# СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 2](#_Toc421336046)

[1 ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ 4](#_Toc421336047)

[1.1 Описание предметной области системы учета материалов на предприятии 4](#_Toc421336048)

[2 ПОСТАНОВКА ЗАДАЧИ И ОБЗОР МЕТОДОВ ЕЕ РЕШЕНИЯ 6](#_Toc421336049)

[2.1 Задачи, решаемые системой учета материалов на предприятии 6](#_Toc421336050)

[2.2 Требования к системе 8](#_Toc421336051)

[2.3 Спецификация вариантов использования системы учета материалов на предприятии 8](#_Toc421336052)

[3 МОДЕЛИ ПРЕДСТАВЛЕНИЯ СИСТЕМЫ И ИХ ОПИСАНИЕ 11](#_Toc421336053)

[3.1 Модели представления системы 11](#_Toc421336054)

[4 ИНФОРМАЦИОННАЯ МОДЕЛЬ СИСТЕМЫ И ЕЕ ОПИСАНИЕ 22](#_Toc421336055)

[5 ОПИСАНИЕ АЛГОРИТМОВ РЕАЛИЗУЮЩИХ БИЗНЕС-ЛОГИКУ СИСТЕМЫ УЧЕТА МАТЕРИАЛОВ НА ПРЕДПРИЯТИИ 24](#_Toc421336056)

[6 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 25](#_Toc421336057)

[ЗАКЛЮЧЕНИЕ 34](#_Toc421336058)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 35](#_Toc421336059)

[ПРИЛОЖЕНИЕ А 36](#_Toc421336060)

[ПРИЛОЖЕНИЕ Б 40](#_Toc421336061)

[ПРИЛОЖЕНИЕ В 43](#_Toc421336062)

[ПРИЛОЖЕНИЕ Г 79](#_Toc421336063)

# ВВЕДЕНИЕ

Согласно Карлу Марксу, труд является естественной жизнедеятельностью человека. Именно поэтому, а также согласно другим причинам и потребностям человека, производство благ является таким важным и незаменимым явлением в любом обществе.

Производство – это  процесс активного преобразования людьми [природных ресурсов](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B8%D1%80%D0%BE%D0%B4%D0%BD%D1%8B%D0%B5_%D1%80%D0%B5%D1%81%D1%83%D1%80%D1%81%D1%8B) в какой-либо продукт. В современном мире под производством чаще всего понимают промышленное производство. Для промышленного производства характерны такие черты, как узкая специализация, автоматизация, масштабность, непрерывность. Для достижения максимальной эффективности в промышленном производстве необходима строгая система администрирования и учета, затрагивающая все аспекты производства, в том числе организация доставки и распределения на предприятии материальных ресурсов.

Учет распределения материальных ресурсов играет ключевую роль в организации работы предприятия. Без нормально функционирующей системы учета необходимые для производства ресурсы будут доставлены не вовремя, не в необходимом количестве, либо не рационально распределены на предприятии. Учет материалов должен включать в себя непосредственную работу с поставщиками материальных ресурсов, процесс оформления заказов на необходимые ресурсы и администрирование мест хранения доставленных ресурсов.

В Республике Беларусь проблема учета и администрирования производства является одной из ключевых в производственном секторе. Согласно статистике, Беларусь является государством с индустриальным типом экономики. Это означает, что в Республике Беларусь больше половины ВВП приходится на промышленную сферу производства. Именно поэтому для нашей страны так актуальна проблема учета производственного процесса.

В современном мире технологии позволяют автоматизировать и запрограммировать сам процесс учета, что позволяет отказаться от устаревших форм администрирования, бумажной работы, документации, сложных и утомительных расчетов, однообразной работы. Также автоматизация учета предполагает сокращение штата бухгалтеров, повышение точности расчетов и многократное уменьшение вероятности ошибки расчетов.

Целью данного курсового проекта является повышение эффективности и качества управления промышленным предприятием путем автоматизации учета материальных ресурсов.

Для достижения данной цели курсового проекта необходимо выполнить следующие задачи:

* Изучить особенности регистрации и учета материалов на промышленном предприятии;
* Изучить классификацию материалов в связи с проблематикой их учета;
* Изучить специфику процесса получения материальных ресурсов на предприятии;
* Ознакомиться с основными статистическими данными, собираемыми в процессе учета материальных ресурсов на производстве;
* Разработать систему учета материалов на предприятии.

Результатом выполнения выше перечисленных задач станет автоматизация процесса учета материальных ресурсов на промышленном предприятии.

# 1 ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

# 1.1 [Описание предметной области системы](#_Toc279251791) учета материалов на предприятии

Предприятие — обособленная специализированная единица, основным признаком которой является профессионально организованный трудовой коллектив, способный с помощью имеющихся в его распоряжении средств производства изготовить нужные потребителю товары (выполнить работы, оказать услуги) соответствующего значения, профиля и ассортимента. Производственное предприятие также называют [производством](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B8%D0%B7%D0%B2%D0%BE%D0%B4%D1%81%D1%82%D0%B2%D0%BE).

В данной курсовой работе рассматривается проблематика учета материальных ресурсов на промышленном предприятии. Прежде всего, промышленному предприятию необходимы ресурсы для производства продукции. Как правило, заниматься одновременно разработкой материальных ресурсов и изготовлением из них готовой продукции нерентабельно, поэтому большинство промышленных предприятий приобретают материалы у сторонних фирм-поставщиков.

В большинстве своем фирма-поставщик также является узкоспециализированной, что означает, что она занимается доставкой определенных видов материальных ресурсов. С другой стороны, для производства обычно требуется несколько различных ресурсов. Это объясняет тот факт, что производственному предприятию приходится заключать договоры на поставку материалов с несколькими фирмами-поставщиками.

Важнейшую роль в учете материальных ресурсов играет учет их стоимости. В стоимость материальных ресурсов включается стоимость покупки ресурсов у поставщика и стоимость хранения материалов на складе. В то же время в стоимость покупки ресурсов включается стоимость доставки ресурсов на склад и собственная стоимость материальных ресурсов.

Сам учет представляет собой регистрацию поставщиков (фиксирование названия организации поставщика, данных о поставляемом материале, и стоимости поставки), после регистрацию заказа на требуемый материальный ресурс (фиксирование даты подписания и даты выполнения условий договора, а также объем и стоимость заказанных материалов). В дальнейшем, по истечению срока выполнения, необходимо принять полученные материалы на складе и учесть стоимость их поступления и хранения.

# 2 ПОСТАНОВКА ЗАДАЧИ И ОБЗОР МЕТОДОВ ЕЕ РЕШЕНИЯ

# 2.1 Задачи, решаемые системой учета материалов на предприятии

Цель контекстной диаграммы IDEF0 – рассмотреть процесс учета материалов на производстве. Точка зрения – экономист-аналитик, отвечающий за учет материальных ресурсов.

Описание бизнес-процесса учета материалов на предприятии нулевого уровня представлено на рисунке 1.1.

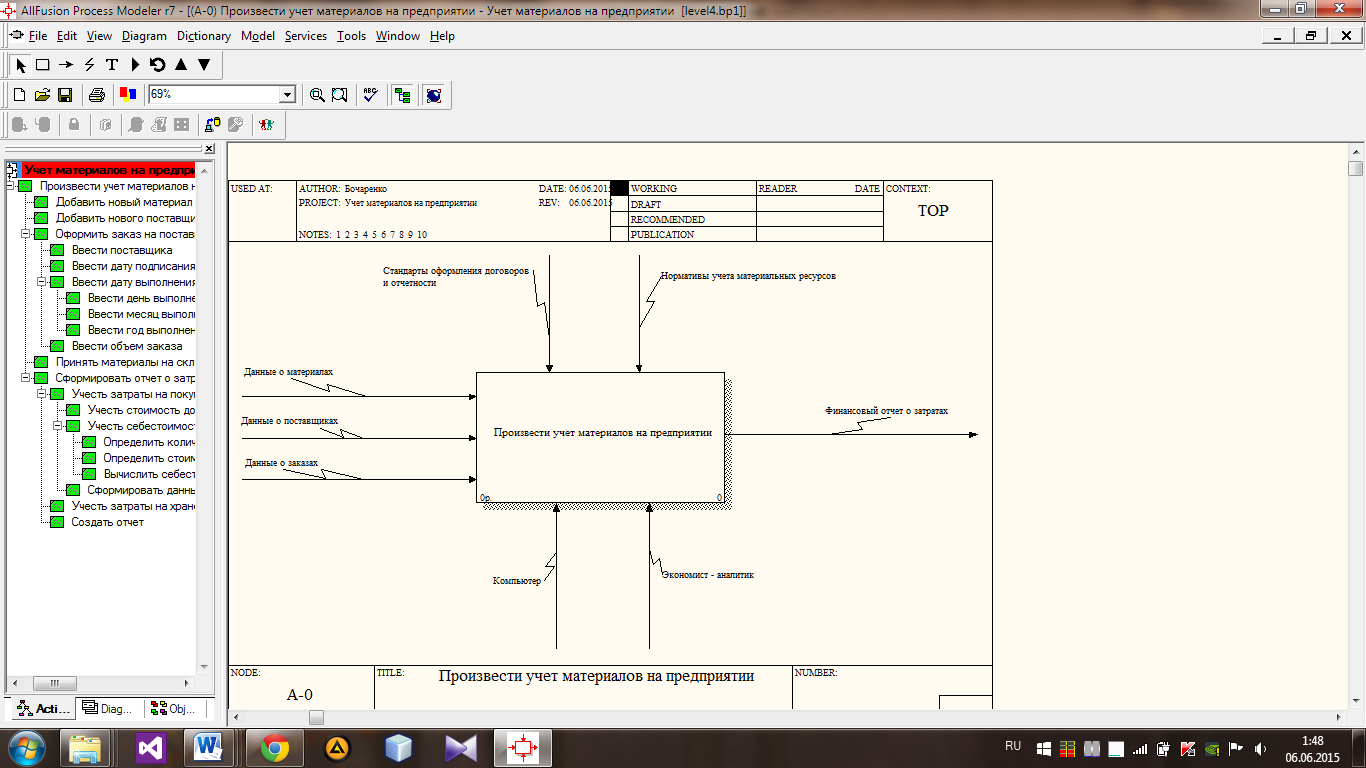


Рисунок 1.1 – Контекстный уровень диаграммы IDEF0

Описание потоков материальных (информационных) объектов:

- Входные данные – данные о материалах, данные о поставщиках, данные о заказах.

- Выходные данные (результат) – финансовый отчет о затратах на покупку и содержание материалов.

- Механизмы – администратор, компьютер и клиент.

- Управление – нормативно-справочная документация и правила оформления регистрационных карточек.



Рисунок 1.2 – Декомпозиция контекстного уровня IDEF0

Рассмотрим декомпозицию основного процесса «Произвести учет материалов на предприятии». Учет материалов на предприятии включает в себя учет собственно материалов, фирм-поставщиков, заказов на поставку и учет материалов на складе. Таким образом, сначала необходимо внести информацию о материалах в базу данных. Затем внести информацию о поставщиках материалов в базу данных. Затем оформить заказ на поставку определенных материалов. Далее необходимо дождаться сроков выполнения заказа, после чего материалы будут получены и оприходованы на складе. После этого формируется отчет о затратах предприятия на закупку и хранение материалов.

Описание других бизнес-процессов учета материалов на предприятии, всех уровней изображено в приложение А.

## 2.2 Требования к системе

В связи с поставленными задачами, необходимо придерживаться следующих требований к системе:

1) Приложение должно осуществлять свою работу в рамках архитектуры клиент-сервер. В качестве языка для реализации выбран объектно-ориентированный язык Java. В данной работе будут реализованы: собственная иерархия классов, расширение базовых классов, предоставляемых SDK, реализация паттерна проектирования Command[1] и Singleton[2], сокрытие данных (инкапсуляция), перегрузка методов, переопределение методов, параметризированные классы, интерфейсы и абстрактные классы, статические методы, обработка исключительных ситуаций[5].

2) Бизнес-логика системы реализована на сервере данного приложения.

3) СУБД – MySQL 5.6. Для разработки информационной модели системы должен быть использован пакет моделирования ERWin. Доступ к данным в СУБД осуществляется через драйвер, предоставляемый производителем СУБД.

3) База данных приведена к 3-ей нормальной форме[3].

4) База данных должна генерироваться sql-запросом, выполняемым приложением.

5) Функционал серверной части не менее 6 Use Case’ов (вариантов использования).

6) Паттерн реализации: «MVC»[4]

7) Транспортный уровень. Взаимодействие между серверной и клиентскими частями должно осуществляется с использованием протокола HTTP.

# 2.3 Спецификация вариантов использования системы учета материалов на предприятии

Суть диаграммы вариантов использования состоит в следующем: проектируемая система представляется в виде множества сущностей или актеров, взаимодействующих с системой с помощью так называемых вариантов использования. При этом актером (actor) или действующим лицом называется любая сущность, взаимодействующая с системой извне. В свою очередь, вариант использования (use case) служит для описания сервисов, которые система предоставляет актеру. Другими словами, каждый вариант использования определяет некоторый набор действий, совершаемый системой при диалоге с актером.

Диаграмма вариантов использования подсистемы учета материалов на предприятии представлена на рисунке 2.3.1



Рисунок 2.3.1 – Диаграмма вариантов использования

В данной программе учета материалов на предприятии есть один актер: экономист – аналитик (администратор).

Рассмотрим функции, которые можно будет осуществлять на правах администратора. Администратор может производить следующие действия:

* Учет материалов. Учет материалов включает в себя добавление, обновление, удаление и чтение материалов из базы данных;
* Учет поставщиков. Это действие включает в себя добавление, обновление, удаление и чтение поставщиков из базы данных;
* Управление заказами. Это действие включает в себя добавление, обновление, удаление и чтение заказов из базы данных;
* Управление складом. Это действие включает в себя просмотр склада и разгрузку склада;
* Изменение текущей даты. Это действие реализует изменение запасов материалов на складе в случае изменения завершенности заказов;
* Поиск материалов, поставщиков, заказов и складских запасов по базе данных;
* Сортировку материалов, поставщиков, заказов и складских запасов в базе данных.

# 3 МОДЕЛИ ПРЕДСТАВЛЕНИЯ СИСТЕМЫ И ИХ ОПИСАНИЕ

# 3.1 Модели представления системы

**3.1.1 Диаграмма классов системы учета материалов на предприятии**

Диаграмма классов:

* Является центральным звеном объектно-ориентированного подхода;
* Содержит информацию об объектах системы и статических связях межд1у объектами;
* Отражает декларативные знания о предметной области;
* Оперирует понятиями класса, объекта, отношения, пакета.

Класс – это множество объектов, которые обладают одинаковой структурой, поведением и отношениями с объектами из других классов [7].

Диаграмма классов сущностей базы данных представлена на рисунке 3.1.1.



Рисунок 3.1.1 – Диаграмма классов – сущностей базы данных

Диаграмма классов слоя доступа к базе данных представлена на рисунке 3.1.2.



Рисунок 3.1.2 – Диаграмма классов слоя доступа к БД

Диаграмма классов слоя Command представлена на рисунке 3.1.3.



Рисунок 3.1.3 – Диаграмма классов слоя Command

Диаграмма классов пакета by.bsuir.bocharenko.rms.servlet представлена на рисунке 3.1.4.



Рисунок 3.1.4 – Диаграмма классов пакета by.bsuir.bocharenko.rms.servlet

Диаграмма классов пакета by.bsuir.bocharenko.rms. controller.exceptions представлена на рисунке 3.1.5.



Рисунок 3.1.5 – Диаграмма классов пакета by.bsuir.bocharenko.rms. controller.exceptions

Диаграмма классов пакета by.bsuir.bocharenko.rms. entity.sortmaterial представлена на рисунке 3.1.6.



Рисунок 3.1.6 – Диаграмма классов пакета by.bsuir.bocharenko.rms. entity.sortmaterial

Диаграмма классов пакета by.bsuir.bocharenko.rms. entity.sortsupplier представлена на рисунке 3.1.7.



Рисунок 3.1.7 – Диаграмма классов пакета by.bsuir.bocharenko.rms. entity.sortsupplier

Диаграмма классов пакета by.bsuir.bocharenko.rms. entity.sortorder представлена на рисунке 3.1.8.



Рисунок 3.1.8 – Диаграмма классов пакета by.bsuir.bocharenko.rms. entity.sortorder

Диаграмма классов пакета by.bsuir.bocharenko.rms. entity.sortstorage представлена на рисунке 3.1.9.



Рисунок 3.1.9 – Диаграмма классов пакета by.bsuir.bocharenko.rms. entity.sortstorage

**3.1.2 Диаграмма последовательности**

Диаграмма последовательности используется для того, чтобы показать, как несколько объектов поддерживают выполнение функций прецедента. Диаграммы последовательности хорошо подходят для отображения того, как объекты взаимодействуют в прецеденте, но они малоэффективны для описания специфических алгоритмов системы.

Диаграмма последовательности добавления материала представлена на рисунке 3.1.2



Рисунок 3.2.1 – Диаграмма последовательности добавления материала в БД

**3.1.3 Диаграммы состояния основных объектов системы учета материалов на предприятии**

Диаграммы состояний используются для описания поведения, реализуемого в рамках варианта использования, или поведения экземпляров класса (объектов), компонента, узла или системы в целом. Поведение моделируется через автомат (англ. state machine), описывающий возможные последовательности состояний экземпляра сущности и переходы между ними на протяжении его жизненного цикла, начиная от создания и заканчивая уничтожением [8].

Диаграмма состояния для объекта «заказ» представлена на рисунке 3.3.1



Рисунок 3.3.1 – Диаграмма состояния для объекта «Заказ»

**3.1.4 Диаграмма компонентов подсистемы учета постояльцев в гостинице**

Диаграмма компонентов – статическая диаграмма, показывающая разбиение программной системы на компоненты, а также связи между ними. Чаще всего в качестве компонентов выступают различные библиотеки, исполняемые файлы, пакеты, интерфейсы, протоколы взаимодействия и т.д.

Диаграмма компонентов представлена на рисунке 3.4.1



Рисунок 3.4.1 – Диаграмма компонентов

**3.1.5 Диаграмма развертывания системы**

Диаграмма развертывания показывает, как будет выглядеть система, когда она будет введена в эксплуатацию. Данный вид диаграмм показывает логические элементы системы, их физическое размещение, то, как эти элементы взаимодействуют, а также их количество.

Диаграмма развертывания показывает, как приведенные ниже элементы системы связаны друг с другом, какие протоколы они используют, какие операционные системы и физические устройства, включая компьютеры и другие вычислительные средства, имеются в информационной системе и в каком порядке необходимо проводить развертывание системы.

Диаграмма развертывания представлена на рисунке 3.5.1



Рисунок 3.5.1 – Диаграмма развертывания

# 4 ИНФОРМАЦИОННАЯ МОДЕЛЬ СИСТЕМЫ И ЕЕ ОПИСАНИЕ

Информационной моделью в данном курсовом проекте является Erwin.

AllFusion ERwin Data Modeler (ранее ERwin) — CASE-средство для проектирования и документирования баз данных, которое позволяет создавать, документировать и сопровождать базы данных, хранилища и витрины данных. Модели данных помогают визуализировать структуру данных, обеспечивая эффективный процесс организации, управления и администрирования таких аспектов деятельности предприятия, как уровень сложности данных, технологий баз данных и среды развертывания[1].

Логическая и физическая модели базы данных «Система управления ресурсами» представлены на рисунке 4.1 и рисунке 4.2 соответственно.

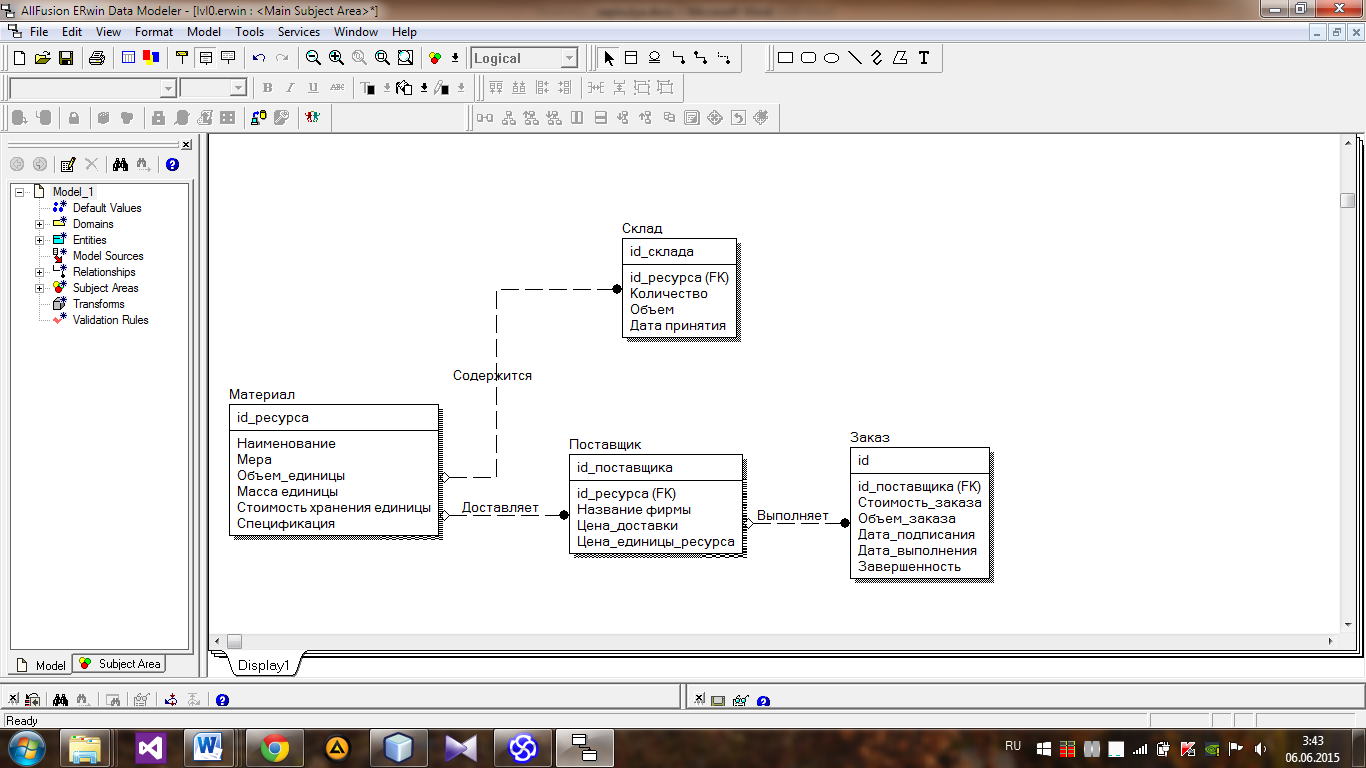


Рисунок 4.1 – Логическая модель данных

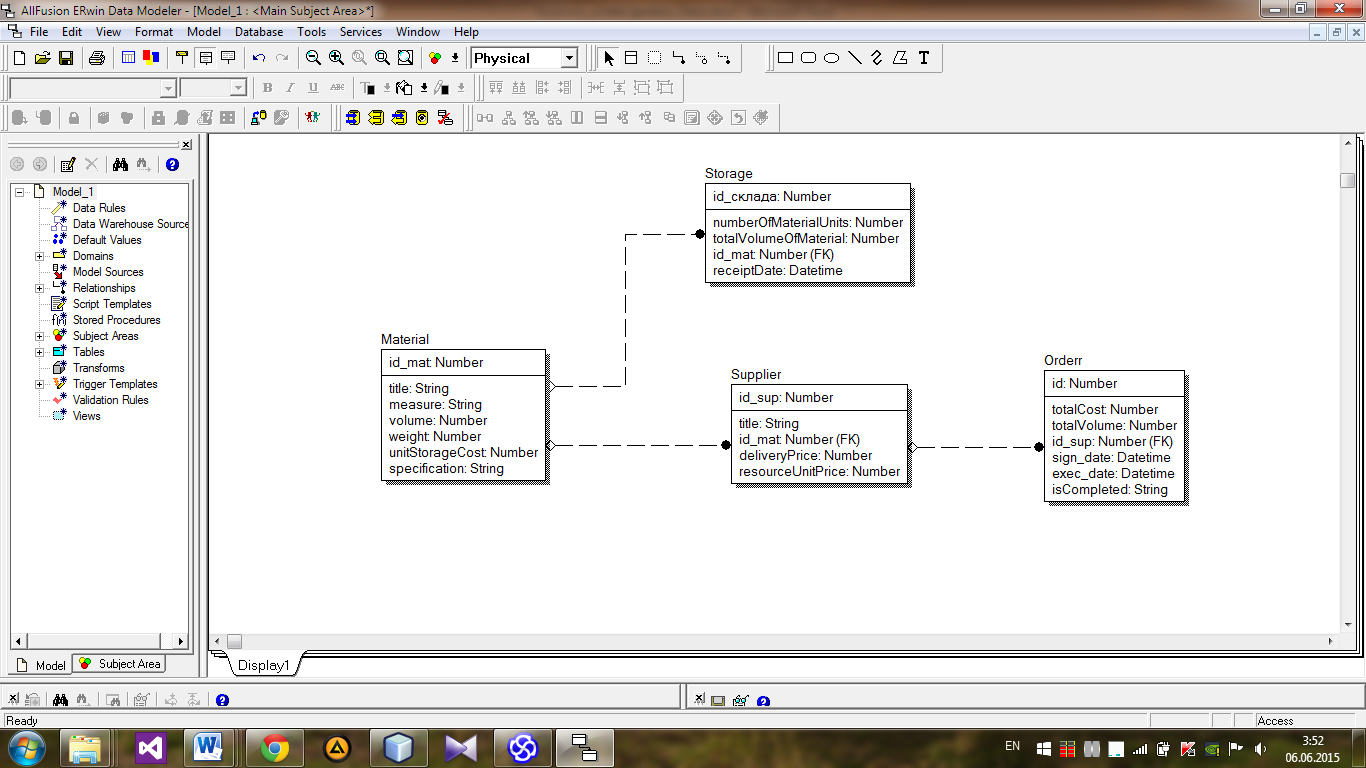


Рисунок 4.2 – Физическая модель данных

Таблица «Материал» связана с таблицей «Поставщик» связью один ко многим, т.к. один материал могут поставлять несколько поставщиков.

Таблица «Поставщик» связана с таблицей «Заказ» связью один ко многим, т.к. один поставщик может принимать несколько заказов.

Таблица «Материал» связана с таблицей «Склад» связью один ко многим, т.к. один материал может занимать несколько ячеек на складе, если он поступает в различное время.

# 5 ОПИСАНИЕ АЛГОРИТМОВ РЕАЛИЗУЮЩИХ БИЗНЕС-ЛОГИКУ СИСТЕМЫ УЧЕТА МАТЕРИАЛОВ НА ПРЕДПРИЯТИИ

Данное web-приложение выполнено в архитектуре паттерна MVC. Это означает, что в приложение разделено на 3 слоя: слой представления, слой управления и слой доступа к базе данных.

Слой представления представлен JSP – страницами. Они используются для взаимодействия пользователя с приложением. Они осуществляют вывод данных, а также сбор и упаковку данных, введенных пользователем. При разработке слоя представления были использована стандартная библиотека jsp-тегов JSTL 1.2 и каскадные таблицы стилей (CSS).

Слой управления представлен сервлетом. Сервлет перенаправляет запросы, полученные от jsp-страниц на классы-команды, реализующие слой доступа к базе данных и бизнес-логики. Все операции обмена между jsp и базой данных протекают через сервлет. Сам сервлет не выполняет никаких действий над данными, он является посредником и контроллером запросов.

Классы бизнес-логики и доступа к базе данных обрабатывают данные, отправленные пользователем с jsp, выполняют запросы к базе данных, осуществляют операции вычисления над данными, генерируют исключения в случае некорректности данных и возвращают данные через сервлет на jsp.

Запуск web-приложения происходит с вывода страницы index.jsp, являющейся стартовой страницейй сайта. Далее пользователь, пользуясь сайтом, вызывает регулярное перенаправление от jsp-страниц к сервлету-контроллеру. Сервлет-контроллер в свою очередь создает объект класса Command, который содержит список возможных команд. Далее сервлет-контроллер определяет команду, пришедшую из JSP, и вызывает реализованный метод класса Command , и осуществляет передачу параметров классу-обработчику конкретной команды. В результате осуществляется вызов страницы ответа на полученный запрос.

В приложении Б представлена блок-схема работы приложения.

# 6 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

После загрузки сайта пользователь увидит страницу index.jsp формой для ввода данных для доступа к базе данных, представленной на рисунке 6.1. В форме необходимо указать имя пользователя, пароль и порт для сервера MySQL, установленного на компьютере сервера, а также имя базы данных, которая будет использоваться приложением. База будет создана либо подключена, если она уже существует.

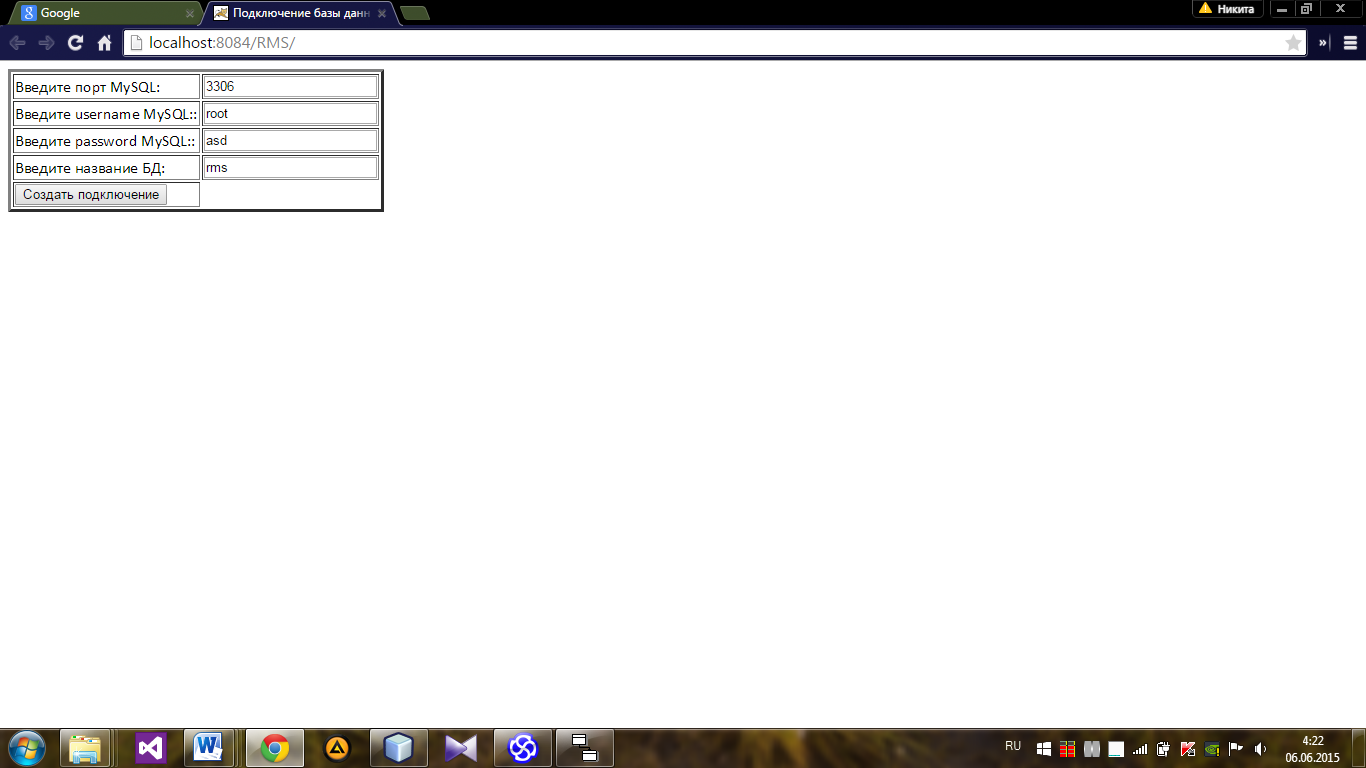


Рисунок 6.1 – Страница index.jsp

После нажатия на кнопку «Создать подключение» пользователь увидит главную страницу сайта, представленную на рисунке 6.2.

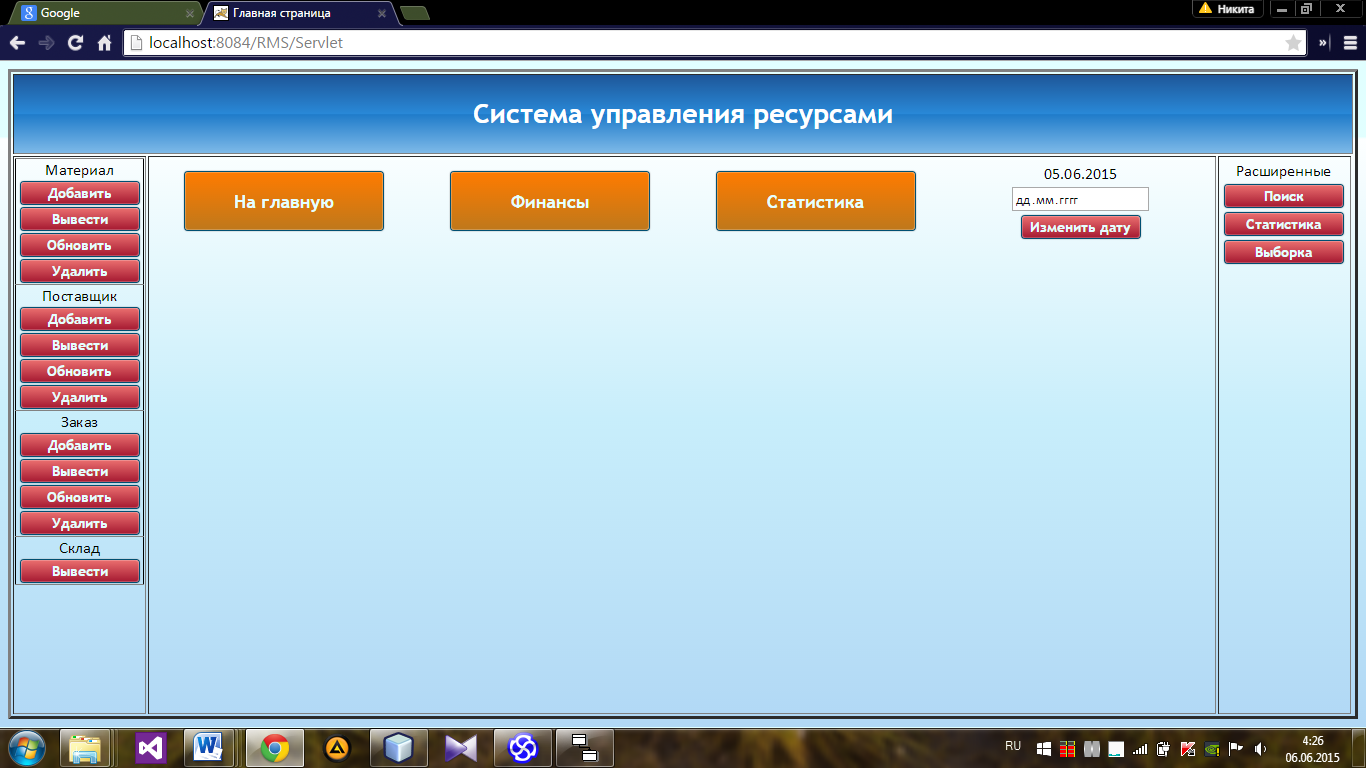


Рисунок 6.2 – Главная страница сайта

С главной страницы сайта пользователь может осуществлять все действия по учету материалов. Пользователь может добавить материал в базу данных. Страница добавления материала представлена на рисунке 6.3.

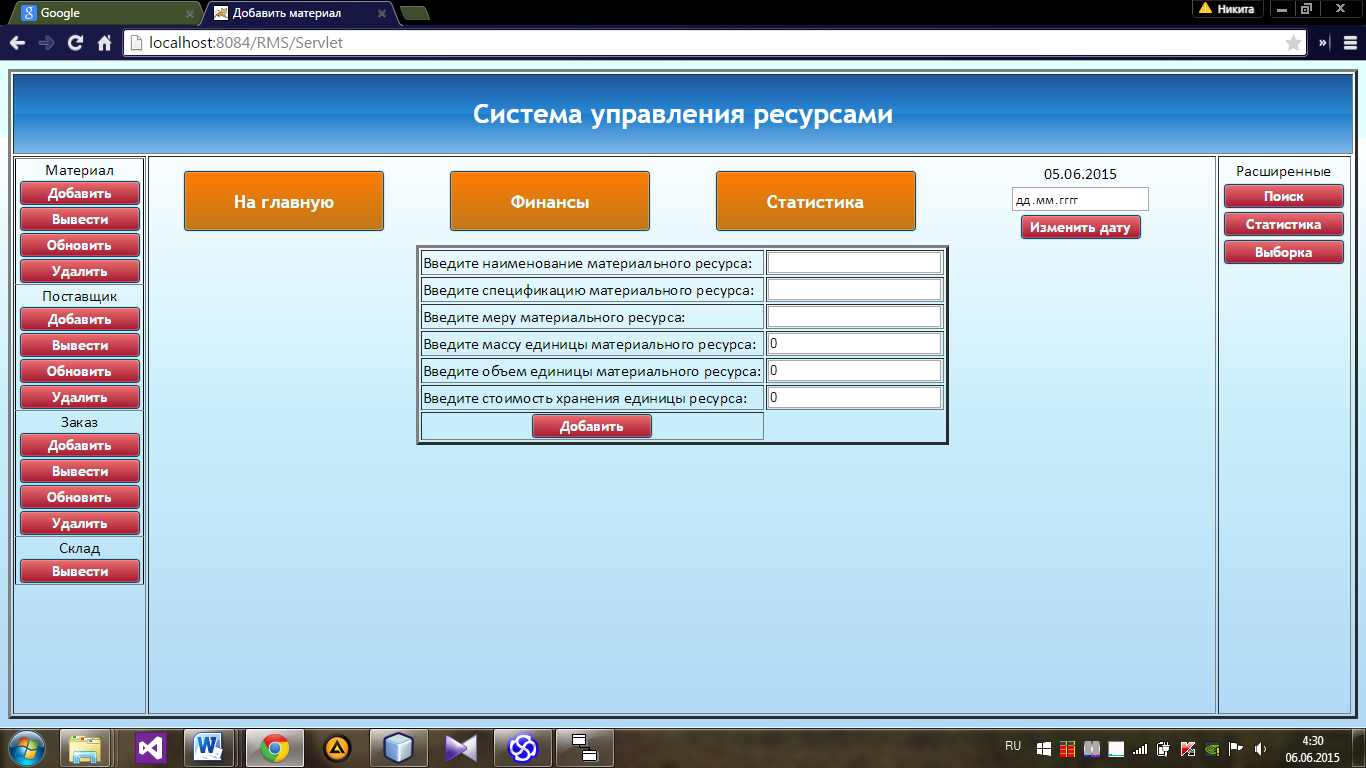


Рисунок 6.3 – Страница добавления материала в БД

Далее пользователь может просмотреть записи обо всех материалах. Страница просмотра записей материалов представлена на рисунке 6.4.

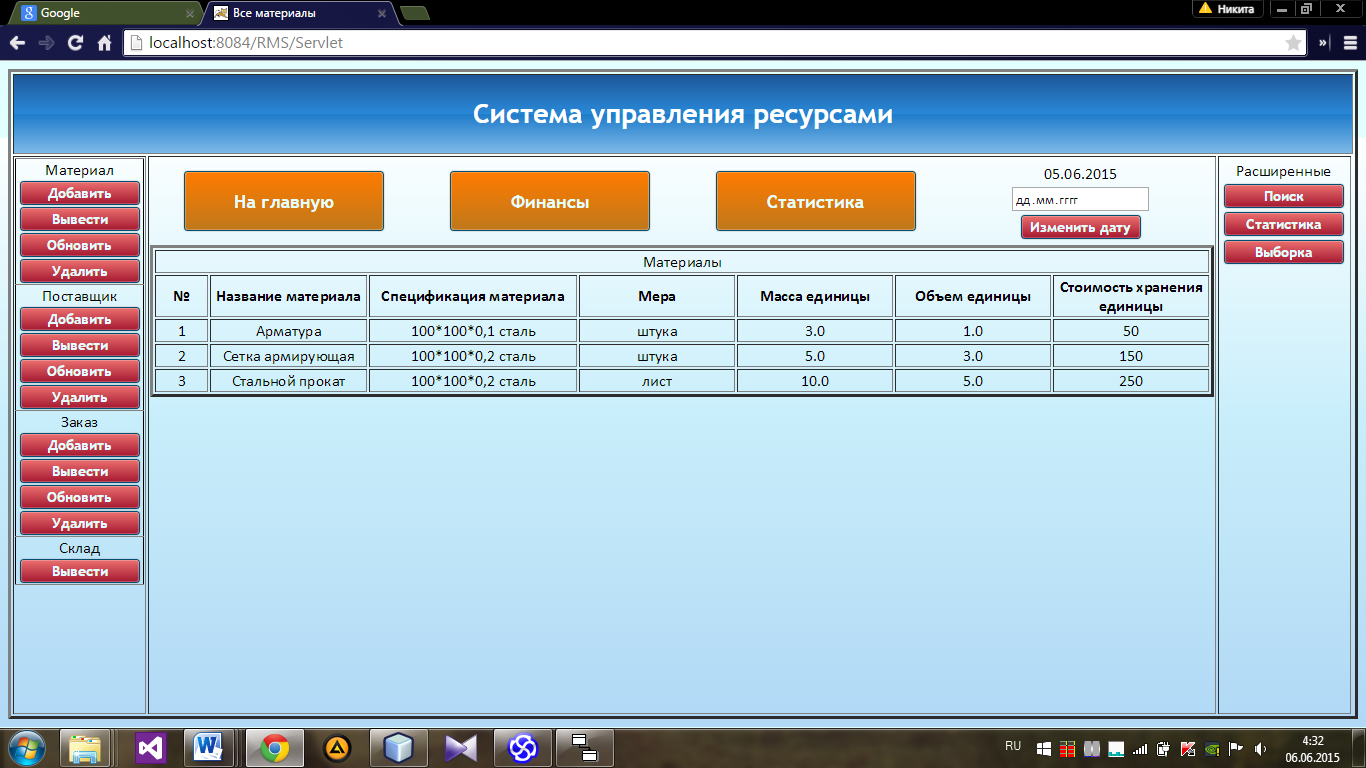


Рисунок 6.4 – Страница чтения материалов из БД

Страница редактирования записи материала представлена на рисунке 6.5.

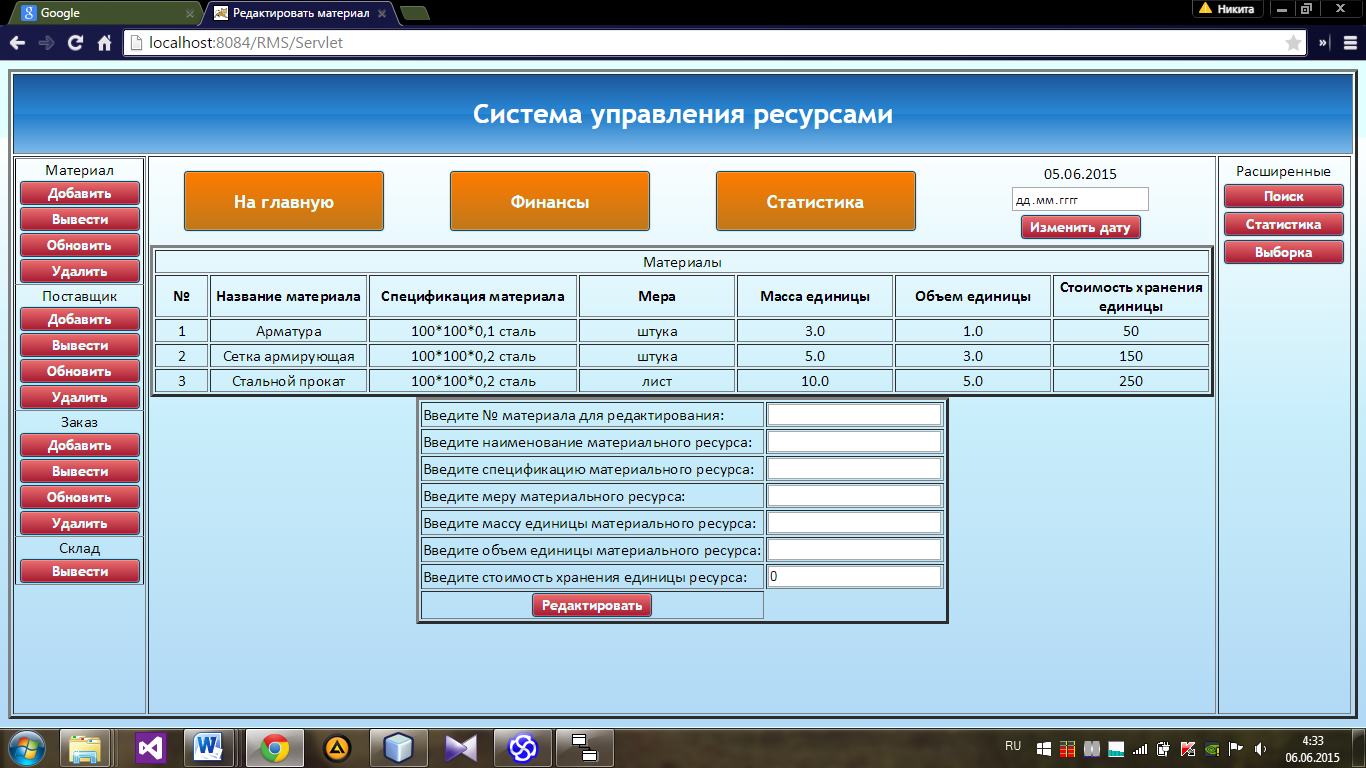


Рисунок 6.5 – Страница редактирования материалов

Страница удаления записи материала из БД представлена на рисунке 6.6.

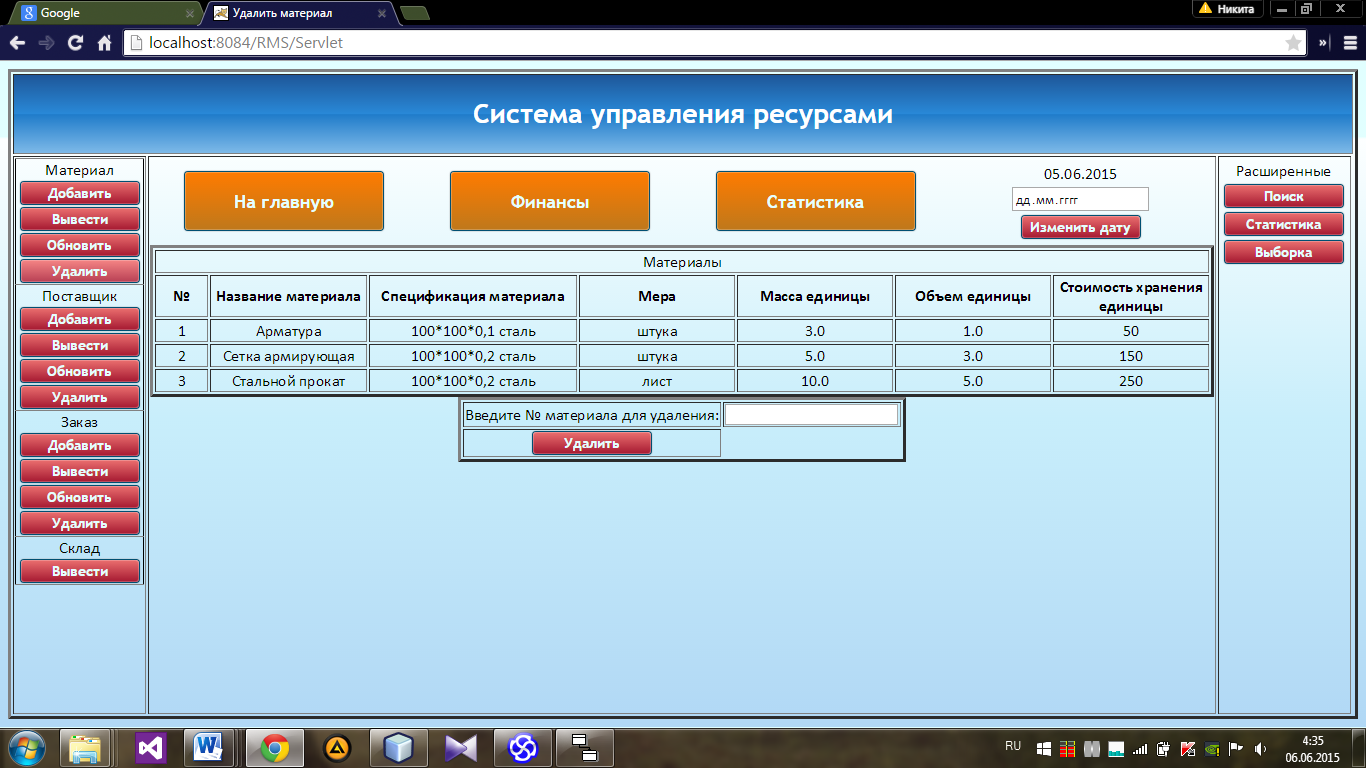


Рисунок 6.6 – Страница удаления материалов

Пользователь может осуществлять операции добавления, чтения, обновления и удаления для записей сущностей «Поставщик» и «Заказ».

Страница добавления поставщика представлена на рисунке 6.7

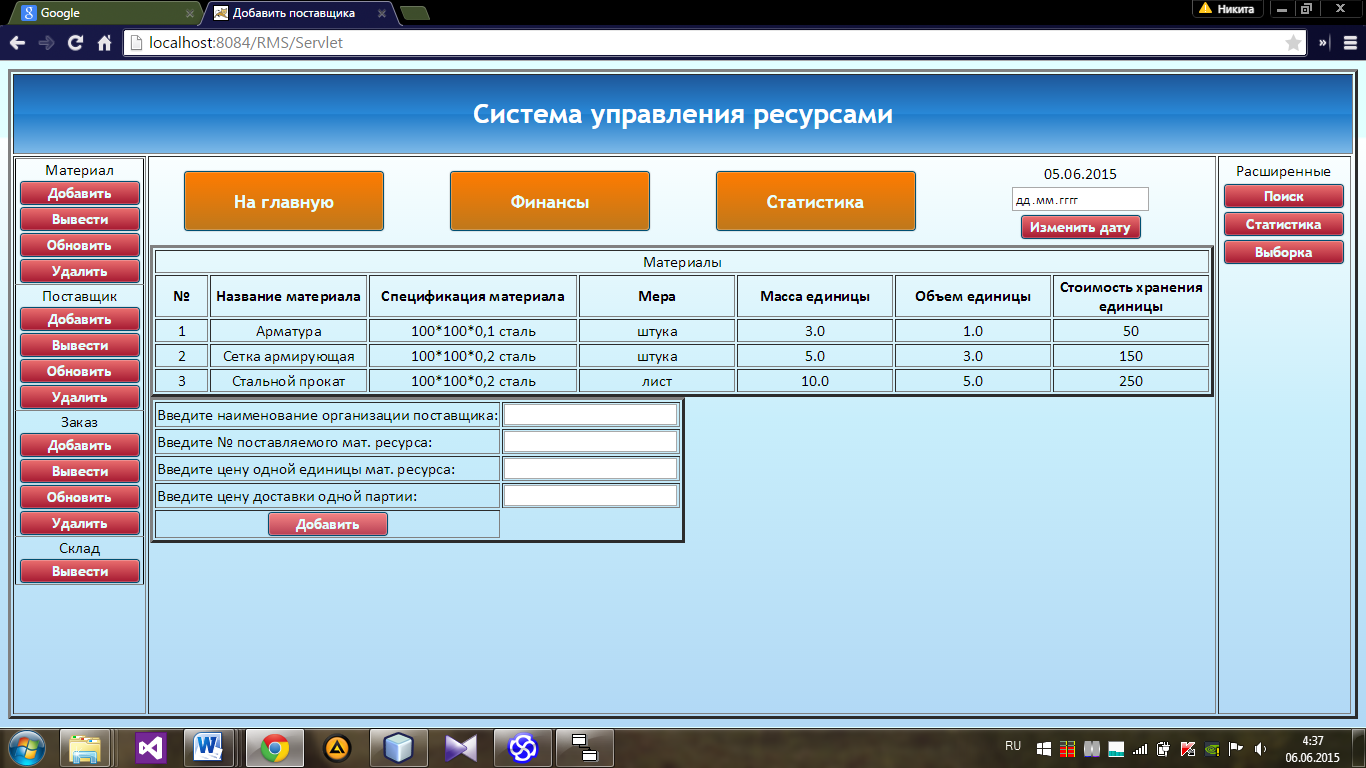


Рисунок 6.7 – Страница добавления поставщиков

В случае неверного заполнения формы пользователь увидит уведомление об ошибке, представленной на рисунке 6.8.

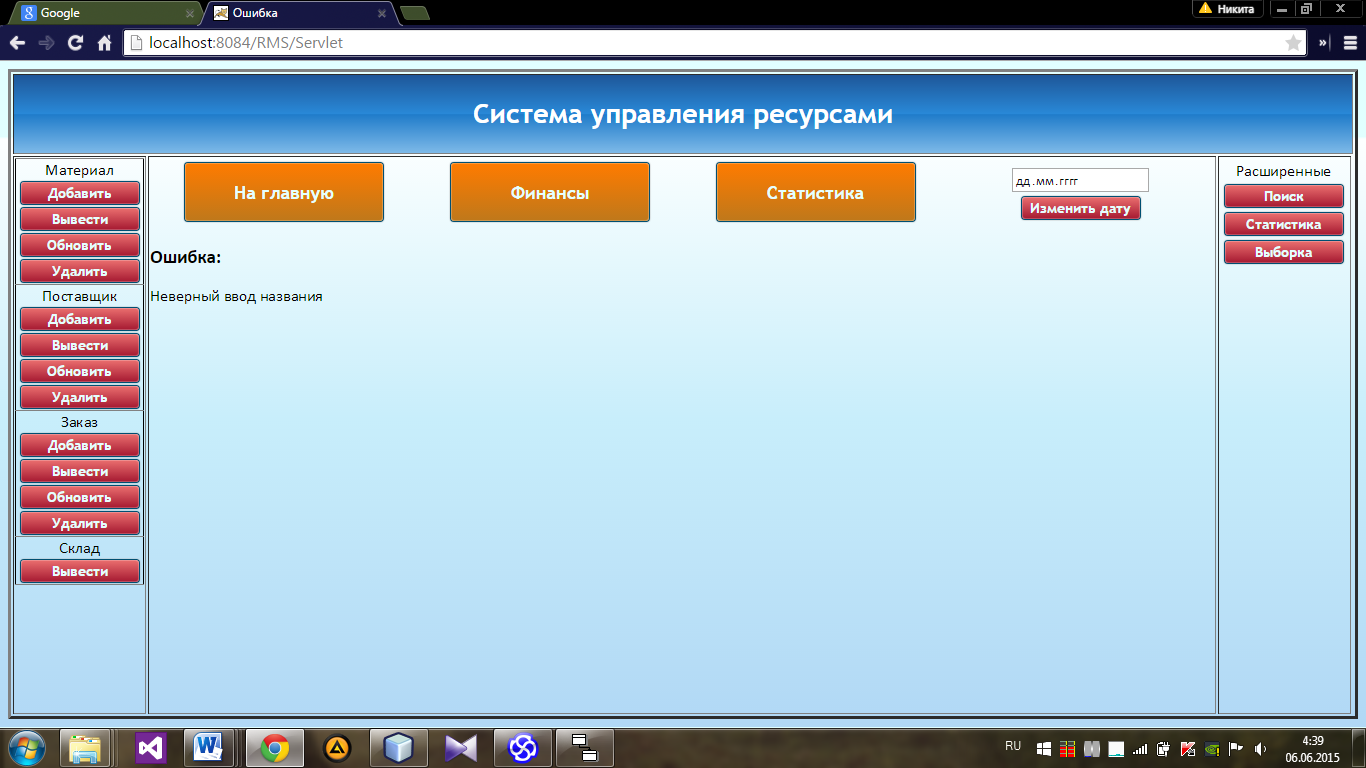


Рисунок 6.8 – Уведомление об ошибке при вводе формы

Страница чтения записей сущности поставщик представлена на рисунке 6.9.

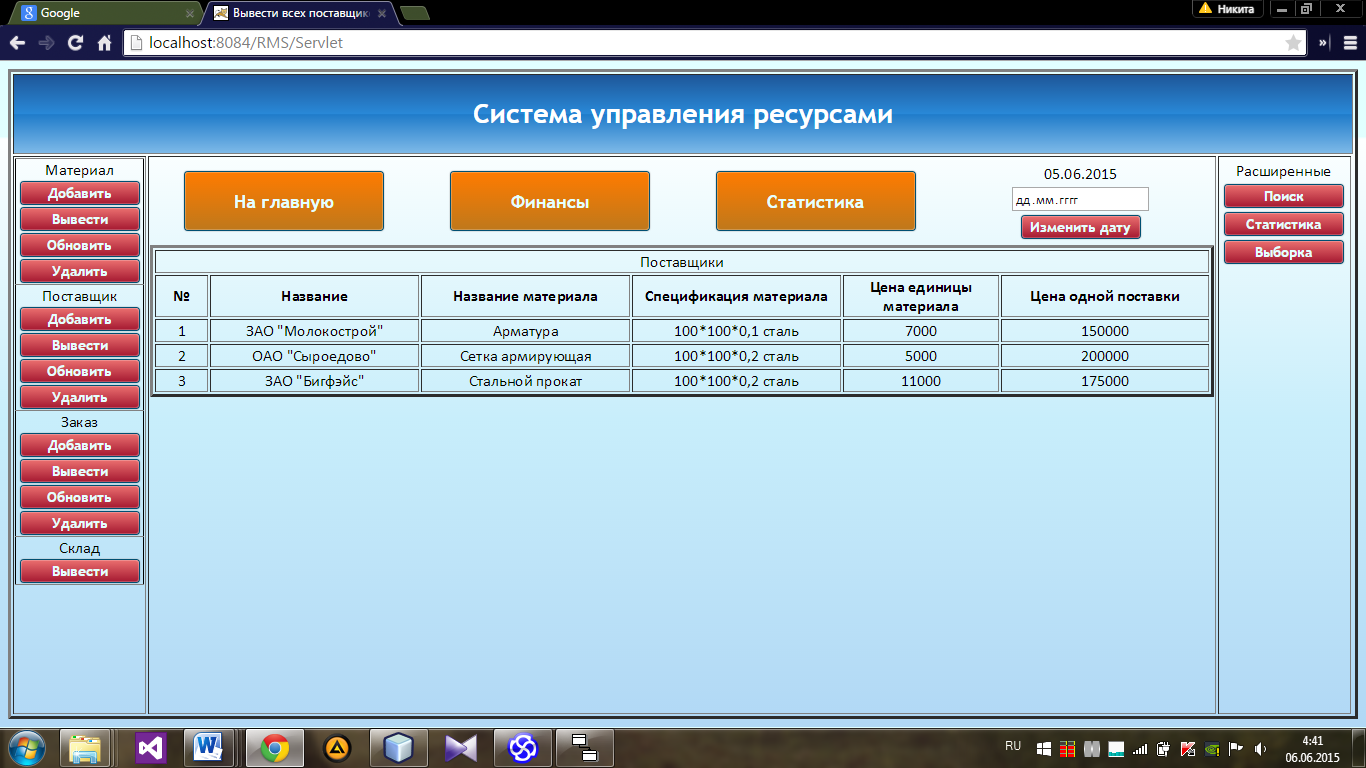


Рисунок 6.9 – Страница чтения поставщиков из БД

Страница обновления записей сущности «Поставщик» представлена на рисунке 6.10.

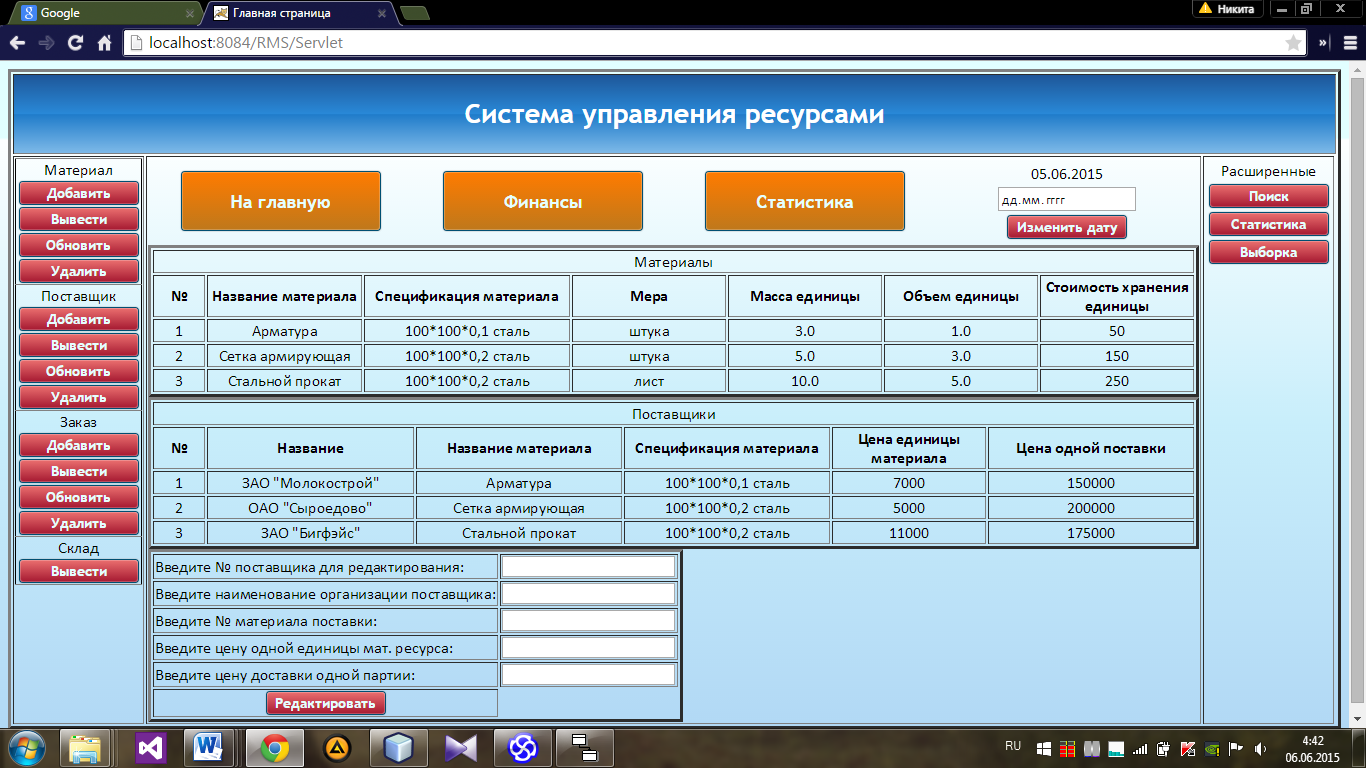


Рисунок 6.10 – Страница редактирования поставщиков

Страница удаления поставщиков из БД представлена на рисунке 6.11

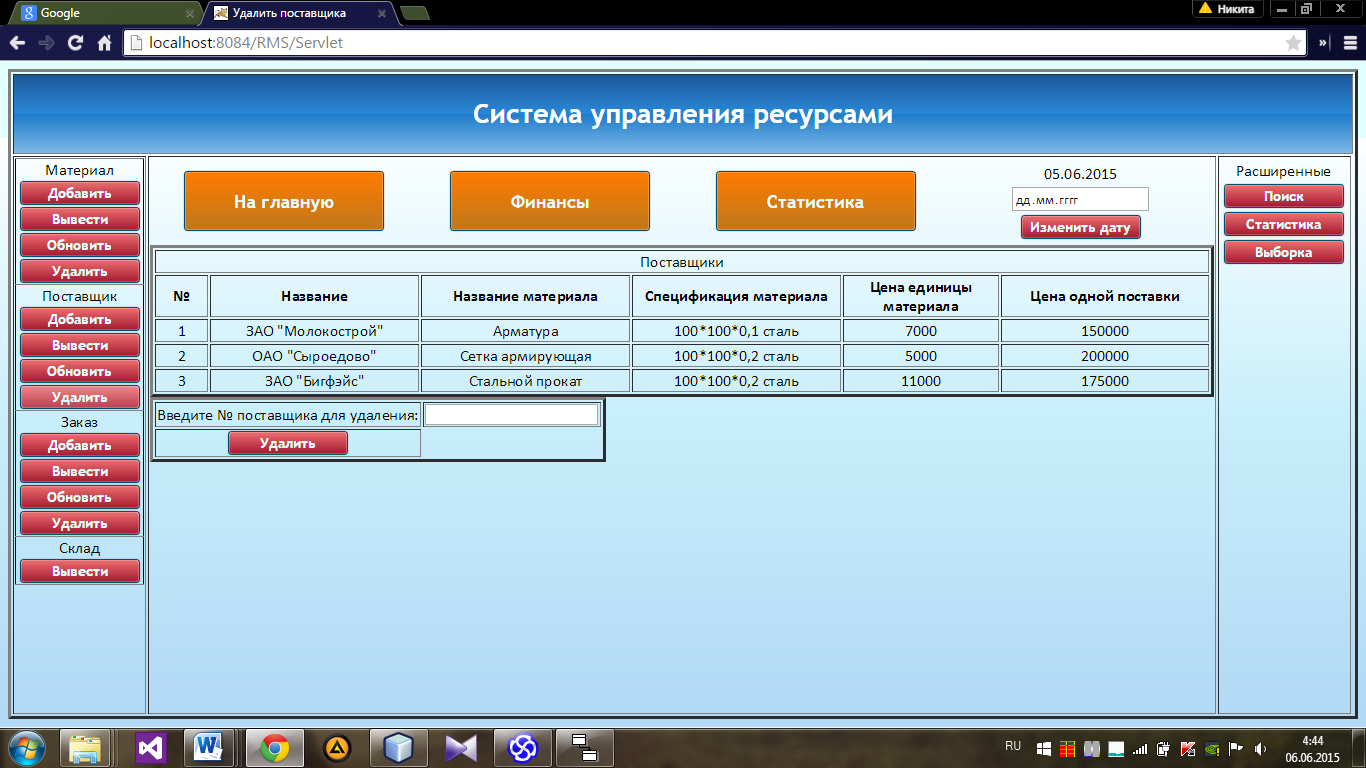


Рисунок 6.11 – Страница удаления поставщиков

Страницы для добавления, чтения, обновления и удаления заказов, а также для чтения склада выглядят аналогично соответствующим страницам материалов и поставщиков.

Пользователь может выполнить поиск по базе данных. Для этого с любой страницы ему необходимо нажать кнопку «Поиск», расположенную в правом верхнем углу. Результат нажатия кнопки «Поиск» представлен на рисунке 6.12.

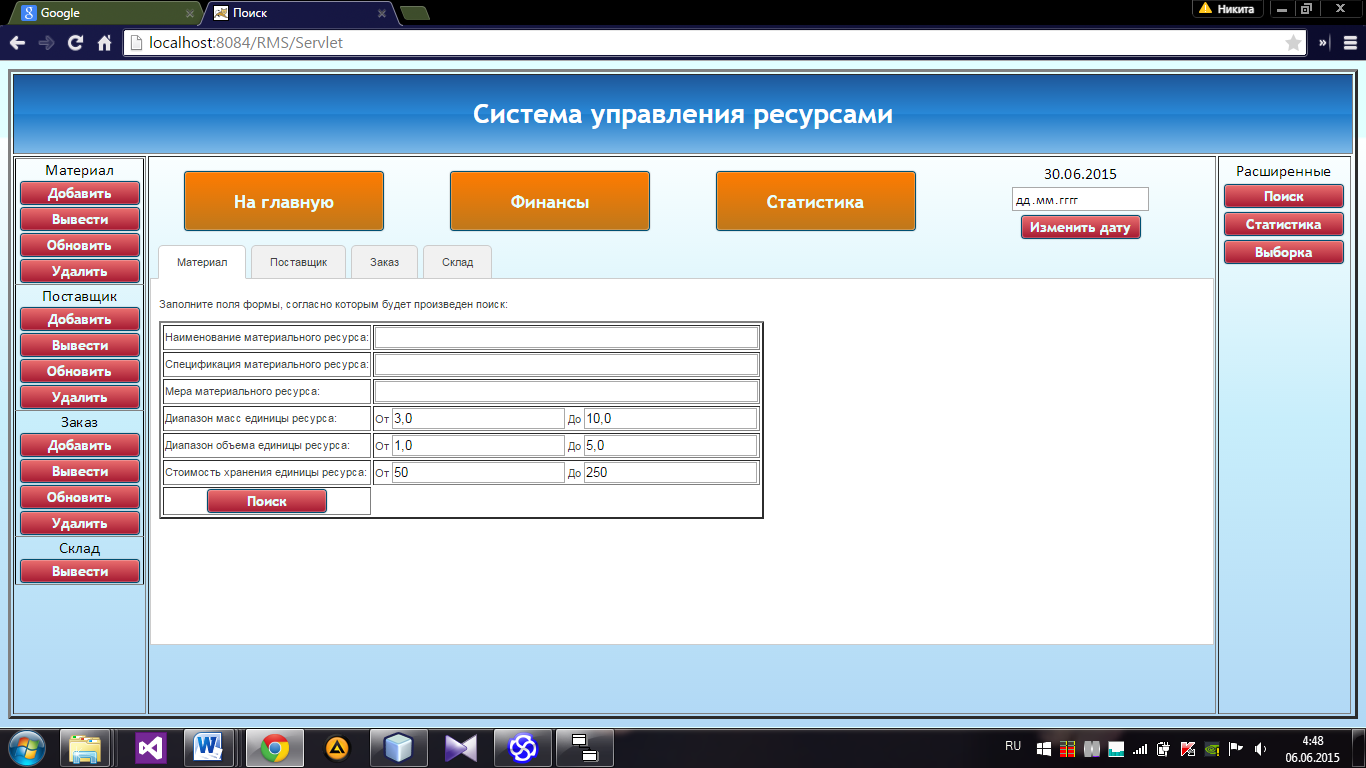


Рисунок 6.11 – Страница поиска в базе данных

Меню поиска выполнено в виде вкладок. Каждой из вкладок соответствует форма для поиска в базе данных одной из сущностей: «Материал», «Поставщик», «Заказ» и «Склад». Вкладки поименованы в соответствии с сущностями для поиска. В любой момент времени на странице поиска видна только одна форма поиска одной из сущностей, вкладка которой активна. Остальные три формы скрыты. Для открытия скрытой формы одной из сущностей пользователь должен нажать на название соответствующей вкладки. На рисунке 6.12 представлена страница поиска с активной вкладкой «Поставщик».

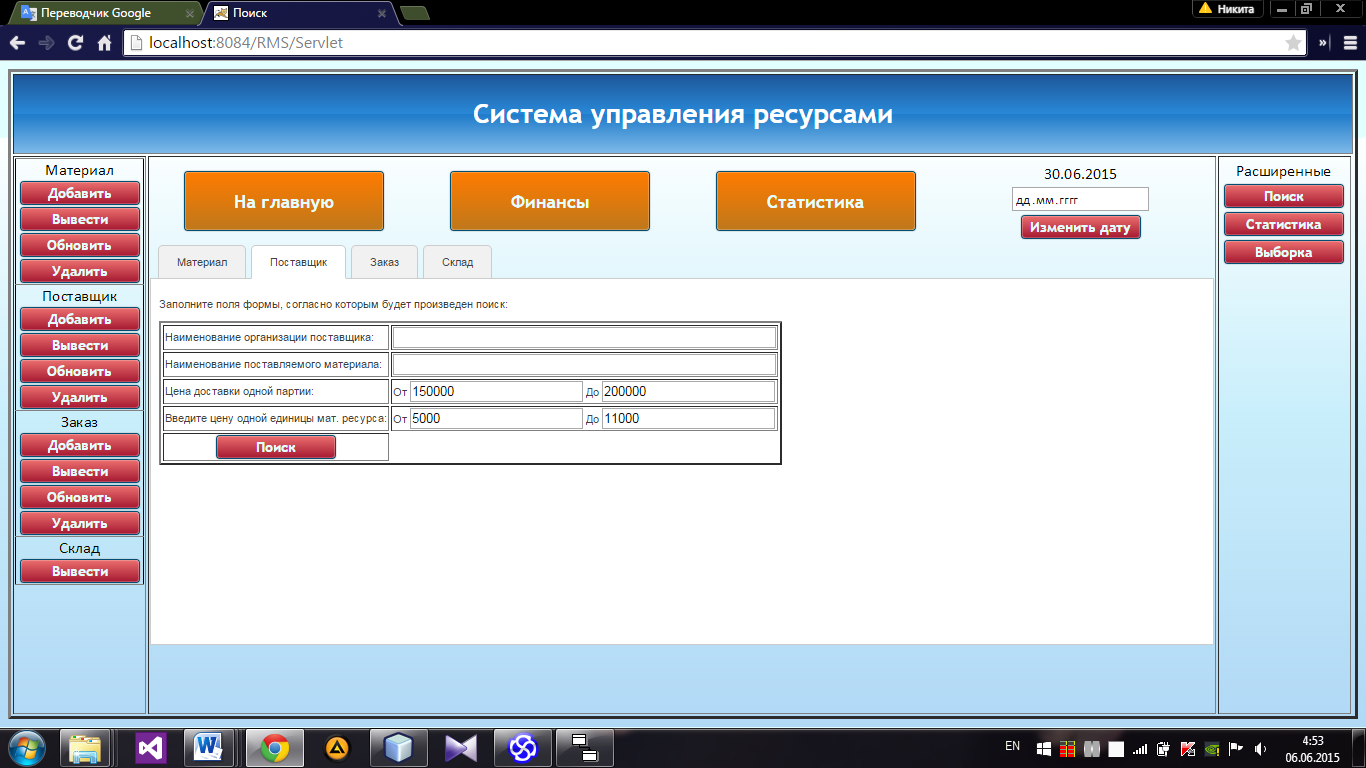


Рисунок 6.12 – Страница поиска с активной вкладкой «Поставщик»

В форме поиска для каждой из сущностей требуется ввести данные в те поля, по которым необходимо произвести поиск. Некоторые поля приложение заполняет автоматически: в них указываются граничные (максимальное и минимальное) значения для полей сущностей, содержащихся в данный момент в базе данных. Для осуществление поиска необходимо нажать на кнопку «Поиск» в требуемой форме. Результаты поиска представлены на рисунке 6.13.

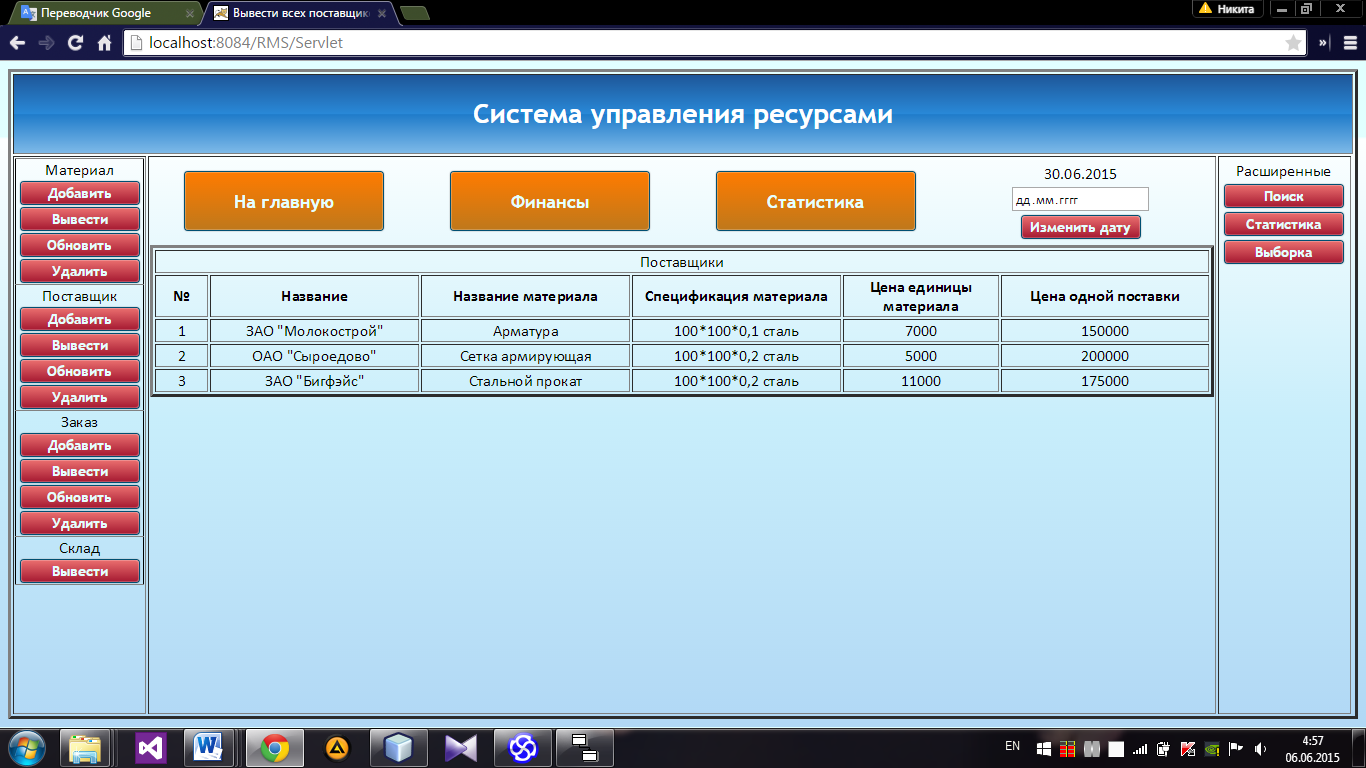


Рисунок 6.13 – Результаты поиска сущности «Поставщик»

Чтобы получить финансовый отчет о затратах предприятия на закупку и хранение материалов, пользователь должен нажать кнопку «Финансы» на любой странице приложения. Результат нажатия кнопки «Финансы» и стоимостной отчет представлены на рисунке 6.14.

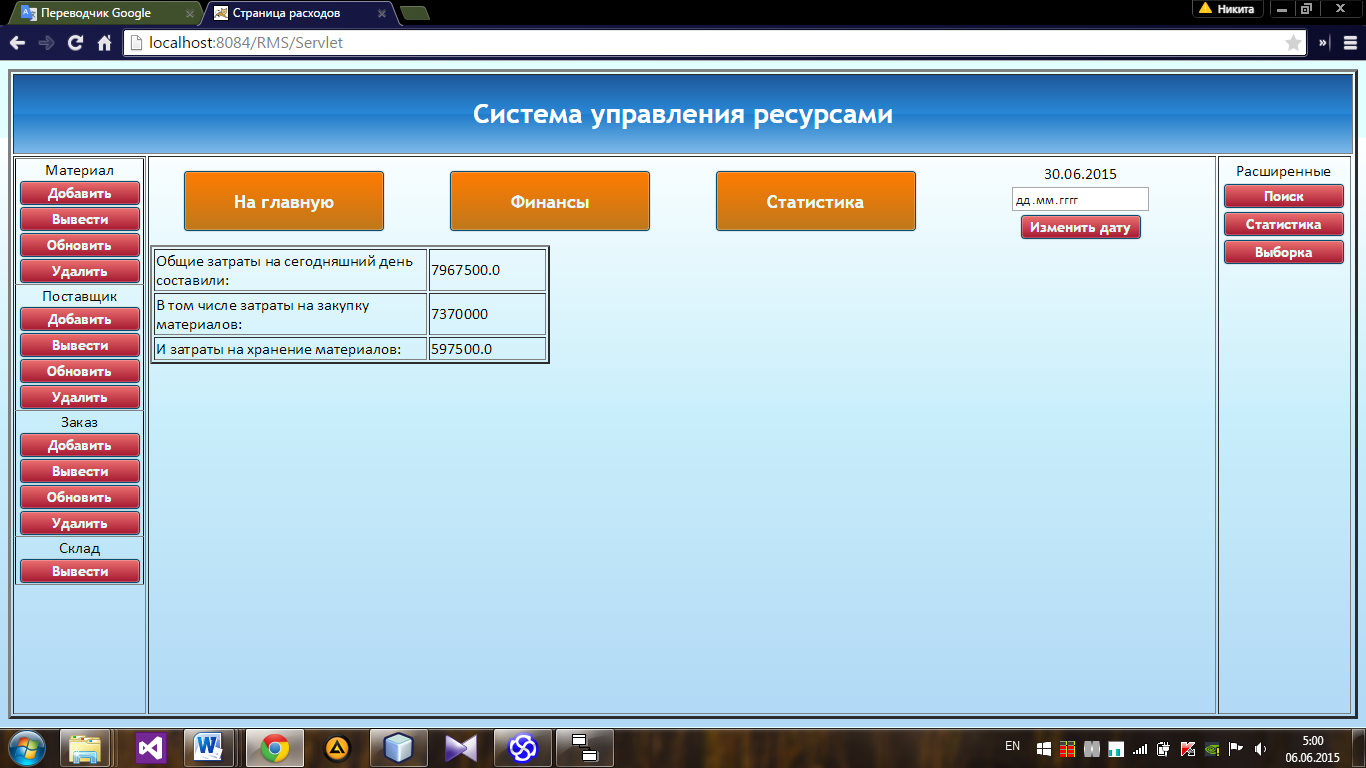


Рисунок 6.14 – Страница отчета о затратах

В приложении реализована функция изменения текущей даты. При запуске приложения дата автоматически устанавливается в текущую. Тем не менее, ее можно изменить. Это необходимо для моделирования выполнения заказов на поставку материалов на склад. В случае, если дата выполнения заказа окажется раньше выставленной даты, материалы данного заказа будут добавлены на склад. Если выставленная дата окажется раньше даты выполнения заказа, то соответствующие материалы будут изъяты со склада. При добавлении и изъятии материалов со склада меняется стоимостной отчет о затратах предприятия.

# ЗАКЛЮЧЕНИЕ

В данной курсовой работе было разработано приложение по учету материалов на предприятии, и была достигнута поставленная цель – повысить эффективность и качества управления промышленным предприятием путем автоматизации учета ресурсов. Эта цель была достигнута в результате решения следующих задач:

* Изучить особенности регистрации и учета материалов на промышленном предприятии;
* Изучить классификацию материалов в связи с проблематикой их учета;
* Изучить специфику процесса получения материальных ресурсов на предприятии;
* Ознакомиться с основными статистическими данными, собираемыми в процессе учета материальных ресурсов на производстве;
* Разработать систему учета материалов на предприятии.

В дальнейшем программу можно усовершенствовать путем добавления таблицы производств и распределения имеющихся на складе материалов по производствам, с получением в дальнейшем готовой продукции и получения прибыли от ее реализации.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Wikipedia [Электронный ресурс] – электронная энциклопедия. – Электронные данные. – Режим доступа: <http://www.wikipedia.org>

[2] Киммел, П. UML Универсальный язык проектирования / П. Киммел. – М.: НТ Пресс, 2008.

[3] Герберт Ш. Java. Полное руководство / Ш. Герберт. – М.: Вильямс,2012.

[4] JUnit [Электронный ресурс] – программно-ориентированный фреймворк для тестирования программ на языке Java. – Электронные данные. – Режим доступа: <http://www.junit.org>.

[5] JFreeChart[Электронный ресурс] – построение графиков для JavaSwing. – Электронные данные. – Режим доступа: <http://www.jfree.org/jfreechart/>

[7] Informicus [Электронный ресурс] – диаграмма классов. – Электронные данные. – Режим доступа:[http://www.informicus.ru/Default.aspx/](http://www.informicus.ru/Default.aspx/%20)

[8] it-gost [Электронный ресурс] – Теория и практика UML. Диаграмма состояний. – Электронные данные. – Режим доступа:[http://www.it-gost.ru/articles/view\_articles/](http://www.it-gost.ru/articles/view_articles/97)

ПРИЛОЖЕНИЕ А

**(обязательное)**

**Функциональная модель (IDEF0)**

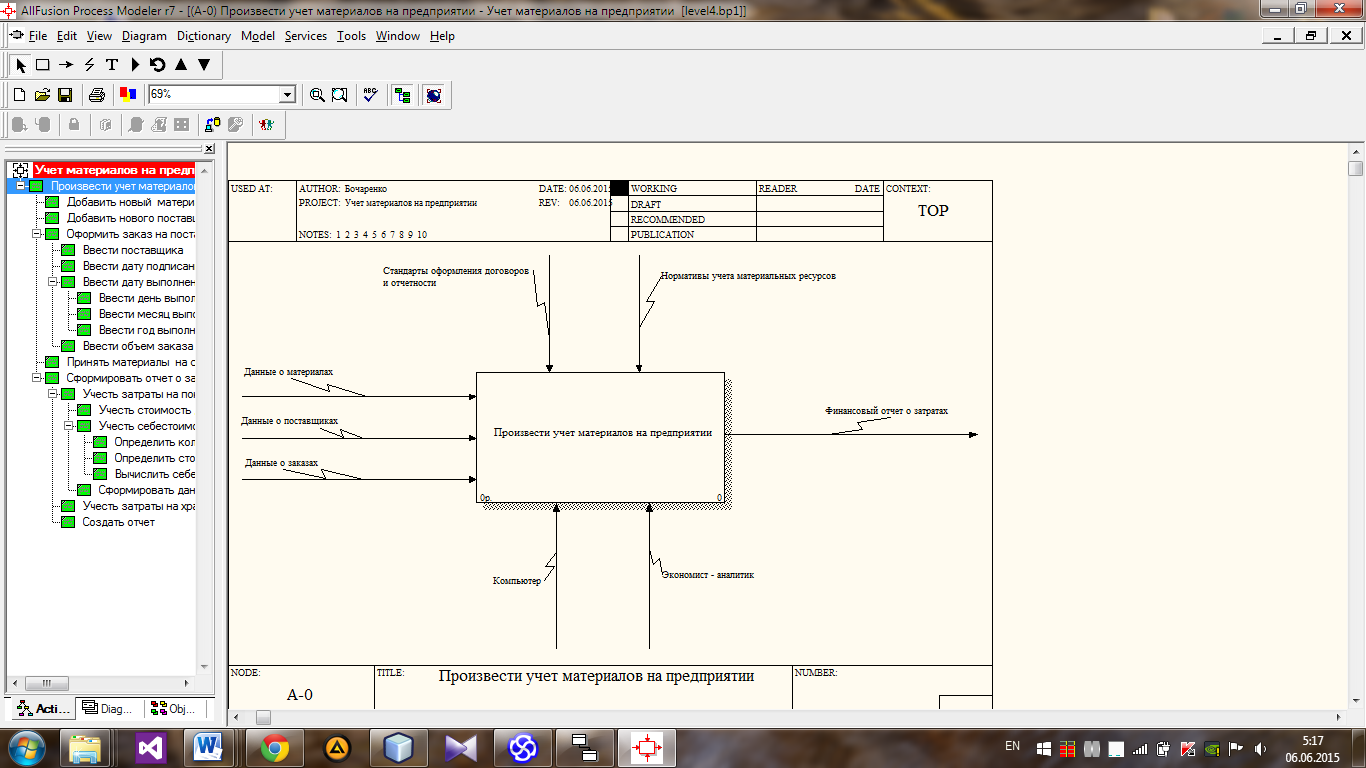


Рисунок А.1 – Главный процесс

Продолжение приложения А

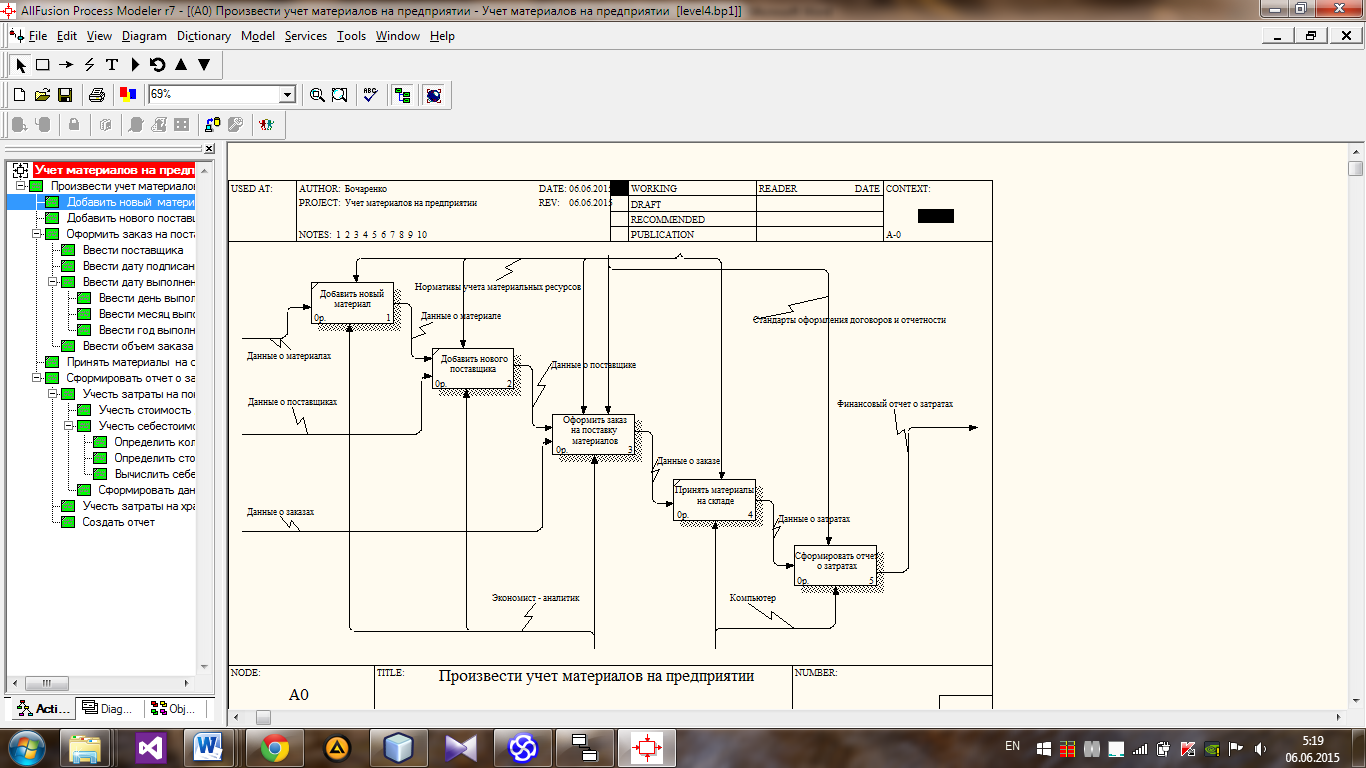


Рисунок А.2 – Декомпозиция главного процесса

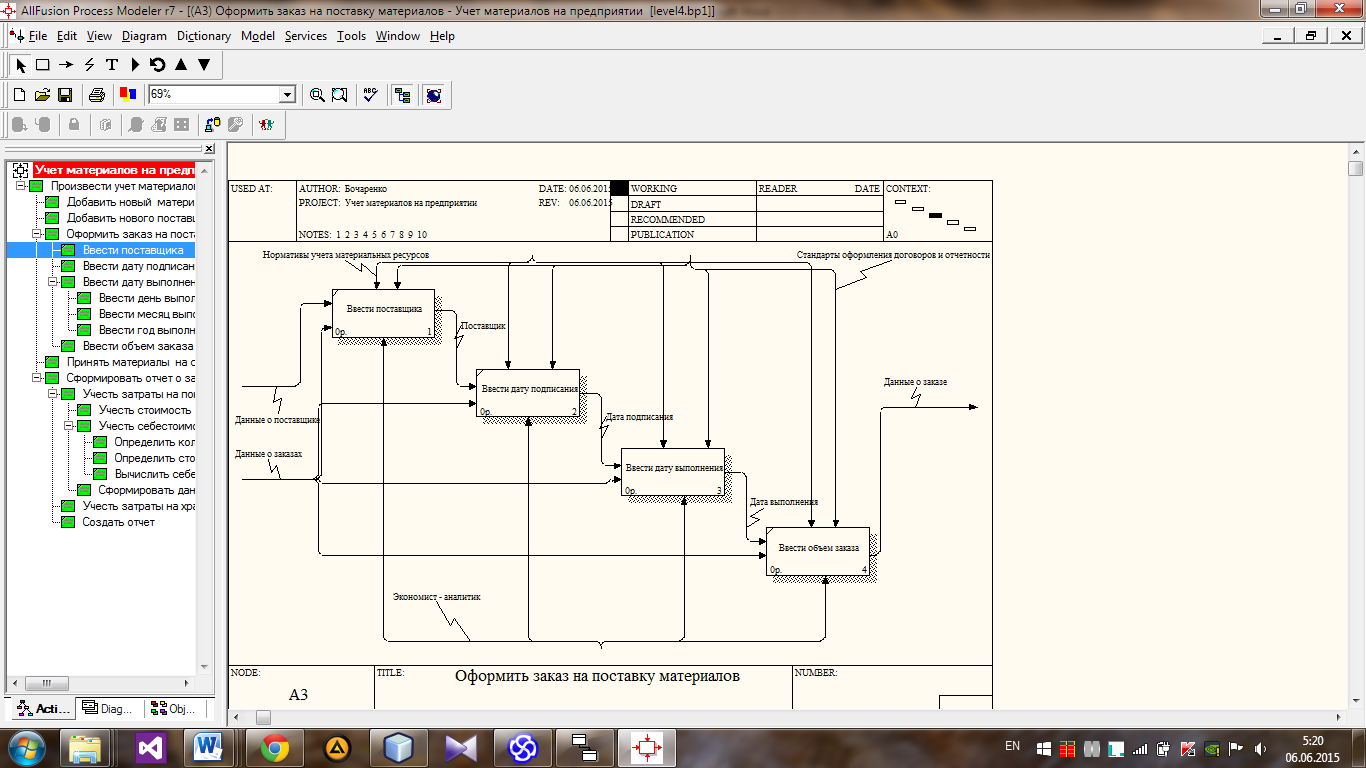


Рисунок А.3 – Декомпозиция процесса «Оформить заказ на поставку материалов»

Продолжение приложения А

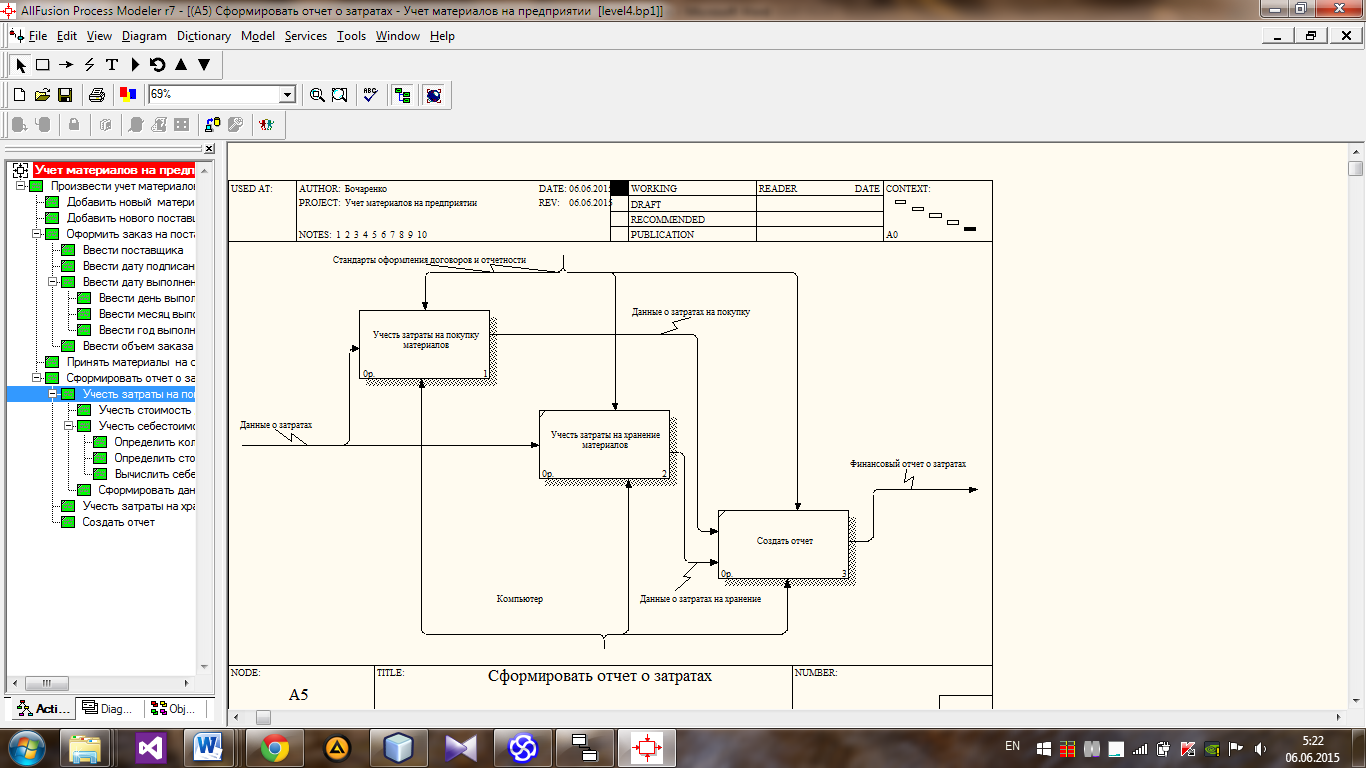


Рисунок А.4 – Декомпозиция процесса «Сформировать отчет о затратах»

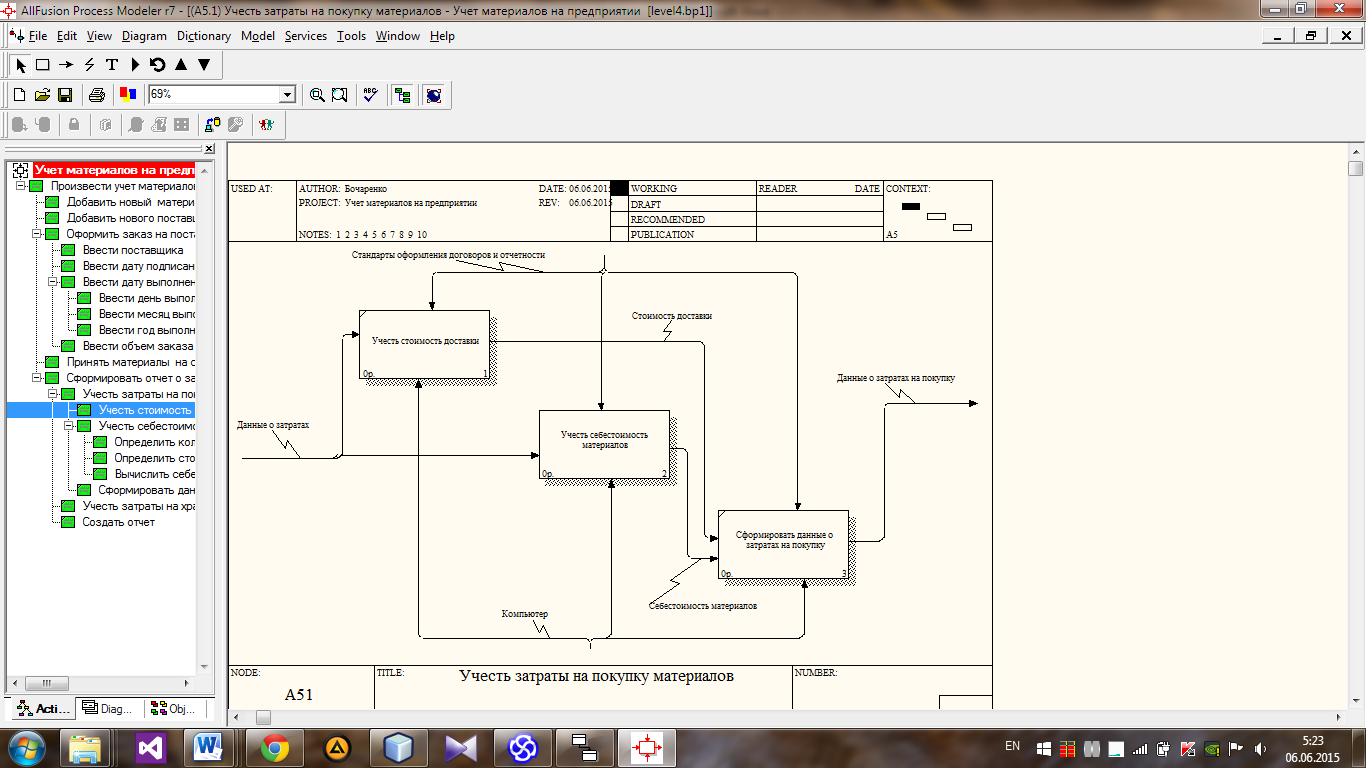


Рисунок А.5 – Декомпозиция процесса «Учесть затраты на покупку материалов»

Продолжение приложения А

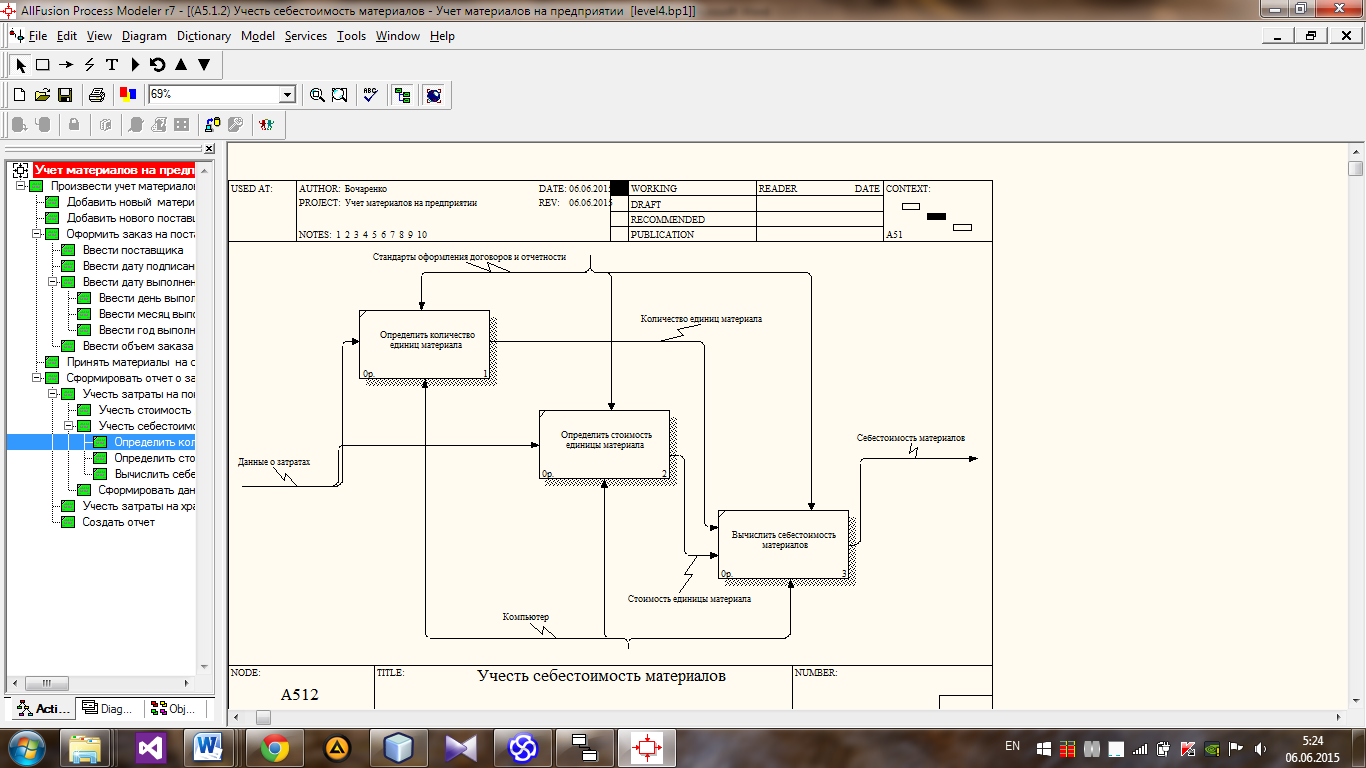


Рисунок А.6 – Декомпозиция процесса «Учесть себестоимость материалов»

# ПРИЛОЖЕНИЕ Б

**(обязательное)**

**Блок-схемы**



Рисунок Б.1 –Блок-схема алгоритма функции CreateMaterialCommand

Продолжение приложения Б



Продолжение приложения Б



Рисунок Б.1 –Блок-схема алгоритма функции ChangeDateCommand

# ПРИЛОЖЕНИЕ В

**(обязательное)**

**Листинг кода**

package by.bsuir.bocharenko.rms.entity;

import java.io.Serializable;

import java.util.List;

import java.util.Objects;

import javax.persistence.CascadeType;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.FetchType;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.OneToMany;

import javax.persistence.Table;

/\*\*

\*

\* @author user

\*/

@Entity

@Table(name="material")

public class Material implements Serializable{

private static final long serialVersionUID = 1L;

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

private Long id\_mat;

@Column(name="title")

private String title;

@Column(name="specification")

private String specification;

@Column(name="measure")

private String measure;

@Column(name="volume")

private Float volume;

@Column(name="weight")

private Float weight;

@Column(name="unitStorageCost")

private Integer unitStorageCost;

@OneToMany(mappedBy="material", fetch = FetchType.EAGER, cascade={CascadeType.MERGE, CascadeType.REMOVE})

protected List<Supplier> suppliers;

Продолжение приложения В

@OneToMany(mappedBy="material", fetch = FetchType.EAGER, cascade={CascadeType.MERGE, CascadeType.REMOVE})

protected List<Storage> storage;

public Material() {

}

public String getTitle() {

return title;

}

public void setTitle(String title) {

this.title = title;

}

public String getMeasure() {

return measure;

}

public void setMeasure(String measure) {

this.measure = measure;

}

public Float getVolume() {

return volume;

}

public void setVolume(Float volume) {

this.volume = volume;

}

public Float getWeight() {

return weight;

}

public void setWeight(Float weight) {

this.weight = weight;

}

public Long getId() {

return id\_mat;

}

public void setId(Long id) {

this.id\_mat = id;

}

public List<Supplier> getSuppliers() {

return suppliers;

}

Продолжение приложения В

public void setSuppliers(List<Supplier> suppliers) {

this.suppliers = suppliers;

}

public List<Storage> getStorage() {

return storage;

}

public void setStorage(List<Storage> storage) {

this.storage = storage;

}

public String getSpecification() {

return specification;

}

public void setSpecification(String specification) {

this.specification = specification;

}

public Integer getUnitStorageCost() {

return unitStorageCost;

}

public void setUnitStorageCost(Integer unitStorageCost) {

this.unitStorageCost = unitStorageCost;

}

@Override

public int hashCode() {

int hash = 0;

hash += (id\_mat != null ? id\_mat.hashCode() : 0);

return hash;

}

@Override

public boolean equals(Object obj) {

if (obj == null) {

return false;

}

if (getClass() != obj.getClass()) {

return false;

}

final Material other = (Material) obj;

if (!Objects.equals(this.id\_mat, other.id\_mat)) {

return false;

}

if (!Objects.equals(this.title, other.title)) {

return false;

}

return true;

Продолжение приложения В

}

/\*@Override

public boolean equals(Object object) {

// TODO: Warning - this method won't work in the case the id fields are not set

if (!(object instanceof Material)) {

return false;

}

Material other = (Material) object;

if ((this.id == null && other.id != null) || (this.id != null && !this.id.equals(other.id))) {

return false;

}

return true;

}\*/

@Override

public String toString() {

return "Material{" + "id=" + id\_mat + ", title=" + title + ", measure=" + measure + ", volume=" + volume + ", weight=" + weight + ", suppliers=" + suppliers + '}';

}

}

package by.bsuir.bocharenko.rms.entity;

import java.io.Serializable;

import java.util.List;

import javax.persistence.CascadeType;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.FetchType;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.JoinColumn;

import javax.persistence.ManyToOne;

import javax.persistence.OneToMany;

/\*\*

\*

\* @author user

\*/

@Entity

public class Supplier implements Serializable {

private static final long serialVersionUID = 1L;

Продолжение приложения В

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

private Long id\_sup;

@Column(name="title")

private String title;

@Column(name="deliveryPrice")

private Integer deliveryPrice;

@Column(name="resourceUnitPrice")

private Integer resourceUnitPrice;

@OneToMany(mappedBy="supplier", fetch = FetchType.EAGER, cascade={CascadeType.MERGE, CascadeType.REMOVE})

protected List<Orderr> orders;

@ManyToOne(fetch = FetchType.EAGER,cascade = CascadeType.PERSIST)

@JoinColumn(name = "id\_mat", nullable = false)

protected Material material;

public Supplier() {

}

public String getTitle() {

return title;

}

public void setTitle(String title) {

this.title = title;

}

public Integer getDeliveryPrice() {

return deliveryPrice;

}

public void setDeliveryPrice(Integer deliveryPrice) {

this.deliveryPrice = deliveryPrice;

}

public Integer getResourceUnitPrice() {

return resourceUnitPrice;

}

public void setResourceUnitPrice(Integer resourceUnitPrice) {

this.resourceUnitPrice = resourceUnitPrice;

}

public Long getId() {

return id\_sup;

}

Продолжение приложения В

public void setId(Long id) {

this.id\_sup = id;

}

public Material getMaterial() {

return material;

}

public void setMaterial(Material material) {

this.material = material;

}

public List<Orderr> getOrders() {

return orders;

}

public void setOrders(List<Orderr> orders) {

this.orders = orders;

}

@Override

public int hashCode() {

int hash = 0;

hash += (id\_sup != null ? id\_sup.hashCode() : 0);

return hash;

}

@Override

public boolean equals(Object object) {

// TODO: Warning - this method won't work in the case the id fields are not set

if (!(object instanceof Supplier)) {

return false;

}

Supplier other = (Supplier) object;

if ((this.id\_sup == null && other.id\_sup != null) || (this.id\_sup != null && !this.id\_sup.equals(other.id\_sup))) {

return false;

}

return true;

}

@Override

public String toString() {

return "Supplier{" + "id=" + id\_sup + ", title=" + title + ", deliveryPrice=" + deliveryPrice + ", resourceUnitPrice=" + resourceUnitPrice + ", orders=" + orders + ", material=" + material + '}';

}

Продолжение приложения В

}

package by.bsuir.bocharenko.rms.entity;

import java.io.Serializable;

import java.util.Date;

import javax.persistence.CascadeType;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.FetchType;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.JoinColumn;

import javax.persistence.ManyToOne;

import javax.persistence.Table;

import javax.persistence.Temporal;

import javax.persistence.TemporalType;

@Entity

@Table(name="Orderr")

public class Orderr implements Serializable {

private static final long serialVersionUID = 1L;

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

private Long id\_ord;

public Orderr() {

}

public Long getId() {

return id\_ord;

}

public void setId(Long id) {

this.id\_ord = id;

}

@Column(name = "sign\_date")

@Temporal(TemporalType.DATE)

private Date signDate;

@Column(name = "exec\_date")

@Temporal(TemporalType.DATE)

private Date execDate;

@Column(name = "totalVolume")

private Integer totalVolume;

Продолжение приложения В

@Column(name = "totalCost")

private Integer totalCost;

@Column(name = "isCompleted")

private String isCompleted;

@ManyToOne(fetch = FetchType.EAGER,cascade = CascadeType.PERSIST)

@JoinColumn(name = "id\_sup", nullable = false)

protected Supplier supplier;

public Date getSignDate() {

return signDate;

}

public void setSignDate(Date signDate) {

this.signDate = signDate;

}

public Date getExecDate() {

return execDate;

}

public void setExecDate(Date execDate) {

this.execDate = execDate;

}

public Integer getTotalVolume() {

return totalVolume;

}

public void setTotalVolume(Integer totalVolume) {

this.totalVolume = totalVolume;

}

public Integer getTotalCost() {

return totalCost;

}

public void setTotalCost(Integer totalCost) {

this.totalCost = totalCost;

}

public Supplier getSupplier() {

return supplier;

}

public void setSupplier(Supplier supplier) {

this.supplier = supplier;

}

public String getIsCompleted() {

Продолжение приложения В

return isCompleted;

}

public void setIsCompleted(String isCompleted) {

this.isCompleted = isCompleted;

}

@Override

public int hashCode() {

int hash = 0;

hash += (id\_ord != null ? id\_ord.hashCode() : 0);

return hash;

}

@Override

public boolean equals(Object object) {

// TODO: Warning - this method won't work in the case the id fields are not set

if (!(object instanceof Orderr)) {

return false;

}

Orderr other = (Orderr) object;

if ((this.id\_ord == null && other.id\_ord != null) || (this.id\_ord != null && !this.id\_ord.equals(other.id\_ord))) {

return false;

}

return true;

}

@Override

public String toString() {

return "Orderr{" + "id=" + id\_ord + ", signDate=" + signDate + ", execDate=" + execDate + ", totalVolume=" + totalVolume + ", totalCost=" + totalCost + ", supplier=" + supplier + '}';

}

}

package by.bsuir.bocharenko.rms.entity;

import java.io.Serializable;

import java.util.Date;

import javax.persistence.CascadeType;

import javax.persistence.Column;

import javax.persistence.Entity;

Продолжение приложения В

import javax.persistence.FetchType;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.JoinColumn;

import javax.persistence.ManyToOne;

import javax.persistence.Temporal;

import javax.persistence.TemporalType;

/\*\*

\*

\* @author user

\*/

@Entity

public class Storage implements Serializable {

private static final long serialVersionUID = 1L;

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

private Long id;

@ManyToOne(fetch = FetchType.EAGER,cascade = CascadeType.PERSIST)

@JoinColumn(name = "id\_mat", nullable = false)

protected Material material;

@Column(name="numberOfMaterialUnits")

private Integer numberOfMaterialUnits;

@Column(name="totalVolumeOfMaterial")

private Float totalVolumeOfMaterial;

@Column(name="receiptDate")

@Temporal(TemporalType.DATE)

private Date receiptDate;

public Storage() {

}

public Long getId() {

return id;

}

public Date getReceiptDate() {

return receiptDate;

}

public void setReceiptDate(Date receiptDate) {

this.receiptDate = receiptDate;

}

public void setId(Long id) {

this.id = id;

Продолжение приложения В

}

public Material getMaterial() {

return material;

}

public void setMaterial(Material material) {

this.material = material;

}

public Integer getNumberOfMaterialUnits() {

return numberOfMaterialUnits;

}

public void setNumberOfMaterialUnits(Integer numberOfMaterialUnits) {

this.numberOfMaterialUnits = numberOfMaterialUnits;

}

public Float getTotalVolumeOfMaterial() {

return totalVolumeOfMaterial;

}

public void setTotalVolumeOfMaterial(Float totalVolumeOfMaterial) {

this.totalVolumeOfMaterial = totalVolumeOfMaterial;

}

@Override

public int hashCode() {

int hash = 0;

hash += (id != null ? id.hashCode() : 0);

return hash;

}

@Override

public boolean equals(Object object) {

// TODO: Warning - this method won't work in the case the id fields are not set

if (!(object instanceof Storage)) {

return false;

}

Storage other = (Storage) object;

if ((this.id == null && other.id != null) || (this.id != null && !this.id.equals(other.id))) {

return false;

}

return true;

}

@Override

public String toString() {

Продолжение приложения В

return "Storage{" + "id=" + id + ", material=" + material + ", numberOfMaterialUnits=" + numberOfMaterialUnits + ", totalVolumeOfMaterial=" + totalVolumeOfMaterial + '}';

}

}

package by.bsuir.bocharenko.rms.controller;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.SQLException;

import java.util.HashMap;

import java.util.Map;

import java.util.logging.Level;

import java.util.logging.Logger;

import javax.persistence.EntityManagerFactory;

import javax.persistence.Persistence;

public class EMFactory {

private static EMFactory instance;

private EntityManagerFactory emf;

private static String dbname;

private static String port;

private static String username;

private static String password;

private String URL = "jdbc:mysql://localhost/";

private String Driver = "com.mysql.jdbc.Driver";

public EMFactory() throws SQLException{

try {

Class.forName(Driver);

java.util.Properties connectionProperties = new java.util.Properties();

connectionProperties.put("driver", Driver);

connectionProperties.put("user", username);

connectionProperties.put("password", password);

Connection con = DriverManager.getConnection(URL, connectionProperties);

String query = "CREATE DATABASE IF NOT EXISTS " + dbname;

PreparedStatement ps = con.prepareStatement(query);

ps.executeUpdate();

} catch (ClassNotFoundException | SQLException ex) {

ex.printStackTrace();

}

Продолжение приложения В

Map properties = new HashMap();

properties.put("hibernate.connection.url", "jdbc:mysql://localhost:"+port+"/"+dbname);

properties.put("hibernate.connection.username", username);

properties.put("hibernate.connection.password", password);

emf = Persistence.createEntityManagerFactory("RMS", properties);

}

public static EMFactory getInstance(){

if(instance==null)

try {

instance = new EMFactory();

} catch (SQLException ex) {

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

}

return instance;

}

public EntityManagerFactory getEntityManagerFactory(){

return emf;

}

public static String getPort() {

return port;

}

public static void setPort(String port) {

EMFactory.port = port;

}

public static String getUsername() {

return username;

}

public static void setUsername(String username) {

EMFactory.username = username;

}

public static String getPassword() {

return password;

}

public static void setPassword(String password) {

EMFactory.password = password;

}

public static String getDbname() {

return dbname;

}

public static void setDbname(String dbname) {

Продолжение приложения В

EMFactory.dbname = dbname;

}

}

package by.bsuir.bocharenko.rms.controller;

import by.bsuir.bocharenko.rms.controller.exceptions.NonexistentEntityException;

import by.bsuir.bocharenko.rms.entity.Material;

import by.bsuir.bocharenko.rms.entity.Orderr;

import by.bsuir.bocharenko.rms.entity.Storage;

import java.io.Serializable;

import javax.persistence.Query;

import javax.persistence.criteria.CriteriaQuery;

import javax.persistence.criteria.Root;

import by.bsuir.bocharenko.rms.entity.Supplier;

import java.util.ArrayList;

import java.util.List;

import javax.persistence.EntityManager;

import javax.persistence.EntityManagerFactory;

/\*\*

\*

\* @author user

\*/

public class MaterialJpaController implements Serializable {

public MaterialJpaController(EntityManagerFactory emf) {

this.emf = emf;

}

private EntityManagerFactory emf = null;

public EntityManager getEntityManager() {

return emf.createEntityManager();

}

public void create(Material material) {

if (material.getSuppliers() == null) {

material.setSuppliers(new ArrayList<>());

}

EntityManager em = null;

try {

em = getEntityManager();

em.getTransaction().begin();

em.persist(material);

em.getTransaction().commit();

} finally {

Продолжение приложения В

if (em != null) {

em.close();

}

}

}

public void edit(Material material) throws NonexistentEntityException{

EntityManager em = null;

if(findMaterial(material.getId())==null)

throw new NonexistentEntityException("Материал с таким номером " + material.getId() + " не зарегистрирован в БД.");

try {

em = getEntityManager();

em.getTransaction().begin();

Material persistentMaterial = em.find(Material.class, material.getId());

List<Supplier> suppliersOld = persistentMaterial.getSuppliers();

List<Supplier> suppliersNew = new ArrayList();

if(suppliersOld!=null){

for (Supplier oldSup : suppliersOld) {

oldSup.setMaterial(material);

em.merge(oldSup);

suppliersNew.add(oldSup);

}

}

material.setSuppliers(suppliersNew);

List<Storage> storageOld = persistentMaterial.getStorage();

List<Storage> storageNew = new ArrayList();

if(storageOld!=null){

for (Storage oldstorage : storageOld) {

oldstorage.setMaterial(material);

em.merge(oldstorage);

storageNew.add(oldstorage);

}

}

material.setStorage(storageNew);

em.merge(material);

em.getTransaction().commit();

} finally {

if (em != null) {

em.close();

}

}

}

public void destroy(Long id) throws NonexistentEntityException {

EntityManager em = null;

if(findMaterial(id)==null)

throw new NonexistentEntityException("Материал с таким номером " + id + " не зарегистрирован в БД.");

try {

Продолжение приложения В

em = getEntityManager();

em.getTransaction().begin();

Material material;

material = em.getReference(Material.class, id);

List<Supplier> orphanSuppliers = material.getSuppliers();

if(orphanSuppliers!=null){

for(Supplier orphanSup : orphanSuppliers){

List<Orderr> orphanOrders = orphanSup.getOrders();

if(orphanOrders!=null)

for(Orderr orphanOrd : orphanOrders)

em.remove(orphanOrd);

em.remove(orphanSup);

}

}

List<Storage> orphanStorages = material.getStorage();

if(orphanStorages!=null){

for(Storage orphanStorage : orphanStorages)

em.remove(orphanStorage);

}

em.remove(material);

em.getTransaction().commit();

} finally {

if (em != null) {

em.close();

}

}

}

public List<Material> findMaterialEntities() {

return findMaterialEntities(true, -1, -1);

}

public List<Material> findMaterialEntities(int maxResults, int firstResult) {

return findMaterialEntities(false, maxResults, firstResult);

}

private List<Material> findMaterialEntities(boolean all, int maxResults, int firstResult) {

EntityManager em = getEntityManager();

try {

CriteriaQuery cq = em.getCriteriaBuilder().createQuery();

cq.select(cq.from(Material.class));

Query q = em.createQuery(cq);

if (!all) {

q.setMaxResults(maxResults);

q.setFirstResult(firstResult);

}

return q.getResultList();

} finally {

Продолжение приложения В

em.close();

}

}

public Material findMaterial(Long id) {

EntityManager em = getEntityManager();

try {

return em.find(Material.class, id);

} finally {

em.close();

}

}

public int getMaterialCount() {

EntityManager em = getEntityManager();

try {

CriteriaQuery cq = em.getCriteriaBuilder().createQuery();

Root<Material> rt = cq.from(Material.class);

cq.select(em.getCriteriaBuilder().count(rt));

Query q = em.createQuery(cq);

return ((Long) q.getSingleResult()).intValue();

} finally {

em.close();

}

}

}

package by.bsuir.bocharenko.rms.controller;

import by.bsuir.bocharenko.rms.controller.exceptions.NonexistentEntityException;

import java.io.Serializable;

import javax.persistence.Query;

import javax.persistence.criteria.CriteriaQuery;

import javax.persistence.criteria.Root;

import by.bsuir.bocharenko.rms.entity.Material;

import by.bsuir.bocharenko.rms.entity.Orderr;

import by.bsuir.bocharenko.rms.entity.Supplier;

import java.util.ArrayList;

import java.util.List;

import javax.persistence.EntityManager;

import javax.persistence.EntityManagerFactory;

/\*\*

\*

\* @author user

\*/

public class SupplierJpaController implements Serializable {

Продолжение приложения В

public SupplierJpaController(EntityManagerFactory emf) {

this.emf = emf;

}

private EntityManagerFactory emf = null;

public EntityManager getEntityManager() {

return emf.createEntityManager();

}

public void create(Supplier supplier) throws NonexistentEntityException {

EntityManager em = null;

try {

em = getEntityManager();

em.getTransaction().begin();

Material material = em.getReference(Material.class, supplier.getMaterial().getId());

if (material == null)

throw new NonexistentEntityException("Материал с таким номером " + supplier.getMaterial().getId() + " не зарегистрирован в БД.");

material = em.getReference(Material.class, material.getId());

supplier.setMaterial(material);

em.persist(supplier);

List<Supplier> suppliers = material.getSuppliers();

suppliers.add(supplier);

material.setSuppliers(suppliers);

em.merge(material);

em.getTransaction().commit();

} finally {

if (em != null) {

em.close();

}

}

}

public void edit(Supplier supplier) throws NonexistentEntityException, Exception {

EntityManager em = null;

if (findSupplier(supplier.getId()) == null)

throw new NonexistentEntityException("Поставщик с таким номером " + supplier.getId()

+ " не зарегестрирован в БД");

try {

em = getEntityManager();

em.getTransaction().begin();

//Изменяется родительская сущность Материал

Supplier persistentSupplier = em.find(Supplier.class, supplier.getId());

Material materialOld = persistentSupplier.getMaterial();

Material materialNew = em.find(Material.class, supplier.getMaterial().getId());

if(materialNew==null)

Продолжение приложения В

throw new NonexistentEntityException("Материал с таким номером " + supplier.getMaterial().getId()

+ " не зарегистрирован в БД.");

if (!materialNew.equals(materialOld)){

List<Supplier> oldSuppliers = materialOld.getSuppliers();

oldSuppliers.remove(persistentSupplier);

materialOld.setSuppliers(oldSuppliers);

em.merge(materialOld);

List<Supplier> newSuppliers = materialNew.getSuppliers();

newSuppliers.add(supplier);

materialNew.setSuppliers(newSuppliers);

em.merge(materialNew);

}

else {

List<Supplier> oldSuppliers = materialOld.getSuppliers();

oldSuppliers.remove(persistentSupplier);

oldSuppliers.add(supplier);

materialOld.setSuppliers(oldSuppliers);

em.merge(materialOld);

}

supplier.setMaterial(materialNew);

//Каскадно изменяются заказы данного поставщика

List<Orderr> ordersOld = persistentSupplier.getOrders();

List<Orderr> ordersNew = new ArrayList();

if(ordersOld!=null){

for (Orderr oldOrder : ordersOld) {

oldOrder.setSupplier(supplier);

em.merge(oldOrder);

ordersNew.add(oldOrder);

}

}

supplier.setOrders(ordersNew);

em.merge(supplier);

em.getTransaction().commit();

} finally {

if (em != null) {

em.close();

}

}

}

public void destroy(Long id) throws NonexistentEntityException {

EntityManager em = null;

if(findSupplier(id)==null)

throw new NonexistentEntityException("Поставщик с таким номером " + id

+ " не зарегестрирован в БД");

try {

em = getEntityManager();

em.getTransaction().begin();

//Редактирование родительской сущности Material

Supplier supplier = em.getReference(Supplier.class, id);

Продолжение приложения В

Material material = supplier.getMaterial();

List<Supplier> suppliers = material.getSuppliers();

suppliers.remove(supplier);

material.setSuppliers(suppliers);

em.merge(material);

//Удаление заказов-сирот

List<Orderr> orphanOrders = supplier.getOrders();

if(orphanOrders!=null)

for(Orderr ord : orphanOrders){

em.remove(ord);

}

em.remove(supplier);

em.getTransaction().commit();

} finally {

if (em != null) {

em.close();

}

}

}

public List<Supplier> findSupplierEntities() {

return findSupplierEntities(true, -1, -1);

}

public List<Supplier> findSupplierEntities(int maxResults, int firstResult) {

return findSupplierEntities(false, maxResults, firstResult);

}

private List<Supplier> findSupplierEntities(boolean all, int maxResults, int firstResult) {

EntityManager em = getEntityManager();

try {

CriteriaQuery cq = em.getCriteriaBuilder().createQuery();

cq.select(cq.from(Supplier.class));

Query q = em.createQuery(cq);

if (!all) {

q.setMaxResults(maxResults);

q.setFirstResult(firstResult);

}

return q.getResultList();

} finally {

em.close();

}

}

public Supplier findSupplier(Long id) {

EntityManager em = getEntityManager();

try {

return em.find(Supplier.class, id);

} finally {

Продолжение приложения В

em.close();

}

}

public int getSupplierCount() {

EntityManager em = getEntityManager();

try {

CriteriaQuery cq = em.getCriteriaBuilder().createQuery();

Root<Supplier> rt = cq.from(Supplier.class);

cq.select(em.getCriteriaBuilder().count(rt));

Query q = em.createQuery(cq);

return ((Long) q.getSingleResult()).intValue();

} finally {

em.close();

}

}

}

package by.bsuir.bocharenko.rms.controller;

import by.bsuir.bocharenko.rms.controller.exceptions.NonexistentEntityException;

import by.bsuir.bocharenko.rms.entity.Orderr;

import by.bsuir.bocharenko.rms.entity.Supplier;

import java.io.Serializable;

import java.util.ArrayList;

import java.util.List;

import javax.persistence.EntityManager;

import javax.persistence.EntityManagerFactory;

import javax.persistence.Query;

import javax.persistence.criteria.CriteriaQuery;

import javax.persistence.criteria.Root;

/\*\*

\*

\* @author user

\*/

public class OrderrJpaController implements Serializable {

public OrderrJpaController(EntityManagerFactory emf) {

this.emf = emf;

}

private EntityManagerFactory emf = null;

public EntityManager getEntityManager() {

return emf.createEntityManager();

}

public void create(Orderr orderr) throws NonexistentEntityException{

EntityManager em = null;

Продолжение приложения В

try {

em = getEntityManager();

em.getTransaction().begin();

Supplier supplier = em.getReference(Supplier.class, orderr.getSupplier().getId());

if (supplier == null)

throw new NonexistentEntityException("Поставщик с таким номером " + orderr.getSupplier().getId()

+ " не зарегистрирован в БД.");

orderr.setSupplier(supplier);

em.persist(orderr);

if (supplier.getOrders()==null)

supplier.setOrders(new ArrayList<>());

List<Orderr> orders = supplier.getOrders();

orders.add(orderr);

supplier.setOrders(orders);

em.merge(supplier);

em.getTransaction().commit();

} finally {

if (em != null) {

em.close();

}

}

}

public void edit(Orderr orderr) throws NonexistentEntityException, Exception {

EntityManager em = null;

if(findOrderr(orderr.getId())==null)

throw new NonexistentEntityException("Заказ с таким номером " + orderr.getId() + " не зарегистрирован в БД.");

try {

em = getEntityManager();

em.getTransaction().begin();

Orderr persistentOrder = em.find(Orderr.class, orderr.getId());

Supplier supplierOld = persistentOrder.getSupplier();

Supplier supplierNew = em.getReference(Supplier.class, orderr.getSupplier().getId());

if(supplierNew==null)

throw new NonexistentEntityException("Поставщик с таким номером " + orderr.getSupplier().getId()

+ " не зарегистрирован в БД.");

if (!supplierNew.equals(supplierOld)){

List<Orderr> oldOrders = supplierOld.getOrders();

oldOrders.remove(persistentOrder);

supplierOld.setOrders(oldOrders);

em.merge(supplierOld);

List<Orderr> newOrders = supplierNew.getOrders();

newOrders.add(orderr);

supplierNew.setOrders(newOrders);

em.merge(supplierNew);

}

Продолжение приложения В

else {

List<Orderr> oldOrders = supplierOld.getOrders();

oldOrders.remove(persistentOrder);

oldOrders.add(orderr);

supplierOld.setOrders(oldOrders);

em.merge(supplierOld);

}

orderr.setSupplier(supplierNew);

em.merge(orderr);

em.getTransaction().commit();

} finally {

if (em != null) {

em.close();

}

}

}

public void destroy(Long id) throws NonexistentEntityException {

EntityManager em = null;

if(findOrderr(id)==null)

throw new NonexistentEntityException("Заказ с таким номером " + id + " не зарегистрирован в БД.");

try {

em = getEntityManager();

em.getTransaction().begin();

Orderr order = em.getReference(Orderr.class, id);

Supplier supplier = order.getSupplier();

List<Orderr> orders = supplier.getOrders();

orders.remove(order);

supplier.setOrders(orders);

em.merge(supplier);

em.remove(order);

em.getTransaction().commit();

} finally {

if (em != null) {

em.close();

}

}

}

public List<Orderr> findOrderrEntities() {

return findOrderrEntities(true, -1, -1);

}

public List<Orderr> findOrderrEntities(int maxResults, int firstResult) {

return findOrderrEntities(false, maxResults, firstResult);

}

private List<Orderr> findOrderrEntities(boolean all, int maxResults, int firstResult) {

EntityManager em = getEntityManager();

Продолжение приложения В

try {

CriteriaQuery cq = em.getCriteriaBuilder().createQuery();

cq.select(cq.from(Orderr.class));

Query q = em.createQuery(cq);

if (!all) {

q.setMaxResults(maxResults);

q.setFirstResult(firstResult);

}

return q.getResultList();

} finally {

em.close();

}

}

public Orderr findOrderr(Long id) {

EntityManager em = getEntityManager();

try {

return em.find(Orderr.class, id);

} finally {

em.close();

}

}

public int getOrderrCount() {

EntityManager em = getEntityManager();

try {

CriteriaQuery cq = em.getCriteriaBuilder().createQuery();

Root<Orderr> rt = cq.from(Orderr.class);

cq.select(em.getCriteriaBuilder().count(rt));

Query q = em.createQuery(cq);

return ((Long) q.getSingleResult()).intValue();

} finally {

em.close();

}

}

}

package by.bsuir.bocharenko.rms.controller;

import by.bsuir.bocharenko.rms.controller.exceptions.NonexistentEntityException;

import java.io.Serializable;

import javax.persistence.Query;

import javax.persistence.criteria.CriteriaQuery;

import javax.persistence.criteria.Root;

import by.bsuir.bocharenko.rms.entity.Material;

import by.bsuir.bocharenko.rms.entity.Storage;

import java.util.List;

import javax.persistence.EntityManager;

import javax.persistence.EntityManagerFactory;

Продолжение приложения В

/\*\*

\*

\* @author user

\*/

public class StorageJpaController implements Serializable {

public StorageJpaController(EntityManagerFactory emf) {

this.emf = emf;

}

private EntityManagerFactory emf = null;

public EntityManager getEntityManager() {

return emf.createEntityManager();

}

public void create(Storage storage) throws NonexistentEntityException{

EntityManager em = null;

try {

em = getEntityManager();

em.getTransaction().begin();

Material material = em.getReference(Material.class, storage.getMaterial().getId());

if (material == null)

throw new NonexistentEntityException("Материал с таким номером " + storage.getMaterial().getId()

+ " не зарегистрирован в БД.");

material = em.getReference(Material.class, material.getId());

storage.setMaterial(material);

em.persist(storage);

List<Storage> storages = material.getStorage();

storages.add(storage);

material.setStorage(storages);

em.merge(material);

em.getTransaction().commit();

} finally {

if (em != null) {

em.close();

}

}

}

public void edit(Storage storage) throws NonexistentEntityException, Exception {

EntityManager em = null;

if (findStorage(storage.getId()) == null)

throw new NonexistentEntityException("Материал с таким номером " + storage.getId()

+ " не зарегестрирован на складе");

try {

em = getEntityManager();

em.getTransaction().begin();

//Изменяется родительская сущность Материал

Storage persistentStorage = em.find(Storage.class, storage.getId());

Material materialOld = persistentStorage.getMaterial();

Material materialNew = em.find(Material.class, storage.getMaterial().getId());

if(materialNew==null)

Продолжение приложения В

throw new NonexistentEntityException("Материал с таким номером " + storage.getMaterial().getId()

+ " не зарегистрирован в БД.");

if (!materialNew.equals(materialOld)){

List<Storage> oldStorages = materialOld.getStorage();

oldStorages.remove(persistentStorage);

materialOld.setStorage(oldStorages);

em.merge(materialOld);

List<Storage> newStorages = materialNew.getStorage();

newStorages.add(storage);

materialNew.setStorage(newStorages);

em.merge(materialNew);

}

else {

List<Storage> oldStorages = materialOld.getStorage();

oldStorages.remove(persistentStorage);

oldStorages.add(storage);

materialOld.setStorage(oldStorages);

em.merge(materialOld);

}

storage.setMaterial(materialNew);

em.merge(storage);

em.getTransaction().commit();

} finally {

if (em != null) {

em.close();

}

}

}

public void destroy(Long id) throws NonexistentEntityException {

EntityManager em = null;

if(findStorage(id)==null)

throw new NonexistentEntityException("Материал под номером " + id

+ " не зарегестрирован на складе");

try {

em = getEntityManager();

em.getTransaction().begin();

Storage storage = em.getReference(Storage.class, id);

Material material = storage.getMaterial();

List<Storage> storages = material.getStorage();

storages.remove(storage);

material.setStorage(storages);

em.merge(material);

em.remove(storage);

em.getTransaction().commit();

} finally {

if (em != null) {

em.close();

}

}

}

public List<Storage> findStorageEntities() {

return findStorageEntities(true, -1, -1);

}

Продолжение приложения В

public List<Storage> findStorageEntities(int maxResults, int firstResult) {

return findStorageEntities(false, maxResults, firstResult);

}

private List<Storage> findStorageEntities(boolean all, int maxResults, int firstResult) {

EntityManager em = getEntityManager();

try {

CriteriaQuery cq = em.getCriteriaBuilder().createQuery();

cq.select(cq.from(Storage.class));

Query q = em.createQuery(cq);

if (!all) {

q.setMaxResults(maxResults);

q.setFirstResult(firstResult);

}

return q.getResultList();

} finally {

em.close();

}

}

public Storage findStorage(Long id) {

EntityManager em = getEntityManager();

try {

return em.find(Storage.class, id);

} finally {

em.close();

}

}

public int getStorageCount() {

EntityManager em = getEntityManager();

try {

CriteriaQuery cq = em.getCriteriaBuilder().createQuery();

Root<Storage> rt = cq.from(Storage.class);

cq.select(em.getCriteriaBuilder().count(rt));

Query q = em.createQuery(cq);

return ((Long) q.getSingleResult()).intValue();

} finally {

em.close();

}

}

}

package by.bsuir.bocharenko.rms.servlet;

import by.bsuir.bocharenko.rms.webappcommand.ICommand;

import java.io.IOException;

import java.util.logging.Level;

import java.util.logging.Logger;

import javax.servlet.RequestDispatcher;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

Продолжение приложения В

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

@WebServlet("/Servlet")

public class Servlet extends HttpServlet {

Requesthelper rh=Requesthelper.getInstance();

@Override

protected void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

String page=null;

request.setCharacterEncoding("UTF-8");

ICommand command=rh.getCommand(request);

page=command.execute(request, response);

if(page!=null){

RequestDispatcher dispatcher=getServletContext().getRequestDispatcher(page);

try{dispatcher.forward(request, response);

}catch (ServletException ex) {

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

}catch (IOException ex) {

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

}

}

}

@Override

protected void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

doGet(request, response);

}

@Override

public String getServletInfo() {

return "Short description";

}// </editor-fold>

}

package by.bsuir.bocharenko.rms.servlet;

import by.bsuir.bocharenko.rms.webappcommand.material.SearchMaterialCommand;

import java.util.HashMap;

import by.bsuir.bocharenko.rms.webappcommand.\*;

import by.bsuir.bocharenko.rms.webappcommand.date.\*;

Продолжение приложения В

import by.bsuir.bocharenko.rms.webappcommand.dbconnection.CreateConnectionCommand;

import by.bsuir.bocharenko.rms.webappcommand.go\_to.\*;

import by.bsuir.bocharenko.rms.webappcommand.order.\*;

import by.bsuir.bocharenko.rms.webappcommand.supplier.\*;

import by.bsuir.bocharenko.rms.webappcommand.material.\*;

import by.bsuir.bocharenko.rms.webappcommand.storage.\*;

import javax.servlet.http.HttpServletRequest;

public class Requesthelper {

private static Requesthelper rh;

HashMap<String,ICommand> hm=new HashMap<>();

private Requesthelper(){

hm.put("GoToCreateMaterial", new GoToCreateMaterialCommand());

hm.put("CreateMaterial",new CreateMaterialCommand());

hm.put("ReadMaterial", new ReadMaterialCommand());

hm.put("GoToUpdateMaterial", new GoToUpdateMaterialCommand());

hm.put("UpdateMaterial", new UpdateMaterialCommand());

hm.put("GoToDeleteMaterial", new GoToDeleteMaterialCommand());

hm.put("DeleteMaterial", new DeleteMaterialCommand());

hm.put("GoToCreateSupplier", new GoToCreateSupplierCommand());

hm.put("CreateSupplier",new CreateSupplierCommand());

hm.put("ReadSupplier", new ReadSupplierCommand());

hm.put("GoToUpdateSupplier", new GoToUpdateSupplierCommand());

hm.put("UpdateSupplier", new UpdateSupplierCommand());

hm.put("GoToDeleteSupplier", new GoToDeleteSupplierCommand());

hm.put("DeleteSupplier", new DeleteSupplierCommand());

hm.put("GoToCreateOrder", new GoToCreateOrderCommand());

hm.put("CreateOrder",new CreateOrderCommand());

hm.put("ReadOrder", new ReadOrderCommand());

hm.put("GoToUpdateOrder", new GoToUpdateOrderCommand());

hm.put("UpdateOrder", new UpdateOrderCommand());

hm.put("GoToDeleteOrder", new GoToDeleteOrderCommand());

hm.put("DeleteOrder", new DeleteOrderCommand());

hm.put("ReadStorage", new ReadStorageCommand());

hm.put("ChangeDate", new ChangeDateCommand());

hm.put("GoToMain", new GoToMainCommand());

hm.put("GoToFinance", new GoToFinanceCommand());

hm.put("GoToStatistics", new GoToStatisticsCommand());

hm.put("GoToSearch", new GoToSearchCommand());

hm.put("GoToSelection", new GoToSelectionCommand());

hm.put("SearchMaterial", new SearchMaterialCommand());

hm.put("SearchSupplier", new SearchSupplierCommand());

hm.put("SearchOrder", new SearchOrderCommand());

hm.put("SearchStorage", new SearchStorageCommand());

hm.put("CreateConnection", new CreateConnectionCommand());

hm.put("NoCommand", new NoCommand());

}

public static Requesthelper getInstance(){

if(rh==null)

Продолжение приложения В

rh=new Requesthelper();

return rh;

}

public ICommand getCommand (HttpServletRequest request){

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

ICommand command=hm.get(action);

if(command==null)

command=new NoCommand();

return command;

}

}

package by.bsuir.bocharenko.rms.webappcommand;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

public interface ICommand {

public String execute (HttpServletRequest request, HttpServletResponse response);

}

package by.bsuir.bocharenko.rms.webappcommand.material;

import by.bsuir.bocharenko.rms.controller.EMFactory;

import by.bsuir.bocharenko.rms.controller.MaterialJpaController;

import by.bsuir.bocharenko.rms.entity.Material;

import by.bsuir.bocharenko.rms.webappcommand.ICommand;

import by.bsuir.bocharenko.rms.webappcommand.date.DateParser;

import java.text.ParseException;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

/\*\*

\*

\* @author user

\*/

public class CreateMaterialCommand implements ICommand{

@Override

public String execute(HttpServletRequest request, HttpServletResponse response){

String w = request.getParameter("weight");

String v = request.getParameter("volume");

String u = request.getParameter("unitStorageCost");

if(w.equals("") || v.equals("") || u.equals("")){

Продолжение приложения В

request.setAttribute("errorMessage", "Неверный ввод формы");

return "jsp/error.jsp";

}

Integer unitStorageCost = new Integer(u);

Float weight = new Float(w);

Float volume = new Float(v);

Material material = new Material();

material.setTitle(request.getParameter("title"));

material.setSpecification(request.getParameter("specification"));

material.setMeasure(request.getParameter("measure"));

material.setWeight(weight);

material.setVolume(volume);

material.setUnitStorageCost(unitStorageCost);

MaterialJpaController dbconn = new MaterialJpaController(EMFactory.getInstance().getEntityManagerFactory());

dbconn.create(material);

try {

request.setAttribute("date", DateParser.parseStandartDate(request, "dateForvarded"));

} catch (ParseException ex) {

request.setAttribute("errorMessage", "Неверный ввод даты");

return "/jsp/error.jsp";

}

return "/jsp/main.jsp";

}

}

package by.bsuir.bocharenko.rms.webappcommand.material;

import by.bsuir.bocharenko.rms.controller.EMFactory;

import by.bsuir.bocharenko.rms.controller.MaterialJpaController;

import by.bsuir.bocharenko.rms.controller.exceptions.NonexistentEntityException;

import by.bsuir.bocharenko.rms.webappcommand.ICommand;

import by.bsuir.bocharenko.rms.webappcommand.date.DateParser;

import java.text.ParseException;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

/\*\*

\*

\* @author user

\*/

public class DeleteMaterialCommand implements ICommand{

@Override

public String execute (HttpServletRequest request, HttpServletResponse response){

MaterialJpaController dbconn = new MaterialJpaController(EMFactory.getInstance().getEntityManagerFactory());

Продолжение приложения В

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

if(id.equals("")){

request.setAttribute("errorMessage", "Неверный ввод номера");

return "/jsp/error.jsp";

}

Long n = new Long(id);

try {

dbconn.destroy(n);

} catch (NonexistentEntityException ex) {

request.setAttribute("errorMessage", ex.getMessage());

return "/jsp/error.jsp";

}

try {

request.setAttribute("date", DateParser.parseStandartDate(request, "dateForvarded"));

} catch (ParseException ex) {

request.setAttribute("errorMessage", "Неверный ввод даты");

return "/jsp/error.jsp";

}

return "/jsp/main.jsp";

}

}

package by.bsuir.bocharenko.rms.webappcommand.material;

import by.bsuir.bocharenko.rms.controller.EMFactory;

import by.bsuir.bocharenko.rms.controller.MaterialJpaController;

import by.bsuir.bocharenko.rms.entity.Material;

import by.bsuir.bocharenko.rms.webappcommand.ICommand;

import by.bsuir.bocharenko.rms.webappcommand.date.DateParser;

import java.text.ParseException;

import java.util.List;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

/\*\*

\*

\* @author user

\*/

public class ReadMaterialCommand implements ICommand{

@Override

public String execute (HttpServletRequest request, HttpServletResponse response){

MaterialJpaController dbconn = new MaterialJpaController(EMFactory.getInstance().getEntityManagerFactory());

List<Material> list = dbconn.findMaterialEntities();

request.setAttribute("List", list);

try {

Продолжение приложения В

request.setAttribute("date", DateParser.parseStandartDate(request, "dateForvarded"));

} catch (ParseException ex) {

request.setAttribute("errorMessage", "Неверный ввод даты");

return "/jsp/error.jsp";

}

return "/jsp/material/ReadMaterial.jsp";

}

}

package by.bsuir.bocharenko.rms.webappcommand.material;

import by.bsuir.bocharenko.rms.controller.EMFactory;

import by.bsuir.bocharenko.rms.controller.MaterialJpaController;

import by.bsuir.bocharenko.rms.entity.Material;

import by.bsuir.bocharenko.rms.webappcommand.ICommand;

import by.bsuir.bocharenko.rms.webappcommand.date.DateParser;

import java.text.ParseException;

import java.util.ArrayList;

import java.util.List;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

/\*\*

\*

\* @author user

\*/

public class SearchMaterialCommand implements ICommand{

@Override

public String execute (HttpServletRequest request, HttpServletResponse response){

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

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

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

Integer minUnitStorageCost = new Integer(request.getParameter("minUnitStorageCost"));

Integer maxUnitStorageCost = new Integer(request.getParameter("maxUnitStorageCost"));

Float minWeight = new Float(request.getParameter("minWeight"));

Float maxWeight = new Float(request.getParameter("maxWeight"));

Float minVolume = new Float(request.getParameter("minVolume"));

Float maxVolume = new Float(request.getParameter("maxVolume"));

MaterialJpaController dbconn = new MaterialJpaController(EMFactory.getInstance().getEntityManagerFactory());

List<Material> list = dbconn.findMaterialEntities();

List<Material> modifiedList = new ArrayList();

System.out.println(list.size());

if(!title.equals("")){

for(Material material : list){

if(material.getTitle().equals(title))

modifiedList.add(material);

Продолжение приложения В

}

list.clear();

list.addAll(modifiedList);

modifiedList.clear();

}

System.out.println(list.size());

if(!specification.equals("")){

for(Material material : list){

if(material.getSpecification().equals(specification))

modifiedList.add(material);

}

list.clear();

list.addAll(modifiedList);

modifiedList.clear();

}

System.out.println(list.size());

if(!measure.equals("")){

for(Material material : list){

if(material.getMeasure().equals(measure))

modifiedList.add(material);

}

list.clear();

list.addAll(modifiedList);

modifiedList.clear();

}

System.out.println(list.size());

for(Material material : list){

if(material.getWeight() >= minWeight && material.getWeight() <= maxWeight

&& material.getVolume() >= minVolume && material.getVolume() <= maxVolume

&& material.getUnitStorageCost() >= minUnitStorageCost && material.getUnitStorageCost() <= maxUnitStorageCost)

modifiedList.add(material);

}

list.clear();

list.addAll(modifiedList);

modifiedList.clear();

System.out.println(list.size());

request.setAttribute("List", list);

try {

request.setAttribute("date", DateParser.parseStandartDate(request, "dateForvarded"));

} catch (ParseException ex) {

request.setAttribute("errorMessage", "Неверный ввод даты");

return "/jsp/error.jsp";

}

return "/jsp/material/ReadMaterial.jsp";

}

}

Продолжение приложения В

package by.bsuir.bocharenko.rms.webappcommand.material;

import by.bsuir.bocharenko.rms.controller.EMFactory;

import by.bsuir.bocharenko.rms.controller.MaterialJpaController;

import by.bsuir.bocharenko.rms.controller.exceptions.NonexistentEntityException;

import by.bsuir.bocharenko.rms.entity.Material;

import by.bsuir.bocharenko.rms.webappcommand.ICommand;

import by.bsuir.bocharenko.rms.webappcommand.date.DateParser;

import java.text.ParseException;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

/\*\*

\*

\* @author user

\*/

public class UpdateMaterialCommand implements ICommand{

@Override

public String execute (HttpServletRequest request, HttpServletResponse response){

String n = request.getParameter("id");

if(n.equals("")){

request.setAttribute("errorMessage","Неверный ввод номера материала");

return "/jsp/error.jsp";

}

String w = request.getParameter("weight");

String v = request.getParameter("volume");

String u = request.getParameter("unitStorageCost");

if(w.equals("") || v.equals("") || u.equals("")){

request.setAttribute("errorMessage", "Неверный ввод формы");

return "jsp/error.jsp";

}

Float weight = new Float(w);

Float volume = new Float(v);

Integer unitStorageCost = new Integer(u);

Long id = new Long(n);

Material material = new Material();

material.setId(id);

material.setTitle(request.getParameter("title"));

material.setSpecification(request.getParameter("specification"));

material.setMeasure(request.getParameter("measure"));

material.setWeight(weight);

material.setVolume(volume);

material.setUnitStorageCost(unitStorageCost);

MaterialJpaController dbconn = new MaterialJpaController(EMFactory.getInstance().getEntityManagerFactory());

try {

dbconn.edit(material);

} catch (NonexistentEntityException ex) {

request.setAttribute("errorMessage", ex.getMessage());

return "jsp/error.jsp";

}

Продолжение приложения В

try {

request.setAttribute("date", DateParser.parseStandartDate(request, "dateForvarded"));

} catch (ParseException ex) {

request.setAttribute("errorMessage", "Неверный ввод даты");

return "/jsp/error.jsp";

}

return "/jsp/main.jsp";

}

}

ПРИЛОЖЕНИЕ Г

(обязательное)

Листинг скрипта генерации базы данных

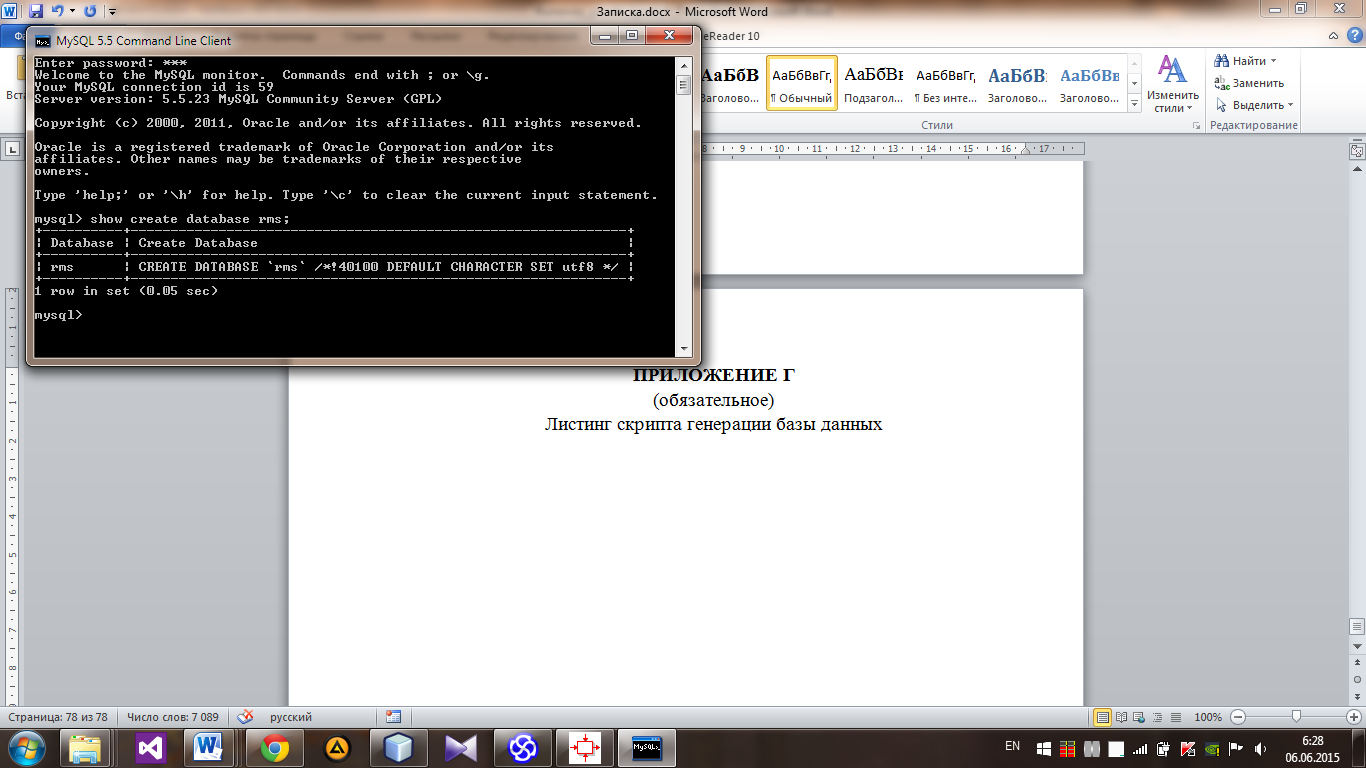


Рисунок Г.1 – Команда создания базы данных

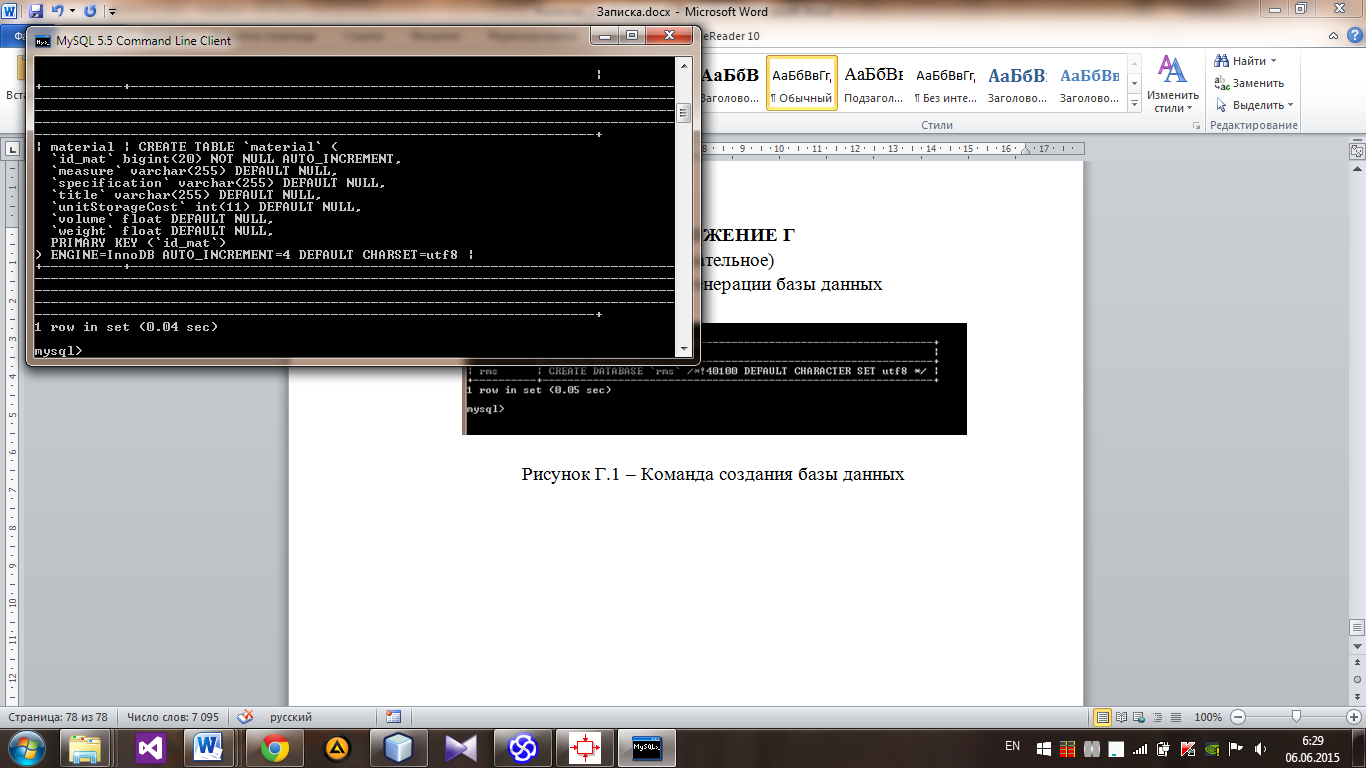


Рисунок Г.2 – Команда создания таблицы Material

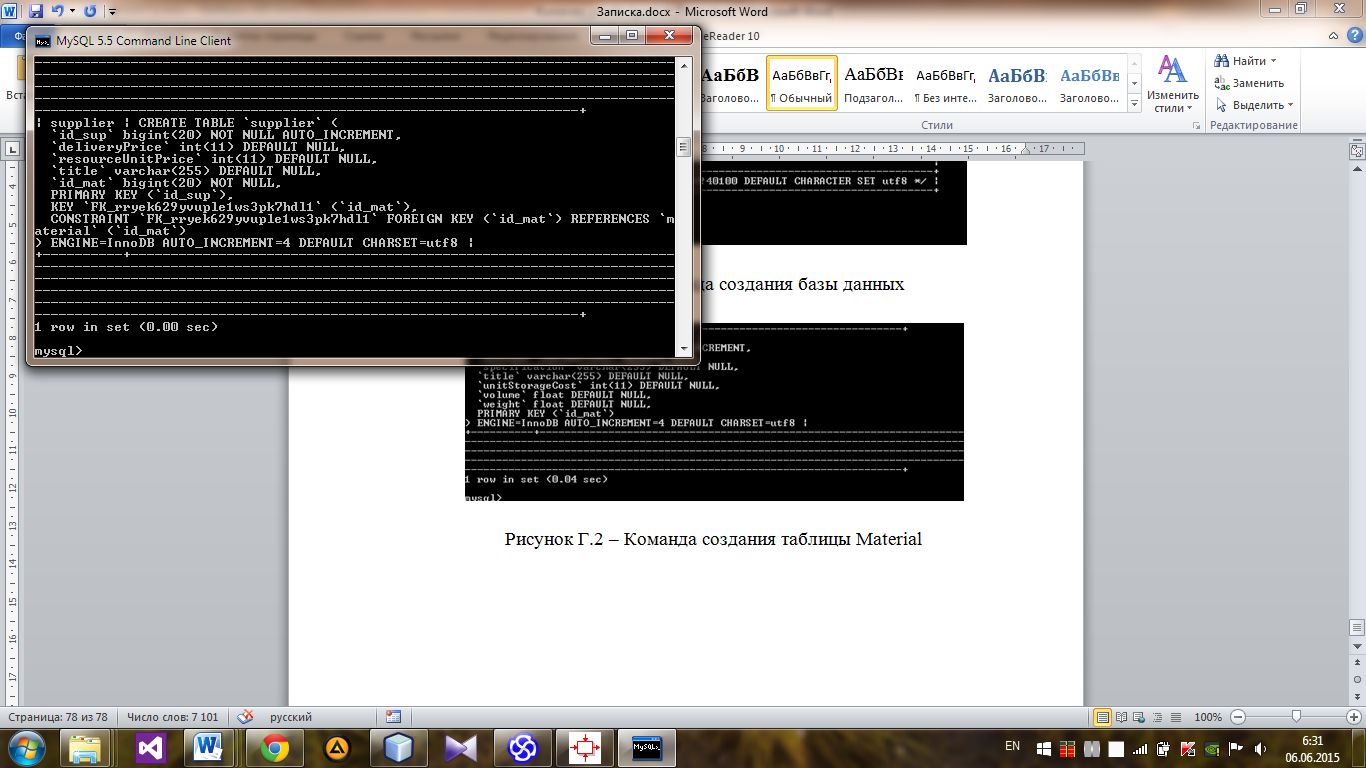


Рисунок Г.3 – Команда создания таблицы Supplier

Продолжение приложения Г

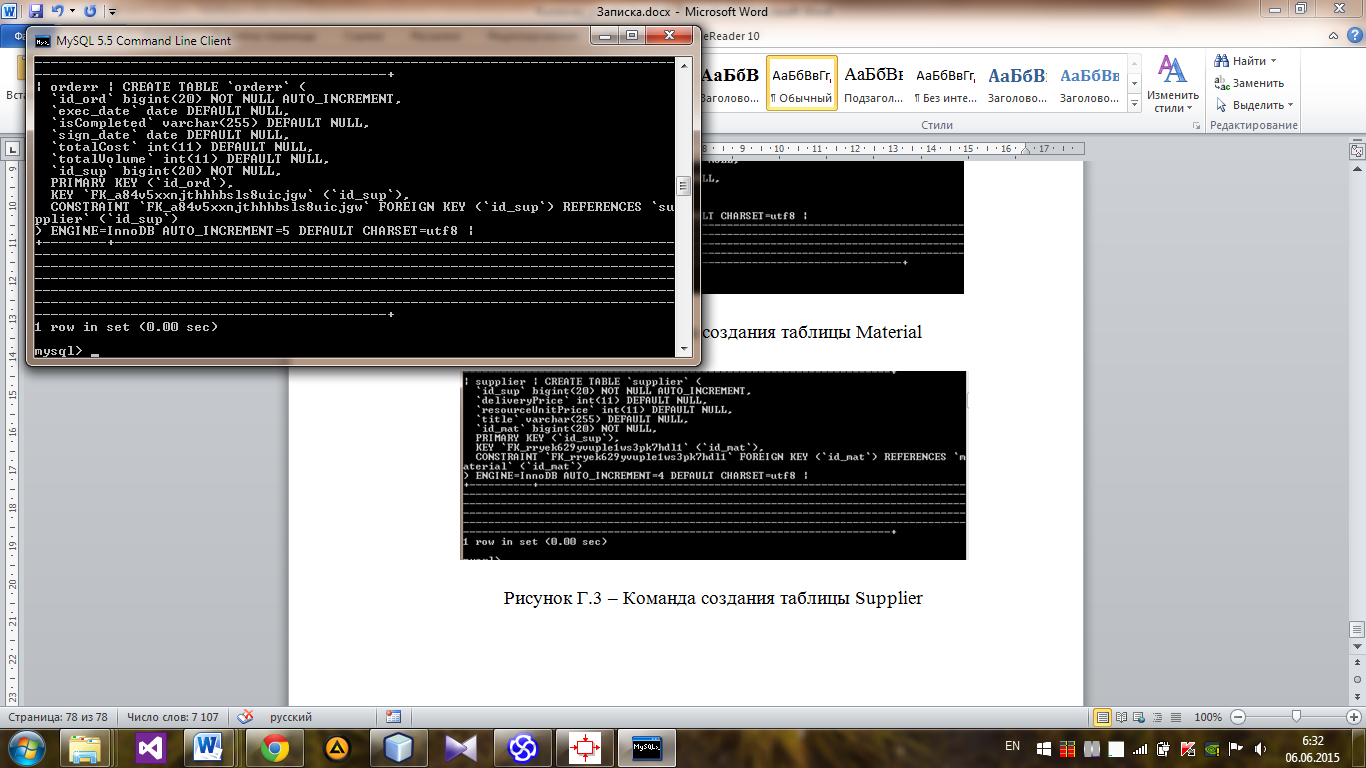


Рисунок Г.4 – Команда создания таблицы Orderr

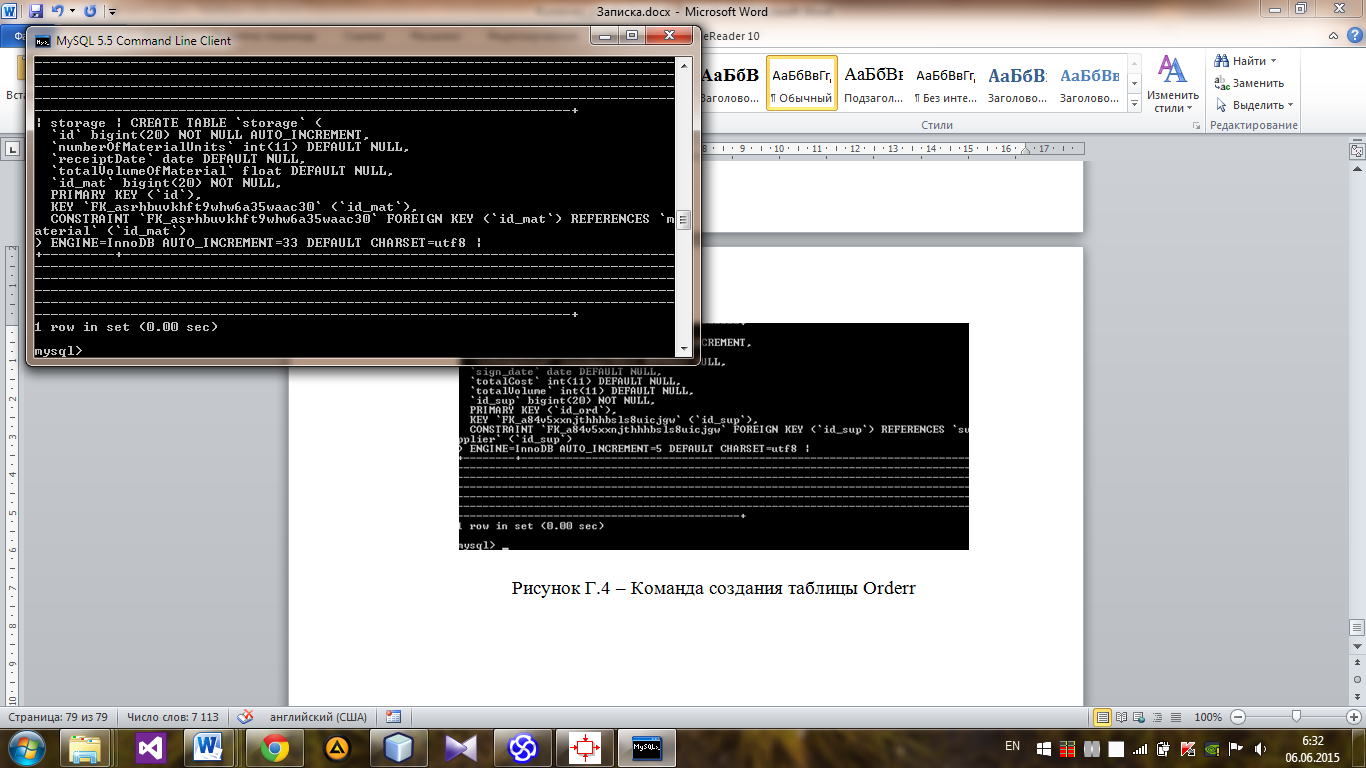


Рисунок Г.4 – Команда создания таблицы Storage