**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

#### Учреждение образования

«Гомельский государственный университет имени Франциска Скорины»

##### **Математический факультет**

Кафедра математических проблем управления

Допущена к защите

Зав. кафедрой \_\_\_\_\_\_\_ Смородин В. С.

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 201\_\_\_\_\_ г.

**РАЗРАБОТКА СИСТЕМЫ АВТОМАТИЗАЦИИ РАБОТЫ**

**БИБЛИОТЕКИ ВЫСШЕГО УЧЕБНОГО ЗАВЕДЕНИЯ**

**ДИПЛОМНАЯ РАБОТА**

Исполнитель:

студент группы ПОИТ-51 \_\_\_\_\_\_\_\_\_ Чураков Н. С.

Научный руководитель:

ассистент кафедры МПУ \_\_\_\_\_\_\_\_\_ Короткевич Л. И.

Рецензент: \_\_\_\_\_\_\_\_\_ Бондарева В. В.

к. т. н. доцент кафедры

информационно-вычислительных

систем БТЭУ

Гомель 2011

**СОДЕРЖАНИЕ**

ВВЕДЕНИЕ 3

1 НАЗНАЧЕНИЕ РЕАЛИЗОВАННОГО ПРИЛОЖЕНИЯ 4

1.1 Функции разработанного приложения 4

1.2 Описание ролей 4

1.3 Функциональная схема работы приложения 6

2 ОПИСАНИЕ СРЕДСТВ РАЗРАБОТКИ ПРИЛОЖЕНИЯ 7

2.1 Система управления реляционными базами данных Microsoft SQL Server 7

2.2 Веб-сервер Apache Tomcat 8

2.3 Технология J2EE 8

2.4 Среда разработки Eclipse 9

2.5 Фреймворк Struts 2 9

2.6 Язык разметки HTML 10

2.7 Каскадные таблицы стилей 10

2.8 Язык JavaScript 12

2.9 Технология AJAX 14

2.10 Библиотека jQuery 14

2.11 Система управления версиями 15

2.12 Инструменты отладки и тестирования отображаемых страниц 17

3 СТРУКТУРА БАЗЫ ДАННЫХ ПРИЛОЖЕНИЯ 18

4 СХЕМА ДИАЛОГА ПОЛЬЗОВАТЕЛЯ С РАЗРАБОТАННЫМ ПРИЛОЖЕНИЕМ 22

4.1 Регистрация на сайте библиотеки высшего учебного заведения 22

4.2 Описание интерфейса для администратора 24

4.2.1 Администрирование пользователей 24

4.2.2 Редактирование справочников 26

4.2.3 Информация о библиотеке высшего учебного заведения 27

4.2.4 Статистика работы библиотеки высшего учебного заведения 28

4.3 Описание интерфейса для пользователя 28

4.3.1 Администрирование профиля пользователя 28

4.3.2 Обзор реальных книг 29

4.3.3 Обзор электронных книг 30

4.3.4 Просмотр персональных задолженностей 32

4.4 Описание интерфейса для библиотекаря 32

4.4.1 Редактирование описаний книг 33

4.4.2 Редактирование реальных книг 34

4.4.3 Редактирование электронных книг 36

4.4.4 Выдача книг читателям 37

4.4.5 Возврат книг в библиотеку 40

4.4.6 Отчет по работе библиотекарей 41

ЗАКЛЮЧЕНИЕ 43

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 44

ПРИЛОЖЕНИЕ А ЛИСТИНГ ПРОГРАММЫ 45

**ВВЕДЕНИЕ**

Современный мир характеризуется наличием огромного объёма накопленной информации. Развитие науки и экономики, рост численности населения способствуют прогрессивному увеличению объёма информации, а также усложнению ее структуры и взаимосвязей. Для организации и упрощения обработки информации создаются информационные системы. Неотъемлемой частью любой информационной системы является база данных, в которой хранятся сведения о конкретных объектах реального мира.

Для решения проблем, связанных с работой со сложными информационными системами, часто создаются специализированные и многофункциональные веб-приложения. В таком приложении [клиентом](http://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B8%D0%B5%D0%BD%D1%82_%28%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0%29) выступает [браузер](http://ru.wikipedia.org/wiki/%D0%91%D1%80%D0%B0%D1%83%D0%B7%D0%B5%D1%80), а [сервером](http://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%80%D0%B2%D0%B5%D1%80_%28%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5%29) выступает [веб-сервер](http://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80). Логика веб-приложения распределена между сервером и клиентом, хранение данных осуществляется на сервере базы данных, обмен информацией происходит по [сети](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D1%81%D0%B5%D1%82%D1%8C). Одним из преимуществ такого подхода является то, что клиенты не зависят от конкретной операционной системы пользователя, по этому веб-приложения являются межплатформенными сервисами.

На любом предприятии актуальной является проблема рациональной организации и удобного использования имеющейся информации. Решение данной проблемы позволит уменьшить затраченное время на поиск и обработку необходимой информации. Так автоматизация библиотек позволяет работникам библиотечной сферы избавиться от многих рутинных процессов, что обеспечивает наиболее быстрое и качественное обслуживание читателей.

В данной дипломной работе были разработаны компоненты системы «Библиотека высшего учебного заведения». В ходе разработки, была создана база данных системы, разработаны программные компоненты серверной и клиентской части. Реализованная система обеспечивает разграничение возможностей для различных ролей пользователей, что позволяет им успешно выполнять свои функции в системе. Каждая роль строго определяет перечень функций, которые может выполнять пользователь в системе. Разработанное приложение поддерживает работу, как с реальными книгами, так и с электронными версиями книг. Данная особенность позволяет пользователям системы скачивать электронные книги, что уменьшает очереди на реальные книги и упрощает доступ к материалам библиотеки.

Данная дипломная работа состоит из четырех разделов.

В первом разделе было рассмотрено назначение и возможности разработанных программных средств.

Во втором разделе содержится описание средств разработки приложения и использованных технологий.

В третьем разделе описана структура базы данных приложения.

В четвёртом разделе описан интерфейс диалога пользователя с разработанными программными средствами.

**1 НАЗНАЧЕНИЕ РЕАЛИЗОВАННОГО ПРИЛОЖЕНИЯ**

В данной работе была поставлена следующая задача: разработка системы автоматизации работы библиотеки высшего учебного заведения. При выполнении этой дипломной работы было реализовано веб-приложение, позволяющее автоматизировать функционирование библиотеки.

**1.1 Функции разработанного приложения**

Разработанная система предоставляет пользователю выполнение следующих функций:

1. Работа с реальными книгами. Данная возможность включает учет, редактирование реальных книг.
2. Выдача и возврат реальных книг, позволяет ускорить книгооборот и снизить время на рутинные операции библиотекаря.
3. Поиск и сортировка каталога книг, что способствует уменьшению временных затрат при осуществлении поиска нужной литературы.
4. Организация очереди на книги. Эта функция позволяет уменьшить время ожидания книги для читателя.
5. Работа с электронными книгами. Реализованная приложение поддерживает возможность организации хранилища для электронных книг. После загрузки книги в систему, каждый пользователь может найти и скачать подходящую книгу.
6. Поддержка общения пользователей через комментарии к электронным книгам. Данная характеристика позволяет обмениваться мнениями по поводу определенной книги.
7. Возможность просматривать персональный и общий список задолженностей. Облегчает поиск должников и выданной литературы.
8. Предоставление статистических данных в виде графиков и отчетов по работе библиотеки, позволяет оценить результаты работы библиотеки.
9. Поддержка русского и английского языка для отображаемого интерфейса.
10. Возможность настройки доступа к книгам для разных категорий пользователей.

**1.2 Описание ролей**

Все пользователи реализованного приложения разделяются на следующие роли: «Новый», «Пользователь», «Библиотекарь», «Администратор». Каждая роль определяет перечень функции, которые может выполнять в системе пользователь.

Пользователь получает роль «Новый» сразу же после регистрации на сайте. Новый пользователь не имеет доступа к основным элементам сайта библиотеки и должен ждать активации его учетной записи администратором.

Роль «Администратор» введена для выполнения операций администрирования пользователей. В полномочия администратора входит управление учетными записями пользователей. Главными же задачами являются активация новых пользователей и изменение ролей, категорий. Категория обеспечивает разделение доступа к электронным и реальным книгам. Также администратор может произвести администрирование всех справочников. Функции администратора представлены на рисунке 1.1

Администратор

Администрирование пользователей

Администрирование справочников

Просмотр общей статистики

Рисунок 1.1 – Функции администратора

Роль «Пользователь» предназначена для обычных пользователей. Пользователь может видеть только те книги, которые разрешены для его категории. Пользователь может производить редактирование своего профиля. Также осуществлять поиск по каталогу электронных и реальных книг, становится в очередь на книгу, оставлять комментарии, скачивать электронные книги, просматривать персональные задолженности и общую статистика по библиотеке. На рисунке 1.2 показаны функции пользователя.

Пользователь

Редактирование профиля

Просмотр списка реальных книг

Обзор персональных задолженностей

Добавление и удаление в очередь на книгу

Просмотр списка электронных книг

Работа с комментариями

Просмотр общей статистики

Рисунок 1.2 – Функции пользователя

Роль «Библиотекарь» введена для обеспечения возможности библиотекарю выполнять свои функции в приложении. Данная роль расширяет функциональные возможности роли «Пользователь» следующими функциями: модификация информации о книгах, удаление комментариев, закачка новых книг, редактирование очереди на книгу, выдача и возврат книг, просмотр всех задолженностей, настройка видимости для книги, просмотр отчета по работе библиотекарей. Дополнительные функции библиотекаря приведены на рисунке 1.3.

Библиотекарь

Модификация информации о книгах

Удаление всех комментариев

Обзор всех задолженностей

Закачка электронных книг

Редактирование очередей на книги

Выдача и возврат книг

Просмотр отчета по деятельности библиотекарей

Настройка видимости для книги

Рисунок 1.3 – Дополнительные функции библиотекаря

**1.3 Функциональная схема работы приложения**

Общая функциональная схема работы веб-приложения, автоматизирующего работу библиотеки высшего учебного заведения, представлена на рисунке 1.4.

Веб-приложение

Пользователь

Администратор

Библиотекарь

Веб-сервер

Сервер базы данных

Рисунок 1.4 – Функциональная схема работы приложения

**2 ОПИСАНИЕ СРЕДСТВ РАЗРАБОТКИ ПРИЛОЖЕНИЯ**

В ходе выполнения данной дипломной работы были использованы различные средства разработки и технологии. Так для реализации серверной части был использован Microsoft SQL Server. Веб-приложение был развернуто на Apache Tomcat веб-сервере. Для программирования логики был использован фреймворк Struts 2, в основе которого лежит технология J2EE. Реализация веб-приложения проводилась в среде Eclipse. Для создания клиентской части были применены следующие технологии: HTML, CSS, JavaScript, jQuery, Ajax. При работе над проектом также был использован инструмент управления версиями tortoiseSVN. Для отладки и тестирования отображаемых страниц широко применялись инструменты браузеров FireFox и Internet Explorer. Это Firebug и Internet Explorer Developer соответственно. Далее упомянутые технологии будут описаны более детально.

**2.1** [**Система управления реляционными базами данных**](http://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BB%D1%8F%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D0%A1%D0%A3%D0%91%D0%94) **Microsoft SQL Server**

Microsoft SQL Server – это законченное решение для управления и анализа данных, позволяющее оперативно развертывать масштабируемые приложения нового поколения. SQL Server – ключевой компонент поддержки электронной коммерции, интерактивных деловых приложений и хранилищ данных, обеспечивающий масштабируемость, необходимую для поддержки растущих, динамических сред. В SQL Server предусмотрена широчайшая поддержка XML и других форматов, используемых в Интернете, функций производительности и доступности, гарантирующих своевременное решение поставленных задач, а также развитой функциональности управления и настройки, позволяющей автоматизировать выполнение рутинных задач и снизить совокупную стоимость владения.

SQL Server является реляционной СУБД, которая использует язык Transact-SQL для пересылки сообщений между компьютером клиента и компьютером, на котором работает SQL Server. Реляционная СУБД состоит из механизма баз данных, собственно баз данных и приложений, необходимых для управления данными и компонентами реляционной СУБД. Реляционная СУБД организует данные в виде связанных строк и столбцов, составляющих базу данных. Реляционная СУБД отвечает за поддержку структуры базы данных и решает следующие задачи:

* 1. поддерживает связи между данными в базе;
  2. гарантирует корректное хранение данных и выполнение правил, регламентирующих связи между ними;
  3. восстанавливает данные после аварии системы, переводя их в согласованное состояние, зафиксированное до сбоя.

Более детальное описание данной СУБД приведено в источниках [1-6].

**2.2 Веб-сервер Apache Tomcat**

Apache Tomcat – это [программа](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0)-[контейнер сервлетов](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D1%82%D0%B5%D0%B9%D0%BD%D0%B5%D1%80_%D1%81%D0%B5%D1%80%D0%B2%D0%BB%D0%B5%D1%82%D0%BE%D0%B2), написанная на языке [Java](http://ru.wikipedia.org/wiki/Java) и преобразующие [сервлет](http://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%80%D0%B2%D0%BB%D0%B5%D1%82)ы в [Java Server Pages](http://ru.wikipedia.org/wiki/JavaServer_Pages) ([JSP](http://ru.wikipedia.org/wiki/JSP)), которые являются стандартами для разработки [веб-приложений](http://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5) на языке [Java](http://ru.wikipedia.org/wiki/Java). Tomcat позволяет запускать [веб-приложения](http://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5), содержит ряд программ для конфигурирования.

Разработка и поддержка Tomcat осуществляется фондом [Apache Software Foundation](http://ru.wikipedia.org/wiki/Apache_Software_Foundation) и добровольцами. Пользователи имеют свободный доступ к исходным кодам и бинарным файлам Tomcat.

Сервер содержит следующие компоненты: Catalina, Сoyote, Jasper. Catalina – [контейнер сервлетов](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D1%82%D0%B5%D0%B9%D0%BD%D0%B5%D1%80_%D1%81%D0%B5%D1%80%D0%B2%D0%BB%D0%B5%D1%82%D0%BE%D0%B2) Tomcat’а. Coyote – компонент стека HTTP Tomcat’а, который поддерживает протокол HTTP 1.1 для веб-серверов или контейнера приложений. Coyote прослушивает входящие соединения на определённом TCP порте сервера и пересылает запросы в механизм Tomcat для обработки запросов и отправляет ответ назад запрашивающему клиенту. Jasper – механизм JSP Tomcat’а. Jasper анализирует JSP-файлы, чтобы компилировать их в Java код, как сервлеты (которые могут быть обработаны с помощью Catalina). Во время выполнения, Jasper может автоматически обнаруживать изменения JSP-файла и перекомпилировать его. Инструкцию по настройке и руководство по использованию данного веб-сервера можно найти в источнике [7].

**2.3 Технология J2EE**

Java Platform Enterprise Edition, сокращенно Java EE – набор спецификаций и соответствующей документации для языка [Java](http://ru.wikipedia.org/wiki/Java), описывающей архитектуру серверной платформы для задач средних и крупных предприятий.

Спецификации детализированы настолько, чтобы обеспечить переносимость программ с одной реализации платформы на другую. Основная цель спецификаций – обеспечить [масштабируемость](http://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%81%D1%88%D1%82%D0%B0%D0%B1%D0%B8%D1%80%D1%83%D0%B5%D0%BC%D0%BE%D1%81%D1%82%D1%8C) приложений и целостность данных во время работы системы. J2EE во многом ориентирована на использование её через [веб](http://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1) как в [интернете](http://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82), так и в [локальных сетях](http://ru.wikipedia.org/wiki/%D0%9B%D0%92%D0%A1). Вся [спецификация](http://ru.wikipedia.org/wiki/%D0%A1%D0%BF%D0%B5%D1%86%D0%B8%D1%84%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D1%8F) создаётся и утверждается через [JCP](http://ru.wikipedia.org/wiki/JCP) (Java Community Process) в рамках инициативы [Sun Microsystems Inc](http://ru.wikipedia.org/wiki/Sun_Microsystems).

J2EE является промышленной технологией и в основном используется в высокопроизводительных проектах, в которых необходима надежность, масштабируемость, гибкость.

Популярности J2EE также способствует то, что Sun предлагает бесплатный комплект разработки, позволяющий предприятиям разрабатывать свои системы, не тратя больших средств. В этот комплект входит сервер приложений с лицензией для разработки. Для углубленного освоения данной технологии были изучены источники [8-9].

**2.4 Среда разработки Eclipse**

Eclipse – [свободн](http://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D0%BE%D0%B1%D0%BE%D0%B4%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5)о распространяемая [интегрированная среда разработки](http://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D1%80%D0%B5%D0%B4%D0%B0_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8) [модульных](http://ru.wikipedia.org/wiki/Plug-in_Development_Environment) кроссплатформенных приложений. Развивается и поддерживается [Eclipse Foundation](http://ru.wikipedia.org/wiki/Eclipse_Foundation).

Eclipse – в первую очередь полноценная Java IDE, нацеленная на групповую разработку: среда интегрирована с системами управления версиями. В силу бесплатности и высокого качества, Eclipse во многих организациях является корпоративным стандартом для разработки приложений.

Второе назначение Eclipse – служить платформой для разработки новых расширений, чем он и завоевал популярность: любой разработчик может расширить Eclipse своими модулями. Множество расширений дополняет среду Eclipse менеджерами для работы с базами данных, серверами приложений и др.

Eclipse написана на Java, потому является платформо-независимым продуктом, за исключением библиотеки SWT, которая разрабатывается для всех распространённых платформ. Библиотека SWT используется вместо стандартной для Java библиотеки [Swing](http://ru.wikipedia.org/wiki/Swing). Она полностью опирается на нижележащую платформу (операционную систему), что обеспечивает быстроту и натуральный внешний вид пользовательского интерфейса, но иногда вызывает на разных платформах проблемы совместимости и устойчивости приложений.

**2.5 Фреймворк Struts 2**

Struts 2 – фреймворк с открытым исходным кодом для создания [Java EE](http://ru.wikipedia.org/wiki/Java_EE) веб-приложений. Основывается на [Java Servlet API](http://ru.wikipedia.org/w/index.php?title=Java_Servlet_API&action=edit&redlink=1) и расширяет его, в архитектурном плане реализует паттерн Model-View-Controller. Фреймворк создан Крейгом Мак Кланаханом и передан [Apache Foundation](http://ru.wikipedia.org/w/index.php?title=Apache_Foundation&action=edit&redlink=1) в мае [2000 года](http://ru.wikipedia.org/wiki/2000_%D0%B3%D0%BE%D0%B4). Сначала находился в составе [Apache Jakarta Project](http://ru.wikipedia.org/wiki/Apache_Jakarta_Project) и был известен как Jakarta Struts. С [2005 года](http://ru.wikipedia.org/wiki/2005_%D0%B3%D0%BE%D0%B4) является проектом Apache верхнего уровня.

В обычном [Java EE](http://ru.wikipedia.org/wiki/Java_Platform,_Enterprise_Edition) веб-приложении клиент, как правило, отправляет информацию на сервер из [формы](http://ru.wikipedia.org/w/index.php?title=%D0%A4%D0%BE%D1%80%D0%BC%D0%B0_%28%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%29&action=edit&redlink=1). Введённые данные передаются [сервлету](http://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%80%D0%B2%D0%BB%D0%B5%D1%82), который обрабатывает её, при необходимости взаимодействуя с базой данных, и формирует [HTML](http://ru.wikipedia.org/wiki/HTML) ответ. Для больших проектов, такой подход является неадекватным, так как объединяет [бизнес логику](http://ru.wikipedia.org/wiki/%D0%91%D0%B8%D0%B7%D0%BD%D0%B5%D1%81-%D0%BB%D0%BE%D0%B3%D0%B8%D0%BA%D0%B0) и логику отображения, что делает поддержку чрезвычайно сложной.

Struts был создан для того, чтобы чётко разделить модель, представление и контроллер. Struts предоставляет стандартный контроллер – сервлет под именем Action Servlet и различные средства для создания страниц представления. Разработчик веб-приложения отвечает за написание кода модели и создание конфигурационного файла struts.xml, который связывает воедино модель, представление и контроллер.

Запросы от клиента передаются контроллеру в виде «Actions», определённых в конфигурационном файле. Когда контроллер получает такой запрос, он передаёт его соответствующему Action классу. Последний взаимодействует с кодом модели и возвращает контроллеру, строку, определяющую страницу для отправления клиенту. Информация передаётся между моделью и представлением в виде особых [Java Beans](http://ru.wikipedia.org/wiki/JavaBeans). Богатая библиотека тегов позволяет получать данные из бинов и записывать их без Java кода.

Struts также поддерживает [i18n](http://ru.wikipedia.org/wiki/I18n) (интернационализацию), облегчает валидацию данных полученных из веб-формы и предоставляет механизм создания шаблонов под названием «Tiles», который кроме всего прочего позволяет наследовать веб-страницы. При изучении этого фрейморка активно использовался источник [10].

**2.6 Язык разметки HTML**

Hyper Text Markup Language – стандартный [язык разметки](http://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D1%80%D0%B0%D0%B7%D0%BC%D0%B5%D1%82%D0%BA%D0%B8) документов во [Всемирной паутине](http://ru.wikipedia.org/wiki/%D0%92%D1%81%D0%B5%D0%BC%D0%B8%D1%80%D0%BD%D0%B0%D1%8F_%D0%BF%D0%B0%D1%83%D1%82%D0%B8%D0%BD%D0%B0). Большинство [веб-страниц](http://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D1%82%D1%80%D0%B0%D0%BD%D0%B8%D1%86%D0%B0) создаются при помощи языка HTML или [XHTML](http://ru.wikipedia.org/wiki/XHTML). Язык HTML интерпретируется [браузером](http://ru.wikipedia.org/wiki/%D0%91%D1%80%D0%B0%D1%83%D0%B7%D0%B5%D1%80) и отображается в виде документа, в удобной для человека форме.

Язык HTML был разработан [британским](http://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%BB%D0%B8%D0%BA%D0%BE%D0%B1%D1%80%D0%B8%D1%82%D0%B0%D0%BD%D0%B8%D1%8F) учёным [Тимом Бернерсом-Ли](http://ru.wikipedia.org/wiki/%D0%91%D0%B5%D1%80%D0%BD%D0%B5%D1%80%D1%81-%D0%9B%D0%B8,_%D0%A2%D0%B8%D0%BC) приблизительно в [1989](http://ru.wikipedia.org/wiki/1989) - [1991 годах](http://ru.wikipedia.org/wiki/1991_%D0%B3%D0%BE%D0%B4) в стенах [Европейского совета по ядерным исследованиям](http://ru.wikipedia.org/wiki/CERN) в [Женеве](http://ru.wikipedia.org/wiki/%D0%96%D0%B5%D0%BD%D0%B5%D0%B2%D0%B0) ([Швейцария](http://ru.wikipedia.org/wiki/%D0%A8%D0%B2%D0%B5%D0%B9%D1%86%D0%B0%D1%80%D0%B8%D1%8F)). HTML создавался как язык для обмена научной и технической документацией, пригодный для использования людьми, не являющимися специалистами в области [вёрстки](http://ru.wikipedia.org/wiki/%D0%92%D1%91%D1%80%D1%81%D1%82%D0%BA%D0%B0). HTML успешно был упрощен путём определения небольшого набора структурных и [семантических](http://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D0%BC%D0%B0%D0%BD%D1%82%D0%B8%D0%BA%D0%B0) элементов - [дескрипторов](http://ru.wikipedia.org/wiki/%D0%94%D0%B5%D1%81%D0%BA%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D1%80). Дескрипторы также часто называют «[тегами](http://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%B3_%28%D1%8F%D0%B7%D1%8B%D0%BA%D0%B8_%D1%80%D0%B0%D0%B7%D0%BC%D0%B5%D1%82%D0%BA%D0%B8%29)». С помощью HTML можно легко создать относительно простой, но красиво оформленный документ. Помимо упрощения структуры документа, в HTML внесена поддержка гипертекста. [Мультимедийные](http://ru.wikipedia.org/wiki/%D0%9C%D1%83%D0%BB%D1%8C%D1%82%D0%B8%D0%BC%D0%B5%D0%B4%D0%B8%D0%B0) возможности были добавлены позже. Изначально язык HTML был задуман и создан как средство структурирования и форматирования документов без их привязки к средствам воспроизведения. В идеале, текст с разметкой HTML должен был без стилистических и структурных искажений воспроизводиться на оборудовании с различной технической оснащённостью (цветной экран современного компьютера, монохромный экран органайзера, ограниченный по размерам экран мобильного телефона или устройства и программы голосового воспроизведения текстов). Однако современное применение HTML очень далеко от его изначальной задачи. Многие приемы оформления страниц реализованного приложения были взяты из источника [11].

**2.7 Каскадные таблицы стилей**

Cascading Style Sheets используется для задания цветов, шрифтов, расположения и других аспектов представления документа. Основной целью разработки CSS являлось разделение содержимого и представления документа. Это разделение может увеличить доступность документа, предоставить большую гибкость и возможность управления его представлением, а также уменьшить сложность и повторяемость в структурном содержимом. Кроме того, CSS позволяет представлять один и тот же документ в различных стилях или методах вывода, таких как экранное представление, печать, чтение голосом.

CSS при отображении страницы может быть взята из различных источников:

- Авторские стили (информация стилей, предоставляемая автором страницы).

- Пользовательские стили.

- Стиль браузера.

Таблица стилей состоит из набора правил. Каждое правило, в свою очередь, состоит из одного или нескольких селекторов, разделённых запятыми, и блока определений. Блок определений же обрамляется фигурными скобками, и состоит из набора свойств и их значений.

Схематически это можно показать так:

селектор, селектор {

свойство: значение;

свойство: значение;

свойство: значение;

}

Стандарт CSS определяет приоритеты, в порядке которых применяются правила стилей, если для какого-то элемента подходят свойства нескольких правил одновременно. Это называется «каскадом», в котором для свойств рассчитываются приоритеты или «веса», что делает результаты предсказуемыми.

Приоритеты рассчитываются таким образом (от большего к меньшему):

- свойство задано при помощи !important;

- стиль прописан напрямую в теге;

- количество идентификаторов (#id) в селекторе (чем больше, тем больше приоритет);

- количество классов (.class) и псевдоклассов (:pseudoclass) в селекторе;

- количество имён тегов в селекторе.

Кроме того, имеет значение относительный порядок расположения свойств: свойство, указанное позже, имеет приоритет.

До появления CSS оформление веб-страниц осуществлялось непосредственно внутри содержимого документа. Однако с появлением CSS стало возможным принципиальное разделение содержания и представления документа. За счёт этого нововведения стало возможным лёгкое применение единого стиля оформления для массы схожих документов, а также быстрое изменение этого оформления.

Преимущества:

- несколько дизайнов страницы для разных устройств просмотра. Например, на экране дизайн будет рассчитан на большую ширину, во время печати меню не будет выводиться, а на КПК и сотовом телефоне меню будет следовать за содержимым;

- уменьшение времени загрузки страниц сайта за счет переноса правил представления данных в отдельный CSS-файл. В этом случае браузер загружает только структуру документа и данные, хранимые на странице, а представление этих данных загружается браузером только один раз и могут быть закешированы;

- простота последующего изменения дизайна. Не нужно править каждую страницу, а лишь изменить CSS-файл;

- дополнительные возможности оформления. Например, с помощью CSS-вёрстки можно сделать блок текста, который остальной текст будет обтекать (например для меню) или сделать так, чтобы меню было всегда видно при прокрутке страницы.

Недостатки:

- различное отображение вёрстки в различных браузерах, которые по- разному интерпретируют одни и те же данные CSS;

- часто встречающаяся необходимость на практике исправлять не только один CSS-файл, но и теги HTML, которые сложным и ненаглядным способом связаны с селекторами CSS, что иногда сводит на нет простоту применения единых файлов стилей и значительно удлиняет время редактирования и тестирования. Особенности каскадных таблиц стилей были изучены на основе материалов из источника [12].

**2.8 Язык JavaScript**

JavaScript – это объектно-ориентированный скриптовый язык программирования. JavaScript обычно используется как встраиваемый язык для программного доступа к объектам приложений. Наиболее широкое применение находит в браузерах как язык сценариев для придания интерактивности веб-страницам.

Основные архитектурные черты: динамическая типизация, слабая типизация, автоматическое управление памятью, прототипное программирование, функции как объекты первого класса.

На JavaScript оказали влияние многие языки, при разработке была цель сделать язык похожим на Java, но при этом лёгким для использования непрограммистами. Языком JavaScript не владеет какая-либо компания или организация, что отличает его от ряда языков программирования, используемых в веб-разработке.

JavaScript обладает рядом свойств объектно-ориентированного языка, но реализованное в языке прототипирование обуславливает отличия в работе с объектами по сравнению с традиционными объектно-ориентированными языками. Кроме того, JavaScript имеет ряд свойств, присущих функциональным языкам – функции как объекты первого класса, объекты как списки, анонимные функции, замыкания – что придаёт языку дополнительную гибкость.

Несмотря на схожий с Си синтаксис, JavaScript по сравнению с языком Си имеет коренные отличия:

- объекты, с возможностью интроспекции;

- функции как объекты первого класса;

- автоматическое приведение типов;

- автоматическая сборка мусора;

- анонимные функции.

В языке отсутствуют такие полезные вещи, как:

- модульная система: JavaScript не предоставляет возможности управлять зависимостями и изоляцией областей видимости;

- стандартная библиотека: в частности, отсутствует интерфейс программирования приложений по работе с файловой системой, управлению потоками ввода/вывода, базовых типов для бинарных данных;

- стандартные интерфейсы к веб-серверам и базам данных;

- система управления пакетами, которая бы отслеживала зависимости и автоматически устанавливала их.

Структурно JavaScript можно представить в виде объединения трёх чётко различимых друг от друга частей:

- ядро (ECMA Script);

- объектная модель браузера (Browser Object Model или BOM);

- объектная модель документа (Document Object Model или DOM).

Если рассматривать Java Script в отличных от браузера окружениях, то объектная модель браузера и объектная модель документа могут не поддерживаться.

Объектную модель документа иногда рассматривают как отдельную от JavaScript сущность, что согласуется с определением DOM как независимого от языка интерфейса документа. В противоположность этому ряд авторов находят BOM и DOM тесно взаимосвязанными.

Объектная модель браузера – браузер специфичная часть языка, являющаяся прослойкой между ядром и объектной моделью документа. Основное предназначение объектной модели браузера – управление окнами браузера и обеспечение их взаимодействия. Каждое из окон браузера представляется объектом window, центральным объектом BOM. Объектная модель браузера на данный момент не стандартизирована, однако спецификация находится в разработке.

Помимо управления окнами, в рамках объектной модели браузера, браузерами обычно обеспечивается поддержка следующих сущностей:

- управление фреймами;

- поддержка задержки в исполнении кода и зацикливания с задержкой;

- системные диалоги;

- управление адресом открытой страницы;

- управление информацией о браузере;

- управление информацией о параметрах монитора;

- ограниченное управление историей просмотра страниц;

- поддержка работы с HTTP cookie.

Объектная модель документа – интерфейс программирования приложений для HTML и XML-документов. Согласно DOM документу можно поставить в соответствие дерево объектов, обладающих рядом свойств, которые позволяют производить с ним различные манипуляции:

- получение узлов;

- изменение узлов;

- изменение связей между узлами;

- удаление узлов.

Более подробный материал по Java Script приведен в источнике [13].

**2.9 Технология AJAX**

Asynchronous Javascript And XML – подход к построению интерактивных [пользовательских интерфейсов](http://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D1%81%D0%BA%D0%B8%D0%B9_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81) [веб-приложений](http://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5), заключающийся в фоновом [обмене данными](http://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D0%BC%D0%B5%D0%BD_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D0%BC%D0%B8) [браузера](http://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D0%B1%D1%80%D0%B0%D1%83%D0%B7%D0%B5%D1%80) с [веб-сервером](http://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80). В результате, при обновлении данных, [веб-страница](http://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D1%82%D1%80%D0%B0%D0%BD%D0%B8%D1%86%D0%B0) не перезагружается полностью и веб-приложения становятся более быстрыми и удобными.

AJAX – не самостоятельная технология, а концепция использования нескольких смежных технологий. AJAX базируется на двух основных принципах:

* + - использование технологии динамического обращения к [серверу](http://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%80%D0%B2%D0%B5%D1%80_%28%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5%29) «на лету», без перезагрузки всей страницы полностью;
    - использование [DHTML](http://ru.wikipedia.org/wiki/DHTML) для динамического изменения содержания страницы.

В качестве формата передачи данных обычно используются [JSON](http://ru.wikipedia.org/wiki/JSON) или [XML](http://ru.wikipedia.org/wiki/XML). Эта технология была изучена при активном использовании источника [14].

## **2.10 Библиотека jQuery**

jQuery – библиотека Java Script, фокусирующаяся на взаимодействии JavaScript и HTML. Библиотека jQuery помогает легко получать доступ к любому элементу DOM, обращаться к атрибутам и содержимому элементов DOM, манипулировать ими. Также библиотека jQuery предоставляет удобный API по работе с Ajax.

Возможности:

- переход по дереву DOM;

- события;

- визуальные эффекты;

- AJAX-дополнения;

- JavaScript-плагины.

Точно так же, как CSS отделяет визуализацию от структуры HTML, jQuery отделяет поведение от структуры HTML. Например, вместо прямого указания на обработчик события нажатия кнопки, управление передаётся jQuery, идентифицирующей кнопки и затем преобразовывающий его в обработчик события клика.

Библиотека jQuery содержит функционал, полезный для максимально широкого круга задач. Тем не менее, разработчиками библиотеки не ставилась задача совмещения в jQuery функций, которые подошли бы всюду, поскольку это привело бы к большому коду, большая часть которого не востребована. Поэтому была реализована архитектура компактного универсального ядра библиотеки и плагинов. Это позволяет собрать для ресурса именно тот JavaScript-функционал, который на нём был бы востребован.

Одним из главных преимуществ работы с jQuery является способ поиска элементов в документе. Он основан на специальных текстовых селекторах, которые идентичны селекторам CSS. Подробное описание данной библиотеки можно найти в источнике [15].

## **2.11 Система управления версиями**

Система управления версиями – [программное обеспечение](http://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) для облегчения работы с изменяющейся информацией. Система управления версиями позволяет хранить несколько версий одного и того же документа, при необходимости, возвращаться к более ранним версиям, определять, кто и когда сделал то или иное изменение и многое другое.

Такие системы наиболее широко применяются при разработке программного обеспечения, для хранения [исходных кодов](http://ru.wikipedia.org/wiki/%D0%98%D1%81%D1%85%D0%BE%D0%B4%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4) разрабатываемой программы. Однако, они могут с успехом применяться и в других областях, в которых ведётся работа с большим количеством непрерывно изменяющихся электронных документов.

Ситуация, когда электронный документ за время своего существования претерпевает ряд изменений, достаточно типична. При этом часто бывает важно иметь не только последнюю версию, но и несколько предыдущих. В простейшем случае, можно просто хранить несколько вариантов документа, соответствующим образом их нумеруя. Но такой способ неэффективен, требует повышенного внимания и дисциплины и часто ведёт к ошибкам.

Традиционные системы управления версиями используют централизованную модель, когда имеется единое хранилище документов, управляемое специальным [сервером](http://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%80%D0%B2%D0%B5%D1%80_%28%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5%29), который и выполняет большую часть функций по управлению версиями. Пользователь, работающий с документами, должен сначала получить нужную ему версию документа из хранилища; обычно создаётся локальная копия документа. Может быть получена последняя версия или любая из предыдущих, которая может быть выбрана по номеру версии или дате создания. После того, как в документ внесены нужные изменения, новая версия помещается в хранилище. В отличие от простого сохранения файла, предыдущая версия не стирается, а тоже остаётся в хранилище и может быть оттуда получена в любое время. Сервер может использовать [дельта-компрессию](http://ru.wikipedia.org/wiki/%D0%94%D0%B5%D0%BB%D1%8C%D1%82%D0%B0-%D0%BA%D0%BE%D0%BC%D0%BF%D1%80%D0%B5%D1%81%D1%81%D0%B8%D1%8F) – такой способ хранения документов, при котором сохраняются только изменения между последовательными версиями, что позволяет уменьшить объём хранимых данных.

Часто бывает, что над одним проектом одновременно работают несколько человек. Если два человека изменяют один и тот же [файл](http://ru.wikipedia.org/wiki/%D0%A4%D0%B0%D0%B9%D0%BB), то один из них может случайно отменить изменения, сделанные другим. Системы управления версиями отслеживают такие конфликты и предлагают средства их решения. Большинство систем может автоматически объединить изменения, сделанные разными разработчиками. Однако такое автоматическое объединение изменений, обычно, возможно только для [текстовых файлов](http://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BA%D1%81%D1%82%D0%BE%D0%B2%D1%8B%D0%B9_%D1%84%D0%B0%D0%B9%D0%BB) и при условии, что изменялись разные части этого файла. Такое ограничение связано с тем, что большинство систем управления версиями ориентированы на поддержку процесса разработки программного обеспечения, а исходные коды программ хранятся в текстовых файлах. Если автоматическое объединение выполнить не удалось, система может предложить решить проблему вручную.

Часто выполнить слияние невозможно ни в автоматическом, ни в ручном режиме, например, если [формат файла](http://ru.wikipedia.org/wiki/%D0%A4%D0%BE%D1%80%D0%BC%D0%B0%D1%82_%D1%84%D0%B0%D0%B9%D0%BB%D0%B0) неизвестен или слишком сложен. Некоторые системы управления версиями дают возможность заблокировать файл в хранилище. Блокировка не позволяет другим пользователям получить рабочую копию или препятствует изменению рабочей копии файла и обеспечивает, таким образом, исключительный доступ только тому пользователю, который работает с документом.

Многие системы управления версиями предоставляют ряд других возможностей:

- Позволяют создавать разные варианты одного документа с общей историей изменений до точки ветвления и с разными – после неё.

- Дают возможность узнать, кто и когда добавил или изменил конкретный набор строк в файле.

- Ведут журнал изменений, в который пользователи могут записывать пояснения о том, что и почему они изменили в данной версии.

- Контролируют права доступа пользователей, разрешая или запрещая чтение или изменение данных, в зависимости от того, кто запрашивает это действие.

В данной работе был использован инструмент управления версиями tortoiseSVN – это клиент для системы контроля версий, выполненный как расширение оболочки [Windows](http://ru.wikipedia.org/wiki/Windows).

Данный инструмент обладает следующими возможностями:

- Интеграция с оболочкой Windows (опции пакета отображены в контекстном меню).

- Для работы не требуется [IDE](http://ru.wikipedia.org/wiki/%D0%A1%D1%80%D0%B5%D0%B4%D0%B0_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F).

- Информационные значки на иконках файлов для индикации состояния файла.

- На сайте доступно множество версий локализаций пакета.

## **2.12 Инструменты отладки и тестирования отображаемых страниц**

Для отладки и тестирования содержания страниц на клиенте были использованы специальные расширения для браузеров Fire Fox и Internet Explorer: Firebug, Internet Explorer Developer.

Firebug – расширение для браузера [Firefox](http://ru.wikipedia.org/wiki/Firefox), являющееся [консолью](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D1%81%D0%BE%D0%BB%D1%8C), [отладчиком](http://ru.wikipedia.org/wiki/%D0%9E%D1%82%D0%BB%D0%B0%D0%B4%D1%87%D0%B8%D0%BA), и [DOM](http://ru.wikipedia.org/wiki/Document_Object_Model)-инспектором [JavaScript](http://ru.wikipedia.org/wiki/JavaScript), [DHTML](http://ru.wikipedia.org/wiki/DHTML), [CSS](http://ru.wikipedia.org/wiki/CSS), [XMLHttpRequest](http://ru.wikipedia.org/wiki/XMLHttpRequest). Firebug показывает в консоли вызвавшую ошибку функцию, [стек вызовов](http://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B5%D0%BA_%D0%B2%D1%8B%D0%B7%D0%BE%D0%B2%D0%BE%D0%B2) функций, вызвавших эту ошибку. Он предупреждает, что CSS-правило или JavaScript-метод или свойство, которое вы пытаетесь использовать, не существует. Firebug предоставляет разработчику следующие возможности:

- Удобный просмотр [HTML](http://ru.wikipedia.org/wiki/HTML)-кода страницы. Позволяет точно определить местонахождение тега того или иного элемента, просмотреть все «привязанные» к нему свойства и стили.

- Редактирование HTML и [CSS](http://ru.wikipedia.org/wiki/CSS) прямо в браузере. Можно изменять атрибуты тегов и значения свойств для того, чтобы пронаблюдать изменения. Удобно для тех случаев, когда нужно путём экспериментов найти наиболее приемлемый вариант оформления создаваемой страницы.

- Отладка [JavaScript](http://ru.wikipedia.org/wiki/JavaScript).

- Отслеживание процесса загрузки страницы.

- Просмотр [HTTP-заголовков](http://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%B3%D0%BE%D0%BB%D0%BE%D0%B2%D0%BA%D0%B8_HTTP) обычных и [AJAX](http://ru.wikipedia.org/wiki/AJAX)-запросов.

[Internet Explorer Developer – панель инструментов](http://ru.wikipedia.org/wiki/%D0%9F%D0%B0%D0%BD%D0%B5%D0%BB%D1%8C_%D0%B8%D0%BD%D1%81%D1%82%D1%80%D1%83%D0%BC%D0%B5%D0%BD%D1%82%D0%BE%D0%B2) для веб-разработчика. Отображает дополнительную информацию о странице. Является [консолью](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D1%81%D0%BE%D0%BB%D1%8C), [отладчиком](http://ru.wikipedia.org/wiki/%D0%9E%D1%82%D0%BB%D0%B0%D0%B4%D1%87%D0%B8%D0%BA), [DOM](http://ru.wikipedia.org/wiki/Document_Object_Model)-инспектором [JavaScript](http://ru.wikipedia.org/wiki/Javascript), [CSS](http://ru.wikipedia.org/wiki/CSS), [XHTML](http://ru.wikipedia.org/wiki/XHTML).

**3 СТРУКТУРА БАЗЫ ДАННЫХ ПРИЛОЖЕНИЯ**

В ходе выполнения данной курсовой работы была разработана база данных библиотеки высшего учебного заведения, предназначенная для использования веб-приложением. База данных состоит из 16 таблиц, описание которых приведено в таблице 3.1.

Таблица 3.1 – Состав таблиц

|  |  |
| --- | --- |
| Название таблицы | Сведения о содержимом |
| users | сведения о пользователях |
| roles | перечень ролей пользователей |
| user\_categories | наименование категорий пользователей |
| book\_descriptions | перечень описаний книг |
| book\_categories | информация о категориях книг |
| real\_books | перечень реальных книг |
| debts | сведения о задолженностях |
| languages | справочник языков |
| comments | информация о комментариях |
| electronic\_books | перечень электронных книг |
| queues | информация об очередях за книгами |
| departments | информация о факультетах |
| divisions | сведения о подразделениях |
| privileges\_electronic | информация о видимости электронных книг |
| privileges\_real | информация о видимости реальных книг |
| reports | сведения для отчета |

В ходе проектирования базы данных были определены связи между таблицами. На рисунке 3.1 приведена схема связей между таблицами.

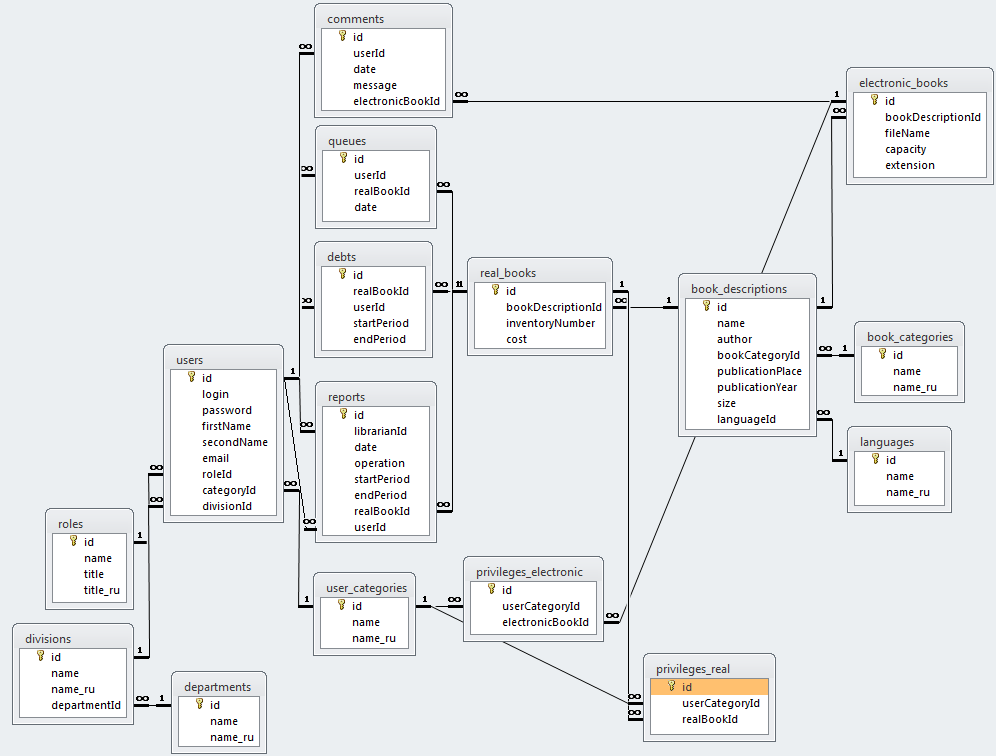


Рисунок 3.1 – Схема связей между таблицами

В таблицах 3.2 - 3.17 приведено подробное описание структуры таблиц спроектированной базы данных.

Таблица 3.2 – Структура таблицы «users»

|  |  |  |
| --- | --- | --- |
| Пользователи | | |
| Наименование поля | Формат поля | Содержимое поля |
| id | Счетчик | Идентификатор записи |
| login | Текстовый | Логин пользователя |
| password | Текстовый | Пароль |
| firstName | Текстовый | Имя |
| secondName | Текстовый | Фамилия |
| email | Текстовый | Почта |
| roleId | Числовой | Код роли (связь с таблицей «roles») |
| categoryId | Числовой | Код категории (связь с таблицей «user\_categories») |
| divisionId | Числовой | Код подразделения (связь с таблицей «divisions») |

Таблица 3.3 – Структура таблицы «user\_categories»

|  |  |  |
| --- | --- | --- |
| Категории пользователей | | |
| Наименование поля | Формат поля | Содержимое поля |
| id | Счетчик | Идентификатор записи |
| name | Текстовый | Название категории пользователя |
| name\_ru | Текстовый | Название категории пользователя на русском языке |

Таблица 3.4 – Структура таблицы «book\_descriptions»

|  |  |  |
| --- | --- | --- |
| Описания книг | | |
| Наименование поля | Формат поля | Содержимое поля |
| id | Счетчик | Идентификатор записи |
| name | Текстовый | Название книги |
| author | Текстовый | Автор |
| bookCategoryId | Числовой | Код категории книги (связь с таблицей «book\_categories») |
| publicationPlace | Текстовый | Место публикации |
| publicationYear | Числовой | Год публикации |
| size | Числовой | Количество страниц |
| languageId | Числовой | Код языка (связь с таблицей «languages») |

Таблица 3.5 – Структура таблицы «departments»

|  |  |  |
| --- | --- | --- |
| Факультеты | | |
| Наименование поля | Формат поля | Содержимое поля |
| id | Счетчик | Идентификатор записи |
| name | Текстовый | Название факультета |
| name\_ru | Текстовый | Название факультета на русском |

Таблица 3.6 – Структура таблицы «reports»

|  |  |  |
| --- | --- | --- |
| Отчет | | |
| Наименование поля | Формат поля | Содержимое поля |
| id | Счетчик | Идентификатор записи |
| librarianId | Числовой | Код библиотекаря (связь с таблицей «users») |
| date | Дата/Время | Дата выполнения операции |
| operation | Текстовый | Название операции |
| startPeriod | Дата/Время | Дата начала периода |
| endPeriod | Дата/Время | Дата окончания периода |
| realBookId | Числовой | Код реальной книги (связь с таблицей «real\_books») |
| userId | Числовой | Код пользователя (связь с таблицей «users») |

Таблица 3.7 – Структура таблицы «queues»

|  |  |  |
| --- | --- | --- |
| Очереди | | |
| Наименование поля | Формат поля | Содержимое поля |
| id | Счетчик | Идентификатор записи |
| userId | Числовой | Код пользователя (связь с таблицей «users») |
| realBookId | Числовой | Код реальной книги (связь с таблицей «real\_books») |
| date | Дата/Время | Время становления пользователя в очередь |

Таблица 3.8 – Структура таблицы «real\_books»

|  |  |  |
| --- | --- | --- |
| Реальные книги | | |
| Наименование поля | Формат поля | Содержимое поля |
| id | Счетчик | Идентификатор записи |
| bookDescriptionId | Числовой | Код описания книги (связь с таблицей «book\_descriptions») |
| inventoryNumber | Числовой | Инвентарный номер |
| cost | Числовой | Цена |

Таблица 3.9 – Структура таблицы «debts»

|  |  |  |
| --- | --- | --- |
| Задолженности | | |
| Наименование поля | Формат поля | Содержимое поля |
| id | Счетчик | Идентификатор записи |
| realBookId | Числовой | Код реальной книги (связь с таблицей «real\_books») |
| userId | Числовой | Код пользователя (связь с таблицей «users») |
| startPeriod | Дата/Время | Дата начала срока |
| endPeriod | Дата/Время | Дата конца срока |

Таблица 3.10 – Структура таблицы «divisions»

|  |  |  |
| --- | --- | --- |
| Подразделения | | |
| Наименование поля | Формат поля | Содержимое поля |
| id | Счетчик | Идентификатор записи |
| name | Текстовый | Название подразделения |
| name\_ru | Текстовый | Название подразделения на русском |
| departmentId | Числовой | Код факультета (связь с таблицей «departments») |

Таблица 3.11– Структура таблицы «comments»

|  |  |  |
| --- | --- | --- |
| Комментарии | | |
| Наименование поля | Формат поля | Содержимое поля |
| id | Счетчик | Идентификатор записи |
| userId | Числовой | Код пользователя (связь с таблицей «users») |
| date | Дата/Время | Дата создания комментария |
| message | Текстовый | Содержимое комментария |
| electronicBookId | Числовой | Код электронной книги (связь с таблицей «electronic\_books») |

Таблица 3.12 – Структура таблицы «electronic\_books»

|  |  |  |
| --- | --- | --- |
| Электронные книги | | |
| Наименование поля | Формат поля | Содержимое поля |
| id | Счетчик | Идентификатор записи |
| bookDescriptionId | Числовой | Код описания книги (связь с таблицей «book\_descriptions») |
| fileName | Текстовый | Название файла |
| capacity | Числовой | Размер файла |
| extension | Текстовый | Расширение файла |

Таблица 3.13 – Структура таблицы «roles»

|  |  |  |
| --- | --- | --- |
| Роли | | |
| Наименование поля | Формат поля | Содержимое поля |
| id | Счетчик | Идентификатор записи |
| name | Текстовый | Название роли |
| title | Текстовый | Отображаемое имя по умолчанию |
| title\_ru | Текстовый | Отображаемое имя ну русском языке |

Таблица 3.14 – Структура таблицы «languages»

|  |  |  |
| --- | --- | --- |
| Языки | | |
| Наименование поля | Формат поля | Содержимое поля |
| id | Счетчик | Идентификатор записи |
| name | Текстовый | Название языка |
| name\_ru | Текстовый | Название языка на русском языке |

Таблица 3.15 – Структура таблицы «privileges\_electronic»

|  |  |  |
| --- | --- | --- |
| Привилегии для электронных книг | | |
| Наименование поля | Формат поля | Содержимое поля |
| id | Счетчик | Идентификатор записи |
| userCategoryId | Числовой | Код категории пользователя (связь с таблицей «user\_categories») |
| electronicBookId | Числовой | Код электронной книги (связь с таблицей «electronic\_books») |

Таблица 3.16 – Структура таблицы «privileges\_real»

|  |  |  |
| --- | --- | --- |
| Привилегии для реальных книг | | |
| Наименование поля | Формат поля | Содержимое поля |
| id | Счетчик | Идентификатор записи |
| userCategoryId | Числовой | Код категории пользователя (связь с таблицей «user\_categories») |
| realBookId | Числовой | Код реальной книги (связь с таблицей «real\_books») |

Таблица 3.17 – Структура таблицы «book\_categories»

|  |  |  |
| --- | --- | --- |
| Категории книг | | |
| Наименование поля | Формат поля | Содержимое поля |
| id | Счетчик | Идентификатор записи |
| name | Текстовый | Название категории книги |
| name\_ru | Текстовый | Название категории книги на русском языке |

**4 СХЕМА ДИАЛОГА ПОЛЬЗОВАТЕЛЯ С РАЗРАБОТАННЫМ ПРИЛОЖЕНИЕМ**

Работа с приложением «Библиотека высшего учебного заведения» начинается со стартовой страницы, которая отображена на рисунке 4.1.

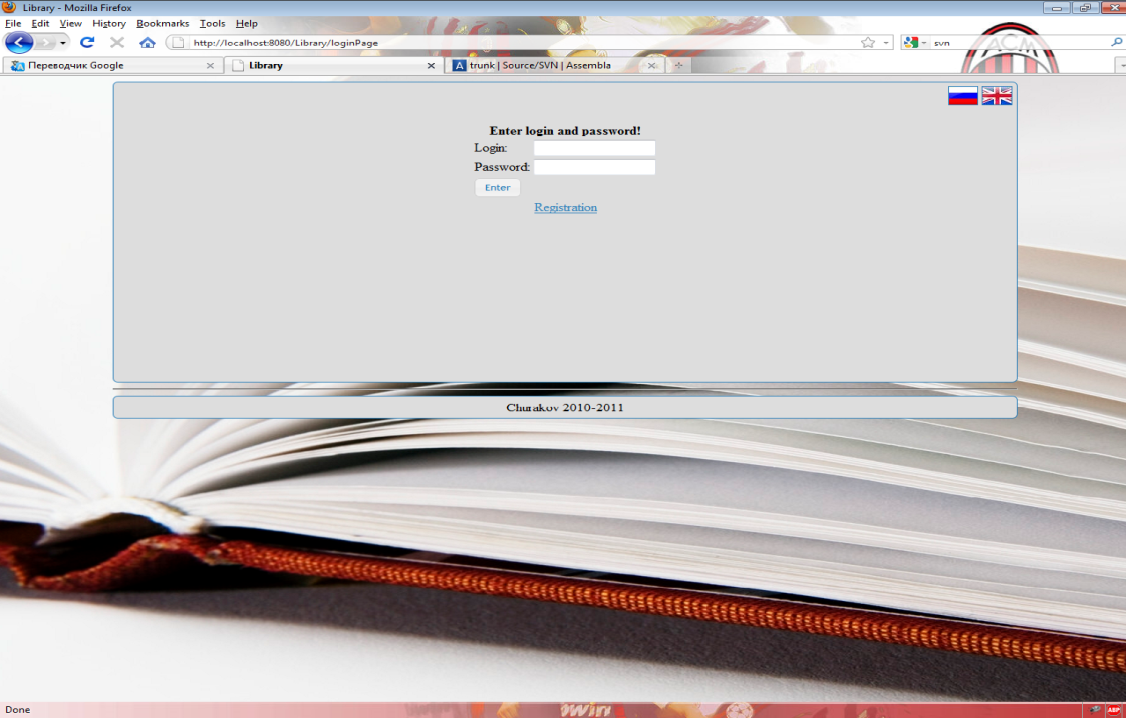


Рисунок 4.1 – Стартовая страница приложения

На данной странице пользователь может поменять язык отображения текстовых сообщений в приложении. Доступны два языка для выбора: русский, английский. Для смены языка необходимо кликнуть на нужном флажке.

**4.1 Регистрация на сайте библиотеки высшего учебного заведения**

Для регистрации новой учетной записи пользователю необходимо на стартовой странице кликнуть на ссылке «Регистрация». Далее он перейдёт на страницу регистрации. Эта страница приведена на рисунке 4.2.

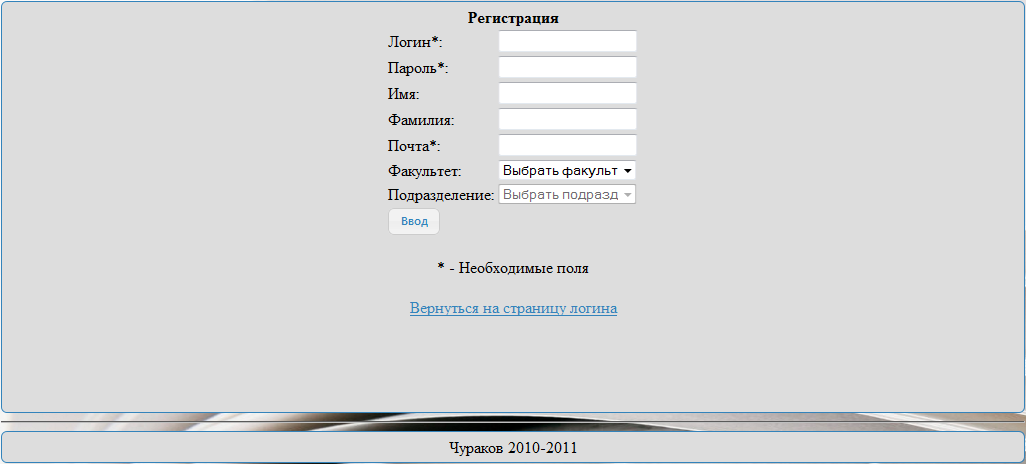


Рисунок 4.2 – Страница регистрации нового пользователя

Со страницы регистрации пользователь может вернуться на стартовую страницу, нажав на ссылке «Вернуться на страницу логина».

Для регистрации нужно заполнить обязательные поля, которые отмечены символом «\*», и кликнуть на кнопке «Ввод» или нажать клавишу «Enter». Если обязательное поле не будет заполнено, то появится предупреждающее сообщение, пример которого приведен на рисунке 4.3.

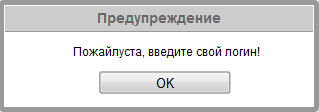


Рисунок 4.3 – Предупреждение о незаполненном поле

Если регистрация прошла успешно, то отобразится стартовая страница с сообщением об успешной регистрации и необходимости ожидания активации. Данное сообщение приведено на рисунке 4.4.

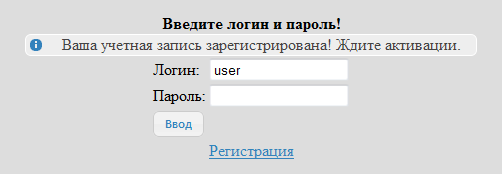


Рисунок 4.4 – Сообщение об успешной регистрации

В том случае если зарегистрированный пользователь, которого еще не активировал администратор, попытается авторизоваться на сайте библиотеки, то будет показано сообщение о необходимости активации. Это сообщение присутствует на рисунке 4.5.

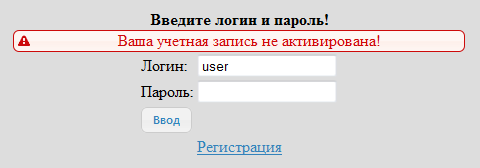


Рисунок 4.5 – Сообщение о необходимости активации

При авторизации пользователь должен ввести свой логин и пароль и нажать кнопку «Ввод». Если они не будут введены, то при попытке авторизации появится соответствующее предупреждение.

Когда пользователь пытается авторизоваться, используя логин и пароль не существующей учетной записи, тогда появится предупреждение об ошибке.

После авторизации любой пользователь может увидеть в шапке страницы своё полное имя, а также роль и категорию. Под личной информацией пользователя находится меню для осуществления навигации.

**4.2 Описание интерфейса для администратора**

После того как администратор войдет в систему, он может увидеть в главном меню следующие пункты: «Пользователи», «Справочники», «Статистика», «Информация», «Выход». Для того чтобы выйти из системы нужно кликнуть на пункте «Выход».

**4.2.1 Администрирование пользователей**

После успешной авторизации администратор входит в основную часть приложение и может увидеть страницу для администрирования пользователей. Рисунок этой страницы приведен на рисунке 4.6.

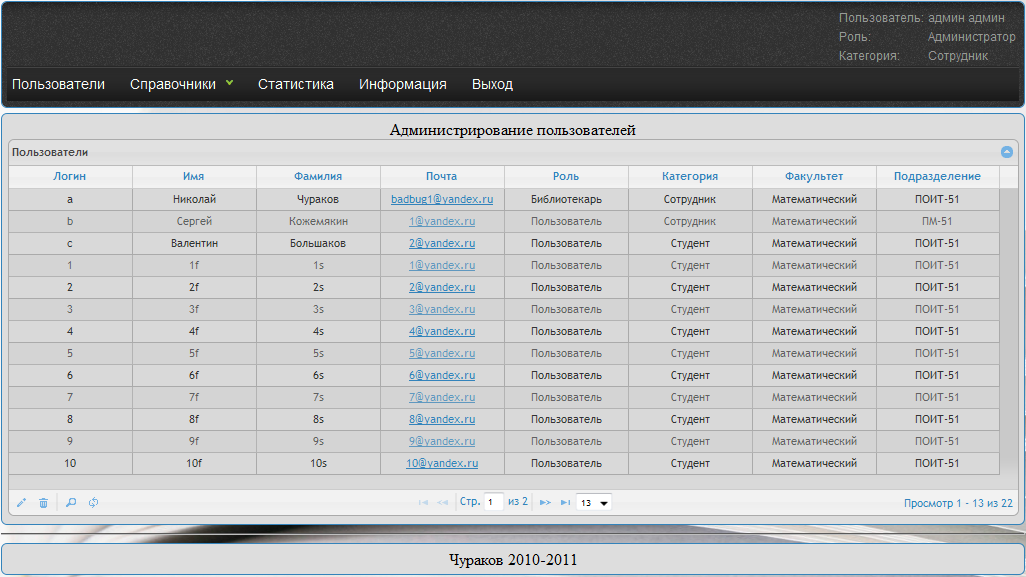


Рисунок 4.6 – Страница администрирования пользователей

На странице администрирования пользователей отображена таблица, содержащая информацию о всех пользователях. Для удобства работы с большими списками пользователей предусмотрено разбиение списка на страницы и возможность осуществлять навигацию по ним. Для этого можно использовать ссылки с изображением стрелочек либо ввести номер необходимой страницы и нажать клавишу «Enter». Также можно изменять количество отображаемых записей на одной странице. Данную настройку можно произвести справа от стрелочек пагинации, выбрав пункт в выпадающем списке. Внизу таблицы также приведено общее количество записей и диапазон номеров отображаемых записей.

Если необходимо отсортировать записи по определенному полю, нужно кликнуть на соответствующем названии колонки, повторный клик отсортирует в обратном порядке.

В случае если на компьютере установлен почтовый клиент, то при клике на почте пользователя произойдет отображение окна создания нового письма для выбранного адресанта. Пример окна почтового клиента приведен на рисунке 4.7.

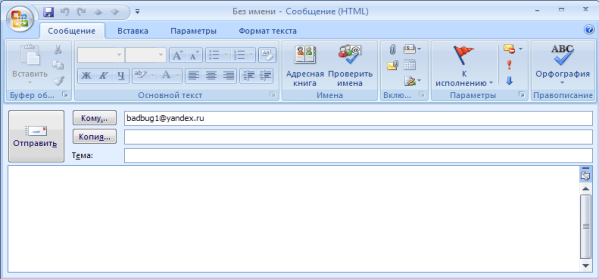


Рисунок 4.7 – Отправка письма пользователю

Внизу таблицы с пользователями находится панель с кнопками. Данная панель содержит следующие кнопки: «Обновить», «Поиск», «Удалить», «Редактировать».

Если нажать на кнопку «Обновить», то произойдет перезагрузка списка, а также будет очищен фильтр поиска.

При нажатии на кнопку «Поиск» появится окно настройки фильтра поиска. Окно фильтра изображено на рисунке 4.8.

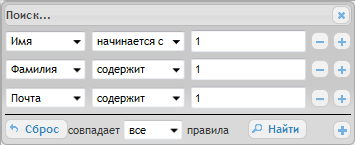


Рисунок 4.8 – Окно фильтра для поиска пользователей

Окно фильтра позволяет задать гибкие критерии поиска. Пользователь может указать по каким полям искать, какие критерии поиска к ним применять и какие значения искать в этих полях. Также можно указать правило совпадения (все, любое). Для строковых полей можно применить следующие критерии поиска: «начинается с», «содержит», «равно». Затем нужно нажать кнопку «Найти» и будут отображены записи, удовлетворяющие критериям поиска.

Для того чтобы удалить пользователя нужно выбрать его в списке и нажать на панели кнопку «Удалить». После этого появится запрос подтверждения операции, он показан на рисунке 4.9.

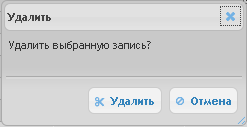


Рисунок 4.9 – Подтверждение удаления записи

Если пользователь подтвердит удаление и нажмет кнопку «Удалить» произойдет проверка возможных связей с этой записью и, если такие не обнаружатся, произойдет удаление. В противном же случае будет показано предупреждение как на рисунке 4.10.

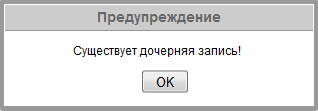


Рисунок 4.10 – Предупреждение о существовании дочерней записи

Чтобы выполнить редактирование данных пользователя нужно выбрать его и нажать на панели кнопку «Редактировать». Затем появится окно с данными выбранного пользователя. Это окно отображено на рисунке 4.11.

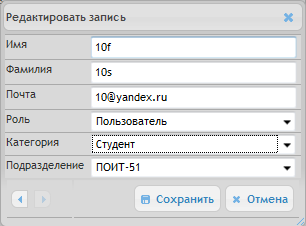


Рисунок 4.11 – Окно редактирования данных пользователя

В появившемся окне можно перемещаться по записям, используя стрелочки. При редактировании нужно корректно изменить значения полей и нажать кнопку «Сохранить». Если будет введен некорректный почтовый адрес, то отобразится предупреждение, которое можно увидеть на рисунке 4.12.

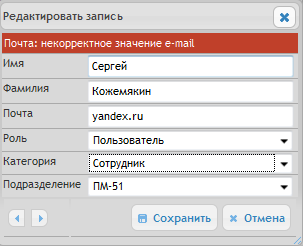


Рисунок 4.12 – Предупреждение о некорректном почтовом адресе

**4.2.2 Редактирование справочников**

Администратор имеет возможность редактировать следующие справочники: «факультеты», «категории книг», «языки», «категории пользователей», «подразделения». Для осуществления редактирования нужно выбрать в пункте меню «Справочники» подпункт соответствующий необходимому справочнику. Затем появится таблица с содержанием справочника, пример приведен на рисунке 4.13. Работа с содержимым справочника не отличается от работы со списком пользователей.

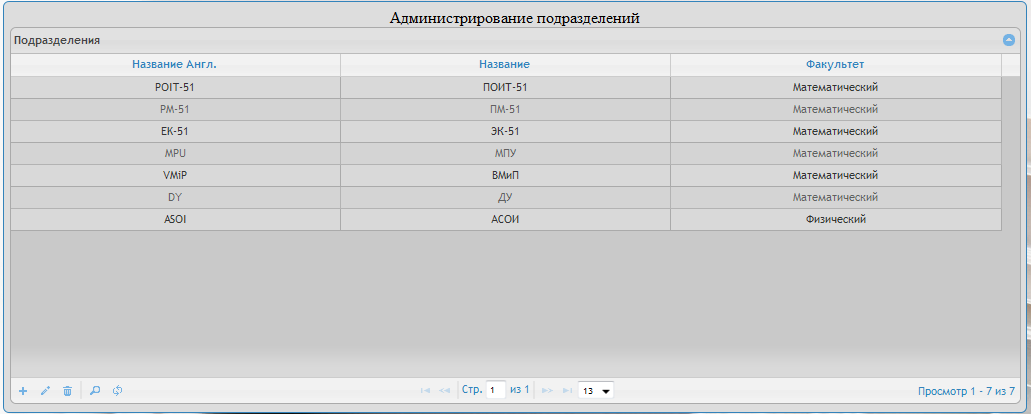


Рисунок 4.13 – Содержимое справочника «Подразделения»

**4.2.3 Информация о библиотеке высшего учебного заведения**

Если в главном меню кликнуть на пункте «Информация», то произойдет переход на информационную страницу. Данная страница содержи две вкладки: «Информация о библиотеке», «Интернет-ресурсы».

Вкладка «Информация о библиотеке» содержит контактную информацию, место расположения и режим работы библиотеки. Эта вкладка отображена на рисунке 4.14.

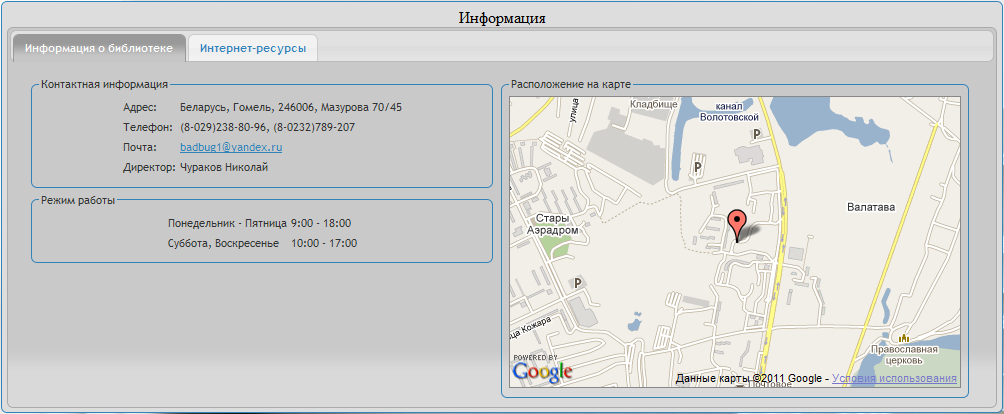


Рисунок 4.14 – Вкладка «Информация о библиотеке»

На вкладке «Интернет-ресурсы» приведен список интернет-ресурсов библиотечного направления, в число которых входят различные электронные библиотеки. Содержимое вкладки «Интернет-ресурсы» приведено на рисунке 4.15.

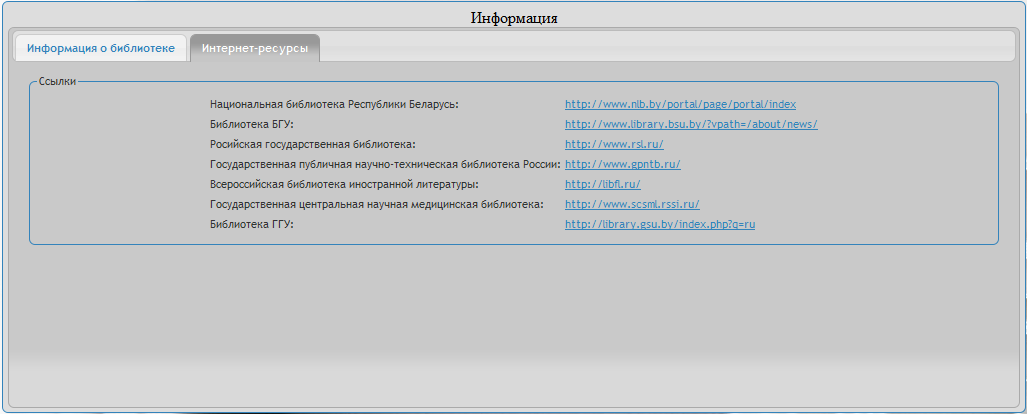


Рисунок 4.15 – Вкладка «Интернет-ресурсы»

**4.2.4 Статистика работы библиотеки высшего учебного заведения**

Для просмотра статистика нужно перейти на страницу «Статистика». Данная страница содержит общие сведения о работе библиотеки и ее пример можно увидеть на рисунке 4.16.

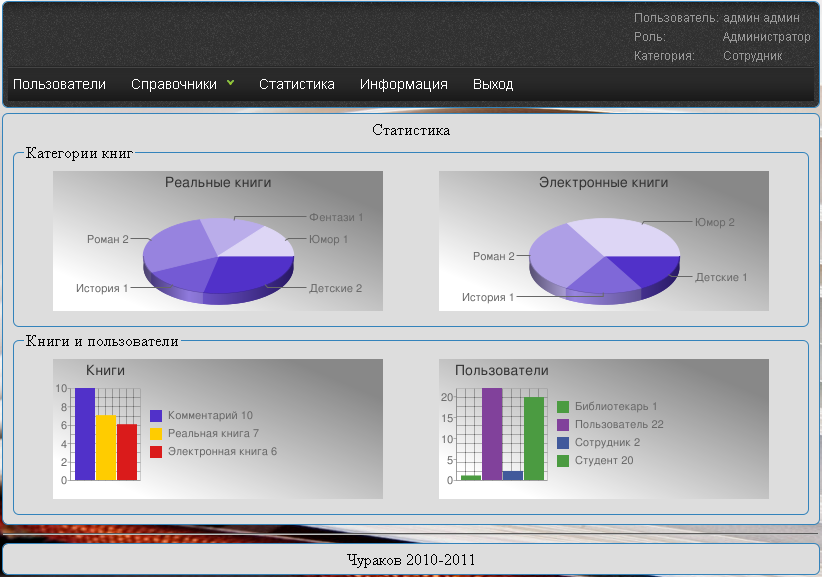


Рисунок 4.16 – Страница «Статистика»

Эта страница содержит 4 графика. График «Реальные книги» показывает распределение имеющихся реальных книг по книжным категориям, а график «Электронные книги» - распределение электронных книг. График «Книги» отображает число реальных, электронных книг и количество оставленных комментариев. График «Пользователи» показывает общее количество пользователей и библиотекарей, а также количество пользователей в зависимости от категории.

**4.3 Описание интерфейса для пользователя**

После того как пользователь войдет в систему, он может увидеть в главном меню следующие пункты: «Профиль», «Книги», «Долги», «Статистика», «Информация», «Выход».

**4.3.1 Администрирование профиля пользователя**

После успешной авторизации пользователь входит в основную часть приложение и может увидеть страницу для администрирования своего профиля. Рисунок этой страницы приведен на рисунке 4.17.

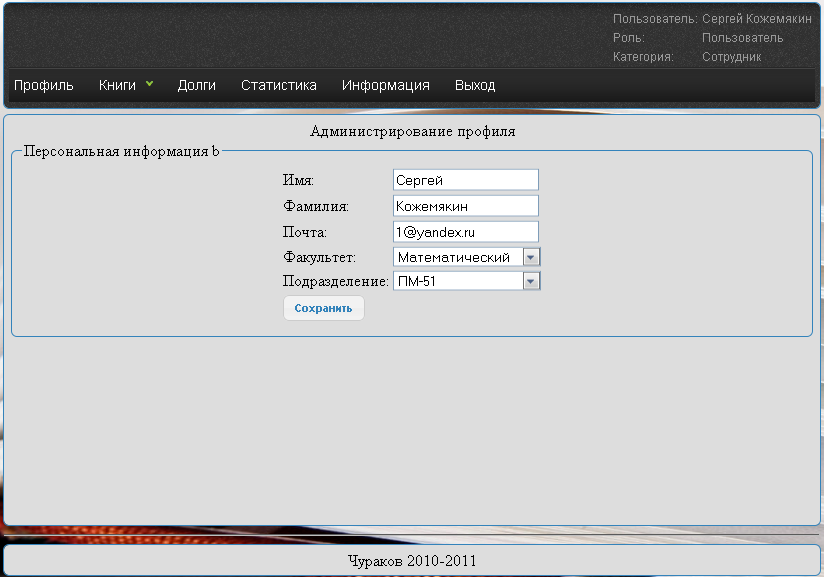


Рисунок 4.17 – Страница администрирования профиля

Данной странице соответствует пункт меню «Профиль». Здесь пользователь может поменять свои личные данные, изменив соответствующие поля и кликнув кнопку «Сохранить». Если почтовый адрес был введен корректно, то изменения будут сохранены, иначе появится предупреждение о неверном почтовом адресе. Также пользователь может изменить информации о факультете и подразделении, к которому он относится.

В пункте меню «Книги» есть два подпункта: «Реальные книги», «Эл.книги». Их можно увидеть на рисунке 4.18.

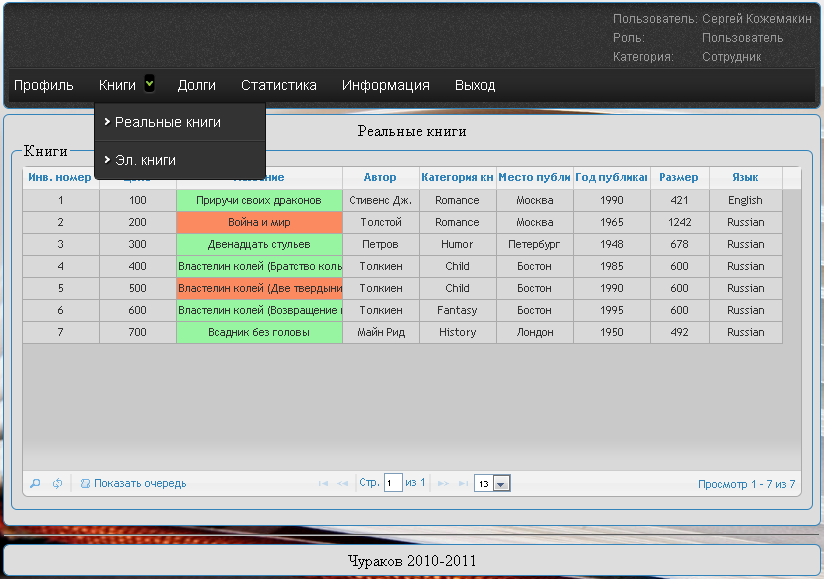


Рисунок 4.18 – Подпункты элемента меню «Книги»

**4.3.2 Обзор реальных книг**

Если выбрать пункт «Реальные книги», тогда отобразится страница реальных книг, смотри рисунок 4.18. Здесь отображена следующая информация о книгах: инвентарный номер, цена, название, автор, категория книги, место публикации, год публикации, размер, язык. Книги, имеющиеся в наличии, выделены зеленым цветом, а те книги, которые выданы – красным.

Аналогично, как и со списком пользователей, для администратора пользователь может осуществлять сортировку, поиск и обновление таблицы.

Однако для числовых полей можно применить специфические критерии поиска: «начинается с», «равно», «меньше», «меньше или равно», «больше», «больше или равно».

Также пользователь может просмотреть очередь для данной книги. Для этого он должен выбрать книгу и нажать на кнопку «Показать очередь».

Когда книга выбрана и нажата кнопка «Показать очередь», тогда будет отображено окно содержащее очередь на выбранную книгу. Это окно изображено на рисунке 4.19.

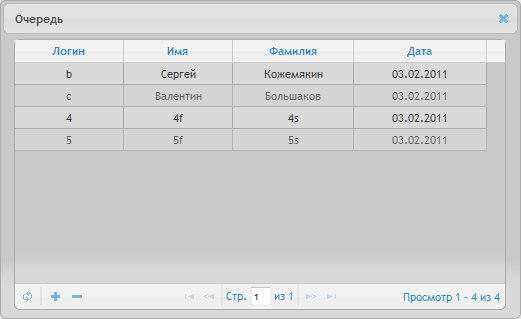


Рисунок 4.19 – Очередь для выбранной книги

В этом окне показана информация о пользователях, которые стали в очередь, а также дата появления в очередь. Пользователь может добавить себя в очередь, нажав кнопку «+». Однако если он уже добавлен, то будет показано предупреждение.

Если необходимо удалить себя из очереди, то нужно нажать кликнуть на кнопку «-». После этого появится подтверждение удаление.

При попытке пользователем удалить себя из очереди, в которой он не находится, произойдет появление предупреждающего сообщения.

**4.3.3 Обзор электронных книг**

Если выбрать пункт «Эл.книги», тогда отобразится страница электронных книг, которая приведена на рисунке 4.20.

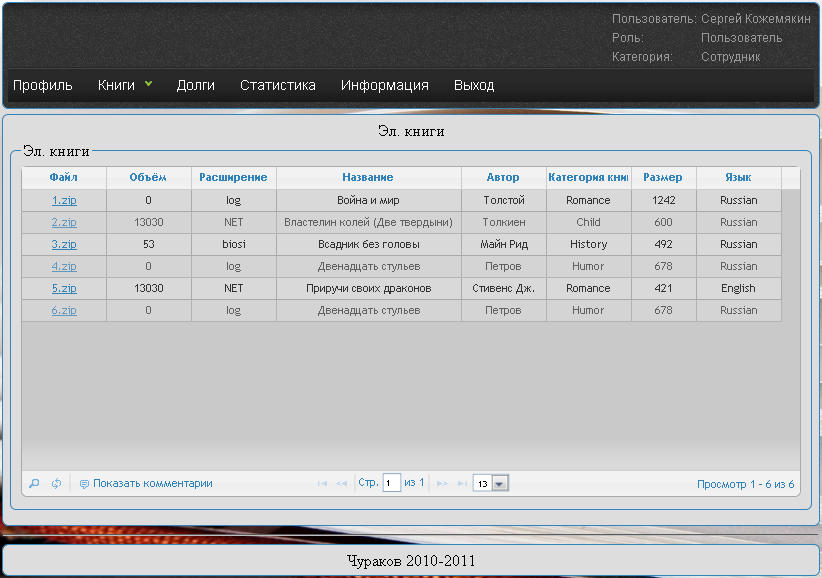


Рисунок 4.20 – Страница электронных книг

На данной странице отображен список электронных книг, имеющихся в библиотеке высшего учебного заведения. В отличие от списка реальных книг, здесь есть информация о формате, объеме файла электронной книги. Все электронные книги хранятся в формате ZIP архива. Для того чтобы скачать необходимую книгу нужно кликнуть на соответствующей ссылке в поле «Файл». Затем появится диалог для загрузки файла. Пример загрузки приведен на рисунки 4.21.

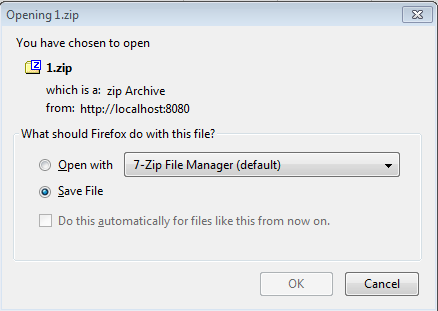


Рисунок 4.21 – Диалог для загрузки файла

Также пользователь может осуществлять поиск и обновления таблицы, используя соответствующие кнопки, находящиеся на панели таблицы электронных книг.

Любой пользователь может оставлять комментарии для определенной книги. Для этого нужно выбрать электронную книгу и нажать кнопку «Показать комментарии», затем будет отображено окно для работы с комментариями. Это окно изображено на рисунке 4.22.

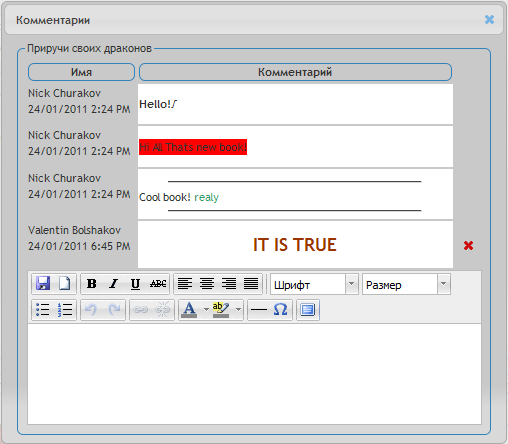


Рисунок 4.22 – Окно для работы с комментариями

В данном окне отображены все комментарии с информацией о времени и об авторе, для выбранной электронной книги. Также пользователь может добавить новые комментарии, используя для форматирования встроенный текстовый редактор. С помощью редактора можно задать размер, шрифт, стиль, выравнивание, перечисление, цвет, фон для текста, а также добавить разделитель, дополнительные символы. После нужно нажать иконку «Сохранить», и новый комментарий будет сохранен. Также пользователь может удалять комментарии, которые он добавил. Для этого нужно кликнуть на значке «Удалить» напротив соответствующего комментария. После нужно подтвердить удаление. Данное подтверждение показано на рисунке 4.23.

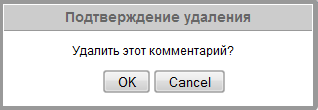


Рисунок 4.23 – Подтверждение удаления комментария

**4.3.4 Просмотр персональных задолженностей**

Для просмотра задолженностей нужно перейти на страницу «Долги». На этой страницы пользователь может просмотреть все свои задолженности. Пример этой страницы приведен на рисунке 4.24.

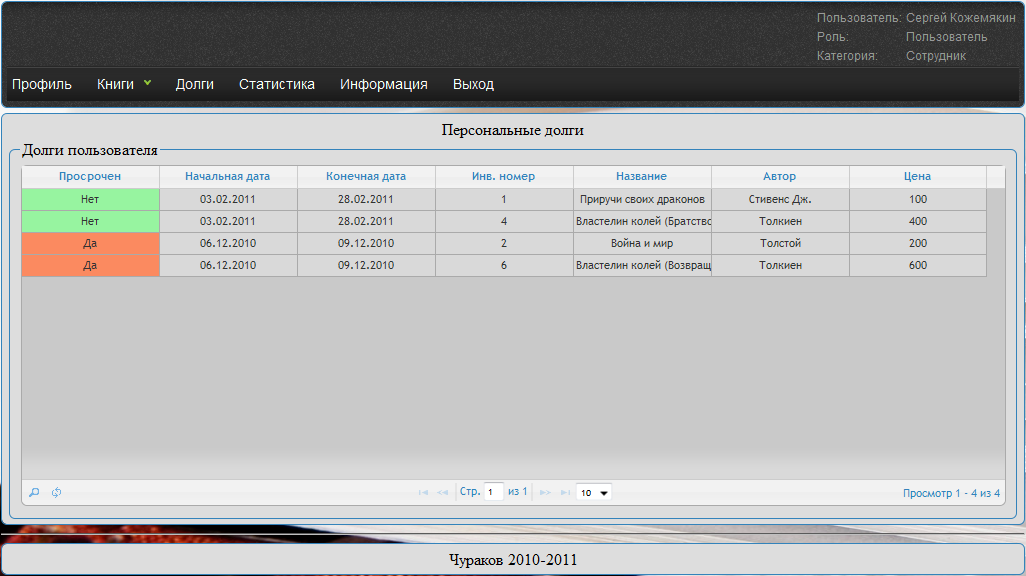


Рисунок 4.24 – Задолженности пользователя

Данная страница содержит таблицу задолженностей, пользователь может выполнить операцию обновления и поиска, аналогично, как и в выше описанных таблицах. В колонке «Просрочен» содержится информации о том, просрочена ли книга или нет. Если книга просрочена то, она выделяется красным цветом, иначе зеленым. Также данная таблица содержит информацию о дате выдачи и планируемой сдачи книги, инвентарном номере, названии книги, авторе и цене книги.

Пункты меню «Статистика» и «Информация» переводят пользователя на страницы, аналогичные, как и у администратора.

**4.4 Описание интерфейса для библиотекаря**

После того как библиотекарь войдет в систему, он может увидеть в главном меню следующие пункты: «Профиль», «Книги», «Читатели», «Долги», «Отчет», «Статистика», «Информация», «Выход».

**4.4.1 Редактирование описаний книг**

Пункт меню «Книги» имеет три подпункта: «Описания книг», «Реальные книги», «Эл.книги». Пример данных пунктов приведен на рисунке 4.25.

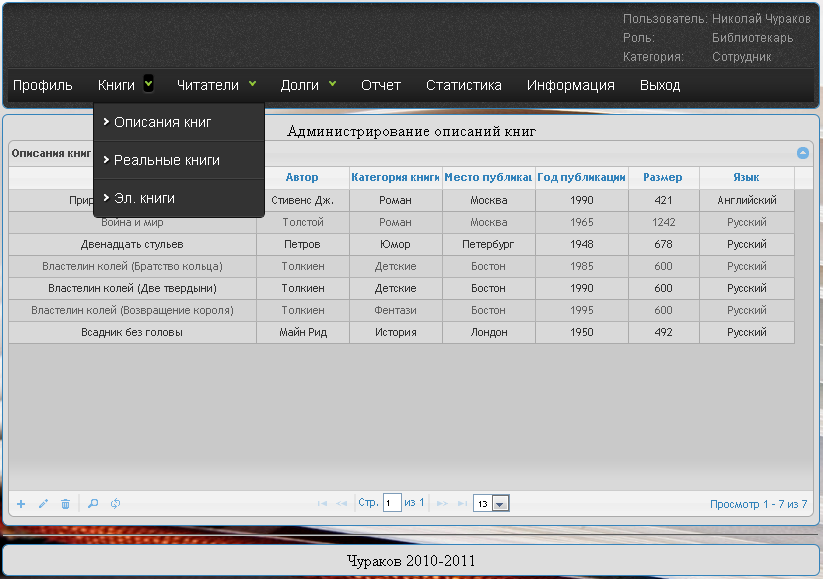


Рисунок 4.25 – Страница описаний книг

Если библиотекарь выберет в пункте меню «Книги» подпункт «Описания книг», тогда отобразится страница для работы с описаниями книг. Ее пример приведен на рисунке 4.25. Описания книг содержат следующую информацию: название книги, автор, категория книги, место публикации, год публикации, размер, язык.

Библиотекарь может осуществлять сортировку, поиск, обновление таблицы, аналогично, как в таблицах описанных выше.

Для того чтобы удалить описание книги, нужно ее выбрать и нажать кнопку «Удалить». И если нет связи с дочерней таблицей, то запись будет удалена.

Чтобы добавить новое описание книги нужно нажать кнопку «Добавить», затем появится окно для введения информации о новом описании книги, его пример представлен на рисунке 4.26.

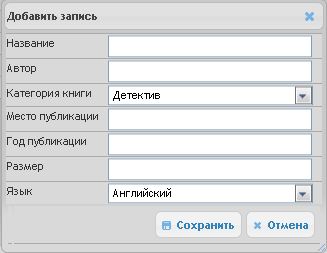


Рисунок 4.26 – Добавление нового описания книги

В данном окне необходимо заполнить поле «Название», «Автор» также нужно задать «Год публикации» в диапазоне от 1900 до 2020 и «Размер» в диапазоне от 1 до 9999. Затем следует нажать кнопку «Сохранить». Если же данные введены не корректно, то будет выведена соответствующая ошибка, пример которой показан на рисунке 4.27.

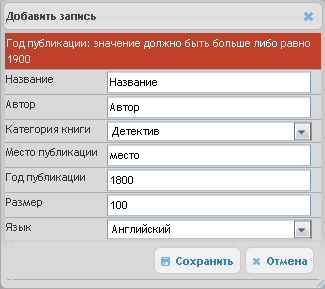


Рисунок 4.27 – Пример ошибки при добавлении нового описания книги

Чтобы отредактировать уже имеющееся описание книги, нужно выбрать его и нажать кнопку «Редактировать». Затем появится окно редактирования, которое показано на рисунке 4.28.

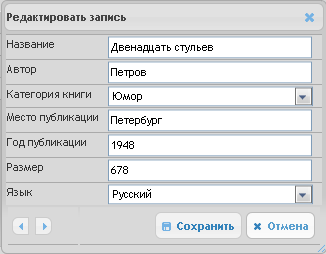


Рисунок 4.28 – Окно для редактирования описания книги

В этом окне нужно изменить данные и нажать кнопку «Сохранить», также можно осуществлять перемещение по записям, кликая на соответствующие стрелочки.

**4.4.2 Редактирование реальных книг**

Если библиотекарь выберет в пункте меню «Книги» подпункт «Реальные книги», тогда отобразится страница для работы с реальными книгами. Ее пример приведен на рисунке 4.29.

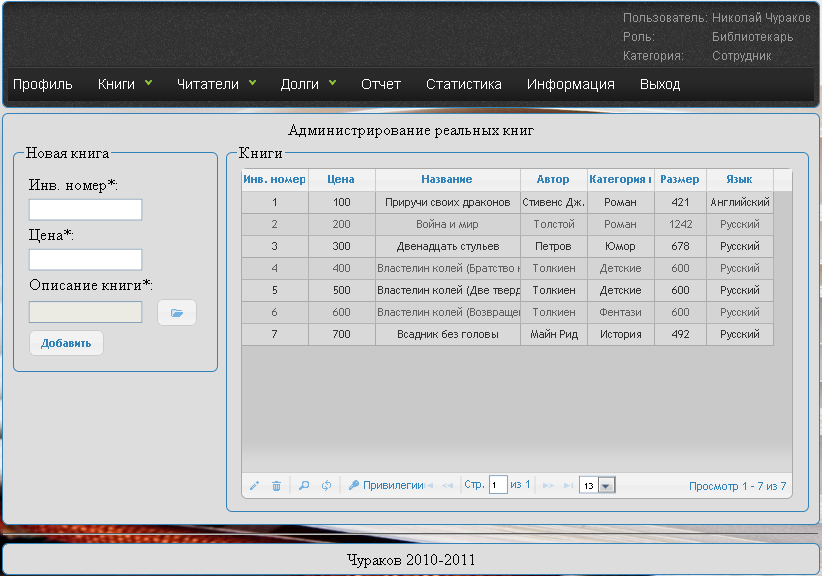


Рисунок 4.29 – Страница для работы с реальными книгами

На данной странице библиотекарь может осуществлять сортировку, поиск, обновление и удаление реальных книг. Также можно изменить цену для существующей книги. Для этого нужно выбрать книгу и нажать кнопку «Редактировать». Затем появится окно для редактирования цены реальной книги, ее пример отображен на рисунке 4.30.

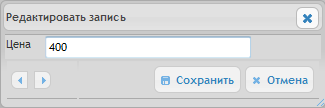


Рисунок 4.30 – Редактирование реальной книги

Требуется ввести новое значение и нажать кнопку «Сохранить».

Для того чтобы добавить новую реальную книгу, нужно ввести для нее инвентарный номер, затем цену, а потом нужно выбрать необходимое описание книги. Для этого нужно напротив поля «Описание книги» нажать на кнопку. Затем появится диалог для выбора описания книги. Нужно выбрать описание и нажать кнопку «Ок». Пример выбора описания приведен на рисунке 4.31.

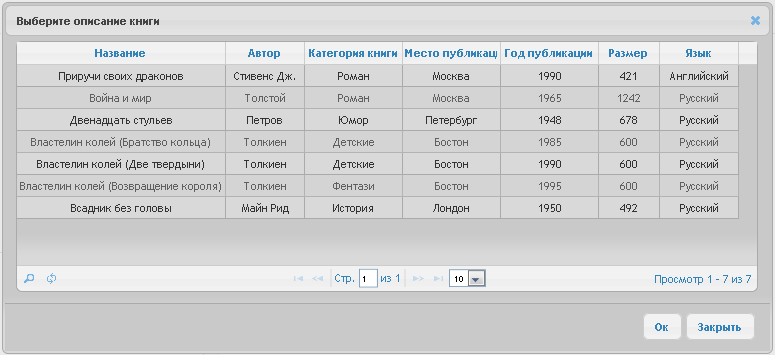


Рисунок 4.31 – Выбор описания книги

При создании новой книги, после того как будут заполнены все поля, нужно нажать на кнопку «Добавить». Если хотя бы одно поле не будет заполнено, появится предупреждение о необходимости заполнить соответствующее поле. После успешного добавления появится сообщение об успешном добавлении.

Если же библиотекарь введет инвентарный номер, который уже имеется в базе данных, то отобразится предупреждение о дублировании инвентарного номера.

Для того чтобы настроить область видимости для определенной книги, нужно выбрать ее и нажать кнопку «Привилегии». Затем появится окно для выбора категорий пользователей, которые смогут видеть данную книгу. Требуется выбрать нужные категории и нажать кнопку «Ок». Пример настройки привилегий приведен на рисунке 4.32.

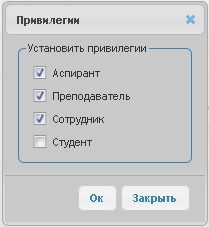


Рисунок 4.32 – Настройка привилегий

**4.4.3 Редактирование электронных книг**

Если библиотекарь выберет в пункте меню «Книги» подпункт «Электронные книги», тогда отобразится страница для работы с электронными книгами. Ее пример приведен на рисунке 3.33.

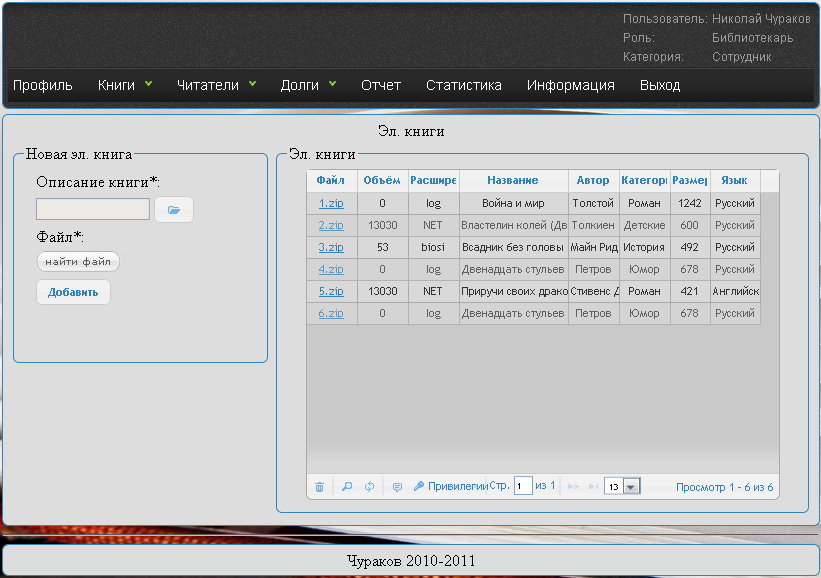


Рисунок 4.33 – Страница для работы с электронными книгами

На данной странице библиотекарь может осуществлять сортировку, поиск, обновление и удаление электронных книг.

Для того чтобы добавить новую электронную книгу, нужно выбрать подходящее описание книги, аналогично, как и при добавлении реальных книг. Затем нужно нажать кнопку «Найти файл» для выбора файла электронной книги. Затем появится окно выбора файла, пример приведен на рисунке 4.34.

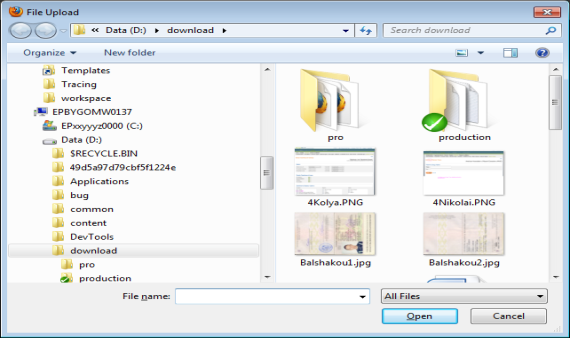


Рисунок 4.34 – Выбор файла для электронной книги

После выбора файла в соответствующем поле появится его название и иконка обозначающая расширение файла, смотри рисунок 4.35.

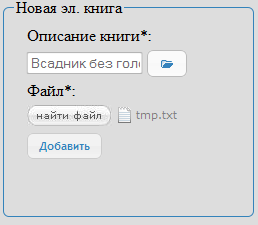


Рисунок 4.35 – Пример выбранного файла

Затем нужно нажать кнопку «Добавить» и, если все поля были заполнены, начнется загрузка файла на сервер. Библиотекарь может видеть состояние загрузки, пример отображения прогресса загрузки приведен на рисунке 4.36.

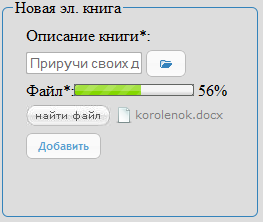


Рисунок 4.36 – Состояние загрузки файла на сервер

Настройка привилегий для электронных книг производится также как и для реальных книг.

Также библиотекарь может просматривать список комментариев для выбранной электронной книги. Для этого нужно выбрать электронную книгу и нажать кнопку «Показать комментарии». Работа с комментариями аналогична, как и у пользователя, за исключением того, что библиотекарь имеет возможность удалять любой комментарий.

**4.4.4 Выдача книг читателям**

Для того чтобы выдать читателю книгу нужно нажать в пункте меню «Читатели» подпункт «Выдать книгу», тогда появится страница выдачи книги. Ее пример отображен на рисунке 4.37.

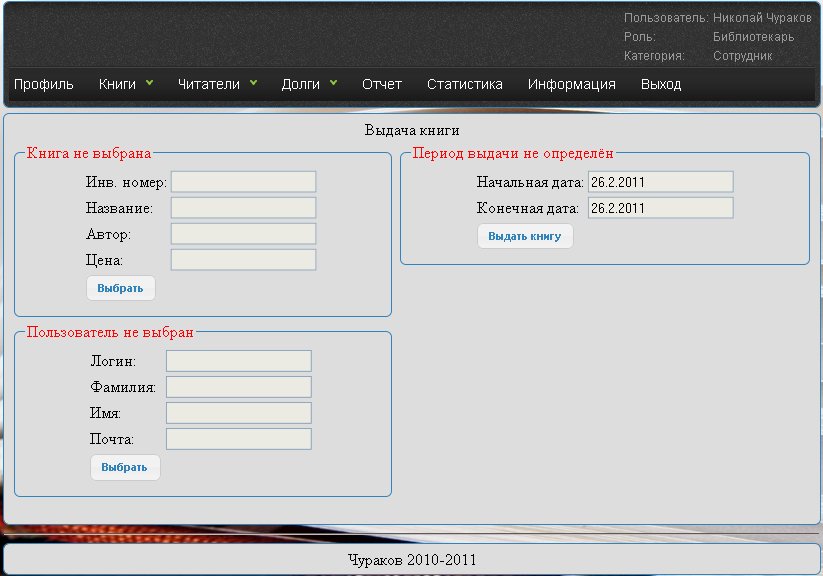


Рисунок 4.37 – Страница выдачи книги

Для успешной выдачи книги нужно выбрать книгу. Для этого нужно нажать кнопку «Выбрать» в секции с информацией о книге. После появится список книг. Пример приведен на рисунке 4.38.

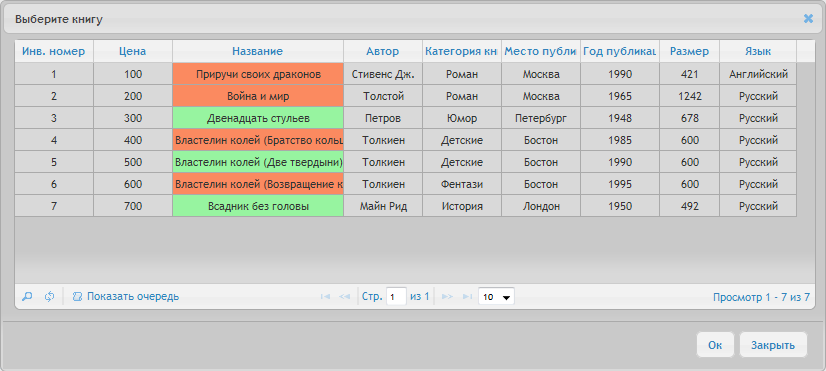


Рисунок 4.38 – Диалог выбора книги для выдачи

Красными цветом выделены книги, которых на данный момент нет в наличии, и при их выборе будет отображено сообщение об ошибке.

В диалоге выбора книги библиотекарь также может выбрать и пользователя, которому будет выдана книга. Пользователя можно взять из имеющейся очереди на определенную книгу. Для того чтобы выбрать пользователя нужно выбрать книгу и нажать кнопку «Показать очередь». Затем появится окно отображающее очередь на данную книгу. Пример приведен на рисунке 4.39.

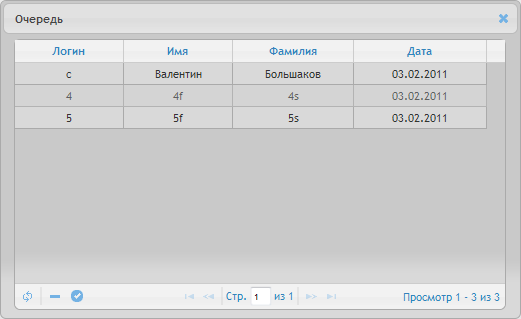


Рисунок 4.39 – Выбор пользователя из очереди на книгу

Затем следует выбрать пользователя и нажать на кнопку «Выбрать пользователя». Также в этом окне можно удалить пользователя из очереди. Для этого нужно выбрать его и нажать кнопку «Удалить из очереди». Затем появится сообщение с подтверждением удаления из очереди.

После того как книга выбрана, секция с информацией о книге будет заполнена, смотри рисунок 4.40.

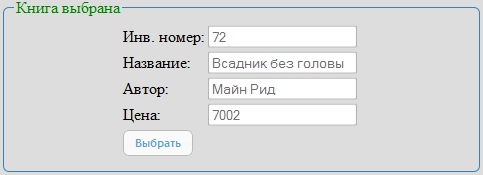


Рисунок 4.40 – Информация о выдаваемой книге

После выбора книги следует выбрать пользователя, которому данная книга будет выдана. Если при выборе книги пользователь не был выбран из очереди, то его можно добавить из общего списка пользователей. Для этого нужно в секции с информации о пользователе нажать кнопку «Выбрать». Затем появится окно со списком пользователей. Пример приведен на рисунке 4.41.

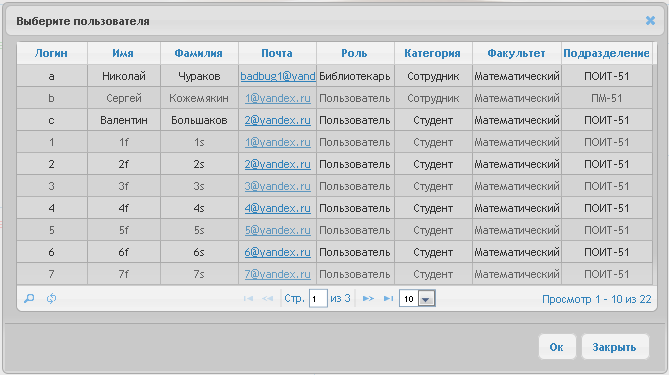


Рисунок 4.41 – Выбор пользователя для выдачи книги

Затем необходимо выбрать пользователя и нажать кнопку «Ок». После этого будет заполнена секция с информацией о пользователе. Пример приведен на рисунке 4.42.

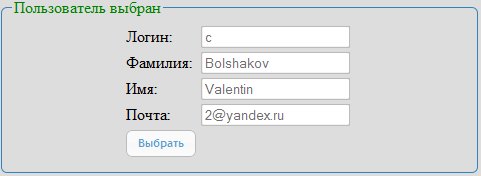


Рисунок 4.42 – Информация о пользователе, которому выдается книга

Также необходимо задать период, на который выдается книга. Для этого нужно используя выпадающий календарь, задать начальную дату и конечную дату. Пример задания даты показан на рисунке 4.43.

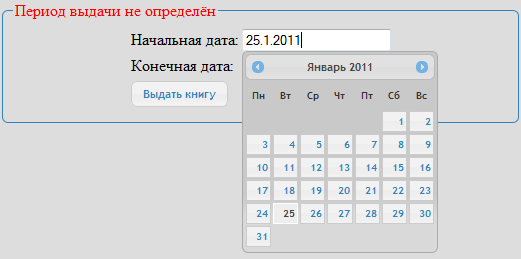


Рисунок 4.43 – Задание периода выдачи книги

После корректного задания периода, секция будет обозначена зеленым подтверждением, пример приведен на рисунке 4.44

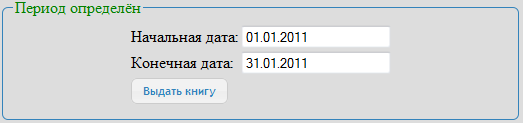


Рисунок 4.44 – Информация о периоде, на который выдается книга

Когда все секции будут заполнены, нужно нажать кнопку «Выдать книгу». Если хотя бы одна секция не будет заполнена, то появится соответствующее предупреждение. В случае успешной выдачи книги появится подтверждающее сообщение.

**4.4.5 Возврат книг в библиотеку**

Для того чтобы возвратить книгу нужно нажать в пункте меню «Читатели» подпункт «Вернуть книгу», тогда появится страница возврата книг. Ее пример отображен на рисунке 4.45.

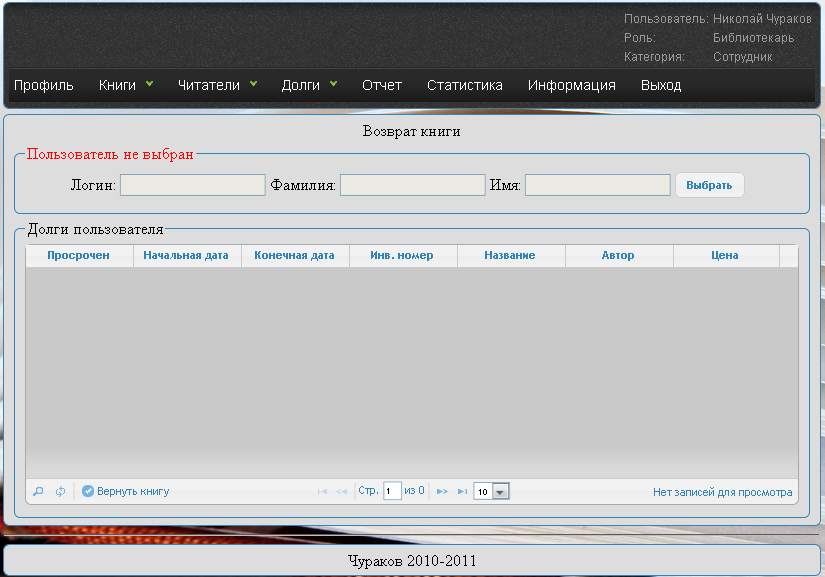


Рисунок 4.45 – Страница возврата книг

Чтобы вернуть книгу, сначала нужно выбрать пользователя, для этого в секции с информацией о пользователе нужно нажать кнопку «Выбрать». Затем появится окно со списком пользователей, пример показан на рисунке 4.46.

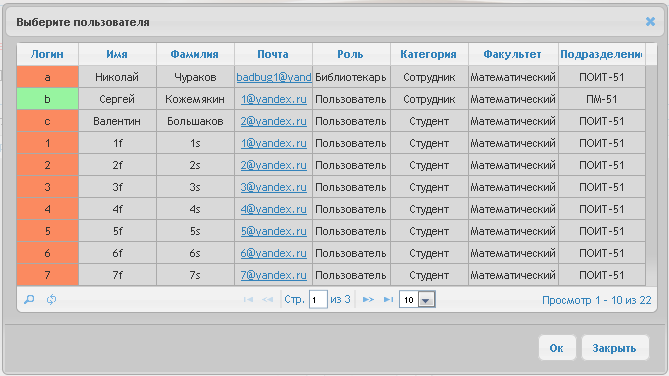


Рисунок 4.46 – Выбор пользователя для возврата книги

В данном списке красным цветом выделены пользователи, которые не имеют задолженностей и при попытке их выбрать появится предупреждение.

Зеленым цветом выделены пользователи, у которых есть задолженности. После выбора одного из них нужно нажать кнопку «Ок».

Затем заполнится секция с информацией о пользователе, а также отобразится список задолженностей для выбранного пользователя. Пример страницы «Вернуть книгу» после выбора пользователя, показан на рисунке 4.47.

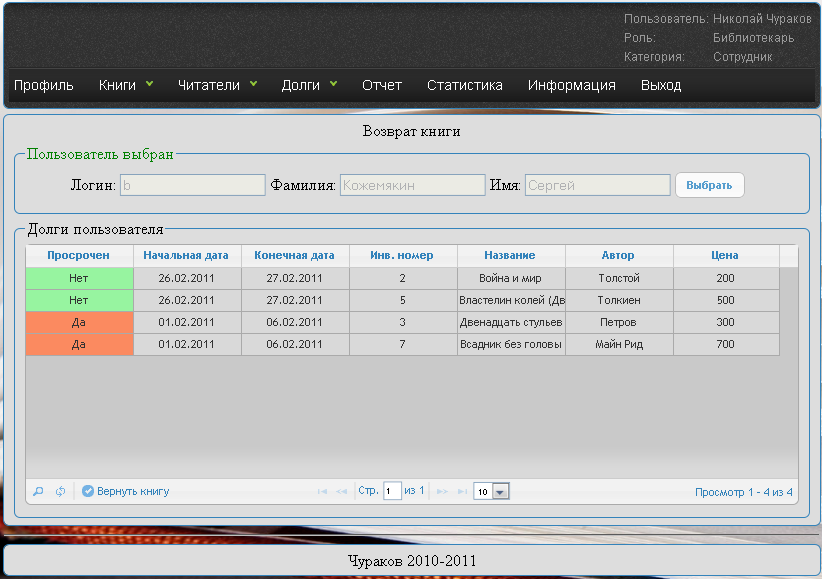


Рисунок 4.47 – Страница возврата книги после выбора пользователя

Затем нужно выбрать задолженность и нажать кнопку «Вернуть книгу». После успешного возврата книги появится подтверждение выдачи книги. После этого данная задолженность исчезнет из списка для выбранного пользователя.

Пункт меню «Долги» содержи подпункт «Все долги» и «Персональные долги». Данные пункты отображают страницу с задолженностями, аналогичную, как и страница задолженностей у пользователя. Пункт «Все долги» соответствует списку задолженностей всех пользователей, а пункт «Персональные долги» – только персональным задолженностям библиотекаря.

**4.4.6 Отчет по работе библиотекарей**

Для того чтобы просмотреть отчет по проделанной работе библиотекарями за определенный период, нужно выбрать в главном меню пункт «Отчет». Затем появится диалог для определения периода, пример приведен на рисунке 4.48.

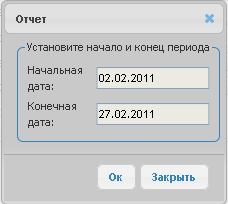


Рисунок 4.48 – Определение периода для отчета

Далее нужно задать период и нажать кнопку «Ок», после этого появится сгенерированный отчет в формате PDF. Пример отчета показан на рисунке 4.49.

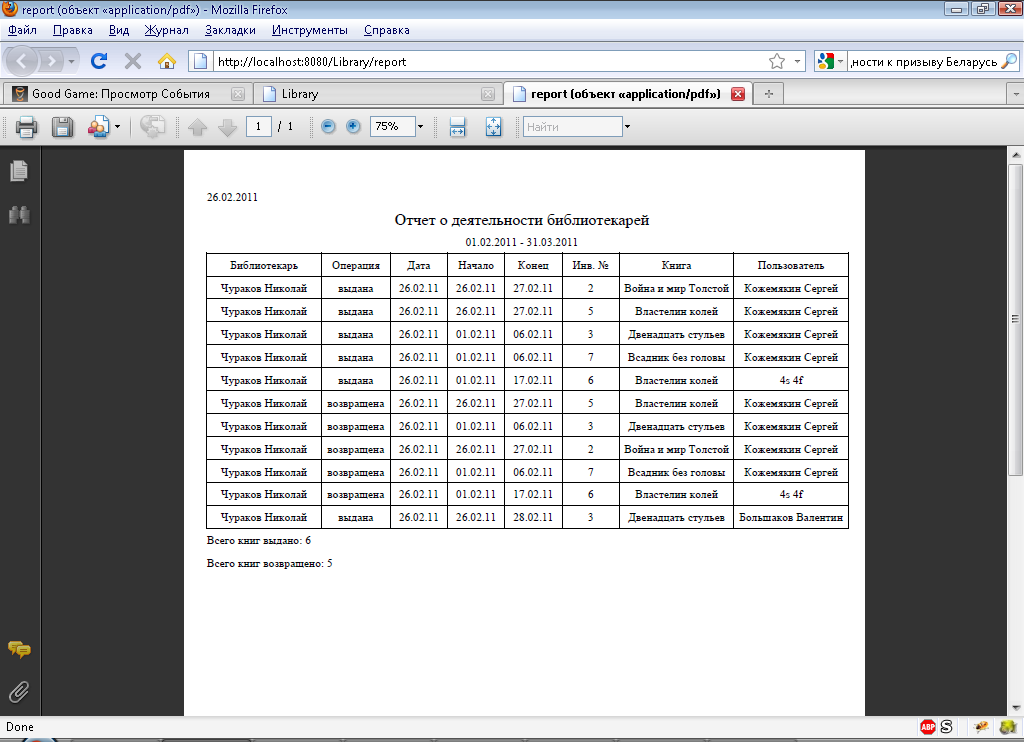


Рисунок 4.49 – Отчет по деятельности библиотекарей

**ЗАКЛЮЧЕНИЕ**

При выполнении данной дипломной работы было разработано веб-приложение «Библиотека высшего учебного заведения», а также детально описаны инструменты разработки и используемые технологии.

Для реализации серверной части был использован Microsoft SQL Server. Веб-приложение был развернуто на Apache Tomcat веб-сервере. Для программирования логики был использован фреймворк Struts 2, в основе которого лежит технология J2EE. Реализация веб-приложения проводилась в среде Eclipse. Для создания клиентской части были применены следующие технологии: HTML, CSS, JavaScript, jQuery, Ajax. При работе над проектом был использован инструмент управления версиями tortoiseSVN. Для отладки и тестирования отображаемых страниц широко применялись инструменты браузеров Fire Fox и Internet Explorer.

Разработанная система предоставляет пользователю выполнение следующих функций:

- Работа с реальными и электронными книгами.

- Выдача и возврат реальных книг.

- Поиск и сортировка каталога книг.

- Организация очереди на книги.

- Механизм комментариев.

- Учет задолженностей

- Предоставление статистических данных в виде графиков и отчетов.

- Поддержка русского и английского языка для отображаемого интерфейса.

- Возможность настройки доступа к книгам для разных категорий пользователей.

По результатам работы были опубликованы тезисы на студенческой научной конференции.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Грабер, М. Введение в SQL [Текст] / М. Грабер – М.: издательство «Лори», 1996. – 382с.
2. Кренке, Д. Теория и практика построения баз данных. 8-е изд. [Текст] / Д. Кренке – СПб.: Питер, 2003. – 800с.
3. Винкоп, С. Использование Microsoft SQL Server 7.0. Специальное издание [Текст] / С. Винкоп – СПб.: Издательский дом «Вильямс», 2001. – 816с.
4. Змитрович, А. И. Базы данных: Учебное пособие для вузов [Текст] / А. И. Змитрович – Мн.: Университетское, 1991. – 271с.
5. Риордан, Р. Программирование в Microsoft SQL Server 2000. Шаг за шагом. [Текст] / Р. Риордан – М.: Эком, 2002. – 608с
6. Хоторн, Р. Разработка баз данных Microsoft SQL Server 2000 на примерах. [Текст] / Р. Хоторн – М.: Бином, 2001. – 464с.
7. Moodie, M. Pro Apache Tomcat 6. [Текст] / M. Moodie – Apress. 2007 – 325 с.
8. Хабибулин, И. Самоучитель Java. Серия "Самоучитель". [Текст] / И. Хабибулин – СПб.: BHV-Санкт-Петербург, 2002 – 464 с.
9. Эккель, Б. Философия Java. Библиотека программиста. [Текст] / Б. Эккель – СПб. 2009 – 640 с.
10. Brown, D. Struts 2 in Action. [Текст] / D. Brown – Manning Publications, 2008 – 450 с.
11. Хейз, Д. Освой самостоятельно HTML. [Текст] / Д.Хейз – СПб. 2007 – 272 с.
12. Мальчук, Е. HTML и CSS. Самоучитель. [Текст] / Е. Мальчук – СПб. 2008 – 415 с.
13. Рева, О. JavaScriptв кармане. [Текст] / О. Рева – Спб. 2008 – 256 с.
14. Crane, D. Ajax - Ajax In Action. [Текст] / D. Crane – Manning Publications Co., 2006 – 681 c.
15. Katz, Y. jQuery in Action. [Текст] / Y. Katz – Manning Publications, 2010 – 475 c.

**ПРИЛОЖЕНИЕ А ЛИСТИНГ ПРОГРАММЫ**

Файл struts.xml:

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

<!DOCTYPE struts PUBLIC

"-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"

"http://struts.apache.org/dtds/struts-2.0.dtd">

<struts>

<constant name="struts.multipart.maxSize" value="5000000000" />

<constant name="struts.devMode" value="true" />

<constant name="struts.enable.DynamicMethodInvocation" value="true" />

<!-- For all -->

<package name="default" namespace="/" extends="tiles-default">

<global-results>

<result name="error" type="tiles">tiles\_error</result>

</global-results>

<global-exception-mappings>

<exception-mapping exception="java.lang.Exception" result="error" />

</global-exception-mappings>

<action name="loginPage" >

<result type="tiles">tiles\_login</result>

</action>

<action name="loginConfirm" class="actions.UserAction" method="loginConfirm">

<result name="input" type="tiles">tiles\_login</result>

<result name="user" type="redirectAction">

<param name="actionName">profile</param>

</result>

<result name="administrator" type="redirectAction">

<param name="actionName">userAdministration</param>

</result>

<result name="librarian" type="redirectAction">

<param name="actionName">profile</param>

</result>

</action>

<action name="registrationConfirm" class="actions.UserAction" method="registrationConfirm">

<result name="input" type="tiles">tiles\_registration</result>

<result name="success" type="tiles">tiles\_login</result>

</action>

<action name="registrationPage" class="actions.UserAction" method="loadDepartments">

<result type="tiles">tiles\_registration</result>

</action>

<action name="changeLanguage" >

<result type="tiles">tiles\_login</result>

</action>

<action name="exit" class="actions.ExitAction">

<result name="success" type="tiles">tiles\_login</result>

</action>

</package>

<!-- For all ajax actions-->

<package name="ajaxActions" extends="json-default" namespace="/">

<interceptors>

<interceptor-stack name="ajaxStack">

<interceptor-ref name="defaultStack" />

</interceptor-stack>

</interceptors>

<default-interceptor-ref name="ajaxStack" />

<global-results>

<result name="login" type="redirect">

login

</result>

<result name="error" type="redirect">

error

</result>

</global-results>

<global-exception-mappings>

<exception-mapping exception="java.lang.Exception" result="error" />

</global-exception-mappings>

<action name="prepareDivisions" class="actions.HandBookAction" method="prepareDivisions">

<result type="json">

<param name="root">result</param>

</result>

</action>

</package>

<!-- For users -->

<package name="userActions" namespace="/" extends="tiles-default">

<interceptors>

<interceptor name="loginInterceptor" class="interceptors.LoginInterceptor" />

<interceptor-stack name="userStack">

<interceptor-ref name="loginInterceptor" />

<interceptor-ref name="defaultStack" />

</interceptor-stack>

</interceptors>

<default-interceptor-ref name="userStack" />

<global-results>

<result name="error" type="tiles">tiles\_error</result>

<result name="login" type="tiles">tiles\_login</result>

</global-results>

<global-exception-mappings>

<exception-mapping exception="java.lang.Exception" result="error" />

</global-exception-mappings>

<action name="profile" class="actions.UserAction" method="profile">

<result name="success" type="tiles">tiles\_profile</result>

</action>

<action name="changeProfile" class="actions.UserAction" method="changeProfile">

<result name="success" type="tiles">tiles\_profile</result>

</action>

<action name="information">

<result name="success" type="tiles">tiles\_information</result>

</action>

<action name="debts" class="actions.DebtAction" method="debts">

<result name="success" type="tiles">tiles\_debts</result>

</action>

<action name="download" class="actions.FileAction" method="download">

<result name="sendFile" type="stream">

<param name="contentType">APPLICATION/OCTET-STREAM</param>

<param name="contentDisposition">attachment;filename="${fileName}"</param>

</result>

</action>

<action name="eBooksForUser" class="actions.ElectronicBookAction" method="eBooks">

<result name="success" type="tiles">tiles\_ebooks\_for\_user</result>

</action>

<action name="realBooksForUser" class="actions.RealBookAction" method="realBooks" >

<result name="success" type="tiles">tiles\_real\_books\_for\_user</result>

</action>

<action name="statistic" class="actions.StatisticAction" method="statistic">

<result name="success" type="tiles">tiles\_statistic</result>

</action>

</package>

<package name="UserAjaxActions" extends="json-default" namespace="/">

<interceptors>

<interceptor name="loginInterceptor" class="interceptors.LoginInterceptor" />

<interceptor-stack name="userStack">

<interceptor-ref name="loginInterceptor" />

<interceptor-ref name="defaultStack" />

</interceptor-stack>

</interceptors>

<default-interceptor-ref name="userStack" />

<global-results>

<result name="login" type="redirect">

login

</result>

<result name="error" type="redirect">

error

</result>

</global-results>

<global-exception-mappings>

<exception-mapping exception="java.lang.Exception" result="error" />

</global-exception-mappings>

<action name="getUserDebtsForUser" class="actions.DebtAction" method="prepareDebtsForUser">

<result type="json">

<param name="root">tableResult</param>

</result>

</action>

<action name="getElectronicBooksForUser" class="actions.ElectronicBookAction" method="prepareElectronicBooksForUser">

<result type="json">

<param name="root">tableResult</param>

</result>

</action>

<action name="getRealBooksForUser" class="actions.RealBookAction" method="prepareRealBooksForUser">

<result type="json">

<param name="root">tableResult</param>

</result>

</action>

<action name="addToQueue" class="actions.QueueAction" method="addToQueue">

<result type="json">

<param name="root">result</param>

</result>

</action>

<action name="deleteFromQueue" class="actions.QueueAction" method="deleteFromQueue">

<result type="json">

<param name="root">result</param>

</result>

</action>

<action name="getQueueForUser" class="actions.QueueAction" method="prepareQueues">

<result type="json">

<param name="root">tableResult</param>

</result>

</action>

<action name="loadCommentsForUser" class="actions.CommentAction" method="loadComments">

<result type="json">

<param name="root">result</param>

</result>

</action>

<action name="addCommentForUser" class="actions.CommentAction" method="addComment">

<result type="json">

<param name="root">result</param>

</result>

</action>

<action name="deleteCommentForUser" class="actions.CommentAction" method="deleteComment">

<result type="json">

<param name="root">result</param>

</result>

</action>

</package>

<!-- For administrator -->

<package name="administratorActions" namespace="/" extends="tiles-default">

<interceptors>

<interceptor name="loginInterceptor" class="interceptors.LoginInterceptor" />

<interceptor name="administratorInterceptor" class="interceptors.AdministratorInterceptor" />

<interceptor-stack name="administratorStack">

<interceptor-ref name="loginInterceptor" />

<interceptor-ref name="administratorInterceptor" />

<interceptor-ref name="defaultStack" />

</interceptor-stack>

</interceptors>

<default-interceptor-ref name="administratorStack" />

<global-results>

<result name="error" type="tiles">tiles\_error</result>

<result name="login" type="tiles">tiles\_login</result>

</global-results>

<global-exception-mappings>

<exception-mapping exception="java.lang.Exception" result="error" />

</global-exception-mappings>

<action name="userAdministration" class="actions.UserAction" >

<result name="success" type="tiles">tiles\_users</result>

</action>

<action name="departmentAdministration" class="actions.HandBookAction" >

<result name="success" type="tiles">tiles\_departments</result>

</action>

<action name="bookCategoryAdministration" class="actions.HandBookAction">

<result name="success" type="tiles">tiles\_book\_categories</result>

</action>

<action name="languageAdministration" class="actions.HandBookAction">

<result name="success" type="tiles">tiles\_languages</result>

</action>

<action name="userCategoryAdministration" class="actions.HandBookAction">

<result name="success" type="tiles">tiles\_user\_categories</result>

</action>

<action name="divisionAdministration" class="actions.HandBookAction" method="loadDepartments">

<result name="success" type="tiles">tiles\_divisions</result>

</action>

</package>

<package name="AdministratorAjaxActions" extends="json-default" namespace="/">

<interceptors>

<interceptor name="loginInterceptor" class="interceptors.LoginInterceptor" />

<interceptor name="administratorInterceptor" class="interceptors.AdministratorInterceptor" />

<interceptor-stack name="administratorStack">

<interceptor-ref name="loginInterceptor" />

<interceptor-ref name="administratorInterceptor" />

<interceptor-ref name="defaultStack" />

</interceptor-stack>

</interceptors>

<default-interceptor-ref name="administratorStack" />

<global-results>

<result name="login" type="redirect">

login

</result>

<result name="error" type="redirect">

error

</result>

</global-results>

<global-exception-mappings>

<exception-mapping exception="java.lang.Exception" result="error" />

</global-exception-mappings>

<action name="getUsers" class="actions.UserAction" method="prepareUsers">

<result type="json">

<param name="root">tableResult</param>

</result>

</action>

<action name="getDepartments" class="actions.HandBookAction" method="prepareDepartments">

<result type="json">

<param name="root">tableResult</param>

</result>

</action>

<action name="editUser" class="actions.UserAction" method="editUser">

<result type="json">

<param name="root">result</param>

</result>

</action>

<action name="editDepartment" class="actions.HandBookAction" method="editDepartment">

<result type="json">

<param name="root">result</param>

</result>

</action>

<action name="getBookCategories" class="actions.HandBookAction" method="prepareBookCategories">

<result type="json">

<param name="root">tableResult</param>

</result>

</action>

<action name="editBookCategory" class="actions.HandBookAction" method="editBookCategory">

<result type="json">

<param name="root">result</param>

</result>

</action>

<action name="getLanguages" class="actions.HandBookAction" method="prepareLanguages">

<result type="json">

<param name="root">tableResult</param>

</result>

</action>

<action name="editLanguage" class="actions.HandBookAction" method="editLanguage">

<result type="json">

<param name="root">result</param>

</result>

</action>

<action name="getUserCategories" class="actions.HandBookAction" method="prepareUserCategories">

<result type="json">

<param name="root">tableResult</param>

</result>

</action>

<action name="editUserCategory" class="actions.HandBookAction" method="editUserCategory">

<result type="json">

<param name="root">result</param>

</result>

</action>

<action name="getDivisions" class="actions.HandBookAction" method="getDivisions">

<result type="json">

<param name="root">tableResult</param>

</result>

</action>

<action name="editDivision" class="actions.HandBookAction" method="editDivision">

<result type="json">

<param name="root">result</param>

</result>

</action>

</package>

<!-- For librarian -->

<package name="librarianActions" namespace="/" extends="tiles-default">

<interceptors>

<interceptor name="loginInterceptor" class="interceptors.LoginInterceptor" />

<interceptor name="librarianInterceptor" class="interceptors.LibrarianInterceptor" />

<interceptor-stack name="librarianStack">

<interceptor-ref name="loginInterceptor" />

<interceptor-ref name="librarianInterceptor" />

<interceptor-ref name="defaultStack" />

</interceptor-stack>

</interceptors>

<default-interceptor-ref name="librarianStack" />

<global-results>

<result name="error" type="tiles">tiles\_error</result>

<result name="login" type="tiles">tiles\_login</result>

</global-results>

<global-exception-mappings>

<exception-mapping exception="java.lang.Exception" result="error" />

</global-exception-mappings>

<action name="bookDescriptions" class="actions.BookDescriptionAction" method="bookDescriptions" >

<result name="success" type="tiles">tiles\_book\_descriptions</result>

</action>

<action name="realBooks" class="actions.RealBookAction" method="realBooks" >

<result name="success" type="tiles">tiles\_real\_books</result>

</action>

<action name="giveBook" class="actions.DebtAction" method="giveBook" >

<result name="success" type="tiles">tiles\_give\_book</result>

</action>

<action name="returnBook" class="actions.DebtAction" method="returnBook" >

<result name="success" type="tiles">tiles\_return\_book</result>

</action>

<action name="eBooks" class="actions.ElectronicBookAction" method="eBooks" >

<result name="success" type="tiles">tiles\_ebooks</result>

</action>

<action name="allDebts" class="actions.DebtAction" method="debts">

<result name="success" type="tiles">tiles\_all\_debts</result>

</action>

</package>

<package name="LibrarianAjaxActions" extends="json-default" namespace="/">

<interceptors>

<interceptor name="loginInterceptor" class="interceptors.LoginInterceptor" />

<interceptor name="librarianInterceptor" class="interceptors.LibrarianInterceptor" />

<interceptor-stack name="librarianStack">

<interceptor-ref name="loginInterceptor" />

<interceptor-ref name="librarianInterceptor" />

<interceptor-ref name="defaultStack" />

</interceptor-stack>

</interceptors>

<default-interceptor-ref name="librarianStack" />

<global-results>

<result name="login" type="redirect">

login

</result>

<result name="error" type="redirect">

error

</result>

</global-results>

<global-exception-mappings>

<exception-mapping exception="java.lang.Exception" result="error" />

</global-exception-mappings>

<action name="getBookDescriptions" class="actions.BookDescriptionAction" method="prepareBookDescriptions">

<result type="json">

<param name="root">tableResult</param>

</result>

</action>

<action name="editBookDescription" class="actions.BookDescriptionAction" method="editBookDescription">

<result type="json">

<param name="root">result</param>

</result>

</action>

<action name="getRealBooks" class="actions.RealBookAction" method="prepareRealBooks">

<result type="json">

<param name="root">tableResult</param>

</result>

</action>

<action name="editRealBook" class="actions.RealBookAction" method="editRealBook">

<result type="json">

<param name="root">result</param>

</result>

</action>

<action name="addRealBook" class="actions.RealBookAction" method="addRealBook">

<result type="json">

<param name="root">result</param>

</result>

</action>

<action name="getUsersForLibrarian" class="actions.UserAction" method="prepareActiveUsers">

<result type="json">

<param name="root">tableResult</param>

</result>

</action>

<action name="giveRealBook" class="actions.DebtAction" method="giveRealBook" >

<result type="json">

<param name="root">result</param>

</result>

</action>

<action name="getUserDebtsForLibrarian" class="actions.DebtAction" method="prepareDebtsForUser">

<result type="json">

<param name="root">tableResult</param>

</result>

</action>

<action name="returnRealBook" class="actions.DebtAction" method="returnRealBook" >

<result type="json">

<param name="root">result</param>

</result>

</action>

<action name="uploadProgress" class="actions.FileAction" method="uploadProgress">

<result type="json">

<param name="root">result</param>

</result>

</action>

<action name="getElectronicBooksForLibrarian" class="actions.ElectronicBookAction" method="prepareElectronicBooks">

<result type="json">

<param name="root">tableResult</param>

</result>

</action>

<action name="editElectronicBook" class="actions.ElectronicBookAction" method="editElectronicBook">

<result type="json">

<param name="root">result</param>

</result>

</action>

<action name="getQueueForLibrarian" class="actions.QueueAction" method="prepareQueues">

<result type="json">

<param name="root">tableResult</param>

</result>

</action>

<action name="deleteFromQueueById" class="actions.QueueAction" method="deleteFromQueueById">

<result type="json">

<param name="root">result</param>

</result>

</action>

<action name="getAllDebts" class="actions.DebtAction" method="prepareAllDebts">

<result type="json">

<param name="root">tableResult</param>

</result>

</action>

<action name="loadComments" class="actions.CommentAction" method="loadComments">

<result type="json">

<param name="root">result</param>

</result>

</action>

<action name="loadCommentsForLibrarian" class="actions.CommentAction" method="loadComments">

<result type="json">

<param name="root">result</param>

</result>

</action>

<action name="addComment" class="actions.CommentAction" method="addComment">

<result type="json">

<param name="root">result</param>

</result>

</action>

<action name="deleteComment" class="actions.CommentAction" method="deleteComment">

<result type="json">

<param name="root">result</param>

</result>

</action>

<action name="getRealPrivileges" class="actions.RealBookAction" method="getRealPrivileges">

<result type="json">

<param name="root">result</param>

</result>

</action>

<action name="setRealPrivileges" class="actions.RealBookAction" method="setRealPrivileges">

<result type="json">

<param name="root">result</param>

</result>

</action>

<action name="getElectronicPrivileges" class="actions.ElectronicBookAction" method="getElectronicPrivileges">

<result type="json">

<param name="root">result</param>

</result>

</action>

<action name="setElectronicPrivileges" class="actions.ElectronicBookAction" method="setElectronicPrivileges">

<result type="json">

<param name="root">result</param>

</result>

</action>

</package>

<package name="ajaxfileupload" namespace="/" extends="ajaxfileupload-default">

<action name="upload" class="actions.FileAction" method="upload">

<interceptor-ref name="fileUploadStack" />

<result name="success" type="httpheader">

<param name="status">200</param>

</result>

</action>

</package>

<package name="jasper" namespace="/" extends="jasperreports-default">

<result-types>

<result-type name="jasper" class="org.apache.struts2.views.jasperreports.JasperReportsResult"/>

</result-types>

<interceptors>

<interceptor name="loginInterceptor" class="interceptors.LoginInterceptor" />

<interceptor name="librarianInterceptor" class="interceptors.LibrarianInterceptor" />

<interceptor-stack name="librarianStack">

<interceptor-ref name="loginInterceptor" />

<interceptor-ref name="librarianInterceptor" />

<interceptor-ref name="defaultStack" />

</interceptor-stack>

</interceptors>

<default-interceptor-ref name="librarianStack" />

<global-results>

<result name="login" type="redirect">

login

</result>

<result name="error" type="redirect">

error

</result>

</global-results>

<global-exception-mappings>

<exception-mapping exception="java.lang.Exception" result="error" />

</global-exception-mappings>

<action name="report" class="actions.ReportAction">

<result name="success" type="jasper">

<param name="location">/reports/report\_compile.jasper</param>

<param name="dataSource">records</param>

<param name="format">PDF</param>

<param name="reportParameters">reportParameters</param>

</result>

</action>

</package>

</struts>

Файл tiles.xml:

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

<!DOCTYPE tiles-definitions PUBLIC

"-//Apache Software Foundation//DTD Tiles Configuration 2.0//EN"

"http://tiles.apache.org/dtds/tiles-config\_2\_0.dtd">

<tiles-definitions>

<definition name="base" template="/WEB-INF/view/verticalTemplate.jsp">

<put-attribute name="head" value="" />

<put-attribute name="title" value="Library" />

<put-attribute name="navigation" value="" />

<put-attribute name="topBlock" value="" />

<put-attribute name="middleBlock" value="" />

<put-attribute name="bottomBlock" value="" />

<put-attribute name="footer" value="/WEB-INF/view/footer.jsp" />

</definition>

<definition name="tiles\_login" extends="base">

<put-attribute name="head" value="/WEB-INF/view/login\_head.tile" />

<put-attribute name="navigation" value="" />

<put-attribute name="topBlock" value="" />

<put-attribute name="middleBlock" value="/WEB-INF/view/login.jsp" />

<put-attribute name="bottomBlock" value="" />

</definition>

<definition name="tiles\_registration" extends="base">

<put-attribute name="head" value="/WEB-INF/view/login\_head.tile" />

<put-attribute name="navigation" value="" />

<put-attribute name="topBlock" value="" />

<put-attribute name="middleBlock" value="/WEB-INF/view/registration.jsp" />

<put-attribute name="bottomBlock" value="" />

</definition>

<definition name="tiles\_profile" extends="base">

<put-attribute name="head" value="/WEB-INF/view/menu\_head.tile" />

<put-attribute name="navigation" value="/WEB-INF/view/header.jsp" />

<put-attribute name="topBlock" value="" />

<put-attribute name="middleBlock" value="/WEB-INF/view/profile.jsp" />

<put-attribute name="bottomBlock" value="" />

</definition>

<definition name="tiles\_users" extends="base">

<put-attribute name="head" value="/WEB-INF/view/menu\_table\_head.jsp" />

<put-attribute name="navigation" value="/WEB-INF/view/header.jsp" />

<put-attribute name="topBlock" value="" />

<put-attribute name="middleBlock" value="/WEB-INF/view/users.jsp" />

<put-attribute name="bottomBlock" value="" />

</definition>

<definition name="tiles\_book\_descriptions" extends="base">

<put-attribute name="head" value="/WEB-INF/view/menu\_table\_head.jsp" />

<put-attribute name="navigation" value="/WEB-INF/view/header.jsp" />

<put-attribute name="topBlock" value="" />

<put-attribute name="middleBlock" value="/WEB-INF/view/bookDescriptions.jsp" />

<put-attribute name="bottomBlock" value="" />

</definition>

<definition name="tiles\_information" extends="base">

<put-attribute name="head" value="/WEB-INF/view/menu\_head.tile" />

<put-attribute name="navigation" value="/WEB-INF/view/header.jsp" />

<put-attribute name="topBlock" value="" />

<put-attribute name="middleBlock" value="/WEB-INF/view/information.jsp" />

<put-attribute name="bottomBlock" value="" />

</definition>

<definition name="tiles\_real\_books" extends="base">

<put-attribute name="head" value="/WEB-INF/view/books\_head.jsp" />

<put-attribute name="navigation" value="/WEB-INF/view/header.jsp" />

<put-attribute name="topBlock" value="" />

<put-attribute name="middleBlock" value="/WEB-INF/view/realBooks.jsp" />

<put-attribute name="bottomBlock" value="" />

</definition>

<definition name="tiles\_give\_book" extends="base">

<put-attribute name="head" value="/WEB-INF/view/books\_head.jsp" />

<put-attribute name="navigation" value="/WEB-INF/view/header.jsp" />

<put-attribute name="topBlock" value="" />

<put-attribute name="middleBlock" value="/WEB-INF/view/giveBook.jsp" />

<put-attribute name="bottomBlock" value="" />

</definition>

<definition name="tiles\_return\_book" extends="base">

<put-attribute name="head" value="/WEB-INF/view/books\_head.jsp" />

<put-attribute name="navigation" value="/WEB-INF/view/header.jsp" />

<put-attribute name="topBlock" value="" />

<put-attribute name="middleBlock" value="/WEB-INF/view/returnBook.jsp" />

<put-attribute name="bottomBlock" value="" />

</definition>

<definition name="tiles\_debts" extends="base">

<put-attribute name="head" value="/WEB-INF/view/books\_head.jsp" />

<put-attribute name="navigation" value="/WEB-INF/view/header.jsp" />

<put-attribute name="topBlock" value="" />

<put-attribute name="middleBlock" value="/WEB-INF/view/debts.jsp" />

<put-attribute name="bottomBlock" value="" />

</definition>

<definition name="tiles\_ebooks" extends="base">

<put-attribute name="head" value="/WEB-INF/view/ebooks\_head.jsp" />

<put-attribute name="navigation" value="/WEB-INF/view/header.jsp" />

<put-attribute name="topBlock" value="" />

<put-attribute name="middleBlock" value="/WEB-INF/view/eBooks.jsp" />

<put-attribute name="bottomBlock" value="" />

</definition>

<definition name="tiles\_ebooks\_for\_user" extends="base">

<put-attribute name="head" value="/WEB-INF/view/ebooks\_head.jsp" />

<put-attribute name="navigation" value="/WEB-INF/view/header.jsp" />

<put-attribute name="topBlock" value="" />

<put-attribute name="middleBlock" value="/WEB-INF/view/eBooks\_for\_user.jsp" />

<put-attribute name="bottomBlock" value="" />

</definition>

<definition name="tiles\_real\_books\_for\_user" extends="base">

<put-attribute name="head" value="/WEB-INF/view/books\_head.jsp" />

<put-attribute name="navigation" value="/WEB-INF/view/header.jsp" />

<put-attribute name="topBlock" value="" />

<put-attribute name="middleBlock" value="/WEB-INF/view/realBooks\_for\_user.jsp" />

<put-attribute name="bottomBlock" value="" />

</definition>

<definition name="tiles\_all\_debts" extends="base">

<put-attribute name="head" value="/WEB-INF/view/books\_head.jsp" />

<put-attribute name="navigation" value="/WEB-INF/view/header.jsp" />

<put-attribute name="topBlock" value="" />

<put-attribute name="middleBlock" value="/WEB-INF/view/all\_debts.jsp" />

<put-attribute name="bottomBlock" value="" />

</definition>

<definition name="tiles\_statistic" extends="base">

<put-attribute name="head" value="/WEB-INF/view/statistic\_head.tile" />

<put-attribute name="navigation" value="/WEB-INF/view/header.jsp" />

<put-attribute name="topBlock" value="" />

<put-attribute name="middleBlock" value="/WEB-INF/view/statistic.jsp" />

<put-attribute name="bottomBlock" value="" />

</definition>

<definition name="tiles\_departments" extends="base">

<put-attribute name="head" value="/WEB-INF/view/menu\_table\_head.jsp" />

<put-attribute name="navigation" value="/WEB-INF/view/header.jsp" />

<put-attribute name="topBlock" value="" />

<put-attribute name="middleBlock" value="/WEB-INF/view/departments.jsp" />

<put-attribute name="bottomBlock" value="" />

</definition>

<definition name="tiles\_book\_categories" extends="base">

<put-attribute name="head" value="/WEB-INF/view/menu\_table\_head.jsp" />

<put-attribute name="navigation" value="/WEB-INF/view/header.jsp" />

<put-attribute name="topBlock" value="" />

<put-attribute name="middleBlock" value="/WEB-INF/view/bookCategories.jsp" />

<put-attribute name="bottomBlock" value="" />

</definition>

<definition name="tiles\_languages" extends="base">

<put-attribute name="head" value="/WEB-INF/view/menu\_table\_head.jsp" />

<put-attribute name="navigation" value="/WEB-INF/view/header.jsp" />

<put-attribute name="topBlock" value="" />

<put-attribute name="middleBlock" value="/WEB-INF/view/languages.jsp" />

<put-attribute name="bottomBlock" value="" />

</definition>

<definition name="tiles\_user\_categories" extends="base">

<put-attribute name="head" value="/WEB-INF/view/menu\_table\_head.jsp" />

<put-attribute name="navigation" value="/WEB-INF/view/header.jsp" />

<put-attribute name="topBlock" value="" />

<put-attribute name="middleBlock" value="/WEB-INF/view/userCategories.jsp" />

<put-attribute name="bottomBlock" value="" />

</definition>

<definition name="tiles\_divisions" extends="base">

<put-attribute name="head" value="/WEB-INF/view/menu\_table\_head.jsp" />

<put-attribute name="navigation" value="/WEB-INF/view/header.jsp" />

<put-attribute name="topBlock" value="" />

<put-attribute name="middleBlock" value="/WEB-INF/view/divisions.jsp" />

<put-attribute name="bottomBlock" value="" />

</definition>

<definition name="tiles\_error" extends="base">

<put-attribute name="middleBlock" value="/WEB-INF/view/error.jsp" />

</definition>

</tiles-definitions>