**Реферат**

Пояснительная записка курсового проекта (работы) 19 с., 4 рис., 1 табл.,

5 источников, 1 прил.

ЯЗЫК ПРОГРАММИРОВАНИЯ C#, MICROSOFT

VISUAL STUDIO, ЭЛЕМЕНТЫ УПРАВЛЕНИЯ, ТЕСТИРОВАНИЕ.

Объектом исследования являются методы и технологии разработки

программных продуктов.

Цель работы состоит в закреплении основ и углублении знаний в

области программирования на языке высокого уровня.

**Содержание**

[Введение 3](#_Toc55902968)

[1 Нормативные документы 4](#_Toc55902969)

[2 Анализ предметной области 5](#_Toc55902970)

[3 Разработка технического задания 5](#_Toc55902971)

[3.1 Введение 5](#_Toc55902972)

[3.2 Требования к программе или программному изделию 6](#_Toc55902973)

[3.2.1 Требования к функциональным характеристикам 6](#_Toc55902974)

[3.2.2 Требования к надежности 6](#_Toc55902975)

[3.2.3 Требования к составу и параметрам технических средств 8](#_Toc55902976)

[3.2.4 Требования к информационной и программной совместимости 8](#_Toc55902977)

[3.2.5 Специальные требования 9](#_Toc55902978)

[3.2.6 Технико-экономические показатели 9](#_Toc55902979)

[3.2.7 Стадии и этапы разработки 9](#_Toc55902980)

[3.2.8 Этапы разработки 10](#_Toc55902981)

[Проектирование классов; 10](#_Toc55902982)

[Разработка пользовательского интерфейса; 10](#_Toc55902983)

[Заключение 10](#_Toc55902984)

[Список использованных источников; 11](#_Toc55902985)

# Введение

Темой данного курсового проекта является разработка приложения для тренажерного зала. Эта тема очень актуальна в данное время, так как спорт пользуется огромной популярностью и все больше и больше открываются разные залы для поддержания своей физической формы. Чтобы облегчить задачу по регистрации и записи в зал будет разработана и спроектирована данная автоматизированная система.

По данной теме существуют уже огромное количество различных и разнообразных автоматизированных программ и сайтов, которые облегчают и снимают часть обязанностей с сотрудников спортивных залов.

В данном курсовом проекте будет спроектирована и разработана автоматизированная программа для тренажерного зала, в которая также включает в себя базу данных и локальный сервер для содержания информации о зарегистрированных пользователях и их записях в зал, что облегчит работу персонала.

Перед данным ПО стоят следующие задачи:

* надежность и безопасность работы системы,
* простое пользование ПО,
* достоверность выдачи информации,
* защищенный доступ к изменению данных в системе,
* создание удобного и интуитивно понятного интерфейса.

# 1 Нормативные документы

В данном курсовом проекте использованы следующие нормативные

ссылки:

ГОСТ Р 1.5-2004 Стандартизация в Российской Федерации. Стандарты национальные Российской Федерации. Правила построения, изложения, оформления и обозначения.

ГОСТ Р 1.12-2004 Стандартизация в Российской Федерации. Термины и определения.

ГОСТ Р 7.0.5-2008 СИБИД. Библиографическая ссылка. Общие

требования и правила составления.

ГОСТ Р ИСО 9000-2008 Системы менеджмента качества. Основные

положения и словарь.

ГОСТ Р ИСО/МЭК 12207-99 Информационная технология. Процессы

жизненного цикла программных средств.

ГОСТ 2.001-93 ЕСКД. Общие положения.

ГОСТ 19.103-77 ЕСПД. Обозначение программ и программных

продуктов.

ГОСТ 19.105-78 ЕСПД. Обозначение программ и программных

продуктов.

ГОСТ 19.401-78 ЕСПД. Текст программы. Требования к содержанию и

оформлению.

ГОСТ 19.101-77 ЕСПД. Виды программ и программных документов.

ГОСТ 19.106-78 Единая система программной документации (ЕСПД).

ГОСТ 51188-98. Защита информации. Испытания программных средств на наличие компьютерных вирусов.

## 2 Анализ предметной области

Предметной областью в рамках данного курсового проекта является разработка программы складского учета.

Основной деятельностью пользователей данной программой будет манипуляция данными в номенклатуре.

Номенклатура содержит следующие данные о товаре: наименование, категория, страна производитель, ставка НДС, плановая цена покупки и плановая цена реализации.

Процедура принятия продукции на склад:

* прибытие продукции на склад в сопровождении экспедитора с приходной накладной;
* проверка приходной накладной и регистрация оной в книге учета входящих документов (накладных), а также осмотр входящей продукции;
* выгрузка товара в свободное место хранения.

Отгрузка товаров со склада проходит следующие стадии:

* подача накладной на отгрузку товара получателем;
* проверка накладной и её регистрация в книге учета входящих документов;
* поиск нужной продукции и её отгрузка;
* сверка полученных товаров со списком из накладной;
* регистрация факта отгрузки товара;
* выдача получателю груза сопроводительный документ по отгрузке товара;
* отгрузка товара.

Проанализировав процедуры, происходящие на складе, будет разработана такая система, которая выполняет следующие функции:

* регистрация документов;
* поиск соответствующего товара в БД и просмотр информации о месте его хранении (номер склада);
* формирование документов отчетности.

В результате вся работа с бумагами будет проводиться с использованием компьютеров, не нужно будет возится с кучей бумаг.

При помощи ЭВМ на складе автоматизирован учет поступления и отгрузки товаров, учет входящих и исходящих документов, количественный учет. В общем объеме учетных работ эти задачи имеют значительный удельный вес. Их автоматизация позволяет сократить ручные операции, ускорить обработку информации, повысить точность учета.

Главное назначение автоматизированной системы в данном случае – повысить эффективность выполнения основных функций работников склада.

Для короткого и максимально понятного описания функционала продукта была использована формулировка целей «User story».

User story – это пользовательско-ориентированное описание целей, которые люди смогут достичь, используя разрабатываемый программный продукт, написанное повседневным языком.

## Цель системы

Система складского учета предназначена для автоматизации процессов, связанных с ведением записей хранения и транспортировки товаров.

Ожидается значительное ускорение обработки накладных на получение и отгрузку товаров, вследствие чего повысится качество работы сотрудников склада.

2) Роли:

* те, кто проверяет приходные и отгрузочные накладные, а также их регистрирует – «Контролеры»;
* те, кто непосредственно выгружают и отгружают товар ­– «Грузчики»;
* системе необходимы «Администраторы», которые будут обеспечивать её поддержку для блага других пользователей.

3) Истории:

* как грузчик я обязан заполнить таблицу со списком полученных товаров и отдать её контролёру;
* как грузчик я не имею доступ к базе данных складской системы;
* как контролёр я обязан сверить список товаров, указанных в накладной с пришедшим грузом;
* как контролёр я могу добавлять новые товары в список товаров;
* как контролёр я могу добавлять, удалять и изменять записи накладных;
* как контролёр я обязан выдать сформированный отчет по накладной экспедитору;
* как контролёр я не могу добавлять новых пользователей;
* как администратор я обязан выдать логин и пароль контроллеру при регистрации новой учетной записи пользователя;
* как администратор я имею доступ ко всей базе данных;
* как администратор при увольнении я обязан удалить свою учетную запись из системы; убрать местоимения

## 3 Разработка технического задания

## 3.1 Введение

В настоящее время в сферу складской логистики повсеместно имплементируются программные технологии, предназначенные для учета товаров, хранящихся на складе. Реализация проекта автоматизированной системы складского учета значительно облегчит работу сотрудников на складе и сократит время обработки внедряемых данных, а также обеспечит освобождение человеческих ресурсов за счет автоматизации поиска и документирования.

Основаниями для разработки являются:

* учебный план направление подготовки 09.03.03 «Прикладная информатика», профиль «Разработка и внедрение прикладных информационных систем;
* рабочая программа по дисциплине «Программная инженерия»;
* указания по распределению тем КП № 25-КТ от 23.09.2020 г. о назначении темы курсового проекта студенту очного отделения группы 19-КБ-ПИ2 Колотову И.В. Тема курсового проекта: “Разработка приложения для складского учета”.

Условное обозначение темы разработки (шифр темы) – «АИС Складской учет».

Функциональным назначением программы является

упрощение работы сотрудников склада за счет автоматизации рабочих процессов. Программа должна эксплуатироваться в профильных подразделениях на объектах заказчика. Пользователями программы должны быть сотрудники профильных подразделений объектов заказчика.

## 3.2 Требования к программе или программному изделию

## 3.2.1 Требования к функциональным характеристикам

Разрабатываемая программа должна обладать следующими функциями:

* учет пользователей системы;
* учет товаров на складе;
* формирование отчетности;
* удобный поиск информации.

Администратор должен обладать:

* добавление/редактирование/удаление данных о пользователе;
* изменение базы данных системы.

Пользователь:

* изменение базы данных системы.

Требования к организации входных данных:

* входные данные программы должны введены пользователем системы;
* только администратор имеет право редактировать данные других пользователей, складов и поставщиков.

Требования к организации выходных данных

* программа должна выводить на экран таблицы базы данных, а также сформировывать накладные и отчеты по ним.

## 3.2.2 Требования к надежности

Надежное (устойчивое) функционирование программы должно быть обеспечено выполнением заказчиком совокупности организационно-технических мероприятий, перечень которых приведен ниже:

* организация бесперебойного питания технических средств;
* использование [лицензионного](javascript:void(9609)) программного обеспечения;
* регулярное выполнение рекомендаций Министерства труда и социального развития РФ, изложенных в Постановлении от 23 июля 1998 г. «Об утверждении межотраслевых типовых норм времени на работы по сервисному обслуживанию ПЭВМ и оргтехники и сопровождению программных средств»;
* регулярное выполнение требований ГОСТ 51188-98.

Время восстановления после отказа, вызванного сбоем электропитания технических средств (иными внешними факторами), не фатальным сбоем (не крахом) операционной системы, не должно превышать пяти минут при соблюдениях условий эксплуатации технических и программных средств.

Время восстановления после отказа, вызванного неисправностью технических средств, фатальным сбоем (крахом) операционной системы, не должно превышать времени, требуемого на устранение неисправностей технических средств и переустановки программных средств.

Отказы программы возможны вследствие некорректных действий оператора (пользователя) при взаимодействии с операционной системой. Во избежание возникновения отказов программы по указанной выше причине следует обеспечить работу пользователя без предоставления ему административных привилегий.

Программа будет прекрасно работать от плюс 5 до плюс 35 °C при относительной влажности 90 % и атмосферном давлении 462 мм.рт.ст., поскольку такие условия приблизительно соответствуют условиям эксплуатации современных компьютеров непромышленного исполнения.

Минимальное количество персонала, требуемого для работы программы, должно составлять не менее 2 штатных единиц – системный администратор и пользователь программы – оператор.

Администратор должен иметь высшее профильное образование и сертификаты компании-производителя операционной системы. В перечень задач, выполняемых системным администратором, должны входить:

* задача поддержания работоспособности технических средств;
* задачи установки (инсталляции) и поддержания работоспособности системных программных средств – операционной системы;
* задача установки (инсталляции) программы.

Пользователь программы (оператор) должен обладать практическими навыками работы с графическим пользовательским интерфейсом операционной системы.

Персонал должен быть аттестован на II квалификационную группу по электробезопасности (для работы с конторским оборудованием).

Персонал, не имеющий II квалификационной группы по электробезопасности, не имеет права даже близко подходить к ПЭВМ и конторскому оборудованию.

## 3.2.3 Требования к составу и параметрам технических средств

В состав технических средств должен входить IBM-совместимый персональный компьютер (ПЭВМ), включающий в себя:

* процессор Intel Core i5 с тактовой частотой 2,9 - ГГц;
* оперативная память объемом 8 Гб;
* жесткий диск стоимостью не менее 3500 рублей и объемом не менее 1 Тб.

## 3.2.4 Требования к информационной и программной совместимости

Требования к информационным структурам (файлов) на входе и выходе, а также к методам решения не предъявляются.

Исходные коды программы должны быть реализованы на языке C#. В качестве интегрированной среды разработки программы должна быть использована среда Microsoft Visual Studio.

Разрабатываемая программа совместима только с операционной системой Windows. Рекомендуется использовать Windows 10 с последними обновлениями

Вместе с программой также должна быть установлено лицензионное приложение Microsoft SQL Server.

Требования к защите информации и программ не предъявляются.

Требования к маркировке и упаковке не рассматриваются в данном курсовом проекте.

## 3.2.5 Специальные требования

Программа должна обеспечивать взаимодействие с пользователем (оператором) посредством графического пользовательского интерфейса, разработанного согласно рекомендациям компании-производителя операционной системы.

## 3.2.6 Технико-экономические показатели

Технико-экономические показатели в данном курсовом проекте не учитываются.

## 3.2.7 Стадии и этапы разработки

Разработка должна быть проведена в три стадии:

1. разработка технического задания;
2. проектирование:
   1. диаграмма вариантов использования;
   2. диаграмма классов;
   3. диаграмма состояний;
   4. диаграмма компонентов;
   5. диаграмма размещения.
3. реализация
   1. idef0 диаграмма;
   2. DFD;
   3. ERD;
   4. диаграмма классов;
   5. диаграмма компонентов;
   6. диаграмма размещения.
4. внедрение.

## 3.2.8 Этапы разработки

Для разработки программного продукта была использована каскадная модель жизненного цикла:

* + 1. анализ предметной области;
    2. проектирование;
    3. разработка программы;
    4. внедрение.

## Проектирование классов

## Проектирование ПО: Выбор языка, среды

## Разработка пользовательского интерфейса

## Концептуальная модель

## Диаграммы UML

концептуальная модель, диаграммы.

Общая характеристика склада как объекта хозяйственной деятельности. В данном курсовом проекте в качестве предметной области используется склад фирмы. Форм отчетности и действий, производимых на складе, весьма широк, но основной задачей является учет товара. Основные функции выполняют заведующий складом, в обязанности которого входит проведение ревизий и составление оборотов за период, и кладовщик, который осуществляет инвентаризацию и прием/выдачу товара. Обоснование актуальности разработки объектно-ориентрованной модели информационной подсистемы. Актуальность разработки объектно-ориентированной модели информационной подсистемы для учета движения товаров на складе обуславливается применением языка UML (Unified Modeling Language). Унифицированный язык моделирования одновременно является простым и мощным средством моделирования, который может быть эффективно использован для построения концептуальных, логических и графических моделей сложных систем самого различного целевого назначения. Язык UML основан на некотором числе базовых понятий, которые могут комбинироваться и расширяться таким образом, что специалисты объектного моделирования получают возможность самостоятельно разрабатывать модели больших и сложных систем в самых различных областях приложений. При этом одним из основных принципов построения моделей сложных систем является принцип абстрагирования, который предписывает включать в модель только те аспекты проектируемой системы, которые имеют непосредственное отношение к выполнению системой своих функций или своего целевого предназначения. При этом все второстепенные детали опускаются, чтобы чрезмерно не усложнять процесс анализа и исследования полученной модели. Формулировка задачи проектирования. Задачей курсового проекта является разработка объектно-ориентированной модели информационной подсистемы для учета движения товаров на складе фирмы с использованием языка UML. В качестве среды разработки информационной подсистемы был использован программный продукт Rational Rose 2000 Enterprise v6.5. Выводы В результате проектирования должна быть получена объектно-ориентированная модель информационной подсистемы для учета движения товаров на складе. Модель будет включать в себя: - диаграмму вариантов использования; - диаграмму последовательности; диаграмму сотрудничества; диаграмму классов; диаграмму состояний; - диаграмму компонентов; - диаграмму размещения; - программный код на языке С++. 2 Создание диаграммы прецедентов Диаграммой прецедентов, или использования (use case diagram) называется диаграмма, на которой показана совокупность прецедентов и актеров, а также отношения (зависимости, обобщения и ассоциации) между ними. Она позволяет выделить внешние системы, контактирующие с системой, основные процессы и их взаимосвязь. Диаграммы прецедентов дают возможность выделить функциональную структуру системы, не вдаваясь в детали ее реализации. Кроме того, производится предварительное выделение объектов системы и их классификация. На основании построенной модели составляется план разработки системы. Элементы диаграммы: - вариант использования - это логическое описание определенной части деятельности системы. Он не представляет собой четкую конструкцию, которую можно напрямую реализовать в программном коде. Каждый вариант использования определяет последовательность действий, которые должны быть выполнены проектируемой системой при взаимодействии ее с соответствующим актером. - актер представляет собой любую внешнюю по отношению к моделируемой системе сущность, которая взаимодействует с системой и использует ее функциональные возможности. Актеры используются для моделирования внешних по отношению к проектируемой системе сущностей, которые взаимодействуют с системой и используют ее в качестве отдельных пользователей. отношения ассоциации устанавливают, какую конкретную роль играет актер при взаимодействии с экземпляром варианта использования. Рисунок 2.1 - Диаграмма прецедентов В качестве актеров на диаграмме (рисунок 2.1) используются объекты: - «zav\_sklad» (заведующий складом), который управляет вариантами использования: - «oborot\_mes» (оборот за месяц); «reviziya» (ревизия); «klad» (кладовщик), управляющий следующими вариантами использования: - «get\_tovar» (принять товар); - «send\_tovar» (отправить товар); - «inventar» (инвентаризация). Между вариантами использования и действующими лицами используется вязь коммуникации (communication). Направление стрелки позволяет понять, кто инициирует коммуникацию. Для построения остальных диаграмм, выбран прецедент «get\_tovar» (принять товар), который описывает получение нового товара от поставщика и создание карточки складского учета. 3 Создание диаграммы последовательности Диаграмма последовательности (sequence diagram) - это диаграмма взаимодействий, акцентирующая внимание на временной упорядоченности сообщений. Она отражают поток событий, происходящих в рамках варианта использования. Конкретный экземпляр потока событий называется сценарием. В диаграммах последовательности действий взаимодействие объектов в системе происходит посредством приема и передачи сообщений объектами-клиентами и обработки этих сообщений объектами-серверами. При этом в разных ситуациях одни и те же объекты могут выступать и в качестве клиентов, и в качестве серверов. На диаграмме последовательности объект изображается в виде прямоугольника на вершине пунктирной вертикальной линии. Эта вертикальная линия называется линией жизни (lifeline) объекта. Она представляет собой фрагмент жизненного цикла объекта в процессе взаимодействия. Каждое сообщение изображается в виде стрелки между линиями жизни двух объектов. Сообщения появляются в том порядке, как они показаны на странице, сверху вниз. Каждое сообщение помечается как минимум именем сообщения; при желании можно добавить также аргументы и некоторую управляющую информацию и, кроме того, показать самоделегирование (self-delegation) -сообщение, которое объект посылает самому себе, при этом стрелка сообщения указывает на ту же самую линию жизни. В данной модели для создания диаграммы последовательности был использован вариант использования «get\_tovar» (принять товар), взятый из предыдущей диаграммы прецедентов (рисунок 3.1). Рисунок 3.1 - Диаграмма последовательности «get\_tovar» (принять товар) На данную диаграмму помещены следующие объекты: - «klad» (кладовщик) - действующее лицо; - «Add/Select Tovar Form» - содержит форму ввода или выбора товара; «Add/Select Postav Form» - содержит форму ввода или выбора поставщика товара; «Card Sklad\_Uchet» - форма карты складского учета, которая создается после ввода всех данных и является итоговым документом; «DataBase» - содержит информацию о поставщиках и товарах, на основании информации этого объекта формируется карта складского учета - «Open» - открыть форму; - «Cancel» - отмена действия; «Query to DataBase» - запрос к базе данных на выбор товара; «Answer from DataBase» - наименование товара; «Query to DataBase on generation Sklad\_Uchet card» - запрос к базе данных на выбор поставщика и генерацию карты складского учета; «Generate» - карта складского учета. После создания объектов и сообщений между ними было выполнено соотнесение объектов с классами, а сообщений с операциями. Все названия объектов и сообщений совпадают с названиями классов и операций соответственно. Выводы 1. На диаграмме последовательности «get\_tovar» (принять товар), размещены пять объектов и девять сообщений между ними. 2. Каждый объект был соотнесен с классом, а сообщение с операцией. 4 Создание диаграммы сотрудничества Вторым видом диаграммы взаимодействия является кооперативная диаграмма (collaboration diagram). Подобно диаграммам последовательности, кооперативные диаграммы отображают поток событий через конкретный сценарий варианта использования. Кооперативные диаграммы заостряют внимание на взаимосвязях вообще, то есть на них отражается наличие сообщений от клиентов к серверам. Диаграмма показывает взаимодействие между объектами, а не классами, то есть является мгновенным снимком объектов системы в некотором состоянии. Ведь объекты, в отличие от созданных на этапе проектирования классов, создаются и уничтожаются на всем протяжении работы программы. И в каждый момент имеется конкретная группа объектов, с которыми осуществляется работа. При создании диаграммы сотрудничества располагают участвующие во взаимодействии объекты в виде вершин графа. Связи между этими объектами, отраженные на диаграмме последовательности, дополняются сообщениями, которые объекты принимают и посылают. Это дает аналитику ясное визуальное преставление о потоке управления в контексте структурной организации кооперирующихся объектов. Согласно созданной выше диаграмме «get\_tovar» (принять товар), была создана диаграмма сотрудничества (рисунок 4.1). На диаграмме расположены объекты: - «sklad»; - «Add/Select Tovar Form»; «Add/Select Postav Form»; «Card Sklad\_Uchet»; «DataBase». Назначение данных объектов аналогично соответствующим объектам диаграммы последовательности (рисунок 3.1). Рисунок 4.1 - Диаграмма сотрудничества для варианта использования «get\_tovar» (принять товар) Выводы Была создана диаграмма сотрудничества «get\_tovar» (принять товар), на которой представлена вся та информация, которая была и на диаграмме последовательности, но кооперативная диаграмма по-другому описывает поток событий. Из нее легче понять связи между объектами, однако труднее разобраться в последовательности событий. 5 Создание диаграммы классов Диаграммы классов (class diagram) позволяет создавать логическое представление системы, на основе которого создается исходный код описанных классов. Значки диаграммы позволяют отображать сложную иерархию систем, взаимосвязи классов (сlasses) и интерфейсов (interfaces). На диаграммах классов изображаются также атрибуты классов, операции классов и ограничения, которые накладываются на связи между классами Данный тип диаграмм противоположен по содержанию диаграмме сотрудничества, на которой отображаются объекты системы. Каждый класс на диаграмме выглядит в виде прямоугольника, разделенного на три части. В первой содержится имя класса, во второй - его атрибуты. В последней части содержатся операции класса, отражающие его поведение (действия, выполняемые классом). Обычно для описания системы создают несколько диаграмм классов. На одних показывают некоторое подмножество классов и отношения между классами подмножества. На других отображают то же подмножество, но вместе с атрибутами и операциями классов. Третьи соответствуют только пакетам классов и отношениям между ними. По умолчанию существует одна диаграмма классов, называемая Главной (Main), на которой показывают пакеты классов модели (рисунок 5.1). Рисунок 5.1 - Главная диаграмма классов информационной подсистемы Внутри каждого пакета также имеется «главная диаграмма», включающая в себя все классы этого пакета (рисунок 5.2, 5.3). Рисунок 5.2 - Диаграмма классов «form» Рисунок 5.3 - Диаграмма классов «DataBase» После создания главой диаграммы классов создается диаграмма классов для сценария «get\_tovar» (принять товар), на которой отражаются все классы, атрибуты и связи между ними (рисунок 5.4). Рисунок 5.4 - Диаграмма классов «get\_tovar» (принять товар) Выводы 1. На диаграмме классов «get\_tovar» (принять товар) были представлены все классы и взаимосвязи между ними. 2. Указанная множественность (multiplicity) показывает, сколько экземпляров одного класса взаимодействуют с помощью этой связи с одним экземпляром другого класса в данный момент времени. Так же были добавлены атрибуты и методы в некоторые классы. 6 Создание диаграммы состояний для классов Диаграммы состояний определяют все возможные состояния, в которых может находиться конкретный объект, а также процесс смены состояний объекта в результате наступления некоторых событий. На диаграмме имеются два специальных состояния - начальное (start) и конечное (stop). Процессы, происходящие в тот момент, когда объект находится в определенном состоянии, называются действиями (actions). С состоянием можно связывать следующие данные: деятельность, входное действие, выходное действие и событие. Входное действие (entry action) - это поведение, которое выполняется, когда объект переходит в данное состояние. Входное действие также показывают внутри состояния, его обозначению предшествуют слово entry (вход) и двоеточие. Выходное действие (exit action) подобно входному действию. Однако оно осуществляется как составная часть процесса выхода из данного состояния. Как и входное, выходное действие является непрерываемым. Переходом (transition) называется перемещение объекта из одного состояния в другое. На диаграмме все переходы изображают в виде стрелки, начинающейся на первоначальном состоянии и заканчивающейся последующим. У перехода существует несколько спецификаций. Они включают события, аргументы, ограждающие условия, действия и посылаемые события. Событие (event) - это то, что вызывает переход из одного состояния в другое. Событие размещают на диаграмме вдоль линии перехода. Большинство переходов должны иметь события, так как именно они, прежде всего, заставляют переход осуществиться. Действие (action) является непрерываемым поведением, осуществляющимся как часть перехода. Входные и выходные действия показывают внутри состояний, поскольку они определяют, что происходит, когда объект входит или выходит из состояния. Диаграммы состояний не надо создавать для каждого класса, они применяются только в сложных случаях. Если объект класса может существовать в нескольких состояниях и в каждом из них ведет себя по-разному, для него может потребоваться диаграмма состояний. В данном курсовом проекте диаграмма состояний создается для варианта использования «get\_tovar» (принять товар). Она представлена на рисунке 6.1. Рисунок 6.1 - Диаграмма состояний для варианта использования «get\_tovar» (принять товар) Выводы На диаграмме состояний расположены следующие состояния: - начальное состояние «Start»; - конечное «exit» состояние; «Initialization» (инициализация); «SomeStop» (выполнение приостановлено); «Cancel» (отменен); «Get» (выполнен). Для каждого из состояний созданы следующие действия: - «StoreDate» (сохранить дату) на входе для состояния «Initialization» (инициализация); - «Info Tovar» (собрать информацию о товаре и поставщиках); «Add» (добавить к набору товаров); «Card Ucheta» (карта учета) - на выходе формируется складская карта учета. 7 Создание диаграммы компонентов Диаграммы компонентов (component diagram) предназначены для распределения классов и объектов по компонентам при физическом проектировании системы. На них изображены компоненты программного обеспечения и связи между ними. При этом на такой диаграмме выделяют два типа компонентов: исполняемые компоненты и библиотеки кода. Часто данный тип диаграмм называют диаграммами модулей. При проектировании больших систем может оказаться, что система должна быть разложена на несколько сотен или даже тысяч компонентов, и этот тип диаграмм позволяет не потеряться в обилии модулей и их связей. Каждый класс модели (или подсистема) преобразуется в компонент исходного кода. После создания они сразу добавляются к диаграмме компонентов. У системы может быть несколько диаграмм компонентов в зависимости от числа подсистем или исполняемых файлов. Каждая подсистема является пакетом компонентов. В общем случае пакеты - это совокупности компонентов. Диаграммы компонентов применяются теми участниками проекта, кто отвечает за компиляцию системы. Из нее видно, в каком порядке надо компилировать компоненты, а также какие исполняемые компоненты будут созданы системой. На такой диаграмме показано соответствие классов реализованным компонентам. Она нужна там, где начинается генерация кода. На рисунке 7.1 показаны компоненты пакета «Form». Они содержат классы пакета «Form» логического представления системы. Рисунок 7.1 - Диаграмма компонентов пакета «Form» На рисунке 7.2 показаны компоненты пакета «DataBase». Они содержат классы пакета «DataBase» логического представления системы. Рисунок 7.2 - Диаграмма компонентов пакета «DataBase» Общая диаграмма компонентов для рассматриваемого варианта использования «get\_tovar» (принять товар) на рисунке 7.3. Рисунок 7.2- Диаграмма компонентов для варианта использования «get\_tovar» (принять товар) Выводы 1. Так как система разрабатывается на языке C++, то у каждого класса имеется свой собственный заголовочный файл и файл с расширением \*.cpp. 2. Для каждого класса была создана спецификация пакета и тело пакета. Они соединены с помощью связей dependency. 8 Создание диаграммы размещения Диаграмма размещения (deployment diagram) отражает физические взаимосвязи между программными и аппаратными компонентами системы. Она показывает размещение объектов и компонентов в распределенной системе. Каждый узел на диаграмме размещения представляет собой некоторый тип вычислительного устройства - в большинстве случаев часть аппаратуры. Эта аппаратура может быть простым устройством или датчиком, а может быть и мэйнфреймом. Диаграмма размещения показывает физическое расположение сети и местонахождение в ней различных компонентов. Для каждой модели создается только одна такая диаграмма, отображающая процессоры (Processor), устройства (Device) и их соединения. Построенная диаграмма размещения показана на рисунке 8.1. Рисунок 8.1 - Диаграмма размещения Как видно из диаграммы, клиентская и серверная программы будут работать на разных местах. Через закрытую сеть будет осуществляться их сообщение. С клиентской частью соединен принтер, так как именно клиентская программа печатает каты складского учета.

# Заключение

# Список использованных источников;