Министерство образования Республики Беларусь

Учреждение образования

Белорусский государственный университет информатики и радиоэлектроники

Инженерно-экономический факультет

Кафедра экономической информатики

Дисциплина: Распределенные системы обработки информации

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему:

РАЗРАБОТКА АВТОМАТИЗИРОВАННОЙ СИСТЕМЫ РАСЧЕТА СТОИМОСТИ ЗАКАЗА ТОВАРОВ И УСЛУГ МАГАЗИНА СПОРТИВНОГО ПИТАНИЯ

Студентка: Ломако Е. Г., гр. 074002

Руководитель: Лыщик А. П.

Минск 2023

**СОДЕРЖАНИЕ**

[Введение 3](#_Toc103206260)

[1 Aнализ и моделирование предметной области системы расчета стоимости заказа 5](#_Toc103206261)

[1.1 Описание предметной области 5](#_Toc103206262)

[1.2 Анализ требований к разрабатываемому программному средству. Спецификация функциональных требований 8](#_Toc103206263)

[2 Разработка функциональной модели системы расчета стоимости заказа 11](#_Toc103206264)

[3 Разработка информационной модели автоматизированной системы расчета стоимости заказа 16](#_Toc103206265)

[4 UML – модели представления программного средства и их описание 21](#_Toc103206267)

[5 Описание применения паттернов проектирования 31](#_Toc103206268)

[6 Описание алгоритмов, реализующих ключевую бизнес-логику ситемы расчета стоимости заказа 35](#_Toc103206270)

[7 Руководство по развертыванию и использованию системы расчета стоимости заказа 42](#_Toc103206273)

[Заключение 55](#_Toc103206274)

[Список использованных источников 56](#_Toc103206275)

[Приложение А 57](#_Toc103206276)

[Приложение Б 68](#_Toc103206277)

# **ВВЕДЕНИЕ**

Целей создания приложения может быть очень много. Например, приложение может являться просто визитной карточкой, элементом фирменного стиля. Также оно может способствовать продвижению торговой марки, продукции или услуг. Приложение может быть эффективным инструментом, который организует работу между клиентами и сотрудниками, между сотрудниками одной или нескольких корпораций.

Целью данного проекта является улучшение производительности экономического отдела магазина за счет частичной автоматизации процесса обработки и расчета стоимости заказа.

Они достигаются посредством решения следующих задач автоматизации технологического процесса:

* реализация клиент-серверного приложения;
* внедрение системы в работу магазина;
* изучение предметной области;
* моделирование системы;
* данные должны храниться в базе данных, связь с базой данных осуществляется на серверной части;
* обеспечить систематизацию данных, хранящихся в базе;
* обеспечить ограниченный доступ к базе данных, содержащей данные клиентов.

С помощью данного клиент-серверного приложения увеличится количество клиентов, ускорятся процессы поиска необходимой информации, а также проще и надежнее станут операции расчета стоимости заказа или доставки.

Основные преимущества автоматизации:

* значительное ускорение операций по обработке данных;
* освобождение большой части рабочего времени для работы над более важными задачами;
* повышение точности данных учета, качества их обработки;
* экономия времени на операциях, которые раньше выполнялись вручную;
* расширение возможностей магазина.

Выбирая систему автоматизации, любая организация получает возможность реализовать следующий спектр задач:

* быстрее и проще организовать и структурировать фонды базы данных;
* расширить спектр оказываемых услуг;
* повысить комфорт работы сотрудников.

Автоматизация технологического процесса — совокупность методов и средств, предназначенная для реализации системы или систем, позволяющих осуществлять управление самим технологическим процессом без непосредственного участия человека, либо оставления за человеком права принятия наиболее ответственных решений.

Основа автоматизации технологических процессов — это перераспределение материальных, энергетических и информационных потоков в соответствии с принятым критерием управления (оптимальности).

Автоматизация помогает быстрее находить нужную информацию о интересующих товарах. Развитие электронных технологий и средств телекоммуникаций позволяет принципиально по-новому организовать процесс каталогизации данных и создания электронных каталогов, сведя к минимуму тысячекратное дублирование трудовых затрат и нерациональное расходование финансовых средств за счет разумного разделения труда каталогизаторов и совместного использования его результатов.

Для реализации каждой из функций были использованы средства Java и SQL. Для написания интерфейса использовался Java Swing. Также была задействована библиотека JFreeChart для построения графиков со статистикой. Для передачи данных был задействован протокол TCP/IP. При проектировании системы использовались паттерны Dao, Mediator и Singleton.

|  |  |  |
| --- | --- | --- |
| **1** | **АНАЛИЗ И МОДЕЛИРОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ СИСТЕМЫ РАСЧЕТА СТОИМОСТИ ЗАКАЗА** | |
| **1.1** | | **Описание предметной области** |

Спортивное питание — это особая группа пищевых продуктов, выпускающаяся преимущественно для людей, ведущих активный образ жизни, занимающихся спортом и фитнесом.

Приём спортивного питания направлен в первую очередь на улучшение спортивных результатов, повышение силы и выносливости, укрепление здоровья, увеличение объёма мышц, нормализацию обмена веществ, достижение оптимальной массы тела и в целом на увеличение качества и продолжительности жизни.

Подавляющее большинство продуктов спортивного питания не имеет ничего общего с допингом.

Виды спортивного питания:

* высокобелковые продукты;
* [углеводно-белковые смеси](https://ru.wikipedia.org/wiki/%D0%93%D0%B5%D0%B9%D0%BD%D0%B5%D1%80_(%D0%BF%D0%B8%D1%89%D0%B5%D0%B2%D0%B0%D1%8F_%D0%B4%D0%BE%D0%B1%D0%B0%D0%B2%D0%BA%D0%B0));
* аминокислоты;
* донаторы оксида азота ([NO-формулы](https://ru.wikipedia.org/w/index.php?title=NO-%D1%84%D0%BE%D1%80%D0%BC%D1%83%D0%BB%D1%8B&action=edit&redlink=1));
* [жиросжигатели](https://ru.wikipedia.org/w/index.php?title=%D0%96%D0%B8%D1%80%D0%BE%D1%81%D0%B6%D0%B8%D0%B3%D0%B0%D1%82%D0%B5%D0%BB%D0%B8&action=edit&redlink=1);
* [левокарнитин](https://ru.wikipedia.org/wiki/%D0%9B%D0%B5%D0%B2%D0%BE%D0%BA%D0%B0%D1%80%D0%BD%D0%B8%D1%82%D0%B8%D0%BD) (L-карнитин);
* [специальные препараты](https://ru.wikipedia.org/w/index.php?title=%D0%A1%D0%BF%D0%B5%D1%86%D0%B8%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B5_%D0%BF%D1%80%D0%B5%D0%BF%D0%B0%D1%80%D0%B0%D1%82%D1%8B&action=edit&redlink=1);
* [креатин](https://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%B5%D0%B0%D1%82%D0%B8%D0%BD%D0%BE%D0%B2%D1%8B%D0%B5_%D0%B4%D0%BE%D0%B1%D0%B0%D0%B2%D0%BA%D0%B8);
* [антикатаболики](https://ru.wikipedia.org/w/index.php?title=%D0%90%D0%BD%D1%82%D0%B8%D0%BA%D0%B0%D1%82%D0%B0%D0%B1%D0%BE%D0%BB%D0%B8%D0%BA%D0%B8&action=edit&redlink=1);
* препараты, повышающие уровень [тестостерона](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D1%81%D1%82%D0%BE%D1%81%D1%82%D0%B5%D1%80%D0%BE%D0%BD);
* [средства для укрепления суставов и связок](https://ru.wikipedia.org/w/index.php?title=%D0%A1%D1%80%D0%B5%D0%B4%D1%81%D1%82%D0%B2%D0%B0_%D0%B4%D0%BB%D1%8F_%D1%83%D0%BA%D1%80%D0%B5%D0%BF%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F_%D1%81%D1%83%D1%81%D1%82%D0%B0%D0%B2%D0%BE%D0%B2_%D0%B8_%D1%81%D0%B2%D1%8F%D0%B7%D0%BE%D0%BA&action=edit&redlink=1);
* [витамины](https://ru.wikipedia.org/wiki/%D0%92%D0%B8%D1%82%D0%B0%D0%BC%D0%B8%D0%BD%D1%8B), минералы и [витамино-минеральные комплексы](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8_%D0%B7%D0%BD%D0%B0%D1%87%D0%B8%D0%BC%D1%8B%D0%B5_%D1%8D%D0%BB%D0%B5%D0%BC%D0%B5%D0%BD%D1%82%D1%8B" \o "Биологически значимые элементы);
* [энергетики](https://ru.wikipedia.org/wiki/%D0%AD%D0%BD%D0%B5%D1%80%D0%B3%D0%B5%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%BD%D0%B0%D0%BF%D0%B8%D1%82%D0%BE%D0%BA);
* [кофеин](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D1%84%D0%B5%D0%B8%D0%BD);
* [изотоники](https://ru.wikipedia.org/w/index.php?title=%D0%98%D0%B7%D0%BE%D1%82%D0%BE%D0%BD%D0%B8%D0%BA%D0%B8&action=edit&redlink=1).

Магазин предоставляет большой выбор спортивного питания. Основными поставщиками спортивного питания являются компании из Америки, России, Беларуси и Германии. Самыми распространенными марками спортивного питания являются Whey, Amino, Bcaa, HardMass и CLA.

Предоставляемые типы спортивного питания:

* протеины;
* креатины;
* аминокислоты;
* BCAA;
* жиросжигатели;
* гейнеры;
* энергетики и изотоники;
* витамины и минералы.

Белки — высокомолекулярные органические вещества, состоящие из альфа-аминокислот, соединённых в цепочку пептидной связью.

Лучший быстрый протеин:

* 100% Whey Gold Standard - смесь изолятов и концентратов. За счет высокой популярности в последнее время имеет неоправданно завышенную стоимость;
* Prostar 100% Whey Protein от Ultimate Nutrition - смесь изолятов и концентратов, оптимальное соотношение цены и качества;
* 100% Pure Platinum Whey от SAN - смесь различных фракций сывороточного белка;
* Triple whey protein от Power System - смесь изолята, гидролизата и концентрата сывороточного протеина. Очень качественная замена дорогим продуктам раскрученных брендов по невысокой цене;
* Elite Whey Protein от Dymatize - бюджетный вариант.

Креатин — азотсодержащая карбоновая кислота, которая встречается в организме позвоночных. Участвует в энергетическом обмене в мышечных и нервных клетках.

Креатин:

* ProMera Sports CON-CRET;
* MuscleTech Platinum 100% Creatine;
* Optimum Nutrition Micronized Creatine Powder;
* Beast Sports Nutrition Creature Powder;
* BPI Sports Best Creatine;
* MuscleTech Creactor;
* RSP Nutrition CreAde;
* Foundation Series Micronized Creatine;
* [MuscleTech CELL-TECH](http://sportwiki.to/Cell-Tech_(MuscleTech)).

Аминокислоты - органические соединения, в молекуле которых одновременно содержатся карбоксильные и аминные группы. Именно от количества, состава, чередования аминокислот зависят свойства белка.

Аминокислоты:

* Optimum Nutrition Essential AmiN.O. Energy;
* [BSN AMINOx](http://sportwiki.to/Amino_X_(BSN));
* SciVation Xtend;
* Cellucor Alpha Amino;
* ErgoGenix ErgoAmino+;
* BPI Sports Best Aminos w/Energy;
* Platinum Series AMINO RECOVERY;
* [Dymatize Super Amino 6000](http://sportwiki.to/Super_Amino_6000_(Dymatize));
* Cellucor COR-Performance Beta-BCAA.

Жиросжигатели, или сжигатели жира — это вид спортивного питания или специальные препараты, которые созданы для редукции лишних жировых отложений. Жиросжигатели способствуют снижению массы тела, делают мышцы более рельефными, позволяют лучше концентрироваться на упражнениях и облегчают тренировки.

Жиросжигатели:

* Lipo-6x от Nutrex - лучший термогеник по версии Спортвики.ту, BB.com и многих других авторитетных ресурсов. В описании добавки вы можете подробно узнать о его преимуществах и недостатках;
* Stimulant X (Anabolic Xtreme) - термогеник, с выраженным стимулирующим эффектом, поднимает настроение, обладает продолжительным эффектом (до 8 часов стимуляции и подавления аппетита);
* Tight Hardcore от SAN - еще один топовый термогеник, отличающийся высоким уровнем качества и эффективности;
* Cheaters Relief‎‎‎‎ от BSN - лучший блокатор углеводов и блокатор жиров;
* L-карнитин - эффективность его не настолько высока как у термогеников, но она существенна. Многие считают карнитин бесполезным, однако независимые исследования доказали, что эта аминокислота обеспечивает сжиганию жира в среднем на 10-15% больше по сравнению с группой плацебо. Ошибочное мнение связано с тем, что эффект его практически невозможно почувствовать, оценить можно только результаты. Карнитин попадает в рейтинг "лучшие жиросжигатели", главным образом, потому что он абсолютно безопасен и даже полезен для здоровья, противопоказания к приему отсутствуют.

Гейнер (от англ. gain — прирост, добавка) — пищевая добавка при спортивном питании. Содержит, главным образом, углеводы (простые либо сложные, от чего во многом зависит цена продукта) и белок (как правило концентрат сывороточного белка, но встречаются и мультикомпонентные по составу белка гейнеры).

Гейнеры:

* [Serious Mass (Optimum Nutrition)](http://sportwiki.to/Serious_Mass_(Optimum_Nutrition));
* [Pro Gainer (Optimum Nutrition)](http://sportwiki.to/Pro_Gainer_(Optimum_Nutrition));
* [True-Mass (BSN)](http://sportwiki.to/True-Mass_(BSN));
* [Super Mass Gainer (Dymatize)](http://sportwiki.to/Super_Mass_Gainer_(Dymatize));
* [Real Gains (Universal Nutrition)](http://sportwiki.to/Real_Gains_(Universal_Nutrition));
* Arnold Schwarzenegger Series Iron Mass;
* Animal Mass Universal Nutrition;
* [True-Mass 1200 (BSN)](http://sportwiki.to/True-Mass_1200_(BSN));
* [Up Your Mass (MHP)](http://sportwiki.to/Up_Your_Mass_(MHP));

Изотоники (англ. Isotonic drinks) — спортивные напитки, которые помогают организму восстанавливаться после продолжительных нагрузок. В изотониках содержится небольшое количество солей и сахаров, в концентрации, схожей с их концентрацией в организме человека. Во время упражнений организм теряет жидкость, а также растворенные в ней соли. Спортивные напитки помогают восполнить эту потерю, кроме того, они пополняют запасы гликогена. В медицине для восполнения жидкости используются регидратанты.

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

При выполнении данного курсового проекта можно выделить две основные задачи: изучение работы магазина спортивного питания и разработка программы по расчету стоимости заказа товаров и услуг.

В ходе решения второй задачи можно выделить следующие пункты:

* разработать приложение на объектно-ориентированном языке;
* приложение должно быть выполнено в архитектуре клиент-сервер;
* бизнес-логика системы должна быть реализована только на серверной части приложения;
* построить функциональную модель системы в IDEF0;
* смоделировать информационную систему с помощью стандарта UML;
* данные должны храниться в базе данных, связь с базой данных осуществляется на серверной части;
* обеспечить систематизацию данных, хранящихся в базе;
* обеспечить ограниченный доступ к базе данных, содержащей данные клиентов.

В программе также должны быть реализованы основные функции и действия:

* регистрация (при первом использовании приложения пользователю необходимо зарегистрироваться, чтобы ему был доступен весь остальной функционал). При попытке зарегистрировать пользователя с уже существующим логином будет выброшена исключительная ситуация. Приложение не подразумевает регистрацию нового администратора, данные о логине и пароли админа хранятся в БД и доступны только ему;
* авторизация (будет выброшена исключительная ситуация, если такого пользователя не существует). При авторизации администратора необходимо ввести логин: admin и пароль: adminadmin, после чего Вам откроется весь функционал системы. Если же вы авторизуетесь как обычный пользователь данные для ввода будут следующие: логин: user и пароль: useruser. И соответственно Ваш функционал будет гораздо меньше;
* добавление товара (данный метод доступен только администратору). Необходимо заполнить соответствующую форму, чтобы оформить новый товар в магазине;
* удаление товара (данный метод доступен только администратору). Необходимо ввести id товара, информацию о котором хотите удалить;
* изменение товара (данный метод доступен только администратору). Необходимо ввести id товара, информацию о котором хотите изменить и заполнить форму с новыми данными;
* управление товарами (данный метод доступен как администратору, так и пользователю). После вызова данного метода в таблицу будет выведена вся информация о товарах магазина;
* заказ товаров (доступен только пользователю). Необходимо заполнить соответствующую форму, в которой указывается тип спортивного питания, количество упаковок, вес упаковки, Ваш регион и т.д. После этого в отдельное поле будет выведена стоимость Вашего заказа в белорусских рублях. Все вышеперечисленные поля будут необходимы для формирования конечной стоимости заказа. Все данные о заказах помещаются в соответствующую таблицу в БД, где всем заказам ставится по умолчанию статус в обработке;
* сортировка (данный метод доступен всем). По нажатию на колонку таблицы все данные будут отсортированы в алфавитном порядке. Сортировка реализована как по убыванию, так и по возрастанию;
* формирование статистики стран-производителей (доступен только пользователю). Данный метод предоставляет возможность получить информацию об основных производителях спортивного питания в мире.

|  |  |
| --- | --- |
| **2** | **РАЗРАБОТКА ФУНКЦИОНАЛЬНОЙ МОДЕЛИ СИСТЕМЫ РАСЧЕТА СТОИМОСТИ ЗАКАЗА** |

IDEF0 — методология функционального моделирования (англ. function modeling) и графическая нотация, предназначенная для формализации и описания бизнес-процессов. Отличительной особенностью IDEF0 является ее акцент на соподчиненность объектов.

Функциональная модель IDEF0 представляет собой набор блоков, каждый из которых представляет собой «черный ящик» со входами и выходами, управлением и механизмами, которые детализируются (декомпозируются) до необходимого уровня. Наиболее важная функция расположена в верхнем левом углу. А соединяются функции между собой при помощи стрелок и описаний функциональных блоков. При этом каждый вид стрелки или активности имеет собственное значение. Данная модель позволяет описать все основные виды процессов, как административные, так и организационные. Стрелки могут быть:

* Входящие – вводные, которые ставят определенную задачу.
* Исходящие – выводящие результат деятельности.
* Управляющие (сверху вниз) – механизмы управления (положения, инструкции и пр).
* Механизмы (снизу-вверх) – что используется для того, чтобы произвести необходимую работу [1].

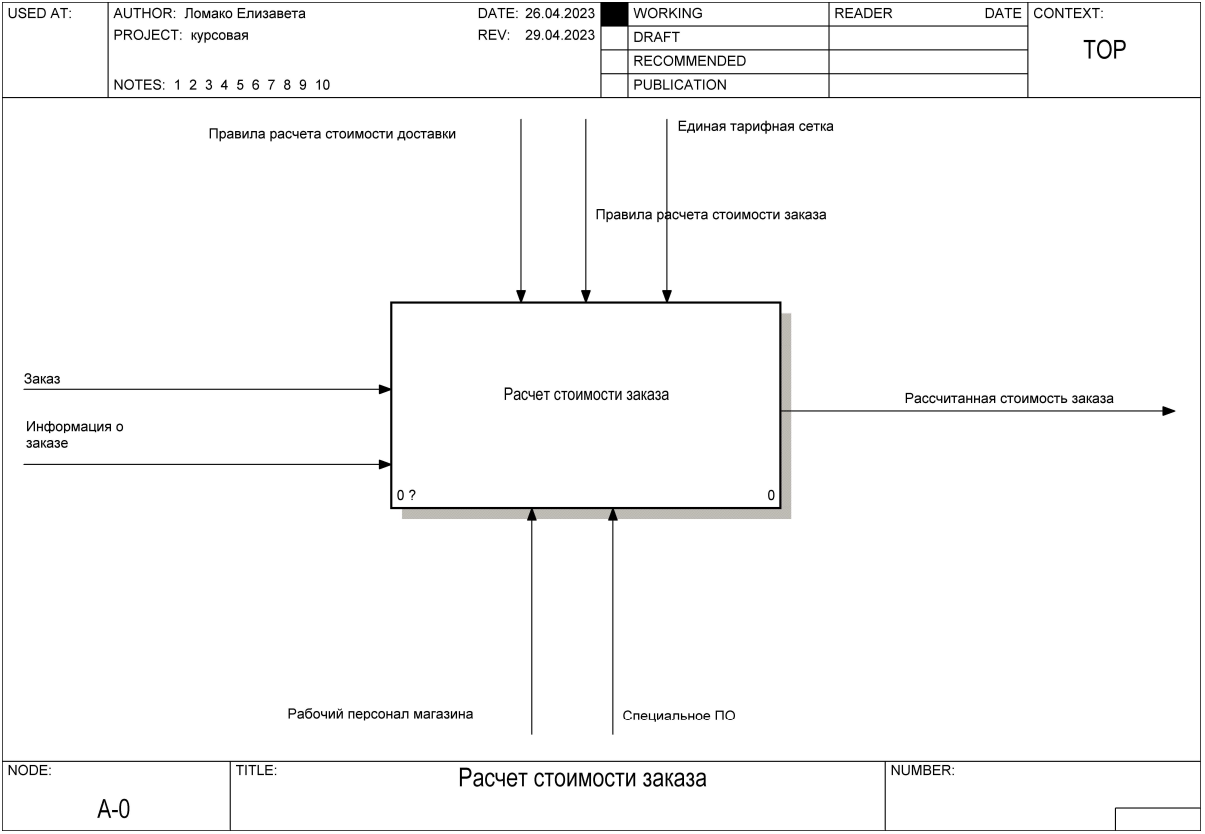
Основной процесс магазина спортивного питания заключается в расчете стоимости заказа товаров и услуг. Поэтому, чтобы понимать, как будет происходить данный процесс была разработана функциональная модель IDEF0.

Входными данными модели будут информация о заказе и сам заказ.

Для того, чтобы рассчитать стоимость необходим сотрудник персонала, который заполняет необходимую форму и специальное ПО с помощью которого и будет рассчитываться конечная стоимость.

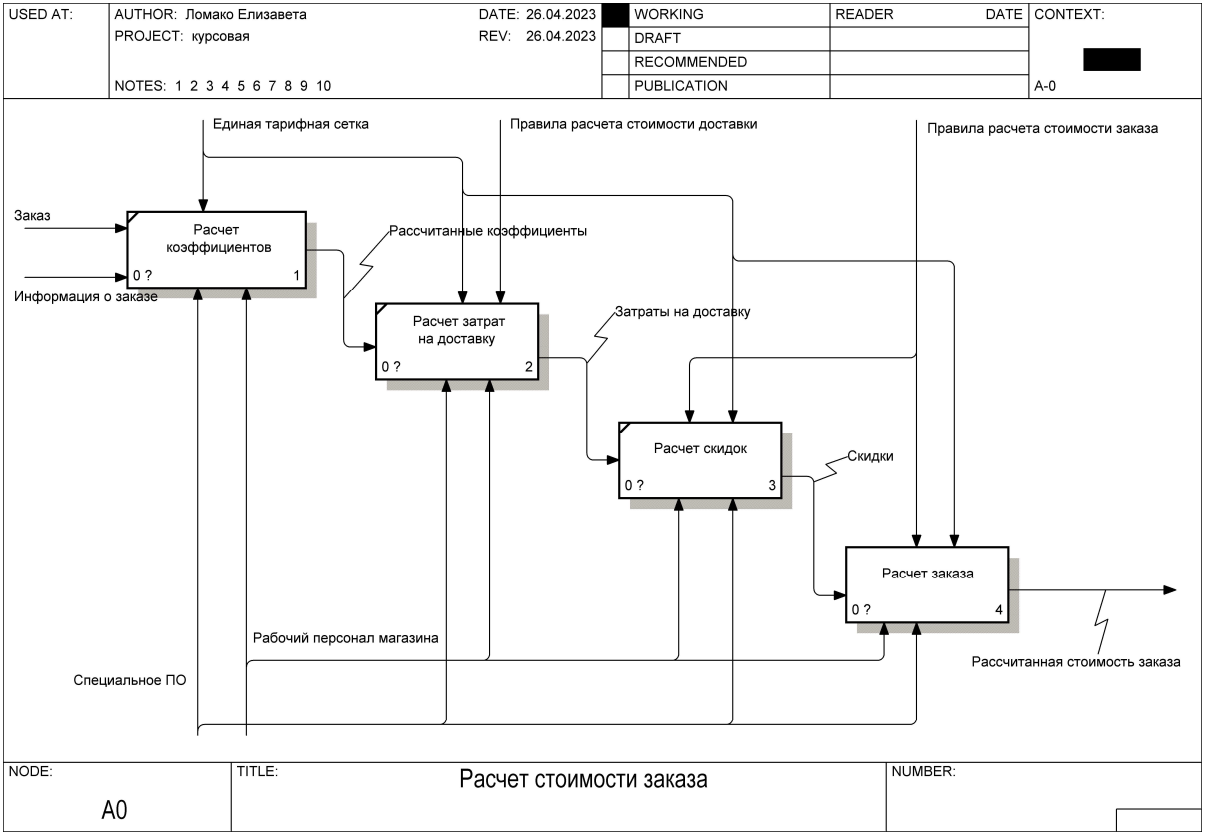
Управляют процессом расчета стоимости правила расчета стоимости доставки и заказа, а также единая тарифная сетка.

Как результат, мы получим рассчитанную стоимость заказа. Функциональная модель IDEF0 по расчету стоимости заказа представлена на рисунке 2.1.

Рисунок 2.1 – Контекстная диаграмма

При декомпозиции контекстной диаграммы (Рисунок 2.2) были определены следующие процессы:

* расчет коэффициентов;
* расчет затрат на доставку;
* расчет скидок;
* расчет заказа.

Рисунок 2.2 – Декомпозиция первого уровня диаграммы IDEF0

Первой задачей является расчет коэффициентов. Коэффициенты зависят от множества факторов: региона доставки, типа спортивного питания, веса упаковки спортивного питания, количества упаковок и страны – производителя. Чем дальше находится покупатель, тем больше будет установлен коэффициент, отвечающий за регион. При заказе спортивного питания из Германии будет установлен коэффициент 1.2, из Америки 1.1, а из России и других стран – 1. При заказе более 5 пачек спортивного питания Вы получите полную скидку на доставку. Чем больше вес упаковки, тем больше стоимость данной упаковки соответственно.

Второй задачей является расчет затрат на доставку. Стоимость доставки зависит от веса и объема перевозимого груза, а также расстояния от пункта отправки до места назначения. Также для расчета доставки учитывается тарифная ставка перевозчика и соответствующие коэффициенты, которые можно узнать из единой тарифной сетки РБ. На величину стоимости доставки также влияет и скорость перевозимого груза. Если спортивное питание доставлена за сутки, то никаких компенсаций не предусмотрено, если же доставка была проведена в течение нескольких дней, то стоимость доставки будет уменьшена на 10%.

Третьей задачей данного процесса является расчет скидок. Здесь все достаточно просто: чем больше спортивного питания Вы приобрели, тем больший процент скидок Вы получите.

Последней, четвертой задачей процесса является расчет заказа. Его рассмотрим подробнее с помощью декомпозиций процесса.

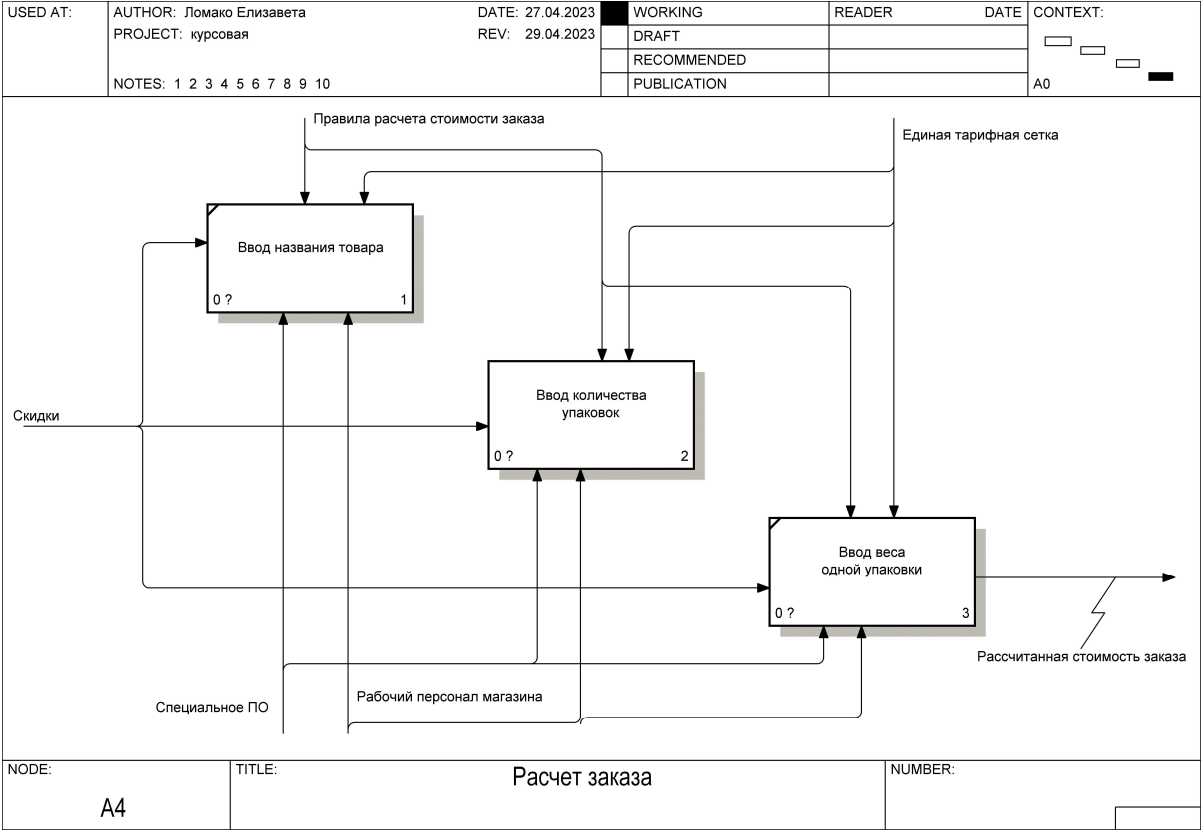
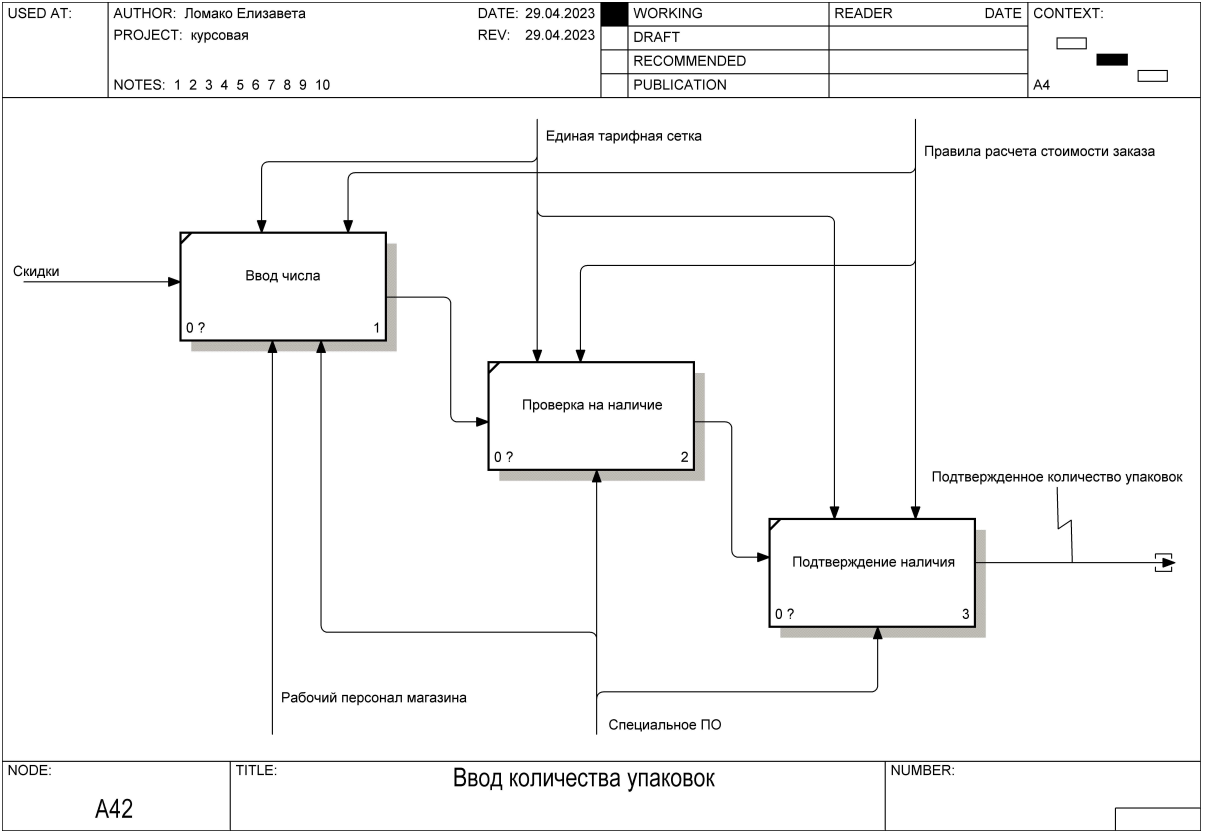


Рисунок 2.3 – Декомпозиция второго уровня диаграммы IDEF0

При декомпозиции процесса «Расчет заказа» (Рисунок 2.3) были определены следующие процессы:

* ввод названия товара;
* ввод количества упаковок;
* ввод веса одной упаковки.

Стоимость заказа будет получена при перемножении всех данных.

Рисунок 2.4 – Декомпозиция третьего уровня диаграммы IDEF0

При декомпозиции процесса «Ввод количества упаковок» (Рисунок 2.4) были определены следующие процессы:

* ввод числа;
* проверка на наличие;
* подтверждение заказа.

На данном уровне выполняется проверка количества товара в магазине. Если определенного товара меньше, чем запрашивается, то вычисления не выполняются.

Общая формула расчета заказа выглядит так: (затраты на доставку + общая сумма заказа – скидки) \* коэффициент.

|  |  |
| --- | --- |
| **3** | **РАЗРАБОТКА ИНФОРМАЦИОННОЙ МОДЕЛИ АВТОМАТИЗИРОВАННОЙ СИСТЕМЫ РАСЧЕТА СТОИМОСТИ ЗАКАЗА** |

ERwin – это не просто средство проектирования, но и инструмент разработки, способный автоматически создавать таблицы и генерировать текст хранимых процедур для всех популярных СУБД. Благодаря интеграции с популярными средами разработки программ, ERwin позволяет ускорить создание приложений для обработки данных [2].

Программа создает визуальное представление (модель данных) для решаемой задачи. Это представление может использоваться для детального анализа, уточнения и распространения как части документации, необходимой в цикле разработки.

Процесс построения информационной модели состоит из следующих шагов:

– определение сущностей;

– определение зависимостей между сущностями;

– задание первичных и альтернативных ключей;

– определение атрибутов сущностей;

– приведение модели к требуемому уровню нормальной формы;

– переход к физическому описанию модели: назначение соответствий имя сущности - имя таблицы, атрибут сущности - атрибут таблицы; задание триггеров, процедур и ограничений;

– генерация базы данных.

В ERwin существуют два уровня представления и моделирования - логический и физический.

Рассмотрим каждый уровень по отдельности.

Логический уровень означает прямое отображение фактов из реальной жизни. На логическом уровне не рассматривается использование конкретной СУБД, не определяются типы данных (например, целое или вещественное число) и не определяются индексы для таблиц.

В результате анализа предметной области было принято решение создать 5 сущностей: Поставщик, Товар, Заказ, Клиент, Регион. Логическая модель системы учета товаров в магазине спортивного питания представлена на рисунке 3.1.

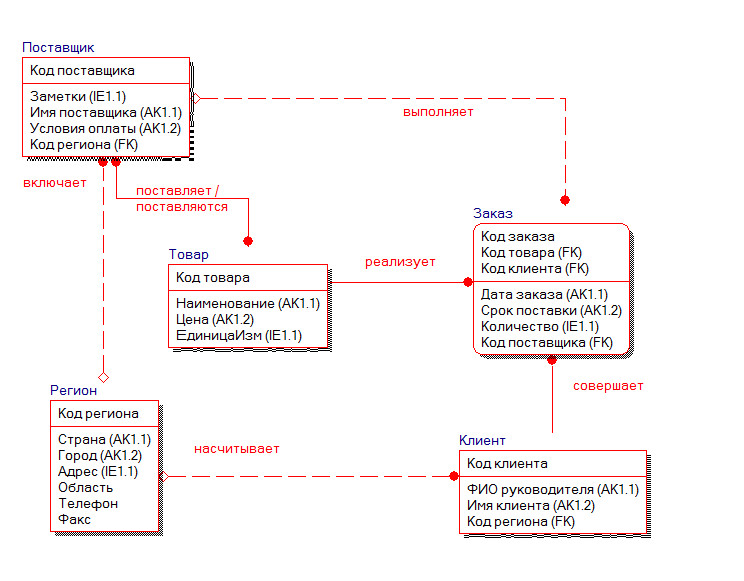


Рисунок 3.1 – Логическая модель представления системы расчета стоимости товаров в магазине спортивного питания

Рассмотрим каждую сущность в отдельности.

Сущность «Клиент» несет в себе всю информацию о покупателях магазина. Названия атрибутов сущности говорят сами за себя.

Сущность «Заказ» отражает всю информацию о заказе. Соединяется с сущностью «Клиент» связью FK\_ Код клиента (связь «один ко многим»). Т.е. любой клиент может оформить несколько заказов. Сущность «Заказ» также соединяется с сущностью «Товар» связью FK\_ Код товара (связь «один ко многим»). Т.е. на один товар может быть оформлено неограниченное число заказов.

Сущность «Регион» содержит основную информацию о поставщике и о клиенте. Соединяется с сущностью «Клиент» связью FK\_ Код региона (связь «один ко многим»). Т.е. в одном регионе может проживать большое число клиентов. Также сущность «Регион» соединяется с сущностью «Поставщик» связью FK\_Код региона (связь «один ко многим»). Аналогично предыдущему пункту в одном регионе может располагаться несколько поставщиков.

Сущность «Товар» необходима для хранения данных о продаваемом спортивном питании. Соединяется с сущностью «Поставщик» (связь «один к одному»). То есть один поставщик может поставлять только один товар в указанный промежуток времени.

Сущность «Поставщик» необходима для определения условий оплаты за предоставляемый товар. Соединяется с сущностью «Заказ» связью FK\_ Код поставщика (связь «один ко многим»). Один поставщик может обрабатывать только один заказ.

Целевая СУБД, имена объектов и типы данных, индексы составляют второй (физический) уровень модели ERwin. Физическая модель представления системы регистрации клиентов медицинского центра отображена на рисунке 3.2.

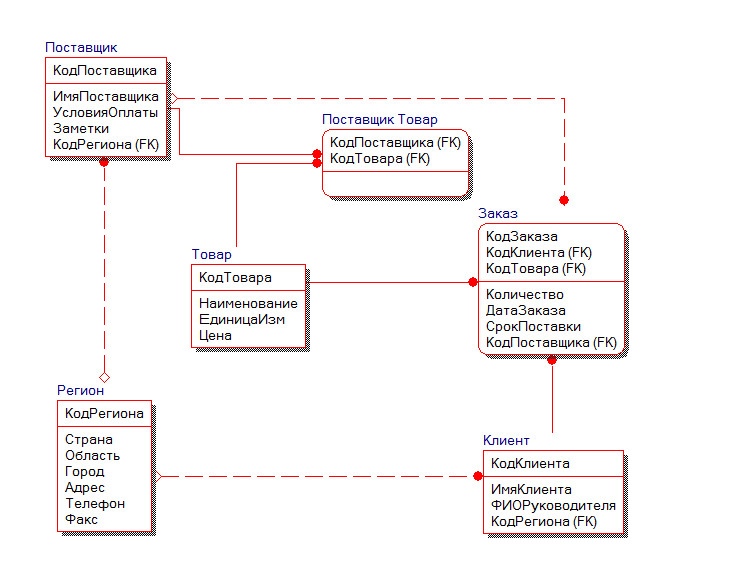


Рисунок 3.2 – Физическая модель представления системы расчета стоимости товаров магазина спортивного питания

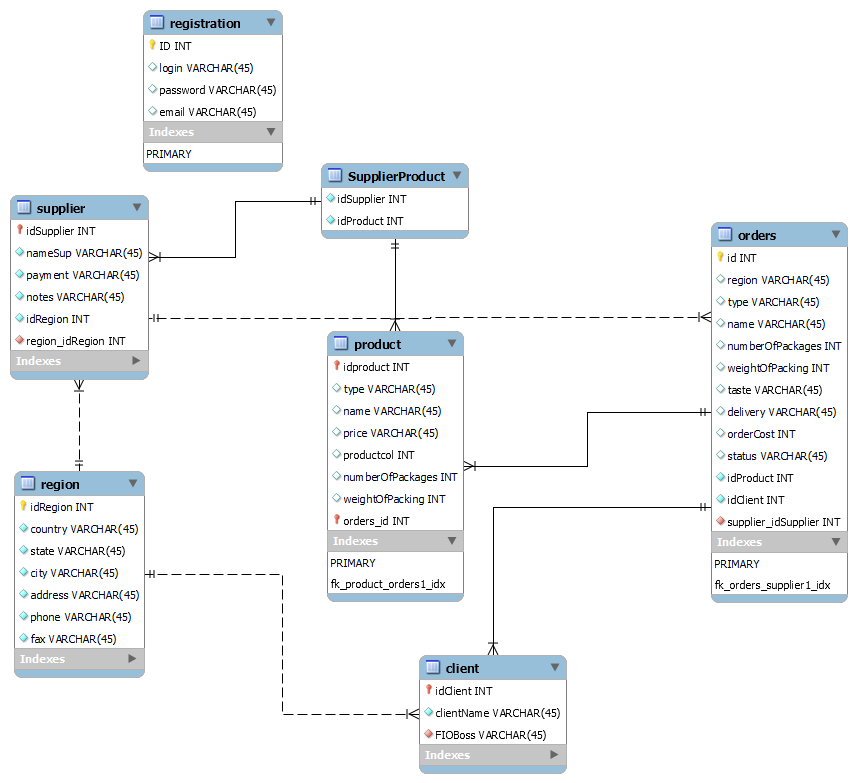


Рисунок 3.3 – Представление в MySQL Workbench

Для доказательства нахождения нашей модели в третьей нормальной форме воспользуемся следующими определениями.

Первая нормальная форма (1НФ) - это обычное отношение и любое отношение автоматически уже находится в 1НФ.

Отношение находится во второй нормальной форме (2НФ) тогда и только тогда, когда отношение находится в 1НФ и нет не ключевых атрибутов, зависящих от части сложного ключа. (Не ключевой атрибут - это атрибут, не входящий в состав никакого потенциального ключа).

В модели нет составных ключей, и она находится в первой нормальной форме, значит, она автоматически находится и во второй нормальной форме.

Отношение находится в третьей нормальной форме (3НФ) тогда и только тогда, когда отношение находится в 2НФ и все не ключевые атрибуты взаимно независимы. (Атрибуты называются взаимно независимыми, если ни один из них не является функционально зависимым от другого).

Следовательно, разработанная информационная модель находится в третьей нормальной форме.

|  |  |
| --- | --- |
| **4** | **UML – МОДЕЛИ ПРЕДСТАВЛЕНИЯ ПРОГРАММНОГО СРЕДСТВА И ИХ ОПИСАНИЕ** |

В данном разделе будет продемонстрировано моделирование информационной системы с помощью стандарта UML, который использует графические обозначения для создания абстрактной модели системы и предназначен для определения, визуализации, проектирования и документирования в основном программных систем. UML позволяет описать систему практически со всех возможных точек зрения и разные аспекты поведения системы.

Для моделирования данной курсовой работы были построены следующие диаграммы: диаграмма вариантов использования, диаграмма последовательности, диаграмма классов, диаграмма развертывания, диаграммы деятельности.

Рассмотрим подробнее каждую из диаграмм.

Магазин занимается продажей спортивного питания и оказанием услуг. В услуги магазина входит возможность заказа товаров через интернет, а также доставка данных товаров по всей территории РБ.

**Диаграмма вариантов использования**

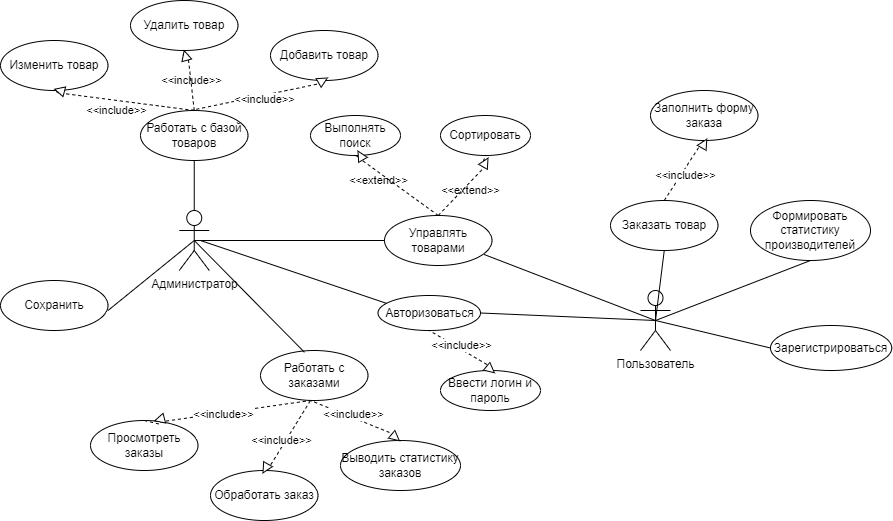


Рисунок 4.1 – Система работы магазина спортивных товаров и услуг

Субъектами в данной системе являются администратор и пользователь. Для каждого из них функционал работы приложения отличается.

Пользователь может осуществлять три функции: «Управлять товарами», «Заказать товар» и «Формировать статистики производителей».

Функция «Заказать» позволяет пользователю заказать товары, при этом рассчитать стоимость заказа. Функция «Формирование статистики производителей» показывает, какая из стран-производителей больше всего пользуется спросом у клиентов магазина.

Функция «Управление товарами» заключается в ознакомлении пользователем или администратором с товарами, которые находятся на складе в данный момент. Также в эту функцию включен прецедент «Сортировка». Эта функция напрямую связана со списком товаров на складе: производится сортировка списка в алфавитном порядке по любому критерию.

У администратора список возможностей намного шире: «Управление товарами», «Работа с заказами», «Сохранение» и «Работа с базой товаров».

«Работа с базой товаров» заключается в добавлении товаров, их изменении и удалении. Работа происходит непосредственно при обновлениях количества товаров на складе, введении новых наименований или продаже товаров вне приложения.

«Работа с заказами» включает в себя несколько прецедентов: возможность просмотреть заказы, обработать их и выводить статистику.

**Диаграмма последовательности**

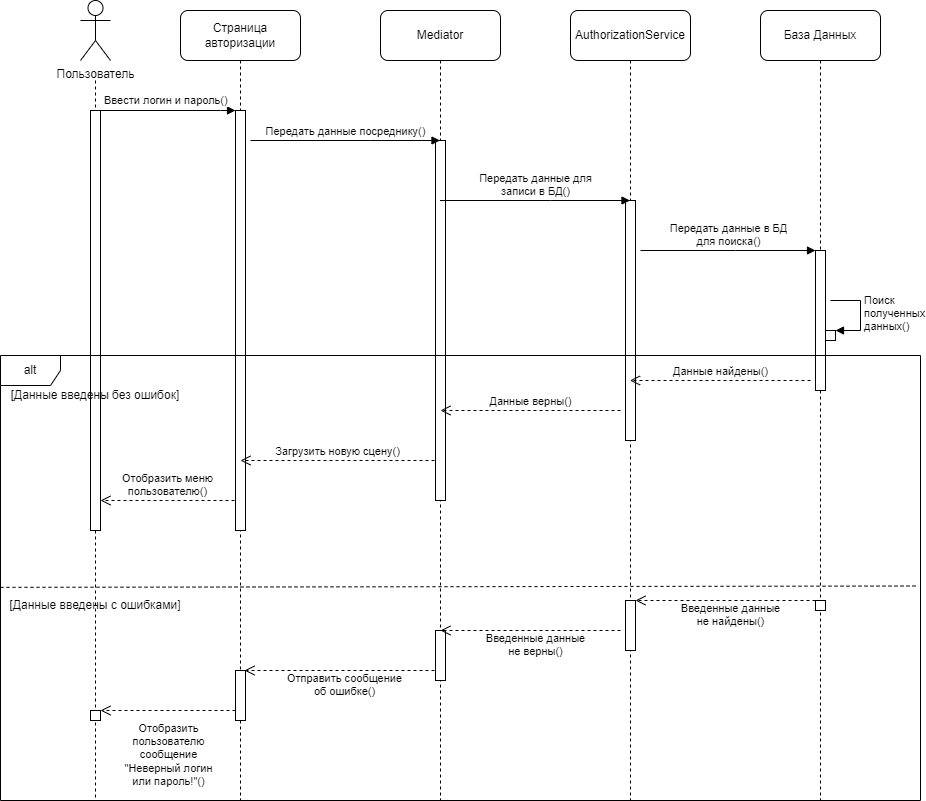


Рисунок 4.2 – Диаграмма последовательности «Добавить новый заказ»

В данной диаграмме последовательности описаны последовательные действия от ввода пароля менеджером по работе с клиентами до сохранения всех данных.

В процессе добавления нового заказа участвуют менеджер по работе с клиентами и менеджер по работе с заказами, они же администраторы. Идет обработка заказа от клиента.

Первоначально пользователь выполняет запрос на авторизацию. Этот запрос отправляется на контроллер, где происходит выбор команды и перенаправление на нужную страницу. Далее пользователь вводит свой логин и пароль и отправляет форму. Данные отправляются на контроллер, который обрабатывает поступившую команду. После определения команды происходит проверка введенного пользователем логина и пароля. Для проведения проверки необходимо обратиться к базе данных пользователей. Из базы данных будет выбран соответствующий пользователь (администратор) и его пароль будет сверен с введенным. При совпадении будет сформирован ответ о том, что авторизация прошла успешно и будет произведено перенаправление на соответствующую страницу. В случае если пользователь с указанным логином отсутствует в базе данных или введен неверный пароль, то будет выведено соответствующее сообщение.

Также, менеджер по работе с клиентами может осуществлять выбор операции «Добавить», «Выбор спортивного питания», «Выбор клиента» и «Сохранить заказ». Сообщение «добавить» приходит к получателю «Параметры работы с заказом». Сообщение «отображение полей ввода» приходит к объекту «Добавление нового заказа».

Сообщение «Выбор спортивного питания» отправляется к объекту «добавление нового заказа», из чего появляется жизненный цикл с исходящими сообщениями «получение списка клиентов» и «отображение списка спортивного питания», «обработка списка клиентов» (сообщения возвращается к тому же объекту).

Менеджер по работе с заказами отправляет сообщение «Получение списка клиентов» «Клиенту» и «Получение списка спортивного питания» «Спортивному питанию».

От объекта «Добавление нового заказа» идет сообщение «передача управления» получателю «менеджер по работе с заказами», он, в свою очередь, посылает сообщение «сохранить» объекту «заказ». На этом сообщении заканчиваются все линии жизни диаграммы последовательности.

**Диаграмма состояний**

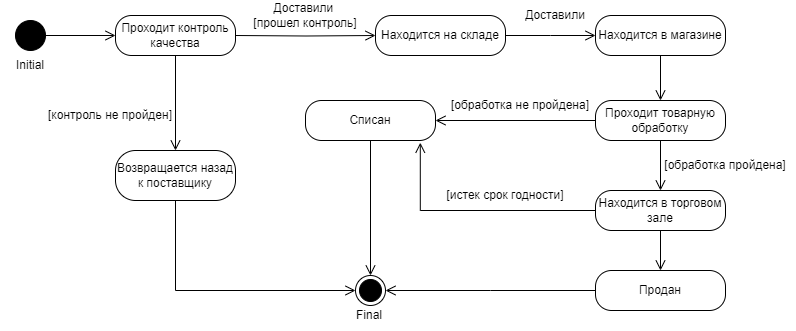


Рисунок 4.3 – Диаграмма состояний «Путь товара»

Путь товара к покупателю начинается с входного контроля качества продукции, которая от поставщиков попадает на один из распределительных центров компании. В буквальном смысле мы проверяем абсолютно все, что нам привозят: на брак, на транспортный бой, смотрим, чтобы не было повреждений упаковки, на соответствие упаковки и маркировки требованиям белорусского законодательства. Вся продукция, не прошедшая проверку на «входе», возвращается поставщику и, соответственно, не попадает в сеть. После одного из основных распределительных центров товар едет в магазины или в регионы – на один из действующих региональных складов, а оттуда попадает в магазины или сразу домой к покупателю – например, при заказе через интернет.

В данной диаграмме существуют восемь состояний товара: проходит контроль качества, находится на складе, находится в магазине, проходит товарную обработку, находится в торговом зале, продан, списан и возвращается назад к поставщику.

Когда товар проходит контроль качества, возможно 2 исхода:

* событие «доставили» с охраняющим действием «прошел контроль», ведет к состоянию товара «находится на складе»;
* действие «контроль не пройден» вызывает состояние «возвращается назад к поставщику».

Если товар возвращается назад к поставщику, товар переходит к конечному состоянию «final».

Если товар находится в состоянии «находится на складе», то с помощью события «доставили», переходит в состояние «находится в магазине».

Далее товар проходит товарную обработку, если товар обработку не проходит, он попадает в состояние «списан», после переходит к конечному состоянию.

Если обработка пройдена, то товар будет в состоянии «находится в торговом зале». При длительном нахождении в зале у товара может истечь срок годности, он попадает в состояние «списан» и переходит к конечному состоянию.

Когда товар в состоянии «находится в торговом зале», он может перейти к состоянию «продан» и перейдет к конечному состоянию.

**Диаграмма развёртывания**

Диаграмма развёртывания предназначена для визуализации элементов и компонентов программы, существующих лишь на этапе ее исполнения [3].

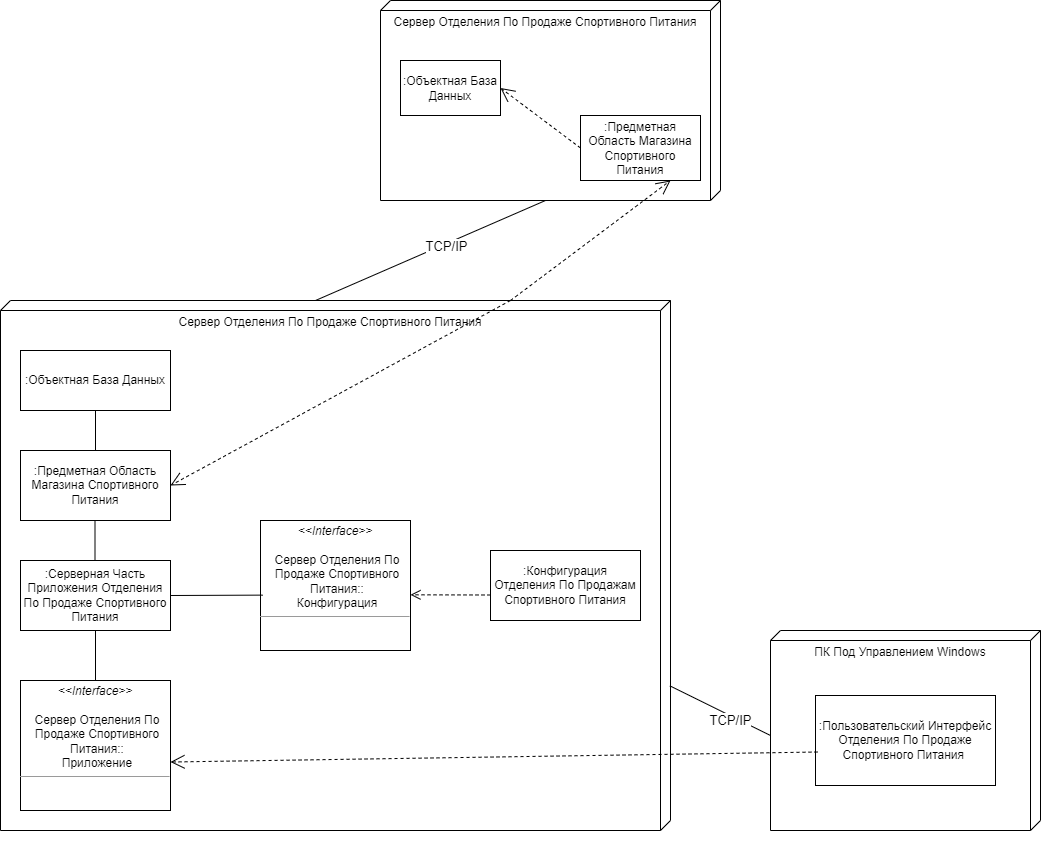


Рисунок 4.3 – Диаграмма развертывания «Взаимодействие клиента и сервера»

Диаграмма развертывания отражает физические взаимосвязи между программными и аппаратными компонентами системы. Она является хорошим средством для того, чтобы показать маршруты перемещения объектов и компонентов в распределенной системе. Каждый узел на диаграмме размещения представляет собой некоторый тип вычислительного устройства.

Диаграмма развертывания была построена для отображения организации компонентов, используемых в приложении. Диаграмма развертывания содержит графические изображения процессоров, устройств, процессов и связей между ними. В отличие от диаграмм логического представления, диаграмма развертывания является единой для системы в целом, поскольку должна всецело отражать особенности ее реализации. Разработка диаграммы развертывания, как правило, является последним этапом спецификации модели программной системы.

На данной диаграмме есть 3 основных узла: сервер отделения по продаже спортивных тренажеров, сервер отделения по продаже спортивного питания и ПК под управлением Windows. Все узлы связаны с помощью программного интерфейса удалённых методов TCP/IP.

В сервере отделения по продаже спортивных тренажеров есть два артефакта: объектная база данных и предметная область магазина спортивного питания (такой же артефакт существует и в сервере отделения по продаже спортивного питания).

В сервере отделения по продаже спортивного питания находятся шесть артефактов: объектная база данных, предметная область магазина спортивного питания, серверная часть приложения, интерфейсы сервера отделения по продаже спортивного питания (приложение и конфигурация) и конфигурация отделения по продажам спортивного питания.

С помощью интерфейса «сервер отделения по продаже спортивного питания: приложение» узел «сервер отделения по продаже спортивного питания» соединяется с узлом «ПК под управлением Windows»

В узле «ПК под управлением Windows» существует артефакт «Пользовательский интерфейс отделения по продаже спортивного питания».

Для обеспечения возможности взаимодействия сервера и базы данных на компьютере должен быть использоваться MySQL, который сохраняет все необходимые таблицы и обеспечивает корректное взаимодействие с ними.

**Диаграмма деятельности**

Диаграмма деятельности (англ. activity diagram) — 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.

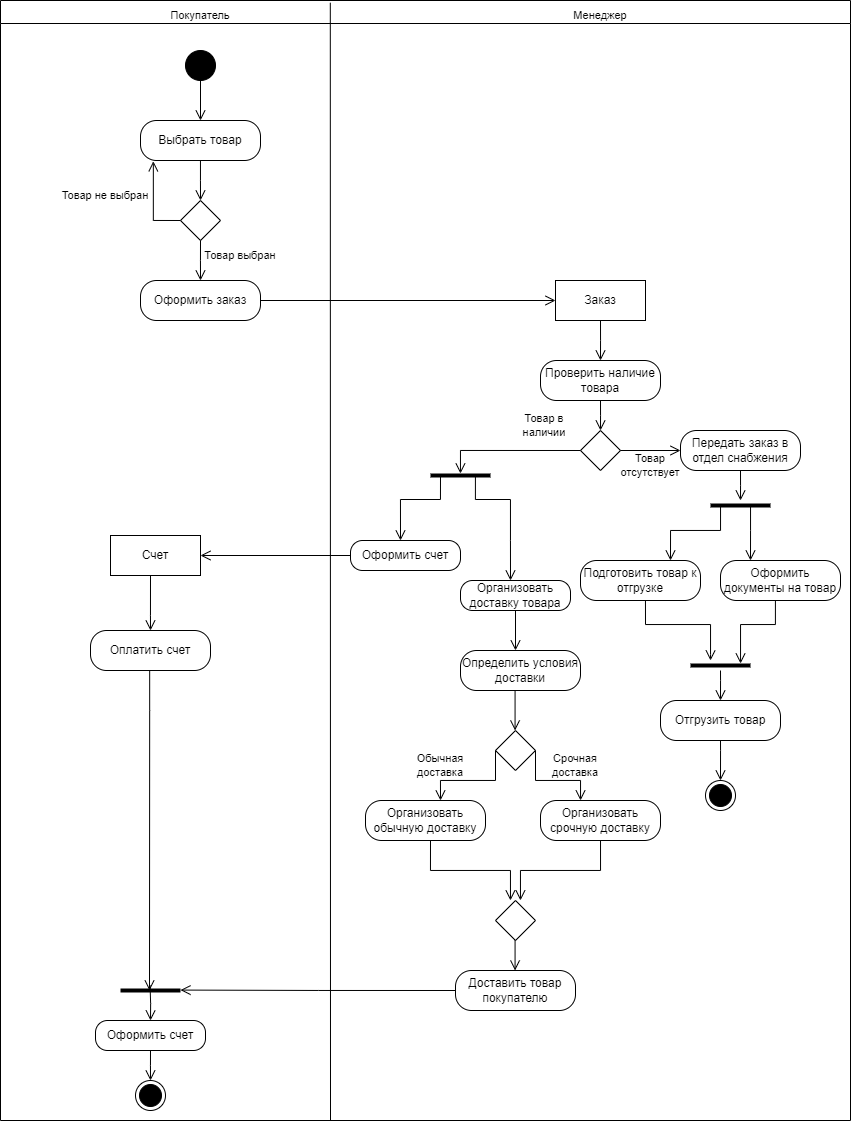


Рисунок 4.4 – Диаграмма деятельности «Оформить заказ»

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

Диаграммы деятельности состоят из ограниченного количества фигур, соединённых стрелками. Основные фигуры:

1. Прямоугольники с закруглениями — действия.
2. Ромбы — решения.
3. Широкие полосы — начало (разветвление) и окончание (схождение) ветвления действий.
4. Чёрный круг — начало процесса (начальное состояние).
5. Чёрный круг с обводкой — окончание процесса (конечное состояние).

Стрелки идут от начала к концу процесса и показывают последовательность переходов.

**Диаграмма классов**

Диаграмма классов описывает структуру программного кода системы, показывая её классы, их атрибуты и операторы, а также взаимосвязи этих классов [4].



Рисунок 4.5 – Диаграмма классов модуля расчета стоимости заказа

Учитывая факт, что все основные функции системы выполняются на серверной части, то необходимо построить диаграмму, описывающую работу одного из основных процессов серверной части.

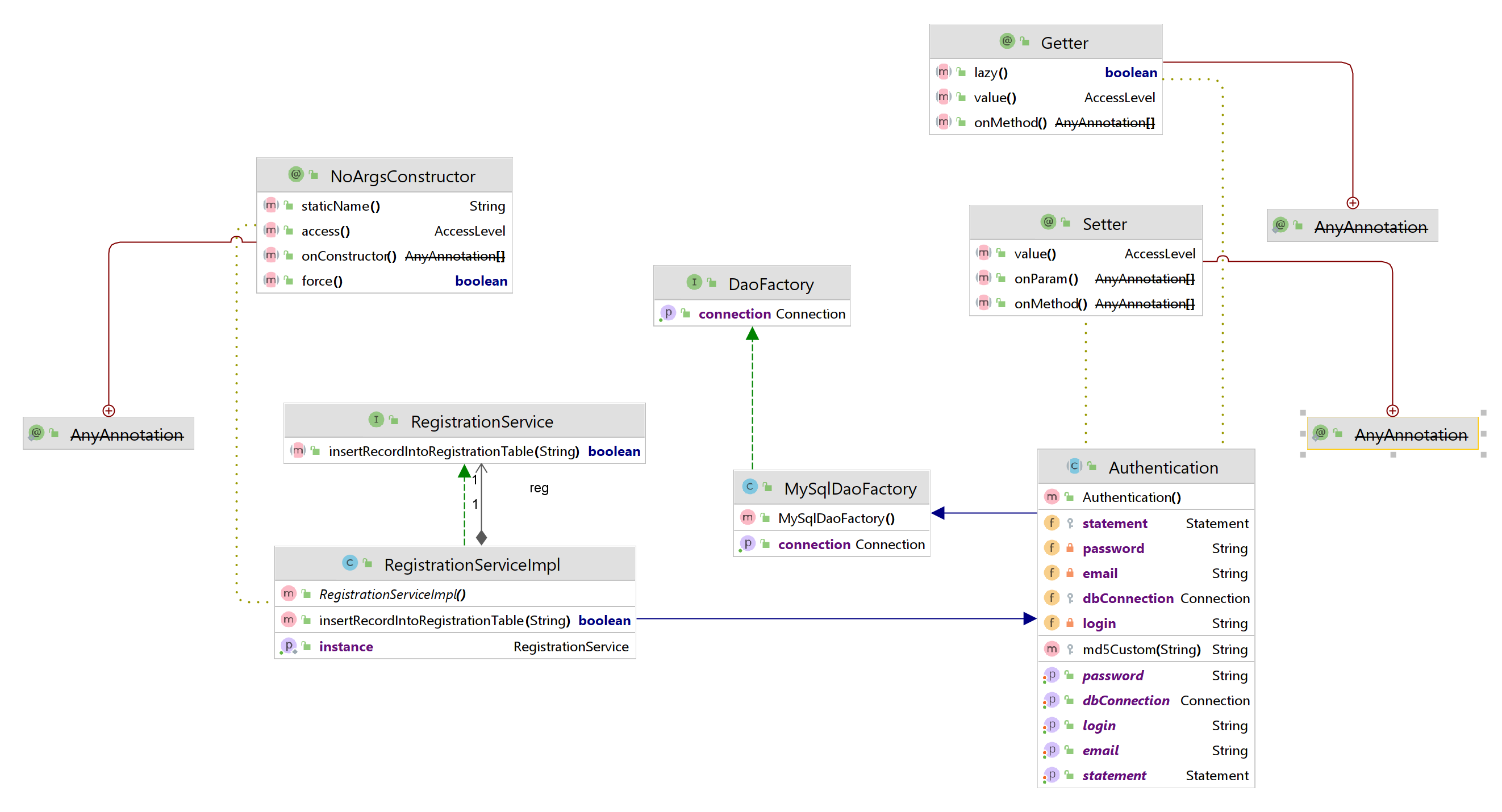


Рисунок 4.6 – Диаграмма классов процесса «Регистрация»

В рамках данной главы было рассмотрено моделирование системы с помощью стандарта UML, а также построены некоторые диаграммы, в которых показаны модели представления работы системы расчета стоимости заказа товаров и услуг магазина спортивного питания.

|  |  |
| --- | --- |
| **5** | **ОПИСАНИЕ ПРИМЕНЕНИЯ ПАТТЕРНОВ ПРОЕКТИРОВАНИЯ** |

При проектировании системы использовались паттерны Dao, Mediator и Singleton.

В программном обеспечении data access object (DAO) — это объект, который предоставляет абстрактный интерфейс к какому-либо типу базы данных или механизму хранения [5]. Определённые возможности предоставляются независимо от того, какой механизм хранения используется и без необходимости специальным образом соответствовать этому механизму хранения. Этот шаблон проектирования применим ко множеству языков программирования, большинству программного обеспечения, нуждающемуся в хранении информации и к большей части баз данных, но традиционно этот шаблон связывают с приложениями на платформе Java Enterprise Edition, взаимодействующими с реляционными базами данных через интерфейс JDBC, потому что он появился в рекомендациях от фирмы Sun Microsystems.

Используется Data Access Object (DAO) для абстрагирования и инкапсулирования доступа к источнику данных. DAO управляет соединением с источником данных для получения и записи данных.

DAO реализует механизм доступа, необходимый для работы с источником данных. Источником данных может быть персистентное хранилище (например, RDBMS), внешняя служба (например, B2B-биржа), репозиторий (LDAP-база данных), или бизнес-служба, обращение к которой осуществляется при помощи протокола CORBA Internet Inter-ORB Protocol (IIOP) или низкоуровневых сокетов. Использующие DAO бизнес-компоненты работают с более простым интерфейсом, предоставляемым объектом DAO своим клиентам. DAO полностью скрывает детали реализации источника данных от клиентов. Поскольку при изменениях реализации источника данных представляемый DAO интерфейс не изменяется, этот паттерн дает возможность DAO принимать различные схемы хранилищ без влияния на клиентов или бизнес-компоненты. По существу, DAO выполняет функцию адаптера между компонентом и источником данных.

DAO в сервере программы:

package serverStore.dao;

import java.sql.Connection;

import java.sql.SQLException;

public interface DaoFactory {

public Connection getConnection() throws SQLException;

}

DAO в Регистрации:

package serverStore.authentication;

import java.sql.SQLException;

public interface AuthorizationDao {

public boolean selectRecordsFromRegistrationTable(String str) throws SQLException;}

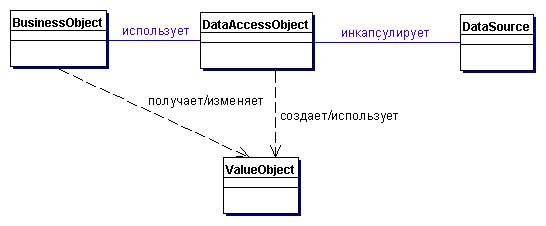


Рисунок 5.1 – Структура DAO

Посредник (англ. Mediator) — поведенческий шаблон проектирования, обеспечивающий взаимодействие множества объектов, формируя при этом слабую связанность и избавляя объекты от необходимости явно ссылаться друг на друга [6].

Паттерн Mediator определяет объект, инкапсулирующий взаимодействие множества объектов. Mediator делает систему слабо связанной, избавляя объекты от необходимости ссылаться друг на друга, что позволяет изменять взаимодействие между ними независимо.

Паттерн Mediator вводит посредника для развязывания множества взаимодействующих объектов.

Заменяет взаимодействие «все со всем» взаимодействием «один со всеми».

public static void getDBConnection() {

MysqlDataSource dataSource = new MysqlDataSource();

dataSource.setUser(loginDB);

dataSource.setPassword(passwordDB);

dataSource.setServerName("localhost");

dataSource.setDatabaseName(DB);

try {

dataSource.setServerTimezone("UTC");

} catch (SQLException throwables) {

throwables.printStackTrace();

}

try {

databaseConnection = dataSource.getConnection();

} catch (SQLException err) {

err.printStackTrace();

}

}

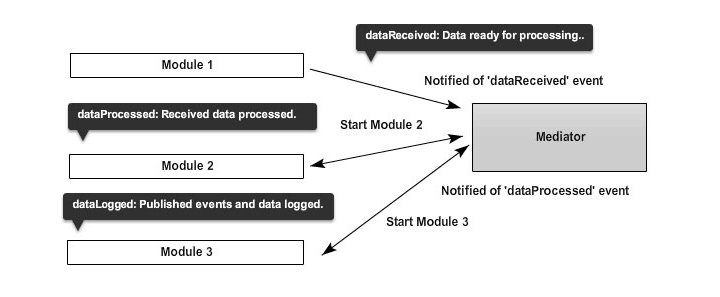


Рисунок 5.2 – Схема взаимодействия модулей и медиатора

Одиночка (англ. Singleton) — порождающий шаблон проектирования, гарантирующий, что в однопоточном приложении будет единственный экземпляр класса с глобальной точкой доступа [7].

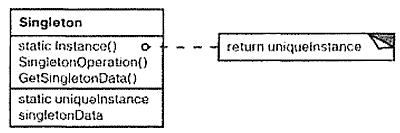


Рисунок 5.3 – Структура Singleton

Существуют несколько различных подходов, реализации шаблона Singleton, но все они имеют общие принципы.

* Private конструктор — для запрета инициализации экземпляра класса из другого класса через конструктор;
* Private static переменную того же класса, которая и будет единственным экземпляром этого класса;
* Public static метод, возвращающий экземпляр класса. Это — глобальная точка доступа для внешнего мира позволяющая получить экземпляр класса Singleton.

Ранняя реализация (Eager initialization). В ранней реализации, экземпляр класса Singleton инициализируется одновременно с загрузкой класса. Это наипростейший вариант создания класса Одиночки, но он имеет недостаток: экземпляр создается в любом случае, даже если им никто так и не воспользуется.

|  |  |
| --- | --- |
| **6** | **ОПИСАНИЕ АЛГОРИТМОВ, РЕАЛИЗУЮЩИХ БИЗНЕС-ЛОГИКУ СИСТЕМЫ РАСЧЕТА СТОИМОСТИ ЗАКАЗА** |

Алгоритм (algorithm) — это любая корректно определенная вычислительная процедура, на вход (input) которой подается некоторая величина или набор величин, и результатом выполнения которой является выходная (output) величина или набор значений. Таким образом, алгоритм представляет собой последовательность вычислительных шагов, преобразующих входные величины в выходные. Алгоритм также можно рассматривать как инструмент, предназначенный для решения корректно поставленной вычислительной задачи (computational problem). В постановке задачи в общих чертах задаются отношения между входом и выходом. В алгоритме описывается конкретная вычислительная процедура, с помощью которой удается добиться выполнения указанных отношений.

Бизнес-логика в разработке информационных систем — совокупность правил, принципов, зависимостей поведения объектов предметной области (области человеческой деятельности, которую система поддерживает). Иначе можно сказать, что бизнес-логика — это реализация правил и ограничений автоматизируемых операций. Является синонимом термина «логика предметной области».

В фазе бизнес-моделирования и разработки требований бизнес-логика может описываться в виде:

* текста;
* концептуальных аналитических моделей предметной области;
* бизнес-правил;
* разнообразных алгоритмов;
* диаграмм деятельности;
* графов и диаграмм перехода состояний;
* моделей бизнес-процессов.

Алгоритм записи в файл данных о товарах, авторизации, и добавления данных о товарах. Для реализации данных функций использовался код на языке Java и SQL.

Код для функции просмотра записи данных о товарах:

@Override

public boolean writeInFileForProduct() throws SQLException {

int flag = 0;

String selectTableSQL = "SELECT \* FROM product";

File FirstFileName = new

File("D:\\Спорт\\Server\\src\\main\\java\\serverStore\\files\\products.txt")

try {

dbConnection = getConnection();

statement = dbConnection.createStatement();

System.out.println(selectTableSQL);

ResultSet rs = statement.executeQuery(selectTableSQL);

BufferedWriter writer = new BufferedWriter(new

FileWriter(FirstFileName));

while (rs.next()) {  
 String idTable = rs.getString("id");  
 String typeTable = rs.getString("type");  
 String nameTable = rs.getString("name");  
 String priceTable = rs.getString("price");  
 String numberOfPackagesTable = rs.getString("numberOfPackages");  
 String weightOfPackingTable = rs.getString("weightOfPacking");  
 String manufacturerTable = rs.getString("manufacturer");  
 String accountingDiscounts = rs.getString("accountingDiscounts");  
 writer.write("ID: " + idTable + '\n');  
 writer.write("Тип спортивного питания: " + typeTable + '\n');  
 writer.write("Название: " + nameTable + '\n');  
 writer.write("Цена: " + priceTable + '\n');  
 writer.write("Количество упаковок: " + numberOfPackagesTable + '\n');  
 writer.write("Вес упаковки: " + weightOfPackingTable + '\n');  
 writer.write("Производитель: " + manufacturerTable + '\n');  
 writer.write("Скидки(%): " + accountingDiscounts + '\n' + '\n');

}

writer.close();

flag = 1;

} catch (FileNotFoundException e) {

System.out.println("Невозможно записать в файл: " + FirstFileName);

} catch (IOException e) {

System.out.println("Ошибка: " + e.toString());

} catch (SQLException e) {

System.out.println(e.getMessage());

} finally {

if (statement != null) {  
 statement.close();

}

if (dbConnection != null) {  
 dbConnection.close();

}

}

if (flag == 0) return false;

else return true;

}



Рисунок 6.1 – Алгоритм сохранения данных о товарах в файл

В первую очередь на сервере необходимо понять в какой момент времени нужно вызывать данную функцию. Для этого на сервер передается id операции, после чего вызывается данный метод. Суть данной функции заключается в записи данных в файл. Сперва необходимо подключиться к БД, далее осуществляется выборка всех данных из таблицы Товар и все данные помещаются в файл.

Код для функции авторизации:

@Override

public boolean selectRecordsFromRegistrationTable(String

clientMessageRecieved) throws SQLException {

String[] message = clientMessageRecieved.split(" ");

String login = message[1];

String password = md5Custom(message[2]);

int flag = 0;

String selectTableSQL = "SELECT login, password FROM registration";

try {

dbConnection = getConnection();

statement = dbConnection.createStatement();

ResultSet rs = statement.executeQuery(selectTableSQL);

while (rs.next()) {

String loginTable = rs.getString("login");

String passwordTable = rs.getString("password");

if (login.equals(loginTable) && password.equals(passwordTable)) {  
 flag = 1;  
 break;  
 } else {  
 flag = 0;  
 }

}

} catch (SQLException e) {

System.out.println(e.getMessage());

} finally {

if (statement != null) {  
 statement.close();

}

if (dbConnection != null) {  
 dbConnection.close();

}

}

if (flag == 1) {

System.out.println("Вы успешно авторизованы!!!");

return true;

} else {

return false;

}

}



Рисунок 6.2 – Алгоритм авторизации пользователя

Для авторизации мне необходимо заполнить форму на клиенте, введя туда логин и пароль. Далее эти данные отправляются на сервер и передаются в функцию авторизации. После этого происходит подключения к БД, выборка данных о пользователях с БД. Данные с БД сравниваются с данными клиента. Если эти данные совпадают, то пользователь успешно авторизован, иначе ошибка авторизации.

Код для функции добавления:

@Override

public boolean insertRecordIntoProductTable(String

clientMessageRecieved) throws SQLException {

int flag = 0;

String[] message = clientMessageRecieved.split(" ");

String type = message[1];

String name = message[2];

String price = message[3];

int iPrice = Integer.parseInt(price);

String numberOfPackages = message[4];

int iNumberOfPackages = Integer.parseInt(numberOfPackages);

String weightOfPacking = message[5];

int iWeightOfPacking = Integer.parseInt(weightOfPacking);

String manufacturer = message[6];

String accountingDiscounts = message[7];

int iAccountingDiscounts = Integer.parseInt(accountingDiscounts);

String insertTableSQL = "INSERT INTO product"  
 + "(type, name, price, numberOfPackages, WeightOfPacking,

manufacturer, accountingDiscounts) " + "VALUES"  
 + "('" + type + "'," + "'" + name + "'," + "'" + iPrice + "'," + "'" +

iNumberOfPackages + "'," + "'" + iWeightOfPacking + "'," + "'" +

manufacturer + "'," + "'" + iAccountingDiscounts + "');";

try {

dbConnection = getConnection();

statement = dbConnection.createStatement();

statement.executeUpdate(insertTableSQL);

flag = 1;

} catch (SQLException e) {

System.out.println(e.getMessage());

} finally {

if (statement != null) {

statement.close();

}

if (dbConnection != null) {  
 dbConnection.close();

}

}

if (flag == 0) return false;

else return true;

}

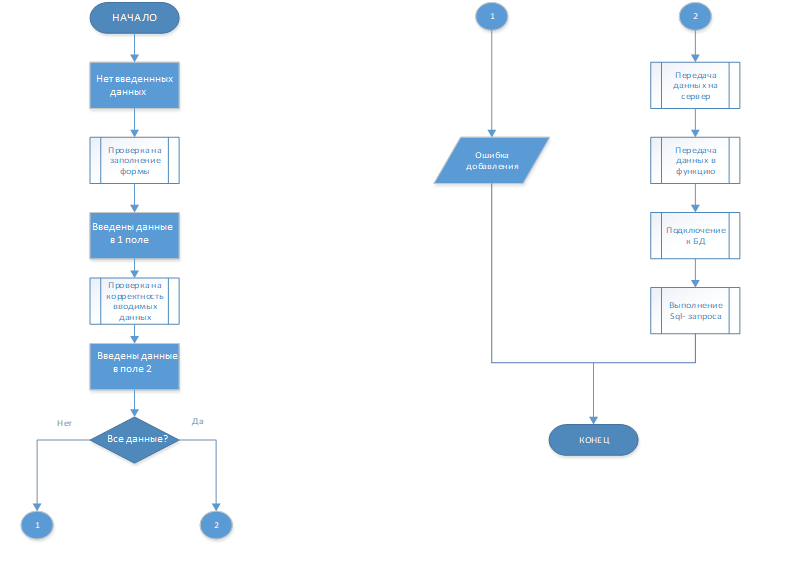


Рисунок 6.3 – Алгоритм добавления данных о товаре

Добавление нового товара доступно только администратору, поэтому в первую очередь необходимо авторизоваться в системе. Далее выберем пункт добавления, необходимо заполнить форму. После заполнения формы все данные будут отправлены на сервер и переданы в соответствующую функцию добавления в БД. С помощью sql-запроса все введенные данные будут помещены в таблицы Products.

В данной главе были рассмотрены и проанализированы алгоритмы работы приложения, а именно алгоритм сохранения данных о товарах в файл, алгоритм авторизации пользователя и алгоритм добавления данных о товаре.

С помощью этих алгоритмов был повышен уровень понимания работы приложения, а также, используя данный анализ, в дальнейшем будет возможно улучшить его работу.

|  |  |
| --- | --- |
| **7** | **РУКОВОДСТВО ПО РАЗВЕРТЫВАНИЮ И ИСПОЛЬЗОВАНИЮ СИСТЕМЫ РАСЧЕТА СТОИМОСТИ ЗАКАЗА** |

После запуска приложение в первую очередь необходимо зарегистрироваться или авторизоваться. Процесс регистрации может избежать только администратор, так как его данные уже занесены в БД вручную. Для начала зарегистрируем нового пользователя и покажем его функционал.

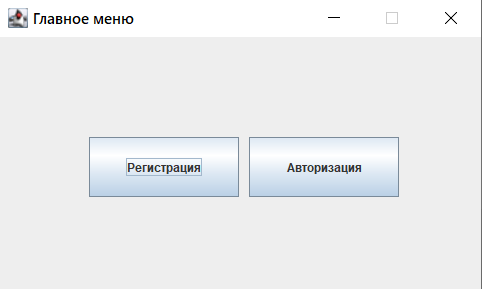


Рисунок 7.1 – Меню аутентификации

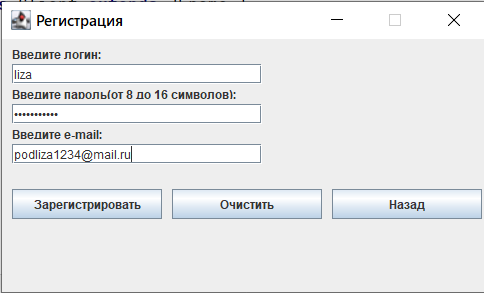


Рисунок 7.2 – Форма регистрации

Если пользователь введет при регистрации существующий логин, то система выдает ошибку.

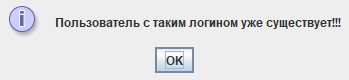


Рисунок 7.3 – Ошибка при регистрации

Если пользователь при регистрации задаст пароль с недостаточным или избыточным количеством символов, система выдаст ошибку.

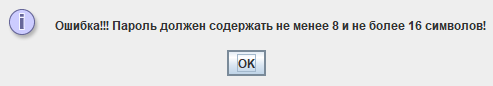


Рисунок 7.4 – Ошибка при регистрации

Если пользователь при регистрации некорректно введет свою почту, система выдаст ошибку.

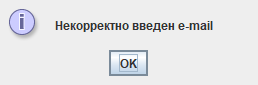


Рисунок 7.5 – Ошибка при регистрации

После регистрации можно авторизоваться и воспользоваться основным функционалом.

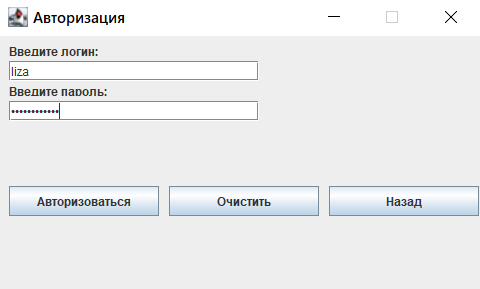


Рисунок 7.6 – Форма авторизации

Если в окно авторизации пользователь введет несуществующий логин или неверный пароль, система выдаст ошибку.

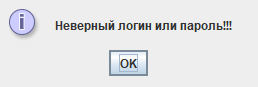


Рисунок 7.7 – Ошибка при авторизации

При успешной авторизации будет выводиться сообщение

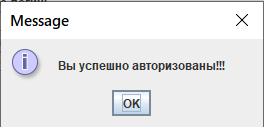


Рисунок 7.8 – Успешная авторизация

После успешной авторизации клиенту будет доступен основной функционал пользователя:

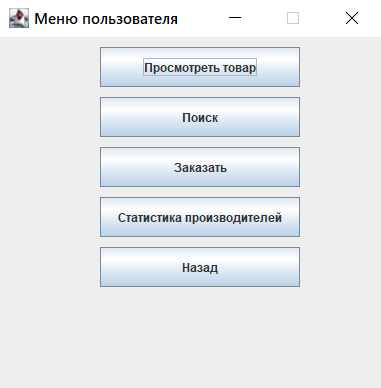


Рисунок 7.9 – Меню пользователя

Выбираем первый пункт «Просмотреть товар». Все данные, которые находятся в таблице товар будут переданы клиенту и перемещены в объект JTable.

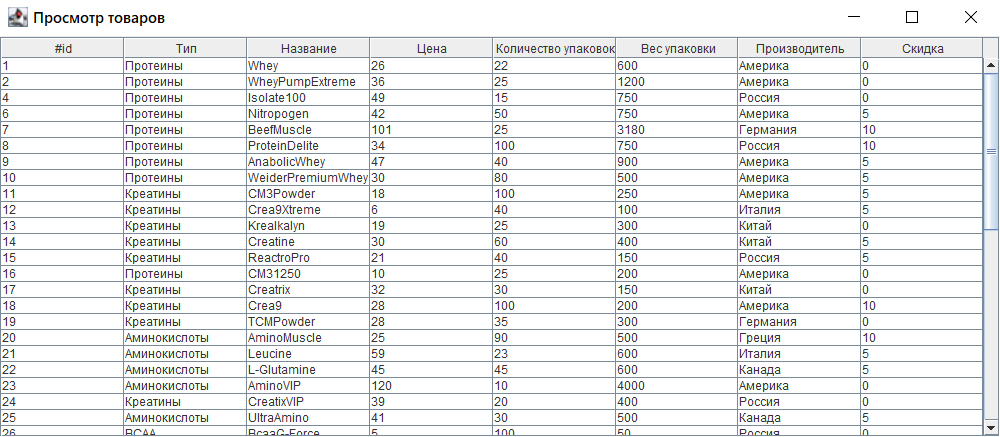


Рисунок 7.10 – Просмотр товаров

Если надо вернуться в «Меню пользователя», следует закрыть данное окно. Теперь воспользуемся функцией «Поиск». Появится диалоговое окно, в котором нужно указать id поиска и ввести искомые данные:

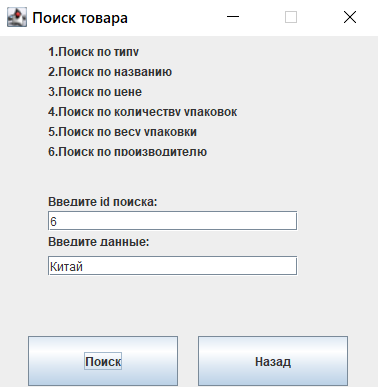


Рисунок 7.11 – Форма поиска

После нажатия кнопки поиск будет выведена вся информация о искомом товаре. Поиск осуществляется по подстроке без учета регистра.

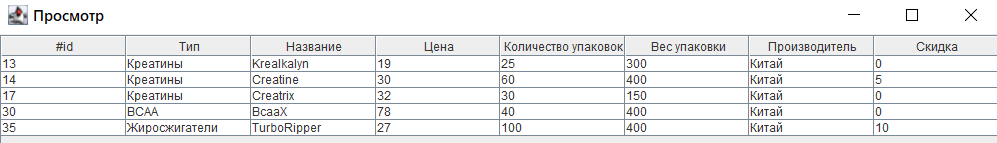


Рисунок 7.12 – Результат поиска

Далее воспользуемся основной функцией пользователя – заказ товара. В первую очередь необходимо заполнить форму заказа. Форма состоит из 2 выпадающих списков, 3 текстовых полей и 4 чекбоксов:

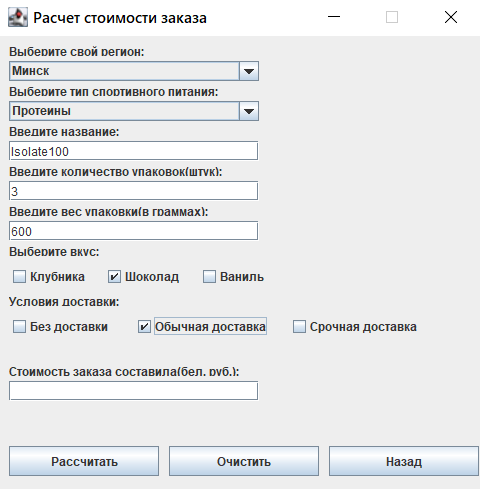


Рисунок 7.13 – Форма заказа

В каждой из форм есть кнопки очистить или вернуться назад, если пользователь ошибется или передумает. Чтобы получить стоимость заказа необходимо нажать кнопку рассчитать. После чего в пустое поле данной формы будет вписан результат расчета стоимости заказа в белорусских рублях:

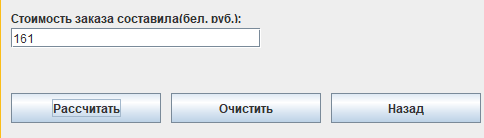


Рисунок 7.14 – Рассчитанная стоимость заказа

Все данные, полученные из формы будут занесены в таблицу «Заказ» на сервере. При этом, к данным будет добавлено новое поле – статус заказа, в котором администратор может увидеть новые заказы и их статус: обработан или в обработке. По умолчанию статус нового заказа – в обработке.

Последней функцией, доступной для пользователя, является просмотр статистики производителей. Эта функция носит информативный характер и показывает пользователю, какие страны являются основными поставщиками спортивного питания в нашей стране.

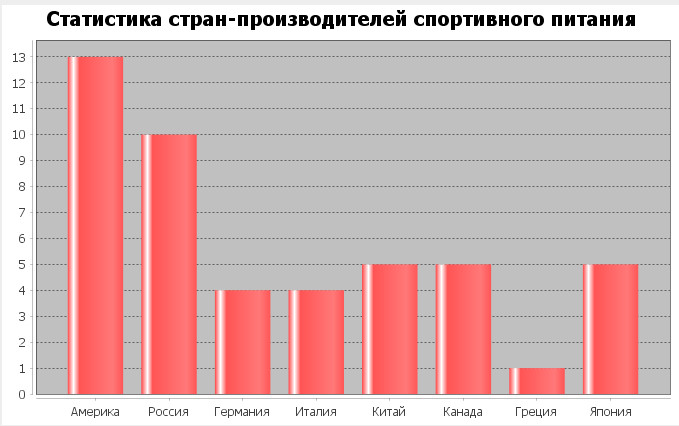


Рисунок 7.15 – Статистика стран-производителей спортивного питания

Столбец слева показывает количества товаров, произведенных определенной страной и которые находятся у нас в ассортименте. На этом функционал пользователя заканчивается. Перейдем к возможностям администратора. Для этого необходимо вернутся в меню аутентификации, выбрать пункт авторизация и заполнить соответствующую форму, введя туда логин - admin и пароль – adminadmin.

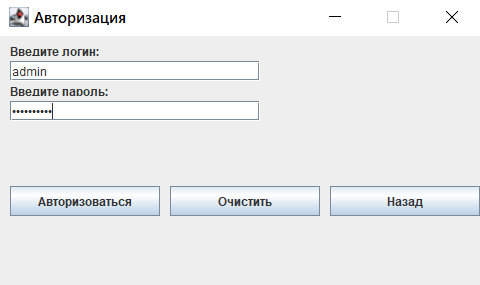


Рисунок 7.16 – Авторизация администратора

После успешной авторизации откроется меню администратора:

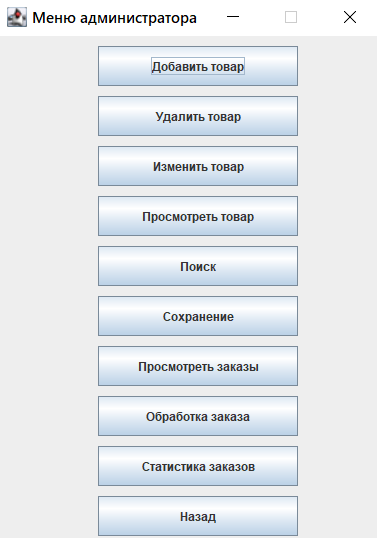


Рисунок 7.17 – Меню администратора

При выборе первого пункта данного меню можно добавить новый товар. Для этого необходимо заполнить форму добавления:

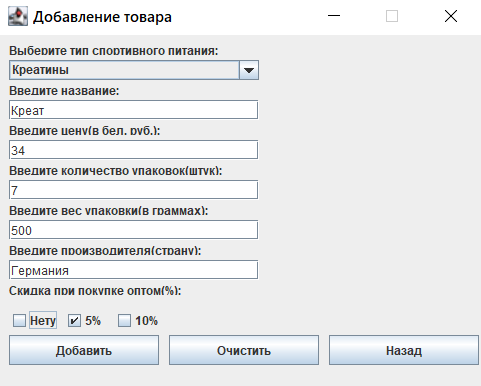


Рисунок 7.18 – Форма добавления нового товара

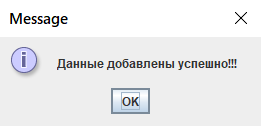


Рисунок 7.19 – Сообщение об успешном добавлении товара

Для добавления данного товара нажму кнопку Добавить. Проверю успешное добавление нового товара. Для этого воспользуюсь функцией Просмотр товаров:

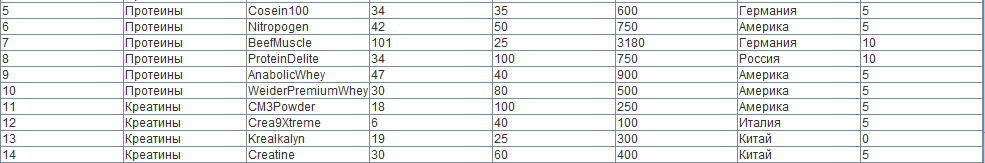


Рисунок 7.20 – Просмотр товаров

Как видно из рисунка 7.20 новый товар был добавлен под номером id 54. Теперь предлагается воспользоваться функцией удаления. Выберете пункт «Удалить товар» в меню администратора. Будет открыта новая форма, в которой необходимо ввести id удаляемого товара:

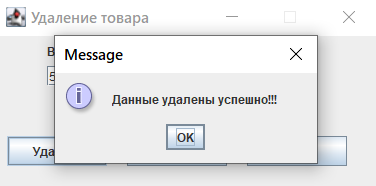


Рисунок 7.21 – Форма удаления и сообщение об успешном удалении

Далее будет продемонстрирована работа функции редактирования данных. Для этого предлагается выбрать пункт «Изменить товар» в меню администратора. Будет выведена форма, в которой необходимо ввести id изменяемого товара, а также новые данные.

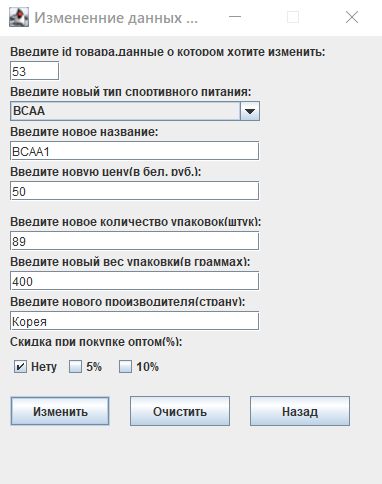


Рисунок 7.22 – Форма редактирования данных о товаре

Следующим пунктом меню является функция поиска данных о товарах. Эта функция реализована аналогично функции поиска у пользователя.

Далее будет продемонстрировано выполнение методов сортировки, которые реализованы в самом объекте JTable. Для этого необходимо получить данные о товарах. Воспользуемся функцией просмотреть товар. Будет выведена таблица с данными о товарах. Для того, чтобы воспользоваться функцией сортировки, необходимо нажать на любую из колонок таблицы один раз, после чего вся колонка будет отсортирована в алфавитном порядке. При нажатии двух раз по колонке сортировка будет произведена в обратную сторону. Для примера отсортируем колонку «Тип»:

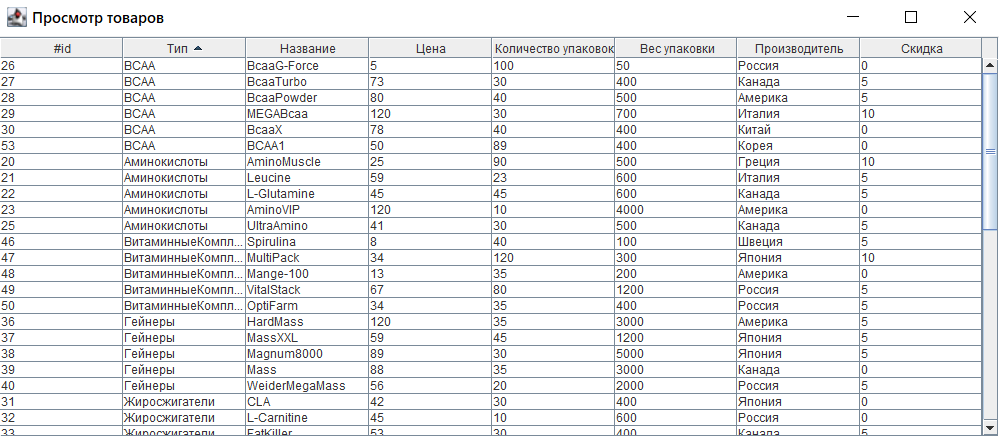


Рисунок 7.23 – Результаты сортировки по колонке «Тип»

Следующим пунктом меню является сохранение данных в файл. Воспользовавшись данной функцией, администратор может сохранить все настроенные изменения. После успешного сохранения все данные о пользователях, товарах и заказах будут записаны в соответствующие файлы:

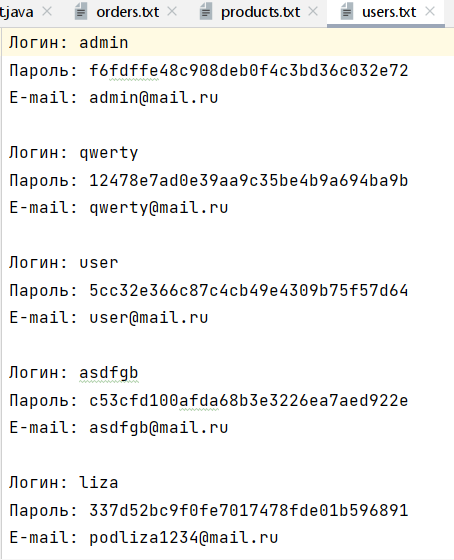


Рисунок 7.24 – Успешное сохранение данных о пользователях в файл

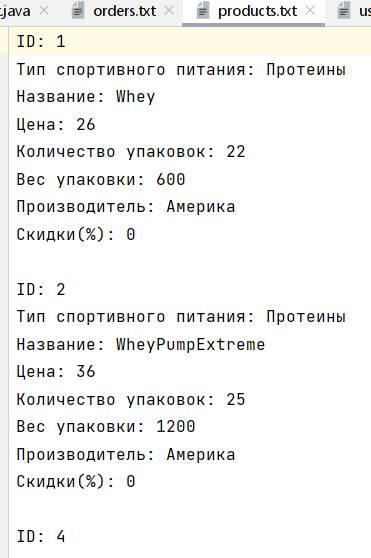


Рисунок 7.25 – Успешное сохранение данных о товарах в файл

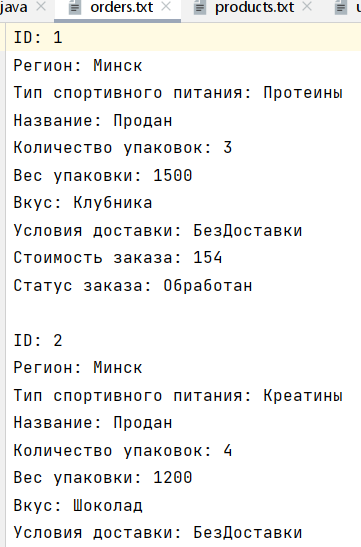


Рисунок 7.26– Успешное сохранения данных о заказах в файл

Следующим пунктом в меню администратора является «Просмотреть заказы». Воспользуемся данной функцией, чтобы получить информацию о заказах:

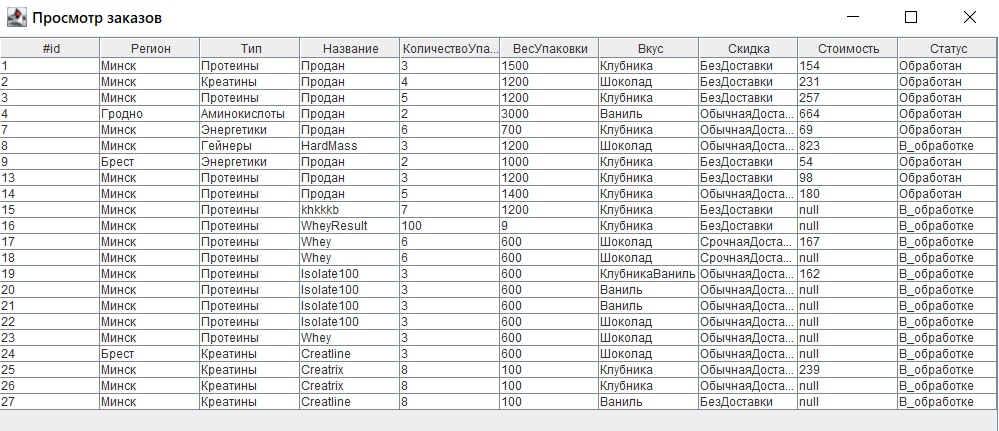


Рисунок 7.27 – Просмотр данных о заказах

Следующий пункт меню является пунктом, реализующим бизнес-логику системы. Этот пункт называется «Обработка заказа». При использовании данного метода появляется форма обработки заказов, в которой необходимо указать id обрабатываемого заказа. Из рисунка 7.26 видно, что заказ под номером 8 находится в обработке. Обработка заказа заключается в том, что статус заказа меняется на обработан и количество упаковок заказа вычитается из количества упаковок в таблице товар. Воспользуемся функцией «Просмотреть товар».

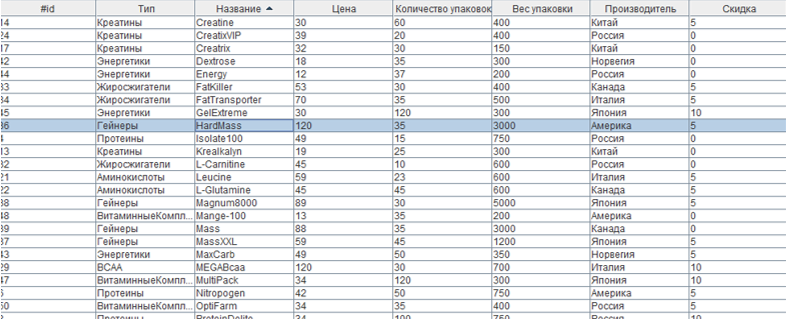


Рисунок 7.28 – Просмотр товара Energy

Из рисунка 7.28 видно, что количество упаковок товара HardMass - 120. Обработаем заказ. Статус заказа поменялся на обработан, а название товара на «Продан». При этом из таблицы Товары было вычтено количество упаковок заказа:

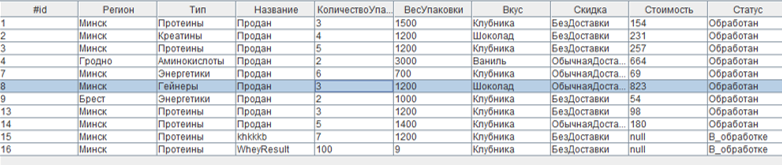


Рисунок 7.29 – Успешная обработка заказа

Последней функцией для администратора является просмотр статистики заказов. Суть данной функции в том, что администратор может видеть тип спортивного питания, на который направлен основной спрос потребителей и соответственно сделать акцент на закупку партий данного товара.

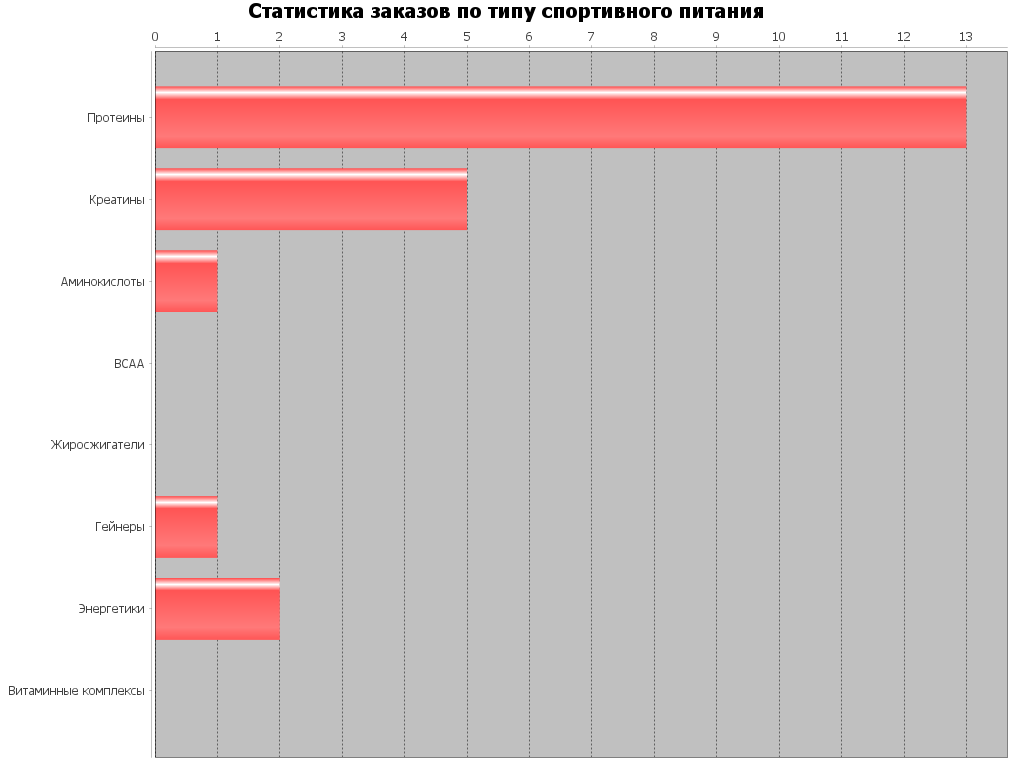


Рисунок 7.30– Статистика заказов по типу спортивного питания

В ходе реализации данной системы были успешно выполнены следующие задачи: улучшилось качество регулирования, была изучена предметная область, а именно магазина спортивного питания, было сделано моделирование системы, а также ее написание, был реализован весь заявленный функционал, произведена автоматизация расчета заказа, доставки и транспортных затрат, улучшилась производительность труда всего персонала магазина, в следствие чего увеличена рентабельность магазина.

# **ЗАКЛЮЧЕНИЕ**

В результате данной курсовой работы была реализована система, которая обладает рядом следующих функций: регистрация, авторизация, поиск по подстроке без учета регистра, добавление, удаление, изменение данных, просмотр данных о товарах и заказах, расчет стоимости доставки, расчет стоимости заказа, сортировка по всем полям, обработка заказа и построение графиков со статистикой. Также выполнена основная задача: автоматизация всех расчетов, увеличение производительности работников и повышение рентабельности магазина.

При разработке данного приложения была использована и изучена методология IDEF0. Для написание данной системы применялись и были применены возможности Java, Swing, JFreeChart, SQL, стек протоколов TCP/IP, а также паттерны DAO, Mediator и Singleton.

На первоначальном этапе проектирования данной системы были разработаны модели, необходимые для её описания, предоставлены требования, исходящие из особенностей работы магазина спортивного питания.

На этапе разработки приложения были учтены все требования, предоставляемые к системе. В результате чего было разработано приложение, позволяющее осуществлять автоматизированный учет товаров магазина, а также поставщиков и покупателей спортивного питания.

Поставленные задачи является выполненными, так как в результате проведено значительное ускорение операций по обработке данных, повысилась точность данных учета, экономия времени на операциях, которые раньше выполняли вручную, расширились возможности магазина по доставке спортивного питания, быстрее и проще стало организовывать и структурировать фонды базы данных, обеспечивать оперативное обслуживание конечных пользователей, а также расширился спектр предоставляемых услуг.

В будущем приложение будет совершенствоваться посредством добавления новых методов, подключением различных библиотек, улучшением дизайна и внедрением новых возможностей.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

[1] https://trinion.org/blog/idef0-znakomstvo-s-notaciey-i-primer-ispolzovaniya

[2] Википедия [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://ru.wikipedia.org/wiki/Диаграмма_развертывания>

[3] Википедия [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://ru.wikipedia.org/wiki/Диаграмма_компонентов>

[4] Википедия [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://ru.wikipedia.org/wiki/Диаграмма_классов>

[5] Хорстманн, К. С. JAVA2. Том 1. Основы. Пер. с англ. Справочное пособие/ Корнелл, Г. – М: ООО «Вильямс», 2003 – 849с.

[6] Аткинсон, Л. MySQL. Библиотека профессионала. Справочное пособие – М: Издательский дом «Вильяме», 2009 – 624с.

[7] Хорстманн, К. С. JAVA2. Том 2. Тонкости программирования. Пер. с англ. Справочное пособие/ Корнелл, Г. – М: ООО «Вильямс», 2003 – 1120с.

# **ПРИЛОЖЕНИЕ А**

**(обязательное)**

**Листинг программного кода**

//интерфейс Products  
package org.example.service.product;  
import java.sql.SQLException;  
public interface Products {  
 String selectRecordFromProductTableForStatistics()throws SQLException;  
 boolean insertRecordIntoProductTable(String clientMessageRecieved) throws SQLException;  
 boolean deleteRecordFromProductTable(String clientMessageRecieved) throws SQLException;  
 boolean changeRecordIntoProductTable(String clientMessageRecieved) throws SQLException;  
 String selectRecordFromProductTable() throws SQLException;  
 String selectRecordFromProductTableForSearch(String clientMessageRecieved)throws SQLException;  
}

// класс ProductsImpl со всем функционаломpackage org.example.service.product;  
import lombok.AllArgsConstructor;  
import lombok.Getter;  
import lombok.Setter;  
import org.example.db.MySqlDaoFactory;  
import org.example.entity.Product;  
import java.sql.Connection;  
import java.sql.ResultSet;  
import java.sql.SQLException;  
import java.sql.Statement;

@Getter  
@Setter  
@AllArgsConstructor  
public class ProductsImpl extends MySqlDaoFactory implements Products {  
  
 private static ProductsImpl *productsImpl*;  
 private static Product *product*;  
  
 public static synchronized ProductsImpl getInstance() {  
 if (*productsImpl* == null) {

Продолжение приложения А

*productsImpl* = new ProductsImpl();  
 }  
 return *productsImpl*;  
 }  
 private static Connection *dbConnection* = null;  
 private static Statement *statement* = null;  
 @Override  
 public String selectRecordFromProductTableForStatistics() {  
 String res = "";  
 int countUSA = 0;  
 int countRussia = 0;  
 int countGermany = 0;  
 int countItaly = 0;  
 int countChina = 0;  
 int countCanada = 0;  
 int countGreece = 0;  
 int countJapan = 0;  
 String selectTableSQL = "SELECT manufacturer FROM product";  
 try (Connection dbConnection = getConnection(); Statement statement = dbConnection.createStatement()) {  
 ResultSet rs = statement.executeQuery(selectTableSQL);  
 while (rs.next()) {  
 String manufacturer = rs.getString("manufacturer");  
 if (manufacturer.equals("Америка")) {  
 countUSA++;  
 }  
 if (manufacturer.equals("Россия")) {  
 countRussia++;  
 }  
 if (manufacturer.equals("Германия")) {  
 countGermany++;  
 }  
 if (manufacturer.equals("Италия")) {  
 countItaly++;  
 }  
 if (manufacturer.equals("Китай")) {  
 countChina++;  
 }  
 if (manufacturer.equals("Канада")) {  
 countCanada++;  
 }  
 if (manufacturer.equals("Греция")) {

Продолжение приложения А

countGreece++;  
 }  
 if (manufacturer.equals("Япония")) {  
 countJapan++;  
 }  
 }  
 res = countUSA + " " + countRussia + " " + countGermany + " " +  
 countItaly + " " + countChina + " " + countCanada + " " +  
 countGreece + " " + countJapan;  
 } catch (SQLException e) {  
 System.*out*.println(e.getMessage());  
 }  
 return res;  
 }  
 @Override  
 public boolean insertRecordIntoProductTable(String clientMessageRecieved) throws SQLException {  
 int flag = 0;  
 String[] message = clientMessageRecieved.split(" ");  
 String type = message[1];  
 String name = message[2];  
 String price = message[3];  
 int iPrice = Integer.*parseInt*(price);  
 String numberOfPackages = message[4];  
 int iNumberOfPackages = Integer.*parseInt*(numberOfPackages);  
 String weightOfPacking = message[5];  
 int iWeightOfPacking = Integer.*parseInt*(weightOfPacking);  
 String manufacturer = message[6];  
 String accountingDiscounts = message[7];  
 int iAccountingDiscounts = Integer.*parseInt*(accountingDiscounts);  
 String insertTableSQL = "INSERT INTO product"  
 + "(type, name, price, numberOfPackages, WeightOfPacking, manufacturer, accountingDiscounts) " + "VALUES"  
 + "('" + type + "'," + "'" + name + "'," + "'" + iPrice + "'," + "'" + iNumberOfPackages + "'," + "'" + iWeightOfPacking + "'," + "'" + manufacturer + "'," + "'" + iAccountingDiscounts + "');";  
 try {  
 *dbConnection* = getConnection();  
 *statement* = *dbConnection*.createStatement();  
 *statement*.executeUpdate(insertTableSQL);  
 flag = 1;  
 } catch (SQLException e) {

Продолжение приложения А

System.*out*.println(e.getMessage());  
 } finally {  
 if (*statement* != null) {  
 *statement*.close();  
 }  
 if (*dbConnection* != null) {  
 *dbConnection*.close();  
 }  
 }  
 return flag != 0;  
 }  
 @Override  
 public boolean deleteRecordFromProductTable(String clientMessageRecieved) throws SQLException {  
 int flag = 0;  
 String[] message = clientMessageRecieved.split(" ");  
 String id = message[1];  
 String selectTableSQL = "SELECT id FROM product";  
 String deleteTableSQL = "DELETE FROM `registrationdb`.`product` WHERE `id`='" + id + "'";  
 try {  
 *dbConnection* = getConnection();  
 *statement* = *dbConnection*.createStatement();  
 ResultSet rs = *statement*.executeQuery(selectTableSQL);  
 while (rs.next()) {  
 String idTable = rs.getString("id");  
 if (id.equals(idTable)) {  
 flag = 1;  
 break;  
 }  
 }  
 if (flag == 1) {  
 *statement*.execute(deleteTableSQL);  
 }  
 } catch (SQLException e) {  
 System.*out*.println(e.getMessage());  
 } finally {  
 if (*statement* != null) {  
 *statement*.close();  
 }  
 if (*dbConnection* != null) {  
 *dbConnection*.close();

Продолжение приложения А

}  
 }  
 return flag != 0;  
 }  
 @Override  
 public boolean changeRecordIntoProductTable(String clientMessageRecieved) throws SQLException {  
 int flag = 0;  
 String[] message = clientMessageRecieved.split(" ");  
 String id = message[1];  
 String type = message[2];  
 String name = message[3];  
 String price = message[4];  
 int iPrice = Integer.*parseInt*(price);  
 String numberOfPackages = message[5];  
 int iNumberOfPackages = Integer.*parseInt*(numberOfPackages);  
 String weightOfPacking = message[6];  
 int iWeightOfPacking = Integer.*parseInt*(weightOfPacking);  
 String manufacturer = message[7];  
 String accountingDiscounts = message[8];  
 int iAccountingDiscounts = Integer.*parseInt*(accountingDiscounts);  
 String selectTableSQL = "SELECT id FROM product";  
 String updateTableSQL = "UPDATE `registrationdb`.`product` SET `type`='" + type + "'" +  
 ", `name`='" + name + "'" + ", `price`='" + iPrice + "'" +  
 ", `numberOfPackages`='" + iNumberOfPackages + "'" + ", `weightOfPacking`='" + iWeightOfPacking + "'" +  
 ", `manufacturer`='" + manufacturer + "'" + ", `accountingDiscounts`='" + iAccountingDiscounts + "'" + " WHERE `id`='" + id + "'";  
 try {  
 *dbConnection* = getConnection();  
 *statement* = *dbConnection*.createStatement();  
 ResultSet rs = *statement*.executeQuery(selectTableSQL);  
 while (rs.next()) {  
 String idTable = rs.getString("id");  
 if (id.equals(idTable)) {  
 flag = 1;  
 break;  
 }  
 }  
 if (flag == 1) {  
 *statement*.execute(updateTableSQL);

Продолжение приложения А

}  
 } catch (SQLException e) {  
 System.*out*.println(e.getMessage());  
 } finally {  
 if (*statement* != null) {  
 *statement*.close();  
 }  
 if (*dbConnection* != null) {  
 *dbConnection*.close();  
 }  
 }  
 return flag != 0;  
 }  
 @Override  
 public String selectRecordFromProductTable() throws SQLException {  
 Connection dbConnection = null;  
 Statement statement = null;  
 String selectTableSQL = "SELECT \* FROM product";  
 String result = "";  
 try {  
 dbConnection = getConnection();  
 statement = dbConnection.createStatement();  
 ResultSet rs = statement.executeQuery(selectTableSQL);  
 while (rs.next()) {  
 *product*.setId(rs.getString("id"));  
 *product*.setType(rs.getString("type"));  
 *product*.setName(rs.getString("name"));  
 *product*.setPrice(rs.getString("price"));  
 *product*.setNumberOfPackages(rs.getString("numberOfPackages"));  
 *product*.setWeightOfPacking(rs.getString("weightOfPacking"));  
 *product*.setManufacturer(rs.getString("manufacturer"));  
 *product*.setAccountingDiscounts(rs.getString("accountingDiscounts"));  
 String id = rs.getString("id");  
 String type = rs.getString("type");  
 String name = rs.getString("name");  
 String price = rs.getString("price");  
 String numberOfPackages = rs.getString("numberOfPackages");  
 String weightOfPacking = rs.getString("weightOfPacking");  
 String manufacturer = rs.getString("manufacturer");  
 String accountingDiscounts = rs.getString("accountingDiscounts");  
 result += id + " " + type + " " + name + " " + price + " " + numberOfPackages + " " +

Продолжение приложения А

weightOfPacking + " " + manufacturer + " " + accountingDiscounts + " ";  
 }  
  
 } catch (SQLException e) {  
 System.*out*.println(e.getMessage());  
 } finally {  
 if (statement != null) {  
 statement.close();  
 }  
 if (dbConnection != null) {  
 dbConnection.close();  
 }  
 }  
 return result;  
 }  
 @Override  
 public String selectRecordFromProductTableForSearch(String clientMessageRecieved) {  
 String[] message = clientMessageRecieved.split(" ");  
 String searchID = message[1];  
 int searchIntID = Integer.*parseInt*(searchID);  
 String searchData = message[2];  
 String result = "";  
 switch (searchIntID) {  
 case 1: {  
 String selectTableSQL = "SELECT \* FROM product WHERE type LIKE '%" + searchData + "%'";  
 try {  
 *dbConnection* = getConnection();  
 *statement* = *dbConnection*.createStatement();  
 ResultSet rs = *statement*.executeQuery(selectTableSQL);  
 while (rs.next()) {  
 String id = rs.getString("id");  
 String type = rs.getString("type");  
 String name = rs.getString("name");  
 String price = rs.getString("price");  
 String numberOfPackages = rs.getString("numberOfPackages");  
 String weightOfPacking = rs.getString("weightOfPacking");  
 String manufacturer = rs.getString("manufacturer");  
 String accountingDiscounts = rs.getString("accountingDiscounts");  
 result += id + " " + type + " " + name + " " + price + " " +

Продолжение приложения А

numberOfPackages + " " +  
 weightOfPacking + " " + manufacturer + " " + accountingDiscounts + " ";  
 }

} catch (SQLException e) {  
 System.*out*.println(e.getMessage());  
 }  
 break;  
 }  
 case 2: {  
 String selectTableSQL = "SELECT \* FROM product WHERE name LIKE '%" + searchData + "%'";  
 try {  
 *dbConnection* = getConnection();  
 *statement* = *dbConnection*.createStatement();  
 ResultSet rs = *statement*.executeQuery(selectTableSQL);  
 while (rs.next()) {  
 String id = rs.getString("id");  
 String type = rs.getString("type");  
 String name = rs.getString("name");  
 String price = rs.getString("price");  
 String numberOfPackages = rs.getString("numberOfPackages");  
 String weightOfPacking = rs.getString("weightOfPacking");  
 String manufacturer = rs.getString("manufacturer");  
 String accountingDiscounts = rs.getString("accountingDiscounts");  
 result += id + " " + type + " " + name + " " + price + " " + numberOfPackages + " " +  
 weightOfPacking + " " + manufacturer + " " + accountingDiscounts + " ";  
 }  
 } catch (SQLException e) {  
 System.*out*.println(e.getMessage());  
 }  
 break;  
 }  
 case 3: {  
 String selectTableSQL = "SELECT \* FROM product WHERE price LIKE '%" + searchData + "%'";  
 try {  
 *dbConnection* = getConnection();  
 *statement* = *dbConnection*.createStatement();  
 ResultSet rs = *statement*.executeQuery(selectTableSQL);

Продолжение приложения А

while (rs.next()) {  
 String id = rs.getString("id");  
 String type = rs.getString("type");

String name = rs.getString("name");  
 String price = rs.getString("price");  
 String numberOfPackages = rs.getString("numberOfPackages");  
 String weightOfPacking = rs.getString("weightOfPacking");  
 String manufacturer = rs.getString("manufacturer");  
 String accountingDiscounts = rs.getString("accountingDiscounts");  
 result += id + " " + type + " " + name + " " + price + " " + numberOfPackages + " " +  
 weightOfPacking + " " + manufacturer + " " + accountingDiscounts + " ";  
 }  
 } catch (SQLException e) {  
 System.*out*.println(e.getMessage());  
 }  
 break;  
 }  
 case 4: {  
 String selectTableSQL = "SELECT \* FROM product WHERE numberOfPackages LIKE '%" + searchData + "%'";  
 try {  
 *dbConnection* = getConnection();  
 *statement* = *dbConnection*.createStatement();  
 ResultSet rs = *statement*.executeQuery(selectTableSQL);  
 while (rs.next()) {  
 String id = rs.getString("id");  
 String type = rs.getString("type");  
 String name = rs.getString("name");  
 String price = rs.getString("price");  
 String numberOfPackages = rs.getString("numberOfPackages");  
 String weightOfPacking = rs.getString("weightOfPacking");  
 String manufacturer = rs.getString("manufacturer");  
 String accountingDiscounts = rs.getString("accountingDiscounts");  
 result += id + " " + type + " " + name + " " + price + " " + numberOfPackages + " " +  
 weightOfPacking + " " + manufacturer + " " + accountingDiscounts + " ";  
 }  
 } catch (SQLException e) {  
 System.*out*.println(e.getMessage());

Продолжение приложения А

}  
 break;  
 }  
 case 5: {  
 String selectTableSQL = "SELECT \* FROM product WHERE weightOfPacking LIKE '%" + searchData + "%'";  
 try {  
 *dbConnection* = getConnection();  
 *statement* = *dbConnection*.createStatement();  
 ResultSet rs = *statement*.executeQuery(selectTableSQL);  
 while (rs.next()) {  
 String id = rs.getString("id");  
 String type = rs.getString("type");  
 String name = rs.getString("name");  
 String price = rs.getString("price");  
 String numberOfPackages = rs.getString("numberOfPackages");  
 String weightOfPacking = rs.getString("weightOfPacking");  
 String manufacturer = rs.getString("manufacturer");  
 String accountingDiscounts = rs.getString("accountingDiscounts");  
 result += id + " " + type + " " + name + " " + price + " " + numberOfPackages + " " +  
 weightOfPacking + " " + manufacturer + " " + accountingDiscounts + " ";  
 }  
 } catch (SQLException e) {  
 System.*out*.println(e.getMessage());  
 }  
 break;  
 }  
 case 6: {  
 String selectTableSQL = "SELECT \* FROM product WHERE manufacturer LIKE '%" + searchData + "%'";  
 try {  
 *dbConnection* = getConnection();  
 *statement* = *dbConnection*.createStatement();  
 ResultSet rs = *statement*.executeQuery(selectTableSQL);  
 while (rs.next()) {  
 String id = rs.getString("id");  
 String type = rs.getString("type");  
 String name = rs.getString("name");  
 String price = rs.getString("price");  
 String numberOfPackages = rs.getString("numberOfPackages");

Продолжение приложения А

String weightOfPacking = rs.getString("weightOfPacking");  
 String manufacturer = rs.getString("manufacturer");  
 String accountingDiscounts = rs.getString("accountingDiscounts");  
 result += id + " " + type + " " + name + " " + price + " " + numberOfPackages + " " +  
 weightOfPacking + " " + manufacturer + " " + accountingDiscounts + " ";  
 }  
 } catch (SQLException e) {  
 System.*out*.println(e.getMessage());  
 }  
 break;  
 }  
 default: {  
 System.*out*.println("Введено неверное значение!!!");  
 break;  
 }  
 }  
 return result;  
 }  
}

# **ПРИЛОЖЕНИЕ Б**

**(обязательное)**

**SQL – скрипт для генерации базы данных**

create table registrationdb.orders

(

id int auto\_increment

primary key,

region varchar(45) null,

type varchar(45) null,

name varchar(45) null,

numberOfPackages int null,

weightOfPacking int null,

taste varchar(45) null,

delivery varchar(45) null,

orderCost int null,

status varchar(45) null

)

charset = utf8mb3;

create table registrationdb.product

(

id int auto\_increment

primary key,

type varchar(45) null,

name varchar(45) null,

price int null,

numberOfPackages int null,

weightOfPacking int null,

manufacturer varchar(45) null,

accountingDiscounts varchar(45) null

)

charset = utf8mb3;

create table registrationdb.registration

(

id int auto\_increment

primary key,

login varchar(45) null,

Продолжение приложения Б

password varchar(45) null,

email varchar(45) null

)

charset = utf8mb3;

insert into registrationdb.registration (id, login, password, email)

values

(1, 'admin', 'f6fdffe48c908deb0f4c3bd36c032e72', 'admin@mail.ru'),

# pass:adminadmin

(2, 'qwerty', '12478e7ad0e39aa9c35be4b9a694ba9b', 'qwerty@mail.ru'),

# pass:qwertyqwerty

(3, 'user', '5cc32e366c87c4cb49e4309b75f57d64', 'user@mail.ru');

# pass:useruser

insert into registrationdb.product (id, type, name, price, numberOfPackages, weightOfPacking, manufacturer, accountingDiscounts)

values (1, 'Протеины', 'Whey', 26, 22, 600, 'Америка', '0'),

(2, 'Протеины', 'WheyPumpExtreme', 36, 25, 1200, 'Америка', '0'),

(4, 'Протеины', 'Isolate100', 49, 15, 750, 'Россия', '0'),

(5, 'Протеины', 'Cosein100', 34, 35, 600, 'Германия', '5'),

(6, 'Протеины', 'Nitropogen', 42, 50, 750, 'Америка', '5'),

(7, 'Протеины', 'BeefMuscle', 101, 25, 3180, 'Германия', '10'),

(8, 'Протеины', 'ProteinDelite', 34, 100, 750, 'Россия', '10'),

(9, 'Протеины', 'AnabolicWhey', 47, 40, 900, 'Америка', '5'),

(10, 'Протеины', 'WeiderPremiumWhey', 30, 80, 500, 'Америка', '5'),

(11, 'Креатины', 'CM3Powder', 18, 100, 250, 'Америка', '5'),

(12, 'Креатины', 'Crea9Xtreme', 6, 40, 100, 'Италия', '5'),

(13, 'Креатины', 'Krealkalyn', 19, 25, 300, 'Китай', '0'),

(14, 'Креатины', 'Creatine', 30, 60, 400, 'Китай', '5'),

(15, 'Креатины', 'ReactroPro', 21, 40, 150, 'Россия', '5'),

(16, 'Протеины', 'CM31250', 10, 25, 200, 'Америка', '0'),

(17, 'Креатины', 'Creatrix', 32, 30, 150, 'Китай', '0'),

(18, 'Креатины', 'Crea9', 28, 100, 200, 'Америка', '10'),

(19, 'Креатины', 'TCMPowder', 28, 35, 300, 'Германия', '0'),

Продолжение приложения Б

(20, 'Аминокислоты', 'AminoMuscle', 25, 90, 500, 'Греция', '10'),

(21, 'Аминокислоты', 'Leucine', 59, 23, 600, 'Италия', '5'),

(22, 'Аминокислоты', 'L-Glutamine', 45, 45, 600, 'Канада', '5'),

(23, 'Аминокислоты', 'AminoVIP', 120, 10, 4000, 'Америка', '0'),

(24, 'Креатины', 'CreatixVIP', 39, 20, 400, 'Россия', '0'),

(25, 'Аминокислоты', 'UltraAmino', 41, 30, 500, 'Канада', '5'),

(26, 'BCAA', 'BcaaG-Force', 5, 100, 50, 'Россия', '0'),

(27, 'BCAA', 'BcaaTurbo', 73, 30, 400, 'Канада', '5'),

(28, 'BCAA', 'BcaaPowder', 80, 40, 500, 'Америка', '5'),

(29, 'BCAA', 'MEGABcaa', 120, 30, 700, 'Италия', '10'),

(30, 'BCAA', 'BcaaX', 78, 40, 400, 'Китай', '0'),

(31, 'Жиросжигатели', 'CLA', 42, 30, 400, 'Япония', '0'),

(32, 'Жиросжигатели', 'L-Carnitine', 45, 10, 600, 'Россия', '0'),

(33, 'Жиросжигатели', 'FatKiller', 53, 30, 400, 'Канада', '5'),

(34, 'Жиросжигатели', 'FatTransporter', 70, 35, 500, 'Италия', '5'),

(35, 'Жиросжигатели', 'TurboRipper', 27, 100, 400, 'Китай', '10'),

(36, 'Гейнеры', 'HardMass', 120, 35, 3000, 'Америка', '5'),

(37, 'Гейнеры', 'MassXXL', 59, 45, 1200, 'Япония', '5'),

(38, 'Гейнеры', 'Magnum8000', 89, 30, 5000, 'Япония', '5'),

(39, 'Гейнеры', 'Mass', 88, 35, 3000, 'Канада', '0'),

(40, 'Гейнеры', 'WeiderMegaMass', 56, 20, 2000, 'Россия', '5'),

(41, 'Энергетики', 'Redfaster', 10, 120, 150, 'Америка', '10'),

(42, 'Энергетики', 'Dextrose', 18, 35, 300, 'Норвегия', '0'),

(43, 'Энергетики', 'MaxCarb', 49, 50, 350, 'Норвегия', '5'),

(44, 'Энергетики', 'Energy', 12, 37, 200, 'Россия', '0'),

(45, 'Энергетики', 'GelExtreme', 30, 120, 300, 'Япония', '10'),

(46, 'ВитаминныеКомплексы', 'Spirulina', 8, 40, 100, 'Швеция', '5'),

(47, 'ВитаминныеКомплексы', 'MultiPack', 34, 120, 300, 'Япония', '10'),

(48, 'ВитаминныеКомплексы', 'Mange-100', 13, 35, 200, 'Америка', '0'),

(49, 'ВитаминныеКомплексы', 'VitalStack', 67, 80, 1200, 'Россия', '5'),

(50, 'ВитаминныеКомплексы', 'OptiFarm', 34, 35, 400, 'Россия', '5'),

(53, 'Протеины', 'БизонPro', 34, 50, 1200, 'Беларусь', '5');

Продолжение приложения Б

insert into registrationdb.orders (id, region, type, name, numberOfPackages, weightOfPacking, taste, delivery, orderCost, status)

values (1, 'Минск', 'Протеины', 'Продан', 3, 1500, 'Клубника', 'БезДоставки', 154, 'Обработан'),

(2, 'Минск', 'Креатины', 'Продан', 4, 1200, 'Шоколад', 'БезДоставки', 231, 'Обработан'),

(3, 'Минск', 'Протеины', 'Продан', 5, 1200, 'Клубника', 'БезДоставки', 257, 'Обработан'),

(4, 'Гродно', 'Аминокислоты', 'Продан', 2, 3000, 'Ваниль', 'ОбычнаяДоставка', 664, 'Обработан'),

(7, 'Минск', 'Энергетики', 'Продан', 6, 700, 'Клубника', 'ОбычнаяДоставка', 69, 'Обработан'),

(8, 'Минск', 'Гейнеры', 'HardMass', 3, 1200, 'Шоколад', 'ОбычнаяДоставка', 823, 'В\_обработке'),

(9, 'Брест', 'Энергетики', 'Продан', 2, 1000, 'Клубника', 'БезДоставки', 54, 'Обработан'),

(13, 'Минск', 'Протеины', 'Продан', 3, 1200, 'Клубника', 'БезДоставки', 98, 'Обработан'),

(14, 'Минск', 'Протеины', 'Продан', 5, 1400, 'Клубника', 'ОбычнаяДоставка', 180, 'Обработан'),

(15, 'Минск', 'Протеины', 'khkkkb', 7, 1200, 'Клубника', 'БезДоставки', null, 'В\_обработке'),

(16, 'Минск', 'Протеины', 'WheyResult', 100, 9, 'Клубника', 'БезДоставки', null, 'В\_обработке');