МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

**федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ**

**ГИДРОМЕТЕОРОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»**

Институт информационных систем и геотехнологий

Направление подготовки 09.03.03 «Прикладная информатика»

Курсовая работа

Проектирование автоматизированной библиотечной информационной системы (АБИС)

Студент очной формы обучения

3 курса, группы ПИ–Б16–1–1

Курдов Агарустам

Приняла:

Доцент кафедры прикладной информатики

Колбина О.Н.

Санкт–Петербург

2019

**Оглавление**

[Введение 3](#_Toc9391683)

[1. Проектирование информационной системы 4](#_Toc9391684)

[1.1 Анализ бизнес-процессов 4](#_Toc9391685)

[1.2 Анализ аналогичных ИС 6](#_Toc9391686)

[1.3 Разработка концепции ИС 9](#_Toc9391687)

[2. Разработка информационной системы 11](#_Toc9391688)

[2.1 Проектирование баз данных 11](#_Toc9391689)

[2.2 Проектирование приложения 14](#_Toc9391690)

[3. Реализация ИС 16](#_Toc9391691)

[3.1 Функциональные требования 16](#_Toc9391692)

[3.2 Выбор инструментов для разработки 19](#_Toc9391693)

[3.3 Проектирование Базы Данных 20](#_Toc9391694)

[3.4 Разработка клиентского приложения 22](#_Toc9391695)

[Заключение 26](#_Toc9391696)

[Список источников 27](#_Toc9391697)

[Приложение A 28](#_Toc9391698)

[Приложение B 29](#_Toc9391699)

# Введение

В условиях активной и непрекращающейся конкуренции повышение результатов деятельности организации зависит от внедрения информационных технологий, ускоряющих и упрощающих документооборот, расчеты и прочие действия, носящие повторяющийся характер.

Очевидно, что с ростом спроса на подобные продукты, повышается и потребность в специалистах рынка программного обеспечения.

Целью курсовой работы является получение первичных умений и навыков проектирования информационных продуктов посредством выполнения конкретного индивидуального задания.

В процессе работы решаются следующие основные задачи:

* Подготовка к полноценному восприятию последующих дисциплин, ориентированных на получение компетенций в области проектирования информационных систем;
* Выработка необходимых умений и навыков использования компьютерной техники и программного обеспечения в будущей профессиональной деятельности.

Целью проекта является разработка системы для локального рабочего места.

В рамках задания необходимо решить следующие задачи:

* Анализ проблемной ситуации;
* Разработка концепции проекта;
* Разработка сопроводительной документации (в том числе и ТЗ).

Для решения поставленных задач было решено разрабатывать приложение на языке программирования Java с подключением базы данных.

Предметом задания является создание журнала учета выдаваемых на прокат книг и получения отчета по финансовым результатам для библиотеки.

# 1. Проектирование информационной системы

Этап проектирования информационной системы включает в себя ряд процессов, призванных формализовать первичный сбор и обработку исходных данных, для дальнейшей декомпозиции на конкретные задачи в разработке программного продукта.

Важное место занимает анализ требований конкретного заказчика, исходящих из сложившихся бизнес-процессов.

Следующий этап включает анализ аналогичных продуктов и решений, которые может предложить рынок.

После подготовки технического задания, являющегося формализованной формой представления бизнес-требований, сложившихся в результате обработки первичной информации, происходит непосредственное выделение задач, передаваемых инженерам.

Важное место занимает планирование шагов решения выявленных задач, с приближенной эстимацией временных затрат.

Популярным методом подобного рода планирования является создание диаграммы Гантта, с предварительной оценкой времени на реализацию определенного функционала системы и общую иерархию выделенных сторей и тасок в рамках проекта.

## 1.1 Анализ бизнес-процессов

Библиотекой осуществляется дифференцированное библиотечно-информационное обслуживание читателей, систематически ведется дневник библиотеки, в котором учитываются сведения о количестве и составе читателей по группам, об объеме выданных изданий и распределении их по отделам библиотечной классификации; дополнительно в дневник введены графы, характеризующие объем выданных учебников, методической литературы, посещений и книговыдачи.

Основные формы индивидуального обслуживания: беседа при записи в библиотеку, беседа при выдаче документов, беседа о прочитанном; экспресс-консультация, блиц-реклама, анализ читательских формуляров, творческие работы.

Основными задачами библиотеки являются:

* Обеспечение участникам образовательного процесса — обучающимся, педагогическим работникам, - доступа к информации, знаниям, идеям, культурным ценностям посредством использования библиотечно-информационных ресурсов на различных носителях: бумажном (книжный фонд, фонд периодических изданий); коммуникативном (компьютерные сети) и иных носителях;
* Воспитание культурного и гражданского самосознания, помощь в социализации обучающегося, развитии его творческого потенциала;
* Формирование навыков независимого библиотечного пользователя: обучение поиску, отбору и критической оценке информации;
* Совершенствование предоставляемых библиотекой услуг на основе внедрения новых информационных технологий и компьютеризации библиотечно-информационных процессов, формирование комфортной библиотечной среды.

## 1.2 Анализ аналогичных ИС

Автоматизированные библиотечные информационные системы (АБИС) — системы планирования ресурсов предприятий для библиотеки, которые используются для отслеживания библиотечных фондов, от их заказа и приобретения до выдачи посетителям библиотек.

Каждый читатель (посетитель) и экземпляр имеют уникальный идентификатор в базе данных, которая позволяет АБИС отслеживать деятельность.

Большие библиотеки используют АБИС, чтобы заказывать и покупать, каталогизировать, распространять книги и другие фонды, резервировать материалы и отслеживать их возврат. Небольшие библиотеки часто используют некоторые из этих возможностей.

Все крупнейшие библиотеки мира используют АБИС.

Отдельные функции программного обеспечения большинства АБИС представляют собой функционально завершенные модули, которые объединены в общий интерфейс. Примерный перечень модулей включает:

* Приобретение фондов (заказ, выставление счетов и получение);
* Каталогизация (внесение и библиографическое описание экземпляров);
* Обращение (выдача/возврат экземпляров читателям);
* Периодику (отслеживание журналов и газет);
* OPAC — общедоступный электронный каталог и интерфейс для пользователей.

Рассмотрим наиболее широко используемые в России АБИС. Подробнее остановимся на системе, являющейся отечественной разработкой – Колибри. Также, изучим популярный зарубежный продукт – АБИС Koha, являющийся одним из первых бесплатных систем своего рода.

Система «Колибри» работает со всеми европейскими алфавитами, включая греческий. В случае необходимости активизируются диактрические символы. При описании издания могут быть одновременно использованы русский, латинский и один из европейских языков.

Система выполнена в виде модулей, которые работают в едином комплексе. Система использует СУБД BTrieve фирмы Pervasive Technology, сетевой вариант системы работает на серверах под управлением ОС Novell NetWare и в сетях Microsoft.

Информация, накопленная при работе системы, может быть передана в других системы через формат представления данных в стадарте US MARC.

В свою очередь, АБИС Koha имеет большое количество доступных языков и их количество увеличивается ежегодно.

Мощный поисковый модуль и встроенный каталог, интегрированный с поисковыми системами онлайн библиотек, как Amazon, Google, Library Thing, Open Library, Syndetics и т.д.

Koha построен с использованием библиотечных стандартов и протоколов, таких как MARC 21, UNIMARC, z39.50, SRU / SW, SIP2, SIP / NCIP, обеспечивая взаимодействие между Koha и другими системами и технологиями, поддерживая существующие рабочие процессы и инструменты.

Интерфейсы OPAC, circ, управления и самоконтроля Koha основаны на стандартизованных технологиях World Wide Web - XHTML, CSS и Javascript, что делает Koha действительно независимым от платформы решением.

**Сравнение популярных АБИС (Каталог книг)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Автоматизированная библиотечная информационная система (АБИС) | | | | | |
|  | Поиск книг | Учет книг | Интерфейс для библиотекарей | Настройка каталогизации | Импорт/экспорт данных |
| Koha | Х | Х | Х |  |  |
| Колибри | Х | Х | Х | Х | Х |
| Моя библиотека | Х | Х | Х | Х | Х |
| Фолиант | Х | Х | Х | Х |  |
| УФД/Библиотека | Х | Х | Х |  | Х |

**Сравнение популярных АБИС (Читатели)**

|  |  |  |  |
| --- | --- | --- | --- |
| Автоматизированная библиотечная информационная система (АБИС) | | | |
| Интерфейс для читателей | Управление читателями | Списки прочитанного для читателей | Информация о посещении |
| Х | Х | Х |  |
|  | Х | Х | Х |
|  | Х | Х |  |
| Х | Х | Х |  |
| Х | Х | Х |  |

Можно заметить, что некоторая часть функционала является общей абсолютно для каждой ИС:

* Поиск книг;
* Учет книг;
* Интерфейс для библиотекарей;
* Управление читателями;
* Списки прочитанного читателями.

## 1.3 Разработка концепции ИС

В результате проведения анализа бизнес-процессов и аналогов программного обеспечения, предлагаемого рынком, появляется возможность формирования общего плана требований и организация этапов разработки системы.

Программа должна обеспечивать возможность выполнения перечисленных ниже функций.

1. Учет зафиксированной информации о выдаче книг под залог.

1.1. Фиксирование выданной книги.

1.2. Фиксирование данных о читателе.

1.3. Фиксирование дат начала проката и даты возвращения книги.

1.4. Редактирование информации о выданных книгах.

1.5. Просмотр информации о выданных книгах.

1. Учет информации о книгах.

2.1. Ввод информации о книгах.

2.2. Редактирование информации о книгах.

2.3. Просмотр информации о книгах.

1. Учет контактной информации о читателях.

3.1. Ввод информации о читателях.

3.2. Редактирование информации о читателях.

3.3. Просмотр информации о читателях.

В Системе должна быть предусмотрена возможность ввода и редактирования информации о движении книг Библиотеки. Информация о конкретной записи должна состоять из:

1. номера книги;
2. номере читателя;
3. дат выдачи и возвращения книги;
4. наличие штрафа за порчу книги.

В Системе должна быть возможно ввода неограниченного количества записей о движении книг, а также о книгах и читателях.

В Системе должна быть предусмотрена возможность поиска и просмотра информации о выдаче книг. Поиск информации должен выполняться по следующим параметрам:

1. книга;
2. читатель;
3. даты выдачи и возвращения.

В результате поиска пользователю должен быть предоставлен список информационных записей либо, удовлетворяющих условиям поиска. В случае, если не найдена ни одна запись, удовлетворяющая условию, пользователю должно быть выдано соответствующее сообщение.

Аналогичными являются действия и с другими справочниками.

В рамках проекта, была сформирована диаграмма предполагаемого перечня работ, которая выражена в виде диаграммы Гантта.

# 2. Разработка информационной системы

Разработка программного обеспечения подразумевает знание правил и методологий для правильного решения возникающих проблем. Чаще всего, структуру программных продуктов можно разделить на два слоя, проектирование которых требует освоения специализированных технологий.

В нашем случае встает необходимость в проектировании базы данных приложения и его объемлющей составляющей – клиентской части.

В связи с этим, необходимо ознакомиться с теоретической составляющей предстоящей работы.

## 2.1 Проектирование баз данных

База данных – представленная в объективной форме совокупность самостоятельных материалов (статей, расчётов, нормативных актов, судебных решений и иных подобных материалов), систематизированных таким образом, чтобы эти материалы могли быть найдены и обработаны с помощью электронной вычислительной машины [1].

Многие специалисты указывают на распространённую ошибку, состоящую в некорректном использовании термина «база данных» вместо термина «система управления базами данных», и указывают на необходимость различения этих понятий.

Проектирование баз данных – процесс создания схемы базы данных и определения необходимых ограничений целостности.

Включает в себя следующие задачи:

* Обеспечение хранения в БД всей необходимой информации.
* Обеспечение возможности получения данных по всем необходимым запросам.
* Сокращение избыточности и дублирования данных.
* Обеспечение целостности базы данных.

Рассмотрим основные этапы проектирования базы данных:

Концептуальное (инфологическое) проектирование – построение семантической модели предметной области, то есть информационной модели наиболее высокого уровня абстракции. Такая модель создаётся без ориентации на какую–либо конкретную СУБД и модель данных. Термины «семантическая модель», «концептуальная модель» и «инфологическая модель» являются синонимами. Кроме того, в этом контексте равноправно могут использоваться слова «модель базы данных» и «модель предметной области» (например, «концептуальная модель базы данных» и «концептуальная модель предметной области»), поскольку такая модель является как образом реальности, так и образом проектируемой базы данных для этой реальности.

Конкретный вид и содержание концептуальной модели базы данных определяется выбранным для этого формальным аппаратом. Обычно используются графические нотации, подобные ER–диаграммам.

Чаще всего концептуальная модель базы данных включает в себя:

* описание информационных объектов или понятий предметной области и связей между ними.
* описание ограничений целостности, то есть требований к допустимым значениям данных и к связям между ними [3].

Логическое (даталогическое) проектирование – создание схемы базы данных на основе конкретной модели данных, например, реляционной модели данных. Для реляционной модели данных даталогическая модель – набор схем отношений, обычно с указанием первичных ключей, а также «связей» между отношениями, представляющих собой внешние ключи.

Преобразование концептуальной модели в логическую модель, как правило, осуществляется по формальным правилам. Этот этап может быть в значительной степени автоматизирован.

На этапе логического проектирования учитывается специфика конкретной модели данных, но может не учитываться специфика конкретной СУБД.

Физическое проектирование – создание схемы базы данных для конкретной СУБД. Специфика конкретной СУБД может включать в себя ограничения на именование объектов базы данных, ограничения на поддерживаемые типы данных и т. п. Кроме того, специфика конкретной СУБД при физическом проектировании включает выбор решений, связанных с физической средой хранения данных (выбор методов управления дисковой памятью, разделение БД по файлам и устройствам, методов доступа к данным), создание индексов и т. д. [3].

## 2.2 Проектирование приложения

Программное обеспечение – это программа или множество программ, используемых для управления компьютером.

Разработка программного продукта предписывает следование стандартам, выработанным сообществом специалистов отрасли за предшествовавшие годы.

Рассмотренная в этой главе методология разработки зарекомендовала себя как удобная для восприятия и гибкая к модификациям кода. Речь пойдет о проектировании приложения по паттерну MVC.

Model–View–Controller (MVC, «Модель–Представление–Контроллер», «Модель–Вид–Контроллер») – схема разделения данных приложения, пользовательского интерфейса и управляющей логики на три отдельных компонента: модель, представление и контроллер – таким образом, что модификация каждого компонента может осуществляться независимо [4].

* Модель (Model) предоставляет данные и реагирует на команды контроллера, изменяя своё состояние.
* Представление (View) отвечает за отображение данных модели пользователю, реагируя на изменения модели.
* Контроллер (Controller) интерпретирует действия пользователя, оповещая модель о необходимости изменений [4].

Основная цель применения этой концепции состоит в отделении бизнес логики (модели) от её визуализации (представления, вида). За счёт такого разделения повышается возможность повторного использования кода. Наиболее полезно применение данной концепции в тех случаях, когда пользователь должен видеть те же самые данные одновременно в различных контекстах и/или с различных точек зрения. В частности, выполняются следующие задачи:

* К одной модели можно присоединить несколько видов, при этом не затрагивая реализацию модели. Например, некоторые данные могут быть одновременно представлены в виде электронной таблицы, гистограммы и круговой диаграммы;
* Не затрагивая реализацию видов, можно изменить реакции на действия пользователя (нажатие мышью на кнопке, ввод данных) – для этого достаточно использовать другой контроллер;
* Ряд разработчиков специализируется только в одной из областей: либо разрабатывают графический интерфейс, либо разрабатывают бизнес–логику. Поэтому возможно добиться того, что программисты, занимающиеся разработкой бизнес логики (модели), вообще не будут осведомлены о том, какое представление будет использоваться.

Концепция MVC позволяет разделить модель, представление и контроллер на три отдельных компонента.

Модель предоставляет данные и методы работы с ними: запросы в базу данных, проверка на корректность. Модель не зависит от представления (не знает как данные визуализировать) и контроллера (не имеет точек взаимодействия с пользователем) просто предоставляя доступ к данным и управлению ими. Модель строится таким образом, чтобы отвечать на запросы, изменяя своё состояние, при этом может быть встроено уведомление «наблюдателей». Модель, за счёт независимости от визуального представления, может иметь несколько различных представлений для одной «модели».

Представление отвечает за получение необходимых данных из модели и отправляет их пользователю. Представление не обрабатывает введённые данные пользователя. Представление может влиять на состояние модели, сообщая модели об этом.

Контроллер обеспечивает «связи» между пользователем и системой. Контролирует и направляет данные от пользователя к системе и наоборот. Использует модель и представление для реализации необходимого действия.

# 3. Реализация ИС

Данный раздел отчета по учебной практике посвящен анализу требований к приложению, перечислению использованных инструментов и демонстрации готового результата.

## 3.1 Функциональные требования

Основными целями разработки системы могут служить:

* Сокращение времени обработки входной информации и получения результатных показателей;
* Повышение степени достоверности выходной информации;
* Снижение трудоемкости и количества ошибок при решении задачи.

Приложение позволяет отслеживать финансовые показатели библиотеки. В свою очередь, библиотека занимается выдачей напрокат книг, имеющихся в небольшом количестве экземпляров.

В библиотеку обращаются читатели, зарегистрированные в картотеке, содержащей стандартные анкетные данные:

* Фамилия;
* Имя;
* Отчество;
* Адрес;
* Телефон.

Каждый читатель может обращаться в библиотеку несколько раз и все его обращения фиксируются. При этом, по каждому факту выдачи книги запоминаются даты выдачи и ожидаемая дата возврата.

Приведем описание таблиц базы данных:

* Книги (Код книги, Название, Автор, Залоговая стоимость, Стоимость проката, Жанр).
* Читатели (Код читателя, Фамилия, Имя, Отчество, Адрес, Телефон).
* Выданные книги (Код книги, Код читателя, Дата выдачи, Дата возврата).

Приложение предназначено для использования персоналом библиотеки.

В результате анализа постановки задачи было решено спроектировать интерфейс с рядом точек взаимодействия с приложением.

Рассмотрим описание основного окна приложения:

* Основное окно программы содержит таблицу, в которой отображаются записи по выдачам книг;
* Основное окно содержит кнопки:
  + - Картотека «читатели»;
    - Картотека «книги»;
    - Финансовые результаты;
    - Оформить выдачу книги.
* Имеются кнопки по настройке выборок, отображаемых в таблице:
  + - Показать все записи журнала выданных книг;
    - Показать завершенные;
    - Показать незавершенные;

Теперь рассмотрим описание окон картотек:

* Содержится таблица, отображающая все записи соответствующей картотеки;
* Имеется кнопка добавления новой записи;
* Имеется поле поиска по картотеке.

Окно финансовых результатов:

* Имеется поле задания начальной даты расчета;
* Имеется поле задания конечной даты расчета;
* Имеется кнопка начала расчета;
* Результат отображается в виде таблицы со всеми записями из журнала, подходящими по заданным датам;
* Сумма с итогом расчета отображается рядом с кнопкой запуска вычисления.

Добавить новую запись в журнал выданных книг можно вызовом соответствующей формы:

* В первом выпадающем списке выбирается читатель;
* Во втором – выбирается книга;
* В поле задания даты предполагаемого возврата выбирается число;
* Сохранение или отмена записи осуществляется нажатием соответствующих кнопок.

Добавление новой записи в картотеку осуществляется через соответствующую ей форму:

* Данные по новой записи вводятся в соответствующие поля;
* Сохранение или отмена записи осуществляется нажатием соответствующих кнопок.

## 3.2 Выбор инструментов для разработки

Разработка приложения велась на собственном персональном компьютере, на базе процессоров Intel, под управлением операционной системы Windows 10.

Из программных ресурсов, были использованы:

* Среда разработки IntelliJ IDEA;
* Средство работы с СУБД MS SQL Server – SQL Server Management Studio;

Использовались технологии:

* Инструменты разработки приложений на платформе Java – Java 8 SE;
* СУБД MS SQL Server 2014;
* Система сборки – Maven;
* Фреймворк Spring, с подключением модуля Spring JDBC;
* Для разработки интерфейса решено использовать библиотеку JavaFX.

Итоговый результат должен представлять собой настольное приложение, с подключением базы данных.

## 3.3 Проектирование Базы Данных

Анализ функциональных требований выделил 3 сущности, которые будут храниться в базе данных:

* Читатель (reader);
* Книга (book);
* Запись (record).

Связь между таблицами "Читатель" и "Книга" осуществляется посредством таблицы "Запись".

Владея данной информацией, можно создать концептуальную модель данных (рис. 3).

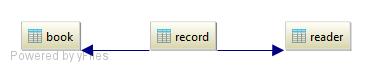


Рис. 3

Из анализа функциональных требований, также, вытекает следующее:

Таблица "Читатель" содержит поля:

* Фамилия;
* Имя;
* Отчество
* Адрес;
* Номер телефона.

Таблица "Книга" содержит поля с названием, автором, залоговой стоимости, стоимостью проката и жанром.

Таблица "Запись" является смежной таблицей, а значит, хранит в себе ключи пациент–заболевание и даты выдачи-возврата.

Следовательно, можно составить логическую схему для реляционной модели данных (рис. 4).

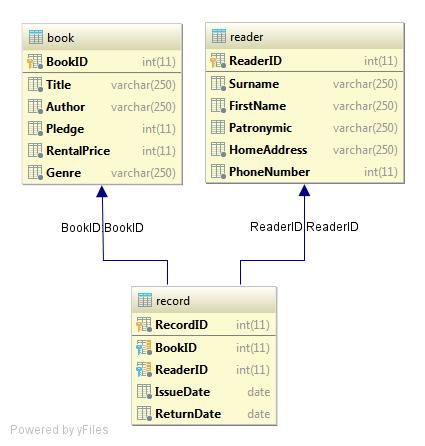


Рис. 4

Следующим этапом является разработка клиентской части приложения и интерфейса системы.

## 3.4 Разработка клиентского приложения

Клиентская часть приложения включает в себя структуру классов, соответствующих сущностям, хранимым в базе данных, а именно:

* Читатель;
* Книга;
* Запись.

Связь клиентской части и базы данных осуществляется инструментарием модуля Spring JDBC через соответствующие классы доступа к БД.

В соответствии с требованиями к интерфейсу приложения, на главном окне отображается таблица с записями журнала и кнопок, открывающих соответствующие формы (рис. 5).

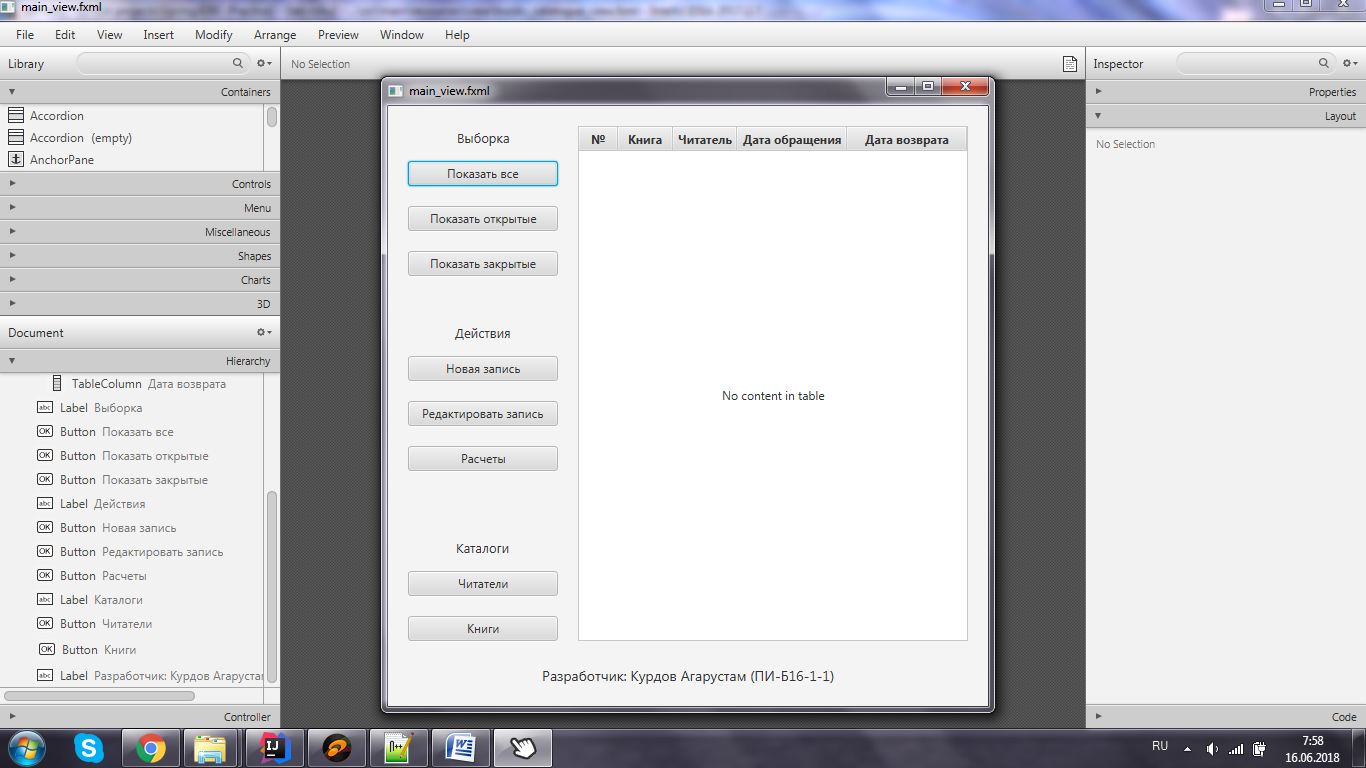


Рис. 5

Вызов нужной нам картотеки, отображает соответствующую форму.

В открывшемся окне есть возможность добавления новых записей и поиска по имеющимся.

Возможно неограниченное количество записей. При этом, удалять существующие записи запрещено (рис. 6).

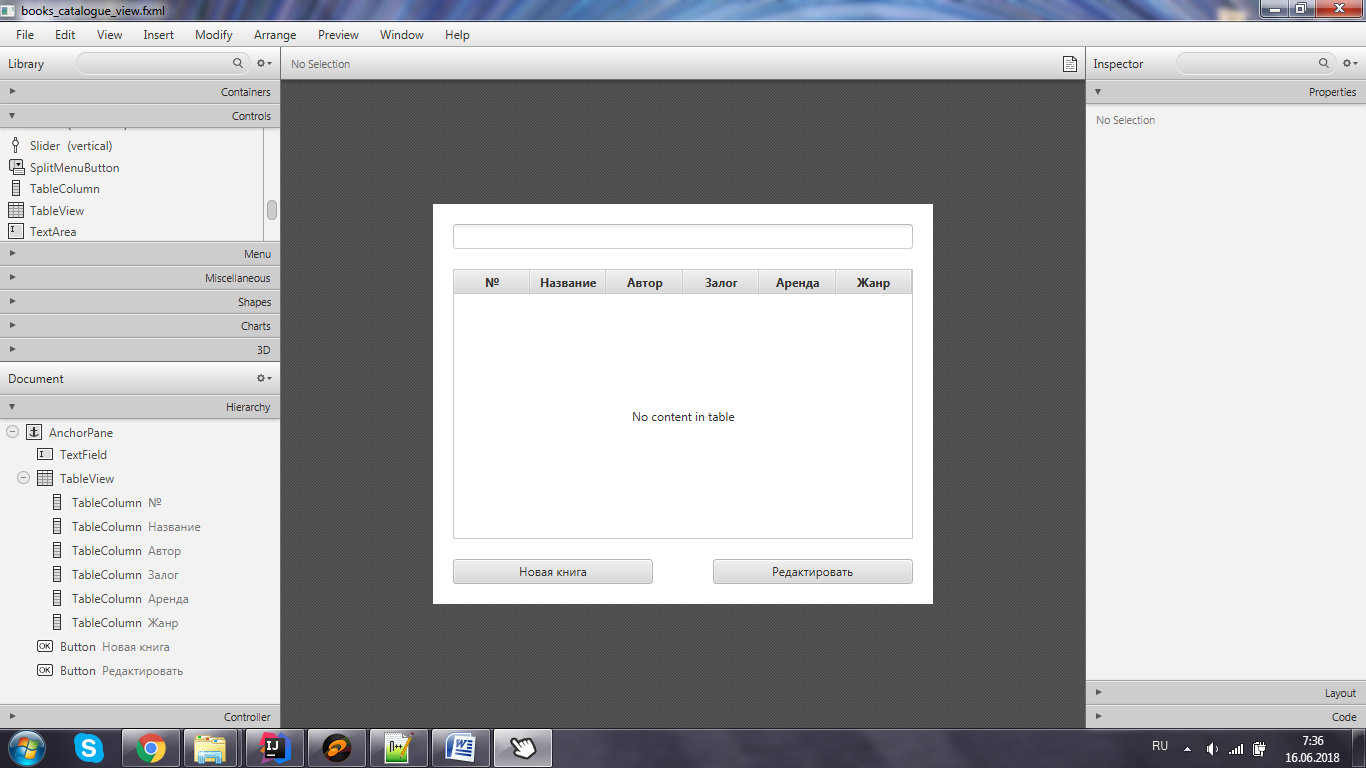


Рис. 6

Нажатие на кнопку редактирования и добавления новой записи в картотеку вызывает форму для заполнения и сохранения данных (рис. 7).

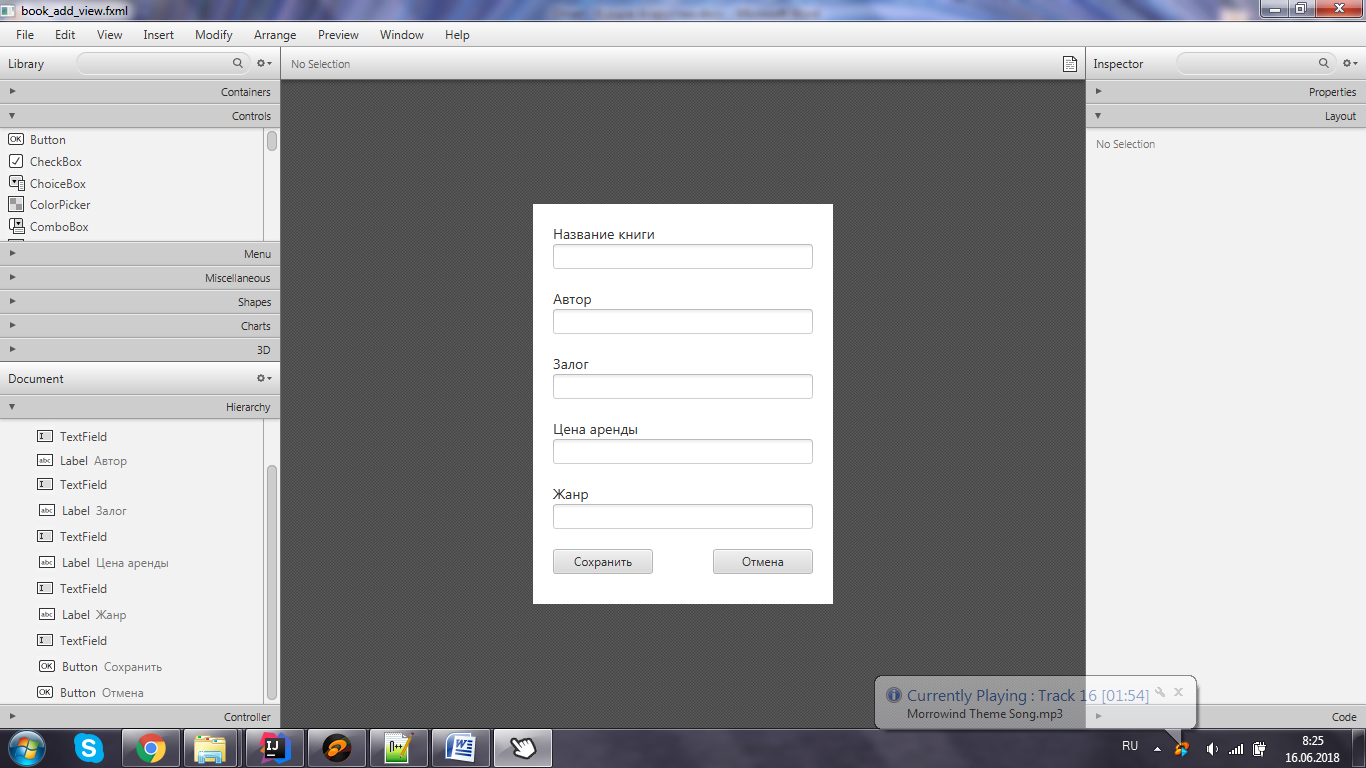


Рис. 7

Окно финансовых результатов позволяет задать начальную и конечную даты и получить подробную информацию (рис. 8).

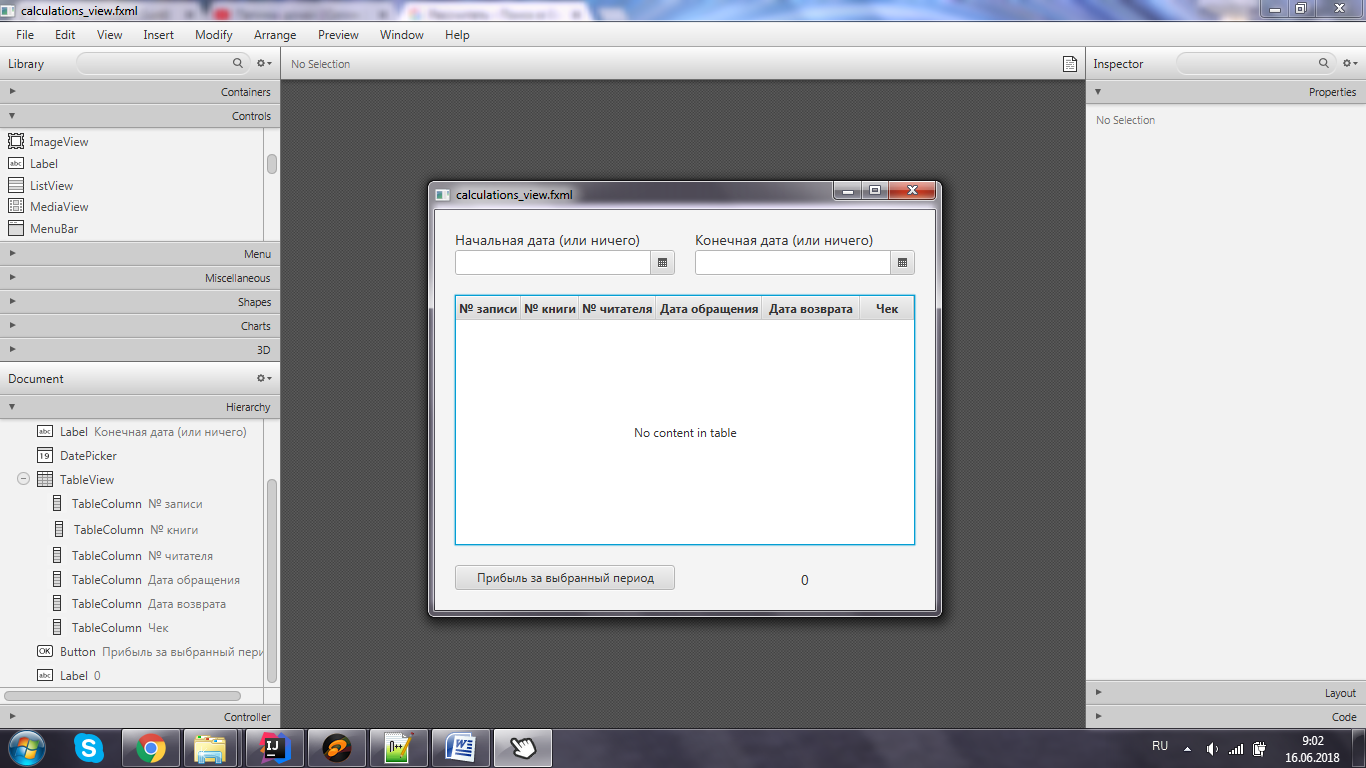


Рис. 8

Создание новой записи в журнале выдачи книг позволяет выбрать читателя, книгу и дату возврата (рис. 9).

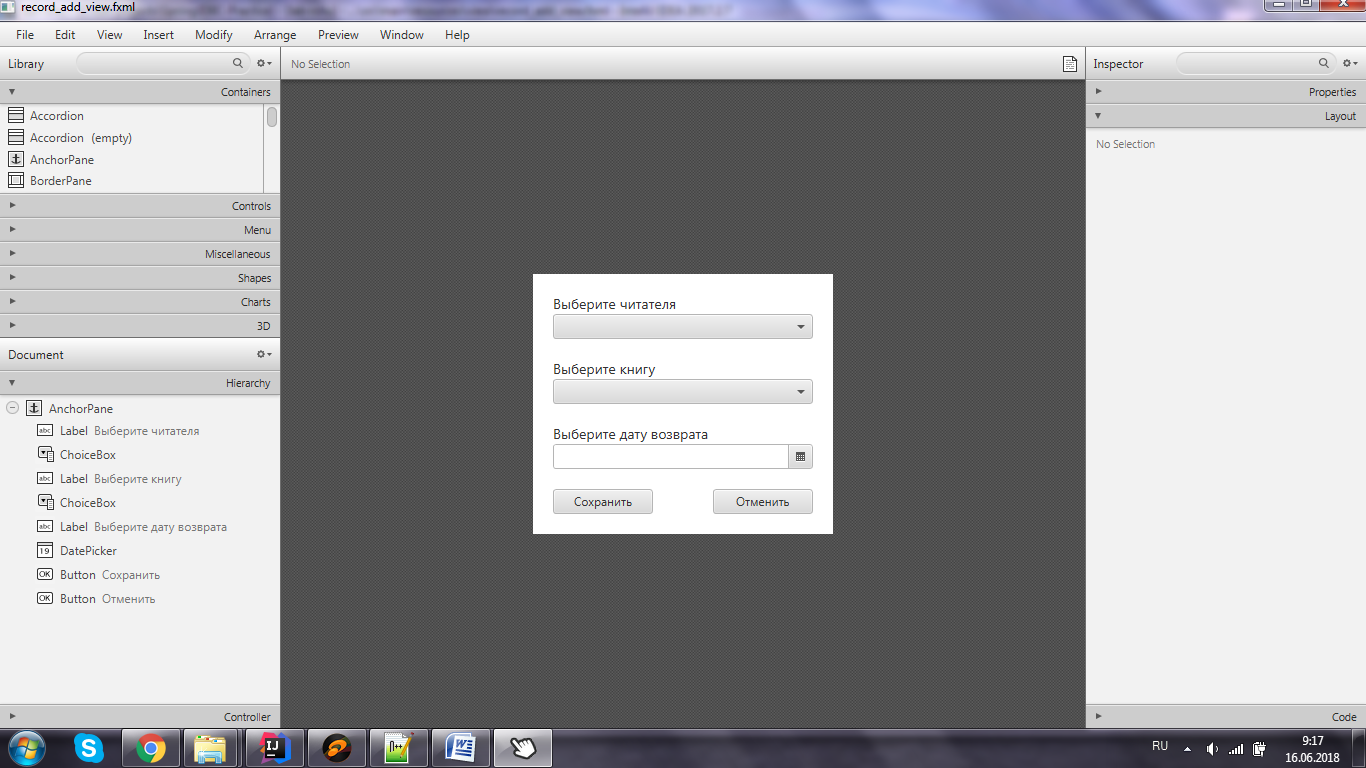


Рис. 9

# Заключение

В результате проведенного исследования были закреплены изученные методологии проектирования программных продуктов.

На практике были закреплены теоретические сведения о шагах проектирования баз данных и клиентского приложения, изучение жизненного цикла разработки проекта.

Для решения поставленных задач, были использованы технологии и фреймворки языка Java, а также СУБД MS SQL Server.

В результате проделанной работы был разработан полноценный программный продукт, полностью отвечающий заявленным требованиям.

Тем не менее, возможно расширение функционала готового приложения, путем добавления редактирования записей картотек особыми категориями пользователей, обладающими соответствующими разрешениями. Добавление данного функционала базируется на существовании риска сохранения неправильных данных первичным оператором. В свою очередь, наличие ограничений на редактирование сохраненных данных связано с контролем над достоверностью информации системы.

Из вышеперечисленного также вытекает потребность в добавлении сущности «Пользователь» и в реализации их регистрации, с заданием административных или пользовательских прав.

Помимо прочего, любое программное обеспечение должно иметь возможности для расширения на другие рынки, что невозможно без добавления соответствующих локализаций. Этот критерий особенно важен в условиях глобализации.

# Список источников

1. Гражданский кодекс РФ, ст. 1260
2. Истомин Е.П., Новиков В.В., Новикова М.В. Высокоуровневые методы информатики и программирования: Учебник.– СПб.: «Андреевский издательский дом», 2006.– 228 с.
3. Гарсиа–Молина Г., Ульман Дж., Уидом Дж. Системы баз данных. Полный курс. – М.: «Вильямс», 2013. – 1088 с.
4. Рогачев С. Обобщённый Model–View–Controller – М.: RSDN, 2007. – 23 с.
5. Кей Хорстманн, Гари Корнелл. Java – Библиотека профессионала. – М.: МАХАОН, 2016. – 866с.
6. Блох Д.Эффективное программирование. – СПб.:Питер, 2014. – 461с.
7. Оти М., Молнар Ш. Будущее SQL Server: прогноз Теда Каммерта // Windows IT Pro/RE: журнал. – Москва: «Открытые системы», 2011. – № 01
8. MySQL [Электронный ресурс] The REGEXP and RLIKE operators: сайт. URL: http://dev.mysql.com/doc/refman/5.7/en/regexp.html (Дата обращения 01.06.2018)
9. JetBrains [Электронный ресурс] Documentation – IntelliJ IDEA: сайт. URL: <https://www.jetbrains.com/idea/documentation/> (Дата обращения 01.06.2018).
10. Spring [Электронный ресурс] // Spring Official Documentation: сайт. URL: [https://docs.spring.io/spring/docs/current/spring–framework–reference/html/index.html](https://docs.spring.io/spring/docs/current/spring-framework-reference/html/index.html) (Дата обращения 01.06.2018).
11. Oracle [Электронный ресурс] JavaFX Official Documentation: сайт. URL: <http://www.oracle.com/technetwork/java/javafx/overview/index.html> (Дата обращения 01.06.2018).
12. http://consortium.ruslan.ru/colibry/colibry/colibry.htm
13. https://koha-community.org/about/
14. http://www.keldysh.ru/papers/2011/source/prep2011\_02.pdf
15. https://knowledge.ru/prog/2c0b65625a3bd69b4d53b88521206c26\_0.html

# Приложение A

**Скрипт создания Базы Данных**

**CREATE TABLE** labrshu.dbo.Book  
(  
 **BookID INT PRIMARY KEY NOT NULL IDENTITY**,  
 **Title VARCHAR**(250) **NOT NULL**,  
 **Author VARCHAR**(250) **NOT NULL**,  
 **Pledge INT NOT NULL**,  
 **RentalPrice INT NOT NULL**,  
 **Genre VARCHAR**(250) **NOT NULL**)  
  
**CREATE TABLE** labrshu.dbo.Reader  
(  
 **ReaderID INT PRIMARY KEY NOT NULL IDENTITY**,  
 **Surname VARCHAR**(250) **NOT NULL**,  
 **FirstName VARCHAR**(250) **NOT NULL**,  
 **Patronymic VARCHAR**(250),  
 **HomeAddress VARCHAR**(250) **NOT NULL**,  
 **PhoneNumber INT NOT NULL**)  
  
**CREATE TABLE** labrshu.dbo.Record  
(  
 **RecordID INT PRIMARY KEY NOT NULL IDENTITY**,  
 **BookID INT NOT NULL**,  
 **ReaderID INT NOT NULL**,  
 **IssueDate DATE NOT NULL**,  
 **ReturnDate DATE NOT NULL  
  
 CONSTRAINT** Record\_Book\_BookID\_fk **FOREIGN KEY** (**BookID**) **REFERENCES** Book (**BookID**),  
 **CONSTRAINT** Record\_Reader\_ReaderID\_fk **FOREIGN KEY** (**ReaderID**) **REFERENCES** Reader (**ReaderID**)  
)

# Приложение B

**Фрагмент кода**

**package** com.rshu.lab.dao;  
  
**import** com.rshu.lab.entity.RecordData;  
**import** org.springframework.jdbc.core.JdbcTemplate;  
  
**import** java.sql.ResultSet;  
**import** java.sql.SQLException;  
**import** java.util.Date;  
**import** java.util.List;  
  
**public class** RecordDataDao **extends** Dao<RecordData> {  
  
 **public** RecordDataDao(JdbcTemplate jdbcTemplate) {  
 **super**(jdbcTemplate);  
 }  
  
 **public** List<RecordData> getBeforeDate(Date till) {  
 **return jdbcTemplate**.query(  
 **"SELECT** *\** **FROM record\_meta\_data WHERE ReturnDate<=?"**,  
 **new** Object[]{till},  
 ((resultSet, i) -> getRecordData(resultSet)));  
 }  
  
 **public** List<RecordData> getAfterDate(Date from) {  
 **return jdbcTemplate**.query(  
 **"SELECT** *\** **FROM record\_meta\_data WHERE IssueDate>=?"**,  
 **new** Object[]{from},  
 ((resultSet, i) -> getRecordData(resultSet)));  
 }  
  
 **public** List<RecordData> getByDates(Date from, Date till) {  
 **return jdbcTemplate**.query(  
 **"SELECT** *\** **FROM record\_meta\_data WHERE IssueDate>=? AND ReturnDate<=?"**,  
 **new** Object[]{from, till},  
 ((resultSet, i) -> getRecordData(resultSet)));  
 }  
  
 **private** RecordData getRecordData(ResultSet resultSet) **throws** SQLException {  
 **return new** RecordData(  
 resultSet.getInt(**"recordID"**),  
 resultSet.getInt(**"BookID"**),  
 resultSet.getString(**"Title"**),  
 resultSet.getInt(**"ReaderID"**),  
 resultSet.getString(**"Surname"**),  
 resultSet.getInt(**"RentalPrice"**),  
 resultSet.getDate(**"IssueDate"**),  
 resultSet.getDate(**"ReturnDate"**),  
 resultSet.getInt(**"Total\_days"**),  
 resultSet.getInt(**"Total\_rent"**)  
 );  
 }  
}