МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Институт **математики, физики и информационных технологий**

Кафедра **«Прикладная математика и информатика»**

02.03.03 МАТЕМАТИЧЕСКОЕ ОБЕСПЕЧЕНИЕ И АДМИНИСТРИРОВАНИЕ ИНФОРМАЦИОННЫХ СИСТЕМ

ТЕХНОЛОГИЯ ПРОГРАММИРОВАНИЯ

БАКАЛАВРСКАЯ РАБОТА

на тему «**Разработка программного обеспечения оптимального распределения ресурсов производственного предприятия**»

Студент П.В. Еремейчук _____

Руководитель А.И. Туищев _____

Консультант по аннотации Н.В. Ященко _____

Допустить к защите
Заведующий кафедрой к.тех.н., доцент, А.В. Очеповский _____

«_____» _____ 201_ г.

Тольятти 2017

АННОТАЦИЯ

Тема: «Разработка программного обеспечения оптимального распределения ресурсов производственного предприятия».

В данной выпускной квалификационной работе проанализирован процесс распределения ресурсов предприятия заместителем директора производственной организации ООО «Центрон», спроектировано и реализовано программное обеспечение, позволяющее оптимизировать и улучшить работу по распределению производственных ресурсов организации.

Структура ВКР представлена введением, тремя главами, заключением, списком литературы и приложением.

Во введении описывается актуальность проводимого исследования, формулируется цель и ставятся задачи, которые необходимо решить.

В первой главе проводится анализ деятельности производственного предприятия ООО «Центрон» для определения ключевых параметров в процессе распределения производственных ресурсов, которые подлежат автоматизации и оптимизации.

Во второй главе описывается проектирование программного обеспечения, в основе которого лежит алгоритм, позволяющий оптимизировать распределение производственных ресурсов.

В третьей главе будет проведена реализация и тестирование разработанного программного обеспечения, а также представлены результаты его работы.

В заключении представлены выводы по проделанной работе.

В приложении отображен программный код, с помощью которого было реализовано программное обеспечение.

В работе использовано 11 таблиц, 22 рисунков и 2 приложения, список литературы содержит 37 литературных источников. Общий объем выпускной квалификационной работы составляет 79 страниц.

ABSTRACT

The title of the graduation work is «Development of Software for Optimal Distribution of Resources of an Industrial Enterprise».

In this graduation project, the process of resource allocation of the enterprise is analyzed by the deputy director of the industrial organization OOO «Centron», software is designed and implemented that allows to optimizing and to improving the distribution of the company's industrial resources.

The structure of the graduation work is represented by an introduction, three chapters, a conclusion, a list of references and an application.

In the first chapter, the activity of the «Centron» enterprise is analyzed to determine key parameters that are subject to automation and optimization in the process of distributing production resources.

The second chapter describes the design of software, which is based on an algorithm that allows to optimize the distribution of production resources.

In the third chapter, describes the implementation and testing of the developed software, as well as the results of its work.

In conclusion conclusions on the work are presented.

The application displays the program code with which the software was implemented.

The work uses 11 tables, 22 figures and 2 applications, the list of references contains 37 literary sources. The total amount of final qualifying work is 79 pages.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

В современном мире существует огромное количество промышленных предприятий, которые за сутки расходуют большое количество различных материальных ресурсов. Их неэффективное использование приводит к увеличению расходов предприятия и лишним издержкам, которые могут привести к банкротству.

В данной ситуации становится ясно, что любое крупное промышленное предприятие не способно существовать без компетентного управления производственными ресурсами. Несмотря на то, что необходимо иметь штат профессиональных аналитиков, большинство предприятия также использует и специальное программное обеспечение, которое позволяет оптимизировать большинство экономических процессов внутри организации.

Актуальность исследования заключается в том, что при правильной оптимизации материальных ресурсов возрастает доход предприятия, благодаря понижению затрат на производственные ресурсы и сокращению производственных отходов, что в свою очередь повышает конкурентоспособность предприятия.

Объектом исследования является процесс оптимального распределения производственных ресурсов.

Предметом исследования является автоматизация оптимального расчета требуемой краски с учетом складских запасов.

Целью ВКР является разработка программного обеспечения оптимального распределения ресурсов производственного предприятия.

Для достижения поставленной цели, необходимо решить следующие **задачи**:

- изучить и проанализировать предметную область;
- изучить и проанализировать известные методы оптимизации;
- изучить и проанализировать аналоги разрабатываемого программного обеспечения;
- разработать концептуальную модель программного обеспечения;

- описать математическую модель оптимального распределения производственных ресурсов организации;
- спроектировать базу данных программного обеспечения;
- выбрать средства разработки и реализовать программное обеспечение и её базу данных;
- обосновать эффективность использования разработанного программного обеспечения.

В данной выпускной квалификационной работе рассматриваются вопросы по разработке и реализации программного обеспечения оптимального распределения ресурсов производственного предприятия.

Выпускная квалификационная работа состоит из введения, трех глав, заключения, списка использованных источников и приложения.

В первой главе приводится краткий обзор предприятия ООО «Центрон», разработана контекстная диаграмма с декомпозицией основных процессов деятельности предприятия. Проводится анализ методов, благодаря которым возможно реализовать оптимальное распределение производственных ресурсов. Предоставляется результат анализа и сравнения аналогичных программных решений, на основании которых были полученных ценные сведения, благодаря которым были сформулированы основные требования к разрабатываемому программному обеспечению.

Во второй главе описывается моделирование программного обеспечения оптимального распределения ресурсов производственного предприятия. Сформулированы основные функциональные требования для программного обеспечения. Реализована математическая модель оптимального распределения ресурсов. Смоделирована база данных программного обеспечения, проведен краткий обзор системы управления базами данных.

В третьей главе был проведён анализ основных средств реализации программного обеспечения, на основании которого были выбраны наиболее предпочтительные средства, с помощью которых будет выполняться реализация программного обеспечения. Были описаны основные принципы

работы разработанного программного обеспечения и приведены результаты тестирования методом «Чёрного ящика». После тестирования программного приложения, происходит оценка и обоснование экономической эффективности программного обеспечения.

В заключении приводятся результаты и выводы о проделанной работе.

Итогом выпускной квалификационной работы является разработанное программное обеспечение оптимального распределения ресурсов производственного предприятия. Данное приложение предназначено для оптимизации процесса распределения производственных ресурсов предприятия. Также данный программный продукт позволяет создавать отчетную документацию, платежную квитанцию.

Глава 1 ФУНКЦИОНАЛЬНОЕ МОДЕЛИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ОПТИМАЛЬНОГО РАСПРЕДЕЛЕНИЯ РЕСУРСОВ ПРОИЗВОДСТВЕННОГО ПРЕДПРИЯТИЯ

1.1 Технико-экономическая характеристика деятельности ООО «Центрон»

ООО «Центрон» - является одной из ведущих организаций в городе Тольятти, которая занимается производством и покраской металлоконструкций любой сложности.

ООО «Центрон» появилась на рынке в 2006 году, зарекомендовав себя конкурентоспособной и высококвалифицированной организацией. На данный момент организации производит около 200 тонн металлоконструкций в месяц. Данная производительность обеспечивается благодаря большой площади производственных помещений, которая составляет 5000 квадратных метров, и одной из самых больших покрасочных камер в городе Тольятти [26].

Производство данной организации увеличивается с каждым годом, в связи с чем возникают проблемы в таких сферах деятельности как:

1) логистика;

2) управление персоналом;

3) управление производственным ресурсами.

В данном случаи наибольший интерес представляет проблема управления производственными ресурсами, так как при неэффективном управлении ими организация может нести большие убытки, которых можно избежать.

Особо остро данная проблема ощущается при больших объемах производства, когда заключаются контракты с фабриками или заводами. В данной ситуации тяжело рассчитать не только необходимое количество ресурсов, но и издержки от таких заказов, которые, если не учитывать, будут сильно бить по бюджету организации.

Таким образом, компетентное управление производственными ресурсами способно значительно улучшить экономическое положение организации, так как благоприятно сказывается на таких факторах как:

- увеличение прибыли;
- повышение производительности производства;
- повышение эффективности труда;
- повышение качества применяемых решений;
- повышение качества продукции;
- снижение рисков.

В ООО «Центрон» за управление производственными ресурсами отвечает заместитель директора. Данная обязанность тесно связана с большим объемом вычислительных операций, в которых необходимо учитывать много факторов таких как:

- объем работы;
- количество необходимых ресурсов;
- стоимость производственных ресурсов;
- издержки при выполнении работы;
- время выполнение работы;
- требования заказчика к продукции.

Из вышесказанного можно легко прийти к выводу, что ошибки, допускаемые при управлении производственными ресурсами, могут легко приводить к неприятным финансовым затруднениям.

На данный момент в ООО «Центрон» отсутствует какой-то определенный подход к вычислению оптимального решения, так как достаточно продолжительное время компания не занималась большими заказами и лишь пару лет назад она вышла абсолютно на новый уровень.

Таким образом, можно сделать вывод о том, что заместитель директора принимает самостоятельное управленческое решение в зависимости от выполняемого заказа о выборе расчета производственных ресурсов для получения наибольшей выгоды при минимизации потерь.

1.2 Концептуальное моделирование процесса распределения производственных ресурсов ООО «Центрон»

Рассмотрим процесс распределения ресурсов по выполнению заказов производственной компании с позиции функционально-структурного моделирования бизнес-процессов IDEF0 (Integration Definition for Function Modeling) – нотация описания бизнес-процессов, основанная на методологии SADT [16].

SADT (Structured Analysis and Design Technique, технология структурного анализа и проектирования) - графические обозначения и подход к описанию систем.

Необходимо построить диаграмму «КАК ЕСТЬ» («AS-IS»), предназначенную для описания текущих процессов на предприятии. На рисунке 1.1 представлена контекстная диаграмма «КАК ЕСТЬ» («AS-IS»), на которой отражён основной процесс «Оптимального распределение производственных ресурсов».

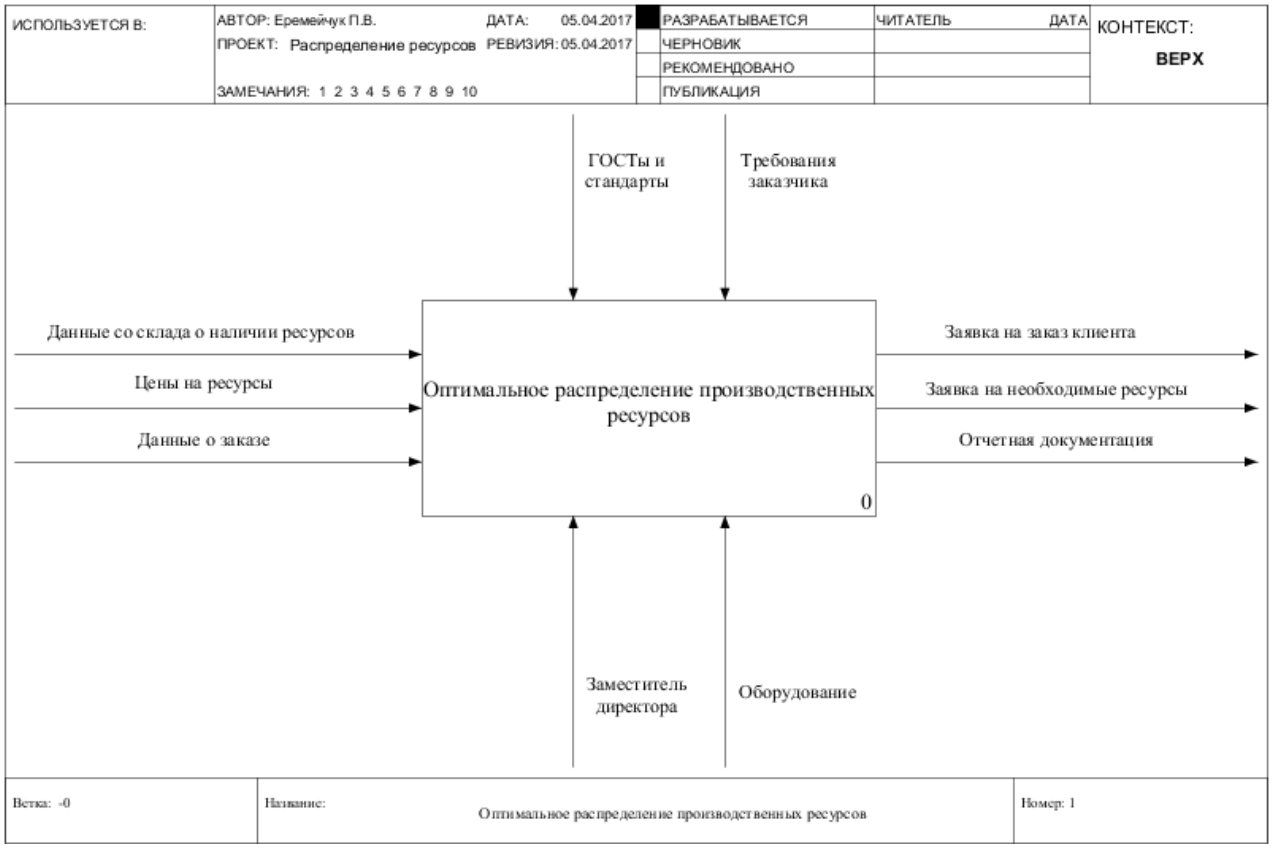


Рисунок 1.1 – Контекстная диаграмма «Оптимальное распределение производственных ресурсов»

Процесс «Оптимального распределения производственных ресурсов» включает в себя выполнение следующих процессов: «Определение необходимого количества ресурсов для выполнения заказа», «Оценка имеющихся ресурсов на складе», «Формирование списка недостающих ресурсов», «Формирование заказа на недостающие ресурсы», «Формирование заявки на заказ клиента» и «Формирование отчетной документации». Декомпозиция процесса «Оптимального распределения производственных ресурсов» представлена на рисунке 1.2.

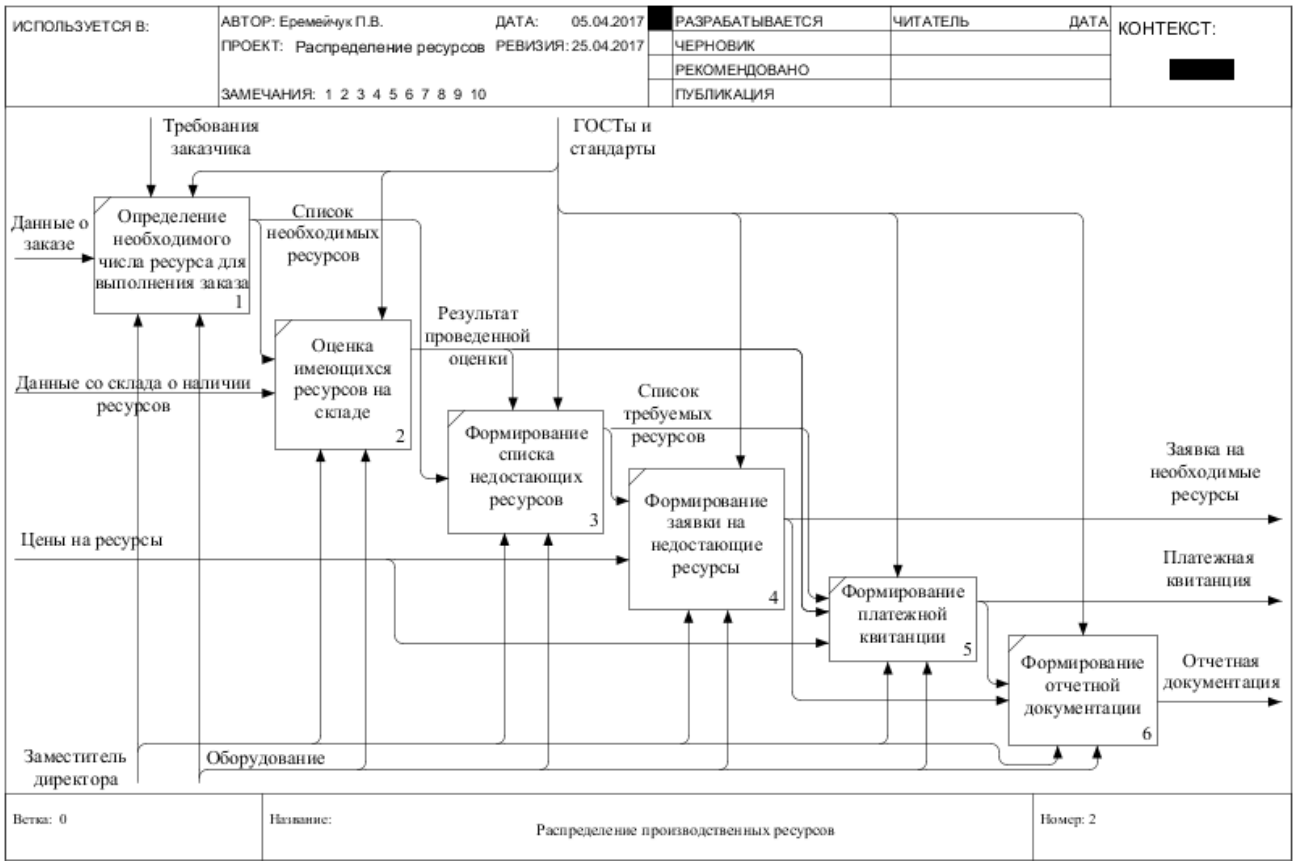


Рисунок 1.2 – Декомпозиция процесса «Оптимальное распределение производственных материалов»

Процесс: A1 – Определение необходимого числа ресурсов для выполнения заказа.

Входные данные: данные о заказе.

Алгоритм выполнения: заместитель директора, изучив данные о заказе, формирует список необходимых ресурсов и определяет их общее количество, которое необходимо для выполнения заказа.

Процесс: A2 – Оценка имеющихся ресурсов на складе.

Входные данные: данные со склада о наличии ресурсов и список необходимых ресурсов.

Алгоритм выполнения: После выполнения процесса A1, заместитель директора сопоставляет информацию, полученную со склада, со списком необходимых ресурсов, благодаря чему приходит к некому результату оценки, в который входят необходимые отчеты и документы.

Процесс: A3 – Формирование списка недостающих ресурсов.

Входные данные: список недостающих.

Алгоритм выполнения: После выполнения процесса A2, заместитель директора, опираясь на данные полученных раннее, определяет каких ресурсов недостаточно и в каком количестве, тем самым формируя список недостающих ресурсов.

Процесс: A4 – Формирование заявки на недостающие ресурсы.

Входные данные: список требуемых ресурсов и цены на эти ресурсы.

Алгоритм выполнения: После выполнения процесса A3, заместитель директора, опираясь на список требуемых ресурсов и цен на них, формирует заявку для их покупки.

Процесс: A5 – Формирование заявки на заказ клиента.

Входные данные: результат проведенной оценки, список требуемых ресурсов и цены на ресурсы.

Алгоритм выполнения: После выполнения процесса A4, заместитель директора, опираясь на сформированные ранее списки о ресурсах и цены на них, составляет платежную квитанцию.

Процесс: A6 – Формирование отчетной документации.

Входные данные: заявка на требуемые ресурсы и заявка на заказ клиента.

Алгоритм выполнения: На данном этапе, заместитель директора обновляет имеющуюся информацию о ресурсах, заказах, прибыли и убытках. Далее заместитель директора формирует соответствующие отчетные документы, которые более наглядно отображают реальное положение дел организации.

При проведении анализа распределения ресурсов заместителем директора ООО «Центрон» с использованием структурного моделирования были выявлены следующие недостатки:

- распределение производственных ресурсов – это трудоемкий процесс, требующий больших вычислительных процессов, что в свою очередь занимает много времени;
- высокая вероятность допустить ошибки при расчетах.

Решение обозначенных проблем может быть представлено в виде программного обеспечения, основанного на использовании методов или алгоритмов оптимизации производственных ресурсов. Программное обеспечение позволит минимизировать количество допущенных ошибок при расчетах и сократит время, необходимое для осуществления оптимального распределения производственных ресурсов в организации.

1.3 Методы оптимального распределения производственных ресурсов в организации

Основная задача оптимального распределения ресурсов в организации заключается в максимизации/минимизации необходимых показателей в процессе принятия решения с учетом заданных определенными условиями реальной задачи ограничений (например, общее затраченное время, стоимость, фактическую продолжительность).

Задача оптимизации заключается в поиске экстремума функции, заданной на некотором множестве [6]:

$$f: \mathbb{X} \rightarrow \mathbb{R}$$

$$f\ x\ \rightarrow min, x \in \mathbb{X} \subseteq \mathbb{G} \quad (1)$$

где $\mathbb{X}$ – множество элементов x;

$\mathbb{R}$ – множество вещественных или действительных чисел;

$\mathbb{G}$ – множество образующих.

Как правило, под задачей оптимизации также подразумевается поиск элемента x, при котором целевая функция $f\ x$ достигает экстремума.

$$x_* = arg\ \min_{x \in \mathbb{X}} f\ x\ , \mathbb{X} \subseteq \mathbb{G} \quad (2)$$

Для того, чтобы корректно поставить задачу оптимизации необходимо задать:

1. Допустимое множество $\mathbb{X}$.

2. Целевую функцию $f: \mathbb{X} \to \mathbb{R}$;

3. Критерий поиска (max или min).

Тогда для того чтобы решить задачу $f\ x \to \min_{x \in \mathbb{X}}$ необходимо:

1. Показать что $\mathbb{X} = \varnothing$;

2. Показать, что целевая функция $f\ x$ не ограничена;

3. Найти x_*;

4. Если не существует x_*, то найти $\inf_{x \in \mathbb{X}} f(x)$.

Если допустимое множество $\mathbb{X} = \mathbb{G}$, то такая задача называется задачей безусловной оптимизации, в противном случае – задачей условной оптимизации.

Таким образом, были рассмотрены основные понятия задачи оптимизации и условия, которые необходимо соблюсти, чтобы корректно сформулировать задачу оптимизации. Далее рассмотрим несколько методов оптимизации, которые способны справиться с поставленной перед нами задачей, а именно:

- метод Нелдера-Мида;
- метод множителей Лагранжа;
- симплекс-метод.

1.3.1 Метод Нелдера-Мида для решения задачи оптимизации

Основой для метода Нелдера-Мида является метод Спендли, Хекста и Химсворта. Регулярным симплексом в данном случае является выпуклая оболочка множества $(n+1)$-й равноудаленной точки в n-мерном пространстве. Данная выпуклая оболочка исследуется в методе Спендли, Хекста и Химсворта. К примеру, в двухмерном пространстве треугольник будет являться регулярным симплексом, а уже в трехмерном – это будет тетраэдр. Основная идея метода заключается в сравнении значений функции в $n+1$ вершинах симплекса и передвижении его в сторону оптимальной точки благодаря итерационной процедуре. В первоначальном варианте предложенного симплексного метода, необходимо было использовать регулярный симплекс на каждом этапе. В связи с чем Нелдер и Мид предоставили несколько усовершенствований этого метода, которые допускали, возможность неправильных симплексов. В итоге, был сформулирован очень надежный метод прямого поиска, который является одним из самых эффективных при $n \leq 6$ [22].

На рисунке 1.3 представлена блок-схема алгоритма Нелдера-Мида.

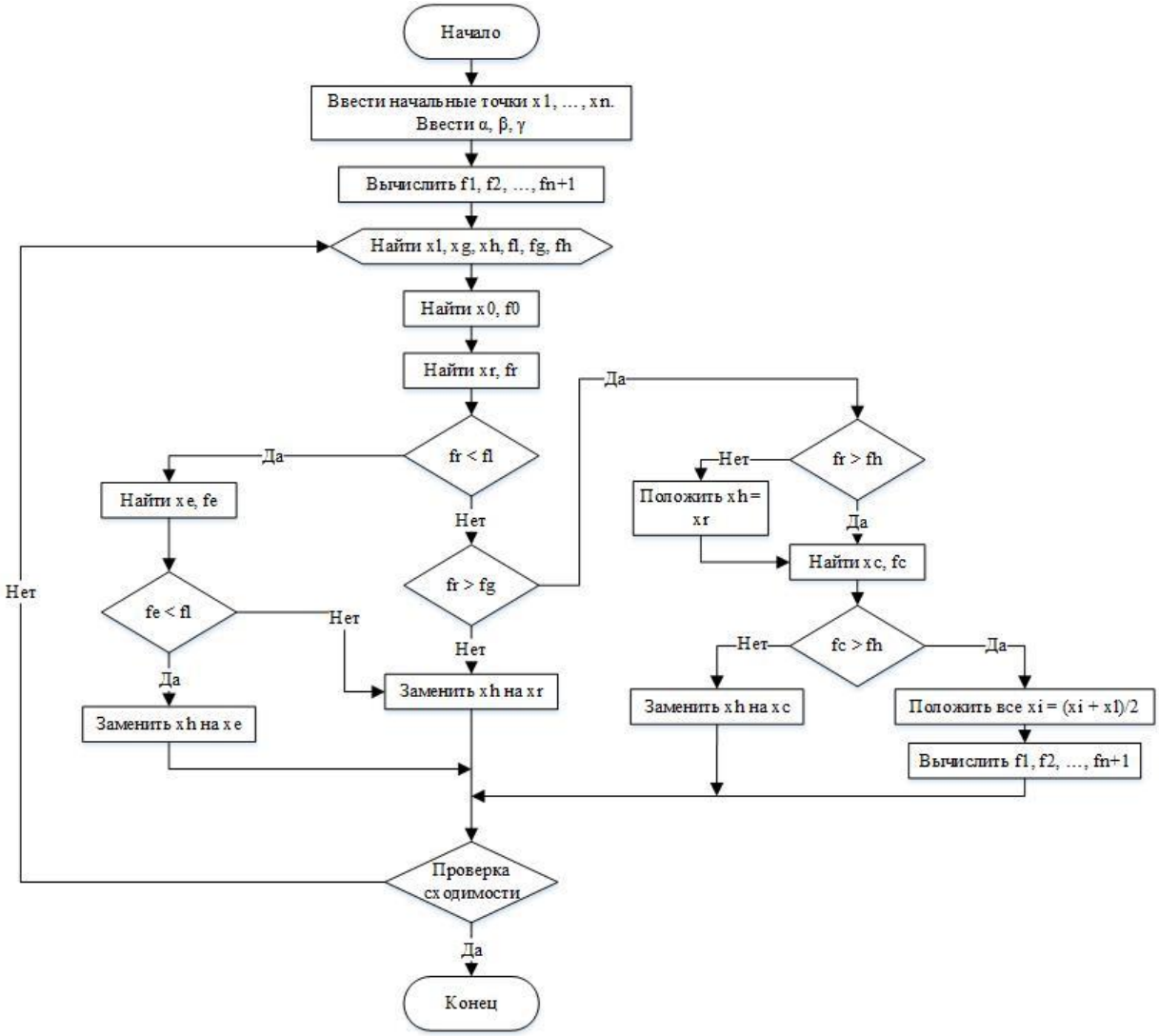


Рисунок 1.3 – Блок-схема алгоритма Нелдера-Мида

Рассмотрим данный алгоритм более подробно. Параметрами метода являются:

- коэффициент отражения $\alpha > 0$, обычно выбираются равным 1;
- коэффициент сжатия $\beta > 0$, обычно выбирается равным 0.5;
- коэффициент растяжения $\gamma > 0$, обычно выбирается равным 2.

Инициализация. Произвольным образом выбирается $n + 1$ точки $x_i = ({x_i}^1, {x_i}^2, \dots, {x_i}^n)$, образующие симплекс n -мерного пространства. В этих точках вычисляются значения функции:

$$f_1 = f\ \ {x_i}^1\ \ , f_2 = f\ \ {x_i}^2\ \ , \dots, f_{n+1} = f\ \ {x_i}^{n+1} \tag{3}$$

Происходит выбор трех точек из вершин симплекса: x_h с самым большим из выбранных точек значением функции f_h, x_g вторая по величине значения f_g

и x_l, которая обладает наименьшим значением функции f_l. Основной целью дальнейших действий является по крайней уменьшение f_h.

1. Сортировка. Происходит выбор трех точек из вершин симплекса: x_h с самым большим из выбранных точек значением функции f_h, x_g, которая обладает вторым по величине значением f_g, и x_l, которая обладает наименьшим значением функции f_l. Основной целью дальнейших действий является по крайней уменьшение f_h.

2. Необходимо найти центр тяжести данных точек, за исключением x_h: $x_c = \frac{1}{n} \sum_{i \neq h} x_i$. Вычислять $f_c = f(x_c)$ не нужно.

3. Отражение. Отражаем точку x_h относительно точки x_c с коэффициентом α, который является центральной симметрией, так как равен единице (в общем случае – гомотетия). В итоге получаем точку x_r, в которой вычисляется значении функции: $f_r = f(x_r)$. Координаты новой точки высчитываются по формуле (4):

$$x_r = (1 + \alpha)\, x_c - \alpha x_h \tag{4}$$

4. Далее происходит сравнение значения f_r со значениями f_h, f_g, f_l.

4a. Если $f_r < f_l$, то происходит растяжение. Высчитывается новая точка $x_e = (1 - \gamma)\, x_c + \gamma x_r$ и значение функции в ней $f_e = f(x_e)$.

Если $f_e < f_l$, то происходит замена точки x_h на x_e и заканчивается итерация (переходим к шагу 8).

Если $f_e < f_l$, то происходит замена точки x_h на x_r и заканчивается итерация (переходим к шагу 8).

4b. Если $f_l < f_r < f_g$, то происходит замена точки x_h на x_r и осуществляется переход на шаг 8.

4c. Если $f_h > f_r > f_g$, то значения данных точек меняются x_r, x_h (и соответствующие значения функции) местами и после происходит переход на шаг 5.

4d. Если $f_r > f_h$, то осуществляем переход на шаг 5.

5. Сжатие. Необходимо вычислить значение точки $x_s = \beta x_h + (1-\beta)x_c$ и её значение функции f_s.

6. Если $f_s < f_h$, то происходит замена точки x_h на x_s и осуществляется переход на шаг 8.

7. Если $f_s > f_h$, то необходимо произвести сжатие комплекса, то есть гомотетию к точке с наименьшим значением x_0: $x_i \to x_0 + (x_i - x_0)/2$ для всех требуемых точек x_i.

8. На последнем шаге происходит проверка сходимости. Данное действие может быть реализовано по-разному, например, оценкой дисперсии набора точек. Цель данной проверки заключается в том, чтобы убедиться во взаимной близости полученных вершин симплекса, что означает и их близость к искомому минимуму. Если необходимая точность пока ещё не достигнута, можно начать алгоритм сначала, то есть вернуться к шагу 1.

Рассматривая данный метод стало ясно, что изучение сходимости алгоритма Нелдера-Мида является сложной математической задачей. На данный момент все известные результаты о сходимости симплекс-методов основаны на таких предположениях как:

- симплекс не должен вырождаться при итерациях алгоритма;
- на гладкость функции накладываются некоторые условия.

Стоит отметить, что для метода Нелдера-Мида оба эти предположения не выполняются сразу, отсюда следует, что об условиях сходимости известно не так уж и много.

Алгоритм Нелдера-Мида отличается тем, что уже при начальных итерациях способен выдать сильное уменьшение значения функции, а также способен быстро достигать необходимой точности. В основном, алгоритм совершает одно или пару вычислений функции на каждой итерации, если не брать во внимание сжатие, используемое на практике довольно редко. Данная особенность крайне важна тогда, когда вычисление значений функции очень трудозатратный или трудоемкий по времени процесс. Для таких задач алгоритм Нелдера-Мида будет гораздо выгоднее многих других методов, которым

необходимо выполнить не менее n вычислений значений функции на каждой итерации [6].

Простота и эффективность данного метода являются, наверное, его самыми основными преимуществами.

Но стоит отметить, что из-за отсутствия теории сходимости, на практике метод способен привести к ошибочному ответу даже для гладких функций. Ещё возможна ситуация, когда рабочий симплекс расположен достаточно далеко от искомой точки, а алгоритм при этом производит большое число итерации, которые недостаточно сильно изменяют значения функции.

Таким образом, симплекс-метод Нелдера-Мида является одним из самых эффективных алгоритмом поиска экстремума функции для многих переменных, который не накладывает ограничений на гладкость функции. За время одной итерации алгоритма производится обычно одно-два вычисления значений функции, что несомненно эффективно, особенно если данные вычисления очень медленны. Основным же недостатком симплекс-метода Нелдера-Мида является отсутствие теории сходимости и существование примеров, при которых метод расходится даже на гладких функциях.

1.3.2 Метод множителей Лагранжа для решения задачи оптимизации

Метод множителей Лагранжа — это метод нахождения решения $x_1, x_2, \dots, x_n$, при котором достигается минимизация или максимизация значения функции $f(x_1, x_2, \dots, x_n)$ при ограничениях $g_1\ x_1, x_2, \dots, x_n = b_1, g_2\ x_1, x_2, \dots, x_n = b_2, g_m\ x_1, x_2, \dots, x_n = b_m$.

Суть метода множителей Лагранжа состоит в построении специальной функции Лагранжа для задачи условной оптимизации, нахождении частных производных и решении системы из этих производных и ограничений [6] [21].

Задачи условной оптимизации делятся на два вида:

1. Задача условной минимизации выполняется с использованием формулы (5).

$$f\ x_1, x_2, \dots, x_n \rightarrow min \quad \Leftrightarrow \quad f\ x_1, x_2, \dots, x_n \rightarrow min \qquad (5)$$

$$g_i\ x_1, x_2, \dots, x_n\ = b_i, \forall i \in N_m \qquad \begin{matrix} g_1\ x_1, x_2, \dots, x_n\ = b_1 \\ g_2\ x_1, x_2, \dots, x_n\ = b_2 \\ \dots \\ g_m\ x_1, x_2, \dots, x_n\ = b_m \end{matrix}$$

2. Задача условной максимизации выполняется с использованием формулы (6).

$$\begin{matrix} f\ x_1, x_2, \dots, x_n\ \to max \\ g_i\ x_1, x_2, \dots, x_n\ = b_i, \forall i \in N_m \end{matrix} \Leftrightarrow \begin{matrix} f\ x_1, x_2, \dots, x_n\ \to max \\ g_1\ x_1, x_2, \dots, x_n\ = b_1 \\ g_2\ x_1, x_2, \dots, x_n\ = b_2 \\ \dots \\ g_m\ x_1, x_2, \dots, x_n\ = b_m \end{matrix} \quad (6)$$

Рассмотрим основные шаги данного метода:

Шаг 1. Ввести набор переменных $\lambda_1, \lambda_2, \dots, \lambda_m$, называемых множителями Лагранжа, и составить функцию Лагранжа:

$$L\ x_1, x_2, \dots, x_n, \lambda_1, \lambda_2, \dots, \lambda_n\ = f\ x_1, x_2, \dots, x_n\ + \ \sum_{i=1}^{m} \lambda_i\ b_i - g_i(x_1, x_2, \dots, x_n) \quad (7)$$

Шаг 2. Нахождение частных производных:

$$\frac{\delta L}{\delta x_j}\ j = 1, n\ , \frac{\delta L}{\delta \lambda_i}(i = 1, m) \quad (8)$$

Шаг 3. Составление системы $n + m$ уравнений и с $n + m$ неизвестными $x_1, x_2, \dots, x_n, \lambda_1, \lambda_2, \dots, \lambda_n$:

$$\begin{matrix} \frac{\delta F}{\delta x_j} = \frac{\delta f}{\delta x_j} - \ \sum_{i=1}^{m} \lambda_i \frac{\delta g_i}{\delta x_j} = 0, j = 1, n \\ \frac{\delta F}{\delta \lambda_i} = b_i - g_i(x_1, x_2, \dots, x_n) = 0, i = 1, m \end{matrix} \quad (9)$$

Шаг 4. Решение полученной системы (9). При большом количестве ограничений имеет смысл применить метод Гаусса.

Шаг 5. После выполнения шага 4 выявляются все точки, в которых функция (7) может иметь экстремальные значения. Дальнейшее исследование найденных точек проводят так же, как и в случае безусловного экстремума.

Таким образом, данный метод является классическим примером решения задач нелинейного программирования. Основными преимуществами данного

метода является его доступность и понятность, но стоит отметить, что при увеличении количества ограничений возрастает трудность вычислений.

1.3.3 Симплекс-метод для решения задачи оптимизации

Основным метод для решения задач линейного программирования является симплекс-метод. В начале задачи рассматривается одна из вершин многогранника условий. С помощью математических операций проводят проверку на соответствие точки максимума (минимума), если она не соответствует, то выбирает следующую точку, увеличивая или уменьшая значение функции, данное действие зависит от того что требуется найти максимум или минимум. Получается, что с каждом последующим шагом значение функции становится лучше. Благодаря тому, что количество вершин многогранника ограничено, то за конечное количество шагов будет точно найдено оптимальное значение или будет решено, что задача нерешаема [6].

Данный метод универсален, может применятся к любой задаче линейного программирования в канонической форме. В данном случае система ограничений является системой линейных уравнений, где количество неизвестных больше количества уравнений. Если ранг системы равен r, то мы можем выбрать r неизвестных, с помощью которых выразим остальные неизвестные. В основном система линейных ограничений выглядит так:

$$\begin{aligned} x_1 &= b_1 + a_{1r+1}x_{r+1} + \cdots + a_{1n}x_n, \\ x_2 &= b_2 + a_{2r+1}x_{r+1} + \cdots + a_{1n}x_n, \\ &\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ x_r &= b_r + a_{rr+1}x_{r+1} + \cdots + a_m x_n. \end{aligned} \tag{10}$$

Устанавливая определенные значения свободным переменным и вычисляя значения базисных, будут формироваться различные решения нашей системы ограничений. На практике необходимы только те решения, когда свободные переменные равны нулю. Данные решения называются базисными, их количество совпадает с количеством базисных видов данной системы ограничений. Базисное решение является допустимым базисным решением или опорным решением тогда, когда значения переменных в нем неотрицательны.

Например, если в качестве базисных взяты переменные $X_1, X_2, \dots, X_r$, то решение $\{b_1, b_2, \dots, b_r, 0, \dots, 0\}$ будет опорным при условии, что $b_1, b_2, \dots, b_r \geq 0$.

В основе симплекс-метода лежит фундаментальная теорема симплекс-метода. Данная теорема гласит, что среди оптимальных планов задачи линейного программирования в канонической форме однозначно есть опорное решение ее системы ограничений. Таким образом, если оптимальный план задачи однозначно определен, то он совпадает с некоторым опорным решением. Количество разных опорных решений системы ограничений конечно. В связи с чем становиться понятно, что решение задачи в канонической форме можно искать обыкновенным перебором опорных решений и определения среди них того, для которого значение F самое большое (маленькое). Но стоит упомянуть, что, во-первых, абсолютно все опорные решения неизвестны и поэтому их сначала нужно найти, а, во-вторых, на практике количество этих решений может достигать большого множества и прямой перебор вряд ли возможен или по крайней мере малоэффективен. Симплекс-метод является процедурой направленного перебора опорных решений. Основываясь на найденное заранее опорное решение по определенному алгоритму симплекс-метода, возникает возможность вычислить новое опорное решение, на котором значение целевой функции F не меньше, чем на предыдущем. В конечном итоге, после ряда итераций формируется опорное решение, которое является оптимальным планом [6].

Таким образом, симплексный метод позволяет структурировать метод нахождения как первого (исходного) базисного решения, так и последующих базисных решений. Идея метода заключается в следующем.

Имеется система ограничений, которая приведена к общему виду, то есть к системе m линейных уравнений с n переменными $(m < n)$, на основании данной системы вычисляют любое базисное решение данной системы, в данном случаи важно лишь как можно быстрее его найти.

Если первое найденное базисное решение является допустимым, то его проверяют на оптимальность. Если проверка на оптимальность не дала результатов, то, происходит переход к следующему, обязательно допустимому базисному решению.

Важно понимать, что симплексный метод позволяет быть абсолютно уверенным, что при нахождении нового решения линейная форма, если и не достигнет оптимума, то станет ближе к нему. Данную процедуру повторяют до тех пор, пока не будет найдено решение, которое является оптимальным.

Бывают случаи, когда первое найденное базисное решение является недопустимым. В такой ситуации с помощью симплексного метода производят переход к другим базисным решениям, которые приближают нас к области допустимых решений, как только решение, либо базисное решение становится допустимым к нему применяют алгоритм симплексного метода, либо происходит подтверждение о противоречивости системы ограничений.

Также на рисунке 1.4 представлена блок-схема алгоритма симплекс-метода.

Таким образом, применение симплексного метода можно разделить на два основных этапа: процесс нахождения допустимого базисного решения системы ограничений или подтверждения того, что она несовместима; процесс нахождение оптимального решения.

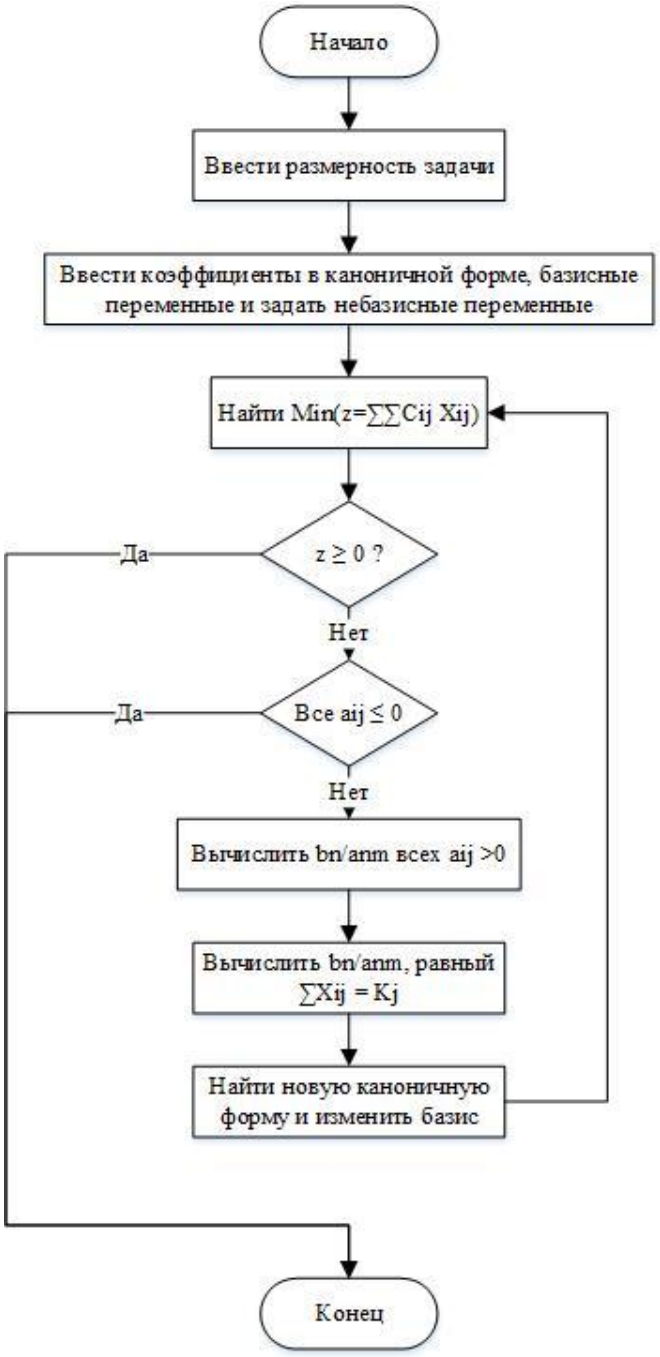


Рисунок 1.4 – Блок-схема алгоритма симплекс-метода

Таким образом, применение симплексного метода можно разделить на два основных этапа: процесс нахождения допустимого базисного решения системы ограничений или подтверждения того, что она несовместима; процесс нахождение оптимального решения.

При изучении симплексного метода был выявлен его алгоритмический характер, что позволяет с уверенностью сказать о легкости его реализации.

1.4 Разработка модели бизнес-процесса «КАК ДОЛЖНО БЫТЬ»

Для устранения недостатков, указанных ранее, была предложена функциональная модель «КАК ДОЛЖНО БЫТЬ» («TO BE»), в которой представлена автоматизация распределения ресурсов заместителем директора ООО «Центрон».

На рисунке 1.5 изображена контекстная диаграмма «TO-BE», на которой можно увидеть благодаря чему будет происходить автоматизация оптимального распределения ресурсов.

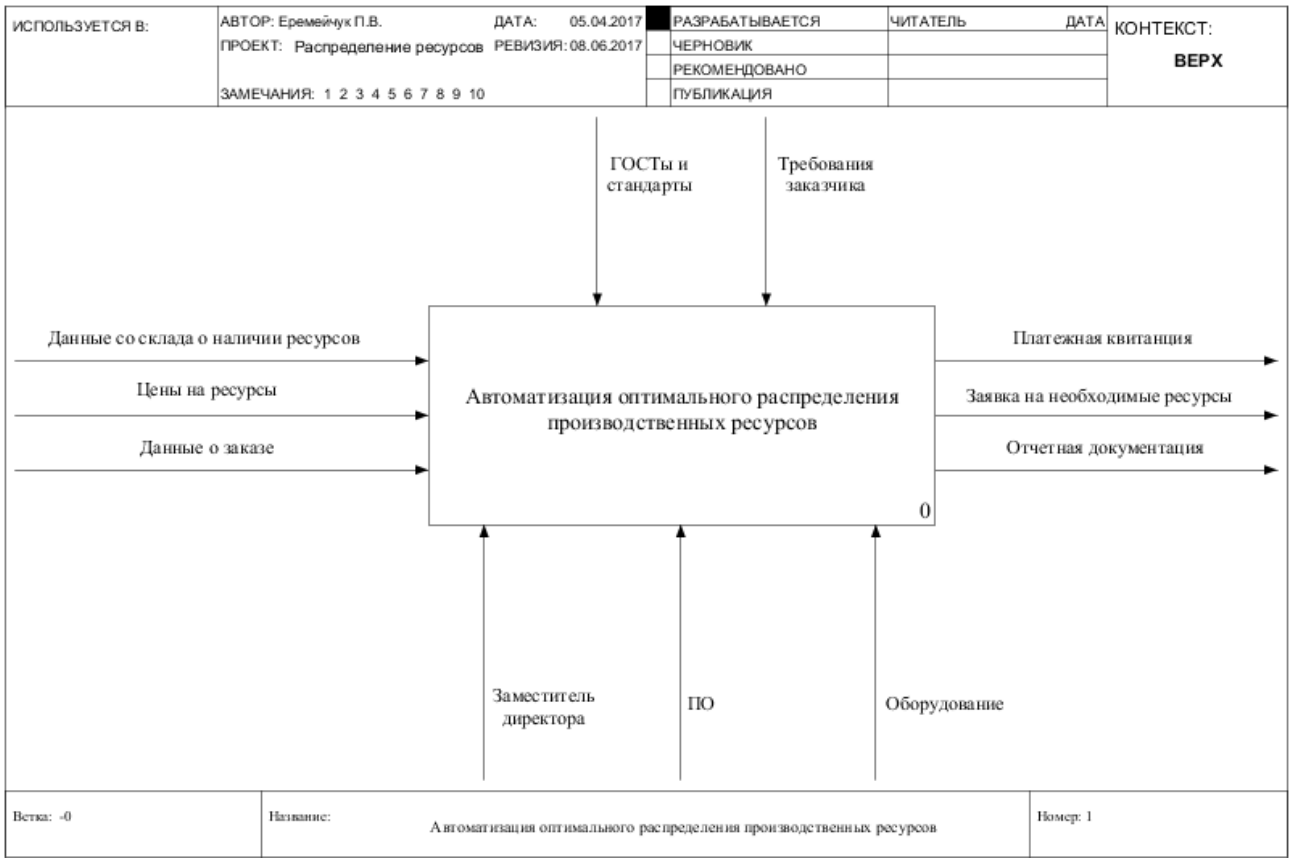


Рисунок 1.5 - Контекстная диаграмма «Автоматизация оптимального распределения производственных ресурсов»

В данном случае можно видеть, что к механизмам бизнес-процесса добавилось программное обеспечение, в основе которого лежит алгоритм, позволяющий оптимизировать распределение ресурсов предприятия и функции, которые отвечают за автоматизацию расчета ресурсов и составления различного рода документации.

Декомпозиция бизнес-процесса «Автоматизация оптимального распределения производственных ресурсов» представлена на рисунке 1.6.

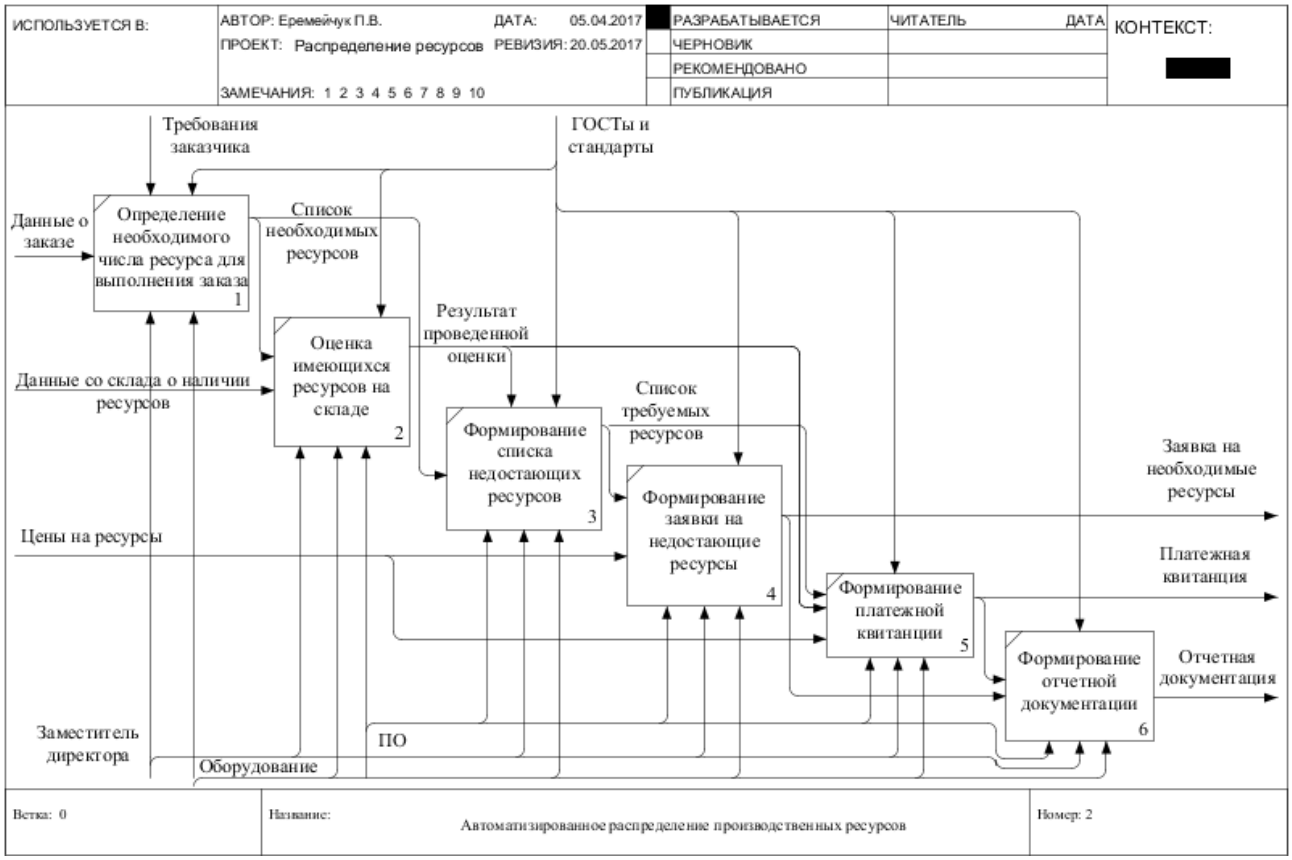


Рисунок 1.6 – Декомпозиция процесса «Автоматизация оптимального распределения производственных материалов»

В данном случаи можно видеть, что изменениям подвергнуться такие процессы как: «Оценка имеющихся ресурсов на складе», «Формирование списка недостающих ресурсов», «Формирование заявки на недостающие ресурсы», «Формирование платежной квитанции» и «Формирование отчетной документации».

Основным изменением во всех вышеперечисленных процессах является появление автоматизации. Но при этом стоит упомянуть и о внедрении нового алгоритма, который будет отвечать за распределение производственных ресурсов.

1.5 Обзор программных аналогов

Как говорилось выше, производственные мощности ООО «Центрон» с каждым годом становятся все больше и больше из-за чего, появляется всё больше клиентов с заказами любой сложности. В связи с чем увеличивается объем и сложность расчетных вычислений как на закупку материалов, так и на продажу выпускаемой продукции.

На данный момент, необходимо решить такие задачи в организации как:

- минимизировать количество допущенных ошибок при расчетах;
- сократить время, необходимое для осуществления оптимального распределения производственных ресурсов;
- повысить качество принимаемых решений.

Одним из самых очевидных решений в данной ситуации является применение программного обеспечения, которое специализируется на автоматизации процессов, протекающих в организации, а также оптимизации расходуемых ресурсов в ней.

Для начала рассмотрим программное обеспечение «Dynamics NAV», разработанное Microsoft. Данное программное обеспечение является легко адаптируемым в области планирования ресурсов предприятия (ERP) для малого и среднего бизнеса, которое позволяет автоматизировать и интегрировать процессы продаж, закупок, операционной деятельности, бухгалтерского учета и управления запасами [24].

Функционал данного программного обеспечения затрагивает различные сферы деятельности предприятия, который представлен в таблице 1.1:

Таблица 1.1 – Функционал программы «Dynamics NAV»

Тип деятельности	Функции
Управления финансами и бухгалтерский учет	Позволяет управлять наличностью, активами и банковскими услугами.
Логистическая цепочка, производство и операционная деятельность	Обеспечивает отслеживание производственных запасав, процессов, заказов и поставщиков, а также позволяет управлять ими.
Продажи и обслуживание	Позволяет управлять контактами, возможностями продажи и контрактами на обслуживание.

Продолжение таблицы 1.1

Управление проектами	Обеспечивает создание прогнозов, управление ресурсами и позволяет контролировать исполнение проектов.
Бизнес-аналитика и отчетность	Позволяет сформировать всестороннее представление о своем бизнесе и принимайте обоснованные решения.

Также «Dynamics NAV» обладает дополнительными возможностями такими как:

1. Поддержка иностранных валют, что повышает глобальную конкурентоспособность, а также позволяет работать с различными валютами и на разных языках.

2. Гибкость развертывания. Позволяет использовать систему как локально, так и в облаке в зависимости от ваших задач и возможностей.

Данный программный продукт обладает рядом полезных функции, которые позволяет пользователю достаточно подробно управлять деятельностью своего предприятия, например, прогнозирование и контролирование исполняемых проектов, но несмотря на это, в приложении отсутствует модуль, который позволил бы автоматизировать вычисление оптимального решения при распределении ресурсов предприятия.

Подобным программным обеспечением является «Конкорд XAL». Данная система относительно молода. «Конкорд XAL появилось не так давно на в России, но несмотря на это оно набирает неплохую популярность среди предпринимателей, так около 20 российских и западных компаний используют её локализованную версию.

Данное программное обеспечение является системой управления предприятием, которое представляет собой полностью интегрированное решение. Количество пользователь, которые могут работать одновременно в системе составляет 400 человек. Данные пользователи могу работать на различных платформах – от DOS и Windows до различных версий Unix. База данных системы является единой для всех. Системой обеспечивается поддержка СУБД MS SQL Server, Oracle, Sybase, DB2. «Конкорд XAL»

основано на клиент-серверных технологиях, также в него встроен аппаратно-независимый язык четвертого поколения XAL (eXtended Application Language), благодаря чему обеспечивается гибкая наращиваемость системы, которая способна подстраиваться под изменчивые требования бизнеса [20].

«Конкорд XAL» содержит все необходимые для систем такого класса модули - от «Управления производством» до «Управления персоналом». Особенность системы является детальное разбиение ее возможностей на отдельные компоненты. Это несомненно заинтересует небольшие предприятий, которые не способны приобрести продукт целиком, но могут постепенно укомплектовываться необходимыми функциями по мере возможности. Стоит отметить, что стоимость минимальной конфигурации проекта по внедрению системы составляет 20 тысяч долларов.

Заострить внимание хотелось бы на том, что «Конкорд XAL» содержит в себе практически все основные методы, с помощью которых происходит расчет объема производства, который основан на полученных заказа и плана реализации, определение параметров загрузки производственных мощностей, принятие решения относительно использования имеющихся ресурсов и т. д.

«Конкорд XAL» разработана в основном для крупного бизнеса, где присутствует сложная структура, то есть данная система обеспечивает организацию работ отдельных филиалов, подразделений и дочерних фирм с последующей консолидацией данных.

Также стоит отметить, что «Конкорд XAL» содержит в себе генератор российских отчетов, которые регламентированы российским законодательством формы строгой отчетности, система также обеспечивает обработку и оформление возврата НДС по предоплате согласно действующим инструкциям, вести электронную книгу продаж и т. д.

Таким образом, данный программный продукт является отличным решением для владельцев крупного бизнеса, так как позволяет видеть картину не только одного предприятия, но и всех филиалов, подразделений и дочерних фирм. Также стоит упомянуть и обширный функционал данного решения,

который способен выполнить практически любую задачу, но, стоит упомянуть, что стоимость минимальной конфигурации составляет 20 тысяч долларов. Данная цена для владельца малого бизнеса является очень сильным ударом по бюджету.

Аналогичным программным продуктом является «Система Галактика – Производство 9.1», в основе которого лежит концепция MRP-II, которая позволяет поддерживать задачи [28]:

- технической подготовки производства;
- управления заказами клиентов;
- планирования и управленческого учета затрат;
- расчета себестоимости;
- управление материально-техническим обеспечением;
- контроллинга.

Версия 9.1 системы “Галактика“ ориентирована на широкую массу промышленных предприятий и включает в себя следующие функциональные этапы планирования:

- планирование сбыта;
- планирование производства;
- планирование снабжения.

Планирование производства реализовано от единичного до массового изготовления продукции на заказ и на склад. возможности «Галактики» позволяют управлять долгосрочным и оперативным планированием производственных процессов – с консолидацией и декомпозицией планов. С помощью версии 9.1 системы «Галактика» возможно обеспечить полный цикл управленческого воздействия на производственный процесс, эффективно решать задачи планирования и учета от уровня предприятия до внутрицехового уровня.

При анализе данного программного обеспечения наибольший интерес представляли такие модули как [28]:

1. Управление заказами. Данный модуль обеспечивает объединение разрозненных заявок подразделений и заказов клиентов в единый портфель заказов. Средства модуля улучшают точность планирования сбыта, производства и закупок, а также позволяют быстро реагировать на изменения спроса на продукцию предприятия.

2. Материально-техническое обеспечение (МТО). Данный модуль обеспечивает объединение разрозненных заявок подразделений в единый портфель заявок на МТО. Средства данного модуля улучшают точность и оперативность планирования снабжения, а также снижает уровень страховых запасов.

3. Учет в производстве. Данный модуль реализован для того, чтобы значительно повысить оперативность производственного учета на предприятии, а также предоставлять руководителям эффективный инструмент контроля за текущим состоянием производства.

4. Производственный контроллинг. Является одним из самых важных модулей в системе, так как позволяет организовывать планирование затрат и вычисление себестоимости на стандартный бухгалтерский период и более крупные периоды.

Таким образом, данный программный продукт является отличным решением для многих предприятий, производство которых стало достаточно масштабным, а также имеется достаточно обширный персонал, который отвечает за ту или иную деятельность в организации.

В таблице 1.2 подведены итоги сравнения данных программных продуктов.

Таблица 1.2 – Сравнение программных продуктов

Программа	**Программа «Dynamics NAV»**	**Программа «Конкорд XAL»**	**Программа «Система Галактика – Производство 9.1»**
Производственный контроллинг	Отсутствует	Присутствует	Присутствует
Сложность использования	Средняя	Высокая	Средняя

Продолжение таблицы 1.2

Автоматизация при создании документации	Присутствует	Присутствует	Присутствует
Сложность использования	Средняя	Высокая	Средняя
Стоимость	В зависимости от числа рабочих мест - от 7185 рублей и выше	Около 20000 долларов	Зависит от предпочтений заказчика
Масштаб	Малый и средний бизнес	Крупный бизнес	Средний и крупный бизнес

Во время анализа данных программных аналогов было принято решение о создании такого программного продукта, который обладал бы схожими модулями, реализованными в «Галактика – Производство 9.1», но при этом был бы прост в использовании как «Dynamics NAV», а также не маловажной часть является ориентированность продукта на представителей малого бизнеса.

Вывод по первой главе

После анализа деятельности ООО «Центрон» было выявлено, что на данный момент в организации отсутствует какой-то определенный подход к вычислению оптимального решения при управлении производственными ресурсами в связи с чем, заместитель директора не застрахован от случайных ошибок при расчетах, а также отсутствие автоматизации некоторых процессов требует от него больших временных затрат.

Автоматизация данных процессов позволит значительно сократить время необходимое заместителю директора на составление различной отчетной документации и расчет оптимального решения, которое в свою очередь будет устойчиво к ошибкам в расчетах.

Таким образом, было принято решение о создании программного обеспечения, которое позволит оптимизировать управление ресурсами предприятия. Для достижения данной задачи требуется выбрать средства и технологию программирования, после чего реализовать данное программное приложение.

Глава 2 МОДЕЛИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ОПТИМАЛЬНОГО РАСПРЕДЕЛЕНИЯ РЕСУРСОВ ПРОИЗВОДСТВЕННОГО ПРЕДПРИЯТИЯ

2.1 Обоснование архитектуры проектируемого программного обеспечения

В связи с тем, что разрабатываемое программное обеспечение будет локальным и пользоваться им в основном будет только заместитель директора, было решено реализовать архитектуру «файл-сервер».

Файл-сервер – это централизованное хранилище информации, доступ к дискам которого имеют подключенные в локальную сеть персональные компьютеры. Основная задача файлового сервера сводится к надежному сохранению данных и бесперебойному доступу к ней, а в случае повреждения файлов – полному их восстановлению [27].

Как было сказано выше, основным пользователем разрабатываемого приложения будет заместитель директора, поэтому проблема увеличения роста пользователей вряд ли даст о себе знать в ближайшее время также, как и проблема крупных денежных вложений на модернизацию и сопровождение пользовательских станций, так как в данном случаи она всего лишь одна.

Модель архитектуры «файл-сервер» представлена на рисунке 2.1.

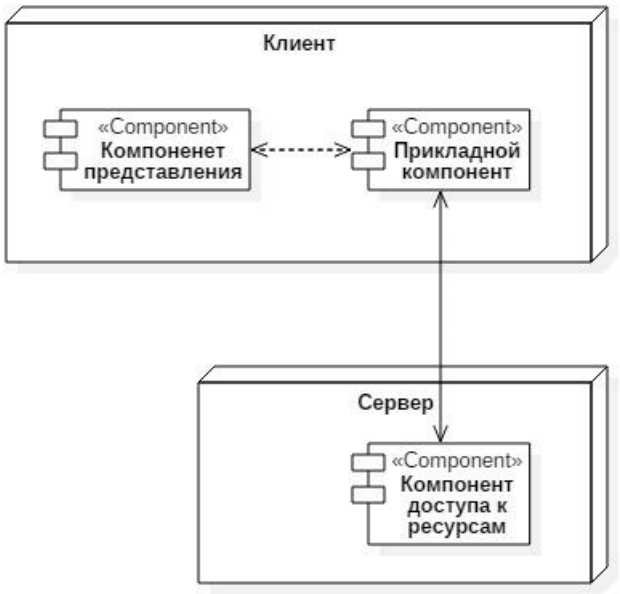


Рисунок 2.1 – Модель архитектуры «файл-сервер»

Также необходимо определить основные требования к будущему программному обеспечению. В данном случае было принято решения воспользоваться методологией FURPS+.

FURPS+ является классификацией требований к программным системам [31]. В приложении А представлены требования к программному приложению по данной методологии.

Таким образом, были выявлены основные требования разрабатываемого программного обеспечения по методологии FURPS+ и выбрана архитектура «файл-сервер».

Далее необходимо смоделировать диаграмму вариантов использования для определения общих границ и контекста моделируемой предметной области, общих требований к функциональному поведению системы.

2.2 Концептуальное моделирование программного приложения для производственного предприятия

2.2.1 Описание процесса взаимодействия пользователя и программного приложения

Описание взаимоотношений и зависимостей между группами вариантов использования и действующими лицами, участвующими в процессе, возможно с применение диаграммы вариантов использования.

Диаграмма вариантов использования предназначена для упрощения взаимодействия с будущими пользователями системы, с клиентами, и особенно пригодится для определения необходимых характеристик системы. Другими словами, диаграмма вариантов использования говорит о том, что система должна делать, не указывая на применяемые методы [29].

Создание диаграммы вариантов использования будем производить с помощью графического языка моделирования UML. Для этого было решено воспользоваться программой StarUML.

UML – это унифицированный графический язык моделирования для описания, визуализации, проектирования и документирования объектно-

ориентированных систем. UML призван поддерживать процесс моделирования программного обеспечения на основе объектно-ориентированного подхода, организовывать взаимосвязь концептуальных и программных понятий, отражать проблемы масштабирования сложных систем [29].

В таблице 2.1 представлено описание прецедентов, которые будут реализованы в разрабатываемом программном обеспечении.

Таблица 2.1 – Таблица прецедентов

№	Прецедент	Краткое описание
1	Составление отчетной документации	Ведение учета поступлений и расходов ресурсов, а также учет заказов и клиентов
2	Определение необходимого количества ресурсов	Расчет необходимых ресурсов для выполнения заказа на основании предоставленных заказчиком требований
3	Формирование платежной квитанции	Расчет стоимости заказа
4	Формирование заявки на недостающие ресурсы	Определение количества недостающих ресурсов на складе с их последующей закупкой

На рисунке 2.2 изображена диаграмма вариантов использования разрабатываемого программного обеспечения заместителем директора.

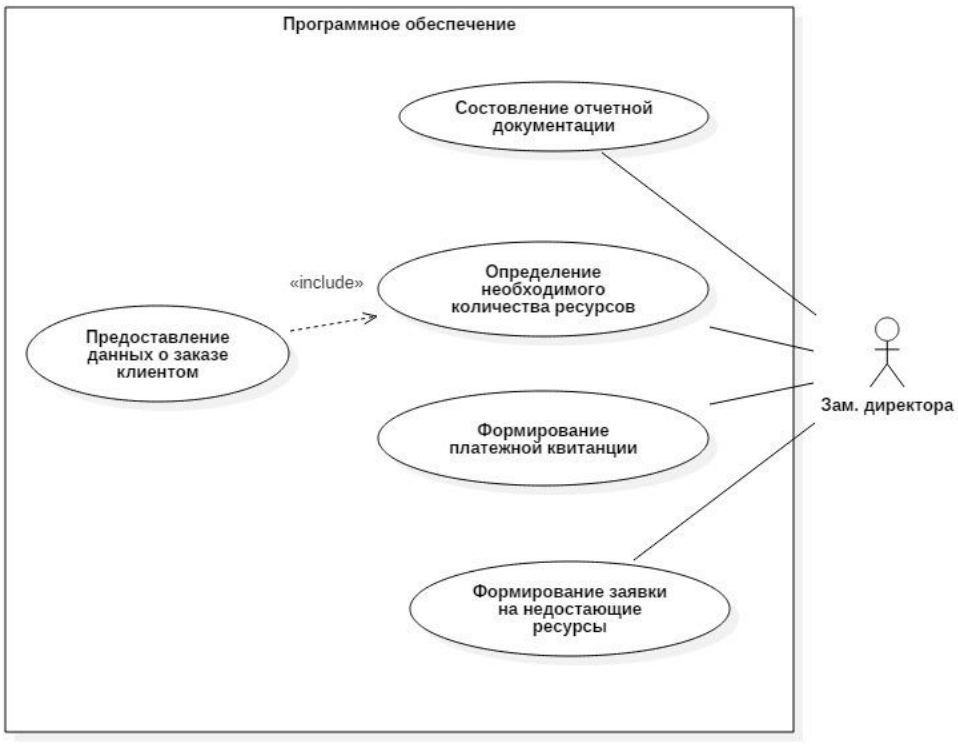


Рисунок 2.2 – Диаграмма вариантов использования разрабатываемого программного обеспечения заместителем директора

«Зам. директора» имеет четыре варианта использования «Составление отчетной документации», «Определение необходимого количества ресурсов» «Формирование платежной квитанции» и «Формирование заявки на недостающие ресурсы».

Далее в таблицах 2.2 – 2.5 представлены спецификации основных прецедентов.

Таблица 2.2 – Описание прецедента «Составление отчетной документации»

Прецедент: Составление отчетной документации
ID: 1
Краткое описание: Ведение учета поступлений и расходов ресурсов, а также учет заказов и клиентов
Главные актеры: Заместитель директора
Второстепенные актеры:
Предусловие: Прецедент начинается по инициативе заместителя директора
Основной поток: Составление отчетной документации
Постусловия: Составлена отчетная документация
Альтернативные потоки: Нет

Таблица 2.3 – Описание прецедента «Определение необходимого количества ресурсов»

Прецедент: Определение необходимого количества ресурсов
ID: 2
Краткое описание: Расчет необходимых ресурсов для выполнения заказа на основании предоставленных заказчиком требований
Главные актеры: Заместитель директора
Второстепенные актеры:
Предусловие: Прецедент начинается по инициативе заместителя директора

Основной поток: 1. Предоставление данных о заказе клиентом 2. Определение необходимого количества ресурсов

Продолжение таблицы 2.3

Постусловия: Определено необходимое количество ресурсов
Альтернативные потоки: Нет

Таблица 2.4 – Описание прецедента «Формирование платежной квитанции»

Прецедент: Формирование платежной квитанции
ID: 3
Краткое описание: Расчет стоимости заказа
Главные актеры: Заместитель директора
Второстепенные актеры:
Предусловие: Прецедент начинается по инициативе заместителя директора
Основной поток: Формирование платежной квитанции
Постусловия: Сформулирована платежная квитанция
Альтернативные потоки: Нет

Таблица 2.5 – Описание прецедента «Формирование заявки на недостающие ресурсы»

Прецедент: Формирование заявки на недостающие ресурсы
ID: 4
Краткое описание: Определение количества недостающих ресурсов на складе с их последующей закупкой
Главные актеры: Заместитель директора
Второстепенные актеры:
Предусловие: Прецедент начинается по инициативе заместителя директора
Основной поток: Формирование заявки на недостающие ресурсы
Постусловия:

Сформулирована заявка на недостающие ресурсы
Альтернативные потоки: Нет

Модель вариантов использования отображает границы моделируемой предметной области, формулирует общие требования к функциональному поведению проектируемого web-представительства. При этом были выделены функции, которые система будет исполнять. Необходимо проверить функции проектируемой логической схемы.

2.2.2 Логическое моделирование программного обеспечения

Диаграмма классов – это статическая структурная диаграмма, которая описывает структуру системы, демонстрирует её классы, их атрибуты, методы и зависимости между классами [29]. На рисунке 2.3 изображена диаграмма классов разрабатываемого программного приложения.

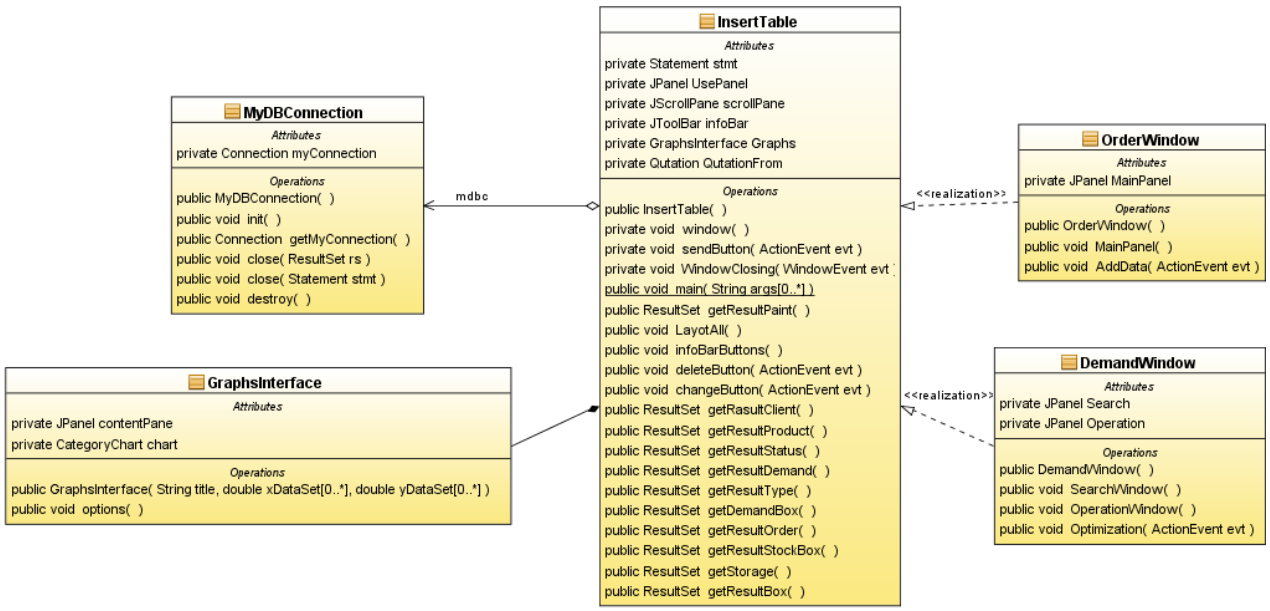


Рисунок 2.3 – Диаграмма классов программного приложения

Класс «InsertTable» включает в себя атрибуты: «stmt» - состояние подключения к БД, «UsePanel» - окно дополнительных функций, «scrollPane» - окно с элементом прокрутки, «InfoBar» - панель инструментов, «Graphs» - график-отчет, «QutationForm» - объект платежная квитанция, так же класс включает в себя 19 методов: Конструктор, LayoutAll() – функция размещения

элементов на окне, InfoBarButtons() – функция создания кнопок на панели инструментов, UseJPanel() – функция создания окна дополнительных функций, deleteButton(ActionEvent evt) – удаляет запись из таблицы, changeButton(ActionEvent evt) – изменяет значение в таблице, sendButton(ActionEvent evt) – добавляет значение в таблицу. Остальные 11 методов отвечают за отображение информации из БД в соответствующей таблице, например, getResultClient() – функция, которая отображает информацию из таблицы базы данных в таблице приложения «Клиенты».

Класс «MyDBConnection», включающий в себя атрибут: «myConnection» - отвечает за соединение с БД, так же класс включает в себя 6 методов: Конструктор, init() – создает подключение, getMyConnection() – возвращает состояние подключения, close(ResultSet rs) – закрывает соединение с таблицой, close(Statement stmt) -–закрывает соединение с БД, destroy() – разрушает объект MyBDConenntction.

Класс «GraphsInterface» включает в себя атрибуты «ContentPane» - панель для графика, «chart» -координата, а также класс включает в себя GraphsInterface(String title, double xDataSet[0..*], double yDataSet[0..*]) – конструктор, options() – настройки графика.

Класс «OrderWindow» включает в себя атрибуты «MainPanel» - форма, а также класс включает в себя конструктор, MainPanel(), AddData (ActionEvent evt) – добавление данных в таблицу.

Класс «DemandWindow» включает в себя атрибуты «Search» - панель с критериями поиска, «Operation» - панель результатов, а также класс включает в себя конструктор, SearchWindow() – создание окна поиска, OperationWindow() – создание окна результатов, Optimization(ActionEvent evt) – функция расчета оптимального распределения.

При помощи диаграммы классов была выполнена проверка логики информационной системы. Выделены конкретные классы, которые будут основой реализации проектируемого программного обеспечения.

2.3 Описание математической модели оптимального распределения ресурсов производственного предприятия

В данном случае, основываясь на анализе, проведенном в первой главе, было решено отказаться от методов линейного программирования для нахождения оптимального решения, даже несмотря на то, что они хорошо изучены и реализованы как в виде алгоритмов, так и в виде программ.

Из-за нестандартности поставленной задачи необходимо было найти другие пути поиска решений задачи исследования вместо алгоритмов линейного программирования. Поэтому во время анализа также были представлены наиболее мощные и удобные в применении к задаче долгосрочного планирования и управления промышленным предприятием алгоритмы поиска экстремума.

На основании чего, был сделан выбор в сторону метода множителей Лагранжа. Одними из основных преимуществ данного метода является простота реализации и точность вычислений.

Далее стоит рассмотреть постановку задачи, определение переменных и ограничений, а также формирование целевой функции.

Постановка задачи формулируется следующим образом:

«Предприятию необходимо выполнить заказ, бюджет которого не должен превышать некоторой суммы S. Известно, что на выполнение всего заказа необходимо P килограмм порошковой краски. Порошковая краска продается в коробках по Z_1 или Z_2 килограмм. Стоимость за коробку без учета скидки составляет Y_1 и Y_2 соответственно. Каждая коробка имеет свой объем V_1 и V_2. Площадь, которая выделена под хранение данных ресурсов, имеет объем равный V_s. Организация обладает надежным поставщиком производственных ресурсов, в связи с чем имеет скидку на оптовую закупку, которая составляет 5% от стоимости коробки при покупке в количестве, равным 5 коробкам, и возрастает на 1% при покупке последующих коробок, то есть при покупке 8 коробок скидка на них будет уже 8%. Данная скидка возрастает до 40%».

Данные о коробках для наглядности представлены в таблице 2.6.

Таблица 2.6 – Характеристика коробок с порошковой краской

Тип коробки	**Вес коробки, кг**	**Объем коробки, м^3**	**Цена за коробку**
1	Z_1	V_1	Y_1
2	Z_2	V_2	Y_2
…	…	…	…
N	Z_N	V_N	Y_N

Таким образом, стоимость коробок с учетом скидки задается формулой:

$$D_N = Y_N - \frac{x_N * Y_N}{100}, \quad (11)$$

где: x_n – это количество покупаемых коробок типа N,

$D_N -$ цена коробки с учетом скидки.

Таким образом, необходимо определить, сколько коробок каждого вида необходимо взять, чтобы минимизировать остаток краски на складе.

Далее необходимо составить математическую модель нахождения оптимального плана закупки коробок с порошковой краской.

Для этого необходимо определить:

$x_1 -$ план закупки коробок 1-го типа;

$x_2 -$ план закупки коробок 2-го типа;

$x_N -$ план закупки коробок N-го типа.

Данные величины являются переменными модели. Обязательства из условий задачи дают следующие ограничения на их значения: $x_1, x_2, \dots, x_N \geq 0$. Так как общий бюджет на заказ составляет S, то возникает следующее ограничение:

$$Y_1 - \frac{x_1 * Y_1}{100} * x_1 + Y_2 - \frac{x_2 * Y_2}{100} * x_2 + \cdots + Y_N - \frac{x_N * Y_N}{100} * x_N \leq S. \quad (11)$$

Также возникает ограничение, связанное с вместительностью склада, которое имеет вид:

$$V_1 * x_1 + V_2 * x_2 + \cdots + V_N * x_N \leq V_S. \quad (12)$$

Последним ограничением является количество краски, которое необходимо для выполнения заказа:

$$Z_1 * x_1 + Z_2 * x_2 + \cdots + Z_N * x_N \geq P. \quad (13)$$

Целевая функция, отвечающая за расчет остатков краски на складе после выполнения заказа, выглядит так:

$$F = Z_1 * x_1 + Z_2 * x_2 + \cdots + Z_N * x_N - P \quad (14)$$

В итоге, математическая модель задачи минимизации остатка краски на складе имеет вид:

$$F = Z_1 * x_1 + Z_2 * x_2 + \cdots + Z_N * x_N - P \rightarrow min,$$
$$Y_1 - \frac{x_1 * Y_1}{100} * x_1 + Y_2 - \frac{x_2 * Y_2}{100} * x_2 + \cdots + Y_N - \frac{x_N * Y_N}{100} * x_N \leq S,$$
$$V_1 * x_1 + V_2 * x_2 + \cdots + V_N * x_N \leq V_S,$$
$$Z_1 * x_1 + Z_2 * x_2 + \cdots + Z_N * x_N \geq P,$$
$$x_1, x_2, \ldots, x_N \geq 0.$$

Таким образом, была сформулирована постановка задачи, определены искомые переменные и система ограничений, а также создана целевая функция.

2.4 Проектирование базы данных программного обеспечения

2.4.1 Концептуальное проектирование модели данных программного обеспечения

На данном шаге необходимо разработать концептуальную модель данных программного обеспечения. Средством моделирования предметной области на данном этапе является модель «сущность-связь» или ER-моделью. В ней моделирование структуры данных предметной области базируется на использовании графических средств – ER-диаграмм. В наглядном виде они представляют связи между сущностями [18].

На рисунке 2.4 изображена ER-модель данных.

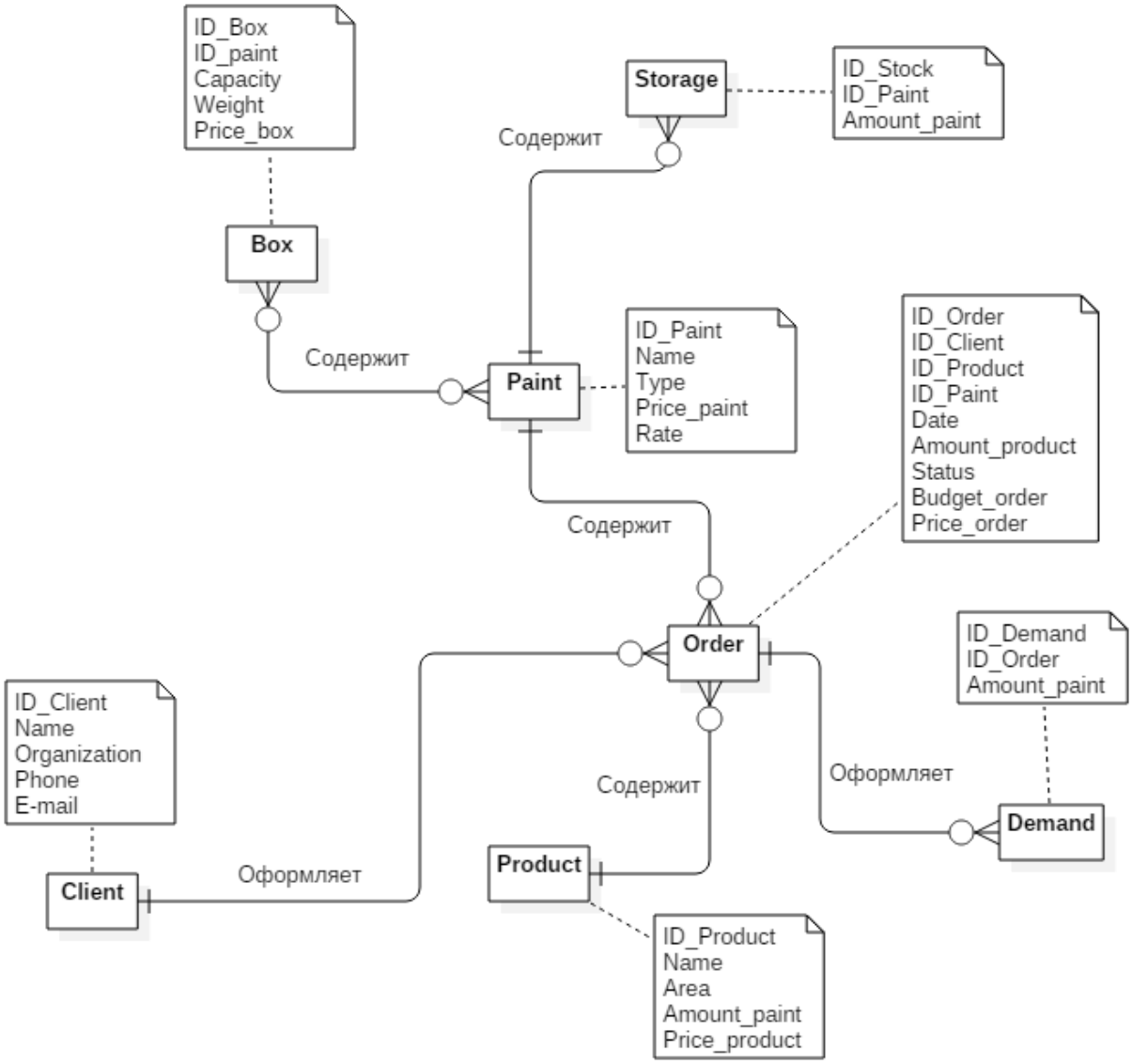


Рисунок 2.4 - Концептуальная ER-модель данных

На данной ER-модели представлено 7 сущностей:

1. Сущность «Client», которая имеет такие атрибуты как: «Name», «Organization», «Phone», «E-mail» и «ID_Client», который является первичным ключом. Сущность «Client» связана с сущностью «Order» с помощью связи «Оформляет», которая имеет мощность «один ко многим».

2. Сущность «Order», которая имеет такие атрибуты как: «ID_Client», «ID_Product», «ID_Paint», «Amount_paint», «Date», «Status», «Budget_order», «Price_order» и «ID_order», который является первичным ключом. Сущность «Order» связана с сущностями «Paint» и «Product» по средству связи

«Добавляет», которая имеет мощность «один ко многим». А также имеет связь «Формирует», мощность которой «один ко многим», с сущностью «Demand».

3. Сущность «Paint», которая имеет атрибуты: «Name», «Type», «Price_paint», «Rate» и «ID_Paint», который является первичным ключом. Сущность «Paint» связана с сущностью «Storage» с помощью связи «Содержит», которая имеет мощность «один ко многим», а также связана с сущностью «Box» с помощью связи «Содержит», которая имеет мощность «многие ко многим».

4. Сущность «Product», которая имеет такие атрибуты как: «Name», «Area», «Amount_paint», «Price_product и «ID_Product», который является первичным ключом. Данная сущность имеет связь только сущностью «Order».

5. Сущность «Storage», которая имеет такие атрибуты как: «ID_Paint», «Amount_paint» и «ID_stock», который является первичным ключом. Данная сущность имеет связь «Содержит», мощность которой «один ко многим», с сущностью «Paint».

6. Сущность «Demand», которая имеет такие атрибуты как: «ID_Order», «Amount_paint» и «ID_Demand», который является первичным ключом.

7. Сущность «Box», которая имеет такие атрибуты как: «ID_Paint» «Capacity» «Weight» «Price_box» и «ID_Box», который является первичным ключом.

Благодаря ER-модели были определены семь основных сущностей, а также мощности связей между ними. На следующем этапе, чтобы избежать логических ошибок, следует провести нормализацию.

2.4.2 Построение логической модели данных

На этом этапе происходит проектирование логической модели, которая представляет из себя создание схемы базы данных на основе конкретной модели данных.

Логическая модель содержит набор схем отношений, в которых указан первичный ключ, «связи» между отношениями, который являются внешними ключами [11].

Перед построением логической модели необходимо обязательно провести нормализацию данных, что позволит уменьшить количество ошибочной информации, которая содержится в базе данных, для этого стоит использовать «Нормальные формы».

Под нормальной формой подразумевают свойство конкретного отношения в модели данных, характеризующего его с точки зрения избыточности, которая способствует появлению ошибочных результатов выборки или изменения данных [23].

Нормальная форма – это совокупность требований, которым должно удовлетворять отношение.

Нормализацией схемы базы данных является процедура, которая происходит над базой данных для того, чтобы устранить в ней избыточность. Нормализация позволяет добиться не малых преимуществ. К примеру, в нормализованной базе данных уменьшается шанс возникновения ошибок и размер базы данных на жестком диске [18].

В данном случаи были использованы две нормальные формы:

- первая форма нормализации;
- вторая форма нормализации.

Первая нормальная форма отвечает за то, чтобы все записи в базе данных представляли из себя один экземпляр сущности.

Таким образом, необходимо:

- разделить атрибут «Name», который находится в сущности «Client» на атрибуты: «First_name», «Second_name», «Third_name»;
- разделить атрибут «Date», который принадлежит сущности «Order» на атрибуты: «Begin» и «End».

Условие второй нормальной формы, гласит, что переменная отношения находится во второй нормальной форме тогда и только тогда, когда она

находится в первой нормальной форме и каждый не ключевой атрибут неприводимо зависит от её потенциального ключа, то есть каждый столбец, не являющийся ключом, зависит от первичного ключа.

На рисунке 2.5 изображена логическая модель после процесса нормализации.

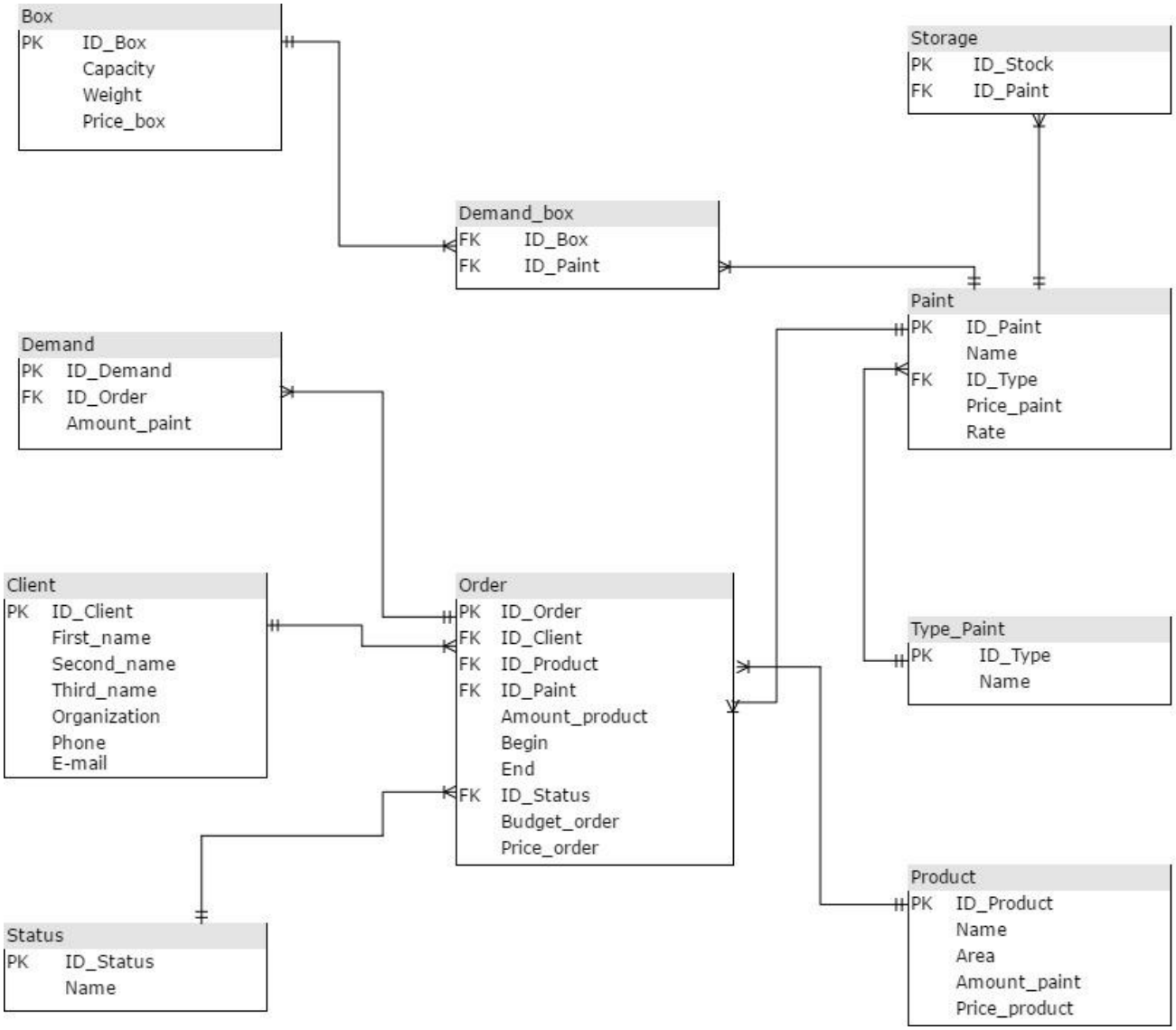


Рисунок 2.5 – Логическая модель данных

«Client» имеет первичный ключ – «ID_Client». Сущность «Client» имеет связь с «Order», благодаря первичному ключу «ID_Client» и вторичному «ID_Client (FK)», мощность связи один ко многим.

«Product» имеет первичный ключ - «ID_Product». Сущность «Product» имеет связь с сущностью «Order» благодаря первичному ключу «ID_Product» и вторичному «ID_Product (FK)», мощность связи один ко многим.

«Paint» имеет первичный ключ - «ID_Paint». Сущность «Paint», имеет связь с сущностью «Order», благодаря первичному ключу «ID_Paint» и вторичному «ID_Paint (FK)», мощность связи один ко многим. Также данная сущность имеет связь с сущностью «Storage» и «Demand_box» с помощью первичного ключа «ID_Paint» и вторичного ключа «ID_Paint (FK)», мощность связи один ко многим.

«Order» имеет первичный ключ - «ID_Order». Сущность «Order» имеет связь с сущностью «Demand» благодаря первичному ключу «ID_Order» и вторичному «ID_Order (FK)», мощность связи один ко многим.

«Status» имеет первичный ключ – «ID_Status». Сущность «Status» имеет связь с сущностью «Order» благодаря первичному ключу «ID_Status» и вторичному «ID_Status(FK)», мощность связи один ко многим.

«Type_Paint» имеет первичный ключ - «ID_Type». Сущность «Type_Paint» имеет связь с сущностью «Paint» благодаря первичному ключу «ID_Type» и вторичному «ID_Type (FK)», мощность связи один ко многим.

«Box» имеет связь с сущностью «Demand_box» с помощью первичного ключа «ID_Box» и вторичного ключа «ID_Box (FK)», мощность связи один ко многим.

Таким образом, была построена логическая модель данных, выполнен процесс нормализации двух уровней, а также была осуществлена проверка на логические ошибки.

2.4.3 Обоснование выбора системы управления базам данных программного обеспечения

Для создания и управления БД используются системы управления базами данных (СУБД), которые являются совокупностью языковых и программных

средств, предназначенных для создания, ведения и совместного использования БД многими пользователями [18].

Современные СУБД позволяют улучшить и ускорить процесс разработки программного обеспечения, благодаря доступному пользовательскому интерфейсу, с помощью которого можно добавлять новые записи, выводить имеющиеся и изменять их, а также позволяет выполнять поиск в БД. Кроме этого СУБД имеет как набор средств для поддержки таблиц и отношений между ними, так и средства программирования, применяемые для разработки собственных приложений.

Таким образом, необходимо сделать правильный выбор СУБД. Для того чтобы не допустить ошибки при выборе следует провести сравнительный анализ, который представлен в таблице 2.7.

Данный анализ был произведен на основе результатов исследований и статистики, представленных в источнике [17].

Таблица 2.7 – Сравнительный анализ СУБД

СУБД	SQLite	MySQL	PostgreSQL
Быстродействие	8	10	7
Безопасность	7	8	9
Лицензия	10	10	10
Открытость кода	10	10	10
Надежность	7	7	9
Ресурсы	9	8	7
Переносимость	10	8	7
Опыт работы	4	8	4
Итог	65	69	63

По итогам проведения сравнительного анализа имеем такой результат: SQLite – 65 баллов, MySQL – 69 баллов и PostgreSQL – 63 балла.

В нашем случаи основными критериями являлись: быстродействие и опыт работы, так как для базы данных, которая используется в приложении, необходима большая скорость передачи данных, так как ранее было принято решение о принятии архитектуры «файл-сервер». В данном случаи СУБД MySQL работает быстрее, чем другие представленные аналоги, к тому же

присутствует значительный опыт работы в ней, что ускорит время её разработки, а также позволит избежать некоторого ряда ошибок. Именно поэтому было принято решение выбрать MySQL, хотя PostgreSQL во многих моментах имеет преимущества.

2.5 Физическое моделирование программного обеспечения

На данном этапе было реализована физическая модель данных БД, которая изображена на рисунке 2.6.

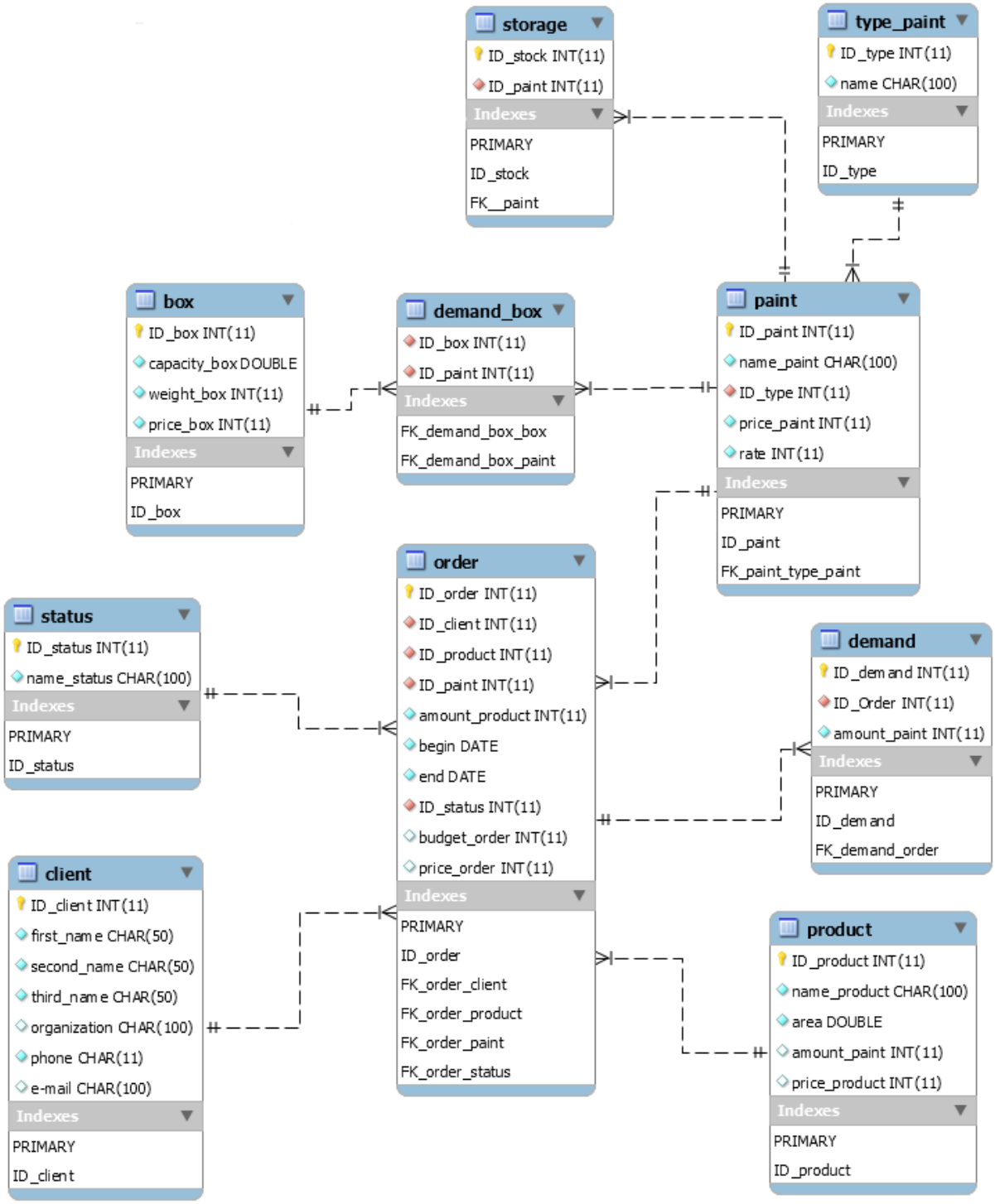


Рисунок 2.6 – Физическая модель данных программного обеспечения

Физическое проектирование базы данных – процесс подготовки описания реализации базы данных на вторичных запоминающих устройствах; на этом этапе рассматриваются основные отношения, организация файлов и индексов, предназначенных для обеспечения эффективного доступа к данным, а также все связанные с этим ограничения целостности и средства защиты [11] [32].

На модели, представленной на рисунке 2.5, представлено 10 таблиц: «Client», «Order», «Product», «Paint», «Storage», «Demand», «Status», «Type_paint», «Box», «Demand_box».

Абсолютно каждый атрибут, который определен в той или иной таблице, имеет конкретный тип данных и способен обладать некоторыми свойствами.

Связь между таблицами осуществляется с помощью первичного ключа.

Физическое проектирование является третьим и последним этапом создания проекта базы данных. Теперь необходимо приступить к самой реализации данной базы данных.

2.6 Схемы реализации программного обеспечения

И, напоследок, стоит рассмотреть структуру взаимодействия компонентов разрабатываемого программного обеспечения, для этого необходимо построить диаграмму компонентов.

Диаграмма компонентов – статическая структурная диаграмма, которая наглядно демонстрирует разбиение проектируемой системы на структурные компоненты и зависимости между этими компонентами [29].

На рисунке 2.6 изображена диаграмма компонентов программного обеспечения.

Данная диаграмма включает в себя компонент «Main», который является управляющим для программного приложения.

Компоненты: «Module_Order», «Module_Calculation», «Module_Rect» и «Module_Report» связаны с файлом «Main» с помощью связи «зависимость». Данные компоненты отвечают за интерфейс этих модулей.

Компонент «Module_Order» содержит в себя компоненты «DataBase_Order» и «DataBase_Client», которые являются базами данных, содержащих информацией о заказах и клиентах соответственно. Данный компонент отвечает за формирование заказа.

Компонент «Module_Rect» включает в себя идентичные компоненты, что и компонент «Module_Order». Данный компонент отвечает за формирование платежной квитанции.

Компонент «Module_Report» включает в себя компоненты: «DataBase_Order» и «DataBase_Storage». Данный модуль отвечает за формирование различной отчетной документации. Также «DataBase_Storage» является базой данных складских запасов.

Компонент «Module_Calculation» включает в себя компоненты «Database_Order» и «DataBase_Demond». Данный модуль отвечает за расчет оптимального распределения ресурсов и формирование цены. Также «DataBase_Demond» является базой данных складских требований.

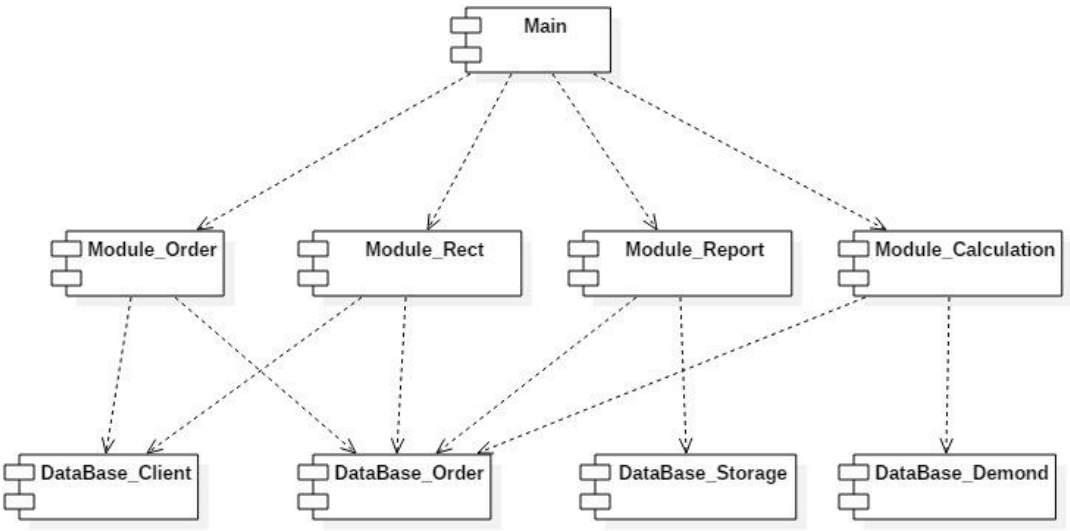


Рисунок 2.7 – Диаграмма компонентов программного обеспечения оптимального распределения ресурсов производственного предприятия

Таким образом, для наглядной демонстрации структурных компонентов программного обеспечения связей между ними была построена диаграмма компонентов. Разрабатываемое приложение состоит из главного компонента «Main» и четырех основных модулей. Данные модули имеют в своём распоряжении по несколько таблиц из базы данных.

Вывод по главе 2

Итогом данной главы является спроектированное программное обеспечение, которое позволит оптимизировать распределение производственных ресурсов организации, а также автоматизировать некоторые процессы. В частном порядке были смоделированы концептуальная, логическая и физическая модели данных. Создана диаграмма компонентов программного обеспечения, которая более подробнее отображает структурные компоненты приложения и связи между ними. Таким образом, настало время перейти к реализации программного обеспечения.

Глава 3 РЕАЛИЗАЦИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ОПТИМАЛЬНОГО РАСПРЕДЕЛЕНИЯ РЕСУРСОВ ПРОИЗВОДСТВЕННОГО ПРЕДПРИЯТИЯ

3.1 Выбор средств реализации программного приложения

Данный этап является очень важным, так как сделав правильный выбор, можно добиться не только быстрой и удобной реализации программного обеспечения, но и повысить его работоспособность. В данным случаи для сравнительного анализа были выбраны наиболее распространенные языки программирования: C++, C#, Java.

C++ – компилируемый строго типизированный язык программирования общего назначения. Поддерживает разные парадигмы программирования: процедурную, обобщённую, функциональную; наибольшее внимание уделено поддержке объектно-ориентированного программирования.

C# – язык программирования, сочетающий объектно-ориентированные и контекстно-ориентированные концепции. Является основным языком разработки приложений для платформы Microsoft .NET. Данный язык имеет строгую статическую типизацию, поддерживает полиморфизм, перегрузку операторов, указатели на функции-члены классов, атрибуты, события, свойства, исключения, комментарии в формате XML [14].

Java – сильно типизированный объектно-ориентированный язык программирования, разработанный компанией Sun Microsystems (в последующем приобретённой компанией Oracle). Приложения Java обычно транслируются в специальный байт-код, поэтому они могут работать на любой компьютерной архитектуре, с помощью виртуальной Java-машины [15] [18].

Сравнительный анализ данных языков программирования представлен в таблице 3.1. Данный анализ был выполнен благодаря информации, предоставленной из источников [15] [18] [19] [30]. В данных источниках приводятся основные характеристики языков и их сравнения в виде: статистики, исследований и опыта работы.

Таблица 3.1 – Сравнительный анализ языков программирования

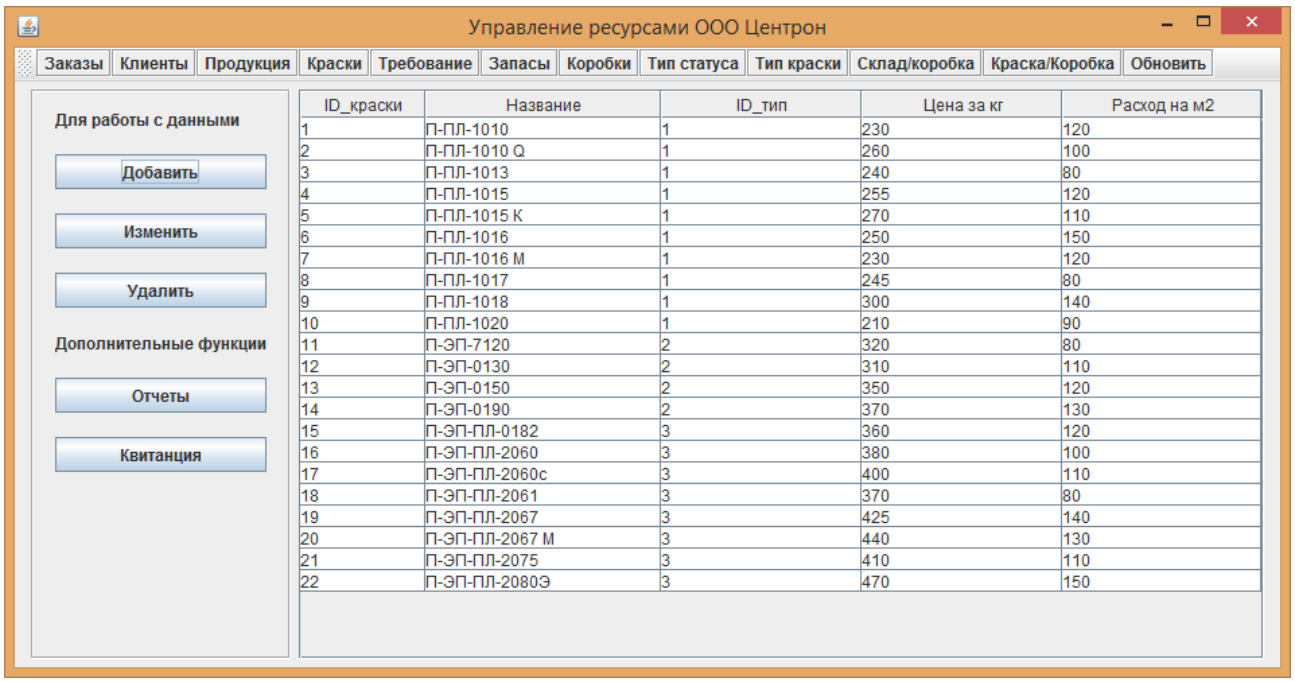
Функции и возможности	Языки программирования		
	Java	C++	C#
Скорость разработки	7	6	8
Производительность кода и требовательность к ресурсам	5	7	6
Библиотеки	6	5	6
Удобство отладки	7	5	6
Язык и синтаксис	6	5	7
Самодостаточность приложений	8	7	5
Удобство сборки	7	5	8
Опыт работы	8	6	5
Итого:	54	46	50

Таким образом, при проведении сравнительного анализа был выбран язык программирования – Java. Данный язык является удобным и достаточно простым в использовании, а также к нему предоставляется большой набор различных библиотек, которые способны решать ряд специфичных задач. Также немало важным фактором является и опыт работы, что несомненно увеличит скорость разработки и снизит количество ошибок в реализации программного кода.

3.2 Описание основных принципов работы реализуемого программного приложения

Основная задача реализуемого программного обеспечения – это оптимизация процесса распределения ресурсов производственного предприятия, возможность оформления заказа и создание отчетной документации и платежной квитанции.

При запуске пользователем приложения перед ним всплывает окно подключения к БД. В данном окне необходимо заполнить три поля: «База данных», «Пользователь» и «Пароль», если данные были введены корректно, то приложение успешно запуститься и откроется основное окно, которое изображено на рисунке 3.1. Если же данные были введены неправильно, то всплывет окно, оповещающее пользователя об ошибке при подключении.



ID_краски	Название	ID_тип	Цена за кг	Расход на м2
1	П-ПЛ-1010	1	230	120
2	П-ПЛ-1010 Q	1	260	100
3	П-ПЛ-1013	1	240	80
4	П-ПЛ-1015	1	255	120
5	П-ПЛ-1015 К	1	270	110
6	П-ПЛ-1016	1	250	150
7	П-ПЛ-1016 М	1	230	120
8	П-ПЛ-1017	1	245	80
9	П-ПЛ-1018	1	300	140
10	П-ПЛ-1020	1	210	90
11	П-ЭП-7120	2	320	80
12	П-ЭП-0130	2	310	110
13	П-ЭП-0150	2	350	120
14	П-ЭП-0190	2	370	130
15	П-ЭП-ПЛ-0182	3	360	120
16	П-ЭП-ПЛ-2060	3	380	100
17	П-ЭП-ПЛ-2060с	3	400	110
18	П-ЭП-ПЛ-2061	3	370	80
19	П-ЭП-ПЛ-2067	3	425	140
20	П-ЭП-ПЛ-2067 М	3	440	130
21	П-ЭП-ПЛ-2075	3	410	110
22	П-ЭП-ПЛ-2080Э	3	470	150

Рисунок 3.1 – Главное окно приложения

Как видно из рисунка 3.1, сверху окна представлена панель кнопок, названия которых соответствуют таблицам подключенной БД. Например, при нажатии на кнопку «Заказы» в основной области окна отобразиться таблица, которая содержит всю необходимую информацию о заказе. Аналогично работают и все остальные кнопки на панели, кроме кнопки «Обновить». Данная кнопка необходима для проверки статуса заказа.

В левой части окна представлена дополнительная панель, на которой размещены основные функциональные кнопки, которые позволяют как работать с данными в таблице, так и вывести необходимые отчеты или создать платежную квитанцию.

Кнопки «Изменить» и «Удалить» работают для всех таблиц одинаково, в отличии от кнопки «Добавить», функциональные возможности которой зависят от того какая таблица отображается в основной части окна. Окна кнопки «Изменить» представлено на рисунке 3.2.

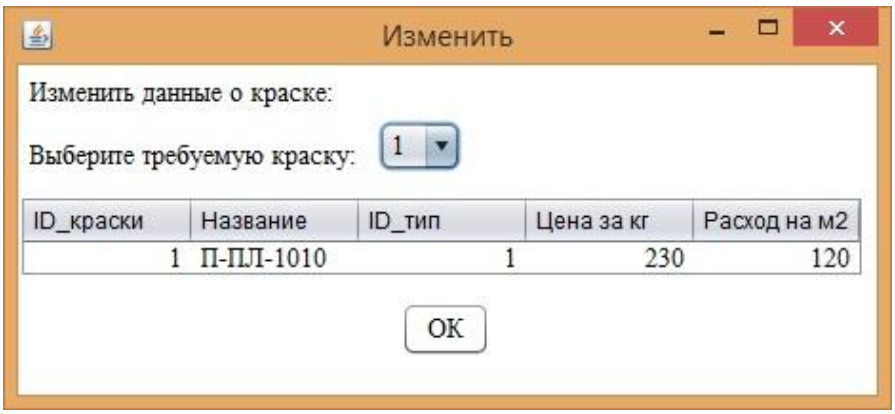


Рисунок 3.2 – Окно кнопки «Изменить» на примере таблицы «Краски»

Как было упомянуто выше, функциональные возможности кнопки «Добавить» зависят от активной таблицы. В данном случаи можно выделить три состояния вызываемого окна при нажатии на кнопку «Добавить», которые, кроме стандартного вида, внешний вид которого идентичен кнопки «Изменить», изображены на рисунках 3.3, 3.4 соответственно.

Рисунок 3.3 – Окно кнопки «Добавить» при нажатии с активной таблицей «Заказы»

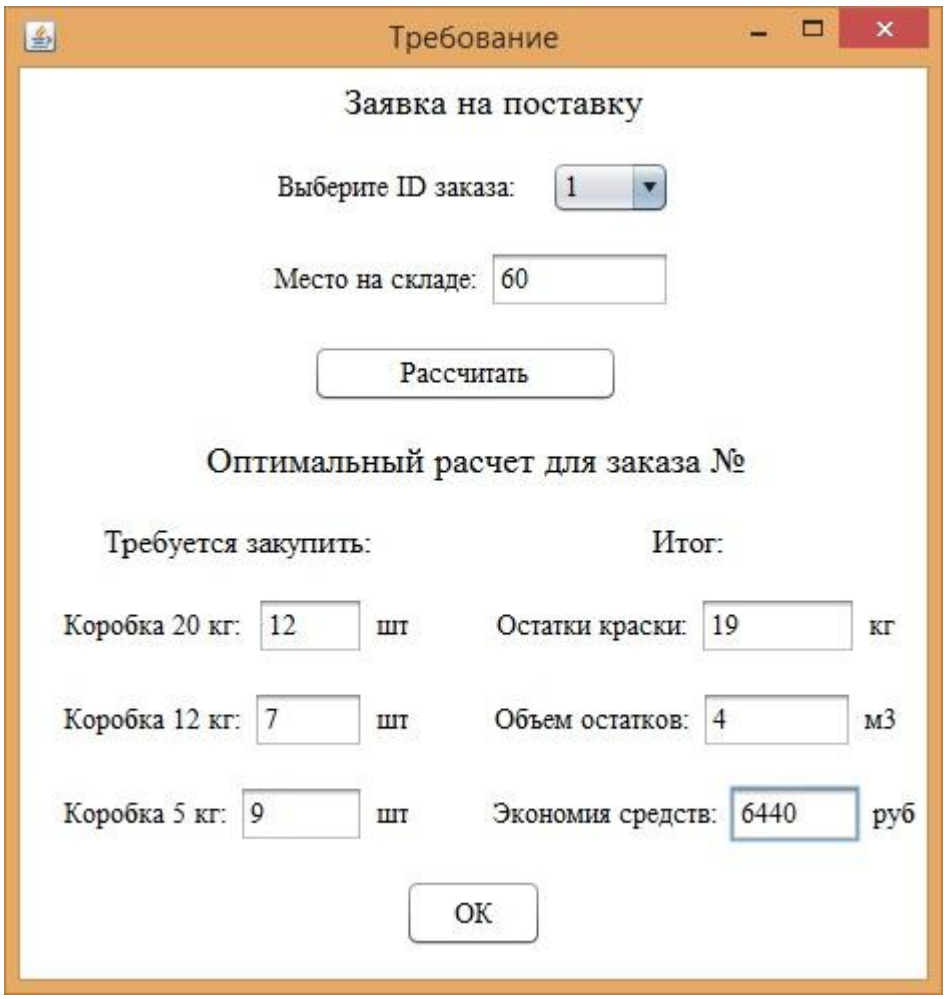


Рисунок 3.4 – Окно кнопки «Добавить» при нажатии с активной таблицей «Требования»

Стоит подробнее рассмотреть окно, изображенное на рисунке 3.4, так как кнопка «Рассчитать» отвечает за оптимизацию при составлении требования на закупку краски, а также позволяет минимизировать количество складских запасов. Программный код данной кнопки представлен в приложении Б.

Кнопка «Отчеты» позволяет отобразить в виде графиков основные результаты деятельности предприятия. На данном этапе приложение поддерживает такие графики как: «Затраты», «Закупки» и «Выполнение заказов за полгода (за год)». При реализации данной функции применялась внешняя библиотека XChart, которая позволяет создавать графики. Программный код формирования графиков представлен в приложении Б. График «Выполнение заказов за полгода (за год) представлен на рисунке 3.5.

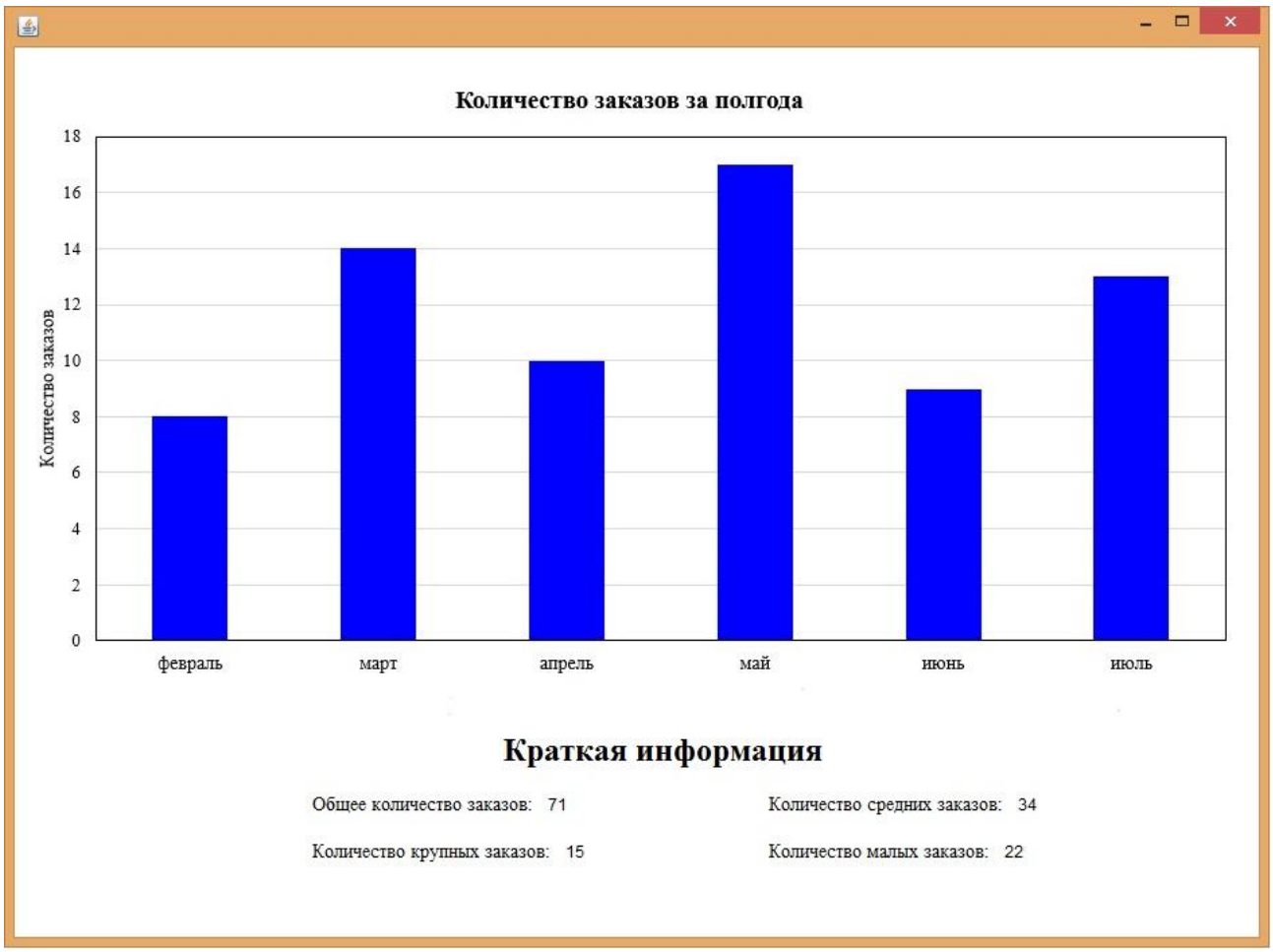


Рисунок 3.5 – График «Выполнение заказов за полгода (за год)

Кнопка «Квитанция» необходима для автоматического формирования платежной квитанции. При нажатии отображается диалоговое окно, в котором необходимо выбрать заказ, после нажать на кнопку «ОК». Далее будет сформирован Excel-файл, изображение которого представлено на рисунке 3.6.

Для того чтобы создать Excel-файл необходимо воспользоваться Apache POI, который является межплатформенным API для Java, предоставляющих доступ к чтению файлов и записи в них в форматах офисных приложений. Программный код данной функции представлен в приложении Б.

ООО "Центрон"

ПОЛУЧЕНО

[Адрес] Офицерская ул., 78, Тольятти, Самарская обл.
[Город] Тольятти
Тел.: 8 (848) 261-67-61

СЧЕТ #		ДАТА
21		10.12.2016
КОД ЗАКАЗЧИКА		УСЛОВИЯ
13		В течении 30 дней

ПОСТАВЩИК	ПОКУПАТЕЛЬ
[ИМЯ] Грачев А.А.	[ИМЯ] Рябцев В.А.
[ОРГАНИЗАЦИЯ] Центрон	[ОРГАНИЗАЦИЯ] ИП Рябцев
[ПОЧТОВЫЙ АДРЕС] Офицерская ул., 78, Тольятти, Самарская обл.	[ТЕЛЕФОН] 89270238073
[ГОРОД] Тольятти	
[ТЕЛЕФОН] 8 (848) 261-67-61	

ОПИСАНИЕ	К-ВО	ЦЕНА	СТОИМОСТЬ
Секция забора	200	1 000,00	200 000,00
Декоративные ворота	10	15 000,00	150 000,00
БЛАГОДАРИМ ЗА СОТРУДНИЧЕСТВО!	**ИТОГО:**		**350 000,00**

Рисунок 3.6 – Платежная квитанция

Таким образом, было реализовано программное обеспечения оптимального распределения ресурсов производственного предприятия, которое имеет различные функциональные возможности, которые были подробно описаны с помощью созданных экранных форм и фрагментов программного кода, которые отвечают, как за создание графического пользовательского интерфейса, так и за функциональные возможности приложения.

3.3 Оценка и обоснование экономической эффективности разработанного программного приложения

На данном этапе известно, что раньше заместителю директора на оформление заказа и процесс распределения требуемых ресурсов для его

реализации требовалось примерно 120 минут. После внедрения разработанного программного приложение данное время снизилось до 20 минут.

Также известно, что при выполнении 5 крупных заказов остатки бюджета на краску составляют около 14 тысяч рублей. После внедрения созданного программного обеспечения остатки бюджета на краску составляют чуть больше 31 тысяч рублей.

Стоит упомянуть и о том объеме краски, который остается не используемым после выполнения 5 производственных заказов. Так до внедрения разработанного программного приложения на склад отправлялось краски объемом около 2,5 м^3. Теперь же данное значение равно 0,875 м^3.

Для того чтобы рассчитать значение прямой эффективности от использования созданного программного приложения необходимо рассчитать показатели трудовых затрат, экономические показатели при расходовании бюджета и показатели, которые отвечают за неиспользуемый объем краски после выполнения заказа.

Абсолютное снижение трудовых затрат, рассчитывается по формуле:

$$_{\Delta}T = T_0 - T_1 = 600\text{-}100 = 500(\text{мин}) \quad (15)$$

где T_0 – время, необходимое на выполнение операций до внедрения программного приложения;

T_1 – время, требуемое на выполнение операций после внедрения (в расчете применяется время необходимое для оформления и процесса распределения производственных ресурсов пяти заказов).

Коэффициент относительного снижения трудовых затрат КТ (в процентах) рассчитывается по следующей формуле:

$$КТ = (_{\Delta}T / T_0) * 100\% = (500/600)*100\%=83{,}33\% \quad (16)$$

Индекс снижения трудовых затрат рассчитывается по формуле:

$$Y_T = T_0 / T_1=600/100=6 \quad (17)$$

К экономически показателям относятся абсолютное повышение, коэффициент относительного повышения и индекс повышения сэкономленных денежных средств, выделенных на бюджет.

Абсолютное повышение сэкономленных денежных средств, выделенных на бюджет, рассчитывается по формуле:

$$C = C_1 – C_0 = 31000\text{-}14000 = 17000 \quad (18)$$

где C_0 является остатком денежных средств, выделенных на бюджет до внедрения программного обеспечения;

C_1 – является остатком денежных средств, выделенных на бюджет при использовании разработанного приложения (при расчете пяти крупных заказов, поступивших от клиентов).

Коэффициент относительного повышения сэкономленных денежных средств, выделенных на бюджет, рассчитывается по формуле:

$$K_Э = (C_0 / C) * 100\% = 14000/17000*100\% = 82,35\% \quad (19)$$

Индекс повышения сэкономленных денежных средств, выделенных на бюджет, рассчитывается по формуле:

$$Y_Э = C_1 / C_0 = 31000/14000 = 2,21 \quad (20)$$

Коэффициент $K_Э$ и $Y_Э$ характеризуют улучшение процесса использования ресурсов при их распределении за счет внедрения более экономичного варианта проектного решения.

К показателям, отвечающих за неиспользуемый объем краски после выполнения заказа, относятся: абсолютное понижение, коэффициент относительного понижения и индекс понижения.

Абсолютное понижение неиспользуемого объема краски после выполнения заказа рассчитывается по формуле:

$$V = V_0 – V_1 = 2,5\text{-}0,875 = 1,625 \quad (21)$$

где V_0 является неиспользуемым объемом краски после выполнения заказа, до внедрения программного обеспечения;

V_1 – является неиспользуемым объемом краски после выполнения заказа при использовании программного обеспечения (при расчете пяти крупных заказов, поступивших от клиентов).

Коэффициент относительного понижения неиспользуемого объема краски после выполнения заказа рассчитывается по формуле:

$$K_П = (V / V_0) * 100\%=1,625/2,5*100\%=65\% \quad (22)$$

Индекс понижения неиспользуемого объема краски после выполнения заказа рассчитывается по формуле:

$$Y_П = C_0 / C_1 = 2,5/0,875 = 2, 86 \quad (23)$$

Коэффициент $K_П$ и $Y_П$ характеризуют увеличение эффективности при управлении ресурсами предприятия благодаря внедрению более экономичного варианта проектного решения.

Далее необходимо выполнить расчет показателей экономической эффективности, представленных в таблице 3.2.

Таблица 3.2 - Показатели эффективности от внедрения программного продукта

	Затраты времени, мин				
	Базовый вариант	Проектный вариант	Абсолютное изменение	Коэффициент изменения	Индекс изменения
Трудоемкость	T_0	T_1	$T=T_0-T_1$	$K_T=T/T_0*100\%$	$Y_T=T_0/T_1$
	120	20	100	83,33%	6
	Экономия выделенного бюджета, руб				
Прибыль	C_0	C_1	$C=C_1-C_0$	$K_{Э=}C_0/C*100\%$	$Y_Э=C_1/C_0$
	14000	31000	17000	82, 35%	2, 21
	Излишки краски, $м^3$				
Объем	V_0	V_1	$V=V_0-V_1$	$K_П=V/V_0*100\%$	$Y_П=V_0/V_1$
	2,5	0,875	1,625	65%	2,86

Для более лучшего восприятия расчетов экономии трудовых и стоимостных затрат, представленных в таблице 3.2, построим диаграммы.

На рисунке 3.7 изображена диаграмма, показывающая отличия между показателями трудоемкости до внедрения программного обеспечения и после внедрения.

Данная диаграмма отображает, что время, требуемое на прием и обработку одного заказа, снижено в 6 раз благодаря внедрению программного обеспечения в деятельность заместителя директора.

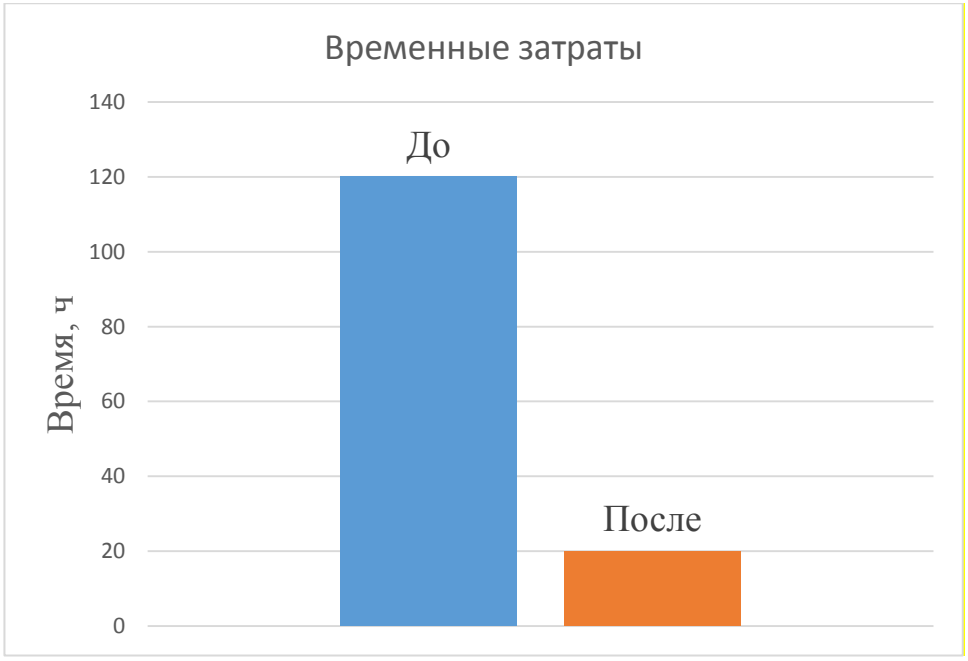


Рисунок 3.7 – Диаграмма изменения трудоемкости

На рисунке 3.8 представлена диаграмма изменения бюджет средств при закупке необходимого количества краски.

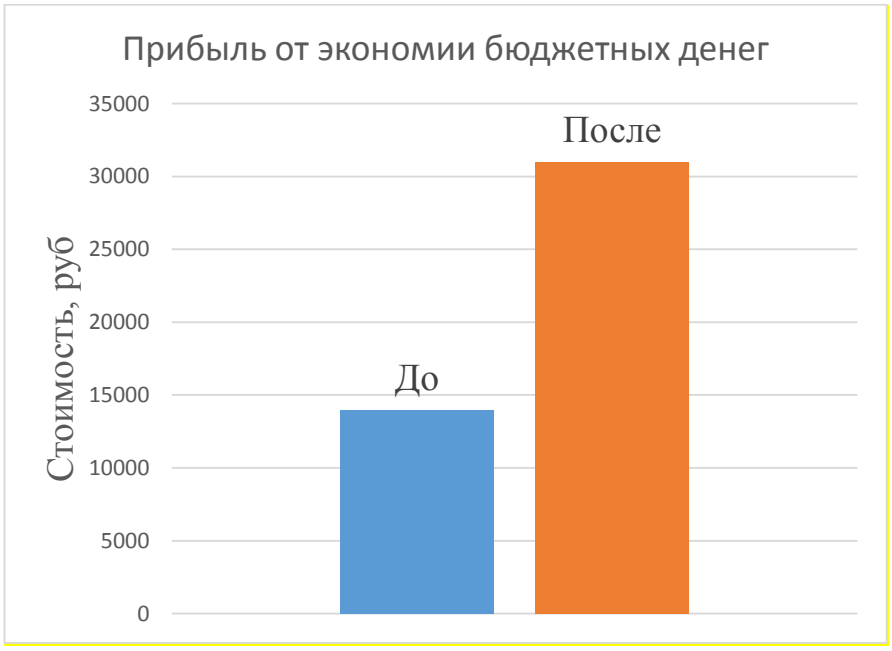


Рисунок 3.8 – Диаграмм прибыли за счет экономии бюджетных денег

На диаграмме показано, что при использовании разработанного программного обеспечения увеличивается прибыль в 2,21 раз за счет экономии денежных средств при распределении бюджетных денег на ресурсы.

На рисунке 3.9 изображена диаграмма, которая показывает объем неиспользуемой краски после завершения заказа.

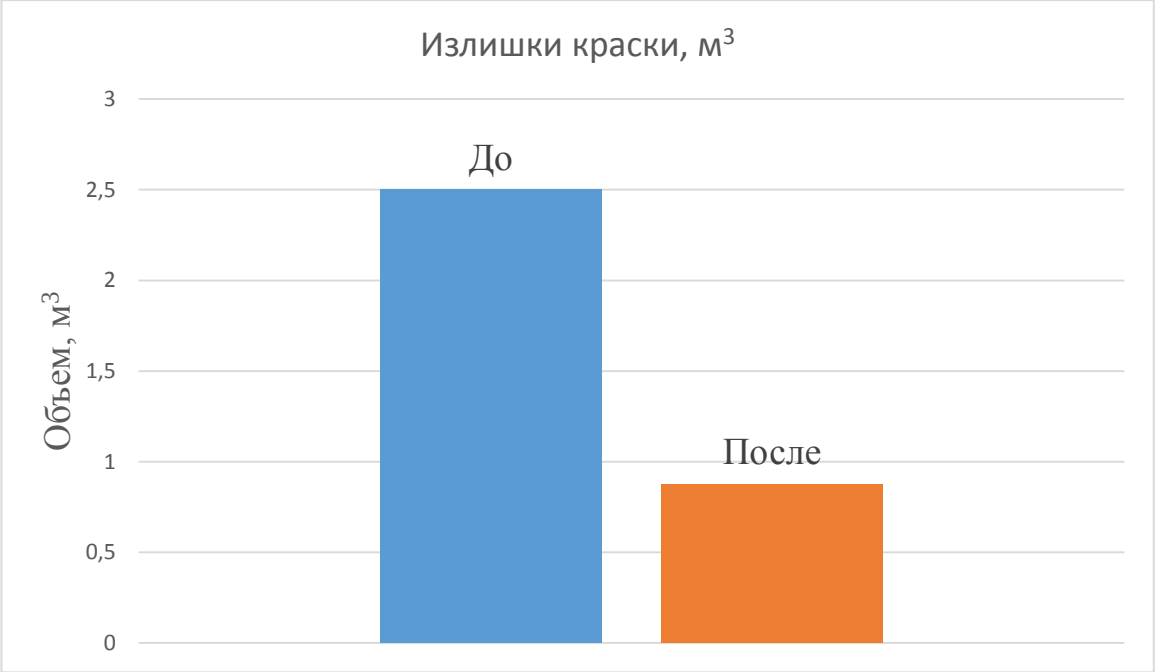


Рисунок 3.9 – Диаграмма излишек краски

В итоге, разработанное программное обеспечение для ООО «Центрон» является актуальным решением для увеличения количества обработанных заказов, экономии бюджетных средств и минимизации складских излишек.

Вывод по 3 главе

Разработанное программное обеспечение соответствует всем ранее заявленным требованиям. Данное программное обеспечение позволило увеличить скорость работы заместителя директора при формировании заказов, увеличить прибыль с помощью экономии бюджетных средств, а также минимизировать количество излишек.

Проделанный анализ экономической эффективности, выявил, что внедрение разработанного программного обеспечения, значительно повышает качество управления ресурсами.

ЗАКЛЮЧЕНИЕ

Использование мощных программных средств для оптимизирования процессов управления производственными ресурсами является большим толчком в развитии любого предприятия. Компетентное управление ресурсами предприятия позволяет улучшить финансовое положение и добиться лидирующих позиций на рынке.

В данной бакалаврской работе был выделен объект исследования, изучена предметная область, определены цели и задачи. Выполнен анализ деятельности работы заместителя директора для выявления недостатков при распределении ресурсов предприятия с целью предложить наиболее эффективный способ решения данной задачи. Для решения проблем с оптимизацией процесса распределения ресурсов был проведен анализ существующих методов оптимизации, на основании которого было решено применить метод множителей Лагранжа. Проведен анализ аналогичных программных продуктов, во время которого были выделены преимущества и недостатки существующих приложений. Благодаря проведенному анализу определены функциональные требования к разрабатываемому продукту.

Были проанализированы основные бизнес-процессы, которые требовали усовершенствования. Для того чтобы проверить корректность логического функционирования и формирования общих требований к функциональному поведению системы, были разработаны диаграммы: вариантов использования и классов с помощью графического языка объектно-ориентированного моделирования – UML. Также была сформулирована математическая модель оптимального распределения ресурсов производственного предприятия.

Средствами для реализации программного обеспечения стали язык программирования – Java и СУБД MySQL.

Итогом работы было полностью разработанное программное обеспечение оптимального распределения ресурсов производственного предприятия для ООО «Центрон» со следующими возможностями: добавление, изменение, и удаление заказов, требований на закупку и т.д.; вычисление оптимального

распределения производственных ресурсов; формирование отчетной документации и платежной квитанции.

Таким образом, реализованное программное обеспечение позволит минимизировать количество складских издержек и максимизировать прибыль при закупке необходимых ресурсов, а также позволит ускорить вычислительные операции и минимизировать количество ошибок, которые можно допустить при расчетах.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

Нормативно-правовые акты

1. ГОСТ 19.701-90. Единая система программной документации. Схемы алгоритмов, программ, данных и систем. Обозначения условные и правила выполнения (ИСО 5807-85). Введ. 1992-01-01.- М.: Изд-во стандартов, 1992. – 14с.

2. ГОСТ 34.003-90. Информационная технология. Комплекс стандартов на автоматизированные системы. Термины и определения. Взамен ГОСТ 24.003-84, ГОСТ 22487-77; введ. 1992-01-01.- М.: Изд-во стандартов, 1992. – 14с.

3. ГОСТ 34.320-96. Информационные технологии. Система стандартов по базам данных. Концепции и терминология для концептуальной схемы и информационной базы. Введ. 2001-07-01.- М.: Изд-во стандартов, 2001. – 46с. - (Основополагающие стандарты)

4. ГОСТ 34.602-89. Информационная технология. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы. – Введ. 1990-01-01.-М.: Изд-во стандартов, 1990. – 12с. - (Основополагающие стандарты)

5. ГОСТ 7.82-2001 Система стандартов по информации, библиотечному и издательскому делу. Библиографическая запись. Библиографическое описание электронных ресурсов. Общие требования и правила составления. – Введ. 2002—07—01.-М.:Изд-во стандартов, 2001. – 23с.

Научная и методическая литература

6. Аттетков А. В. Методы оптимизации. Учеб. пособие / А. В. Аттетков, В. С. Зарубин, А. Н. Канатников. - Москва : РИОР: Инфра-М, 2013. - 270 с. - (Высшее образование. Бакалавриат).

7. Васильев А. Н. Самоучитель Java с примерами и программами [учеб. пособие] / А. Н. Васильев. - 3-е изд. - Санкт-Петербург : Наука и Техника, 2016. - 368 с. - ISBN 978-5-94387-985-2.

8. Вигерс, К. Разработка требований к программному обеспечению. 3-е изд., дополнительное / К. Вигерс, Д. Битти., Пер. с англ. – М.: Издательство «Русская редакция» ; СПб.: БХВ-Петербург, 2014. – 736 стр.

9. Гагарина Л.Г., Кокорева Е.В., Виснадул Б.Д. Технология разработки программного обеспечения. – М.: ИД. «Форум»: ИНФРА-М, 2013. – 400 с.

10. Голицына, О.Л., Попов, И.И. Программирование на языках высокого уровня: учеб. Пособие для СПО. – М.: Форум, 2013. – 496 с.

11. Карпова И. П. Базы данных : курс лекций и материалы для практ. занятий : учеб. пособие для студентов техн. фак. / И. П. Карпова. - Санкт-Петербург : Питер, 2013. - 240 с. : ил. - (Учебное пособие). - Библиогр.: с. 233-234. - Прил.: с. 211-232. - Алф. указ.: с. 235-240. - ISBN 978-5-496-00546-3 : 418-60.

12. Лафоре Р. Структура данных и алгоритмы Java = Data Structures & Algorithms in Java / Р. Лафоре ; [пер. с англ. Е. Матвеев]. - 2-е изд. - Санкт-Петербург : Питер, 2013. - 701 с. : ил. - (Классика computer science). - Прил.: 678-694. - Алф. указ.: с. 695-704. - ISBN 978-5-496-00740-5. - ISBN 978-0672324536 (англ.) : 756-00.

13. Маклафлин Б. Объектно-ориентированный анализ и проектирование = Object-Oriented Analysis and Design / Б. Маклафлин, Г. Поллайс, Д. Уэст. - Санкт-Петербург : Питер, 2013. - 601 с. : ил. - (Head First). - Прил.: с. 571-601. - ISBN 978-5-496-00144-1 : 780-00.

Электронные источники

14. C# - энциклопедия языков программирования / [Электронный ресурс] – URL: http://progopedia.ru/language/csharp/, Режим доступа: свободный.

15. IBM Knowledge Center / [Электронный ресурс] – URL: https://www.ibm.com/support/knowledgecenter/ru/ssw_aix_61/com.ibm.aix.performance/advantages_java.htm, Режим доступа: свободный.

16. IDEF0 - методология функционального моделирования / Itstan [Электронный ресурс]. – URL: http://www.itstan.ru/funk-strukt-analiz/idef0-metodologija-funkcionalnogo-modelirovanija.html/, Режим доступа: свободный.

17. SQLite vs MySQL vs PostgreSQL: сравнение систем управления базами данных / [Электронный ресурс] – URL: http://devacademy.ru/posts/sqlite-vs-mysql-vs-postgresql/, Режим доступа: свободный.

18. Википедия – свободная энциклопедия / [Электронный ресурс] – URL: https://ru.wikipedia.org/wiki/, Режим доступа: свободный.

19. Выбор между C++ и C# / [Электронный ресурс] – URL: https://habrahabr.ru/post/262461/, Режим доступа: свободный.

20. Конкорд XAL: автоматизация работы предприятия / [Электронный ресурс] – URL: https://www.weekit.ru/themes/detail.php?ID=49011, Режим доступа: свободный.

21. Метод множителей Лагранжа / [Электронный ресурс] – URL: http://www.math.mrsu.ru/text/courses/method/metod_mnogitelei_lagranga.htm, Режим доступа: свободный.

22. Метод Нелдера Мида / [Электронный ресурс]. – URL: http://www.machinelearning.ru/wiki/index.php, Режим доступа: свободный.

23. Модель «сущность-связь» / [Электронный ресурс] – URL: http://www.bseu.by/it/tohod/lekcii4_3.htm, Режим доступа: свободный.

24. Обзор Microsoft Dynamics NAV / [Электронный ресурс] – URL: https://www.microsoft.com/ru-ru/dynamics365/nav-overview, Режим доступа: свободный.

25. Общая характеристика языка UML / [Электронный ресурс] – URL: http://www.informicus.ru/default.aspx?SECTION=6&id=73&subdivisionid=2, Режим доступа: свободный.

26. ООО «Центрон» / [Электронный ресурс]. – URL: http://centron63.ru/, Режим доступа: свободный.

27. Основные типы серверов / [Электронный ресурс] – URL: http://www.administrator-pro.ru/articles/osnovniye_tipiy_serverov.html, Режим доступа: свободный.

28. Отечественная, российская ERP система Галактика / [Электронный ресурс] – URL: https://www.galaktika.ru/erp/, Режим доступа: свободный.

29. Сайт национальный открытый университет ИНТУИТ/ [Электронный ресурс]. – URL: http://intuit.ru, Режим доступа: свободный.

30. Сравнительный анализ языков C# и C++ / [Электронный ресурс] – URL:https://habrahabr.ru/post/273331/, Режим доступа: свободный.

31. Требования к системе: классификации FURPS+ / [Электронный ресурс] – URL: https://sysana.wordpress.com/2010/09/16/furps/, Режим доступа: свободный.

32. Физическое проектирование базы данных / [Электронный ресурс] – URL: http://bourabai.ru/dbt/dbms/03.htm, Режим доступа: свободный.

Литература на иностранном языке

33. L. Emiliano Sanchez, J. Andres Diaz-Pace, Alejandro Zunino, Sabine Moisan and Jean-Paul Rigault, An approach based on feature models and quality criteria for adapting component-based systems [Text] / L. Emiliano Sanchez / Journal of Software Engineering Research and Development. - Volume 3, Issue 10, 2015.

34. Md. Nasar, Prashant Johri, Udayan Chanda, A Genetic Algorithm Approach for Optimal Allocation of Software Testing Effort [Text] / Md. Nasar // International Journal of Computer Applications. - Volume 68, Issue 5, 2013 – PP. 21-25.

35. Md. Nasar, Prashant Johri, Udayan Chanda, I.J., Software Testing Resource Allocation and Release Time Problem: A Review [Text] / Md. Nasar / Modern Education and Computer Science. – Volume 6, Issue 2, 2014. – PP. 48-55.

36. Shireesh Verma, Kiran Ramineni and Ian G. Ian, 4. A Control-Oriented Coverage Metric and its Evaluation for Hardware Designs [Text] / Shireesh Verma / Journal of Computer Science. - Volume 5, Issue 4, 2010 – PP. 302-310.

37. Walid M. Aly and Mohamed S. Abuelnasr, Electronic Design Automation Using Object Oriented Electronics [Text] / Walid M. Aly // American Journal of Engineering and Applied Sciences. - Volume 3, Issue 1, 2010. – PP. 121-127.

ПРИЛОЖЕНИЕ А

Требования FURPS+ к разработке программного приложения

Таблица А1 – Основные требования к разработке ПО

ID	Требование	Статус	Полезность	Риск	Стабильность	Целевая версия
Functionality – Функциональные требования						
1	Формирования заказа	Одобренные	Критичное	Средний	Низкая	1.0.0.0
2	Обработка информации о заказе	Одобренные	Критичное	Средний	Средняя	1.0.0.0
3	Возможность производить оптимальный расчет ресурсов	Одобренные	Критичное	Высокий	Средняя	1.0.0.0
4	Формирование отчетной документации	Одобренные	Критичное	Средний	Низкая	1.0.0.0
5	Формирование платежной квитанции	Одобренные	Критичное	Средний	Низкая	1.0.0.0
Usability – Требования к удобству использования						
6	Текст не должен содержать грамматических ошибок	Одобренные	Критичное	Низкий	Низкая	1.0.0.0
7	Интерфейс приложения должен быть легким в использовании	Одобренные	Критичное	Низкий	Низкая	1.0.0.0
8	Читаемость текста	Одобренные	Критичное	Низкий	Низкая	1.0.0.0

Продолжение таблицы А1

9	Навигация приложения должна быть простой в использовании	Одобренные	Критичное	Низкий	Низкая	1.0.0.0
Reliability – Требования к надежности						
10	Доступ при включении приложения	Одобренные	Критичное	Низкий	Низкая	1.0.0.0
11	Пользователь имеет возможность добавлять, изменять и удалять записи из базы данных только с помощью встроенных форм	Одобренные	Критичное	Средний	Низкая	1.0.0.0
Performance – Требования к производительности						
12	Время открытия приложения 3 секунд	Одобренные	Критичное	Средний	Средняя	1.0.0.0
13	Время необходимое на расчет оптимального количества ресурсов 4 секунды	Одобренные	Критичное	Средний	Средняя	1.0.0.0
Supportability – Требования к поддержке						
14	Время восстановления работоспособности не более 30 минут	Предложенные	Критичное	Средний	Средний	1.0.0.0

Продолжение таблицы А1

Ограничения проектирования						
15	Управление БД производиться через СУБД MySQL	Одобренные	Критичное	Средний	Средняя	1.0.0.0
Ограничения реализации						
16	Модули и функции должны быть реализованы на языке Java	Одобренные	Критичное	Средний	Низкая	1.0.0.0

ПРИЛОЖЕНИЕ Б

Программный код приложения

Программный код соединения с БД

```
package mysqloperation;
import java.sql.*;
public class MyDBConnection {
    private Connection myConnection;
    public MyDBConnection() {
    }
    public void init(){
       try{
        Class.forName("com.mysql.jdbc.Driver");
        myConnection=DriverManager.getConnection(
                "jdbc:mysql://localhost/centron_db",   "root",
"root");
        }
        catch(Exception e){
            System.out.println("Failed to get connection");
            e.printStackTrace();
        }
    }

    public Connection getMyConnection(){
        return myConnection;
    }
    public void close(ResultSet rs){
        if(rs !=null){
            try{
               rs.close();
            }
            catch(Exception e){}
        }
    }
    public void close(Statement stmt){
        if(stmt !=null){
            try{
               stmt.close();
            }
            catch(Exception e){}
        }
    }
    public void destroy(){
    if(myConnection !=null){
         try{
               myConnection.close();
            }
            catch(Exception e){}
    }
  }
}
```

Программный код главного окна

```
package mysqloperation;
import java.util.ArrayList;
import mysqloperation.CarTableModel;
import mysqloperation.MyDBConnection;
import java.sql.*;
public class InsertTable extends JFrame {
    public InsertTable() throws  Exception{
        mdbc=new MyDBConnection();
        mdbc.init();
        Connection conn=mdbc.getMyConnection();
        stmt= conn.createStatement();

        initComponents();
    }
    private void initComponents() {
        dataPanel = new JPanel();
        idLabel = new JLabel();
        idField = new JTextField();
        nameLabel = new JLabel();
        nameField = new JTextField();
        vendorLabel = new JLabel();
        vendorField = new JTextField();
        typeLabel = new JLabel();
        typeCombo = new JTextField();
        tablePane = new JScrollPane();
        carTable = new JTable();
        sendPanel = new JPanel();
        sendButton = new JButton();
        commentLabel = new JLabel();
        RateLabel = new JLabel();
        RateField = new JTextField();


setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
        addWindowListener(new event.WindowAdapter() {
            public  void  windowClosing(event.WindowEvent  evt)
{
                formWindowClosing(evt);
            }
        });
        dataPanel.setLayout(new GridLayout(5, 2, 1, 2));
        idLabel.setText("ID_paint:");
        dataPanel.add(idLabel);
        dataPanel.add(idField);
        nameLabel.setText("name_paint:");
        dataPanel.add(nameLabel);
        dataPanel.add(nameField);
        vendorLabel.setText("ID_type:");
        dataPanel.add(vendorLabel);
        dataPanel.add(vendorField);
        typeLabel.setText("Price_paint:");
```

```
            dataPanel.add(typeLabel);
            dataPanel.add(typeCombo);
            RateLabel.setText("Rate:");
            dataPanel.add(RateLabel);
            dataPanel.add(RateField);
            getContentPane().add(dataPanel, BorderLayout.NORTH);
            ResultSet rs=getResultFromCars();
            carTable.setModel(new CarTableModel(rs));
            mdbc.close(rs);
            tablePane.setViewportView(carTable);
            getContentPane().add(tablePane, BorderLayout.SOUTH);
            sendPanel.setLayout(new GridLayout(2, 0));
            sendButton.setText("Send");
            sendButton.addActionListener(new
event.ActionListener() {
                public   void   actionPerformed(event.ActionEvent
evt) {
                    sendButtonActionPerformed(evt);
                }
            });
            sendPanel.add(sendButton);
            commentLabel.setText("Click button to send data");
            sendPanel.add(commentLabel);
            getContentPane().add(sendPanel, BorderLayout.CENTER);
            Dimension                screenSize                =
Toolkit.getDefaultToolkit().getScreenSize();
            setBounds((screenSize.width-400)/2,
(screenSize.height-600)/2, 400, 600);
        }//GEN-END:initComponents

        private  void  sendButtonActionPerformed(event.ActionEvent
evt) {//GEN-FIRST:event_sendButtonActionPerformed
            // TODO add your handling code here:
            String ID_paint=idField.getText();
            String name_paint=nameField.getText();
            String ID_type=vendorField.getText();
            String price_paint=typeCombo.getText();
            String rate=RateField.getText();
            String insertStr="";
            try{
                insertStr="insert    into    paint    (ID_paint,
name_paint, ID_type, price_paint, rate) values("
                        +quotate(ID_paint)+","
                        +quotate(name_paint)+","
                        +quotate(ID_type)+","
                        +quotate(price_paint)+","
                        +quotate(rate)
                        +")";
                int done=stmt.executeUpdate(insertStr);
                commentLabel.setText("1 row inserted");
                getContentPane().removeAll();
                initComponents();
            }
```

```
            catch(Exception e){
                commentLabel.setText("Error occurred in inserting
data");
                e.printStackTrace();
            }
        }//GEN-LAST:event_sendButtonActionPerformed
        private void formWindowClosing(java.awt.event.WindowEvent
evt) {//GEN-FIRST:event_formWindowClosing
            // TODO add your handling code here:
            mdbc.close(stmt);
            mdbc.destroy();
        }//GEN-LAST:event_formWindowClosing

        public static void main(String args[]) {
            java.awt.EventQueue.invokeLater(new Runnable() {
                public void run() {
                    try{
                            new InsertTable().setVisible(true);
                        }
                    catch(Exception e){
                    }
                }
            });
        }

       public ResultSet getResultFromCars() {
            ResultSet rs=null;
            try{
                rs=stmt.executeQuery("Select * from paint");

            }
            catch(SQLException e){}
            return rs;
        }
        public String quotate(String content){
            return "'"+content+"'";
        }
         private MyDBConnection mdbc;
         private Statement stmt;
        //   Variables   declaration   -   do   not   modify//GEN-
BEGIN:variables
        private JTable carTable;
        private JLabel commentLabel;
        private JPanel dataPanel;
        private JTextField idField;
        private JLabel idLabel;
        private JTextField nameField;
        private JLabel nameLabel;
        private JButton sendButton;
        private JPanel sendPanel;
        private JScrollPane tablePane;
        private JTextField typeCombo;
        private JLabel typeLabel;
```

```
        private JTextField vendorField;
        private JLabel vendorLabel;
        private JTextField RateField;
        private JLabel RateLabel;
        // End of variables declaration//GEN-END:variables
    }
```

Программный код табличной модели на примере таблицы Краски

```
    package mysqloperation;
    import javax.swing.table.AbstractTableModel;
    import java.sql.*;
    import java.util.ArrayList;
    public class PaintTableModel extends AbstractTableModel {
        private int colnum=5;
        private int rownum;
        private String[] colNames={
            "ID","Name","Type","Price","Rate"
        };
        private  ArrayList<String[]> ResultSets;
        public PaintTableModel(ResultSet rs) {
          ResultSets=new ArrayList<String[]>();
          try{
            while(rs.next()){
                  String[] row={

rs.getString("ID_paint"),rs.getString("name_paint"),
rs.getString("ID_type"),rs.getString("price_paint"),
                        rs.getString("Rate")

                };
                ResultSets.add(row);
             }
          }
          catch(Exception e){
              System.out.println("Exception in CarTableModel");
                }
        }
        @Override
        public Object getValueAt(int rowindex, int columnindex) {
           String[] row=ResultSets.get(rowindex);
           return row[columnindex];
        }
        @Override
        public int getRowCount() {
            return ResultSets.size();
        }
        @Override
        public int getColumnCount() {
            return colnum;
        }
        @Override
        public String getColumnName(int param) {
```

```
            return colNames[param];
        }
    }
```

Программный код построения графика

```
    import java.awt.BorderLayout;
    import java.awt.Dimension;
    import java.awt.Toolkit;
    import javax.swing.JFrame;
    import javax.swing.JPanel;
    import org.knowm.xchart.CategoryChart;
    import org.knowm.xchart.CategoryChartBuilder;
    import org.knowm.xchart.XChartPanel;
    import org.knowm.xchart.style.Styler;

    public class GraphsInterface extends JFrame {
        private final JPanel contentPane = new JPanel();
        private CategoryChart chart;
        public GraphsInterface(String title, double[] xDataSet,
double[] yDataSet) {
            options();
            chart.addSeries(title, xDataSet, yDataSet);
            contentPane.add(new XChartPanel(chart));
            contentPane.validate();
        }
        private void options(){
            setDefaultCloseOperation(DISPOSE_ON_CLOSE);
            Dimension                screenSize                =
Toolkit.getDefaultToolkit().getScreenSize();
            int sizeWidth = 940;
            int sizeHeight = 700;
            int locationX = (screenSize.width - sizeWidth) / 2;
            int locationY = (screenSize.height - sizeHeight) / 2;
            setBounds(locationX,       locationY,       sizeWidth,
sizeHeight);
            contentPane.setLayout(new BorderLayout(0, 0));
            setContentPane(contentPane);

            chart = new CategoryChartBuilder().
                width(800).height(600).title("Количество   заказов
за полгода").
                xAxisTitle("ID     Заказа").yAxisTitle("Количество
заказов").
                theme(Styler.ChartTheme.Matlab).build();
        }
    }
```

Программный код формирования платежной квитанции

```
    import org.apache.poi.hssf.usermodel.HSSFCellStyle;
    import org.apache.poi.hssf.usermodel.HSSFFont;
    import org.apache.poi.hssf.usermodel.HSSFSheet;
    import org.apache.poi.hssf.usermodel.HSSFWorkbook;
    import org.apache.poi.ss.usermodel.Cell;
    import org.apache.poi.ss.usermodel.Font;
```

```
    import org.apache.poi.ss.usermodel.Row;
    import ua.com.prologistic.model.DataModel;
    import java.io.File;
    import java.io.FileInputStream;
    import java.io.FileOutputStream;
    import java.io.IOException;
    import java.text.ParseException;
    import java.util.ArrayList;
    import java.util.List;
    public class ExcelWorker {
        public static void main(String[] args) throws
ParseException, IOException {
             createSimpleExcelFile();
        }
        private static void setBoldStyle() throws IOException {
            FileInputStream file = new FileInputStream(new
File("C:\\Apache POI Excel File.xls"));
            HSSFWorkbook workbook = new HSSFWorkbook(file);
            HSSFSheet sheet = workbook.getSheetAt(0);
            HSSFFont font = workbook.createFont();
            font.setBoldweight(HSSFFont.BOLDWEIGHT_BOLD);
            HSSFCellStyle style = workbook.createCellStyle();
            style.setFont(font);
            Row row = sheet.getRow(0);
            for (int i = 0; i < row.getPhysicalNumberOfCells();
i++) {
                row.getCell(i).setCellStyle(style);
            }
            try (FileOutputStream out = new FileOutputStream(new
File("C:\\Apache POI Excel File.xls"))) {
                workbook.write(out);
            } catch (IOException e) {
                e.printStackTrace();
            }
            System.out.println("Excel файл успешно создан!");
        }
        private static void setFormula() throws IOException {
            FileInputStream file = new FileInputStream(new
File("F:\\Apache POI Excel File.xls"));
            HSSFWorkbook workbook = new HSSFWorkbook(file);
            HSSFSheet sheet = workbook.getSheetAt(0);
            Row row = sheet.createRow(4);
            Cell sum = row.createCell(3);
            sum.setCellFormula("D2+D3+D4");
            HSSFFont font = workbook.createFont();
            font.setColor(Font.COLOR_RED);
            HSSFCellStyle style = workbook.createCellStyle();
            HSSFCellStyle style = workbook.createCellStyle();
            style.setFont(font);
            sum.setCellStyle(style);
            try (FileOutputStream out = new FileOutputStream(new
File("F:\\Apache POI Excel File.xls"))) {
                workbook.write(out);
```

```
            } catch (IOException e) {
                e.printStackTrace();
            }
            System.out.println("Excel файл успешно обновлен!");
        }
        private static void createSimpleExcelFile() {
            HSSFWorkbook workbook = new HSSFWorkbook();
            HSSFSheet   sheet   =   workbook.createSheet("Просто
лист");
            List<DataModel> dataList = fillData();
            int rowNum = 0;
            Row row = sheet.createRow(rowNum);
            row.createCell(0).setCellValue("Заказчик");
            row.createCell(1).setCellValue("Стоимость");
            row.createCell(2).setCellValue("Оплачено");
            row.createCell(3).setCellValue("Подтверждено");
            for (DataModel dataModel : dataList) {
                createSheetHeader(sheet, ++rowNum, dataModel);
            }
            try (FileOutputStream out = new FileOutputStream(new
File("C:\\Apache POI Excel File.xls"))) {
                workbook.write(out);
            } catch (IOException e) {
                e.printStackTrace();
            }
            System.out.println("Excel файл успешно обновлен!");
        }
        private  static  void  createSheetHeader(HSSFSheet  sheet,
int rowNum, DataModel dataModel) {
            Row row = sheet.createRow(rowNum);
            row.createCell(0).setCellValue(dataModel.getName());
          row.createCell(1).setCellValue(dataModel.getSurname());
            row.createCell(2).setCellValue(dataModel.getCity());
           row.createCell(3).setCellValue(dataModel.getSalary());
        }
        private static List<DataModel> fillData() {
            List<DataModel> dataModels = new ArrayList<>();
            dataModels.add(new   DataModel("Howard",   "Wolowitz",
"Massachusetts", 90000.0));
            dataModels.add(new  DataModel("Leonard",  "Hofstadter",
"Massachusetts", 95000.0));
            dataModels.add(new   DataModel("Sheldon",   "Cooper",
"Massachusetts", 120000.0));
            return dataModels;
        }
    }
```