**СОДЕРЖАНИЕ**

Введение 2

1 Анализ предметной области 3

1.1 Постановка задачи 3

1.2 Анализ требований к автоматизированной системе 3

2 Системное проектирование программного средства. 4

3 Логическое моделирование 7

3.1 Диаграмма вариантов использования 7

3.2 Диаграмма классов 8

3.3 Диаграмма деятельности 9

3.4 Диаграмма коопераций 11

3.5 Диаграмма последовательности 12

3.6 Диаграмма развёртывания 13

3.7 Диаграмма компонентов 14

4 Генерация кода 16

Заключение 22

Список использованных источников 21

ВВЕДЕНИЕ

В сегодняшнем технологическом мире все больше увеличивается значительность информации в самом широком смысле этого слова и связанных с ней информационных технологий. Для больших объемов информации главным качеством является их структурированность, так как именно от характера структуры данных зависит скорость обработки поиска информации.

База данных на сегодняшний день — это самый распространенный «повод» для написания программ. Все современные языки программирования содержат в себе встроенные возможности для быстрого и удобного создания СУБД.

СУБД — это, конечно, нечто большее, чем просто набор структурированных данных. СУБД можно назвать «умной» средой, с помощью которой можно обрабатывать, искать, передавать, хранить данные. От качества СУБД зависит эффективность работы с базой данных в целом. Современные СУБД — это «высокоинтеллектуальные» системы, позволяющие работать не только с базами данных, но и с базами знаний. Это направление, связанное с накоплением, получением, сортировкой и использованием знаний, является довольно новым. С понятием «база знаний» тесно связанно понятие искусственного интеллекта, что говорит об огромных масштабах работ, которые потребуются провести для совершенствования сегодняшних СУБД, так как искусственный интеллект — это новый горизонт в нашей компьютерной эпохе. При работе баз данных и связанных с ними программ обработки и поиска информации необходимо учитывать специфику предметной области — одно из важнейших достоинств базы, данных. Конечно, здесь можно говорить о том, что уже давно существуют универсальные формы для баз данных, универсальные СУБД, универсальный модуль обработки данных. Но эта универсальность требует больших, (если не огромных) затрат машинного времени и ресурсов, не говоря уж о стоимости всего ПО. Поэтому чаще всего используют специализированные БД и СУБД.

В данной курсовой работе также будет создана специализированная база данных для учёта товаров на складе магазина.

1. **Анализ предметной области**

Магазин одежды — юридическое лицо, которое выполняет функции розничной торговли.

Магазин ведет непосредственную работу с клиентами по вопросам розничной торговли товаров.

Установлена четкая процедура принятия решений о определении конечной цены товара и оформлении сделки по покупке. Общая процедура продажи товаров регламентируется специальными регламентами и инструкциями.

Клиенты магазина — физические лица, для которых предусмотрены распространяемые виды товаров.

Особенности:

– у каждого товара есть разные размеры, у каждого размера есть свое количество

– пользователь может заказать один и тот же товар в нескольких размерах

– один пользователь может заказать несколько товаров в одном заказе

– заказы могут производить как зарегистрированные, так и не зарегистрированные пользователи

– администратор может просматривать все заказы которые находятся в обработке.

## **Постановка задачи**

В данной контрольной работе необходимо реализовать базу данных «Магазин одежды».

* 1. **Анализ требований к автоматизированной системе**

Процесс проектирования архитектуры программного обеспечения состоит в проектировании структуры всех его компонент, функционально связанных с решаемой задачей, включая сопряжения между ними и требования к ним.

Архитектура программного обеспечения в традиционном смысле включает определение всех модулей программ, их иерархии и сопряжения между ними и данными.

Во время разработки архитектуры программного обеспечения выполняется его модульно-иерархическое построение.

1. **Системное проектирование программного средства**

IDEF0 — методология функционального моделирования (англ. function modeling) и графическая нотация, предназначенная для формализации и описания бизнес-процессов. Отличительной особенностью IDEF0 является её акцент на соподчинённость объектов. В IDEF0 рассматриваются логические отношения между работами, а не их временная последовательность (поток работ).

Стандарт IDEF0 представляет организацию как набор модулей, здесь существует правило — наиболее важная функция находится в верхнем левом углу, кроме того есть правило стороны:

* стрелка входа приходит всегда в левую кромку активности,
* стрелка управления — в верхнюю кромку,
* стрелка механизма — нижняя кромка,
* стрелка выхода — правая кромка.

Описание выглядит как «чёрный ящик» с входами, выходами, управлением и механизмом, который постепенно детализируется до необходимого уровня. Также для того чтобы быть правильно понятым, существуют словари описания активностей и стрелок. В этих словарях можно дать описания того, какой смысл вы вкладываете в данную активность либо стрелку.

Также отображаются все сигналы управления, которые на DFD (диаграмме потоков данных) не отображались. Данная модель используется при организации бизнес-проектов и проектов, основанных на моделировании всех процессов: как административных, так и организационных.

Графический язык IDEF0 удивительно прост и гармоничен. В основе методологии лежат четыре основных понятия:

Первым из них является понятие функционального блока (Activity Box). Функциональный блок графически изображается в виде прямоугольника   
(см. рис. 2.1) и олицетворяет собой некоторую конкретную функцию в рамках рассматриваемой системы. По требованиям стандарта название каждого функционального блока должно быть сформулировано в глагольном наклонении (например, «производить услуги», а не «производство услуг»).

IDEF1X является методом для разработки реляционных баз данных и использует условный синтаксис, специально разработанный для удобного построения концептуальной схемы. Концептуальной схемой мы называем универсальное представление структуры данных в рамках коммерческого предприятия, независимое от конечной реализации базы данных и аппаратной платформы. Будучи статическим методом разработки, IDEF1X изначально не предназначен для динамического анализа по принципу "AS IS", тем не менее, он иногда применяется в этом качестве, как альтернатива методу IDEF1. Использование метода IDEF1X наиболее целесообразно для построения логической структуры базы данных после того, как все информационные ресурсы исследованы (скажем с помощью метода IDEF1) и решение о внедрении реляционной базы данных, как части корпоративной информационной системы, было принято.

Хотя терминология IDEF1X практически совпадает с терминологией IDEF1, существует ряд фундаментальных отличий в теоретических концепциях этих методологий. Сущность в IDEF1X описывает собой совокупность или набор экземпляров похожих по свойствам, но однозначно отличаемых друг от друга по одному или нескольким признакам. Каждый экземпляр является реализацией сущности. Таким образом, сущность в IDEF1X описывает конкретный набор экземпляров реального мира, в отличие от сущности в IDEF1, которая представляет собой абстрактный набор информационных отображений реального мира.

Декомпозиция контекстной диаграммы функциональной моделипоказана на рис. 2.1. Контекстная диаграмма модели показана на рисунке 2.2. Для построения модели использовался продукт BPwin 4.0 фирмы Computer Associates.

Субъектом моделирования является программное средство «Администрирование магазина», цель — описать функциональность работы программного средства, точка зрения– пользователь, администратор.

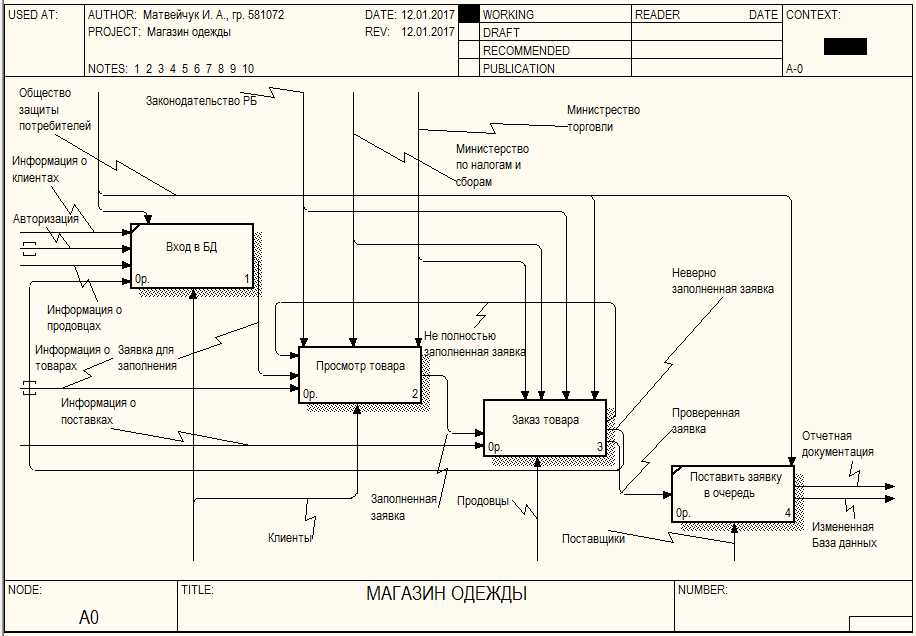


Рисунок 2.1 — Декомпозиция контекстной диаграммы

функциональной модели

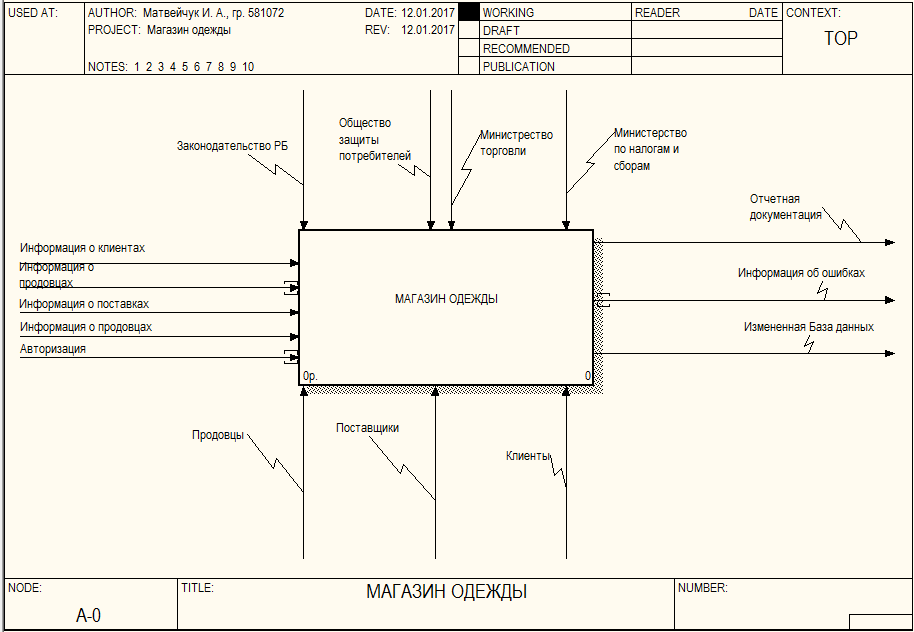


Рисунок 2.2 — Контекстная диаграмма модели

1. **Логическое моделирование**

Основное назначение логического представления состоит в анализе структурных и функциональных отношений между элементами модели системы. Различные элементы логического представления, такие как классы, ассоциации, состояния, сообщения, не существуют материально или физически. Они лишь отражают наше понимание структуры физической системы или аспекты ее поведения.

### **Диаграмма вариантов использования**

Модель вариантов использования предназначается для определения требований к системе. Она включает в себя актеров, варианты использования и связи между ними. Для отображения этой модели язык UML предлагает использовать диаграммы Use Case (вариант использования) совместно с моделями State Diagram (диаграммы состояний) и Activity Diagram (диаграммы деятельности/активности). Последние используются для конкретизации вариантов использования системы

Разработка диаграммы вариантов использования преследует цели:

* определить общие границы и контекст моделируемой предметной области на начальных этапах проектирования системы;
* сформулировать общие требования к функциональному поведению проектируемой системы;
* разработать исходную концептуальную модель системы для ее последующей детализации в форме логических и физических моделей;
* подготовить исходную документацию для взаимодействия разработчиков системы с ее заказчиками и пользователями.

Диаграмма вариантов использования представлена на рисунке 3.1.1.

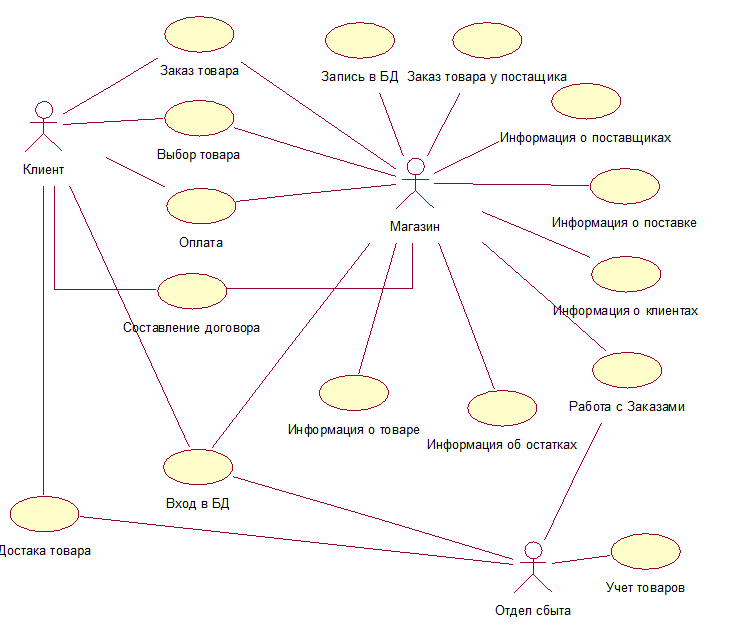


Рисунок 3.1.1 — Диаграмма вариантов использования

### **3.2 Диаграмма классов**

Диаграмма классов — диаграмма, демонстрирующая классы системы, их атрибуты, методы и взаимосвязи между ними. Входит в UML.

Существует два вида:

* статический вид диаграммы рассматривает логические взаимосвязи классов между собой;
* аналитический вид диаграммы рассматривает общий вид и взаимосвязи классов, входящих в систему.

Существуют разные точки зрения на построение диаграмм классов в зависимости от целей их применения:

* концептуальная точка зрения — диаграмма классов описывает модель предметной области, в ней присутствуют только классы прикладных объектов;
* точка зрения спецификации — диаграмма классов применяется при проектировании информационных систем;
* точка зрения реализации — диаграмма классов содержит классы, используемые непосредственно в программном коде (при использовании объектно-ориентированных языков программирования).

Диаграмма классов представлена на рисунке 3.2.1.

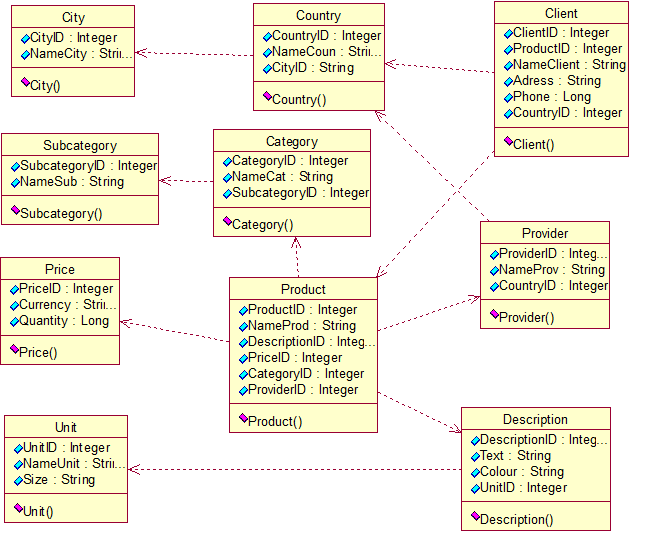


Рисунок 3.2.1 — Диаграмма классов

**3.3 Диаграмма деятельности**

Диаграмма деятельности — UML-диаграмма, на которой показано разложение некоторой деятельности на её составные части. Под деятельностью (англ. activity) понимается спецификация исполняемого поведения в виде координированного последовательного и параллельного выполнения подчинённых элементов — вложенных видов деятельности и отдельных действий англ. action, соединённых между собой потоками, которые идут от выходов одного узла ко входам другого.

Диаграммы деятельности используются при моделировании бизнес-процессов, технологических процессов, последовательных и параллельных вычислений.

Диаграмма деятельности представлена на рисунке 3.3.1.

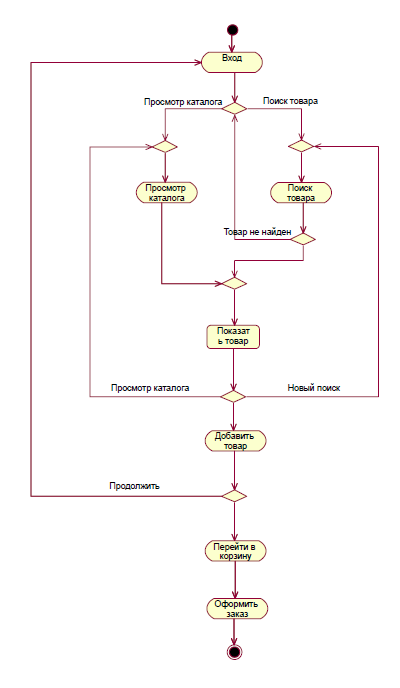


Рисунок 3.3.1 — Диаграмма деятельности

### **3.4 Диаграмма коопераций**

Главная особенность диаграммы кооперации заключается в возможности графически представить не только последовательность взаимодействия, но и все структурные отношения между объектами, участвующими в этом взаимодействии.

Прежде всего, на диаграмме кооперации в виде прямоугольников изображаются участвующие во взаимодействии объекты, содержащие имя объекта, его класс и, возможно, значения атрибутов. Далее, как и на диаграмме классов, указываются ассоциации между объектами в виде различных соединительных линий. При этом можно явно указать имена ассоциации и ролей, которые играют объекты в данной ассоциации. Дополнительно могут быть изображены динамические связи — потоки сообщений. Они представляются также в виде соединительных линий между объектами, над которыми располагается стрелка с указанием направления, имени сообщения и порядкового номера в общей последовательности инициализации сообщений.

В отличие от диаграммы последовательности, на диаграмме кооперации изображаются только отношения между объектами, играющими определенные роли во взаимодействии. На этой диаграмме не указывается время в виде отдельного измерения. Поэтому последовательность взаимодействий и параллельных потоков может быть определена с помощью порядковых номеров. Следовательно, если необходимо явно специфицировать взаимосвязи между объектами в реальном времени, лучше это делать на диаграмме последовательности.

Диаграмма коопераций представлена на рисунке 3.4.1.



Рисунок 3.4.1 — Диаграмма коопераций

### **3.5 Диаграмма последовательности**

Диаграмма последовательности — диаграмма, на которой показано взаимодействие объектов, упорядоченное по времени, с отражением продолжительности обработки и последовательности их проявления. Используется в языке UML.

Основными элементами диаграммы последовательности являются обозначения объектов (прямоугольники с названиями объектов), вертикальные «линии жизни», отображающие течение времени, прямоугольники, отражающие деятельность объекта или исполнение им определенной функции (прямоугольники на пунктирной «линии жизни»), и стрелки, показывающие обмен сигналами или сообщениями между объектами.

Диаграмма последовательности представлена на рисунке 3.5.1.

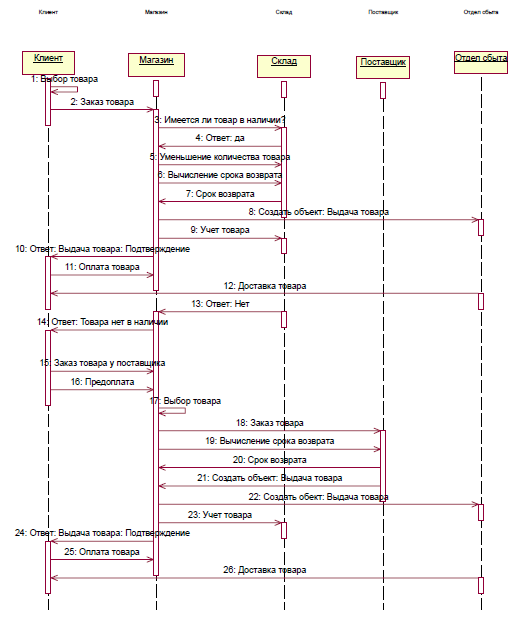


Рисунок 3.5.1 — Диаграмма последовательности

### **Диаграмма развёртывания**

Диаграмма развёртывания в UML моделирует физическое развертывание артефактов на узлах. Например, чтобы описать веб-сайт диаграмма развертывания должна показывать, какие аппаратные компоненты («узлы») существуют (например, веб-сервер, сервер базы данных, сервер приложения), какие программные компоненты («артефакты») работают на каждом узле (например, веб-приложение, база данных), и как различные части этого комплекса соединяются друг с другом (например, JDBC, REST, RMI).

Узлы представляются как прямоугольные параллелепипеды с артефактами, расположенными в них, изображенными в виде прямоугольников. Узлы могут иметь под узлы, которые представляются как вложенные прямоугольные параллелепипеды. Один узел диаграммы развертывания может концептуально представлять множество физических узлов, таких как кластер серверов баз данных.

Существует два типа узлов:

– узел устройства;

– узел среды выполнения.

Узлы устройств — это физические вычислительные ресурсы со своей памятью и сервисами для выполнения программного обеспечения, такие как обычные ПК, мобильные телефоны. Узел среды выполнения — это программный вычислительный ресурс, который работает внутри внешнего узла и который предоставляет собой сервис, выполняющий другие исполняемые программные элементы.

Диаграмма развёртывания представлена на рисунке 3.6.1.

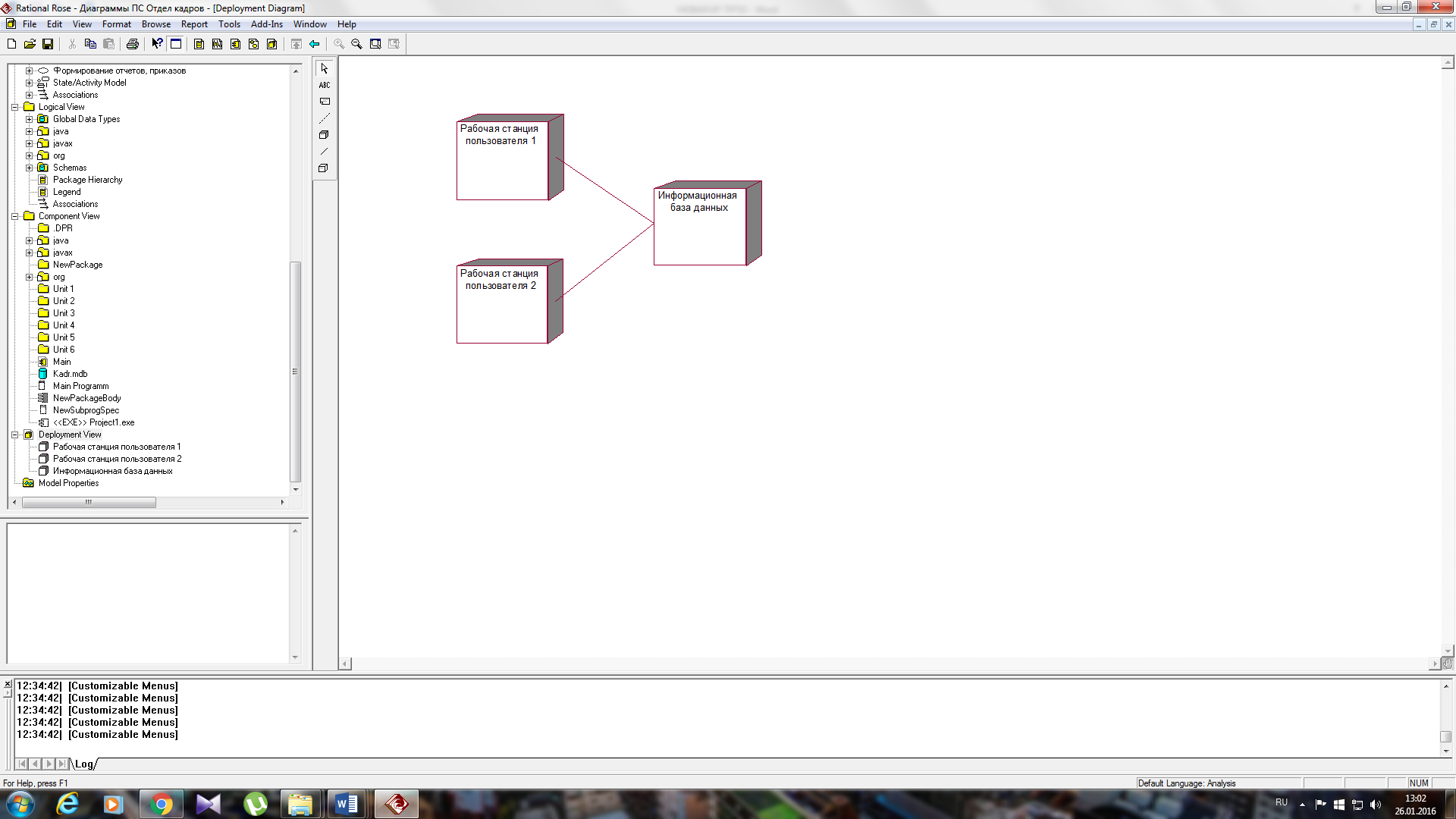


Рисунок 3.6.1 — Диаграмма развёртывания

### **Диаграмма компонентов**

Диаграмма компонентов — статическая структурная диаграмма, показывает разбиение программной системы на структурные компоненты и связи (зависимости) между компонентами. В качестве физических компонентов могут выступать файлы, библиотеки, модули, исполняемые файлы, пакеты и т. п.

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 представлена на рисунке 3.7.1.

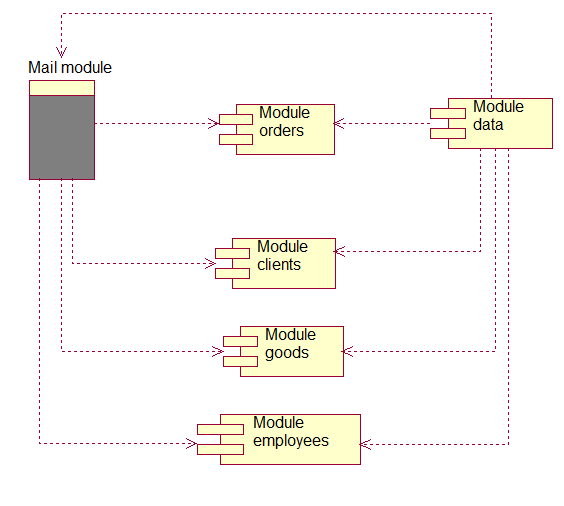


Рисунок 3.7.1 — Диаграмма компонентов

### **Генерация кода**

Сгенерированный код:

public class Unit

{

public Integer UnitID;

public String NameUnit;

public String Size;

/\*\*

\* @roseuid 58472FEB005C

\*/

public Unit()

{

}

}

public class Subcategory

{

public Integer SubcategoryID;

public String NameSub;

/\*\*

\* @roseuid 58472FEB002C

\*/

public Subcategory()

{

}

}

public class Provider

{

public Integer ProviderID;

public String NameProv;

public Integer CountryID;

/\*\*

\* @roseuid 58472FEB00AF

\*/

public Provider()

{

}

}

public class Product

{

public Integer ProductID;

public String NameProd;

public Integer DescriptionID;

public Integer PriceID;

public Integer CategoryID;

public Integer ProviderID;

/\*\*

\* @roseuid 58472FEB00CF

\*/

public Product()

{

}

}

public class Price

{

public Integer PriceID;

public String Currency;

public Long Quantity;

/\*\*

\* @roseuid 58472FEB0044

\*/

public Price()

{

}

}

public class Description

{

public Integer DescriptionID;

public String Text;

public String Colour;

public Integer UnitID;

/\*\*

\* @roseuid 58472FEB0090

\*/

public Description()

{

}

}

public class Country

{

public Integer CountryID;

public String NameCoun;

public String CityID;

/\*\*

\* @roseuid 58472FEA03E5

\*/

public Country()

{

}

}

public class Client

{

public Integer ClientID;

public Integer ProductID;

public String NameClient;

public String Adress;

public Long Phone;

public Integer CountryID;

/\*\*

\* @roseuid 58472FEB00F9

\*/

public Client()

{

}

}

public class City

{

public Integer CityID;

public String NameCity;

/\*\*

\* @roseuid 58472FEA03C4

\*/

public City()

{

}

}

public class Category

{

public Integer CategoryID;

public String NameCat;

public Integer SubcategoryID;

/\*\*

\* @roseuid 58472FEB0074

\*/

public Category()

{

}

}

**ЗАКЛЮЧЕНИЕ**

Предметной областью курсового проектирования является программное средство «Магазин одежды». Разработанное программное средство оптимизирует ведение учета поступления материалов, создание документов движения, заполнение справочников.

Пояснительная записка состоит из следующих разделов:

1. Постановка задачи. В данном разделе была поставлена задача по разработке программного средства «Магазин одежды».

2. Описание основного процесса предметной области с использованием стандарта IDEF0.

3. Описание информационной модели с использованием стандарта IDEF1.

4. Описание моделей информационной системы с использованием языка UML.

5 Генерация кода на основе моделей Rational Rose.

В ходе написания контрольной работы были закреплены теоретические знания по таким разделам как составление функциональной модели (IDEF0), разработка логической и физической модели IDEF1X, проектирование системы с помощью [унифицированного языка моделирования](http://edu.dvgups.ru/METDOC/GDTRAN/YAT/ITIS/PROEK_INF_SIS/METOD/UMK_DO/frame/UMK_DO/M6/L11.htm#11_1) (UML).

Контрольная работа выполнялась в соответствии с методическими требованиями и указаниями.

Список использованных источников

1. ГОСТ 2.105-95 ЕСКД. Введ. 1996–07–01. — М.: Изд-во стандартов, 1996.
2. ГОСТ 19.104–78 — Единая система программной документации. Основные надписи.
3. Маклаков С.В. BPwin и ERwin: CASE — средства для разработки информационных систем.
4. Федотов Д.Э., Семенов Ю.Д., Чижик К.Н. Практикум для высших учебных заведений. CASE-технологии.-157с.
5. Интернет обучение [Электронный ресурс] / Котляр Д.С. Использование CASE-средства ERwin для автоматизации проектирования и разработки базы данных — Режим доступа: http://royallib.com/read/ bezopasnost — Дата доступа: 15.12.2015.
6. Интернет обучение [Электронный ресурс] / Информационные системы и технологии — Режим доступа: <http://www.narfu.ru> — Дата доступа: 19.12.2015.
7. Интернет обучение [Электронный ресурс] / Нормализация структурны данных — Режим доступа: http://infostart.ru/public/269803/ — Дата доступа: 15.01.2016.
8. Интернет обучение [Электронный ресурс] / Программа компьютерного моделирования BpWin — Режим доступа: http://bourabai.kz/cm/bpwin.htm — Дата доступа: 11.11.2016.