Минобрнауки России | РГРТУ | Кафедра ВПМ

Дисциплина  
«СИСТЕМЫ УПРАВЛЕНИЯ БИЗНЕС-ПРОЦЕССАМИ»

**Отчет о практических работах**

Выполнил:

Малистова Мария Романовна

студент группы 044м

электронная почта malistovam@mail.ru

Проверил:

Пруцков Александр Викторович

д-р техн. наук, профессор кафедры ВПМ

Рязань 2023

# Описание сетевой информационной системы

## Цель работы

Цель работы – целью работы является автоматизация бизнес-процесса выкладывания статей в журнал с помощью создания сетевой информационной системы для упрощения работы и сокращения вовлечения людей в процесс.

## Концептуальная и логическая модели базы данных

В ходе разработки информационной системы были выявлены следующие сущности:

1) Пользователь

Первичный ключ: Код\_Пользователя

2) Группа\_Пользователей

Первичный ключ: Код\_Группы

3) Статья

Первичный ключ: Код\_Статьи

4) Этап

Первичный ключ: Код\_Этапа

5) Выпуск

Первичный ключ: Код\_Выпуска

6) Доступ\_к\_Статье

Первичный ключ: Код\_Доступа

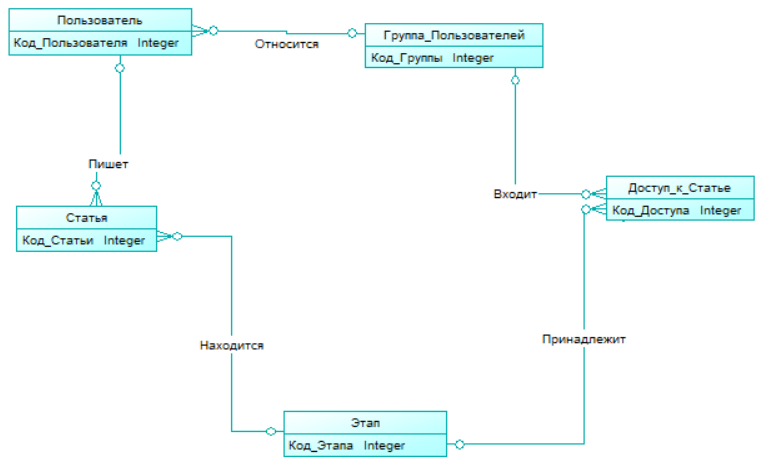


Рисунок 1 – Концептуальная модель базы данных.

Таблицы содержат следующие атрибуты:

1) Пользователь

Код\_Пользователя,

Роль,

Логин,

Пароль,

Доступность.

2) Группа\_Пользователей

Код\_Группы,

Название\_Группы.

3) Статья

Код\_Статьи,

Автор,

Текст,

Этап,

Модератор,

Редактор,

Корректор.

4) Этап

Код\_Этапа,

Название\_Этапа.

5) Доступ\_к\_Статье

Код\_Доступа,

Группа,

Этап.

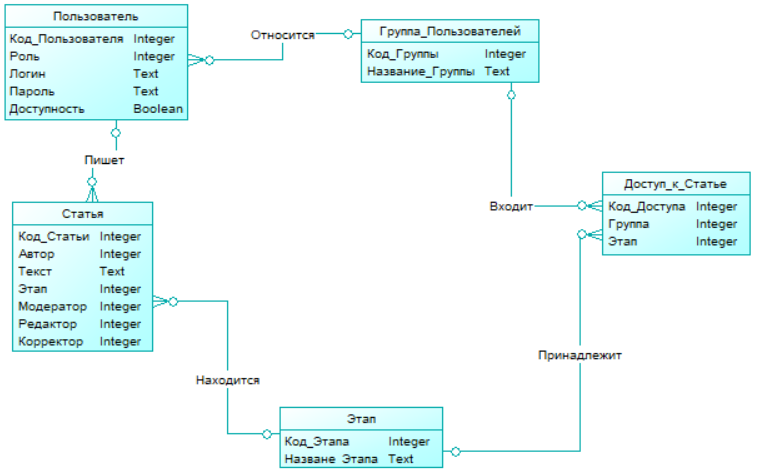


Рисунок 2 – Логическая модель базы данных.

## Логика работы сетевой информационной системы, её связь с моделью базы данных

Действия групп пользователей изменяют данные в базе данных (таблица 1).

Таблица 1 – Действия групп пользователей и соответствующие им изменения в базе данных

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Действие | Группа пользователей | | Изменения данных в базе данных | |
| Добавление/удаление пользователя | | Администратор | | В таблице «Пользователь» добавилась/удалилась информация о пользователе |
| Редактирование информации о пользователе | | Администратор | | В таблице «Пользователь» изменилась информация о пользователе |
| Блокирование/разблокирование пользователя | | Модератор | | В таблице «Пользователь» изменится значение поля «Доступность» |
| Удаление статьи пользователя | | Модератор | | В таблице «Статья» удалится запись со статьей пользователя |
| Принятия статьи пользователя | | Модератор | | В таблице «Статья» изменится значение поля «Этап» у статьи пользователя |
| Публикация статьи | | Автор | | В таблице «Статья» добавится новая запись |
| Редактирование статьи | | Редактор | | В таблице «Статья» изменится значения полей «Этап» и «Текст» у статьи пользователя |
| Корректирование статьи | | Корректор | | В таблице «Статья» изменится значения полей «Этап» и «Текст» у статьи пользователя |

## Отчеты, предоставляемые администратору, модератору и другим группам пользователей

Группам пользователей выдаются различные отчеты (таблица 2).

Таблица 2 – Отчеты для групп пользователей

|  |  |
| --- | --- |
| Группа пользователей | Наименование отчета |
| Модераторы | Отчет о редактировании |
| Модераторы | Отчет о корректировании |
| Администраторы | Отчеты о статьях |
| Автор | Отчет о статье пользователя |

## Описание автоматизируемого бизнес-процесс

Процесс состоит из пяти этапов:

1. Прислать статью

Пользователь присылает свою статью и она отправляется на рассмотрение модератором;

2. Рассмотрение

Модератор читает статью и решает, будет ли она помещена в журнал, если да, то статья передается на редакцию;

3. Редактор редактирует статью, после чего она отправляется на корректирование;

4. Корректор исправляет ошибки в статье, и она публикуется;

5. Автор просматривает опубликованную статью.

Статьи автоматически передаются от одних групп пользователей к другим.

# Требуемое программное обеспечение

Для функционирования разработанной сетевой информационной системы требуется программное обеспечение (таблица 3).

Таблица 3 – Требуемое программное обеспечение

|  |  |  |
| --- | --- | --- |
| Название | Версия | Назначение |
| JDK | 1.8 | Библиотеки классов и компилятор |
| SQL Developer | 20.4.1 | Среда разработки БД |
| Oracle XE Express Edition | 18c | СУБД |
| Apache Tomcat | 9.0 | Сервер приложения |
| JDBC | 6 | Взаимодействие с СУБД |
| JSTL | 1.2 | Тэги для jsp |

# Список используемых запросов на языке SQL

## Запросы создания таблиц базы данных

--create table users that contains information about app's users

CREATE TABLE users (

user\_id INTEGER GENERATED ALWAYS as IDENTITY(START with 1 INCREMENT by 1),

user\_role INTEGER,

login VARCHAR2(50),

user\_password VARCHAR2(50),

has\_access NUMBER(1),

CONSTRAINT login\_unique UNIQUE (login));

--create table user\_groups that contains information about users' groups

CREATE TABLE user\_groups (

user\_group\_id INTEGER,

group\_name VARCHAR2(30),

CONSTRAINT group\_name\_unique UNIQUE (group\_name));

--create table article that contains information about articles

CREATE TABLE article (

article\_id INTEGER GENERATED ALWAYS as IDENTITY(START with 1 INCREMENT by 1),

author INTEGER,

text LONG,

step INTEGER,

moderator INTEGER,

editor INTEGER,

corrector INTEGER);

--create table article\_access that contains information about which user groups have an access to articles at which step

CREATE TABLE article\_access (

access\_id INTEGER,

user\_role INTEGER,

step INTEGER);

--create table steps that contains information about steps that articles have

CREATE TABLE steps (

step\_id INTEGER,

step\_role VARCHAR2(20),

CONSTRAINT step\_role\_unique UNIQUE (step\_role));

--add primary key to the table users

ALTER TABLE users ADD (

CONSTRAINT users\_pk PRIMARY KEY (user\_id));

--add primary key to the table user\_groups

ALTER TABLE user\_groups ADD (

CONSTRAINT user\_groups\_pk PRIMARY KEY (user\_group\_id));

--add primary key to the table article

ALTER TABLE article ADD (

CONSTRAINT article\_pk PRIMARY KEY (article\_id));

--add primary key to the table article\_access

ALTER TABLE article\_access ADD (

CONSTRAINT article\_access\_pk PRIMARY KEY (access\_id, user\_role));

--add primary key to the table steps

ALTER TABLE steps ADD (

CONSTRAINT step\_pk PRIMARY KEY (step\_id));

--add foreign key on the table users to the table user\_groups

ALTER TABLE users

ADD FOREIGN KEY (user\_role) REFERENCES user\_groups(user\_group\_id);

--add foreign key on the table article\_access to the table user\_groups

ALTER TABLE article\_access

ADD FOREIGN KEY (user\_role) REFERENCES user\_groups(user\_group\_id);

--add foreign key on the table article\_access to the table steps

ALTER TABLE article\_access

ADD FOREIGN KEY (step) REFERENCES steps(step\_id);

--add foreign key on the table article to the table users

ALTER TABLE article

ADD FOREIGN KEY (step) REFERENCES users(user\_id);

--add foreign key on the table article to the table users

ALTER TABLE article

ADD FOREIGN KEY (author) REFERENCES users(user\_id);

--add foreign key on the table article to the table users

ALTER TABLE article

ADD FOREIGN KEY (moderator) REFERENCES users(user\_id);

--add foreign key on the table article to the table users

ALTER TABLE article

ADD FOREIGN KEY (editor) REFERENCES users(user\_id);

--add foreign key on the table article to the table users

ALTER TABLE article

ADD FOREIGN KEY (corrector) REFERENCES users(user\_id);

## Запросы выборки данных из таблиц базы данных

--select user by login and password

select users.user\_id, user\_groups.user\_group\_id, user\_groups.group\_name, users.login, users.user\_password, users.has\_access from users inner join user\_groups on users.user\_role = user\_groups.user\_group\_id where users.login = ? and users.user\_password = ?

--select user groups by group name

select user\_group\_id, group\_name from user\_groups where group\_name = ?

--select users that has specific group

select u.user\_id, u.login, u.user\_password, u.has\_access,ug.user\_group\_id, ug.group\_name from users u inner join user\_groups ug on u.user\_role = ug.user\_group\_id

--select all user groups

select user\_group\_id, group\_name from user\_groups

--select user with specific id

select u.user\_id, u.login, u.user\_password, u.has\_access,ug.user\_group\_id, ug.group\_name from users u inner join user\_groups ug on u.user\_role = ug.user\_group\_id where u.user\_id = ?

--select articles that admin has access to

select a.article\_id, a.author, u.login as author\_login, a.text, a.step, ac.user\_role, ug.group\_name, s.step\_role, a.moderator, um.login as moder, a.editor, ue.login as edit, a.corrector, uc.login as correct from article a join users u on a.author = u.user\_id inner join steps s on a.step = s.step\_id left join users um on a.moderator = um.user\_id left join users ue on a.editor = ue.user\_id left join users uc on a.corrector = uc.user\_id inner join article\_access ac on a.step = ac.step inner join user\_groups ug on ac.user\_role = ug.user\_group\_id where a.step = (select steps.step\_id from steps inner join article\_access on article\_access.step = steps.step\_id inner join user\_groups on article\_access.user\_role = user\_groups.user\_group\_id where user\_groups.group\_name='Administrator')

--select users of a specific group

select u.user\_id, u.login, ug.group\_name from users u inner join user\_groups ug on u.user\_role = ug.user\_group\_id where ug.group\_name = ?

-- select articles that moderator has access to

select a.article\_id, a.author, u.login as author\_login, a.text, a.step, ac.user\_role, ug.group\_name, s.step\_role, a.moderator, um.login as moder, a.editor, ue.login as edit, a.corrector, uc.login as correct from article a join users u on a.author = u.user\_id inner join steps s on a.step = s.step\_id left join users um on a.moderator = um.user\_id left join users ue on a.editor = ue.user\_id left join users uc on a.corrector = uc.user\_id inner join article\_access ac on a.step = ac.step inner join user\_groups ug on ac.user\_role = ug.user\_group\_id where a.step = (select steps.step\_id from steps inner join article\_access on article\_access.step = steps.step\_id inner join user\_groups on article\_access.user\_role = user\_groups.user\_group\_id where user\_groups.group\_name='Moderator') and a.moderator = ?

-- select articles assigned to a specific editor

select a.article\_id, a.author, u.login as author\_login, a.text, a.step, ac.user\_role, ug.group\_name, s.step\_role, a.moderator, um.login as moder, a.editor, ue.login as edit, a.corrector, uc.login as correct from article a join users u on a.author = u.user\_id inner join steps s on a.step = s.step\_id left join users um on a.moderator = um.user\_id left join users ue on a.editor = ue.user\_id left join users uc on a.corrector = uc.user\_id inner join article\_access ac on a.step = ac.step inner join user\_groups ug on ac.user\_role = ug.user\_group\_id where a.step = (select steps.step\_id from steps inner join article\_access on article\_access.step = steps.step\_id inner join user\_groups on article\_access.user\_role = user\_groups.user\_group\_id where user\_groups.group\_name='Editor') and a.editor = ?

-- select articles assigned to a specific corrector

select a.article\_id, a.author, u.login as author\_login, a.text, a.step, ac.user\_role, ug.group\_name, s.step\_role, a.moderator, um.login as moder, a.editor, ue.login as edit, a.corrector, uc.login as correct from article a join users u on a.author = u.user\_id inner join steps s on a.step = s.step\_id left join users um on a.moderator = um.user\_id left join users ue on a.editor = ue.user\_id left join users uc on a.corrector = uc.user\_id inner join article\_access ac on a.step = ac.step inner join user\_groups ug on ac.user\_role = ug.user\_group\_id where a.step = (select steps.step\_id from steps inner join article\_access on article\_access.step = steps.step\_id inner join user\_groups on article\_access.user\_role = user\_groups.user\_group\_id where user\_groups.group\_name='Corrector') and a.corrector = ?

--select article with specific id

select a.article\_id, a.author, u.login as author\_login, a.text, a.step, ac.user\_role, ug.group\_name, s.step\_role, a.moderator, um.login as moder, a.editor, ue.login as edit, a.corrector, uc.login as correct from article a join users u on a.author = u.user\_id inner join steps s on a.step = s.step\_id left join users um on a.moderator = um.user\_id left join users ue on a.editor = ue.user\_id left join users uc on a.corrector = uc.user\_id inner join article\_access ac on a.step = ac.step inner join user\_groups ug on ac.user\_role = ug.user\_group\_id where a.article\_id = ?

--select step with specific step id

select s.step\_id, s.step\_role from steps s where s.step\_id = ?

--select articles that have been edited and correced

select a.article\_id, a.author, u.login as author\_login, a.text, a.step, ac.user\_role, ug.group\_name, s.step\_role, a.moderator, um.login as moder, a.editor, ue.login as edit, a.corrector, uc.login as correct from article a join users u on a.author = u.user\_id inner join steps s on a.step = s.step\_id left join users um on a.moderator = um.user\_id left join users ue on a.editor = ue.user\_id left join users uc on a.corrector = uc.user\_id inner join article\_access ac on a.step = ac.step inner join user\_groups ug on ac.user\_role = ug.user\_group\_id where a.step = (select steps.step\_id from steps inner join article\_access on article\_access.step = steps.step\_id inner join user\_groups on article\_access.user\_role = user\_groups.user\_group\_id where user\_groups.group\_name='Author')

## Запросы изменения данных в таблицах базы данных

--change user's login and password

update users set login = ?, user\_password = ? where user\_id = ?

--add new user

insert into users(user\_role, login, user\_password, has\_access) values (?,?,?,1)

--delete user

delete from users where user\_id = ?

--add new article

insert into article(author, text, step) values (?,?,1)

--assign moderator to an article

update article set moderator = (select user\_id from users where login = ?), step = (select step\_id from steps where step\_role = 'Reading') where article\_id = ?

--block or unblock a user

update users set has\_access = ? where user\_id = ?

--assign editor and corrector to an article

update article set editor = (select user\_id from users where login = ?), corrector = (select user\_id from users where login = ?),step = (select step\_id from steps where step\_role = 'Editing') where article\_id = ?

--delete an article

delete from article where article\_id = ?

--change article's text

update article set text = ? where article\_id = ?

--chance article's step

update article set step = (select step\_id from steps where step\_role = ?) where article\_id = ?

# Инструкции по работе с сетевой информационной системой

## Инструкция администратора

Пройдя авторизацию, администратор оказывается на своей основной странице, где он может видеть всех пользователей системы, их роль, логин, пароль и возможность доступа к системе (рисунок 3).

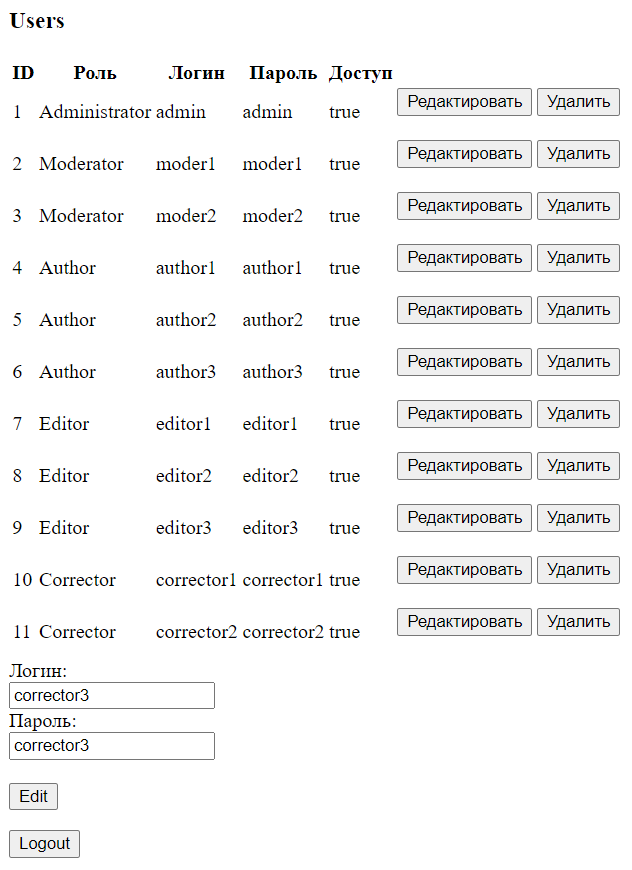


Рисунок 3 – Список пользователей системы.

Администратор может удалить пользователя, нажав на кнопку "Удалить" рядом с соответствующим пользователем. Также он может отредактировать данные пользователя, нажав на кнопку "Редактировать" рядом с пользователем, после этого откроется форма редактирования, в которой можно поменять логин и пароль пользователя и сохранить изменения, нажав кнопку "Edit".

Также администратор может добавить нового пользователя. Для этого на странице есть форма, где нужно ввести логин, пароль и роль нового пользователя, после этого нажать "Добавить". Администратор посмотреть статьи, присланные авторами и назначить к ним модератора. Для этого из выпадающего списка около нужной статьи надо выбрать модератора, который будет ответственным за эту статью, и нажать кнопку "Назначить".

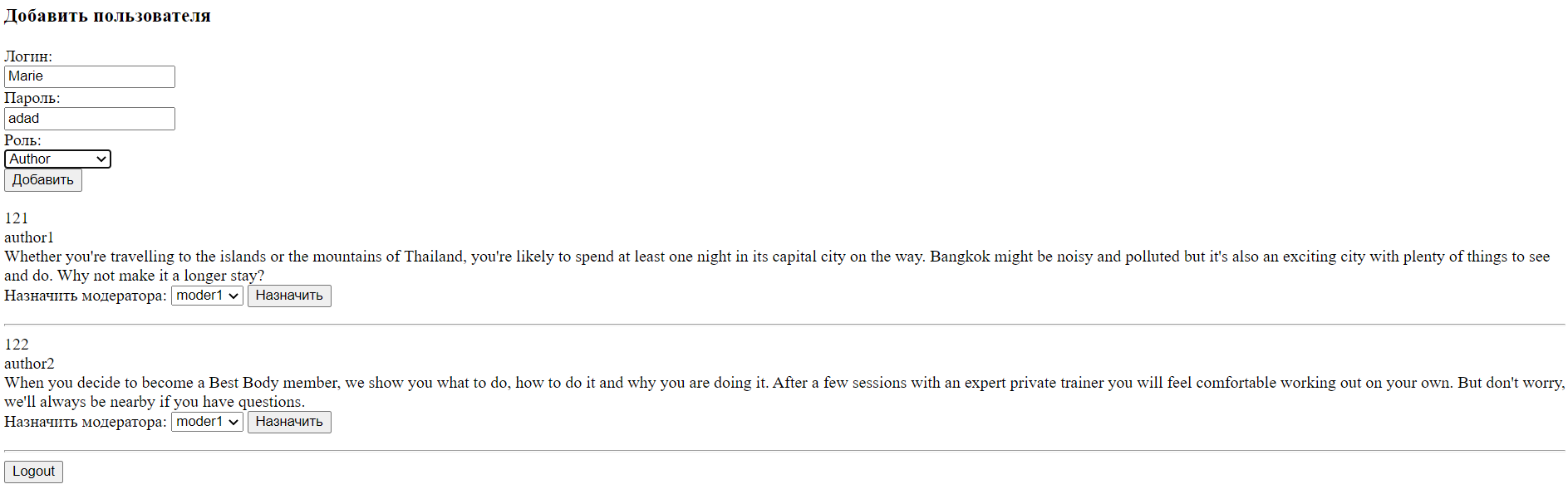


Рисунок 4 – Добавление пользователя и назначение модератора статье.

## Инструкция модератора

Модератор видит всех пользователей системы, их логин, роль и возможность доступа. Он может заблокировать любого пользователя кроме администратора. Для этого рядом с этим пользователем нужно нажать кнопку "Заблокировать", после чего этот пользователь не сможет входить в систему. Чтобы вернуть пользователю доступ, надо нажать кнопку "Разблокировать" рядом с этим пользователем.

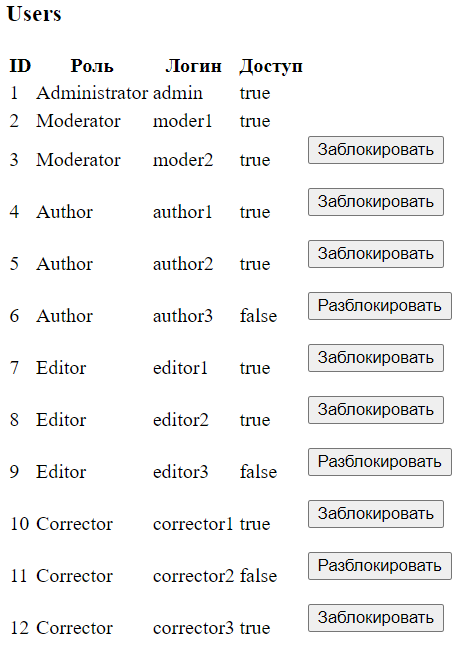


Рисунок 5 – Информация о пользователях.

Также модератор может просматривать статьи, которые ему назначены. Он может удалить статью, либо отправить ей на дальнейшую редакцию. Для этого надо выбрать из выпадающих списков редактора и корректора, которые будут редактировать и корректировать эту статью соответственно, после чего нажать "Назначить".

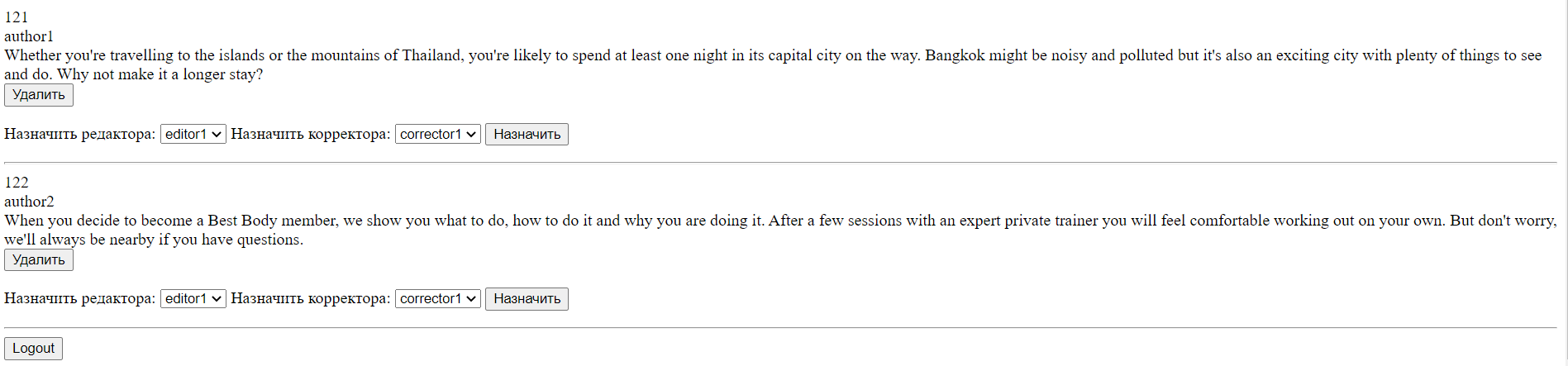


Рисунок 6 – Просмотр статей.

## Инструкция автора

Автор может добавлять свои статьи в систему, для этого текст статьи нужно ввести в поле ввода, после чего нажать кнопку "Добавить". Также автор может просматривать уже опубликованные статьи с информацией об их авторе, модераторе, корректоре и редакторе.

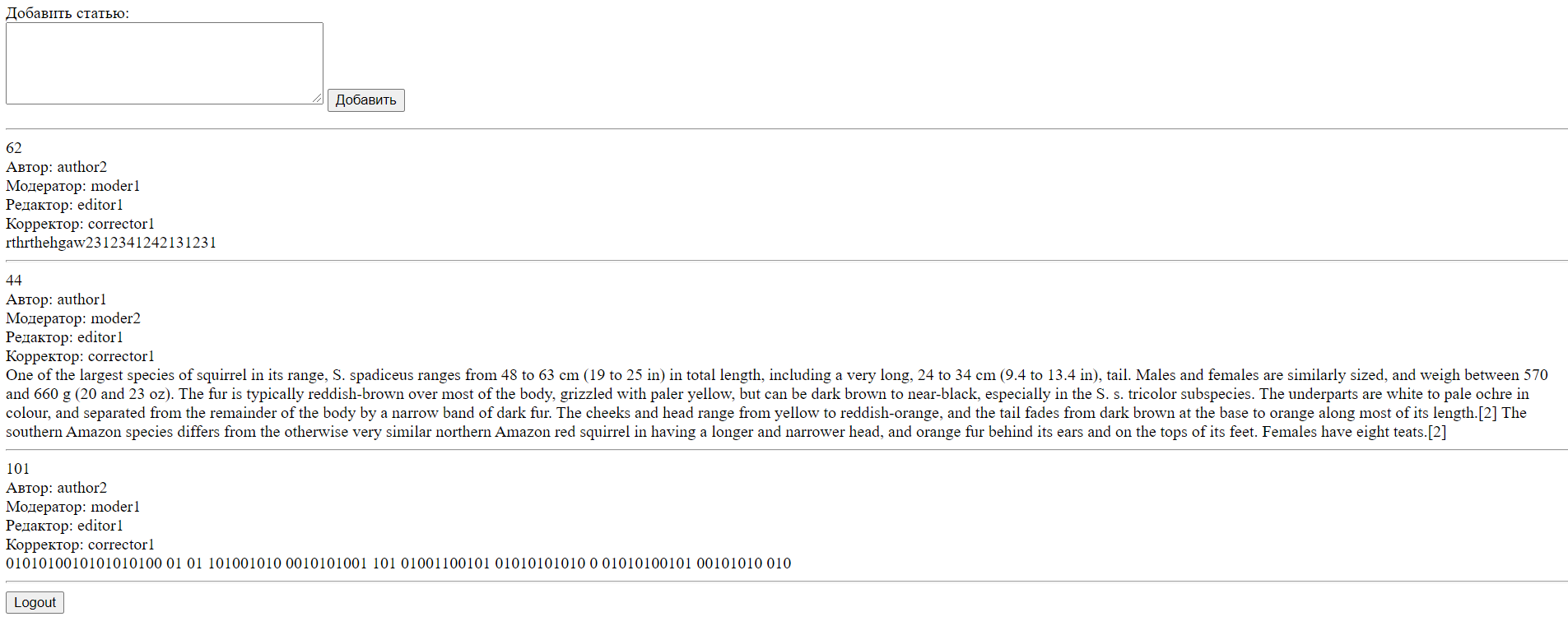


Рисунок 6 – Страница автора.

## Инструкция редактора/корректора

На странице данного пользователя отображаются назначенные ему статьи. Редактор и корректор могут исправить в них ошибки. Редактор исправляет содержание текста, а корректор исправляет ошибки.

Чтобы отредактировать статью, нужно нажать на кнопку "Edit" рядом с ней, после чего можно будет исправить её текст. Чтобы сохранить изменения, следует нажать кнопку "Save". Чтобы закончить работу со статьей, нужно нажать кнопку "Завершить редактирование".

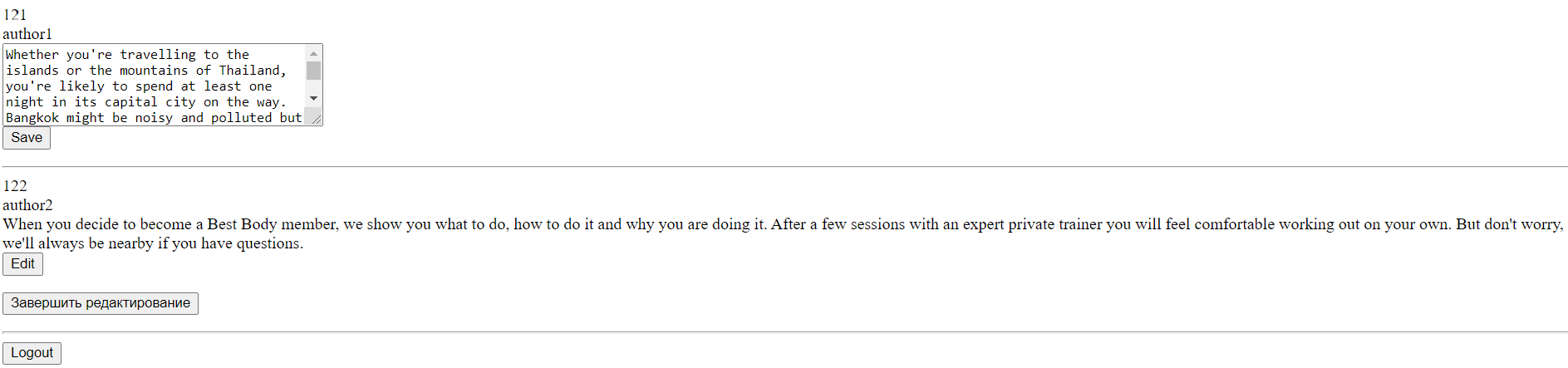


Рисунок 7 – Редактирование статьи.

# Листинг основных классов программы с комментариями Javadoc

## FrontController

public class FrontController extends HttpServlet {

public final static String ADMIN\_URL = "/02\_06\_malistova/journal?action=admin";

public final static String LOGIN\_URL = "/02\_06\_malistova/journal?action=login";

public final static String AUTHOR\_URL = "/02\_06\_malistova/journal?action=author";

public final static String MODERATOR\_URL = "/02\_06\_malistova/journal?action=moderator";

public final static String EDITS\_URL = "/02\_06\_malistova/journal?action=edits";

HttpSession session = null;

private Users user;

private UsersDao userDao = new UsersDao();

private UserGroupDao groupDao = new UserGroupDao();

private ArticleDao articleDao = new ArticleDao();

/\*\*

\* Urls for the method POST

\* @param request

\* @param response

\* @throws ServletException

\* @throws IOException

\*/

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

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

session = request.getSession();

if (action != null) {

switch (action) {

case "login":

try {

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

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

user = userDao.findUser(login, password);

if (user != null) {

String role = user.getUserGroup().getGroupName();

session.setAttribute("name", user.getUserId());

session.setAttribute("userGroup", role);

switch (role) {

case "Administrator":

response.sendRedirect(ADMIN\_URL);

break;

case "Author":

response.sendRedirect(AUTHOR\_URL);

break;

case "Moderator":

response.sendRedirect(MODERATOR\_URL);

break;

case "Editor":

case "Corrector":

response.sendRedirect(EDITS\_URL);

break;

default:

response.sendRedirect(LOGIN\_URL);

break;

}

} else {

response.sendRedirect(LOGIN\_URL);

}

} catch (SQLException e) {

e.printStackTrace();

}

break;

case "adminedituser":

String editedUserId = request.getParameter("getEditedId");

try {

List<Users> users = userDao.selectUsers();

List<UserGroups> roles = groupDao.selectRoles();

Users editedUser = userDao.getUserById(Integer.parseInt(editedUserId));

request.setAttribute("users", users);

request.setAttribute("roles", roles);

request.setAttribute("editedUser", editedUser);

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

} catch (SQLException e) {

e.printStackTrace();

}

break;

case "useredited":

String newUserLogin = request.getParameter("login");

String newPassword = request.getParameter("password");

String userId = request.getParameter("editedUserId");

try {

userDao.updateUser(newUserLogin, newPassword, Integer.parseInt(userId));

} catch (SQLException e) {

e.printStackTrace();

}

response.sendRedirect(ADMIN\_URL);

break;

case "adminadduser":

try {

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

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

String roleName = request.getParameter("role");

UserGroups role = groupDao.getUserGroupName(roleName);

userDao.addUser(role.getUserGroupId(), login, password);

response.sendRedirect(ADMIN\_URL);

} catch (SQLException e) {

e.printStackTrace();

}

break;

case "admindeleteuser":

String deletedUserId = request.getParameter("getDeletedId");

try {

userDao.deleteUser(Integer.parseInt(deletedUserId));

} catch (SQLException e) {

e.printStackTrace();

}

response.sendRedirect(ADMIN\_URL);

break;

case "authoraddarticle":

try {

String text = request.getParameter("articletext");

int author = (int) session.getAttribute("name");

articleDao.addArticle(author, text);

response.sendRedirect(AUTHOR\_URL);

} catch (SQLException e) {

e.printStackTrace();

}

break;

case "assignmoder":

try {

String article = request.getParameter("chosenarticle");

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

articleDao.assignModer(article, moder);

response.sendRedirect(ADMIN\_URL);

} catch (SQLException e) {

e.printStackTrace();

}

break;

case "blockuser":

try {

String blockedUser = request.getParameter("getBlockedId");

userDao.blockUser(Integer.parseInt(blockedUser), false);

response.sendRedirect(MODERATOR\_URL);

} catch (SQLException e) {

e.printStackTrace();

}

break;

case "unblockuser":

try {

String unblockedUser = request.getParameter("getUnblockedId");

userDao.blockUser(Integer.parseInt(unblockedUser), true);

response.sendRedirect(MODERATOR\_URL);

} catch (SQLException e) {

e.printStackTrace();

}

break;

case "deletearticle":

String deletedArticle = request.getParameter("deletedarticle");

try {

articleDao.deleteArticle(Integer.parseInt(deletedArticle));

} catch (SQLException e) {

e.printStackTrace();

}

response.sendRedirect(MODERATOR\_URL);

break;

case "assigneditorcorrector":

try {

String article = request.getParameter("chosenarticle");

String editor = request.getParameter("edit");

String corrector = request.getParameter("correct");

articleDao.assignEditorCorrector(article, editor, corrector);

response.sendRedirect(MODERATOR\_URL);

} catch (SQLException e) {

e.printStackTrace();

}

break;

case "editarticle":

String articleId = request.getParameter("getEditedId");

String articleText = request.getParameter("articletext");

try {

articleDao.updateArticle(articleText, Integer.parseInt(articleId));

} catch (SQLException e) {

e.printStackTrace();

}

response.sendRedirect(EDITS\_URL);

break;

case "articleedits":

String editedArticleId = request.getParameter("getEditedArticleId");

try {

Users user = userDao.getUserById((int)session.getAttribute("name"));

List<Articles> articles = articleDao.selectArticles(user.getUserId(), user.getUserGroup().getGroupName());

Articles editedArticle = articleDao.getArticleById(Integer.parseInt(editedArticleId));

request.setAttribute("editedArticle", editedArticle);

request.setAttribute("articles", articles);

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

} catch (SQLException e) {

e.printStackTrace();

}

break;

case "assignnext":

try {

String editedDone = request.getParameter("getEditedDoneId");

articleDao.assignNext(articleDao.getArticleById(Integer.parseInt(editedDone)));

response.sendRedirect(EDITS\_URL);

} catch (SQLException e) {

e.printStackTrace();

}

break;

case "logout":

session.setAttribute("name", null);

session.setAttribute("userGroup", null);

response.sendRedirect(LOGIN\_URL);

break;

default:

break;

}

}

}

/\*\*

\* Urls for the method GET

\* @param request

\* @param response

\* @throws ServletException

\* @throws IOException

\*/

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

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

session = request.getSession();

if (action != null) {

switch (action) {

case "login":

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

break;

case "admin":

try {

List<Users> users = userDao.selectUsers();

Users user = userDao.getUserById((int)session.getAttribute("name"));

List<Articles> articles = articleDao.selectArticles(user.getUserId(), user.getUserGroup().getGroupName());

List<UserGroups> roles = groupDao.selectRoles();

List<Users> moders = userDao.selectSpecificRole("Moderator");

request.setAttribute("users", users);

request.setAttribute("roles", roles);

request.setAttribute("articles", articles);

request.setAttribute("editedUser", null);

request.setAttribute("moders", moders);

request.setAttribute("session", session.getAttribute("name"));

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

} catch (SQLException e) {

e.printStackTrace();

}

break;

case "author":

try {

List<Articles> articles = articleDao.selectArticles(user.getUserId(), user.getUserGroup().getGroupName());

request.setAttribute("articles", articles);

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

} catch (SQLException e) {

e.printStackTrace();

}

break;

case "moderator":

try {

List<Users> users = userDao.selectUsers();

Users user = userDao.getUserById((int)session.getAttribute("name"));

List<Articles> articles = articleDao.selectArticles(user.getUserId(), user.getUserGroup().getGroupName());

List<Users> editors = userDao.selectSpecificRole("Editor");

List<Users> correctors = userDao.selectSpecificRole("Corrector");

request.setAttribute("session", session.getAttribute("name"));

request.setAttribute("users", users);

request.setAttribute("articles", articles);

request.setAttribute("editors", editors);

request.setAttribute("correctors", correctors);

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

} catch (SQLException e) {

e.printStackTrace();

}

break;

case "edits":

try {

Users user = userDao.getUserById((int)session.getAttribute("name"));

List<Articles> articles = articleDao.selectArticles(user.getUserId(), user.getUserGroup().getGroupName());

System.out.println(articles);

request.setAttribute("session", session.getAttribute("name"));

request.setAttribute("articles", articles);

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

} catch (SQLException e) {

e.printStackTrace();

}

break;

default:

break;

}

}

}

}

## Users

public class Users {

private int userId;

private UserGroups userGroup;

private String login;

private String userPassword;

private Boolean hasAccess;

/\*\*

\* Getter for userId

\* @return

\*/

public int getUserId() { return userId; }

/\*\*

\* Setter for userId

\* @param userId

\*/

public void setUserId(int userId) { this.userId = userId; }

/\*\*

\* Getter for userGroup

\* @return

\*/

public UserGroups getUserGroup() { userGroup; }

/\*\*

\* Setter for userGroup

\* @param role

\*/

public void setUserGroup(UserGroups role) { this.userGroup = role; }

/\*\*

\* Getter for login

\* @return

\*/

public String getLogin() { return login; }

/\*\*

\* Setter for login

\* @param login

\*/

public void setLogin(String login) { this.login = login; }

/\*\*

\* Getter for userPassword

\* @return

\*/

public String getUserPassword() { return userPassword; }

/\*\*

\* Setter for userPassword

\* @param userPassword

\*/

public void setUserPassword(String userPassword) { this.userPassword = userPassword; }

/\*\*

\* Getter for hasAccess

\* @return

\*/

public Boolean getHasAccess() { return hasAccess; }

/\*\*

\* Setter for hasAccess

\* @param hasAccess

\*/

public void setHasAccess(Boolean hasAccess) { this.hasAccess = hasAccess; }

/\*\*

\* Constructor with all attributes

\* @param userId

\* @param userGroup

\* @param login

\* @param userPassword

\* @param hasAccess

\*/

public Users(int userId, UserGroups userGroup, String login, String userPassword, Boolean hasAccess) {

this.userId = userId;

this.userGroup = userGroup;

this.login = login;

this.userPassword = userPassword;

this.hasAccess = hasAccess;

}

/\*\*

\* Constructor with id and login

\* @param userId

\* @param login

\*/

public Users(int userId, String login) {

this.userId = userId;

this.login = login;

}

}

## UserGroups

public class UserGroups {

private int userGroupId;

private String groupName;

/\*\*

\* Getter for userGroupId

\* @return

\*/

public int getUserGroupId() { userGroupId; }

/\*\*

\* Setter for userGroupId

\* @param userGroupId

\*/

public void setUserGroupId(int userGroupId) { this.userGroupId = userGroupId; }

/\*\*

\* Getter for groupName

\* @return

\*/

public String getGroupName() { return groupName; }

/\*\*

\* Setter for groupName

\* @param groupName

\*/

public void setGroupName(String groupName) { this.groupName = groupName; }

/\*\*

\* Constructor

\* @param userGroupId

\* @param groupName

\*/

public UserGroups(int userGroupId, String groupName) {

this.userGroupId = userGroupId;

this.groupName = groupName;

}

}

## Articles

public class Articles {

private int articleId;

private Users author;

private String text;

private Steps step;

private Users moderator;

private Users editor;

private Users corrector;

/\*\*

\* Getter for articleId

\* @return

\*/

public int getArticleId() { articleId; }

/\*\*

\* Setter for articleId

\* @param articleId

\*/

public void setArticleId(int articleId) { this.articleId = articleId; }

/\*\*

\* Getter for author

\* @return

\*/

public Users getAuthor() { return author; }

/\*\*

\* Setter for author

\* @param author

\*/

public void setAuthor(Users author) { this.author = author; }

/\*\*

\* Getter for text

\* @return

\*/

public String getText() { return text; }

/\*\*

\* Setter for text

\* @param text

\*/

public void setText(String text) { this.text = text; }

/\*\*

\* Getter for step

\* @return

\*/

public Steps getStep() { return step; }

/\*\*

\* Setter for step

\* @param step

\*/

public void setStep(Steps step) { this.step = step; }

/\*\*

\* Getter for moderator

\* @return

\*/

public Users getModerator() { return moderator; }

/\*\*

\* Setter for moderator

\* @param moderator

\*/

public void setModerator(Users moderator) { this.moderator = moderator; }

/\*\*

\* Getter for editor

\* @return

\*/

public Users getEditor() { return editor; }

/\*\*

\* Setter for editor

\* @param editor

\*/

public void setEditor(Users editor) { this.editor = editor; }

/\*\*

\* Getter for corrector

\* @return

\*/

public Users getCorrector() { return corrector; }

/\*\*

\* Setter for corrector

\* @param corrector

\*/

public void setCorrector(Users corrector) { this.corrector = corrector; }

/\*\*

\* Constructor

\* @param articleId

\* @param author

\* @param text

\* @param step

\* @param moderator

\* @param editor

\* @param corrector

\*/

public Articles(int articleId, Users author, String text, Steps step, Users moderator, Users editor, Users corrector) {

this.articleId = articleId;

this.author = author;

this.text = text;

this.step = step;

this.moderator = moderator;

this.editor = editor;

this.corrector = corrector;

}

}

## Steps

public class Steps {

private int stepId;

private String stepName;

/\*\*

\* Getter for stepId

\* @return

\*/

public int getStepId() { return stepId; }

/\*\*

\* Setter for stepId

\* @param stepId

\*/

public void setStepId(int stepId) { this.stepId = stepId; }

/\*\*

\* Getter for stepName

\* @return

\*/

public String getStepName() { return stepName; }

/\*\*

\* Setter for stepName

\* @param stepName

\*/

public void setStepName(String stepName) { this.stepName = stepName; }

/\*\*

\* Constructor

\* @param stepId

\* @param stepName

\*/

public Steps(int stepId, String stepName) {

this.stepId = stepId;

this.stepName = stepName;

}

}

## UsersDao

public class UsersDao {

private Connection connection;

/\*\*

\* Find user by login and password

\* @param login

\* @param password

\* @return

\* @throws SQLException

\*/

public Users findUser(String login, String password) throws SQLException {

Users user = null;

connection = ConnectionCreator.getConnection();

String sql = Resourcer.getString("query.user.find");

PreparedStatement statement = connection.prepareStatement(sql);

statement.setString(1, login);

statement.setString(2, password);

ResultSet result = statement.executeQuery();

while (result.next()) {

user = new Users(result.getInt("user\_id"),

new UserGroups(result.getInt("user\_group\_id"),

result.getString("group\_name")),

result.getString("login"),

result.getString("user\_password"),

result.getBoolean("has\_access"));

}

System.out.println(user);

if (user != null) {

if (!user.getHasAccess()) {

user = null;

}

}

connection.close();

return user;

}

/\*\*

\* Get list of users

\* @return

\* @throws SQLException

\*/

public List<Users> selectUsers() throws SQLException {

List<Users> users = new ArrayList<>();

connection = ConnectionCreator.getConnection();

Statement statement = connection.createStatement();

String sql = Resourcer.getString("query.user.select");

ResultSet result = statement.executeQuery(sql);

while (result.next()) {

users.add(new Users(result.getInt("user\_id"),

new UserGroups(result.getInt("user\_group\_id"),

result.getString("group\_name")),

result.getString("login"),

result.getString("user\_password"),

result.getBoolean("has\_access")));

}

connection.close();

return users;

}

/\*\*

\* Find user by id

\* @param userId

\* @return

\* @throws SQLException

\*/

public Users getUserById(int userId) throws SQLException {

Users user = null;

connection = ConnectionCreator.getConnection();

String sql = Resourcer.getString("query.user.getbyid");

PreparedStatement statement = connection.prepareStatement(sql);

statement.setInt(1, userId);

ResultSet result = statement.executeQuery();

while (result.next()) {

user = new Users(result.getInt("user\_id"),

new UserGroups(result.getInt("user\_group\_id"),

result.getString("group\_name")),

result.getString("login"),

result.getString("user\_password"),

result.getBoolean("has\_access"));

}

connection.close();

return user;

}

/\*\*

\* Update user's login and password

\* @param login

\* @param password

\* @param userId

\* @throws SQLException

\*/

public void updateUser(String login, String password, int userId) throws SQLException {

connection = ConnectionCreator.getConnection();

String sql = Resourcer.getString("query.user.update");

PreparedStatement statement = connection.prepareStatement(sql);

statement.setString(1, login);

statement.setString(2, password);

statement.setInt(3, userId);

statement.executeUpdate();

connection.close();

}

/\*\*

\* Add new user

\* @param roleId

\* @param login

\* @param password

\* @throws SQLException

\*/

public void addUser(int roleId, String login, String password) throws SQLException {

connection = ConnectionCreator.getConnection();

String sql = Resourcer.getString("query.user.add");

PreparedStatement statement = connection.prepareStatement(sql);

statement.setInt(1, roleId);

statement.setString(2, login);

statement.setString(3, password);

statement.executeUpdate();

connection.close();

}

/\*\*

\* Delete user

\* @param deletedUserId

\* @throws SQLException

\*/

public void deleteUser(int deletedUserId) throws SQLException {

connection = ConnectionCreator.getConnection();

String sql = Resourcer.getString("query.user.delete");

PreparedStatement statement = connection.prepareStatement(sql);

statement.setInt(1, deletedUserId);

statement.executeUpdate();

connection.close();

}

/\*\*

\* Get users of a specific group

\* @param userRole

\* @return

\* @throws SQLException

\*/

public List<Users> selectSpecificRole(String userRole) throws SQLException {

List<Users> specUsers = new ArrayList<>();

connection = ConnectionCreator.getConnection();

String sql = Resourcer.getString("query.user.selectspecific");

PreparedStatement statement = connection.prepareStatement(sql);

statement.setString(1, userRole);

ResultSet result = statement.executeQuery();

while (result.next()) {

specUsers.add(new Users(result.getInt("user\_id"),

result.getString("login")));

}

connection.close();

return specUsers;

}

/\*\*

\* Block and unblock user

\* @param blockedUser

\* @param access

\* @throws SQLException

\*/

public void blockUser(int blockedUser, boolean access) throws SQLException {

connection = ConnectionCreator.getConnection();

String sql = Resourcer.getString("query.user.block");

PreparedStatement statement = connection.prepareStatement(sql);

statement.setBoolean(1, access);

statement.setInt(2, blockedUser);

statement.executeUpdate();

connection.close();

}

}

## UserGroupDao

public class UserGroupDao {

Connection connection;

/\*\*

\* Get user's group name

\* @param name

\* @return

\* @throws SQLException

\*/

public UserGroups getUserGroupName(String name) throws SQLException {

UserGroups userGroup = null;

connection = ConnectionCreator.getConnection();

String sql = Resourcer.getString("query.userGroup.byName");

PreparedStatement statement = connection.prepareStatement(sql);

statement.setString(1, name);

ResultSet result = statement.executeQuery();

while (result.next()) {

userGroup = new UserGroups(result.getInt("user\_group\_id"),

result.getString("group\_name"));

}

connection.close();

return userGroup;

}

/\*\*

\* Get list of user's groups

\* @return

\* @throws SQLException

\*/

public List<UserGroups> selectRoles() throws SQLException {

List<UserGroups> roles = new ArrayList<>();

connection = ConnectionCreator.getConnection();

Statement statement = connection.createStatement();

String sql = Resourcer.getString("query.userGroup.select");

ResultSet result = statement.executeQuery(sql);

while (result.next()) {

roles.add(new UserGroups(result.getInt("user\_group\_id"),

result.getString("group\_name")));

}

connection.close();

return roles;

}

}

## ArticleDao

public class ArticleDao {

private Connection connection;

/\*\*

\* Add new article

\* @param author

\* @param text

\* @throws SQLException

\*/

public void addArticle(int author, String text) throws SQLException {

connection = ConnectionCreator.getConnection();

String sql = Resourcer.getString("query.article.add");

PreparedStatement statement = connection.prepareStatement(sql);

statement.setInt(1, author);

statement.setString(2, text);

statement.executeUpdate();

connection.close();

}

/\*\*

\* Get list of articles

\* @param userId

\* @param role

\* @return

\* @throws SQLException

\*/

public List<Articles> selectArticles(int userId, String role) throws SQLException {

List<Articles> articles = new ArrayList<>();

String sql = null;

Statement statement;

PreparedStatement pstatement;

ResultSet result = null;

connection = ConnectionCreator.getConnection();

if (role.equals("Administrator")) {

statement = connection.createStatement();

sql = Resourcer.getString("query.articles.selectAdmin");

result = statement.executeQuery(sql);

} else if (role.equals("Author")) {

statement = connection.createStatement();

sql = Resourcer.getString("query.articles.selectdone");

result = statement.executeQuery(sql);

} else {

switch (role) {

case "Moderator":

sql = Resourcer.getString("query.articles.selectModer");

break;

case "Editor":

sql = Resourcer.getString("query.articles.assigneditor");

break;

case "Corrector":

sql = Resourcer.getString("query.articles.assigncorrector");

break;

default:

break;

}

pstatement = connection.prepareStatement(sql);

pstatement.setInt(1, userId);

result = pstatement.executeQuery();

}

while (result.next()) {

articles.add(new Articles(result.getInt("article\_id"),

new Users(result.getInt("author"),

result.getString("author\_login")),

result.getString("text"),

new Steps(result.getInt("step"),

result.getString("step\_role")),

new Users(result.getInt("moderator"),

result.getString("moder")),

new Users(result.getInt("editor"),

result.getString("edit")),

new Users(result.getInt("corrector"),

result.getString("correct"))));

}

connection.close();

return articles;

}

/\*\*

\* Add attribute "Moderator" to an article

\* @param article

\* @param moder

\* @throws SQLException

\*/

public void assignModer(String article, String moder) throws SQLException {

connection = ConnectionCreator.getConnection();

String sql = Resourcer.getString("query.article.assignmoder");

PreparedStatement statement = connection.prepareStatement(sql);

statement.setString(1, moder);

statement.setInt(2, Integer.parseInt(article));

statement.executeUpdate();

connection.close();

}

/\*\*

\* Delete article

\* @param deletedArticle

\* @throws SQLException

\*/

public void deleteArticle(int deletedArticle) throws SQLException {

connection = ConnectionCreator.getConnection();

String sql = Resourcer.getString("query.article.delete");

PreparedStatement statement = connection.prepareStatement(sql);

statement.setInt(1, deletedArticle);

statement.executeUpdate();

connection.close();

}

/\*\*

\* Add attributes "Editor" and "Corrector" to an article

\* @param article

\* @param editor

\* @param corrector

\* @throws SQLException

\*/

public void assignEditorCorrector(String article, String editor, String corrector) throws SQLException {

connection = ConnectionCreator.getConnection();

String sql = Resourcer.getString("query.article.assigneditorcorrector");

PreparedStatement statement = connection.prepareStatement(sql);

statement.setString(1, editor);

statement.setString(2, corrector);

statement.setInt(3, Integer.parseInt(article));

statement.executeUpdate();

connection.close();

}

/\*\*

\* Update article's text

\* @param text

\* @param articleId

\* @throws SQLException

\*/

public void updateArticle(String text, int articleId) throws SQLException {

connection = ConnectionCreator.getConnection();

String sql = Resourcer.getString("query.article.update");

PreparedStatement statement = connection.prepareStatement(sql);

statement.setString(1, text);

statement.setInt(2, articleId);

statement.executeUpdate();

connection.close();

}

/\*\*

\* Get article by its id

\* @param articleId

\* @return

\* @throws SQLException

\*/

public Articles getArticleById(int articleId) throws SQLException {

Articles article = null;

connection = ConnectionCreator.getConnection();

String sql = Resourcer.getString("query.article.getbyid");

PreparedStatement statement = connection.prepareStatement(sql);

statement.setInt(1, articleId);

ResultSet result = statement.executeQuery();

while (result.next()) {

article = new Articles(result.getInt("article\_id"),

new Users(result.getInt("author"),

result.getString("author\_login")),

result.getString("text"),

new Steps(result.getInt("step"),

result.getString("step\_role")),

new Users(result.getInt("moderator"),

result.getString("moder")),

new Users(result.getInt("editor"),

result.getString("edit")),

new Users(result.getInt("corrector"),

result.getString("correct")));

}

connection.close();

return article;

}

/\*\*

\* Change article's step

\* @param article

\* @throws SQLException

\*/

public void assignNext(Articles article) throws SQLException {

connection = ConnectionCreator.getConnection();

String articleStep = article.getStep().getStepName();

String sql = Resourcer.getString("query.articles.assignnext");

PreparedStatement statement = connection.prepareStatement(sql);

String nextStep = null;

switch (articleStep) {

case "Editing":

nextStep = "Correcting";

break;

case "Correcting":

nextStep = "Ready";

break;

default:

break;

}

statement.setString(1, nextStep);

statement.setInt(2, article.getArticleId());

statement.executeUpdate();

connection.close();

}

}

## StepsDao

public class StepsDao {

private Connection connection;

/\*\*

\* Get step by its id

\* @param stepId

\* @return

\* @throws SQLException

\*/

public Steps getStepById(int stepId) throws SQLException {

Steps step = null;

connection = ConnectionCreator.getConnection();

String sql = Resourcer.getString("query.steps.getbyid");

PreparedStatement statement = connection.prepareStatement(sql);

statement.setInt(1, stepId);

ResultSet result = statement.executeQuery();

while (result.next()) {

step = new Steps(result.getInt("step\_id"),

result.getString("step\_role"));

}

connection.close();

return step;

}

}

# Библиографический список

1. Блинов И. Н., Романчик В. С. Java. Методы программирования: учеб.-метод. пособие. - Минск: Изд-во "Четыре четверти", 2013. - 896 с.