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

Институт математики, физики и информационных технологий
(наименование института полностью)

Кафедра «Прикладная математика и информатика»
(наименование)

09.04.03 Прикладная информатика
(код и наименование направления подготовки)

Информационные системы и технологии корпоративного управления
(направленность (профиль))

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ)

на тему «Распределенная информационная система сети общественного питания»

Студент Е.В. Масленников _____
(И.О. Фамилия) (личная подпись)

Руководитель к.т.н., доцент, А.В. Очеповский
(ученая степень, звание, И.О. Фамилия)

Тольятти 2020

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

В современном мире активно развиваются различные информационные технологии. Одной из предметных областей, где постоянно применяются подобные технологии, являются области, где требуется какой-либо учет. Особенно сложной является проблема организации учета на территориально-распределенных производственных объектах, где требуется интеграция различных информационных систем, распределенных в пространстве. В качестве примера такого объекта можно привести организацию общественного питания. Сеть кофейных заведений – вид деятельности, который относится к ведению бизнеса, требующего учета приобретаемого пищевого материала и учета продаж. Данный учет осложнен большим количеством географически распределенных объектов: самих заведений, их поставщиков и работников, и используемых ими учетных систем.

Для решения таких задач учета часто требуется использовать ту или иную распределенную базу данных, управляемую системой управления базами данных (СУБД). В базе данных можно хранить всю необходимую информацию. Это могут быть наименования поставщиков продуктов, адреса, телефоны, фотографии продукции, в базах данных можно вести учет платежей за приобретенную продукцию. Для этого в базе данных существуют таблицы. Непосредственно заполнять таблицы достаточно неудобно. Особенно для конечного пользователя, который далек от администрирования СУБД.

Конечно, непосредственно в СУБД тоже можно создавать формы для ввода данных, запросы и отчеты. Однако, от конечного пользователя это будет требовать более глубоких знаний основ работы с персональным компьютером. Вторым минусом такого подхода будет являться то, что у конечного пользователя должна быть установлена СУБД в рамках которой

будет функционировать база данных. Это накладывает дополнительную нагрузку на бюджет юридического лица, а так же на технические знания пользователя. Поэтому наиболее предпочтительным будет являться база данных, управлять которой будет стороннее приложение, которое может быть написано для решения задач конкретного предприятия.

Поскольку речь идет о сети предприятий, выполняющих одну и ту же функцию, то рационально будет использование не локальной учетной информационной системы, а распределенной информационной системы, выполненной на базе технологии клиент-сервер.

При этом применение коробочных продуктов, таких как ERP-системы нецелесообразно по причине несоответствия размеру бизнеса и затратами на интеграцию и сопровождение. В данном случае экономически целесообразным является разработка ПО под нужды конкретного предприятия, список бизнес-процессов которого ограничен и не имеет ожидаемых изменений в будущем.

Актуальность темы заключается в сложности организации учета на территориально-распределенных производственных объектах, где требуется интеграция различных информационных систем, распределенных в пространстве. Предметная область сети кофейных заведений подходит в качестве примера исследования указанной темы.

Объектом исследования будет выступать распределенная информационная система, построенная на базе двухуровневой архитектуры клиент-сервер, решающая задачи организации приема и учета заказов в различных точках сети кофейных заведений с использованием единой базы данных.

Субъектом исследования будет являться распределенная информационная система сети кофейных заведений.

Предметом исследования будет являться организация распределенной системы учета поступающих продуктов от поставщиков и учет продаж

продуктов клиентам в различных точках продаж с применением технологии клиент-сервер.

Методы исследования: системный анализ, концептуальный анализ, моделирование, методы построения распределенных информационных систем, общенаучные методы: анализ и синтез, индукция и дедукция.

Цель исследования заключается в исследовании профессионального применения технологий распределенных информационных систем для организации учета на географически удаленных производственных объектах.

Задачами исследования являются:

- исследование истории, назначения и области применения распределенных информационных систем;
- описание бизнес-процессов предприятия, постановка задач;
- моделирование и декомпозиция бизнес-процессов, информационных потоков;
- обоснование архитектуры распределённого приложения;
- проектирование распределенной базы данных, создание таблиц и определение связей между ними;
- создание пользовательского интерфейса работы с базами данных (приложения-клиента);
- организация выборок из базы данных;
- создание отчетов с возможностью вывода на печать.

Публикации по теме исследования. Основные результаты теоретической части исследования изложены в статье: Масленников Е. В. Распределенные информационные системы: особенности применения и построения. Молодой ученый. — 2019. — №22. — С. 59-61.

На защиту выносятся:

1. Результаты исследования области применения распределенных ИС и принципов их построения с обоснованием выбора архитектуры «клиент-сервер» для решения поставленной задачи

2. Распределенная информационная система учета для сети кофеен и результаты тестирования решения

Работа состоит из четырех глав. Первая глава является теоретической и описывает назначение и характеристику распределенных информационных систем.

Вторая глава, также является теоретической и описывает основные подходы и методы построения распределенных информационных систем, обосновывает применение архитектуры «клиент-сервер» для построения РИС в рамках исследуемой предметной области (сеть кофейных заведений).

Третья глава носит практический (технологический) характер. В этой главе описываются все этапы разработки распределенной базы данных и проектирование форм и отчетов.

Третья глава является технологической и носит практический характер. В ней обосновываются выбранные программные средства для построения системы, а так же описывается само построение базы данных и клиентского программного обеспечения.

Последняя глава является тестирующей и апробирующей результат разработанной распределенной информационной системы.

Работа изложена на 104 страницах и включает 61 рисунок, 2 таблицы, 61 источник.

ГЛАВА 1 АНАЛИЗ НАЗНАЧЕНИЯ И ОБЛАСТИ ПРИМЕНЕНИЯ РАСПРЕДЕЛЕННЫХ ИНФОРМАЦИОННЫХ СИСТЕМ

1.1 Понятие распределенных информационных систем

Распределенной информационной системой можно назвать систему, в которой осуществляется распределение функций, ресурсов между множеством элементов (узлов) и для которой свойственно отсутствие единого управляющего центра, поэтому выход из строя одного из узлов не должен привести к неработоспособности всей системы. Если рассматривать понятие в узком смысле, то распределенная информационная система с позиции программного обеспечения представляет собой совокупность взаимодействующих друг с другом программных компонентов. Каждая из которых, в свою очередь, может рассматриваться как программный модуль, исполняемый в рамках отдельного процесса. Обмен информацией выполняется согласно правилам, определенных централизованно. Таким образом, распределённая ИС — система, для которой отношения местоположений элементов играют ключевую роль с точки зрения функционирования системы, а, следовательно, и с точки зрения анализа и синтеза системы. Распределённые информационные системы разделяют на[1]:

- файл-серверные информационные системы (информационные системы с архитектурой «файл-сервер»);

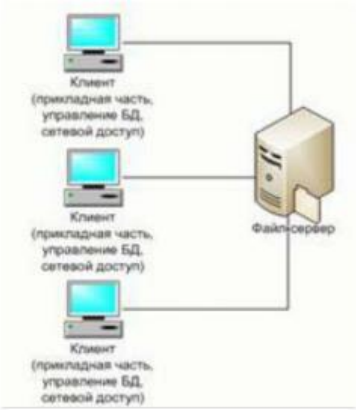


Рисунок 1.1 – Классическое представление архитектуры «файл-сервер»

[1] Радченко, Г.И. Распределенные вычислительные системы: учебное пособие / Г.И. Радченко. - Челябинск: Фотохудожник, 2012. - 184 с.

- клиент-серверные информационные системы (информационные системы с архитектурой «клиент-сервер»)

Рисунок 1.2 – Двухуровневая архитектура «клиент-сервер»

- пиринговые (одноранговые), и вытекающие из них технологии, такие как torrent, blockchain[2] и т.п.

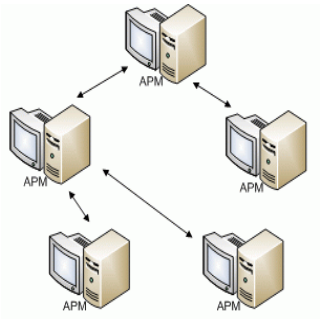


Рисунок 1.3 – Одноранговая РИС

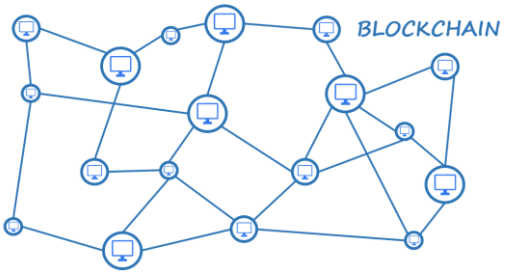


Рисунок 1.4 – РИС на базе архитектуры «Blockchain»

[2] Фомин О.В. Содержание технологии «блокчейн» и характеристика основных участников системы // научное сообщество студентов: междисциплинарные исследования: сб. ст. по мат. XLVI междунар. студ. науч.-практ. конф. № 11(46).

В файл-серверных информационных системах база данных находится на файловом сервере, а СУБД и клиентские приложения находятся на функционирующих станциях.

В клиент-серверных информационных системах база данных и СУБД находятся на сервере, а на рабочих станциях находятся только клиентские приложения. В свою очередь, клиент-серверные ИС разделяют на двухзвенные и многозвенные.

В двухзвенных информационных системах всего два типа «звеньев»: сервер базы данных (на котором находятся БД и СУБД), и рабочие станции (на которых находятся клиентские приложения). Клиентские приложения обращаются к СУБД напрямую.

В многозвенных информационных системах добавляются промежуточные «звенья»: серверы приложений. Пользовательские клиентские приложения не обращаются к системам управления базами данных напрямую, а взаимодействуют с промежуточными звеньями.

Классическим примером использования трёхзвенной архитектуры являются современные Web-приложения, которые используют базы данных.

Организация РИС необходима для предприятий и организаций, которые занимаются различными видами распределенной в пространстве производственной деятельности. Это делается с целью упрощения дальнейшего централизованного анализа данных, а также формирования отчетов из сводной базы, как по всему предприятию в целом, так и в отдельности по каждому структурному подразделению. Внедрение распределенной информационной системы осуществляется при необходимости обеспечения централизованного контроля над изменением данных в дистанционно удаленных подразделениях организации.

1.2 Современные РИС. Высоконагруженные и децентрализованные системы и их основные признаки

Одной из современных областей применения РИС являются высоконагруженные информационные системы, такие как популярные веб-сайты типа поисковых систем или социальных сетей, видеохостингов, а также различные облачные хранилища файлов. Все они являются распределенными системами, построенными по архитектуре «клиент-сервер». Однако в таких системах само понятие сервер, является абстрактным, но при этом прозрачным для пользователя.

В качестве клиента в таких системах выступает пользовательское приложение, такое как веб-браузер, мобильное приложение. За абстрактным понятием сервер в таких ИС может стоять целая цепочка физического оборудования, такого как балансировщики нагрузки, серверы приложений, серверы баз данных, обрабатывающих запросы по цепочке и объединенных в кластеры. Цель подобных современных РИС – быстрая обработка больших массивов данных и больших групп пользователей.

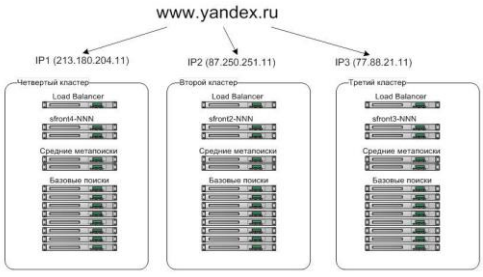


Рисунок 1.5 – Упрощенное представление многоуровневой архитектуры клиент-сервер поисковой системы Яндекс[3]

Для планирования использования ресурсов компьютерных систем с целью решения вычислительных задач в настоящее время используется

3 Архитектура Поиска Яндекса. Лекция для Малого ШАДа [Электронный ресурс] https://habr.com/ru/company/yandex/blog/204282/ (дата обращения 04.12.2019).

комплекс различных методов. Высоконагруженными называют системы безостановочного доступа, то есть те структуры, запрос данных из которых позволяет получать информацию без длительной задержки при беспрерывной работе. Обычно для такой работы используются мультисерверные решения.

Сервера могут разделяться на 3 типа:

- Front (сервера взаимодействия с конечным потребителем);
- Middle (сервера агрегации запросов, кэширования, обработки);
- BackEnd (системы доступа к хранилищу данных).

Если система высоконагруженная, но не имеет сложной модели кэширования или кодирования информации, то роль MiddleWare может быть опущена. Высоконагруженные системы состоят обычно из нескольких front и back серверов или групп серверов. Для географически-распределенных проектов, местонахождения front/back групп может быть разнесено в разные ЦОДы в зависимости от потребителя ресурсов.

Облако. При облачном объединении сервера обмениваются процессорными ресурсами. Вывод из строя одной или нескольких машин не приведет к сбою сервиса, нагрузка с него перераспределится между остальными участниками облака. Обычно программное обеспечение разрабатывается специализированно для работы с многопоточными облачными вычислениями. Взаимодействие между модулями приложения может иметь более глубокие возможности. Для данной схемы, серверные приложения Front и Back могут находиться внутри одного облака.

Распределенная система. При таком взаимодействии, экосистема строится на создании изолированных серверных машин, общающихся только на уровне приложений. Масштабируемость в данной структуре более проста и прозрачна, чем при облачных вычислениях.

Связь Front и Back серверов назначается как много ко многим, но не ко всем, то есть один Front сервер перераспределяет запросы к ограниченному

числу Back серверов, это необходимо для предотвращения выхода из строя всех машин, в случае атак или сбоев. Очередь запросов распределяется равномерно или по весовой доле между различными Back серверами.

Обычно высоконагруженные системы разрабатываются с распределенными вычислениями и системами хранения. Для обеспечения бесперебойной работы без вовлечения человека. Для повышения уровня отказоустойчивости, создаем системы самодиагностики и устранения неполадок при сбоях. При обнаружении неисправностей, модуль отчетов (bug tracer, bug logger) передает в службу поддержки отладочную информацию, при получении которой делаются доработки системы и передаем алгоритмы обновления заказчику.

Разработка высоконагруженных систем производится с передачей сопроводительной документации и исходных кодов заказчику. Программное обеспечение имеет модульное тестирование. Наиболее важным моментом в этом является невмешательство третьих лиц, либо при доработке функционала третьими лицами необходим своевременное информирование нас и передача документации.

Балансировка нагрузки (Load Balancing) применяется для организации распределенных вычислений с помощью распределённой (параллельной) вычислительной системы. Балансировка нагрузки предполагает обеспечение балансировщиками равномерно нагрузки определенных вычислительных узлов, входящих в состав распределенной системы.

При поступлении задач на обработку запросов, балансировщик должен принять решение о том, на каком нижестоящем вычислительном узле следует выполнять вычисления, связанные с этим заданием. Кроме того, балансировка предполагает перенос на лету части вычислений с наиболее загруженных вычислительных узлов на менее загруженные узлы.

Среди задач, требующих больших вычислительных мощностей можно выделить довольно разнообразные направления.

Современный информационный мир все больше переходит на интернет-приложения, где в качестве клиента выступает браузер операционной системы. Ярким примером задачи, требующих огромных вычислительных модностей является порталы социальных сетей, например "В контакте". Ежедневная посещаемость этой сети достигает более 90 миллионов человек. Обработать такое количество активных пользователей базы данных не удастся не одной системе типа файл-сервер. Поэтому для решения подобных задач единственным правильным решением будет обращение за помощью к средствам построения клиент-серверной платформы в той или иной конфигурации.

Другим примером является социальная сеть видео – YouTube. Ежедневная посещаемость составляет сотни миллионов пользователей и неуклонно растет. Каждую минуту на портал загружается тысячи часов видеороликов. Для обеспечения бесперебойной работы такого портала требуются очень большие вычислительные мощности, сопряженные с грамотно-построенным приложением, работающим в архитектуре клиент-сервер.

Другая интересная сфера, где требуется огромные вычислительные мощности – это трейдинг на финансовых рынках.

С развитием интернет-технологий торговля на финансовых рынках, таких, как Forex, Nyse, Московская биржа и других ведется из пользовательских торговых терминалов, таких как MT4, Rumus 2 и подобных. Широкая реклама и пропаганда по завлечения масс населения в "короткие" инвестиции с целью получения быстрого дохода привела к тому, что большая часть населения планеты ведет электронную торговлю на фондовых, фьючерсных и валютных рынках не в здании биржи, а электронно. Чтобы быстро обработать многомиллионные запросы пользователей по всему миру требуется огромная вычислительная мощность, обеспечивающая быструю синхронизацию, обновление котировок без

запаздываний, почти мгновенное исполнение запросов на покупку или продажу актива. Если брокер будет работать медленнее, чем конкурент, то финансовые спекулянты и инвесторы уйдут к другому, с более быстрым исполнением команд и надежной репутацией. Поэтому, для решения этих задач необходима исключительно распределенная архитектура.

Важным свойством распределенной информационной системы является уровень доступности системы. Уровень доступности является важной характеристикой распределенной системы и определяется не только доступностью определенного элемента в момент времени t, но и принципами организации защиты РС.

Доступность системы в общем случае достигается за счет применения трех групп мер, направленных на повышение:

- безотказности;
- отказоустойчивости
- обслуживаемости.

Одним из основных признаков и требований к РИС является отказоустойчивость.

Согласно теоретическим положениям, отказоустойчивость определяется способностью системы сохранять работоспособность при отказе или выходе из строя одного или нескольких вычислительных узлов. Можно выделить несколько компонентных составляющих общей отказоустойчивости: ПО, аппаратного обеспечения, отдельных модулей внутри устройства (например, отказоустойчивость жестких дисков блоков питания), площадки (в случае, если ИТ-система имеет географически распределенную архитектуру).

В настоящее время единственным архитектурным решением обеспечения отказоустойчивости является избыточность входящих в нее аппаратных компонентов.

Отказоустойчивость на программном уровне. С целью ее предоставления выполняется кластеризация. В случае программного либо аппаратного перебоя в работе кластером определенная задача одинаково перераспределяется между правильно работающими узлами. Иными словами, ПО кластера определяет неправильно работающий программный компонент и «выключает» этот модуль с работы, а в случае аппаратной неисправности – выключает определенный узел и распределяет его нагрузку между правильно работающими узлами.

Отказоустойчивость аппаратного обеспечения. Отказоустойчивость аппаратного обеспечения осуществляется резервированием компонентов, по схеме N+1 (один резервный элемент) также 2N (двойная чрезмерность, предполагающая удвоение элементов).

Аппаратная отказоустойчивость прибора гарантируется его изготовителем. Способности опции во данном случае, равно как принцип, минимальные, но введение перемен во схему осуществлении отказоустойчивости допустимо только лишь изготовителем посредством развитие микрокода аппаратного приборы. При этом схема аппаратной реализации отказоустойчивости выбирается архитектором системы из жестко заданных производителем оборудования вариантов. Применяются схемы N+1, N+2, 2N, а также множество производных схем, которые могут быть рекомендованы производителем оборудования или правилами построения системы.

Стоит также отметить, что такого рода решения могут после проведения диагностики предусматривать автоматическое устранение отказа через некоторый период времени. Во избежание потери данных при выходе из строя носителей информации отказоустойчивость отдельных модулей хранения реализуется во всех критичных системах, а также системах корпоративного класса. Для ИТ-персонала компаний это не представляет

особой сложности, но для правильного выбора схемы обеспечения отказоустойчивости рекомендуется использовать проектный подход.

Все большее распространение в наши дни получают распределенные вычислительные системы. Наиболее перспективными являются технологии GRID. Grid-технологии дают возможность реализовать на практике географически распределенные вычислительные инфраструктуры. Под ними понимается определенная форма распределенных вычислений, где основной виртуальный «суперкомпьютер» образован из некоторых кластеров, соединённых по сети Интернет, компьютеров, которые вместе выполняют совместную работу в целях решения большого числа производственных заданий, таких как добровольные вычисления.

1.3 Применение РИС для объединения территориально-распределенных рабочих мест и производственных объектов

В современном мире развитие крупной компании уже не помещается в привычные рамки, с каждым годом структура организации только усложняется. Сейчас тяжело себе представить современно развивающуюся компания с одним головным офисом. Специфика нашей страны заключается, в частности, в том, что в России существует большое количество крупных предприятий, занимающих обширные территории, такие как предприятия топливно-энергетического комплекса, банковской сферы, общественного питания и т.п. С каждым годом география все больше расширяется, филиалы крупных федеральных компаний открываются в новых городах, странах и даже континентах. К числу таких компаний относятся строительные и добывающие организации, торговые организации с наличием розничных точек, туристические агентства, банковский сектор и многие другие. И вполне закономерно, что все больше предприятий прибегают к внедрению распределенных информационных систем связи на базе современных программных продуктов.

Внедрение в организацию, имеющую в своей структуре территориально-распределенные рабочие места и производственные объекты распределенной информационной системы приведет к уменьшению затрат и издержек, вызванных необходимостью ручного переноса данных из одной информационной системы в другую. Создание распределенной базы данных позволит выполнять агрегированные запросы сразу для всей сети, получать статистику в режиме реального времени. Переход на одну программу исключит необходимость выполнения операций импорта-экспорта в ручном режиме, из таких систем как интернет-банк, системы бухгалтерской и налоговой отчетности, базы данных и т.п., что сократит количество ошибок и расходы на ручной труд.

Подводят итог и объединяя понятия распределенной информационной системы и территориально-распределенной структуры можно вывести определение, что территориально-распределенный информационный комплекс – это инфраструктурный комплекс, который виртуально и логически объединен в единую систему общим информационным полем. Он строится с учетом реальных бизнес-процессов и информационных потоков в компании.

1.4 Постановка задачи на разработку ПО

В качестве предметной области и проектной части по теме диссертации выбрана распределенная система учета в сети кофейных заведений. Сеть кофейных заведений – вид деятельности, который относится к ведению бизнеса, требующего учета приобретаемого пищевого материала и учета продаж. Данный учет осложнен большим количеством географически распределенных объектов: самих заведений, их поставщиков и работников, и используемых ими учетных систем. Поэтому для решения указанной проблемы целесообразно построение распределенной информационной системы на базе архитектуры «клиент-сервер».

Для решения поставленной задачи необходимо выполнить следующие действия:

- описать технологию сбора данных для проведения диссертационного исследования;
- провести исследование истории, назначения и области применения распределенных информационных систем;
- выполнить описание бизнес-процессов предприятия, постановку задач;
- осуществить моделирование бизнес-процессов, информационных потоков;
- обосновать архитектуру распределённого приложения;
- описать методы обработки результатов, оценки их достоверности и достаточности для завершения работы над диссертацией;
- разработать алгоритмы реализации проектируемого программного продукта, необходимого для решения обозначенной в исследовании проблемы;
- показать технологию преобразования разработанных алгоритмов в программный продукт;
- выполнить проектирование распределенной базы данных, создание таблиц и определение связей между ними;
- создать пользовательского интерфейса работы с базами данных (приложения-клиента);
- протестировать программный продукт и показать его значимость для исследования.

Выводы по 1 главе

В результате выполнения первой главы магистерской диссертации были рассмотрены история, назначение и область применения распределенных информационных систем. Это необходимо сделать для того,

чтобы выбрать архитектуру системы, которая будет рассмотрена в практической части.

Исследованы основные разновидности РИС. Одним из первых типов РИС были GRID. Под ними понимается определенная форма распределенных вычислений, где основной виртуальный «суперкомпьютер» сформирован из некоторых вычислительных узлов, которые вместе выполняют работу в целях решения большого числа производственных заданий.

Следующим подразделом являются Системы, имеющие требования к отказоустойчивости и высокой доступности. Это различные диспетчерские системы, системы управления критическими объектами инфраструктуры и т.п. Эти системы имеют многократное резервирование, которое обеспечивает отказоустойчивость.

Следующей областью применения РИС является высоконагруженные системы. Это различные популярные веб-сайты, имеющие миллионы одновременных подключений. Для обработки такой нагрузки стоятся распределенные системы.

Еще одной областью применения РИС является территориально распределенные рабочие места и производственные объекты.

ГЛАВА 2 ВЫБОР И ОБОСНОВАНИЕ АРХИТЕКТУРНОГО РЕШЕНИЯ И ИСПОЛЬЗУЕМЫХ ТЕХНОЛОГИЙ

2.1 Теоретические основы ведения кофейного бизнеса

Современная экономическая система представляет из себя тесно интегрированные формы бизнеса, которые существуют и зависят от существования других видов бизнеса. Большинство из них тесно связаны с информационными технологиями в той или иной степени.

Не исключением являются и малые виды бизнеса. Кто-то из малого бизнеса отдает свое предпочтение таким программным продуктам по ведению учета как 1С Предприятие, кто-то другим системам. А некоторые фирмы заказывают индивидуальные программы, которые наиболее точно охватывают потребности по ведению учета.

При ведении кофейного бизнеса необходимо вести учет товаров, продуктов, учет поставщиков товаров. Данные товары используются для дальнейшего приготовления продукции. Такими полуфабрикатами могут выступать сахар, кофе в зернах, молоко, сливки и другие. При смешивании их образуется конечный продукт, который продается пользователю.

Весь этот механизм необходимо учитывать, чтобы облегчить труд и исключить по возможности ошибки, связанные с человеческим фактором.

Для этого предназначены компьютерные программы, связанные с учетом и отчетностью. Они помогают держать в поле зрения сведения обо всех поставщиках, товарах, обо всех проданных и поставщиках товарах за месяц, неделю, день.

При закрытии кассы следует так же подсчитывать все операции, которые произвел оператор-барист.

Так же, необходимо вести учет по всей сети кофейных заведений.

Таким образом, кофейный бизнес – это сложный механизм, где нужно рассчитывать все нюансы по взаимодействию с поставщиками и клиентами. В этом помогает программное обеспечение.

Сеть кофейных заведений – вид деятельности, который относится к ведению бизнеса, требующего учета приобретаемого пищевого материала и учета продаж. Данный учет осложнен большим количество географически распределенных объектов: самих заведений, их поставщиков и работников, и используемых ими учетных систем. Поэтому для решения указанной проблемы целесообразно построение распределенной информационной системы на базе архитектуры «клиент-сервер».

2.2 Моделирование бизнес-процессов

Перед проектированием информационной системы необходимо провести моделирование бизнес-процессов, которые будет выполнять система. Это очень удобно сделать в программе Microsoft Visio по методологии IDEF[4] или BPMN. К бизнес-процессам относится организация продаж, учет, работа персонала, учет нормативных и регулирующих документов.

Методология моделирования бизнес-процессов IDEF0 предписывает построение строгой иерархической системы диаграмм - единичных описаний фрагментов процесса в определенной системе. Первоначально осуществляется описание системы в целом и ее взаимодействия с внешними системами (контекстная диаграмма), после чего проводится функциональная декомпозиция – моделируемая система разбивается на входящие в ее состав подсистемы и каждая подсистема описывается отдельно (стоятся диаграммы декомпозиции). Затем каждая подсистема разбивается на более мелкие объекты до достижения нужной степени детализации процесса.

Основной блок состоит из определенной бизнес-функции, в которую есть Вход и Выход, а также блоки Управления и Механизмов.

[4] Черемных, С.В. Структурный анализ систем: IDEF-технологии / С.В. Черемных, И.О. Семенов, В.С. Ручкин. - М.: Финансы и статистика, 2001. - 208 с.

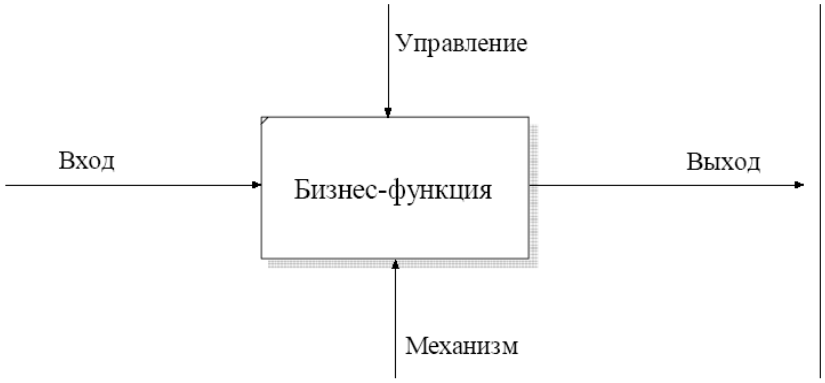


Рисунок 2.1 – Основные типы стрелок в методологии IDEF0

Первоначально, работа с приложением начинается с настройки пользователей и сведений об организации. Эту информацию может редактировать только администратор системы. Следующим шагом в работе является первоначальное заполнение справочников продуктов, которые будут поступать и которые будут расходоваться для изготовления конечного продукта. Эта функция так же доступна пользователю с учетной записью администратора. Проведем моделирование основного бизнес-процесса.

Например, к потокам Управления относятся нормативные документы, такие как локальные акты (рецептура), государственные, такие как санитарные нормы и правила.

К потокам Механизмов относится персонал, сырье, инфраструктура и т.п.

К Входам относятся заказы, поставка.

К Выходам конечные продукты и отчетность.

Рисунок 2.2 – Схема основного бизнес-процесса уровень 0

Следующим этапом является моделирование процесса продаж. При оформлении заказа происходит проверка наличия необходимого сырья на локальном складе кофейни. При его наличии, заказ передается в производство. После производства, заказ переходит к конечному потребителю и ведется его учет.

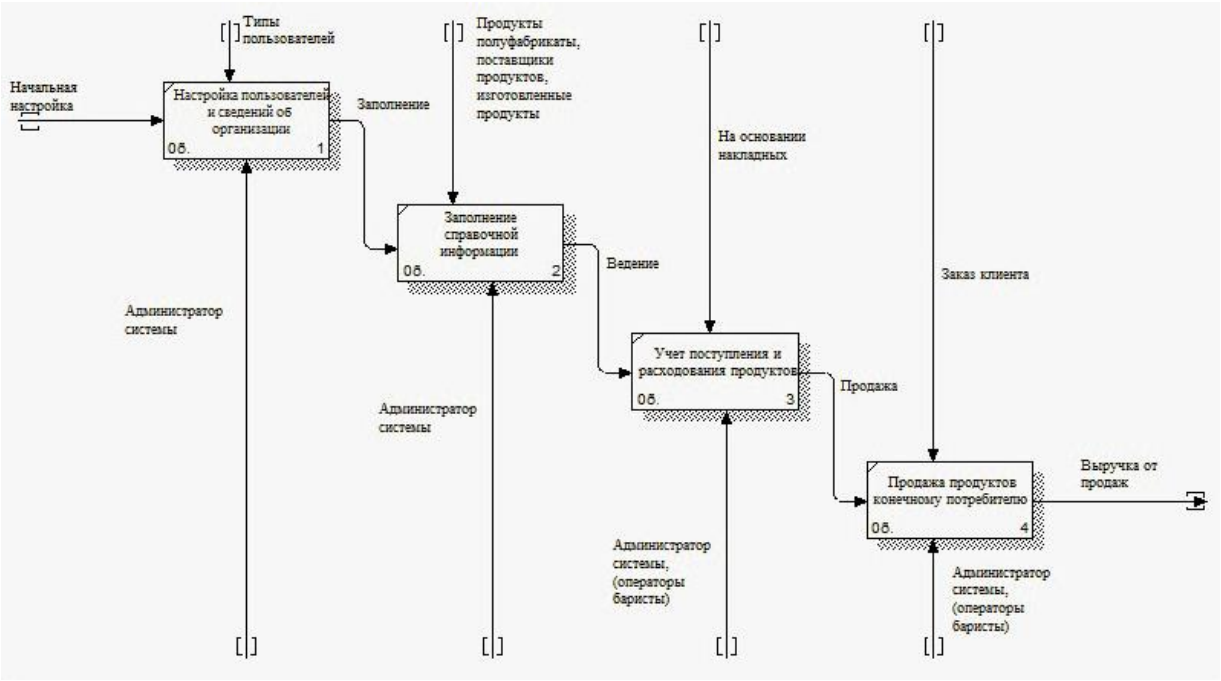


Рисунок 2.3 - Схема процесса работы в системе уровень А0

При отсутствии в достаточном количестве необходимого сырья на локальном складе, заказ не подтверждается. После этого формируется автоматическая заявка на поставку с удаленного склада.

После получения заявки на удаленный склад, происходит проверка наличия сырья на нем. При его наличии, оно поставляется в определенную кофейню. При его отсутствии, оформляется заказ у поставщиков.

Аналогично система ведет себя при приближении остатков продукции к концу. В данном случае все необходимые действия по оформлению заказов производятся заблаговременно, без отказа в приеме заказа у клиента.

Завершающим этапом любой операции является ее учет: учет продаж, бухгалтерский учет, подготовка отчетов.

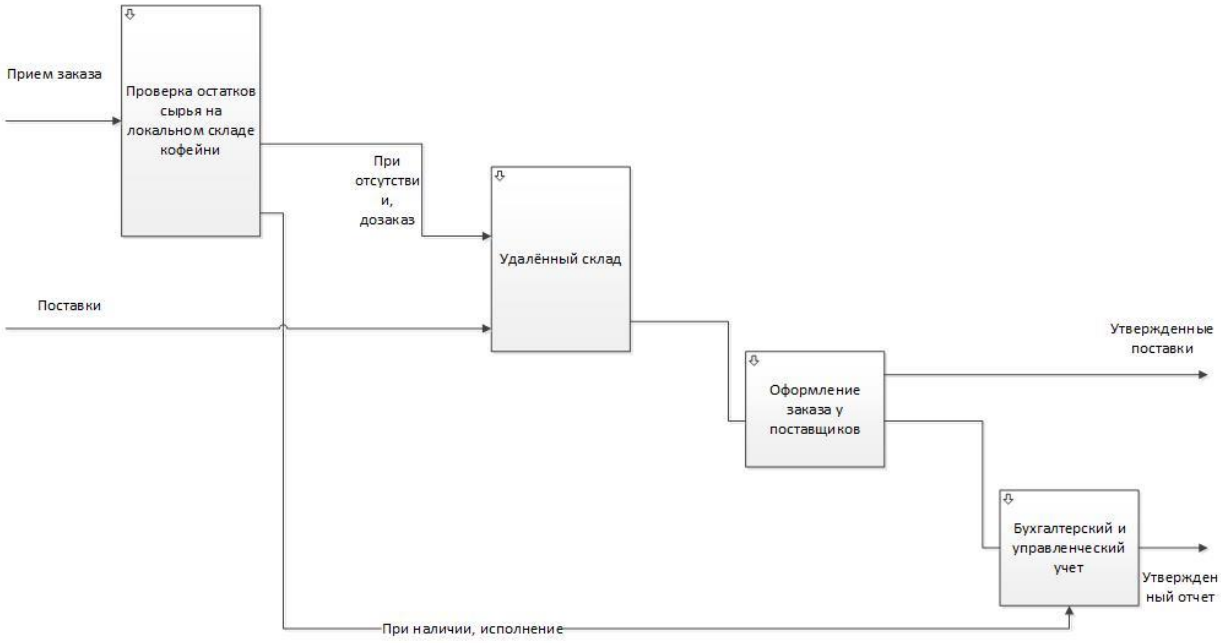


Рисунок 2.4 – Схема процесса проверки остатков на локальном (удаленном) складе и их дозаказа у поставщиков уровень 1

Далее, на основании введенной в справочную систему информации ведется учет ведется учет поступления от поставщиков и расходования продуктов-полуфабрикатов для изготовления конечных продуктов.

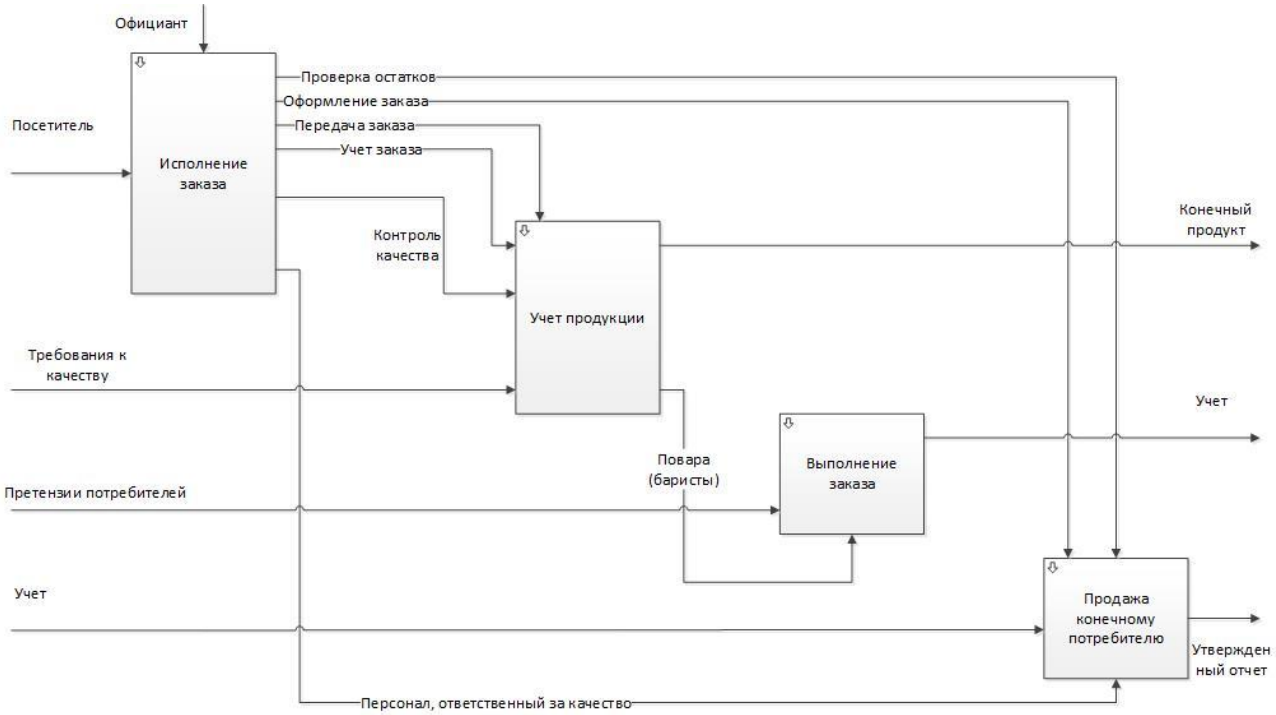


Рисунок 2.5 – Схема процесса учета произведенной продукции, сырья и продаж уровень 1

Последним этапом в бизнес-модели является продажа конечного продукта посетителям. Моделирование произведено в Microsoft Visio.

2.3 Разработка архитектуры информационной системы

2.3.1 Выбор и обоснование архитектуры «клиент-сервер»

В технологической части будет рассматриваться сеть кофейных заведений, т.е. сеть территориально-распределенных производственных объектов, использующих различные информационные системы. Для реализации РИС указанной сети следует применить архитектуру «клиент-сервер». Использование подобной архитектуры можно обосновать необходимостью осуществления централизации, для ведения единого учета продаж и остатков продукции, планирования поставок. РИС можно представить как двухуровневую систему, содержащую компоненты:

КЛИЕНТ -> БАЗА ДАННЫХ

где:

- «Клиент»: пользователь ИС (официант, барист, менеджер, бухгалтер, кассир и т.п.), работающий в определенной точке (кофейне) через «клиента» (модуль учетной программы, рассматриваемой далее)
- «Сервер/База данных»: расположена на отдельном сервере, размещенном в главном офисе (дата центре), которая является единой доля всех «клиентов», т.е. содержит единых меню, поставщиков и т.п.

Применение удаленной базы данной в рассматриваемой системе необходимо для упрощения выполнения агрегированных операций, таких как ведение статистики, учет закупок и расхода сырья.

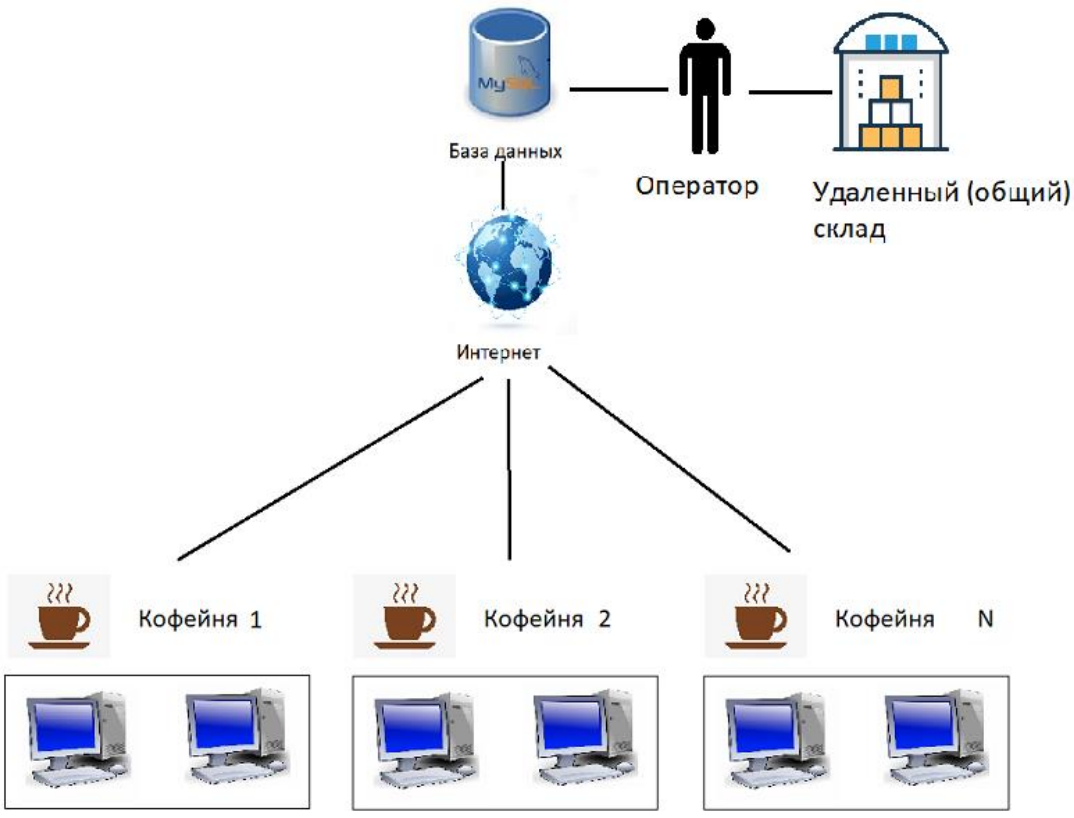


Рисунок 2.6 – Упрощенное представление архитектуры ИС

2.3.2 Теоретический обзор архитектуры

Построение информационных систем является довольно сложным процессом. Системы, работающие в экономических предметных областях, обрабатывающие бизнес-процессы, либо ведущие учет, строятся на основе технологий баз данных.

Несмотря на постепенное развитие технологии NoSQL, все же в настоящее время значительно преобладает технология построения реляционных баз данных.

В мире построения реляционных баз данных существует множество форматов данных. Обрабатываются данные, так называемыми СУБД, системами управления баз данных. Так вот, глобально принято делить СУБД на локальные и промышленные.

Локальные СУБД предприятия для обработки относительно небольших задач в рамках одного предприятия. Например, это может быть начисление заработной платы, учет товаров на складе. Да и вообще абсолютно любая задача, требующая учета может быть решена с применением технологий локальных СУБД.[5] У локальных СУБД есть два существенных ограничения. Первое ограничение связано с максимально допустимым числом полей в таблице, числом записей в таблице, размером таблицы и так далее. Например в Visual FoxPro максимально допустимое количество записей в таблице не может превышать 1 миллиарда записей, а максимальный размер файла таблицы не может превышать 2 гигабайт. Таким образом, в локальных базах данных есть определенные ограничения. Для маленького и среднего по величине предприятия достичь 1 миллиарда записей в таблице весьма трудно, поэтому подобные ограничения не существенны. К тому же, можно делать архивные копии, перезаписывая часть старых записей в виде архива в отдельную базу данных.

[5] Джеймс, Р. Грофф SQL. Полное руководство / Джеймс Р. Грофф, Пол Н. Вайнберг, Эндрю Дж. Оппель. - М.: Вильямс, 2014. - 960 с.

Существенные ограничения проявляются в количестве пользователей, которые одновременно работают с СУБД. Их количество, как правило не может превышать одного-двух десятков. При хорошо работающей сети, число пользователей возрастает до нескольких десятков. Но все же, когда пользователей слишком много, система начинает работать медленно. И виной тому является технология файл-серверной системы, которые используют локальные СУБД.

Следующим этапом в развитии явилась клиент-серверная технология построение систем, работающих на основе баз данных.

Таким образом, существуют две основные модели в построении базы данных, называемые архитектурой СУБД: файл-сервер, подходящий для проектирования небольших систем и клиент-сервер, предназначен для систем корпоративного уровня.[6]

Системы файл-серверного типа работают следующим образом. Когда программа посылает запрос базе данных, то в результате программе передаются все данные и обработку этих данных, например массовую замену одного значения на другое, ведет клиентская программа. При этом на сеть создается большая нагрузка. Когда компьютеров, обращающихся к одной и той же базе данных становится слишком много, сеть начинает давать сбои в работе.

При клиент-серверном подходе дело обстоит иначе. Программа-клиент посылает данные на сервер в виде запроса. Сервер обрабатывает запрос клиента и посылает ему лишь нужный результат, а не все данные. Таким образом, сетевой трафик остается не загруженным.

Как правило, базы данных, относящиеся к клиент-серверным технологиям располагаются на отдельных компьютерах сети, называемых серверами сети. Программа (СУБД), которая обрабатывает эту базу данных применяет язык

6 Кригель, А. SQL. Библия пользователя / А. Кригель. - М.: Диалектика / Вильямс, 2016.

SQL, поэтому такие программы в рамках клиент-серверной технологии принято называть SQL-серверами.

Клиентское же приложение обращается, прежде всего к SQL-серверу, то есть, к СУБД, которое в свою очередь, обрабатывает базу данных. Таким образом, достигается централизация в управлении безопасностью доступа к данным. Например, триггеры и хранимые процедуры, написанные на языке SQL и хранящиеся в базе данных (управляемые при этом SQL-сервером), будут действовать для всех клиентов.

Количество клиентов при использовании технологии клиент-сервера практически не ограничено. Примером может служить любой крупный интернет-ресурс. Доступ, например к порталу ok.ru может достигать множество тысяч пользователей одновременно, работающих в системе. Их количество может быть ограничено только физической производительностью кластера серверов, на котором расположена СУБД.[7]

Локальную базу данных, например, Access, то же можно расположить на отдельном сетевом компьютере, а клиенты будут расположены на разных машинах сети, но это не сделает данную систему клиент-серверной. Она по-прежнему будет являться системой с файл-серверной архитектурой сети.

Клиент-серверной систему делает лишь промышленная СУБД.

Помимо практически неограниченного числа пользователей информации в системе клиент-сервера, у нее есть ряд других преимуществ по сравнению с файл-серверной технологией:

- Достаточно сильная централизация управления данными, что жестко сказывается на безопасности и целостности данных.
- Центральное хранилище файлов.
- Сервера совместно используют имеющееся программное и техническое обеспечение.

7 Селко, Джо SQL для профессионалов. Программирование / Джо Селко. - М.: ЛОРИ, 2015.

- Гибкая управляемость при очень большом количестве пользователей.

Но у данной системы есть и свои недостатки:

- Дорогое техническое обеспечение. Прежде всего это дорогостоящие сервера, которые должны обрабатывать огромное количество информации.
- Дорогостоящие лицензии на серверные операционные системы.
- Требуется администратор для выполнения массовых операций над данными.

Клиент-серверные системы, в общем своем случае делятся на двухуровневые и многоуровневые системы.

Двухуровневая система представляет из себя условно простую архитектуру клиент-сервер, состоящую и сервера, являющегося одним уровнем и компьютеров пользователей, представляющих из себя второй уровень. При двухуровневой архитектуре и СУБД и база данных хранятся на одном компьютере сети – сервере.[8]

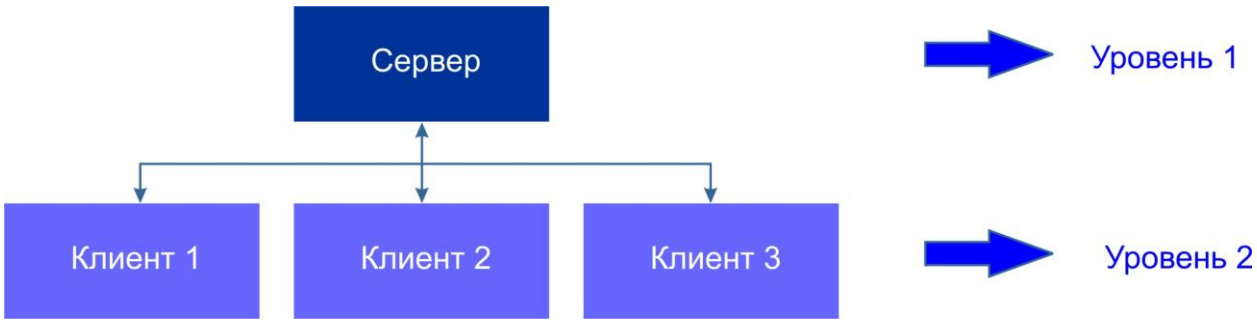


Рисунок 2.7 – Двухуровневая архитектура клиент-сервер

При многоуровневой организации архитектуры клиент-сервер между уровнем клиентов и уровнем сервером добавляется промежуточный уровень – уровень сервера приложений.

8 Маркин, А. В. Построение запросов и программирование на SQL. Учебное пособие / А.В. Маркин. - М.: Диалог-Мифи, 2014. - 384 с.

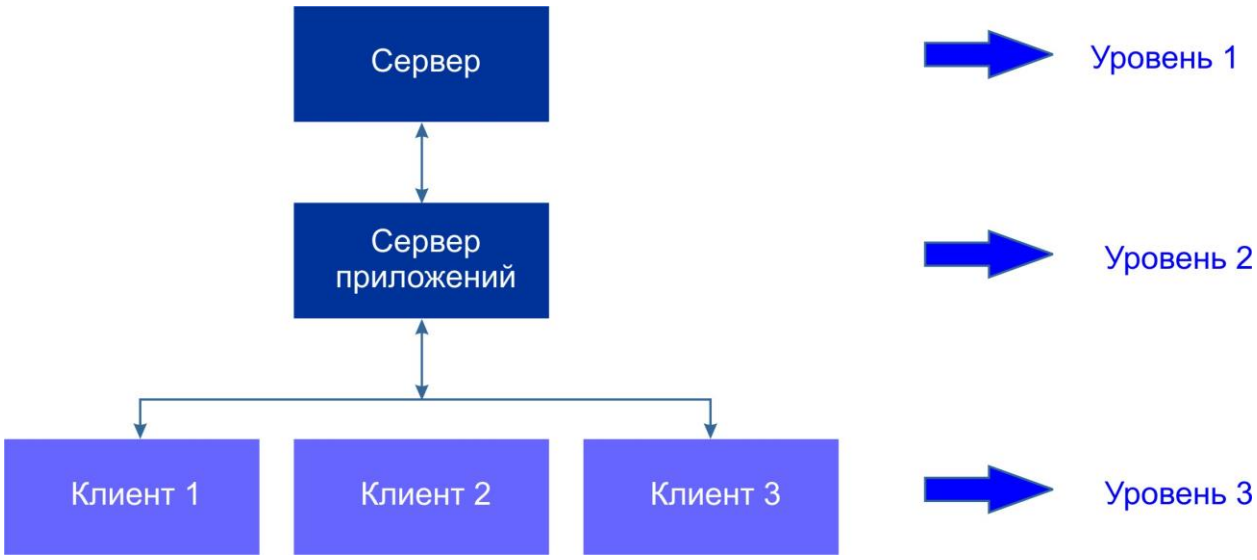


Рисунок 2.8 – Многоуровневая архитектура клиент-сервер

При такой архитектуре на уровне приложений хранится СУБД, а база данных хранится отдельно, на сервере – первом уровне.

Рассматриваемая многоуровневая архитектура позволяет более гибко распределять функции системы и нагрузку между компонентами ПО, а также может снизить требования к аппаратным ресурсам рабочих мест клиентов.[9]

Клиентское приложение взаимодействует с приложением сервером через провайдера данных (поставщика данных). Провайдер обеспечивает передачу информации посредством пакетов данных, представляющих из себя блоки двоичных кодов.

Рисунок 2.9 – Трехуровневая архитектура клиент-сервер

9 Дунаев, В. В. Базы данных. Язык SQL для студента / В.В. Дунаев. - М.: БХВ-Петербург, 2017. - 288 с.

После обработки данные пересылаются обратно, в дельта – пакетах. В этих пакетах содержится информация до внесения изменений в запись и после внесения. Перед физическими изменениями в базе данных на сервере базы данных, сервер приложений осуществляет анализ корректности этих изменений. Например, если запись пытается изменить первый клиент, а запись в это время уже была изменена вторым клиентом, то сделанные изменения первого клиента будут отвергнуты сервером приложений.

Сервера приложений создаются на основе модулей данных:

- Remote Data Module – для серверов DCOM, TCP/IP и OLEnterprise;
- Transactional Data Module – для сервером MTS;
- WebSnap Data Module – для сервера Web;
- SOAP Server Data Module – для серверов SOAP.

Построение приложения "тонкого" клиента отличается от приложения "толстого" клиента.

- Для "тонкого" клиента следует организовать взаимодействие между сервером приложений и "тонким" клиентом.
- Обеспечить взаимодействие по обмену данными между клиентом и сервером.

Таким образом, построение конкретного программного комплекса по технологии клиент-сервер зависит от того, является ли создаваемая система двух или трех-уровневой, а так же от реализации задач для конкретной предметной области.

2.3.3 Преимущества и недостатки архитектуры

Перед окончательным выбором архитектуры приложения и способа взаимодействия с БД необходимо рассмотреть их преимущества и недостатки, возможные риски и угрозы. Например, как будет работать сеть кофеен, в случае пропадания физического доступа к БД, например при потере соединения с интернетом или отказе сервера?

Указанный инцидент приведет к полной невозможности приема заказов, осуществления их учета и приведет к остановке деятельности. Для нейтрализации этого риска можно предусмотреть две схемы – введение временного «бумажного» учета или ведения локальной копии БД с ее синхронизацией с общим сервером. Однако полный отказ от клиент-серверного решения и перехода на автономный (локальный) невозможен, в силу необходимости учета остатков на централизованном складе сети заведений. Поэтому в случае реализации сценария «отказ сервера» возможно продолжение деятельности с временным введением локальной копии БД и последующей ее синхронизацией с удаленной базой. Вероятность отказа модуля «клиент» маловероятна, поскольку каждая точка оборудована несколькими ПК, тем самым осуществляется дублирование. Отказ электропитания приведет к полной остановке деятельности всего заведения в силу пропадания освещения, остановки плит, что приведет к невозможности производства продукции и не имеет смысл детально рассматриваться.

Преимуществом модели взаимодействия и архитектуры клиент-сервер является то, что исходный код клиентского приложения и серверного разделен. Также в данной модели предъявляются пониженные требования к аппаратным ресурсам машин клиентов, так как большая часть вычислительных операций будет производиться на сервере, а также архитектура клиент-сервер довольно гибкая.

К недостаткам рассматриваемой архитектуры можно отнести то, что стоимость серверного оборудования значительно выше клиентского. Сервер должен обслуживать специально обученный и подготовленный персонал. Если в сети откажет сервер, то и клиенты не смогут взаимодействовать с ним, что приведет к затруднению в осуществлении деятельности.

2.3.4 Расчет объема данных и нагрузки

При разработке распределенной информационной системы (ИС) встает вопрос о требующих аппаратных ресурсах – количестве серверов (виртуальных машин) и их аппаратным характеристиках.

В данном исследовании пропускается исследование аппаратной платформы «клиентов» и рассчитываются только необходимые характеристики «сервера».

Базовые показатели:

- Количество торговых точек: 11
- Условное количество клиентов: 30-50
- Количество одновременно работающих клиентов: <10 (пик по данным PhpMyadmin)
- Обращений в минуту: 37
- Средний объем данных, генерируемых за день: 18 МБ

Количество клиентов, объем генерируемых данных - переменная величина регулируемая при расчете/моделировании.

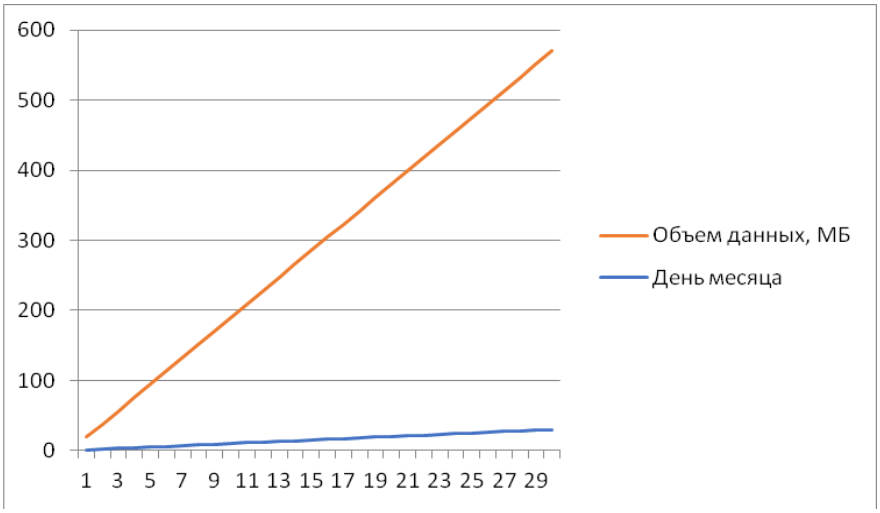
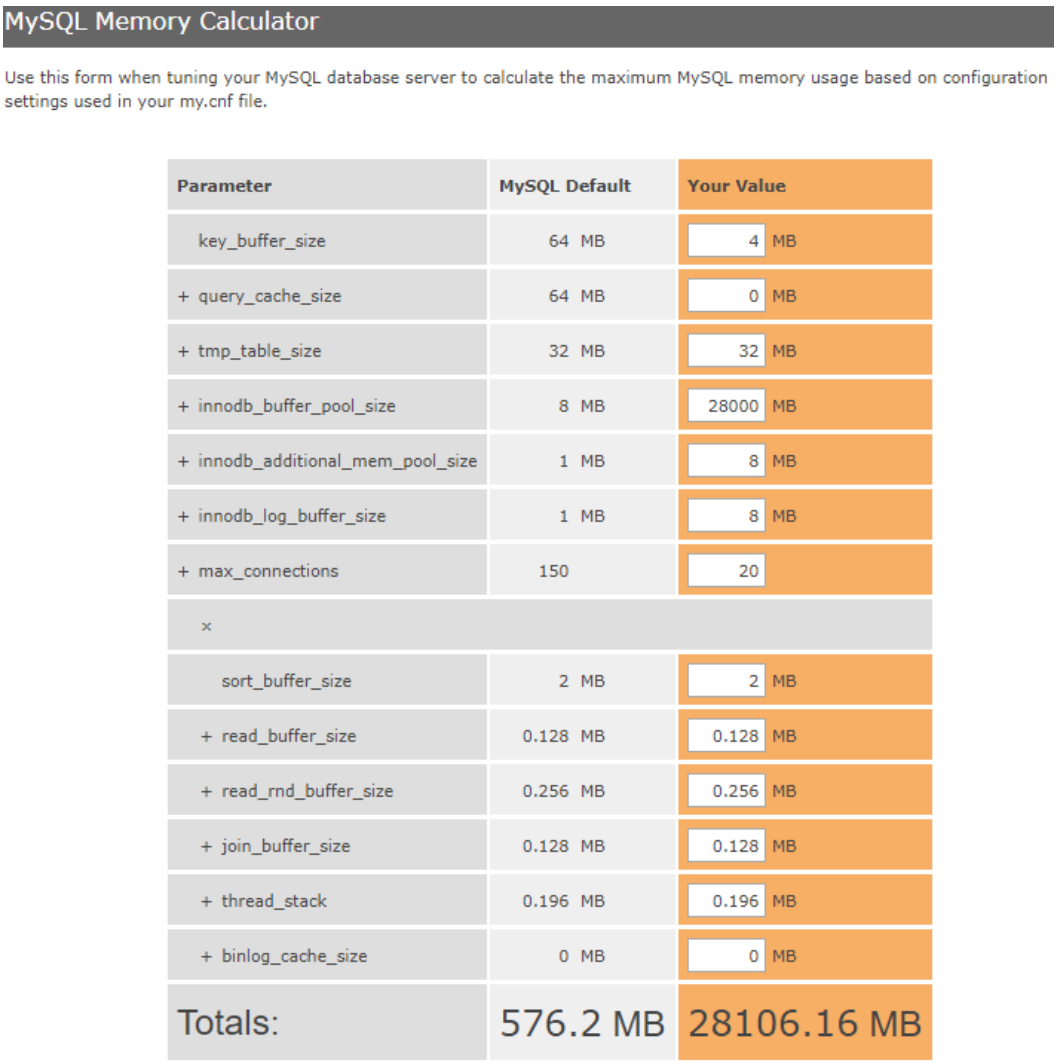


Рисунок 2.10 – Расчет месячного прироста объема данных для целей выбора жесткого диска

Учитывая месячный прирост в объеме не более 600 МБ, при подборе аппаратной конфигурации сервера можно выбрать жесткий диск (SSD) самой минимальной ценовой категории, поскольку его переполнение с данной динамикой случится позднее, чем пройдет его срок эксплуатации и он выйдет из строя из-за механического износа.

На нижерасположенном рисунке выполнен расчет объема БД в памяти для сервера с 32 ГБ ОЗУ. При этом размер буфера InnoDB установлен в 28 ГБ в соответствии с рекомендацией выделения до 80% ОЗУ на сервере, который выполняет только роль сервера БД и не содержит иных приложений, создающих нагрузку. Выделение такого объема памяти позволит кешировать и разместить всю БД в ОЗУ, что многократно ускорит работу с базой, в том числе выполнения сложных агрегирующих запросов, типа COUNT(*). При отсутствии кеширования базы в память, при выполнении подобного запроса, например, для подсчета остатков, высокая нагрузка может парализовать возможность выполнения других запросов, обработка которых требуется в оперативном режиме, например, на оформление заказов.



MySQL Memory Calculator

Use this form when tuning your MySQL database server to calculate the maximum MySQL memory usage based on configuration settings used in your my.cnf file.

Parameter	MySQL Default	Your Value
key_buffer_size	64 MB	4 MB
+ query_cache_size	64 MB	0 MB
+ tmp_table_size	32 MB	32 MB
+ innodb_buffer_pool_size	8 MB	28000 MB
+ innodb_additional_mem_pool_size	1 MB	8 MB
+ innodb_log_buffer_size	1 MB	8 MB
+ max_connections	150	20
×		
sort_buffer_size	2 MB	2 MB
+ read_buffer_size	0.128 MB	0.128 MB
+ read_rnd_buffer_size	0.256 MB	0.256 MB
+ join_buffer_size	0.128 MB	0.128 MB
+ thread_stack	0.196 MB	0.196 MB
+ binlog_cache_size	0 MB	0 MB
Totals:	576.2 MB	28106.16 MB

Рисунок 2.11 – Расчет необходимого количества оперативной памяти

Обобщая вышеприведенные расчеты, в качестве аппаратной платформы под сервер БД рекомендован выбор сервера эконом класса, поскольку учитывая малое количество одновременно работающих клиентов и запросов отсутствуют высокие прогнозируемые нагрузки.

2.4 Выбор и обоснование средств проектирования ПО

2.4.1 Выбор целевой СУБД

В настоящее время существует огромное множество разнообразных средств разработки ПО, многие из которых по функционалу очень похожи друг на друга.

Для решения определенной проблемы и предназначен тот или иной язык программирования или среда разработки. В настоящий момент можно сделать вывод, что выбор среды разработки уже зависит больше не от задачи, которую предстоит решить, а все более от предпочтения программистом того или иного языка. Это связано со все большей универсальностью и развитием систем разработки.

Учет в распределенной информационной системе сети кофеен удобно осуществлять посредством СУБД (Система управления базами данных) и прикладного языка программирования.

СУБД представляет собой систему, которая предназначена для управления имеющейся базой данных[10]. Среди СУБД существует множество решений. Следует упомянуть, что прежде всего, СУБД делятся на настольные и промышленные.

Настольные СУБД предназначены для организации работы с базами данных в рамках мелких, средних и крупных предприятий.

Промышленные же, СУБД используются для организации работы с базами данных, к которым обращаются миллионы пользователей. Например, тот, же сайт государственной статистики, который ежедневно посещают тысячи пользователей всей страны. Данные этого, как и любого другого сайта хранятся в базе данных, по формату относящихся к промышленным СУБД.

Недостатком настольных СУБД является то, что с увеличением числа записей и роста пользователей, система начинает медленно работать. Для решения этой проблемы применяется кеширование результатов запросов и буферизация. Например, система хранения данных InnoDB для СУБД MySQL имеет возможность указать размер буфера, который будет хранится в

10 Аллен, Г. Тейлор SQL для чайников / Аллен Г. Тейлор. - М.: Диалектика, Вильямс, 2015. - 416 с.

ОЗУ. Если размер ОЗУ больше размена базы, все БД поместится в оперативной памяти и будет работать на порядок быстрее.

И, если при увеличении объема записей (данных) в базе данных таких СУБД можно решить вопрос, путем архивирования старых, хронологически первых записей и чистки базы данных, то с по мере увеличения числа пользователей и нагрузки на СУБД вопрос остается актуальным.[11] Именно вопрос, касающийся нагрузки и призвана решить промышленная СУБД.

Среди настольных СУБД следует выделить такие как dBase и Visual dBase, Microsoft Access , Microsoft FoxPro и Visual FoxPro , Paradox .

Среди промышленных СУБД можно выделить Oracle и Microsoft SQL Server, MySQL. Для осуществления данного проекта была выбрана СУБД MySQL Server 5.5.MySQL представляет из себя реляционную систему управления базами данных. Может устанавливаться в операционную систему в качестве службы, что является очень удобным.

Классическим подходом к управлению базой данных MySQL, включая создание самой базы данных, создание таблиц и связей между ними, а так же наполнение таблиц и другие операции осуществляется посредством клиентского приложение в виде командного окна, похожего на обычное командное окно Windows.

В настоящее время существует достаточно много клиентских программ, выполняющих администрирование баз данных MySQL в графическом режиме, так, как это делается в Access. Конечно, механизм визуального формирования здесь весьма отличается.

11 Кузнецов, С. Д. Основы баз данных / С.Д. Кузнецов. - М.: Бином. Лаборатория знаний, Интернет-университет информационных технологий, 2017. - 488 с.

Основными программами, позволяющими осуществлять графическое проектирование базы данных MySQL являются PHPMyAdmin, MySQL Workbench, dbForge Studio for MySQL и многие другие программы.

Для осуществления данного проекта была выбрана программа администрирования MySQL Workbench 5.5.

Основой для работы СУБД MySQL является стандартизированный язык управления данными SQL. В операционную систему устанавливается сервер, представляющий из себя программное обеспечение. Данный сервер и есть непосредственно СУБД, который через посылаемые администратором команды SQL управляет базами данных MySQL через клиентское приложение.

Система управления базами данных MySQL характеризуется высокой скоростью работы и высокой отказоустойчивостью.

В настоящее время поддержку и развитие MySQL осуществляет компания ORACLE. Она же сопровождает и клиентское приложения для администрирования MySQL Workbench.

Отличительной особенностью MySQL от MS SQL Server является то, что последний в настоящее время наиболее ориентирован на работу с Net-языками, в то время, как MySQL работает со многими другими языками программирования.[12]

Основным отличием для большинства разработчиков является все же различие в синтаксисах самого языка SQL.

Ниже приводятся основные отличие трех перечисленных промышленных СУБД.

12 Бьюли, Алан Изучаем SQL / Алан Бьюли. - М.: Символ-плюс, 2014. - 0 с.

Характеристика	Oracle	MySQL	SQL Server
Интерфейс	GUI, SQL	SQL	GUI, SQL, другое
Поддержка языков	C, C++, Java, Ruby, Objective C и др.	C, C++, Java, Ruby, Objective C и др.	Java, Ruby, Python, VB, .Net, PHP
Операционная система	Windows, Linux, Solaris, HP-UX, OS X, z/OS, AIX	Windows, Linux, OS X, FreeBSD, Solaris	Windows
Лицензия	Проприетарная	Свободная	Проприетарная

Рисунок 2.12 – Сравнительная таблица промышленных СУБД

По сравнению с MS SQL Server и ORACLE, MySQL предназначен больше для проектов средней величины. Пожалуй самой популярной предметной областью для использования MySQL является сайтостроение. Это связано прежде всего с открытостью кода и бесплатным распространением MySQL. Тем не менее, надежность и модность MySQL позволяет строить как корпоративные приложения, так и приложения государственного уровня.

Именно объемы и надежность СУБД MySQL, а так же возможность дальнейшего масштабирования и обеспечили ее выбор для реализации поставленной задачи.

2.4.2 Выбор языка программирования

В настоящее время для работы разработки ПО существует достаточно множество различных сред. Это и C++, C#, JAVA и многие другие.

В каждом из распространенных языков есть свои преимущества и недостатки, а также специфическая область применения. Но в настоящее время все языки достаточно развиты и стандартизированы, что выбор того или иного языка аргументируется больше предпочтениями ли навыками конкретного программиста, нежели поставленными задачами. Так же выбор языка аргументируется совместной разработкой. Когда программист устраивается на определенную работу, где ведется командная разработка ПО,

то он вынужден писать программный код на том языке, на котором работает остальная команда.

Для реализации проекта была выбрана среда разработки JAVA, а конкретно JAVA SE 12 в среде разработки IntelliJ IDEA Community Edition 2019.2.3 x64.

JAVA SE 12 - это современный популярный язык программирования с гибкими возможностями. JAVA базируется на концепциях современного объектно-ориентированного программирования:

- Полиморфизм
- Наследование
- Инкапсулирование

Полиморфизм - это возможность, при которой объекты, имея одинаковую спецификацию способны иметь различную реализацию.

Наследование представляет собой технологию, когда дочерний объект наследует атрибуты объекта родителя.

Инкапсулирование - это объединение данных и функций в едином компоненте.

JAVA 12 был разработан компанией Sun. В настоящее время JAVA принадлежит компании Oracle. [13]

Стоит сказать так же, что JAVA - это многоплатформенная среда разработки. Современные версии Java позволяют писать приложения для Андроид.

Для работы с базой данных JAVA представляет из себя многочисленные технологии. Это технология JDBC и многие другие.

Для реализации этого проекта была выбрана технология JDBC. Это «коннектор» к базе данных.

13 Гупта, Арун Java EE 7. Основы / Арун Гупта. - М.: Вильямс, 2014. - 336 с.

Таким образом, JAVA 12 - это одна из самых мощных сред разработки прикладного программного обеспечения, в том числе и для работы с базами данных в распределенных системах.

2.5 Выбор технологий проектирования системы

Для проектирования системы была использована технология доступа к данным Java Database.

Доступ к данным в JDBC осуществляется посредством использования драйвером доступа к данным.

Для доступа же к базам данных MySQL применяются свои драйвера, версия которых зависит от версии MySQL.

В данном проекте применяется java.sql.Connection.

С помощью технологии JDBC можно подключиться к физической таблице базы данных, получить из нее, посредством SQL запроса набор необходимых данных, выполнить необходимые манипуляции с данными и осуществить физическое сохранение из набора в таблицу. Так же, технология JDBC позволяет осуществлять редактирование базы данных без формирования набора данных.[14]

JDBC была разработана главным образом для реализации архитектуры "Клиент-сервер".

14 Мартишин, С. А. Проектирование и реализация баз данных в СУБД MySQL с использованием MySQL Workbench. Учебное пособие / С.А. Мартишин, В.Л. Симонов, М.В. Храпченко. - М.: Форум, Инфра-М, 2015. - 160 с.

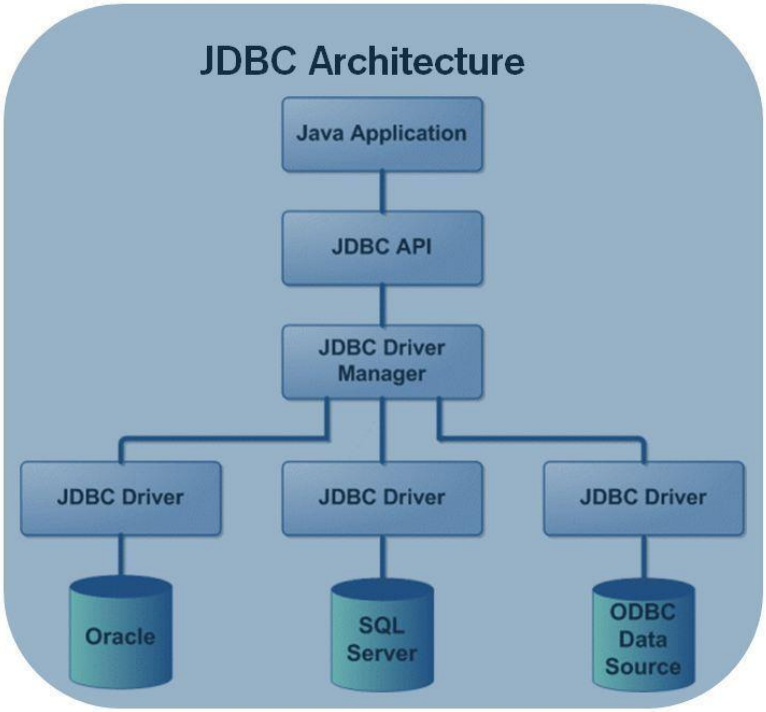


Рисунок 2.13 – Схема работы технологии JDBC

Так, в реляционном способе доступа для получения данных из той или иной таблицы базы данных используется запрос Select. Таким образом, применяя данный запрос можно сформировать выборку, в которую попадут не все данные таблицы, а только необходимые. Под этим подразумевается, что кроме необходимых записей, так же можно ограничиться лишь необходимыми полями. Данные же обрабатывает сервер.

С помощью команды Select легко создавать и многотабличные запросы, выборка которых будет формировать объект Recordset, который будет являться источником данных для создания пользовательских форм.

В качестве среды разработки программного кода выбрана платформа IntelliJ IDEA Community Edition 2019.2.3 x64.

Выводы по 2 главе

В ходе выполнения главы было произведено моделировании бизнес-процессов, выполнен обзор архитектуры. Выбрано и обосновано применение

архитектуры «клиент-сервер» для решения поставленной задачи. Произведены расчеты объема данных и нагрузки на ИС. Рассмотрены потенциальные языки программирования и СУБД, а также средства проектирования.

В результате проделанной работы принято решение выполнить проектную часть на языке программирования Java и СУБД MYSQL. При этом проектирование БД будет выполняться в MySQL Workbench, а написание программного кода в IntelliJ IDEA.

ГЛАВА 3 РАЗРАБОТКА РАСПРЕДЕЛЕННОЙ ИС СЕТИ КОФЕЙНЫХ ЗАВЕДЕНИЙ

3.1 Блок-схема решения задачи

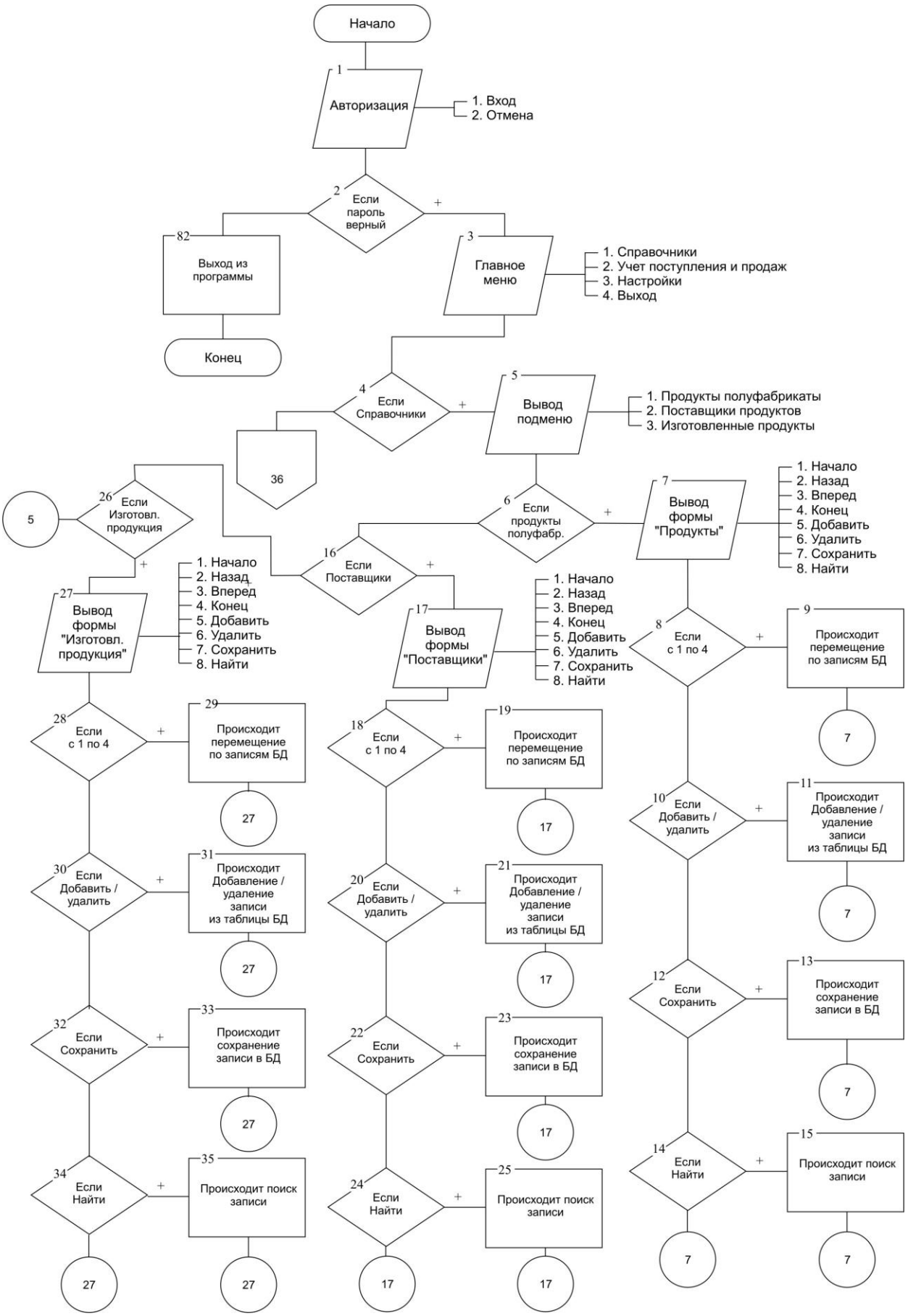
Перед разработкой программного кода и структуры базы данных необходимо выполнить блок-схему, в которой визуально представляются основные процессы, которые будет выполнять ПО[15], такие как авторизация в системе, учет заказов, поступления и расходов продукции.

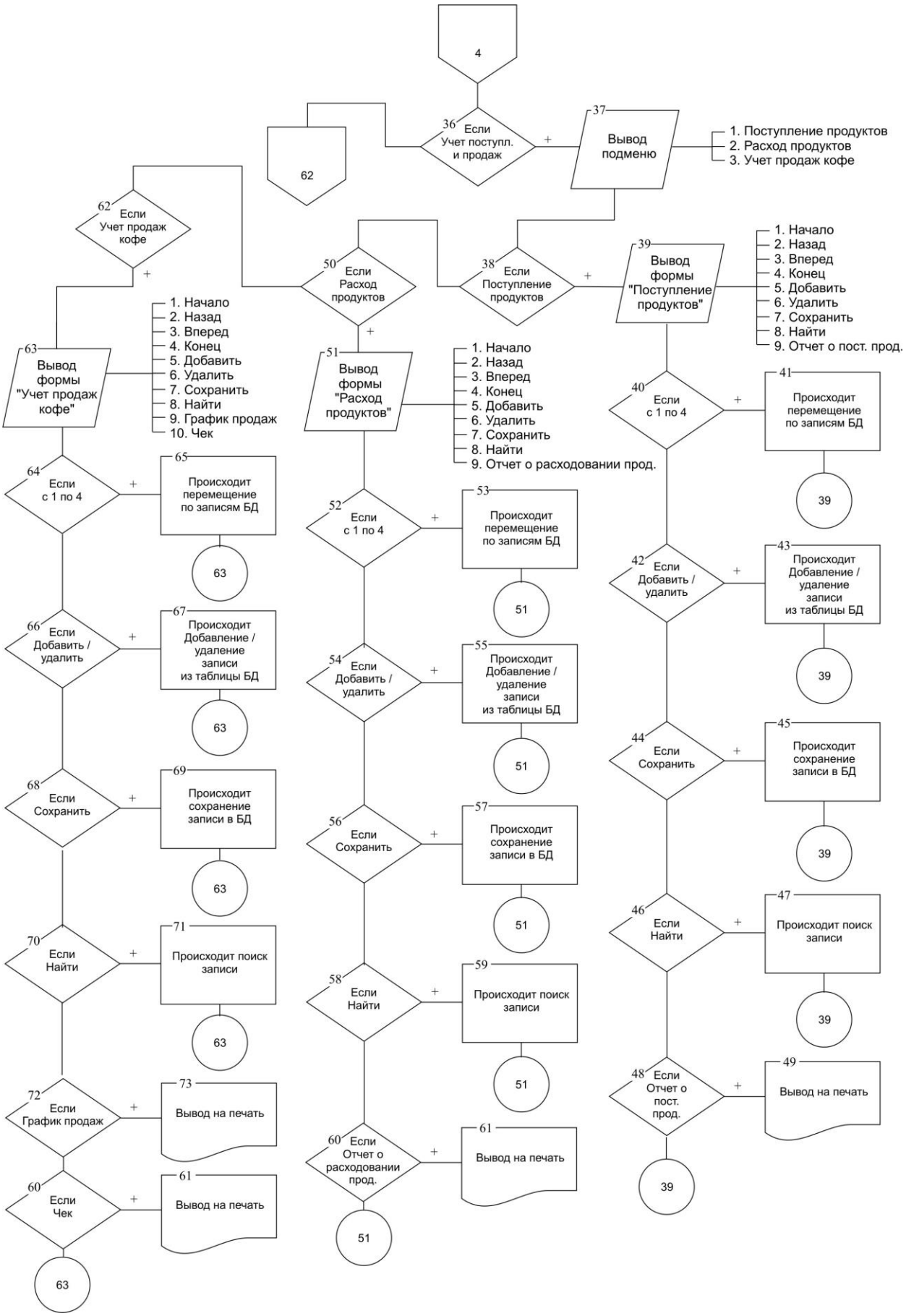
Блок-схемой является унифицированный тип схем (графических моделей), которые описывают алгоритмы или процессы, в которых отдельные шаги изображаются в виде блоков различной формы, соединенных между собой линиями, указывающими направление последовательности.

Правила составления блок-схем регламентируются ГОСТ 19.701-90 «Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения». Стандарт в частности регулирует способы построения схем и внешний вид их структурных элементов.

Блок-схема основного процесса работы ПО и взаимодействия с БД описана на рисунке 3.1.

15 Флах Петер. Наука и искусство построения алгоритмов, которые извлекают знания из данных. Учебник; ДМК Пресс - М., 2015. - 400 с.


Начало
1
Авторизация
1. Вход
2. Отмена
2
Если пароль верный
+
82
Выход из программы
Конец
3
Главное меню
1. Справочники
2. Учет поступления и продаж
3. Настройки
4. Выход
4
Если Справочники
+
5
Вывод подменю
1. Продукты полуфабрикаты
2. Поставщики продуктов
3. Изготовленные продукты
36
6
Если продукты полуфабр.
+
7
Вывод формы "Продукты"
1. Начало
2. Назад
3. Вперед
4. Конец
5. Добавить
6. Удалить
7. Сохранить
8. Найти
8
Если с 1 по 4
+
9
Происходит перемещение по записям БД
7
10
Если Добавить / удалить
+
11
Происходит Добавление / удаление записи из таблицы БД
7
12
Если Сохранить
+
13
Происходит сохранение записи в БД
7
14
Если Найти
+
15
Происходит поиск записи
7
16
Если Поставщики
+
17
Вывод формы "Поставщики"
1. Начало
2. Назад
3. Вперед
4. Конец
5. Добавить
6. Удалить
7. Сохранить
8. Найти
18
Если с 1 по 4
+
19
Происходит перемещение по записям БД
17
20
Если Добавить / удалить
+
21
Происходит Добавление / удаление записи из таблицы БД
17
22
Если Сохранить
+
23
Происходит сохранение записи в БД
17
24
Если Найти
+
25
Происходит поиск записи
17
5
26
Если Изготовл. продукция
+
27
Вывод формы "Изготовл. продукция"
1. Начало
2. Назад
3. Вперед
4. Конец
5. Добавить
6. Удалить
7. Сохранить
8. Найти
28
Если с 1 по 4
+
29
Происходит перемещение по записям БД
27
30
Если Добавить / удалить
+
31
Происходит Добавление / удаление записи из таблицы БД
27
32
Если Сохранить
+
33
Происходит сохранение записи в БД
27
34
Если Найти
+
35
Происходит поиск записи
27
27


4
36 Если Учет поступл. и продаж
37 Вывод подменю
1. Поступление продуктов
2. Расход продуктов
3. Учет продаж кофе
62
62 Если Учет продаж кофе
50 Если Расход продуктов
38 Если Поступление продуктов
39 Вывод формы "Поступление продуктов"
1. Начало
2. Назад
3. Вперед
4. Конец
5. Добавить
6. Удалить
7. Сохранить
8. Найти
9. Отчет о пост. прод.
63 Вывод формы "Учет продаж кофе"
1. Начало
2. Назад
3. Вперед
4. Конец
5. Добавить
6. Удалить
7. Сохранить
8. Найти
9. График продаж
10. Чек
51 Вывод формы "Расход продуктов"
1. Начало
2. Назад
3. Вперед
4. Конец
5. Добавить
6. Удалить
7. Сохранить
8. Найти
9. Отчет о расходовании прод.
64 Если с 1 по 4
65 Происходит перемещение по записям БД
63
66 Если Добавить / удалить
67 Происходит Добавление / удаление записи из таблицы БД
63
68 Если Сохранить
69 Происходит сохранение записи в БД
63
70 Если Найти
71 Происходит поиск записи
63
72 Если График продаж
73 Вывод на печать
60 Если Чек
61 Вывод на печать
63
52 Если с 1 по 4
53 Происходит перемещение по записям БД
51
54 Если Добавить / удалить
55 Происходит Добавление / удаление записи из таблицы БД
51
56 Если Сохранить
57 Происходит сохранение записи в БД
51
58 Если Найти
59 Происходит поиск записи
51
60 Если Отчет о расходовании прод.
61 Вывод на печать
51
40 Если с 1 по 4
41 Происходит перемещение по записям БД
39
42 Если Добавить / удалить
43 Происходит Добавление / удаление записи из таблицы БД
39
44 Если Сохранить
45 Происходит сохранение записи в БД
39
46 Если Найти
47 Происходит поиск записи
39
48 Если Отчет о пост. прод.
49 Вывод на печать
39

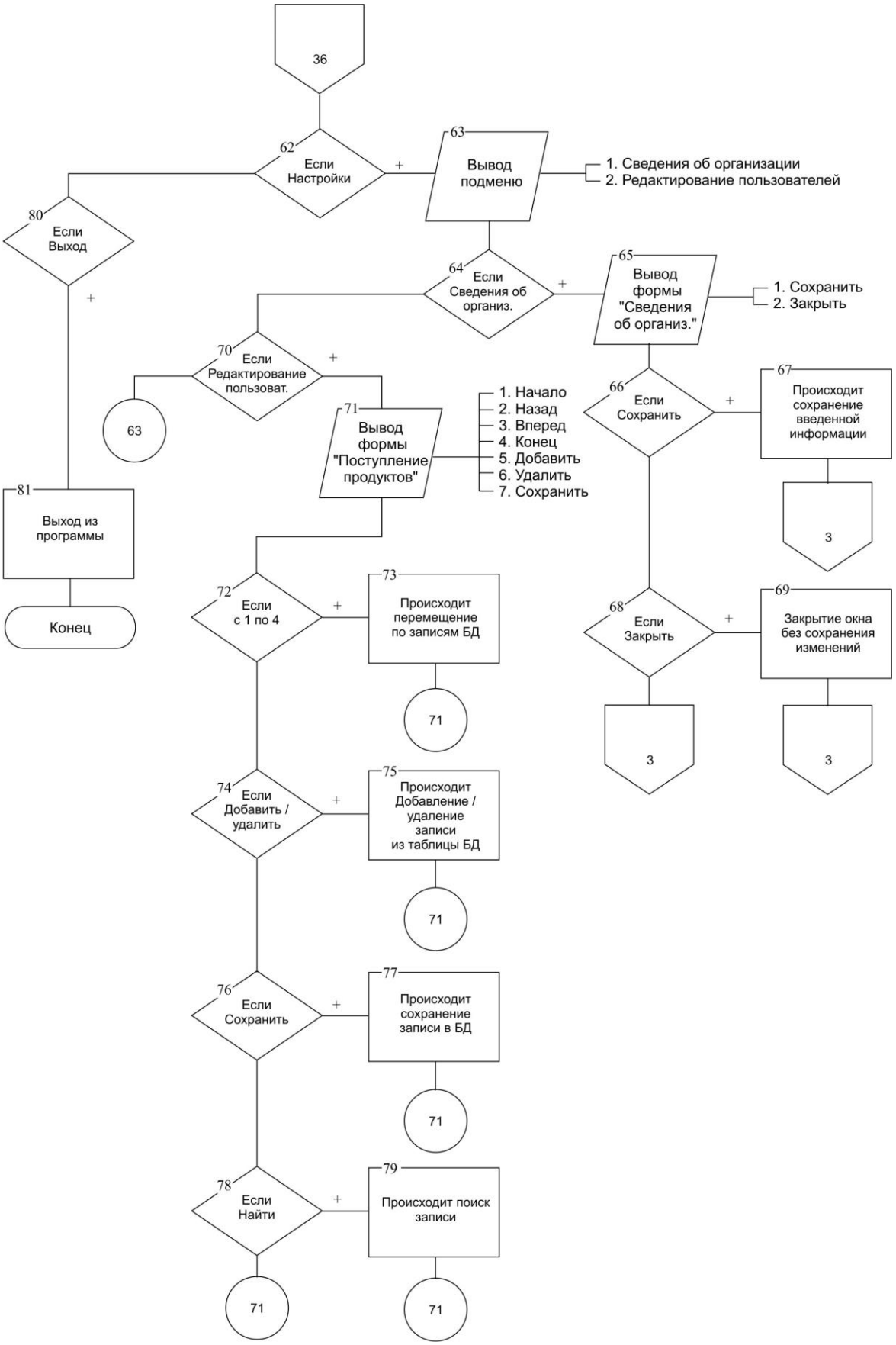


Рисунок 3.1 – Блок схема алгоритма взаимодействия с БД

3.2 Логическое проектирование базы данных

Перед началом проектирования БД ее необходимо визуализировать, т.е. представить ее структуру, таблицы, строки и связи между ними.

Проектирование баз данных — процесс создания визуальной схемы базы данных и определения внутреннего строения входящих в нее элементов и необходимых ограничений целостности.

Логическое проектирование БД — это процедура создания визуальной схемы базы данных на основе конкретной формальной модели данных, например, реляционной модели данных. Для реляционной модели данных строится даталогическая модель — набор схем связей и отношений, обычно с указанием первичных ключей, а также «связей» между отношениями, представляющих собой внешние ключи.

Преобразование первоначальной концептуальной модели базы в логическую модель, как правило, осуществляется по формальным правилам. Этот этап может быть в значительной степени автоматизирован с помощью специальных средств визуального и физического проектирования БД.

На этапе логического проектирования учитывается специфика конкретной модели данных, но может не учитываться специфика конкретной СУБД.

Поскольку у всех объектов в сети кофеен одинаковые поставщики, меню, цены, продажи и общая база сотрудников, целесообразно создать одну единую базу, размещенную на сервере, к которой будут обращаться «клиенты».

«Клиент» это пользователь ИС (официант, барист, менеджер, бухгалтер, кассир и т.п.), работающий в определенной точке (кофейне) через «клиента» (модуль учетной программы, рассматриваемой далее). Клиенты распределены, а сама база – централизована.

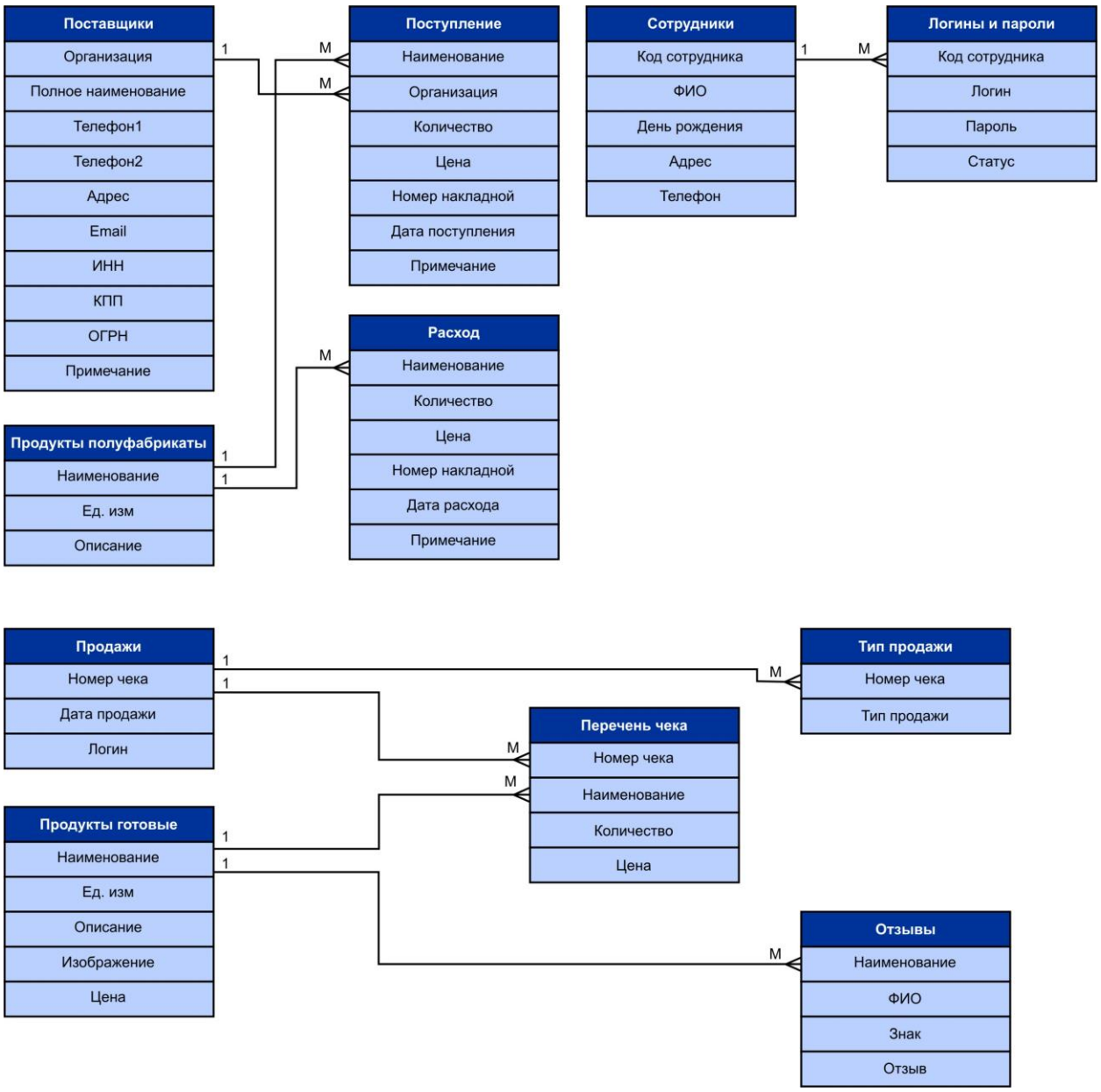


Рисунок 3.2 - Информационно-логическая модель с определенными сущностями и связями между таблицами

Данная БД содержит 11 сущностей. Рассмотрим основные из них и связь между ним.

Сущность поставщики содержит 10 атрибутов. Организация, полное наименование, телефон 1, телефон 2, адрес, email, ИНН, КПП, ОГРН, примечание. В этой сущности содержатся основные сведения о поставщиках (для целей бухучета). Между наименованием организации и сущности

«Поставщики» и сущности «Поступления» имеется связь для однозначной идентификации поставщика.

Сущность Поступление содержит 7 атрибутов: наименование товара, поставщика, количество, цена закупки, номер накладной, дата поступления, примечание.

Сущность Сотрудники содержит 5 атрибутов для целей кадрового учета. Данная сущность связана с сущностью Логины и пароли.

Сущность Полуфабрикаты содержит 3 атрибута, наименование связано с сущностью Расход. В Сущности расход ведется учет продаж для целей контроля расхода полуфабрикатов и дозаказа их на удаленном складе.

Все связи в БД являются связями один-ко-многим. Связи между сущностями являются идентифицирующими, так как вторичный ключ попадает в атрибуты первичного ключа другой сущности.

3.3 Физическое проектирование базы данных

Проектирование базы данных для данного проекта осуществлялось в MySQL Workbench 5.5.

Работа с данным графическим клиентом начинается с формирования локальной модели базы данных. Это не физическая база данных, а виртуальная схема с полями и связями между полями, а так же другими элементами базы данных.

Модель строится аналогичным образом, как это делается в MS ACCESS. Логически данные представлены в виде объектов. К основным элементам структуры БД относятся объекты, представленные в таблице 1.1.

Таблица 1.1 - структура таблиц базы данных

Наименование поля	**Содержание поля**	**Тип поля**	**Размер поля**	**Значение по умолчанию**	**Условие на значение**	**Ключ или индекс**
Логины и пароли (LogPass)						
Login	Логин	VARCHAR	50		Уникальное NOT NULL	ПК, Индекс

Наименование поля	**Содержание поля**	**Тип поля**	**Размер поля**	**Значение по умолчанию**	**Условие на значение**	**Ключ или индекс**
Password	Пароль	VARCHAR	50	md5(rand())		
Status	Статус	INT	11	0	NOT NULL	
Поставщики (Post)						
Org	Краткое наименование организации	VARCHAR	150		Уникальное NOT NULL	ПК, Индекс
Name	Полное наименование организации	VARCHAR	200		Уникальное	
Tel1	Телефон	VARCHAR	20			
Tel2	Дополнительный телефон	VARCHAR	20			
Adres	Адрес	VARCHAR	255			
E-mail	E-mail	VARCHAR	50			
INN	ИНН	VARCHAR	12			
KPP	КПП	VARCHAR	10			
OGRN	ОГРН	VARCHAR	15			
Prim	Примечание	MEDIUMTEXT				
Продукты полуфабрикаты (ProdPol)						
Name	Наименование продукта	VARCHAR	150		Уникальное NOT NULL	ПК, Индекс
EdIzm	Единица измерения	VARCHAR	15			
Opis	Описание	MEDIUMTEXT	20			
Продукты готовые (ProdGot)						
Name	Наименование продукта	VARCHAR	150		Уникальное NOT NULL	ПК, Индекс
EdIzm	Единица измерения	VARCHAR	15			
Opis	Описание	MEDIUMTEXT	20			
Image	Изображение	MEDIUMBLOB				
Cena	Цена	DECIMAL	10,2		NOT NULL	
Продажи (Prod)						
NomChecka	Номер чека	INT	150	Автоинкре-ментное	Уникальное NOT NULL	ПК, Индекс
DataProd	Дата продажи	VARCHAR	10			
Login	Логин	VARCHAR	50			
Перечень чека (Perchecka)						
NomChecka	Номер чека	INT				Индекс
Name	Наименование продукта	VARCHAR	150			Индекс
Kolvo	Количество	INT				
Cena	Цена	DECIMAL	10,2			
ID	ID	BIGINT		Автоинкре-ментное	Уникальное NOT NULL	ПК, Индекс
Поступление (Postup)						
Name	Наименование	VARCHAR	150			Индекс

Наименование поля	**Содержание поля**	**Тип поля**	**Размер поля**	**Значение по умолчанию**	**Условие на значение**	**Ключ или индекс**
	продукта					
Org	Организация-поставщик	VARCHAR	150			Индекс
Kolvo	Количество	INT				
Cena	Цена	DECIMAL	10,2			
NomNakl	Номер накладной	VARCHAR	50			
DatePost	Дата поступления	VARCHAR	10			
Prim	Примечание	VARCHAR	255			
ID	ID	INT		Автоинкре-ментное	Уникальное NOT NULL	ПК, Индекс
Отзывы (Otziv)						
ID	ID	INT		Автоинкре-ментное	Уникальное NOT NULL	ПК, Индекс
Name	Наименование продукта	VARCHAR	150			Индекс
FIO	ФИО	VARCHAR	100			
Znak	Знак	VARCHAR	1			
Otz	Отзывы	VARCHAR	255			
Расход (Rashod)						
Name	Наименование продукта	VARCHAR	150			Индекс
Kolvo	Количество	INT				
Cena	Цена	DECIMAL	10,2			
NomNakl	Номер накладной	VARCHAR	50			
DateRash	Дата поступления	VARCHAR	10			
Prim	Примечание	VARCHAR	255			
ID	ID	INT		Автоинкре-ментное	Уникальное NOT NULL	ПК, Индекс
Сотрудники (Sotr)						
KodSotr	Код сотрудника	INT		Автоинкре-ментное	Уникальное NOT NULL	ПК, Индекс
FIO	Количество	VARCHAR	100			
Adres	Цена	VARCHAR	255			
Tel	Номер накладной	VARCHAR	22			
Birsday	Дата поступления	VARCHAR	10			
Тип продаж (TipProd)						
ID	ID	INT		Автоинкре-ментное	Уникальное NOT NULL	ПК, Индекс
NomChecka	Номер чека	INT				
TipProdaj	Тип продаж	VARCHAR	50			

Структура таблиц БД состоит из следующих основных элементов. описанных ниже.

Таблицы Логины и пароли (LogPass) и Сотрудники (Sotr) содержат сведения о логине, пароле и статусе (активен или нет) учетной записи и ее роли. При этом по результатам тестирования принято решение в целях безопасности хешировать пароли алгоритмом md5(), а пароль генерировать генератором псевдослучайных чисел rand(). Для обеспечения совместимости, это будет реализовано в последующих версиях программы, с добавлением дополнительного признака «тип пароля»: открытый или хешированный. Эта база ведется централизованно, на удаленном сервере и заполняется руководством организации из главного офиса. Основанием для внесения сведений является приказ по кадрам.

Таблица Поступление (Postup) содержит сведения о поступлениях товаров от поставщиков на удаленный склад, и содержит наименование продуктов, их артикулы, сведения о поставщиках для целей бухучета.

Таблица Отзывы (Otziv) содержит отзывы, которые переносятся из бумажных носителей, таких как книга отзывов и предложений, или устных замечаний, зафиксированных документально. База ведется для целей ведения претензионной работы.

Таблица Расход (Rashod) ведется локально и служит для целей учета остатков сырья на локальном складе и оформления заказов на поставки на удаленный склад.

Таблица Тип продаж (TipProd) содержит сведения для целей налогового учета и содержит номер кассового чек и тип продажи. Данные передаются в центральную БД по окончанию смены для генерации статистики и отчетов по продажам.

База должна соответствовать реляционной модели данных, для этого необходимо построить диаграмму отношений (EER-диаграмму), приведенную на рисунке 3.3.

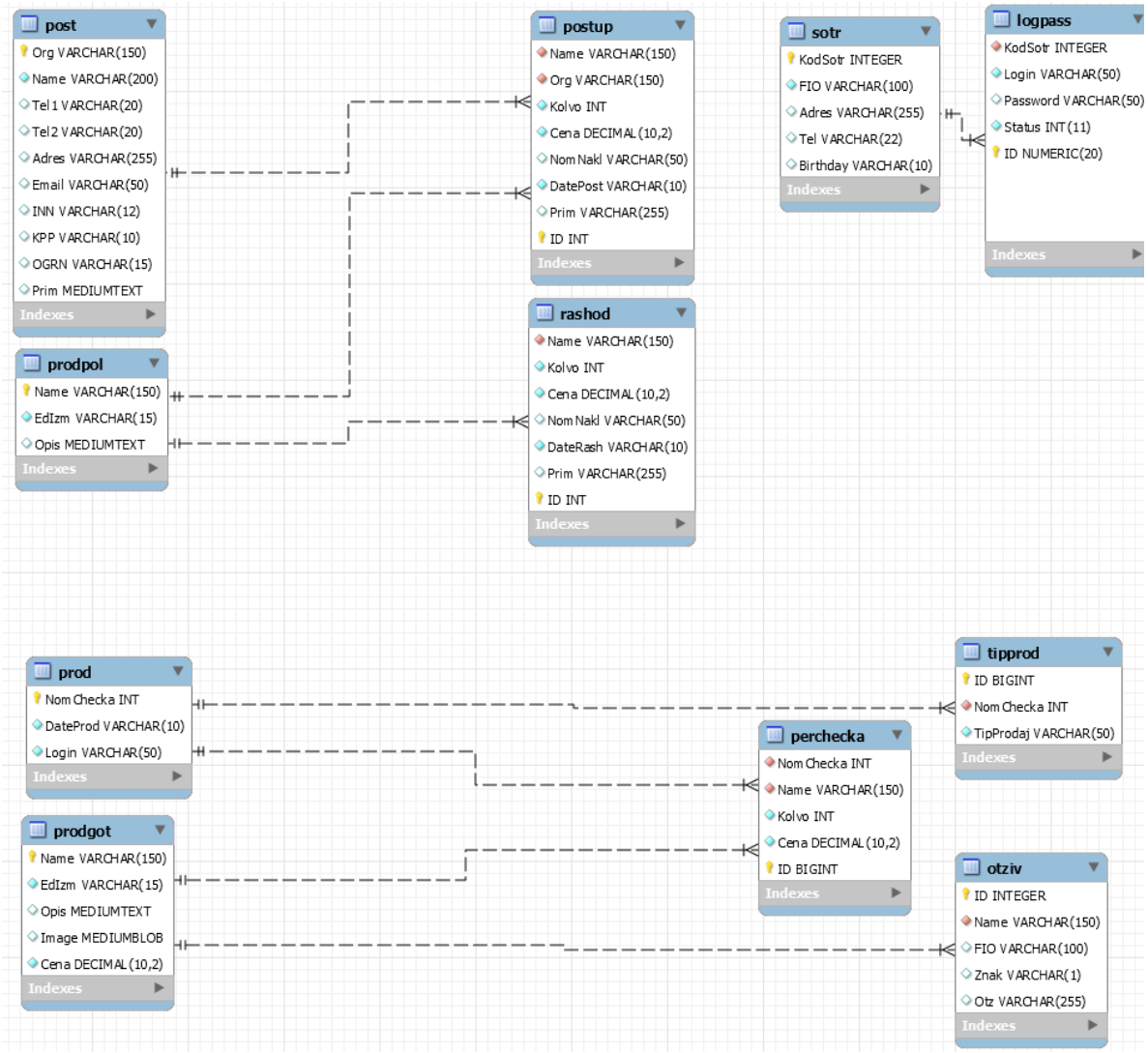


Рисунок 3.3 – EER-диаграмма (диаграмма отношений)

Диаграмма отношений позволяет проверить соответствие базы реляционной модели, ее логической целостности и правильности построения связей.

После построения модели базы данных необходимо сформировать скрипт на языке SQL. Данный скрипт можно сохранить в файл. Смысл в данной операции в том, чтобы не набирать команды вручную. Графическое проектирование в значительной степени экономит время и уменьшает вероятность ошибок.

Следующим этапом будет являться формирование физической базы данных на основании созданного скрипта, фрагмент которого представлен на рисунке 3.4.

```
USE `database`;

CREATE  TABLE IF NOT EXISTS `database`.`logpass` (
  `Login` VARCHAR(50) NOT NULL ,
  `Password` VARCHAR(50) NULL DEFAULT NULL ,
  `Status` INT(11) NOT NULL DEFAULT '0' ,
  PRIMARY KEY (`Login`) ,
  UNIQUE INDEX `Login_UNIQUE` (`Login` ASC) )
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;


CREATE  TABLE IF NOT EXISTS `database`.`prodgot` (
  `Name` VARCHAR(150) NOT NULL ,
  `EdIzm` VARCHAR(15) NOT NULL ,
  `Opis` MEDIUMTEXT NULL DEFAULT NULL ,
  `Image` MEDIUMBLOB NULL DEFAULT NULL ,
  `Cena` DECIMAL(10,2) NOT NULL ,
  PRIMARY KEY (`Name`) ,
  UNIQUE INDEX `Name_UNIQUE` (`Name` ASC) )
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;

CREATE  TABLE IF NOT EXISTS `database`.`prod` (
  `NomChecka` INT NOT NULL AUTO_INCREMENT ,
  `DateProd` VARCHAR(10) NOT NULL ,
  `Login` VARCHAR(50) NOT NULL ,
  PRIMARY KEY (`NomChecka`) )
ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8;

```

Рисунок 3.4. – скрипт импорта структуры БД

Данный скрипт содержит SQL-запросы на проверку существование таблицы и ее создание при отсутствии. Следующим этапом является создание полей таблиц и указание их типов данных – текстовых (string), числовых (int) и т.п., а также их предельная длина и указание на допустимость отсутствия значения.

В дальнейшем выбирается первичный ключ, например, порядковый номер записи в таблице.

В завершение выбирается тип таблиц и ее кодировка.

Фрагмент скрипта импорта первоначальной структуры БД приведен в приложении А.

3.4 Разработка архитектуры приложения и программного кода

В первую очередь перед написанием программного кода необходимо построить диаграмму компонентов, которая описывает особенности физического и логического представления проектируемой информационной системы. Диаграмма компонентов приложения позволяет определить внутреннюю архитектуру приложения и строение разрабатываемой информационной системы, установив зависимости и связи между программными компонентами.. Во многих средах разработки наименования модуля или компонента программы соответствует подключаемому файлу. Основными графическими элементами при построении диаграммы компонентов являются компоненты, интерфейсы и зависимости между ними. В качестве среды разработки ПО выбрана платформа IntelliJ IDEA Community Edition 2019.2.3 x64.

Рисунок 3.5 – Упрощенное представление архитектуры приложения[16]

В самом начале определен необходимый набор основных подключаемых модулей, таких как модули для соединения с БД, модули для построения графического интерфейса (GUI).

16 Мартин Фаулер. Архитектура корпоративных программных приложений, 2016

Следующим этапом идет определение списка необходимых специализированных библиотек. К таким библиотекам относятся библиотеки для построения графиков и отчетов, для специализированного взаимодействия по API с операторами фискальных данных и платежными системами. На рисунке 3.6 представлен фрагмент исходного кода, отвечающего за инициализацию пользовательского интерфейса приложения.

```
import java.io.*;
import javax.swing.JFrame;
import javax.swing.SwingUtilities;
import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartPanel;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.plot.PlotOrientation;
import org.jfree.data.xy.XYDataset;
import org.jfree.data.xy.XYSeries;
import org.jfree.data.xy.XYSeriesCollection;
public class Main {
  public static void main(String[] args) {
    XYSeries series = new XYSeries("sin(a)");

    for(float i = 0; i < Math.PI; i+=0.1){
      series.add(i, Math.sin(i));
    }

    XYDataset xyDataset = new XYSeriesCollection(series);
    JFreeChart chart = ChartFactory
        .createXYLineChart("y = sin(x)", "x", "y",
                           xyDataset,
                           PlotOrientation.VERTICAL,
                           true, true, true);
    JFrame frame =
        new JFrame("MinimalStaticChart");
    frame.getContentPane()
        .add(new ChartPanel(chart));
    frame.setSize(400,300);
    frame.show();
  }
}
```

Рисунок 3.6 – Фрагмент модуля, инициирующего создание пользовательского интерфейса приложения

Данный модуль импортирует необходимые для работы модули и библиотеки, и создает окно определенного размера. Его графические элементы подгружаются из файлов, а текстовые значения из переменных.

Первым модулем разрабатывается «прием заказов», который содержит такие сущности и их содержимое, как номер кофейни, номер столика, статус брони, статус заказа, состав заказа, чек. Например, для работы модуля приема заказа необходимо создать массив (array), в котором ключом будет порядковый номер в заказе, а значениями – наименования товарной позиции, количество, внутренний id товара.

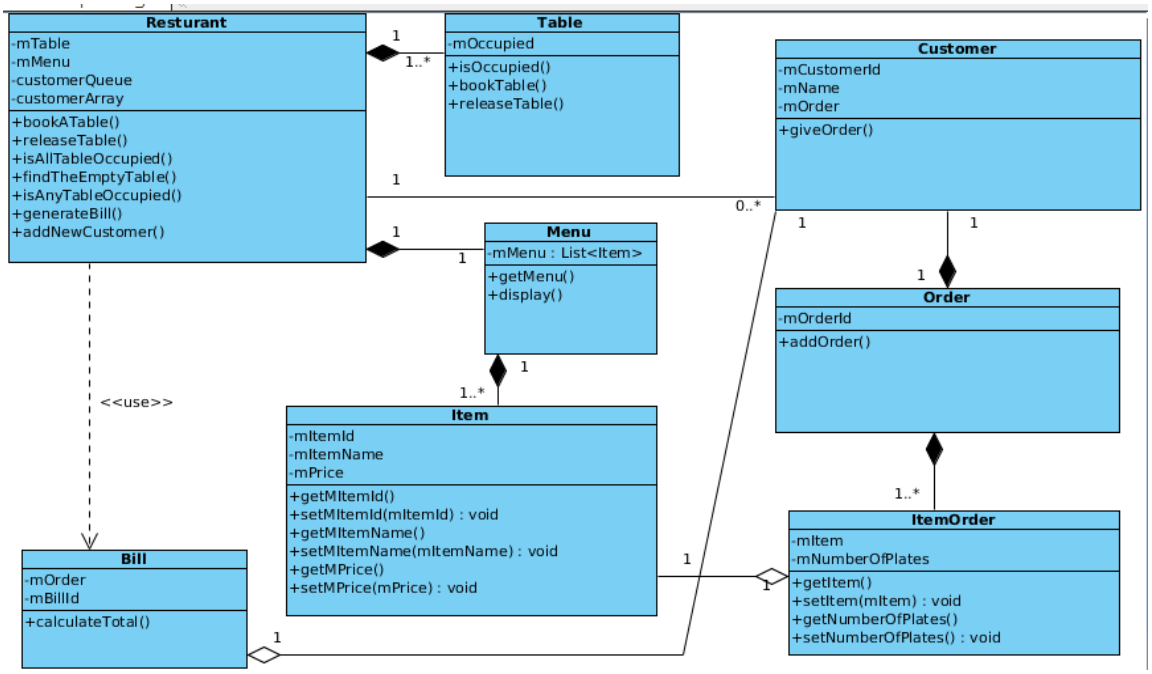


Рисунок 3.7 – Диаграмма компонентов модуля «прием заказов»

На представленном выше рисунке визуализируется диаграмма компонентов модуля приема заказов. В ней установлена четкая связь между рестораном Restaraunt и определенном столиком Table в нем, статусом резервации is_occuped, а также обслуживающем его официанте, составе заказа Order. Данный модуль формирует массив (array), по итогам которого формируется SQL-запрос на добавление в БД новой записи о заказе, его составе, продолжительности исполнения, а также вычете из остатков сырья определенного количества наименований.

```
package tgu_disser;
public class Customer{
private int mCustomerId;
private Order mOrder;
private String mCustomerName;
public Customer(){
mCustomerId = 0;
mOrder = new Order(mCustomerId);
}
public Customer(int customerId){
mCustomerId = customerId;
mOrder = new Order(customerId);
}
public int getCustomerId() {
return mCustomerId;
}
public void setCustomerId(int mCustomerId) {
this.mCustomerId = mCustomerId;
}
public void giveOrder(Item item, int numberOfPlates){

ItemOrder newItemOrder = new ItemOrder(item,  numberOfPlate
mOrder.addOrder(newItemOrder);
}
public void iAmDone(){
}
public Order getOrder() {
return mOrder;
}
public void setOrder(Order mOrder) {
this.mOrder = mOrder;
}
public String getCustomerName() {
return mCustomerName;
}
public void setCustomerName(String mCustomerName) {
this.mCustomerName = mCustomerName;
}
}
```

Рисунок 3.8 – Фрагмент программного кода модуля «прием заказов»

Следующим модулем идет модуль проверки остатков на удаленном складе при их приближении к исчерпанию. При оформлении заказа происходит проверка наличия необходимого сырья на локальном складе кофейни. При его наличии, заказ передается в производство. После производства, заказ переходит к конечному потребителю и ведется его учет.

При отсутствии в достаточном количестве необходимого сырья на локальном складе, заказ не подтверждается. После этого формируется автоматическая заявка на поставку с удаленного склада.

После получения заявки на удаленный склад, происходит проверка наличия сырья на нем. При его наличии, оно поставляется в определенную кофейню. При его отсутствии, оформляется заказ у поставщиков.

Аналогично система ведет себя при приближении остатков продукции к концу. В данном случае все необходимые действия по оформлению заказов производятся заблаговременно, без отказа в приеме заказа у клиента.

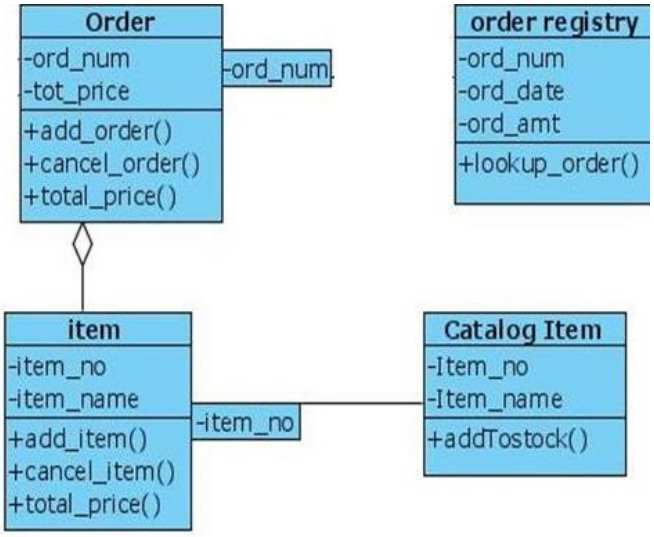


Рисунок 3.9 – Диаграмма компонентов модуля «проверка на локальном/удаленном складе»

На представленном выше рисунке диаграммы классов описана архитектура класса по заказу на удаленном складе. Проверяются остатки на удаленном складе и при их отсутствии вызывается класс заказа у поставщиков. При положительном остатке, формируется массив заказа Order, который содержит сведения о необходимых для поставки компонентах Item и их количестве total_price, а также сведения о конкретном подразделении, принимающем сырье. Полученный массив преобразуется в запрос и отправляется в базу, к которой происходит регистрация заказа. По итогам обработки класс order_registry формирует запрос к к БД на присвоение статуса исполненного.

```
public void setOrder(Order mOrder) {
this.mOrder = mOrder;
}
public float calculateTotal(){
float retValue = 0;
Iterator<ItemOrder> it = mCustomer.getOrder().getItemOrder().iterator();
while (it.hasNext() == true){
ItemOrder element = it.next();
retValue+= (element.getItem().getItemPrice())* (element.getNumberOfPlates());
}
return retValue;
}
public int getBillId() {
return mBillId;
}
public void setBillId(int mBillId) {
this.mBillId = mBillId;
}
}
```

Рисунок 3.10 – Фрагмент программного кода модуля «проверка остатков»

Данные из массива отправляются запросом к БД для проверки на наличие остатков указанных товаров. В случае положительного ответа БД, создается новая запись в таблице заказов.

```
SELECT * from products
WHERE id=(id продукта)
AND ostatok >= (кол-во)

INSERT INTO Sales
VALUES (1,'Capucchino',1,1);
```

Рисунок 3.11 – Фрагмент запроса к БД на языке SQL

Полученный ответ от базы обрабатывается клиентом, после чего формируется диалоговое окно. После получения результата выполнения запроса в пользовательский интерфейс выводится результат его выполнения в виде текста. Текст хранится в переменных типа String.

Фрагменты исходного кода основных модулей программы вынесены в приложение Б.

Выводы по 3 главе

В ходе выполнения главы произведено физическое и логическое моделирование БД, написание программного кода. База должна

соответствовать реляционной модели данных, для этого необходимо построить диаграмму отношений. Перед началом проектирования БД ее необходимо визуализировать, т.е. представить ее структуру, таблицы, строки и связи между ними.

Далее производится логическое проектирование, в ходе которого логическое содержание базы с определенными сущностями и связями между таблицами. После чего производится описание сущностей. Завершается проектирование построением диаграммы отношений.

Перед написанием программного кода необходимо построить диаграмму компонентов, которая описывает особенности физического представления программного обеспечения. Далее выполняется написание исходного кода. Каждая программа начинается с импорта основных модулей. Далее подгружаются библиотеки, например, для построения пользовательского интерфейса и его содержание, такого как меню, графики. Следующим этапом идет написание компонентов приложения. Для этого используются функции, переменные и массивы. Компоненты разрабатываются в соответствии с бизнес-логикой и требованиям к функциям ПО.

ГЛАВА 4 ТЕСТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ

4.1 Цель, задачи и методика тестирования

При тестировании программного продукта подтверждается работа бизнес-правил, установленных при проектировании приложения и базы данных, а именно, осуществляется каскадное обновление и удаление записей в таблицах базы данных, попытка добавления некорректных значений, реакция на ошибки и т.п.

Для проведения тестирования распределенной информационной системы сети общественного питания разработана программа и методика, представленные ниже.

Объектом тестирования является программное обеспечение (ПО) распределенной информационной системы сети общественного питания, а именно клиентский модуль, выполняемый на базе виртуальной машины Java и базы данных Mysql. Предъявляемое для тестирования ПО должно быть представлено в составе, достаточном для проведения полнофункционального тестирования в соответствии с предусмотренными программой и методикой тестирования.

Рисунок 4.1 - Общая схема основных видов тестирования

Целью тестирования распределенной информационной системы сети общественного питания является:

- проверка ПО распределенной информационной системы сети общественного питания на соответствие утвержденному проекту разработки и внедрения системы;
- проверка работоспособности ПО распределенной информационной системы сети общественного питания и выявление возможных ошибок;
- проверка качества интерфейса пользователя ПО распределенной информационной системы сети общественного питания;
- проверка качества информационного обмена между отдельными модулями распределенной информационной системы сети общественного питания.
- проверка информационной безопасности

Тестирование состоит из двух этапов – автоматического и ручного.

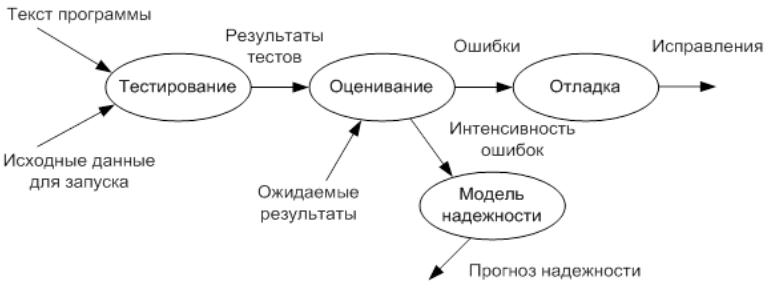


Рисунок 4.2 – Общий алгоритм тестирования

4.2 Автоматическое тестирование

Автоматизированное тестирование ПО — процесс тестирования п, при котором ключевые функции и шаги теста, такие как запуск, инициализация, выполнение, анализ и выдача результата, производятся автоматически специальным ПО с помощью инструментов для автоматизированного тестирования. Тестирование распределенной информационной системы сети общественного питания проводится по методу автоматизированного тестирования с использованием программы jUnit.

Автоматизированное тестирование основано на использовании специального ПО для контроля выполнения тестов и сравнения фактических полученных результатов с прогнозируемыми результатами. В процессе автоматизированного тестирования используются специальное приложение – менеджер тестирования.

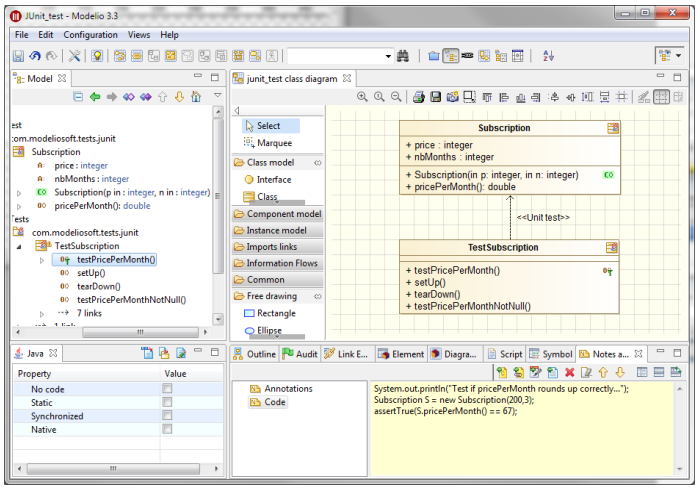


Рисунок 4.3 – Программа автоматического тестирования

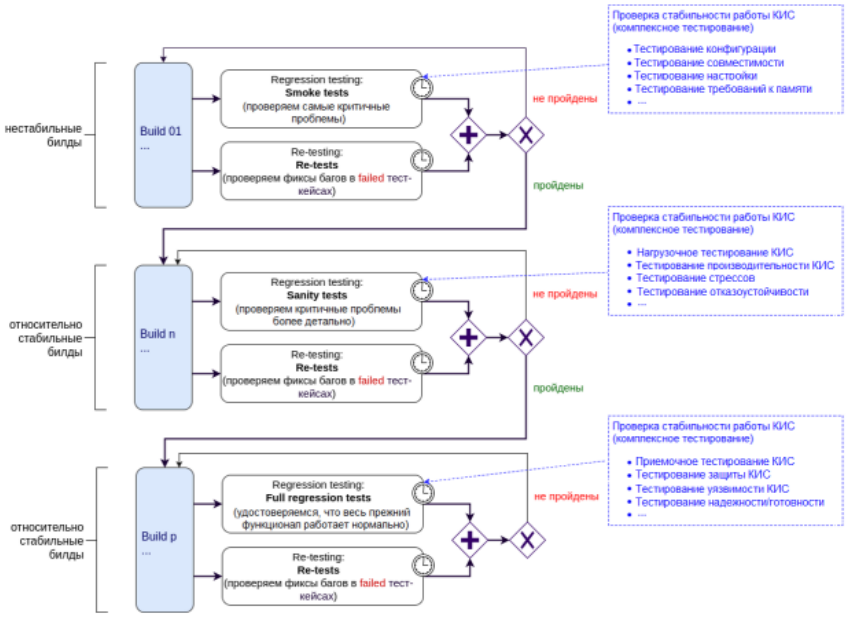


Рисунок 4.4 - Сценарий комплексного тестирования

Название:	Тест авторизации	
Функция:	Вход в программу	
Действие	Ожидаемый результат	Результат теста:
Предусловие:		
Откройте программу	Открытое главное окно программы	пройден
Шаги теста:		
Введите логин и пароль пользователя	Поля заполнены	пройден
Нажмите кнопку «Вход»	Открытие главного окна программы	пройден

Рисунок 4.5 – Тест авторизации

Название:	Тест меню программы	
Функция:	Переход в другие окна приложения	
Действие	Ожидаемый результат	Результат теста:
Предусловие:		
Откройте программу	Открытое главное окно программы	пройден
Шаги теста:		
Выберите пункты из предоставленного меню	Модули для открытия выбраны	пройден
Нажмите кнопку перехода в другой модуль	Переход на выбранный модуль осуществлен	пройден

Рисунок 4.6 – Тест главного меню

По результатам автоматического тестирования были выявлены и исправлены некоторые ошибки, связанные с недостаточной фильтрацией данных при запросах к базе, таких как передача текста (String) в поле, тип которого число (int) и т.п., что могло бы привести к падению программы или ее некорректным результатам работы.

Нагрузочное тестирование (load testing) – позволяет оценить поведение распределенной системы при возрастающей нагрузке и большом количестве пользователей. Целью нагрузочного тестирования является определение максимальной нагрузки, которую может выдержать система.

Во время нагрузочного тестирования могут осуществляться операции, которые позволяют более точно измерить производительность и определить «узкое место» системы:

- затрачиваемое время при определенных интенсивностях выполнения этих операций;
- определение количества клиентов, одновременно работающих с приложением;
- определение границ приемлемой производительности и быстродействия при увеличении нагрузки.[17]

В роли нагрузки может выступать количество пользователей (клиентов), а также количество операций на сервере.

Рисунок 4.7 - Динамика подачи нагрузки

По результатам диаграммы видно, что нагрузка на процессор и время отклика системы удовлетворяет требованиям производительности при расчетной нагрузке 30 пользователей.

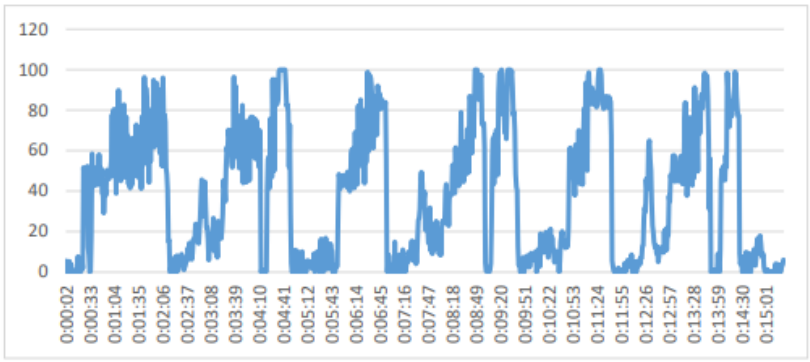


Рисунок 4.8 - Потребление ресурсов CPU сервера БД

[17] Кулямин В. В. Тестирование на основе моделей, URL: http://panda.ispras.ru/~kuliamin/lectures-mbt/Lecture04.pdf. (дата обращения 07.12. 2019)

4.3 Ручное тестирование пользовательского интерфейса

Ручное тестирование проводится через пользовательский интерфейс ПО. В ходе ручного тестирования осуществляется каскадное обновление и удаление записей в таблицах базы данных, попытка добавления некорректных значений, реакция на ошибки, такие как деление на ноль, попытка введения буквенных значений и поля, предназначенные только для чисел (integer) и наоборот, проверяется работа всех модулей программы и выполняется сравнение их результата работы с ожидаемым.

Так, обновление наименования продукта в таблице "Продукты полуфабрикаты" приводит к обновлению наименования этого же продукта в таблице "Поступление". Соответственно, при удалении продукта из справочной таблицы, происходит удаление всех записей в учетной таблице, в которых наименование равно наименованию удаленного продукта из справочной таблицы.

Если изменить название или удалить организацию из таблицы "Поставщики", то это изменение аналогичным образом отразится на таблице "Поступление".

Точно такие же бизнес-правила организованы для таблиц "Продажи", "Продукты готовые" и "Перечень чека". Таблица "Продажи" и таблица "Продукты готовые" являются справочными таблицами и связаны соотношением один ко многим с таблицей "Перечень чека". Точно так же, изменение в этой связке таблиц отражается на подчиненной таблице, изменяя или удаляя из нее записи в соответствии с правилами каскадного обновления и удаления.

Все ошибки программного продукта, включая потенциальные ошибки пользователей при неправильном или ошибочном вводе обрабатываются программным исключением с выводом соответствующих предупреждений.

Например, при незаполненных обязательных к заполнению полях и попытке сохранить запись на экран выводится предупреждение, и программа останавливает сове действие до момента выполнения ее требований.

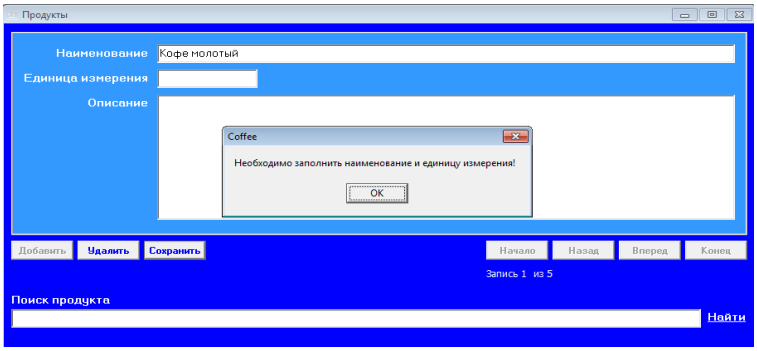


Рисунок 4.9 – Предупреждение о незаполненном поле

При вводе неуникального значения в ключевое поле пользователю выводится соответствующее сообщение. Реакция ПО на попытку добавления пустого значения (поле БД «IS NOT NULL», т.е. не может быть пустым).

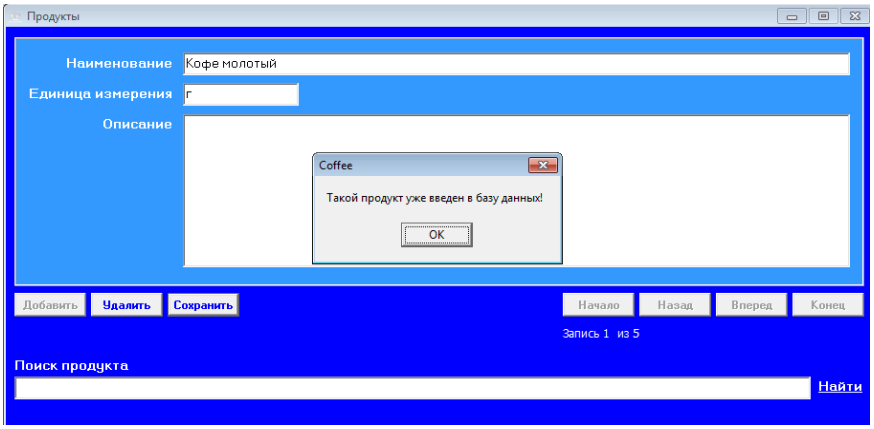


Рисунок 4.10 – Ошибка дублирования данных. Реакция на добавление записи с повторным ID или названием

Видимой (клиентской) частью РИС является пользовательский модуль, обращающийся к серверу. Далее будет рассмотрен пользовательский интерфейс при работе со стационарного компьютера администратора, менеджера или бухгалтера определенной кофейни.

Данный клиентский модуль обращается по зашифрованному соединению к серверу БД, хранящего единую базу продукции, поставщиков,

сотрудников и т.п. для всей сети. Интерфейс создан с помощью библиотеки SWING для JAVA, которая позволяет создать пользовательский интерфейс (GUI).

При запуске программы появляется окно запуска и выбора пользователя.

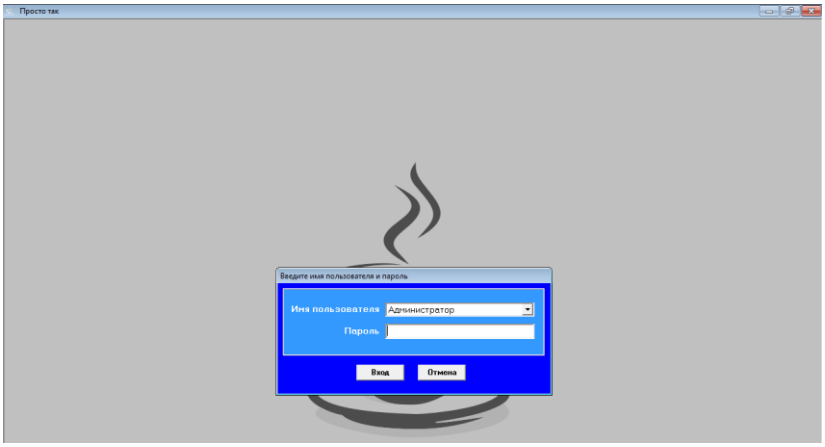


Рисунок 4.11 – Окно авторизации в системе

Пароль для администратора первоначальный генерируется функцией rand(), который впоследствии можно изменить. Хранится в БД в хешированном виде (MD5).

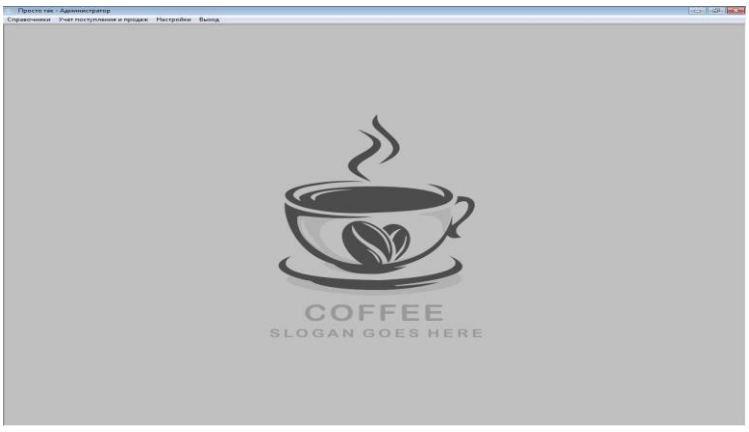


Рисунок 4.12 – Главное окно программы

В верхней части главного окна приложения имеется меню для доступа к другим функциям программы.

Основное меню содержит пункты, которые имеют такое же название, как и таблицы базы данных.

Каждый пункт этого меню, вызывает окно, информация в котором отображается из одноименной таблицы базы данных.

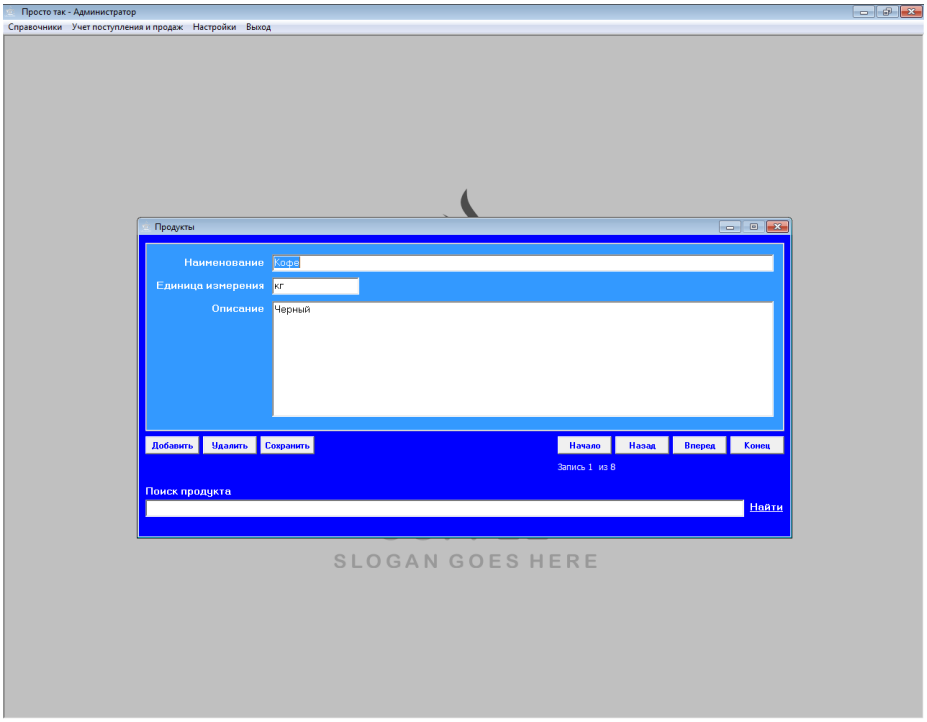


Рисунок 4.13 – Окно "Продукты"

Окно "Продукты" предназначено для ввода и хранения информации о полуфабрикатах и товарах. Добавление нового продукта осуществляется одноименной кнопкой навигации данного окна. В поле "Наименование" вводится наименование продукта, в поле "Единица измерения" указывается единица измерения продукта. Это может быть килограммы, литры, граммы и так далее. Поле "Описание" предназначено для хранения любой информации о продукте.

Для сохранения добавленной записи необходимо нажать одноименную кнопку. Кнопка "Удалить" служит для удаления записи о продукте из файла. Кнопки "Начало", "Назад", "Вперед" и "Конец" предназначены для перемещения по записям таблицы.

В нижней части окна предусмотрен поиск продукта по наименованию. Поиск допускается по неполному совпадению. Это удобно, когда не имеется возможность вспомнить точное наименование продукта. После ввода в строку "Поиск продукта", необходимо нажать на кнопку "Поиск". При этом название кнопки изменится на "Отключить фильтр".

Если поиск окажется неудачным, то результат будет нулевым. Если же будет найдена запись, то она покажется на экран.

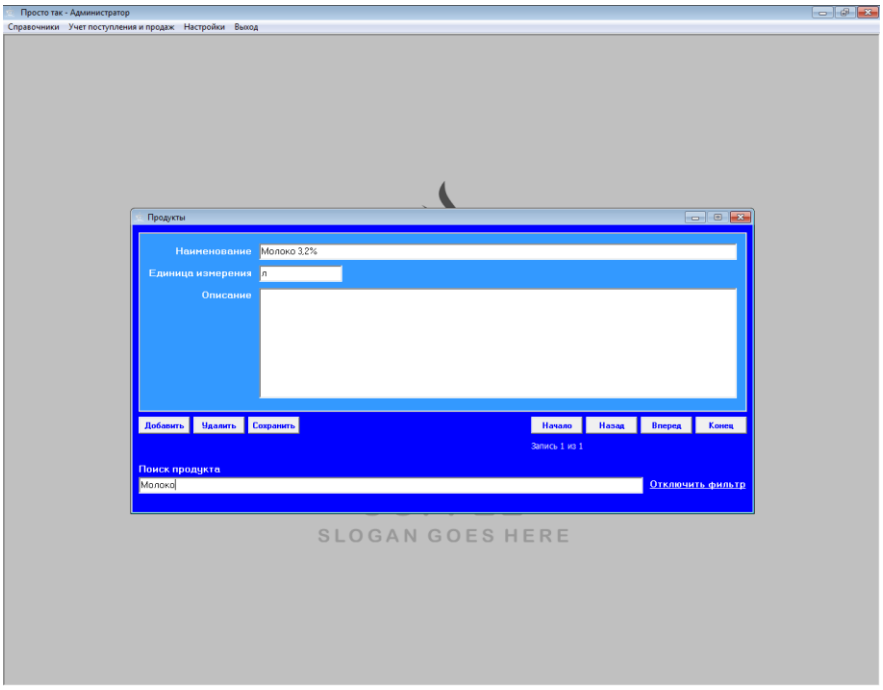


Рисунок 4.14 – Поиск продукта по неполному ключу

Если же в результате поиска под введенный критерий будет подходить несколько записей, то индикатор количества записей укажет, сколько записей найдено. По найденным записям так же можно перемещаться с помощью кнопок навигации, как и по записям при отключенном фильтре.

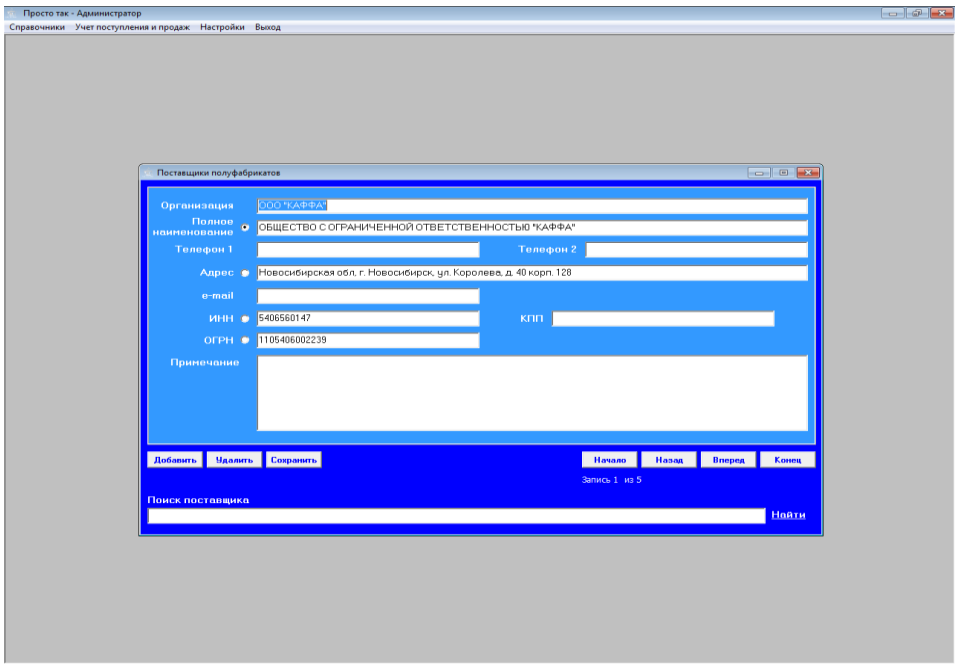


Рисунок 4.15 – Окно "Поставщики полуфабрикатов"

Окно "Поставщики продуктов" содержит в себе окно с информацией о поставщиках. Поле "Организация" содержит краткое наименование организации. Оно удобно для отображения в других окнах. Поле "Полное наименование" содержит полное наименование организации.

Все остальные поля содержат информацию, необходимую для идентификации организации по правилам российского законодательства. В нижней части экрана так же расположен поиск поставщика.

Чтобы найти поставщика необходимо выставить переключатель кнопки по тому критерию, по которому планируется поиск. Например, если необходимо найти поставщика по его ИНН, то следует выставить переключатель напротив ИНН и в строку "Поиск поставщика" ввести его ИНН. Затем нажать "Найти".

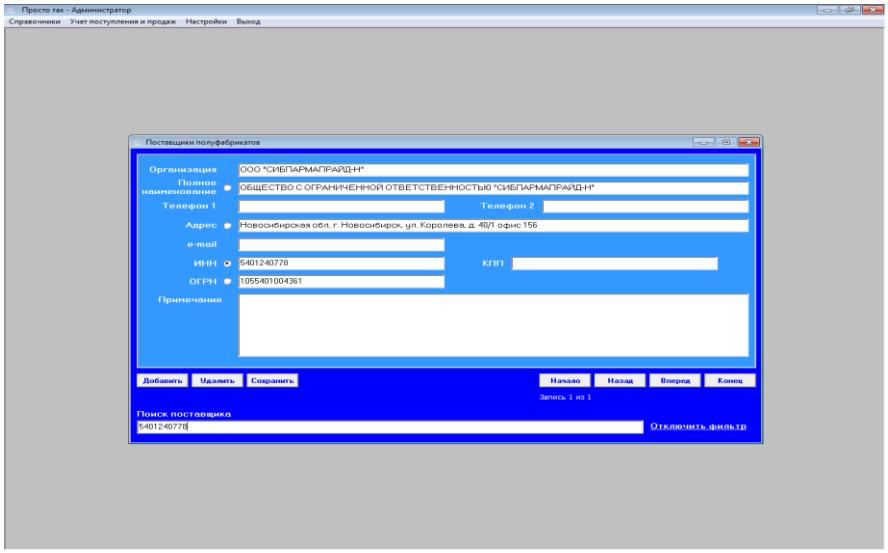


Рисунок 4.16 – Поиск поставщика по ИНН.

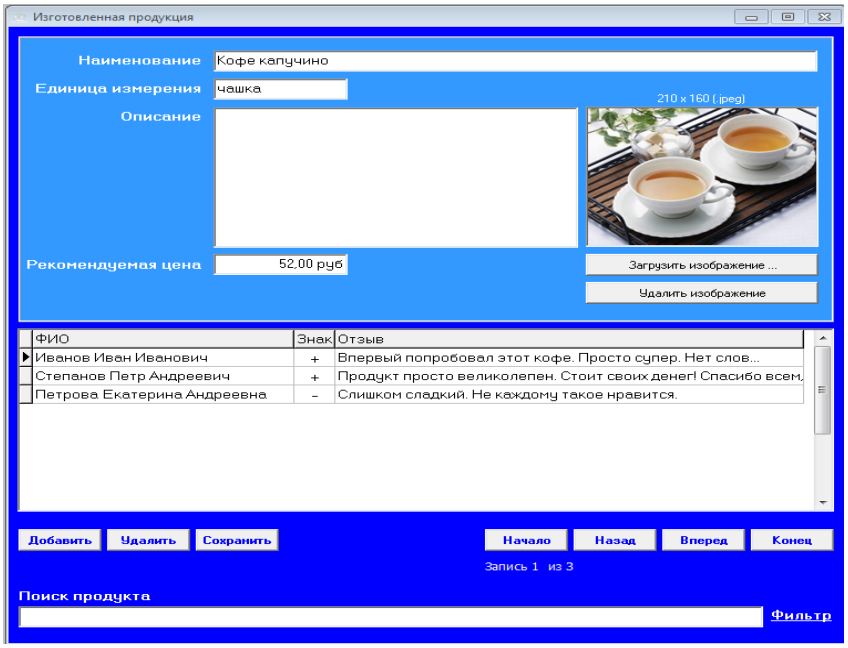


Рисунок 4.17 – Окно "Изготовленная продукция".

Если выбрать в главном меню из справочной информации пункт "Изготовленная продукция", то на экран выйдет соответствующее окно.

Название полей идентичны окно "Продукты полуфабрикаты". Новыми полями здесь являются "Рекомендуемая цена" и изображение товара.

Изображение товара можно загружать из файла типа *.*jpg*, *.*jpeg*. Размер изображения должен быть 210 х 160 пикселей.

Первоначально продукт следует сфотографировать и подготовить фотографию нужного размера в графическом редакторе. Затем в форме нажать кнопку "Загрузить изображение ..." и выбрать из стандартного окна Windows нужный файл картинки.

Первоначально с программой в каталоге "Ресурсы" поставляется три образца изображения для демонстрации работы.

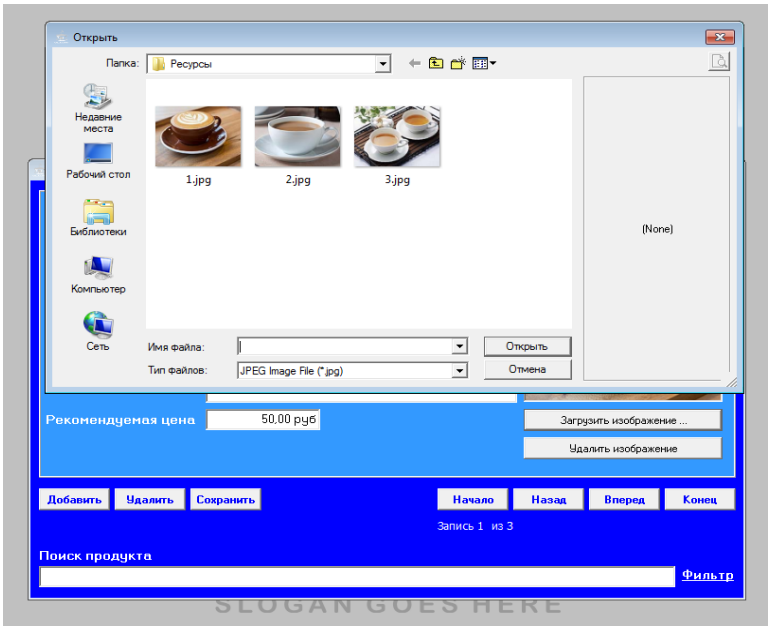


Рисунок 4.18 – Выбор изображения для загрузки в поле

Удаление изображения осуществляется посредством нажатия на кнопку "Удалить изображение". Поле является необязательным и изображение можно не загружать.

При выборе "Поступление продуктов" из главного меню на экран выходит одноименная форма.

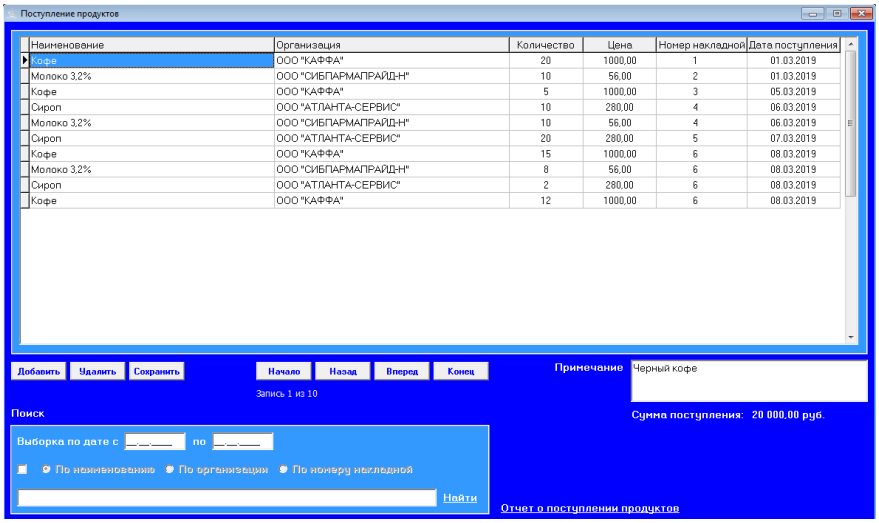


Рисунок 4.19 – Поступление продуктов

Добавлять новую запись можно тремя способами: нажатием кнопки "Добавить", нажатием клавиши "Стрелка вниз" на клавиатуре, когда курсор установлен на последнюю запись и нажатием клавиши "Insert" на клавиатуре.

Далее появляется пустая запись. Наименование и организацию следует выбирать только из раскрывающегося списка.

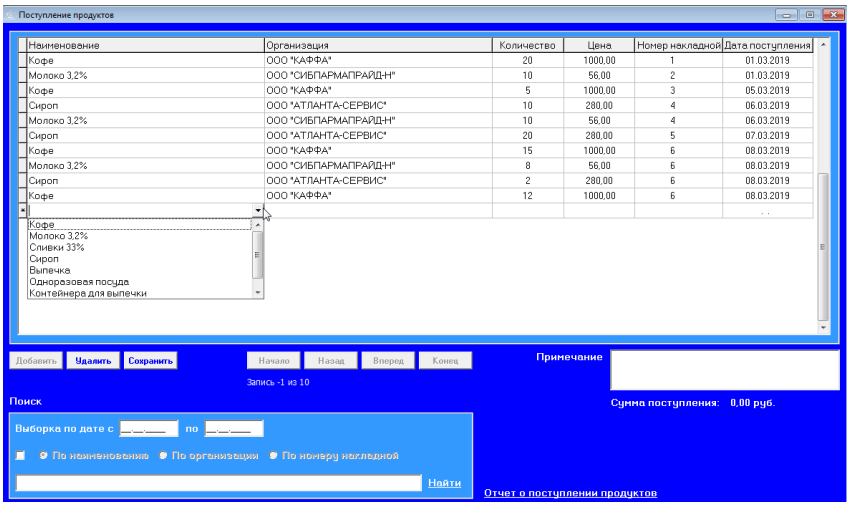


Рисунок 4.20 – Добавление товара из списка полуфабрикатов

После ввода всех строк, запись следует сохранить. Строка "Сумма поступления" отражает сумму по текущей строке.

Внизу окна представлена развернутая система поиска. Поиск можно осуществлять по диапазону дат. Для этого следует ввести начальную и

конечную даты. Если даты совпадают, то будут найдены записи именно с этой датой.

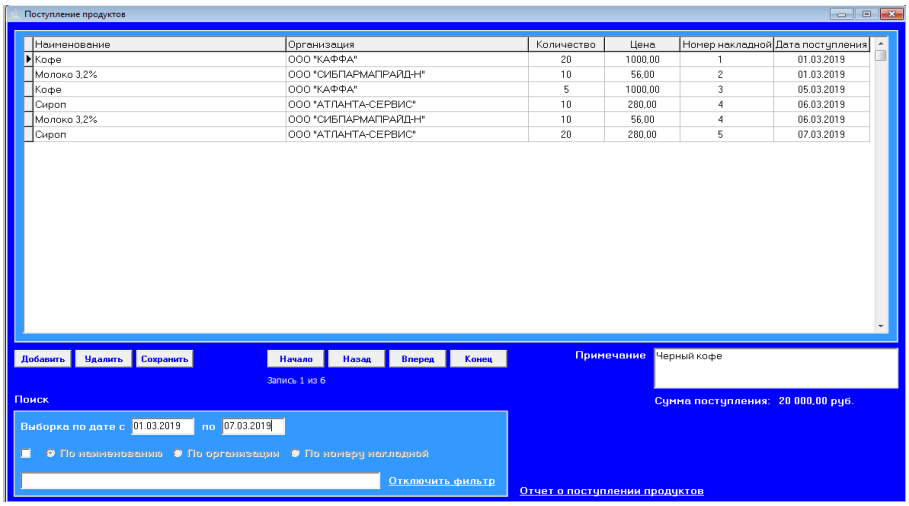


Рисунок 4.21 – Найденные записи по диапазону дат

Критерий фильтра можно уточнить. Например, из данного диапазона дат необходимо выбрать только определенный продукт. Для этого нужно поставить флажок, чтобы активировать дополнительный поиск. Далее выбрать "По наименованию", ввести наименование, либо его часть и нажать "Найти".

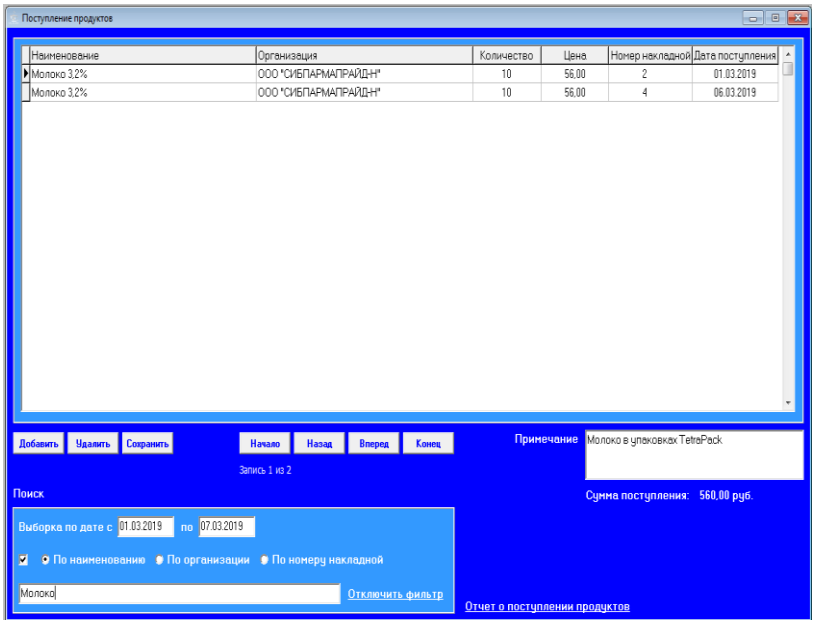


Рисунок 4.22 – Критерий уточнен до наименования "Молоко"

Точно так же можно осуществить поиск и по другим параметрам, например, по номеру накладной.

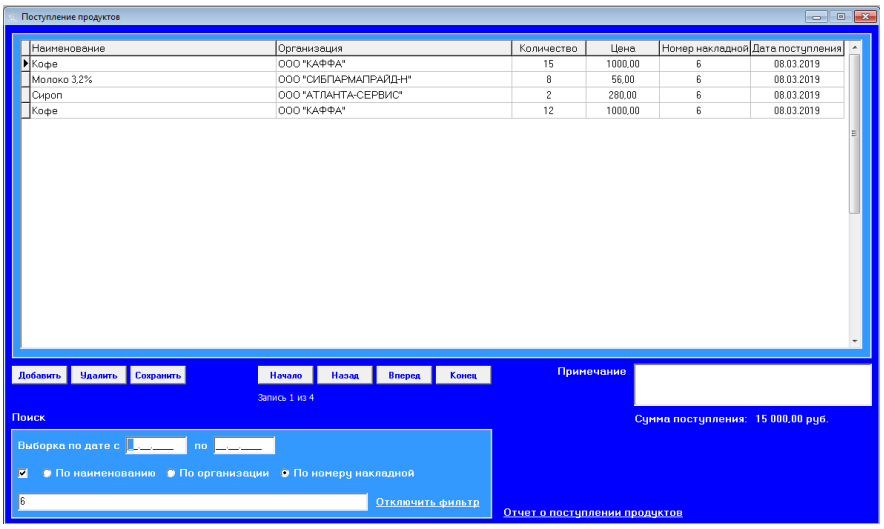


Рисунок 4.23 – Поиск по номеру накладной.

Следует заметить, что дополнительные критерии можно использовать и без выбора диапазона дат. Таким образом, представленный поиск является достаточно гибким.

Так же в нижней части экрана присутствует строка-кнопка "Отчет о поступлении продуктов" Она уникальна тем, что в отчет попадают только найденные записи. Если фильтр отключен, то в отчет выведутся все записи из базы данных.

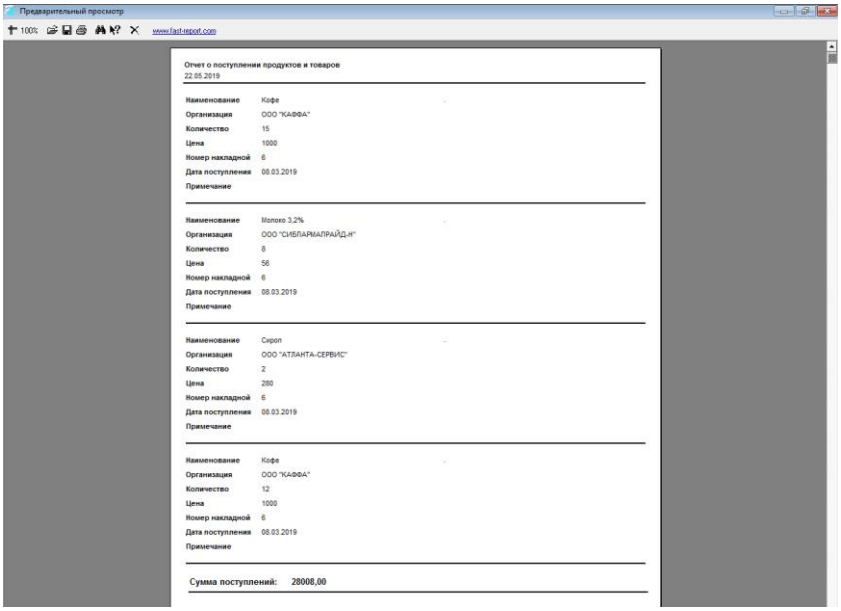


Рисунок 4.24 – Вывод на печать отчета о поступлении продуктов.

При выборе пункта меню "Расход продуктов в производство" открывается одноимённая форма.

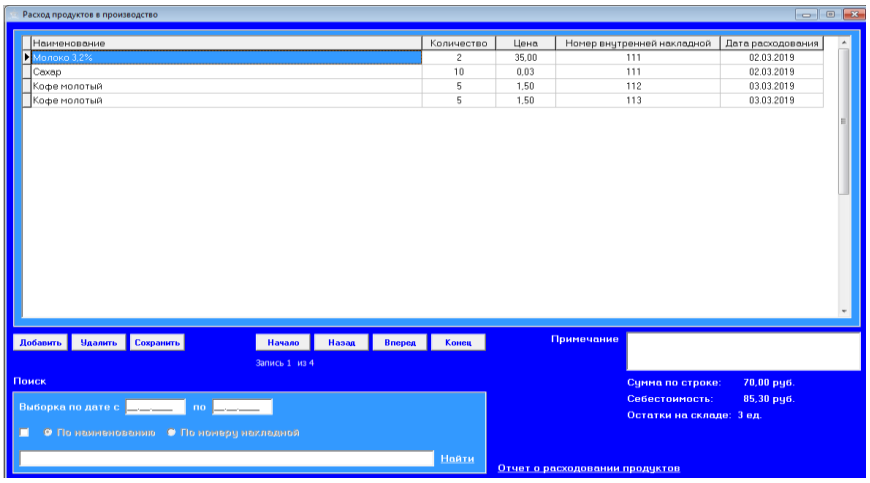


Рисунок 4.25 – Расход продуктов

Окно "Расход продуктов" предназначено для ввода информации о расходовании продуктов-полуфабрикатов. Окно по своей структуре аналогично поступлению. Основное отличие состоит в том, что в нижней части окна расходования вычисляется показатели себестоимости и остатков продуктов на складе кофейного заведения.

Пункт меню "Учет продаж кофе" открывает форму по учету продаж. Это основное рабочее окно. Окно делится на четыре части.

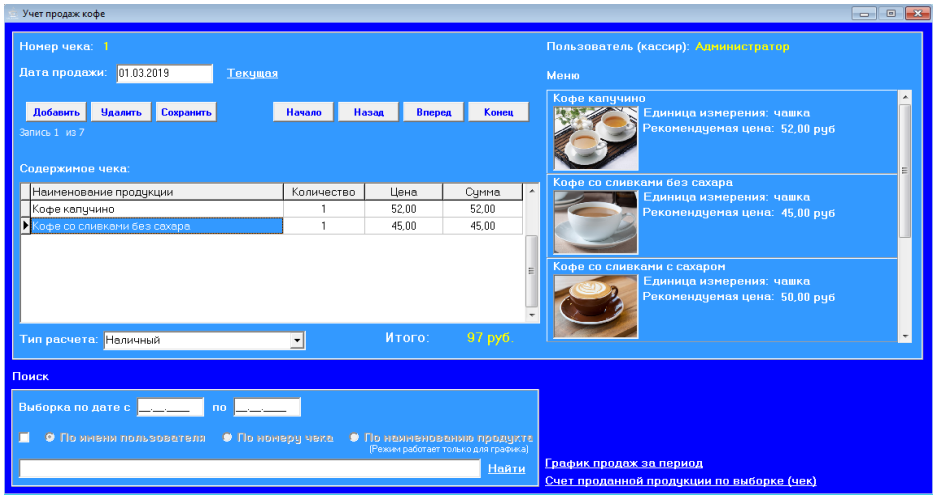


Рисунок 4.26 – Окно "Учет продаж кофе"

Для того, чтобы осуществить продажу товара клиенту, необходимо нажать на кнопку "Добавить". Будет сформирована пустая запись. В поле "Дата продажи" можно выставить любую дату. Кнопка "Текущая" добавляет текущую дату. Далее следует сохранить запись. После этого автоматически присвоится номер чека. Так же в верхней части окна присутствует пользователь, который добавлял записи до этого. Не следует путать с текущим пользователем, под которым осуществлен вход. Имя текущего пользователя (бариста), под которым в данный момент создается чек отображается в главном окне программы после ее названия.

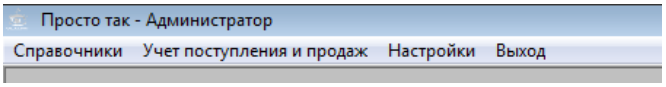


Рисунок 4.27 – Имя текущего пользователя

После того, как сформирован чек, следует начинать добавлять в него продукты, которые заказывает клиент. Продукты можно добавлять двумя способами. Первый все так же из раскрывающегося списка. Перед этим необходимо так же нажать кнопку "Стрелка вниз" на клавиатуре, стоя на последней записи в таблице или клавишу "Insert".

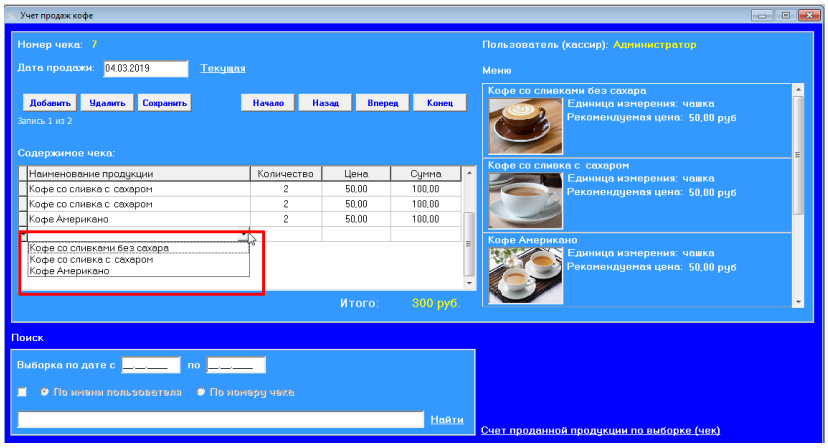


Рисунок 4.28 – Добавление продукта

Более удобный способ добавления товара является его выбор из меню справа. Вся информация загружается из справочника "Изготовленные продукты".

Одинарный щелчок мышью на необходимом изображении добавит запись. Пустым останется только количество и сумма. Количество следует ввести пользователю, а сумма будет посчитана после нажатия на клавишу "Enter". После добавления товара в данный список можно добавлять новый товар.

Графа "Сумма" отражает суму по текущей строке. Внизу таблицы присутствует итоговая сумма по всему чеку. Система поиска работает точно так же, как и в случае с поступлением продуктов.

Так же в самом низу находится пункт "Счет по проданной продукции по выборке (чек)". Это кнопка, которая выводит на экран кассовый чек для клиента и взаимодействует с оператором фискальных данных.

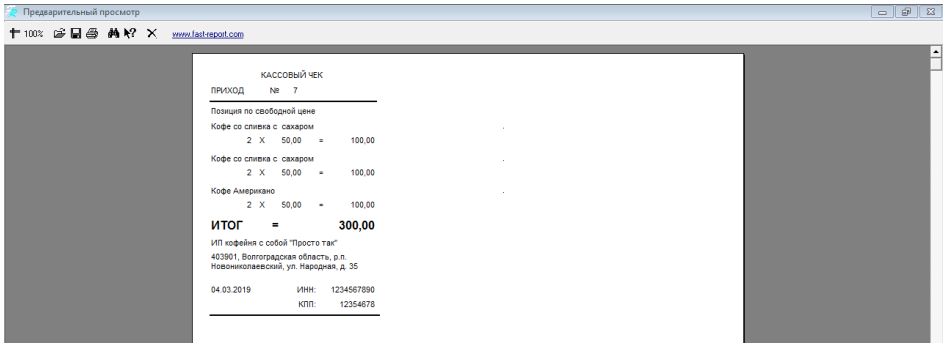


Рисунок 4.29 – Кассовый чек при покупке

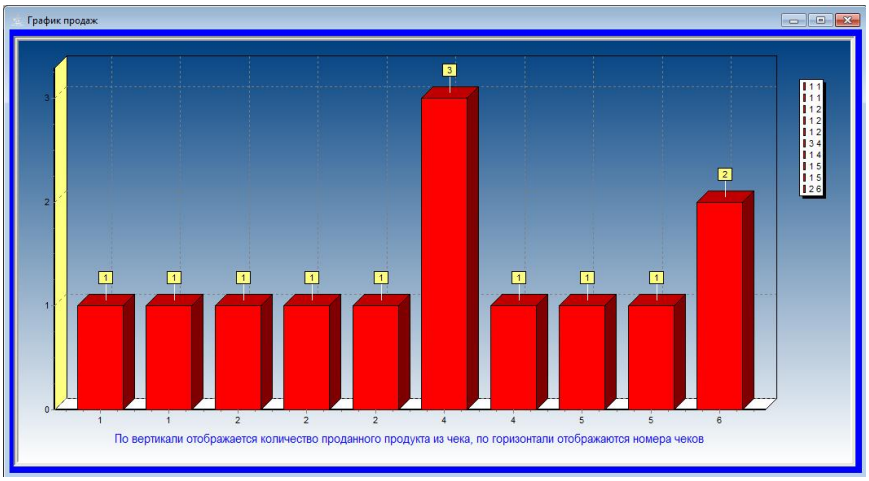


Рисунок 4.30 – Графический анализ продаж

Графический анализ работает только при условии ввода критерия поиска.

Следующим пунктом основного меню является пункт "Настройки". Если раскрыть этот пункт, то появится подменю, состоящее из трех пунктов:

– Резервное копирование БД;

– Редактирование пользователей;

– Сведения об организации.

Если был выбран пункт меню "Редактирование пользователей", то на экране появится одноименное окно.

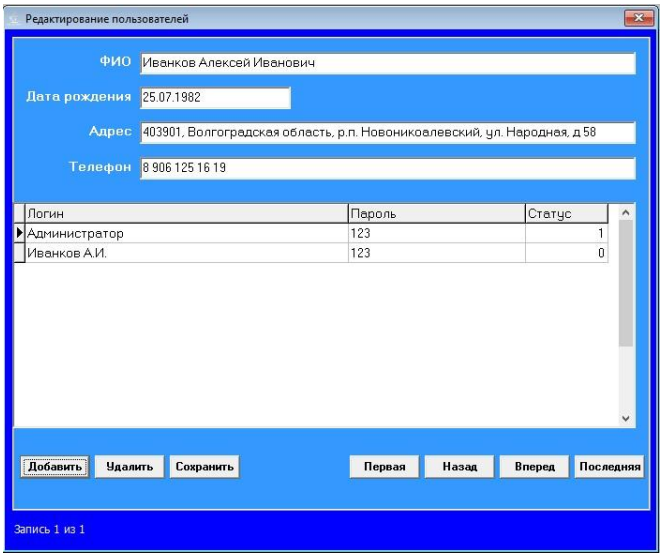


Рисунок 4.31 – Окно редактирования пользователей системы

В этом окне все пользователи, которые отображаются при запуске программы. Список пользователей можно редактировать путем добавления, удаления или изменения уже введенных пользователей. Администратор определяется значением, введенным в соответствующее поле. Значение "1" соответствует администратору. Любое другое значение соответствует учетной записи кассира-бариста.

Если выбран пункт "Сведения об организации", то появляется окно ввода постоянных настроек.

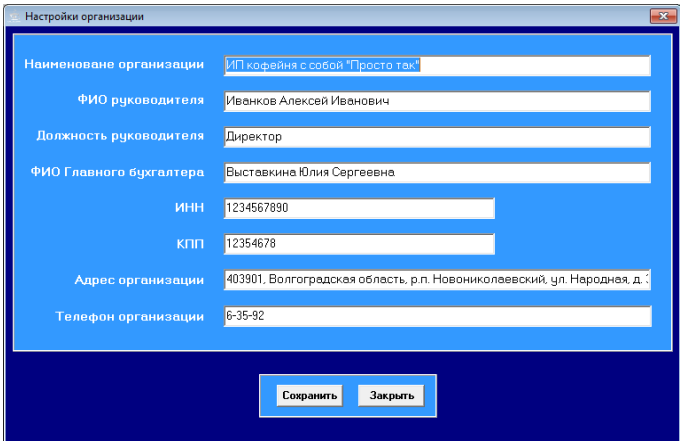


Рисунок 4.32 – Настройки параметров организации

Здесь, можно ввести наименование организации, сведения об основных материально-ответственных лицах и другие сведения.

Таким образом, интерфейс программы является интуитивно понятным, единообразным для более легкого и комфортного восприятия. Исследованное приложение может работать только на стационарном ПК, при наличии установленной виртуальной машины Java.

4.4 Протокол тестирования

По результатам тестирования составлен сводный протокол, представленный в таблице 4.1. В нее включено описание проводимого теста, результат проведения теста, а также при наличии – рекомендации по

модификации ПО. Условием допуска программы к эксплуатации является только полное прохождение тестов без критических замечаний.

Таблица 4.1 – сводный протокол тестирования ПО

№	Описание	Результат	Рекомендации
1	проверка ПО на соответствие утвержденному проекту разработки и внедрения системы	соответствует	
2	автоматическая проверка работоспособности ПО и выявление возможных ошибок	соответствует	
3	ручная проверка работоспособности ПО и выявление возможных ошибок	соответствует	
4	проверка качества интерфейса пользователя ПО	соответствует	
5	проверка качества информационного обмена между отдельными модулями ПО	соответствует	
6	проверка информационной безопасности	Существенных рисков не обнаружено	1. Использовать хеширование паролей (md5) в базе 2. Использовать зашифрованное соединение (или VPN) для соединения с БД по открытым каналам

Заключение по результатам тестирования. В результате проектирования было разработано приложение, деятельность которого в значительной степени облегчает работу бариста, а так же минимизирует ошибки, возникающие при работе. Внедрение в сеть кофеен распределенной информационной системы привело к уменьшению затрат и издержек, вызванных необходимостью ручного переноса данных из одной информационной системы в другую.

Создание распределенной базы данных позволило выполнять агрегированные запросы сразу для всей сети, получать статистику в режиме реального времени.

Переход на одну программу исключил необходимость выполнения операций импорта-экспорта в ручном режиме, из таких систем как интернет-банк, системы бухгалтерской и налоговой отчетности, базы данных и т.п., что сократило количество ошибок и расходы на ручной труд.

Выводы по 4 главе

При тестировании программного продукта подтверждается работа бизнес-правил, установленных при проектировании приложения и базы данных, а именно, осуществляется каскадное обновление и удаление записей в таблицах базы данных, попытка добавления некорректных значений, реакция на ошибки и т.п.

Для проведения тестирования распределенной информационной системы сети общественного питания разработана программа и методика, представленные выше.

Объектом тестирования является программное обеспечение (ПО) распределенной информационной системы сети общественного питания, а именно клиентский модуль, выполняемый на базе виртуальной машины Java и базы данных Mysql. Предъявляемое для тестирования ПО должно быть представлено в составе, достаточном для проведения полнофункционального

тестирования в соответствии с предусмотренными программой и методикой тестирования.

Разработаны программы тестов, проведено ручное и автоматическое тестирование. По результатам тестирования составлен сводный протокол.

ЗАКЛЮЧЕНИЕ

В результате исследования была проанализирована история и назначение распределенных ИС. В ходе выполнения технологической части разработана распределенная информационная система по учету продаж в сети кофейных заведений, а так же база данных и клиентский и серверный модули ИС.

Проектная часть включила стадию создания архитектуры, разработки базы данных, стадию проектирования приложения для работы с созданной базой данных и программного кода. Для создания приложения была выбрана СУБД MySQL, язык программирования JAVA 12 и набор готовых библиотек (таких как JBDC для подключения к БД, SWING для создания пользовательского интерфейса). Разработанная система имеет архитектуру клиент-сервер и направлена на организацию учета в территориально-распределенных объектах, список бизнес-процессов которого ограничен и не имеет ожидаемых изменений в будущем. В данном случае экономически целесообразным является разработка ПО под нужды конкретного предприятия вместо интеграции определенного решения. Применение удаленной базы данной в рассматриваемой системе необходимо для упрощения выполнения агрегированных операций, таких как ведение статистики, учет закупок и расхода сырья, когда данные передаются в центральную БД по окончанию смены для генерации статистики и отчетов по продажам.

Перед разработкой было проведено исследование предметной области, выполнены электронные чертежи бизнес-процессов, а так же были выполнены модели информационно-логической модели базы данных.

Непосредственно, перед разработкой программы была выполнена блок-схема ее работы.

Следующим этапом разработки стало кодирование программной части.

Итоговым этапом разработки было проведение автоматического и ручного теста работы приложения на предмет некорректного ввода данных пользователем в приложение и корректную работу бизнес-правил по каскадному удалению и каскадному обновлению данных в таблицах базы данных.

Можно отметить, что программа получилась универсальной и может функционировать в любой подобной организации общественного питания с любым ассортиментом продукции.

Для разработки клиентского модуля был выбран объектно-ориентированный визуально-программируемый язык высокого уровня JAVA 12, содержащий в себе возможности по обработке базы данных, сопоставимые с возможностями СУБД. А широкий ассортимент методов и моделей доступа к данным позволил спроектировать приложения максимально комфортным способом.

Приложение использует концепцию JDBC для доступа к данным. Это абстрактная модель для доступа к локальным и удаленным данным. Ее применение дает возможность с относительной легкостью распространять созданный программный продукт на другие персональные компьютеры, поскольку нет необходимости задумываться о том, что бы на компьютере пользователя был установлен драйвер доступа к данным.

В качестве базы данных была выбрана СУБД MySQL. Этот формат является достаточно надежным и удобным. Используя связку JAVA + MySQL была построена двухуровневая клиент-серверная система.

Рамки данного проекта и ограниченное время не позволили выполнить интеграцию созданного приложения с другими информационными системами, но такие возможности вполне можно реализовать посредством API.

Интерфейс программной оболочки представляет из себя стандартный GUI-интерфейс в стиле многодокументного приложения, когда в качестве

главной формы выступает форма запуска, а дочерние открываются в ее рамках.

Созданная в ходе работы информационная система предназначена для установки на настольные ПК под управлением ОС Windows 10. В силу кроссплатформенности платформы Java, возможна эксплуатация указанной ИС и на других операционных системах в режиме ограниченной функциональности.

В системе можно достаточно легко регистрировать новые продажи, а так же проводить редактирование введенной информации в базу данных.

Программа обладает удобными инструментами по фильтрации данных. Данные можно фильтровать по частичному совпадению, что позволяет пользователю, не зная, например, точного наименования, ввести часть названия и система выдаст ему результат поиска. Отобранные данные можно, в случае необходимости отредактировать.

В разработанной системе спроектированы отчеты о поступлении, а также квитанции (чеки) с возможность взаимодействия с операторами фискальных данных через API. Благодаря системе поиска и фильтрации информации из данных отчетов можно получить различные комбинации данных.

В качестве перспективы развития можно назвать создание клиентского мобильного модуля программы, работающего на платформе Android. Подобный модуль позволит в полной мере использовать возможности РИС, например, с помощью него можно организовать собственную службу доставки сети общественного питания, создав мобильное приложение для курьеров. Также перспективной альтернативой является интеграции данной РИС с сервисами по доставке еды на дом.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

Учебники и учебные пособия

1. Абрамов, С.А. Математические построения и программирование / С.А. Абрамов. - М.: Наука, 2016. - 192 с.

2. Алиев, В.С. Практикум по бизнес-планированию с использованием программы Project Expert / В.С. Алиев. - М.: Инфра-М, Форум, 2017. - 593 с.

3. Аллен, Г. Тейлор SQL для чайников / Аллен Г. Тейлор. - М.: Диалектика, Вильямс, 2015. - 416 с.

4. Балдин, К. В. Информационные системы в экономике / К.В. Балдин. - М.: ИНФРА-М, 2018. - 224 с.

5. Бен, Форта SQL за 10 минут / Форта Бен. - М.: Диалектика / Вильямс, 2015.

6. Бьюли, Алан Изучаем SQL / Алан Бьюли. - М.: Символ-плюс, 2014.

7. Ван, Тассел Д. Стиль, разработка, эффективность, отладка и испытания программ / Ван Тассел Д.. - М.: Мир, 2017. - 332 с.

8. Вдовенко, Л. А. Информационная система предприятия: Уч. пос./Л.А.Вдовенко-2-е изд., пераб. и доп.-М.:Вузовский уч. / Л.А. Вдовенко. - Москва: Машиностроение, 2016. - 143 с.

9. Вендров, А. М. Практикум по проектированию программного обеспечения экономических информационных систем / А.М. Вендров. - М.: Финансы и статистика, 2017. - 192 с.

10. Вирт, Н. Алгоритмы и структуры данных / Н. Вирт. - М.: Мир, 2016. - 360 с.

11. Голицына, О.Л. Основы алгоритмизации и программирования: Учебное пособие / О.Л. Голицына, И.И. Попов. - М.: Форум; Издание 2-е, 2015. - 432 с.

12. Грабер, Мартин SQL для простых смертных / Мартин Грабер. - М.: ЛОРИ, 2014. - 378 с.

13. Граничин О. Н., Кияев В. И. Информационные технологии и системы в современном менеджменте. — СПб: Изд-во BBM, 2014. — 897 с.

14. Гудсон, Джон Практическое руководство по доступу к данным (+ DVD-ROM) / Джон Гудсон , Роб Стюард. - М.: БХВ-Петербург, 2013. - 304 с.

15. Дейт, К. Дж. SQL и реляционная теория. Как грамотно писать код на SQL / К.Дж. Дейт. - М.: Символ-плюс, 2016.

16. Джеймс, Р. Грофф SQL. Полное руководство / Джеймс Р. Грофф, Пол Н. Вайнберг, Эндрю Дж. Оппель. - М.: Вильямс, 2014. - 960 с.

17. Дунаев, В. В. Базы данных. Язык SQL для студента / В.В. Дунаев. - М.: БХВ-Петербург, 2017. - 288 с.

19. Ивасенко, А.Г. Информационные технологии в экономике и управлении. Учебное пособие / А.Г. Ивасенко. - М.: КноРус, 2017. - 354 с.

20. Информационные системы в экономике. Практикум. - М.: КноРус, 2017. - 254 с.

21. Информационные технологии в маркетинге. Учебник и практикум. - Москва: Мир, 2016. - 368 с.

22. Информационные технологии в менеджменте. Учебное пособие / В.И. Карпузова и др. - М.: Вузовский учебник, Инфра-М, 2017. - 304 с.

23. Информационные технологии в управлении персоналом. Учебник и практикум / Ю.Д. Романова и др. - М.: Юрайт, 2015. - 291 с.

24. Информационные технологии. Учебник. В 2 томах. Том 1-2 (комплект из 2 книг) / В.В. Трофимов и др. - М.: Юрайт, 2016. - 632 с.

25. Исаев, Г. Н. Информационные системы в экономике / Г.Н. Исаев. - М.: Омега-Л, 2018. - 464 с.

26. Карвин, Билл Программирование баз данных SQL. Типичные ошибки и их устранение / Билл Карвин. - М.: Рид Групп, 2018. - 336 с.

27. Кириллов А. Г. Условия эффективного применения информационных технологий в управлении / А. Г. Кириллов // Преподаватель XXI век. – 2013. – Т.1, № 2. – С. 12-15.

28. Коробов, Н. А. Информационные технологии в торговле / Н.А. Коробов, А.Ю. Комлев. - М.: Академия, 2016. - 176 с.

29. Кригель, А. SQL. Библия пользователя / А. Кригель. - М.: Диалектика / Вильямс, 2016. -

30. Кудрявцев, Е.М. Методы решения организационных задач. Учебник / Е.М. Кудрявцев. - М.: Ассоциация строительных вузов (АСВ), 2015. - 150 с.

31. Кулемина, Ю. В. Информационные системы в экономике. Краткий курс / Ю.В. Кулемина. - М.: Окей-книга, 2015. - 112 с.

32. Лихтенштейн, В. Е. Информационные технологии в бизнесе. Практикум / В.Е. Лихтенштейн, Г.В. Росс. - М.: Финансы и статистика, 2017. - 512 с.

33. Майкл, Дж. Хернандес SQL-запросы для простых смертных. Практическое руководство по манипулированию данными в SQL / Майкл Дж. Хернандес, Джон Л. Вьескас. - М.: ЛОРИ, 2014. - 480 с.

34. Маркин, А. В. Построение запросов и программирование на SQL. Учебное пособие / А.В. Маркин. - М.: Диалог-Мифи, 2014. - 384 с.

35. Мартишин, С. А. Проектирование и реализация баз данных в СУБД MySQL с использованием MySQL Workbench. Учебное пособие / С.А. Мартишин, В.Л. Симонов, М.В. Храпченко. - М.: Форум, Инфра-М, 2015. - 160 с.

36. Молинаро, Энтони SQL. Сборник рецептов / Энтони Молинаро. - М.: Символ-плюс, 2016.

37. Оппель, Эндрю Дж. SQL. Полное руководство / Оппель Эндрю Дж.. - М.: Диалектика / Вильямс, 2016. -

38. Пржиялковский, В. В. Введение в Oracle SQL / В.В. Пржиялковский. - М.: Бином. Лаборатория знаний, Интернет-университет информационных технологий, 2016. - 320 с.

39. Сапков, В. В. Информационные технологии и компьютеризация делопроизводства / В.В. Сапков. - М.: Академия, 2017. - 288 с.

40. Селко, Джо SQL для профессионалов. Программирование / Джо Селко. - М.: ЛОРИ, 2015. - 464 с.

41. Скотт, Т. Основы программирования. Курс программированного обучения / Т. Скотт. - М.: Советское радио, 2016. - 490 с.

42. Тейлор, Аллен SQL для чайников / Аллен Тейлор. - М.: Вильямс, 2014. - 416 с.

43. Туманов, В. Е. Проектирование хранилищ данных для систем бизнес-аналитики / В.Е. Туманов. - М.: Интернет-университет информационных технологий, Бином. Лаборатория знаний, 2016. - 616 с.

44. Уткин, В. Б. Информационные системы в экономике / В.Б. Уткин, К.В. Балдин. - М.: Academia, 2018. - 288 с.

45. Форта, Бен Освой самостоятельно SQL за 10 минут / Бен Форта. - М.: Вильямс, 2015. - 288 с.

46. Хабибуллин И. Ш. Разработка Web-служб средствами Java / И. Ш. Хабибуллин. – СПб.: БХВ-Петербург, 2003. – 400 с

47. Черников, Б. В. Информационные технологии управления / Б.В. Черников. - М.: Инфра-М, Форум, 2017. - 368 с.

48. Чудновский, А. Д. Информационные технологии управления в туризме / А.Д. Чудновский. - М.: КноРус, 2017. - 104 с.

49. Шаран, Кишори Java 9. Полный обзор нововведений. Для быстрого ознакомления и миграции / Кишори Шаран. - М.: ДМК Пресс, 2017. - 690 с.

50. Шлыкова, О. В. Компьютерная Анимация: Учебная Программа Курса / О.В. Шлыкова. - Москва: СПб. [и др.] : Питер, 2016. - 701 с.

51. Шпаргалка по информационным системам в экономике. - М.: Окей Книга, 2018. - 640 с.

52. Эйри, Джоунс Функции SQL. Справочник программиста / Джоунс Эйри. - М.: Диалектика / Вильямс, 2013.

Электронные ресурсы

53. Василенко Н. В., Макаров В. А. Оценка надежности программного обеспечения // Вестник НовГУ. 2005. №30. [Электронный ресурс]. – Режим доступа: https://cyberleninka.ru/article/n/otsenka-nadezhnosti-programmnogoobespecheniya (дата обращения: 10.12.2019).

54. Автоматизация тестирования Java EE веб-сервисов с помощью SoapUI и Arquillian // Хабр [Электронный ресурс]. - Режим доступа: https://habr.com/ru/post/267301/ (дата обращения 10.12.2019).

55. Оценка надежности программного обеспечения // Dev Harmony [Электронный ресурс]. - Режим доступа: http://kirnosenko.com/2011/03/07/software-reliability-estimation/ (дата обращения 17.12.2019).

Литература на иностранном языке

56. AntiPatterns // W. Brown, R. Malveau, H. I. McCormick, T. Mowbray. – N.Y.: John Wiley, 1998. - 156 pp.

57. Chen P. P., The Entity-Relationship Model: Toward a Unified View of Data. ACM Trans. on Database Systems, 1976, Vol. 1, – P 9-36.

58. Cox B. J. Object oriented programming: An Evolutionary Approach / B. J. Cox // ill Reading, Mass.: Addison-Wesley Pub. Co., 1. – 1986 – 274 pp.

59. Date C. J. Database Design and Relational Theory: Normal Forms and All That Jazz (Theory in Practice) / C. J. Date // O'Reilly Media, 2012. – 276 pp

60. Fisher A. S. CASE: Using Software Development Tools / A. S. Fisher. – N.Y.: J. Willey and Sons Inc., 1988. – 287 pp.

61. Fowler, M. Patterns of Enterprise Application Architecture / M. Fowler. - Wiley Publishing, Inc., 2012.

Приложение А

Фрагмент скрипта импорта структуры БД

```
USE `database` ;
-- -----------------------------------------------------
-- Table `database`.`logpass`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `database`.`logpass` (
  `Login` VARCHAR(50) NOT NULL ,
  `Password` VARCHAR(50) NULL DEFAULT NULL ,
  `Status` INT(11) NOT NULL DEFAULT '0' ,
  PRIMARY KEY (`Login`) ,
  UNIQUE INDEX `Login_UNIQUE` (`Login` ASC) )
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;
-- -----------------------------------------------------
-- Table `database`.`prodgot`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `database`.`prodgot` (
  `Name` VARCHAR(150) NOT NULL ,
  `EdIzm` VARCHAR(15) NOT NULL ,
  `Opis` MEDIUMTEXT NULL DEFAULT NULL ,
  `Image` MEDIUMBLOB NULL DEFAULT NULL ,
  `Cena` DECIMAL(10,2) NOT NULL ,
  PRIMARY KEY (`Name`) ,
  UNIQUE INDEX `Name_UNIQUE` (`Name` ASC) )
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;
-- -----------------------------------------------------
-- Table `database`.`prod`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `database`.`prod` (
  `NomChecka` INT NOT NULL AUTO_INCREMENT ,
  `DateProd` VARCHAR(10) NOT NULL ,
  `Login` VARCHAR(50) NOT NULL ,
  PRIMARY KEY (`NomChecka`) )
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `database`.`perchecka`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `database`.`perchecka` (
  `NomChecka` INT NOT NULL ,
  `Name` VARCHAR(150) NOT NULL ,
  `Kolvo` INT NOT NULL ,
  `Cena` DECIMAL(10,2) NOT NULL ,
  `ID` INT NOT NULL AUTO_INCREMENT ,
```

```
  PRIMARY KEY (`ID`) ,
  INDEX `ProdGotName_idx` (`Name` ASC) ,
  INDEX `ProdNomCheka_idx` (`NomChecka` ASC) ,
  CONSTRAINT `ProdGotName`
    FOREIGN KEY (`Name` )
    REFERENCES `database`.`prodgot` (`Name` )
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `ProdNomChecka`
    FOREIGN KEY (`NomChecka` )
    REFERENCES `database`.`prod` (`NomChecka` )
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;

CREATE  TABLE IF NOT EXISTS `database`.`post` (
  `Org` VARCHAR(150) NOT NULL ,
  `Name` VARCHAR(200) NOT NULL ,
  `Tel1` VARCHAR(20) NULL DEFAULT NULL ,
  `Tel2` VARCHAR(20) NULL DEFAULT NULL ,
  `Adres` VARCHAR(255) NULL DEFAULT NULL ,
  `Email` VARCHAR(50) NULL DEFAULT NULL ,
  `INN` VARCHAR(12) NULL DEFAULT NULL ,
  `KPP` VARCHAR(10) NULL DEFAULT NULL ,
  `OGRN` VARCHAR(15) NULL DEFAULT NULL ,
  `Prim` MEDIUMTEXT NULL DEFAULT NULL ,
  PRIMARY KEY (`Org`) )
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;

CREATE  TABLE IF NOT EXISTS `database`.`prodpol` (
  `Name` VARCHAR(150) NOT NULL ,
  `EdIzm` VARCHAR(15) NOT NULL ,
  `Opis` MEDIUMTEXT NULL DEFAULT NULL ,
  PRIMARY KEY (`Name`) )
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;
-- -----------------------------------------------------
-- Table `database`.`postup`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `database`.`postup` (
  `Name` VARCHAR(150) NOT NULL ,
  `Org` VARCHAR(150) NOT NULL ,
  `Kolvo` INT NOT NULL ,
  `Cena` DECIMAL(10,2) NOT NULL ,
  `NomNakl` VARCHAR(50) NULL DEFAULT NULL ,
  `DatePost` VARCHAR(10) NOT NULL ,
  `Prim` VARCHAR(255) NULL DEFAULT NULL ,
  `ID` INT NOT NULL AUTO_INCREMENT ,
  PRIMARY KEY (`ID`) ,
  INDEX `ProdPolName_idx` (`Name` ASC) ,
```

```
  INDEX `PostOrg_idx` (`Org` ASC) ,
  CONSTRAINT `ProdPolName`
    FOREIGN KEY (`Name` )
    REFERENCES `database`.`prodpol` (`Name` )
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `PostOrg`
    FOREIGN KEY (`Org` )
    REFERENCES `database`.`post` (`Org` )
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;

USE `database` ;

SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

Приложение Б

Фрагмент программного кода клиентского модуля

```
package tgudisser;

//Подключение к базе данных на внешнем сервере
public class ConnectToDB {

    public static final String DB_URL = "jdbc:ip:3336";
    private static final String user = "";
    private static final String password = "";

    public static void main(String[] args) {
        try {
            Class.forName(DB_Driver);
            Connection                connection                =
DriverManager.getConnection(DB_URL);
            //System.out.println("Соединение        с        СУБД
выполнено.");
               boolean is_ok = true;

        } catch (ClassNotFoundException e) {
            e.printStackTrace();     //     обработка     ошибки
Class.forName
            System.out.println("JDBC   драйвер   для   СУБД   не
найден!");
        } catch (SQLException e) {
            e.printStackTrace();     //     обработка     ошибок
DriverManager.getConnection
            System.out.println("Ошибка SQL !");
        }
    }
}

// Класс инициализации пользовательского интерфейса
public class MainFrame extends JFrame {

    public MainFrame() {
        setBounds(800, 600, 800, 600); // Размеры окна
    }

    public static void main(String[] args) {

        SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                MainFrame mf = new MainFrame();
                mf.setDefaultCloseOperation(EXIT_ON_CLOSE);
                mf.setVisible(true);
            }
        });
```

```
    }
}
// Класс приема заказа и помещения ингредиентов в массив
public class MainOrder {
    public static void main(String[] args) {

        Map<String, Integer> sostavzakaza = new
LinkedHashMap<String, Integer>();
        getStorageData(sostavzakaza);
        Map<String, Integer> OrderMap = new
LinkedHashMap<String, Integer>();
        if (sostavzakaza.isEmpty()) {
            System.out.println(Texts.sostavpust);
            return;
        }

        printkomponents(Texts.AVAILABLE_komponentS,
sostavzakaza);
        showHowTo();
        Scanner sc = new Scanner(System.in);
        while (sc.hasNext()) {
            String value = sc.nextLine();
            if ("q".equalsIgnoreCase(value)) {
                System.exit(0);
            }
            if ("DONE".equalsIgnoreCase(value)) {
                break;
            }
            processInput(value, sostavzakaza, OrderMap);

            if (!OrderMap.isEmpty()) {
                System.out.println(Texts.REMOVE);
            }
        }
        printkomponents(Texts.YOUR_Order_END, OrderMap);
    }


    private static void showHowTo() {
        System.out.println(Texts.TEXT_CHOOSE);
        System.out.println(Texts.TEXT_EXIT);
        System.out.println(Texts.TEXT_FINISH);
    }


    private static void processInput(String value,
            Map<String, Integer> sostavzakaza, Map<String,
Integer> OrderMap) {
        String[] splittedValue = value.split(" ");
        if (splittedValue.length == 2) {
            int NumIng = stringToInt(splittedValue[0]);
            int komponentCount = stringToInt(splittedValue[1]);
```

```
            if (NumIng <= 0 || NumIng > sostavzakaza.size()) {

System.out.println(Texts.komponent_NUMBER_LIMIT);
                showHowTo();
                return;
            }

            processOrderMap(NumIng,                komponentCount,
sostavzakaza, OrderMap);
        } else {
            System.out.println(Texts.SELECTION_ERROR);
            showHowTo();
        }
    }

    private   static   void   processOrderMap(int   NumIng,   int
komponentCount, Map<String, Integer> sostavzakaza, Map<String,
Integer> OrderMap) {
        StringBuilder key = new StringBuilder();
        if  (komponentCount  >  0  &&  iskomponentPresent(NumIng,
komponentCount, sostavzakaza, key)) {
            if (OrderMap.containsKey(key.toString())) {
                int amount = OrderMap.get(key.toString());
                amount += komponentCount;
                OrderMap.put(key.toString(), amount);
            } else {
                OrderMap.put(key.toString(), komponentCount);
            }

            if (sostavzakaza.containsKey(key.toString())) {
                int amount = sostavzakaza.get(key.toString());
                amount -= komponentCount;
                sostavzakaza.put(key.toString(), amount);
            }
            printkomponents(Texts.YOUR_Order, OrderMap);
            showHowTo();
            printkomponents(Texts.AVAILABLE_komponents,
sostavzakaza);
        }     else     if     (komponentCount     <     0     &&
iskomponentPresentInOrder(NumIng,    komponentCount,    OrderMap,
key)) {

            int value = OrderMap.get(key.toString());
            value += komponentCount;
            OrderMap.put(key.toString(), value);

            value = sostavzakaza.get(key.toString());
            value -= komponentCount;
            sostavzakaza.put(key.toString(), value);
            printkomponents(Texts.YOUR_Order, OrderMap);
```

```
            showHowTo();
            printkomponents(Texts.AVAILABLE_komponents,
sostavzakaza);
        } else {

System.out.println(Texts.komponent_UNAVAILABLE_AMOUNT);
            showHowTo();
        }
    }

    private static boolean iskomponentPresentInOrder(int NumIng,
int komponentCount, Map<String, Integer> OrderMap, StringBuilder
key) {
        int i = 1;
        for (Entry<String, Integer> entry : OrderMap.entrySet())
{
            if (i == NumIng) {
                komponentCount = -1 * komponentCount;
                if               (komponentCount               <=
OrderMap.get(entry.getKey())) {
                    key.append(entry.getKey());
                    return true;
                }
            }
            i++;
        }
        return false;
    }

    private static boolean iskomponentPresent(int NumIng, int
komponentCount, Map<String, Integer> sostavzakaza, StringBuilder
key) {
        int i = 1;
        for       (Entry<String,       Integer>       entry       :
sostavzakaza.entrySet()) {
            if (i == NumIng) {
                if  (komponentCount  >  0  &&  komponentCount  <=
entry.getValue()) {
                    key.append(entry.getKey());
                    return true;
                }
            }
            i++;
        }
        return false;
    }

    private static int stringToInt(String str) {
        try {
            return Integer.parseInt(str);
        } catch (Exception e) {
            return Texts.ERROR;
```

```
        }
    }

    private static void printkomponents(String prefix,
            Map<String, Integer> komponentMap) {
        System.out.println(prefix);
        int i = 1;
        for      (Entry<String,      Integer>      entry      :
komponentMap.entrySet()) {
            System.out.println(i + ". " + entry.getKey() + ": "
                    + entry.getValue());
            i++;
        }
    }


    private static class Texts {
        private static String sostavpust = "На локальном складе
нет достаточного количества";
        private static String AVAILABLE_komponentS = "Доступные
ингредиенты исходя из остатков сырья:";
        private static String TEXT_CHOOSE = "Выберите ингредиент
и его количество";
        private   static   String   TEXT_EXIT   =   "Подтвердите
завершения смены";
        private  static  String  TEXT_FINISH  =  "Завершите  выбор
компонентов";
        private   static   String   SELECTION_ERROR   =   "Ошибка
выполнения команды";
        private  static  String  komponent_NUMBER_LIMIT  =  "Вы
указали неправильный внутренний номер продукта";
        private static final String komponent_UNAVAILABLE_AMOUNT
= "Выбраное количество сырья не доступно.";
        private   static   String   YOUR_Order   =   "\n\n\nСостав
заказа:";
        private  static  String  YOUR_Order_END  =  "\n\n\Заказ
состоит из:";
        private  static  final  String  REMOVE  =  "У  вас  есть
возможность убрать добавленные ингредиенты.";
        private static final int ERROR = -1;
    }
}
```