**Приложение А (обязательное)**

**Диаграммы модели разрабатываемого приложения в формате IDEF0**



Рисунок А.1 – Контекстная диаграмма

Приложение А (продолжение)



Рисунок А.2 – Диаграмма декомпозиции, уровень А-0

Приложение А (продолжение)



Рисунок А.3 – Диаграмма декомпозиции, уровень А0

Приложение А (продолжение)



Рисунок А.4 – Диаграмма декомпозиции, уровень А1

Приложение А (продолжение)



Рисунок А.5 – Диаграмма декомпозиции, уровень А13

**Приложение Б (обязательное)**

**UML-диаграммы разрабатываемой системы**



Рисунок Б.1 – Диаграмма вариантов использования

Приложение Б (продолжение)



Рисунок Б.2 – Диаграмма классов (модели)

Приложение Б (продолжение)



Рисунок Б.3 – Диаграмма классов (DAO)

Приложение Б (продолжение)



Рисунок Б.4 – Диаграмма последовательности

**Приложение В (обязательное)**

**Листинг кода скрипта генерации базы данных**

CREATE DATABASE IF NOT EXISTS `fin\_dept\_docs`;

USE `fin\_dept\_docs`;

CREATE USER 'dbadmin'@'localhost' IDENTIFIED BY 'dbadmin';

GRANT ALL PRIVILEGES ON \* . \* TO 'dbadmin'@'localhost';

--

-- Table structure for table `document`

--

DROP TABLE IF EXISTS `document`;

CREATE TABLE `document` (

 `id` int(10) unsigned NOT NULL AUTO\_INCREMENT,

 `input\_number` varchar(10) NOT NULL,

 `reg\_date` date NOT NULL,

 `name` varchar(200) NOT NULL,

 `type\_id` int(10) unsigned NOT NULL,

 `performer\_id` int(10) unsigned NOT NULL,

 `initiator\_id` int(10) unsigned NOT NULL,

 `deadline` date NOT NULL,

 `status\_id` int(10) unsigned NOT NULL,

 `file` varchar(200) DEFAULT NULL,

 `description` varchar(500) DEFAULT NULL,

 PRIMARY KEY (`id`),

 UNIQUE KEY `doc\_id\_UNIQUE` (`id`),

 UNIQUE KEY `file\_UNIQUE` (`file`),

 KEY `FK\_initiator` (`initiator\_id`),

 KEY `FK\_status` (`status\_id`),

 KEY `FK\_type` (`type\_id`),

 KEY `FK\_performer` (`performer\_id`)

) ENGINE=InnoDB AUTO\_INCREMENT=11 DEFAULT CHARSET=utf8;

--

-- data for table `document`

--

LOCK TABLES `document` WRITE;

INSERT INTO `document` VALUES (1,'01/16','2015-11-15','План реализации продукции на 1 кв. 2016 г.',5,2,3,'2015-12-15',5,'План реализации продукции на 1 кв. 2016 г..pdf',''),(2,'02/16','2016-02-16','План реализации \n\nпродукции на 2 кв. 2016 г.',5,3,3,'2016-03-16',5,NULL,NULL),(3,'03/16','2016-05-17','План реализации продукции на 3 кв. 2016 г.',5,4,3,'2016-06-17',3,NULL,NULL),(5,'01/17','2016-11-26','План реализации продукции на 1 кв. 2017 г.',5,9,3,'2016-12-26',1,NULL,NULL),(6,'02/17','2017-01-28','План реализации продукции на 2 кв. 2017 \n\nг.',5,7,3,'2017-02-28',2,NULL,NULL),(22,'5/17','2017-01-01','План амортизационных отчислений на 2017 год',6,17,2,'2017-12-31',2,'План амортизационных отчислений на 2017 год.pdf',''),(23,'5/16','2016-01-01','План амортизационных отчислений на 2016 год',6,17,2,'2016-12-31',1,NULL,''),(24,'5/15','2015-01-01','План амортизационных отчислений на 2015 год',6,17,2,'2015-12-31',3,NULL,''),(25,'6/16','2016-01-01','Кассовый план на 2 полугодие 2016',4,2,3,'2016-12-31',1,NULL,''),(26,'6/17','2017-01-01','Кассовый план на 1 полугодие 2017',4,10,3,'2017-03-31',2,'Кассовый план на 1 полугодие 2017.pdf',''),(27,'7/17','2017-04-01','Кассовый план на 2 полугодие 2017',4,2,3,'2017-06-30',2,NULL,''),(28,'8/17','2017-02-23','Кредитная заявка Беларусбанк',11,12,1,'2017-03-02',2,NULL,NULL),(29,'9/17','2017-02-25','Кредитная заявка МТБанк',11,11,1,'2017-02-26',2,'Кредитная заявка МТБанк.pdf',''),(30,'10/17','2017-02-16','Кредитная заявка БПС-Сбербанк',11,13,1,'2017-02-23',2,NULL,''),(31,'11/16','2016-12-14','План капитального ремонта здания ул.Филимонова, 53',8,5,3,'2017-02-05',2,NULL,''),(32,'6/16','2016-11-01','План

Приложение В (продолжение)

модернизации VoIP-телефонии офиса',8,10,3,'2017-03-16',2,'План модернизации VoIP-телефонии офиса.pdf',''),(33,'7/16','2016-09-13','Строительство торгового павильона',7,11,3,'2017-06-24',2,NULL,''),(34,'12/17','2017-02-22','Выплаты из прибыли 2016 года',9,7,2,'2017-04-07',2,NULL,''),(36,'7/16','2016-11-01','Разработка Положения о закупках',2,6,1,'2017-04-01',2,NULL,''),(37,'13/16','2016-09-06','Разработка Положения о премировании',2,3,5,'2017-04-13',2,NULL,''),(38,'14/16','2016-09-20','Разработка Учетной политики по налоговому учету на 2017 год',1,14,6,'2016-12-31',2,NULL,''),(39,'7/15','2015-08-11','Разработка Учетной политики по налоговому учету на 2016 год',1,6,6,'2016-01-01',1,NULL,''),(40,'15/17','2017-02-01','Кредитная заявка Беларусбанк',11,4,1,'2017-02-11',2,NULL,''),(41,'15/46','2017-02-20','План распределения балансовой прибыли на 2 кв. 2017 года',9,1,2,'2017-03-15',2,NULL,''),(42,'15/47','2017-02-01','Кредитная заявка Беларусбанк (в евро)',11,4,1,'2017-03-11',2,NULL,NULL),(43,'15/48','2017-02-26','Кредитная заявка Беларусбанк (в долларах США)',11,9,1,'2017-03-17',2,NULL,NULL),(44,'15/79','2017-02-24','Кредитная заявка МТБанк (в евро)',11,4,1,'2017-03-22',2,NULL,NULL),(45,'15/88','2017-02-25','Кредитная заявка БПС-Сбербанк (в евро)',11,9,1,'2017-03-22',2,NULL,NULL),(46,'16/106','2017-02-01','План капитального ремонта складского помещения (филиал 1)',8,1,3,'2017-03-15',2,'План капитального ремонта складского помещения (филиал 1).pdf',NULL),(47,'16/106','2017-02-11','План капитального ремонта складского помещения (филиал 2)',8,17,3,'2017-02-27',2,NULL,NULL),(48,'16/106','2017-02-17','План капитального ремонта складского помещения (филиал 3)',8,14,3,'2017-03-02',2,NULL,NULL),(49,'16/106','2017-02-05','План капитального ремонта складского помещения (филиал 4)',8,16,3,'2017-03-10',2,NULL,NULL),(50,'50/81','2017-02-11','План реализации продукции (позиция 006-01) на 2 кв. 2017 г.',5,2,4,'2017-03-03',2,'План реализации продукции (позиция 006-01) на 2 кв. 2017 г..pdf',NULL),(51,'50/82','2017-02-12','План реализации продукции (позиция 006-02) на 2 кв. 2017 г.',5,3,4,'2017-03-06',2,NULL,NULL),(52,'50/83','2017-02-13','План реализации продукции (позиция 006-03) на 2 кв. 2017 г.',5,5,4,'2017-03-14',2,NULL,NULL),(53,'50/84','2017-02-14','План реализации продукции (позиция 006-04) на 2 кв. 2017 г.',5,6,4,'2017-03-06',2,NULL,NULL),(54,'50/85','2017-02-15','План реализации продукции (позиция 006-05) на 2 кв. 2017 г.',5,18,4,'2017-03-03',2,NULL,NULL),(55,'50/93','2017-02-16','План реализации продукции (позиция 006-06) на 2 кв. 2017 г.',5,16,4,'2017-03-08',2,NULL,NULL),(56,'50/94','2017-02-17','План реализации продукции (позиция 006-07) на 2 кв. 2017 г.',5,12,4,'2017-03-04',2,NULL,NULL),(57,'50/96','2017-02-18','План реализации продукции (позиция 006-08) на 2 кв. 2017 г.',5,2,4,'2017-03-11',2,NULL,NULL),(58,'50/99','2017-02-19','План реализации продукции (позиция 006-09) на 2 кв. 2017 г.',5,3,4,'2017-03-22',2,NULL,NULL),(59,'17/105','2017-02-03','План капитального ремонта заготовительного цеха (филиал 7)',8,1,3,'2017-04-17',2,NULL,''),(60,'17/106','2017-02-05','План капитального ремонта производственного цеха (филиал 2)',8,2,3,'2017-04-15',2,NULL,NULL),(61,'17/13','2017-02-18','План капитального ремонта производственного цеха (филиал 3)',8,13,3,'2017-04-18',2,NULL,NULL),(62,'12/98','2017-02-10','План капитального ремонта производственного цеха (филиал 4)',8,1,3,'2017-04-11',2,NULL,NULL),(63,'15/87','2017-02-10','План капитального ремонта производственного цеха (филиал 5)',8,18,3,'2017-04-10',2,NULL,NULL);

UNLOCK TABLES;

--

-- Table structure for table `initiator`

--

DROP TABLE IF EXISTS `initiator`;

CREATE TABLE `initiator` (

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

 `name` varchar(45) NOT NULL,

 PRIMARY KEY (`id`),

 UNIQUE KEY `id\_UNIQUE` (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=7 DEFAULT CHARSET=utf8;

--

-- data for table `initiator`

--

LOCK TABLES `initiator` WRITE;

INSERT INTO `initiator` VALUES (1,'Финансовый отдел'),(2,'Бухгалтерия'),(3,'Планово-экономический отдел'),(4,'Отдел сбыта'),(5,'Отдел труда и зарплаты'),(6,'Отдел налогообложения');

Приложение В (продолжение)

UNLOCK TABLES;

--

-- Table structure for table `performer`

--

DROP TABLE IF EXISTS `performer`;

CREATE TABLE `performer` (

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

 `name` varchar(45) NOT NULL,

 `post` varchar(100) DEFAULT NULL,

 `phone\_in` varchar(10) DEFAULT NULL,

 `phone\_out` varchar(20) DEFAULT NULL,

 `nickname` varchar(20) DEFAULT NULL,

 PRIMARY KEY (`id`),

 UNIQUE KEY `id\_UNIQUE` (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=19 DEFAULT CHARSET=utf8;

--

-- data for table `performer`

--

LOCK TABLES `performer` WRITE;

INSERT INTO `performer` VALUES (1,'Иванова А.А.','Заместитель начальника отдела','102','(017) 215-17-89',NULL),(2,'Синицын А.Ю.','Экономист 2 категории','103','(017) 215-17-

89',NULL),(3,'Павлова А.Н.','Экономист 1 категории','104','(017) 215-17-96',NULL),(4,'Варламов Н.Е.','Специалист','105','(017) 215-17-98',NULL),(5,'Постоловский

А.Б.','Экономист 2 категории','106','(017) 215-18-15',NULL),(6,'Карасев Е.Ю.','Начальник отдела','101','(017) 215-18-20',NULL),(7,'Филлимонов К.Е.','Финансовый

аналитик','107','(017) 215-18-25',NULL),(8,'Беляева О.И.','Ведущий специалист','108','(017) 215-18-27',NULL),(9,'Никифоров А.Ю.','Экономист 2 категории','109','(017) 215-18-

26',NULL),(10,'Горький С.А.','Экономист 1 категории','110','(017) 215-18-27',NULL),(11,'Пилюлин П.Р.','Финансовый аналитик','111','(017) 215-18-88',NULL),(12,'Шохин

Е.К.','Специалист','112','(017) 215-18-36',NULL),(13,'Туров М.С.','Экономист 1 категории','113','(017) 215-18-64',NULL),(14,'Александрова И.И.','Финансовый

аналитик','114','(017) 215-18-17',NULL),(15,'Болтрукевич В.В.','Ведущий специалист','115','(017) 215-18-74',NULL),(16,'Сошников А.А.','Экономист 2 категории','116','(017)

215-18-66',NULL),(17,'Панкратов П.П.','Ведущий специалист','117','(017) 215-18-78',NULL),(18,'Прохоров Д.Л.','Экономист 1 категории','118','(017) 215-18-33',NULL);

UNLOCK TABLES;

--

-- Table structure for table `status`

--

DROP TABLE IF EXISTS `status`;

CREATE TABLE `status` (

 `id` int(10) unsigned NOT NULL AUTO\_INCREMENT,

 `name` varchar(45) NOT NULL,

 PRIMARY KEY (`id`),

 UNIQUE KEY `status\_id\_UNIQUE` (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=6 DEFAULT CHARSET=utf8;

--

-- data for table `status`

--

LOCK TABLES `status` WRITE;

INSERT INTO `status` VALUES (1,'Обработан/выполнен'),(2,'Принят к обработке'),(3,'Обработка отменена'),(4,'Не требует обработки'),(5,'Выведен из документооборота');

Приложение В (продолжение)

UNLOCK TABLES;

--

-- Table structure for table `type`

--

DROP TABLE IF EXISTS `type`;

CREATE TABLE `type` (

 `id` int(10) unsigned NOT NULL AUTO\_INCREMENT,

 `name` varchar(80) NOT NULL,

 PRIMARY KEY (`id`),

 UNIQUE KEY `type\_id\_UNIQUE` (`id`),

 UNIQUE KEY `type\_name\_UNIQUE` (`name`)

) ENGINE=InnoDB AUTO\_INCREMENT=12 DEFAULT CHARSET=utf8;

--

-- Dumping data for table `type`

--

LOCK TABLES `type` WRITE;

INSERT INTO `type` VALUES (3,'График'),(4,'Кассовый план'),(11,'Кредитная заявка'),(6,'План амортизационных отчислений'),(10,'План по налогу \n\nс оборота'),(9,'План

распределения балансовой прибыли'),(5,'План реализации продукции'),(8,'План финансирования капитального ремонта основных фондов'),(7,'План финансирования капитальных

вложений'),(2,'Положение'),(1,'Проект документа');

UNLOCK TABLES;

-- FK

ALTER TABLE `fin\_dept\_docs`.`document`

ADD CONSTRAINT `FK\_initiator`

 FOREIGN KEY (`initiator\_id`)

 REFERENCES `fin\_dept\_docs`.`initiator` (`id`)

 ON DELETE NO ACTION

 ON UPDATE NO ACTION,

ADD CONSTRAINT `FK\_perfomer`

 FOREIGN KEY (`performer\_id`)

 REFERENCES `fin\_dept\_docs`.`performer` (`id`)

 ON DELETE NO ACTION

 ON UPDATE NO ACTION,

ADD CONSTRAINT `FK\_status`

 FOREIGN KEY (`status\_id`)

 REFERENCES `fin\_dept\_docs`.`status` (`id`)

 ON DELETE NO ACTION

 ON UPDATE NO ACTION,

ADD CONSTRAINT `FK\_type`

 FOREIGN KEY (`type\_id`)

 REFERENCES `fin\_dept\_docs`.`type` (`id`)

 ON DELETE NO ACTION

 ON UPDATE NO ACTION;

**Приложение Г (обязательное)**

**Листинг кода разрабатываемого приложения**

***DocumentDAO.java***

package DAO;

import java.sql.SQLException;

import java.util.Collection;

import models.Document;

public interface DocumentDAO {

 public void addDocument(Document doc) throws SQLException ;

 public void newDocument(String input\_number, String reg\_date, String name,

 String type\_id, String initiator\_id, String status\_id, String deadline,

 String performer\_id, String description) throws SQLException ;

 public void updateDocument(int id, Document doc) throws SQLException;

 public void updateDocument(int id, String input\_number, String reg\_date, String name,

 String type\_id, String initiator\_id, String status\_id, String deadline, String performer\_id,

 String file, String description);

 public Document getLastDocument() throws SQLException;

 public Document getDocumentById(int id) throws SQLException;

 public void deleteDocumentById(int id) throws SQLException;

 public Collection getDocumentsByPerformer(int performer\_id) throws SQLException;

 public Collection getAllDocuments() throws SQLException;

 public Collection getDocumentsExpiringNextDays(int days) throws SQLException;

 public Collection getDocumentsExpired() throws SQLException;

 public Collection getDocumentsStatistics() throws SQLException;

 public Collection getDocumentsByType(int id) throws SQLException;

 public Collection getDocumentsByInitiator(int id) throws SQLException;

 public Collection getDocumentsByStatus(int id) throws SQLException;

}

***DocumentDAOImpl.java***

package DAO;

import java.sql.DriverManager;

import java.sql.SQLException;

import java.text.DateFormat;

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.ArrayList;

import java.util.Collection;

import java.util.Date;

import java.util.List;

import java.util.Locale;

import java.util.logging.Level;

import java.util.logging.Logger;

import models.Document;

import models.Initiator;

import models.Type;

import models.Performer;

import models.Status;

import org.hibernate.Criteria;

import org.hibernate.HibernateException;

Приложение Г (продолжение)

import org.hibernate.Query;

import org.hibernate.SQLQuery;

import org.hibernate.Session;

import util.HibernateUtil;

public class DocumentDAOImpl implements DocumentDAO {

 @Override

 public void addDocument(Document doc) throws SQLException {

 Session session = null;

 try {

 session = HibernateUtil.getSessionFactory().getCurrentSession();

 session.beginTransaction();

 session.save(doc);

 session.getTransaction().commit();

 } catch (HibernateException e) {

 DriverManager.println("Ошибка при вставке");

 } finally {

 if (session != null && session.isOpen()) {

 session.close();

 }

 }

 }

 @Override

 public void updateDocument(int id, Document doc) throws SQLException {

 Session session = null;

 try {

 session = HibernateUtil.getSessionFactory().getCurrentSession();

 session.beginTransaction();

 String hql = "update fin\_dept\_docs.document set"

 + " document.input\_number = :input\_number, "

 + " document.reg\_date = :reg\_date, "

 + " document.name = :name, "

 + " document.type\_id = :type\_id, "

 + " document.initiator\_id = :initiator\_id, "

 + " document.status\_id = :status\_id, "

 + " document.deadline = :deadline, "

 + " document.performer\_id = :performer\_id, "

 + " document.file = :file, "

 + " document.description = :description "

 + "WHERE id = :id";

 Query query = (Query) session.createSQLQuery(hql);

 query.setParameter("input\_number", doc.getInputNumber());

 query.setParameter("reg\_date", doc.getRegDate());

 query.setParameter("name", doc.getName());

 query.setParameter("type\_id", doc.getType().getId());

 query.setParameter("initiator\_id", doc.getInitiator().getId());

 query.setParameter("status\_id", doc.getStatus().getId());

 query.setParameter("deadline", doc.getDeadline());

 query.setParameter("performer\_id", doc.getPerformer().getId());

 query.setParameter("file", doc.getFile());

 query.setParameter("description", doc.getDescription());

 query.setParameter("id", id);

Приложение Г (продолжение)

 query.executeUpdate();

 session.getTransaction().commit();

 } catch (HibernateException e) {

 throw e;

 } finally {

 if (session != null && session.isOpen()) {

 session.close();

 }

 }

 }

 @Override

 public void updateDocument(int id, String input\_number, String reg\_date, String name,

 String type\_id, String initiator\_id, String status\_id, String deadline, String performer\_id,

 String file, String description) {

 Session session = null;

 try {

 session = HibernateUtil.getSessionFactory().openSession();

 session.beginTransaction();

 String hql = "update document set"

 + " input\_number = :input\_number, "

 + " reg\_date = :reg\_date, "

 + " name = :name, "

 + " type\_id = :type\_id, "

 + " initiator\_id = :initiator\_id, "

 + " status\_id = :status\_id, "

 + " deadline = :deadline, "

 + " performer\_id = :performer\_id, ";

 if (file != "") {

 hql += " file = :file, ";

 }

 hql += " description = :description "

 + "WHERE id = :id";

 Query query = (Query) session.createSQLQuery(hql);

 query.setParameter("input\_number", input\_number);

 query.setParameter("reg\_date", reg\_date);

 query.setParameter("name", name);

 query.setParameter("type\_id", type\_id);

 query.setParameter("initiator\_id", initiator\_id);

 query.setParameter("status\_id", status\_id);

 query.setParameter("deadline", deadline);

 query.setParameter("performer\_id", performer\_id);

 if (file != "") {

 query.setParameter("file", file);

 }

 query.setParameter("description", description);

 query.setParameter("id", id);

 query.executeUpdate();

 session.getTransaction().commit();

 } catch (HibernateException e) {

 throw e;

Приложение Г (продолжение)

 } finally {

 if (session != null && session.isOpen()) {

 session.close();

 }

 }

 }

 @Override

 public Collection getDocumentsByPerformer(int performer\_id) throws SQLException {

 Session session = null;

 List documents = new ArrayList<>();

 try {

 session = HibernateUtil.getSessionFactory().openSession();

 String sql = "SELECT \* FROM document WHERE performer\_id = " + performer\_id + " ORDER BY deadline ASC";

 SQLQuery query = session.createSQLQuery(sql);

 query.setResultTransformer(Criteria.ALIAS\_TO\_ENTITY\_MAP);

 documents = query.list();

 } catch (Exception e) {

 } finally {

 if (session != null && session.isOpen()) {

 session.close();

 }

 }

 return documents;

 }

 @Override

 public Document getDocumentById(int id) throws SQLException {

 Session session = null;

 Document doc = new Document();

 try {

 session = HibernateUtil.getSessionFactory().openSession();

 doc = (Document) session.load(Document.class, id);

 } finally {

 if (session != null && session.isOpen()) {

 session.close();

 }

 }

 return doc;

 }

 @Override

 public Collection getAllDocuments() throws SQLException {

 Session session = null;

 List<Document> documents = new ArrayList<>();

 try {

 session = HibernateUtil.getSessionFactory().openSession();

 documents = session.createCriteria(Document.class).list();

 } catch (Exception e) {

 throw e;

Приложение Г (продолжение)

 } finally {

 if (session != null && session.isOpen()) {

 session.close();

 }

 }

 return documents;

 }

 @Override

 public void newDocument(String input\_number, String reg\_date, String name,

 String type\_id, String initiator\_id, String status\_id, String deadline,

 String performer\_id, String description) throws SQLException {

 Session session = null;

 try {

 session = HibernateUtil.getSessionFactory().getCurrentSession();

 session.beginTransaction();

 Document doc = new Document();

 DateFormat format = new SimpleDateFormat("yyyy-MM-dd", Locale.ENGLISH);

 doc.setInputNumber(input\_number);

 Date regDate = format.parse(reg\_date);

 doc.setRegDate(regDate);

 doc.setName(name);

 Type type = new Type();

 type.setId(Integer.parseInt(type\_id));

 doc.setType(type);

 Initiator initiator = new Initiator();

 initiator.setId(Integer.parseInt(initiator\_id));

 doc.setInitiator(initiator);

 Status status = new Status();

 status.setId(Integer.parseInt(status\_id));

 doc.setStatus(status);

 Date deadlineDate = format.parse(deadline);

 doc.setDeadline(deadlineDate);

 Performer performer = new Performer();

 performer.setId(Integer.parseInt(performer\_id));

 doc.setPerformer(performer);

 doc.setDescription(description);

 doc.setFile(null);

 session.save(doc);

 session.getTransaction().commit();

 } catch (HibernateException e) {

 throw e;

 } catch (ParseException ex) {

 Logger.getLogger(DocumentDAOImpl.class.getName()).log(Level.SEVERE, null, ex);

Приложение Г (продолжение)

 } finally {

 if (session != null && session.isOpen()) {

 session.close();

 }

 }

 }

 @Override

 public Document getLastDocument() throws SQLException {

 Session session = null;

 Document lastDoc = new Document();

 List<Document> documents = new ArrayList<>();

 try {

 session = HibernateUtil.getSessionFactory().openSession();

 documents = session.createCriteria(Document.class).list();

 lastDoc = documents.get(documents.size() - 1);

 } catch (HibernateException e) {

 throw e;

 } finally {

 if (session != null && session.isOpen()) {

 session.close();

 }

 }

 return lastDoc;

 }

 @Override

 public void deleteDocumentById(int id) throws SQLException {

 Session session = null;

 try {

 session = HibernateUtil.getSessionFactory().openSession();

 session.beginTransaction();

 session.delete(getDocumentById(id));

 session.getTransaction().commit();

 } finally {

 if (session != null && session.isOpen()) {

 session.close();

 }

 }

 }

 @Override

 public Collection getDocumentsExpiringNextDays(int days) throws SQLException {

 Session session = null;

 List documents = new ArrayList<>();

 try {

 session = HibernateUtil.getSessionFactory().openSession();

 String sql = "SELECT \n"

 + "document.id, \n"

 + "document.input\_number, \n"

 + "document.reg\_date,\n"

 + "document.`name`,\n"

 + "performer.name AS performerName,\n"

 + "initiator.name AS initiatorName,\n"

 + "deadline,\n"

Приложение Г (продолжение)

 + "status.name AS statusName,\n"

 + "DATEDIFF(deadline,CURDATE()) AS expDays"

 + "\n"

 + "FROM fin\_dept\_docs.document \n"

 + "\n"

 + "JOIN fin\_dept\_docs.initiator ON document.initiator\_id = initiator.id\n"

 + "JOIN fin\_dept\_docs.performer ON document.performer\_id = performer.id\n"

 + "JOIN fin\_dept\_docs.status ON document.status\_id = status.id\n"

 + "\n"

 + "WHERE deadline >= curdate() AND DATEDIFF(deadline,CURDATE()) <=" + days + "\n"

 + "\n"

 + "ORDER BY deadline ASC";

 SQLQuery query = session.createSQLQuery(sql);

 query.setResultTransformer(Criteria.ALIAS\_TO\_ENTITY\_MAP);

 documents = query.list();

 } catch (Exception e) {

 } finally {

 if (session != null && session.isOpen()) {

 session.close();

 }

 }

 return documents;

 }

 @Override

 public Collection getDocumentsExpired() throws SQLException {

 Session session = null;

 List documents = new ArrayList<>();

 try {

 session = HibernateUtil.getSessionFactory().openSession();

 String sql = "SELECT \n"

 + "document.id, \n"

 + "document.input\_number, \n"

 + "document.reg\_date,\n"

 + "document.`name`,\n"

 + "performer.name AS performerName,\n"

 + "initiator.name AS initiatorName,\n"

 + "deadline,\n"

 + "status.name AS statusName,\n"

 + "DATEDIFF(CURDATE(), deadline) AS expDays"

 + "\n"

 + "FROM fin\_dept\_docs.document \n"

 + "\n"

 + "JOIN fin\_dept\_docs.initiator ON document.initiator\_id = initiator.id\n"

 + "JOIN fin\_dept\_docs.performer ON document.performer\_id = performer.id\n"

 + "JOIN fin\_dept\_docs.status ON document.status\_id = status.id\n"

 + "\n"

 + "WHERE deadline < curdate() AND document.status\_id = 2\n"

 + "\n"

 + "ORDER BY deadline ASC";

 SQLQuery query = session.createSQLQuery(sql);

 query.setResultTransformer(Criteria.ALIAS\_TO\_ENTITY\_MAP);

Приложение Г (продолжение)

 documents = query.list();

 } catch (Exception e) {

 } finally {

 if (session != null && session.isOpen()) {

 session.close();

 }

 }

 return documents;

 }

 @Override

 public Collection getDocumentsStatistics() throws SQLException {

 Session session = null;

 List documents = new ArrayList<>();

 try {

 session = HibernateUtil.getSessionFactory().openSession();

 String sql = "SELECT \n"

 + "`document`.performer\_id,\n"

 + "`performer`.name AS performer,\n"

 + "(\n"

 + " SELECT count(`document`.id)\n"

 + " FROM `fin\_dept\_docs`.document\n"

 + " WHERE `document`.status\_id = 2 AND `performer`.id = `document`.performer\_id\n"

 + " ) AS documents,\n"

 + "(\n"

 + " SELECT count(`document`.id)\n"

 + " FROM `fin\_dept\_docs`.document\n"

 + " WHERE `document`.deadline < curdate() AND `document`.status\_id = 2 AND `performer`.id = `document`.performer\_id\n"

 + " ) AS exp\n"

 + " \n"

 + "FROM `fin\_dept\_docs`.document\n"

 + "JOIN `fin\_dept\_docs`.performer ON `performer`.id = `document`.performer\_id\n"

 + "\n"

 + "group by performer\n"

 + "order by `performer`.id";

 SQLQuery query = session.createSQLQuery(sql);

 query.setResultTransformer(Criteria.ALIAS\_TO\_ENTITY\_MAP);

 documents = query.list();

 } catch (Exception e) {

 } finally {

 if (session != null && session.isOpen()) {

 session.close();

 }

 }

 return documents;

 }

 @Override

Приложение Г (продолжение)

 public Collection getDocumentsByType(int id) throws SQLException {

 Session session = null;

 List<Document> tempDocuments = new ArrayList<>();

 List<Document> documents = new ArrayList<>();

 try {

 session = HibernateUtil.getSessionFactory().openSession();

 tempDocuments = session.createCriteria(Document.class).list();

 if (!tempDocuments.isEmpty()) {

 for (int i = 0; i < tempDocuments.size(); i++) {

 if (tempDocuments.get(i).getType().getId() == id) {

 documents.add(tempDocuments.get(i));

 }

 }

 }

 } catch (Exception e) {

 throw e;

 } finally {

 if (session != null && session.isOpen()) {

 session.close();

 }

 }

 return documents;

 }

 @Override

 public Collection getDocumentsByInitiator(int id) throws SQLException {

 Session session = null;

 List<Document> tempDocuments = new ArrayList<>();

 List<Document> documents = new ArrayList<>();

 try {

 session = HibernateUtil.getSessionFactory().openSession();

 tempDocuments = session.createCriteria(Document.class).list();

 if (!tempDocuments.isEmpty()) {

 for (int i = 0; i < tempDocuments.size(); i++) {

 if (tempDocuments.get(i).getInitiator().getId() == id) {

 documents.add(tempDocuments.get(i));

 }

 }

 }

 } catch (Exception e) {

 throw e;

 } finally {

 if (session != null && session.isOpen()) {

 session.close();

 }

 }

 return documents;

 }

 @Override

 public Collection getDocumentsByStatus(int id) throws SQLException {

 Session session = null;

 List<Document> tempDocuments = new ArrayList<>();

Приложение Г (продолжение)

 List<Document> documents = new ArrayList<>();

 try {

 session = HibernateUtil.getSessionFactory().openSession();

 tempDocuments = session.createCriteria(Document.class).list();

 if (!tempDocuments.isEmpty()) {

 for (int i = 0; i < tempDocuments.size(); i++) {

 if (tempDocuments.get(i).getStatus().getId() == id) {

 documents.add(tempDocuments.get(i));

 }

 }

 }

 } catch (Exception e) {

 throw e;

 } finally {

 if (session != null && session.isOpen()) {

 session.close();

 }

 }

 return documents;

 }

}

***InitiatorDAO.java***

package DAO;

import java.sql.SQLException;

import java.util.Collection;

public interface InitiatorDAO {

 public Collection getAllInitiators() throws SQLException;

}

***InitiatorDAOImpl.java***

package DAO;

import java.sql.SQLException;

import java.util.ArrayList;

import java.util.Collection;

import java.util.List;

import models.Initiator;

import org.hibernate.Session;

import util.HibernateUtil;

public class InitiatorDAOImpl implements InitiatorDAO {

 @Override

 public Collection getAllInitiators() throws SQLException {

 Session session = null;

 List<String> initiators = new ArrayList<>();

 try {

Приложение Г (продолжение)

 session = HibernateUtil.getSessionFactory().openSession();

 initiators = session.createCriteria(Initiator.class).list();

 } catch (Exception e) {

 throw e;

 } finally {

 if (session != null && session.isOpen()) {

 session.close();

 }

 }

 return initiators;

 }

}

***PerformerDAO.java***

package DAO;

import java.sql.SQLException;

import java.util.Collection;

import models.Performer;

public interface PerformerDAO {

 public Collection getAllPerformers() throws SQLException;

 public Performer getPerformerById(int id) throws SQLException;

}

***PerformerDAOImpl.java***

package DAO;

import java.sql.SQLException;

import java.util.ArrayList;

import java.util.List;

import models.Performer;

import org.hibernate.Session;

import util.HibernateUtil;

public class PerformerDAOImpl implements PerformerDAO {

 @Override

 public List<Performer> getAllPerformers() throws SQLException {

 Session session = null;

 List<Performer> performers = new ArrayList<>();

 try {

 session = HibernateUtil.getSessionFactory().openSession();

 performers = session.createCriteria(Performer.class).list();

 } catch (Exception e) {

 throw e;

 } finally {

 if (session != null && session.isOpen()) {

Приложение Г (продолжение)

 session.close();

 }

 }

 return performers;

 }

 @Override

 public Performer getPerformerById(int id) throws SQLException {

 Session session = null;

 Performer performer = new Performer();

 try {

 session = HibernateUtil.getSessionFactory().openSession();

 performer = (Performer) session.get(Performer.class, id);

 } finally {

 if (session != null && session.isOpen()) {

 session.close();

 }

 }

 return performer;

 }

}

***StatusDAO.java***

package DAO;

import java.sql.SQLException;

import java.util.Collection;

public interface StatusDAO {

 public Collection getAllStatuses() throws SQLException;

}

***StatusDAOImpl.java***

package DAO;

import java.sql.SQLException;

import java.util.ArrayList;

import java.util.Collection;

import java.util.List;

import models.Status;

import org.hibernate.Session;

import util.HibernateUtil;

public class StatusDAOImpl implements StatusDAO {

 @Override

 public Collection getAllStatuses() throws SQLException {

 Session session = null;

 List<String> statuses = new ArrayList<>();

 try {

 session = HibernateUtil.getSessionFactory().openSession();

 statuses = session.createCriteria(Status.class).list();

 } catch (Exception e) {

Приложение Г (продолжение)

 throw e;

 } finally {

 if (session != null && session.isOpen()) {

 session.close();

 }

 }

 return statuses;

 }

}

***TypeDAO.java***

package DAO;

import java.sql.SQLException;

import java.util.Collection;

public interface TypeDAO {

 public Collection getAllTypes() throws SQLException;

}

***TypeDAOImpl.java***

package DAO;

import java.sql.SQLException;

import java.util.ArrayList;

import java.util.Collection;

import java.util.List;

import models.Type;

import org.hibernate.Session;

import util.HibernateUtil;

public class TypeDAOImpl implements TypeDAO {

 @Override

 public Collection getAllTypes() throws SQLException {

 Session session = null;

 List<String> types = new ArrayList<>();

 try {

 session = HibernateUtil.getSessionFactory().openSession();

 types = session.createCriteria(Type.class).list();

 } catch (Exception e) {

 throw e;

 } finally {

 if (session != null && session.isOpen()) {

 session.close();

 }

 }

 return types;

 }

}

Приложение Г (продолжение)

***SessionFactory.java***

package DAO;

public class SessionFactory {

 private static DocumentDAO documentDAO = null;

 private static TypeDAO typeDAO = null;

 private static StatusDAO statusDAO = null;

 private static PerformerDAO performerDAO = null;

 private static InitiatorDAO initiatorDAO = null;

 private static SessionFactory instance = null;

 public static synchronized SessionFactory getInstance() {

 if (instance == null) {

 instance = new SessionFactory();

 }

 return instance;

 }

 public DocumentDAO getDocumentDAO() {

 if (documentDAO == null) {

 documentDAO = new DocumentDAOImpl();

 }

 return documentDAO;

 }

 public TypeDAO getTypeDAO() {

 if (typeDAO == null) {

 typeDAO = new TypeDAOImpl();

 }

 return typeDAO;

 }

 public StatusDAO getStatusDAO() {

 if (statusDAO == null) {

 statusDAO = new StatusDAOImpl();

 }

 return statusDAO;

 }

 public PerformerDAO getPerformerDAO() {

 if (performerDAO == null) {

 performerDAO = new PerformerDAOImpl();

 }

 return performerDAO;

 }

 public InitiatorDAO getInitiatorDAO() {

 if (initiatorDAO == null) {

 initiatorDAO = new InitiatorDAOImpl();

 }

 return initiatorDAO;

 }

}

Приложение Г (продолжение)

***addDocument.java***

package servlets;

import DAO.SessionFactory;

import java.io.IOException;

import java.io.PrintWriter;

import java.sql.SQLException;

import java.util.List;

import java.util.logging.Level;

import java.util.logging.Logger;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import models.Document;

public class addDocument extends HttpServlet {

 protected void processRequest(HttpServletRequest request, HttpServletResponse response)

 throws ServletException, IOException {

 response.setContentType("text/html;charset=UTF-8");

 try (PrintWriter out = response.getWriter()) {

 /\* TODO output your page here. You may use following sample code. \*/

 out.println("<!DOCTYPE html>");

 out.println("<html>");

 out.println("<head>");

 out.println("<title>Servlet addDocument</title>");

 out.println("</head>");

 out.println("<body>");

 out.println("<h1>Servlet addDocument at " + request.getContextPath() + "</h1>");

 out.println("</body>");

 out.println("</html>");

 }

 }

 @Override

 protected void doGet(HttpServletRequest request, HttpServletResponse response)

 throws ServletException, IOException {

 request.setCharacterEncoding("UTF-8");

 try {

 List<String> allTypes = (List<String>) SessionFactory.getInstance().getTypeDAO().getAllTypes();

 request.setAttribute("allTypes", allTypes);

 List<String> allStatuses = (List<String>) SessionFactory.getInstance().getStatusDAO().getAllStatuses();

 request.setAttribute("allStatuses", allStatuses);

 List<String> allPerformers = (List<String>) SessionFactory.getInstance().getPerformerDAO().getAllPerformers();

 request.setAttribute("allPerformers", allPerformers);

 List<String> allInitiators = (List<String>) SessionFactory.getInstance().getInitiatorDAO().getAllInitiators();

 request.setAttribute("allInitiators", allInitiators);

 Document newDoc = new Document();

 request.setAttribute("newDoc", newDoc);

 request.getRequestDispatcher("/newdoc.jsp").forward(request, response);

Приложение Г (продолжение)

 } catch (SQLException ex) {

 Logger.getLogger(showAllDocs.class.getName()).log(Level.SEVERE, null, ex);

 }

 }

 @Override

 protected void doPost(HttpServletRequest request, HttpServletResponse response)

 throws ServletException, IOException {

 request.setCharacterEncoding("UTF-8");

 try {

 String input\_number = request.getParameter("input\_number");

 String reg\_date = request.getParameter("reg\_date");

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

 String type\_id = request.getParameter("type");

 String initiator\_id = request.getParameter("initiator");

 String status\_id = request.getParameter("status");

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

 String performer\_id = request.getParameter("performer");

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

 SessionFactory.getInstance().getDocumentDAO().newDocument(input\_number, reg\_date, name, type\_id, initiator\_id, status\_id, deadline, performer\_id, desc);

 List<Document> allDocuments = null;

 allDocuments = (List<Document>) SessionFactory.getInstance().getDocumentDAO().getAllDocuments();

 request.setAttribute("allDocuments", allDocuments);

 request.getRequestDispatcher("/alldocs.jsp").forward(request, response);

 } catch (SQLException ex) {

 Logger.getLogger(addDocument.class.getName()).log(Level.SEVERE, null, ex);

 request.getRequestDispatcher("/errordocs.jsp").forward(request, response);

 }

 }

 @Override

 public String getServletInfo() {

 return "Short description";

 }// </editor-fold>

}

***deleteDocument.java***

package servlets;

import DAO.SessionFactory;

import java.io.IOException;

import java.io.PrintWriter;

import java.sql.SQLException;

import java.util.List;

import java.util.logging.Level;

import java.util.logging.Logger;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

Приложение Г (продолжение)

import models.Document;

public class deleteDocument extends HttpServlet {

 protected void processRequest(HttpServletRequest request, HttpServletResponse response)

 throws ServletException, IOException {

 response.setContentType("text/html;charset=UTF-8");

 try (PrintWriter out = response.getWriter()) {

 /\* TODO output your page here. You may use following sample code. \*/

 out.println("<!DOCTYPE html>");

 out.println("<html>");

 out.println("<head>");

 out.println("<title>Servlet delDocument</title>");

 out.println("</head>");

 out.println("<body>");

 out.println("<h1>Servlet delDocument at " + request.getContextPath() + "</h1>");

 out.println("</body>");

 out.println("</html>");

 }

 }

 @Override

 protected void doGet(HttpServletRequest request, HttpServletResponse response)

 throws ServletException, IOException {

 try {

 int id = Integer.parseInt(request.getParameter("docid"));

 SessionFactory.getInstance().getDocumentDAO().deleteDocumentById(id);

 List<String> allTypes = (List<String>) SessionFactory.getInstance().getTypeDAO().getAllTypes();

 request.setAttribute("allTypes", allTypes);

 List<String> allStatuses = (List<String>) SessionFactory.getInstance().getStatusDAO().getAllStatuses();

 request.setAttribute("allStatuses", allStatuses);

 List<String> allPerformers = (List<String>) SessionFactory.getInstance().getPerformerDAO().getAllPerformers();

 request.setAttribute("allPerformers", allPerformers);

 List<String> allInitiators = (List<String>) SessionFactory.getInstance().getInitiatorDAO().getAllInitiators();

 request.setAttribute("allInitiators", allInitiators);

 List<Document> allDocuments = (List<Document>) SessionFactory.getInstance().getDocumentDAO().getAllDocuments();

 request.setAttribute("allDocuments", allDocuments);

 request.getRequestDispatcher("/alldocs.jsp").forward(request, response);

 } catch (SQLException ex) {

 Logger.getLogger(showAllDocs.class.getName()).log(Level.SEVERE, null, ex);

 request.getRequestDispatcher("/errordocs.jsp").forward(request, response);

 }

 }

 @Override

 protected void doPost(HttpServletRequest request, HttpServletResponse response)

 throws ServletException, IOException {

 processRequest(request, response);

 }

Приложение Г (продолжение)

 @Override

 public String getServletInfo() {

 return "Short description";

 }// </editor-fold>

}

***downloadFile.java***

package servlets;

import DAO.SessionFactory;

import java.io.File;

import java.io.FileInputStream;

import java.io.IOException;

import java.io.OutputStream;

import java.net.URLEncoder;

import java.sql.SQLException;

import java.util.logging.Level;

import java.util.logging.Logger;

import javax.servlet.ServletContext;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

public class downloadFile extends HttpServlet {

 private static final String SAVE\_DIR = "data\\files";

 @Override

 protected void doGet(HttpServletRequest request,

 HttpServletResponse response) throws ServletException, IOException {

 response.setCharacterEncoding("UTF-8");

 request.setCharacterEncoding("UTF-8");

 int docid = Integer.parseInt(request.getParameter("docid"));

 String appPath = getServletContext().getRealPath("");

 String savePath = appPath + File.separator + SAVE\_DIR;

 String docFileName = null;

 try {

 docFileName = SessionFactory.getInstance().getDocumentDAO().getDocumentById(docid).getFile();

 String filePath = savePath + File.separator + docFileName;

 if (docFileName == null) {

 request.getRequestDispatcher("/errordownload.jsp").forward(request, response);

 } else {

 File downloadFile = new File(filePath);

 String fileName = URLEncoder.encode(downloadFile.getName(), "UTF-8");

 OutputStream outStream;

 try (FileInputStream inStream = new FileInputStream(downloadFile)) {

Приложение Г (продолжение)

 String relativePath = getServletContext().getRealPath("");

 System.out.println("relativePath = " + relativePath);

 ServletContext context = getServletContext();

 String mimeType = context.getMimeType(filePath);

 if (mimeType == null) {

 mimeType = "application/octet-stream";

 }

 response.setContentType(mimeType);

 response.setContentLength((int) downloadFile.length());

 response.setHeader("Content-Disposition", "attachment;filename=" + fileName);

 outStream = response.getOutputStream();

 byte[] buffer = new byte[4096];

 int bytesRead = -1;

 while ((bytesRead = inStream.read(buffer)) != -1) {

 outStream.write(buffer, 0, bytesRead);

 }

 }

 outStream.close();

 }

 } catch (SQLException ex) {

 Logger.getLogger(downloadFile.class.getName()).log(Level.SEVERE, null, ex);

 }

 }

 @Override

 public String getServletInfo() {

 return "Short description";

 }// </editor-fold>

}

***editDocument.java***

package servlets;

import DAO.SessionFactory;

import java.io.IOException;

import java.sql.SQLException;

import java.util.List;

import java.util.logging.Level;

import java.util.logging.Logger;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import models.Document;

public class editDocument extends HttpServlet {

 @Override

Приложение Г (продолжение)

 protected void doGet(HttpServletRequest request, HttpServletResponse response)

 throws ServletException, IOException {

 try {

 int id = Integer.parseInt(request.getParameter("docid"));

 Document currentDoc = SessionFactory.getInstance().getDocumentDAO().getDocumentById(id);

 List<String> allTypes = (List<String>) SessionFactory.getInstance().getTypeDAO().getAllTypes();

 request.setAttribute("allTypes", allTypes);

 List<String> allStatuses = (List<String>) SessionFactory.getInstance().getStatusDAO().getAllStatuses();

 request.setAttribute("allStatuses", allStatuses);

 List<String> allPerformers = (List<String>) SessionFactory.getInstance().getPerformerDAO().getAllPerformers();

 request.setAttribute("allPerformers", allPerformers);

 List<String> allInitiators = (List<String>) SessionFactory.getInstance().getInitiatorDAO().getAllInitiators();

 request.setAttribute("allInitiators", allInitiators);

 if (currentDoc != null) {

 request.setAttribute("currentDoc", currentDoc);

 request.getRequestDispatcher("/editdocs.jsp").forward(request, response);

 } else {

 request.getRequestDispatcher("/errordocs.jsp").forward(request, response);

 }

 } catch (SQLException ex) {

 Logger.getLogger(showAllDocs.class.getName()).log(Level.SEVERE, null, ex);

 request.getRequestDispatcher("/errordocs.jsp").forward(request, response);

 }

 }

 @Override

 protected void doPost(HttpServletRequest request, HttpServletResponse response)

 throws ServletException, IOException {

 processRequest(request, response);

 }

 @Override

 public String getServletInfo() {

 return "Short description";

 }// </editor-fold>

 private void processRequest(HttpServletRequest request, HttpServletResponse response) {

 throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.

 }

}

***performers.java***

package servlets;

import DAO.SessionFactory;

import java.io.IOException;

import java.io.PrintWriter;

import java.sql.SQLException;

Приложение Г (продолжение)

import java.util.List;

import java.util.logging.Level;

import java.util.logging.Logger;

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 models.Document;

import models.Performer;

@WebServlet(name = "performers", urlPatterns = {"/performers"})

public class performers extends HttpServlet {

 protected void processRequest(HttpServletRequest request, HttpServletResponse response)

 throws ServletException, IOException {

 response.setContentType("text/html;charset=UTF-8");

 try (PrintWriter out = response.getWriter()) {

 /\* TODO output your page here. You may use following sample code. \*/

 out.println("<!DOCTYPE html>");

 out.println("<html>");

 out.println("<head>");

 out.println("<title>Servlet performers</title>");

 out.println("</head>");

 out.println("<body>");

 out.println("<h1>Servlet performers at " + request.getContextPath() + "</h1>");

 out.println("</body>");

 out.println("</html>");

 }

 }

 @Override

 protected void doGet(HttpServletRequest request, HttpServletResponse response)

 throws ServletException, IOException {

 try {

 int id = Integer.parseInt(request.getParameter("performerid"));

 if (id == 0) { // страница по умолчанию

 List<Performer> allPerformers = (List<Performer>) SessionFactory.getInstance().getPerformerDAO().getAllPerformers();

 request.setAttribute("allPerformers", allPerformers);

 request.getRequestDispatcher("/performers.jsp").forward(request, response);

 }

 if (id > 0) { // страница с выбранным исполнителем

 List<Performer> allPerformers = (List<Performer>) SessionFactory.getInstance().getPerformerDAO().getAllPerformers();

 request.setAttribute("allPerformers", allPerformers);

 Performer currentPerformer = (Performer) SessionFactory.getInstance().getPerformerDAO().getPerformerById(id);

 request.setAttribute("currentPerformer", currentPerformer);

 List<Document> performerDocs = (List<Document>) SessionFactory.getInstance().getDocumentDAO().getDocumentsByPerformer(id);

 request.setAttribute("performerDocs", performerDocs);

Приложение Г (продолжение)

 List<Document> performersStat = (List<Document>) SessionFactory.getInstance().getDocumentDAO().getDocumentsStatistics();

 request.setAttribute("performersStat", performersStat);

 request.getRequestDispatcher("/performers.jsp").forward(request, response);

 } else {

 request.getRequestDispatcher("/errordocs.jsp").forward(request, response);

 }

 } catch (SQLException ex) {

 Logger.getLogger(performers.class.getName()).log(Level.SEVERE, null, ex);

 }

 }

 @Override

 protected void doPost(HttpServletRequest request, HttpServletResponse response)

 throws ServletException, IOException {

 processRequest(request, response);

 }

 @Override

 public String getServletInfo() {

 return "Short description";

 }// </editor-fold>

}

***showAllDocs.java***

package servlets;

import DAO.SessionFactory;

import java.io.IOException;

import java.sql.SQLException;

import java.util.List;

import java.util.logging.Level;

import java.util.logging.Logger;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import models.Document;

public class showAllDocs extends HttpServlet {

 @Override

 protected void doGet(HttpServletRequest request, HttpServletResponse response)

 throws ServletException, IOException {

 try {

 List<Document> allDocuments = (List<Document>) SessionFactory.getInstance().getDocumentDAO().getAllDocuments();

 request.setAttribute("allDocuments", allDocuments);

 request.getRequestDispatcher("/alldocs.jsp").forward(request, response);

 } catch (SQLException ex) {

 Logger.getLogger(showAllDocs.class.getName()).log(Level.SEVERE, null, ex);

Приложение Г (продолжение)

 }

 }

 @Override

 protected void doPost(HttpServletRequest request, HttpServletResponse response)

 throws ServletException, IOException {

 //processRequest(request, response);

 }

 @Override

 public String getServletInfo() {

 return "Short description";

 }// </editor-fold>

}

***showDocsByInitiator.java***

package servlets;

import DAO.SessionFactory;

import java.io.IOException;

import java.sql.SQLException;

import java.util.List;

import java.util.logging.Level;

import java.util.logging.Logger;

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 models.Document;

@WebServlet(name = "showDocsByInitiator", urlPatterns = {"/showdocsbyinitiator"})

public class showDocsByInitiator extends HttpServlet {

 @Override

 protected void doGet(HttpServletRequest request, HttpServletResponse response)

 throws ServletException, IOException {

 try {

 int id = Integer.parseInt(request.getParameter("initiatorid"));

 List<Document> allDocuments = (List<Document>) SessionFactory.getInstance().getDocumentDAO().getDocumentsByInitiator(id);

 request.setAttribute("allDocuments", allDocuments);

 request.getRequestDispatcher("/alldocs.jsp").forward(request, response);

 } catch (SQLException ex) {

 Logger.getLogger(showAllDocs.class.getName()).log(Level.SEVERE, null, ex);

 }

 }

 @Override

 protected void doPost(HttpServletRequest request, HttpServletResponse response)

 throws ServletException, IOException {

 //processRequest(request, response);

 }

Приложение Г (продолжение)

 @Override

 public String getServletInfo() {

 return "Short description";

 }// </editor-fold>

}

***showDocsByStatus.java***

package servlets;

import DAO.SessionFactory;

import java.io.IOException;

import java.sql.SQLException;

import java.util.List;

import java.util.logging.Level;

import java.util.logging.Logger;

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 models.Document;

@WebServlet(name = "showDocsByStatus", urlPatterns = {"/showdocsbystatus"})

public class showDocsByStatus extends HttpServlet {

 @Override

 protected void doGet(HttpServletRequest request, HttpServletResponse response)

 throws ServletException, IOException {

 try {

 int id = Integer.parseInt(request.getParameter("statusid"));

 List<Document> allDocuments = (List<Document>) SessionFactory.getInstance().getDocumentDAO().getDocumentsByStatus(id);

 request.setAttribute("allDocuments", allDocuments);

 request.getRequestDispatcher("/alldocs.jsp").forward(request, response);

 } catch (SQLException ex) {

 Logger.getLogger(showAllDocs.class.getName()).log(Level.SEVERE, null, ex);

 }

 }

 @Override

 protected void doPost(HttpServletRequest request, HttpServletResponse response)

 throws ServletException, IOException {

 //processRequest(request, response);

 }

 @Override

 public String getServletInfo() {

 return "Short description";

 }// </editor-fold>

}

***showDocsByType.java***

package servlets;

Приложение Г (продолжение)

import DAO.SessionFactory;

import java.io.IOException;

import java.sql.SQLException;

import java.util.List;

import java.util.logging.Level;

import java.util.logging.Logger;

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 models.Document;

@WebServlet(name = "showDocsByType", urlPatterns = {"/showdocsbytype"})

public class showDocsByType extends HttpServlet {

 @Override

 protected void doGet(HttpServletRequest request, HttpServletResponse response)

 throws ServletException, IOException {

 try {

 int id = Integer.parseInt(request.getParameter("typeid"));

 List<Document> allDocuments = (List<Document>) SessionFactory.getInstance().getDocumentDAO().getDocumentsByType(id);

 request.setAttribute("allDocuments", allDocuments);

 request.getRequestDispatcher("/alldocs.jsp").forward(request, response);

 } catch (SQLException ex) {

 Logger.getLogger(showAllDocs.class.getName()).log(Level.SEVERE, null, ex);

 }

 }

 @Override

 protected void doPost(HttpServletRequest request, HttpServletResponse response)

 throws ServletException, IOException {

 //processRequest(request, response);

 }

 @Override

 public String getServletInfo() {

 return "Short description";

 }// </editor-fold>

}

***showTerms.java***

package servlets;

import DAO.SessionFactory;

import java.io.IOException;

import java.sql.SQLException;

import java.util.List;

import java.util.logging.Level;

import java.util.logging.Logger;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

Приложение Г (продолжение)

import javax.servlet.http.HttpServletResponse;

import models.Document;

@WebServlet(name = "showTerms", urlPatterns = {"/showterms"})

public class showTerms extends HttpServlet {

 @Override

 protected void doGet(HttpServletRequest request, HttpServletResponse response)

 throws ServletException, IOException {

 try {

 List<Document> expiredDocuments = (List<Document>) SessionFactory.getInstance().getDocumentDAO().getDocumentsExpired();

 request.setAttribute("expiredDocuments", expiredDocuments);

 List<Document> expiringNextDaysDocuments = (List<Document>) SessionFactory.getInstance().getDocumentDAO().getDocumentsExpiringNextDays(10);

 request.setAttribute("expiringNextDaysDocuments", expiringNextDaysDocuments);

 request.getRequestDispatcher("/terms.jsp").forward(request, response);

 } catch (SQLException ex) {

 Logger.getLogger(showAllDocs.class.getName()).log(Level.SEVERE, null, ex);

 }

 }

 @Override

 protected void doPost(HttpServletRequest request, HttpServletResponse response)

 throws ServletException, IOException {

 //processRequest(request, response);

 }

 @Override

 public String getServletInfo() {

 return "Short description";

 }

}

***updateDocument.java***

package servlets;

import DAO.SessionFactory;

import java.io.IOException;

import java.sql.SQLException;

import java.util.List;

import java.util.logging.Level;

import java.util.logging.Logger;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import models.Document;

public class updateDocument extends HttpServlet {

 protected void processRequest(HttpServletRequest request, HttpServletResponse response)

 throws ServletException, IOException {

 response.setContentType("text/html;charset=UTF-8");

Приложение Г (продолжение)

 }

 @Override

 protected void doGet(HttpServletRequest request, HttpServletResponse response)

 throws ServletException, IOException {

 request.setCharacterEncoding("UTF-8");

 try {

 int id = Integer.parseInt(request.getParameter("docid"));

 Document currentDoc = SessionFactory.getInstance().getDocumentDAO().getDocumentById(id);

 List<String> allTypes = (List<String>) SessionFactory.getInstance().getTypeDAO().getAllTypes();

 request.setAttribute("allTypes", allTypes);

 List<String> allStatuses = (List<String>) SessionFactory.getInstance().getStatusDAO().getAllStatuses();

 request.setAttribute("allStatuses", allStatuses);

 List<String> allPerformers = (List<String>) SessionFactory.getInstance().getPerformerDAO().getAllPerformers();

 request.setAttribute("allPerformers", allPerformers);

 List<String> allInitiators = (List<String>) SessionFactory.getInstance().getInitiatorDAO().getAllInitiators();

 request.setAttribute("allInitiators", allInitiators);

 String input\_number = request.getParameter("input\_number");

 String reg\_date = request.getParameter("reg\_date");

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

 String type\_id = request.getParameter("type");

 String initiator\_id = request.getParameter("initiator");

 String status\_id = request.getParameter("status");

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

 String performer\_id = request.getParameter("performer");

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

 String desc = request.getParameter("description");

 SessionFactory.getInstance().getDocumentDAO().updateDocument(id, input\_number, reg\_date, name,

 type\_id, initiator\_id, status\_id, deadline, performer\_id, file, desc);

 if (currentDoc != null) {

 List<Document> allDocuments = null;

 allDocuments = (List<Document>) SessionFactory.getInstance().getDocumentDAO().getAllDocuments();

 request.setAttribute("allDocuments", allDocuments);

 request.getRequestDispatcher("/alldocs.jsp").forward(request, response);

 } else {

 request.getRequestDispatcher("/errordocs.jsp").forward(request, response);

 }

 } catch (SQLException ex) {

 Logger.getLogger(showAllDocs.class.getName()).log(Level.SEVERE, null, ex);

 }

 }

 @Override

 protected void doPost(HttpServletRequest request, HttpServletResponse response)

 throws ServletException, IOException {

 processRequest(request, response);

 }

 @Override

Приложение Г (продолжение)

 public String getServletInfo() {

 return "Short description";

 }// </editor-fold>

}

***uploadFile.java***

package servlets;

import DAO.SessionFactory;

import java.io.IOException;

import java.sql.SQLException;

import java.util.List;

import java.util.logging.Level;

import java.util.logging.Logger;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import models.Document;

public class updateDocument extends HttpServlet {

 protected void processRequest(HttpServletRequest request, HttpServletResponse response)

 throws ServletException, IOException {

 response.setContentType("text/html;charset=UTF-8");

 }

 @Override

 protected void doGet(HttpServletRequest request, HttpServletResponse response)

 throws ServletException, IOException {

 request.setCharacterEncoding("UTF-8");

 try {

 int id = Integer.parseInt(request.getParameter("docid"));

 Document currentDoc = SessionFactory.getInstance().getDocumentDAO().getDocumentById(id);

 List<String> allTypes = (List<String>) SessionFactory.getInstance().getTypeDAO().getAllTypes();

 request.setAttribute("allTypes", allTypes);

 List<String> allStatuses = (List<String>) SessionFactory.getInstance().getStatusDAO().getAllStatuses();

 request.setAttribute("allStatuses", allStatuses);

 List<String> allPerformers = (List<String>) SessionFactory.getInstance().getPerformerDAO().getAllPerformers();

 request.setAttribute("allPerformers", allPerformers);

 List<String> allInitiators = (List<String>) SessionFactory.getInstance().getInitiatorDAO().getAllInitiators();

 request.setAttribute("allInitiators", allInitiators);

 String input\_number = request.getParameter("input\_number");

 String reg\_date = request.getParameter("reg\_date");

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

 String type\_id = request.getParameter("type");

 String initiator\_id = request.getParameter("initiator");

 String status\_id = request.getParameter("status");

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

 String performer\_id = request.getParameter("performer");

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

Приложение Г (продолжение)

 String desc = request.getParameter("description");

 SessionFactory.getInstance().getDocumentDAO().updateDocument(id, input\_number, reg\_date, name,

 type\_id, initiator\_id, status\_id, deadline, performer\_id, file, desc);

 if (currentDoc != null) {

 List<Document> allDocuments = null;

 allDocuments = (List<Document>) SessionFactory.getInstance().getDocumentDAO().getAllDocuments();

 request.setAttribute("allDocuments", allDocuments);

 request.getRequestDispatcher("/alldocs.jsp").forward(request, response);

 } else {

 request.getRequestDispatcher("/errordocs.jsp").forward(request, response);

 }

 } catch (SQLException ex) {

 Logger.getLogger(showAllDocs.class.getName()).log(Level.SEVERE, null, ex);

 }

 }

 @Override

 protected void doPost(HttpServletRequest request, HttpServletResponse response)

 throws ServletException, IOException {

 processRequest(request, response);

 }

 @Override

 public String getServletInfo() {

 return "Short description";

 }// </editor-fold>