МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Кафедра «Информационно-вычислительные системы»

Курсовая работа

по дисциплине «Моделирование и проектирование систем»

на тему «Проектирование и программная реализация автоматизированной информационной системы магазина рыболовных принадлежностей»

ПГУ 1.090501.6

Специальность 09.05.01 «Применение и эксплуатация автоматизированных систем специального назначения»

Специализация №12 «Автоматизированные системы обработки информации и управления специального назначения»

|  |  |
| --- | --- |
|  | Выполнил студент гр. 18ВО1  \_\_\_\_\_\_\_\_\_\_Кравчук М.В.  Руководитель: к.т.н., доцент кафедры ИВС  \_\_\_\_\_\_\_\_\_\_Дрождин В.В. |

|  |  |
| --- | --- |
| Работа защищена с оценкой  Преподаватели  Дата защиты | \_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_ |

2022

Реферат

Изм.

Лист

№ докум.

Подпись

Дата

Лист

3

ПГУ 1.090501.06

Разраб.

Кравчук М.В.

Провер.

Дрождин В.В.

Н. Контр.

.

Утв.

Проектирование и программная реализация автоматизированной информационной системы магазина рыболовных принадлежностей

Пояснительная записка

Лит.

Листов

82

гр. 18ВО1

Пояснительная записка состоит из 82 листов, 20 рисунков, 10 таблиц, 2 приложения.

Объектом исследования является проектирование и программная реализация автоматизированной информационной системы магазина рыболовных принадлежностей.

Цель работы – спроектировать и реализоваться программно-автоматизированную систему магазина рыболовных принадлежностей.

В результате проделанной работы была спроектирована и программно-реализована автоматизированная система магазина рыболовных принадлежностей.

При разработке программного приложения использовалась среда визуального программирования Microsoft Visual Studio 2019 Community.

Содержание

[Введение 6](#_Toc103159148)

[1 Анализ предметной области 8](#_Toc103159149)

[1.1 Проведение обследования автоматизируемого объекта 8](#_Toc103159150)

[1.2 Построение модели предметной области «как есть» (AS-IS) 10](#_Toc103159151)

[1.2.1 Разработка диаграммы пакетов 10](#_Toc103159152)

[1.2.2 Разработка диаграммы вариантов использования 11](#_Toc103159153)

[1.2.3 Разработка диаграмм последовательности 12](#_Toc103159154)

[1.2.4 Разработка диаграмм деятельности 13](#_Toc103159155)

[1.3 Построение модели предметной области «как должно быть» (TO -BE) 16](#_Toc103159156)

[1.3.1 Разработка диаграммы пакетов 17](#_Toc103159157)

[1.3.2 Разработка диаграммы вариантов использования 17](#_Toc103159158)

[1.3.3 Разработка диаграммы последовательности 18](#_Toc103159159)

[1.3.4 Разработка диаграммы деятельности 18](#_Toc103159160)

[2 Техническое задание 22](#_Toc103159161)

[2.1 Основание для разработки 22](#_Toc103159162)

[2.2 Назначение разработки 22](#_Toc103159163)

[2.3.1 Требования к функциональным характеристикам 22](#_Toc103159164)

[2.3.2 Требования к составу и параметрам технических средств 23](#_Toc103159165)

[2.3.3 Требования к информационной и программной совместимости 23](#_Toc103159166)

[2.4 Требования к программной документации 23](#_Toc103159167)

[2.5 Стадии и этапы разработки 23](#_Toc103159168)

[2.6 Порядок контроля и приемки 24](#_Toc103159169)

[3 Проектирование автоматизированной системы 26](#_Toc103159170)

[3.1 Технико-экономическое обоснование 26](#_Toc103159171)

[3.1.1 Техническое обоснование 26](#_Toc103159172)

[3.1.2 Экономическое обоснование 27](#_Toc103159173)

[3.2 Разработка UML – модели автоматизированной системы 35](#_Toc103159174)

[3.2.1 Разработка диаграммы пакетов АС 35](#_Toc103159175)

[3.2.2 Разработка диаграммы вариантов использования АС 35](#_Toc103159176)

[3.2.3 Разработка диаграммы последовательности АС 35](#_Toc103159177)

[3.2.4 Разработка диаграммы деятельности АС 36](#_Toc103159178)

[3.2.5 Разработка диаграммы классов 36](#_Toc103159179)

[3.2.6 Разработка диаграммы развертывания 38](#_Toc103159180)

[3.3 Проектирование базы данных 38](#_Toc103159181)

[4 Программа и методика испытаний 41](#_Toc103159182)

[4.1 Объект испытаний 41](#_Toc103159183)

[4.2 Цель испытаний 41](#_Toc103159184)

[4.3 Требования к программе 41](#_Toc103159185)

[4.4 Средства и порядок испытаний 41](#_Toc103159186)

[4.5 Тестирование системы 42](#_Toc103159187)

[5 Руководство по использованию 44](#_Toc103159188)

[5.1 Руководство администратора системы 44](#_Toc103159189)

[5.2 Руководство пользователя 44](#_Toc103159190)

[Заключение 46](#_Toc103159191)

[Список использованных источников 47](#_Toc103159192)

[ТЕКСТ ПРОГРАММЫ 48](#_Toc103159193)

[ЭКРАННЫЕ ФОРМЫ 75](#_Toc103159196)

# Введение

В современных требованиях, предъявляемых к качеству работы торгово-розничных предприятий, отмечается, что эффективная работа сотрудников таких предприятий полностью зависит от оснащения их информационными средствами и их успешного использования в системе автоматизированного учёта товаров [1].

Компьютерный учёт товара полностью отличается от классического, рукописного. Компьютерные программы упрощают учёт товаров, сокращают время, требуемое на оформление документов для анализа торговой деятельности, следовательно, при применении компьютерных программ, повышается эффективность работы персонала торгового предприятия, уменьшается время обучения персонала [2].

Результаты выполнения торговых операций заносятся в соответствующие журналы, что позволяет автоматически их сохранять и использовать в дальнейшем.

Основные преимущества автоматизации учёта: экономия времени, сжатие хранимых данных с экономией объёма памяти и снижение затрат на операции обновления данных. При этом информационная система автоматизирует и ведёт учёт товаров, поставку и отпуск товаров со склада предприятия. Всё это будет сопутствовать более качественному обслуживанию, повышению результативности работы предприятия, повышению точности учёта и снижению потерь товара.

Актуальность темы данной выпускной квалификационной работы связана с необходимостью автоматизирования процесса учёта товаров в организации, которая занимается розничной торговлей рыболовных принадлежностей.

Цель данной курсовой работы состоит в разработке информационной системы учёта товаров, которая позволит повысить производительность труда персонала организации, в виде программного приложения.

Для достижения поставленной цели курсовой работы необходимо решить следующие задачи:

1) провести анализ предметной области, в которой требуется применение информационной системы;

2) спроектировать структуру будущей информационной системы;

3) спроектировать структуру базы данных;

4) выполнить программную реализацию информационной системы;

5) тестировать информационную систему.

Объектом исследования курсовой работы является программная реализация автоматизированной информационной системы магазина рыболовных принадлежностей.

# 1 Анализ предметной области

## 1.1 Проведение обследования автоматизируемого объекта

Магазин рыболовных принадлежностей — место, предоставляющее услуги по покупке всевозможного снаряжения, используемого рыболовами при ловле рыбы. Магазин рыболовных принадлежностей выполняет функции координации и обеспечению торгового процесса, ведения делопроизводства. Магазин рыболовных принадлежностей контролирует работу сотрудников магазина и покупателей на предмет купли-продажи товара, осуществляет общее руководство работой сотрудников. Магазин рыболовных принадлежностей занимается контролем закупкой и продажей товара [4].

Организация производственного процесса в магазине невозможна без четкого взаимодействия всех сотрудников. Именно сотрудники отвечают за потребность магазина в различной продукции, занимаются ее складированием, учетом и продажей.

Сотрудникам магазина рыболовных принадлежностей при решении данной задачи предстоит изучение спроса и предложения по всей потребляемой продукции, а также анализ уровня и колебания цен на продукцию, на услуги посредников, поиск самого выгодного варианта товародвижения, оптимизация своих запасов с учетом сокращения складских и транспортно-заготовительных издержек.

В процессе работы магазина рыболовных принадлежностей возникает вопрос в автоматизации функций учета и обработки, собираемой им информации. Данная проблема решается путем внедрения в работу отдела снабжения автоматизированной информационной системы.

На рынке представлен ряд АИС, специализирующихся на автоматизации работы магазинов. Одной из таких систем является «1С Розничная торговля магазин "Рыбалка"» (Далее 1С) – современное многофункциональное решение для предприятия.

В возможности данной системы входят:

* учет закупок;
* учет данных о поставщиках;
* учет услуг;
* учет закупок для нужд магазина;
* учет закупки ПО.

К достоинствам данного решения можно отнести:

* облачное хранение данных;
* возможность применения как для крупных предприятий, так и для малых бизнесов.

Минусами АИС 1С являются:

* высокая стоимость;
* необходимость обучения персонала работе с данной системой, что влечет за собой дополнительные материальные и временные расходы.
* необходимость доработки АИС под конкретный магазин;
* необходимость заказывать услуги поддержки 1С;
* платные обновления продуктов;
* низкая безопасность и защищенность информации.

Рассмотрев приведенное выше программное решение, можно сделать вывод, что оно не подходит для автоматизации работы магазина рыболовных принадлежностей по ряду причин:

* + 1. Имеет избыточный функционал, который не будет востребован в работе, но будет влиять на итоговую стоимость продукта.
    2. Слабый механизм отчетности, что является ключевым моментом для руководителей организации, приобретающей данную систему.
    3. Стоимость решения, так как ключевым фактором для приобретения системы являются финансовые возможности организации.

Таким образом, задача автоматизации работы отдела снабжения, путем разработки новой автоматизированной системы, подходящей под требования магазина и устраняющей недостатки уже существующей системы является актуальной.

## 1.2 Построение модели предметной области «как есть» (AS-IS)

Модель AS – IS «как есть» - функциональная модель предназначена для описания существующей бизнес модели– по разработке и ведению автоматизированной информационной системы магазина рыболовных принадлежностей. Модель AS-IS разрабатывается на языке UML.

### 1.2.1 Разработка диаграммы пакетов

Диаграммы пакетов унифицированного языка моделирования (UML) отображают зависимости между пакетами, составляющими модель [3].

Целью диаграммы пакетов, в первую очередь является организация элементов в группы по какому – либо признаку с целью упрощения структуры и организации работы с моделью системы.

Пакет (package) – это конструкция UML, предназначенная для упорядочивания UML – моделей, а также для группировки классов.

Пакет, во – первых, выполняет служебную роль, позволяя организовать порядок в создаваемых UML – моделях и распределить различные модельные конструкции, а также диаграммы, по разным «папкам».

Во – вторых, в пакеты традиционно помещают классы системы, особенно если проект большой и их много.

Диаграмма пакетов модели «как есть» представлена на рисунке 1.



Рисунок 1 – Диаграмма пакета

Для более подробного описания системы, были разработаны соответствующие диаграммы вариантов использования.

### 1.2.2 Разработка диаграммы вариантов использования

Диаграммы вариантов использования описывают взаимоотношения и зависимости между группами вариантов использования и действующих лиц, участвующих, в процессе.

Диаграммы вариантов использования не предназначены для отображения проекта и не могут описывать внутреннее устройство системы, они предназначены для упрощения взаимодействия с будущими пользователями системы, с клиентами [3].

Для пакета «Магазин рыболовных принадлежностей» была разработана диаграмма вариантов использования «как есть», которая включает в себя: одного ACTOR (Сотрудник магазина) и 4 варианта использования (Учет поступления и наличия рыболовных принадлежностей, Учет продаж рыболовных принадлежностей, Учет видов продаваемых товаров, Формирование отчетов).

Диаграмма вариантов использования «как есть» представлена на рисунке 2.

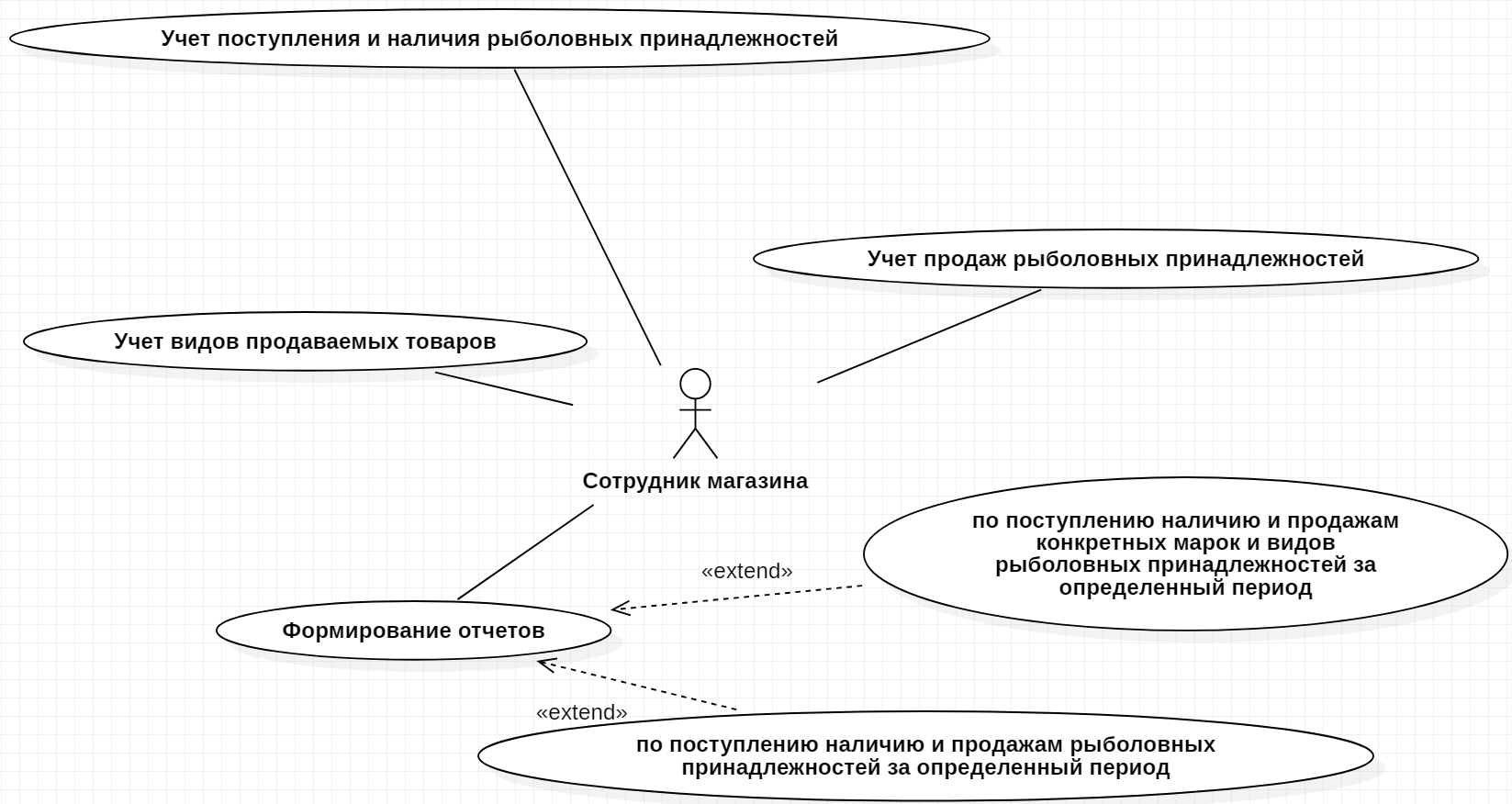


Рисунок 2 – Диаграмма вариантов использования «как есть»

### 1.2.3 Разработка диаграмм последовательности

Диаграмма последовательности (sequence diagram) — диаграмма, на которой для некоторого набора объектов на единой временной оси показан жизненный цикл какого-либо определённого объекта (создание-деятельность-уничтожение некой сущности) и взаимодействие актёров (действующих лиц) АС рамках какого-либо определённого прецедента (отправка запросов и получение ответов) [3].

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

На данной диаграмме объекты располагаются слева направо.

Диаграмма последовательности «как есть» представлена на рисунке 3.

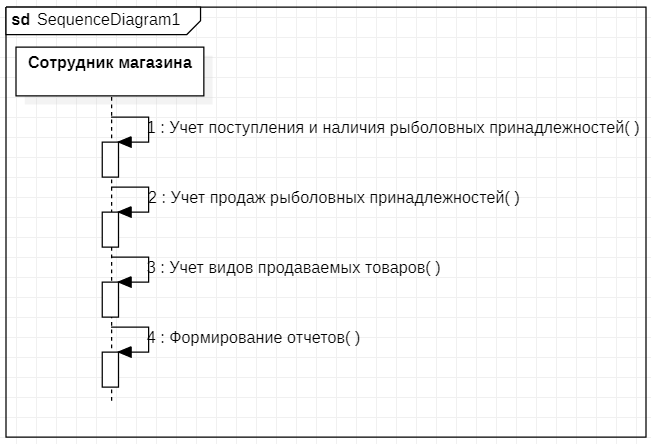


Рисунок 3 – Диаграмма последовательности «как есть»

### 1.2.4 Разработка диаграмм деятельности

Диаграммы деятельности – один из видов диаграмм, применяемых в UML, для моделирования динамических аспектов поведения системы. Диаграмма деятельности – это, по существу, блок – схема, которая показывает, как поток управления переходит от одной деятельности к другой [3].

Диаграммы деятельности являются представлением алгоритмов неких действий (активностей), выполняющихся в системе.

Диаграммы деятельности позволяют моделировать сложный жизненный цикл объекта, с переходами из одного состояния в другое. Они применимы для детализации некоторой конкретной операции. Диаграммы деятельности описывают переход от одной деятельности к другой, в отличие от диаграмм взаимодействия, где акцент делается на переходы потока управления от объекта к объекту.

Диаграммы деятельности «как есть» представлены на рисунках 4-7.

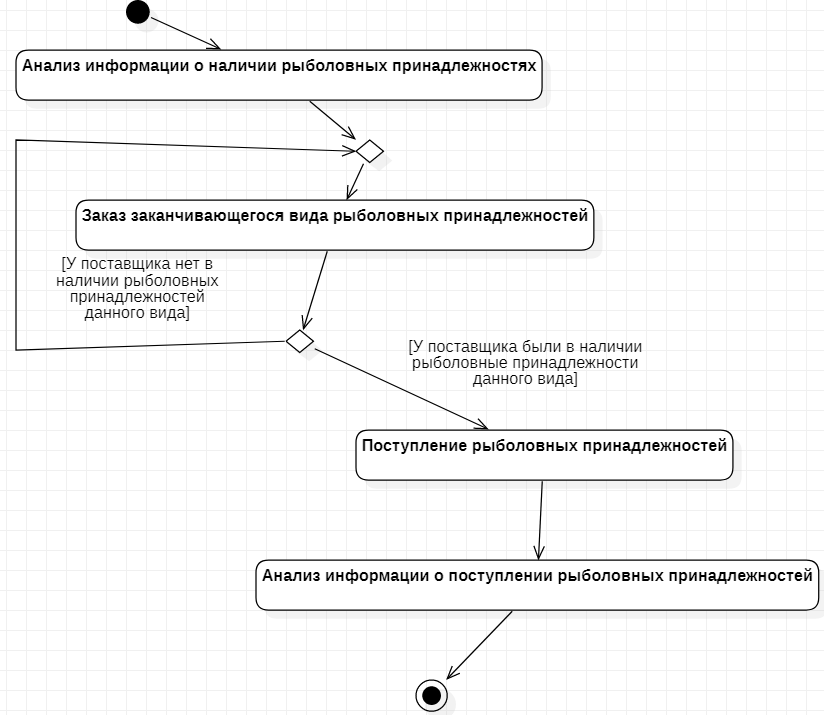


Рисунок 4 — Диаграмма деятельности «Учет поступления и наличия рыболовных принадлежностей»

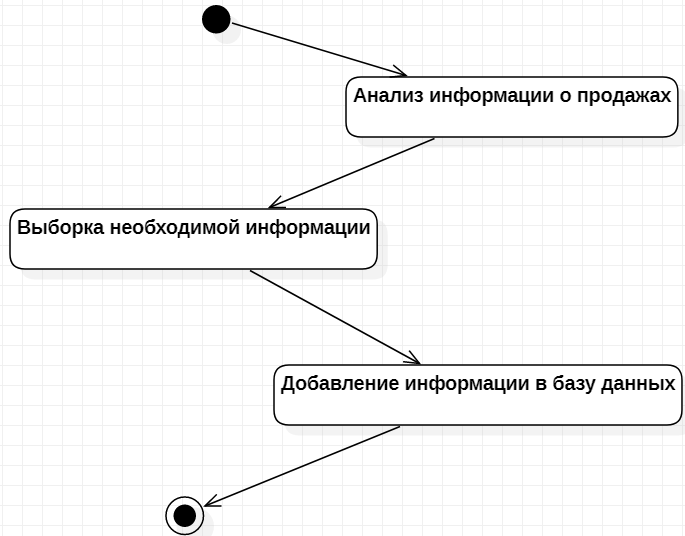


Рисунок 5 — Диаграмма деятельности «Учет продаж рыболовных принадлежностей»

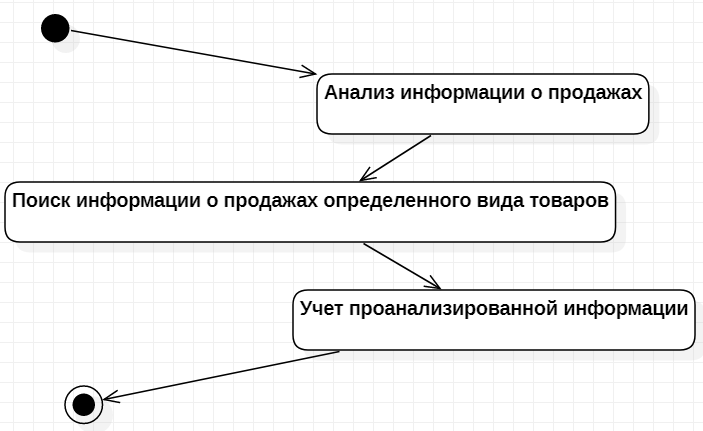


Рисунок 6 — Диаграмма деятельности «Учет видов продаваемых товаров»

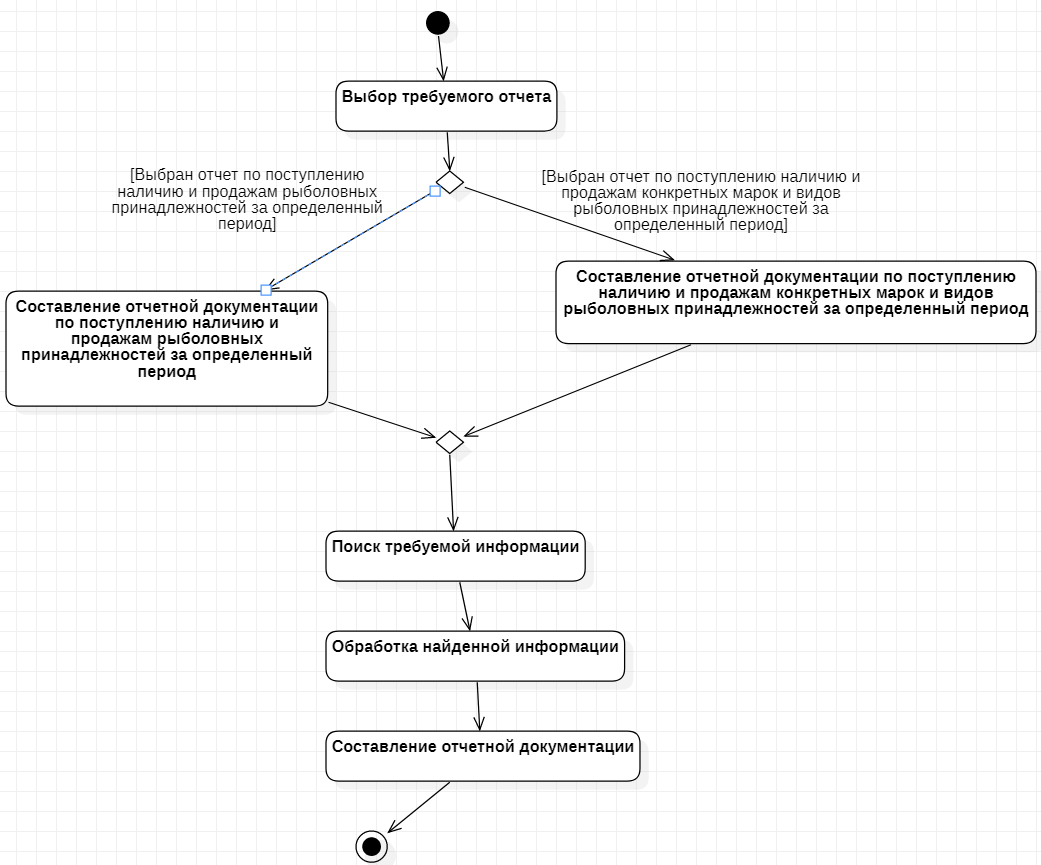


Рисунок 7 — Диаграмма деятельности «Формирование отчетов»

Таким образом была построена UML-модель «как есть» (AS – IS). Главным недостатком текущего мониторинга магазина рыболовных принадлежностей являются большие затраты людских ресурсов, обусловленные тем, что продавцу(дальнейшему пользователю АС), требуется проводить большое количество времени за формированием отчетной документации, тем самым отвлекая его от прямых обязанностей (обслуживание покупателя), чтобы снять с продавца данную задачу требуется образовать отдел по экономическому обеспечению производства, что является невыгодно для магазина рыболовных принадлежностей. Исходя из этого было принято, что целесообразно вести разработку новой АС.

В создаваемой новой АС было принято, добавить новый функционал реализующий: ведения списка поставщиков, ведение списка партнеров, ведения списка товаров. Так же для разгрузки продавца была сделана клиентская часть, предназначенная для покупателя, данная часть содержит в себе следующий функционал: добавление в корзину, учет предпочтения покупателя.

## 1.3 Построение модели предметной области «как должно быть» (TO -BE)

Найденные в модели AS-IS недостатки устраняются в модели ТО-ВЕ (как будет), т.е. модели новой организации процессов в системе магазин рыболовных принадлежностей. Создание и внедрение АС приводит к изменению условий выполнения отдельных операций, структуры процессов и магазина в целом. Это приводит к необходимости изменения системы правил, используемых в магазине, модификации должностных инструкций сотрудников.

Функциональная модель TO-BE позволяет уже на стадии проектирования будущей АС определить эти изменения. Применение функциональной модели TO-BE позволяет не только сократить сроки внедрения информационной системы, но также снизить риски, связанные с невосприимчивостью персонала к информационным технологиям. Модель ТО-ВЕ нужна для анализа альтернативных (лучших) путей выполнения функции и документирования того, как компания будет делать бизнес в будущем.

Функциональная модель TO-BE позволит четко определить распределение ресурсов между операциями делового процесса, что дает возможность оценить эффективность использования ресурсов после предлагаемого реинжиниринга.

В ходе анализа разработанной модели «как есть» был выявлен ряд проблем, показывающих что в системе есть определенные «дыры». Например, в системе не отлажен процесс составления расписания и хранение данных в одном месте. Чтобы исправить эти недостатки в систему необходимо добавить определенные компоненты для того чтобы избавиться от «дыр» внутри системы.

### 1.3.1 Разработка диаграммы пакетов

Разработанная диаграмма пакетов модели «как должно быть» соответствует диаграмме на рисунке 1.

### 1.3.2 Разработка диаграммы вариантов использования

В результате анализа выявленных недостатков было принято решение что для полноценного функционирования автоматизированной системы магазин рыболовных принадлежностей необходимо добавить на диаграмму вариантов использования двух ACTOR (Сотрудник магазина и покупатель) и 8 вариантов использования (Учет поступления и наличия рыболовных принадлежностей, Покупка, Формирование отчетов, Формирование договоров, Ведение списка поставщиков, Ведение списка партнеров магазина, Учет покупателей рыболовных принадлежностей, Добавление в корзину).

Разработанная диаграмма вариантов использования «как должно быть» представлена на рисунке 8.



Рисунок 8 – Диаграмма вариантов использования «как должно быть»

### 1.3.3 Разработка диаграммы последовательности

В соответствии с изменениями, внесенными в диаграмму вариантов использования были внесены изменения в диаграмму последовательности. Диаграмма последовательности была обобщена для удобства анализа последовательности действий, производимых в автоматизированной системе магазина рыболовных принадлежностей.

Разработанная диаграмма последовательности представлена на рисунке 9.

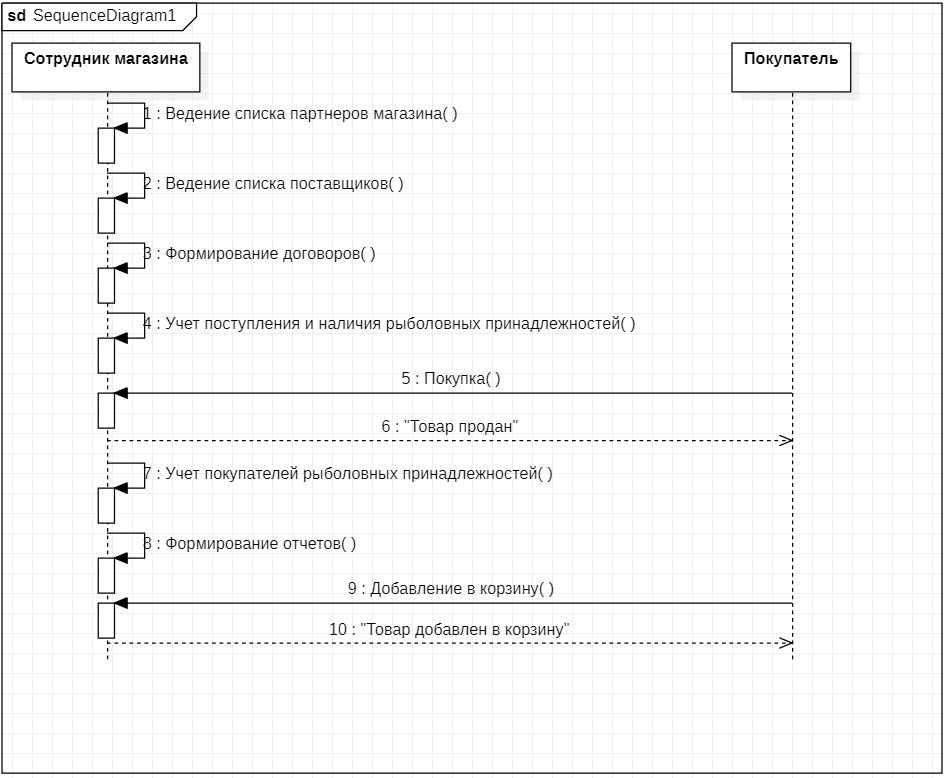


Рисунок 9 – Диаграмма последовательности «как должно быть»

### 1.3.4 Разработка диаграммы деятельности

В соответствии с изменениями, внесенными в диаграмму пакетов и диаграмму вариантов использования были внесены изменения в диаграмму деятельности. Диаграмма деятельности была обобщена для удобства анализа действий, производимых в автоматизированной системе магазина рыболовных принадлежностей.

Разработанные диаграммы деятельности «как должно быть» совпадают с диаграммами на рисунках 4-7. Новые диаграммы представлены на рисунках 10 - 14.

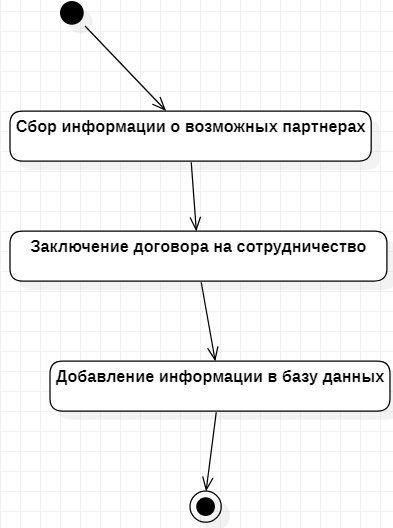


Рисунок 10 — Диаграмма деятельности «Учет сведений о потребностях покупателей в товарах»

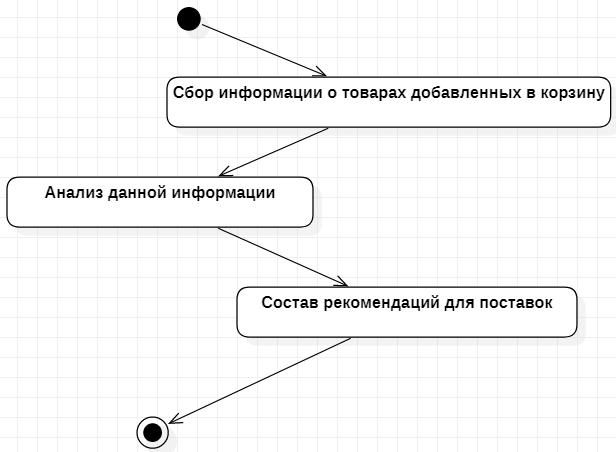


Рисунок 11 — Диаграмма деятельности «Учет информации о потребностях покупателей»

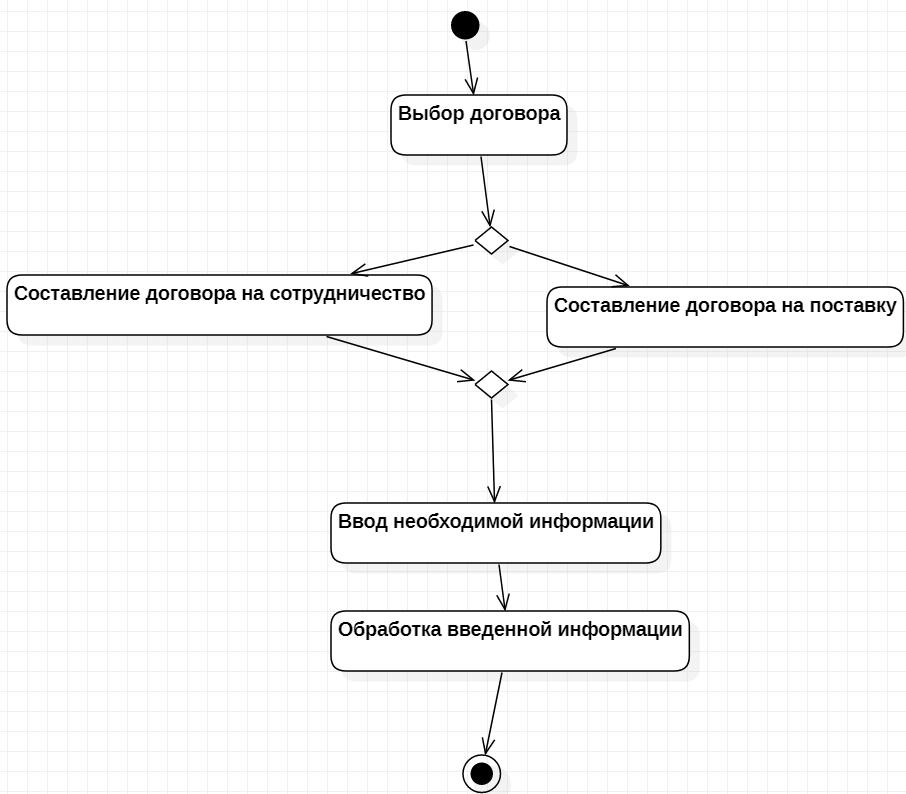


Рисунок 12 — Диаграмма деятельности «Формирование договоров»

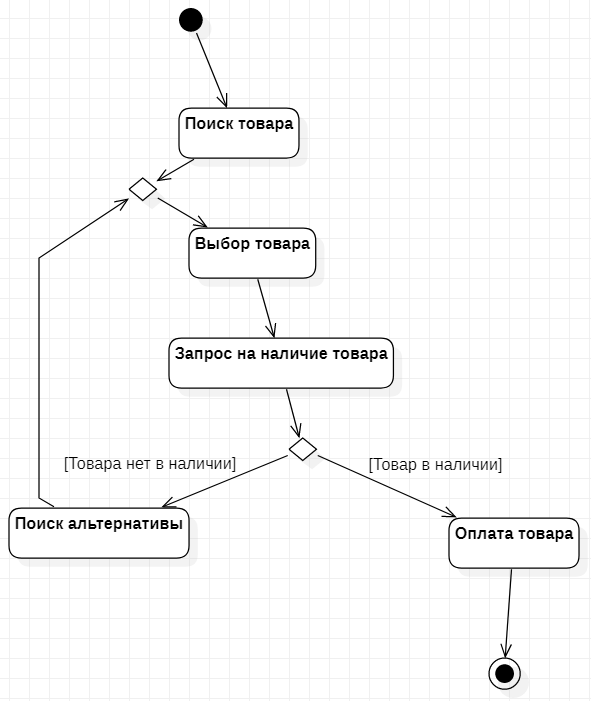


Рисунок 13 — Диаграмма деятельности «Покупка»

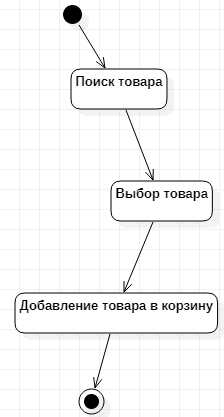


Рисунок 14 — Диаграмма деятельности «Добавление в корзину»

Таким образом была построена UML-модель «как должно быть» (TO – BE). Которая учла в себе все выявленные недостатки UML-модели «как есть». Исходя из UML-модели «как должно быть» продавцу больше не требуется проводить большое количество времени за формированием отчетной документации, с него была снята данная обязанность. Теперь АС самостоятельно формирует ту отчетную документацию, которая требуется. Нужная в отделе по экономическому обеспечению производства больше не является первостепенной

В построенной UML-модели «как должно быть» был введен, требуемый функционал реализующий: учет видов продаваемых товаров, учет поступления и наличия рыболовных принадлежностей, учет продаж рыболовных принадлежностей, ведение списка поставщиков, ведение списка партнеров, ведения списка товаров, а так же формирование отчетов по поступлению, наличию и продажам конкретных марок и видов рыболовных принадлежностей за определенный период, по поступлению, наличию и продажам рыболовных принадлежностей за определенный период. Так же для разгрузки продавца была сделана клиентская часть, предназначенная для покупателя, данная часть содержит в себе следующий функционал: добавление в корзину, учет предпочтения покупателя.

# 2 Техническое задание

## 2.1 Основание для разработки

Автоматизированная система должна быть разработана на основании задания на курсовое проектирование, выданного к.т.н. доцентом кафедры ИВС Дрождиным В.В. и утвержденного заведующей кафедры ИВС к.т.н., доцентом Бобрышевой Г.В.

## 2.2 Назначение разработки

Разрабатываемая автоматизированная система предназначена для ведения автоматизированной информационной системы магазина рыболовных принадлежностей.

### 2.3.1 Требования к функциональным характеристикам

Программа должна выполнять следующие функции:

* + учет видов продаваемых товаров;
  + учет поступления и наличия рыболовных принадлежностей;
  + учет продаж рыболовных принадлежностей;
  + формирование отчетов по поступлению, наличию и продажам конкретных марок и видов рыболовных принадлежностей за определенный период, по поступлению, наличию и продажам рыболовных принадлежностей за определенный период.

### 2.3.2 Требования к составу и параметрам технических средств

Программа должна быть предназначена для использования как на локальном компьютере, так и в сети. Локальный компьютер или рабочая станция сети должны иметь следующие технические характеристики:

* тактовая частота процессора - не менее 1,5 МГц;
* оперативная память – не менее 2048 Мбайт;
* на жестком диске при установке используется около 300 Мбайт;
* объем жестокого диска должен быть не менее 1024 Мбайт.

### 2.3.3 Требования к информационной и программной совместимости

Программа должна быть предназначена для использования на локальном компьютере. Разработанное приложение должно работать под управлением операционной системы Windows 7 и выше.

### 2.4 Требования к программной документации

Программная документация на систему должна содержать следующие документы:

* описание программы;
* программа и методика испытаний;
* руководство по применению, включающее руководство администратора системы и руководство пользователя;
* текст программы.

## 2.5 Стадии и этапы разработки

Стадии и этапы разработки проекта указаны в таблице 2.1.

Таблица 2.1 - Стадии выполнения проекта

|  |  |  |  |
| --- | --- | --- | --- |
| Стадии разработки | Этап работ | Срок | Стадии разработки |
| 1. Формирование требований к АС | Предпроектное  обследование |  | Кравчук М.В. |
| Разработка  модели  предметной  области |  | Кравчук М.В. |
| 2. Эскизное проектирование | Разработка  технического  задания |  | Кравчук М.В. |
| Технико-  экономическое  обоснование |  | Кравчук М.В. |
| 3. Техническое  проектирование | Разработка  модели системы |  | Кравчук М.В. |
| Разработка  алгоритма работы  системы |  |  |
| Разработка схемы  базы данных |  | Кравчук М.В. |
| 4. Создание системы | Написание и  отладка  программного  обеспечения |  | Кравчук М.В. |
| Создание базы  данных |  | Кравчук М.В. |
| Тестирование и отладка системы |  | Кравчук М.В. |
| Оформление  пояснительной  записки |  | Кравчук М.В. |

## 2.6 Порядок контроля и приемки

Контроль правильности работы программы осуществляется на основе выполнения тестовых примеров, обеспечивающих контроль функций системы.

Правильность работы программы определяется по результатам выполнения тестовых примеров.

Проверка правильности программы осуществляется путем проверки выполнения заложенных в нее функций. Для этого требуется проверить функцию учета видов продаваемых товаров, функцию учета поступления и наличия рыболовных принадлежностей, функцию учета продаж рыболовных принадлежностей, формирование отчетов:

* по поступлению, наличию и продажам конкретных марок и видов рыболовных принадлежностей за определенный период;
* по поступлению, наличию и продажам рыболовных принадлежностей за определенный период.

Приёмка работоспособной программы производится по результатам выполнения всех тестовых заданий и при наличии пояснительной записки, включающей полный комплект программной документации.

# 3 Проектирование автоматизированной системы

## 3.1 Технико-экономическое обоснование

### 3.1.1 Техническое обоснование

Использование АС позволяет добиться снижения численности управленческого аппарата, повысить эффективность и оперативность управления, освободить персонал от большого объема рутинной работы, создав условия для максимального использования его творческих способностей, в кратчайшие сроки обеспечить специалистов различных уровней необходимой информацией и решить многие иные проблемы.

АС в процессе своей работы должна выполнять следующие функции:

* сбор, обработка и анализ информации о состоянии объекта управления (посредством АС собирается информация о каждом товаре, марке и покупателе с целью проведения анализа полученной информации и подбора лучшего предложения для покупателя);
* передача управляющих воздействий на исполнение и контроль их передачи;
* обмен информацией с другими связанными с ней автоматизированными системами (например, показатели работы АС направляются в офис генерального директора компании с целью проведения анализа окупаемости и рентабельности магазина в определенной местности).

АС является двухуровневой, многопользовательской, рассчитанной примерно на 100 пользователей, имеющая доступ к базе данных через клиентское приложение, позволяющее заносить, удалять или изменять данные.

### 3.1.2 Экономическое обоснование

Расчет затрат на разработку АС можно выполнить по формуле:

Кп = Фз/п[(1+βд)(1+ βс)+ βн+βпр]+tЭВМСм–ч

где

Фз/п – фонд основной заработной платы разработчиков и других исполнителей работ, р.;

βд – коэффициент дополнительной зарплаты, можно принимать равным 0,1;

βс – коэффициент отчислений на социальные нужды от основной и дополнительной заработной платы, равен 0,3;

βн – коэффициент накладных расходов организации, разрабатывающей проект, можно принимать 0,7;

βпр – коэффициент прочих расходов, принимать 0,1;

tЭВМ – машинное время, затраченное для отладки модулей скрипта, ч.,

См–ч – стоимость машино-часа работы ЭВМ, р.

Укрупненный расчет фонда основной заработной платы исполнителей работ по разработке АС производится по формуле:

где

– суммарная трудоемкость работ по разработке АС, чел.-ч. (чел.-дн.), например, 32 чел.-дн. C – тарифная ставка часовая (дневная) разработчиков и других исполнителей работ, рассчитывается по формуле:

,

где ЗПМ – среднемесячная заработная плата разработчиков и других исполнителей работ, равна 27000р.;

Д – количество рабочих дней в месяце, равно 20 дней. Дневная тарифная ставка С численно равна:

С = 27000/20 = 1350 р.

Себестоимость машино-часа работы ПК определяется по формуле:

где Зп – затраты на заработную плату обслуживающего персонала с учетом всех отчислений, р.;

А – годовая сумма амортизации, р.;

Зэ – затраты на силовую электроэнергию, р.;

Зр – затраты на ремонт и обслуживание оборудования в год, р.;

3м – затраты на материалы в год, р.;

Зн – накладные расходы, р.;

Фд – действительный годовой фонд времени работы КСА, ч.

Так как нет необходимости в обслуживании ПК, то расчет затрат на заработную плату обслуживающего персонала проводить не надо (Зп=0).

Годовые амортизационные отчисления по КСА считаются по формуле:

где СКСА—стоимость ПК, используемого при отладке;

Стоимость ПК, необходимой для разработки АС составляет 40000 р.

На – норма амортизации, 25%. Таким образом, сумма годовой амортизации составит:

А=((40000)\*0,25)=10000 р.

Затраты на электроэнергию в год Зэ определяются следующим образом:

где Р — установленная мощность, кВт;

ЦЭ — стоимость силовой электроэнергии, р/кВт;

ФВ — время, в течение года, когда КСА потребляет электроэнергию, ч.

Затраты на текущие ремонты Зр и на материалы Зм в год берутся по данным бухгалтерии. При укрупненном расчете их сумма может быть принята от 4,5 до 10 % от стоимости ЭВМ.

Зр+Зм =40000 ∙0,1=4000 р.

В накладные расходы включаются затраты на оплату труда АУП, затраты на амортизацию и содержание площадей, затраты на отопление, освещение и прочие. Примем их равными 20% от стоимости ЭВМ.

Годовой фонд времени Фд устанавливается, исходя из номинального фонда времени и времени профилактики оборудования и ремонтов:

Фд =S∙h∙D – TПР

где S — продолжительность смены, ч.;

h — количество смен;

D — число рабочих дней в году, дн.;

TПР— время ремонтов и профилактики оборудования в год, ч.

Исходные данные для расчета затрат на разработку приведены в таблице 3.1.

Таблица 3.1 – Исходные данные для расчета затрат на разработку

|  |  |
| --- | --- |
| Параметр | Значение |
| Коэффициент дополнительной заработной платы - βд | 0,1 |
| Коэффициент отчислений на социальные нужды - βс | 0,3 |
| Коэффициент накладных расходов - βн | 0,7 |
| Коэффициент прочих расходов - βпр | 0,1 |
| Дневная ставка разработчика - С (р.) | 1350 |
| Суммарная трудоемкость -1 (чел-дней) | 20 |
| Продолжительность смены - S (ч) | 6 |
| Количество смен - h | 1 |
| Число рабочих дней в году - D | 247 |
| Время ремонта и профилактики оборудования в год - Тпр | 36 |
| Срок эксплуатации ПЭВМ – tэкспл (мес) | 48 |
| Стоимость ПЭВМ - Скса(р) | 40000 |
| Потребляемая мощность ПЭВМ - Wy (кВт~ч) | 0,3 |
| Стоимость электроэнергии Сэ (р/кВт-ч) | 2,05 |

Результаты расчета затрат на разработку АС приведены в таблице 3.2.

Таблица 3.2- Результаты расчета затрат на разработку АС

|  |  |
| --- | --- |
| Параметр | Значение |
| 1 | 2 |
| Действительный годовой фонд времени КСА - Фд (ч) | 1482 |
| Затраты на электроэнергию в год - Зэ (р) | 911.43 |
| Годовые амортизационные отчисления А (р) | 10000 |

Продолжение таблицы 3.2

|  |  |
| --- | --- |
| 1 | 2 |
| Накладные расходы - Зн (р/год) | 33345 |
| Затраты на ремонт - Зр (р/год) | 2200 |
| Затраты на материалы - Зм (р/год) | 1800 |
| Стоимость машино-часа работы ЭВМ См.ч (р) | 30.305 |
| Фонд основной заработной платы – Фз/п (р) | 333450 |
| Затраченное машинное время tэвм (ч) | 1482 |
| Суммарные затраты на разработку Кп (р) | 751156.43 |

Минимальная цена разработки Zmin складывается из полных затрат на разработку КП и минимально необходимой суммы прибыли nmin, размер которой позволял бы на минимальном уровне осуществить самофинансирование организации-разработчика после всех обязательных платежей и выплаты налогов.

Zmin = КП + Пmin.

Сумма прибыли Пmin рассчитывается исходя из планируемого минимального уровня рентабельности затрат организации-разработчика.

где Rmin — минимальный уровень рентабельности, 20%.

Таким образом, минимальная сумма прибыли:

Пmin = 751156.43\*0,20 =150231.29 р.

Следовательно,

Zmin= 751156.43+150231.29 = 901387.72 р.

Так как система разрабатывается под заказчика, то цена продажи будет договорной. Учитывая, что затраты на разработку будут эффективными, если доходы покроют их полностью, и будет получена минимально необходимая сумма прибыли, цену продажи с учётом НДС 20% установили равной 44149.6 р.

Единовременные затраты на внедрение системы К определяются по формуле:

К = Зпр + КК,

где ЗПР – затраты на приобретение модуля с учетом затрат на адаптацию и обучение персонала;

КК – величина капитальных затрат;

δ – коэффициент участия КСА.

Так как на подготовку к разработке данной системы не требуются дополнительные затраты, примем Зпр=0.

Коэффициент участия КСА используется только для данной регистратуры, поэтому коэффициент участия δ равен 1.

Комплекс средств автоматизации используется для одной регистратуры и коэффициент участия равен единице.

Величина капитальных затрат определяется по формуле:

Кк = Кктс+ Км + Кинв + Кзд + Кос + Ктр + Ксоп - Квыс,

где Кктс — сметная стоимость КТС, 15000 р.;

Км, Кинв, Кзд — затраты на установку, монтаж и запуск КТС в работу, на производственный инвентарь, на строительство и реконструкцию зданий для размещения КТС. Так как для установки КТС не требуется дополнительной реконструкции здания, примем Кзд = 0, а Км и Кинв равными 10% от стоимости КТС, т.е. 1500 р.;

Кос — сумма оборотных средств, 0 р.;

Ктр — транспортно-заготовительные расходы, примем равными 10% от стоимости КТС, т. е. 1500 р.;

Ксоп — сметная стоимость системы стандартного обеспечения применения КТС, примем равной 15% от стоимости КТС, т. е. 3750 р.;

Квыс ~ сумма высвобожденных средств в результате ввода в действие КСА, 0 р. Таким образом, единовременные затраты на внедрение АС составят:

Квн= 90138.77 + 15000 + 1500 + 1500 + 1500 + 3750 = 113388.77 р.

Для расчета текущих затрат на функционирование АС воспользуемся методом, который предполагает определение текущих затрат посредством основных составляющих:

где Зкса— годовые текущие затраты на эксплуатацию КСА, р. / год;

Ззп — годовые затраты на заработную плату специалистов в условиях функционирования модуля с начислениями, р. / год.

Внедрение АС подразумевает высвобождение специалиста, занимающегося обработкой данных в больнице и, соответственно, не требует затрат на заработную плату специалиста. Поэтому Ззп=0.

Затраты ЗКСА определяются по формуле:

где — годовые затраты на эксплуатацию КТС без учета заработной платы персонала, р. / год;

— годовые затраты на поддержание и актуализацию системы обеспечения применения КТС (хранение, обновление, контроль данных, программ и другие операции), р. / год;

– годовые затраты на содержание и ремонт производственных помещений, р. / год.

= 1920 \* 7,85 =15072 р.

=8000р.

=5000р.

= 15072 + 8000 = 23072 р.

Таким образом, годовые затраты на функционирование АС:

= 5000 + 23072 = 28072 р.

При внедрении АС высвобождается три специалиста, занимающихся обработкой данных в регистратуре. Годовая экономия на зарплате с учетом страховых взносов ЭЗ при ставке зарплаты специалиста 35 256 руб. составит:

ЭЗ=3 \* 35 256р.\*12 мес.\*1,3=1649980.8 р.

Чтобы получить чистую годовую экономию, необходимо вычесть затраты на функционирование АС:

ЭГ=ЭЗ-ЗТЕК=1649980.8 р. – 28072 р. = 1621908.8 р.

Для оценки эффективности затрат на создание (внедрение) АС можно применять следующие методы:

* метод расчета чистой дисконтированной стоимости;
* метод полного возмещения инвестиций.

Метод расчета чистой дисконтированной стоимости предполагает, что предприятие заранее задает минимально допустимую ставку процента, при которой капитальные вложения могут считаться эффективными. Такая ставка процента называется расчетной ставкой процента ("субъективная" ставка процента фирмы).

Базисом для установления расчетной ставки может быть ставка процента на заемный капитал, который предприятие должно выплачивать кредитору. Если процентная ставка не учитывает инфляцию, то ее называют номинальной ставкой процента in. Реальная ставка процента ir учитывает уровень инфляции In.

Реальная ставка процента ir рассчитывается по формуле Фишера:

ir = (in – In) / (1 + In),

где. ir, in, In заданы в десятичных дробях.

При низких процентных ставках и низкой инфляции реальная ставка процента может быть найдена по формуле:

ir = in - In

Чистая дисконтированная стоимость — это суммарный эффект за период функционирования инвестиций с учетом приведения всех результатов и затрат к начальному году (дисконтирование с помощью расчетной ставки процента).

Чистая дисконтированная стоимость (чистый дисконтированный доход или интегральный эффект) рассчитывается по формуле:

где

Т – период функционирования инвестиций, г.;

Kj – инвестиции b j-м году, т. р.;

– экономический результат (экономия, прибыль) b j-м году, т.р.;

δj – коэффициент дисконтирования для года j.

Коэффициент дисконтирования δj можно рассчитать по формуле:

Возьмем

in = 30%, Iinf=10%,

Тогда расчетная ставка процента будет равна:

ir = 100% • (0.30 – 0.10) / (1 + 0.10) = 18,2%

Расчет чистой дисконтированной стоимости приведен в таблице 3.3.

Таблица 3.3 – Расчет чистой дисконтированной стоимости

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Год | Инвестиц. вложения, р. | Экономия (дополн. прибыль), р. | Ряд платежей и поступлений, р. | Расчётная процентная ставка, % | | Чистая дисконт. стоимость, р. |
| Расчётная процентная ставка, % | Текущая дисконт. стоимость, р. |
| 0 | 118400 |  | -118400 | 1,0000 | -118400 | -118400 |
| 1 |  | 112000 | 112000 | 0,8460 | 94752 | -23648 |
| 2 |  | 112000 | 112000 | 0,7158 | 80169 | 56521 |
| 3 |  | 112000 | 112000 | 0,6055 | 67816 | 124337 |
| ∑ | 118400 | 336000 | 217600 |  | 124337 | x |

Затраты на разработку системы – 751156.43 р. С учетом того, что систему можно тиражировать, а число потенциальных потребителей 100, то затраты на одно лицо без НДС и дополнительных расходов равна 7511,56 р. без учета расходов на внедрение АС.

Стоимость аналогичных АС на рынке 79000 р. (за 10 копий). Следовательно, разработка АС целесообразна, потому что затраты на покупку АС на рынке на 100 лиц составляет 75115,6 без учета расходов на внедрение.

## 3.2 Разработка UML – модели автоматизированной системы

### 3.2.1 Разработка диаграммы пакетов АС

Разработанная диаграмма пакетов АС представлена на рисунке 15.



Рисунок 13 — Диаграмма пакетов

### 3.2.2 Разработка диаграммы вариантов использования АС

Разработанная диаграмма вариантов использования АС представлена на рисунке 16.

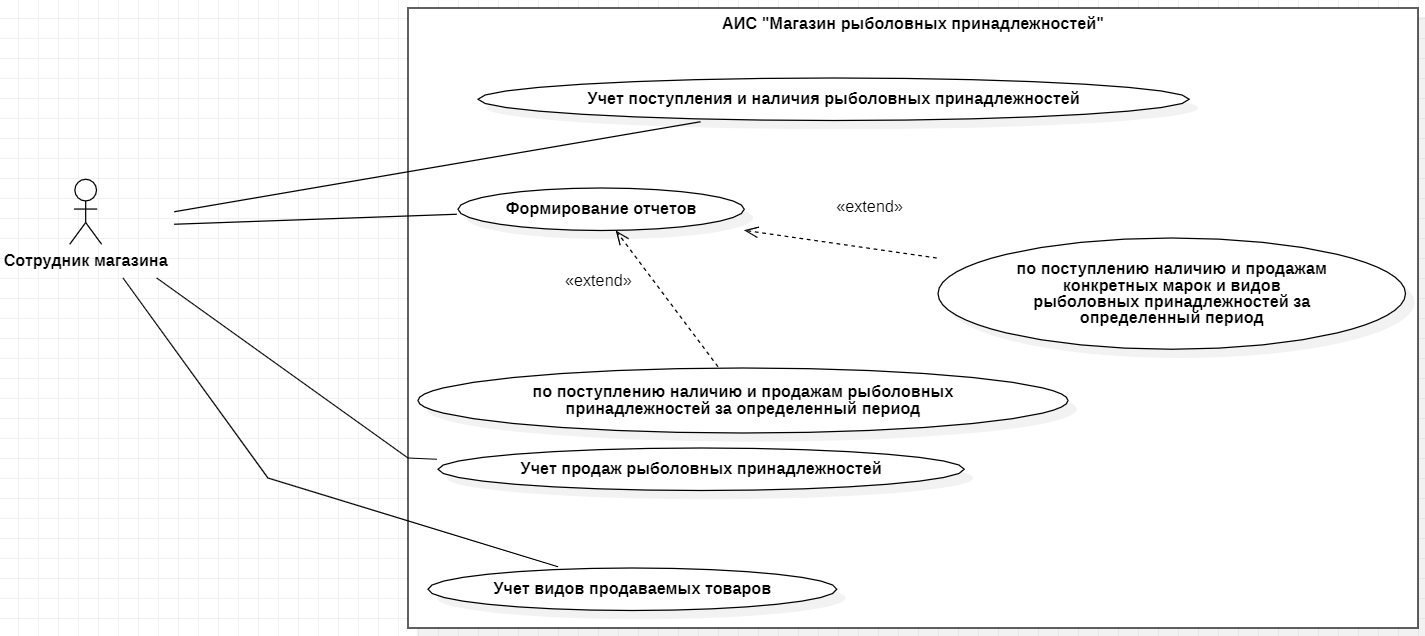


Рисунок 16 — Диаграмма вариантов использования АС

### 3.2.3 Разработка диаграммы последовательности АС

Разработанная диаграмма последовательности АС представлена на рисунке 17.

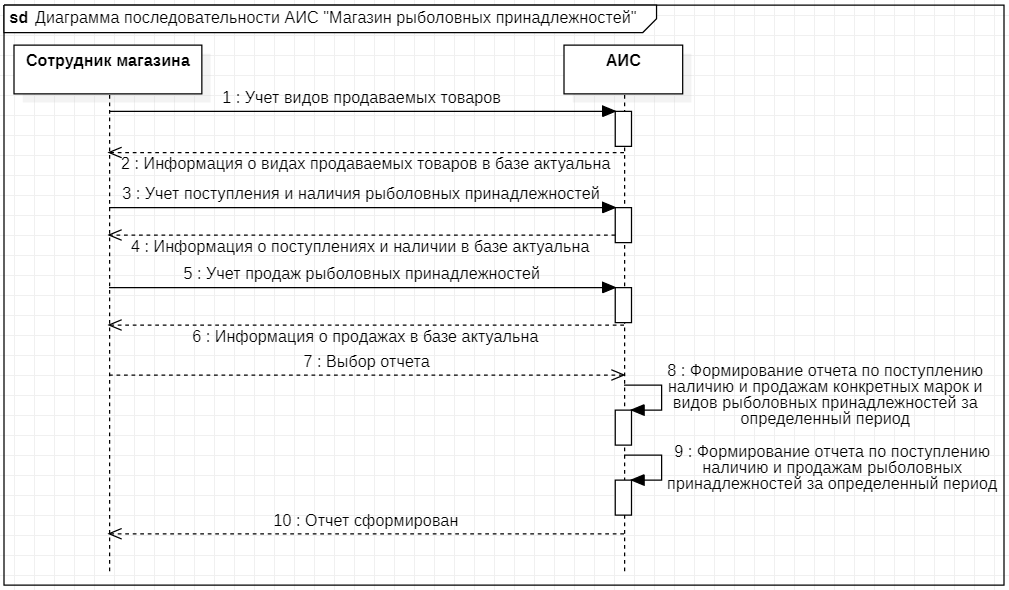


Рисунок 17 — Диаграмма последовательности АС

### 3.2.4 Разработка диаграммы деятельности АС

Диаграммы деятельности вариантов использования: «Учет поступления и наличия рыболовных принадлежностей», «Учет продаж рыболовных принадлежностей», «Учет продаж рыболовных принадлежностей» соответствуют диаграммам на рисунках 4, 5, 7.

### 3.2.5 Разработка диаграммы классов

Целью диаграммы классов, является определение необходимых объектов для нормального функционирования проектируемой системы, а также определение основных методов и функций.

В соответствии с техническим заданием была разработана диаграмма классов, представленная на рисунке 18.

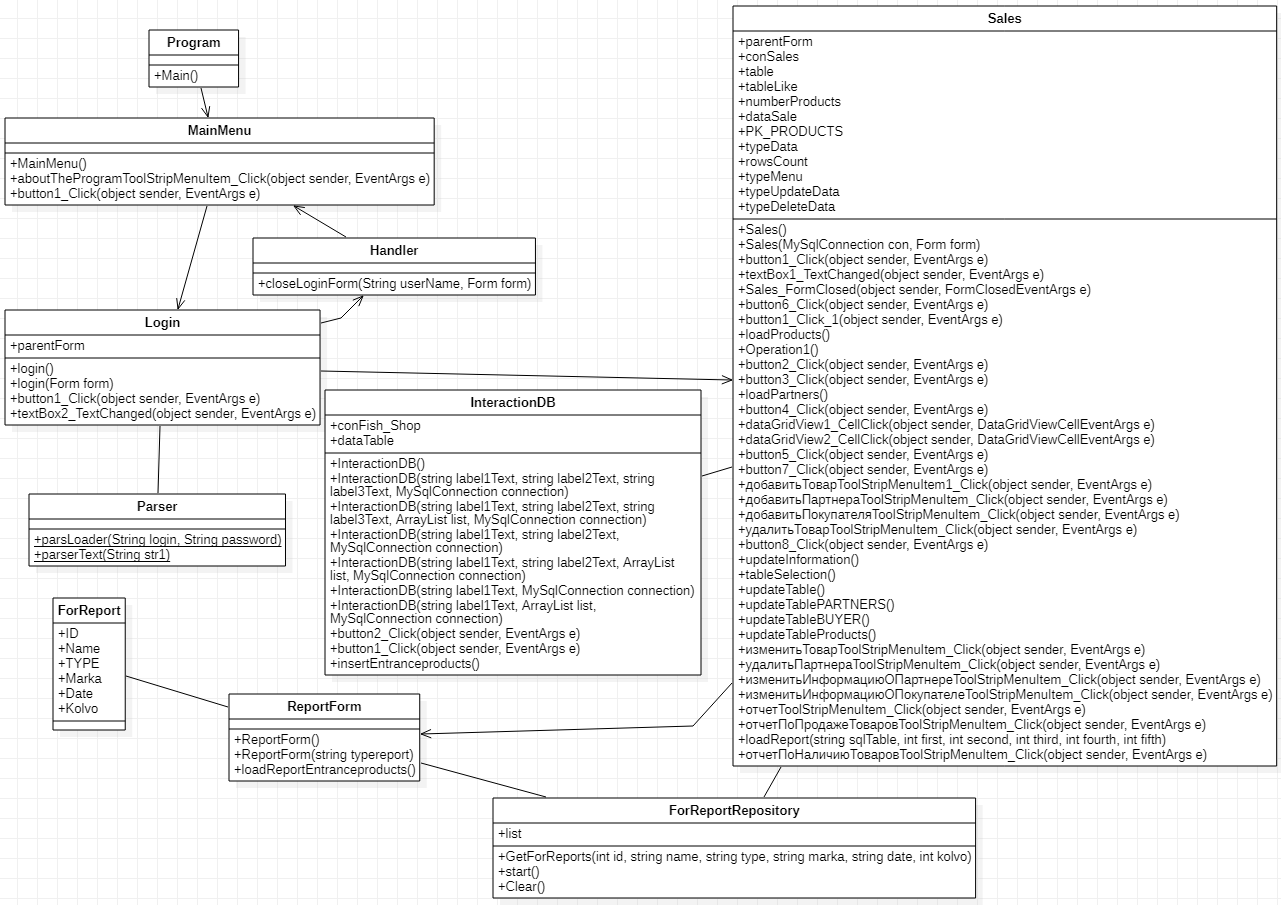


Рисунок 18 – Диаграмма классов

Описание классов приведено в таблице 3.4.

Таблица 3.4 — Описание классов автоматизированной информационной системы магазина рыболовных принадлежностей»

|  |  |
| --- | --- |
| Класс | Описание класса |
| Program | Класс основная точная входа в программу |
| MainMenu | Класс для отображения главного окна приложения |
| Login | Класс для отображения окна входа |
| Parser | Класс для работы с конфигурационными файлами |
| Handler | Класс для выполнения общих задач |
| Sales | Класс для отображения окна продавца |
| InteractionDB | Класс для отображения окна добавления или изменения информации о товарах, партнерах и пользователях |
| ReportForm | Класс для отображения окна вывода отчетов |
| ForReport | Класс «интерфейс» для наполнения коллекции |
| ForReportRepository | Класс для работы с коллекцией |

### 3.2.6 Разработка диаграммы развертывания

Диаграмма развертывания представляет узлы выполнения программных компонентов реального времени, а также процессов и объектов. Применяется для представления общей конфигурации и топологии распределенной программной системы, и содержит изображение размещения компонентов по отдельным узлам системы. Показывает наличие физических соединений, маршрутов передачи информации между аппаратными устройствами.

В соответствии с техническим заданием была разработана диаграмма развертывания, представленная на рисунке 19.

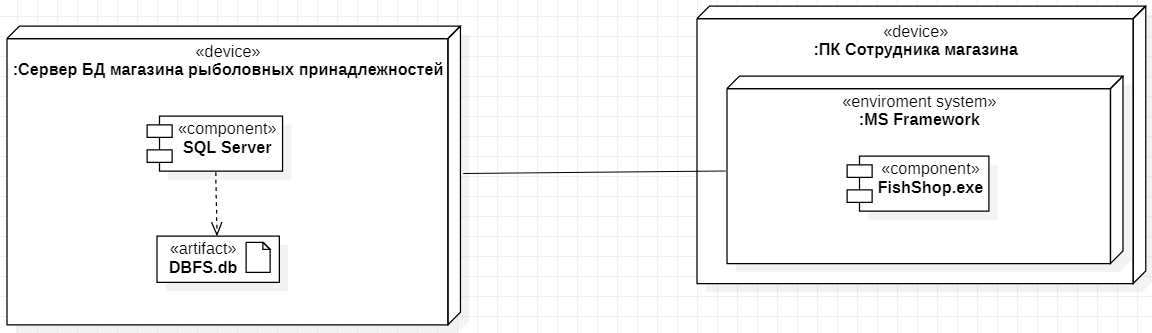


Рисунок 19 — Диаграмма развертывания АС

## 3.3 Проектирование базы данных

В соответствии с техническим заданием была разработана физическая модель БД, представленная на рисунке 20.

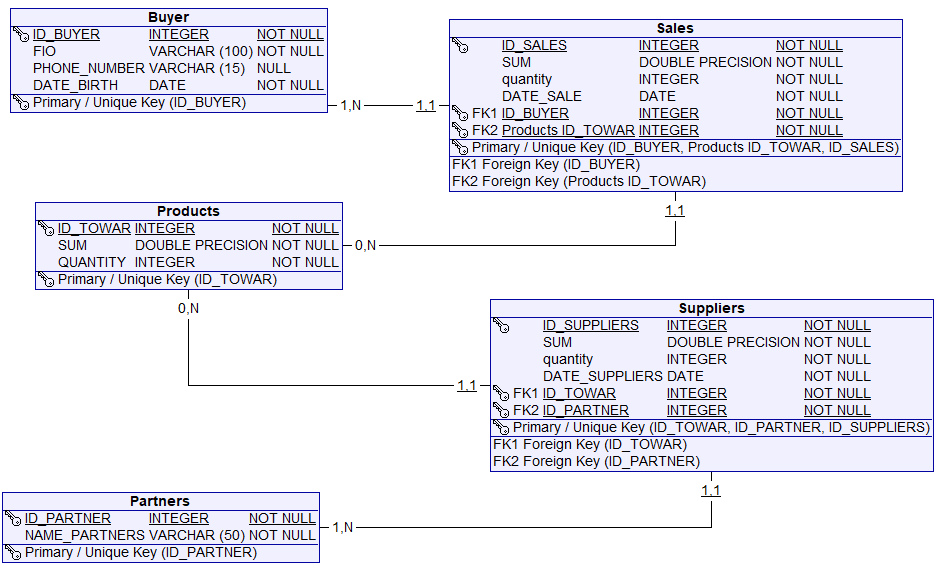


Рисунок 20 — Физическая модель БД

Описание сущностей приведено в таблицах 3.5 — 3.9.

Таблица 3.5 - Описание полей таблицы BUYER

|  |  |  |  |
| --- | --- | --- | --- |
| Имя атрибута | Нулевое значение | Ключевое поле | Тип данных |
| ID\_BUYER | Недопустимо | PK | INTEGER |
| FIO | Недопустимо | - | VARCHAR(100) |
| PHONE\_NUMBER | Допустимо | - | VARCHAR(15) |
| DATE\_BIRTH | Недопустимо | - | DATE |

Таблица 3.6 — Описание таблицы SALES

|  |  |  |  |
| --- | --- | --- | --- |
| Имя атрибута | Нулевое значение | Ключевое поле | Тип данных |
| ID\_SALES | Недопустимо | PK | INTEGER |
| SUM | Недопустимо | - | DOUBLE PRECISION |
| quantity | Недопустимо | - | INTEGER |
| DATE\_SALE | Недопустимо | - | DATE |
| ID\_BUYER | Недопустимо | FK | INTEGER |
| ID\_TOWAR | Недопустимо | FK | INTEGER |

Таблица 3.7 — Описание таблицы PRODUCTS

|  |  |  |  |
| --- | --- | --- | --- |
| Имя атрибута | Нулевое значение | Ключевое поле | Тип данных |
| ID\_TOWAR | Недопустимо | PK | INTEGER |
| SUM | Недопустимо | - | DOUBLE PRECISION |
| QUANTITY | Недопустимо | - | INTEGER |

Таблица 3.8 — Описание таблицы SUPPLIERS

|  |  |  |  |
| --- | --- | --- | --- |
| Имя атрибута | Нулевое значение | Ключевое поле | Тип данных |
| ID\_SUPPLIERS | Недопустимо | PK | INTEGER |
| SUM | Недопустимо | - | DOUBLE PRECISION |
| quantity | Недопустимо | - | INTEGER |
| DATE\_SUPPLIERS | Недопустимо | - | DATE |
| ID\_TOWAR | Недопустимо | FK | INTEGER |
| ID\_PARTNER | Недопустимо | FK | INTEGER |

Таблица 3.9 — Описание таблицы PARTNERS

|  |  |  |  |
| --- | --- | --- | --- |
| Имя атрибута | Нулевое значение | Ключевое поле | Тип данных |
| ID\_PARTNER | Недопустимо | PK | INTEGER |
| NAME\_PARTNERS | Недопустимо | - | VARCHAR(50) |

На основе физической модели данных сгенерирован файл «MAGAZIN.sql», содержащий скрипт базы данных на языке SQL.

# 4 Программа и методика испытаний

## 4.1 Объект испытаний

Объектом испытаний является приложение, предназначенное для мониторинга магазина рыболовных принадлежностей.

## 4.2 Цель испытаний

Испытания проводятся с целью проверки работоспособности и надежности программы. Для проверки правильности работы программы необходимо испытать её на тестовом примере. Сравнивая полученные результаты, можно определить правильность работы программы.

## 4.3 Требования к программе

Приложение должно выполнять все функции, указанные в техническом задании. Работа приложения не должна приводить к сбою. Общим требованием является создание дружественного интерфейса, с помощью которого пользователь мог бы легко и быстро найти нужную информацию и выполнить необходимые функции.

## 4.4 Средства и порядок испытаний

Во время испытания приложения необходимо использовать персональный компьютер с установленной операционной системой Windows 7 и выше. На компьютере должна быть установлена MS Visual Studio 2019.

Для проверки правильности работы системы был принят следующий порядок действий:

а) запустить программу на выполнение;

б) провести тестирование программы по тестам;

в) сравнить реакции программы с ожидаемым результатом;

г) сделать выводы по результатам тестирования о работоспособности программы.

## 4.5 Тестирование системы

Для проверки правильности функционирования АС магазина рыболовных принадлежностей, необходимо запустить программу на выполнение путем двойного клика мыши на файл «FishShop.exe» — на экране появится главная форма приложения (рисунок Б.1). После чего, требуется войти в свою учетную запись при помощи нажатия на главном окне на кнопку «Выполнить вход». После нажатия, появится форма для реализации входа в учетную запись (рисунок Б.2).

Затем, для проверки функции учета видов продаваемых товаров, необходимо выбрать кнопку «За покупками» расположенную на форме (рисунок Б.3). На экране появится таблица со всевозможными рыболовными принадлежностями (рисунок Б.4). Далее, для проверки функции учета видов продаваемых товаров необходимо выбрать из выпадающего списка, который находится на форме с таблицей со всевозможными рыболовными принадлежностями (рисунок Б.5), вид для реализации его учета (Рисунок Б.6).

Для проверки функции учета поступления и наличия рыболовных принадлежностей, необходимо в пункте меню «File», выбрать пункт меню «Просмотр поступлений» и нажать на него (рисунок Б.7). После чего, будет выведена форма учета поступления товаров в магазин (рисунок Б.8). Для учета наличия товаров требуется на форме «Страница продавца» нажать на кнопку «За покупками» (рисунок Б.3). На экране появится таблица со всевозможными рыболовными принадлежностями (рисунок Б.4).

Для проверки функции учета продаж рыболовных принадлежностей, необходимо в пункте меню «File», выбрать пункт меню «Учет продаж» и нажать на него (рисунок Б.7). После чего, будет выведена форма учета продаж товаров в магазине (рисунок Б.8).

Для проверки функций формирования отчетов:

* по поступлению, наличию и продажам конкретных марок и видов рыболовных принадлежностей за определенный период;
  + по поступлению, наличию и продажам рыболовных принадлежностей за определенный период.

Необходимо выбрать пункт меню «Report» и в нем выбрать соответствующий подпункт: «Отчет по поступлению товаров», «Отчет по наличию товаров», «Отчет по продаже товаров» (рисунок Б.9). После нажатия на выбранный подпункт будет выведено окно для отображения отчётов (рисунок Б.11). На данном окне требуется выбрать пункт меню «File -> Сортировать», а после чего выбрать требующийся метод сортировки: «Конкретные виды и марки по дате» или «По дате» (рисунок Б.10).

На экране появится отчет, выбранный пользователем (рисунки Б.11-Б.13).

# 5 Руководство по использованию

## 5.1 Руководство администратора системы

Для начала работы в АС магазина рыболовных принадлежностей требуется разархивировать данное программное приложение на ПК. После чего, необходимо заменить файл базы данных, находящийся в папке с программой, на уже имеющийся, в противном же случае, оставить все без изменений, и использовать файл БД, идущий вместе с программным приложением.

Запуск программы осуществляется путем открытия исполняющего файла FishShop.exe.

## 5.2 Руководство пользователя

Использование АС начинается с ее запуска путем открытия исполняющего файла FishShop.exe. После чего, требуется войти в свою учетную запись при помощи нажатия на главном окне на кнопку «Выполнить вход». После нажатия, появится форма для реализации входа в учетную запись (рисунок Б.2).

В процессе эксплуатации пользователь может использовать все заложенные в программу функции.

При необходимости у пользователя есть возможность добавления, удаления и модификации информации о товарах, покупателях и партнерах магазина рыболовных принадлежностей.

Для добавления нового товара пользователю требуется перейти в меню «File», выбрать подменю «Взаимодействие с товарами», после чего выбрать подменю «Добавить товар».

Для добавления нового партнера пользователю требуется перейти в меню «File», выбрать подменю «Взаимодействие с партнерами», после чего выбрать подменю «Добавить партнера».

Для добавления нового покупателя пользователю требуется перейти в меню «File», выбрать подменю «Взаимодействие с покупателями», после чего выбрать подменю «Добавить покупателя».

Для просмотра видов продаваемых товаров, необходимо выбрать кнопку «За покупками» расположенную на форме (рисунок Б.3). После чего, на экране появится таблица со всевозможными рыболовными принадлежностями (рисунок Б.4). Далее, по необходимости выбрать из выпадающего списка, который находится на форме с таблицей со всевозможными рыболовными принадлежностями (рисунок Б.5), вид для реализации его просмотра (Рисунок Б.6).

Для просмотра поступления и наличия рыболовных принадлежностей, необходимо в пункте меню «File», выбрать пункт меню «Просмотр поступлений» и нажать на него (рисунок Б.7). После чего, будет выведена форма для просмотра поступления товаров в магазин (рисунок Б.8). Для просмотра наличия товаров требуется на форме «Страница продавца» нажать на кнопку «За покупками» (рисунок Б.3). На экране появится таблица со всевозможными рыболовными принадлежностями (рисунок Б.4).

Для просмотра продаж рыболовных принадлежностей, необходимо в пункте меню «File», выбрать пункт меню «Учет продаж» и нажать на него (рисунок Б.7). После чего, будет выведена для просмотра продаж товаров в магазине (рисунок Б.8).

Для просмотра отчетов необходимо выбрать пункт меню «Report» и в нем выбрать соответствующий подпункт: «Отчет по поступлению товаров», «Отчет по наличию товаров», «Отчет по продаже товаров» (рисунок Б.9). После нажатия на выбранный подпункт будет выведено окно для отображения отчётов (рисунок Б.11).

# Заключение

В ходе выполнения курсового проекта была спроектирована и программно-реализована автоматизированная система магазина рыболовных принадлежностей. В качестве СУБД была взята реляционная СУБД MySQL. В качестве языка программирования был выбран язык Visual C#, в качестве среды разработки — Microsoft Visual Studio 2019. Полученные результаты показали, что программа удовлетворяет требованиям технического задания. Таким образом, можно сделать вывод, что задание выполнено в полном объеме.

# Список использованных источников

* 1. .Калянов. CASE и структурный системный анализ. М.: ЛОРИ, 2009. 262
  2. Карамазов А. "1С:Франчайзи - Объединение "Все для Главбуха"
  3. Моделирование UML. Диаграммы. - <http://book.uml3.ru/>
  4. Бузукова Е.А. Ассортимент магазина. Методы анализа и практические советы, Изд: Питер,2007
  5. ГОСТ 18617-83. ОХОТНИЧЬИ И РЫБОЛОВНЫЕ ПРИНАДЛЕЖНОСТИ

# ТЕКСТ ПРОГРАММЫ

# Приложение А

# (обязательное)

// file “Program.CS”

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace FishShop

{

static class Program

{

/// <summary>

/// Главная точка входа для приложения.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new MainMenu());

}

}

}

// file “MainMenu.cs”

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace FishShop

{

public partial class MainMenu : Form

{

public MainMenu()

{

InitializeComponent();

}

private void aboutTheProgramToolStripMenuItem\_Click(object sender, EventArgs e)

{

MessageBox.Show("This software is designed for the operation of a fishing accessories store. " +

"Thank you for using this software. " +

"For support questions, please follow the link: https://github.com/young-dead-body", "About the program",

MessageBoxButtons.OK, MessageBoxIcon.None);

}

private void button1\_Click(object sender, EventArgs e)

{

//ДЛЯ ПООДКЛЮЧЕНИЯ К БД

login login = new login(this);

login.ShowDialog();

}

}

}

// file “Login.cs”

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Xml;

using MySql.Data.MySqlClient;

namespace FishShop

{

public partial class login : Form

{

public login()

{

InitializeComponent();

}

Form parentForm;

public login(Form form)

{

InitializeComponent();

parentForm = form;

}

private void button1\_Click(object sender, EventArgs e)

{

if (textBox1.Text != "" && textBox2.Text != "")

{

String login = textBox1.Text;

String password = textBox2.Text;

bool start = parser.parsLoader(login,password);

if (start)

{

//=======================================

String connection = $"Database=fish\_shop;Data Source=localhost;User Id={login};Password={password}";

MySqlConnection con = new MySqlConnection(connection);

con.Open();

/\*String query = "UPDATE products SET SUM = 300 WHERE ID\_PRODUCT = 1";

MySqlCommand update\_table = new MySqlCommand(query, con);

update\_table.ExecuteNonQuery();\*/

//=======================================

Handler.closeLoginForm(login, parentForm);

//========================================

Sales sales = new Sales(con, parentForm);

sales.Show();

//========================================

parentForm.WindowState = FormWindowState.Minimized;

Close();

}

else

{

MessageBox.Show("Error login/password", "error");

}

}

}

private void textBox2\_TextChanged(object sender, EventArgs e)

{

if (Control.IsKeyLocked(Keys.CapsLock))

{

MessageBox.Show("The Caps Lock key is ON.");

}

}

}

}

// file “Parser.cs”

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Xml;

using System.Xml.Linq;

using System.Xml.XPath;

namespace FishShop

{

public class parser

{

public static bool parsLoader(String login, String password)

{

var filepath = "loader.xml";

string name, pwd; // Новые переменные имени и пароля

// Объявляем и забиваем файл в документ

XmlDocument xd = new XmlDocument();

FileStream fs = new FileStream(filepath, FileMode.Open);

xd.Load(fs);

XmlNodeList list = xd.GetElementsByTagName("user"); // Создаем и заполняем лист по тегу "user"

for (int i = 0; i < list.Count; i++)

{

// XmlElement id = (XmlElement)xd.GetElementsByTagName("user")[i]; // Забиваем id в переменную

XmlElement user = (XmlElement)xd.GetElementsByTagName("login")[i]; // Забиваем login в переменную

XmlElement pass = (XmlElement)xd.GetElementsByTagName("password")[i]; // Забиваем password в переменную

name = user.InnerText;

if (name == login)

{

pwd = pass.InnerText;

if (pwd == password)

{

fs.Close();

return true;

}

}

}

// Закрываем поток

fs.Close();

return false;

}

public static String parserText(String str1)

{

String str2 = "";

for (int i = 0; i < str1.Length; i++)

{

if (str1[i] == '\'')

{

str2 += "\\\"";

}

else

{

str2 += str1[i];

}

}

return str2;

}

}

}

// file “Handler.cs”

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace FishShop

{

class Handler

{

public static void closeLoginForm(String userName, Form form)

{

form.Text = $"Главное меню [{userName}]";

}

}

}

// file “Sales.cs”

using System;

using System.Collections;

using System.Collections.Generic;

using System.Data;

using System.Windows.Forms;

using MySql.Data.MySqlClient;

namespace FishShop

{

public partial class Sales : Form

{

public Sales()

{

InitializeComponent();

panel3.Visible = false;

dataGridView2.Enabled = false;

panel5.Visible = false;

}

Form parentForm;

MySqlConnection conSales;

public MySqlConnection ConSales { get => conSales; set => conSales = value; }

public Sales(MySqlConnection con, Form form)

{

InitializeComponent();

ConSales = con;

panel3.Visible = false;

parentForm = form;

dataGridView2.Enabled = false;

panel5.Visible = false;

comboBox2.Visible = false;

}

private void button1\_Click(object sender, EventArgs e)

{

panel1.Visible = true;

panel1.Dock = DockStyle.Fill;

}

private void Sales\_Load(object sender, EventArgs e)

{

// TODO: данная строка кода позволяет загрузить данные в таблицу "fish\_shopDataSet1.partners". При необходимости она может быть перемещена или удалена.

//this.partnersTableAdapter.Fill(this.fish\_shopDataSet1.partners);

// TODO: данная строка кода позволяет загрузить данные в таблицу "fish\_shopDataSet.products". При необходимости она может быть перемещена или удалена.

//this.productsTableAdapter.Fill(this.fish\_shopDataSet.products);

}

String table;

String tableLike;

private void textBox1\_TextChanged(object sender, EventArgs e)

{

if (table == "products")

{

MySqlDataAdapter data = new MySqlDataAdapter($"SELECT ID\_PRODUCT, NAME\_PRODUCT, QUANTITY, SUM, TYPE\_PRODUCT, MARKA\_PRODUCT FROM {table} WHERE {tableLike} LIKE '%{textBox1.Text}%'", conSales);

DataSet dstFish\_Shop = new DataSet("fish\_shop");

data.Fill(dstFish\_Shop, $"{table}");

DataTable dataTable;

dataTable = dstFish\_Shop.Tables[$"{table}"];

dataGridView1.DataSource = dataTable;

}

else

{

MySqlDataAdapter data = new MySqlDataAdapter($"SELECT \* FROM {table} WHERE {tableLike} LIKE '%{textBox1.Text}%'", conSales);

DataSet dstFish\_Shop = new DataSet("fish\_shop");

data.Fill(dstFish\_Shop, $"{table}");

DataTable dataTable;

dataTable = dstFish\_Shop.Tables[$"{table}"];

dataGridView1.DataSource = dataTable;

}

}

private void Sales\_FormClosed(object sender, FormClosedEventArgs e)

{

parentForm.Text = "Главное меню";

conSales.Close();

}

int numberProducts = 0;

ArrayList dataSale = new ArrayList();

ArrayList PK\_PRODUCTS = new ArrayList();

private void button6\_Click(object sender, EventArgs e)

{

panel4.Visible = false;

panel3.Visible = true;

int kolvo = 0;

if (comboBox1.SelectedIndex == -1) {

kolvo = 1;

}

else {

kolvo = comboBox1.SelectedIndex + 1;

}

dataGridView2.Rows.Add(new Object[] { dataGridView1.Rows[dataGridView1.CurrentRow.Index].Cells[1].Value,

kolvo});

numberProducts++;

dataSale.Add(dataGridView1.Rows[dataGridView1.CurrentRow.Index].Cells[1].Value);

dataSale.Add(dataGridView1.Rows[dataGridView1.CurrentRow.Index].Cells[4].Value);

dataSale.Add(dataGridView1.Rows[dataGridView1.CurrentRow.Index].Cells[5].Value);

PK\_PRODUCTS.Add(dataGridView1.Rows[dataGridView1.CurrentRow.Index].Cells[0].Value);

}

private void button1\_Click\_1(object sender, EventArgs e)

{

typeMenu = "Выбрать";

panel4.Visible = false;

panel1.Visible = false;

panel2.Visible = true;

panel2.Dock = DockStyle.Fill;

table = "PRODUCTS";

tableLike = "NAME\_PRODUCT";

comboBox2.Visible = true;

loadProducts();

}

private void loadProducts() {

MySqlDataAdapter data = new MySqlDataAdapter("SELECT ID\_PRODUCT, NAME\_PRODUCT, QUANTITY, SUM, TYPE\_PRODUCT, MARKA\_PRODUCT " +

$"FROM {table}", conSales);

DataSet dstFish\_Shop = new DataSet("fish\_shop");

data.Fill(dstFish\_Shop, $"{table}");

DataTable dataTable;

dataTable = dstFish\_Shop.Tables[$"{table}"];

dataGridView1.DataSource = dataTable;

//=================================================

dataGridView1.Columns[0].HeaderText = "НОМЕР";

dataGridView1.Columns[0].Width = 70;

dataGridView1.Columns[1].HeaderText = "Имя товара";

dataGridView1.Columns[1].Width = 80;

dataGridView1.Columns[2].HeaderText = "Количество";

dataGridView1.Columns[2].Width = 90;

dataGridView1.Columns[3].HeaderText = "Сумма";

dataGridView1.Columns[3].Width = 70;

dataGridView1.Columns[4].HeaderText = "Тип";

dataGridView1.Columns[4].Width = 90;

dataGridView1.Columns[5].HeaderText = "Марка";

dataGridView1.Columns[5].Width = 90;

//dataGridView1.Columns[6].HeaderText = "Дата";

dataGridView1.Width = 550;

Width = 740;

textBox1.Location = new System.Drawing.Point(565, 3);

textBox1.Width = 140;

button2.Location = new System.Drawing.Point(565, 315);

panel4.Location = new System.Drawing.Point(565, 28);

panel5.Location = new System.Drawing.Point(565, 28);

comboBox2.Location = new System.Drawing.Point(565, 250);

comboBox2.Width = 140;

}

private void button2\_Click(object sender, EventArgs e)

{

panel1.Visible = true;

panel2.Visible = false;

panel4.Visible = false;

panel5.Visible = false;

typeMenu = "";

Width = 600;

comboBox2.Visible = false;

}

private void button3\_Click(object sender, EventArgs e)

{

panel1.Visible = false;

panel2.Visible = true;

panel2.Dock = DockStyle.Fill;

table = "PARTNERS";

tableLike = "NAME\_PARTNERS";

loadPartners();

}

private void loadPartners()

{

MySqlDataAdapter data = new MySqlDataAdapter($"SELECT \* FROM {table}", conSales);

DataSet dstFish\_Shop = new DataSet("fish\_shop");

data.Fill(dstFish\_Shop, $"{table}");

DataTable dataTable;

dataTable = dstFish\_Shop.Tables[$"{table}"];

dataGridView1.DataSource = dataTable;

dataGridView1.Columns[0].HeaderText = "НОМЕР";

dataGridView1.Columns[1].HeaderText = "Имя партнера";

dataGridView1.Columns[1].Width = 200;

dataGridView1.Width = 400;

textBox1.Location = new System.Drawing.Point(420, 3);

panel4.Location = new System.Drawing.Point(420, 28);

button2.Location = new System.Drawing.Point(420, 315);

panel5.Location = new System.Drawing.Point(420, 28);

textBox1.Width = 140;

}

private void button4\_Click(object sender, EventArgs e)

{

panel1.Visible = false;

panel2.Visible = true;

panel2.Dock = DockStyle.Fill;

table = "buyer";

tableLike = "fio";

//ДОДЕЛАЙ

MySqlDataAdapter data = new MySqlDataAdapter($"SELECT \* FROM {table}", conSales);

DataSet dstFish\_Shop = new DataSet("fish\_shop");

data.Fill(dstFish\_Shop, $"{table}");

DataTable dataTable;

dataTable = dstFish\_Shop.Tables[$"{table}"];

dataGridView1.DataSource = dataTable;

dataGridView1.Columns[0].HeaderText = "НОМЕР";

dataGridView1.Columns[0].Width = 70;

dataGridView1.Columns[1].HeaderText = "ФИО";

dataGridView1.Columns[1].Width = 100;

dataGridView1.Columns[2].HeaderText = "Номер телефона";

dataGridView1.Columns[2].Width = 100;

dataGridView1.Columns[3].HeaderText = "Дата регистрации";

dataGridView1.Columns[3].Width = 90;

textBox1.Location = new System.Drawing.Point(420, 3);

panel4.Location = new System.Drawing.Point(420, 28);

button2.Location = new System.Drawing.Point(420, 315);

panel5.Location = new System.Drawing.Point(420, 28);

textBox1.Width = 140;

}

String typeData = "";

private void dataGridView1\_CellClick(object sender, DataGridViewCellEventArgs e)

{

dataGridView2.ColumnCount = 2;

dataGridView2.Columns[0].HeaderText = "Наименование товара";

dataGridView2.Columns[0].Width = 90;

dataGridView2.Columns[1].HeaderText = "Количество штук";

dataGridView2.Columns[1].Width = 90;

switch (typeMenu)

{

case "Удаление":

panel5.Visible = true;

label5.Text = $"Вы выбрали {typeData}";

label6.Text = dataGridView1.Rows[dataGridView1.CurrentRow.Index].Cells[1].Value.ToString();

break;

case "Выбрать":

comboBox1.Items.Clear();

panel4.Visible = true;

System.Text.StringBuilder sb = new System.Text.StringBuilder();

label2.Text = dataGridView1.Rows[dataGridView1.CurrentRow.Index].Cells[1].Value.ToString();

try

{

int quantity = Convert.ToInt32(dataGridView1.Rows[dataGridView1.CurrentRow.Index].Cells[2].Value.ToString());

for (int j = 0; j < quantity; j++)

{

comboBox1.Items.Add(j + 1);

}

}

catch

{

MessageBox.Show("Вы выбрали пустую строку! В следующий раз будьте отснорожнее.", "Упс...");

}

break;

case "Изменить":

panel5.Visible = true;

label5.Text = $"Вы выбрали {typeData}";

button8.Text = "Изменить";

label6.Text = dataGridView1.Rows[dataGridView1.CurrentRow.Index].Cells[1].Value.ToString();

break;

}

}

int rowsCount;

private void dataGridView2\_CellClick(object sender, DataGridViewCellEventArgs e)

{

rowsCount = dataGridView2.CurrentRow.Index;

}

private void button5\_Click(object sender, EventArgs e)

{

dataGridView2.Rows.RemoveAt(rowsCount);

}

private void button7\_Click(object sender, EventArgs e)

{

for (int k = 0; k < numberProducts; k++)

{

//String nameProducts = dataGridView2.CurrentRow.Cells[0].Value.ToString();

// String quantityProducts = dataGridView2.CurrentRow.Cells[1].Value.ToString();

String nameProducts = dataGridView2.Rows[k].Cells[0].Value.ToString();

String quantityProducts = dataGridView2.Rows[k].Cells[1].Value.ToString();

//==============================================

String querySQL = $"SELECT QUANTITY " +

$"FROM PRODUCTS " +

$"WHERE ID\_PRODUCT = {PK\_PRODUCTS[k].ToString()}";

MySqlCommand command = new MySqlCommand(querySQL, conSales);

int oldQuantity = Convert.ToInt32(command.ExecuteScalar().ToString());

int buyerQuantity = Convert.ToInt32(quantityProducts);

//==============================================

if (oldQuantity < buyerQuantity)

{

MessageBox.Show($"Извините, покупка товара {nameProducts} в количестве: {buyerQuantity} штук невозможна. " +

$"Так как на складу находится лишь: {oldQuantity} штук", "Ошибочка вышла...");

}

else

{

String queryUpdateQuantity = $"UPDATE PRODUCTS SET QUANTITY = {oldQuantity - buyerQuantity} " +

$"WHERE ID\_PRODUCT = {PK\_PRODUCTS[k].ToString()}";

MySqlCommand commandUpdate = new MySqlCommand(queryUpdateQuantity, conSales);

commandUpdate.ExecuteNonQuery();

if (oldQuantity == buyerQuantity) // если равны по удаляем

{

String queryDelete = $"DELETE FROM PRODUCTS WHERE (ID\_PRODUCT = {PK\_PRODUCTS[k].ToString()})";

commandUpdate = new MySqlCommand(queryDelete, conSales);

commandUpdate.ExecuteNonQuery();

}

//======================================

String kolvo = dataGridView2.Rows[k].Cells[1].Value.ToString();

String date = $"{DateTime.Now.Year}-{DateTime.Now.Month}-{DateTime.Now.Day}";

String name\_product = dataSale[k \* 3].ToString();

String type\_product = dataSale[k \* 3 + 1].ToString();

String marka\_product = dataSale[k \* 3 + 2].ToString();

String insertQuery = $"INSERT INTO SALEPRODUCTS (name\_product, type\_product, marka\_product, data\_sale, kolvo\_saleproduct) " +

$"VALUES ('{name\_product}', '{type\_product}', '{marka\_product}', '{date}', {kolvo});";

commandUpdate = new MySqlCommand(insertQuery, conSales);

commandUpdate.ExecuteNonQuery();

//======================================

}

}

updateTable();

//ВЕРНИСЬ

Width = 600;

dataGridView2.Rows.Clear();

panel3.Visible = false;

numberProducts = 0;

}

private void добавитьТоварToolStripMenuItem1\_Click(object sender, EventArgs e)

{

InteractionDB interactionDB = new InteractionDB("Наименование товара",

"Количество",

"Цена за штуку", conSales);

interactionDB.ShowDialog();

}

private void добавитьПартнераToolStripMenuItem\_Click(object sender, EventArgs e)

{

InteractionDB interactionDB = new InteractionDB("Наименование партнера", conSales);

interactionDB.ShowDialog();

}

private void добавитьПокупателяToolStripMenuItem\_Click(object sender, EventArgs e)

{

InteractionDB interactionDB = new InteractionDB("ФИО",

"Номер телефона",

conSales);

interactionDB.ShowDialog();

}

String typeMenu = "";

private void удалитьТоварToolStripMenuItem\_Click(object sender, EventArgs e)

{

typeData = "товар";

typeDeleteData = "данный товар";

button8.Text = "Удалить";

panel4.Visible = false;

panel1.Visible = false;

panel2.Visible = true;

panel2.Dock = DockStyle.Fill;

table = "PRODUCTS";

tableLike = "NAME\_PRODUCT";

loadProducts();

typeMenu = "Удаление";

//panel5.Visible = true;

}

String typeUpdateData = "";

String typeDeleteData = "";

private void button8\_Click(object sender, EventArgs e)

{

if (button8.Text == "Удалить")

{

var result = MessageBox.Show($"Вы уверены что хотите удалить {typeDeleteData}?", "Проверка на удаление",

MessageBoxButtons.YesNo, MessageBoxIcon.Question);

var id = dataGridView1.Rows[dataGridView1.CurrentRow.Index].Cells[0].Value.ToString();

String nameColums = tableSelection();

if (result == DialogResult.Yes)

{

String SQLScript = $"DELETE FROM {table} WHERE({nameColums} = {id})";

MySqlCommand commandDelete = new MySqlCommand(SQLScript, conSales);

commandDelete.ExecuteNonQuery();

updateTable();

panel5.Visible = false;

}

}

else if (button8.Text == "Изменить")

{

var result = MessageBox.Show($"Вы уверены что хотите изменить информацию о данном {typeUpdateData}?", "Проверка на изменение",

MessageBoxButtons.YesNo, MessageBoxIcon.Question);

var id = dataGridView1.Rows[dataGridView1.CurrentRow.Index].Cells[0].Value.ToString();

String nameColums = tableSelection();

//=======================ПЕРЕДЕЛАТЬ ПОД ИЗМЕНЕНИЕ

if (result == DialogResult.Yes)

{

updateInformation();

updateTable();

panel5.Visible = false;

}

//===============================================================

}

}

private void updateInformation()

{

ArrayList list = new ArrayList();

InteractionDB interactionDB;

//===========================

switch (table)

{

case "PRODUCTS":

list.Add(dataGridView1.Rows[dataGridView1.CurrentRow.Index].Cells[0].Value.ToString());

list.Add(dataGridView1.Rows[dataGridView1.CurrentRow.Index].Cells[1].Value.ToString());

list.Add(dataGridView1.Rows[dataGridView1.CurrentRow.Index].Cells[2].Value.ToString());

list.Add(dataGridView1.Rows[dataGridView1.CurrentRow.Index].Cells[3].Value.ToString());

list.Add(dataGridView1.Rows[dataGridView1.CurrentRow.Index].Cells[4].Value.ToString());

list.Add(dataGridView1.Rows[dataGridView1.CurrentRow.Index].Cells[5].Value.ToString());

interactionDB = new InteractionDB("Наименование товара",

"Количество",

"Цена за штуку",

list, conSales);

interactionDB.ShowDialog();

break;

case "BUYER":

list.Add(dataGridView1.Rows[dataGridView1.CurrentRow.Index].Cells[0].Value.ToString());

list.Add(dataGridView1.Rows[dataGridView1.CurrentRow.Index].Cells[1].Value.ToString());

list.Add(dataGridView1.Rows[dataGridView1.CurrentRow.Index].Cells[2].Value.ToString());

interactionDB = new InteractionDB("ФИО", "Номер телефона", list, conSales);

interactionDB.ShowDialog();

break;

case "PARTNERS":

list.Add(dataGridView1.Rows[dataGridView1.CurrentRow.Index].Cells[0].Value.ToString());

list.Add(dataGridView1.Rows[dataGridView1.CurrentRow.Index].Cells[1].Value.ToString());

interactionDB = new InteractionDB("Наименование партнера",

list, conSales);

interactionDB.ShowDialog();

break;

default:

MessageBox.Show("Непонятное завершение процесса. Обрабитесь к администратору", "ERROR");

//код, выполняемый если выражение не имеет ни одно из выше указанных значений

break;

}

}

private string tableSelection()

{

String nameColums = "";

switch (table)

{

case "PRODUCTS":

nameColums = "ID\_PRODUCT";

break;

case "BUYER":

nameColums = "ID\_BUYER";

break;

case "PARTNERS":

nameColums = "ID\_PARTNER";

break;

default:

MessageBox.Show("Непонятное завершение процесса. Обрабитесь к администратору", "ERROR");

//код, выполняемый если выражение не имеет ни одно из выше указанных значений

break;

}

return nameColums;

}

private void updateTable()

{

switch (table)

{

case "PRODUCTS":

updateTableProducts();

break;

case "BUYER":

updateTableBUYER();

break;

case "PARTNERS":

updateTablePARTNERS();

break;

default:

MessageBox.Show("Непонятное завершение процесса. Обрабитесь к администратору", "ERROR");

//код, выполняемый если выражение не имеет ни одно из выше указанных значений

break;

}

}

private void updateTablePARTNERS()

{

loadPartners();

}

private void updateTableBUYER()

{

MySqlDataAdapter data = new MySqlDataAdapter($"SELECT \* FROM {table}", conSales);

DataSet dstFish\_Shop = new DataSet("fish\_shop");

data.Fill(dstFish\_Shop, $"{table}");

DataTable dataTable;

dataTable = dstFish\_Shop.Tables[$"{table}"];

dataGridView1.DataSource = dataTable;

dataGridView1.Columns[0].HeaderText = "НОМЕР";

dataGridView1.Columns[0].Width = 70;

dataGridView1.Columns[1].HeaderText = "ФИО";

dataGridView1.Columns[1].Width = 100;

dataGridView1.Columns[2].HeaderText = "Номер телефона";

dataGridView1.Columns[2].Width = 100;

dataGridView1.Columns[3].HeaderText = "Дата регистрации";

dataGridView1.Columns[3].Width = 90;

textBox1.Location = new System.Drawing.Point(420, 3);

panel4.Location = new System.Drawing.Point(420, 28);

button2.Location = new System.Drawing.Point(420, 315);

panel5.Location = new System.Drawing.Point(420, 28);

textBox1.Width = 140;

}

private void updateTableProducts()

{

loadProducts();

}

private void изменитьТоварToolStripMenuItem\_Click(object sender, EventArgs e)

{

typeData = "товар";

typeUpdateData = "товаре";

typeMenu = "Изменить";

table = "PRODUCTS";

panel2.Visible = true;

loadProducts();

}

private void удалитьПартнераToolStripMenuItem\_Click(object sender, EventArgs e)

{

typeData = "партнера";

typeDeleteData = "данного партнера";

button8.Text = "Удалить";

panel4.Visible = false;

panel1.Visible = false;

panel2.Visible = true;

panel2.Dock = DockStyle.Fill;

table = "PARTNERS";

//======================================

loadPartners();

//=================================================

typeMenu = "Удаление";

}

private void изменитьИнформациюОПартнереToolStripMenuItem\_Click(object sender, EventArgs e)

{

typeData = "партнера";

typeUpdateData = "партнере";

typeMenu = "Изменить";

table = "PARTNERS";

panel2.Visible = true;

//======================================

loadPartners();

}

private void изменитьИнформациюОПокупателеToolStripMenuItem\_Click(object sender, EventArgs e)

{

typeData = "покупателя";

typeUpdateData = "покупателе";

typeMenu = "Изменить";

table = "BUYER";

panel2.Visible = true;

//======================================

MySqlDataAdapter data = new MySqlDataAdapter($"SELECT \* FROM {table}", conSales);

DataSet dstFish\_Shop = new DataSet("fish\_shop");

data.Fill(dstFish\_Shop, $"{table}");

DataTable dataTable;

dataTable = dstFish\_Shop.Tables[$"{table}"];

dataGridView1.DataSource = dataTable;

dataGridView1.Columns[0].HeaderText = "НОМЕР";

dataGridView1.Columns[0].Width = 70;

dataGridView1.Columns[1].HeaderText = "ФИО";

dataGridView1.Columns[1].Width = 100;

dataGridView1.Columns[2].HeaderText = "Номер телефона";

dataGridView1.Columns[2].Width = 100;

dataGridView1.Columns[3].HeaderText = "Дата регистрации";

dataGridView1.Columns[3].Width = 90;

textBox1.Location = new System.Drawing.Point(420, 3);

panel4.Location = new System.Drawing.Point(420, 28);

button2.Location = new System.Drawing.Point(420, 315);

panel5.Location = new System.Drawing.Point(420, 28);

textBox1.Width = 140;

}

private void отчетToolStripMenuItem\_Click(object sender, EventArgs e)

{

String sqlTable = "entranceproducts";

loadReport(sqlTable, 1, 2, 3, 4, 5);

ReportForm form1 = new ReportForm("Отчет о поступлении товаров", this);

form1.Show();

}

private void отчетПоПродажеТоваровToolStripMenuItem\_Click(object sender, EventArgs e)

{

String sqlTable = "saleproducts";

loadReport(sqlTable,1, 2, 3, 4, 5);

ReportForm form1 = new ReportForm("Отчет о продаже товаров", this);

form1.Show();

}

private void loadReport(string sqlTable, int first, int second, int third, int fourth, int fifth)

{

MySqlDataAdapter data = new MySqlDataAdapter($"SELECT \* FROM {sqlTable}", conSales);

DataSet dstFish\_Shop = new DataSet("fish\_shop");

data.Fill(dstFish\_Shop, $"{sqlTable}");

DataTable dataTable;

dataTable = dstFish\_Shop.Tables[$"{sqlTable}"];

dataGridView1.DataSource = dataTable;

//////////

String querySQL = $"SELECT COUNT(\*) FROM {sqlTable}";

MySqlCommand command = new MySqlCommand(querySQL, conSales);

int count = Convert.ToInt32(command.ExecuteScalar().ToString());

/////////

ForReportRepository.Clear();

for (int i = 0; i < count; i++)

{

int ID = Convert.ToInt32(dataGridView1.Rows[i].Cells[0].Value);

string Name = dataGridView1.Rows[i].Cells[first].Value.ToString();

string TYPE = dataGridView1.Rows[i].Cells[second].Value.ToString();

string Marka = dataGridView1.Rows[i].Cells[third].Value.ToString();

string Date = dataGridView1.Rows[i].Cells[fourth].Value.ToString();

int Kolvo = Convert.ToInt32(dataGridView1.Rows[i].Cells[fifth].Value);

ForReportRepository.GetForReports(ID, Name, TYPE, Marka, Date, Kolvo);

}

}

private void отчетПоНаличиюТоваровToolStripMenuItem\_Click(object sender, EventArgs e)

{

String sqlTable = "products";

loadReport(sqlTable,3,4,5,6,2);

ReportForm form1 = new ReportForm("Отчет о наличии товаров", this);

form1.Show();

}

private void comboBox2\_SelectedIndexChanged(object sender, EventArgs e)

{

MySqlDataAdapter data = new MySqlDataAdapter($"SELECT ID\_PRODUCT, NAME\_PRODUCT, QUANTITY, SUM, TYPE\_PRODUCT, MARKA\_PRODUCT FROM {table} WHERE TYPE\_PRODUCT LIKE '%{comboBox2.Text}%'", conSales);

DataSet dstFish\_Shop = new DataSet("fish\_shop");

data.Fill(dstFish\_Shop, $"{table}");

DataTable dataTable;

dataTable = dstFish\_Shop.Tables[$"{table}"];

dataGridView1.DataSource = dataTable;

}

private void просмотрПоступленийToolStripMenuItem\_Click(object sender, EventArgs e)

{

textBox1.Visible = false;

panel2.Visible = true;

panel1.Visible = false;

panel3.Visible = false;

panel4.Visible = false;

panel5.Visible = false;

dataGridView1.Width = 670;

Width = 800;

button2.Location = new System.Drawing.Point(700, 310);

MySqlDataAdapter data = new MySqlDataAdapter($"SELECT \* FROM entranceproducts", conSales);

DataSet dstFish\_Shop = new DataSet("fish\_shop");

data.Fill(dstFish\_Shop, $"entranceproducts");

DataTable dataTable;

dataTable = dstFish\_Shop.Tables[$"entranceproducts"];

dataGridView1.DataSource = dataTable;

dataGridView1.Columns[0].HeaderText = "Номер";

dataGridView1.Columns[0].Width = 70;

dataGridView1.Columns[1].HeaderText = "Наименование";

dataGridView1.Columns[1].Width = 100;

dataGridView1.Columns[2].HeaderText = "Вид";

dataGridView1.Columns[2].Width = 100;

dataGridView1.Columns[3].HeaderText = "Марка";

dataGridView1.Columns[3].Width = 90;

dataGridView1.Columns[4].HeaderText = "Дата поступления";

dataGridView1.Columns[4].Width = 90;

dataGridView1.Columns[5].HeaderText = "Количество";

dataGridView1.Columns[5].Width = 90;

}

private void учетПродажToolStripMenuItem\_Click(object sender, EventArgs e)

{

textBox1.Visible = false;

panel2.Visible = true;

panel1.Visible = false;

panel3.Visible = false;

panel4.Visible = false;

panel5.Visible = false;

dataGridView1.Width = 670;

Width = 800;

button2.Location = new System.Drawing.Point(700, 310);

MySqlDataAdapter data = new MySqlDataAdapter($"SELECT \* FROM saleproducts", conSales);

DataSet dstFish\_Shop = new DataSet("fish\_shop");

data.Fill(dstFish\_Shop, $"saleproducts");

DataTable dataTable;

dataTable = dstFish\_Shop.Tables[$"saleproducts"];

dataGridView1.DataSource = dataTable;

dataGridView1.Columns[0].HeaderText = "Номер";

dataGridView1.Columns[0].Width = 70;

dataGridView1.Columns[1].HeaderText = "Наименование";

dataGridView1.Columns[1].Width = 100;

dataGridView1.Columns[2].HeaderText = "Вид";

dataGridView1.Columns[2].Width = 100;

dataGridView1.Columns[3].HeaderText = "Марка";

dataGridView1.Columns[3].Width = 90;

dataGridView1.Columns[4].HeaderText = "Дата продажи";

dataGridView1.Columns[4].Width = 90;

dataGridView1.Columns[5].HeaderText = "Количество";

dataGridView1.Columns[5].Width = 90;

}

public void reportParser1(string sqlTable, string vid, string marka, string ondate, string offdate)

{

MySqlDataAdapter data = new MySqlDataAdapter($"SELECT \* FROM {sqlTable} " +

$"WHERE (TYPE\_PRODUCT LIKE '%{vid}%') AND (MARKA\_PRODUCT LIKE '%{marka}%')" +

$"AND (data\_entrance BETWEEN '{ondate}' AND '{offdate}')", conSales);

DataSet dstFish\_Shop = new DataSet("fish\_shop");

data.Fill(dstFish\_Shop, $"{sqlTable}");

DataTable dataTable;

dataTable = dstFish\_Shop.Tables[$"{sqlTable}"];

dataGridView1.DataSource = dataTable;

//////////

String querySQL = $"SELECT COUNT(\*) FROM {sqlTable} " +

$"WHERE (TYPE\_PRODUCT LIKE '%{vid}%') AND (MARKA\_PRODUCT LIKE '%{marka}%')" +

$"AND (data\_entrance BETWEEN '{ondate}' AND '{offdate}')";

MySqlCommand command = new MySqlCommand(querySQL, conSales);

int count = Convert.ToInt32(command.ExecuteScalar().ToString());

/////////

ForReportRepository.Clear();

for (int i = 0; i < count; i++)

{

int ID = Convert.ToInt32(dataGridView1.Rows[i].Cells[0].Value);

string Name = dataGridView1.Rows[i].Cells[1].Value.ToString();

string TYPE = dataGridView1.Rows[i].Cells[2].Value.ToString();

string Marka = dataGridView1.Rows[i].Cells[3].Value.ToString();

string Date = dataGridView1.Rows[i].Cells[4].Value.ToString();

int Kolvo = Convert.ToInt32(dataGridView1.Rows[i].Cells[5].Value);

ForReportRepository.GetForReports(ID, Name, TYPE, Marka, Date, Kolvo);

}

}

public void reportParser2(string sqlTable, string vid, string marka, string ondate, string offdate)

{

MySqlDataAdapter data = new MySqlDataAdapter($"SELECT \* FROM {sqlTable} " +

$"WHERE (TYPE\_PRODUCT LIKE '%{vid}%') AND (MARKA\_PRODUCT LIKE '%{marka}%')" +

$"AND (data\_sale BETWEEN '{ondate}' AND '{offdate}')", conSales);

DataSet dstFish\_Shop = new DataSet("fish\_shop");

data.Fill(dstFish\_Shop, $"{sqlTable}");

DataTable dataTable;

dataTable = dstFish\_Shop.Tables[$"{sqlTable}"];

dataGridView1.DataSource = dataTable;

//////////

String querySQL = $"SELECT COUNT(\*) FROM {sqlTable} " +

$"WHERE (TYPE\_PRODUCT LIKE '%{vid}%') AND (MARKA\_PRODUCT LIKE '%{marka}%')" +

$"AND (data\_sale BETWEEN '{ondate}' AND '{offdate}')";

MySqlCommand command = new MySqlCommand(querySQL, conSales);

int count = Convert.ToInt32(command.ExecuteScalar().ToString());

/////////

ForReportRepository.Clear();

for (int i = 0; i < count; i++)

{

int ID = Convert.ToInt32(dataGridView1.Rows[i].Cells[0].Value);

string Name = dataGridView1.Rows[i].Cells[1].Value.ToString();

string TYPE = dataGridView1.Rows[i].Cells[2].Value.ToString();

string Marka = dataGridView1.Rows[i].Cells[3].Value.ToString();

string Date = dataGridView1.Rows[i].Cells[4].Value.ToString();

int Kolvo = Convert.ToInt32(dataGridView1.Rows[i].Cells[5].Value);

ForReportRepository.GetForReports(ID, Name, TYPE, Marka, Date, Kolvo);

}

}

public void reportParser3(string sqlTable, string vid, string marka, string ondate, string offdate)

{

MySqlDataAdapter data = new MySqlDataAdapter($"SELECT \* FROM {sqlTable} " +

$"WHERE (TYPE\_PRODUCT LIKE '%{vid}%') AND (MARKA\_PRODUCT LIKE '%{marka}%')" +

$"AND (DATE\_PRODUCT BETWEEN '{ondate}' AND '{offdate}')", conSales);

DataSet dstFish\_Shop = new DataSet("fish\_shop");

data.Fill(dstFish\_Shop, $"{sqlTable}");

DataTable dataTable;

dataTable = dstFish\_Shop.Tables[$"{sqlTable}"];

dataGridView1.DataSource = dataTable;

//////////3,4,5,6,2

String querySQL = $"SELECT COUNT(\*) FROM {sqlTable} " +

$"WHERE (TYPE\_PRODUCT LIKE '%{vid}%') AND (MARKA\_PRODUCT LIKE '%{marka}%')" +

$"AND (DATE\_PRODUCT BETWEEN '{ondate}' AND '{offdate}')";

MySqlCommand command = new MySqlCommand(querySQL, conSales);

int count = Convert.ToInt32(command.ExecuteScalar().ToString());

/////////

ForReportRepository.Clear();

for (int i = 0; i < count; i++)

{

int ID = Convert.ToInt32(dataGridView1.Rows[i].Cells[0].Value);

string Name = dataGridView1.Rows[i].Cells[3].Value.ToString();

string TYPE = dataGridView1.Rows[i].Cells[4].Value.ToString();

string Marka = dataGridView1.Rows[i].Cells[5].Value.ToString();

string Date = dataGridView1.Rows[i].Cells[6].Value.ToString();

int Kolvo = Convert.ToInt32(dataGridView1.Rows[i].Cells[2].Value);

ForReportRepository.GetForReports(ID, Name, TYPE, Marka, Date, Kolvo);

}

}

}

}

// file “InteractionDB.cs”

using System;

using System.Collections;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using MySql.Data.MySqlClient;

namespace FishShop

{

public partial class InteractionDB : Form

{

public InteractionDB()

{

InitializeComponent();

}

MySqlConnection conFish\_Shop;

public InteractionDB(string label1Text, string label2Text, string label3Text, MySqlConnection connection)

{

InitializeComponent();

Text = "Добавить товар";

label1.Text = label1Text;

label2.Text = label2Text;

label3.Text = label3Text;

dateTimePicker1.Visible = false;

button1.Text = "Добавить";

button2.Text = "Отменить";

conFish\_Shop = connection;

label4.Text = "Тип товара";

label5.Text = "Марка товара";

}

ArrayList dataTable;

public InteractionDB(string label1Text, string label2Text, string label3Text,

ArrayList list, MySqlConnection connection)

{

InitializeComponent();

Text = "Изменить товар";

label1.Text = label1Text;

label2.Text = label2Text;

label3.Text = label3Text;

textBox1.Text = list[1].ToString();

textBox2.Text = list[2].ToString();

textBox3.Text = list[3].ToString();

comboBox1.Text = list[4].ToString();

comboBox2.Text = list[5].ToString();

dateTimePicker1.Visible = false;

button1.Text = "Изменить";

button2.Text = "Отменить";

conFish\_Shop = connection;

dataTable = list;

label4.Text = "Тип товара";

label5.Text = "Марка товара";

}

public InteractionDB(string label1Text, string label2Text, MySqlConnection connection)

{

InitializeComponent();

Text = "Добавить покупателя";

label1.Text = label1Text;

label2.Text = label2Text;

label3.Visible = false;

textBox3.Visible = false;

dateTimePicker1.Visible = false;

button1.Text = "Добавить";

button2.Text = "Отменить";

conFish\_Shop = connection;

label4.Visible = false;

label5.Visible = false;

comboBox1.Visible = false;

comboBox2.Visible = false;

}

public InteractionDB(string label1Text, string label2Text, ArrayList list, MySqlConnection connection)

{

InitializeComponent();

Text = "Изменить покупателя";

label1.Text = label1Text;

textBox1.Text = list[1].ToString();

label2.Text = label2Text;

textBox2.Text = list[2].ToString();

label3.Visible = false;

textBox3.Visible = false;

dateTimePicker1.Visible = false;

button1.Text = "Изменить";

button2.Text = "Отменить";

conFish\_Shop = connection;

dataTable = list;

label4.Visible = false;

label5.Visible = false;

comboBox1.Visible = false;

comboBox2.Visible = false;

}

public InteractionDB(string label1Text, MySqlConnection connection)

{

InitializeComponent();

Text = "Добавить партнера";

label1.Text = label1Text;

label2.Visible = false;

textBox2.Visible = false;

label3.Visible = false;

textBox3.Visible = false;

dateTimePicker1.Visible = false;

button1.Text = "Добавить";

button2.Text = "Отменить";

conFish\_Shop = connection;

label4.Visible = false;

label5.Visible = false;

comboBox1.Visible = false;

comboBox2.Visible = false;

}

public InteractionDB(string label1Text, ArrayList list,MySqlConnection connection)

{

InitializeComponent();

Text = "Изменить партнера";

label1.Text = label1Text;

textBox1.Text = list[1].ToString();

label2.Visible = false;

textBox2.Visible = false;

label3.Visible = false;

textBox3.Visible = false;

dateTimePicker1.Visible = false;

button1.Text = "Изменить";

button2.Text = "Отменить";

conFish\_Shop = connection;

dataTable = list;

label4.Visible = false;

label5.Visible = false;

comboBox1.Visible = false;

comboBox2.Visible = false;

}

private void button2\_Click(object sender, EventArgs e)

{

Close();

}

private void button1\_Click(object sender, EventArgs e)

{

String date = $"{DateTime.Now.Year}-{DateTime.Now.Month}-{DateTime.Now.Day}";

if (button1.Text == "Изменить")

{

MySqlCommand commandUpdate;

String SQLScript;

switch (Text)

{

case "Изменить товар":

SQLScript = $"UPDATE PRODUCTS SET " +

$"SUM = {textBox3.Text}, QUANTITY = {textBox2.Text}, " +

$"NAME\_PRODUCT = '{textBox1.Text}', TYPE\_PRODUCT = '{comboBox1.SelectedItem}', " +

$"MARKA\_PRODUCT = '{comboBox2.SelectedItem}', DATE\_PRODUCT = '{date}' " +

$"WHERE(ID\_PRODUCT = {dataTable[0].ToString()})";

commandUpdate = new MySqlCommand(SQLScript, conFish\_Shop);

commandUpdate.ExecuteNonQuery();

insertEntranceproducts();

break;

case "Изменить партнера":

SQLScript = $"UPDATE PARTNERS SET " +

$"NAME\_PARTNERS = '{textBox1.Text}'"+

$"WHERE(ID\_PARTNER = {dataTable[0].ToString()})";

commandUpdate = new MySqlCommand(SQLScript, conFish\_Shop);

commandUpdate.ExecuteNonQuery();

break;

case "Изменить покупателя":

SQLScript = $"UPDATE BUYER SET " +

$"FIO = '{textBox1.Text}', " +

$"PHONE\_NUMBER = '{textBox2.Text}', " +

$"DATE\_BUYER = '{date}'" +

$"WHERE(ID\_BUYER = {dataTable[0].ToString()})";

commandUpdate = new MySqlCommand(SQLScript, conFish\_Shop);

commandUpdate.ExecuteNonQuery();

break;

default:

MessageBox.Show("Непонятное завершение процесса. Обрабитесь к администратору", "ERROR");

//код, выполняемый если выражение не имеет ни одно из выше указанных значений

break;

}

}

else

{

String queryUpdateQuantity;

MySqlCommand commandInsert;

switch (Text)

{

case "Добавить товар":

queryUpdateQuantity = $"INSERT INTO PRODUCTS (NAME\_PRODUCT, QUANTITY, SUM, TYPE\_PRODUCT, MARKA\_PRODUCT, DATE\_PRODUCT) " +

$"VALUES ('{parser.parserText(textBox1.Text)}', {textBox2.Text}, " +

$"{textBox3.Text}, '{comboBox1.SelectedItem}', '{comboBox2.SelectedItem}', '{date}')";

commandInsert = new MySqlCommand(queryUpdateQuantity, conFish\_Shop);

commandInsert.ExecuteNonQuery();

insertEntranceproducts();

break;

case "Добавить партнера":

queryUpdateQuantity = $"INSERT INTO PARTNERS (NAME\_PARTNERS) " +

$"VALUES ('{parser.parserText(textBox1.Text)}')";

commandInsert = new MySqlCommand(queryUpdateQuantity, conFish\_Shop);

commandInsert.ExecuteNonQuery();

break;

case "Добавить покупателя":

queryUpdateQuantity = $"INSERT INTO BUYER (FIO, PHONE\_NUMBER, DATE\_BUYER) " +

$"VALUES ('{textBox1.Text}', " +

$"'{textBox2.Text}', " +

$"'{date}')";

commandInsert = new MySqlCommand(queryUpdateQuantity, conFish\_Shop);

commandInsert.ExecuteNonQuery();

break;

default:

MessageBox.Show("Непонятное завершение процесса. Обрабитесь к администратору", "ERROR");

//код, выполняемый если выражение не имеет ни одно из выше указанных значений

break;

}

}

Close();

}

private void insertEntranceproducts() //ДОРАБОТАТЬ

{

String date = $"{DateTime.Now.Year}-{DateTime.Now.Month}-{DateTime.Now.Day}";

String name\_product = textBox1.Text;

String type\_product = comboBox1.Text;

String marka\_product = comboBox2.Text;

int kolvo\_entranceproduct = 0;

if (Text == "Добавить товар")

{

kolvo\_entranceproduct = Convert.ToInt32(textBox2.Text);

}

else

{

// сделай скрипт для проверки прошлого значения

// из текущего вычти прошлое

// проверку тоже сделай все таки продавец дурачок <3

}

String insertQuery = $"INSERT INTO entranceproducts (name\_product, type\_product, marka\_product, data\_entrance, kolvo\_entranceproduct) " +

$"VALUES ('{name\_product}', '{type\_product}', '{marka\_product}', '{date}', {kolvo\_entranceproduct});";

MySqlCommand commandUpdate = new MySqlCommand(insertQuery, conFish\_Shop);

commandUpdate.ExecuteNonQuery();

}

}

}

// file “ReportForm.cs”

using Microsoft.Reporting.WinForms;

using MySql.Data.MySqlClient;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace FishShop

{

public partial class ReportForm : Form

{

public ReportForm()

{

InitializeComponent();

panel2.Visible = false;

}

String sqlTable = "";

Sales parentFrom;

public ReportForm(string typereport, Sales pF)

{

InitializeComponent();

panel2.Visible = false;

Text = typereport;

switch (typereport)

{

case "Отчет о поступлении товаров":

sqlTable = "entranceproducts";

break;

case "Отчет о продаже товаров":

sqlTable = "saleproducts";

break;

case "Отчет о наличии товаров":

sqlTable = "products";

break;

}

loadReportEntranceproducts();

parentFrom = pF;

}

private void loadReportEntranceproducts()

{

reportViewer1.Clear();

panel1.Visible = true;

List<ForReport> list; //get list of students

list = ForReportRepository.start();

reportViewer1.LocalReport.DataSources.Clear(); //clear report

reportViewer1.LocalReport.ReportEmbeddedResource = "FishShop.Report1.rdlc"; // bind reportviewer with .rdlc

Microsoft.Reporting.WinForms.ReportDataSource dataset = new Microsoft.Reporting.WinForms.ReportDataSource("DataSet1", list); // set the datasource

reportViewer1.LocalReport.DataSources.Add(dataset);

dataset.Value = list;

reportViewer1.LocalReport.Refresh();

reportViewer1.RefreshReport(); // refresh report

}

private void сортироватьToolStripMenuItem\_Click(object sender, EventArgs e)

{

//panel2.Visible = true;

//panel2.Dock = DockStyle.Fill;

}

private void button1\_Click(object sender, EventArgs e)

{

panel2.Visible = false;

DateTime date = dateTimePicker1.Value;

string onDate = $"{date.Year}-{date.Month}-{date.Day}";

date = dateTimePicker2.Value;

string offDate = $"{date.Year}-{date.Month}-{date.Day}";

switch (sqlTable)

{

case "entranceproducts":

parentFrom.reportParser1(sqlTable, comboBox2.Text,

comboBox1.Text, onDate, offDate);

break;

case "saleproducts":

parentFrom.reportParser2(sqlTable, comboBox2.Text,

comboBox1.Text, onDate, offDate);

break;

case "products":

parentFrom.reportParser3(sqlTable, comboBox2.Text,

comboBox1.Text, onDate, offDate);

break;

default:

MessageBox.Show("Неожиданное завершение процесса обратитесь к администратору", "Ошибка");

break;

}

loadReportEntranceproducts();

}

private void конкретныеВидыИМаркиПоДатеToolStripMenuItem\_Click(object sender, EventArgs e)

{

dateTimePicker1.CustomFormat = "yyyy-mm-dd";

dateTimePicker2.CustomFormat = "yyyy-mm-dd";

//============================

label1.Visible = true;

label2.Visible = true;

label3.Visible = true;

label4.Visible = true;

//============================

comboBox1.Visible = true;

comboBox2.Visible = true;

dateTimePicker1.Visible = true;

dateTimePicker2.Visible = true;

//============================

panel2.Visible = true;

panel2.Dock = DockStyle.Fill;

}

private void поДатеToolStripMenuItem\_Click(object sender, EventArgs e)

{

//============================

label1.Visible = false;

label4.Visible = false;

label2.Visible = true;

label3.Visible = true;

//============================

comboBox1.Visible = false;

comboBox2.Visible = false;

dateTimePicker1.Visible = true;

dateTimePicker2.Visible = true;

//============================

panel2.Visible = true;

panel2.Dock = DockStyle.Fill;

}

private void closeToolStripMenuItem\_Click(object sender, EventArgs e)

{

Close();

}

}

}

// file “ForReport.cs”

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace FishShop

{

public class ForReport

{

public int ID { get; set; }

public string Name { get; set; }

public string TYPE { get; set; }

public string Marka { get; set; }

public string Date { get; set; }

public int Kolvo { get; set; }

}

public class ForReportRepository

{

public static List<ForReport> list = new List<ForReport> { };

public static List<ForReport> GetForReports(int id, string name, string type, string marka, string date, int kolvo)

{

list.Add(new ForReport

{

ID = id,

Name = name,

TYPE = type,

Marka = marka,

Date = date,

Kolvo = kolvo

}

);

return list;

}

public static List<ForReport> start()

{

return list;

}

public static void Clear()

{

list.Clear();

}

}

}

# ЭКРАННЫЕ ФОРМЫ

# Приложение Б

# (обязательное)

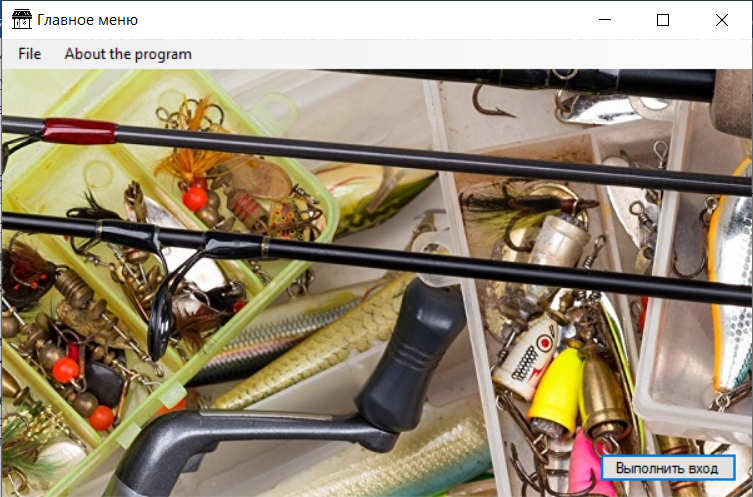


Рисунок Б.1 - Главное окно приложения

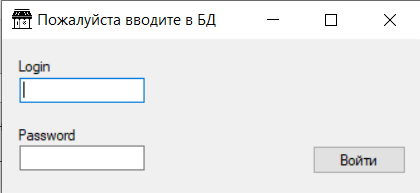


Рисунок Б.2 – Вход в свою учетную запись

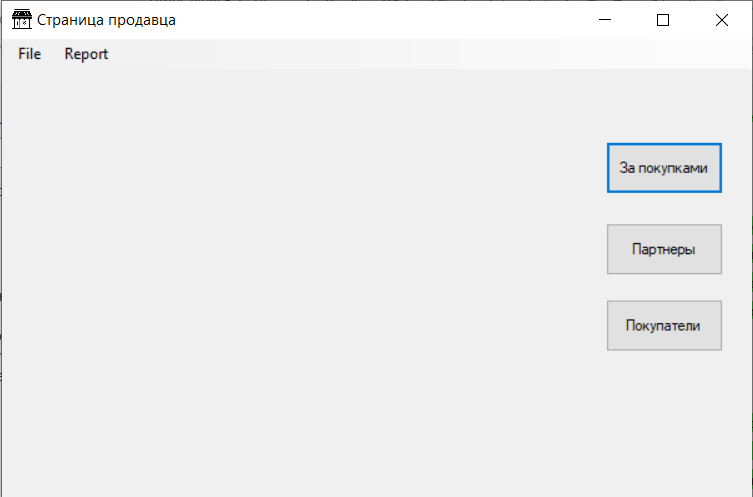


Рисунок Б.2 – Окно для работы продавца

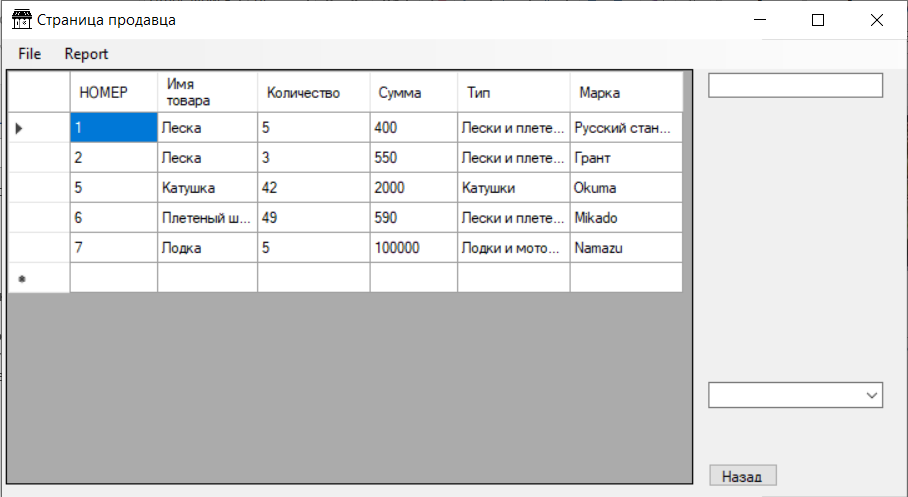


Рисунок Б.3 – Окно для выбора товара

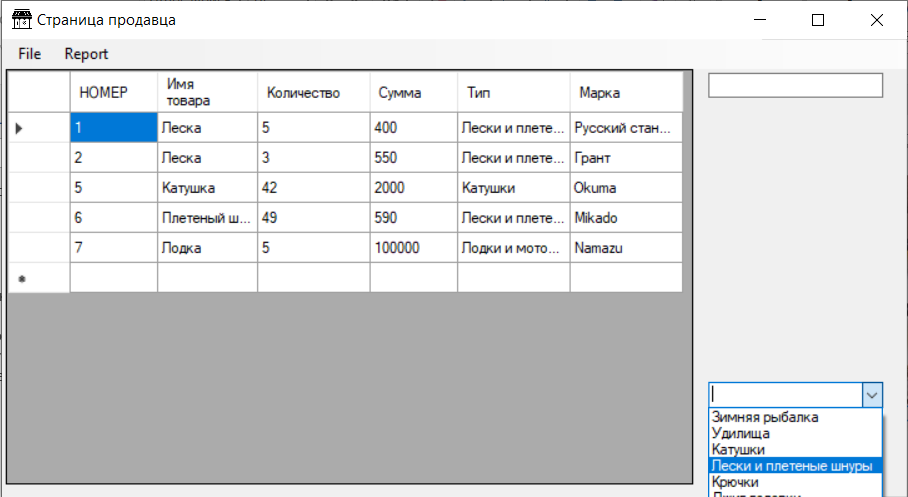


Рисунок Б.4 – Выпадающий список на окне для выбора товара

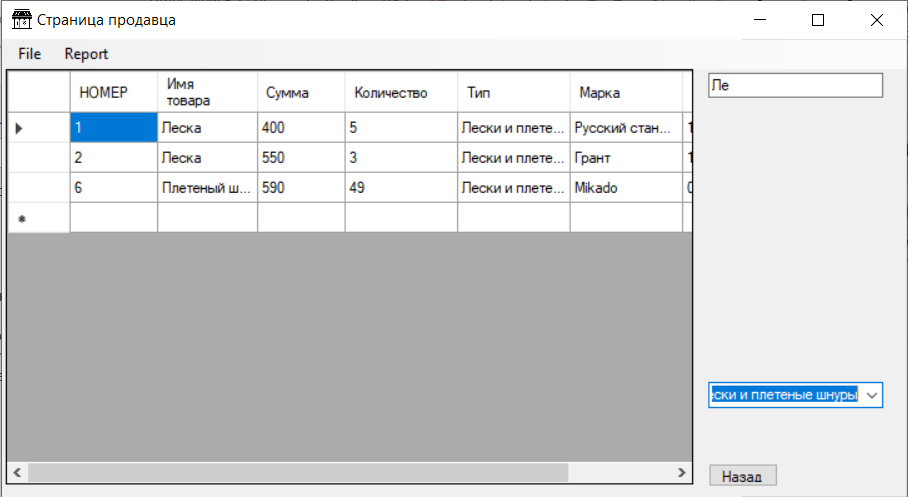


Рисунок Б.5 – Реализация учета вида продаваемых товаров

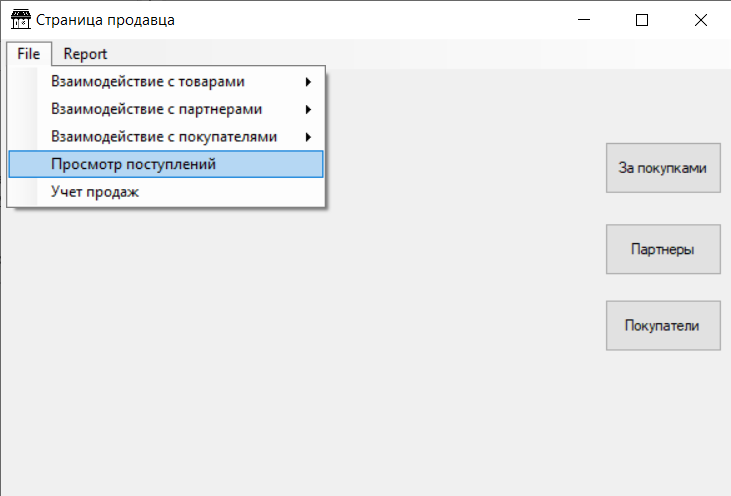


Рисунок Б.6 – Пункт меню «File»

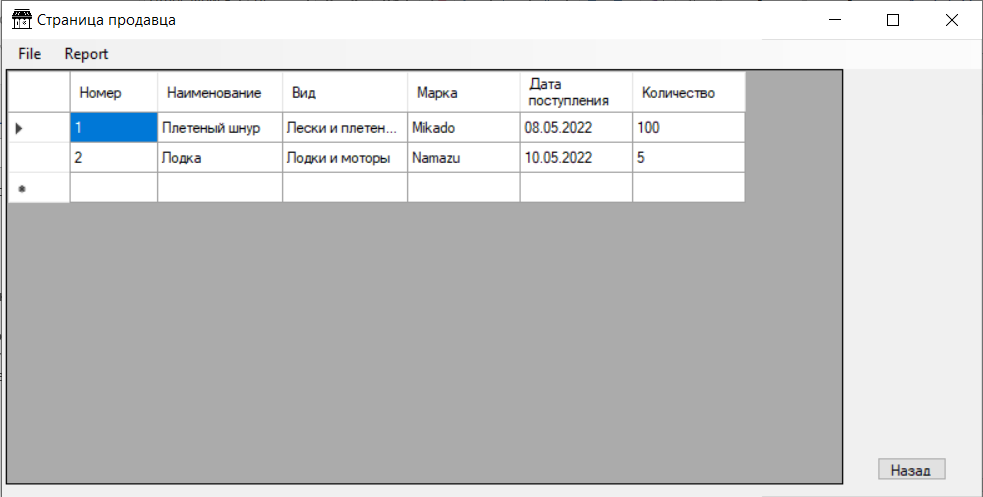


Рисунок Б.7 – Реализация учета поступления товаров

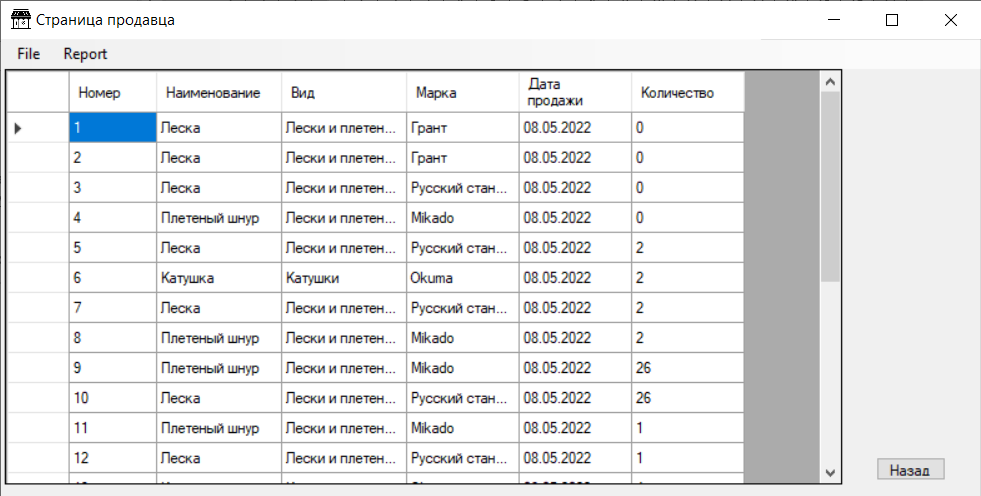


Рисунок Б.8 – Реализация учета продаж

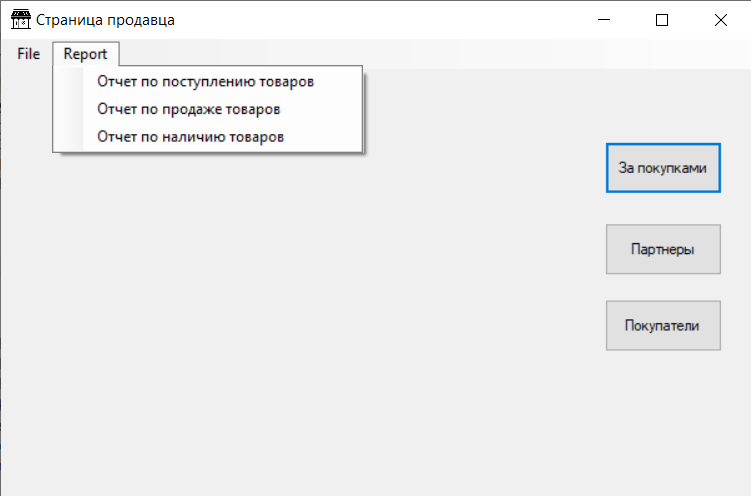


Рисунок Б.9 – Пункт меню «Report»

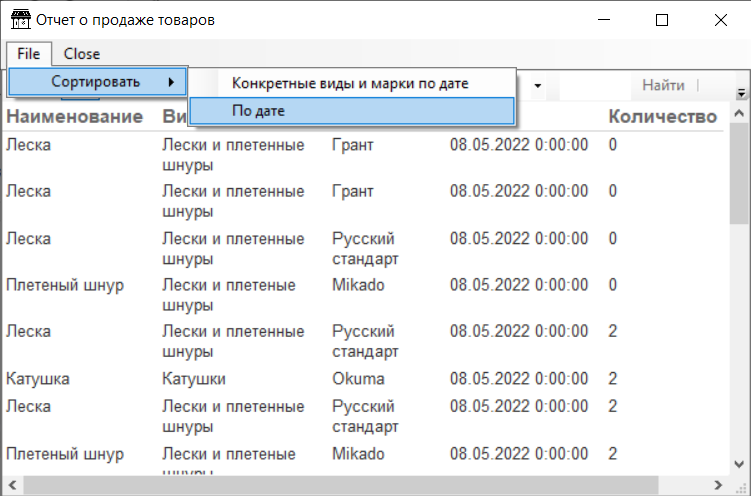


Рисунок Б.10 – Подменю «Сортировать» меню «File»

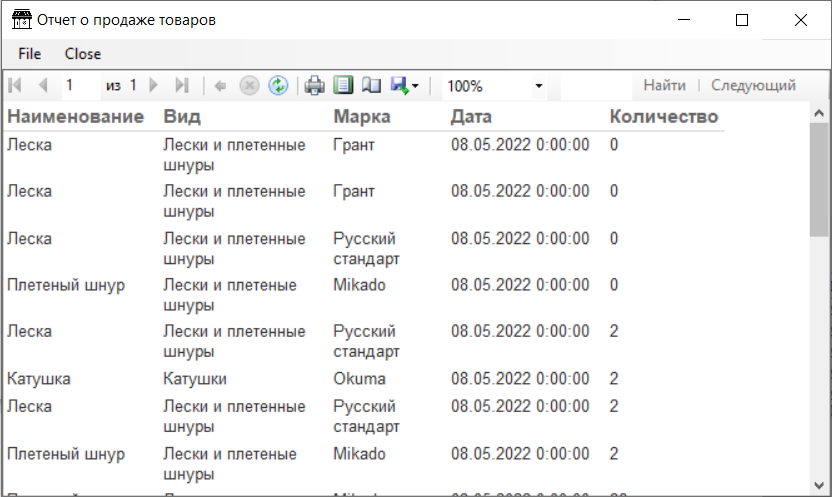


Рисунок Б.11 – Окно вывода отчетов

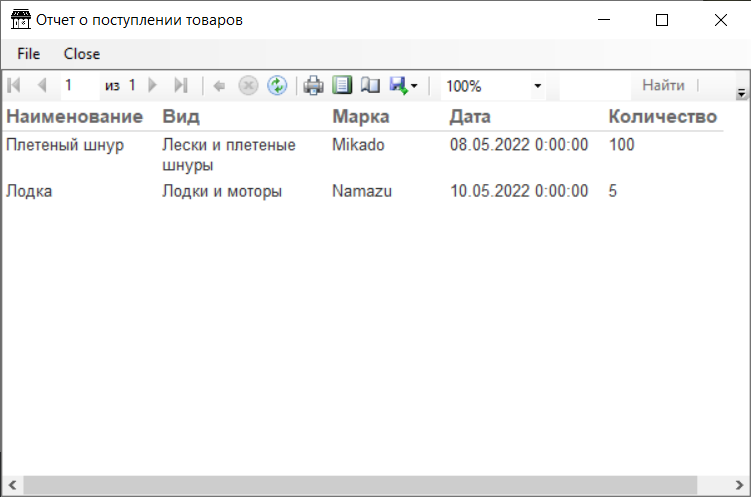


Рисунок Б.12 – Окно вывода отчета о поступлении товаров

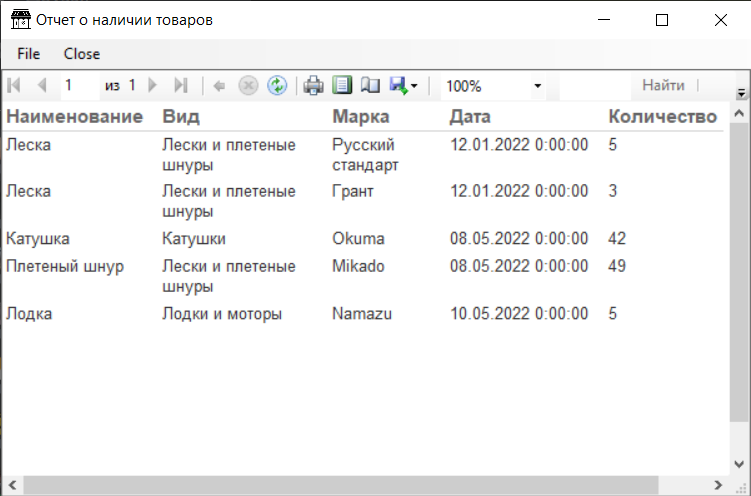


Рисунок Б.13 – Окно вывода отчета о наличии товаров