Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет   
информатики и радиоэлектроники»

Факультет иженерно-экономический

Кафедра экономической информатики

Дисциплина «Программирование сетевых приложений»

|  |  |
| --- | --- |
|  | «К ЗАЩИТЕ ДОПУСТИТЬ» |
|  | Руководитель курсового проекта  старший преподаватель  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Т.М. Унучек |
|  | \_\_\_.\_\_\_\_.2021 |

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовому проекту

на тему:

**«****АВТОМАТИЗИРОВАННАЯ СИСТЕМА ПОДДЕРЖКИ УПРАВЛЕНИЯ ДИСКОНТНОЙ ПРОГРАММНОЙ СЕТИ МАГАЗИНОВ»**

БГУИР КП 1-40 05 01-10 024 ПЗ

|  |  |
| --- | --- |
|  | Выполнила студентка группы 914301  СТАНКУС Александра Леонидовна  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись студентки) |
|  | Курсовой проект представлен на проверку 15.12.2021  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись студентки) |

Минск 2021

**СОДЕРЖАНИЕ**

Введение5

1 Описание предметной области8

2 Постановка задачи и обзор методов ее решения13

2.1 Детализация задач в области разработки системы13

2.2 Обзор методов решения13

3 Функциональное моделирование на основе стандарта IDEF0…………18

4 Информационная модель системы и ее описание25

5 Описание алгоритмов, реализующих бизнес-логику серверной части проектируемой системы29

6 Модели представления системы и их описание………………………..31

6.1 Диаграмма классов…....………………………………………………….31

6.2 Диаграмма компонентов…………………………………………………33

6.3 Диаграмма развертывания……………………………………………….34

6.4 Диаграмма состояний…………………………………………………….35

6.5 Диаграмма последовательности…………………………………………36

6.6 Диаграмма вариантов использования……………………………………38

7 Руководство пользователя41

8 Результаты тестирования разработанной системы46

Заключение49

Список использованных источников50

Приложение А (обязательное) 51

Приложение Б (обязательное) 53

Приложение В (обязательное) 55

# **ВВЕДЕНИЕ**

Прямой выгодой для покупателя в торговле сегодня является скидка – дисконтная программа, повышающая привлекательность магазина, от которой владелец имеет гораздо лучшие результаты, чем его конкуренты. Автоматизация системы учета дисконтной программы с помощью современного компьютерного обеспечения позволяет собирать информацию, присвоенную карте, мгновенно. Техника не только считывает данные с магнитной, чиповой карты или штрих-кода, она определяет клиента в базе, позволяет создать историю покупок, автоматически рассчитывает скидку, а также показывает эффективность дисконтной программы при помощи аналитических расчетов. Однако самое важное, что нужно учитывать, — это цели вашего бизнеса, которых должна помогать достигнуть дисконтная программа [1].

Дисконтная программа — это разработанная система скидок для клиента от магазина, представителя сферы услуг или общепита. В первую очередь дисконтная программа скидок предназначена для повышения лояльности клиента за счет положительных эмоций от покупки, тем самым она напрямую влияет на повышение спроса.

В современных условиях возрастает роль дисконтных программ и скидок в борьбе за покупателя. Система дисконтных карт сегодня является одним наиболее распространенных и эффективных инструментов, применяемых для удержания покупателей и привлечения новых клиентов, которая при грамотной организации может существенно увеличить объем продаж и повысить имидж предприятия в глазах клиентов и конкурентов.

Дисконтные системы разделяются на простые и сложные. В простых программах скидка или иная льгота клиенту не зависит от его покупательской активности, но может зависеть от суммы его разовой покупки. Сложные дисконтные системы в свою очередь делятся на накопительные программы (или программы лояльности) и расчетные (или торговые) дисконтные программы. В программах лояльности скидка зависит от объема покупок потребителя в данной организации торговли или сферы услуг. Учет покупок может вестись в отношении определенной номенклатуры товаров или услуг, либо по всему ассортименту товаров или услуг, которые предоставляются данной фирмой. Соответственно используемым системам, дисконтные карты также делятся на простые дисконтные карты и карточки со сложным дисконтом, подразделяющиеся, в свою очередь, на карты лояльности и расчетные (заемные, предоплатные, кредитные) дисконтные карты [2].

Дисконтно-бонусная программа дает возможности построения живого общения с покупателем по поводу: информирования о накопленных баллах, их остатке; информации о том, на что можно потратить бонусы с акцентом на продвигаемый товар. Лучшие дисконтные программы – это те, которые грамотно реализованы и способны запустить достаточное количество промо-акций, позволяющих продавцу достигать следующих задач: распродажа товаров; продвижение новинок; формирование портрета покупателя как прибыльного, так и среднестатистического; равномерное распределение нагрузки.

Целью проекта является повышение эффективности работы магазина, что можно достичь автоматизацией системы поддержки управления дисконтной программы сети магазинов.

Задачи проекта:

– провести описание предметной области;

– осуществить подбор инструментария;

– разработать функциональную модель на основе стандарта IDEF0;

– составить описание информационной модели (приведение к 3-ей нормальной форме);

– составить руководство пользователя;

– автоматизировать управление информационной системой;

– проанализировать результаты тестирования автоматизированной системы.

# **1 ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ**

Дисконтная программа — это определённая система скидок, которые предоставляются магазином или другим заведением клиенту. Дисконтные программы бывают разными. Официально их делят на несколько групп, и само деление проводят с разных точек зрения [3].

Дисконтные программы с географической точки зрения разделяют на:

1. международные, они предоставляют привилегии владельцу карты на территории нескольких стран;
2. национальные, они предоставляют привилегии на территории нескольких регионов;
3. региональные, они предоставляют привилегии на территории одного региона;
4. местные, они предоставляют привилегии на территории какой–то части региона.

Характеристика по географическому признаку касается, главным образом, независимых программ и программ на основе платёжных карт. Если же программа организована торговыми и сервисными компаниями самостоятельно или на основе объединения, то она всегда будет ограничена географией деятельности этих компаний и не будет выходить за пределы региона.

Дисконтные программы с точки зрения состава организации, выпускающей дисконтные карты, и состава предприятий–участников разделяют на:

1. локальные, когда предприятие само выпускает дисконтные карты и само эти карты обслуживает, а в других местах их карты недействительны;
2. межфирменные, они бывают двух видов: альянс и клуб. По альянсовой программе у каждого предприятия есть своя организация, которая выпускает для него дисконтные карты. Эти организации и предприятия договариваются между собой предоставлять скидки не только по своим картам, но и по картам друг друга. По клубной программе организация выпускает карты, которые обслуживаются и в других, не принадлежащих ей торговых точках;
3. независимые, когда разрабатывает программу и выпускает карты специализированная компания — она продаёт дисконтные карты и другие услуги клиентам и именно с этого получает прибыль.

Дисконтные программы с точки зрения профиля деятельности предприятий–участников разделяют на:

1. специализированные, при этом в программе участвуют предприятия одного профиля деятельности;
2. объединённые, в том случае, если в программе участвует ограниченное количество фирм, но их профили не пересекаются, благодаря чему отсутствует конкуренция внутри программы. В основном объединёнными являются межфирменные программы типа «альянс»;
3. универсальные, в которых участвуют компании различного профиля, в том числе и конкурирующие друг с другом.

Дисконтные программы с точки зрения предоставляемой привилегии, льготы разделяют на:

1. программы с фиксированной скидкой. Они позволяют покупать товар и оплачивать услуги по сниженной цене. Размер скидки обычно фиксированный, но может и зависеть от суммы разовой покупки или от вида товара, услуги;
2. программы с накопительной скидкой. Такие программы предусматривают увеличение размера скидки по мере увеличения общей суммы, которая была потрачена на покупки в той или иной фирме. Чем больше сумма, тем большую скидку предусматривает программа, например при сумме 5 000 рублей — 5 % скидки; 10 000 рублей — 10 %; 15 000 рублей — 15 %;
3. программы бонусные. В зависимости от стоимости приобретаемых товаров на счёт клиента начисляется то или иное количество неких условных единиц — бонусов, по которым можно получить бесплатно или какой–либо товар, заказав его по специальному бонусному каталогу, или какую–либо услугу. Каждая организация продумывает свой порядок действия бонусов.

Некоторые организации предусматривают ещё одну дисконтную услугу — предоставление скидок на товар только при условии предоплаты.

При введении дисконтной программы предприятие все же должно понимать, что ее простота важна для клиента, однако самой компании придется провести ряд мероприятий:

1) определение «порога для входа» — необходимая для участия в дисконтной программе стоимость покупки. Данный показатель ничем не регламентирован, а потому может вызывать сложности. Как правило, берется средний чек, и его сумма увеличивается в 2-3 раза. Также частой практикой является заимствование опыта конкурентов, эффективно использующих какую-либо дисконтную программу. Затем уже на базе фактических результатов «порог входа» корректируется;

2) размер скидок, величина которых разнообразна в зависимости от сферы деятельности предприятия. Супермаркеты не уходят выше порога в 5 %, ритейлы бытовой техники и электроники остаются в пределах 5-15 %, рестораны и магазины одежды могут подарить скидку и в 25 %. Тут стоит понимать, что завышенная скидка вызывает недоверие. Грамотный покупатель сразу заподозрит неладное, засомневается в качестве товара или адекватности изначальной цены, и стратегия вашей дисконтной программы станет проигрышной;

3) на каких скидках строить программу. Фиксированные они будут или накопительные — что привлечет потребителя и при этом будет простым для понимания? Как создать схему зависимости итоговой скидки от общей суммы покупок при ее увеличении? Принцип накопления привлекателен тем, что покупатель участвует в некой игре, подогревающей его интерес. При этом потребитель знает, какие действия нужно совершить для получения своей выгоды.

Участие покупателей в системе дисконтной программы магазина без ограничений приводит компанию к ситуации, когда теряется приличная часть выручки. Уйти от данной проблемы можно, проведя пересмотр условий и определение минимального порога денежной суммы, которую нужно потратить потребителю для участия в дисконтной программе.

Несмотря на популярность дисконтных программ, большинство экспертов сходятся во мнении, что фиксированная скидка не является залогом выстраивания длительных взаимоотношений между клиентом и магазином. Скидки если и создают лояльность, то к поощрениям, а не компании, их предлагающей

Так, 20% опрошенных абсолютно точно готовы отложить покупку в случае отсутствия дисконтной карты с собой. При этом еще 40% респондентов готовы отложить покупку, если размер скидки, предоставляемой по дисконтной карте, значителен.

Такие выводы привели к переосмыслению программ лояльности в сторону повышения значимости для клиентов. На смену дисконтным программам стали приходить бонусные. При этом бонусы могут быть в виде накопленной скидки, в виде подарка от «родной» компании, либо призов от компаний-партнеров. Самое главное — предлагать такой вид поощрения, который интересен покупателю.

Можно выделить следующие положительные стороны в использовании дисконтной программы для магазина:

1. Нет необходимости задавать пороги изменения размера скидки.
2. Универсальность методов, применяемых к клиентской базе.
3. В случае небольшого размера установленной скидки - возможность привлечения большего количества клиентов.

Отрицательной сторойнно использованния системы является:

1. Нет механизма выстраивания долгосрочных связей с клиентом.
2. Аналогичные предложения у конкурентов и отсутствие индивидуальности программы.

У потребителя могут быть десятки дисконтных карт, но это не свидетельствует о его лояльности этим компаниям. Дисконтная система без обратной связи с клиентом программой лояльности не является. Ведь если клиент получает скидку в зависимости от наличия у него дисконтной карты, а не в зависимости от его покупательской активности, разве это формирует лояльность?

Карточные программы совершенствуются в сторону персонализации. Имя держателя пластиковой карты используется не только при рассылке, чтобы обратиться к клиенту по имени, но и для выявления клиентского профиля и оптимизации рассылаемых сообщений под этот профиль. Помимо рассылок компании получают возможность собирать аналитику по клиентам-держателям карт: отследить поведение новых и постоянных клиентов, мотивы их покупок, а динамика и по тем и другим позволяет оценить эффективность рекламных акций [4].

Существуют некоторые готовые системы для реализации дисконтной программы магазина. Примером является SLS-Дисконт.

Комплекс программ SLS-Дисконт позволит реализовать дисконтную политику в рамках территориально распределенной сети магазинов розничной торговли в соответствии с вашими требованиями и пожеланиями.

Программный комплекс SLS-Дисконт состоит из двух частей:

1. SLS-Дисконт Офис — программа управления дисконтной системой. Устанавливается в центральном офисе (ЦО). С помощью этой части комплекса ведется общая база данных (БД) покупателей торговой сети, сделанных ими покупок, задаются правила предоставления скидок. Также она позволяет провести операцию сбора и обработки информации, поступившей в центральный офис от всех магазинов сети, и обеспечивает подготовку файлов экспорта информации для каждого магазина торговой сети.
2. SLS-Дисконт Магазин — программа для сбора и обработки первичной информации о новых покупателях торговой сети и о покупках, сделанных в магазине с использованием дисконтных карт. Обеспечивает предоставление информации кассовой программе о величине действующей скидки для каждого покупателя из БД, а также регистрацию информации о совершенных покупках с использованием дисконтных карт, подготовленной кассовой программой, и подготовку для ЦО файлов обмена информацией.

Все функции управления дисконтной политикой компании осуществляются в центральном офисе. Вся информация о покупателях торговой сети хранится в базе данных. Каждому покупателю присваивается карточка, в которой записаны номер дисконтной карты и анкетные данные. В этой карточке накапливается информация о покупках, сделанных клиентом в магазинах торговой сети, рассчитывается текущая сумма покупок и процент скидки. Центральный офис назначает магазин, ответственный за конкретного покупателя. В БД этого магазина хранятся анкетные данные о покупателе, и этот же магазин оповещает клиента о проводимых рекламных акциях, новых скидках и пр.

Магазины сети обеспечивают предоставление скидок, сбор информации о клиентах, выдачу дисконтных карт и регистрацию в собственных БД собранной информации. При обмене данными с центральным офисом магазины торговой сети не зависят друг от друга. Обмен между БД магазина и центрального офиса происходит с учетом разницы во времени между магазином и ЦО.

Все процессы экспорта и импорта информации для передачи из центрального офиса в магазины и обратно инициируются операторами систем (оператор выбирает соответствующий пункт меню в системе). Чтобы минимизировать ошибки операторов, в базах данных дисконтной системы описываются и хранятся всевозможные настройки, определяющие параметры подключения к ресурсам и передачи информации (порты, пароли подключения, размещение файлов экспорта-импорта и пр.).

Схема обмена между магазинами и центральным офисом реализована двумя способами: либо по протоколу FTP обмена файлами через FTP-сервер (в предположении, что между компьютерами в центральном офисе и в магазине не существует прямой связи), либо с помощью файлового обмена внутри локально-вычислительной сети [5].

**2 ПОСТАНОВКА ЗАДАЧИ И МЕТОДЫ ЕЕ РЕШЕНИЯ**

**2.1 Детализация задач в области разработки системы**

Задача данной программы заключается в обеспечении пользователя всей необходимой информацией об предоставленной скидке. Необходимо, чтобы программа была построена на основе клиент-серверной архитектуры.

В приложении необходимо реализовать:

1. Авторизация/регистрация пользователей (для разделения прав доступа администратору и пользователю);
2. Пользователь должен видеть всю имеющуюся информация о предоставленных скидках.
3. Администратор должен иметь возможность добавлять, редактировать и удалять уже хранящуюся в базе данных информацию.

**2.2 Обзор методов решения**

Задача курсового проекта решалась с помощью приложения, в основе работы которого лежит так называемая модель взаимодействия клиент-сервера на основе TCP соединения.

Transmission Control Protocol/Internet Protocol (TCP/IP) — это промышленный стандарт стека протоколов, разработанный для глобальных сетей.

Лидирующая роль стека TCP/IP объясняется следующими его свойствами:

1. Это наиболее завершенный стандартный и в то же время популярный стек сетевых протоколов, имеющий многолетнюю историю.
2. Почти все большие сети передают основную часть своего трафика с помощью протокола TCP/IP.
3. Это метод получения доступа к сети Internet.
4. Этот стек служит основой для создания intranet — корпоративной сети, использующей транспортные услуги Internet и гипертекстовую технологию WWW, разработанную в Internet.
5. Все современные операционные системы поддерживают стек TCP/IP.
6. Это гибкая технология для соединения разнородных систем как на уровне транспортных подсистем, так и на уровне прикладных сервисов.
7. Это устойчивая масштабируемая межплатформенная среда для приложений клиент-сервер.

Стек был разработан по инициативе Министерства обороны США (Department of Defence, DoD) более 20 лет назад для связи экспериментальной сети ARPAnet с другими сателлитными сетями как набор общих протоколов для разнородной вычислительной среды. Сеть ARPA поддерживала разработчиков и исследователей в военных областях. В сети ARPA связь между двумя компьютерами осуществлялась с использованием протокола Internet Protocol (IP), который и по сей день является одним из основных в стеке TCP/IP и фигурирует в названии стека [6].

Было реализовано GUI-приложение, которое работает в связке с сервером, а тот в свою очередь с базой данных. В качестве СУБД используется PostgreSQL.

PostgreSQL — это популярная свободная объектно-реляционная система управления базами данных. PostgreSQL базируется на языке SQL и поддерживает многочисленные возможности.

Преимущества PostgreSQL:

* поддержка БД неограниченного размера;
* мощные и надёжные механизмы транзакций и репликации;
* расширяемая система встроенных языков программирования и поддержка загрузки C-совместимых модулей;
* наследование;
* легкая расширяемость.

PostgreSQL не просто реляционная, а объектно-реляционная СУБД. Это даёт ему некоторые преимущества над другими SQL базами данных с открытым исходным кодом, такими как MySQL, MariaDB и Firebird.

Фундаментальная характеристика объектно-реляционной базы данных — это поддержка пользовательских объектов и их поведения, включая типы данных, функции, операции, домены и индексы. Это делает Постгрес невероятно гибким и надежным. Среди прочего, он умеет создавать, хранить и извлекать сложные структуры данных.

Существует обширный список типов данных, которые поддерживает PostgreSQL. Кроме числовых, с плавающей точкой, текстовых, булевых и других ожидаемых типов данных (а также множества их вариаций), PostgreSQL может похвастаться поддержкой uuid, денежного, перечисляемого, геометрического, бинарного типов, сетевых адресов, битовых строк, текстового поиска, xml, json, массивов, композитных типов и диапазонов, а также некоторых внутренних типов для идентификации объектов и местоположения логов. Справедливости ради стоит сказать, что MySQL, MariaDB и Firebird тоже имеют некоторые из этих типов данных, но только PostgreSQL поддерживает их все [7].

Вся реализация производилась на языке Java, так как он является объектно-ориентированным и имеет в комплектации достаточно объемную библиотеку классов, которая значительно облегчает разработку программного обеспечения, предлагая программисту мощные средства решения распространенных задач.

Java — строго типизированный объектно-ориентированный язык программирования общего назначения, разработанный компанией Sun Microsystems (в последующем приобретённой компанией Oracle). Разработка ведётся сообществом, организованным через Java Community Process; язык и основные реализующие его технологии распространяются по лицензии GPL. Права на торговую марку принадлежат корпорации Oracle [8].

В качестве среды разработки была выбрана IntelliJ IDEA.

IntelliJ IDEA — это ведущая среда быстрой разработки на языке Java. IntelliJ IDEA представляет собой высокотехнологичный комплекс тесно интегрированных инструментов программирования.

С помощью Draw.io были созданы UML-диаграммы.

В данной работе использовались два паттерна проектирования:

1. singletone — это порождающий паттерн проектирования, который гарантирует, что у класса есть только один экземпляр, и предоставляет к нему глобальную точку доступа.;
2. delegate — основной шаблон проектирования, в котором объект внешне выражает некоторое поведение, но в реальности передаёт ответственность за выполнение этого поведения связанному объекту.

В курсовом проекте присутствует авторизация пользователей, возможность просмотра, редактирования, удаления данных и создания новых записей. Для работы двух и более пользователей со вкладами одновременно, сервер обладает возможностью параллельной обработки запросов.

Для создания проекта был выбран шаблон MVC.

Model-View-Controller (MVC) (Модель-представление-контроллер) –схема разделения данных приложения, пользовательского интерфейса и управляющей логики на три отдельных компонента: модель, представление и контроллер – таким образом, что модификация каждого компонента может осуществляться независимо.

Модель – часть вашего приложения, которая хранит данные. Проще всего представить это как библиотеку (не для языка программирования, а реальную, с читальным залом): в ней есть книги, а ещё есть правила их выдачи и пользования ими. Это похоже на то, чем занимается модель, только вместо книг данные: этот код отвечает за то, какие из них хранить, при помощи чего и в каком виде.

Представление – то, что видит пользователь, с чем взаимодействует. Можно считать это приборной панелью самолёта: кнопки, рычажки, благода-ря которым, система может понять, чего хочет пользователь.

Контроллер – нечто посередине между двумя предыдущими сущностями. На основе того, как пользователь взаимодействует с представлением, контроллер решает, что и с какими моделями необходимо делать.

В курсовм проекте был использован JavaFX.

JavaFX — это набор инструментов для создания кроссплатформенных приложений на основе языка программирования Java. Этот инструментарий был создан с целью упростить написание приложений. Другими словами, JavaFX — это декларативный язык сценариев со статической типизацией.

Особенности JavaFX:

1. Структура и API-интерфейсы предназначены для создания анимации, 2D- и 3D-геометрии, диаграмм, специальных эффектов, цветовых градиентов, графических элементов управления и простых манипуляций с мультимедиа, включая аудио, видео и изображения. Итог: можно создавать сложный и богатый контент с помощью JavaFX намного проще, чем используя Swing и Java 2D.
2. Визуализация графики JavaFX пользуется возможностями аппаратного ускорения. Это обеспечивает хорошую производительность рендеринга графики и анимации.
3. Можно встраивать содержимое JavaFX в панели Swing с помощью JFXPanel. Магия этого специализированного компонента Swing позволяет создавать сложные приложения платформы NetBeans.

JavaFX задумывался как декларативный язык сценариев (JavaFX Script), который был бы построен на основе Java. Хотя разработчикам нравилась простота декларативного сценария, было сложно интегрировать JavaFX Script с существующими приложениями Swing.

JavaFX 2.0 был выпущен в 2011 году и был основан на API Java. Уже в Java 7 JavaFX был включен в стандартный выпуск, а начиная с Java 8, который включает в себя возможности 3D, все библиотеки JavaFX (файлы JAR) включены в стандартный путь к классам.

Это означает, что JavaFX теперь является частью стандарта Java при загрузке Java Development Kit (JDK) [9].

Для выполнения UML-моделей в стандарте IDEF0 использовалось CASE-средство CA AllFusion Process Modeler r7 (BPwin). Для информационного моделирования применялось средство CA AllFusion ERwin Data Modeler r7 (ERwin).

BPwin это программный продукт, разработанный компанией ltd. Logic Works. Он предназначен для поддержки процесса создания информационных систем. Относится к категории CASE средств верхнего уровня. Первая версия BPwin была выпущена в 1995 г. совместно с другим CASE средством - ERwin, предназначенным для моделирования данных. В дальнейшем, развитием и поддержанием BPwin занималась компания Platinum Technology, а последние версии разрабатывала компания CA Technologies.

BPwin является достаточно развитым средством моделирования, позволяющим проводить анализ, документирование и улучшение бизнес-процессов. С его помощью можно моделировать действия в процессах, определять их порядок и необходимые ресурсы. Модели BPwin создают структуру, необходимую для понимания бизнес-процессов, выявления управляющих событий и порядка взаимодействия элементов процесса между собой.

BPwin поддерживает функциональное моделирование, моделирование потока работ и потока данных. Соответствующие диаграммы реализованы на основе стандартов IDEF0, IDEF3 и DFD. Функциональное моделирование дает возможность осуществлять систематизированный анализ бизнес-процессов, обращая внимание на регулярно выполняемые задачи (функции). Моделирование потока работ обеспечивает анализ логики выполнения процесса. Моделирование потока данных позволяет сконцентрировать внимание на обмене данными между различными задачами. Кроме того, что в BPwin создаются отдельные модели, также могут создаваться и смешанные модели.

Для анализа работы организации в комплексе, и построения больших моделей, в BPwin предусмотрена детализация. Модели могут быть разбиты на группы. Каждая модель представляется на более низком уровне детализации. При этом взаимосвязь между моделями и их элементами сохраняется. С помощью BPwin модель можно разделить на составляющие части, провести работу отдельно с каждой из них, а затем интегрировать обратно в единую модель [10].

# **3 ФУНКЦИОНАЛЬНОЕ МОДЕЛИРОВАНИЕ НА**

# **ОСНОВЕ СТАНДАРТА IDEF0**

Наилучшим способом моделирования является функциональная стратегия, которая базируется на функциональных взаимоотношениях действий системы, т.к. она заставляет автора точно определить: что делает система, независимо от того, как она работает. Такая чисто функциональная ориентация является принципиальной — функции системы анализируются независимо от объектов, которыми они оперируют. Это позволяет более четко смоде-лировать логику и взаимодействие процессов организации.

Функциональное моделирование — это процесс моделирования функций, выполняемых рассматриваемой информационной системой/объектом, путем создания описательного структурированного графического изображения, показывающего что, как и кем делается в рамках функционирования объекта и объектов, связывающих эти функции, с учетом имеющейся информации.

Целью создания функциональной модели процесса является точная спецификация всех функций, осуществляемых в рамках процесса более высокого уровня иерархии, а также характера взаимосвязей между ними. Будучи построенной, такая модель способна обеспечить полное представление, как о функционировании обследуемого процесса, так и обо всех имеющих в нем место потоках информации и материалов.

Функциональная модель позволяет четко определить распределение ресурсов между операциями делового процесса, что дает возможность оценить эффективность их использования.

IDEF0 — это методология функционального моделирования и поэтому имя блока, описывающее функцию, должно быть глаголом или глагольным оборотом. Примеры имен функций: производить детали, наблюдать за выполнением.

Стрелки и их сегменты помечаются существительными или оборотами существительных. Примеры меток стрелок: менеджер, бюджет.

Каждая сторона блока имеет своё определенное значение с точки зрения связи блок-стрелка. Верхняя сторона имеет значение «управление», левая - «вход», правая - «выход», а нижняя - «механизм». В свою очередь, сторона блока, к которой присоединена стрелка, однозначно определяет ее роль.

В IDEF0 различают пять классов стрелок - стрелка входа, стрелка выхода, стрелка управления, стрелка механизма, стрелка вызова.

Стрелка входа — это материал или данные, которые преобразуются или расходуются функцией, чтобы создать то, что появится на ее выходе. Стрелка входа рисуется как входящая в левую грань блока. Допускается, что функция может не иметь ни одной стрелки входа. Часто бывает сложно определить, являются ли данные входом, или управлением. В том случае, когда данные изменяются или перерабатываются, это вход, если нет - управление.

Стрелка управления — это правила, стратегии, процедуры, стандарты, которые определяют условия, необходимые функции, чтобы произвести правильный выход. Стрелка управления рисуется как входящая в верхнюю грань блока. Каждая функция должна иметь хотя бы одну стрелку управления. Управление влияет на функцию, но не преобразуется функцией. Если цель функции - изменить процедуру, то такая процедура будет для функции входом. В случае возникновения неопределенности в классифицировании стрелки (вход или управление) рекомендуется создавать стрелку управления.

Стрелка выхода — это данные или материальные объекты, произведенные функцией. Стрелка выхода рисуется как выходящая из правой грани блока. Каждая функция должна иметь хотя бы одну стрелку выхода. Функция без выхода не имеет смысла и не должна моделироваться.

Стрелка механизма — это ресурсы (персонал, техника, оборудование), поддерживающие выполнение функции. Стрелка механизма рисуется как входящая в нижнюю грань блока. Стрелка механизма может не изображаться на модели.

Стрелка вызова — это стрелка, указывающая на другую модель. Стрелка вызова рисуется как исходящая из нижней грани блока. Такая стрелка используется как указание на то, что некоторая функция выполняется за пределами моделируемой системы. На верхнем уровне моделирования системы аэропорта информация, которая подаётся на вход, является основополагающей. Данный класс стрелок отображает данные, которые преобразуются функцией. Этими данными являются: пассажир, запрос на багаж, личные данные, предложения авиакомпаний.

Благодаря корректной семантике разработанного первого уровня модели появилась возможность получить полное, достоверное описание системы. При формулировании названия диаграммы необходимо учитывать, что будет отображаться перечень вопросов, которые необходимо решить данной системы.

Наиболее важные свойства объекта выявляются на верхнем уровне иерархии, по мере декомпозиции функции верхнего уровня и разбиения ее на подфункции эти свойства уточняются. Каждая подфункция декомпозируется на элементы следующего уровня. Декомпозиция происходит до тех пор, пока не будет получена релевантная структура, позволяющая ответить на вопросы, сформулированные в цели моделирования [11].

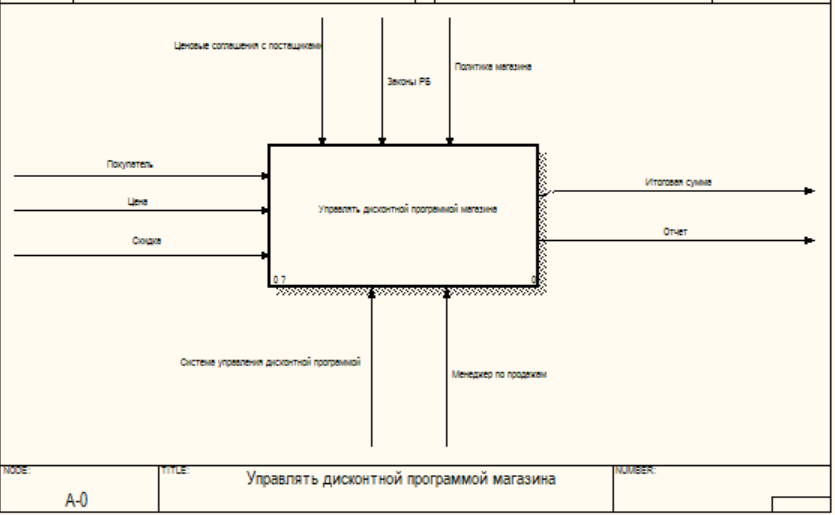


Рисунок 3.1 – Контекстная диаграмма

При этом входными данными для контесктной модели (см. рис 3.1) будут служить покупатель, цена, скидка, а выходными – итоговая сумма и отчет. Механизмами служат менеджер по прадажам, выполняющий свои обязанности, система управления дисконтной программой. Управление служат как законы РБ, политика магазина и ценовые соглашения с поставщиками.

Как упоминалось ранее, для полного определения перечня вопросов и структуры необходимо провести декомпозицию главного процесса. Для приобретения более чётких и подробных описаний проектируемой системы необходимо осуществить декомпозицию верхнего уровня диаграммы (рис 3.2). Особое внимание следует уделить при моделировании данного уровня диаграмм связыванию функций стрелками для решения обозначенных в цели моделирования задач.

Следующая диаграмма отображает основные цели, преследуемые системой. Так как автоматизация позволяет повысить [производительность труда](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B8%D0%B7%D0%B2%D0%BE%D0%B4%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%BE%D1%81%D1%82%D1%8C_%D1%82%D1%80%D1%83%D0%B4%D0%B0), улучшить [качество продукции](https://ru.wikipedia.org/wiki/%D0%9A%D0%B0%D1%87%D0%B5%D1%81%D1%82%D0%B2%D0%BE_%D0%BF%D1%80%D0%BE%D0%B4%D1%83%D0%BA%D1%86%D0%B8%D0%B8), [оптимизировать](https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D1%82%D0%B8%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%D1%80%D0%B5%D1%88%D0%B5%D0%BD%D0%B8%D0%B5) процессы управления, следовательно необходимо автоматизировать основные процессы.

Деятельность дисконтной программы включает в себя следующие этапы (см. рис. 3.2):

– зарегистрировать покупателя в системе;

– добавить информацию в базу данных;

– рассчитать итоговую сумму;

– сформировать отчет.

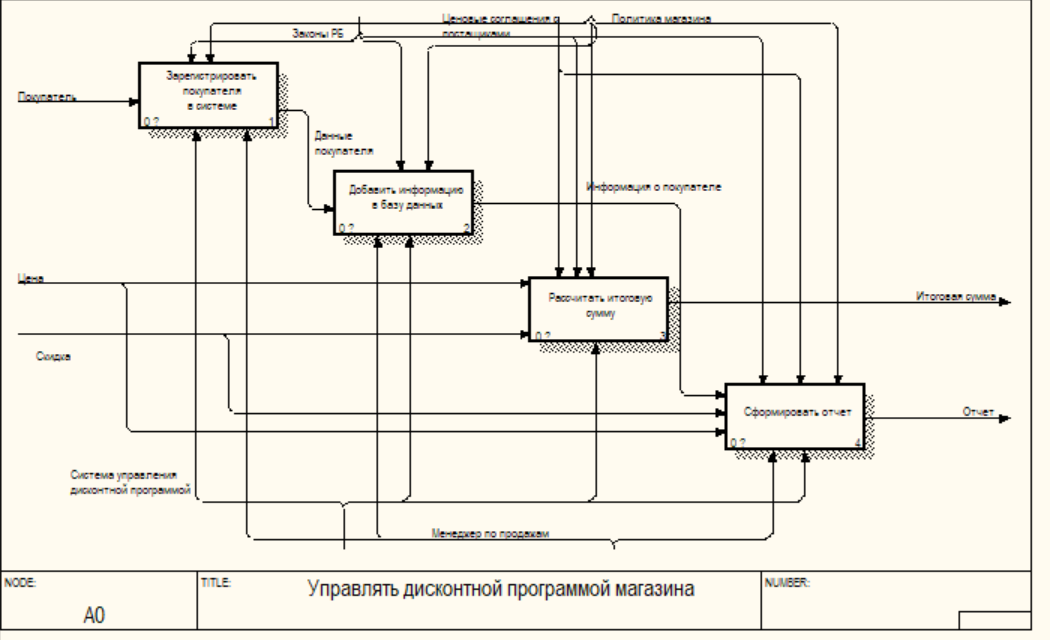


Рисунок 3.2 – Декомпозиция контекстной модели

Далее декомпозируем процесс регистрации:

– проверить наличиепокупателя в системе;

– заполнить данные о покупателе;

– определить уровень доступа к данным.

В результате данного этапа на выходе мы получаем данные покупателя (см.рис.3.3).

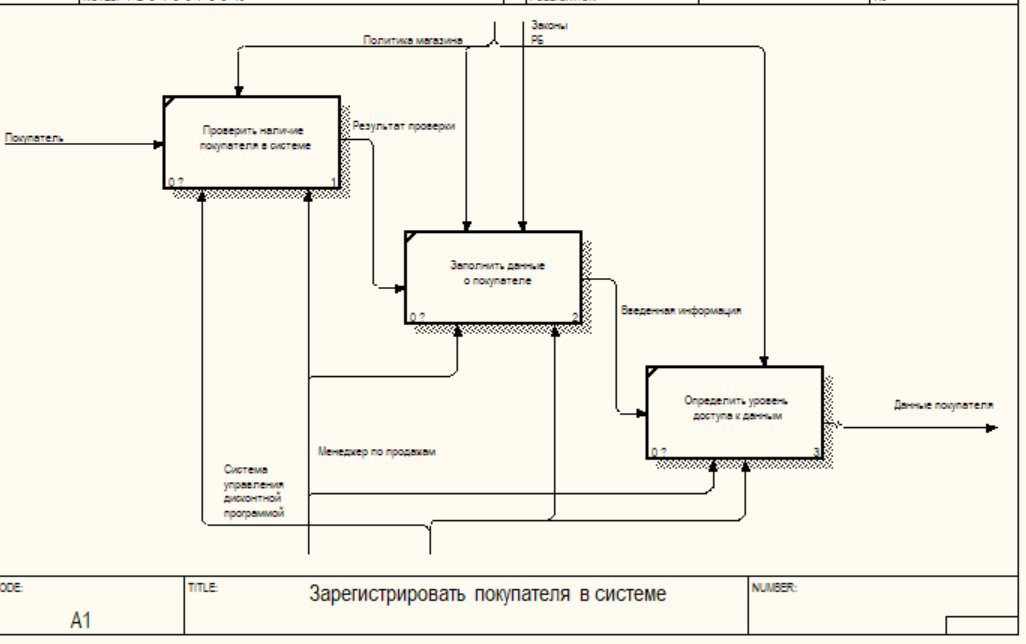


Рисунок 3.3 – Декомпозиция блока «Зарегистрировать покупателя в системе»

Декомпозиция процесса добавления информации в базу данных:

– проверить уровень доступа;

– заполнить данные;

– проверить правильность ввода;

– добавить иеформацию.

В результате данного этапа на выходе мы получаем информацию о покупателях, зарегистрированных в системе и находящихся в базе данных (см. рис. 3.4).

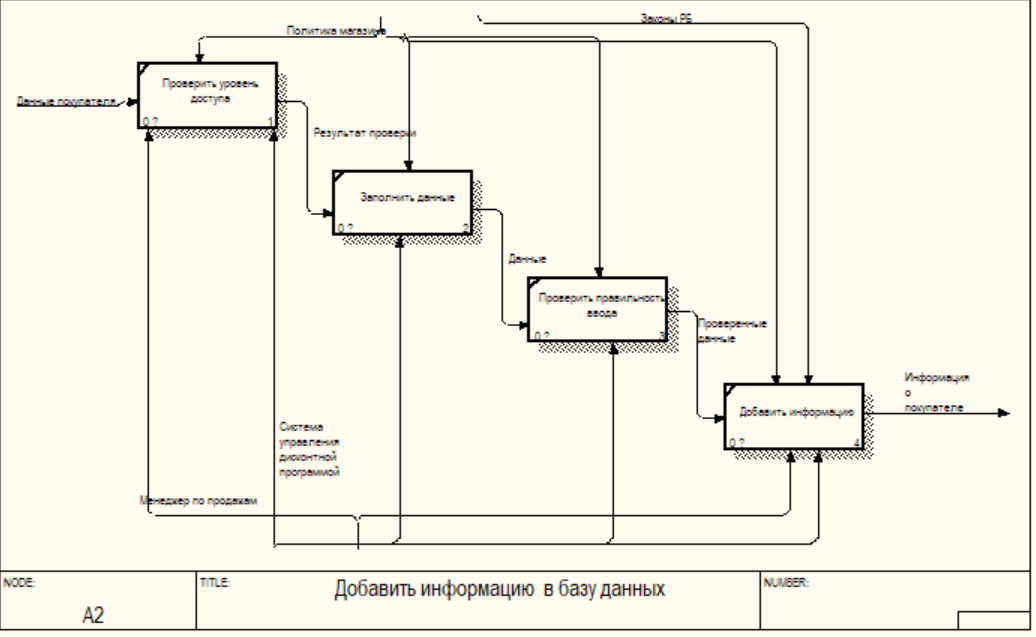


Рисунок 3.4 – Декомпозиция блока «Добавить информацию в базу данных»

Далее декомпозируем процесс рассчета суммы (см. рис. 3.5):

–заполнение ячеек;

– проверка на правильность данных;

– вывод расчета.

Декомпозиция процесса проверки на правильность (см. рис. 3.6):

– проверка на целостность;

– проверка на валидность.

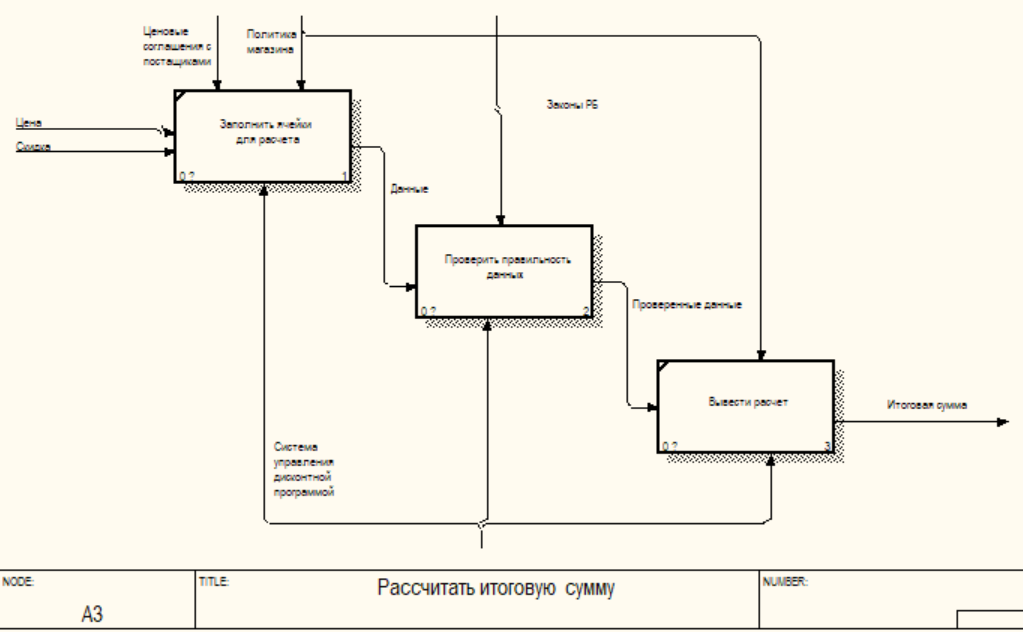


Рисунок 3.5 – Декомпозиция блока «Рассчитать итоговую сумму»

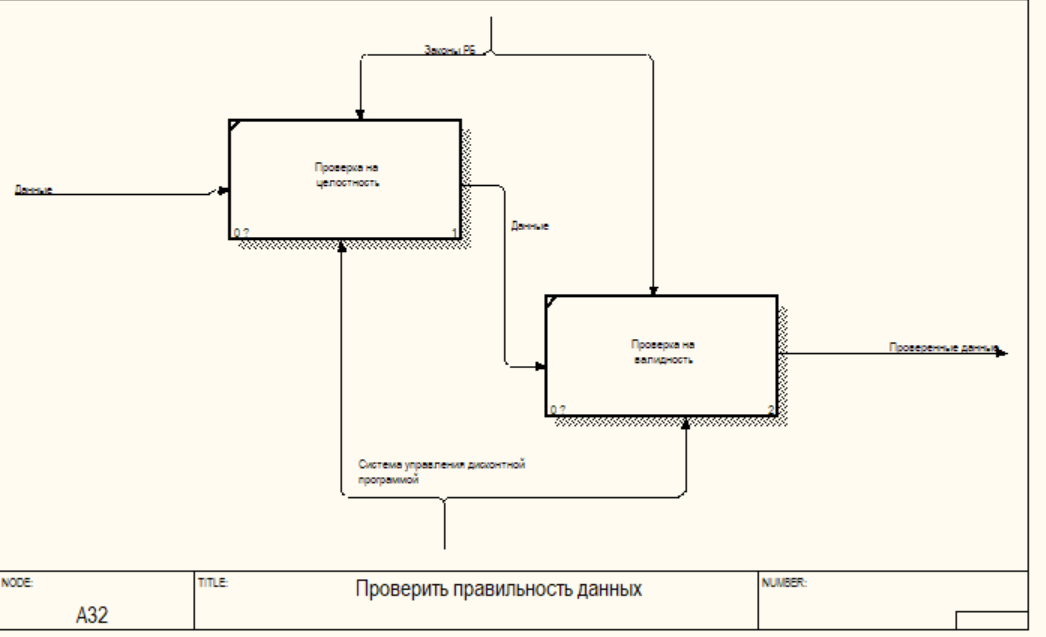


Рисунок 3.6 – Декомпозиция блока «Проверить правильность данных»

Декомпозиция процесса формирования отчета (см. рис. 3.7):

– выьрать необходимые данные;

– добавить данные в отчет.

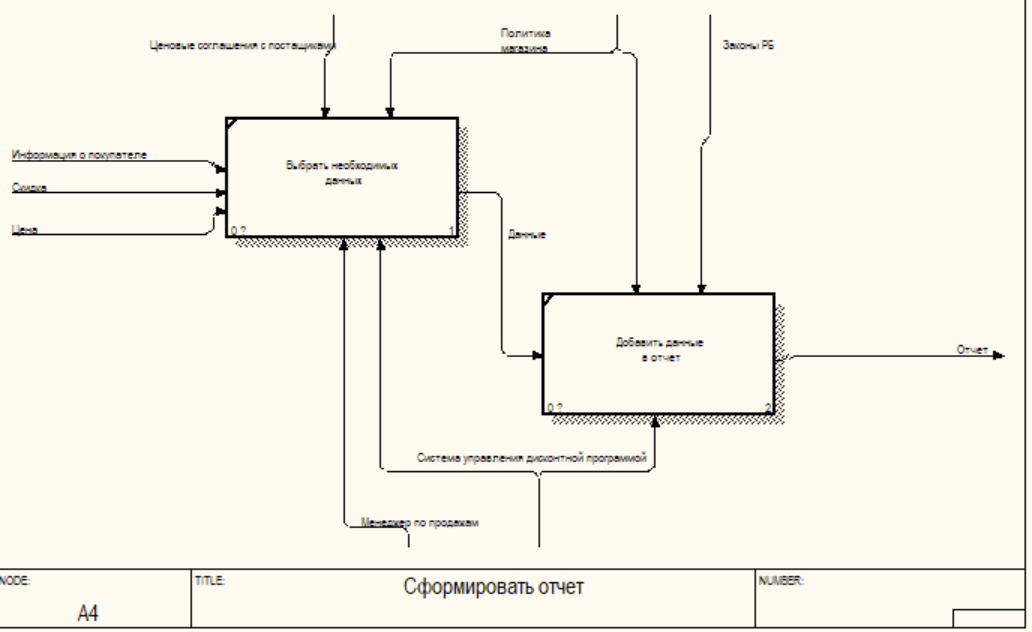


Рисунок 3.7 – Декомпозиция блока «Сформировать отчет»

Целью построение функциональной модели является описание всех необходимых процессов с точностью, достаточной для однозначного моделирования деятельности системы.

Одним из положительных результатов построения функциональ­ной модели оказывается прояснение границ моделирования системы в целом и ее основных компонентов. Хотя и предполагается, что в процессе работы над моделью будет происходить некоторое измене­ние границ моделирования, их вербальное (словесное) описание должно поддерживаться с самого начала для обеспечения координа­ции работы участвующих в проекте аналитиков. Как и при определе­нии цели моделирования, отсутствие границ затрудняет оценку степе­ ни завершенности модели, поскольку границы моделирования имеют тенденцию к расширению с ростом размеров модели.

# **4 ИНФОРМАЦИОННАЯ МОДЕЛЬ СИСТЕМЫ И ЕЕ ОПИСАНИЕ**

Информационная система – это коммуникационная и вычислительная система по сбору, хранению, обработке и передаче информации, снаб­жающая работников различного ранга той информацией, которая необходима им для реализации функций управления.

Информационная модель представляет собой схему движения входных, промежуточных и результативных потоков и функций предметной области. Кроме того, она объясняет, на основе каких входных документов и какой нормативно-справочной информации происходит выполнение функций по обработке данных и формирование конкретных выходных документов.

Так же информационная модель системы отражает информацию о предметной области, так называемой части реального мира, данные о которой должны храниться в проектируемой базе данных. В данном случае предметной областью является дисконтная программа. При этом информация, необходимая для описания информационной системы, зависит от типа самой программы.

Информационные модели делятся на описательные и формальные.

Описательные информационные модели — это модели, созданные на естественном языке (т.е. на любом языке общения между людьми: английском, русском, китайском, мальтийском и т.п.) в устной или письменной форме.

Формальные информационные модели — это модели, созданные на формальном языке (т.е. научном, профессиональном или специализированном). Примеры формальных моделей: все виды формул, таблицы, графы, карты, схемы и т.д.

Хроматические (информационные) модели — это модели, созданные на естественном языке семантики цветовых концептов и их онтологических предикатов (т.е. на языке смыслов и значений цветовых канонов, репрезентативно воспроизводившихся в мировой культуре). Примеры хроматических моделей: "атомарная" модель интеллекта (АМИ), межконфессиональная имманентность религий (МИР), модель аксиолого-социальной семантики (МАСС) и др., созданные не базе теории и методологии хроматизма.

Основная задача моделирования – получить дополнительную информацию об объекте. Модели применяются для изучения объектов, которые не доступны для натурного эксперимента. Для этого происходим замена реального объекта ему подобным, который отражает определенные свойства.

С помощью такой замены доступно:

1.Изучение недоступных для натурного эксперимента объектов. Например, используя ускоренную съемку можно наблюдать за ростом цветка.

2.Исследование гипотетических объектов или реальных объектов в гипотетических условиях.

3.Выполнение имитации на модели и получение дополнительной информации в условиях информационной неопределённости.

4.Осуществление обучения людей в различных областях науки и техники.

5.Формирование справочно-экспертных систем.

6.Создание автоматизированных систем проектирования и управления производством.

7.Объединение неформальных и формальных методов исследования для формирования основы создания искусственного интеллекта.

Возможные недостатки проведения экспериментов на моделях:

1.Сложная конструкция модели или высокая стоимость эксперимента.

2.Сложность интерпретации результатов эксперимента на модели.

3.Сложность в оценке достоверности полученных результатов моделирования.

Для того, чтобы обеспечить минимальную избыточность данных и физического объёма данных, а также ускоренный доступ, необходимо привести информационную модель к нормальной форме. При этом система называется нормализованной только в том случае, если она способна удовлетворять следующим требованиям: надежное хранение и обновление данных. Это поможет сократить риск потери данных или их искажения при внесении в БД [12].

Для моделирования системы используются три этапа проектирования:

– концептуальное проектирование;

– логическое проектирование;

– физическое проектирование;

Концептуальное (инфологическое) проектирование – анализ предметной области и ее описание. Этот этап осуществляется без ориентации на какие-либо конкретные программные или технические средства [13].

Даталогическое (логическое) проектирование – описание логической структуры данных средствами системы управления базами данных (СУБД), для которой проектируется БД. Описание данной модели основывается на построении концептуальной модели. Даталогическое проектирование включает в себя:

− описание таблиц;

− описание связей между таблицами;

− описание атрибутов.

На данном этапе проектирования информационной модели осуществляется приведение базы к нормальным формам, что позволяет на этапе физического проектирования осуществить корректную работы всей проектируемой системы. Модель системы изображена на рисунке 4.1:

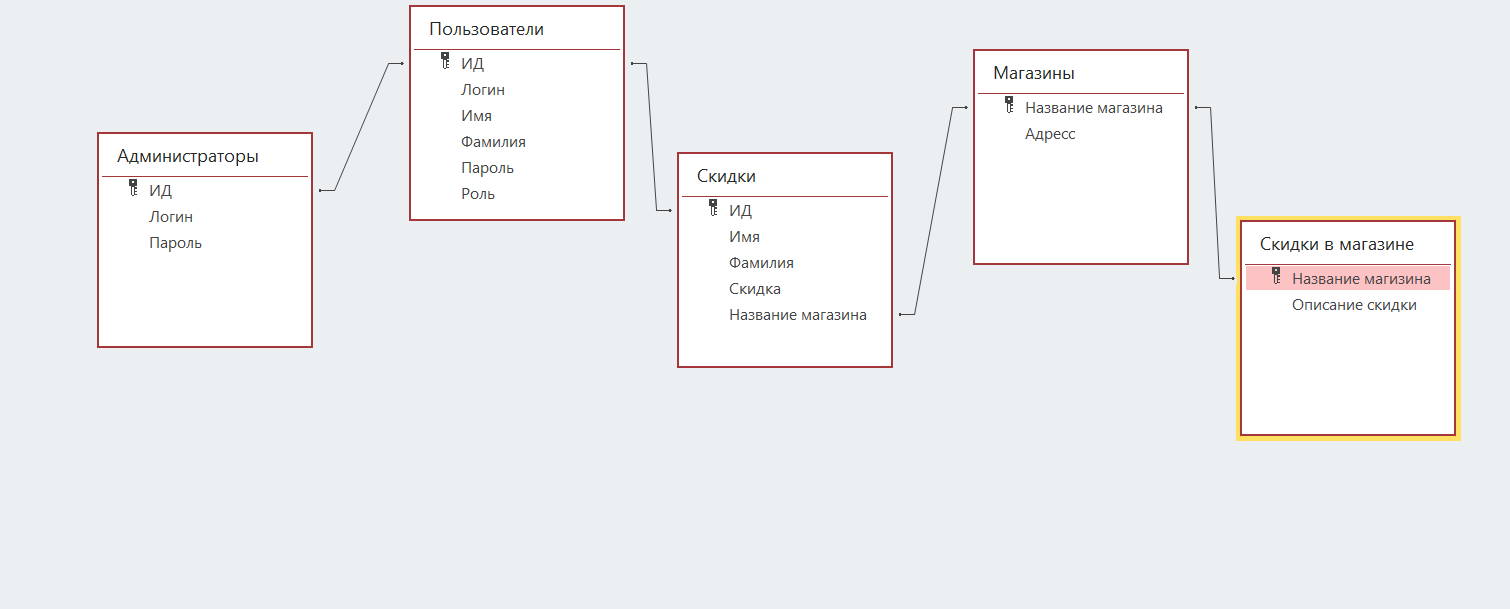


Рисунок 4.1 – Логическая модель системы

Для создания ER-модели необходимо выделить сущности данной предметной области, которые образуют структуру проектируемой информационной системы:

– Администраторы;

– Пользователи;

– Скидки;

– Скидки в магазине;

– Магазины.

Данная модель данных, состоящая из 5 таблиц, представлена на рисунке 4.1.

В таблице «Пользователи» хранится информация о всех пользователях. Атрибутами данной сущности являются:

– уникальный номер пользователя;

– фамилия пользователя;

– имя пользователя;

– логин пользователя;

– пароль пользователя;

– роль.

В таблице «Администраторы» хранится информация о всех администраторах. Атрибутами данной сущности являются:

– уникальный номер администратора;

– логин;

– пароль.

Роль определяет функционал пользователя системы. Функционал различается в зависимости от роли: пользователь или администартор.

В таблице «Магазины» хранится информация о всех магазинах:

– название магазина;

– адресс.

В таблице «Скидки в магазине» хранится информация о представленных скидках в магазинах:

– название магазина;

– описание скидки.

В таблице «Скидки» хранится информация о полученных скидках:

– уникальный номер пользователя;

– фамилия пользователя;

– имя пользователя;

– скидка;

– название магазина.

Для приведения таблиц в базе данных к третьей нормальной форме (3НФ) в таблицах, приведенных ко 2НФ, необходимо устранить зависимость неключевых полей от других неключевых полей. Если упростить данное правило, то можно сформулировать его так: необходимо выносить все неключе-вые поля, содержимое которых может относиться к нескольким записям таб-лицы, в отдельные таблицы.

Разработанная база данных приведена к третьей нормальной форме, так как у каждой таблицы есть всего один первичный ключ, и каждое неключевое поле не транзитивно зависит от первичного ключа, что означает, если мы изменим значение в одном столбце, то изменение в другом столбце не потребуется.

**5 ОПИСАНИЕ АЛГОРИТМА, РЕАЛИЗУЮЩЕГО**

**БИЗНЕС-ЛОГИКУ СЕРВЕРНОЙ ЧАСТИ СИСТЕМЫ**

Бизнес-логика — в разработке информационных систем — совокупность правил, принципов, зависимостей поведения объектов предметной области (области человеческой деятельности, которую система поддерживает). Иначе можно сказать, что бизнес-логика — это реализация правил и ограничений автоматизируемых операций. Является синонимом термина «логика предметной области» (англ. domain logic). Бизнес-логика задает правила, которым подчиняются данные предметной области.

Проще говоря, бизнес-логика — это реализация предметной области в информационной системе. К ней относятся, например, формулы расчёта ежемесячных выплат по ссудам (в финансовой индустрии), автоматизированная отправка сообщений электронной почты руководителю проекта по окончании выполнения частей задания всеми подчиненными (в системах управления проектами), отказ от отеля при отмене рейса авиакомпанией (в туристическом бизнесе) и т. д.

В фазе бизнес-моделирования и разработки требований бизнес-логика может описываться в виде:

* текста;
* концептуальных аналитических моделей предметной области (онтологии);
* бизнес-правил;
* разнообразных алгоритмов;
* диаграмм деятельности;
* графов и диаграмм перехода состояний;
* моделей бизнес-процессов.

В фазе анализа и проектирования системы бизнес-логика воплощается в различных диаграммах языка UML или ему подобных. В фазе программирования бизнес-логика воплощается в коде классов и их методов, в случае использования объектно-ориентированных языков программирования, или процедур и функций, в случае применения процедурных языков [14].

Приложение разработано на основе архитектуры «клиент-сервер». Для создания графического интерфейса была использована библиотека JavaFX.  
 Для начала работы необходимо осуществить идентификацию для базы данных. Пользователь вводит данные на стороне клиента. При этом необходимо подключить клиента по уникальному порту. После обработки данных в консоли сервера выводится сообщения об успешном установлении соединения. Затем сервер отправляет сообщение клиенту о результатах выполнения своей работы. Выведенное сообщение отображает результат выполненных работ.

Далее пользователю необходимо авторизоваться. Для этого в соответствующей форме необходимо ввести уникальные пароль и логин. После авторизации данные передаются на сервер. Данные обрабатываются на стороне базы данных, где хранится информация о зарегистрированных пользователях. В случае успешной авторизации выводится сообщение о выполненной работе.

Алгоритм авторизации пользователя представлен на рисунке 5.1.

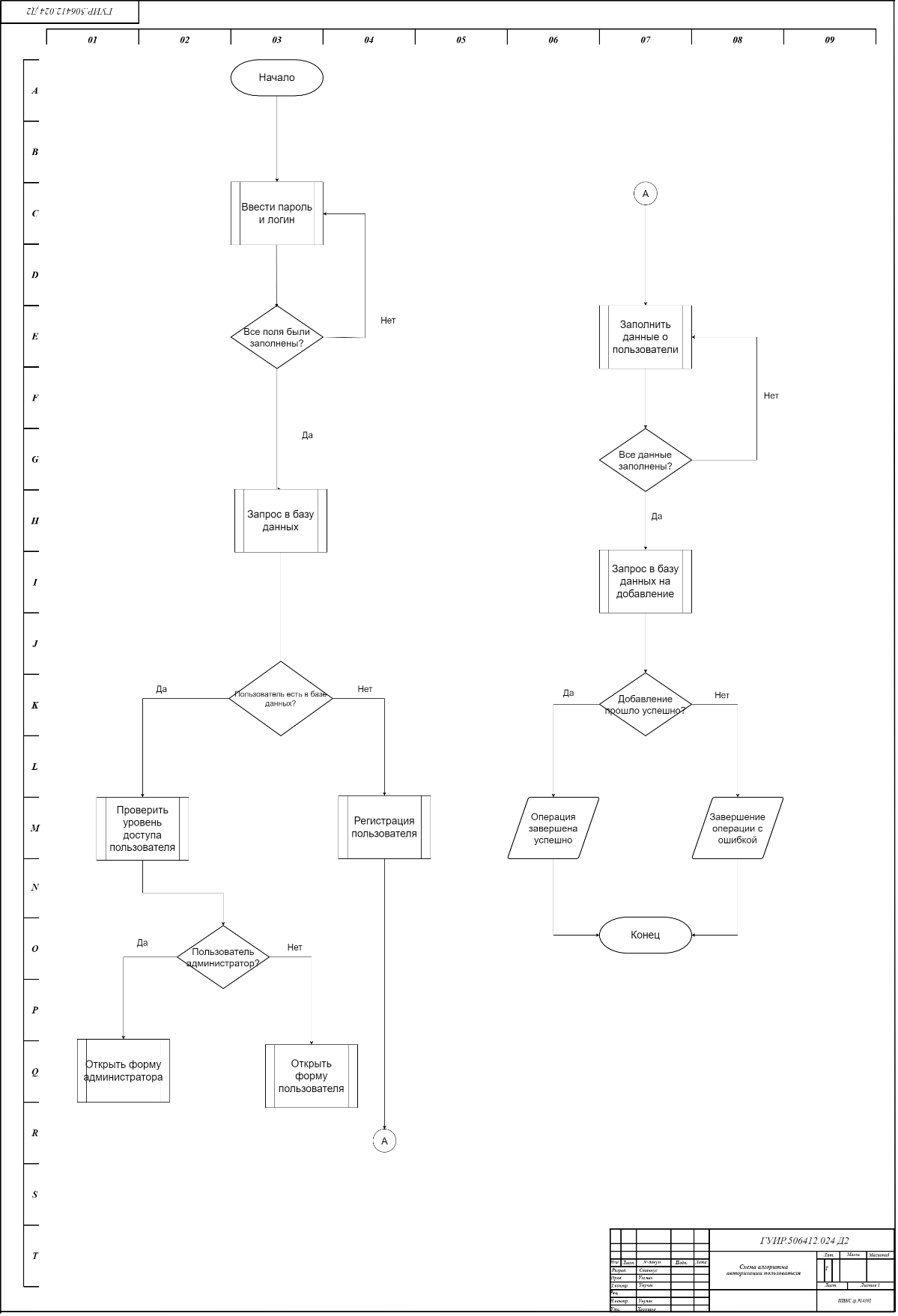
****

Рисунок 5.1 – Алгоритм авторизации пользователя

# **6 МОДЕЛИ ПРЕДСТАВЛЕНИЯ СИСТЕМЫ И ИХ ОПИСАНИЕ**

UML — это унифицированный графический язык моделирования для описания, визуализации, проектирования и документирования ОО систем. UML призван поддерживать процесс моделирования ПС на основе ОО подхода, организовывать взаимосвязь концептуальных и программных понятий, отражать проблемы масштабирования сложных систем. Модели на UML используются на всех этапах жизненного цикла ПС, начиная с бизнес-анализа и заканчивая сопровождением системы. Разные организации могут применять UML по своему усмотрению в зависимости от своих проблемных областей и используемых технологий.

Модель представляется в виде сущностей и отношений между ними, которые показываются на диаграммах.

Сущности — это абстракции, являющиеся основными элементами моделей. Имеется четыре типа сущностей — структурные (класс, интерфейс, компонент, вариант использования, кооперация, узел), поведенческие (взаимодействие, состояние), группирующие (пакеты) и аннотационные (комментарии). Каждый вид сущностей имеет свое графическое представление. Сущности будут подробно рассмотрены при изучении диаграмм. Отношения показывают различные связи между сущностями [15].

В UML предусмотрены различные диаграммы.

Диаграммы, описывающие поведение системы:

1. Диаграммы состояний (State diagrams),
2. Диаграммы деятельностей (Activity diagrams),
3. Диаграммы объектов (Object diagrams),
4. Диаграммы последовательностей (Sequence diagrams),
5. Диаграммы взаимодействия (Collaboration diagrams);

Диаграммы, описывающие физическую реализацию системы:

1. Диаграммы компонент (Component diagrams);
2. Диаграммы развертывания (Deployment diagrams).

6.1 Диаграмма классов

Диаграммы классов используются при моделировании ПС наиболее часто. Они являются одной из форм статического описания системы с точки зрения ее проектирования, показывая ее структуру. Диаграмма классов не отображает динамическое поведение объектов изображенных на ней классов. На диаграммах классов показываются классы, интерфейсы и отношения между ними.

Класс — это основной строительный блок ПС. Это понятие присутствует и в ОО языках программирования, то есть между классами UML и программными классами есть соответствие, являющееся основой для автоматической генерации программных кодов или для выполнения реинжиниринга. Каждый класс имеет название, атрибуты и операции. Класс на диаграмме показывается в виде прямоугольника, разделенного на 3 области. В верхней содержится название класса, в средней — описание атрибутов (свойств), в нижней — названия операций — услуг, предоставляемых объектами этого класса.

Атрибуты класса определяют состав и структуру данных, хранимых в объектах этого класса. Каждый атрибут имеет имя и тип, определяющий, какие данные он представляет. При реализации объекта в программном коде для атрибутов будет выделена память, необходимая для хранения всех атрибутов, и каждый атрибут будет иметь конкретное значение в любой момент времени работы программы. Объектов одного класса в программе может быть сколь угодно много, все они имеют одинаковый набор атрибутов, описанный в классе, но значения атрибутов у каждого объекта свои и могут изменяться в ходе выполнения программы.

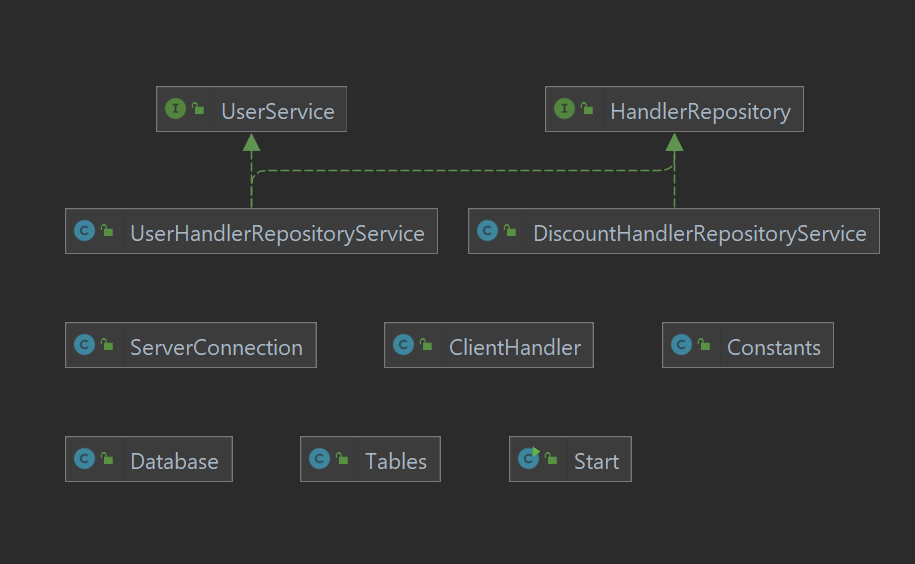
Диаграмма классов проектируемой системы представлена на рисунке 6.1. на диаграмме отображены связи наследования между родительскими и дочерними классами.

Рисунок 6.1 – Диаграмма классов

**6.2 Диаграмма компонентов**

Компонентная диаграмма (component diagram) — первая из двух разновидностей диаграмм реализации (implementation diagrams), моделирующих физические аспекты объектно-ориентированных систем. Компонентная диаграмма показывает организацию набора компонентов и зависимости между компонентами.

Элементами компонентных диаграмм являются компоненты и интерфейсы, а также отношения зависимости и реализации. Как и другие диаграммы, компонентные диаграммы могут включать примечания и ограничения. Кроме того, компо­нентные диаграммы могут содержать пакеты или подсистемы, используемые для группировки элементов модели в крупные фрагменты.

По своей сути компонент является физическим фрагментом реализации системы, который заключает в себе программный код (исходный, двоичный, исполняемый), сценарные описания или наборы команд операционной системы (имеются в виду командные файлы). Язык UML дает следующее определение.

Компонент (component) — физическая и заменяемая часть системы, которая соответствует набору интерфейсов и обеспечивает реализацию этого набора интерфейсов. Графически компонент изображается как прямоугольник с вкладками, обычно включающий имя компонента и, возможно, некоторую дополнительную информацию. Изображение этого символа может незначительно варьироваться в зависимости от характера ассоциируемой с компонентом информации.

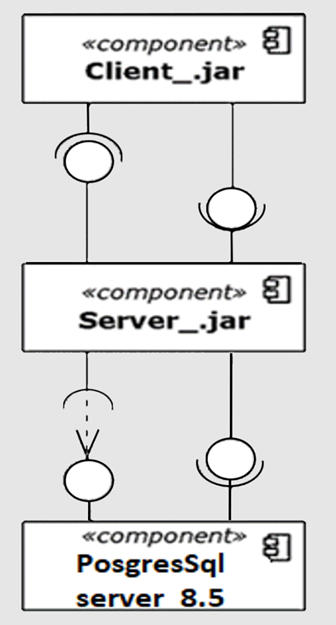
Диаграмма компонентов представлена на рисунке 6.2. Она позволяет определить состав компонентов программы, а также установить зависимость между ними.

Рисунок 6.2 – Диаграмма компонентов

**6.3 Диаграмма развертывания**

Диаграмма развертывания — диаграмма, на которой изображается конфигурация для работающих узлов и экземпляров компонента, а также объектов, которые на них существуют. Компоненты представляют собой выполнимые блоки программного кола. Компоненты, которые не существуют как единицы времени выполнения (в том случае, если они не были задействованы при компиляции), на этих диаграммах не указываются. Их место — на диаграммах компонентов. На диаграмме развертывания изображаются экземпляры компонентов, в то время как на диаграмме компонентов показаны определения их типов.

В представлении развертывания содержатся экземпляры узлов, соединенные между собой коммуникационными связями. Узлы могут содержать экземпляры времени выполнения, такие как экземпляры компонентов и объекты. Экземпляры компонентов и объекты могут, в свою очередь, содержать в себе прочие объекты. Модель может демонстрировать зависимости, существующие между экземплярами и их интерфейсами. Кроме этого, она может отображать миграцию сущностей между узлами и другими контейнерами. У представления развертывания есть две формы - форма дескриптора и форма экземпляра. Форма экземпляра (которую мы описали выше) описывает месторасположение конкретных экземпляров компонентов на конкретных экземплярах узлов в качестве одной из составляющих конфигурации системы.

Форма дескриптора — более привычный вид представления развертывания. Она указывает, на каких видах узлов могут существовать определенные компоненты, а также какие виды узлов могут соединяться между собой. Иными словами, эта форма напоминает диаграмму классов.

Диаграмма развертывания представляет собой схему, состоящую из символов узлов, соединенных маршрутами коммуникационных ассоциаций. Внутри символов узлов могут находиться экземпляры компонентов. Это указывает на то, что компонент существует или выполняется на данном узле. Символы компонентов, в свою очередь, могут содержать в себе символы объектов. Это обозначает, что объект является частью компонента. Компоненты связаны между собой пунктирными стрелками зависимостей (возможно, через интерфейсы). Такие связи обозначают, что один компонент использует услуги другого. Когда на диаграмме нужно точно указать вид зависимости, используют стереотип.

Разработанная диаграмма развертывания представлена на рисунке 6.3. Цели разработки данной диаграммы можно сформулировать так:

– распределение компонентов системы по ее физическим узлам;

– отображение физических связей между узлами системы на этапе ис-

полнения;

– выявление узких мест системы и реконфигурация ее топологии с це-

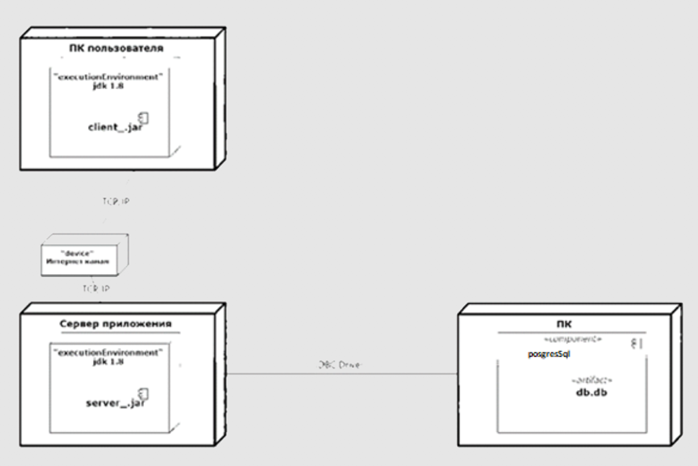
лью достижения требуемой производительности.

Рисунок 6.3 – Диаграмма развертывания

**6.4 Диаграмма состояний**

Диаграмма состояний покажет нам все возможные состояния, в которых может находиться объект, а также процесс смены состояний в результате внешнего влияния.

Основными элементами диаграммы состояний являются «Состояние» и «Переход». Диаграмма состояний имеет схожую семантику с диаграммой деятельности, только деятельность здесь заменена состоянием, переходы символизируют действия. Таким образом, если для диаграммы деятельности отличие между понятиями «Деятельность» и «Действие» заключается в возможности дальнейшей декомпозиции, то на диаграмме состояний деятельность символизирует состояние, в котором объект находится продолжительное количество времени, в то время как действие моментально.

Состояние может содержать только имя или имя и дополнительно список внутренних действий. Список внутренних действий содержит перечень действий или деятельностей, которые выполняются во время нахождения объекта в данном состоянии. Данный список фиксированный. Список основных действий включает следующие значения:

* entry - действие, которое выполняется в момент входа в данное состояние (входное действие);
* exit - действие, которое выполняется в момент выхода из данного состояния (выходное действие);
* do - выполняющаяся деятельность ("do activity") в течение всего времени, пока объект находится в данном состоянии
* defer - событие, обработка которого предписывается в другом состоянии, но после того, как все операции в текущем будут завершены.

На рисунке 6.4 представлена диаграмма состояний.

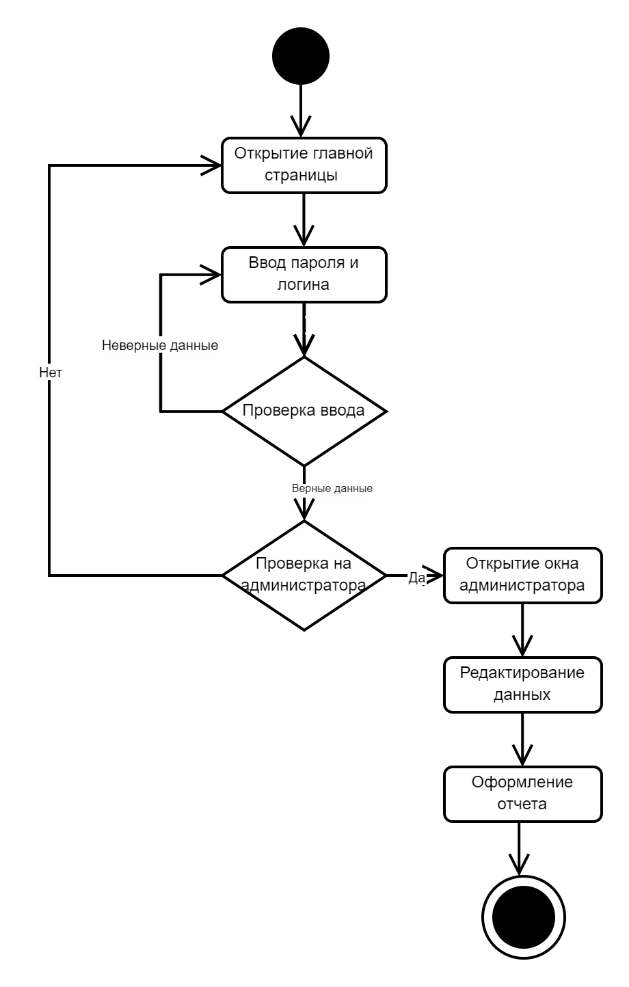


Рисунок 6.4 – Диаграмма состояний

**6.5 Диаграмма последовательности**

Диаграмма последовательности — UML-диаграмма, на которой для некоторого набора объектов на единой временной оси показан жизненный цикл объекта и взаимодействие актеров информационной системы в рамках прецедента.

Диаграммы последовательностей используются для уточнения диаграмм прецедентов, более детального описания логики сценариев использования. Это отличное средство документирования проекта с точки зрения сценариев использования.

Диаграммы последовательностей обычно содержат объекты, которые взаимодействуют в рамках сценария, сообщения, которыми они обмениваются, и возвращаемые результаты, связанные с сообщениями. Впрочем, часто возвращаемые результаты обозначают лишь в том случае, если это не очевидно из контекста.

Объекты обозначаются прямоугольниками с подчеркнутыми именами (чтобы отличить их от классов).

Сообщения (вызовы методов) - линиями со стрелками.

Возвращаемые результаты - пунктирными линиями со стрелками.

Прямоугольники на вертикальных линиях под каждым из объектов показывают “время жизни” (фокус) объектов. Впрочем, довольно часто их не изображают на диаграмме, все это зависит от индивидуального стиля проектирования.

На рисунке 6.5 представлена диаграмма последовательности.

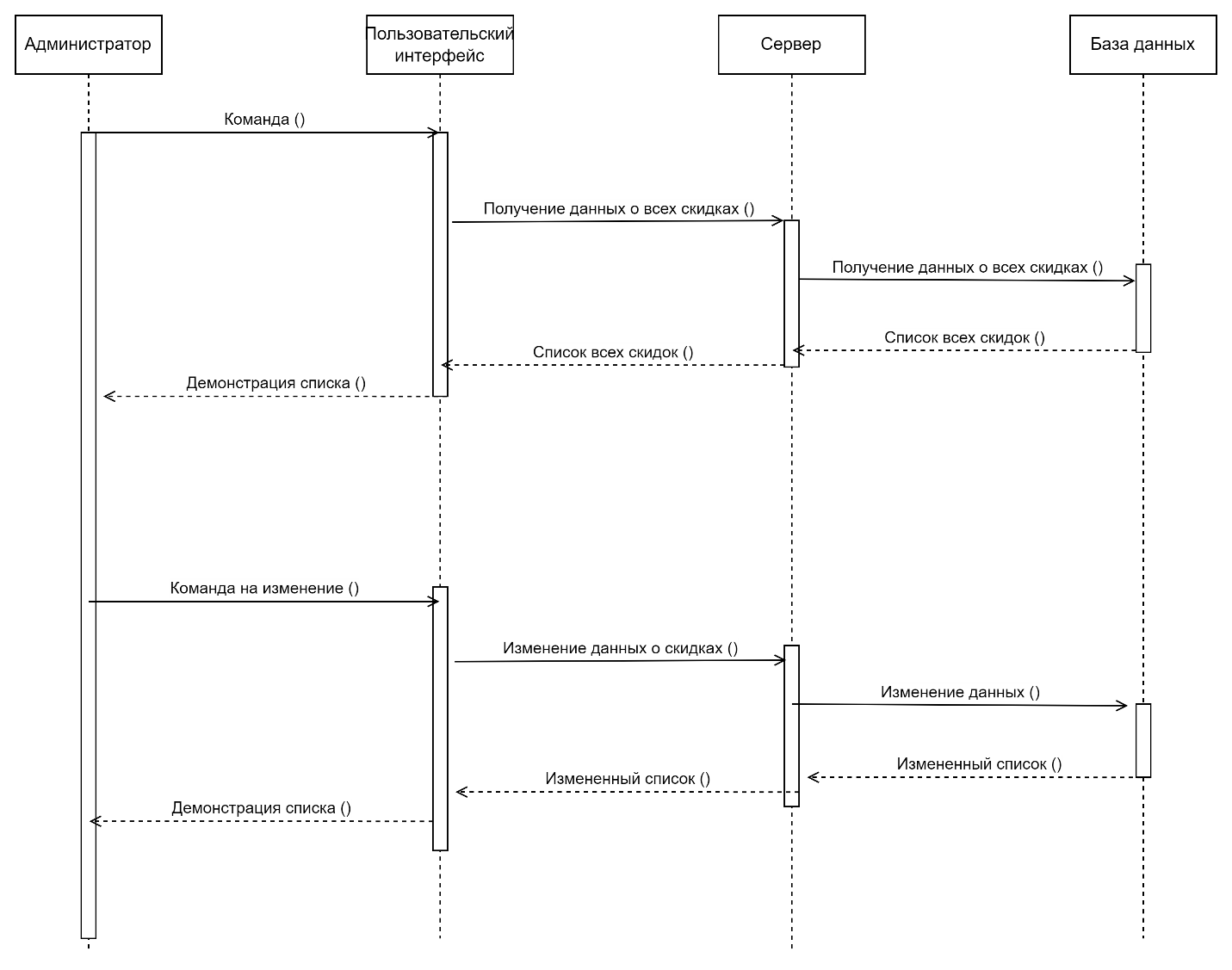


Рисунок 6.5 – Диаграмма последовательности

**6.6 Диаграмма вариантов использования**

Диаграмма вариантов использования в UML — диаграмма, отражающая отношения между акторами и прецедентами и являющаяся составной частью модели прецедентов, позволяющей описать систему на концептуальном уровне.

На диаграммах вариантов использования (ВИ) изображаются актеры и варианты использования, между которыми существуют отношения. Здесь можно показывать и другие элементы UML.

Актером будем называть внешнюю по отношению к ПС сущность, которая может взаимодействовать с системой. Актерами могут быть как люди, так и внешние системы или устройства. Следует всегда помнить, что актер — это не конкретный человек или устройство, а роль (должностная обязанность), в которой он выступает по отношению к программной системе. Например, в качестве актера «Бухгалтер» может выступать весь наличный штат бухгалтерии. В то же время один конкретный человек может играть несколько ролей по отношению к системе. Главный бухгалтер может выступать как актер с таким же именем, но может использовать систему так же, как актер «бухгалтер» (то есть, выполнять работу обычного бухгалтера). В то же время актеры — не обязательно люди. Если ПС должна выполнять какие-либо действия в определенные моменты времени, то в качестве актера, инициирующего выполнение этих действий, может быть указан системный таймер.

Нахождение актеров — один из первых шагов в определении использования любой системы (как реальной, так и программной). Каждый источник внешних событий, с которыми должна взаимодействовать система, представляется как актер. Актер должен иметь имя, которое должно отражать его роль. На диаграммах ВИ актер изображается в виде стилизованной человеческой фигурки, при этом можно использовать другие стереотипы для переопределения изображения.

При взаимодействии актера с системой последняя выполняет ряд работ, которые образуют вариант использования системы (use case). Каждый актер может использовать систему по-разному, то есть инициировать выполнение разных ВИ. Таким образом, каждый ВИ, по существу, есть некоторое функциональное требование к системе (которое может быть разбито на несколько более мелких). ВИ не представляет собой конструкцию, напрямую реализуемую в программном коде. Все его поведение реализуется в виде классов и компонент.

ВИ описывает, что делает ПС, но не как она это делает. Каждый ВИ обычно предполагает наличие нескольких вариантов поведения системы (потоков событий), один из которых является основным, остальные — альтернативными. Основной поток событий определяет последовательность действий системы, направленную на выполнение главной целевой функции данного ВИ. Альтернативные потоки описывают поведение системы в исключительных ситуациях, например, при ошибках. Описание потоков событий может быть выполнено как в текстовой форме, так и с помощью диаграмм UML, которые будут рассматриваться в дальнейшем.

Лучший путь к нахождению ВИ — рассмотреть, что требует каждый актер от системы. Следует помнить, что система существует только для пользователей и должна строиться, исходя из их потребностей.

Каждый ВИ должен иметь название, отвечающее его назначению. Название должно отражать, что достигается при взаимодействии с актерами. На диаграммах ВИ изображается в виде овала.

На диаграмме присутсвуют разлизные отношения.

Ассоциация – единственно возможная связь между актером и ВИ. Она показывает, что актер и ВИ общаются друг с другом, посылая и получая сообщения. Если ассоциация направленная, она показывает направление передачи сообщения.

Если два или более ВИ имеют сходство в структуре и поведении, то целесообразно выделить общий фрагмент и построить новый, родительский ВИ. Исходные ВИ будут являться дочерними по отношению к родительскому. Дочерний ВИ наследует все поведение, описанное в родительском варианте. Отношение обобщения между двумя ВИ означает, что когда осуществляется дочерний ВИ, необходимо исполнение и родительского. В общем случае для того, чтобы создание родительского ВИ имело смысл, необходимо, чтобы у него было бы хотя бы два дочерних. Единственное исключение – это, когда имеются два ВИ и один из них является детализацией другого, но оба могут осуществляться независимо. На диаграммах показывается в виде отношения обобщения.

Отношение включения, помечаемое стереотипом <>, означает, что для полного осуществления основного (базового) ВИ необходимо выполнение и включаемого варианта. В общем случае выделение включаемых ВИ будет целесообразным в тех случаях, когда такой вариант включается в несколько базовых. Об отношении включения “знает” только базовый вариант использования, но не включаемый. Включение показывается пунктирной стрелкой, направленной от базового ВИ к включаемому.

Если ВИ имеет фрагменты, которые по характеру являются необязательными или представляют собой исключения и при этом не способствуют лучшему пониманию основного назначения ВИ, вы можете вынести за скобки такие фрагменты, создав новый, расширяющий ВИ. Начальный вариант становится базовым, который связывается с новым вариантом отношением расширения. Расширение показывается пунктирной стрелкой, направленной к расширяемому ВИ.

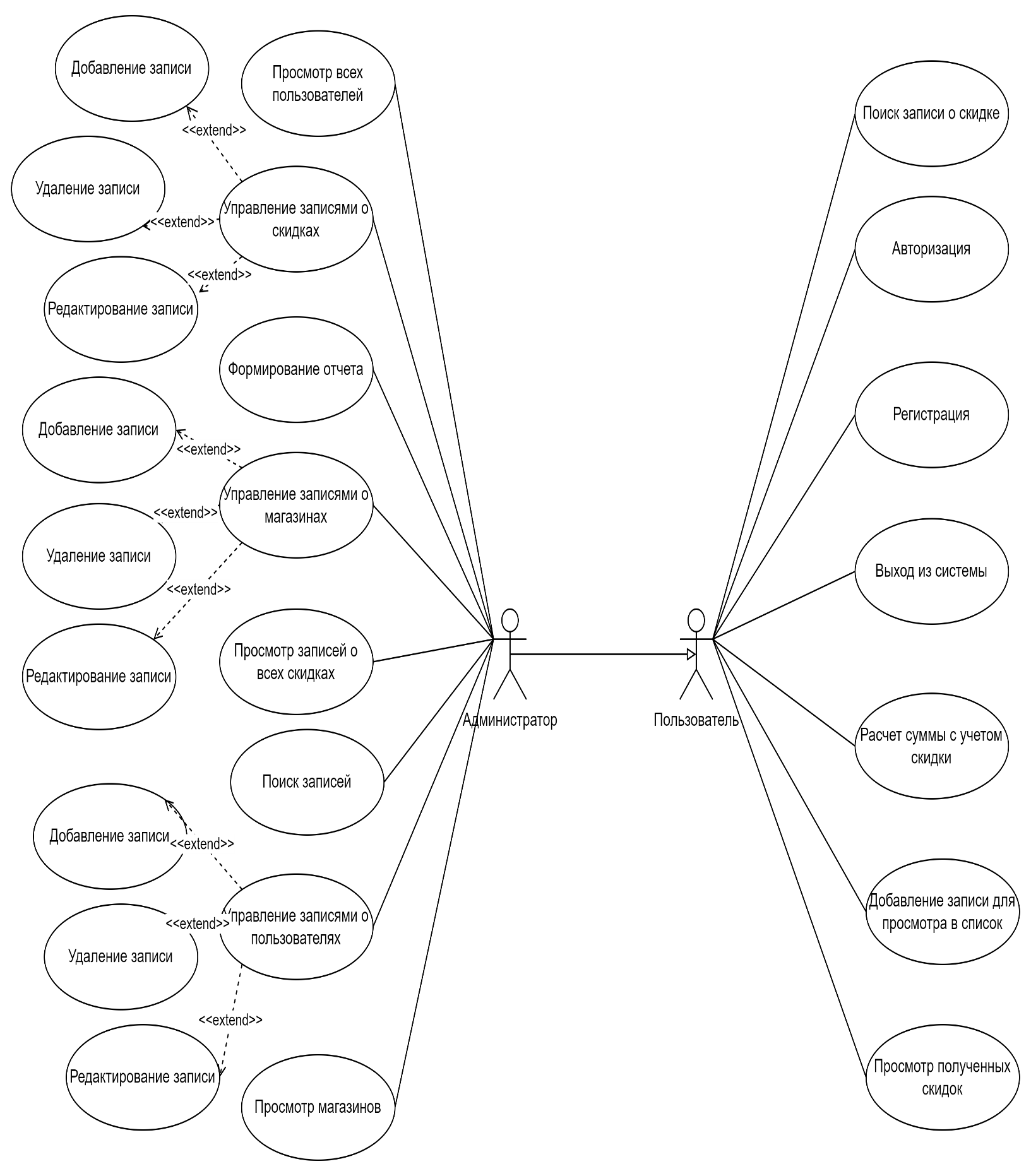


Рисунок 6.6 – Диаграмма вариантов использования

**7 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ**

Для корректного запуска системы, компьютер или рабочая станция, на которой разворачиваются клиентская и серверная части приложения должен обладать следующими минимальными требованиями

* 32х разрядная система ОС Windows 7 и выше;
* СУБД PostgreSQL;
* Java Development Environment (IntelliJ IDEA Ultimate).

Как было отмечено ранее, разработанная система поддержки управления дисконтной программой сети магазинов представляет собой клиент-серверное приложение, позволяющее одновременно работать нескольким пользователям.

Для запуска серверной части будет использован jar-файл. Это можно осуществить c запуска из директории, где располагается данный файл. Также запустить сервер или клиент возможно с помощью приложения «Командная строка».

После успешного запуска серверной части приложения появляется запись о запуске сервера (рисунок 7.1).

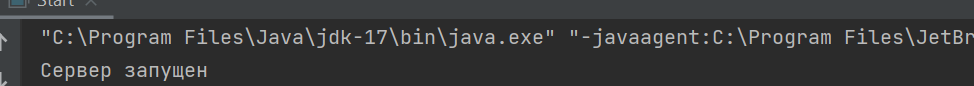


Рисунок 7.1 – Уведомление о запуске сервера

Если база данных уже была создана, то появляется запись о подключении к уже готовой базе данных, иначе таблицы базы данных создаются, о чем система уведомляет (рисунок 7.2).

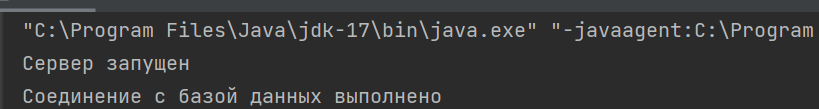


Рисунок 7.2 – Соединение с базой данных

Далее следуте запуск клиентской части приложения. Для запуска клиентской части будет использован jar-файл. Это можно осуществить c запуска из директории, где располагается данный файл. Также запустить сервер или клиент возможно с помощью приложения «Командная строка».

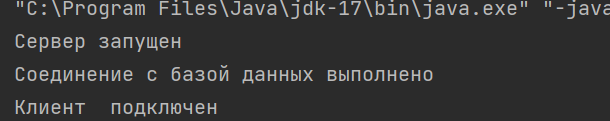
После успешного запуска клиентской части приложения и покдючения клиента к серверу появляется запись о запуске сервера (рисунок 7.3).

Рисунок 7.3 – Подключение клиента

После установки соединения появляется форма авторизации пользователя. В данном окне есть возможность зарегистрироваться. Если пользователь уже существует в системе, то для продолжения работы необходимо нажать «Войти». Для регистрации нового пользователя необходимо ввести имя, фамилию логин и пароль. При необходимости выхода из системы требуется нажать «Выйти». Форма авторизации пользователя изображена на рисунке 7.4.

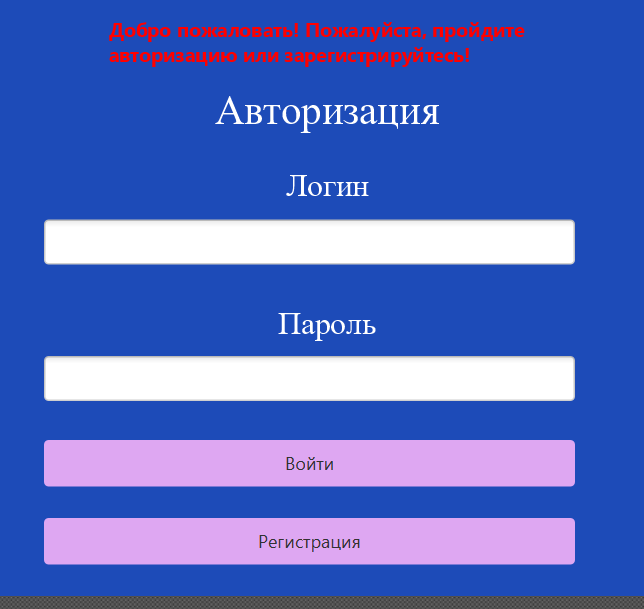


Рисунок 7.4 – Форма авторизации пользователя

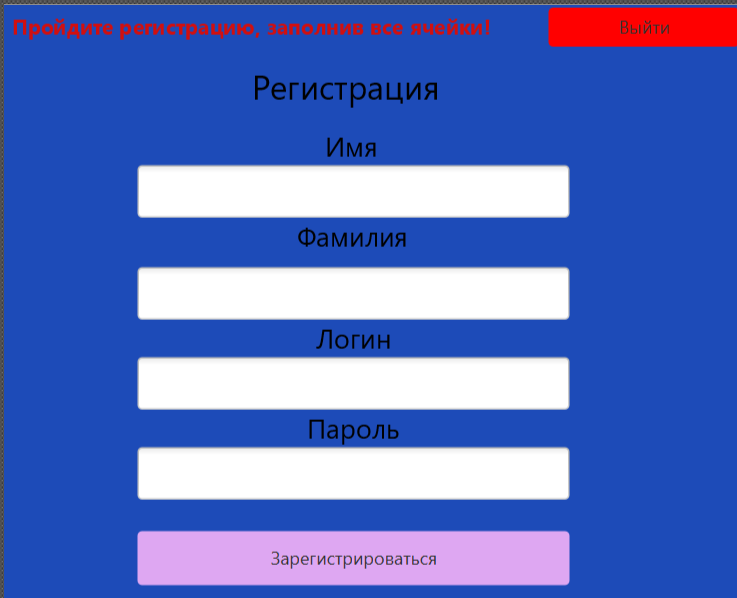
На рисунке 7.5 представлена форма для регистрации пользователя.

Рисунок 7.5 – Форма регистрации пользователя

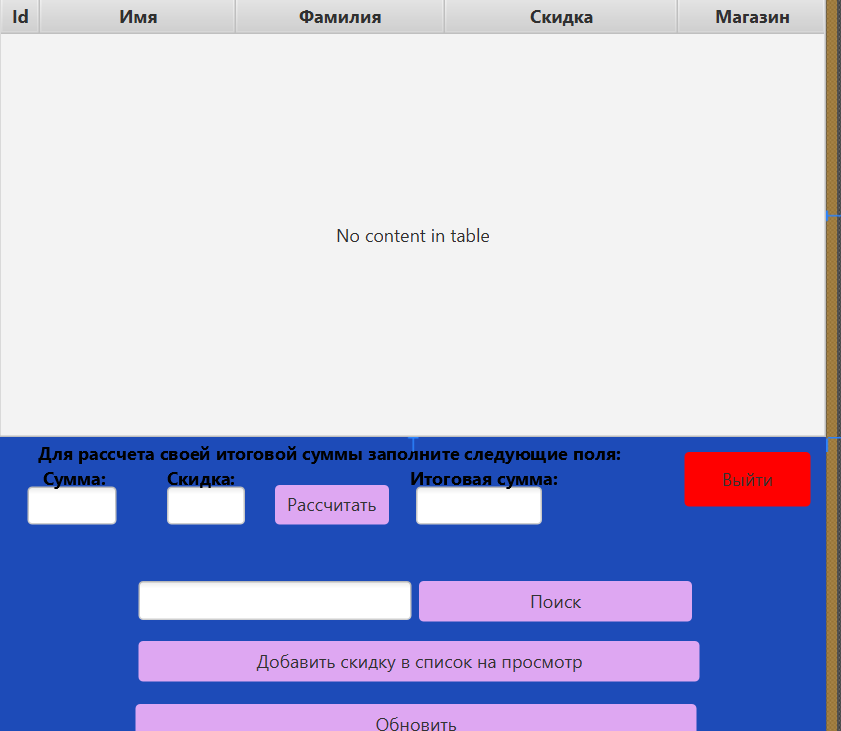
 При успешной регистрации пользователю необходимо ввести данные для авторизации в системе. После авторизации пользователя появляется окно, которое позволяет просматривать данные. Для поиска необходимой скидки нужно заполнить поле поиска и нажать кнопнку «Поиск». Форма пользователя представлена на рисунке 7.6.

Рисунок 7.6 – Форма регистрации пользователя

Для расчета полученной итоговой суммы пользователю следует заполнить ячейки и нажать кнокпку «Рассчитать». Форма для расчета отображена на рисунке 7.7.

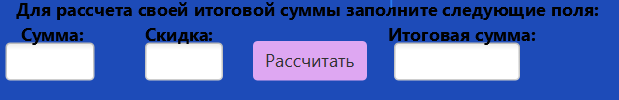


Рисунок 7.7 – Форма расчета

Если при авторизации пользователь идентифицирован как администратор, то он увидит форму с более расширенными возможностями (рисунок 7.8).

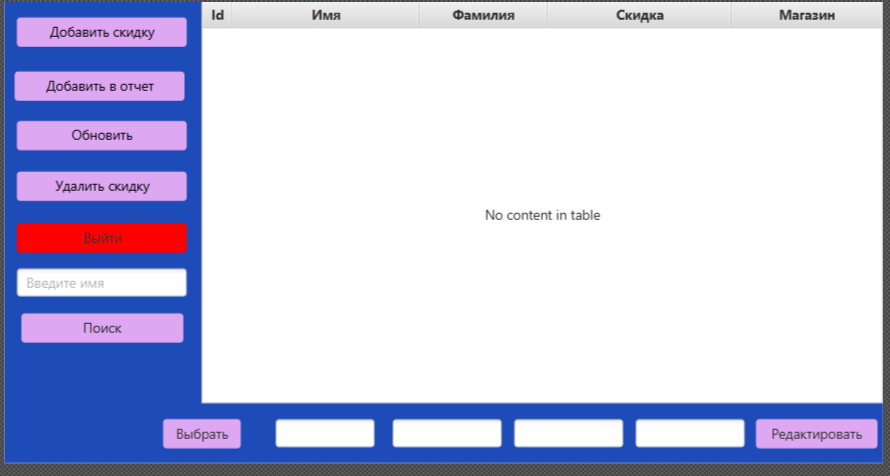


Рисунок 7.8 – Форма администратора

Администратору представлена возможность для редактирования записей. Для этого ему необходимо нажать на одну из записей в таблице, затем нажать «Выбрать». После нажатия данные из выбранной записи будт записаны в ячейки, а администратор получит возможность редактировать каждую из них. После окончания необходимых действий для сохранения результата нужно нажать на кнопку «Редактировать.

Интерфейс для этого представлен на рисунке 7.9.



Рисунок 7.9 – Редактирование данных

Также администратору представлена возможность добавления новой записи о скидках. Для этого необходимо нажать на кнопку «Добавить скидку». После нажатия появится форма для добавления новой записи. В ней представлены ячейки для заполнения информации о имени, фамили, скидке и магазине, который ее предоставляет. Чтобы сохранить новые данные нужно нажать кнопку «Добавить», и новая запись появится в таблице. Форма для добавления записи представлена на рисунке 7.10.

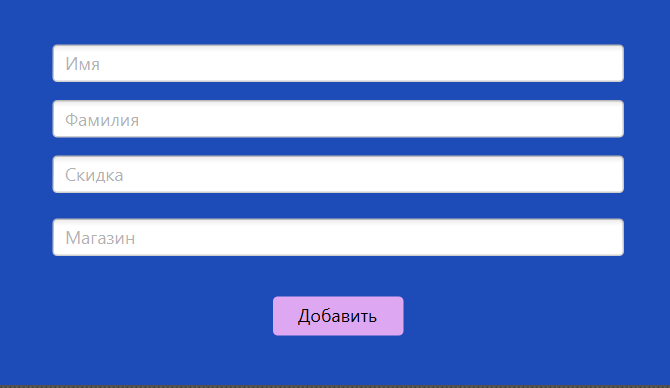


Рисунок 7.10 – Форма для добавления

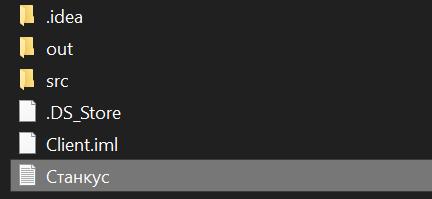
 Администратору предоставляется возможность формирования отчета. Для того, чтобы сформировать отчет необходимо выбрать нужную запись и нажать на кнопку «Добавить в отчет». После этого в папке проекта появится текстовый документ с записью из таблицы( рисунок 7.11)

Рисунок 7.11 – Формирование отчета

**8 Результаты тестирования разработанной системы**

Тестирование — это процесс оценки системы или ее компонентов с целью выяснить, удовлетворяет ли она указанным требованиям или нет. Проще говоря, тестирование — это выполнение системы с целью выявления пробелов, ошибок или отсутствующих требований, противоречащих фактическим требованиям.

В широком смысле, тестирование – это одна из техник контроля качества (Quality Control), которая включает планирование, составление тестов, непосредственно выполнение тестирования и анализ полученных результатов.

В соответствии со стандартом ANSI / IEEE 1059 тестирование можно определить как — процесс анализа элемента программного обеспечения для выявления различий между существующими и требуемыми условиями (то есть дефектами / ошибками / ошибками) и для оценки характеристик элемента программного обеспечения.

Тестирование является одним из этапов жизненного цикла программного обеспечения, направленным на улучшение качественных характеристик. В случае программного обеспечения тестирование — это процесс многократного выполнения программы для обнаружения ошибок.

Принципы тестирования:

– процесс тестирования является более эффективным, если проводится не разработчиком программы;

– описание ожидаемых значений результатов тестовых запусков должно быть необходимой частью набора тестовых данных;

– нужно внимательно изучать результаты применения каждого теста;

– испытания неправильных и непреднамеренных входных данных должны разработаны так же внимательно, как для правильных;

– необходимо проверить не только, делает ли программа то, для чего она предназначена, но и делает ли она то, что она не должна делать;

– вероятность наличия необнаруженных ошибок в части программы пропорциональна числу ошибок, уже обнаруженных в этой части.

Своевременное начало тестирования снижает затраты и время на доработку и создание безошибочного программного обеспечения, которое доставляется клиенту. Однако в жизненном цикле разработки программного обеспечения (SDLC) тестирование можно начинать с этапа сбора требований и продолжать до развертывания программного обеспечения.

Это также зависит от используемой модели развития. Например, в модели «Водопад» формальное тестирование проводится на этапе тестирования; но в инкрементальной модели тестирование выполняется в конце каждого приращения / итерации, и все приложение тестируется в конце.

Трудно определить, когда прекратить тестирование, поскольку тестирование является бесконечным процессом, и никто не может утверждать, что программное обеспечение протестировано на 100%. Следующие аспекты должны быть рассмотрены для остановки процесса тестирования:

1. Сроки тестирования;
2. Завершение выполнения тестового примера;
3. Завершение функционала и покрытие кода до определенной точки;
4. Уровень ошибок падает ниже определенного уровня, и высокоприоритетных ошибок не обнаружено;
5. Управленческое решение.

В результате тестирования курсового проекта были выявлены некоторые ситуации, которые затрудняли использование приложения.

В случае если пользователь на этапе авторизации не заполнил одно или несколько, обязательных для заполнения, текстовых полей — он получит уведомление (рисунок 8.1).

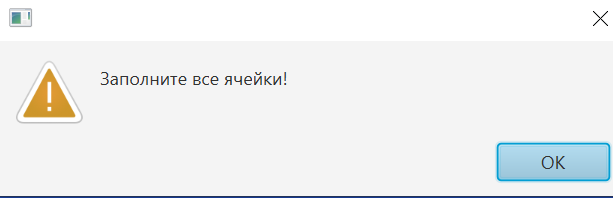


Рисунок 8.1 – Уведомление о незаполненных ячейках

При авторизации пользователя/администратора и введении не верных данных, в случае если такие данные не были обнаружены в базе данных пользователей, то будет показано уведомление (рисунок 8.2).

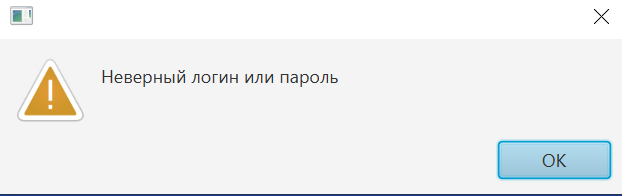


Рисунок 8.2 – Уведомление о неверных входных данных

Для удобства использования приложения на стороне серверной части приложения в случае, если были введены неверный логин иои пароль, будет выведено сообщение о том, что пользовательс представленными данными не был найден. Это отображено на рисунке 8.3.

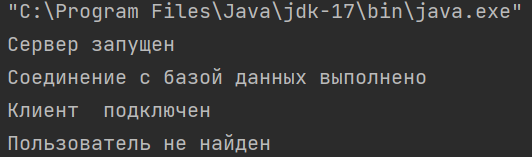


Рисунок 8.3 – Уведомление о не найденом пользователе

# **ЗАКЛЮЧЕНИЕ**

Прямой выгодой для покупателя в торговле сегодня является скидка – дисконтная программа, повышающая привлекательность магазина, от которой владелец имеет гораздо лучшие результаты, чем его конкуренты. Автоматизация системы учета дисконтной программы с помощью современного компьютерного обеспечения позволяет собирать информацию, присвоенную карте, мгновенно.

В результате разработки была создано программное средство по управлению дисконтной программы сети магазинов.

Приложение разработано на объектно-ориентированном языке программирования Java. Для создания графического пользовательского интерфейса была использована библиотека JavaFX.

Данные, с которыми работает приложение, записываются и хранятся в базе данных, которая связана с СУБД PostgreSQL с помощью менеджера баз данных JDBC.

Была достигнута основная цель проекта, а также выполнены задачи.

Целью проекта являлась повышение эффективности работы магазина, что можно достичь автоматизацией системы поддержки управления дисконтной программы сети магазинов.

Задачами проекта были:

1. Провести описание предметной области;
2. Осуществить подбор инструментария;
3. Разработать функциональную модель на основе стандарта IDEF0;
4. Составить описание информационной модели (приведение к 3-ей нормальной форме);
5. Составить руководство пользователя;
6. Автоматизировать управление информационной системой;
7. Проанализировать результаты тестирования автоматизированной системы.

Подобные программы уже давно существуют и используются, можно сказать, повсеместно. Они используются потому, что выполнять те же задачи без информационной системы стало попросту невозможно. И, хоть такие программы не являются чем-то новым и неизведанным, была совершена попытка сделать удобное и функциональное приложение, которое способно выполнить поставленные перед ним задачи.

Так же проект можно расширить, добавив новые функции. Например, добавить возможность оформить и заказать дисконтную карту онлайн. Однако программа будет постоянно зависеть от сети интернет, что само собой усложнит ее работу в случае, если сеть неожиданно пропадет.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

[1] Маркетинг и реклама [Электронный ресурс]. – Электронные данные. – Режим доступа: https://www.kom-dir.ru/article/1760-diskontnaya-programma

[2] Морозова, С.И. Применение дисконтных программ в организациях потребительской кооперации / С.И. Морозова // Молодой ученый. – 2009. – № 2. – С. 74-76

[3] Дисконтная программа [Электронный ресурс]. – Электронные данные. – Режим доступа: <http://www.belelectro-m.by/program_kinds.html>

[4] Что интереснее покупателю? [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://dinect.com/what-customers-want/>

[5] SLS-Дисконт - инструмент повышения лояльности покупателя [Электронный ресурс]. – Электронные данные. – Режим доступа: <http://www.sls.ru/produce/discount.php>

[6] Стек протоколов TCP/IP [Электронный ресурс]. – Электронные данные. – Режим доступа: http://citforum.ru/nets/ip/glava\_2.shtml

[7] Чем PostgreSQL лучше других SQL баз данных с открытым исходным кодом. Часть 1 [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://habr.com/ru/post/282764/>

[8] Java [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://ru.wikipedia.org/wiki/Java>

[9] Что такое JavaFX и как он поможет при создании приложений [Электронный ресурс]. – Электронные данные. – Режим доступа:https://highload.today/chto-takoe-javafx-i-kak-on-pomozhet-pri-sozdanii-prilozhenij/

[10] BPwin [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://www.kpms.ru/Automatization/BPwin.htm>

[11] Семантика IDEF0 [Электронный ресурс]. – Электронные данные. – Режим доступа: <http://www.kgau.ru/istiki/umk/mbp/ch06s10s03.html>

[12] Базы данных [Электронный ресурс]. – Режим доступа: file:///D:/EORD\_bazy\_dannykh.html

[13] Батин, Н.В. Проектирование баз данных / Батин Н.В., Минск: Беларусь -2007. – 56 с.

[14] Бизнес-логика [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://ru.wikipedia.org/wiki/Бизнес-логика>

[15] Общая характеристика языка UML [Электронный ресурс]. – Электронные данные. – Режим доступа: <http://www.informicus.ru/default.aspx?SECTION=6&id=73&subdivisionid=2>

**Приложение А**

**(обязательное)**

**Листинг алгоритмов, реализующих бизнес-логику**

public void chekUserInDatabase(Button button){

try {

String answer = acceptMessage.readLine();

if(!answer.equals("false")){

String[] string = answer.split(" ");

User.currentUser = new User(string[0],string[1],string[2]);

if(User.currentUser.getRoll().equals("admin")) {

NewWindowOpen.newWindowOpen.openWindow(button, "../views/adminWindow.fxml");

}else{

NewWindowOpen.newWindowOpen.openWindow(button, "../views/userWindow.fxml");

}

}else{

AlertWindow.alertWindow.alertWindow("Неверный логин или пароль");

}

} catch (IOException e) {

e.printStackTrace();

}

}

public LinkedList<Discount> allDiscounts() throws IOException {

send("вывести все скидки");

LinkedList<Discount> discounts = new LinkedList<>();

int sizeList = Integer.parseInt(acceptMessage.readLine());

for(int i=0;i<sizeList;i++){

addDiscountInList(acceptMessage.readLine(), discounts);

}

return discounts;

}

private void addDiscountInList(String string, LinkedList<Discount> discounts){

String[] product;

product = string.split(" ");

discounts.add(new Discount(Integer.parseInt(product[0]),product[1],product[2],product[3],product[4]));

}

public void redactionDiscount(int id, String name, String surname, String dnumber, String shopName) {

send("редактировать скидку");

send(id+" "+name+" "+surname+" "+dnumber+" "+shopName);

}

public void deleteDiscount(int id) {

send("удалить скидку");

send(String.valueOf(id));

}

public void addNewDiscountInDataBase(String name, String surname, String dnumber, String shopName) {

send("добавить скидку");

send(name+" "+surname+" "+dnumber+" "+shopName);

}

public void addNewUserInDataBase(String name, String lastname, String login, String password) {

send("добавить пользователя");

send(login+" "+name+" "+lastname+" "+password);

}

**Приложение Б**

**(обязательное)**

**Листинг основных элементов программы**

public DiscountHandlerRepositoryService(int countUser, Socket socket, Database database, BufferedReader messageFromServer, BufferedWriter writeMessage) {

this.countUser = countUser;

this.socket = socket;

this.messageFromServer = messageFromServer;

this.writeMessage = writeMessage;

this.database = database;

}

@Override

public void showAll() {

LinkedList<String> listCinemas = database.showAllDiscounts();

ServerConnection.usersConnected.get(countUser).sendMessage(String.valueOf(listCinemas.size()));

for(String user:listCinemas){

ServerConnection.usersConnected.get(countUser).sendMessage(user);

}

}

@Override

public void delete() {

try {

database.deleteDiscountInDataBase(Integer.parseInt(messageFromServer.readLine()));

} catch (IOException e) {

e.printStackTrace();

}

}

@Override

public void addInDataBase() {

try {

String[] line = messageFromServer.readLine().split(" ");

database.addDiscountInDataBase(line[0],(line[1]),(line[2]),line[3]);

} catch (IOException e) {

e.printStackTrace();

}

}

@Override

public void redaction() {

try {

String[] line = messageFromServer.readLine().split(" ");

database.redactionDiscountInDataBase(Integer.parseInt(line[0]),line[1],(line[2]),(line[3]),line[4]);

} catch (IOException e) {

e.printStackTrace();

}

}

**Приложение В**

**(обязательное)**

**Листинг скрипта генерации базы данных**

public class Tables {

private final Connection connection;

private final Statement statement;

public Tables(Connection connection, Statement statement) {

this.connection = connection;

this.statement = statement;

}

public void createTablesInDataBase(){

addTableUserInDateBase();

addTableProduct();

addTableAdminsInDateBase();

addTableShopDiscInDateBase();

addTableShopInDateBase();

}

private void addTableUserInDateBase(){

if(tableExists(Constants.USERS\_TABLE)) {

try {

String SQL = "CREATE TABLE "+Constants.USERS\_TABLE +

"( " +

" id SERIAL PRIMARY KEY," +

" login VARCHAR (50), " +

" name VARCHAR (50), " +

" lastName VARCHAR (50), " +

" password VARCHAR (50), " +

" roll VARCHAR (50)"+

")";

System.out.println("Таблица 1 создана!");

statement.executeUpdate(SQL);

} catch (SQLException e) {

e.printStackTrace();

}

}

}

private void addTableProduct(){

if(tableExists(Constants.DISCOUNTS\_TABLE)) {

try {

String SQL = "CREATE TABLE "+Constants.DISCOUNTS\_TABLE +

"( " +

" id SERIAL PRIMARY KEY," +

" name VARCHAR (50), " +

" surname VARCHAR (50), " +

" dnumber VARCHAR (50), " +

" shopName VARCHAR (50) " +

")";

System.out.println("Таблица 2 создана!");

statement.executeUpdate(SQL);

} catch (SQLException e) {

e.printStackTrace();

}

}

}

private void addTableShopInDateBase(){

if(tableExists(Constants.SHOP\_TABLE)) {

try {

String SQL = "CREATE TABLE "+Constants.SHOP\_TABLE +

"( " +

" id SERIAL PRIMARY KEY," +

" shopName VARCHAR (50), " +

" address VARCHAR (50) " +

")";

System.out.println("Таблица 3 создана!");

statement.executeUpdate(SQL);

} catch (SQLException e) {

e.printStackTrace();

}

}

}

private void addTableAdminsInDateBase(){

if(tableExists(Constants.ADMIN\_TABLE)) {

try {

String SQL = "CREATE TABLE "+Constants.ADMIN\_TABLE +

"( " +

" id SERIAL PRIMARY KEY," +

" login VARCHAR (50), " +

" password VARCHAR (50) " +

")";

System.out.println("Таблица 4 создана!");

statement.executeUpdate(SQL);

} catch (SQLException e) {

e.printStackTrace();

}

}

}

private void addTableShopDiscInDateBase(){

if(tableExists(Constants.SHOPDISC\_TABLE)) {

try {

String SQL = "CREATE TABLE "+Constants.SHOPDISC\_TABLE +

"( " +

" id SERIAL PRIMARY KEY," +

" shopDiscName VARCHAR (50), " +

" discList VARCHAR (50) " +

")";

System.out.println("Таблица 2 создана!");

statement.executeUpdate(SQL);

} catch (SQLException e) {

e.printStackTrace();

}

}

}

private boolean tableExists(String nameTable){

try{

DatabaseMetaData md = connection.getMetaData();

ResultSet rs = md.getTables(null, null, nameTable, null);

rs.last();

return rs.getRow() <= 0;

}catch(SQLException ignored){

}

return true;

}}