**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

**к проекту**

**по дисциплине «Спецификация, проектирование и архитектура программных систем»**

|  |  |  |
| --- | --- | --- |
| Студентка гр. 0304 |  | Говорющенко А.В. |
| Студент гр. 0304 |  | Люлин Д.В. |
| Студент гр. 0304 |  | Алексеев Р.В. |
| Студент гр. 0304 |  | Никитин Д.Э. |
| Студент гр. 0304 |  | Максименко Е.М. |
| Руководитель |  | Романенко С.А. |

Санкт-Петербург

2022

**введение**

Работа представляет собой проект по созданию автоматизированной системы для автоматизации процессов сети пекарен-булочных. Было разработано ТЗ, в соответствии с которым разрабатывалась архитектура системы и описывалась предметная область. Результат работы описан в пояснительной записке.

**1.Общие положения**

* 1. **Полное наименование системы**

Автоматизированная система «Пекарня-булочная».

Условное обозначение: АС «Пекарня-булочная».

* 1. **Цели, назначение и области использования АС**

Система предназначена для прогнозирования количества заказов продукции каждой из пекарен сети и необходимого сырья для их выполнения, автоматизации учёта склада готовой продукции и сырья, обработки заказов пользователей, сделанных, в том числе и онлайн через сторонние службы доставки, приема оплаты заказов, а также для автоматизированной отправки запросов в центральный офис на поставку сырья.

В результате создания системы ожидается повышение уровня автоматизации процессов управления сетью пекарен на 90%.

**2. Описание процесса деятельности**

**2.1. Описание терминов предметной области**

**2.1.1. Словарь терминов**

* Клиент – клиент сети пекарен-булочных или сотрудник сторонней службы доставки
* Персонал – сотрудники сети пекарен-булочных: пекари, менеджеры, администраторы системы
* Регистрация – процесс создания учетной записи пользователя в системе
* Авторизация – процесс входа пользователем в систему
* Валидация – проверка корректности введенных данных
* Логирование – процесс сохранения данных о работе системы в журнал (лог)

**2.2. Описание технологических процессов предметной области в нотации IDEF0**

**2.2.1. Контекстная диаграмма IDEF0**

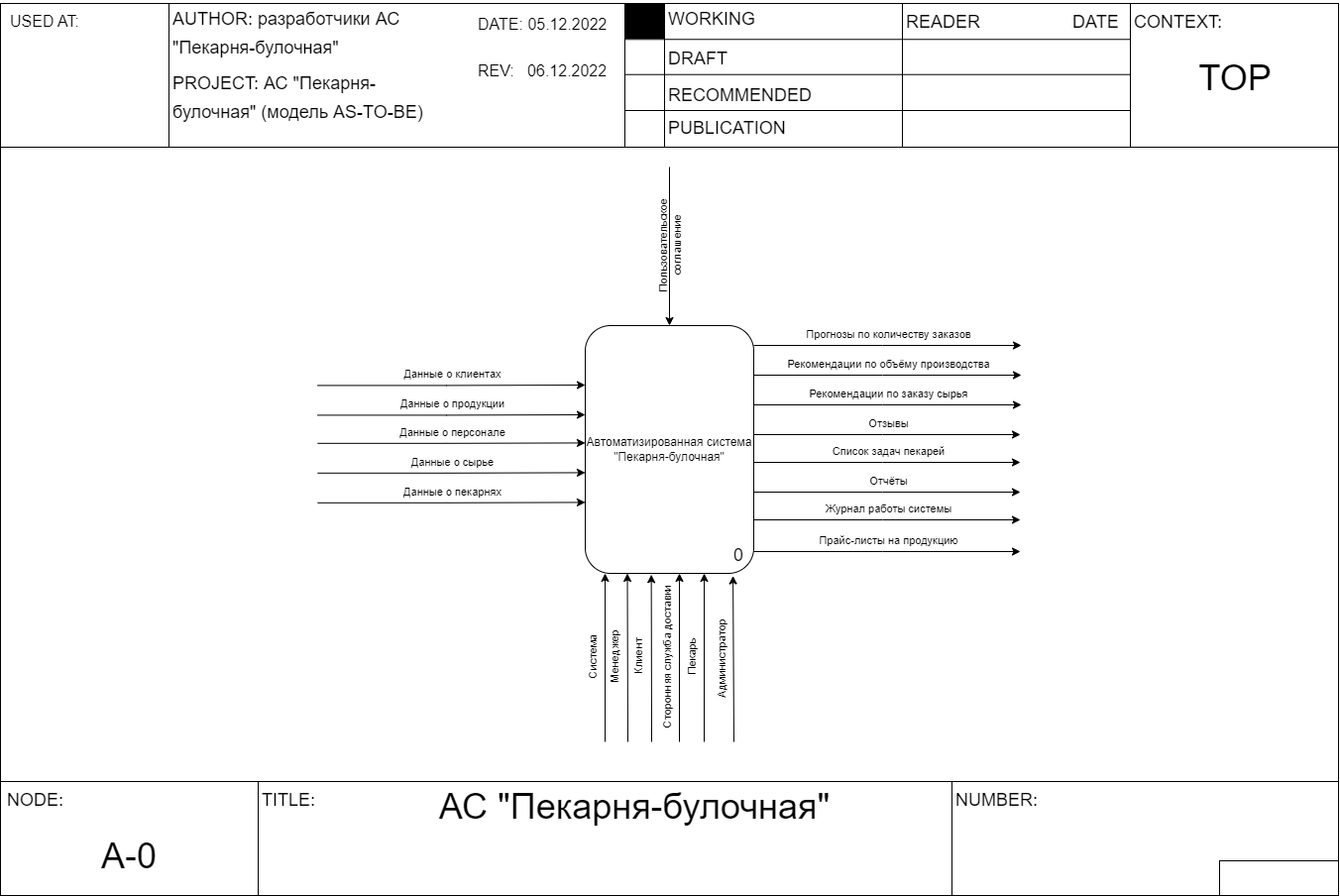


Рисунок 1. Контекстная диаграмма IDEF0

Данная диаграмма определяет границы основной задачи системы: автоматизации процессов в сети пекарен-булочных. Согласно п. 3.2 технического задания имеются 5 категорий пользователей: клиенты, сотрудники сторонней службы доставки, менеджеры, пекари и администраторы. Для возможности регистрации и авторизации пользователей системы (клиентов и сотрудников сети) предусмотрены входные данные о клиентах и о персонале. Для возможности оформления и обработки заказа предусмотрены входные данные о пекарнях. Для возможности составления прайс-листов предусмотрены входные данные о продукции и пекарнях. Также, для заказа сырья из центрального офиса, предусмотрены входные данные об имеющемся и необходимом сырье. На выходе система выдает рекомендации по объему производства продукции пекарнями, рекомендации по заказу сырья для пекарен на основе данных за предыдущие дни, список задач для пекарей, отзывы о работе сети пекарен от клиентов, отчеты, журнал работы системы, прайс-листы на продукцию и прогнозы по количеству заказов.

**2.2.2. Диаграмма IDEF0 первого уровня**

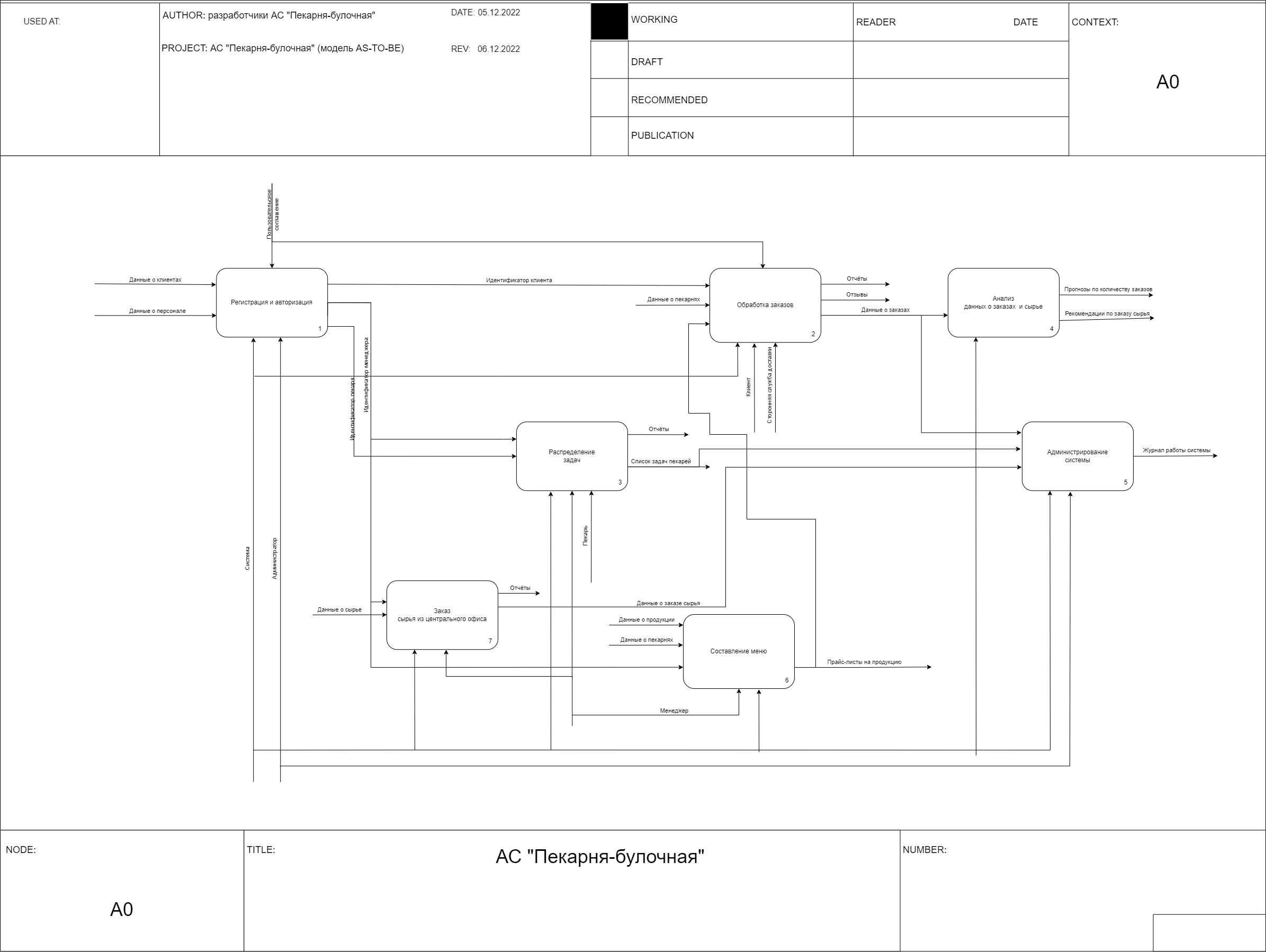


Рисунок 2. Диаграмма IDEF0 первого уровня

С помощью блока “Регистрация и авторизация” клиенты (клиентами могут быть сотрудники сторонней службы доставки) и персонал сети пекарен имеют возможность зарегистрироваться и авторизоваться в системе. Для регистрации персонала требуется, чтобы администратор выдал им соответствующие права пользователя. Таким образом, на вход подаются данные о клиенте и данные о персонале, на выходе получаются идентификаторы пользователей.

С помощью блока “Обработка заказа” клиенты могут сформировать заказа, оплатить его, а также оставить отзыв. При оплате заказа происходит перенаправление на систему оплаты, после чего данный блок получает подтверждение об оплате. Поэтому на вход подается идентификатор клиента, который оформляет заказ, данные о пекарнях и прайс-лист. На выходе получаются данные о заказах, отзывы и отчеты.

С помощью блока “Распределение задач” менеджер формирует список задач на изготовление продукции, а также распределяет задачи между пекарями. Пекари, в свою очередь, имеют возможность отметить задачу как выполненную. Поэтому на вход блока подаются идентификаторы менеджера и пекаря. На выходе имеется список задач пекарей, а также отчеты.

С помощью блока “Заказ сырья из центрального офиса” менеджер осуществляет заказ сырья из центрального офиса. Для этого на вход поступают данные о сырье, а также идентификатор менеджера. На выходе имеются данные о заказе сырья и отчеты.

С помощью блока “Анализ данных о заказах и сырье” вычисляется среднее количество заказов за определенные промежутки времени, а также среднее количество требуемого сырья. На основе этих данных составляются рекомендации для менеджера. Поэтому на вход подаются данные о заказах, на выходе получаются прогнозы по количеству заказов и рекомендации по заказу сырья.

С помощью блока “Администрирование системы” происходит запись логов обо всех действиях в системе, а также составление общего журнала. Поэтому на вход подаются данные о заказах, список задач пекарей и данные о заказе сырья. На выходе имеется журнал работы системы.

С помощью блока “Составление меню” происходит формирование прайс-листа на продукцию с возможностью установки скидок и акций на определенные позиции. Для этого на вход поступают данные о продукции и пекарнях. На выходе имеется прайс-лист на продукцию.

**2.2.3. Диаграммы IDEF0 второго уровня**



Рисунок 3. Диаграмма IDEF0 второго уровня, декомпозирующая функцию “Регистрация и авторизация”

С помощью блока «Валидация» происходит проверка данных клиентов на корректность. Таким образом, на вход подаются непроверенные данные о клиентах и персонале, а на выходе получаются уже проверенные данные.

С помощью блока «Проверка на наличие пользователя в системе» осуществляется поиск пользователя в базе данных системы, в случае, если пользователь найден, он перенаправляется на блок «Авторизация», в противном случае — на блок «Регистрация пользователя».

С помощью блока «Регистрация пользователя» происходит регистрация пользователя в системе и сохранение его данных в базу данных системы. Входными и выходными данными являются данные о пользователе.

С помощью блока «Выбор администратором роли пользователя» администратор выбирает роль новому пользователю. Изначально все пользователи являются клиентами, если необходимо, администратор может изменить роль на пекаря или менеджера. На вход блок получает данные пользователя, а на выходе получаются идентификаторы менеджера, клиента и пекаря.

С помощью блока «Авторизация» происходит выдача клиентам их прав. При этом входными и выходными данными являются идентификаторы клиентов.

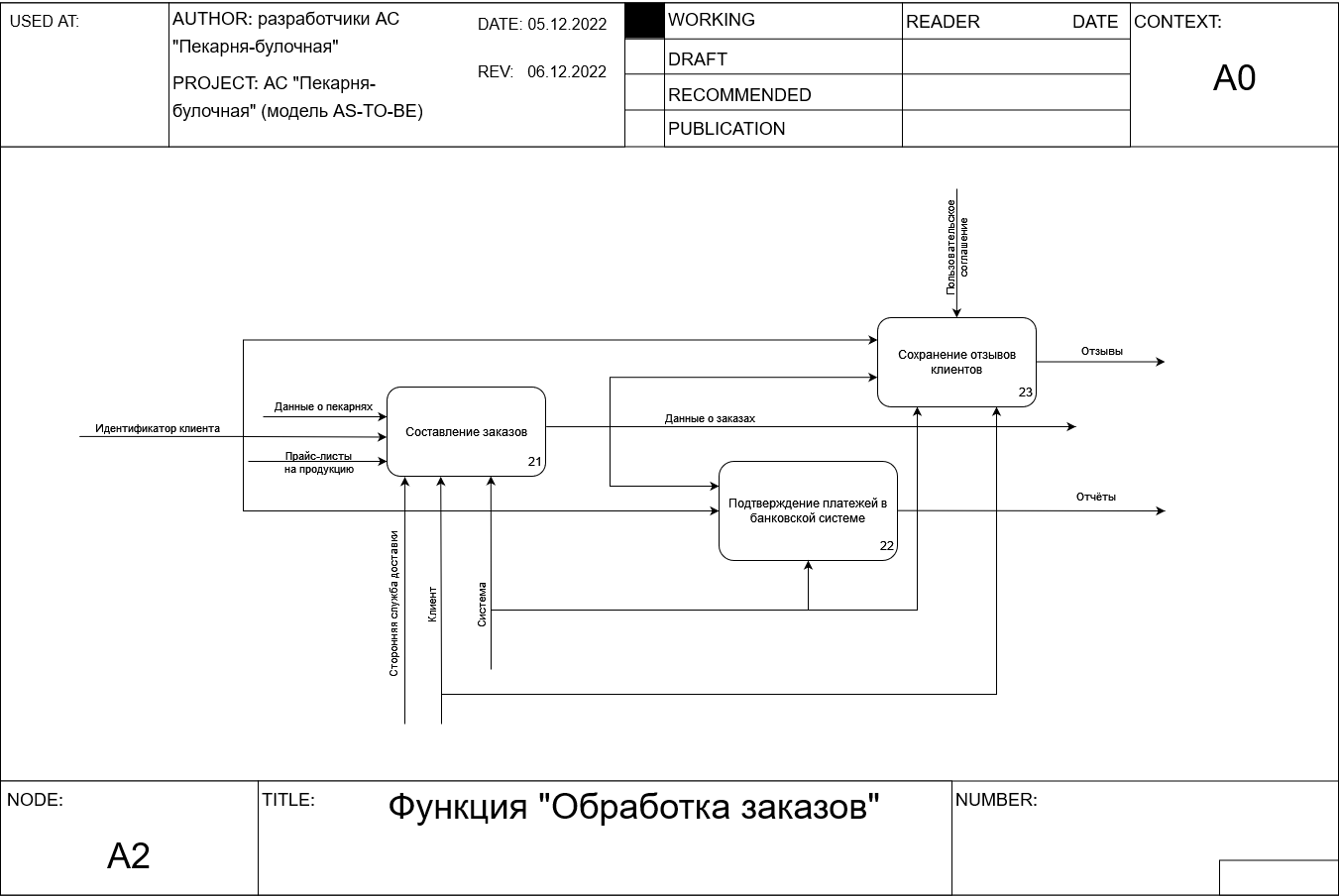


Рисунок 4. Диаграмма IDEF0 второго уровня, декомпозирующая функцию “Обработка заказов”

С помощью блока «Составление заказов» составляется список заказываемой продукции и определяется её стоимость. На вход блок получает идентификатор клиента (в качестве клиента может выступать сотрудник сторонней службы доставки), данные о пекарнях и прайс-листы на продукцию. На выход передаются данные о заказах.

С помощью блока «Подтверждение платежей в банковской системе» происходит подтверждение оплаты заказа. На вход блок получает идентификатор клиента и данные о заказах. Если оплата прошла успешно, данные о заказе сохраняются в отчёт, получаемый на выходе из блока.

С помощью блока «Сохранение отзывов клиентов» клиенты могут сохранять в системе отзывы о заказах и пекарне. На вход блока поступает идентификатор клиента и данные о заказах. На выходе из блока передаются отзывы пользователей. Сохранение отзыва происходит в соответствии с пользовательским соглашением.

Рисунок 5. Диаграмма IDEF0 второго уровня, декомпозирующая функцию “Распределение задач”

С помощью блока “Добавление задач для пекарей” происходит добавление менеджером задач пекарей в список всех задач для пекарей. Поэтому на вход функции подается идентификатор менеджера, добавляющего задачу. На выходе функции имеется список задач пекарей.

С помощью блока “Распределение задач между пекарями” происходит распределение менеджером задач между пекарями. Таким образом, на вход функции подается список задач пекарей из предыдущего блока, а также идентификатор менеджера, распределяющего задачи. На выходе функции имеются задачи для каждого пекаря.

С помощью блока “Отметка выполненных задач” происходит отметка пекарями выполненных задач. На вход функции подается список задач пекаря, а также идентификатор пекаря. На выходе функции имеется отчет о выполненных задачах.



Рисунок 6.Диаграмма IDEF0 второго уровня, декомпозирующая функцию “Анализ данных о заказах и сырье”

С помощью блока “Вычисление среднего количества заказов” вычисляется среднее количество заказа продукции за определенный период времени. На вход блок получает данные о заказах. На выходе получается вычисленное среднее количество заказов в сутки.

С помощью блока “Вычисление среднего количества заказа продукции каждого типа” вычисляется среднее количество заказанной продукции каждого типа за определенный период времени. На вход блок получает данные о заказах. На выходе получается вычисленное среднее количество продукции каждого типа в сутки.

С помощью блока “Обработка данных” происходит формирование рекомендаций по заказу сырья и объему производства. На вход блок получает среднее количество заказов и продукции в сутки, и на выходе формирует прогнозы по количеству заказов продукции и рекомендации по заказу сырья.



Рисунок 7. Диаграмма IDEF0 второго уровня, декомпозирующая функцию “Администрирование системы”

С помощью блока “Журналирование заказов продукции” происходит запись логов о заказах продукции. На вход блок получает данные о заказах. На выходе получается журнал с соответствующими записями о заказах.

С помощью блока “Журналирование выполнения задач пекарей” происходит запись логов о выполнении задач пекарей. На вход блок получает список задач пекарей. На выходе получается журнал с соответствующими записями о задачах пекарей.

С помощью блока “Журналирование заказов сырья” происходит запись логов о заказах сырья для производства. На вход блок получает данные о заказе сырья. На выходе получается журнал с соответствующими записями о заказах сырья.

С помощью блока “Экспорт системного журнала” происходит формирование общего журнала работы системы. На вход блок получает журналы работ предыдущих трех блоков и идентификатор администратора, и на выходе формирует журнал работы всей системы.

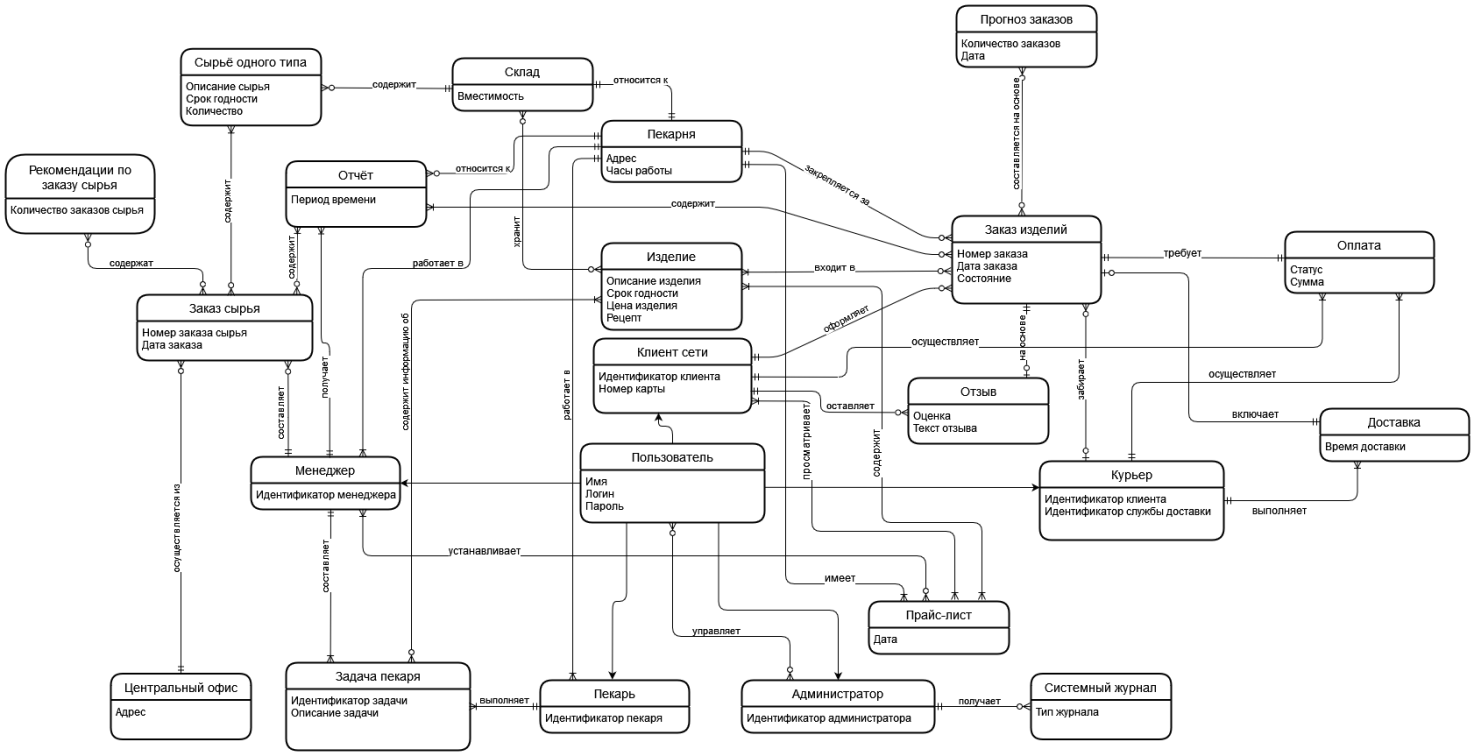
Рисунок 8 - Диаграмма IDEF0 второго уровня, декомпозирующая функцию “Составление меню”

С помощью блока “Составление прайс-листа” создается прайс-лист на продукцию на основе данных о продукции и пекарен. На вход блок получает данные о продукции и о пекарнях, на выходе получается прайс-лист на продукцию.

С помощью блока “Установление скидок и акций” на созданный ранее прайс-лист менеджер может установить различные акции и скидки или может не устанавливать их. Блок получает на вход прайс-лист на продукцию и идентификатор менеджера, на выходе получается прайс-лист на продукцию.

С помощью блока “Экспорт прайс-листа” происходит формирование итогового прайс-листа. Блок получает на вход и выдает прайс-лист на продукцию.

**2.3. Модель предметной области.**

Рисунок 9. Модель предметной области

Сущность “Пользователь” является общей: от нее наследуются сущности “Клиент сети”, “Менеджер”, “Курьер”, “Пекарь” и “Администратор”. Сущность “Пользователь” определяет персональные данные пользователя (имя), а также данные для входа в систему: логин и пароль. Сущность связана с другими следующим образом:

* “Администратор”: связь многие ко многим (администратор управляет пользователями и назначает им роли).

Сущность “Администратор” содержит информацию об администраторе системы: его идентификатор и данные, определенные в сущности “Пользователь”. Сущность связана с другими следующим образом:

* “Пользователь”: связь многие ко многим (администраторы управляют пользователями и назначают им роли);
* “Системный журнал”: связь один ко многим (администратор получает системные журналы).

Сущность “Пекарь” содержит информацию о пекаре: его идентификатор и данные, определенные в сущности “Пользователь”. Сущность связана с другими следующим образом:

* “Задача пекаря”: связь один ко многим (пекарь выполняет задачи);
* “Пекарня”: связь многие к одному (пекари работают в пекарне).

Сущность “Менеджер” содержит информацию о менеджере: его идентификатор и данные, определенные в сущности “Пользователь”. Сущность связана с другими следующим образом:

* “Пекарня”: связь многие к одному (менеджеры работают в пекарне);
* “Заказ сырья”: связь один ко многим (менеджер составляет заказы сырья);
* “Отчёт”: связь один ко многим (менеджер получает отчеты о работе пекарни);
* “Задача пекаря”: связь один ко многим (менеджер составляет задачи пекарей);
* “Прайс-лист”: связь многие ко многим (менеджер устанавливает и редактирует прайс-листы).

Сущность “Курьер” содержит информацию о курьере: его идентификатор, идентификатор службы доставки, в которой он работает, и данные, определенные в сущности “Пользователь”. Сущность связана с другими следующим образом:

* “Заказ изделий”: связь один ко многим (курьер забирает заказы);
* “Оплата”: связь один ко многим (курьер осуществляет оплату заказов);
* “Доставка”: связь один ко многим (курьер выполняет доставку заказов).

Сущность “Клиент сети” содержит информацию о клиенте сети пекарен: его идентификатор, номер банковской карты и данные, определенные в сущности “Пользователь”. Сущность связана с другими следующим образом:

* “Заказ изделий”: связь один ко многим (клиент оформляет заказы изделий);
* “Отзыв”: связь один ко многим (клиент оставляет отзывы);
* “Прайс-лист”: связь многие ко многим (клиент просматривает прайс-лист для составления заказа);
* “Оплата”: связь один ко многим (клиент осуществляет оплату заказа).

Сущность “Системный журнал” содержит данные о работе системы. Данная сущность определяется типом журнала. Сущность связана с другими следующим образом:

* “Администратор”: связь многие к одному (администратор получает системный журнал).

Сущность “Задача пекаря” содержит информацию о задаче пекаря: идентификатор задачи и описание задачи. Сущность связана с другими следующим образом:

* “Менеджер”: связь многие к одному (менеджер составляет задачи пекарей);
* “Изделие”: связь многие к одному (задача пекаря содержит информацию об изделии);
* “Пекарь”: связь многие к одному (пекарь выполняет задачи).

Сущность “Заказ сырья” содержит информацию о заказе сырья из центрального офиса: номер и дату заказа. Сущность связана с другими следующим образом:

* “Менеджер”: связь многие к одному (менеджер составляет заказы сырья);
* “Центральный офис”: связь многие к одному (заказ сырья осуществляется из центрального офиса);
* “Рекомендации по заказу сырья”: связь многие ко многим (рекомендации содержат заказы сырья);
* “Сырье одного типа”: связь многие ко многим (заказ сырья содержит сырье разных типов);
* “Отчёт”: связь многие ко многим (отчет содержит информацию о заказах сырья).

Сущность “Центральный офис” содержит информацию об офисе, такую как его адрес. Сущность связана с другими следующим образом:

* “Заказ сырья”: связь один ко многим одному (заказ сырья осуществляется из центрального офиса).

Сущность “Рекомендации по заказу сырья” содержит информацию о количестве заказов сырья. Сущность связана с другими следующим образом:

* “Заказ сырья”: связь многие ко многим (рекомендации содержат заказы сырья).

Сущность “Сырье одного типа” содержит информацию о сырье на складе. Сущность связана с другими следующим образом:

* “Заказ сырья”: связь многие ко многим (заказ сырья содержит сырье разных типов);
* “Склад”: связь многие к одному (склад содержит сырье различных типов).

Сущность “Отчет” содержит информацию о заказах изделий и заказах сырья за определенный период времени. Содержит информацию о периоде времени. Сущность связана с другими следующим образом:

* “Менеджер”: связь многие ко многим (менеджер получает отчеты);
* “Заказ сырья”: связь многие ко многим (отчет содержит информацию о заказах сырья);
* “Пекарня”: связь многие к одному (отчёт относится к пекарне);
* “Заказ изделий”: связь многие ко многим (отчет содержит информацию о заказах изделий).

Сущность “Пекарня” содержит информацию о пекарне, такую как ее адрес и часы ее работы. Сущность связана с другими следующим образом:

* “Заказ изделий”: связь один ко многим (заказ закрепляется за пекарней);
* “Склад”: связь один к одному (склад относится к пекарне);
* “Отчёт”: связь один ко многим (отчёт относится к пекарне);
* “Менеджер”: связь один ко многим (менеджер работает в пекарне);
* “Пекарь”: связь один ко многим (пекарь работает в пекарне);
* “Прайс-лист”: связь один ко многим (пекарня имеет прайс-листы на различные дни).

Сущность “Склад” содержит информацию о складе пекарни, такую как его вместимость. Сущность связана с другими следующим образом:

* “Пекарня”: связь один к одному (склад относится к пекарне);
* “Сырье одного типа”: связь один ко многим (склад содержит сырье различных типов).

Сущность “Изделие” содержит информацию об изделии пекарни, такую как описание изделия, срок годности, цена изделия и рецепт. Сущность связана с другими следующим образом:

* “Заказ изделий”: связь многие ко многим (изделия входят в заказы);
* “Склад”: связь многие ко многим (склад хранит изделие);
* “Задача пекаря”: связь многие ко многим (задача пекаря содержит информацию об изделии);
* “Прайс-лист”: связь многие ко многим (прайс-листы содержат цены на изделия).

Сущность “Прайс-лист” содержит информацию о меню и ценах на продукцию, такую как дата, в которую будет действовать прайс-лист. Сущность связана с другими следующим образом:

* “Пекарня”: связь многие к одному (пекарня имеет прайс-листы на различные дни);
* “Менеджер”: связь многие ко многим (менеджер устанавливает и редактирует прайс-листы);
* “Клиент сети”: связь многие ко многим (клиент просматривает прайс-лист для составления заказа);
* “Изделие”: связь многие ко многим (прайс-листы содержат цены на изделия).

Сущность “Заказ изделий” содержит информацию о заказе изделий пекарни, такую как номер заказа, дата заказа и состояние заказа (оформлен, оплачен, выдан, доставлен). Сущность связана с другими следующим образом:

* “Изделие”: связь многие ко многим (изделия входят в заказы);
* “Пекарня”: связь многие к одному (заказ закрепляется за пекарней);
* “Отчет”: связь многие ко многим (отчет содержит информацию о заказах изделий);
* “Клиент сети”: связь многие к одному (клиент оформляет заказы);
* “Отзыв”: связь один к одному (отзыв составляется на основе заказа);
* “Прогноз заказов”: связь многие ко многим (прогнозы составляются на основе заказов);
* “Курьер”: связь многие к одному (курьер забирает заказ);
* “Оплата”: связь один к одному (заказ требует оплату);
* “Доставка”: связь один к одному (заказ может включать доставку).

Сущность “Прогноз заказов” содержит информацию о прогнозе заказов пекарни, такую как количество заказов и дата, на которую составляется прогноз. Прогноз составляется на основе информации о текущих заказах. Сущность связана с другими следующим образом:

* “Заказ изделий”: связь многие ко многим (прогнозы составляются на основе заказов).

Сущность “Отзыв” содержит информацию об отзыве клиента о заказе, такую как оценка заказа и текст отзыва. Сущность связана с другими следующим образом:

* “Заказ изделий”: связь один к одному (отзыв составляется на основе заказа);
* “Клиент сети”: связь многие к одному (клиент оставляет отзывы).

Сущность “Оплата” содержит информацию об оплате заказа, такую как оценка статус оплаты и сумма оплаты. Сущность связана с другими следующим образом:

* “Заказ изделий”: связь один к одному (заказ требует оплату)
* “Клиент сети”: связь многие к одному (клиент осуществляет оплату заказа);
* “Курьер”: связь многие к одному (курьер осуществляет оплату заказа).

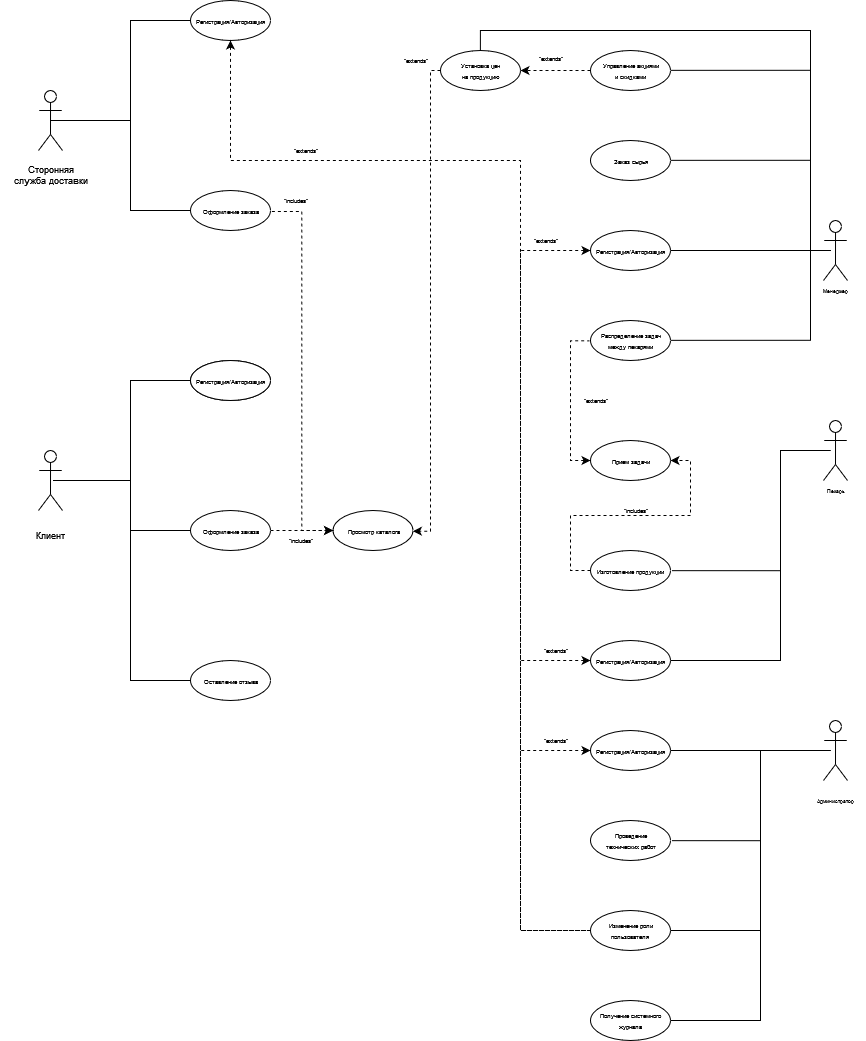
Сущность “Доставка” содержит информацию о доставке заказа курьером, такую как предполагаемое время доставки. Сущность связана с другими следующим образом:

* “Заказ изделий”: связь один к одному (заказ может включать доставку);
* “Курьер”: связь многие к одному (курьер выполняет доставку заказов).

**3. основные технологические решения**

**3.1. Диаграммы Use Case.**

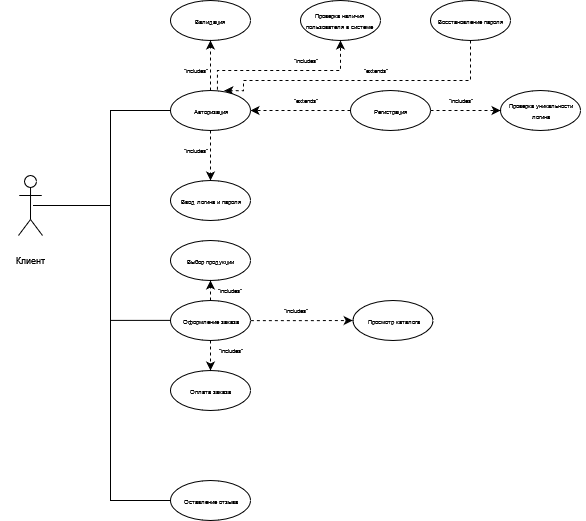
На рисунке 9 представлена диаграмма Use Case для АС «Пекарня-булочная».

Рисунок 9 — Use Case диаграмма АС «Пекарня-Булочная»

На рисунке 10 представлена диаграмма для клиента. Клиент для взаимодействия с приложением должен пройти авторизацию. Если он пользуется приложением впервые, то необходимо зарегистрироваться при этом логин должен быть уникальным. В ином случае производится ввод логина и пароля. Если же пароль был забыт, то его можно восстановить.

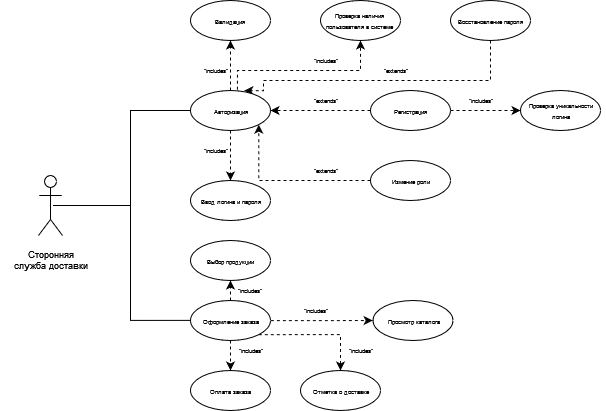
Клиент имеет возможность оформить заказ. Для этого необходимо посмотреть каталог, выбрать интересующую продукцию и оплатить заказ.

Клиент также имеет возможность оставить отзыв о работе пекарни.

Рисунок 10 — Use Case диаграмма для клиента

На рисунке 11 представлена диаграмма для сторонней службы доставки. Сторонняя служба доставки для взаимодействия с приложением должна пройти авторизацию. Если используется приложение впервые, то необходимо зарегистрироваться при этом логин должен быть уникальным. В ином случае производится ввод логина и пароля. Если же пароль был забыт, то его можно восстановить. Изменить роль пользователя на стороннюю службу доставки может админ системы.

Сторонняя служба доставки имеет возможность оформить заказ. Для этого необходимо посмотреть каталог, выбрать интересующую продукцию и оплатить заказ. После доставки продукции требуется отметить заказ выполненным.

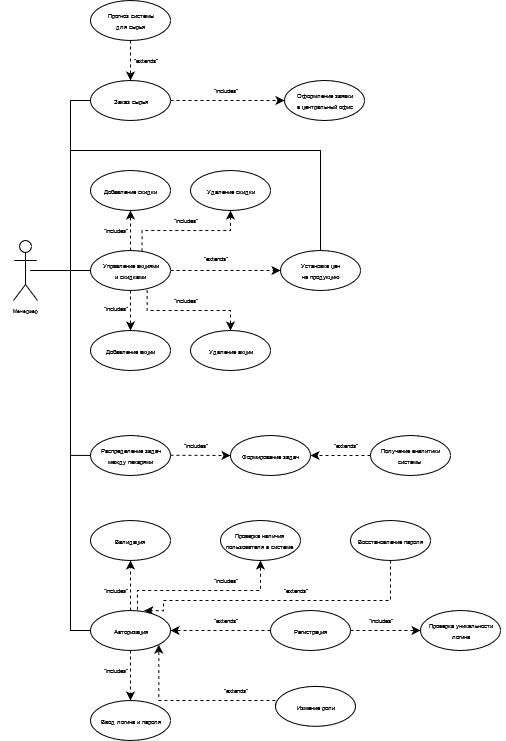
Рисунок 11 — Use Case диаграмма для сторонней службы доставки

На рисунке 12 представлена Use Case диаграмма для менеджера. Менеджер для взаимодействия с приложением должен пройти авторизацию. Если он пользуется приложением впервые, то необходимо зарегистрироваться при этом логин должен быть уникальным. В ином случае производится ввод логина и пароля. Если же пароль был забыт, то его можно восстановить. Изменить роль пользователя на менеджера может админ системы.

У него есть возможность выполнить заказ сырья с учетом прогноза системы для сырья. Для этого необходимо оформить заявку в центральный офис.

Менеджер может оказывать влияние на цены двумя способами: первый — напрямую за счет установки цены на товар из каталога, второй — косвенно за счет акций и скидок, которые он может добавить или удалить.

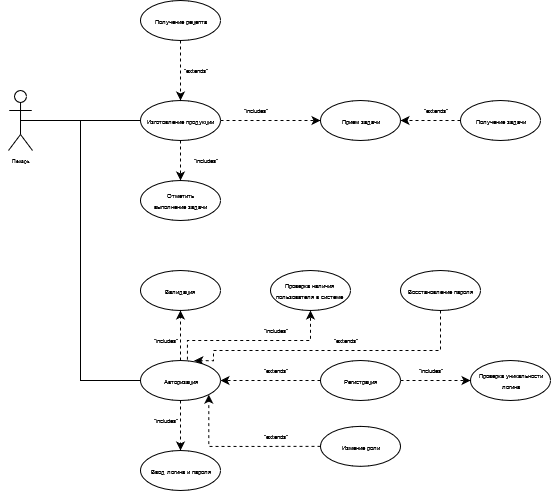
Также менеджер распределяет задачи между пекарями. Для этого он формирует задачи с учетом полученной аналитики от системы.

Рисунок 12 — Use Case диаграмма для менеджера

На рисунке 13 представлена диаграмма для пекаря. Пекарь для взаимодействия с приложением должен пройти авторизацию. Если он пользуется приложением впервые, то необходимо зарегистрироваться при этом логин должен быть уникальным. В ином случае производится ввод логина и пароля. Если же пароль был забыт, то его можно восстановить. Изменить роль пользователя на пекаря может админ системы.

Его основная задача — изготовление продукции на основе задачи, которую сформировал менеджер. После этого ему необходимо отметить задачу выполненной.

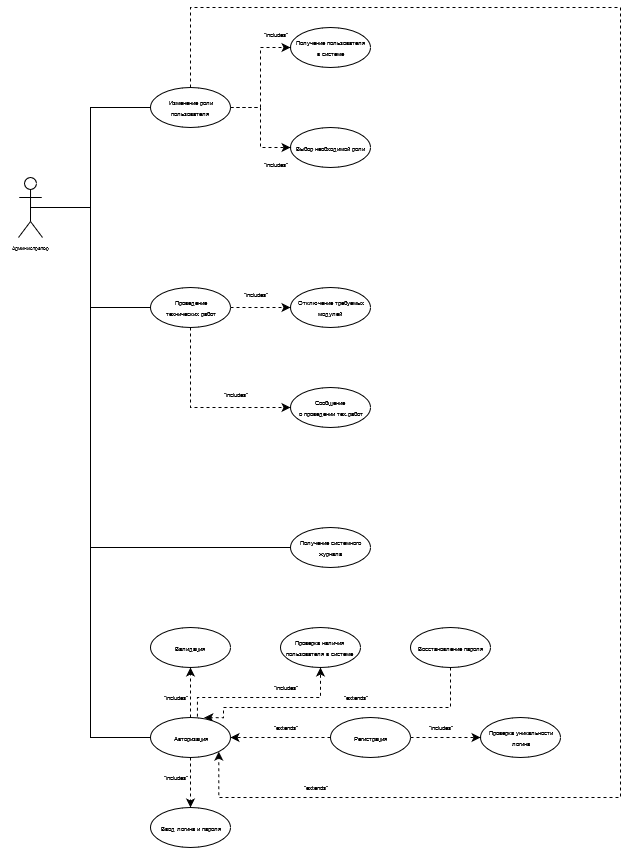
Пекарь может посмотреть рецепт при необходимости.

Рисунок 13 — Use Case диаграмма для пекаря

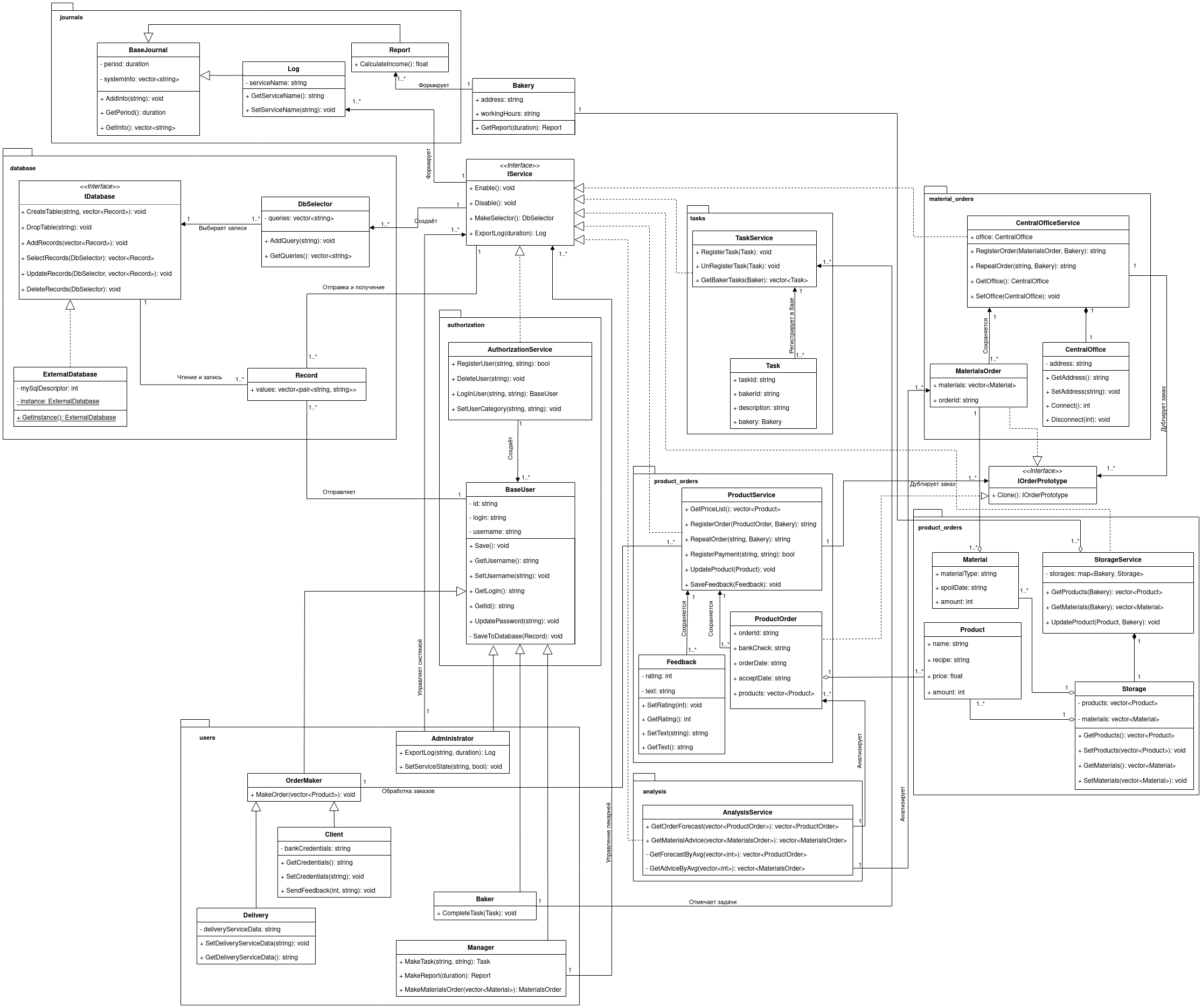
На рисунке 14 представлена Use Case диаграмма для администратора. Администратор идет вместе с приложением. Администратор для взаимодействия с приложением должен пройти авторизацию. Если он пользуется приложением впервые, то необходимо зарегистрироваться при этом логин должен быть уникальным. В ином случае производится ввод логина и пароля. Если же пароль был забыт, то его можно восстановить. Изменить роль пользователя на администратора может любой админ системы.

Он имеет возможность изменять роль пользователя за счет нахождения пользователя в системе и выбора необходимой роли.

Также администратор отвечает за проведение технических работ, которые включают в себя оповещение о проведении, а также отключение требуемых модулей на время.

Рисунок 14 — Use Case диаграмма для администратора

**3.2. Диаграмма классов**

Рисунок 15. Диаграмма классов

**3.2.1. Описание сущностей на диаграмме классов**

* IService — интерфейс сервиса приложения
  + Enable(): void — метод включения сервиса.
  + Disable(): void — метод отключения сервиса.
  + MakeSelector(): DbSelector — метод создания выборки из базы данных.
  + ExportLog(duration): Log — метод создания журнала работы сервиса.
* AuthorizationService — сервис авторизации пользователей системы
  + RegisterUser(string, string): bool — метод принимает на вход логин и пароль пользователя и регистрирует его в системе. Если регистрация не удалась, метод возвращает false.
  + DeleteUser(string): void — метод принимает на вход. идентификатор пользователя и удаляет его из системы.
  + LogInUser(string, string): BaseUser — метод принимает на вход логин и пароль пользователя и возвращает сущность пользователя при успешном входе.
  + SetUserCategory(string, string): void — метод принимает идентификатор пользователя и желаемую категорию, после чего добавляет пользователя в указанную категорию.
* BaseUser — абстрактный класс пользователя системы
  + id: string — идентификатор пользователя.
  + login: string — логин пользователя.
  + username: string — имя пользователя.
  + Save(): void — сохранение изменений в модели пользователя.
  + GetUsername(): string — метод получения имени пользователя.
  + SetUsername(string): void — метод установки имени пользователя.
  + GetLogin(): string — метод получения логина пользователя.
  + GetId(): string — метод получения идентификатора пользователя.
  + UpdatePassword(string): void — метод обновления пароля пользователя. Пароль не хранится в сущности пользователя, только в базе данных.
  + SaveToDatabase(Record): void — метод сохранения данных о сущности в базе данных.
* IDatabase — интерфейс базы данных
  + CreateTable(string, vector<Record>): void — метод создания таблицы на основе списка записей.
  + DropTable(string): void — метод удаления таблицы из базы данных.
  + AddRecords(vector<Record>): void — метод добавления записей.
  + SelectRecords(DbSelector): vector<Record> - метод извлечения записей из базы данных. Принимает объект запроса к базе данных.
  + UpdateRecords(DbSelector, vector<Record>): void — метод обновления записей в базе данных. Получает на вход объект запроса к базе данных и обновляемые записи.
  + DeleteRecords(DbSelector): void — метод удаления записей из базы данных.
* ExternalDatabase — класс подключения к базе данных. Реализует паттерн «Одиночка».
  + mySqlDescriptor: int — дескриптор подключения к базе данных.
  + instance: ExternalDatabase — объект подключения. Данный объект существует в единственном экземпляре, что позволяет реализовать паттерн «Одиночка».
  + GetInstance(): ExternalDatabase — метод получения объекта подключения к базе данных.
* DbSelector — класс запроса к базе данных. Реализует паттерн «Команда».
  + queries: vector<string> — список запросов к базе данных. Позволяет объекту базы данных исполнять запросы, ничего не зная об их структуре.
  + AddQuery(string): void — метод добавления нового запроса к базе данных.
  + GetQueries(): vector<string> — метод получения запросов к базе данных из текущей выборки.
* Record — класс записи в базе данных
  + values: vector<pair<string, string>> - список пар поле:значение, отвечающий полям записи в базе данных.
* Administrator — класс администратора системы
  + ExportLog(string, duration): Log — метод, принимающий на вход название системы и промежуток времени, возвращает отчет о работе указанной системы за указанный промежуток времени.
  + SetServiceState(string, bool): void — метод включения/отключения определенного сервиса системы.
* Baker — класс пекаря
  + CompleteTask(Task): void — метод закрытия задачи пекаря.
* OrderMaker — класс пользователя, который может осуществлять заказ в сети пекарен
  + MakeOrder(vector<Product>): void — метод оформления заказа чере систему.
* Client — класс клиента сети пекарен
  + bankCredentials: string — банковские данные клиента (номер карты).
  + GetCredentials(): string — метод получения банковских данных клиента.
  + SetCredentials(string): void — метод установки банковских данных клиента.
  + SendFeedback(int, string): void — метод добавления отзыва о заказе.
* Delivery — класс курьера сторонней службы доставки
  + deliveryServiceData: string — данные о службе доставки, в которой работает курьер.
  + SetDeliveryServiceData(string): void — метод обновления данных о службе доставки курьера.
  + GetDelivertServiceData(): string — метод получения данных о службе доставки курьера.
* Manager — класс менеджера пекарни
  + MakeTask(string, string): Task — метод добавления задачи пекаря. Принимает идентификатор пекаря и описание задачи.
  + MakeReport(duration): Report — метод получения менеджером отчета о работе пекарни.
  + MakeMaterialsOrder(vector<Material>): MaterialsOrder — метод оформления заказа сырья из центрального офиса.
* Bakery — класс пекарни
  + address: string — адрес расположения пекарни.
  + workingHours: string — часы работы пекарни.
  + GetReport(duration): Report — метод получения отчета о работе пекарни за заданной промежуток времени.
* BaseJournal — абстрактный класс отчета/журнала о работе сервиса/пекарни
  + period: duration — промежуток времени, за который составляется журнал.
  + systemInfo: vector<string> - текстовое содержимое журнала.
  + AddInfo(string): void — добавление содержимого в журнал.
  + GetPeriod(): duration — получение промежутка времени, за который составляется журнал.
  + GetInfo(): vector<string> — получение текстового содержимого журнала
* Log — класс журнала о работе системы
  + serviceName: string — название системы, по которой составляется журнал.
  + GetServiceName(): string — получение названия системы.
  + SetServiceName(string): void — получение названия системы.
* Report — класс отчета о работе пекарни
  + CalculateIncome(): float — метод подсчета прибыли пекарни за заданный промежуток времени.
* TaskService — класс сервиса задач пекарей
  + RegisterTask(Task): void — метод добавления задачи пекаря.
  + UnRegisterTask(Task): void — метод удаления задачи пекаря.
  + GetBakerTasks(Baker): vector<Task> — метод получения всех задач определенного пекаря.
* Task — класс задачи пекаря
  + taskId — идентификатор задачи.
  + bakerId — идентификатор пекаря, которому поручена задача.
  + description — описание задачи.
  + bakery: Bakery — пекарня, в которой выполняется данная задача.
* ProductService — класс сервиса заказов продукции
  + GetPriceList(): vector<Product> - метод получения списка продукции, включая цены на продукцию.
  + RegisterOrder(ProductOrder, Bakery): string — метод оформления заказа продукции. Принимает на вход составленный заказ, а также пекарню, в которой происходит заказ. Метод возвращает идентификатор заказа.
  + RepeatOrder(string, Bakery): string — метод повторного оформления заказа продукции. Принимает на вход идентификатор заказа, который необходимо повторить, а также пекарню, в которой происходит заказ. Метод возвращает идентификатор нового заказа.
  + RegisterPayment(string, string): bool — метод подтверждения оплаты заказа продукции. Принимает на вход идентификатор заказа и данные об оплате. Возвращает true в случае успешной оплаты, иначе — false.
  + UpdateProduct(Product): void — метод обновления информации об изделии.
  + SaveFeedback(Feedback): void — метод добавления отзыва клиента о заказе продукции пекарни.
* IOrderPrototype — интерфейс заказа. Реализует паттерн «Прототип».
  + Clone(): IOrderPrototype — метод создания копии сущности с новым идентификатором.
* ProductOrder — класс заказа продукции пекарни
  + orderId: string — идентификатор заказа.
  + bankCheck: string — банковские данные об оплате.
  + orderDate: string — дата оформления заказа.
  + acceptDate: string — дата регистрации и оплаты заказа.
  + products: vector<Product> — список продукции из заказа.
* MaterialsOrder — класс заказа сырья из центрального офиса
  + orderId: string — идентификатор заказа.
  + materials: vector<Material> — список сырья в заказе.
* Feedback — класс отзыва о заказе
  + rating: int — оценка заказа по пятибальной шкале.
  + text: string — текст отзыва.
  + SetRating(int): void — метод изменения оценки.
  + GetRating(): int — метод получения оценки.
  + SetText(string): void — метод установки текста отзыва.
  + GetText(): string — метод получения текста отзыва.
* CentralOfficeService — класс, отвечающий за работу центрального офиса сети пекарен
  + office: CentralOffice — объект офиса.
  + RegisterOrder(MaterialsOrder, Bakery): string — метод оформления заказа сырья в центральном офисе. Принимает на вход заказ сырья и пекарню, для которой сделан заказ. Возвращает идентификатор заказа.
  + RepeatOrder(string, Bakery): string — метод повтора заказа сырья в центральном офисе. Принимает на вход идентификатор заказа и пекарню, для которой сделан заказ. Возвращает идентификатор нового заказа.
  + GetOffice(): CentralOffice — метод получения объекта офиса.
  + SetOffice(CentralOffice): void — метод изменения объекта офиса.
* CentralOffice — класс центрального офиса сети пекарен
  + address: string — адреса офиса.
  + GetAddress(): string — метод получения адреса офиса.
  + SetAddress(string): void — метод изменения адреса офиса.
  + Connect(): int — метод подключения к сервису данного офиса. Возвращает дескриптор соединения с сервисом офиса.
  + Disconnect(int): void — метод отключения от сервиса офиса.
* StorageService — класс сервиса складов сети пекарен
  + storages: map<Bakery, Storage> — словарь, в котором каждой пекарне соответсвует ее склад.
  + GetProducts(Bakery): vector<Product> — метод получения списка изделий со склада пекарни.
  + GetMaterials(Bakery): vector<Material> — метод получения списка сырья со склада пекарни.
  + UpdateProduct(Product, Bakery): void — метод обновления информации о товаре на складе. Позволяет устанавливать различные цены на изделия для пекарен.
* Storage — класс склада пекарни
  + products: vector<Product> — список изделий на складе.
  + materials: vector<Material> — список сырья на складе.
  + GetProducts(): vector<Product> — метод получения списка изделий на складе.
  + SetProducts(vector<Product>): void — метод изменения списка изделий на складе.
  + GetMaterials(): vector<Material> — метод получения списка сырья на складе.
  + SetMaterials(vector<Material>): void — метод изменения списка сырья на складе.
* Product — класс изделия пекарни на складе/в заказе
  + name: string — название изделия.
  + recipe: string — рецепт изделия.
  + price: float — цена изделия.
  + amount: int — количество изделий.
* Material — класс сырья на складе
  + materialType: string — тип сырья.
  + spoilDate: string — срок годности сырья.
  + amount: int — количество сырья.
* AnalysisService — класс сервиса анализа заказов
  + GetOrderForecast(vector<ProductOrder>): vector<ProductOrder> — метод получения прогноза по количеству заказов изделий на основе предыдущих заказов.
  + GetMaterialAdvice(vector<MaterialsOrder>): vector<MaterialsOrder> — метод получения рекомендации по заказу сырья на основе предыдущих заказов сырья.
  + GetForecastByAvg(vector<int>): vector<ProductOrder> — метод получения прогноза по количеству заказов изделий путем подсчета среднего количества заказов определенных изделий.
  + GetAdviceByAvg(vector<int>): vector<MaterialsOrder> — метод получения рекомендации заказу сырья путем подсчета среднего количества заказов определенных типов сырья.

**3.2.2. Описание примененных паттернов проектирования**

При разработке системы были использованы такие паттерны проектирования, как «Одиночка», «Команда», «Прототип».

Паттерн «Одиночка» гарантирует, что существует единственный экземпляр класса в памяти. Это позволяет избежать случайного создания нескольких сущностей одного класса, а также позволяет не передавать экземпляр класса всем использующим его сущностям: вместо этого доступ к экземпляру может быть легко получен в любом месте программы. Паттерн «Одиночка» реализован в классе ExternalDatabase, так как приложению требуется единственный экземпляр соединения с базой данных.

Паттерн «Команда» используется для инкапсуляции информации, необходимой для выполнения действия в более позднее время. Это позволяет вызывающему объекту ничего не знать о конкретной команде, только об интерфейсе команды. Паттерн «Команда» реализован в классе DbSelector, что позволяет выполнять запросы к базе данных, при этом объект соединения с базой данных ничего не знает о запросах.

Паттерн «Прототип» позволяет задавать виды создаваемых объектов через экземпляр-прототип и создавать новые объекты путем копирования прототипа. В системе в качестве интерфейса прототипа выступает интерфейс IOrderPrototype. Классы ProductOrder и MaterialsOrder реализуют данный интерфейс. Это позволяет создавать копии заказов (т. е. делать повторный заказ), не задавая данные заказа с нуля.