

Министерство образования Республики Беларусь  
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Инженерно-экономический факультет  
Кафедра экономической информатики

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

по дисциплине:

«Системный анализ и проектирование информационных систем»

к курсовому проекту на тему:

«Программное средство для принятия решений методом полного попарного  
сравнения (на примере работы пекарни).»

Студентка

Козырева А. И.

группа 872302

Руководитель

Унучек Е. Н.

Минск 2020

## Оглавление

ВВЕДЕНИЕ.....	5
1 Описание предметной области .....	7
2 Постановка задачи на разработку программного средства .....	14
3 Построение базовой аналитической модели .....	15
4 Модели представления .....	18
5 Обоснование выбора программных средств разработки .....	22
6 Описание архитектуры разрабатываемого средства .....	24
7 Алгоритм работы программы .....	26
8 Руководство по развертыванию.....	29
9 Руководство пользователя.....	32
10 Контрольный пример .....	38
ЗАКЛЮЧЕНИЕ .....	43
Список использованных источников .....	44
ПРИЛОЖЕНИЕ А (Листинг кода) .....	45

## ВВЕДЕНИЕ

В современной жизни мы не можем существовать без посещения различных мест питания. Возможно не каждый день, но мы изрядно пользуемся теми услугами, которые нам предоставляет нынешние организации. Существует немало мест, которые мы посещаем сами или со своими друзьями и родными: кафе, ресторан, кино, театр. Люди всегда ценили вкусную пищу и с каждым столетием кулинарные шедевры достигали все больших высот.

Сейчас перед нашими глазами огромный ассортимент изысканных блюд. Все больше строится кафе и ресторанов, а также особую тенденцию набирают пекарни. Открытие любого заведения общепита начинается с выбора разновидности, к которой будет принадлежать объект. Сфера общепита отличается стремительным развитием, вследствие чего на сегодняшний день существует большое количество разновидностей заведений общепита. Мы видим увеличение количества частных предприятий, где каждый кондитер пытается донести идею своего творчества и покориť людей уникальными кулинарными шедеврами.

Пекарня представляет собой небольшое немеханизированное предприятие по выпечке и реализации хлебобулочных и кондитерских изделий, как правило, также реализующее их на месте. Типичный ассортимент пекарен составляют различные хлеб, торты, пирожные и пироги.

Некоторые пекарни также сочетают в себе функции кафе. В них имеются оборудованные для организации общественного питания залы, а ассортимент, помимо собственно хлебобулочных и кондитерских изделий, также включает чай или кофе для клиентов, желающих употребить их продукцию прямо в помещении.

Основная роль пекарни заключается в удовлетворении потребности населения в питании. На мой взгляд реализация данной роли очень важна, так как вкусное питание повышает настроение и работоспособность, что в свою очередь сказывается на эффективности деятельности предприятия. Организация вкусного питания населения во внерабочее время, осуществляемая за счет реализации готовых блюд на предприятиях через кулинарные заведения, сокращает затраты времени на приготовление пищи и способствует облегчению домашнего труда женщин. В данный момент наблюдается сокращение числа многодетных семей, что также обуславливает повышение посещаемости предприятий общественного питания.

В настоящее время вряд ли хотя бы одна организация может обойтись без документирования и компьютерных программ для учёта поступающей

информации. Так же и в работе пекарни: для упрощения работы с клиентами, специалистами, для автоматизации системы заказа на изделие и других важных действий создаются программные обеспечения и приложения, где хранится вся нужная информация, которую можно в любой момент просмотреть, отредактировать, отсортировать для удобства пользования и т.д. И, конечно, с помощью компьютера эту работу можно проделать намного быстрее и эффективнее.

Общественное питание является важным и перспективным сектором в экономике нашей страны. Заведения, оказывающие услуги по питанию, развиваясь быстрыми темпами, приносят существенный доход своим владельцам и в государственную казну, т. к. нет сезонности и среди клиентов развито сарафанное радио. А также современный маркетинг позволяет быстро сделать заведение популярным и посещаемым.

Учитывая всё вышеизложенное актуальность данного курсового проекта вызвана потребностью в новой, современной системе, способной улавливать тенденции любых изменений, приспосабливаться и оперативно реагировать на них.

Таким образом, основная цель данной курсовой работы – обеспечение лёгкого, но надёжного формата взаимодействий между клиентом и самой пекарней, обеспечение удобной формы учёта всей документации и информации, а также принятие правильного решения, основанного на критериях оценки.

Для достижения поставленной цели необходимо решить следующие задачи:

- исследовать организацию работы пекарен;
- разработать классовую иерархию для хранения данных;
- разработать бизнес-функции для удобной работы с приложением;
- разработать функциональную схему работы программы;
- разработать простой и удобный интерфейс приложения;
- предусмотреть обработку исключительных ситуаций;
- разработать систему принятия решения, основанное на оценке параметров.

## 1 ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

Организация работы любого предприятия – непростой процесс, требующий точного отслеживания времени работы учреждения и работающих специалистов.

Просмотрев прайс лист пекарни и отзывы можно понять, где уровень обслуживания намного лучше.

Для упрощения работы разрабатывается специальная программа, которая автоматически учитывает все важнейшие аспекты для осуществления работы с информацией о специалистах и покупателях данной пекарни.

Важнейшим стратегическим фактором успеха пекарни является кадровый потенциал, который в свою очередь определяет успех компании. По мере возможности руководство фирмы старается повышать уровень знаний сотрудников посредством курсов повышения квалификации, проведения семинаров с привлечением иностранных представителей, в совершенстве владеющим кулинарным мастерством.

Вдобавок услуги общепита, как и другие отрасли, активно используют инструменты интернет-маркетинга – посадочные страницы сайта, веб-аналитику, контекстную рекламу и прочее. Пекарни все охотнее обращаются к информационным технологиям: растет спрос на программное обеспечение, необходимое для эффективного управления.

Общая величина прибыли пекарни является отчетным показателем, по которому можно судить лишь о «размерах» и ее значении в экономике.

В то же время на рынке существуют целые информационные системы, контролирующие практически все бизнес-процессы пекарни: учет данных потребителей, складской учет, финансовую и управленческую отчетность. Они же предлагают анализ услуг на основе реальных данных, который затем можно использовать для увеличения среднего чека, корректировки предлагаемых услуг и прочих маркетинговых задач. По сути эти системы очень упрощают жизнь сотрудников, параллельно улучшая для покупателей пользовательский опыт общения с пекарней.

Пекарни все больше внимания уделяют сервисной составляющей. Современный клиент, приходя в кафе, по праву требует качественный сервис. А хороший сервис – это, в числе прочего, и электронные технологии: онлайн-заказ, личный кабинет покупателя, система уведомлений об акциях, оценка пекарни опытными специалистами и т.д. То есть все, что делает пекарню более конкурентоспособной, чем открывшаяся по соседству, которая работает «с бумагой».

И поскольку использование облачных технологий в бизнесе является более экономически эффективным (салон не несет лишних расходов на ИТ-инфраструктуру и ее содержание, на оплату труда дополнительного персонала – программистов, сисадминов, администраторов, аналитиков), то чем больше процессов бизнес организует из облака, тем понятнее ему ежемесячные расходы.

Информационные технологии – это полезный инструмент, который успешно применяется во множестве сфер жизни общества. Деятельность пекарни – не исключение. Прогресс в информационных технологиях положительно сказался на развитии новых направлений организации услуг населению.

Описав предметную область данного проекта, нужно сказать, что процесс учёта деятельности пекарни и принятия решения требует высокого внимания и точности. Благодаря наличию эффективного функционала и удобного интерфейса данная программа способна обеспечить простое и лёгкое взаимодействие клиента и предприятия. Однако основные функции и данные хранения информации имеют ряд особенностей и замечаний, описание которых необходимо для правильной работы. Данные критерии будут описаны в последующих пунктах.

Для данного проекта была разработана организационная структура пекарни, которая показывает иерархию подчинения в данном предприятии (рис. 1.1).

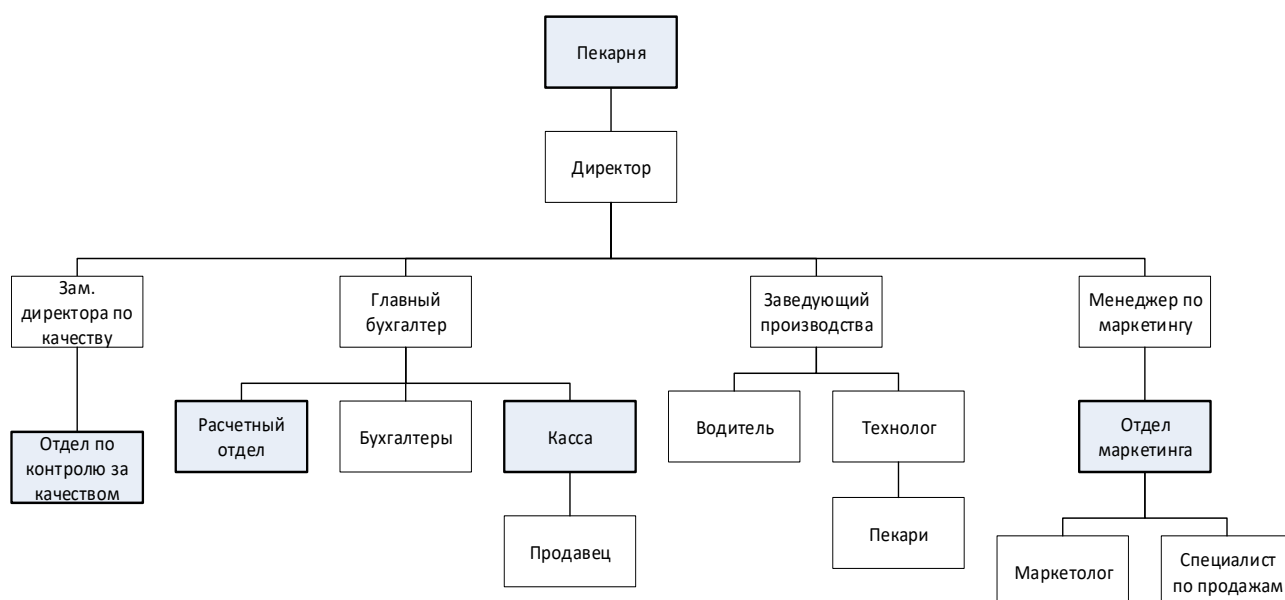


Рисунок 1.1 – Организационная структура пекарни

Далее рассмотрим стратегическую карту для пекарни. Определим для нее цели и показатели, которые направлены на увеличение прибыли и развитие самого производства (рис. 1.2).

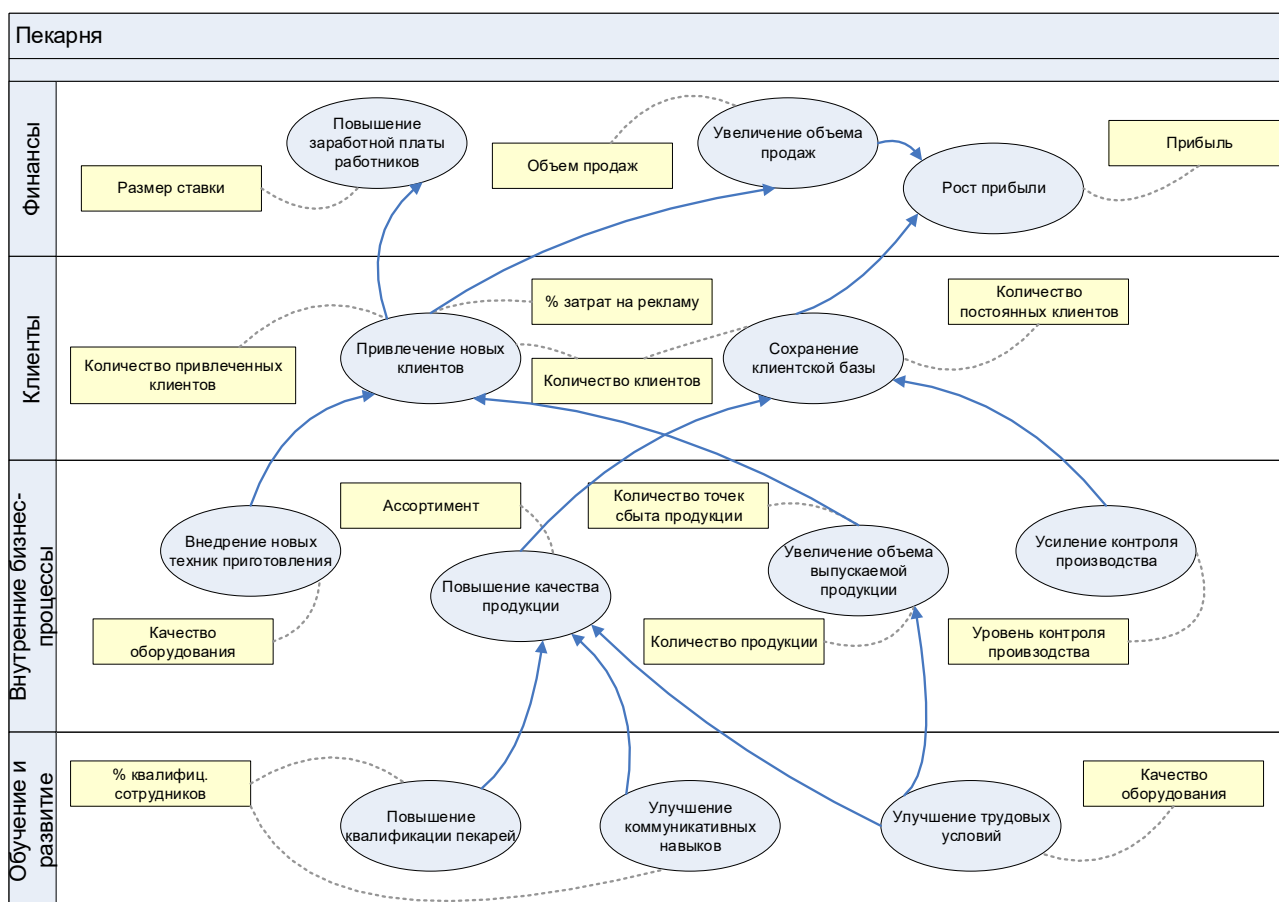


Рисунок 1.2 – Стратегическая карта пекарни

Основным процессом предметной области данного проекта является управление пекарней. Необходимо отметить, что данное действие включает в себя большое количество особенностей. Исходя из этого, существует необходимость в изложении основных пунктов и действий каждой из сторон данной операции.

Основная бизнес-функция системы – это «миссия» системы, ее значение в окружающем мире. При её определении необходимо всегда иметь ввиду цель моделирования и точку зрения на модель. Использовалась методология функционального моделирования IDEF0. Рассмотрим контекстную диаграмму, где представлены основные связи и ресурсы (рис.1.3).



Рисунок 1.3 – Контекстная диаграмма верхнего уровня

Далее представлена декомпозиция контекстной диаграммы, состоящая из четырёх блоков:

- а) Сформировать меню пекарни, основываясь на предпочтениях потребителей;
- б) Изготовить продукцию ;
- в) Проверить качество продукции;
- г) Выставить приготовленную продукцию на продажу.

Эти блоки отражают основные процессы, которые происходят на предприятии, а также определяют потоки, задействованные в процедуре (рис. 1.4).



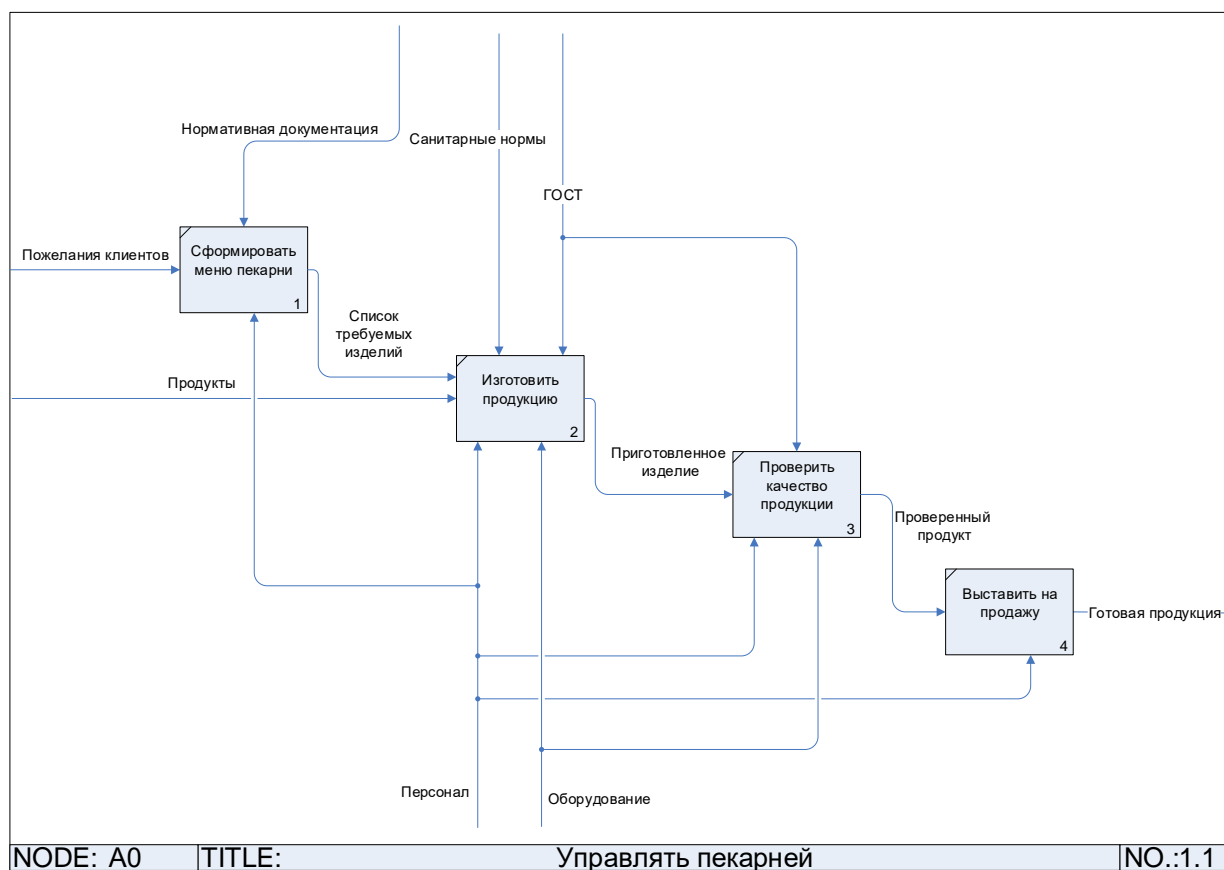


Рисунок 1.4 – Декомпозиция контекстной диаграммы верхнего уровня

Более детально рассмотрим процесс изготовления продукции, так это один из основных и наиболее важных процессов организации. Данный процесс состоит из четырех блоков:

- а) Определить перечень необходимых ингредиентов;
- б) Получить необходимые продукты;
- в) Распределить работу по процессам;
- г) Выполнить процессы по приготовлению.

Данные процессы более подробно описывают сам процесс приготовления и показывают, какие ресурсы для этого необходимы (рис. 1.5).

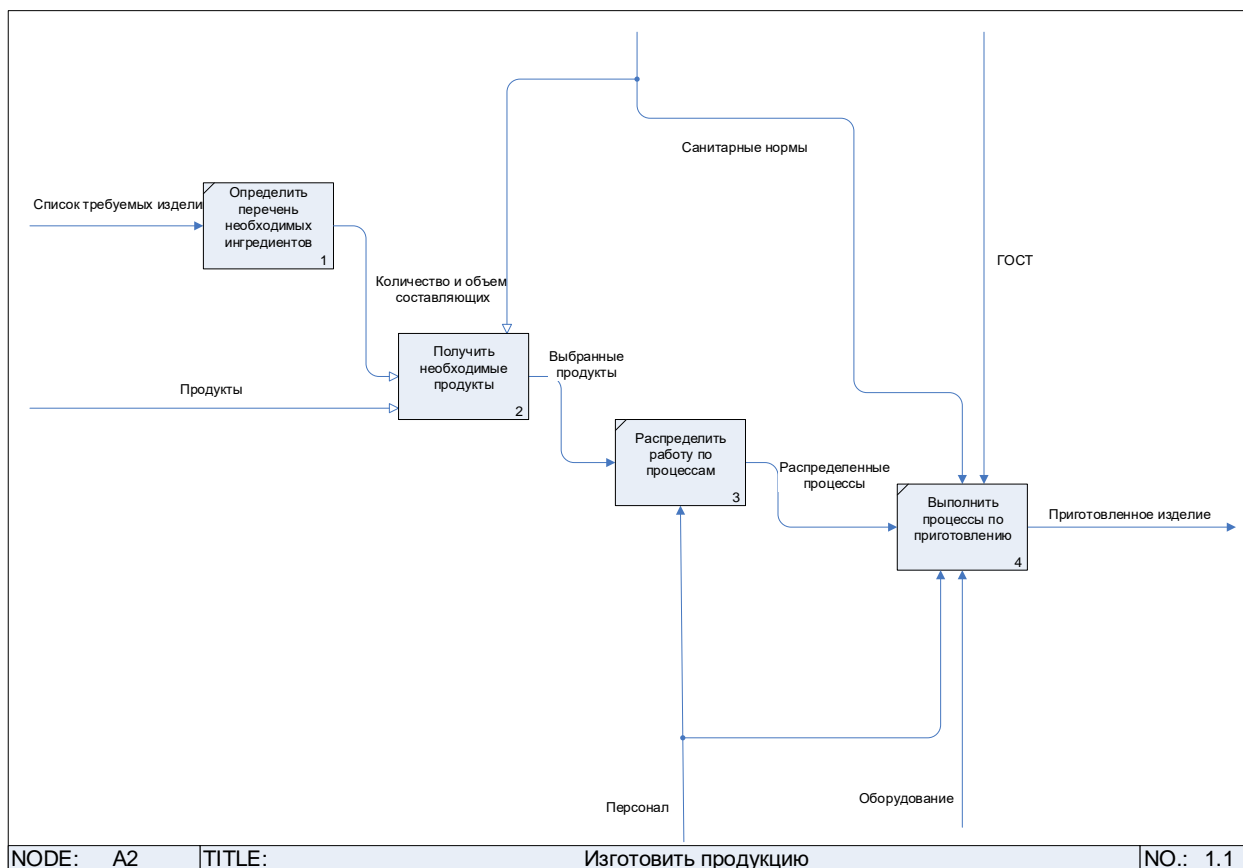


Рисунок 1.5 – Декомпозиция блока «Изготовить продукцию»

Разобьем блок «Получить необходимые продукты на составляющие»:

- а)Выбрать поставщика;
- б)Отправить запрос на поставку;
- в)Оформить договор на поставку;
- г)Получить заказанные продукты.

Здесь более подробно описан сам процесс взаимодействия с поставщиком, поскольку от правильного выбора поставщика зависит качество продуктов, а, следовательно, и качество выпускаемой продукции (рис. 1.6).

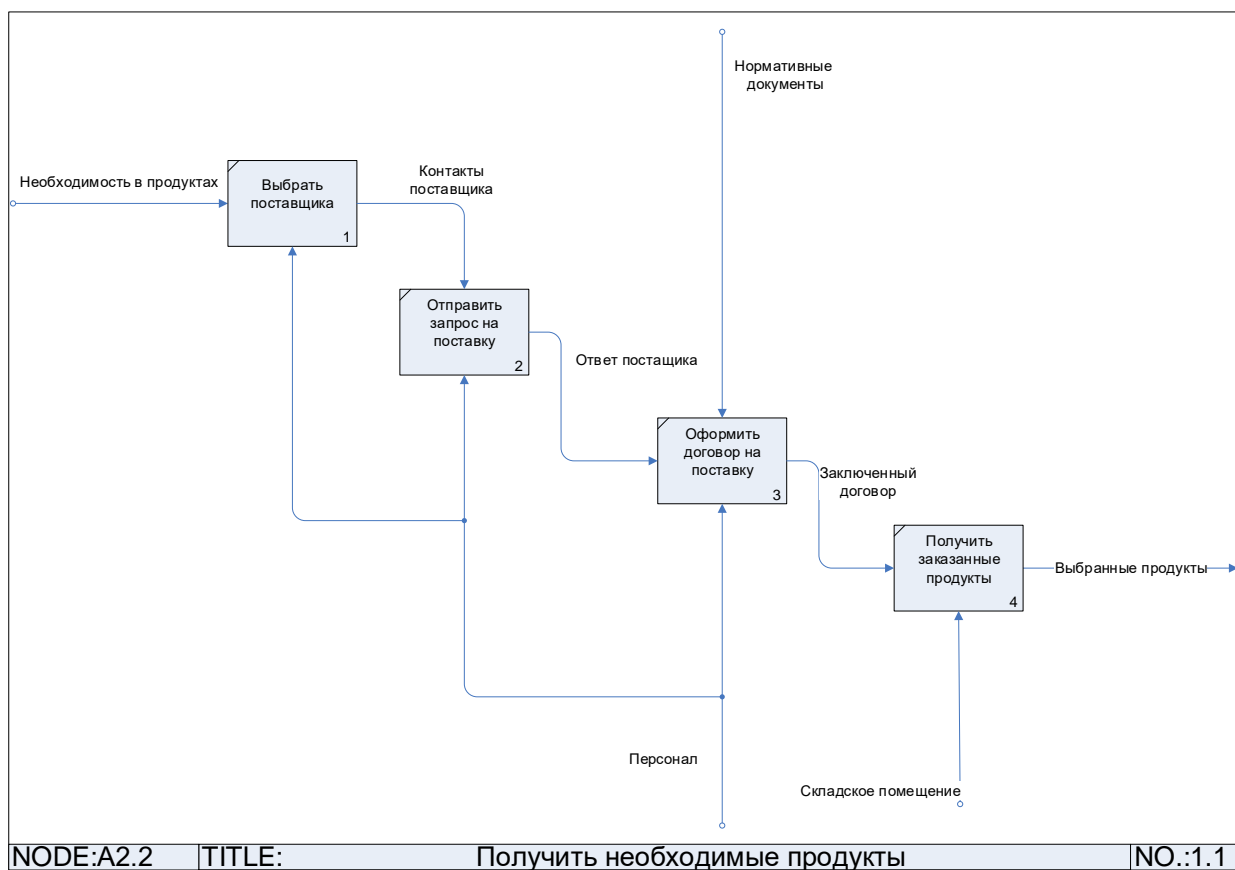


Рисунок 1.6 – Декомпозиция блока «Получить необходимые продукты»

## **2 ПОСТАНОВКА ЗАДАЧИ НА РАЗРАБОТКУ ПРОГРАММНОГО СРЕДСТВА**

В современной жизни информационные системы существенно облегчают проведение многих процессов, которые требуют больших материальных затрат, в особенности затрат времени.

Постановка задачи — точная формулировка условий задачи с описанием входной и выходной информации. В качестве входной информации здесь выступают данные самой пекарни, а также данные пользователя. На выходе мы получаем готовое решение, основанное на соответствующем методе.

Разработка данного программного средства предполагает упрощение взаимодействия клиента и пекарни. Приложение построено на связи клиент-сервер, при которой доступны широкие возможности, такие как просмотр информации, ее редактирование, удаление, добавление новых данных.

Данный программный продукт удобен и прост в использовании и обладает понятным для любого пользователя интерфейсом. Особую пользу данный проект представляет для самой пекарни, обеспечивая возможность контроля своих параметров и позволяя определить, какие дальнейшие перспективы должны быть для развития организации. Программа позволяет с легкостью вести учет на предприятии.

Для реализации данного проекта был выбран язык программирования C++. При решении поставленной задачи оптимально использовать для представления информационных материалов язык C++, который позволяет быстро и эффективно создавать приложения. Информация записывается в файл для удобного и легкого взаимодействия с данными. При разработке программного средства были учтены и обработаны исключительные ситуации. Так, при некорректном вводе информации программа выдаст ошибку и предоставит возможность для повторного ввода данных.

Готовое приложение должно представлять собой надежный и легкий в использовании продукт. Он должен быть работоспособным и осуществлять поставленные задачи по изменению данных, поиску, фильтрации, добавлению и удалению. Программа предназначена для работы с любыми данными. Приложение подразумевает собой следование основным принципам объектно-ориентированного программирования и содержит в себе все основные конструкции для правильной работы.

Таким образом, разработанный продукт значительно упрощает работу и сокращает время на взаимодействие, предоставляя покупателю удобный интерфейс.

### 3 ПОСТРОЕНИЕ БАЗОВОЙ АНАЛИТИЧЕСКОЙ МОДЕЛИ

Каждый процесс принятия решения производит окончательный выбор, который может побуждать или не побуждать действие. Принятие решений — это процесс идентификации альтернатив и выбора среди них, основанный на ценностях и предпочтениях принимающего решение.

Целью применения системного анализа к конкретной проблеме является повышение степени обоснованности принимаемого решения, расширение множества вариантов, среди которых производится выбор, с одновременным указанием способов отбрасывания тех из них, которые заведомо уступают другим.

Метод «Принятие решений» является систематической процедурой, основанной на моделях мышления, которые все мы используем в ситуации выбора. Эта модель мышления состоит из следующих элементов:

1. Принятие необходимости сделать выбор и формулировка решения (определить то, что нужно сделать)
2. Рассмотрение конкретных условий, удовлетворение которых приведет к успешному выбору
3. Решение о варианте действий, удовлетворяющих выбранным условиям
4. Рассмотрение рисков, связанных с принятием выбранного конечного решения и оценка этих рисков с точки зрения их способности повлиять на успешность и безопасность выбранных действий. Оценка возможных негативных последствий.
5. Формулировка решения

Существует множество разновидностей методик системного анализа, однако в большинстве своем они имеют ряд сходных этапов. Методика системного анализа в общем случае включает семь основных этапов:



В данной работе для принятия решений использовался метод полного попарного сравнения. При формировании экспертных оценок часто используется шкала порядка. Вопрос сравнения решается по принципу лучше или хуже, больше или меньше. Это во многом обусловлено особенностями психологии человека, который сравнивает объекты парами. Поэтому при построении шкалы порядка и ранжированного ряда экспертам следует предлагать метод попарного сравнения.

Для оценки значимости потребительских свойств и функций обычно используются метод попарного сравнения свойств и метод расстановки приоритетов.

Каждый эксперт проводит попарное сопоставление целей в прямом и обратном направлениях, формируя матрицу частот, превалирования целей друг над другом, причем общее число суждений эксперта определяется формулой  $N = n \cdot (n-1)$ . В прямом и обратном направлении, т.е. заполняем не только наддиагональную часть. Это более точный метод. В этих условиях веса целей определяются следующим образом:

1. Формируются матрицы частот (каждый эксперт заполняет свою матрицу). Смысл частот: характеризуют предпочтение одной цели перед другой.

$\Xi_j$	$Z_1$	$Z_2$	...	$Z_n$
$Z_1$	-	$f(Z_1/Z_2)_j$	...	$f(Z_1/Z_n)_j$
$Z_2$	$f(Z_2/Z_1)_j$	-	...	$f(Z_2/Z_n)_j$
...	...	...	-	...
$Z_n$	$f(Z_n/Z_1)_j$	$f(Z_n/Z_2)_j$	...	-

2. Определяются оценки предпочтений:

$$f_{kj} = \sum_{l \neq k} \left( \frac{Z_k}{Z_l} \right)_j$$

$$(j = \overline{1, m}; k = \overline{1, n})$$

Сначала задаем  $j$  и так далее.

3. Определяются нормированные оценки:

$$v_{kj} = \frac{f_{ki}}{N}$$

для всех  $(j = \overline{1, m}; k = \overline{1, n})$

4. Вычисляются искомые веса целей:

$$\omega_k = \sum_{j=1}^m v_{kj} / \sum_{k=1}^n \sum_{j=1}^m v_{kj}$$

$$(k = \overline{1, n}), \text{ где } \sum \omega_k = 1.$$

## 4 МОДЕЛИ ПРЕДСТАВЛЕНИЯ

UML представляет собой язык графического описания для объектного моделирования в области разработки программного обеспечения, моделирования бизнес-процессов, системного проектирования и отображения организационных структур.

Унифицированный язык моделирования (UML) является стандартным инструментом для создания «чертежей» программного обеспечения. С помощью UML можно визуализировать, специфицировать, конструировать и документировать артефакты программных систем. UML пригоден для моделирования любых систем: от информационных систем масштаба предприятия до распределенных Web-приложений и даже встроенных систем реального времени.

UML – это язык для визуализации, специфицирования, конструирования и документирования артефактов программных систем. Для понимания любой нетривиальной системы приходится разрабатывать большое количество взаимосвязанных моделей. В применении к программным системам это означает, что необходим язык, с помощью которого можно с различных точек зрения описать представления архитектуры системы на протяжении цикла ее разработки. UML – это язык конструирования, и модели, созданные с его помощью, могут быть непосредственно переведены на различные языки программирования. Иными словами, UML-модель можно отобразить на такие языки, как Java, C++, Visual Basic, и даже на таблицы реляционной базы данных. Можно решить важную задачу: реконструировать модель по имеющейся реализации. UML позволяет решить проблему документирования системной архитектуры и всех ее деталей, предлагает язык для формулирования требований к системе и определения тестов и, наконец, предоставляет средства для моделирования работ на этапе планирования проекта и управления версиями. Использование UML эффективно в:

- информационных системах масштаба предприятия;
- банковских и финансовых услугах;
- телекоммуникациях;
- на транспорте;
- оборонной промышленности, авиации и космонавтике;
- розничной торговле;
- медицинской электронике;

Диаграмма вариантов использования (сценариев поведения, прецедентов) является исходным концептуальным представлением системы в



процессе ее проектирования и разработки. Каждый вариант использования определяет некоторый набор действий, совершаемых системой при взаимодействии с актером. При этом в модели никак не отражается то, каким образом будет реализован этот набор действий. Диаграмма вариантов использования представлена на рисунке 4.1.

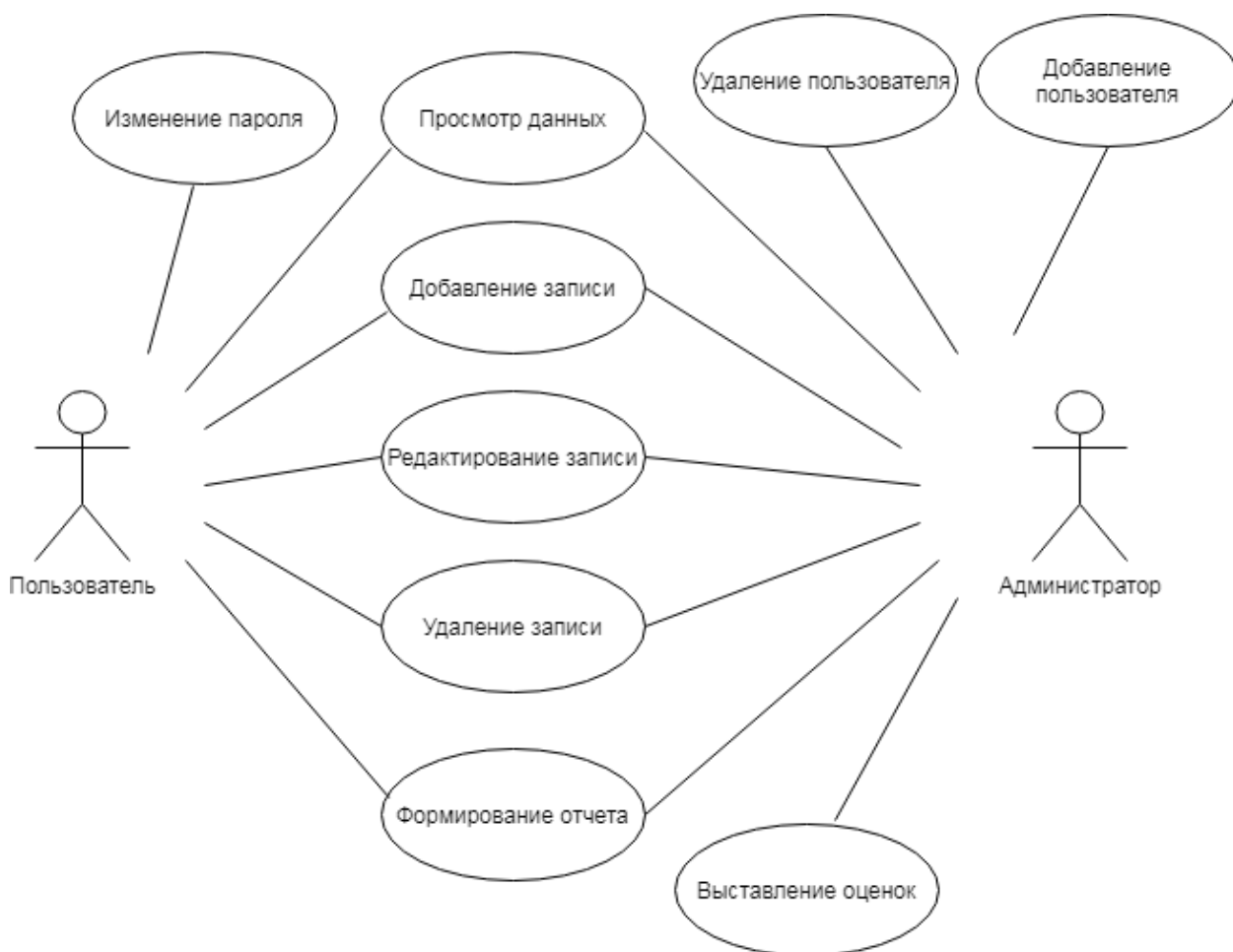


Рисунок 4.1 – Диаграмма вариантов использования

Актёрами на данной диаграмме являются пользователь и администратор. В возможности пользователя входит добавление данных, редактирование и удаление записей. Пользователь может просмотреть лучшие заведения, которые были оценены экспертами по различным критериям. Теперь рассмотрим возможности администратора, по сравнению с пользователем, у него их больше. Администратор может просматривать все учётные записи пользователей, добавлять и удалять их. Также он может добавлять, удалять и редактировать записи. Для администратора доступны функции выставления экспертных оценок, а также проверка этих оценок. Еще одной важной функцией обладает администратор – создание отчета, где

изложена требуемая информация. Одной из самых важных функций администратора является расчет результата на основании экспертных оценок. Описанный выше функционал должен быть реализован в полной мере, с учётом предусмотренных исключительных ситуаций.

Именно диаграмма классов дает нам наиболее полное и развернутое представление о структуре и связях в программном коде. Понимание принципов построения данной диаграммы позволяет кратко и прозрачно выражать свои мысли и идеи. Так диаграммы классов представлены на рисунках 4.2 и 4.3

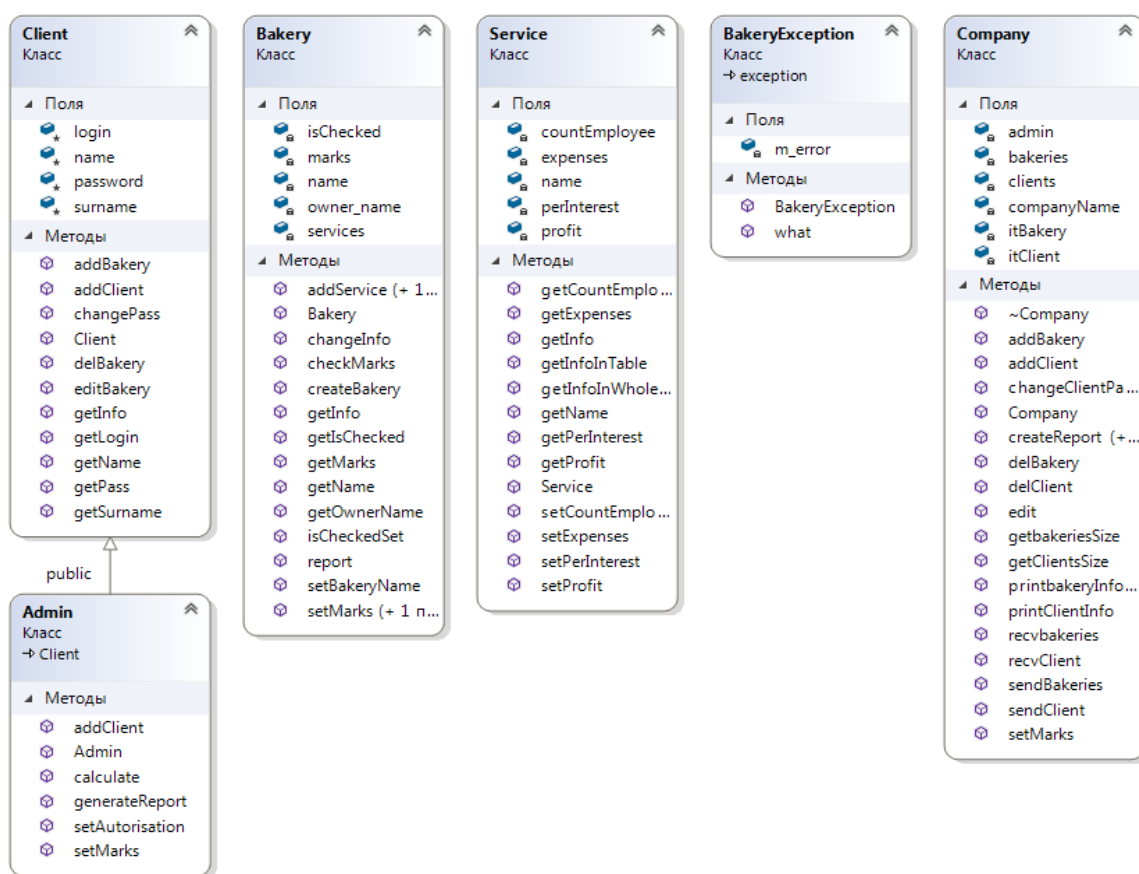


Рисунок 4.2 – Диаграмма классов клиента

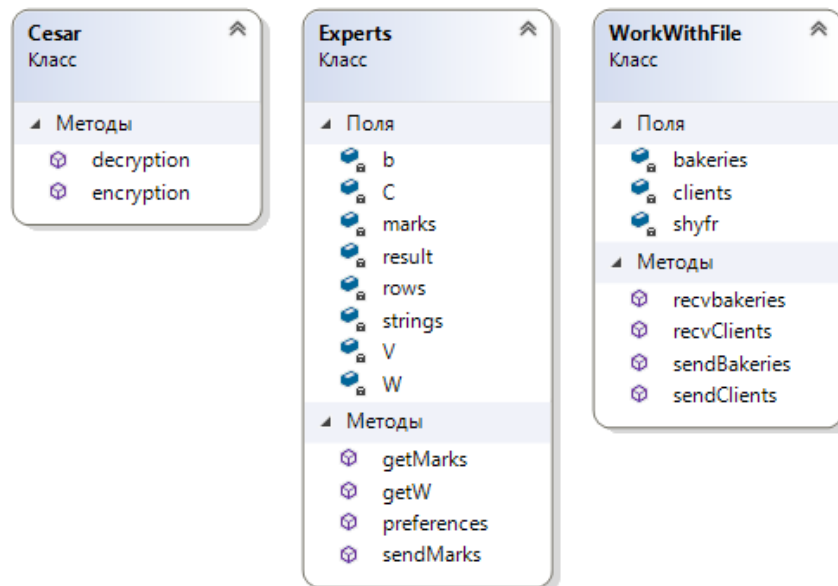


Рисунок 4.3 – Диаграмма классов сервера

Данные классы позволяют в полной мере обеспечить функционирование программы и предусматривают различные исключительные ситуации. В файле клиента содержится 6 классов и соответствующие им методы. В классе Client находится информация о пользователе, его имя, фамилия, логин и пароль, а также методы получения этих полей и функции по добавлению, редактированию и удалению данных. В классе Admin определены методы по получению логина и пароля, а также генерация отчета и другие. Класс Client наследует также методы класса Admin. В классе Bakery определены составляющие пекарни : название, оценки, предоставляемые услуги и т. д, а также соответствующие методы. Класс Company содержит информацию о клиентах, пекарнях и администраторе и содержит методы по изменению данных. Класс Service необходим для учета услуг и содержит поля с названием услуги, затратами, прибылью, количеством сотрудников и методы для работы и услугами. И класс BakeryException предназначен для работы в исключительных ситуациях, когда данные вводятся некорректно. В сервере содержится 3 класса : Cesar, Experts, WorkWithFile. Они предназначены для шифрования методом Цезаря, работы с экспертными оценками и работы с файлом.

## 5 ОБОСНОВАНИЕ ВЫБОРА ПРОГРАММНЫХ СРЕДСТВ РАЗРАБОТКИ

Для создания данного курсового проекта использовались такие инструменты как среда разработки Microsoft Visual Studio 2019 и язык программирования C++.

C++ создавался на основе языка C, и при этом до определенного момента сохранял с ним совместимость. Следовательно, C++ вобрал в себя всю скорострельность языка C. C++ используется во всех сферах деятельности программирования: от высоконагруженных систем до программирования микроконтроллеров. На C++ можно написать как web-сервер, так и игры, любые компьютерные программы, компоненты и так далее.

C++ позволяет писать как в процедурном стиле, так и в объектно-ориентированном и функциональном. Так как язык используется всеми крупными компаниями, то они же его и поддерживают — есть целый комитет по стандартизации C++, в который входят все самые крупные ИТ-компании.

Следует отметить, что в курсовом проекте реализованы основные принципы объектно-ориентированного программирования :

- инкапсуляция;
- полиморфизм;
- наследование;
- абстракция.

**Microsoft Visual Studio** — линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств. Данные продукты позволяют разрабатывать как консольные приложения, так и приложения с графическим интерфейсом, в том числе с поддержкой технологии Windows Forms, а также веб-сайты, веб-приложения, веб-службы как в родном, так и в управляемом кодах для всех платформ, поддерживаемых Windows, Windows Mobile, Windows CE, .NET Framework, Xbox, Windows Phone .NET Compact Framework и Silverlight.

Visual Studio включает в себя редактор исходного кода с поддержкой технологии IntelliSense и возможностью простейшего рефакторинга кода. Встроенный отладчик может работать как отладчик уровня исходного кода, так и отладчик машинного уровня. Остальные встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения, веб-редактор, дизайнер классов и дизайнер схемы базы данных. Visual Studio позволяет создавать и подключать сторонние

дополнения (плагины) для расширения функциональности практически на каждом уровне, включая добавление поддержки систем контроля версий исходного кода (как, например, Subversion и Visual SourceSafe), добавление новых наборов инструментов (например, для редактирования и визуального проектирования кода на предметно-ориентированных языках программирования) или инструментов для прочих аспектов процесса разработки программного обеспечения (например, клиент Team Explorer для работы с Team Foundation Server).

## **6 ОПИСАНИЕ АРХИТЕКТУРЫ РАЗРАБАТЫВАЕМОГО СРЕДСТВА**

В основе работы приложения лежит так называемая модель взаимодействия клиент-сервер.

Стоит заметить, что в основе взаимодействия клиент-сервер лежит принцип того, что такое взаимодействие начинает клиент, сервер лишь отвечает клиенту и сообщает о том может ли он предоставить услугу клиенту и если может, то на каких условиях. Клиентское программное обеспечение и серверное программное обеспечение обычно установлено на разных машинах, но также они могут работать и на одном компьютере.

Данная концепция взаимодействия была разработана в первую очередь для того, чтобы разделить нагрузку между участниками процесса обмена информацией, а также для того, чтобы разделить программный код поставщика и заказчика.

Преимуществом модели взаимодействия клиент-сервер является то, что программный код клиентского приложения и серверного разделен.

Также к преимуществам такой архитектуры относятся:

- Отсутствие дублирования кода программы-сервера программами-клиентами.
- Так как все вычисления выполняются на сервере, то требования к компьютерам, на которых установлен клиент, снижаются.

Приложения не управляют напрямую базой, управлением занимается только сервер. В связи с этим можно обеспечить высокую степень защиты данных.

В данном курсовом проекте работа построена на модели клиент-сервер, поэтому было необходимо создать два проекта. Один проект называется «clientKP». Это файл .cpp, где реализованы основные классы и функции, а также описаны возможности программного проекта, которые включают в себя управление пользователями, работа с информацией и так далее. Второй файл с расширением .cpp называется «serverKP», он представляет собой серверную часть приложения, обеспечивая непосредственно связь с клиентом.

Помимо этих файлов, были созданы и текстовые файлы. В файле «Admin.txt» хранится зашифрованный пароль администратора. В файле «Users.txt» находятся данные пользователя, а именно : имя, фамилия, зашифрованный пароль и логин. Также был создан файл «Bakery.txt», где представлена информация о пекарнях. Текстовые файлы помогают

структурировать информацию и предполагают просмотр данных в любой момент.

## 7 АЛГОРИТМ РАБОТЫ ПРОГРАММЫ

В данном курсовом проекте для принятия решения использовался метод полного попарного сравнения, алгоритм которого представлен на рисунке 7.1.



Рисунок 7.1 – Схема алгоритма решения задачи

Курсовой проект построен на взаимодействии клиент-сервер, алгоритм которого приведен на рисунке 7.2.



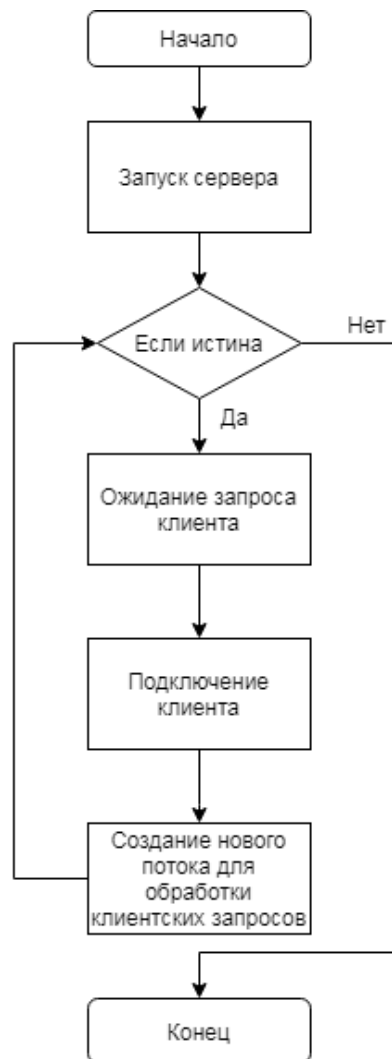


Рисунок 7.2 – Схема клиент-серверного взаимодействия

```

int main(int argc, char* argv[]) {
    system("chcp 1251 >> null");
    set_terminate(term_func);

    WSADATA wsaData;
    WORD DLLVersion = MAKEWORD(2, 1);
    if (WSAStartup(DLLVersion, &wsaData) != 0) {
        std::cout << "Error" << std::endl;
        exit(1);
    }

    SOCKADDR_IN addr;
    int sizeofaddr = sizeof(addr);
    addr.sin_addr.s_addr = inet_addr("127.0.0.1");
    addr.sin_port = htons(1111);
    addr.sin_family = AF_INET;

    Connection = socket(AF_INET, SOCK_STREAM, NULL);
    if (connect(Connection, (SOCKADDR*)& addr, sizeof(addr)) != 0) {
        std::cout << "Error: failed connect to server.\n";
        return 1;
    }
}

```

```

        std::cout << "Connected!\n";

int main(int argc, char* argv[]) {
    system("chcp 1251 >> null");

    WSADATA wsaData;
    WORD DLLVersion = MAKEWORD(2, 1);
    if (WSAStartup(DLLVersion, &wsaData) != 0) {
        std::cout << "Error" << std::endl;
        exit(1);
    }

    SOCKADDR_IN addr;
    int sizeofaddr = sizeof(addr);
    addr.sin_addr.s_addr = inet_addr("127.0.0.1");
    addr.sin_port = htons(1111);
    addr.sin_family = AF_INET;

    SOCKET sListen = socket(AF_INET, SOCK_STREAM, NULL);
    bind(sListen, (SOCKADDR*)& addr, sizeof(addr));
    listen(sListen, SOMAXCONN);

    newConnection = accept(sListen, (SOCKADDR*)& addr, &sizeofaddr);

    if (newConnection == 0) {
        std::cout << "Error #2\n";
    }
    else {
        std::cout << "Client Connected!\n";
    }
}

```

## 8 РУКОВОДСТВО ПО РАЗВЕРТЫВАНИЮ

После запуска программы появляется два консольных окна: сервера и клиента. Если соединение между ними установлено, то выводится сообщение: «Client Connected!» (рис. 8.1).

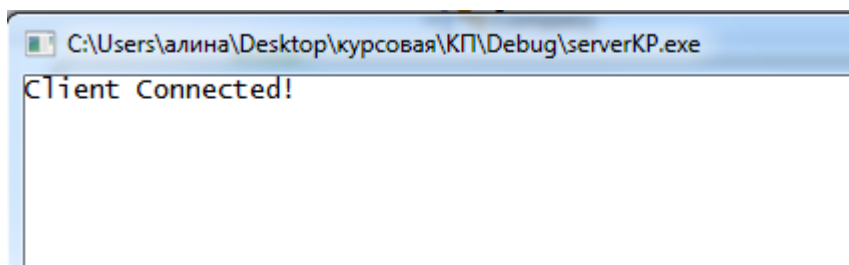


Рисунок 8.1 – Установка соединения сервера с клиентом

После установления соединения клиенту открывается меню, где он может войти как администратор или как пользователь (рис. 8.2)

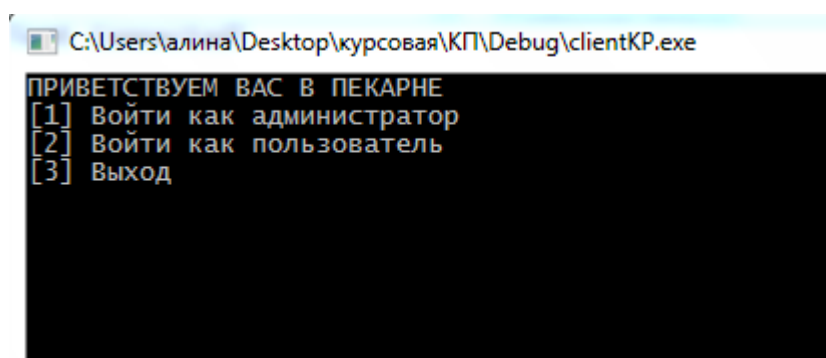


Рисунок 8.2 – Меню клиента

Чтобы войти как администратор, необходимо нажать «1» и ввести логин и пароль (рис. 8.3). Зашифрованный логин и пароль хранятся в файле «Admins.txt». После успешной авторизации откроется меню администратора (рис. 8.4).

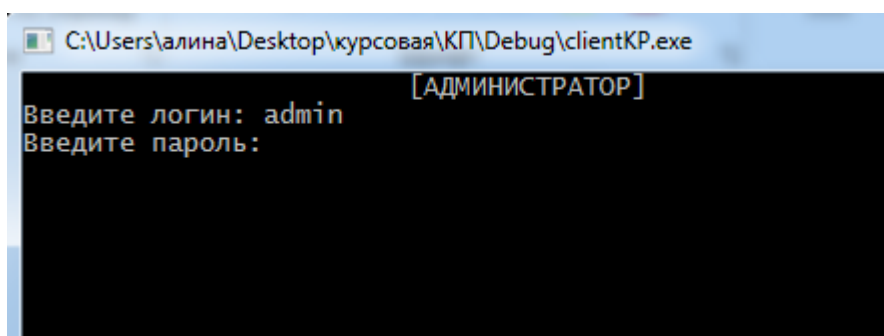


Рисунок 8.3 – Вход под администратором

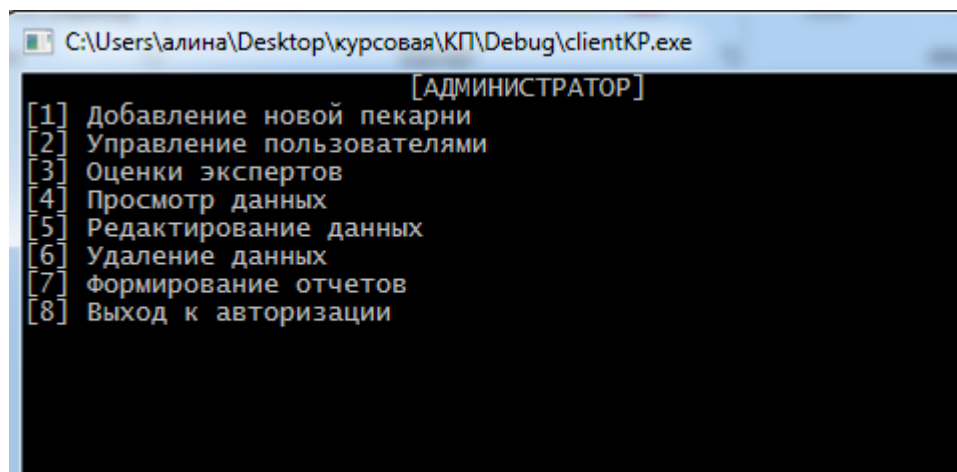


Рисунок 8.4 – Меню администратора

В окне сервера появится сообщение о том, что администратор вошел (рис. 8.5)

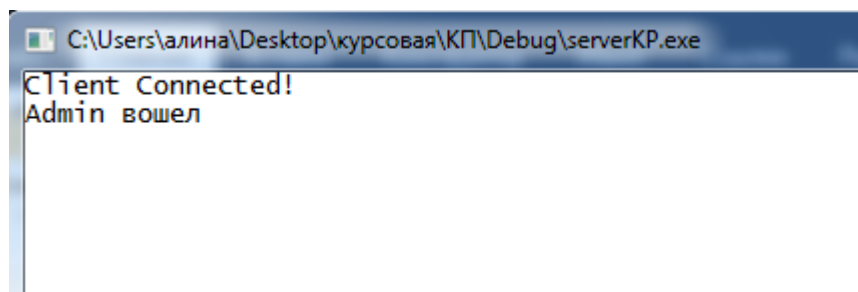


Рисунок 8.5 – Вход администратора

Если выйти из меню администратора, на сервере также появится соответствующее сообщение (рис. 8.6)

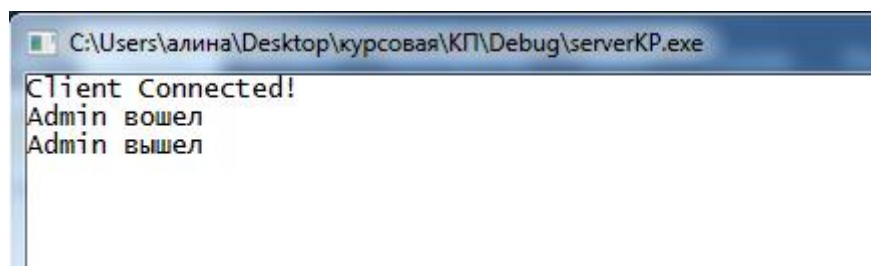


Рисунок 8.6 – Выход администратора

Теперь осуществим вход в качестве пользователя, выбрав пункт меню «2», необходимо ввести логин и пароль (рис. 8.7). При корректном вводе открывается меню (рис. 8.8).

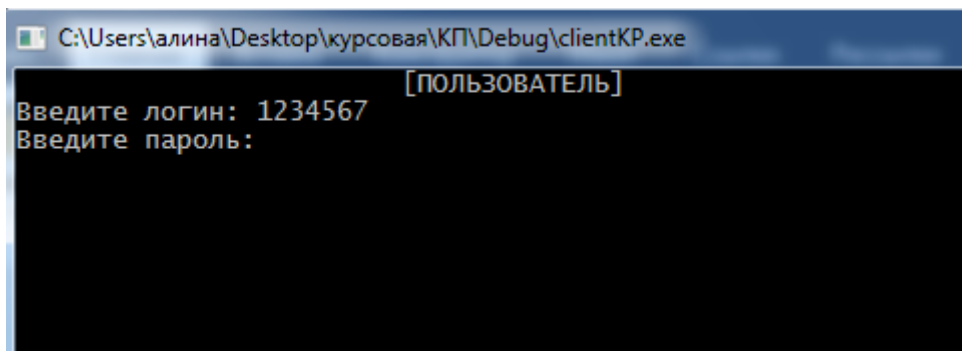


Рисунок 8.7 – Вход под пользователем

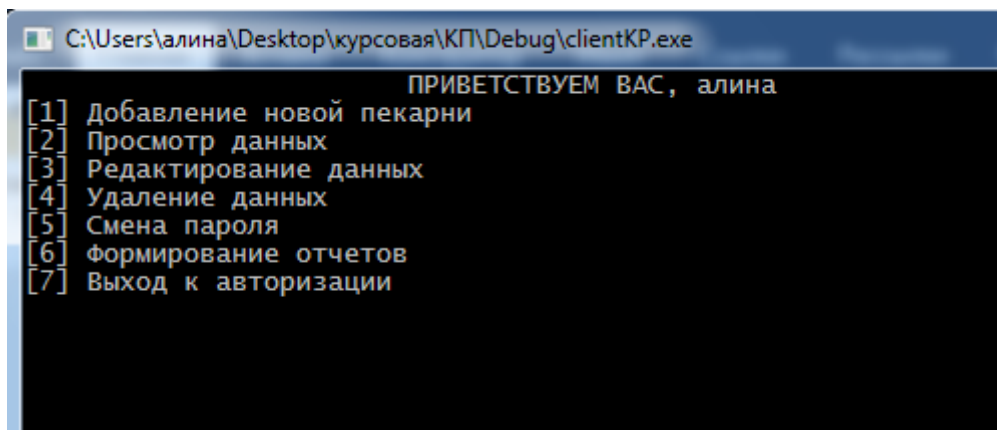


Рисунок 8.8 – Меню пользователя

В это время в окне сервера отображаются происходящие события. При успешном входе пользователя выводится сообщение (рис. 8.9)

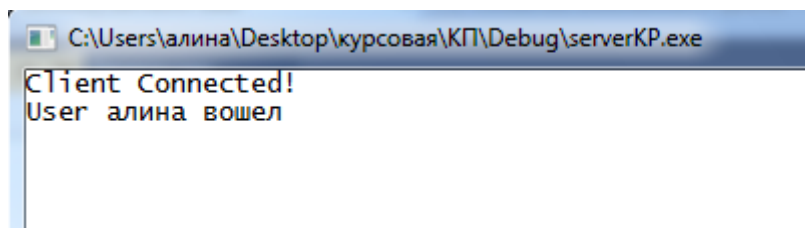


Рисунок 8.9 – Вход пользователя

При выходе пользователя на сервере также отображается сообщение (рис. 8.10).

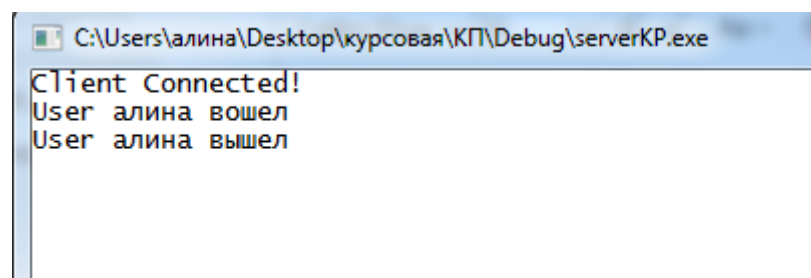


Рисунок 8.10 – Выход пользователя

## 9 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

Система, разработанная в данном курсовом проекте, предполагает вход в качестве пользователя или администратора.

Для того, чтобы войти в качестве администратора, необходимо корректно ввести логин и пароль, присущий данному пользователю. В случае ошибочного ввода программа выходит в меню. Управлять пользователями может администратор. Так, при правильном вводе логина и пароля, открывается меню администратора (рис. 9.1)

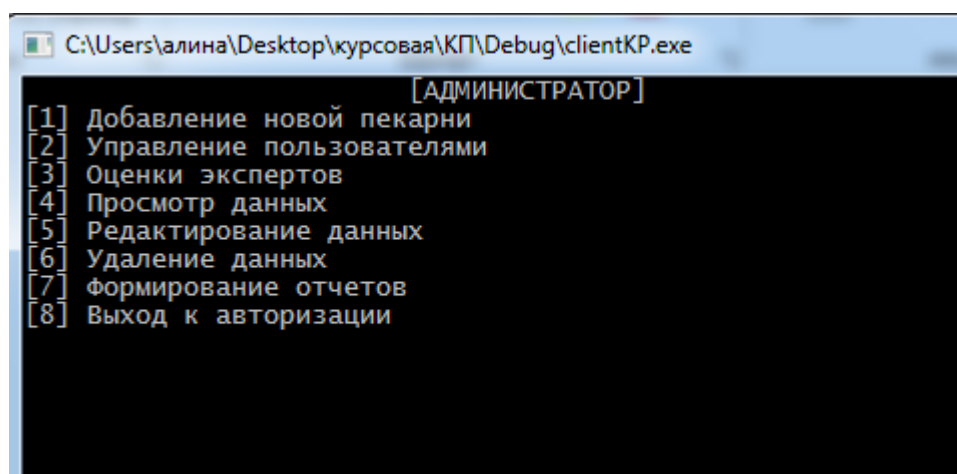


Рисунок 9.1 – Меню администратора

Администратору предоставлена возможность добавления, чтения, редактирования и удаления данных, также он может управлять пользователями, сформировать отчет или выйти в меню авторизации. Важной функцией администратора, которая отличает его от обычного пользователя – выставление оценок экспертами.

Для того, чтобы добавить новую пекарню, необходимо выбрать пункт меню «1» (рис. 9.2).

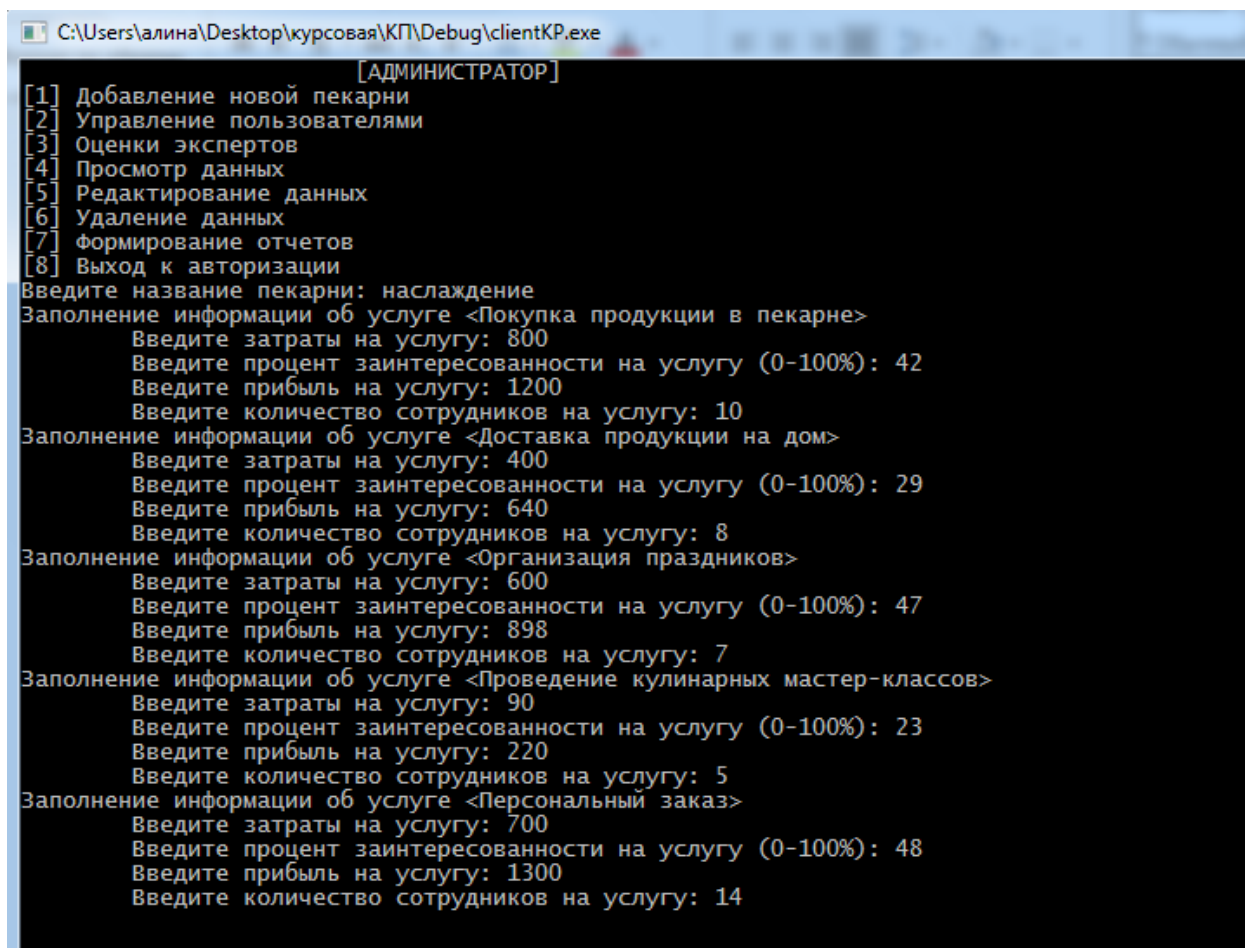


Рисунок 9.2 – Добавление новой пекарни

Администратор может также управлять пользователями, для этого нужно выбрать пункт меню «2» (рис. 9.3).

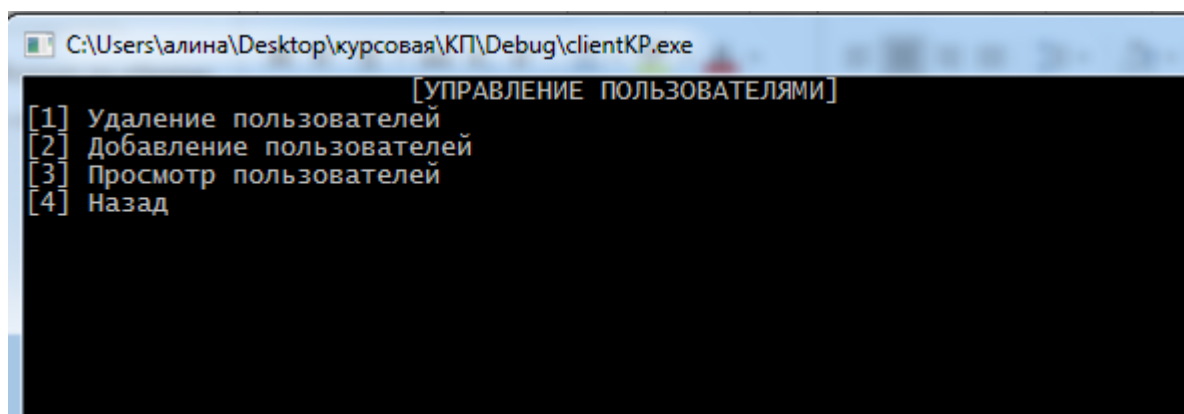


Рисунок 9.3 – Управление пользователями

Добавим нового пользователя, выбрав пункт меню «2» и введем требуемые данные (рис. 9.4).

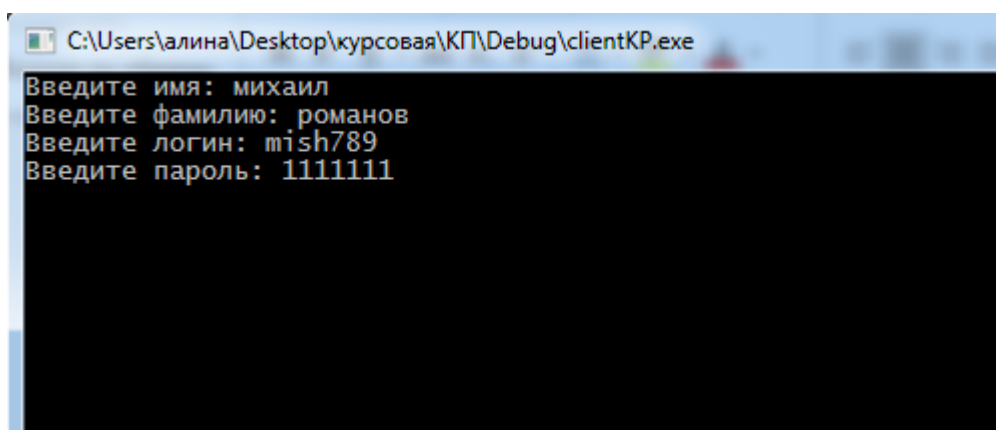


Рисунок 9.4 – Регистрация нового пользователя

После регистрации можем посмотреть существующих пользователей, где отобразится новый добавленный (рис. 9.5).

id	Имя	Фамилия	Логин	Пароль
0	алина	козырева	1234567	1234567
1	аня	иванова	1234555	1234555
2	михаил	романов	mish789	1111111

Рисунок 9.5– Просмотр пользователей

Для просмотра заведений необходимо выбрать пункт меню «4». Отобразятся пекарни, а также их владельцы и оценки. В случае, если оценок нет, отобразится статус «не проверено» (рис. 9.6)

id	Название	Владелец	Проверено	Оценки
0	nik	Admin	Проверено	0.045000 0.060000 0.090000 0.080000 0.040000
1	сладость	аня	Проверено	0.175000 0.155000 0.195000 0.260000 0.120000
2	наслаждение	Admin	Не проверено	0.000000 0.000000 0.000000 0.000000 0.000000
3	мечта	алина	Не проверено	0.000000 0.000000 0.000000 0.000000 0.000000

Нажмите Enter чтобы продолжить

Рисунок 9.6 – Просмотр данных

Также администратор может отредактировать запись о пекарне, выбрав пункт меню «5». Появится выбор, что именно администратор желает изменить. Можно отредактировать название пекарни, либо изменить параметры услуги. Выберем пункт 1 и введем новое название (рис. 9.7). Затем посмотрим отредактированную запись (рис. 9.8).



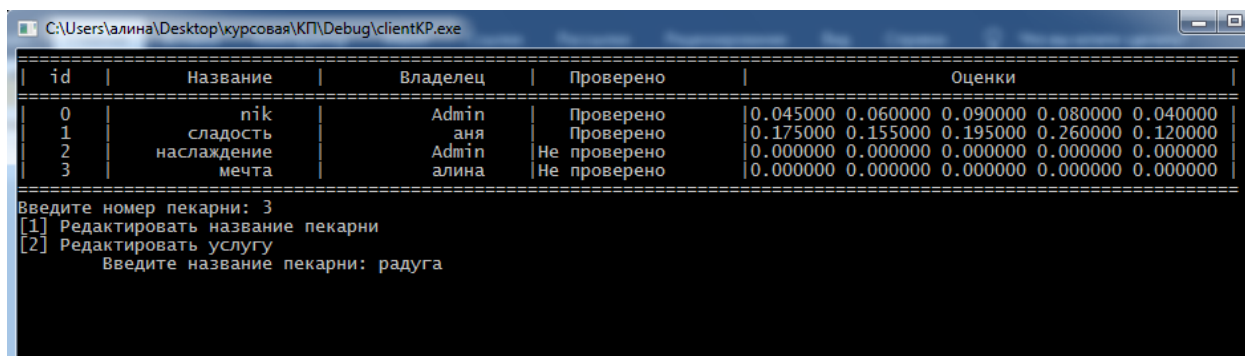


Рисунок 9.7 – Редактирование данных

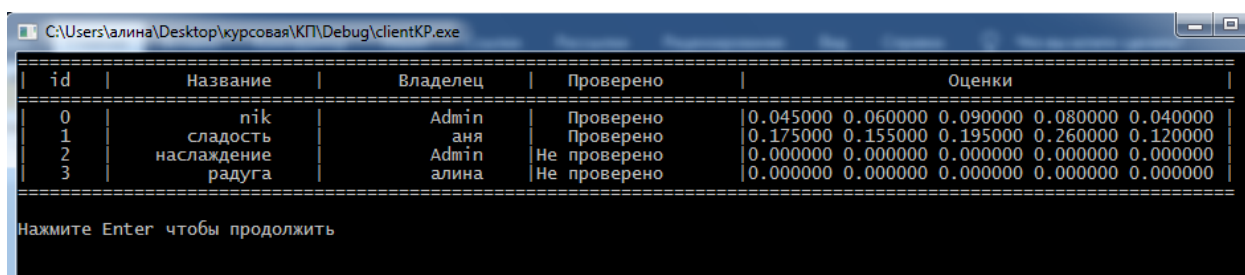


Рисунок 9.8 – Просмотр обновленной записи

Для удаления записи нужно выбрать пункт «б». Удалим пекарню под номером 2 (рис. 9.9), посмотрим обновленные данные (рис. 9.10) .

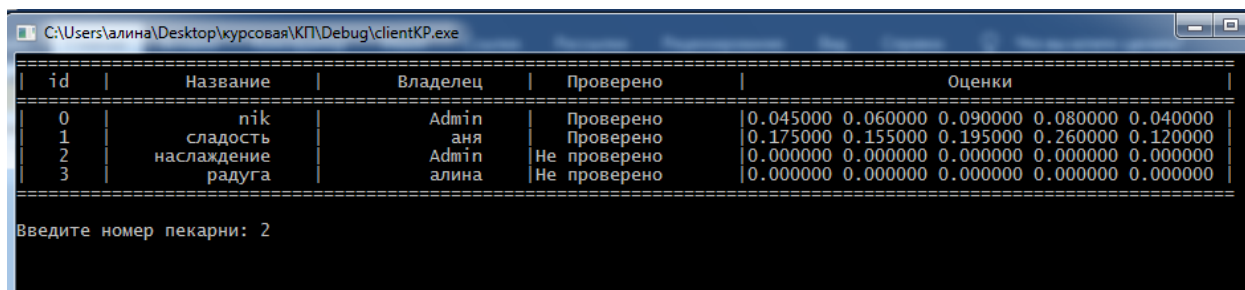


Рисунок 9.9 – Удаление пекарни

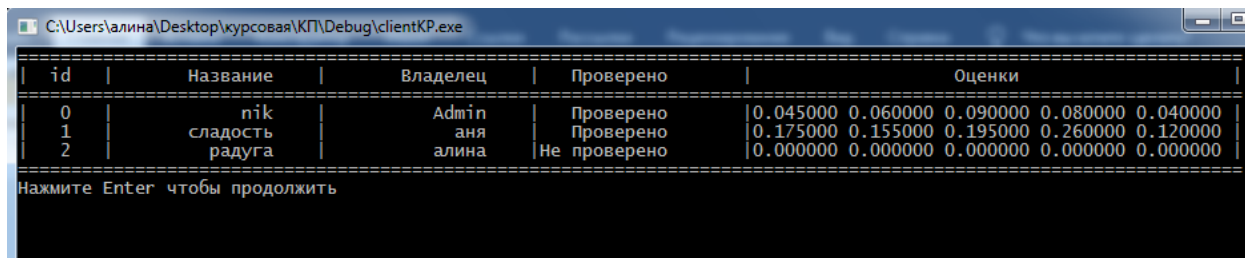


Рисунок 9.8 – Просмотр обновленных данных

Также администратор может сформировать отчеты, выбрав соответственно пункты 7. Для выхода нужно выбрать пункт меню 8, произойдет возврат в меню авторизации. Такую функцию администратора, как

выставление экспертных оценок , мы рассмотрим в качестве контрольного примера.

Далее войдем в систему в качестве пользователя, при верном вводе появится меню пользователя (рис. 9.9)

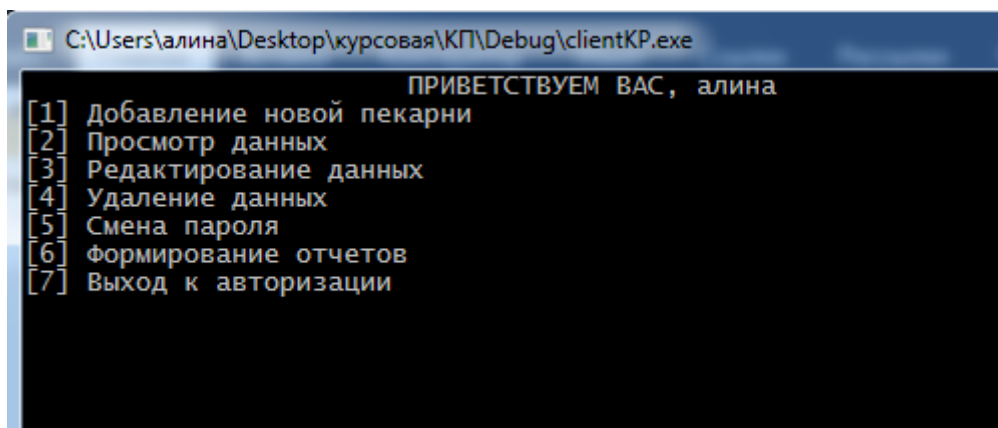


Рисунок 9.9 – Меню пользователя

Функции администратора схожи с пользовательскими, но администратор обладает более широкими возможностями. Рассмотрим пункт пользовательского меню «1» (рис. 9.10)

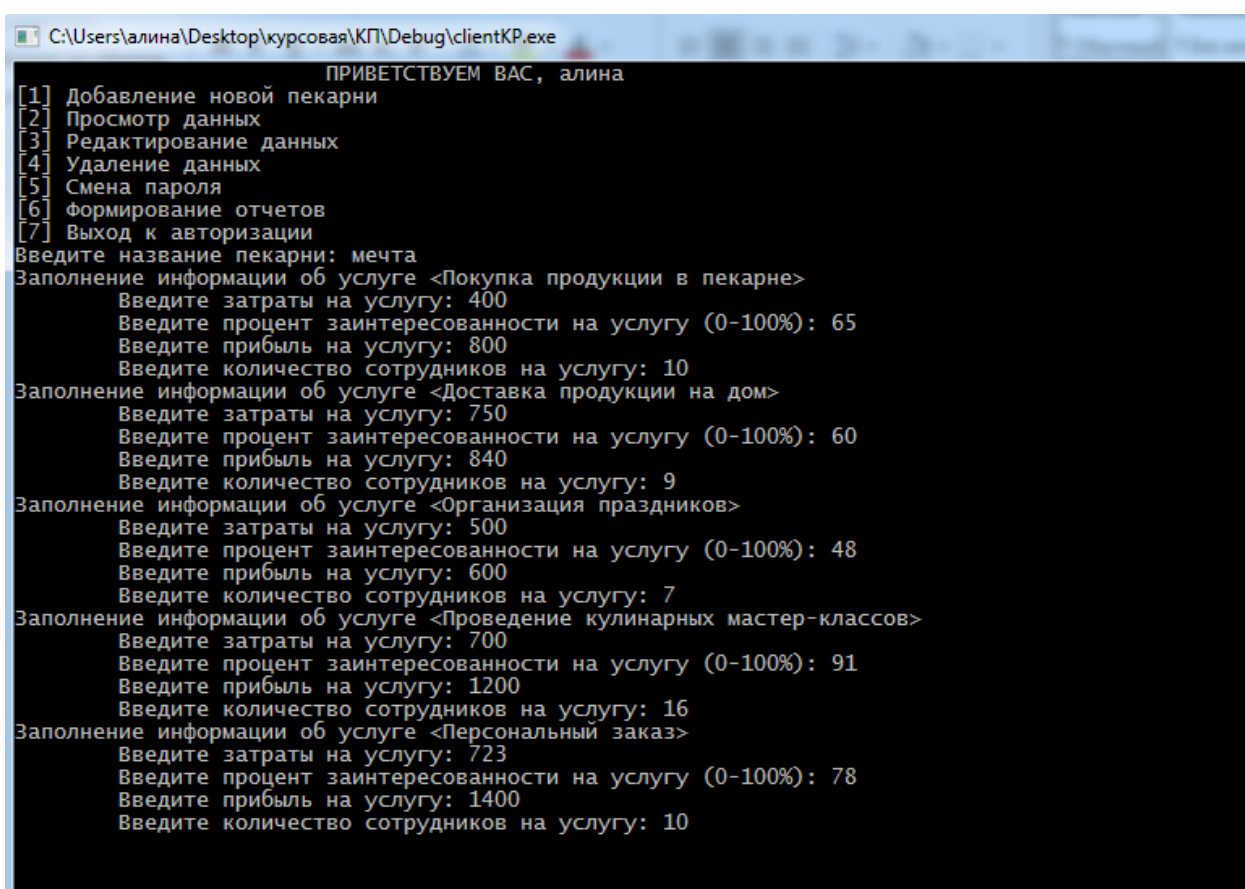
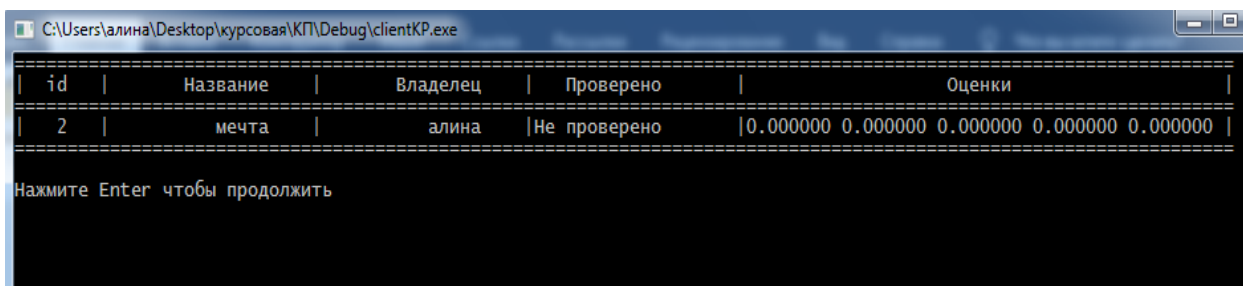


Рисунок 9.10 – Добавление новой пекарни

Теперь посмотрим данные, выбрав пункт меню «2» (рис. 9.11).



id	Название	Владелец	Проверено	Оценки				
2	мечта	алина	Не проверено	0.000000	0.000000	0.000000	0.000000	0.000000

Нажмите Enter чтобы продолжить

Рисунок 9.11 – Просмотр данных

Функции редактирования, удаления данных и формирования отчетов такие же, как и у администратора. Также пользователь может поменять свой пароль, выбрав пункт «5» меню.

## 10 КОНТРОЛЬНЫЙ ПРИМЕР

Особенностью данного курсового проекта является возможность оценки целей экспертами. Провести оценку можно, для этого нужно зайти под администратором и выбрать пункт меню «3» - оценки экспертов (рис. 10.1)

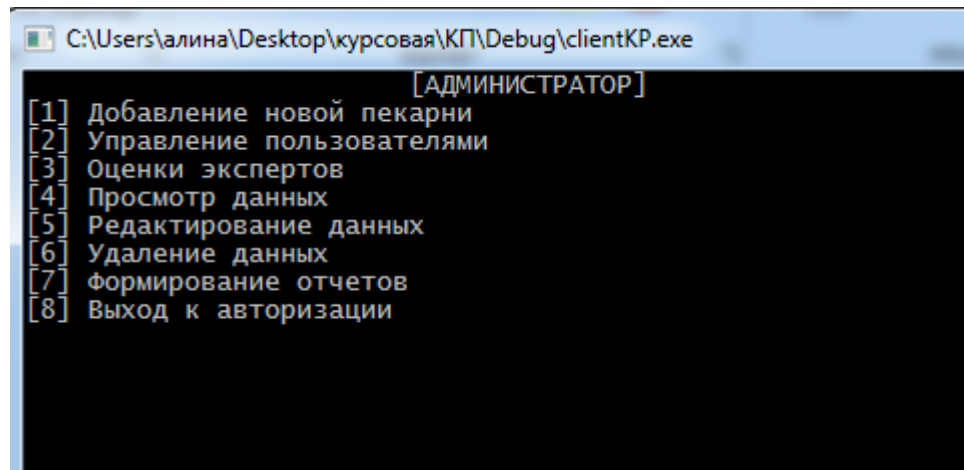
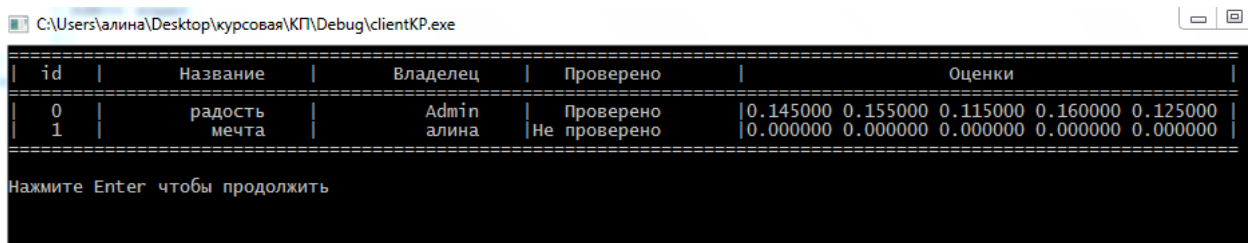


Рисунок 10.1 – Меню администратора

Для начала посмотрим существующие данные, выбрав пункт меню «4» (рис. 10.2).

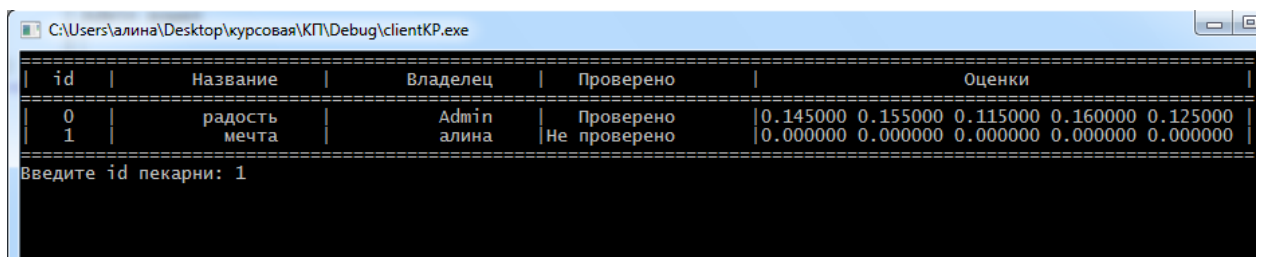


id	Название	Владелец	Проверено	Оценки
0	радость	Admin	Проверено	0.145000 0.155000 0.115000 0.160000 0.125000
1	мечта	алина	Не проверено	0.000000 0.000000 0.000000 0.000000 0.000000

Нажмите Enter чтобы продолжить

Рисунок 10.2 – Просмотр данных

Пекарня под номером один не проверена, тогда выйдем в меню и выберем пункт 3 – оценивание. Введем id пекарни (рис.10.3) и введем количество экспертов (рис. 10.4).



id	Название	Владелец	Проверено	Оценки
0	радость	Admin	Проверено	0.145000 0.155000 0.115000 0.160000 0.125000
1	мечта	алина	Не проверено	0.000000 0.000000 0.000000 0.000000 0.000000

Введите id пекарни: 1

Рисунок 10.3 – Выбор пекарни для оценки

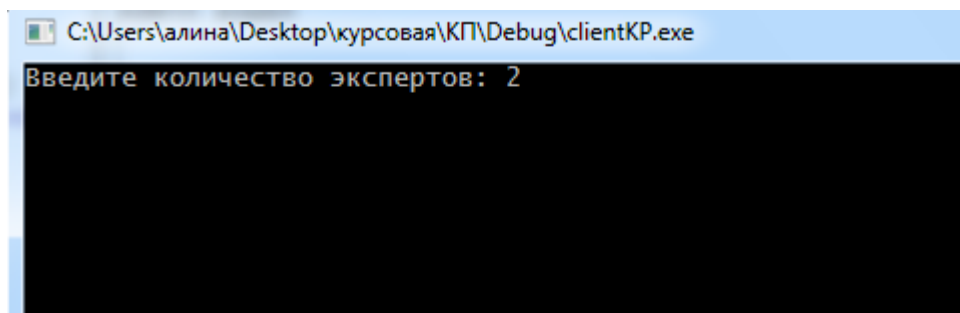


Рисунок 10.4 – Ввод количества экспертов

В курсовом проекте в качестве метода принятия решения выступает метод полного попарного сравнения. Согласно этому методу необходимо заполнить матрицу, где указывается предпочтение одной цели перед другой. Заполним матрицы оценок двух экспертов (рис. 10.5 и рис. 10.6)

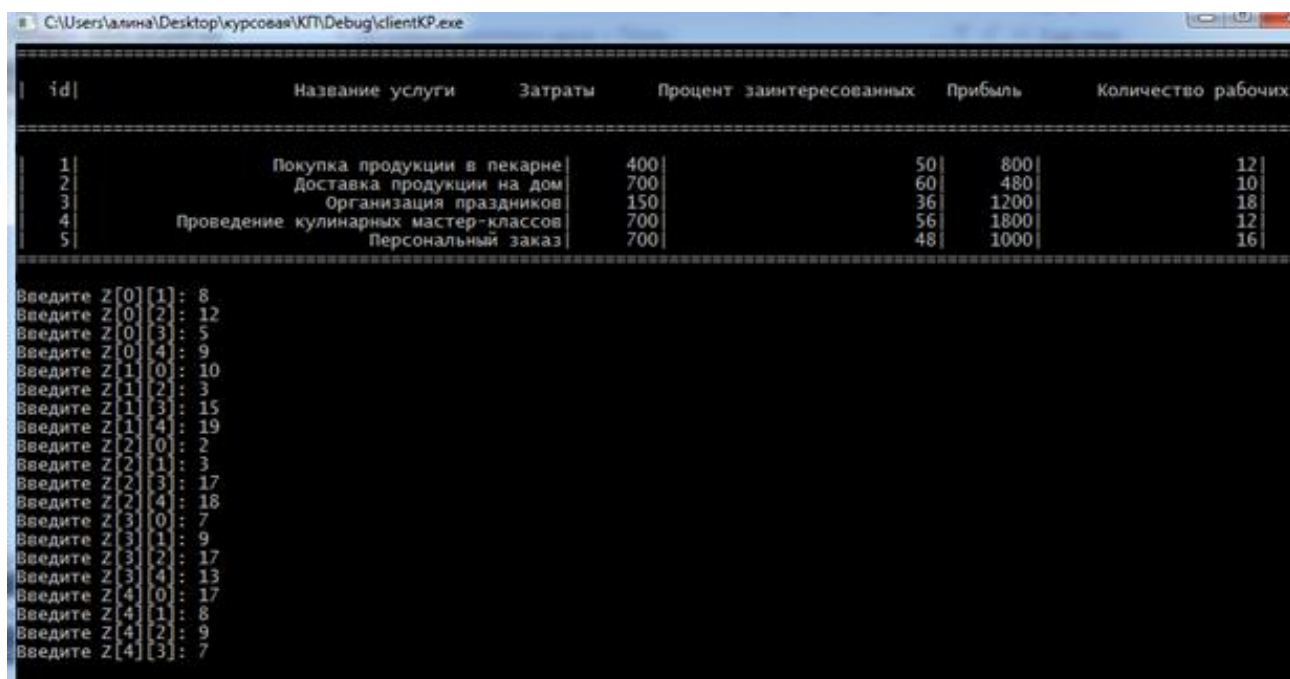


Рисунок 10.5 – Выставление оценок первым экспертом

```

Введите Z[0][1]: 4
Введите Z[0][2]: 8
Введите Z[0][3]: 9
Введите Z[0][4]: 15
Введите Z[1][0]: 2
Введите Z[1][2]: 2
Введите Z[1][3]: 19
Введите Z[1][4]: 18
Введите Z[2][0]: 17
Введите Z[2][1]: 13
Введите Z[2][3]: 14
Введите Z[2][4]: 2
Введите Z[3][0]: 8
Введите Z[3][1]: 9
Введите Z[3][2]: 10
Введите Z[3][4]: 12
Введите Z[4][0]: 7
Введите Z[4][1]: 8
Введите Z[4][2]: 9
Введите Z[4][3]: 7

```

Рисунок 10.6 – Выставление оценок вторым экспертом

Затем по методу были рассчитаны оценки для каждой из пяти услуг пекарни, выйдем в меню администратора и посмотрим данные, где отобразится обновленная информация (рис. 10.7). Теперь у пекарни появится статус «Проверено».

C:\Users\алина\Desktop\хурцовая\КП\Debug\clientKP.exe

id	Название	Владелец	Проверено	Оценки				
0	радость	Admin	Проверено	0.145000	0.155000	0.115000	0.160000	0.125000
1	мечта	алина	Проверено	0.170000	0.235000	0.200000	0.230000	0.205000

Нажмите Enter чтобы продолжить

Рисунок 10.7 – Просмотр данных после оценки экспертами

Теперь сформируем для этой пекарни отчет, выбрав пункт меню «7» (рис.10.8).

C:\Users\алина\Desktop\хурцовая\КП\Debug\clientKP.exe

id	Название	Владелец	Проверено	Оценки				
0	радость	Admin	Проверено	0.145000	0.155000	0.115000	0.160000	0.125000
1	мечта	алина	Проверено	0.170000	0.235000	0.200000	0.230000	0.205000

Введите номер пекарни: 1  
Введите имя файла: мечта

Рисунок 10.8 – Формирование отчета

Выйдем из программы и откроем файл под названием «мечта» (рис. 10.9).

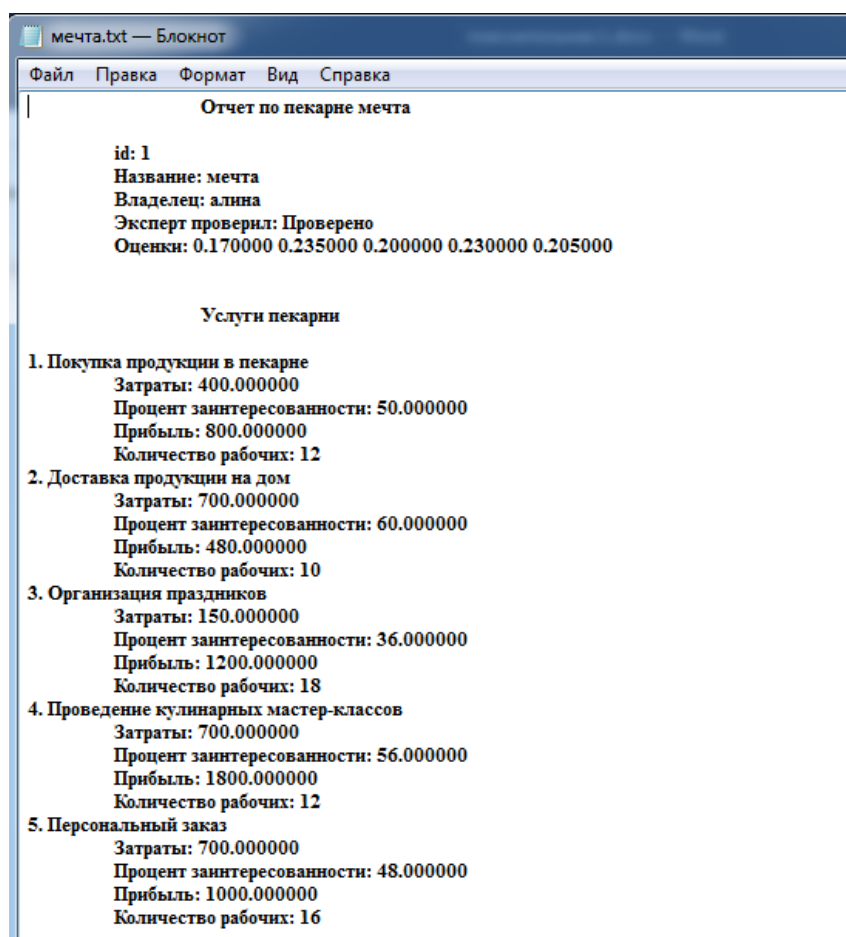


Рисунок 10.9 – Сформированный отчет

В курсовом проекте предусмотрены и обработаны исключения, так, при вводе некорректных данных, программа сообщает об ошибке и предлагает повторно ввести информацию. Рассмотрим некоторые примеры на рисунках 10.10 и 10.11.

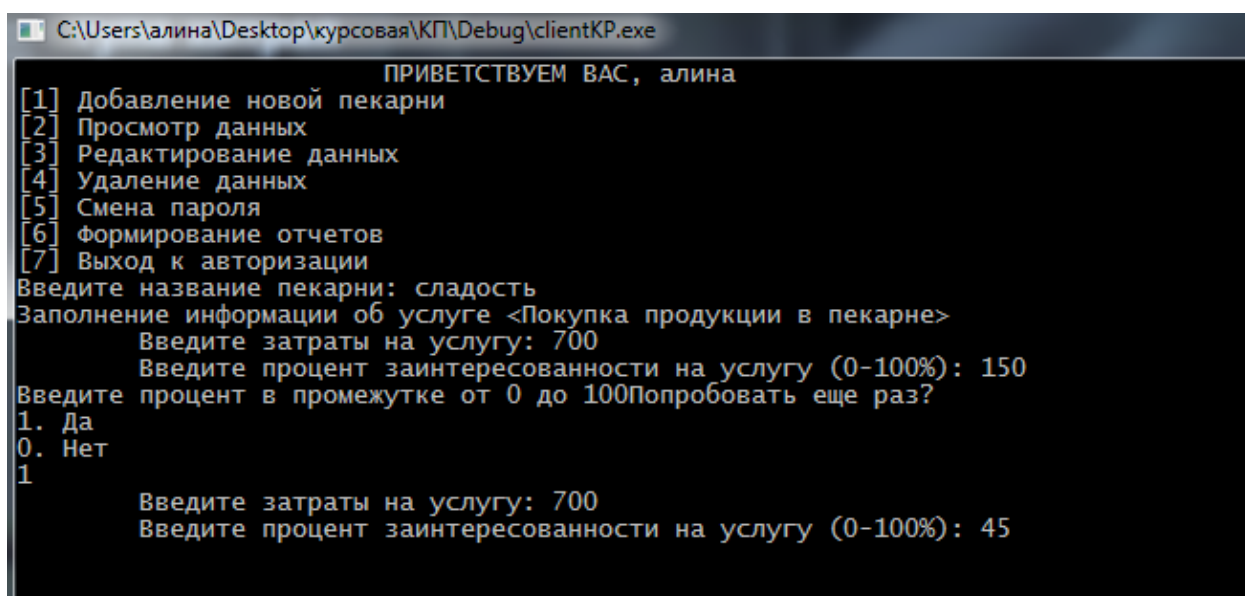


Рисунок 10.10 – Некорректный ввод процентных данных



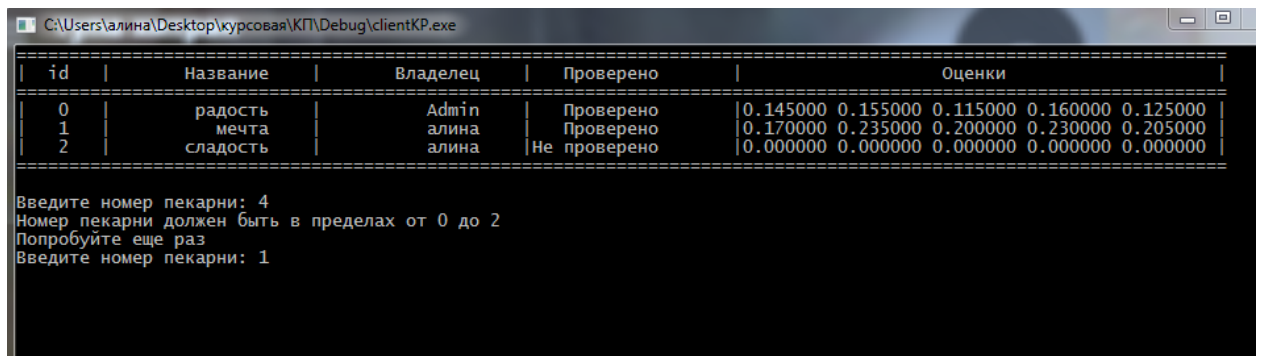


Рисунок 10.11 – Некорректный ввод номера

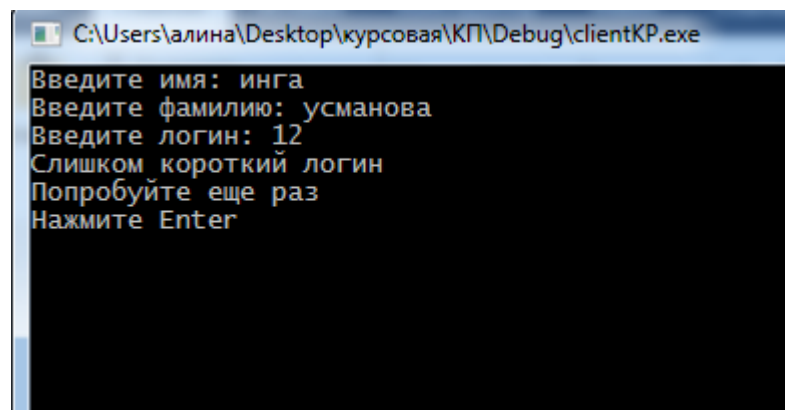


Рисунок 10.12 – Некорректный ввод логина



## **ЗАКЛЮЧЕНИЕ**

Развивающееся производство требует и развития систем. Часто для принятия правильных решений не хватает информации в полном ее объеме, а также статистических данных для анализа сложившейся ситуации. Средство, которое было разработано, позволяет на основе реальных данных и вычислений сформировать для пользователя корректное решение.

В данном приложении создаются условия для удобного хранения информации, работы с ней, предоставление её в приемлемом для пользователя виде. Пользователю предоставляется понятный и простой интерфейс для работы. Также были учтены и предупреждены возможные возникновения ошибок.

При разработке были решены следующие задачи: была исследована предметная область задания; разработаны классы для хранения данных; разработаны пользовательские функции приложения, функциональные схемы приложения; была описана программа.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Каплан Р., Нортон Д. Сбалансированная система показателей. От стратегии к действию / Р. Каплан, Д. Нортон. – М.: ЗАО «Олимп-Бизнес», 2008 г. – 320 с.

[2] Луцик Ю.А., Ковальчук А.М., Лукьянова И.В. «Объектно-ориентированное программирование на языке C++», 2003.

[3] БГУИР [Электронный ресурс] – официальный сайт высшего учебного заведения БГУИР. – Электронные данные. – Режим доступа: <http://www.bsuir.by>

[4] Герберт Шилдт. С: полное руководство, классическое издание – Вильямс, 2010

[5] Репин В.В. Разработка архитектуры бизнес-процессов компании в Business Studio / В.В. Репин. – М.: Манн, Иванов и Фербер, 2019. – 142 с.

[6] Скотт К., Фауер М. UML. Основы. 3-е издание. / К. Скотт, М. Фаулер. – М.: Символ, 2016. – 192 с.

## ПРИЛОЖЕНИЕ А

### Листинг кода

#### Файл serverKP.cpp

```
class Cesar {
public:
    std::string encryption(std::string original) {
        std::string TC;
        for (int i = 0; i < (original.length()); i++) {
            TC = TC + (char)((original[i] + k));
        }
        return TC;
    }
    std::string decryption(std::string encOriginal) {
        std::string UnTC;
        for (int i = 0; i < (encOriginal.length()); i++) {
            UnTC = UnTC + (char)((encOriginal[i] - k));
        }
        return UnTC;
    }
};

class WorkWithFile {
private:
    vector<string> clients;
    vector<string> bakeries;
    Cesar shyfr;
public:
    void sendClients() {
        ifstream clients("Clients.txt");
        char login[BUFF], pass[BUFF], name[BUFF], surname[BUFF];
        while (!clients.eof()) {
            clients >> login >> pass >> name >> surname;
            strncpy(login, shyfr.decryption(login).c_str(), sizeof(login));
            strncpy(pass, shyfr.decryption(pass).c_str(), sizeof(pass));
            this->clients.push_back(string(login) + " " + string(pass) + " " +
string(name) + " " + string(surname));
        }
        clients.close();
        char countClients[BUFF];
        itoa(this->clients.size(), countClients, 10);
        send(newConnection, countClients, sizeof(countClients), NULL); //отправить
КОЛИЧЕСТВО КЛИЕНТОВ
        vector<string>::iterator it;
        for (it = this->clients.begin(); it != this->clients.end(); it++) {

            char clientInfo[BUFF];
            strncpy(clientInfo, it->c_str(), sizeof(clientInfo));
            send(newConnection, clientInfo, sizeof(clientInfo), NULL);
        }
        this->clients.clear();
    }
    void sendBakeries() {
        string bakery;
        ifstream filebakeries("Bakeries.txt");
        while (getline(filebakeries, bakery)) {
            bakeries.push_back(bakery);
        }
        char bakeryCount[BUFF];
        itoa(bakeries.size(), bakeryCount, 10);
        send(newConnection, bakeryCount, sizeof(bakeryCount), NULL);
    }
};
```

```

        vector<string>::iterator it;
        for (it = this->bakeries.begin(); it != this->bakeries.end(); it++) {
            char bakeryInfo[BUFF];
            strncpy(bakeryInfo, it->c_str(), sizeof(bakeryInfo));
            send(newConnection, bakeryInfo, sizeof(bakeryInfo), NULL);
        }
        this->bakeries.clear();
    }
    void recvClients() {
        char countClient[BUFF];
        recv(newConnection, countClient, sizeof(countClient), NULL);
        if (atoi(countClient) == 0)
        {
            ofstream clients("Clients.txt");
            clients.close();
            return;
        }
        ofstream clients("Clients.txt");
        char clientInfo[BUFF];
        recv(newConnection, clientInfo, sizeof(clientInfo), NULL);
        char login[BUFF], pass[BUFF], name[BUFF], surname[BUFF];
        strncpy(login, strtok(clientInfo, " "), sizeof(login));
        strncpy(pass, strtok(NULL, " "), sizeof(pass));
        strncpy(name, strtok(NULL, " "), sizeof(name));
        strncpy(surname, strtok(NULL, " "), sizeof(surname));
        clients << shyfr.encryption(login) << " " << shyfr.encryption(pass) << " "
<< name << " " << surname;
        for (int i = 1; i < atoi(countClient); i++) {
            recv(newConnection, clientInfo, sizeof(clientInfo), NULL);
            char login[BUFF], pass[BUFF], name[BUFF], surname[BUFF];
            strncpy(login, strtok(clientInfo, " "), sizeof(login));
            strncpy(pass, strtok(NULL, " "), sizeof(pass));
            strncpy(name, strtok(NULL, " "), sizeof(name));
            strncpy(surname, strtok(NULL, " "), sizeof(surname));
            clients << endl;
            clients << shyfr.encryption(login) << " " << shyfr.encryption(pass)
<< " " << name << " " << surname;
        }
        clients.close();
        this->clients.clear();
    }
    void recvbakeries() {
        char answer_count[BUFF];
        recv(newConnection, answer_count, sizeof(answer_count), NULL);
        if (strcmp(answer_count, "-1") == 0) {
            ofstream mbakeries("Bakeries.txt");
            mbakeries.close();
            return;
        }

        int count = atoi(answer_count);
        char databakeries[BUFF];
        ofstream mbakeries("Bakeries.txt");
        recv(newConnection, databakeries, sizeof(databakeries), NULL);
        mbakeries << databakeries;
        for (int bakeryNum = 1; bakeryNum < count; bakeryNum++) {
            recv(newConnection, databakeries, sizeof(databakeries), NULL);
            mbakeries << endl;
            mbakeries << databakeries;
        }
        mbakeries.close();
    }
};

```

## Файл clientKP.cpp

```
class Client {
protected:
    char name[BUFF];
    char surname[BUFF];
    char login[BUFF];
    char password[BUFF];
public:
    Client() {
        std::strncpy(this->name, "", sizeof(this->name));
        std::strncpy(this->surname, "", sizeof(this->surname));
        std::strncpy(this->login, "", sizeof(this->login));
        std::strncpy(this->password, "", sizeof(this->password));
    };
    void addBakery();
    void addClient(char name[BUFF], char surname[BUFF], char login[BUFF], char
password[BUFF]) {
        std::strncpy(this->name, name, sizeof(this->name));
        std::strncpy(this->surname, surname, sizeof(this->surname));
        std::strncpy(this->login, login, sizeof(this->login));
        std::strncpy(this->password, password, sizeof(this->password));
    }
    void editBakery();
    void delBakery();
    void changePass(char pass[BUFF]) {
        strncpy(this->password, pass, sizeof(this->password));
    }
    string getName() {
        return string(name);
    };
    string getSurname() {
        return string(surname);
    };
    string getLogin() {
        return string(login);
    };
    string getPass() {
        return string(password);
    };
    string getInfo() {
        string result;
        result += string(login) + " " + string(password) + " " + string(name) + " "
+ string(surname);
        return result;
    }
};

class Admin : public Client {
public:
    Admin() {
        std::strncpy(this->login, "", sizeof(this->login));
        std::strncpy(this->password, "", sizeof(this->password));
    }
    void setAutorisation(const char login[BUFF], const char password[BUFF]) {
        std::strncpy(this->login, login, sizeof(this->login));
        std::strncpy(this->password, password, sizeof(this->password));
    }
    void setMarks();
    void calculate();
    void addClient();
    void generateReport();
};
```

```

class Service {
private:
    string name;
    double expenses; // затраты на услугу
    double perInterest; // процент заинтересованности относительно всех услуг
    double profit; // прибыль
    int countEmployee; // количество сотрудников
public:
    Service(string name) {
        this->name = name;
    };
    void setExpenses(double expenses) {
        this->expenses = expenses;
    }
    void setPerInterest(double perInterest) {
        this->perInterest = perInterest;
    }
    void setProfit(double profit) {
        this->profit = profit;
    }
    void setCountEmployee(int count) {
        this->countEmployee = count;
    }
    string getName() {
        return name;
    }
    string getInfo() {
        string result;
        result += to_string(expenses) + " " + to_string(perInterest) + " "
            + to_string(profit) + " " + to_string(countEmployee);
        return result;
    }
    void getInfoInTable(int id) {
        cout << setw(1) << "|" << setw(4) << id << setw(1) << "|" <<
            setw(45) << this->name << setw(1) << "|" <<
            setw(8) << this->expenses << setw(1) << "|" <<
            setw(25) << this->perInterest << setw(1) << "|" <<
            setw(8) << this->profit << setw(1) << "|" <<
            setw(20) << this->countEmployee << setw(1) << "|";
    }
    void getInfoInWholeTable(int id) {
        cout << setw(110) << setfill('=') << "=" << endl;
        cout << setfill(' ');
        cout << setw(1) << "|" << setw(4) << "id" << setw(1) << "|" <<
            setw(67) << "Название услуги" << setw(1) << "|" <<
            setw(2) << "Затраты" << setw(1) << "|" <<
            setw(25) << "Процент заинтересованных" << setw(1) << "|" <<
            setw(8) << "Прибыль" << setw(1) << "|" <<
            setw(9) << "Количество рабочих" << setw(1) << "|" << endl;
        cout << setw(110) << setfill('=') << "=" << endl;
        cout << setfill(' ');
        this->getInfoInTable(id);
        cout << endl;
        cout << setw(110) << setfill('=') << "=" << endl;
        cout << setfill(' ');
    }
    string getExpenses() {
        return to_string(this->expenses);
    }
    string getPerInterest() {
        return to_string(this->perInterest);
    }
    string getProfit() {

```

```

        return to_string(this->profit);
    }
    string getCountEmployee() {
        return to_string(this->countEmployee);
    }
};

class BakeryException : public std::exception {
private:
    std::string m_error;
public:
    BakeryException(std::string error) :m_error(error) {}
    const char* what() const noexcept {
        return m_error.c_str();
    }
};

string beautiful(string original) {
    for (int i = 0; i < original.length(); i++) {
        if (original[i] == ' ') {
            original[i] = '_';
        }
    }
    return original;
}

class Bakery {
private:
    char name[BUFF];
    char owner_name[BUFF];
    std::vector<Service> services;
    bool isChecked = false;
    double marks[5];
public:
    Bakery(const char owner_name[BUFF]) {
        strncpy(this->name, "", sizeof(this->name));
        strncpy(this->owner_name, owner_name, sizeof(this->owner_name));
        for (int i = 0; i < 5; i++) marks[i] = 0;
    };
    void setMarks(double marks[5]) {
        for (int i = 0; i < 5; i++) {
            this->marks[i] = marks[i];
        }
    }
    bool checkMarks(int expertMark, int marks[5]) {
        for (int i = 0; i < 5; i++) {
            if (marks[i] == expertMark) return true;
        }
        return false;
    }
    void setMarks() {
        system("cls");
        int expertsCount = 0;
        while (true) {
            try
            {
                cout << "Введите количество экспертов: ";
                cin >> expertsCount;
                if (!cin.good() || expertsCount < 2 || expertsCount > 10) {
                    throw BakeryException("Количество экспертов может быть
от 2 до 10");
                }
            }
            break;
        }
    }
};

```

```

    }
    catch (BakeryException& error)
    {
        cout << error.what() << endl;
        cin.clear();
        rewind(stdin);
        cout << "Попробуйте еще раз" << endl;
    }
}
char countOfExperts[BUFF];
itoa(expertsCount, countOfExperts, sizeof(countOfExperts));
send(Connection, countOfExperts, sizeof(countOfExperts), NULL);
for (int i = 0; i < expertsCount; i++) {
    system("cls");
    int expertMarks[5][5];
    for (int i = 0; i < 5; i++)
        for (int j = 0; j < 5; j++)
            expertMarks[i][j] = -1;
    vector<Service>::iterator itService;
    itService = services.begin();
    cout << setw(122) << setfill('=') << "=" << endl;
    cout << setfill(' ');
    cout << setw(1) << "|" << setw(4) << "id" << setw(1) << "|" <<
        setw(35) << "Название услуги" << setw(5) << "|" <<
        setw(8) << "Затраты" << setw(5) << "|" <<
        setw(25) << "Процент заинтересованных" << setw(2) << "|" <<
        setw(8) << "Прибыль" << setw(5) << "|" <<
        setw(20) << "Количество рабочих" << setw(3) << "|" << endl;
    cout << setw(122) << setfill('=') << "=" << endl;
    cout << setfill(' ');
    for (int serviceNum = 0; serviceNum < 5; serviceNum++, itService++) {
        itService->getInfoInTable(serviceNum + 1);
        cout << endl;
    }

    //cout << endl;
    cout << setw(122) << setfill('=') << "=" << endl;
    cout << setfill(' ');
    for (int i1 = 0; i1 < 5; i1++) {
        for (int j = 0; j < 5; j++) {
            if (i1 == j) continue;
            while (true) {
                try {
                    cout << "Введите Z[" << i1 << "][" << j <<
                        ": ";

                    cin >> expertMarks[i1][j];
                    if (!cin.good() || expertMarks[i1][j] < 0
                        || expertMarks[i1][j] > 20) {
                        throw BakeryException("Оценка
                            должна быть от 0 до 20");
                    }
                    break;
                }
                catch (BakeryException& error) {
                    cout << error.what() << endl;
                }
            }
        }
        char sendMarksString[BUFF];
        strcpy(sendMarksString, "");
        char mark[BUFF];
        for (int j = 0; j < 4; j++) {

            itoa(expertMarks[i1][j], mark, 10);

```



```

        strcat(sendMarksString, mark);
        strcat(sendMarksString, " ");
    }
    itoa(expertMarks[i1][4], mark, 10);
    strcat(sendMarksString, mark);
    send(Connection, sendMarksString, sizeof(sendMarksString),
NULL);
    }
}
}

```