Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет   
информатики и радиоэлектроники»

Факультет инженерно-экономический

Кафедра экономической информатики

Дисциплина «Программирование сетевых приложений»

|  |  |
| --- | --- |
|  | «К ЗАЩИТЕ ДОПУСТИТЬ» |
|  | Руководитель курсового проекта  Ассистент кафедры ЭИ  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.П. Лыщик |
|  | \_\_\_.\_\_\_\_.2022 |

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовому проекту

на тему:

**«Подсистема учета и регистрации поступления товаров на склад логистической компании»**

БГУИР КР 1-40 01 02-08 015 ПЗ

|  |  |
| --- | --- |
|  | Выполнил студент группы 073601  Примакович Людмила Васильевна  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (подпись студента) |
|  | Курсовая работа представлена на проверку \_\_\_.\_\_\_\_.2022  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (подпись студента)) |

Минск 2022

**РЕФЕРАТ**

**Оглавление**

[Перечень условных обозначений, символов и терминов 7](#_Toc120533290)

[Введение 9](#_Toc120533291)

[1 АНАЛИЗ И МОДЕЛИРОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ ПРОГРАММНОГО СРЕДСТВА 11](#_Toc120533292)

[**1.1 Описание работы склада** 11](#_Toc120533293)

[**1.2 Разработка функциональной модели регистрации и учета поступления товаров на склад** 12](#_Toc120533294)

[**1.3 Анализ требований к разрабатываемому программному средству. Спецификация функциональных требований** 17](#_Toc120533295)

[**1.4 Разработка информационной модели приема и регистрации поставок** 19](#_Toc120533296)

[**1.5 UML-модели представления программного средства и их описание** 21](#_Toc120533297)

[2 ПРОЕКТИРОВАНИЕ И КОНСТРУИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА 25](#_Toc120533298)

[**2.1 Постановка задачи** 25](#_Toc120533299)

[**2.2 Архитектурные решения** 25](#_Toc120533300)

[**2.3 Описание алгоритмов, реализующих ключевую бизнес-логику разрабатываемого программного средства** 25](#_Toc120533301)

[**2.4 Проектирование пользовательского интерфейса** 25](#_Toc120533302)

[**2.5 Обоснование выбора компонентов и технологий для реализации программного средства** 26](#_Toc120533303)

[3 ТЕСТИРОВАНИЕ И ПРОВЕРКА РАБОТОСПОСОБНОСТИ ПРОГРАММНОГО СРЕДСТВА 27](#_Toc120533304)

[4 ИНСТРУКЦИЯ ПО РАЗВЕРТЫВАНИЮ ПРИЛОЖЕНИЯ И СКВОЗНОЙ ПРИМЕР, НАЧИНАЯ ОТ АВТОРИЗАЦИИ, ДЕМОНСТРИРУЯ РЕАЛИЗАЦИЮ ВСЕХ ВАРИАНТОВ ИСПОЛЬЗОВАНИЯ 28](#_Toc120533305)

[ЗАКЛЮЧЕНИЕ 29](#_Toc120533306)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 30](#_Toc120533307)

[Приложение А 31](#_Toc120533308)

[(обязательное) 31](#_Toc120533309)

[Отчет о проверке на заимствования в системе «Антиплагиат» 31](#_Toc120533310)

[Приложение Б 32](#_Toc120533311)

[(обязательное) 32](#_Toc120533312)

[Листинг кода алгоритмов, реализующих бизнес-логику 32](#_Toc120533313)

[Приложение В 33](#_Toc120533314)

[(обязательное) 33](#_Toc120533315)

[Листинг скрипта генерации базы данных 33](#_Toc120533316)

[ВЕДОМОСТЬ ДОКУМЕНТОВ КУРСОВОГО ПРОЕКТА 37](#_Toc120533317)

# **Перечень условных обозначений, символов и терминов**

|  |  |
| --- | --- |
| БД (база данных) | – представленная в объективной форме совокупность самостоятельных материалов, систематизированных таким образом, чтобы эти материалы могли быть найдены и обработаны с помощью электронной вычислительной машины |
|  |  |
| Информационная система | – система, предназначенная для хранения, поиска и обработки информации, и соответствующие организационные ресурсы (человеческие, технические, финансовые), которые обеспечивают и распространяют информацию |
| Нормальная форма | – свойство отношения в реляционной модели данных, характеризующее его с точки зрения избыточности, потенциально приводящей к логически ошибочным результатам выборки или изменения данных |
| Среда выполнения | – вычислительное окружение, необходимое для выполнения компьютерной программы и доступное во время выполнения компьютерной программы |
| СУБД (система управления базами данных) | – совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных |
| ТЗ (техническое задание) | – документ, содержащий требования заказчика к объекту разработки, определяющий порядок и условия её проведения |
| *API* (*application programming interface*) | – описание способов (набор классов, процедур, функций, структур или констант), которыми одна компьютерная программа может взаимодействовать с другой программой |
| *HTML* (*HyperText Markup Language*) | – стандартизированный язык разметки документов во Всемирной паутине |
| *HTTP* (*HyperText Transfer Protocol*) | – протокол прикладного уровня передачи данных изначально – в виде гипертекстовых документов в формате *html*, в настоящий момент используется для передачи произвольных данных. |
| *IDE* (*Integrated development environment*) | – комплекс программных средств, используемый программистами для разработки программного обеспечения |
| *IDEF0* | – методология функционального моделирования (англ. *function modeling*) и графическая нотация, предназначенная для формализации и описания бизнес-процессов |
| *Java* | – строго типизированный объектно-ориентированный язык программирования общего назначения, разработанный компанией *Sun* *Microsystems* |
|  |  |
| *SQL* (*structured query language*) | – язык структурированных запросов, декларативный язык программирования, применяемый для создания, модификации и управления данными в реляционной базе данных, управляемой соответствующей системой управления базами данных |
| *MySQL Server* | – свободная реляционная система управления базами данных |
| *MySQL Workbench CE 8.0* | – инструмент для визуального проектирования баз данных, интегрирующий проектирование, моделирование, создание и эксплуатацию БД в единое бесшовное окружение для MySQL. |
| *Sybase ASE* (*Adaptive Server Enterprise*) | – реляционная система управления базами данных компании *SAP*, одна из СУБД, использующая в качестве основного процедурного *SQL*-расширения язык *Transact-SQL* |
| *UML* (*Unified Modeling Language*) | – язык графического описания для объектного моделирования в области разработки программного обеспечения, для моделирования бизнес-процессов, системного проектирования и отображения организационных структур |
| *URL* (*Uniform Resource Locator*) | – система унифицированных адресов электронных ресурсов, или единообразный определитель местонахождения ресурса |
| *Usability* | – способность продукта быть понимаемым, изучаемым, используемым и привлекательным для пользователя в заданных условиях |
|  |  |

# **Введение**

Сегодня информационные технологии оказывают большое влияние на протекание процессов ведения бизнеса. Успех деятельности любого предприятия зависит от оснащенности компьютерами, телефонами, современным ПО и т.д. Эти устройства позволяют общаться на больших расстояниях, а также производить обмен файлами.

Очень легко отслеживать и вести учет записей и бизнес-процессов на любом предприятии благодаря современным технологиям, которые позволяют автоматизировать процесс обмена информацией и ускорить процесс ее обработки. Это актуально в век высоких технологий, ведь от скорости работы с информацией сегодня зависит успех и эффективность работы предприятия, а человеку было бы очень трудоемко и долго фиксировать все необходимые данные и сортировать их вручную.

К примеру, на склады логистических компаний постоянно поступают различного рода товары. Отслеживать все соответствия привозимых товаров требованиям склада проблематично. Также существует проблема надлежащего хранения информации о поставках и поставщиках, корректного заполнения сопроводительных документов. Автоматизация процессов приемки и регистрации поступлений на склад значительно упрощает задачу.

Для анализа поставщиков необходимы большие объемы информации об их более ранних поставках. Вся эта информация должна быть хорошо структурирована и отфильтрована. Многочисленные таблички Excel с данными о поставках, поставщиках, продукции, отказах, доходах по складам и т.д. при большом количестве поставщиков и поставок не справляются с поставленной задачей. Система управления взаимодействия поставщиков и работников склада (кладовщиков) необходима на предприятии, чтобы ускорить процесс приемки и сделать его более надежным.

Автоматизация складского учёта влияет на скорость и качество выполнения основных складских процессов, приводит к совершенствованию систем управления и регулирования материальных и информационных потоков на складе. Это достигается путём внедрения современного программного обеспечения и компьютерного оборудования на предприятии.

Разработка автоматизированной системы учета и регистрации поступлений товаров на склад дает толчок к развитию новых форм взаимодействия поставщиков и работников склада, упрощает и ускоряет их взаимодействие, что ведет к осуществлению указанных процессов на более высоком уровне, а также позволяет легко выявлять недобросовестных и надежных поставщиков, что, в свою очередь, оказывает немаловажное влияние на разработку и осуществление стратегии предприятия.

Объектом исследования являются процессы приемки товара на склад и регистрации поступления товарно-материальных ценностей (ТМЦ).

Повышение качества и эффективности управления приемкой ТМЦ путем автоматизации процессов учета поступивших товаров и регистрации новых поставок внутри системы и управления информацией о поставщиках, складах и кладовщиках является целью этого курсового проекта.

Для того, чтобы достичь данной цели, необходимо выполнить ряд задач:

* изучить принципы осуществления приемки товаров на склад и ведения учетной документации;
* разработать достаточный функциональный набор, необходимый для эффективной работы программного средства;
* составить схемы алгоритмов, реализующих основной функционал программного средства;
* выполнить проектирование программного средства, используя UML-диаграммы и методологию IDEF0;
* спроектировать базу данных хранимой информации;
* реализовать серверную часть приложения, которая будет реализовывать бизнес логику, и будет выполнять работу с базой данных;
* реализовать клиентскую часть приложения, с удобным интерфейсом пользователя;
* протестировать полученное программное средство и убедиться, что оно работает корректно и эффективно.

Курсовой проект выполнен самостоятельно, проверен в системе «Антиплагиат». Процент оригинальности составляет %. Цитирования обозначены ссылками на публикации, указанными в «Списке использованных источников». Скриншот проверки приведен в приложении А.

# **1 АНАЛИЗ И МОДЕЛИРОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ ПРОГРАММНОГО СРЕДСТВА**

## **1.1 Описание работы склада**

Успешная работа предприятия складывается из суммарного воздействия различных факторов и грамотного выполнения ключевых функций. При этом стоит отметить, что правильный учёт товара можно отнести к одному из основных условий стабильной работы компании. Без учёта ТМЦ, размещённых на складе, трудно обеспечить их сохранность. Перед тем как доверить кладовщику материальные запасы, с ним, как правило, заключается договор. В нем описываются виды работ, которые выполняет сотрудник, и степень ответственности в случае потери или повреждения продукции, принимаемой на склад и хранящейся на складе. Грамотно организованный процесс учёта поступающих и хранящихся на складе материалов — это очень важный и нужный сегмент деятельности организации [1].

Совокупность работ, выполняемых на различных складах, примерно одинакова. Это объясняется тем, что в разных логистических процессах склады выполняют схожие функции: приемка товаров от поставщиков; временное размещение и хранение материальных запасов; преобразование материальных потоков (транспортировка, складирование); предоставление информации о движении товаров. Однако организация работы на конкретном складе имеет ряд особенностей, которые зависят как от размера склада, так и от его предназначения. Приёмка товаров на склад является важной составной частью технологического процесса. Она осуществляется материально ответственными лицами в соответствии с внутренними правилами приемки.

Приёмка материалов на складах организации заключается в проверке соответствия поступивших ТМЦ данным, указанным в товарно-транспортной накладной (ТТН) поставщика, спецификации, описи, упаковочных ярлыков и др., а также условиям договора. Отсутствие некоторых из данных документов не приостанавливает приёмки: составляется акт о фактическом наличии. В этих случаях в акте делается отметка об отсутствии документов [2].

В общем виде процесс работы склада можно изобразить наглядно (см. рисунок 1.1). На нем видно, что одними из основных проблем организации склада является необходимость грамотного снабжения запасами, т.е. сокращение времени ожидания отгрузки, и надлежащего контроля поставок, т.е. сверка с заявленным количеством и качеством.



Рисунок 1.1 – Логический процесс работы склада

В связи с этим процедура регистрации поступлений на склад и процесс ведения складского учета все чаще и чаще подвергается различного рода автоматизации.

## **1.2 Разработка функциональной модели регистрации и учета поступления товаров на склад**

Основным процессом предметной области курсового проекта является регистрация и учет поступления товара. Данный процесс имеет множество особенностей и включает в себя формальности, требует выполнения всех инструкций и требований, а также заслуживает отдельного внимания. А значит, существует необходимость изложить все пункты и действия всех сторон предметной области.

Методология IDEF0 используется для создания функциональной модели, которая представляет собой структурированное изображение функций производственной системы или среды, а также информации и объектов, которые связывают эти функции. Отличительной особенностью IDEF0 является её акцент на соподчинённость объектов [3].

Результат анализа предметной области можно представить с помощью функциональной модели процесса регистрации и учета поступления товаров на склад. На рисунке 1.2 представлена контекстная диаграмма верхнего уровня.



Рисунок 1.2 – Контекстная диаграмма

Затем представлена декомпозиция приведенной диаграммы, которая состоит из трех блоков (см. рисунок 1.3):

1. Оформить ТТН.
2. Принять товар.
3. Хранить товар на складе.

Декомпозиция – это разделение одной крупной цели на задачи для успешного достижения этой самой цели; простыми словами, декомпозицию применяют для продуктивного распределения времени и ресурсов и чтобы не испытывать страх перед огромной задачей[4].

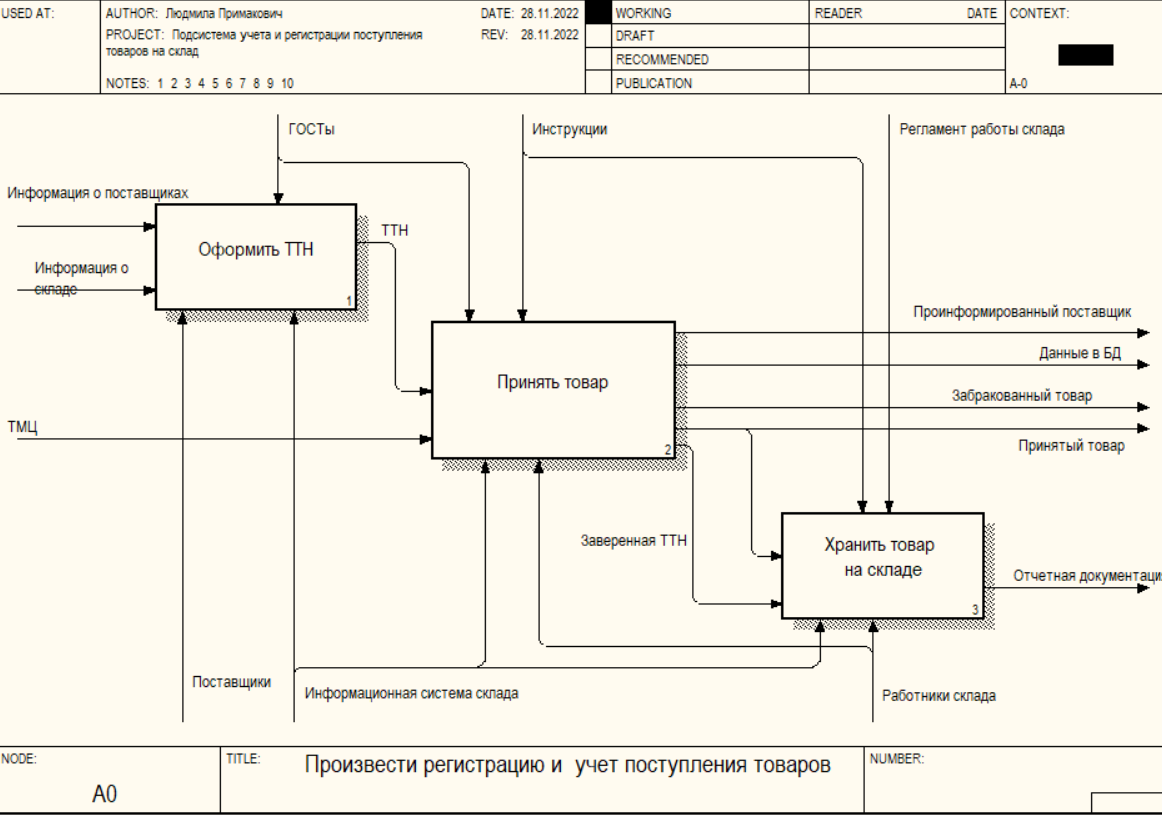


Рисунок 1.3 – Декомпозиция контекстной диаграммы

Процесс «Оформить ТТН» разбивается на три функциональных блока (см. рисунок 1.4):

1. Оформить данные о поставщике.
2. Внести данные о товаре.
3. Выбрать адрес поставки.

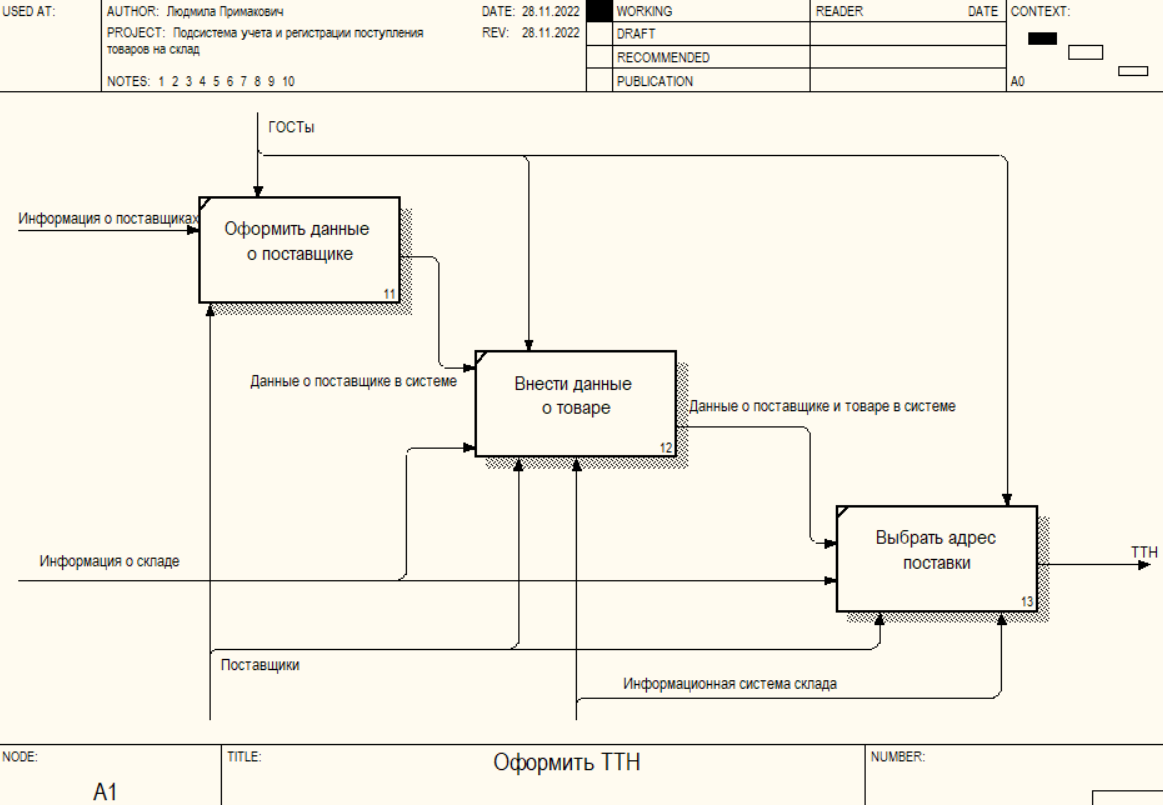


Рисунок 1.4 – Декомпозиция блока «Оформить ТТН»

Этап «Принять товар» разбит на шесть функциональных блоков (см. рисунок 1.5):

1. Проверить ТТН.
2. Проверить ТМЦ.
3. Сообщить поставщику результаты проверки.
4. Заверить ТТН.
5. Передать ТМЦ на хранение.
6. Внести данные в БД.

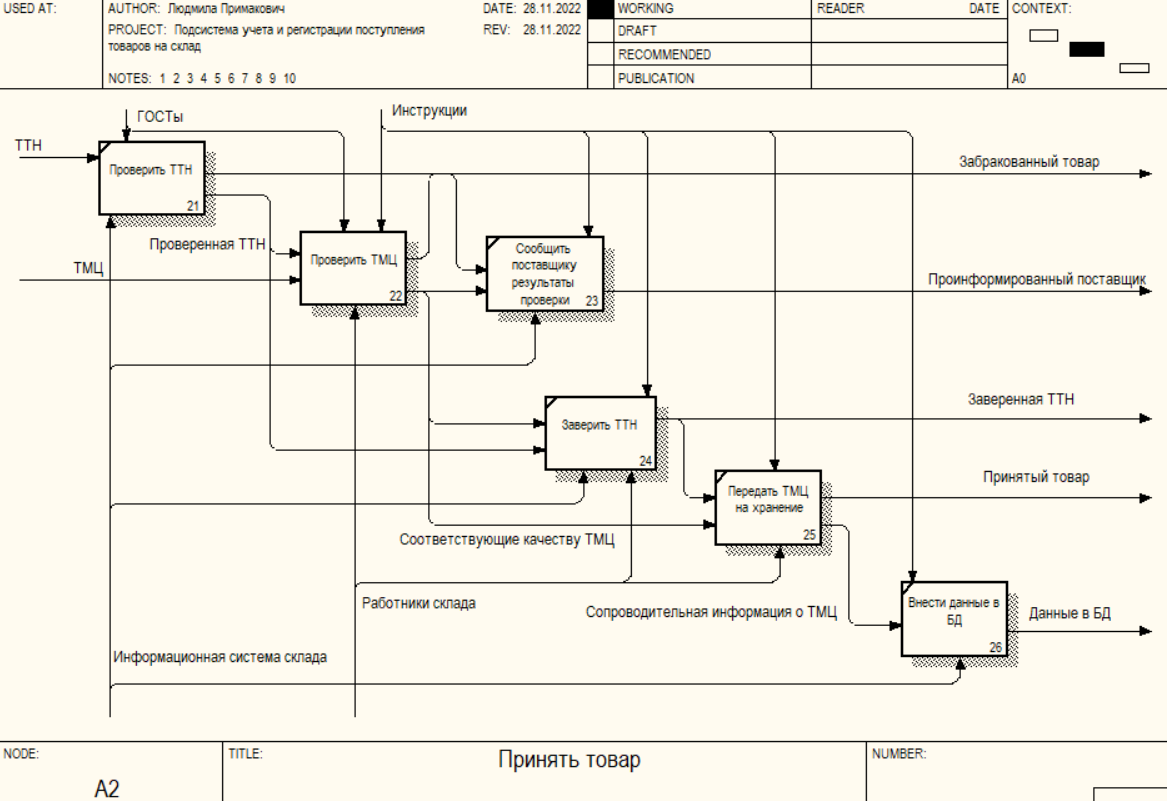


Рисунок 1.5 – Декомпозиция блока «Принять товар»

Данная декомпозиция наиболее полно описывает основную логику рассматриваемой предметной области курсового проекта и является наиболее развернутой декомпозицией всей функциональной модели. Этот факт дает нам право говорить, что именно декомпозиция блока «Принять товар» будет определять опорную логику разрабатываемого впоследствии программного проекта.

Декомпозиция блока «Проверить ТМЦ» данной диаграммы включает в себя два блока (см. рисунок 1.6):

1. Проверить соответствие качеству.
2. Проверить соответствие срокам.

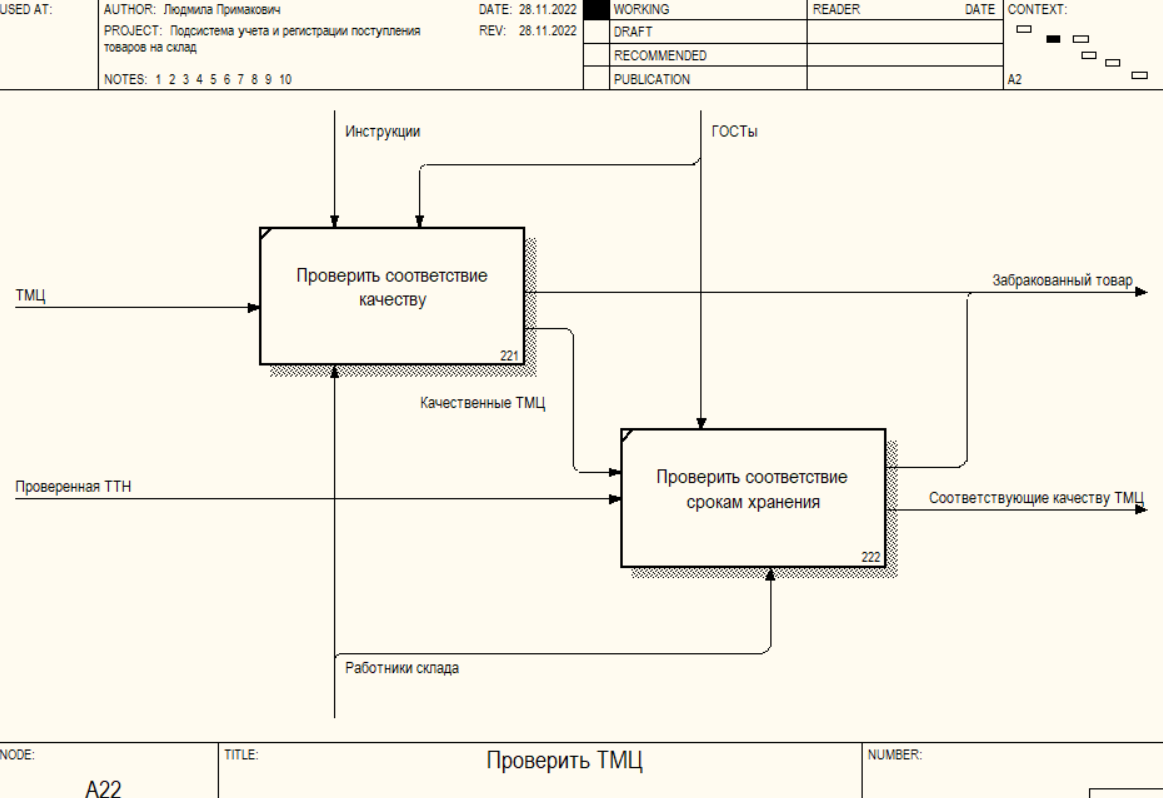


Рисунок 1.6 – Декомпозиция блока «Проверить ТМЦ»

Возвращаясь к декомпозиции контекстной диаграммы, блок «Хранить товар на складе» разбивается на четыре функциональных блока (смотри рисунок 1.7):

1. Разместить товар на складе.
2. Перемещать товар между корпусами.
3. Произвести инвентаризацию.
4. Обновить данные в БД.

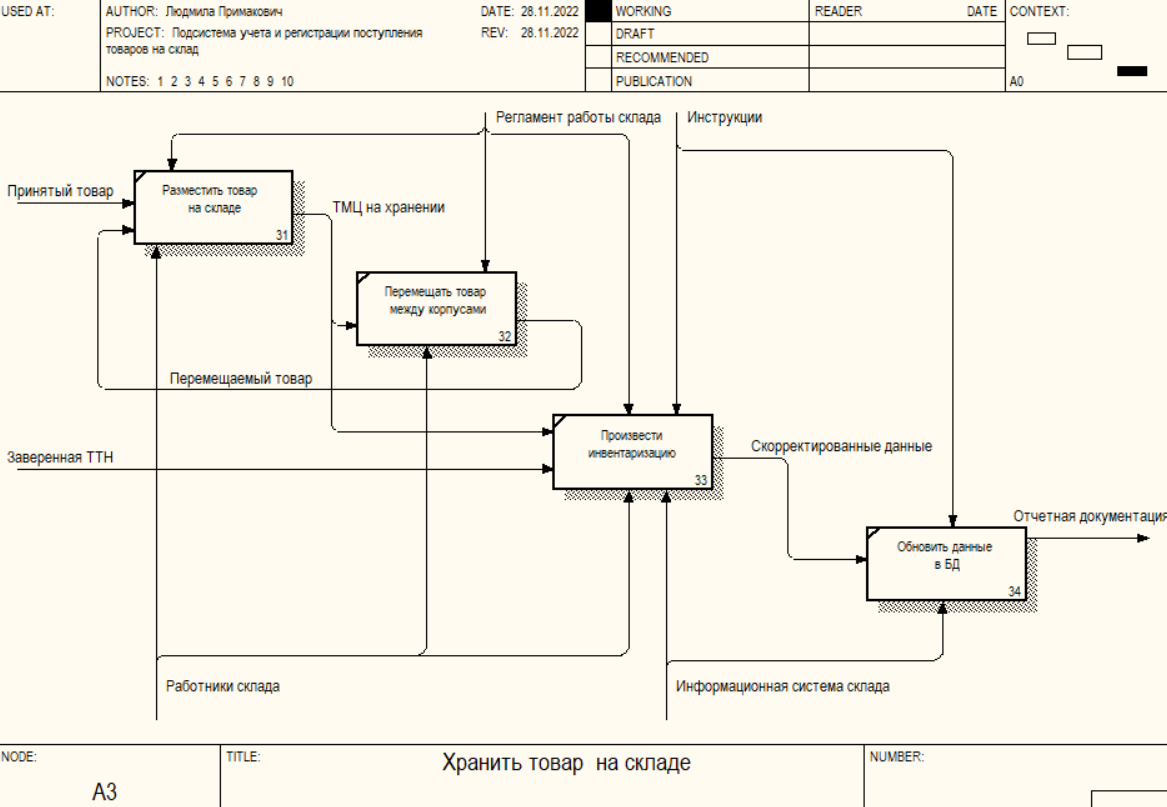


Рисунок 1.7 – Декомпозиция блока «Хранить товар на складе»

Подводя итог проделанной работе по описанию функциональной модели данного курсового проекта, можно отметить, что для организации процесса регистрации и учета поступлений товаров на склад логистической компании необходимо учитывать соотношение принятых и забракованных товаров от каждого конкретного поставщика, для того, чтобы в дальнейшем продолжать сотрудничество лишь с надежными поставщиками, тем самым максимизируя прибыль от деятельности склада и повышая эффективность его работы.

При помощи функционального проектирования были продемонстрированы все этапы бизнес-процессов предметной области.

## **1.3 Анализ требований к разрабатываемому программному средству. Спецификация функциональных требований**

Для разработки программного средства необходимо понять общий принцип осуществления регистрации прибытия товаров на склад, а также предусмотреть ведение учетной документации всех поступлений.

Поскольку каждый поставщик поставляет свой товар, причем это происходит неоднократно, необходимо спроектировать базу данных (БД), которая осуществляла бы хранение информации о каждой поставке в электронном виде. Это позволит упростить ведение учетной деятельности, а также позволит легко производить анализ поставщиков посредством их фильтрации по определенным критериям. Для корректной работы необходимо предусмотреть возможность добавления, редактирования и обновления информации в БД. Для хранения информации будет использован *MYSQL* *Server*. Подключение к нему будет осуществляться при авторизации, и, в зависимости от авторизованной роли, пользователь будет получать определенный набор возможностей для выполнения своей работы.

Для осуществления своей деятельности работники склада должны быть действительными сотрудниками, поэтому их личные данные вносятся администратором. Он же получает возможность удаления их, например, вследствие увольнения работника., а также имеет доступ ко всей базе данных, может производить анализ поставщиков и инвентаризацию, т.е. обновление полей ТТН в соответствии с датой.

Работники должны принимать товарно-транспортные накладные – это их уникальная функция. В результате приемки работник может как принять поступивший товар, так и отклонить, например, при несоответствии сроков годности или наличии повреждений.

Поставщик самостоятельно регистрируется в системе и имеет возможность оформить ТТН. Программа после заполнения полей формы требует оплату для заведения личного счета на каждую поставку.

Диаграмма вариантов использования представлена на рисунке 1.8.

Диаграмма вариантов использования (сценариев поведения, прецедентов) является исходным концептуальным представлением системы в процессе ее проектирования и разработки.

Суть даной диаграммы заключается в том, что проектируемая система представляется в виде множества актеров, взаимодействующих с системой с помощью так называемых вариантов использования (use cases). Поэтому эту диаграму ещё называют usecase-диаграммой. Каждый use case определяет набор действий, которые система совершает при взаимодействии с актерами. При этом на самой диаграмме никаким образом не отображается то, ка именно будет реализован определенный набор действий.

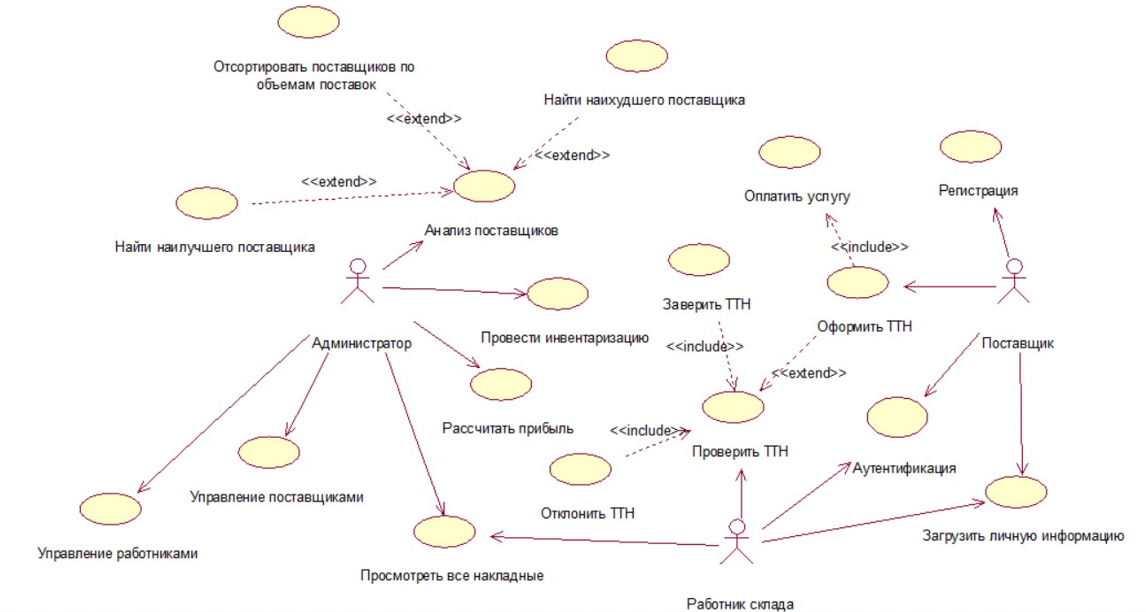


Рисунок 1.9 – Диаграмма вариантов использования

## **1.4 Разработка информационной модели учета и регистрации поставок**

CA ERwin Data Modeler —программный продукт в области реализации средств CASE-технологий [5]. Механизм работы программного обеспечения основан на методологии системного представления данных IDEF1X. Теперь он также поддерживает и другие методологии, включая пространственное моделирование. Программа позволяет строить диаграммы, удобные для чтения и анализа и отображающие потоки данных и управление ими в любой системе (не обязательно в целом информационной). Условно, это можно представить как взаимодействие сущностей и связей, где сущности – это различные процессы системы, а связи представляют собой потоки данных, документов, распоряжений, информации и прочего. Каждая сущность может быть декомпозирована и быть представлена как совокупность более мелких и быстрых процессов и обмена информацией, требуемого для их реализации.

В ходе информационного моделирования были сформированы следующие сущности:

* администраторы;
* работники;
* поставщики;
* ключи авторизации (логин/пароль);
* склады;
* продукция;
* товарно-транспортные накладные (ТТН);
* платежи.

Поставщик при регистрации в системе записывает личные данные о своей компании, которые добавляются в сущность «Поставщики». При оформлении ТТН данные этого поставщика автоматически добавляются в сущность «ТТН» и создается его личный счет для оплаты услуг складирования.

Сведения о работниках склада, принимающих товары, должны вноситься администратором. Для этого вводятся сущности «Администратор» и «Работники». Первая таблица хранит в себе id, логин и пароль, а во второй храниться ещё и личная информация о работнике и склады, на которых он работает и которые хранятся в таблице «Склады».

При входе в систему каждый пользователь в обязательном порядке проходит аутентификацию, вводит личные данные, которые целесообразно где-то хранить. Для этой цели создана сущность «Ключи». В результате данная таблица связывается с тремя дочерними таблицами: «Администратор», «Работники» и «Поставщики» – связью один-к-одному.

Таблица «Платежи» необходима для учета и расчета прибыли. Она содержит поля id личного счета, баланс на счету и связана с таблицей «Поставщик» связью один-к-одному.

В таблице «Склады» храниться информация о складах, которыми располагает компания, а в таблице «Продукция» содержится информация о типах продукции, которую на этих складах можно принимать.

Информационная модель подсистемы учета и регистрации поступления товаров на склад приведена на рисунке 1.9.

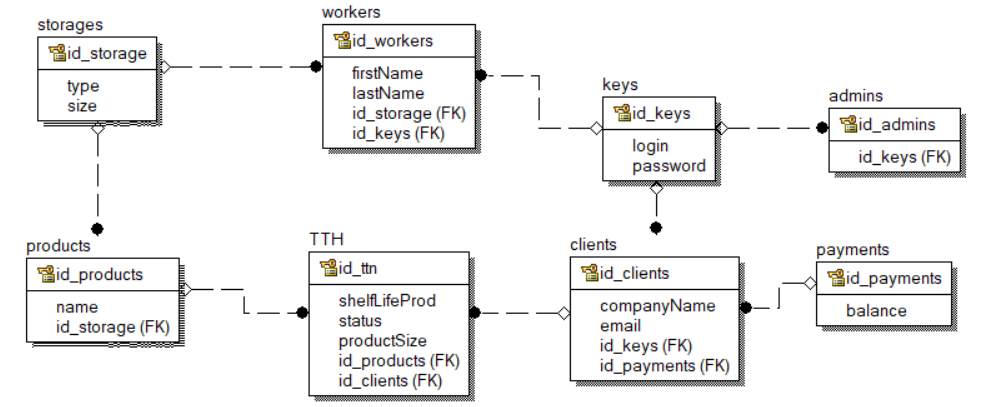


Рисунок 1.8 - Информационная модель подсистемы учета и регистрации поступления товаров на склад

## **1.5 UML-модели представления программного средства и их описание**

Язык UML предназначен для описания моделей, причем для работы с этим языком используется специальные редакторы диаграмм, такие как Rational Rose или online-редакторы. На UML можно содержательно описывать классы, объекты и компоненты в различных предметных областях, часто сильно отличающихся друг от друга [6].

Моделирование необходимо для понимания системы, причем единственной модели никогда не бывает достаточно. Следовательно, приходиться разрабатывать большое количество взаимосвязанных моделей. Рассмотрим некоторые основные модели программного средства: диаграмму состояний, диаграмму последовательности и диаграмму развертывания системы.

Диаграмма состояний показывает, каким образом объект меняет одно состояние на другое. Подобного рода диаграммы 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 того, чье поведение характеризуется реакцией на внешние события.

Пример этой диаграммы представлен на рисунке 1.10.

На данной диаграмме продемонстрирована деятельность администратора в системе: в каком состоянии может пребывать объект и какие части программного средства при этом будут задействованы.

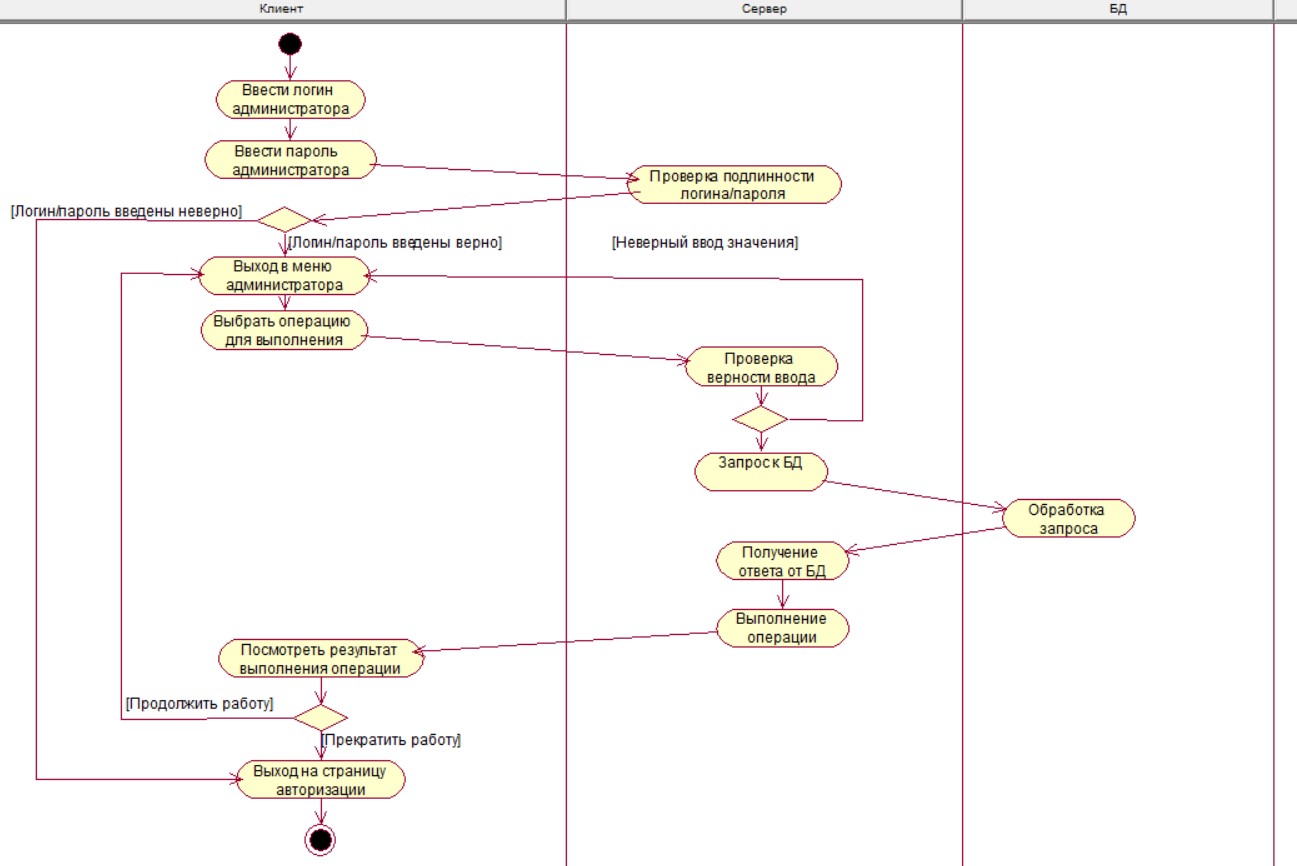


Рисунок 1.10 – Диаграмма состояний для деятельности администратора в системе

На рисунке 1.11 представлена диаграмма последовательности, на которой отображен процесса оформления товарно-транспортной накладной (ТТН). Диаграмма последовательности – одна из подвидов диаграмм взаимодействия, предназначается для моделирования взаимодействия различных объектов системы во времени, сопровождающегося обменом сообщениями. В качестве объектов на данной диаграмме могут выступать пользователи, инициирующие взаимодействие, классы, имеющие поведение, или программные компоненты.

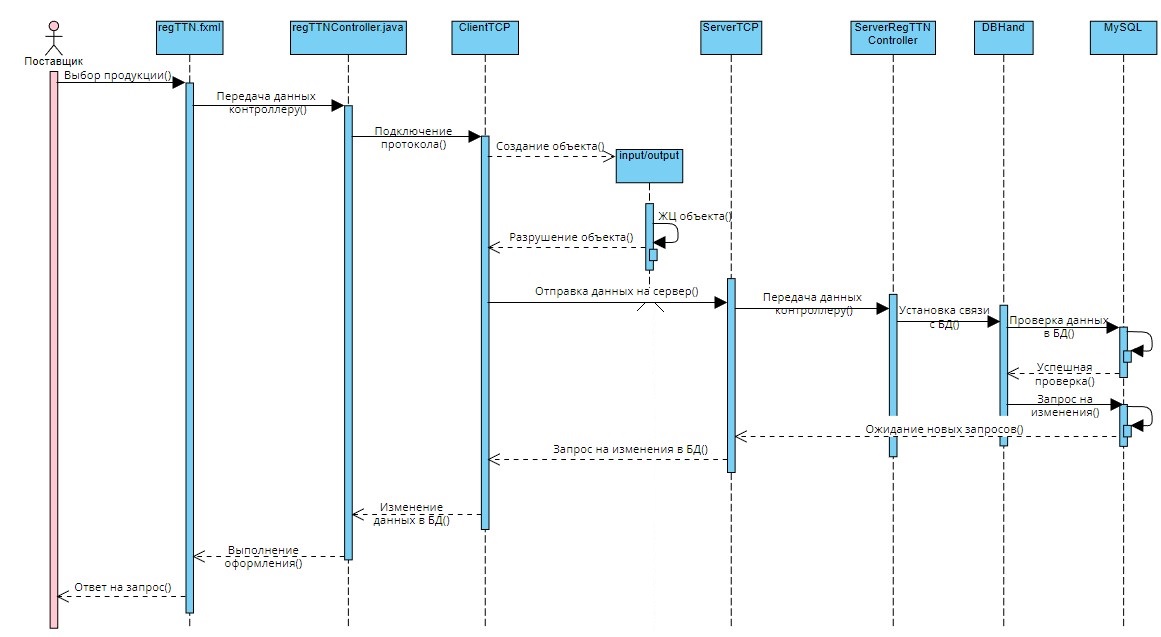


Рисунок 1.11 – Диаграмма последовательности процесса оформления ТТН

На приведенной диаграмме поставщик выбирает тип продукции для оформления ТТН и вводит необходимые данные в поля формы графического приложения, которые считывает контроллер и передает по TCP-протоколу на сервер. Сервер производит обработку данных и подключается к БД.

Там происходят соответствующие изменения и добавления записей, после чего отправляется ответ на сервер. Как в случае успеха, так и в случае неудачи сервер отправляет клиенту соответствующее сообщение, которое поставщик увидит на своем экране приложения.

Диаграмма развёртывания показывает топологию системы и распределение компонентов системы по ее узлам, а также соединения, которые представляют собой маршруты передачи информации между аппаратными узлами, которые на диаграмме обозначаются кубиками.

Диаграмма развертывания представлена на рисунке 1.12.

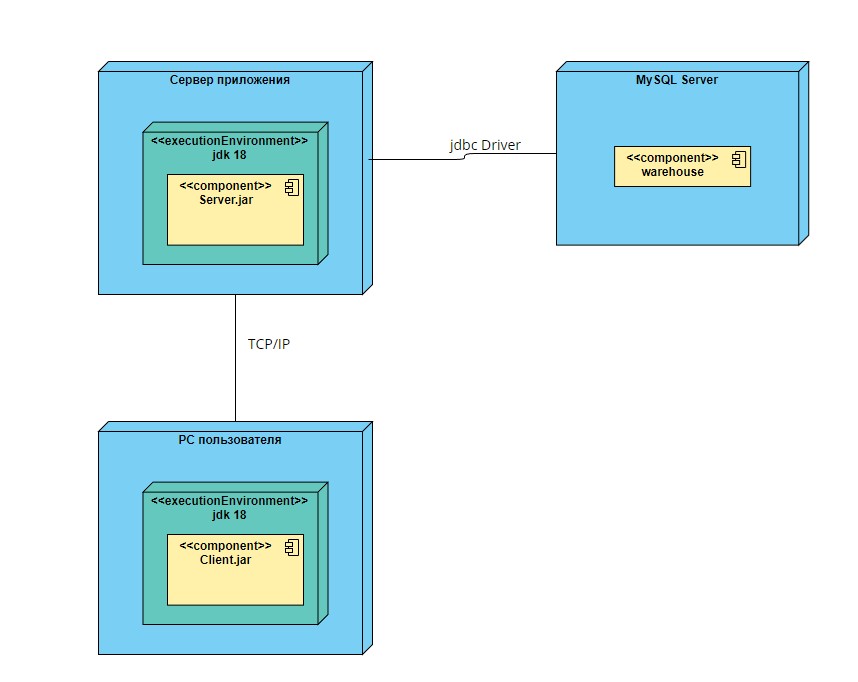


Рисунок 1.12 – Диаграмма развертывания системы

В качестве узлов выступают ПК пользователя, сервер и БД.

Для запуска разрабатываемого приложения необходимо наличие исполняемой среды JDK18 на компьютерах пользователей ПС. Также необходимо наличие MySQL Server для возможности доступа к БД.

Связь узлов ПК с сервером осуществляется по протоколу TCP/IP, а связь сервера с БД осуществляется при использовании драйвера jdbc.

Для успешной работы на устройствах должна быть установлена система Windows 8 и выше.

# **2 ПРОЕКТИРОВАНИЕ И КОНСТРУИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА**

## **2.1 Постановка задачи**

На склады логистических компаний постоянно поступают различного рода товары. Отслеживать все соответствия привозимых товаров требованиям склада проблематично. Также существует проблема надлежащего хранения информации о поставках и поставщиках, корректного заполнения сопроводительных документов.

Следовательно, возникает необходимость в разработке программного приложения, которое позволило бы оперативно собирать и обобщать отчетные данные о приеме и регистрации товаров. Для складской деятельности немаловажна оценка или анализ поставщиков.

Разрабатываемое приложение должно содержать в себе удобный для восприятия и использования интерфейс и функционал, позволяющий осуществить всю описанную ранее деятельность: оформление и заверение накладных, блокировка недобросовестных поставщиков, управление данными персонала, а также редактирование внесенных данных и добавление новых. Также данное приложение должно иметь надежную систему хранения данных, осуществлять клиент-серверное взаимодействие, доступ в систему должен защищаться и разграничиваться системой ввода логинов и паролей.

Помимо прочего разрабатываемое приложение должно осуществлять полезное представление данных. Например, вывод данных обо всех накладных или поиск наилучшего поставщика и под.

В результате разработки пользователи получат готовый к использованию хорошо оптимизированный продукт, который будет выполнять все необходимые функции, хранить данные о поставках и позволит анализировать поставщиков на основе их поставок.

## **2.2 Архитектурные решения**

## **2.3 Описание алгоритмов, реализующих ключевую бизнес-логику разрабатываемого программного средства**

## **2.4 Проектирование пользовательского интерфейса**

## **2.5 Обоснование выбора компонентов и технологий для реализации программного средства**

# **3 ТЕСТИРОВАНИЕ И ПРОВЕРКА РАБОТОСПОСОБНОСТИ ПРОГРАММНОГО СРЕДСТВА**

# **4 ИНСТРУКЦИЯ ПО РАЗВЕРТЫВАНИЮ ПРИЛОЖЕНИЯ И СКВОЗНОЙ ПРИМЕР, НАЧИНАЯ ОТ АВТОРИЗАЦИИ, ДЕМОНСТРИРУЯ РЕАЛИЗАЦИЮ ВСЕХ ВАРИАНТОВ ИСПОЛЬЗОВАНИЯ**

# **ЗАКЛЮЧЕНИЕ**

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

[1] Как организовать складской учет в компании учета [Электронный ресурс]. — Режим доступа: <https://assistentus.ru/vedenie-biznesa/skladskojuchet/>.

[2] Автоматизация складского учета на малом предприятии [Электронный ресурс]. – Электронный данные. – Режим доступа:

https://elar.rsvpu.ru/handle/123456789/29095.

[3] Нотация IDEF0 [Электронный ресурс]. – Электронный данные. – Режим доступа: <https://www.businessstudio.ru/wiki/docs/v4/doku.php/ru/csdesign/bpmodeling/idef0>.

[4] Декомпозиция: как упростить себе жизнь и достичь результатов [Электронный ресурс]. – Электронный данные. – Режим доступа:

<https://lpgenerator.ru/blog/dekompoziciya-chto-eto-takoe-prostymi-slovami/>.

[5] Фаулер, М. UML. Основы. – 3-е изд. – СПб.: Символ-плюс, 2006. –192 с.

[6] Яковлева А.О., Язык UML. Диаграммы UML. – Методическое пособие. – Минск, 2006. – 86 с.

[7]

# **Приложение А**

# **(обязательное)**

# **Отчет о проверке на заимствования в системе «Антиплагиат»**

# **Приложение Б**

# **(обязательное)**

# **Листинг кода алгоритмов, реализующих бизнес-логику**

# **Приложение В**

# **(обязательное)**

# **Листинг скрипта генерации базы данных**

SET @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0;

SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0;

SET @OLD\_SQL\_MODE=@@SQL\_MODE,

SQL\_MODE='ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION';

-- -----------------------------------------------------

-- Schema mydb

-- -----------------------------------------------------

-- -----------------------------------------------------

-- Schema warehouse

-- -----------------------------------------------------

-- -----------------------------------------------------

-- Schema warehouse

-- -----------------------------------------------------

CREATE SCHEMA IF NOT EXISTS `warehouse` DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci ;

USE `warehouse` ;

CREATE TABLE IF NOT EXISTS `warehouse`.`keys` (

`id\_keys` INT NOT NULL AUTO\_INCREMENT,

`login` VARCHAR(45) NOT NULL,

`password` VARCHAR(45) NOT NULL,

PRIMARY KEY (`id\_keys`),

UNIQUE INDEX `login\_UNIQUE` (`login` ASC) VISIBLE)

ENGINE = InnoDB

AUTO\_INCREMENT = 65

DEFAULT CHARACTER SET = utf8mb4

COLLATE = utf8mb4\_0900\_ai\_ci;

-- -----------------------------------------------------

-- Table `warehouse`.`admins`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `warehouse`.`admins` (

`idadmins` INT NOT NULL AUTO\_INCREMENT,

`id\_keys` INT NOT NULL,

PRIMARY KEY (`idadmins`),

UNIQUE INDEX `id\_keys\_UNIQUE` (`id\_keys` ASC) VISIBLE,

CONSTRAINT `fk\_admins\_keys`

FOREIGN KEY (`id\_keys`)

REFERENCES `warehouse`.`keys` (`id\_keys`)

ON DELETE CASCADE

ON UPDATE CASCADE)

ENGINE = InnoDB

AUTO\_INCREMENT = 2

DEFAULT CHARACTER SET = utf8mb4

COLLATE = utf8mb4\_0900\_ai\_ci;

-- -----------------------------------------------------

-- Table `warehouse`.`storages`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `warehouse`.`storages` (

`idstorage` INT NOT NULL AUTO\_INCREMENT,

`type` VARCHAR(45) NOT NULL,

`size` INT NULL default NULL,

PRIMARY KEY (`idstorage`),

UNIQUE INDEX `storage\_UNIQUE` (`type` ASC) VISIBLE)

ENGINE = InnoDB

AUTO\_INCREMENT = 17

DEFAULT CHARACTER SET = utf8mb4

COLLATE = utf8mb4\_0900\_ai\_ci;

-- -----------------------------------------------------

-- Table `warehouse`.`products`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `warehouse`.`products` (

`idproduct` INT NOT NULL AUTO\_INCREMENT,

`typeName` VARCHAR(45) NOT NULL,

`idstorage` INT NOT NULL,

PRIMARY KEY (`idproduct`),

INDEX `fk\_products\_storages\_idx` (`idstorage` ASC) VISIBLE,

CONSTRAINT `fk\_products\_storages`

FOREIGN KEY (`idstorage`)

REFERENCES `warehouse`.`storages` (`idstorage`))

ENGINE = InnoDB

AUTO\_INCREMENT = 13

DEFAULT CHARACTER SET = utf8mb4

COLLATE = utf8mb4\_0900\_ai\_ci;

-- -----------------------------------------------------

-- Table `warehouse`.`workers`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `warehouse`.`workers` (

`idworker` INT NOT NULL AUTO\_INCREMENT,

`firstname` VARCHAR(45) NOT NULL,

`lastname` VARCHAR(45) NOT NULL,

`category` VARCHAR(45) NOT NULL,

`id\_keys` INT NOT NULL,

`idstorage` INT NOT NULL,

PRIMARY KEY (`idworker`),

UNIQUE INDEX `id\_keys\_UNIQUE` (`id\_keys` ASC) VISIBLE,

INDEX `fk\_workers\_keys\_idx` (`id\_keys` ASC) VISIBLE,

INDEX `fk\_workers\_storages\_idx` (`idstorage` ASC) VISIBLE,

CONSTRAINT `fk\_workers\_keys`

FOREIGN KEY (`id\_keys`)

REFERENCES `warehouse`.`keys` (`id\_keys`)

ON DELETE CASCADE

ON UPDATE CASCADE,

CONSTRAINT `fk\_workers\_storages`

FOREIGN KEY (`idstorage`)

REFERENCES `warehouse`.`storages` (`idstorage`))

ENGINE = InnoDB

AUTO\_INCREMENT = 15

DEFAULT CHARACTER SET = utf8mb4

COLLATE = utf8mb4\_0900\_ai\_ci;

-- -----------------------------------------------------

-- Table `warehouse`.`payments`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `warehouse`.`payments` (

`idpayment` INT NOT NULL AUTO\_INCREMENT,

`payment` INT NOT NULL,

`balance` INT NULL DEFAULT NULL,

PRIMARY KEY (`idpayment`),

UNIQUE INDEX `payment\_UNIQUE` (`payment` ASC) VISIBLE)

ENGINE = InnoDB

AUTO\_INCREMENT = 33

DEFAULT CHARACTER SET = utf8mb4

COLLATE = utf8mb4\_0900\_ai\_ci;

-- -----------------------------------------------------

-- Table `warehouse`.`clients`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `warehouse`.`clients` (

`idclient` INT NOT NULL AUTO\_INCREMENT,

`companyName` VARCHAR(45) NOT NULL,

`email` VARCHAR(45) NOT NULL,

`id\_keys` INT NOT NULL,

`idpayment` INT NULL DEFAULT NULL,

PRIMARY KEY (`idclient`),

UNIQUE INDEX `id\_keys\_UNIQUE` (`id\_keys` ASC) VISIBLE,

UNIQUE INDEX `idpayment\_UNIQUE` (`idpayment` ASC) VISIBLE,

INDEX `fk\_clients\_keys\_idx` (`id\_keys` ASC) VISIBLE,

CONSTRAINT `fk\_clients\_keys`

FOREIGN KEY (`id\_keys`)

REFERENCES `warehouse`.`keys` (`id\_keys`)

ON DELETE CASCADE

ON UPDATE CASCADE,

CONSTRAINT `fk\_clients\_payments`

FOREIGN KEY (`idpayment`)

REFERENCES `warehouse`.`payments` (`idpayment`)

ON DELETE CASCADE

ON UPDATE CASCADE)

ENGINE = InnoDB

AUTO\_INCREMENT = 36

DEFAULT CHARACTER SET = utf8mb4

COLLATE = utf8mb4\_0900\_ai\_ci;

-- -----------------------------------------------------

-- Table `warehouse`.`ttns`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `warehouse`.`ttns` (

`idttn` INT NOT NULL AUTO\_INCREMENT,

`productName` varchar(45) NOT NULL,

`status` varchar(45) NOT NULL,

`productSize` INT NULL DEFAULT NULL,

`shelfLifeProd` DATE NULL DEFAULT NULL,

`idproduct` INT NOT NULL,

`idworker` INT NOT NULL,

PRIMARY KEY (`idttn`),

INDEX `fk\_groups\_products\_idx` (`idproduct` ASC) VISIBLE,

INDEX `fk\_groups\_workers\_idx` (`idworker` ASC) VISIBLE,

CONSTRAINT `fk\_ttns\_products`

FOREIGN KEY (`idproduct`)

REFERENCES `warehouse`.`products` (`idproduct`),

CONSTRAINT `fk\_ttbs\_workers`

FOREIGN KEY (`idworker`)

REFERENCES `warehouse`.`workers` (`idworker`))

ENGINE = InnoDB

AUTO\_INCREMENT = 12

DEFAULT CHARACTER SET = utf8mb4

COLLATE = utf8mb4\_0900\_ai\_ci;

# **ВЕДОМОСТЬ ДОКУМЕНТОВ КУРСОВОГО ПРОЕКТА**