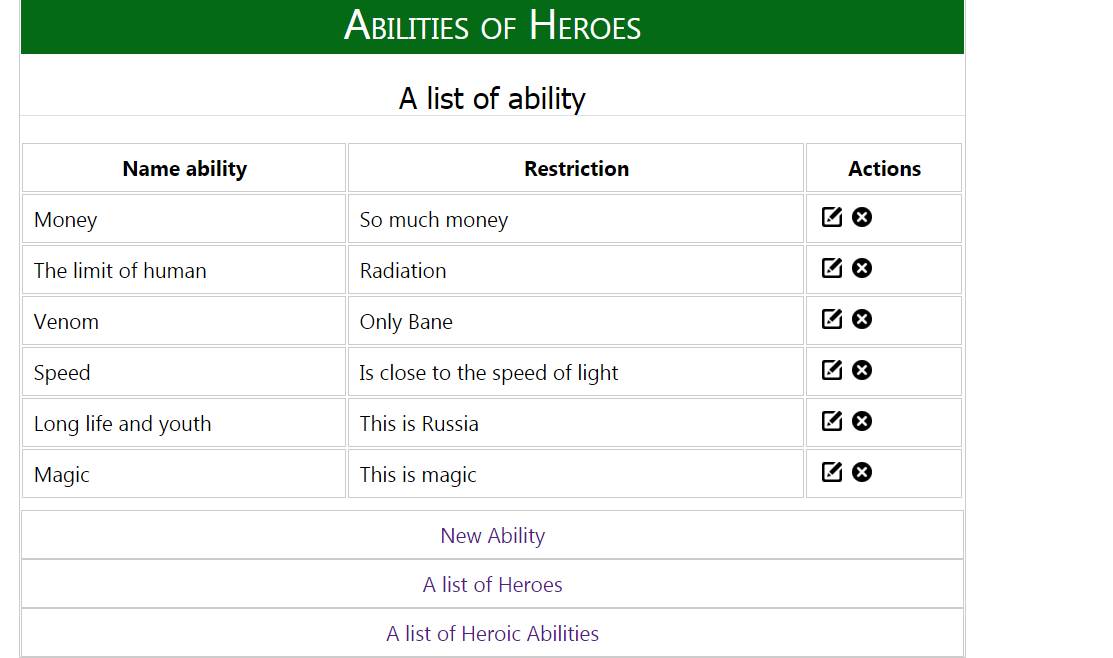


Рисунок 7 – Страница со списком способностей

Общий вид страницы создания новой записи (героя, способности и т.д.) представлен на рисунке 8. Пользователю необходимо заполнить все поля и после нажатия на кнопку «Save», будет осуществлён переход к соответствующему списку. Так же имеется вкладка перехода к необходимому списку, при отмене создания новой записи.

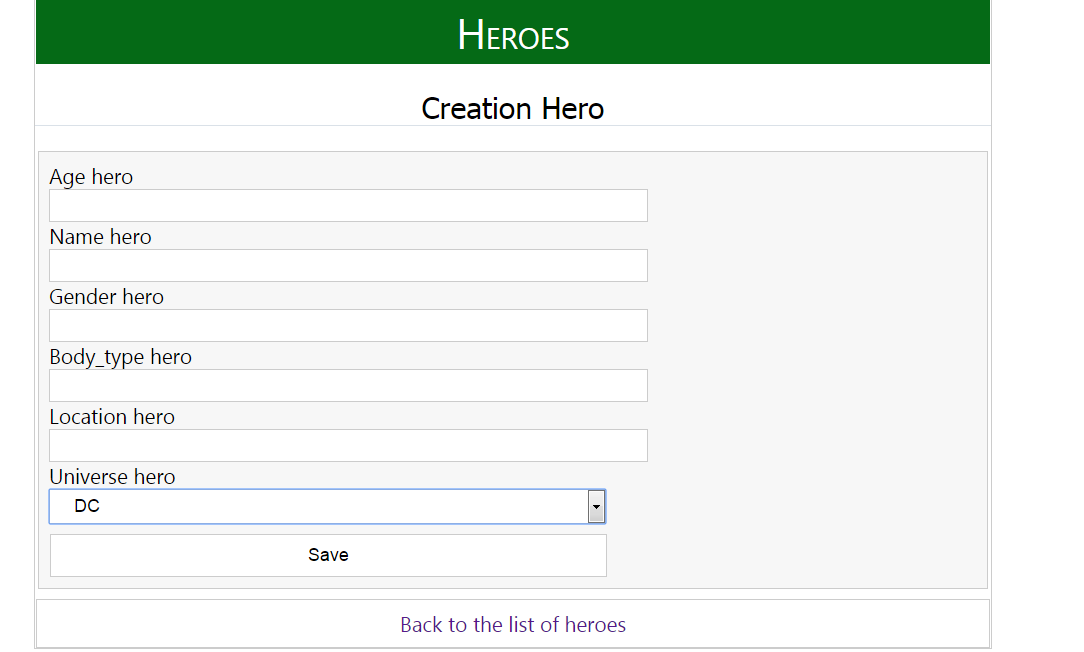


Рисунок 8 – Пример страницы создания новой записи

Страница, предоставляющая список героев и соответствующих им способностей представлена на рисунке 9. Страница также имеет вкладки перехода к списку героев, к списку способностей и вкладку создания новой связи между героем и соответствующей ему способности.

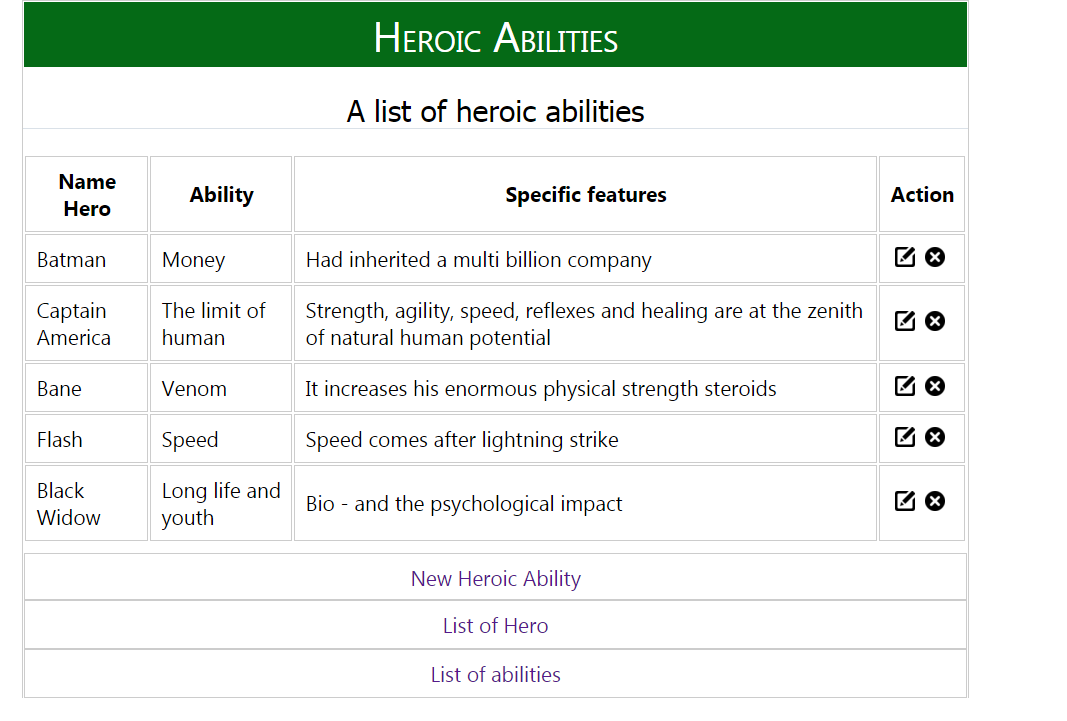


Рисунок 9 – Страница со списком героев и их способностей

ЗАКЛЮЧЕНИЕ

В результате работы разработана информационная система учёта характеристик персонажей комиксов, предоставляющая возможность просмотра и редактирования данных из базы данных. Пользователь может просматривать список героев, способностей, вселенных. Пользователю также доступна возможность удалять, редактировать существующие записи и создавать новые.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Справочное руководство Oracle [Электронный ресурс]. - Режим доступа: http://docs.oracle.com

2 Форум Java программистов [Электронный ресурс]. - Режим доступа: javatalks.ru

3 Форум stackoverflow [Электронный ресурс]. - Режим доступа: stackoverflow.com

4 Форум SQL.ru [Электронный ресурс]. – Режим доступа:

http://www.sql.ru

ПРИЛОЖЕНИЕ А

SQL-скрипты создания и заполнения таблиц

CREATE TABLE UNIVERSES(

ID\_UNIVERSE INTEGER NOT NULL,

TITLE VARCHAR(20) NOT NULL,

FOUNDATION\_YEAR SMALLINT,

TYPE VARCHAR(20),

PRIMARY KEY (ID\_UNIVERSE)

);

CREATE TABLE HEROES(

ID\_HERO INTEGER NOT NULL,

ID\_UNIVERSE INTEGER NOT NULL,

AGE SMALLINT,

NAME VARCHAR(20) NOT NULL,

GENDER VARCHAR(10),

BODY\_TYPE VARCHAR(15),

LOCATION VARCHAR(20),

PRIMARY KEY (ID\_HERO)

);

ALTER TABLE HEROES add

FOREIGN KEY (ID\_UNIVERSE) REFERENCES UNIVERSES (ID\_UNIVERSE);

CREATE TABLE ABILITIES(

ID\_ABILITY INTEGER NOT NULL,

NAME VARCHAR(20) NOT NULL,

RESTRICTIONS VARCHAR(50),

PRIMARY KEY (ID\_ABILITY)

);

CREATE TABLE HEROIC\_ABILITIES

(

ID\_HAB SERIAL NOT NULL,

SPECIFIC\_FEATURES CHARACTER VARYING(255),

ID\_ABILITY INTEGER NOT NULL,

ID\_HERO INTEGER NOT NULL,

PRIMARY KEY (ID\_HAB),

FOREIGN KEY (ID\_ABILITY)

FOREIGN KEY (ID\_HERO)

)

WITH (

OIDS=FALSE

);

INSERT INTO UNIVERSES (id\_universe,title,foundation\_year,type) VALUES(0,'DC',1934,'Comics') ;

INSERT INTO UNIVERSES (id\_universe,title,foundation\_year,type) VALUES(1,'MARVEL',1939,'Comics');

INSERT INTO HEROES (id\_hero,id\_universe,age,name,gender,body\_type,location) VALUES(0,0,34,'Batman','male','muscular','Gotham City');

INSERT INTO HEROES (id\_hero,id\_universe,age,name,gender,body\_type,location) VALUES(1,1,99,'Captain America','male','muscular','New York City');

INSERT INTO HEROES (id\_hero,id\_universe,age,name,gender,body\_type,location) VALUES(2,0,38,'Bane','male','muscular','Gotham City');

INSERT INTO HEROES (id\_hero,id\_universe,age,name,gender,body\_type,location) VALUES(3,0,24,'Flash','male','muscular','Central City');

INSERT INTO HEROES (id\_hero,id\_universe,age,name,gender,body\_type,location) VALUES(4,1,30,'Black Widow','female','slim','SSSR');

INSERT INTO HEROES (id\_hero,id\_universe,age,name,gender,body\_type,location) VALUES(5,1,43,'Doctor Strange','male','muscular','New York City');

INSERT INTO ABILITIES (id\_ability,name,restrictions) VALUES(0,'Money', 'So much money');

INSERT INTO ABILITIES (id\_ability,name,restrictions) VALUES(1,'The limit of human','Radiation');

INSERT INTO ABILITIES (id\_ability,name,restrictions) VALUES(2,'Venom','Only Bane');

INSERT INTO ABILITIES (id\_ability,name,restrictions) VALUES(3,'Speed','Is close to the speed of light');

INSERT INTO ABILITIES (id\_ability,name,restrictions) VALUES(4,'Long life and youth','This is Russia');

INSERT INTO ABILITIES (id\_ability,name,restrictions) VALUES(5,'Magic','This is magic');

INSERT INTO HEROIC\_ABILITIES (id\_hab, id\_hero,id\_ability,specific\_features) VALUES(0,0,0,'Had inherited a multi billion company') ;

INSERT INTO HEROIC\_ABILITIES (id\_hab, id\_hero,id\_ability,specific\_features) VALUES(1,1,1,'Strength, agility, speed, reflexes and healing are at the zenith of natural human potential') ;

INSERT INTO HEROIC\_ABILITIES (id\_hab, id\_hero,id\_ability,specific\_features) VALUES(2,2,2,'It increases his enormous physical strength steroids') ;

INSERT INTO HEROIC\_ABILITIES (id\_hab, id\_hero,id\_ability,specific\_features) VALUES(3,3,3,'Speed comes after lightning strike') ;

INSERT INTO HEROIC\_ABILITIES (id\_hab, id\_hero,id\_ability,specific\_features) VALUES(4,4,4,'Bio - and the psychological impact') ;

INSERT INTO HEROIC\_ABILITIES (id\_hab, id\_hero,id\_ability,specific\_features) VALUES(5,5,5,'even years studying magic') ;

ПРИЛОЖЕНИЕ Б

Пример ORM-компонента

package DAO;

import java.util.List;

import Entity.Heroes;

import org.hibernate.Query;

import org.hibernate.Session;

public class HeroesDAO {

public HeroesDAO(){

}

public List<Heroes> findAll() {

Query query = HibernateUtil.getSession().createQuery("SELECT h FROM Heroes h");

List<Heroes> result = (List<Heroes>) query.list();

return result;

}

public void save(Heroes obj) {

Session session = HibernateUtil.getSession();

session.beginTransaction();

session.saveOrUpdate(obj);

session.getTransaction().commit();

HibernateUtil.closeSession();

}

public void delete(Heroes obj) {

HibernateUtil.closeSession();

Session session = HibernateUtil.getSession();

session.beginTransaction();

session.delete(obj);

session.getTransaction().commit();

}

public Heroes getById(int id) {

Session session = HibernateUtil.getSession();

Heroes result = (Heroes)session.get(Heroes.class, id);

return result;

}

}

ПРИЛОЖЕНИЕ В

Пример JSP - страницы

<%@page import="com.sun.javafx.image.impl.IntArgb"%>

<%@page import="Entity.Universes"%>

<%@page import="java.util.List"%>

<%@page import="Entity.Heroes"%>

<%@page import="DAO.UniversesDAO"%>

<%@page import="DAO.HeroicAbilitiesDAO"%>

<%@page import="DAO.HeroesDAO"%>

<%@page import="DAO.AbilitiesDAO"%>

<%@page import="java.net.URLDecoder"%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>

<%!

private HeroesDAO hdao = new HeroesDAO();

private UniversesDAO undao = new UniversesDAO();

%>

<%

String delId = request.getParameter("delId");

if(delId != null)

{

Heroes deleteHeroes = hdao.getById(Integer.parseInt(delId));

try {

hdao.delete(deleteHeroes);

} catch (NullPointerException e) {

System.err.print(e);

}

response.sendRedirect("");

return;

}

%>

<%

Heroes heroes = null;

String idStr = request.getParameter("editId");

if(idStr != null){

try {

int id = Integer.parseInt(idStr);

heroes = hdao.getById(id);

} catch (NullPointerException e){

System.err.print(e);

}

}

%>

<!DOCTYPE html>

<html>

<head>

<meta http-equiv="Content-Type" content="=text/html; charset = UTF-8">

<title>Heroes among us</title>

<link rel="stylesheet" type="text/css" href="style.css">

</head>

<body>

<h1>Heroes</h1>

<h2><%

if(heroes != null)

out.write("Change Hero");

else

out.write("Creation Hero");

%> </h2>

<p>

<%

if(request.getParameter("save") != null)

{

if (heroes == null)

heroes = new Heroes();

String param;

param = request.getParameter("age");

if (!param.isEmpty())

heroes.setAge(Short.parseShort(param));

param = request.getParameter("name");

if (!param.isEmpty()){

heroes.setName(new String(request.getParameter("name").getBytes("iso-8859-1"),"UTF-8"));

} else {

heroes.setName("");

out.write("<p class = 'err'>Is not given a name</p>");

}

param = request.getParameter("gender");

if (!param.isEmpty()){

heroes.setGender(new String(request.getParameter("gender").getBytes("iso-8859-1"),"UTF-8"));

} else {

heroes.setName("");

out.write("<p class = 'err'>Is not given a name</p>");

}

param = request.getParameter("body\_type");

if (!param.isEmpty()){

heroes.setBodyType(new String(request.getParameter("body\_type").getBytes("iso-8859-1"),"UTF-8"));

} else {

heroes.setBodyType("");

out.write("<p class = 'err'>Is not given a body type</p>");

}

param = request.getParameter("location");

if (!param.isEmpty()){

heroes.setLocation(new String(request.getParameter("location").getBytes("iso-8859-1"),"UTF-8"));

} else {

heroes.setLocation("");

out.write("<p class = 'err'>Is not given a location</p>");

}

param = request.getParameter("id\_universe");

if (!param.isEmpty())

heroes.setIdUniverse(undao.getById(Integer.parseInt(param)));

hdao.save(heroes);

response.sendRedirect("");

return;

}

%>

</p>

<form method = "post">

Age hero<br>

<input type="number" name="age" value="<%=(heroes != null) ? heroes.getAge(): ""%>"/><br>

Name hero<br>

<input type="text" name="name" value="<%=(heroes != null) ? heroes.getName() : ""%>"/><br>

Gender hero<br>

<input type="text" name="gender" value="<%=(heroes != null) ? heroes.getGender(): ""%>"/><br>

Body\_type hero<br>

<input type="text" name="body\_type" value="<%=(heroes != null) ? heroes.getBodyType(): ""%>"/><br>

Location hero<br>

<input type="text" name="location" value="<%=(heroes != null) ? heroes.getLocation(): ""%>"/><br>

Universe hero<br>

<select name="id\_universe">

<%

List<Universes> un = undao.findAll();

for(Universes u: un)

{

out.write("<option ");

if (heroes != null && heroes.getIdUniverse().getIdUniverse() == u.getIdUniverse())

{

out.write("selected ");

}

out.write("value =" + u.getIdUniverse() + ">" + u.getTitle() + "\n");

}

%>

</select><br>

<input type="submit" value="Save" name="save" />

</form>

<div class="foot">

<a class="home" href ="heroeslist.jsp">Back to the list of heroes</a>

</div>

</body>

</html>

ПРИЛОЖЕНИЕ Г

Дескриптор развёртывания web - приложения

<?xml version="1.0" encoding="UTF-8"?>

<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app\_3\_1.xsd"

version="3.1">

<welcome-file-list>

<welcome-file>/heroeslist.jsp</welcome-file>

</welcome-file-list>

</web-app>

ПРИЛОЖЕНИЕ Д

Файл конфигурации Hibernate

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration DTD 3.0//EN" "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">

<hibernate-configuration>

<session-factory>

<property name="hibernate.connection.driver\_class">org.postgresql.Driver</property>

<property name="hibernate.dialect">org.hibernate.dialect.PostgreSQLDialect</property>

<property name="hibernate.connection.url">jdbc:postgresql://localhost:5432/postgres</property>

<property name="hibernate.connection.username">postgres</property>

<property name="hibernate.connection.password">root</property>

<property name="hbm2ddl.auto">update</property>

<property name="show\_sql">true</property>

<mapping class="Entity.Abilities"/>

<mapping class="Entity.Heroes"/>

<mapping class="Entity.HeroicAbilities"/>

<mapping class="Entity.Universes"/>

</session-factory>

</hibernate-configuration>