Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет инженерно-экономический
Кафедра экономической информатики
Дисциплина «Программирование сетевых приложений»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовой работе
на тему

РАЗРАБОТКА АВТОМАТИЗИРОВАННОЙ СИСТЕМЫ РАСЧЕТА ПРИБЫЛИ ОТ ПРОДАЖ В МЕБЕЛЬНОМ САЛОНЕ

БГУИР КП 1-40 05 01-10 ПЗ

Студент Ю. Э. Фишкина
группа 894351

Руководитель Д. А. Сторожев
старший преподаватель
кафедры ЭИ

Минск, 2020

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

На современном уровне развития автоматизация процессов представляет собой один из подходов к управлению процессами на основе применения информационных технологий. Этот подход позволяет осуществлять управление операциями, данными, информацией и ресурсами за счет использования компьютеров и программного обеспечения, которые сокращают степень участия человека в процессе, либо полностью его исключают.

Каждый предприниматель, начиная свою деятельность, должен ясно представлять потребность на будущее в финансовых, материальных, трудовых и интеллектуальных ресурсах, источники их получения, а также уметь четко рассчитывать эффективность их использования в процессе работы фирмы.

В рыночной экономике предприниматели не смогут добиться стабильного успеха, если не будут четко и эффективно планировать свою деятельность, постоянно собирать и аккумулировать информацию, как о состоянии целевых рынков, положении на них конкурентов, так и собственных перспективах, и возможностях. При всем многообразии форм предпринимательства существуют ключевые положения, применимые практически во всех областях коммерческой деятельности для различных фирм. Необходимые для того, чтобы своевременно подготовиться и обойти потенциальные трудности и опасности, тем самым уменьшив риск в достижении поставленных целей.

Данные процессы возможно автоматизировать для повышения качества исполнения процесса. Также, автоматизированный процесс обладает более стабильными характеристиками, чем процесс, выполняемый в ручном режиме. Положительной стороной автоматизации процессов является:

- повышение производительности,
- сокращение времени выполнения процесса,
- снижение стоимости,
- увеличение точности и стабильности выполняемых операций.

Предметом исследования курсовой работы является процесс системы расчета прибыли, регистрации товаров и пользователей.

Объектом исследования является мебельный салон.

Целью курсовой работы является повышение эффективности процесса расчета прибыли и убытков мебельного салона и сокращение затрат (временных, трудовых, финансовых) на проведение операций, которые сопровождают данные процессы.

Для реализации проекта необходимо:

– изучить ситуацию на рынке продажи мебели;

– изучить основные методики расчета прибыли;

– изучить, на основе каких данных производится расчет прибыли;

– разработать функциональную модель основного процесса;

– разработать базу данных;

– проанализировать логическую и физическую модель представления данных;

– реализовать клиент-серверное взаимодействие, позволяющее управлять базой данных и выполнять определённый ряд функций, поставленных целью разработки курсового проекта;

– разработать приложение с удобным и понятным пользовательским интерфейсом;

– предложить варианты улучшения работы и повышения эффективности программного продукта.

Программные продукты автоматизации складского учета направлены на оптимизацию бизнес-процессов анализа и расчета прибыли, а также на получение существенного положительного экономического эффекта.

1 ОПИСАНИЕ СИСТЕМЫ РАСЧЕТА ПРИБЫЛИ ОТ ПРОДАЖ В МЕБЕЛЬНОМ САЛОНЕ

1.1 Основные понятия и бизнес-процессы при подсчете прибыли в мебельном салоне

Мебельный салон «Дом мебели» существует на рынке с 1975 года, базирующимся в Минске. Полное наименование компании – Белорусское республиканское оптово-розничное объединение «Дом мебели». Основным видом деятельности является торговля мягкой мебелью с белорусских фабрик в розницу в специализированных магазинах. Форма собственности – частная. Стратегические задачи компании:

– увеличение прибыли и эффективности;

– привлечение новых клиентов;

– территориальное расширение деятельности.

Мебельный салон имеет небольшой штат сотрудников, некоторые из которых выполняют довольно большой функционал и несут усиленную ответственность за правильное и своевременное исполнение заказов. Компания действует на основании устава. На предприятии ведется финансовая и экономическая отчетность. Компания предоставляет услуги розничной торговли мебели через принадлежащие ей помещения сбыта продукции. Местоположения мебельного салона: г. Минск.

Форма управления – иерархическая.

Принципы управления:

– Четкое разделение труда.

– Процесс обслуживания клиентов и заказов происходит непосредственно благодаря менеджерам.

Иерархия управления представлена на рисунке 1.1:

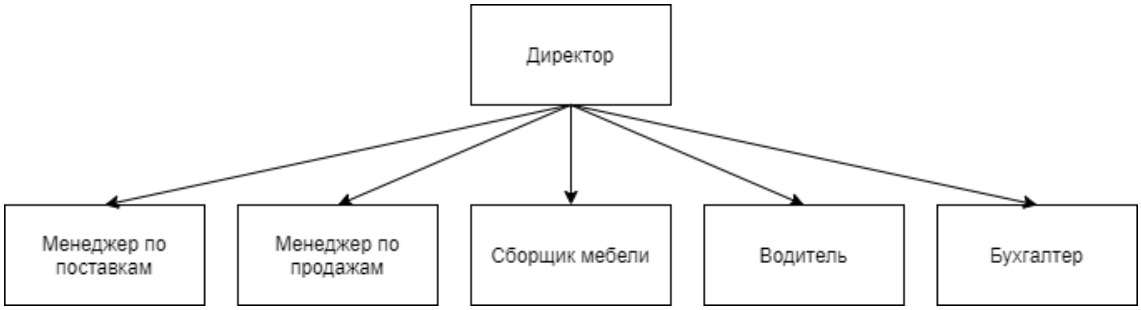

Рисунок 1.1 - организационно-штатная структура мебельного салона

На рисунке 1.1 представлена организационно-штатная структура мебельного магазина, составляющие которой являются:

– директор – является основной фигурой на предприятии. От директора зависит продвижение и развитие данного предприятия. В обязанности директора входят эффективность и контроль работы персонала;

– бухгалтер - является одной из основных обязательных штатных структур в любом бизнесе. Работа специалистов заключается в расчете окладов и начисления заработной платы сотрудникам. Так же они занимаются расчетом налоговых отчислений и себестоимости продукции, проводят счета от поставщиков и субподрядчиков;

– водитель - должен доставлять товар и документы вовремя, в целостности и сохранности;

– сборщик мебели – является лицом компании, работа дома у клиента и работа в салонах сети;

– менеджер по продажам – занимается консультацией клиентов, расчетами заказов, оказывает помощь в подборе дизайна интерьера;

– менеджер по поставкам – человек, который должен управлять, координировать и контролировать процессы поставки мебели на предприятии.

В первую очередь при изучении основного производства нужно разобраться в процессе изготовления продукции, т. е. изучить все стадии от приобретения сырья до продажи товара покупателю (часто первой ступенью считают не приобретение сырья и материалов, а получение заявки от заказчика на приобретение продукции или выполнения работ (услуг)). Основное производство — это фундамент функционирования предприятия, основа его денежных потоков. На рисунке 1.2 представлены основной процесс мебельной компании в виде схемы:

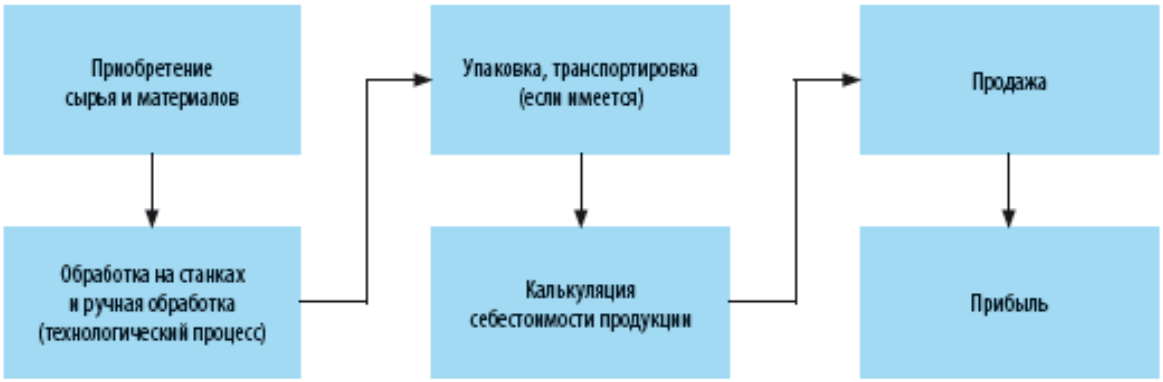

Рисунок 1.2 – Процесс работы мебельного салона

Из рисунка 1.2 видно, что для выполнения основного производства как бизнес-процесса используется совокупность разнообразных ресурсов: кадровых, материальных, технических, финансовых и др. Результат бизнес-процесса мебельного салона — это прибыль за счет реализации продукции (товара, услуги).

Спрос на мебель среди минского потребителя большой. Данные Яндекс.Вордстат показывают, что минчане пользуются информационными запросами о покупке мебели весьма часто. Практические все представленные запросы имеют статистику поиска пользователями выше 10000 раз в месяц, что делает данный бизнес рентабельным:

Поисковый запрос	Раз в месяц
Купить кресло	10 825
Купить пуфик	571
Купить кровать	16789
Купить диван	17800
Купить кушетку	693

Таблица 1.1 – статистика поисковых запросов Яндекс.Вордстат

Остановимся на процессе калькуляции цены мягкой мебели в мебельном салоне. В процессе продажи мягкой мебели фирма имеет тесный контакт с фабрикой, производящей мягкую мебель в другом городе. Менеджер ведет переговоры с менеджером фабрики и договаривается о цене и количестве закупаемой мягкой мебели. Данные о цене товара передаются бухгалтеру, который(-ая) владеет информацией о заработной плате каждого сотрудника, о выплате страховых взносов и об общих расходах компании. Эти данные суммируются и таким образом получается себестоимость товара. Далее ведутся переговоры с директором, который назначает наценку на товар. После чего товар попадает в каталог и выставляется на продажу. Схема действия приведена ниже (рисунок 1.3).

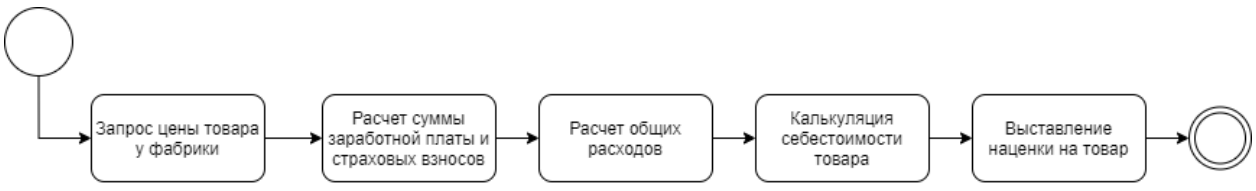

Рисунок 1.3 – Переговоры с фабрикой и выставления цены за товар мебельным салоном

Готовую продукцию реализуют в мебельном салоне, в результате чего предприятие получает выручку от реализации. В конце квартала бухгалтер предоставляет директору финансовые документы. На рисунке 1.4 отчет о прибыльности компании представлен подробно:

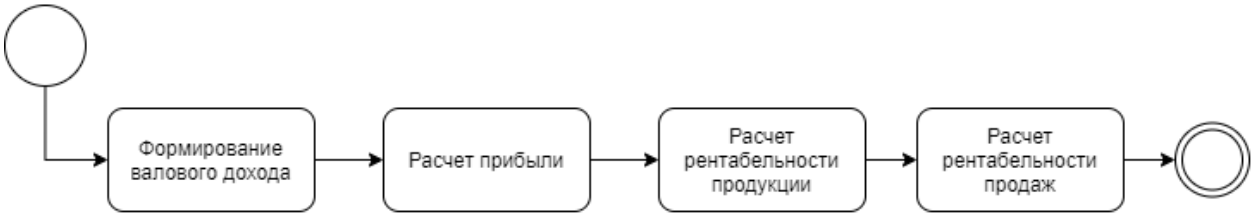

Рисунок 1.4 - Расчет прибыльности мебельного салона

Вначале формируется валовый доход, как суммирование реализованной продукции втечении квартала. Также суммируется себестоимость всех реализованных товаров. Для расчета прибыли, рентабельности продукции и продаж используется формулы:

Прибыль = Валовый доход – Себестоимость

Чтобы оценить интенсивность и эффективность производства, рассчитывают показатели рентабельности.

Рентабельность продукции (ROM) — это отношение прибыли к полной себестоимости. Показывает, сколько получено прибыли на 1 руб. текущих затрат:

ROM = (Прибыль / Себестоимость) × 100 %.

Рентабельность продаж (NPM) — отношение прибыли к выручке. Этот показатель характеризует степень прибыльности работы предприятия и правильность установления цены продажи, отражая долю прибыли в выручке от продажи. Рентабельность продаж показывает, сколько прибыли имеет предприятие с 1 руб. продажи:

NPM = (Прибыль / Валовой доход) × 100 %

1.2 Причины перехода к автоматизированной системе расчета прибыли в компании

Основной целью автоматизации бухгалтерского учета, а именно расчет прибыли, является повышение его качества и оперативности, обеспечение дальнейшего развития и совершенствование. Актуальность внедрения компьютерной техники определяется следующими причинами:

– ростом объемов бухгалтерской информации, ее детализацией;

– повышением требований к скорости и своевременности обработки информации;

– обеспечение точности и достоверности бухгалтерской информации.

Внедрение ЭВМ в обработку учетной информации позволяет сократить сроки выполнения учетных работ, усовершенствовать документацию и документооборот, создать основу безбумажной технологии учета, изменить характер и условия труда учетных работников, повысить его производительность, устранить рутинную, техническую работу, сделать труд бухгалтера более производительным и творческим.

2 ПОСТАНОВКА ЗАДАЧИ ПО АВТОМАТИЗИРОВАНИЮ СИСТЕМЫ РАСЧЕТА ПРИБЫЛИ ОТ ПРОДАЖ В МЕБЕЛЬНОМ САЛОНЕ И ОБЗОР МЕТОДОВ ЕЁ РЕШЕНИЯ

2.1 Постановка задачи расчета прибыли от продаж в мебельном салоне

В результате разработки данного курсового проекта должно быть создано клиент-серверное приложение с удобным интерфейсом и представляющее собой автоматизированную систему расчета прибыли от продаж в мебельном салоне.

В разрабатываемой системе должен быть реализован механизм авторизации пользователей (вход от имени менеджера, заказчика и работника склада), а также предоставление пользователю аналитической информации в виде графиков и диаграмм.

Пользователи, работающие с данной системой, будут двух типов:

– администратор;

– обычный пользователь;

В приложении требуется реализовать:

– функции администратора: управление базой данных пользователей (добавление новых пользователей, редактирование и просмотр имеющихся пользователей). Следует предоставить возможность изменить статус допуска пользователя к автоматизированной системе. Более того, администратор обладает теми же правами доступа, что и пользователь к товарам и анализу прибыли, которые будут описаны ниже.

– функции пользователя: просмотр списка товаров мебельного салоне, (изменение, удаление товаров, добавление новых товаров), возможность сформировать валовую прибыль за месяц путем выбора нужного месяца, а также товаров, проданных за месяц мебельным салоном. Расчет чистой прибыли при вводе операционных расходов и формирование графика динамики чистой прибыли по месяцам;

Диаграмма вариантов использования позволяет провести анализ использования системы, то есть провести описание ее основного предназначения.

Диаграмма вариантов использования разрабатываемой системы представлена на рисунке 2.1.

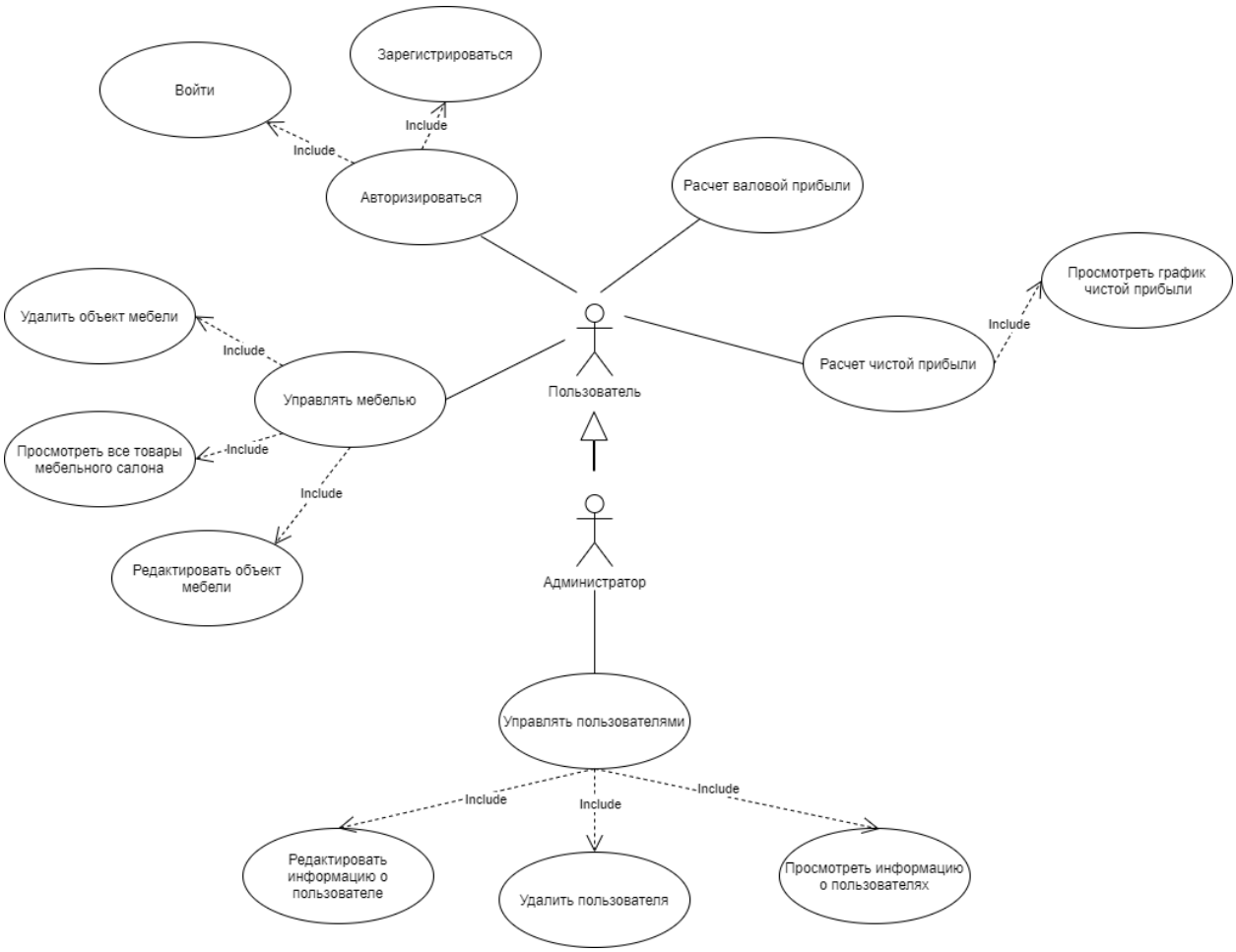

Рисунок 2.1 – Диаграмма вариантов использования.

Требуется реализовать механизм обработки исключительных ситуаций в случае некорректного использования программного продукта, а также при вводе некорректной информации.

Для решения поставленных задач необходимо разработать WEB-приложение с организацией взаимодействия с базой данных (MySQL) на объектно-ориентированном языке Java.

2.2 Технологии, использованные для решения поставленной задачи

Данная система была разработана на языке программирования Java. Главным преимуществом этого языка является то, что Java-приложения являются независимыми от платформы, как на уровне исходного кода, так и на двоичном уровне, следовательно, их можно запускать в различных системах.

Работа приложения основывается на том, что клиентская часть реализует пользовательский интерфейс, формирует запросы к серверу и обрабатывает ответы от него, в то время как серверная часть получает запрос от клиента, выполняет вычисления, после чего формирует ответ и отправляет её клиенту. Выполнение всех описанных сетевых функций приложения без проблем реализуется с помощью Java, так как сетевая работа является одной из сильных сторон данного языка программирования.

Java обеспечивает встроенную поддержку многопоточного программирования, что использовано в приложении при реализации архитектуры «клиент-сервер»[8].

В данном курсовом проекте для хранения информация используется база данных. Базы данных сами заботятся о безопасности информации и её сортировке и позволяют извлекать и размещать информацию при помощи одной строчки. Код с использованием базы данных получается более компактным, и отлаживать его гораздо легче.

Для реализации хранения, обработки и дальнейшего использования информации в данном приложении используется система управления базами данных (СУБД) MySQL. СУБД используются для упорядоченного хранения и обработки больших объемов информации:

- просматривать;
- пополнять;
- изменять;
- удалять
- искать нужные сведения;
- делать любые выборки.

MySQL − компактный многопоточный сервер баз данных. MySQL отличатся хорошей скоростью работы, надежностью, гибкостью. Работа с ней, как правило, не вызывает больших трудностей.

Для общения с СУБД MySQL применяется язык SQL (Stuctured Query Language − язык структурированных запросов). В настоящее время SQL является стандартом работы с базами данных, и все основные СУБД понимают его. SQL включает много разных типов операторов, разработанных для взаимодействия с базами данных[9].

Информационная система должна быть реализована в виде web-приложения на языке Java с использованием технологий Servlet 3, JSP 2.2, EL 2.2, JSTL 1.2, XML/XSLT, HTML 5, CSS 3, ECMAScript 6. Архитектура приложения должна быть выполнена в архитектуре паттерна MVC. Интерфейс приложения должен быть оформлен с использованием каскадных таблиц стилей(CSS). В приложении должна быть предусмотрена проверка данных, вводимых пользователем (на клиентской и серверной части проекта). В приложении должны быть разработаны и использованы собственные библиотеки тэгов. При разработке приложения должен быть использован механизм обработки исключительных ситуаций. При разработке пользовательского интерфейса необходимо предусмотреть элемент «меню», пункты которого должны храниться в виде XML документа, который будет трансформироваться в html представление с помощью технологии XSLT.

Каркас приложения должен быть выполнен в архитектуре MVC. В приложении должны быть чётко определены уровни: модели данных, представления и контроллер. Модель данных должна быть представлена в виде набора классов, соответствующих правилам построения компонентов JavaBeans. В качестве контроллера должен выступать сервлет, который будет осуществлять взаимодействие между моделью и представлением.

Представление данных должно быть реализовано в виде набора JSP страниц. В приложении должны быть выделен слой доступа к данным (классы, в которых будут размещены методы, отвечающие за бизнес-логику системы). Доступ к данным. Доступ к данным должен быть выделен в отдельный слой приложения. Доступ к данным в СУБД должен осуществляться через драйвер JDBC, поставляемый производителем СУБД. Использование интерфейса ODBC запрещено. База данных должна быть приведена к 3-ей нормальной форме. Взаимодействие между серверной и клиентскими частями должно осуществляться с использованием сокетов и протокола HTTP.

Разработанная система должна обладать следующей инфраструктурой: - Исполняемые файлы должны работать в среде 32x разрядной ОС Windows 7 и выше - СУБД (на выбор) – Sybase SQL 11.0+, MS SQL Server 2008 R2+, MySQL 5.5+, PostgreSQL 9.0+, Java DB 10.x+ JDK 7/ JRE7. Сервлет-контейнер: Tomcat 7.0.x. База данных должна генерироваться sql-скриптом. Интерфейс программы и данные должны быть только на русском языке. Приложение должно запускаться без использования интегрированных средств разработки.

2.3. Паттерн MVC

MVC — это набор архитектурных идей и принципов для построения сложных систем с пользовательским интерфейсом.

Model. Первая компонента/модуль — так называемая модель. Она содержит всю бизнес-логику приложения.

View. Вторая часть системы — вид. Данный модуль отвечает за отображение данных пользователю. Все, что видит пользователь, генерируется видом.

Controller. Третьим звеном данной цепи является контроллер. В нем хранится код, который отвечает за обработку действий пользователя (любое действие пользователя в системе обрабатывается в контроллере).

На рисунке 2.2 представлена схема взаимодействия между модулями в паттерне MVC.

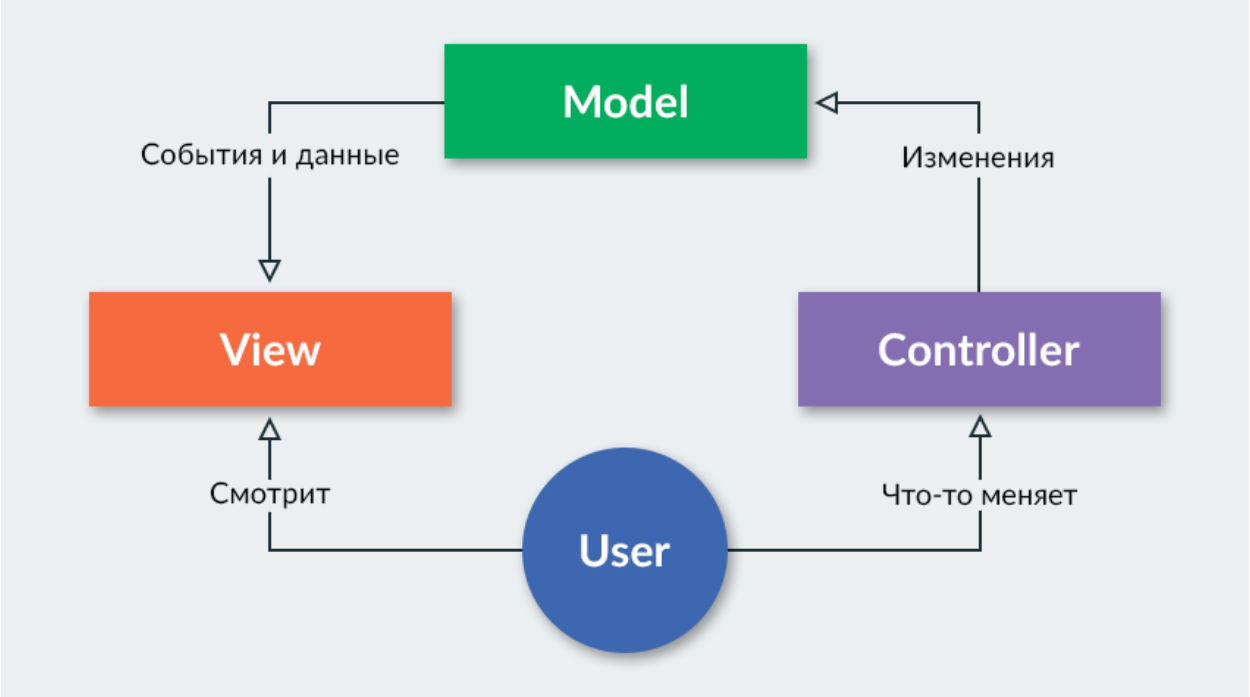

Рисунок 2.2 – Схема взаимодействия модулей MVC

3 ОПИСАНИЕ ПРОЦЕССА РАСЧЕТА ПРИБЫЛИ ДЛЯ МЕБЕЛЬНОГО САЛОНА

3.1. Функциональное моделирование на основе стандарта IDEF0

Основной задачей, решаемой разрабатываемой системой, является расчет прибыли от продаж в мебельном салоне на основе введенных показателей.

На разных этапах пользователь будет взаимодействовать с программным обеспечением и вводить данные, чтобы войти в систему, добавить данные о товарах или ввести операционные расходы. Данные будут сохраняться в определенную базу данных (пользователей/товаров и показателей) и при необходимости нужные данные будут извлекаться для отображения на экране или для расчета прибыли. Итоговой работой данного бизнес-процесса является формирование отчета для анализа прибыли и убытков мебельного салона. Описание бизнес–процесса расчета представлено на рисунке 3.1.

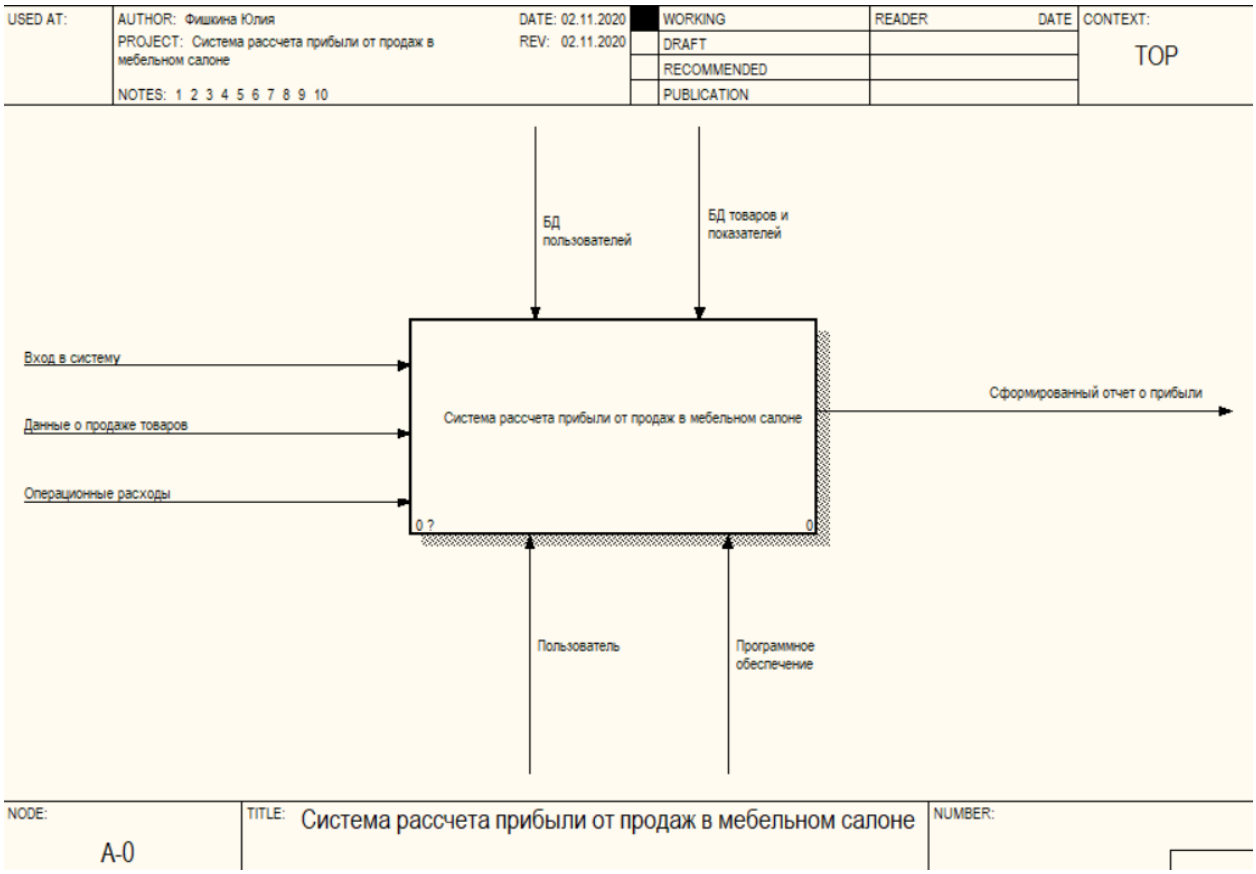

Рисунок 3.1 – Бизнес-процесс расчёта прибыли от продаж.

На рисунке 3.2 представлена декомпозиция бизнес-процесса расчёта прибыли от продаж в мебельном салоне. Она описывает разделение расчета прибыли от продаж, которое состоит из следующих этапов:

– Авторизацию/Регистрация.
– Расчет маржинальной прибыли.
– Формирование отчета о прибыли и убытках.

Стрелка управления блока «Авторизации или регистрации» является база данных пользователей, а управление блоками «Расчета маржинальной прибыли» и «формирование отчета о прибыли и убытках» является база данных товаров и показателей. Для расчета прибыли используются механизмы «Пользователь» и «Программное обеспечение» в каждом блоке бизнес-процесса. На входе вводятся данные для входа в систему на этапе авторизации, далее при необходимости вводятся данные о проданных товарах за месяц в блоке расчета маржинальной (валовой) прибыли. Для подсчета чистой прибыли на этапе формирование отчета о прибыли и убытках необходимо пользователю ввести операционных расходы.

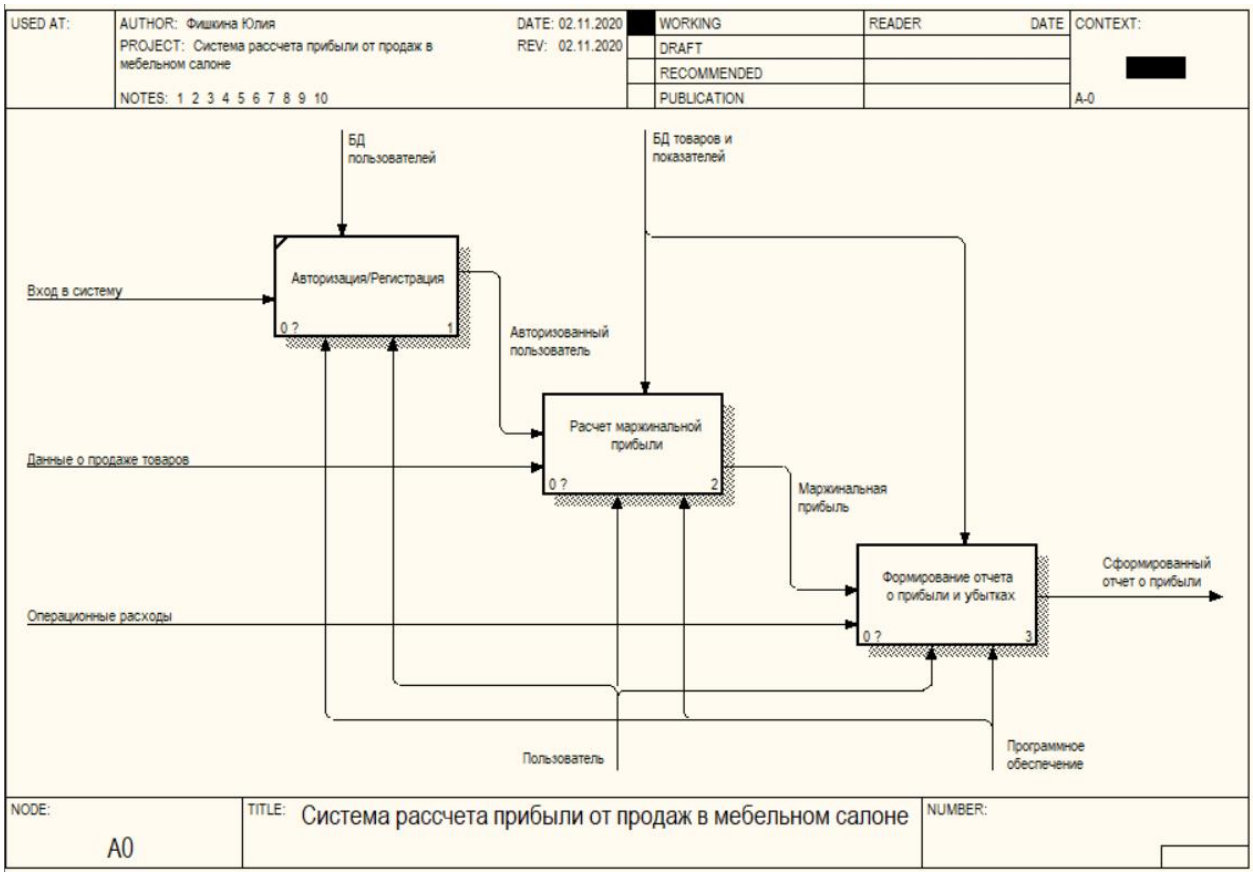

Рисунок 3.2 – Декомпозиция диаграммы верхнего уровня.

На рисунке 3.3 представлена декомпозиция процесса A2 «Расчет маржинальной прибыли». Данная диаграмма состоит из следующих процессов:

– Ввод имени товара.
– Ввод цены товара.
– Ввод себестоимости.
– Ввод количества проданного товара.
– Расчет выручки и себестоимости.
– Расчет валовой прибыли и рентабельности.

В декомпозиции блока расчета маржинальной прибыли пользователю вначале потребуется ввести данные о проданных товарах: имя товара, цена продажи, себестоимость, количество проданного товара. На стороне сервера программное обеспечение рассчитывает выручку и себестоимость, а затем и показатели валовой прибыли и рентабельности продукции.

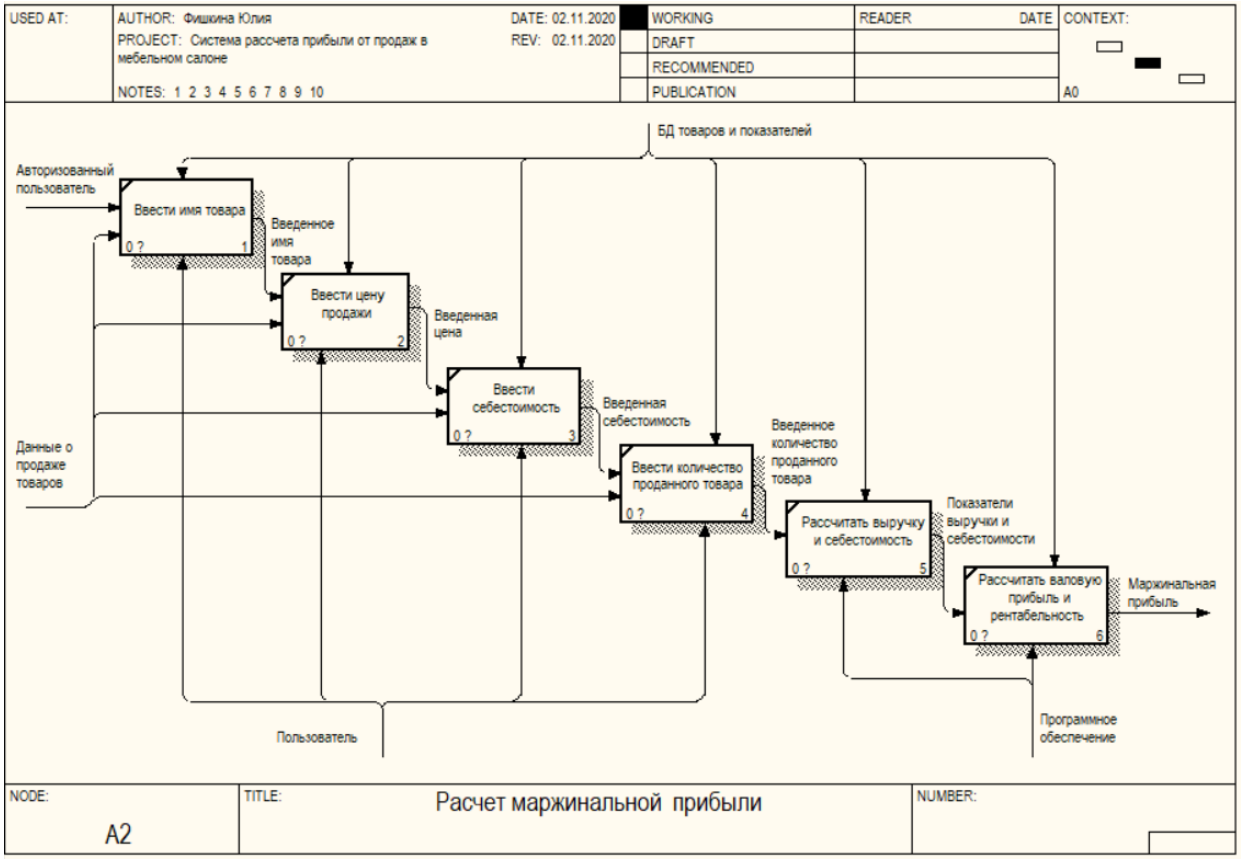

Рисунок 3.3 – Декомпозиция процесса A2

На рисунке 3.4 представлена декомпозиция процесса А3 «Формирование отчета о прибыли и убытках». Данная диаграмма состоит из следующих процессов:

– Ввод операционных расходов.
– Расчет чистой прибыли.
– Расчет рентабельности по чистой прибыли.
– Сложение прибыли каждого товара.
– Вывод отчета на экран.

Входными данными для данной декомпозиции является маржинальная/валовая прибыль и операционные расходы. Для расчета чистой прибыли пользователь обязан ввести операционные расходы компании, в которые входит аренда имущества, амортизация, интернет, канцелярские и командировочные расходы, оплата труда, сотовая связь и страховые взносы. После чего серверная часть приложения произведет расчет чистой прибыли и рентабельности производства, а также отобразит график динамики чистой прибыли за определенный период.

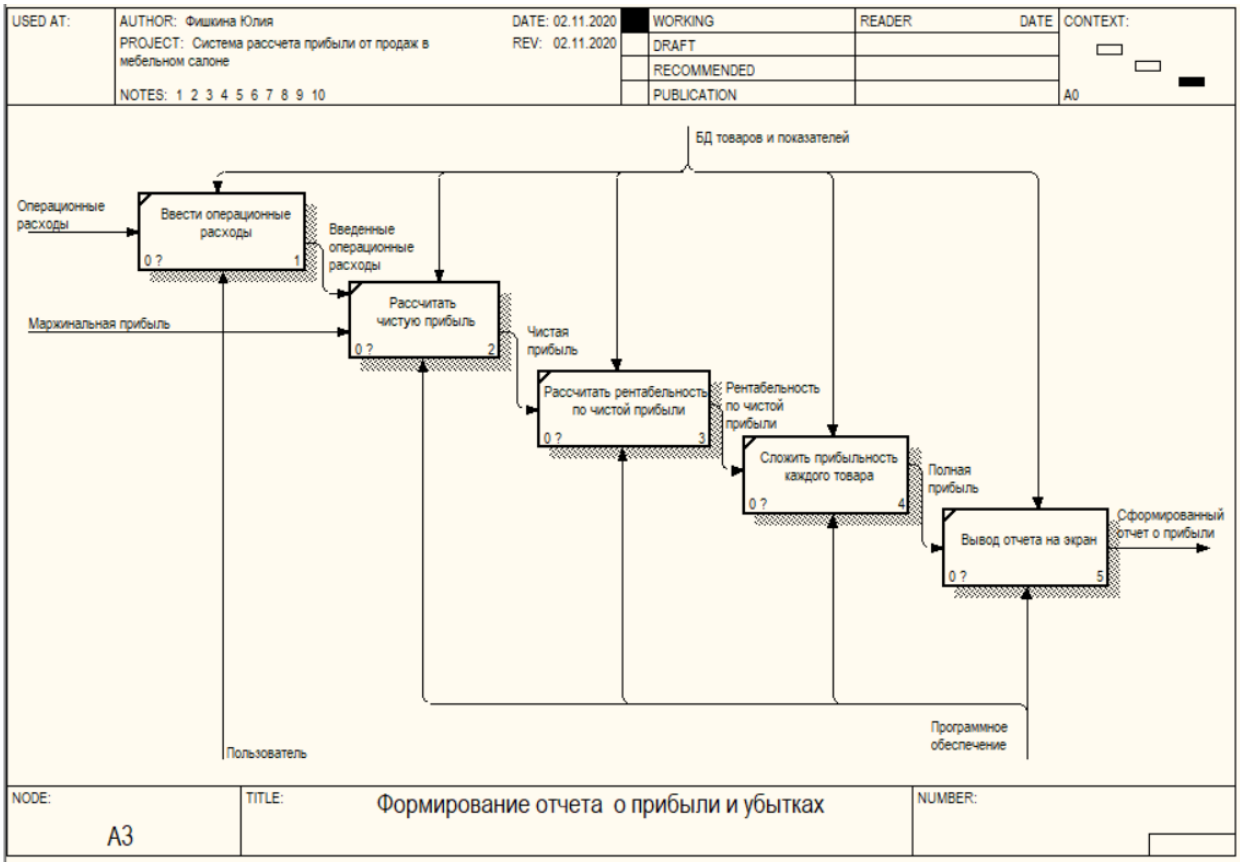

Рисунок 3.4 – Декомпозиция блока А3.

3.2. Модели представления системы

3.2.1. Диаграмма классов системы

Диаграмма классов определяет типы классов системы и различного рода статические связи, которые существуют между ними. На диаграммах классов изображаются также атрибуты классов, операции классов и ограничения, которые накладываются на связи между классами. Вид и интерпретация диаграммы классов существенно зависит от точки зрения (уровня абстракции): классы могут представлять сущности предметной области (в процессе анализа) или элементы программной системы (в процессах проектирования и реализации).

На рисунке 3.5 приведена диаграмма классов обработки данных (Model). Классы Product (сущность: продукт) и SumProfitData (сущность: чистая прибыль ассоциируются с классом ProfitData (сущность: валовая прибыль). Также в данном пакете классов представлен UserAccount для создания экземпляра класс пользователя. Каждый класс симулирует одну таблицу в БД:

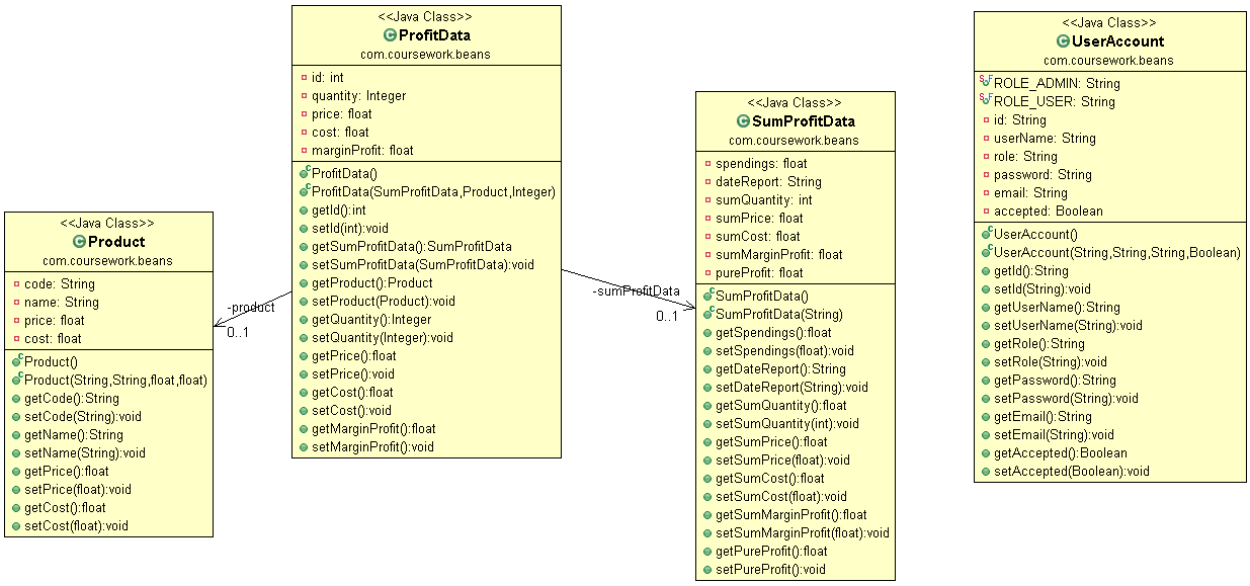

Рисунок 3.5 - Диаграмма классов “Представления”
(пакет com.coursework.beans)

Для установления и дальнейшего получения соединения с базой данной MySQL необходимы классы ConnectionUtils и MySQLConnUtils, собранные в пакете conn. Здесь прописан путь к MySQL драйверу для корректной работы программы. На рисунке 3.6 представлена диаграмма данных классов.

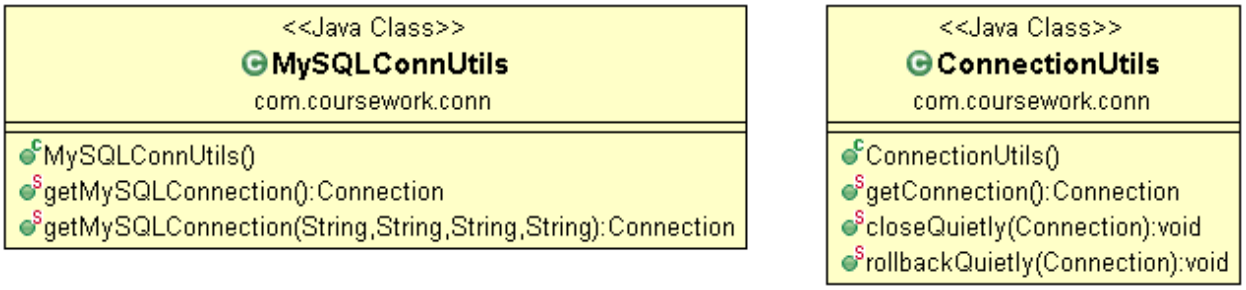

Рисунок 3.6 - Диаграмма классов соединения с БД MySQL
(пакет com.coursework.conn)

В JDBCFilter проверяются, какие запросы действительно вызываются к Сервлету. JDBCFilter с объявлением url-pattern = /*, означает, что любые пользовательские запросы должны пройти через этот фильтр. JDBCFilter проверит запрос для гарантии открытия только соединения JDBC для нужных запросов, например, для Сервлета, во избежание открытия соединения JDBC для обычных запросов, как например файлы картинок, CSS, JS, html. В случае, если пользователь вошел в систему и запомнил информацию прошлого доступа (например, за день до этого). И теперь пользователь возвращается, этот фильтр будет проверять информацию Cookie, которые сохранились браузером и автоматически входит в систему. JDBCFilter и CookieFilter имеют одинаковый url-pattern = /*, конфигурируется для гарантирования, что JDBCFilter выполняется раньше. Порядок объявляется в файле web.xml. В программе расчета прибыли фильтры находятся в пакете com.coursework.filter. Диаграмма классов фильтров представлена на рисунке 3.7:

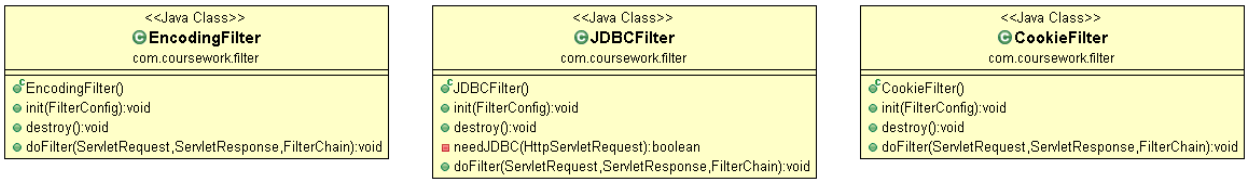

Рисунок 3.7 - Диаграмма классов фильтрации файлов
(пакет com.coursework.filter)

Классы DBUtils и MyUtils являются утилитарными классами для манипулирования данными. В MyUtils классе описаны методы для работы с Cookie и сохранения информации или соединения в атрибут запроса. В DBUtils классе содержатся методы вызова SQL-запросов к базе данных. Диграмма данных классах приведена на рисунке 3.8:

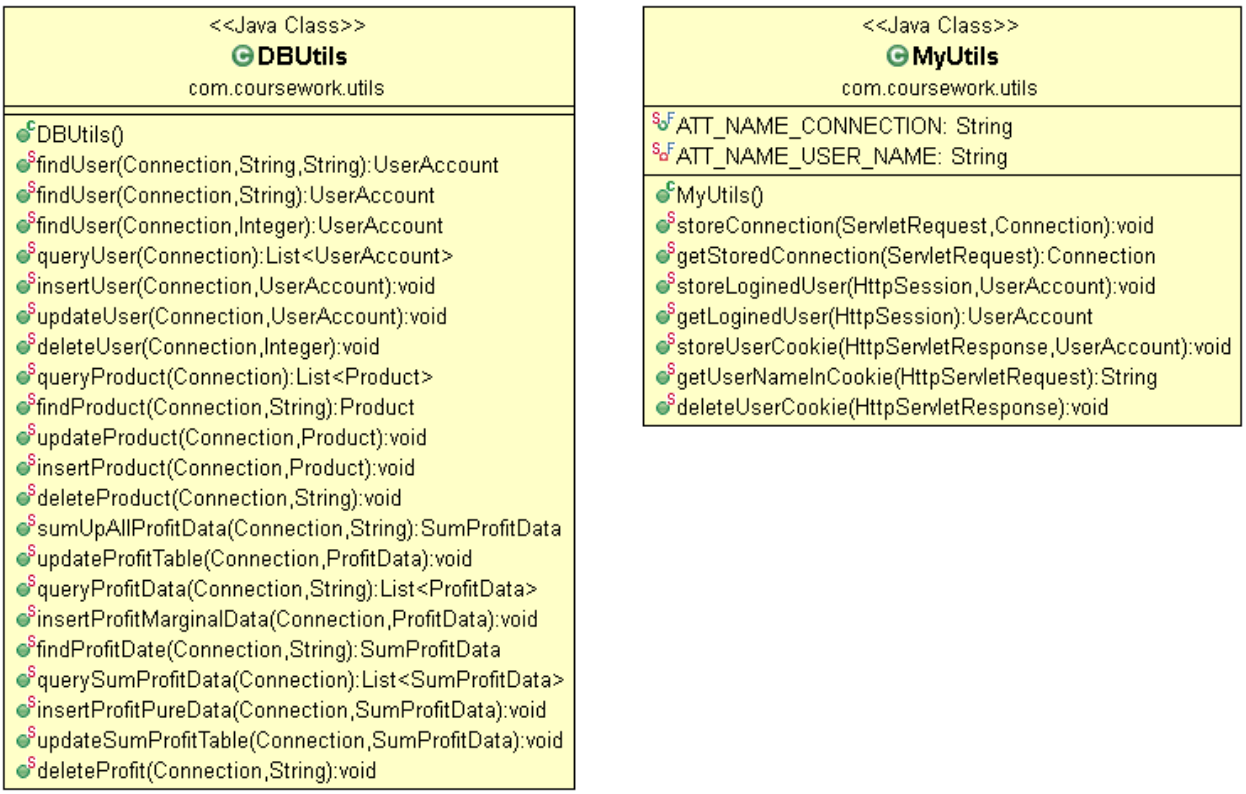

Рисунок 3.8 – Диаграмма вспомогательных классов
(пакет com.coursework.utils)

Диаграмма классов сервлетов (контроллеров) представлена в приложение А. Сервлет выполняет роль управления потоком приложения и обрабатывает программную логику.

3.2.2. Диаграмма последовательности

Диаграммы последовательности (sequence diagram) являются видом диаграмм взаимодействия языка UML, которые описывают отношения объектов в различных условиях. Условия взаимодействия задаются сценарием, полученным на этапе разработки диаграмм вариантов использования.

Диаграммы последовательности отражают поток событий, происходящих в рамках варианта использования. На этих диаграммах изображаются только те объекты, которые непосредственно участвуют во взаимодействии т.к. ключевым моментом является именно динамика взаимодействия объектов во времени и не используются возможные статические ассоциации с другими объектами.

При этом диаграмма последовательности имеет два измерения. Одно – слева направо в виде вертикальных линий, каждая из которых изображает линию жизни отдельного объекта, участвующего во взаимодействии. Второе измерение – вертикальная временная ось, направленная сверху вниз. При этом взаимодействия объектов реализуются посредством сообщений, которые посылаются одними объектами другим. Сообщения изображаются в виде горизонтальных стрелок с именем сообщения и также образуют порядок по времени своего возникновения. Другими словами, сообщения, расположенные на диаграмме последовательности выше, инициируются раньше тех, которые расположены ниже.

На рисунке 3.5 представлена диаграмма последовательности оценки недвижимости. На данной диаграмме представлено несколько объектов: Сервлет, JSP файл, HTTP запрос, Сущность Product, класс обработки БД, с помощью SQL-запросов (DBUtils.java) и База данных. Здесь описана последовательность загрузки веб-страницы “/productList”. Загрузка других веб-страниц будет аналогична. В приложении А диаграмма представлена в увеличенном размере.

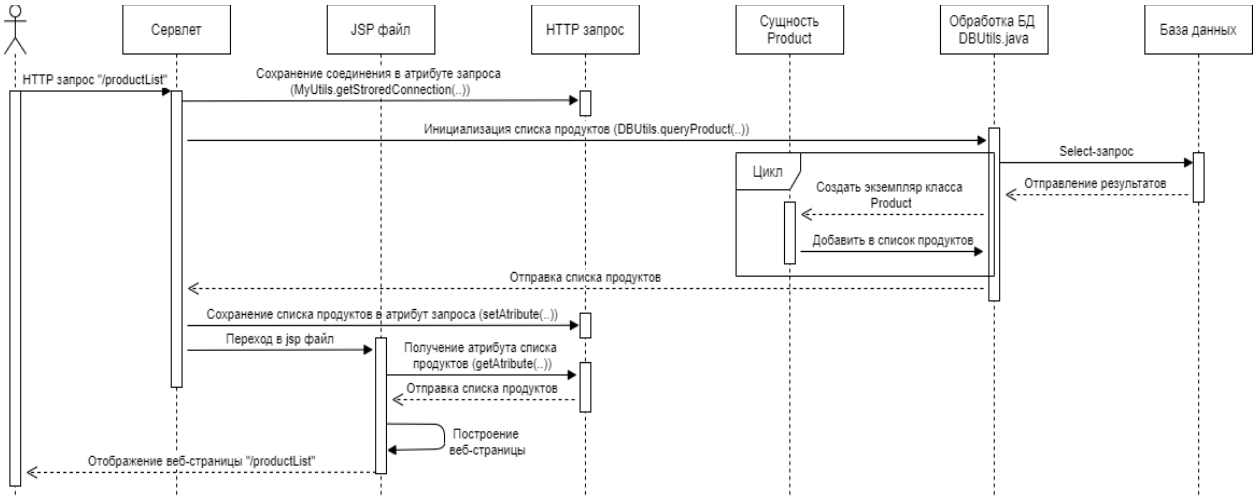

Рисунок 3.5 – Диаграмма последовательности расчета прибыли мебельного салона.

3.2.3. Диаграмма состояния основных объектов системы

Диаграмма состояния показывает, как объект переходит из одного состояния в другое. Диаграммы состояний служат для моделирования динамических аспектов системы. Данная диаграмма полезна при моделировании жизненного цикла объекта. От других диаграмм диаграмма состояний отличается тем, что описывает процесс изменения состояний только одного экземпляра определенного класса - одного объекта, причем объекта реактивного, то есть объекта, поведение которого характеризуется его реакцией на внешние события.

На рисунке 3.6 представлена диаграмма состояния всего процесса авторизации пользователя.

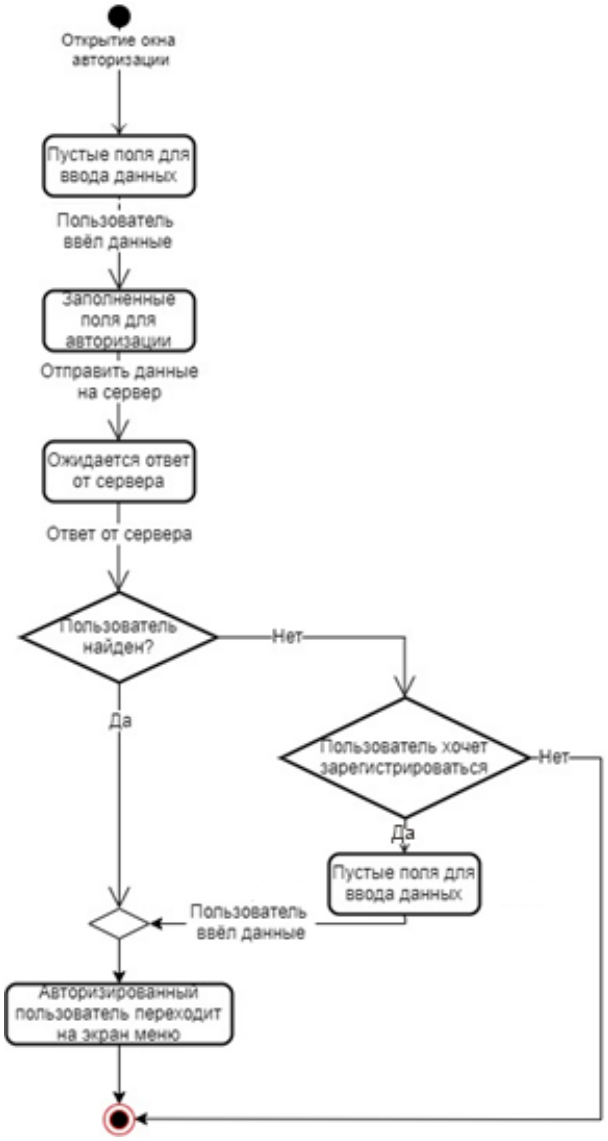

Рисунок 3.6 – Диаграмма состояния процесса авторизации пользователя.

3.2.4. Диаграмма компонентов системы

Диаграмма компонентов – элемент языка моделирования UML, статическая структурная диаграмма, которая показывает разбиение программной системы на структурные компоненты и связи (зависимости) между компонентами. В качестве физических компонентов могут выступать файлы, библиотеки, модули, исполняемые файлы, пакеты.

С помощью диаграммы компонентов представляются инкапсулированные классы вместе с их интерфейсными оболочками, портами и внутренними структурами (которые тоже могут состоять из компонентов и коннекторов).

Компоненты связываются через зависимости, когда соединяется требуемый интерфейс одного компонента с имеющимся интерфейсом другого компонента. Таким образом иллюстрируются отношения клиент-источник между двумя компонентами.

Зависимость показывает, что один компонент предоставляет сервис, необходимый другому компоненту. Зависимость изображается стрелкой от интерфейса или порта клиента к импортируемому интерфейсу.

Когда диаграмма компонентов используется, чтобы показать внутреннюю структуру компонентов, предоставляемый и требуемый интерфейсы составного компонента, могут делегироваться в соответствующие интерфейсы внутренних компонентов.

Делегация показывается связь внешнего контракта компонента с внутренней реализацией этого поведения внутренними компонентами.

Диаграмма компонентов системы представлена на рисунке 3.7.

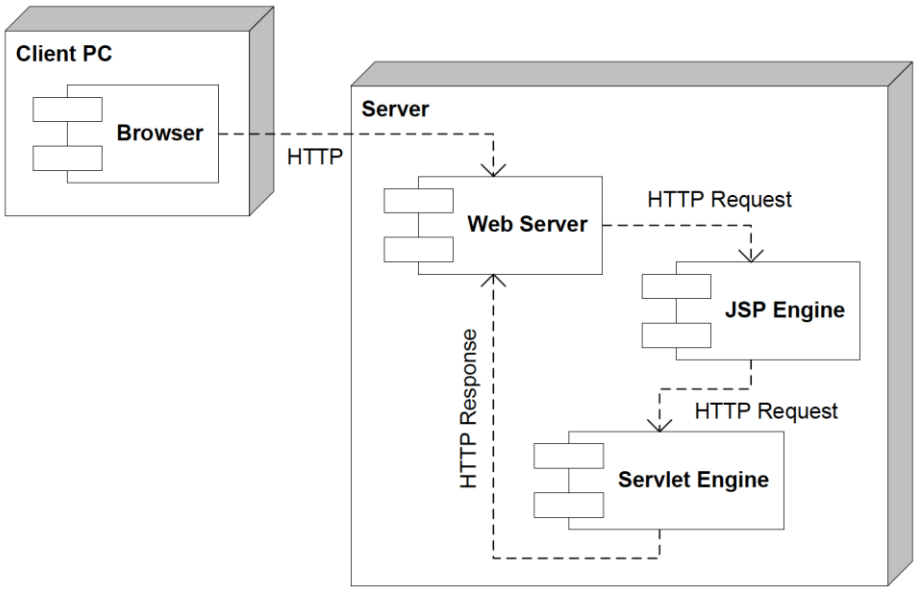

Рисунок 3.7 – Диаграмма компонентов системы.

3.2.5. Диаграмма развертывания системы

Корпоративные приложения часто требуют для своей работы некоторой ИТ-инфраструктуры, хранят информацию в базах данных, расположенных где-то на серверах компании, вызывают веб-сервисы, используют общие ресурсы и т. д. В таких случаях полезно иметь графическое представление инфраструктуры, на которую будет развернуто приложение. Для этого и нужны диаграммы развертывания, которые иногда называют диаграммами размещения.

Такие диаграммы есть смысл строить только для аппаратно-программных систем, тогда как UML позволяет строить модели любых систем, не обязательно компьютерных.

Диаграмма развертывания показывает топологию системы и распределение компонентов системы по ее узлам, а также соединения - маршруты передачи информации между аппаратными узлами. Это единственная диаграмма, на которой применяются “трехмерные” обозначения: узлы системы обозначаются кубиками. Все остальные обозначения в UML - плоские фигуры. Также можно увидеть какие программные компоненты работают на каждом узле, например, база данных – MySQL Workbench с базой данных mebel, и как различные части этого комплекса соединяются друг с другом, в данном случае через протокол HTTP. Соединение сервера и базы данных происходит с помощью подключения драйвера JDBC.

Диаграмма развёртывания представлена на рисунке 3.8.

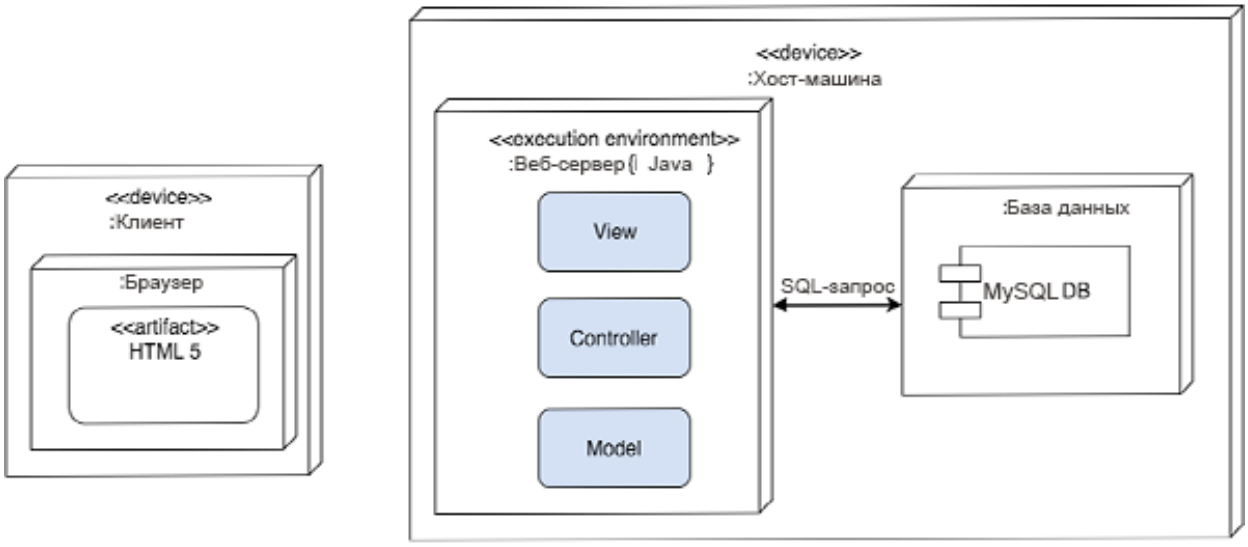

Рисунок 3.8 – Диаграмма развёртывания.

4 ПОСТРОЕНИЕ ИНФОРМАЦИОННОЙ МОДЕЛИ СИСТЕМЫ РАСЧЕТА ПРИБЫЛИ МЕБЕЛЬНОГО САЛОНА

После изучения предметной области данного курсового проекта и анализа требований к формируемой системе была сформирована информационная модель, содержащая следующие сущности:

– Пользователь;
– Продукт;
– Валовая прибыль;
– Чистая прибыль.

На рисунке 4.1 отображены связи между сущностями, а также их поля и типы данных, которыми располагают эти поля.

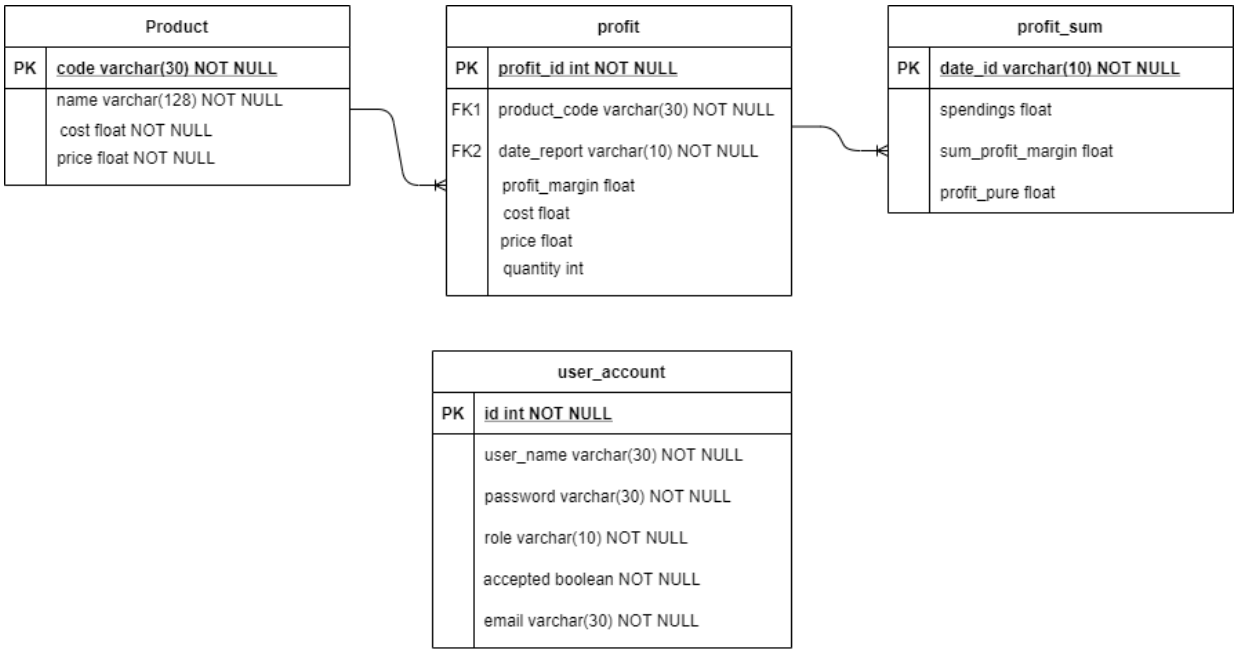

Рисунок 4.1 – Информационная модель базы данных

Сущность «Пользователь» необходима для регистрации и авторизации пользователей. Она содержит следующие атрибуты:

- id, предназначен для хранения уникального номера пользователя;
- user_name – логин для входа в систему и выступает уникальным полем;
- password – пароль для входа в систему;
- role, предназначен для хранения информации о роле, которую выполняет пользователь (администратор, обычный пользователь);

– accepted – статус пользователя системы (если значение равно true, значит у пользователя есть доступ к системе, если значение false, то пользователь не может зайти в систему, пока администратор не подтвердит учетную запись).

– email – электронная почта пользователя.

Рассмотрим более подробно сейчас сущности «Продукт» (product), «Валовая прибыли» (profit) и «Чистая прибыль» (profit_sim). Они непосредственно связаны, сущности «Продукт» и «Чистая прибыль» дополняют некоторой информацией сущность «Валовая прибыль».

Сущность «Продукт» представлена полями:

– code – уникальный код продукта;

– name, предназначен для хранения названия продукта.

– price, цена товара установленная мебельным салоном для сбыта;

– cost – цена мебели, купленной на фабрике (себестоимость продукции).

Атрибуты сущности «Чистая прибыль»:

– date_id – уникальная строка, где указывается месяц и год вычисляемой чистой прибыли и других показателей;

– spendings, предназначен для хранения информации об операционных расходам;

– sum_profit_margin, предназначен для хранения значения валовой прибыли всех товаров за месяц;

– pure_profit, предназначен для хранения чистой прибыли.

В сущности «Валовая прибыль» можно выделить следующие характеристики:

– profit_id – уникальный номер расчета валовой прибыли;

– product_code – внешний ключ от сущности «Продукт», предназначена для хранения идентификатора товара;

– date_report - внешний ключ от сущности «Чистая прибыль», поле предназначено для хранения даты для вычисления прибыли от всех проданных товаров за эту дату;

– profit_margin – валовая прибыль;

– cost – общая себестоимость нескольких одинаковых товаров проданных за месяц;

– price – общая цена нескольких одинаковых товаров проданных за месяц;

– quantity – предназначено для хранения количество проданных товаров за месяц.

Данная база данных соответствует требованиям, потому что:

– все его атрибуты являются простыми, все используемые домены должны содержать только скалярные значения. Не должно быть повторений строк в таблице, а значит находится в 1НФ;

– каждый не ключевой атрибут неприводимо зависит от первичного ключа, а значит находится в 2НФ;

Неприводимость означает, что в составе потенциального ключа отсутствует меньшее подмножество атрибутов, от которого можно также вывести данную функциональную зависимость.

– каждый не ключевой атрибут нетранзитивно зависит от первичного ключа, а значит находится в 3НФ.

Проще говоря, это правило требует выносить все не ключевые поля, содержимое которых может относиться к нескольким записям таблицы в отдельные таблицы. Таким образом, некоторые данные из подсчета прибыли были вынесены в новую сущность «Чистая прибыль».

5 ОБОСНОВАНИЕ ОРИГИНАЛЬНЫХ РЕШЕНИЙ ПО ИСПОЛЬЗОВАНИЮ ТЕХНИЧЕСКИХ И ПРОГРАММНЫХ СРЕДСТВ, НЕ ВКЛЮЧЕННЫХ В ТРЕБОВАНИЯ

В данном курсовом проекте была использована сторонняя JavaScript-библиотека *Chart.js*. А также была подключена к проекту Bootstrap фреймворк для быстрого создания адаптивного сайта.

Плюсы Bootstrap в том, что:

- уменьшает количество времени, затрачиваемого на разработку;
- адаптивность;
- кросс-браузерность;
- легкость в использовании и быстрота в освоении;
- качественный и понятный код;
- позволяют создавать страницы и сайты в едином стиле.

Chart.js - это бесплатная библиотека диаграмм, которая позволяет разработчикам легко отображать анимированные интерактивные диаграммы профессионального качества в своих веб-приложениях.

Обширный набор функций *Chart.js* включает в себя:

– согласованный и хорошо документированный функционал, поддерживающий различные типы диаграмм;

– гибкий дизайн, который легко расширять, и предназначенный как для клиентских, так и для серверных приложений;

6 ОПИСАНИЕ АЛГОРИТМОВ, РЕАЛИЗУЮЩИХ БИЗНЕС-ЛОГИКУ СЕРВЕРНОЙ ЧАСТИ ПРОЕКТИРУЕМОЙ СИСТЕМЫ

В приложение А.3 представлен главный класс серверной части: фильтрация сервлетов. Фильтры используются для следующих задач:

– Работа с ответами сервера перед тем, как они будут переданы клиенту.

– Перехватывание запросов от клиента перед тем, как они будут отправлены на сервер.

Фильтр – это простой Java класс, который имплементирует интерфейс javax.servlet.Filter, который содержит метод doFilter(..), который отображен в приложении А.3.

Схема организации добавления данных о продаже продукта для расчета прибыли 6.1. Надлежит ввести дату и количество продажи товара, а также код товара. При совпадении будет рассчитана валовая прибыль.

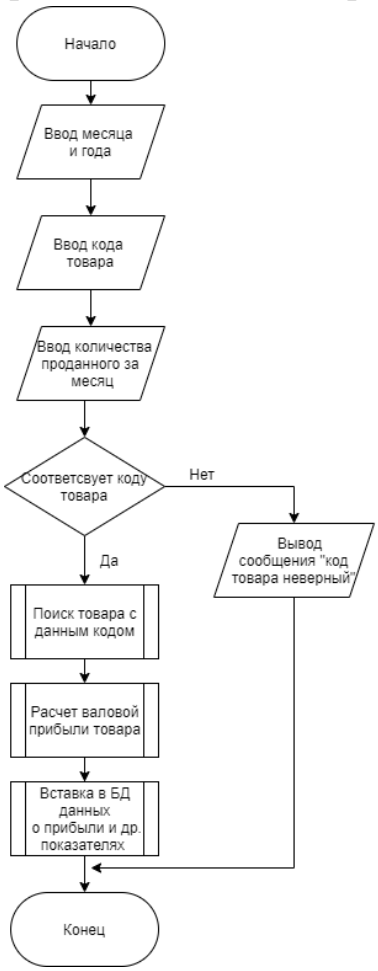

Рисунок 6.1 – Функциональная блок-схема добавления данных о продаже продукта для расчета прибыли

7 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

Первым делом необходимо создать базу данных и добавить стартовые записи для всех таблиц. База данных состоит из себя 3 связанных таблиц, а также таблица данных пользователя, описано ранее в главе 4. Скрипт генерации базы данных и скрипт стартового заполнения базы данных, необходимого для демонстрации программного средства, располагаются в приложении В. При необходимости данные для соединения с БД MySQL нужно изменить, как на рисунке 7.1.

```
String hostName = "localhost:3406";
String dbName = "mebel";
String userName = "root";
String password = "root";
```

Рисунок 7.1 – Данные для соединения с БД MySQL

Далее необходимо запустить сервер Apache Tomcat на порту 8090.

После чего необходимо открыт любой удобный браузер и в строке поисковика написать «http://localhost:8090/SimpleWebApp/». Результатом будет запуск главной страницы веб-приложения с меню (рисунок 7.2).

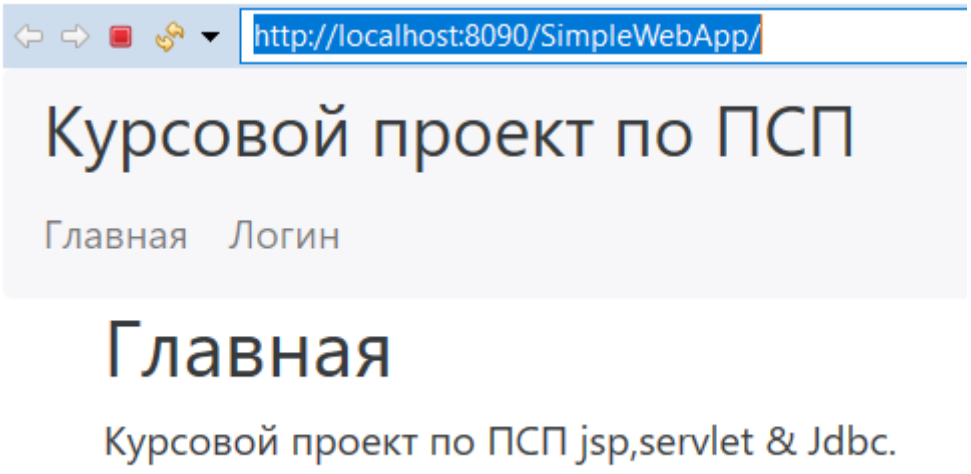

Рисунок 7.2 – Запуск главной страницы веб-приложения

8 РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ РАЗРАБОТАННОЙ СИСТЕМЫ

Процедура входа в систему предполагает ввод логина и пароля (рисунок 8.1). В зависимости от введенных данных пользователю будет предоставлено вид на странице «Мой аккаунт».

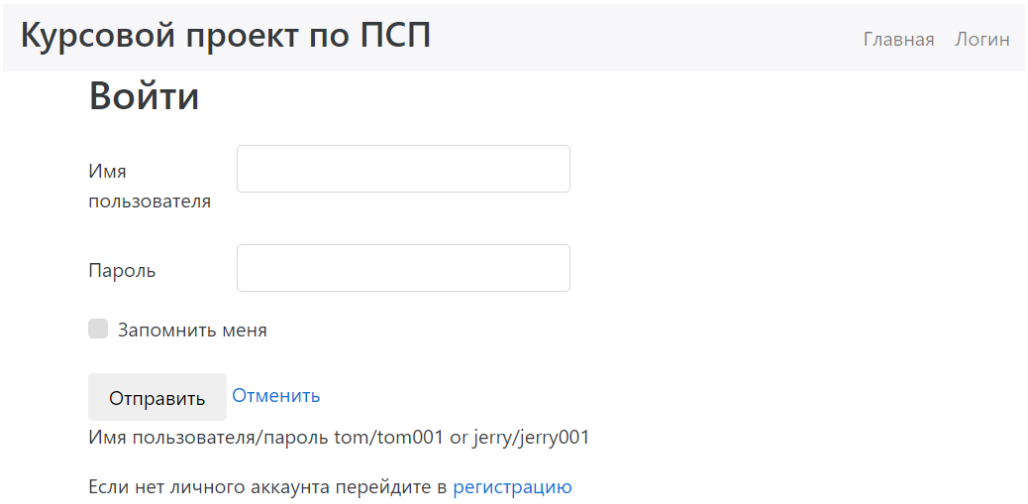

Рисунок 8.1 – Вход в систему

Если у рабочего нет еще аккаунта, то необходимо перейти на страницу регистрации (рисунок 8.2). Данная страница потребует ввести желаемое имя пользователя, пароль и электронную почту.

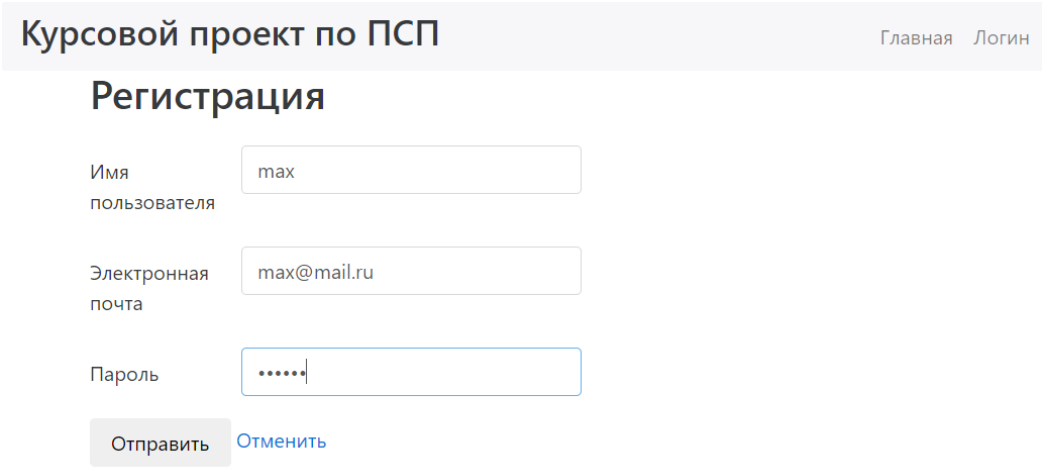

Рисунок 8.2 – Регистрация нового пользователя

По нажатию на кнопку отправить пользователя перенаправит на страницу логина снова. В систему новый зарегистрированный пользователь сможет зайти, когда администратор подтвердит аккаунт зарегистрировавшегося пользователя. Таким образом на рисунке 8.3 видно, что администратор еще не подтвердил пользователя и на экране появилось сообщение «Данный пользователь не подтвержден администратором».

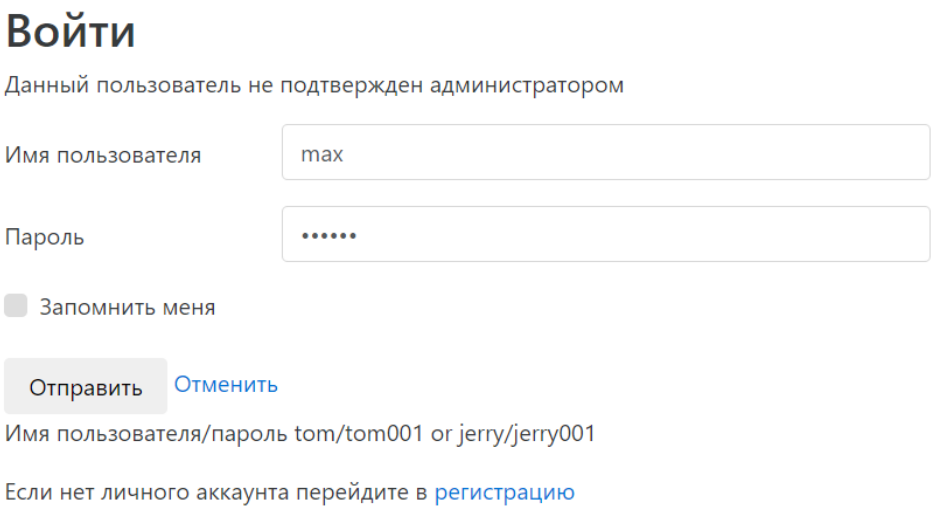

Рисунок 8.3 – Ошибка при входе нового пользователя

В случае несовпадения введенных данных с логинами и паролями, хранящимися в таблице Пользователь в базе данных, программа выдает сообщение о неправильном вводе (рисунок 8.4).

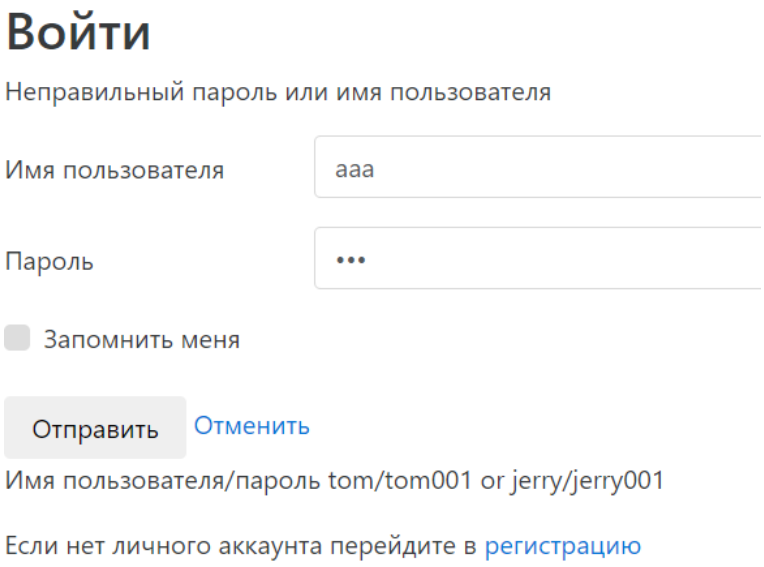

Рисунок 8.4 – Ошибка при вводе неверного логина или пароля

Если же логин и пароль оказались правильными и вход осуществлялся под администратора, то на экране появится страница аккаунта пользователя, изображенное на рисунке 8.5. Администратору предоставлены права просматривать БД пользователей и изменять/удалять данные о пользователях. В случае, если рабочий задет под учетной записью обычного пользователя, то на странице «Мой аккаунт» будет выведено просто имя пользователя.

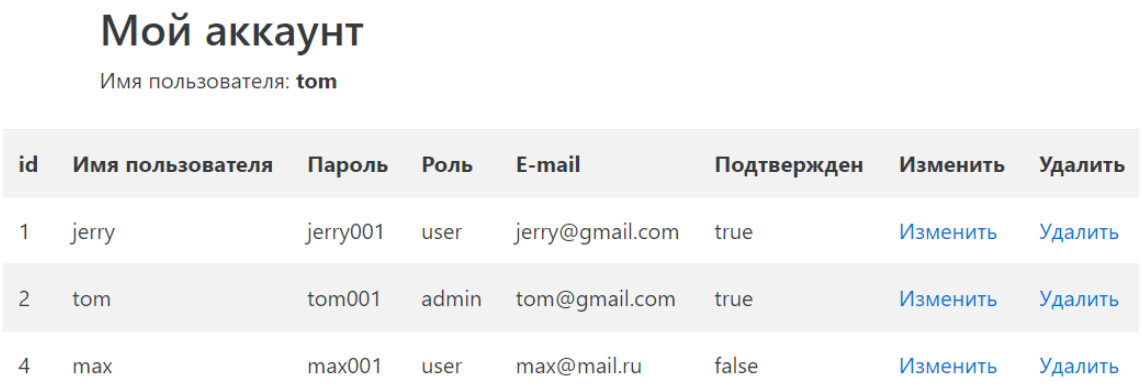

Рисунок 8.5 – Страница администратора «Мой аккаунт»

Рассмотрим редактирование пользователя на рисунке 8.6. В данном случае откроем доступ к системе пользователю «max», у которого в графе «Подтвержден» написано «false» (рисунок 8.5). Для этого необходимо ввести имя пользователя, пароль, электронную почту, выбрать роль и предоставить/отклонить доступ.

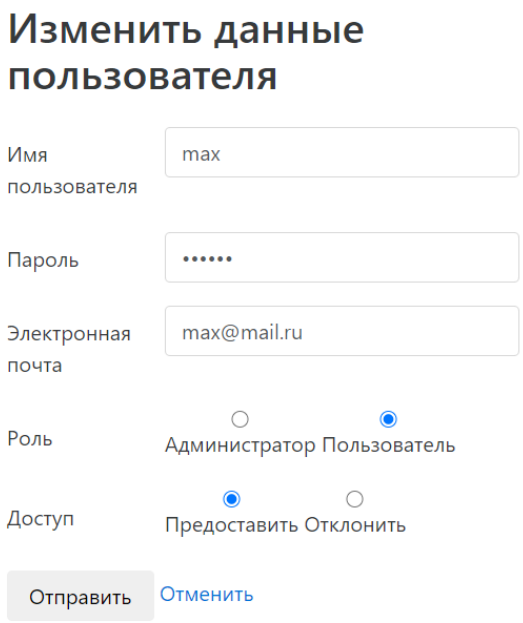

Рисунок 8.6 – Редактирование учетной записи

На рисунке 8.7 видны изменения в учетной записи «max», где доступ к системе поменялся на «true».

2	tom	tom001	admin	tom@gmail.com	true	Изменить	Удалить
4	max	max001	user	max@mail.ru	true	Изменить	Удалить

Рисунок 8.7 – Изменения в БД «Пользователь»

Перейдя в меню на страницу список товаров (рисунок 8.8), на экране отображается все товары, который продает мебельный салон. Записи товаров также, как и учетные записи пользователей можно изменять и удалять.

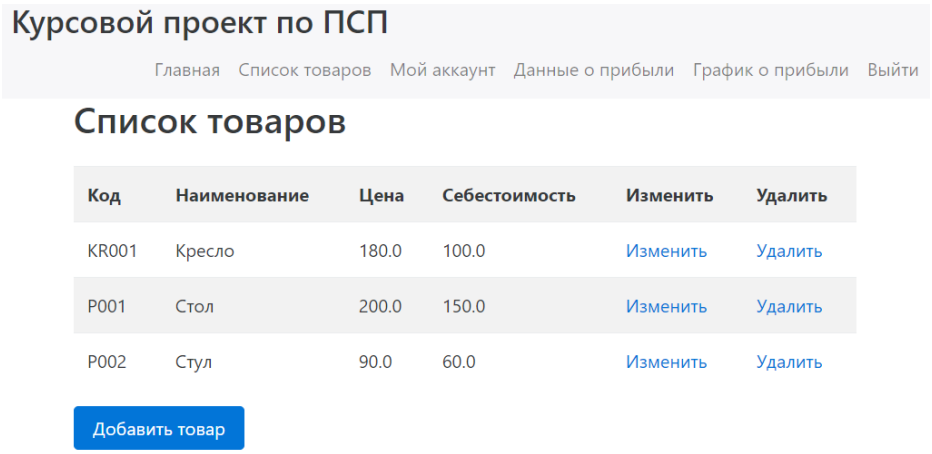

Рисунок 8.8– Страница «Список товаров»

На рисунке 8.9 продемонстрировано добавление нового товара, если некоторые данные были введены некорректно система об этом оповестит (рисунок 8.10).

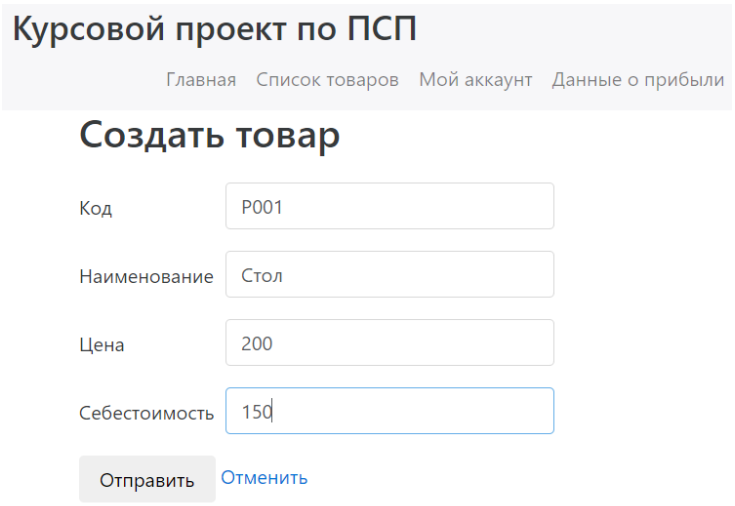

Рисунок 8.9 – Добавление нового товара

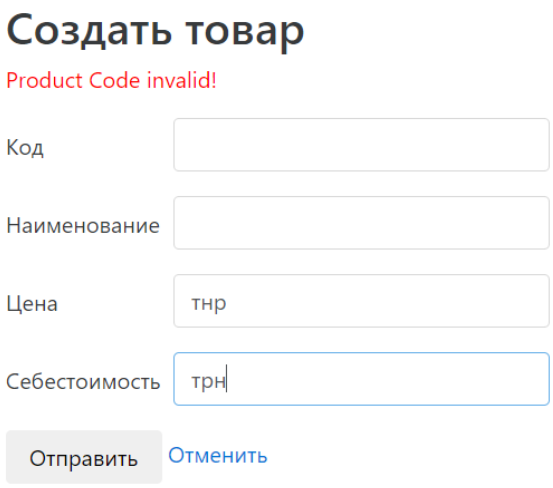

Рисунок 8.10 – Ошибка при добавлении нового товара

При переходе во вкладку «Данные о прибыли» система запрашивает за какой месяц пользователь хочет посмотреть доходы и валовую прибыль мебельного салона (рисунок 8.11).

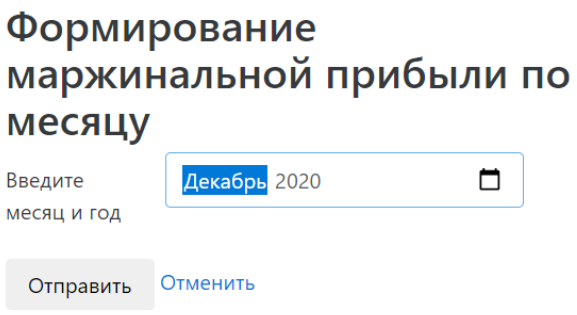

Рисунок 8.11 – Страница ввода месяца и года

После отправки даты система перенаправляет пользователя к странице расчета валовой\маржинальной прибыли. Данная страница отображена на рисунке 8.12. Если пользователь ошибся, то строку с параметрами проданного товара можно удалить из таблицы. Данные о проданных товарах суммируются и выводятся, как последняя строка таблицы. Пользователь может добавить товары, которые были проданы в определенном месяце, нажав на кнопку добавить товар. Страница добавления товара для включения в месячный отчет о доходах и валовой прибыли приведена на рисунке 8.13. Результаты добавления новой строки в таблицу и БД отображены на рисунке 8.14. Для перехода к отчету о чистой прибыли и анализа графика чистой прибыли необходимо ввести операционные расходы и нажать кнопку «Создать отчет за месяц» (рисунок 8.12).

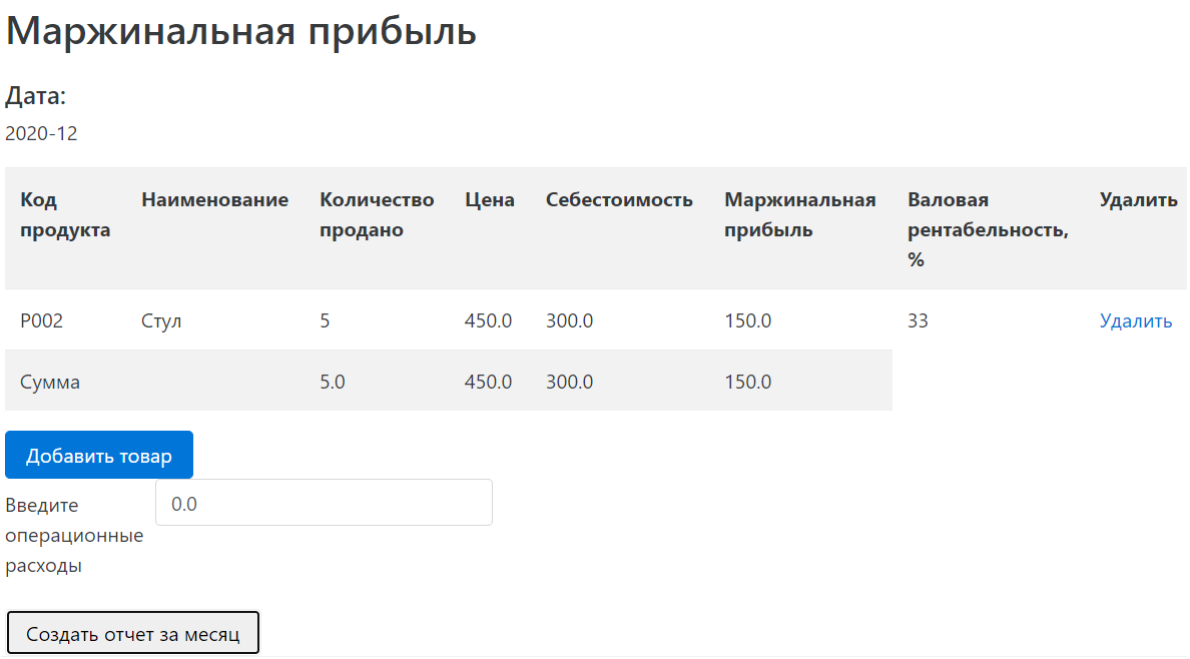

Рисунок 8.12 – Страница расчета маржинальной прибыли

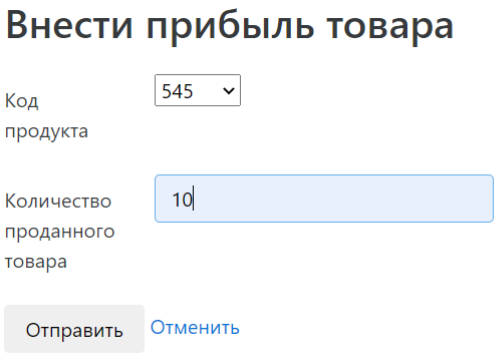

Рисунок 8.13 – Страница добавления новой записи о прибыли товара

Рисунок 8.14 – Обновленная страница расчета маржинальной прибыли

При переходе по кнопке «Создать отчет за месяц» или нажав в меню на пункт «График о прибыли» программа отобразит график и таблицу чистой прибыли по месяцам (рисунки 8.15, 8.16). Таким образом, легко проследить динамику бизнеса.

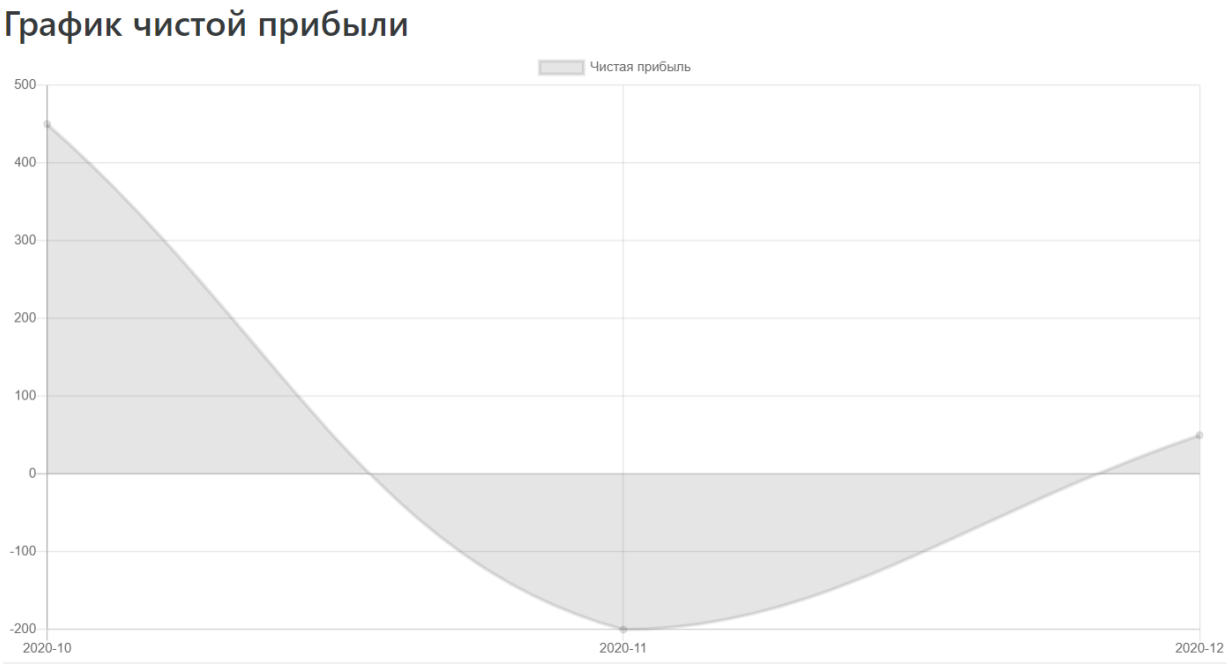

Рисунок 8.15 – График чистой прибыли

Дата	Маржинальная прибыль	Операционные расходы	Чистая прибыль	Рентабельность чистой прибыли
2020-10	950.0	500.0	450.0	20
2020-11	500.0	700.0	-200.0	-15
2020-12	150.0	100.0	50.0	11

Рисунок 8.16 – Таблица расчета чистой прибыли

ЗАКЛЮЧЕНИЕ

Автоматизация расчета прибыли от продаж – эффективный инструмент организации работы мебельного салона, независимо от масштабов и специфики деятельности предприятия.

В курсовой работе разработана программа автоматизированной системы расчета прибыли от продаж в мебельном салоне. Продажи и анализа прибыли и убытков является одним из основных бизнес-процессов мебельной компании, который имеет четкую последовательность действий и процедур, обусловленных действующими стандартами, нормативами и документами. Использование программных продуктов позволяет значительно оптимизировать этот процесс и сократить все виды затрат, сопровождающие его, а также предотвратить возникновение ошибок, присущих персоналу и избежать непредвиденных затрат. Также данное программное обеспечение облегчит работу для бухгалтера и сделает ее более интересной.

Программный продукт разработан для белорусской мебельной компании, которая занимается перепродажей импортной и белорусской мебели с фабрик. В программе использованы анализ и статистика по товарному ассортименту. Все действия над товарами, такие как, добавление в список, удаление, редактирование, отслеживаются непосредственно программой. У пользователя есть возможность просмотреть статистику и проанализировать месячную прибыль компании. Информация о товарах и показателях прибыли хранится в базе данных.

Таким образом, разработанная программа позволяет решить несколько важных задач автоматизированного системы учета поступления и регистрации товара на складе: организовать принятие товаров на склад и обработку данных по поступившим товарам; уменьшить затраты труда на обработку информации; уменьшить вероятность допущения ошибок, сформировать необходимые документы складского учета и снизить количество времени на их формирование, создать условия для дальнейшей работы с товарами при работе с заказом клиента (потребителя); повысить оперативность и актуальность информации.

Программа может быть использована менеджерами, бухгалтером, а также директором, полностью контролирующим мебельную компанию. В дальнейшем программу можно дополнить анализом дополнительных экономических метрик.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Методология IDEF0. – Электронные данные. – Режим доступа: https://itteach.ru/bpwin/metodologiya-idef0\.

[2] Разработка функциональной модели.– Электронные данные. – Режим доступа: https://sites.google.com/site/anisimovkhv/learning/pris/lecture/tema6/tema6_2.

[3] Моделирование бизнес-процессов средствами BPwin. НОУ ИНТУИТ. – Электронные данные. – Режим доступа: https://intuit.ru/studies/courses/2195/55/lecture/1630.

[4] Производство как бизнес-процесс. Справочник экономиста. – Электронные данные. – Режим доступа: https://www.profiz.ru/se/2_2017/opisyvaem_process/

[5] Описание бизнес-процессов компании «Ваша мебель». – Электронные данные. – Режим доступа: https://www.elma-bpm.ru/journal/opisanie-biznes-processov-kompanii-vasha-mebel/.

[6] Чистая прибыль предприятия. Формула. Методы анализа и цели использования. Финансовый анализ. – Электронные данные. – Режим доступа: https://finzz.ru/chistaya-pribyl-predpriyatiya-formula-metody-analiza-celi-ispolzovaniya.html.

[7] Как рассчитывается розничный товарооборот. – Электронные данные. – Режим доступа: https://moneymakerfactory.ru/spravochnik/roznichnyiy-tovarooborot/.

[8] Язык UML. Руководство пользователя. 2-е изд.: Пер. с англ. / Мухин Н. – М.: ДМК Пресс, 2011.

[9] Герберт Шилдт. Java 8. Руководство для начинающих. – Шестое издание: Издательский дом «Вильямс»: Москва, Санкт-Петербург, Киев, 2015 – 32 с.

[10] В. Васвани. MySQL: использование и администрирование MySQL Database. – М.:Питер, 2011. – 25 с.

[11] Java. Методы программирования: уч.-мет. пособие / И. Н. Блинов, В. С. Романчик. — Минск: издательство «Четыре четверти», 2013. – 273 с.

[12] Паттерн MVC (Model-View-Controller) – Электронные данные. – Режим доступа: https://javarush.ru/groups/posts/2536-chastjh-7-znakomstvo-s-patternom-mvc-model-view-controller

ПРИЛОЖЕНИЕ А
(обязательное)
Диаграмма последовательности (к главе 3)

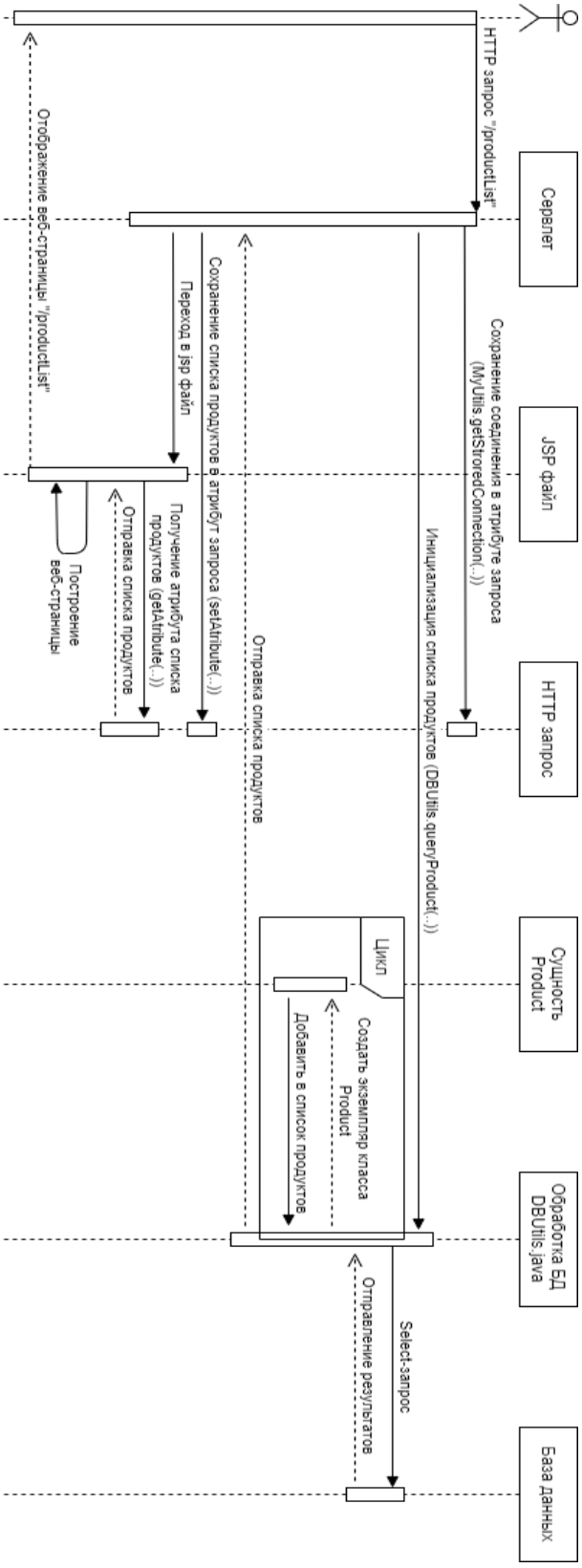

Рисунок А.1 – Диаграмма последовательности

Диаграмма классов Контролера (к главе 3)

Рисунок А.2 – Диаграмма классов Контролера
(пакет com.coursework.servlet)

Схема алгоритма фильтрации сервлетов

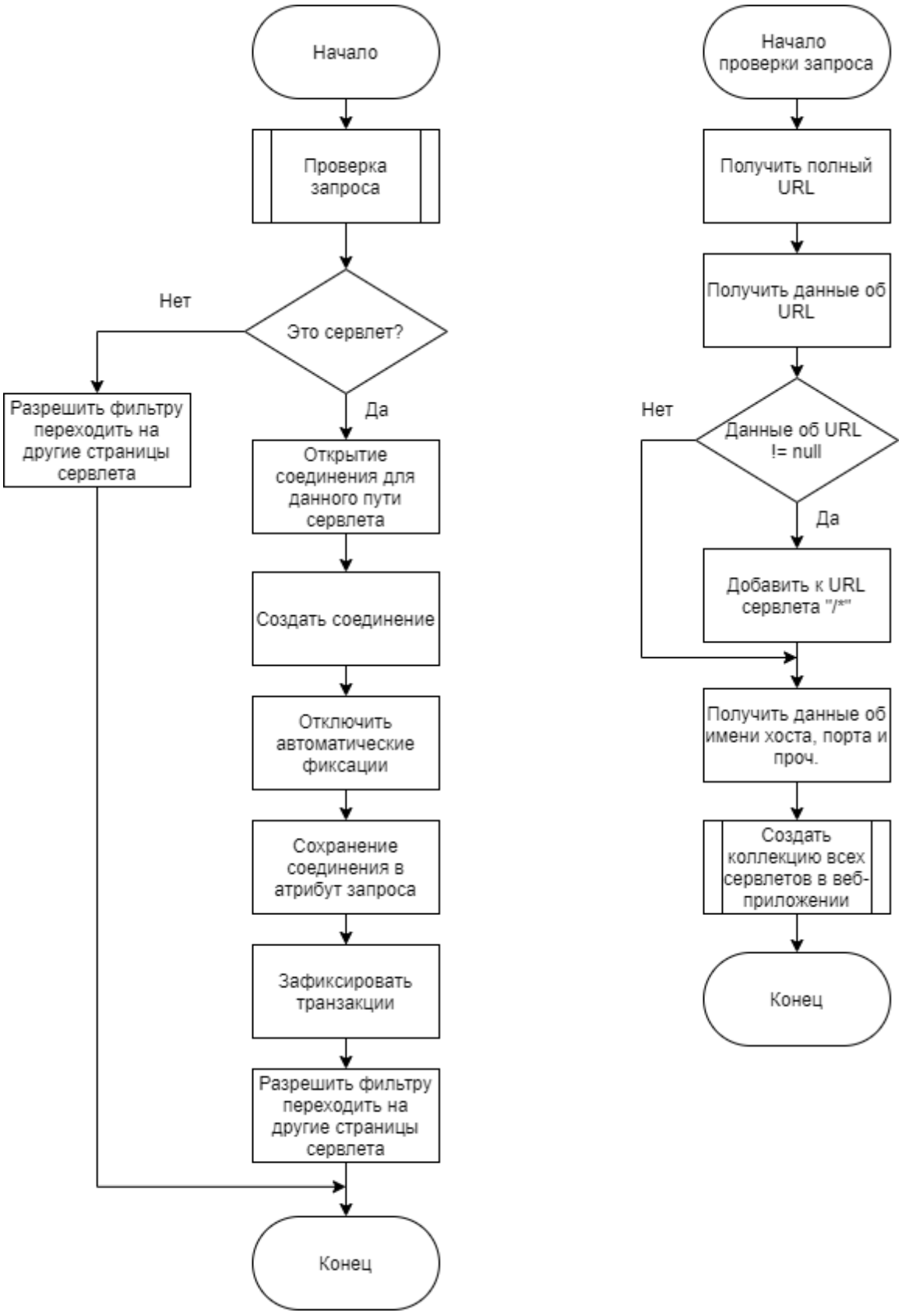

Рисунок А.3 – Схема алгоритма фильтрации сервлетов

ПРИЛОЖЕНИЕ Б
(обязательное)
Скрипт генерации базы данных

```
CREATE DATABASE mebel;

create table User_Account
(
User_Name VARCHAR(30) not null,
Password  VARCHAR(30) not null,
Role VARCHAR(10) not null,
Email VARCHAR(30) not null,
Accepted BOOLEAN not null,
primary key (User_Name)
);

create table Product
(
Code VARCHAR(30) not null,
Name  VARCHAR(128) not null,
Cost FLOAT not null,
Price FLOAT not null,
primary key (Code)
) ;

CREATE TABLE profit (
    profit_id INT AUTO_INCREMENT PRIMARY KEY,
    product_code VARCHAR(30),
    quantity INT,
    profit_margin FLOAT,
    cost FLOAT,
    price FLOAT,
    date_report VARCHAR(10),
    FOREIGN KEY (product_code) REFERENCES product(code),
    FOREIGN KEY (date_report) REFERENCES sum_profit(date_id),
);

CREATE TABLE profit_sum (
    date_id VARCHAR(10) PRIMARY KEY,
    spendings FLOAT,
    sum_profit_margin FLOAT,
    profit_pure FLOAT
);
```

ПРИЛОЖЕНИЕ В
(обязательное)
Листинг кода программы

Web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://xmlns.jcp.org/xml/ns/javaee"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd" id="WebApp_ID" version="3.1">
  <display-name>SimpleWebApp</display-name>
  <welcome-file-list>
    <welcome-file>home</welcome-file>
    <welcome-file>index.html</welcome-file>
  </welcome-file-list>
  <filter-mapping>
    <filter-name>jdbcFilter</filter-name>
    <url-pattern>/*</url-pattern>
  </filter-mapping>
  <filter-mapping>
    <filter-name>cookieFilter</filter-name>
    <url-pattern>/*</url-pattern>
  </filter-mapping>
</web-app>
```

JDBCFilter.java

```
@WebFilter(filterName = "jdbcFilter", urlPatterns = { "/*" })
public class JDBCFilter implements Filter {

  public JDBCFilter() {
  }

  @Override
  public void init(FilterConfig fConfig) throws ServletException {

  }

  @Override
  public void destroy() {

  }

  private boolean needJDBC(HttpServletRequest request) {
    System.out.println("JDBC Filter");
    //
    // Servlet Url-pattern: /spath/*
    //
    // => /spath
    String servletPath = request.getServletPath();
    // => /abc/mnp
    String pathInfo = request.getPathInfo();

    String urlPattern = servletPath;
```

```
    if (pathInfo != null) {
      urlPattern = servletPath + "/*";
    }
    Map<String, ? extends ServletRegistration> servletRegistrations =
request.getServletContext()
        .getServletRegistrations();

    Collection<? extends ServletRegistration> values = servletRegistrations.values();
    for (ServletRegistration sr : values) {
      Collection<String> mappings = sr.getMappings();
      if (mappings.contains(urlPattern)) {
        return true;
      }
    }
    return false;
  }

  @Override
  public void doFilter(ServletRequest request, ServletResponse response, FilterChain
chain)
  throws IOException, ServletException {

    HttpServletRequest req = (HttpServletRequest) request;

    if (this.needJDBC(req)) {

      System.out.println("Open Connection for: " + req.getServletPath());

      Connection conn = null;
      try {
        conn = ConnectionUtils.getConnection();
        conn.setAutoCommit(false);
        MyUtils.storeConnection(request, conn);
        chain.doFilter(request, response);
        conn.commit();
      } catch (Exception e) {
        e.printStackTrace();
        ConnectionUtils.rollbackQuietly(conn);
        throw new ServletException();
      } finally {
        ConnectionUtils.closeQuietly(conn);
      }
    }
    else {

      chain.doFilter(request, response);
    }

  }

}
```

CookieFilter.java

```
@WebFilter(filterName = "jdbcFilter", urlPatterns = { "/*" })
public class JDBCFilter implements Filter {

  public JDBCFilter() {
  }
```

```
  @Override
  public void init(FilterConfig fConfig) throws ServletException {

  }

  @Override
  public void destroy() {

  }

  private boolean needJDBC(HttpServletRequest request) {
    System.out.println("JDBC Filter");
    String servletPath = request.getServletPath();
    String pathInfo = request.getPathInfo();

    String urlPattern = servletPath;

    if (pathInfo != null) {
      urlPattern = servletPath + "/*";
    }

    Map<String, ? extends ServletRegistration> servletRegistrations =
request.getServletContext()
        .getServletRegistrations();


    // Collection of all servlet in your webapp.
    Collection<? extends ServletRegistration> values = servletRegistrations.values();
//получить данные об имени хоста, порт и тд
    for (ServletRegistration sr : values) {
      Collection<String> mappings = sr.getMappings();
      if (mappings.contains(urlPattern)) {
        return true;
      }
    }
    return false;
  }

  @Override
  public void doFilter(ServletRequest request, ServletResponse response, FilterChain
chain)
  throws IOException, ServletException {

    HttpServletRequest req = (HttpServletRequest) request;

    if (this.needJDBC(req)) {

      System.out.println("Open Connection for: " + req.getServletPath());

      Connection conn = null;
      try {
        conn = ConnectionUtils.getConnection();
        conn.setAutoCommit(false);
        MyUtils.storeConnection(request, conn);
        chain.doFilter(request, response);
        conn.commit();
      } catch (Exception e) {
        e.printStackTrace();
        ConnectionUtils.rollbackQuietly(conn);
        throw new ServletException();
```

```
      } finally {
        ConnectionUtils.closeQuietly(conn);
      }
    }

    else {

      chain.doFilter(request, response);
    }

  }

}
```

DBUtils.java

```
public class DBUtils {

      //////////////User DBUtils//////////////
  public static UserAccount findUser(Connection conn, String userName, String
password) throws SQLException {

    String sql = "Select * from User_Account "
                + " where User_Name = ? and Password= ?";

    PreparedStatement pstm = conn.prepareStatement(sql);
    pstm.setString(1, userName);
    pstm.setString(2, password);
    ResultSet rs = pstm.executeQuery();

    if (rs.next()) {
      UserAccount user = new UserAccount();
      user.setUserName(userName);
      user.setPassword(password);
      String role = rs.getString("role");
      Boolean accepted = rs.getBoolean("accepted");
      user.setAccepted(accepted);
      user.setRole(role);
      return user;
    }
    return null;
  }

  public static UserAccount findUser(Connection conn, String userName) throws
SQLException {

    String sql = "Select * from User_Account " + " where User_Name = ? ";

    PreparedStatement pstm = conn.prepareStatement(sql);
    pstm.setString(1, userName);

    ResultSet rs = pstm.executeQuery();

    if (rs.next()) {
      String password = rs.getString("Password");
      UserAccount user = new UserAccount();
      user.setUserName(userName);
      user.setPassword(password);
      return user;
    }
    return null;
```

```
  }

  public static UserAccount findUser(Connection conn, Integer id) throws SQLException
{

          String sql = "Select * from User_Account " + " where id = ? ";

          PreparedStatement pstm = conn.prepareStatement(sql);
          pstm.setInt(1, id);

          ResultSet rs = pstm.executeQuery();

          if (rs.next()) {
            UserAccount user = new UserAccount();
            user.setId(Integer.toString(id));
            return user;
          }
          return null;
        }

  public static List<UserAccount> queryUser(Connection conn) throws SQLException {
          String sql = "Select * from User_Account";

          PreparedStatement pstm = conn.prepareStatement(sql);

          ResultSet rs = pstm.executeQuery();
          List<UserAccount> list = new ArrayList<UserAccount>();
          while (rs.next()) {
            String id = Integer.toString(rs.getInt("id"));
            String userName = rs.getString("User_Name");
            String password = rs.getString("Password");
            String role = rs.getString("Role");
            String email = rs.getString("Email");
            Boolean accepted = rs.getBoolean("Accepted");
            UserAccount user = new UserAccount();
            user.setId(id);
            user.setUserName(userName);
            user.setPassword(password);
            user.setRole(role);
            user.setEmail(email);
            user.setAccepted(accepted);
            list.add(user);
          }
          return list;
  }

  public static void insertUser(Connection conn, UserAccount user) throws
SQLException {
          String sql = "Insert into User_Account(User_Name, Password, role, Email,
accepted) values (?,?,?,?,?)";

          PreparedStatement pstm = conn.prepareStatement(sql);

          pstm.setString(1, user.getUserName());
          pstm.setString(2, user.getPassword());
          pstm.setString(3, user.getRole());
          pstm.setString(4, user.getEmail());
          pstm.setBoolean(5, false);

          pstm.executeUpdate();
```

```
        }

  public static void updateUser(Connection conn, UserAccount user) throws
SQLException {
          String sql = "Update User_Account set User_Name=?, Password=?, role=?,
email=?, Accepted=? where id=? ";

          PreparedStatement pstm = conn.prepareStatement(sql);

          pstm.setString(1, user.getUserName());
          pstm.setString(2, user.getPassword());
          pstm.setString(3, user.getRole());
          pstm.setString(4, user.getEmail());
          pstm.setBoolean(5, user.getAccepted());

          pstm.setString(6, user.getId());
          pstm.executeUpdate();
  }

  public static void deleteUser(Connection conn, Integer id) throws SQLException {
          String sql = "Delete from user_account where id=? ";

          PreparedStatement pstm = conn.prepareStatement(sql);

          pstm.setInt(1, id);

          pstm.executeUpdate();
  }
  ////////////////////////////////////////////////


  //////////////Product DBUtils//////////////
  public static List<Product> queryProduct(Connection conn) throws SQLException {
    String sql = "Select * from Product ";

    PreparedStatement pstm = conn.prepareStatement(sql);

    ResultSet rs = pstm.executeQuery();
    List<Product> list = new ArrayList<Product>();
    while (rs.next()) {
      String code = rs.getString("Code");
      String name = rs.getString("Name");
      float price = rs.getFloat("Price");
      float cost = rs.getFloat("Cost");
      Product product = new Product();
      product.setCode(code);
      product.setName(name);
      product.setPrice(price);
      product.setCost(cost);
      list.add(product);
    }
    return list;
  }

  public static Product findProduct(Connection conn, String code) throws SQLException
{
    String sql = "Select * from Product where Code=?";

    PreparedStatement pstm = conn.prepareStatement(sql);
    pstm.setString(1, code);
```

```
    ResultSet rs = pstm.executeQuery();

    while (rs.next()) {
      String name = rs.getString("Name");
      float price = rs.getFloat("Price");
      float cost = rs.getFloat("Cost");
      Product product = new Product(code, name, price, cost);
      return product;
    }
    return null;
  }

  public static void updateProduct(Connection conn, Product product) throws
SQLException {
    String sql = "Update Product set Name =?, Price=?, Cost=? where Code=? ";

    PreparedStatement pstm = conn.prepareStatement(sql);

    pstm.setString(1, product.getName());
    pstm.setFloat(2, product.getPrice());
    pstm.setFloat(3, product.getCost());
    pstm.setString(4, product.getCode());
    pstm.executeUpdate();
  }

  public static void insertProduct(Connection conn, Product product) throws
SQLException {
    String sql = "Insert into Product(Code, Name, Price, Cost) values (?,?,?,?)";

    PreparedStatement pstm = conn.prepareStatement(sql);

    pstm.setString(1, product.getCode());
    pstm.setString(2, product.getName());
    pstm.setFloat(3, product.getPrice());
    pstm.setFloat(4, product.getCost());

    pstm.executeUpdate();
  }

  public static void deleteProduct(Connection conn, String code) throws SQLException
{
    String sql = "Delete from Product where Code= ?";

    PreparedStatement pstm = conn.prepareStatement(sql);

    pstm.setString(1, code);

    pstm.executeUpdate();
  }
      ////////////////////////////////////////////////

  public static SumProfitData sumUpAllProfitData(Connection conn, String date) throws
SQLException {
        String sql  = "SELECT SUM(quantity), SUM(price), SUM(cost),
SUM(profit_margin) FROM Profit WHERE date_report = ?";

        PreparedStatement pstm = conn.prepareStatement(sql);
        pstm.setString(1, date);
```

```
        ResultSet rs = pstm.executeQuery();
        SumProfitData sumProfit = new SumProfitData();
        if (rs.next()) {
               sumProfit.setSumQuantity(rs.getInt(1));
               sumProfit.setSumPrice(rs.getFloat(2));
               sumProfit.setSumCost(rs.getFloat(3));
               sumProfit.setSumMarginProfit(rs.getFloat(4));
        }
        sumProfit.setDateReport(date);
        return sumProfit;
  }

  public static void updateProfitTable(Connection conn, ProfitData profit) throws
SQLException {
        String sql  =  "UPDATE Profit SET price = ?, cost = ?, profit_margin = ?
WHERE profit_id = ?";

          PreparedStatement pstm = conn.prepareStatement(sql);

          pstm.setFloat(1, profit.getPrice());
          pstm.setFloat(2, profit.getCost());
          pstm.setFloat(3, profit.getMarginProfit());
          pstm.setInt(4, profit.getId());
          pstm.executeUpdate();
  }

  public static List<ProfitData> queryProfitData(Connection conn, String date) throws
SQLException {
          String sql = "Select prod.*, prof.quantity, prof.profit_id from Product
prod, Profit prof Where prod.code = prof.product_code And prof.date_report = ?";

          PreparedStatement pstm = conn.prepareStatement(sql);
          pstm.setString(1, date);

          ResultSet rs = pstm.executeQuery();
          List<ProfitData> list = new ArrayList<ProfitData>();
          while (rs.next()) {
            String code = rs.getString("Code");
            Integer quantity = rs.getInt("Quantity");
            Integer id = rs.getInt("profit_id");
            Product product = findProduct(conn,code);
            ProfitData profit = new ProfitData();
            profit.setId(id);
            profit.setProduct(product);
            profit.setQuantity(quantity);
            profit.setPrice();
            profit.setCost();
            profit.setMarginProfit();
            updateProfitTable(conn, profit);

            list.add(profit);
          }
          return list;
        }

            //вставка для дальнейшего расчета валовой прибыли
        public static void insertProfitMarginalData(Connection conn, ProfitData
profit) throws SQLException {
          String sql = "Insert into Profit(product_code, quantity, date_report)
values (?,?,?)";
```

```
            PreparedStatement pstm = conn.prepareStatement(sql);
            Product product = profit.getProduct();

            pstm.setString(1, product.getCode());
            pstm.setInt(2, profit.getQuantity());
            pstm.setString(3, profit.getSumProfitData().getDateReport());

            pstm.executeUpdate();
          }

        ////////////////////////////////////////////

        //////////////SumProfit DBUtils//////////////

          public static SumProfitData findProfitDate(Connection conn, String date)
throws SQLException {
                  String sql = "Select * from profit_sum where date_id=?";

                  PreparedStatement pstm = conn.prepareStatement(sql);
                  pstm.setString(1, date);

                  ResultSet rs = pstm.executeQuery();

                  while (rs.next()) {
                     SumProfitData profit = new SumProfitData(date);
                     profit.setDateReport(date);

                    return profit;
                  }
                  return null;
                }

          public static List<SumProfitData> querySumProfitData(Connection conn) throws
SQLException {
                  String sql = "Select * from profit_sum";

                  PreparedStatement pstm = conn.prepareStatement(sql);

                  ResultSet rs = pstm.executeQuery();
                  List<SumProfitData> list = new ArrayList<SumProfitData>();
                  while (rs.next()) {
                    String date_id = rs.getString("date_id");
                    Float spendings =rs.getFloat("spendings");
                    Float sum_profit_margin =rs.getFloat("sum_profit_margin");
                   // Float profit_pure =rs.getFloat("profit_pure");
                    Float revenue = rs.getFloat("revenue");
                    SumProfitData profit = new SumProfitData(date_id);
                profit.setSumMarginProfit(sum_profit_margin);
                profit.setSumPrice(revenue);
                profit.setSpendings(spendings);
                profit.setPureProfit();

                    list.add(profit);
                  }
                  return list;
                }

          public static void insertProfitPureData(Connection conn, SumProfitData
profit) throws SQLException {
```

```
                String sql = "Insert into profit_sum(date_id, sum_profit_margin)
values (?,?)";

                PreparedStatement pstm = conn.prepareStatement(sql);

                //pstm.setFloat(1, profit.getSpendings());
                pstm.setString(1, profit.getDateReport());
                pstm.setFloat(2, profit.getSumMarginProfit());

                pstm.executeUpdate();
        }

        public static void updateSumProfitTable(Connection conn, SumProfitData
profit) throws SQLException {
              String sql  =  "UPDATE profit_sum SET spendings = ?, sum_profit_margin
= ?, profit_pure = ?, revenue = ? WHERE date_id = ?";

                PreparedStatement pstm = conn.prepareStatement(sql);

                pstm.setFloat(1, profit.getSpendings());
                pstm.setFloat(2, profit.getSumMarginProfit());
                pstm.setFloat(3, profit.getPureProfit());
                pstm.setFloat(4, profit.getSumPrice());
                pstm.setString(5, profit.getDateReport());

                pstm.executeUpdate();
        }

        public static void deleteProfit(Connection conn, String id) throws
SQLException {
                String sql = "Delete from profit where profit_id= ?";

                PreparedStatement pstm = conn.prepareStatement(sql);

                pstm.setString(1, id);

                pstm.executeUpdate();
              }
      }
```