|  |  |  |  |
| --- | --- | --- | --- |
| МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ  ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  «Национальный исследовательский ядерный университет «МИФИ»  Озерский технологический институт -  филиал федерального государственного автономного образовательного учреждения высшего образования «Национальный исследовательский ядерный университет «МИФИ»  (ОТИ НИЯУ МИФИ) | | | |
| Кафедра прикладной математики | | | |
| Допускается к защите  И. о. зав. кафедрой  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ..  «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 20 \_\_\_ г. | | | |
| Курсовая работа | | | |
| по дисциплине: | Базы данных | | |
| Тема работы: | Автоматизированная система аптечного учета | | |
| Пояснительная записка | | | |
|  | |  |  |
| Руководитель | |  | Е.Н. Сёмин |
|  | |  |  |
| Н. контролер | |  | И.А. Тарасова |
|  | |  |  |
| Выполнил | |  |  |
| студент гр. 1ИВТ-31Д | |  | К.А. Шабалин |
|  | |  |  |
|  | |  |  |

20

|  |
| --- |
| МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ  ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  «Национальный исследовательский ядерный университет «МИФИ»  Озерский технологический институт -  филиал федерального государственного автономного образовательного учреждения высшего образования «Национальный исследовательский ядерный университет «МИФИ»  (ОТИ НИЯУ МИФИ) |

ЛИСТ ОЦЕНКИ КУРСОВОЙ РАБОТЫ

|  |  |
| --- | --- |
| Студент | Шабалин Кирилл Андреевич |
| Группа |  |
| Дисциплина: |  |
| Тема работы: | Автоматизированная система аптечного учета |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Компонент оценки | | | | | Максимальное  значение (баллы) | | Оценка  (баллы) |
| Оценка руководителя за работу в семестре | | | | | 40 | |  |
| Содержание пояснительной записки | | | | | 20 | |  |
| Презентация работы и защита | | | | | 20 | |  |
| Ответы на дополнительные вопросы | | | | | 20 | |  |
| Итого баллов: | | | | | 100 | |  |
| Итого оценка: | A (отл)  90-100 | B (хор)  85-89 | C (хор)  75-84 | D (хор)  70-74 | D (уд)  65-69 | E (уд)  60-64 | F (неуд)  ниже 60 |

Члены комиссии:

|  |  |  |
| --- | --- | --- |
|  |  |  |
| Фамилия И. О. |  | Подпись |
|  |  |  |
| Фамилия И. О. |  | Подпись |
|  |  |  |
| Фамилия И. О. |  | Подпись |
|  |  |  |
| Фамилия И. О. |  | Подпись |

|  |
| --- |
| МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ  ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  «Национальный исследовательский ядерный университет «МИФИ»  Озерский технологический институт -  филиал федерального государственного автономного образовательного учреждения высшего образования «Национальный исследовательский ядерный университет «МИФИ»  (ОТИ НИЯУ МИФИ) |

ЗАДАНИЕ НА КУРСОВОЕ ПРОЕКТИРОВАНИЕ

|  |  |  |
| --- | --- | --- |
| Дисциплина: |  | |
| Тема работы: | Автоматизированная система аптечного учета | |
| Дата выдачи задания: | | 25.02.2024 |
| Дата защиты работы | | 07.07.2024 |

КРАТКОЕ ОПИСАНИЕ ЗАДАНИЯ

Разработать автоматизированную систему управления аптечным учётом, включающую базу данных на PostgreSQL и клиент-серверное веб-приложение. Система должна обеспечивать контроль товарных запасов, управление персоналом, учёт клиентов, поддерживать программы лояльности и акции, а также автоматизировать процессы закупки и поставки медикаментов. Проект должен включать детальное проектирование и реализацию базы данных на PostgreSQL, серверной части на Java Spring и клиентской части с использованием JavaScript. Необходимо также разработать механизм защиты данных от несанкционированного доступа и провести тестирование созданной системы для обеспечения её надёжности и производительности.

Аннотация

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | К.А. Шабалин. Автоматизированная система аптечного учета: . ОТИ НИЯУ МИФИ, 20, 41 стр., 26 ил.  Библиография - 7 наименований.  Вложения - CD-диск с демоверсией. | | | | | |
| В курсовой работе разработана автоматизированная система аптечного учета, направленная на повышение эффективности и точности работы аптек и аптечных сетей. Проект охватывает все этапы от анализа требований и проектирования базы данных до разработки клиент-серверного приложения и его тестирования.  Предлагается использовать:  а) СУБД PostgreSQL для надежного и масштабируемого управления данными;  б) Java Spring для создания безопасного и высокопроизводительного клиент-серверного приложения;  в) среду разработки IntelliJ IDEA для удобной и эффективной разработки, отладки и тестирования кода;  г) инструменты PgAdmin или DBeaver для дополнительно удобства и возможностей при работе с базой данных, включая выполнение запросов и анализ данных.  Эти предложения позволят разработать автоматизированную систему аптечного учета, основываясь на современных принципах проектирования и безопасности. | | | | | | | | | | | |
|  |  |  |  |  | МИФИ.092131.102ПЗ | | | | | | |
|  |  |  |  |  |
| Изм. | Лист | № докум. | Подпись | Дата |
| Разраб. | | Шабалин |  |  | *Автоматизированная система аптечного учета* | |  | | | Лист | Листов |
| Пров. | | Сёмин |  |  | К |  |  | 4 | 41 |
|  | |  |  |  | ОТИ | | | | |
| Н.контр. | |  |  |  |
| Утв. | |  |  |  |

Лит.

Содержание

Введение 6

1 Предметная область 10

1.1 Общая характеристика 10

1.2 Требования к хранимым данным 14

1.3 Специфика и функции 16

1.4 Ограничение и требования к базе данных 17

1.5 Роли в системе 18

1.6 Выбор технологий для серверной части 19

1.7 Бизнес-ограничения 20

2 Проект логической структуры БД 23

3 Описание общего алгоритма работы системы 27

4 Практическая часть 29

4.1 Описание веб-приложения 29

4.2 Описание серверной части 33

5 Установка и эксплуатация 41

6 Информационная безопасность 48

7 Экспериментальная часть 51

7.1 Тестирование ограничений целостности 51

7.2 Тестирование ограничений бизнес-логики 52

Заключение 55

Список использованных источников 57

# Введение

Аптека представляет собой сложное коммерческое предприятие с уникальными характеристиками продукции. Аптеки и аптечные склады работают с широким ассортиментом медикаментов, которые поставляются в разнообразных дозировках и формах. Существует значительные различия в ценах от различных дистрибьюторов и производителей, необходимость контроля и переоценки складских остатков, и многое другое. Кроме того, фармацевтическая отрасль подлежит государственному контролю за ценообразованием и имеет специфические бизнес-модели. Все эти процессы требуют автоматизации для повышения эффективности и прозрачности работы.

Автоматизация в фармацевтической отрасли помогает компаниям оптимизировать производство и сбыт, управлять качеством продукции, наладить взаимодействие с аптечными сетями и отдельными торговыми представителями, а также избегать непрозрачных и незаконных схем на рынке. Программные решения для автоматизации фармацевтической деятельности разрабатываются с учётом специфики задач и процессов данной отрасли. К примеру, CRM-системы позволяют объединить в одном интерфейсе все необходимые инструменты управления, аналитику и базы данных по врачам и аптекам.

На уровне аптек и аптечных сетей автоматизация способствует повышению эффективности работы, скорости и качества обслуживания клиентов, управлению филиалами, организации взаимодействия сотрудников и их обучению. Комплексная автоматизация аптек позволяет эффективно управлять складскими запасами, иметь доступ к реестрам лекарственных средств, организовать бухгалтерский учёт, наладить коммуникацию с покупателями, управлять рекламными кампаниями и программами лояльности и многое другое. К этому также относится автоматизация кассовых рабочих мест в аптеках, системы самообслуживания, маркировка препаратов при собственном производстве, автоматизация учёта движения товара и многое другое.

Исходя из всех перечисленных аспектов, становится очевидной необходимость разработки автоматизированной системы аптечного учёта, которая бы охватывала все эти процессы и улучшала их эффективность.

Таким образом, целью данной курсовой работы является проектирование и реализация такой системы с разработкой базы данных и клиентского веб-приложения. Для достижения этой цели необходимо выполнить следующие задачи:

- сформировать и проанализировать требования к базе данных;

- провести концептуальное проектирование базы данных;

- осуществить физическую реализацию базы данных;

- разработать клиент-серверное приложение для взаимодействия с базой данных;

- произвести тестирование полученной системы и подготовить её описание.

Ожидаемым результатом данной курсовой работы является создание комплексной автоматизированной системы аптечного учета, которая обеспечит:

- повышение точности и оперативности учета медикаментов и других аптечных товаров;

- упрощение и ускорение процесса продаж за счет интеграции с кассовым оборудованием и системами самообслуживания;

- эффективное управление клиентскими данными и программами лояльности, что позволит улучшить сервис и увеличить продажи;

- оптимизацию процесса закупок и взаимодействия с поставщиками, что позволит снизить затраты и обеспечить наличие необходимых товаров;

- повышение прозрачности и контроль за деятельностью сотрудников и соблюдением трудового распорядка;

- обеспечение соответствия законодательным требованиям в части учета и продажи медикаментов;

- возможность быстрого формирования отчетов для анализа и принятия управленческих решений.

Таким образом, результатом выполнения курсовой работы станет разработка надежной и функциональной системы, способной значительно повысить эффективность и качество работы аптек и аптечных сетей, а также обеспечить высокий уровень обслуживания клиентов.

Основное содержание работы построено следующим образом:

- Предметная область. Описание специфики и задач, связанных с автоматизацией аптечного учета, включая ценообразование, учет товаров, программы лояльности, классификацию лекарств, работу с поставщиками, поддержку кассового оборудования и работу с рецептами.

- Проект логической структуры БД. Определение ключевых сущностей и их атрибутов, а также функциональные зависимости и нормализация данных.

- Описание общего алгоритма работы системы. Последовательное описание этапов и процессов, обеспечивающих работу системы.

- Практическая часть. Описание веб-приложения и серверной части, включая взаимодействие клиент-сервер и реализацию функций и процедур в базе данных PostgreSQL.

- Установка и эксплуатация. Инструкции по установке системы и описание её эксплуатации.

- Информационная безопасность. Механизмы защиты данных и управление доступом в системе.

- Экспериментальная часть. Тестирование ограничений целостности и бизнес-логики, а также проверка корректности работы доменных ограничений.

# 1 Предметная область

## 1.1 Общая характеристика

Комплексные программы для автоматизации аптек и аптечных сетей, как правило, состоят из отдельных модулей, каждый из них отвечает за автоматизацию тех или иных задач или процессов аптек и аптечных сетей.

### 1.1.1 Ценообразование

Существуют товары, которые продаются по разным ценовым моделям: некоторые со скидками, другие подлежат государственному регулированию и не могут продаваться по ценам выше определенного уровня или с наценкой не выше определенного процента. Также есть товары, продаваемые по схеме «производитель спонсирует скидку», где разница между розничной и экономически обоснованной ценой компенсируется поставщиком.

В зависимости от стадии жизненного цикла аптеки необходимо также управлять ценовой политикой. Если аптека только что открылась, в ней будет больше акций. Если рядом находится сильный конкурент, то аналогичная ситуация.

Таким образом, ценообразование для аптечной сети является сложной задачей, и система автоматизации аптечного бизнеса должна предоставлять возможность управления различными моделями ценообразования, предоставлять аналитику об эффективности отдельных моделей и категорий товаров, а также желательно предлагать прогнозы по прибыльности или убыточности перед переоценкой.

### 1.1.2 Учет товаров

В аптечных сетях применяются разнообразные модели учета и переучета товаров. Это включает в себя партионный учет, проведение переучета по заранее установленному расписанию, а также переучет при возникновении внешних событий, таких как изменение курса валют, внесение изменений в регуляторные нормы, или истечение срока годности товаров.

Соответственно, в системе должны быть учтены различные аспекты, такие как учет и переоценка товарных остатков, динамическое ценообразование, оперативное формирование дозаказов товаров, а также организация транспортной логистики как внутри аптечной сети, так и от поставщиков.

Эти компоненты позволяют обеспечить точное и своевременное управление товарными запасами, минимизировать потери и оптимизировать процессы снабжения, что, в свою очередь, повышает эффективность работы всей аптечной сети.

### 1.1.3 Программы лояльности и акции.

Аптеки активно переходят на использование скидочных и бонусных программ с индивидуальными условиями для каждого клиента, основанными на его предыдущих покупках, вероятных диагнозах и других данных.

Для клиента это удобно, так как он получает персонализированные предложения на товары, которые он часто или регулярно покупает. Это может включать скидки на любимые продукты, бонусные баллы за покупки определенных товаров или эксклюзивные акции.

Для аптек такие программы также выгодны. Они могут создавать акционные пакеты, которые не только привлекают клиентов, но и помогают продать товары, которые необходимо реализовать. Это позволяет аптекам более эффективно управлять своим товарным запасом и увеличивать продажи, одновременно повышая лояльность клиентов.

### 1.1.4 Классификация лекарств

Управление аптечными товарами осложняется необходимостью использования различных методов классификации, которые упрощают процесс поиска и подбора лекарств для пользователей. Одной из наиболее признанных систем является Анатомо-терапевтико-химическая (АТХ) или Anatomical Therapeutic Chemical (ATC) классификация. Эта система структурирует препараты на основании их анатомических, терапевтических и химических характеристик, что делает её незаменимой в профессиональной среде.

Однако для конечных потребителей важнее ориентировка на симптомы или заболевания. Для удовлетворения потребностей различных пользователей система автоматизации аптечного бизнеса должна поддерживать несколько систем классификации параллельно.

### 1.1.5 Работа с поставщиками

Встроенная система заказа поставщикам играет ключевую роль в эффективном управлении аптечными запасами. В настоящее время автоматизация процессов импорта и экспорта прайс-листов осуществляется с использованием кодов АТХ.

Для аптек важно иметь возможность оперативно получать прайс-листы от поставщиков и формировать заказы на основе оптимальной цены. Это позволяет не только сократить затраты, но и обеспечить наличие необходимых медикаментов в нужное время.

Примеры инструментов автоматизации заказов в фармацевтической отрасли включают LIKIS и его аналоги. Эти системы позволяют выбирать поставщиков по заданным критериям, таким как минимальная цена, что значительно упрощает процесс закупок. Автоматизация этих процессов помогает аптекам избегать человеческих ошибок, ускоряет обработку данных и повышает общую эффективность работы.

### 1.1.6 Поддержка кассового оборудования

Возможность интеграции и работы с различными кассовыми и фискальными устройствами существенно повышает эффективность и точность операций.

К таким устройствам относятся:

- сканеры штрих-кодов, которые позволяют быстро и точно идентифицировать товары, ускоряя процесс обслуживания клиентов и уменьшая вероятность ошибок при вводе данных;

- принтеры этикеток, необходимые для маркировки товаров, что упрощает их идентификацию и учет;

- регистраторы расчетных операций (РРО), которые обеспечивают корректный учет финансовых операций и соответствие требованиям законодательства;

- POS-системы (Point of Sale), которые объединяют функциональность кассовых аппаратов и компьютеров, позволяя вести учет продаж, управлять товарными запасами и анализировать данные.

Интеграция этих устройств в систему автоматизированного аптечного учета позволяет не только ускорить и упростить процесс обслуживания клиентов, но и обеспечить точность учета товаров и финансовых операций, а также соблюдение всех нормативных требований.

### 1.1.7 Работа с рецептами

Единая государственная информационная система в сфере здравоохранения (ЕГИСЗ) - это масштабный проект Министерства здравоохранения России, целью которого является интеграция всех медицинских учреждений, аптек и других организаций, связанных с предоставлением медицинских услуг, в единую информационную сеть.

Основные задачи и функции ЕГИСЗ включают централизацию данных, сбор и хранение информации о пациентах, их медицинских историях, результатах анализов, назначениях и прочих медицинских данных. Перевод медицинских документов в электронный формат облегчит доступ к информации и уменьшит вероятность потери данных. Система электронных рецептов позволяет автоматизировать процесс выдачи лекарств, контролировать их учет и предотвращать ошибки. Улучшение взаимодействия между врачами, больницами и аптеками способствует повышению качества медицинских услуг и снижению времени на обработку документов. Цели и задачи системы подробно описаны в Положении, утвержденном Постановлением Правительства РФ № 140 от 9 февраля 2022 года.

Типичный сценарий продажи препаратов по рецепту: покупатель приходит в аптеку и предоставляет рецепт, выданный врачом. Фармацевт вводит номер рецепта или сканирует его штрихкод в специализированную программу. Программа проверяет рецепт через единую систему учета рецептов, куда уже поступила информация от больницы или поликлиники, выдавшей рецепт. Система предоставляет список лекарств, которые можно выдать по данному рецепту. Фармацевт выдает необходимые лекарства, фиксируя факт их выдачи. Информация об отпущенных лекарствах обновляется в единой базе данных, чтобы избежать повторных выдач по одному и тому же рецепту.

## 1.2 Требования к хранимым данным

Для разработки эффективной системы автоматизации аптечного учета, важно определить, какие конкретные данные необходимо хранить. Ниже приведены требования к хранимым данным в системе, организованные по ключевым сущностям.

a) должности:

- уровень доступа (строка, обязательно);

- название должности (строка, обязательно.

б) каталог товаров:

- название позиции (строка, обязательно);

- дозировка позиции (строка, обязательно);

- производитель позиции (строка, обязательно);

- флаг, показывающий отпускается ли позиция по рецепту (логический, обязательно).

- идентификатор каталога (число, обязательно)

в) классификации:

- код классификации Anatomical Therapeutic chemical (строка, обязательно).

г) персоны:

- фамилия (строка, обязательно);

- имя (строка, обязательно);

- отчество (строка, обязательно);

- дата рождения (дата, обязательно);

- телефон (строка, не обязательно).

д) лекарства:

- серийный номер (строка, обязательно);

- дата изготовления (дата, обязательно);

- дата окончания срока годности (дата, обязательно);

- флаг, указывающий на наличие лекарства на складе (логический, обязательно);

- идентификатор лекарства по каталогу (число, обязательно).

е) покупатели:

- индивидуальная скидка (процент, не обязательно).

ж) поставщики:

- цена (число, обязательно);

- количество товара (число, обязательно);

- идентификатор товара по каталогу (число, обязательно);

- название поставщика (строка, обязательно).

и) сотрудники:

- серия паспорта (строка, обязательно);

- номер паспорта (строка, обязательно);

- персональный код (строка, обязательно);

- идентификатор сотрудника (число, обязательно);

- идентификатор должности (число, обязательно).

к) поставки:

- розничная цена (число, обязательно);

- дата поставки (дата, обязательно);

- наценка (процент, обязательно);

- идентификатор лекарства (число, обязательно);

- идентификатор поставщика (число, обязательно).

л) смены:

- время начала смены (дата и время, обязательно);

- время окончания смены (дата и время, не обязательно);

- идентификатор сотрудника (число, обязательно).

м) продажи:

- временная метка (дата и время, обязательно);

- сумма продажи (число, обязательно);

- идентификатор клиента (число, обязательно);

- идентификатор смены (число, обязательно).

Эти данные обеспечивают полное и корректное функционирование системы автоматизации аптечного учета, позволяя эффективно управлять запасами, продажами, сотрудниками и покупателями.

## 1.3 Специфика и функции

Рассмотрим основные функции, которые должна выполнять проектируемая система:

- контроль наличия товаров на складе, отслеживание сроков годности и автоматическое формирование заказов при достижении минимальных остатков;

- учет персонала, распределение рабочей нагрузки и контроль соблюдения трудового распорядка, а также управление уровнями доступа для обеспечения безопасности данных;

- ведение клиентской базы, поддержка программы лояльности, предоставление индивидуальных скидок и отслеживание предпочтений клиентов для увеличения их удовлетворенности и повышения продаж;

- обеспечение быстрой и точной обработки продаж, возможности фиксировать каждую транзакцию и обновлять состояние складских остатков в реальном времени, а также минимизировать ошибки при расчете стоимости и регистрации проданных товаров;

- управление процессами заказа и поставки медикаментов, создание заказов, отслеживание статуса поставок и проведение сравнительного анализа цен от разных поставщиков для оптимизации закупок;

- формирование отчетов по продажам, закупкам, остаткам на складе и другим параметрам в различных форматах и периодичностях для предоставления руководству актуальной и точной информации для стратегического планирования и оперативного управления.

## 1.4 Ограничение и требования к базе данных

В контексте разработки системы автоматизации аптечного учёта важно учитывать различные ограничения и требования к базе данных. Эти требования обеспечивают стабильность, безопасность и производительность системы. Рассмотрим их подробнее:

- система должна поддерживать интерактивное взаимодействие с пользователем и работать в диалоговом режиме, позволяя выполнять операции в реальном времени;

- время поиска и выдачи информации должно быть минимальным, чтобы не задерживать работу пользователей;

- важно настроить каскадные операции (удаление/обновление) для обеспечения целостности данных при изменении или удалении записей;

- необходимо использовать уникальные индексы для полей, которые должны содержать уникальные значения, например, для идентификаторов товаров, номеров заказов, номеров партий медикаментов и т.д.;

- использование транзакций для выполнения связанных операций, чтобы гарантировать, что-либо все операции выполнятся успешно, либо ни одна не будет выполнена (атомарность);

- использование различных ограничений (NOT NULL, CHECK), доменов и регулярных выражений для обеспечения корректности данных;

- поддержание возможности вертикальной и горизонтальной масштабируемости системы в случае роста данных и числа пользователей.

## 1.5 Роли в системе

В системе автоматизации аптеки предполагается реализация трёх ролей: кассир, менеджер и администратор. Каждая из этих ролей оправдывает определённый уровень доступа и набор функций, что позволяет разграничить ответственность и обеспечить гибкость управления аптекой.

a) кассир:

- осуществление продажи лекарственных препаратов покупателям;

- ввод информации о рецептах, предоставленных покупателями;

- проведение кассовых операций, использование POS-систем и сканеров штрих-кодов;

- формирование кассовых отчетов по итогам смены;

- регистрация новых клиентов.

б) менеджер:

- создание и мониторинг заказов на поставку лекарственных препаратов;

- просмотр и анализ прайс-листов от различных поставщиков;

- контроль за поступлением товаров на склад;

‑ ведение клиентской базы, добавление и удаление клиентов;

- управление программой лояльности и персонализированными предложениями для клиентов.

в) администратор:

- полный доступ к функционалу кассира и менеджера;

‑ регистрация новых сотрудников, внесение изменений в их учетные записи и удаление учетных записей сотрудников;

‑ управление персональными кодами и уровнями доступа.

Таким образом, основываясь на выделенных ролях и их обязанностях, можно сформировать матрицу доступа, приведенную в таблице 1.4.1.

Таблица 1.4.1 ‑ Матрица доступа

|  | **Кассир** | **Менеджер** | **Администратор** |
| --- | --- | --- | --- |
| **Информация о продажах** | В;З | - | В;З;И;У |
| **Информация о поставках** | - | В;З | В;З;И;У |
| **Информация о клиентах** | З | В;З | В;З;И;У |
| **Информация о сотрудниках** | - | - | В;З;И;У |
| В - выбор; З - запись; И - изменение; У - удаление. | | | |

## 1.6 Выбор технологий для серверной части

Для успешной реализации данного проекта было необходимо выбрать эффективные и надёжные технологии для серверной части. Внимательно рассмотрев различные варианты, было принято решение использовать PostgreSQL и Java Spring. Далее рассмотрим причины, обосновывающие этот выбор.

PostgreSQL славится своей высокой надежностью и стабильностью, что критически важно для систем, требующих высокой доступности и непрерывной работы, особенно для автоматизированного аптечного учета, где ошибки в данных могут привести к серьезным последствиям. Полная поддержка транзакционности, соответствие принципам ACID и гарантирование целостности данных позволяют избежать потерь информации или некорректных изменений. Сложные запросы, индексы, триггеры и хранимые процедуры обеспечивают эффективное управление данными, реализацию сложной логики на уровне базы данных и оптимизацию производительности системы. Возможности масштабирования как вертикально, так и горизонтально делают систему гибкой и готовой к росту нагрузки, что важно для поддержания производительности при увеличении количества данных и пользователей. Соответствие стандартам SQL и поддержка различных расширений делают PostgreSQL гибким и мощным инструментом для работы с данными, позволяющим адаптировать базу данных под специфические требования проекта.

Java с использованием фреймворка Spring известна своей надежностью и высоким уровнем безопасности, что особенно важно для систем, работающих с чувствительной информацией, такой как медицинские данные. Будучи платформенно независимой, Java обеспечивает гибкость при выборе серверов и операционных систем. Spring упрощает управление зависимостями и конфигурациями, что делает код более чистым и поддерживаемым благодаря инверсии управления и внедрению зависимостей. С широким набором модулей, таких как Spring Boot для упрощения начальной настройки и Spring Security для обеспечения безопасности, фреймворк предлагает расширяемость на любом уровне. Создание высокопроизводительных и масштабируемых приложений становится простым с Spring, что особенно важно для систем с большим количеством транзакций. Удобные инструменты для создания RESTful сервисов упрощают интеграцию с другими системами и клиентскими приложениями. Совместимость Spring с облачными платформами, такими как AWS, Google Cloud и Azure, позволяет развертывать приложения в облаке и использовать преимущества облачных технологий.

Также был выбран подход с использованием JDBCTemplate в Java для вызова процедур и функций из базы данных PostgreSQL. Это обеспечило высокую гибкость в написании запросов, а также позволило ускорить работу с данными, которые хранятся в базе, так как планировщик PostgreSQL оптимизирует запросы в функциях. Такой подход позволяет максимально эффективно использовать возможности базы данных для обработки данных и реализации бизнес-логики на уровне базы, уменьшая тем самым нагрузку на серверное приложение и улучшая общую производительность системы.

## 1.7 Бизнес-ограничения

В целях обеспечения корректности и эффективности функционирования системы автоматизации аптечного учета, необходимо учитывать определенные бизнес-ограничения. Эти ограничения способствуют соблюдению нормативно-правовых норм и оптимизации бизнес-процессов. Ниже приведены основные бизнес-ограничения:

a) запас лекарств:

- лекарства, срок годности которых истекает в течение следующего месяца, должны быть автоматически идентифицированы для приоритетной реализации;

- минимальные и максимальные уровни запасов лекарств должны быть определены для каждого наименования, чтобы избежать их недостатка или переизбытка.

б) ценообразование:

- лекарства, подлежащие государственному регулированию, не могут иметь наценку выше установленного предела;

- все акции и скидки должны быть утверждены начальством и не могут превышать определенные пороговые значения.

в) продажи и рецепты:

- лекарства, отпускаемые по рецепту, могут быть проданы только при наличии действующего рецепта, зарегистрированного в системе;

- система должна проверять количество оставшегося отпуска данного лекарства по рецепту и выдавать предупреждения при превышении.

г) поставщики и закупки:

- закупка лекарств у поставщиков должна проходить проверку на соответствие качеству и установленным стандартам;

- поставщики должны регулярно обновлять прайс-листы, и система автоматически должна выбирать оптимальные цены среди доступных предложений.

д) персонал и доступ:

- все сотрудники должны проходить регулярное обучение по работе с системой;

- только аттестованные сотрудники могут выполнять определенные операции, такие как отпуск лекарств по рецепту и проведение инвентаризации.

е) программы лояльности:

- скидки по программам лояльности не могут совокупно превышать установленный процент от суммы покупки;

- срок действия бонусных баллов должен быть ограничен и контролироваться системой.

ж) отчеты и аналитика:

- регулярные проверки и валидация данных должны осуществляться для обеспечения точности отчетов;

- руководство должно иметь доступ к ключевым аналитическим данным для принятия стратегических решений.

к) безопасность данных:

- все персональные данные клиентов и сотрудников должны безопасно храниться и обрабатываться в соответствии с законодательством;

- регулярные резервные копии базы данных должны выполняться для предотвращения потери данных.

Комплексное соблюдение перечисленных бизнес-ограничений обеспечивает высокую степень надежности, удобства и эффективности автоматизированной системы учета в аптеке.

# 2 Проект логической структуры БД

## 2.1 Определение атрибутов и функциональных зависимостей

При проектировании базы данных важно определить ключевые атрибуты предметной области. В этом разделе будут представлены атрибуты и функциональные зависимости, описывающие различные аспекты предметной области.

Определение ключевых атрибутов:

- уровень доступа;

- название должности;

- персональный код сотрудника;

- название позиции;

- дозировка позиции;

- производитель позиции;

- флаг, показывающий отпускается ли по рецепту;

- код классификации ATC;

- фамилия персоны;

- имя персоны;

- отчество персоны;

- дата рождения персоны;

- телефон персоны;

- номер паспорта сотрудника;

- серия паспорта сотрудника;

- серийный номер лекарства;

- дата изготовления лекарства;

- дата окончания срока годности лекарства;

- флаг, указывающий на наличие лекарства на складе;

- индивидуальная скидка в процентах;

- название поставщика;

- цена лекарства от поставщика;

- количество товара у поставщика;

- серийный номер лекарства;

- розничная цена лекарства при поставке;

- дата поставки;

- наценка на препарат;

- время начала смены;

- время окончания смены;

- временная метка продажи;

- сумма продажи.

Функциональные зависимости:

- название должности -> уровень доступа.

- (название позиции, дозировка, производитель) -> флаг по рецепту.

- код классификации -> категория.

- (фамилия, имя, отчество, дата рождения) -> телефон.

- серийный номер лекарства -> (дата изготовления, дата окончания срока годности, флаг наличия).

- (фамилия, имя) -> индивидуальная скидка.

- (название поставщика, серийный номер лекарства) -> (цена, количество).

- персональный код сотрудника -> (фамилия, имя, отчество, номер паспорта, серия паспорта, название должности).

- (название поставщика, серийный номер лекарства) -> (розничная цена, дата, наценка).

- (фамилия сотрудника, имя сотрудника, время начала) -> время окончания.

- временная метка продажи -> (сумма, фамилия клиента, имя клиента, фамилия сотрудника, имя сотрудника).

## 2.2 Нормализация

Для обеспечения целостности данных и избежания аномалий данных необходимо пройти этапы нормализации, приведенные ниже:

а) Первая нормальная форма (1НФ). Все данные должны быть атомарными, то есть каждое поле должно содержать только одно значение.

б) Вторая нормальная форма (2НФ). Все неключевые атрибуты должны зависеть функционально полно от первичного ключа. На этом этапе выделяются идентификаторы для каждой сущности, и появляются новые функциональные зависимости.

в) Третья нормальная форма (3НФ). Отношение находится в 3НФ, когда находится во 2НФ и каждый не ключевой атрибут нетранзитивно зависит от первичного ключа.

г) Отношение находится в НФБК, когда каждая нетривиальная и неприводимая слева функциональная зависимость обладает потенциальным ключом в качестве детерминанта.

Далее рассмотрим каждую сущность отдельно и определим, в каких нормальных формах они находятся:

а) Первая нормальная форма (1НФ):

- должности: уровень доступа, название должности;

- каталог товаров: название позиции, дозировка позиции, производитель позиции, флаг по рецепту;

- классификации: код классификации ATC;

- персоны: фамилия, имя, отчество, дата рождения, телефон;

- лекарства: серийный номер лекарства, дата изготовления, дата окончания срока годности, флаг наличия;

- покупатели: индивидуальная скидка;

- поставщики: название поставщика, цена, количество;

- сотрудники: номер паспорта, серия паспорта, персональный код сотрудника;

- поставки: розничная цена, дата, наценка;

- смены: время начала, время окончания;

- продажи: временная метка продажи, сумма;

Перечисленные сущности находятся в 1НФ, так как все их атрибуты содержат атомарные значения и в таблицах отсутствуют повторяющиеся группы данных.

Вторая нормальная форма (2НФ):

- должности: ид\_должности -> (название должности, уровень доступа);

- каталог товаров: ид\_каталога -> (название позиции, дозировка, производитель, флаг по рецепту);

- классификации: ид\_классификации -> код классификации;

- персоны: ид\_персоны -> (фамилия, имя, отчество, дата рождения, телефон);

- лекарства: ид\_лекарства -> (серийный номер лекарства, дата изготовления, дата окончания срока годности, флаг наличия, ид\_каталога);

- покупатели: ид\_клиента -> индивидуальная скидка;

- поставщики: ид\_поставщика -> (название поставщика, цена, количество, ид\_каталога);

- сотрудники: ид\_сотрудника -> (ид\_персоны, номер паспорта, серия паспорта, ид\_должности, персональный код сотрудника);

- поставки: ид\_поставки -> (ид\_поставщика, ид\_лекарства, розничная цена, дата, наценка);

- смены: ид\_смены -> (ид\_сотрудника, время начала, время окончания);

- продажи: ид\_продажи -> (временная метка продажи, сумма, ид\_клиента, ид\_смены).

Таким образом, все перечисленные сущности находятся во 2НФ, так как каждый неключевой атрибут зависит от всего первичного ключа, а не от его части.

Третья нормальная форма (3НФ). Отношение находится в 3НФ, если оно находится во 2НФ и каждый неключевой атрибут нетранзитивно зависит от первичного ключа. Все сущности находятся в 3НФ, так как у них нет транзитивных зависимостей.

Нормальная форма Бойса-Кодда (НФБК). Все сущности полностью соответствуют НФБК, так как для каждой нетривиальной и неприводимой слева функциональной зависимости левое множество является суперключом.

Диаграмма базы данных, удовлетворяющая нормальной форме Бойса-Кодда, изображена в приложении А на рисунке А.1.

# 3 Описание общего алгоритма работы системы

Общий алгоритм работы системы автоматизированного аптечного учета включает в себя последовательность шагов и процессов, обеспечивающих гладкую и эффективную работу системы. Ниже приведено описание ключевых этапов и взаимодействий между различными компонентами системы:

a) авторизация пользователей:

1) при входе в систему пользователю предлагается ввести свои учетные данные (логин и пароль);

2) система проверяет данные и определяет уровень доступа пользователя (кассир, менеджер, администратор);

3) в случае успешной авторизации пользователь получает доступ к соответствующим функциям системы.

б) управление товарными запасами:

1) менеджер вводит данные о новых товарах в систему, включая название, дозировку, производителя и другие характеристики;

2) при изменении цен на товары система позволяет менеджеру обновить информацию о ценах и провести переоценку.

в) работа с поставщиками:

1) менеджер формирует заказы на поставку, выбирая товары и указывая количество;

2) система позволяет отслеживать статус заказов и поступление товаров на склад.

г) продажа товаров:

1) кассир вводит данные о продаваемых товарах и, при необходимости, вводит информацию о рецептах;

2) если покупатель является участником программы лояльности, система автоматически применяет индивидуальные скидки;

3) каждая продажа фиксируется в базе данных, обновляется информация о складских запасах.

д) учет персонала:

1) администратор вводит данные о новых сотрудниках, включая их должности и уровни доступа;

2) система позволяет менеджеру планировать и управлять рабочими сменами сотрудников;

3) уровни доступа сотрудников регулируются в зависимости от их должностей и полномочий.

е) учет клиентов:

1) менеджер или кассир могут добавлять новых клиентов в систему;

2) система позволяет управлять бонусными программами, предоставлять индивидуальные скидки и отслеживать предпочтения клиентов.

ж) формирование отчетов:

1) система предоставляет инструменты для формирования различных отчетов (по продажам, поставкам, остаткам на складе и т.д.);

2) руководство может использовать отчеты для анализа данных и принятия стратегических решений.

# 4 Практическая часть

## 4.1 Описание веб-приложения

Взаимодействие клиента и сервера в данной курсовой работе осуществляется с помощью Spring Java. Рассмотрим основные детали этого процесса, на рисунке 4.1.1 представлена структура Spring проекта.

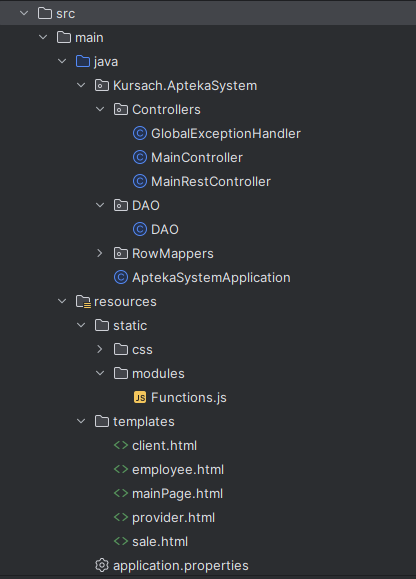


Рисунок 4.1.1 - Структура Spring Java проекта

Начнем с того, что в данном проекте есть два контроллера. «MainController» имеет аннотацию @Controller и возвращает html страницы из папки templates, когда пользователь обращается к определенным адресам, его строение показано на рисунке 4.1.2.

@Controller

public class MainController {

public MainController() {}

@GetMapping(«/mainPage»)

public String showMainPage() {

return «mainPage.html»;

}

@GetMapping(«/sale»)

public String showSale() {

return «sale.html»;

}

@GetMapping(«/provider»)

public String showProvider() {

return «provider.html»;

}

@GetMapping(«/client»)

public String showClient() {

return «client.html»;

}

@GetMapping(«/employee»)

public String showEmployee() {

return «employee.html»;

}

}

Рисунок 4.1.2 - Основной контроллер @Controller

«MainRestController» имеет аннотацию @RestController и является связующим звеном между клиентом и «DAO», которое уже непосредственно сотрудничает с базой данных. Он представляет из себя набор методов, которые «слушают» определенные адреса и получают данные, которые пользователь отправляет по ним с помощью JSON (JavaScript Object Notation). Далее они вызывают соответствующие методы «DAO» для обработки этих запросов. Один из методов представлен на рисунке 4.1.3.

@PostMapping(«/getAllRecords»)

public ResponseEntity<?> getAllRecords(@RequestBody JSONNode JSONNode) {

List<String> data = dao.getAllRecordsFromTable(JSONNode);

return new ResponseEntity<>(data, HttpStatus.OK);

}

Рисунок 4.1.3 - Метод getAllRecords класса «MainRestController»

Класс «DAO» содержит методы, которые обращаются к функциям и процедурам в базе данных. Для этого используется класс «JdbcTemplate» и его методы «queryForList», «queryForObject», «update». Для демонстрации один из методов «DAO» представлен на рисунке 4.1.4.

public List<String> getAllRecordsFromTable(JSONNode JSONNode)

String JSONString = JSONNode.toString();

String sql = «SELECT \* FROM получить\_записи(?::JSONb)»;

return jdbcTemplate.queryForList(sql, String.class, JSONString);

}

Рисунок 4.1.4 - Метод getAllRecordsFromTable класса «DAO»

Класс «GlobalExceptionHandler» предназначен для обработки ошибок, возникающих в системе. Он аннотирован с помощью @ControllerAdvice, что позволяет ему перехватывать управление программой при возникновении ошибок, обработка которых предусмотрена в одном из методов этого класса. В данном проекте обрабатываются только ошибки типа «SQLException», которые могут быть выброшены базой данных при выполнении функций или процедур. Этот класс позволяет обрабатывать как стандартные ошибки, так и пользовательские, имеющие вид «P0000» (где вместо нулей могут быть любые числа). На рисунке 4.1.5 представлен список всех ошибок, где предустановлены определённые сообщения.

static {

sqlErrorMessages.put(«23505», «Дублирование уникального значения.»);

sqlErrorMessages.put(«23503», «Нарушение ограничения целостности данных (foreign key).»);

sqlErrorMessages.put(«23502», «Нарушение ограничения NOT NULL.»);

sqlErrorMessages.put(«23514», «Нарушение ограничения проверки (check constraint).»);

sqlErrorMessages.put(«42501», «Нет прав на редактирование таблицы»);

sqlErrorMessages.put(«22004», «Не найдено записей»);

sqlErrorMessages.put(«P7772», «У поставщика нет указанного товара в таком количестве, или вы пытаетесь купить более 1000 товаров.»);

}

Рисунок 4.1.5 - Сообщения, возвращаемые сервером при возникновении ошибок базы данных

В файле «Functions.js» описаны JavaScript функции, которые создают пользовательский интерфейс, а также обращаются к упомянутому ранее классу «MainRestController» с помощью fetch запросов и JSON. Не будем останавливаться на подробном рассмотрении этих функций в рамках данной работы, на рисунке 4.1.6 приведен код функции «CalculateFullPrice», которая запрашивает с сервера актуальные цена на лекарства.

function CalculateFullPrice(table) {

const idColumnIndex = Array.from(table.querySelectorAll('th')).findIndex(th => th.getAttribute('name') === 'ид\_лекарства');

const priceColumnIndex = Array.from(table.querySelectorAll('th')).findIndex(th => th.getAttribute('name') === 'цена');

const ids = Array.from(table.querySelectorAll('tr td:nth-child(' + (idColumnIndex + 1) + ')')).map(td => parseInt(td.textContent.trim()));

const payload = { 'ид\_лекарств': ids };

fetch(«/calculateFullPrice», {

method: «POST»,

headers: {

«Content-Type»: «application/JSON»

},

body: JSON.stringify(payload)

})

.then(response => {

if (!response.ok) {

return response.JSON().then(errorBody => {

throw new Error(errorBody.error);

});

}

return response.JSON();

})

.then(responseData => {

responseData.forEach(itemStr => {

const item = JSON.parse(itemStr);

const id = item.ид\_лекарства;

const fullPrice = item.полная\_стоимость;

Array.from(table.querySelectorAll('tr')).forEach(tr => {

const idCell = tr.cells[idColumnIndex];

const priceCell = tr.cells[priceColumnIndex];

if (idCell && priceCell && parseInt(idCell.textContent.trim()) === id) {

priceCell.textContent = fullPrice;

}

});

});

})

.catch(error => {

alert(error.message);

});

}

Рисунок 4.1.6 - Функция CalculateFullPrice на языке JavaScript

## 4.2 Описание серверной части

Теперь рассмотрим, как в данной работе реализуются процедуры и функции, хранящиеся в базе данных PostgreSQL. Как видно на рисунке 4.2.1 все функции и процедуры принимают в качестве входного параметра строку «params» которая приводится к типу JSONB.

JSONB - это один из форматов хранения JSON в базе данных PostgreSQL. В отличие от простого формата JSON, JSONB (JSON Binary) сохраняет данные в бинарном виде, что позволяет выполнять быстрые операции поиска, индексирования и манипуляции данными.

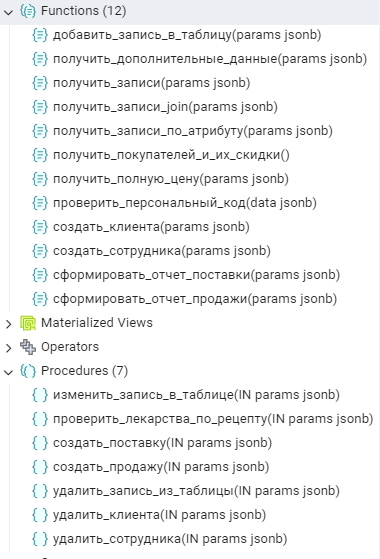


Рисунок 4.2.1 - Список всех функция и процедур в БД

На примере функции «добавить\_запись\_в\_таблицу» рассмотрим принцип работы всех функций и процедур в данной работе. Скрипт ее создания представлен на рисунке 4.2.2.

CREATE OR REPLACE FUNCTION public.»получить\_записи\_по\_атрибуту»(

params JSONb)

RETURNS SETOF JSONb

LANGUAGE 'plpgsql'

COST 100

VOLATILE PARALLEL UNSAFE

ROWS 1000

AS $BODY$

DECLARE

table\_name text;

attributes JSONb;

query text;

conditions text := '';

record JSONb;

key text;

value text;

BEGIN

-- Извлекаем имя таблицы и атрибуты из параметров

table\_name := params->>'tablename';

attributes := params->'attributes';

-- Проверяем, пустые ли атрибуты

IF attributes IS NULL OR attributes = '{}' THEN

-- Если атрибуты пустые, то составляем запрос без условий

query := 'SELECT row\_to\_JSON(t) FROM ' || quote\_ident(table\_name) || ' t';

ELSE

-- Если атрибуты не пустые, составляем условия

FOR key, value IN SELECT \* FROM JSONb\_each\_text(attributes) LOOP

IF conditions != '' THEN

conditions := conditions || ' AND ';

END IF;

conditions := conditions || format('%I = %L', key, value);

END LOOP;

-- Далее составляем запрос с условиями

query := 'SELECT row\_to\_JSON(t) FROM ' || quote\_ident(table\_name) || ' t WHERE ' || conditions;

END IF;

FOR record IN EXECUTE query LOOP

RETURN NEXT record;

END LOOP;

RETURN;

END

$BODY$;

ALTER FUNCTION public.»получить\_записи\_по\_атрибуту»(JSONb)

OWNER TO postgres;

Рисунок 4.2.2 - Функция «получить\_записи\_по\_атрибуту»

В данной работе для создания процедур и функций используется PL/pgSQL (Procedural Language/PostgreSQL Structured Query Language) - это процессуальный язык, который разработан для PostgreSQL. Он позволяет писать более сложные и мощные запросы, чем стандартный SQL, добавляя возможности процедурного программирования, такие как циклы, условия и переменные.

Пример функции «получить\_записи\_по\_атрибуту» демонстрирует, как с помощью PL/pgSQL можно реализовать динамическое построение запросов и их выполнение. Рассмотрим основные этапы работы функции:

а) функция принимает JSONB-объект params, из которого извлекает имя таблицы (table\_name) и атрибуты (attributes). Это позволяет передавать данные в функции в структурированном виде и пользоваться преимуществами JSONB формата.

б) если атрибуты пустые, функция составляет простой запрос без условий. В противном случае, с помощью цикла перебираются все ключи и значения в JSONB-объекте attributes, и составляются условия для SQL-запроса.

в) в зависимости от наличия условий, создается соответствующий SQL-запрос. Запрос выполняется с помощью команды EXECUTE, а результаты возвращаются в виде множества записей (SETOF JSONb).

г) результаты выполнения запроса возвращаются по одной записи с помощью команды RETURN NEXT.

Рассмотрим ситуацию, когда используется функция «получить\_записи\_по\_атрибуту». На входе она получает примерно такой JSON, как представлен на рисунке 4.2.3.

{«tablename»:»лекарство»,»attributes»:{«в\_наличии»:»true»}}

Рисунок 4.2.3 - JSON строка, приходящая в функцию в качестве входного параметра

Если добавить вывод отладочных сообщений в данную функцию, то можно увидеть ситуацию, показанную на рисунке 4.2.4.

ЗАМЕЧАНИЕ: Имя таблицы: лекарство

ЗАМЕЧАНИЕ: Атрибуты: {«в\_наличии»: «true»}

ЗАМЕЧАНИЕ: Условия запроса: «в\_наличии» = 'true'

ЗАМЕЧАНИЕ: Запрос с условиями: SELECT row\_to\_JSON(t) FROM «лекарство» t WHERE «в\_наличии» = 'true'

ЗАМЕЧАНИЕ: Полученная запись: {«в\_наличии»: true, «ид\_каталога»: 22, «ид\_лекарства»: 714, «дата\_окончания»: «2025-05-27», «серийный\_номер»: «6bfebDF», «дата\_изготовления»: «2024-05-27»}

...

// Запрос повторяется в цикле для всех записей

Рисунок 4.2.4 - Отладочные сообщения при выполнении функции

Результат выполнения функции «получить\_записи\_по\_атрибуту» с входными данными, рассмотренными ранее, показан на рисунке 4.2.5.

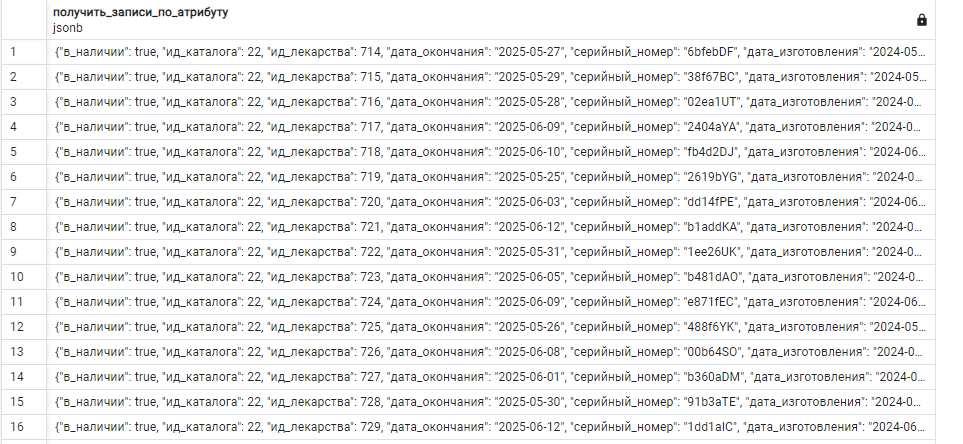


Рисунок 4.2.5 - Результат выполнения функции

Часто возникаю ситуации, когда необходимо выкинуть ошибку. Для этого используется конструкция, представленная на рисунке 4.2.6. Можно выкинуть как ошибку с сообщением, куда вставить какие-то переменные значения, так и просто ошибку с пользовательским кодом, которая далее обработается в классе «GlobalExceptionHandler».

IF v\_quantity > v\_provider\_quantity OR v\_quantity > 1000 THEN

RAISE EXCEPTION USING ERRCODE = 'P7772';

END IF;

Рисунок 4.2.6 - Пример обработки ошибки

Основной пласт ошибок, связанных с базой данных обрабатывается на уровне функций и процедур. Как показано на рисунке 4.2.7, в данном фрагменте кода происходит проверка различных ситуаций, которые могут вызвать нарушение целостности базы. В таком случае функция выкидывает исключение, которое позже обрабатывается в классе «GlobalExceptionHandler».

BEGIN

-- Получение ид\_клиента из JSON объекта

client\_id := params->>'ид\_клиента';

-- Проверка существует ли клиент с данным ид\_клиента

SELECT ид\_персоны INTO person\_id

FROM public.покупатель

WHERE ид\_клиента = client\_id;

IF NOT FOUND THEN

RAISE EXCEPTION 'Клиент с ид\_клиента % не найден', client\_id;

END IF;

-- Проверка не является ли персона сотрудником

SELECT COUNT(\*) INTO person\_count

FROM public.сотрудник

WHERE ид\_персоны = person\_id;

-- Удаление клиента

DELETE FROM public.покупатель

WHERE ид\_клиента = client\_id;

-- Если персона не является сотрудником, удаляем её

IF person\_count = 0 THEN

DELETE FROM public.персона

WHERE ид\_персоны = person\_id;

END IF;

RAISE NOTICE 'Клиент с ид\_клиента % успешно удален', client\_id;

IF person\_count = 0 THEN

RAISE NOTICE 'Персона с ид\_персоны % также удалена', person\_id;

ELSE

RAISE NOTICE 'Персона с ид\_персоны % не удалена, так как является сотрудником', person\_id;

END IF;

END;

Рисунок 4.2.7 - Фрагмент кода, обрабатывающий исключительные случаи внутри процедур или функций PostgreSQL

Далее приведено краткое описание всех функций и процедур, что они принимают и возвращают, а также ключевые особенности их реализации:

а) процедура: «удалить\_запись\_из\_таблицы»:

- принимает: JSON объект с параметрами, включающими имя таблицы и пары ключ-значение для идентификации записей для удаления;

- возвращает: ничего (это процедура);

- ключевые особенности: динамическое формирование sql-запроса для удаления записей из указанной таблицы.

б) процедура: «удалить\_клиента»:

- принимает: JSON объект с параметром «ид\_клиента»;

- возвращает: ничего (это процедура);

- ключевые особенности: удаляет клиента и связанную персону, если она не является сотрудником.

в) процедура: «удалить\_сотрудника»:

- принимает: JSON объект с параметром «ид\_сотрудника»;

- возвращает: ничего (это процедура);

- ключевые особенности: проверяет права доступа (не удаляет администратора) и удаляет сотрудника и связанную персону, если она не связана с другими записями.

г) функция: «получить\_покупателей\_и\_их\_скидки»:

- принимает: ничего;

- возвращает: JSON объекты с информацией о покупателях и их скидках;

- ключевые особенности: формирует динамический sql-запрос для получения данных о покупателях и их скидках.

д) функция: «получить\_полную\_цену»:

- принимает: JSON объект с массивом «ид\_лекарств»;

- возвращает: JSON объекты с полной стоимостью лекарств;

- ключевые особенности: извлекает розничную цену для каждого «ид\_лекарства» и возвращает JSON с полной стоимостью.

е) процедура: «проверить\_лекарства\_по\_рецепту»:

- принимает: JSON объект с массивами «productids» и «recipeids»;

- возвращает: ничего (это процедура);

- ключевые особенности: проверяет, что все продукты разрешены к продаже по данным рецептам, выбрасывает исключения при нарушениях.

ж) функция: «проверить\_персональный\_код»:

- принимает: JSON объект с «employeeid» и «personalcode»;

- возвращает: JSON объект с уровнем доступа;

- ключевые особенности: проверяет персональный код сотрудника и извлекает их уровень доступа.

и) функция: «создать\_клиента»:

- принимает: JSON объект с параметрами для создания клиента;

- возвращает: JSON объект с информацией о новом клиенте;

- ключевые особенности: создает новую запись в таблицах «персона» и «покупатель», возвращает информацию о созданном клиенте.

к) процедура: «создать\_поставку»:

- принимает: JSON объект с параметрами для создания поставки;

- возвращает: ничего (это процедура);

- ключевые особенности: проверяет доступное количество у поставщика и вставляет записи в таблицы «лекарство» и «поставка».

л) процедура: «создать\_продажу»:

- принимает: JSON объект с параметрами для создания продажи («время», «ид\_смены», «ид\_персоны»);

- возвращает: ничего (это процедура);

- ключевые особенности: создает запись в таблице «продажа», вставляет записи в таблице «продажа\_лекарство», обновляет наличие лекарства и вычисляет сумму продажи с учетом скидки.

м) функция: «создать\_сотрудника»:

- принимает: JSON объект с параметрами для создания сотрудника;

- возвращает: JSON объект с информацией о новом сотруднике;

- ключевые особенности: создает новую запись в таблицах «персона» и «сотрудник», возвращает информацию о созданном сотруднике.

н) функция: «сформировать\_отчет\_поставки»:

- принимает: JSON объект с «начальная\_дата» и «конечная\_дата»;

- возвращает: набор JSON объектов с отчетом по поставке за указанный период;

- ключевые особенности: формирует отчет о расходах на поставки, группируя данные по поставщикам и каталогам.

п) функция: «сформировать\_отчет\_продажи»:

- принимает: JSON объект с «начальная\_дата» и «конечная\_дата»;

- возвращает: набор JSON объектов с отчетом по продажам за указанный период;

- ключевые особенности: формирует отчет о продажах, включая информацию о времени, сумме, персонах и лекарствах.

# 5 Экспериментальная часть

## 5.1 Тестирование ограничений целостности

Тест для таблицы «каталог» (Рисунок 5.1.1).

Ограничения:

- проверка уникальности составного ключа (название, производитель, дозировка).

Желаемый результат:

- запись не добавлена, возникает ошибка уникальности.

Входные параметры:

- название (уже существующей записи): «Симвастатин»;

- производитель (уже существующей записи): «АстраЗенка»;

- дозировка (уже существующей записи): «250 mg».

Выходные параметры:

- ошибка уникальности для комбинации названия, производителя и дозировки.

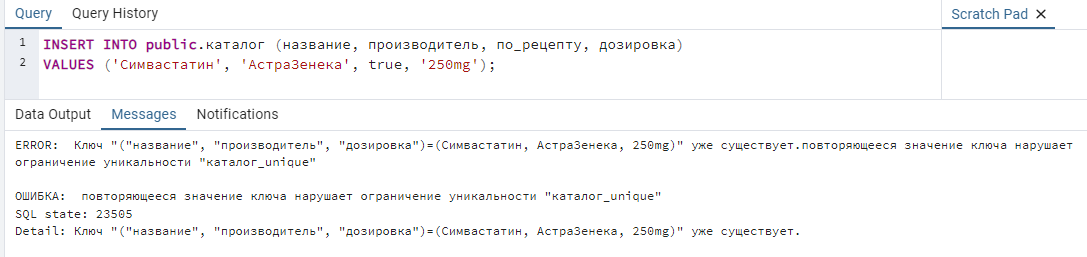


Рисунок 5.1.1 - Ограничение уникальности в таблице «каталог». Название, производитель и дозировка должны быть уникальны для каждой позиции.

Тест для таблицы «лекарство» (Рисунок 5.1.2).

Ограничения:

- проверка уникальности составного ключа (серийный номер, флаг наличия).

Желаемый результат:

- запись не добавлена, возникает ошибка уникальности.

Входные параметры:

- серийный номер (уже существующей записи): «77617LF»;

- дата изготовления: «2024-05-22»;

- дата окончания: «2025-05-22»;

- идентификатор по каталогу: «22»;

- флаг наличия (уже существующей записи): «true».

Выходные параметры:

- ошибка уникальности для комбинации серийного номера и флага наличия лекарства.

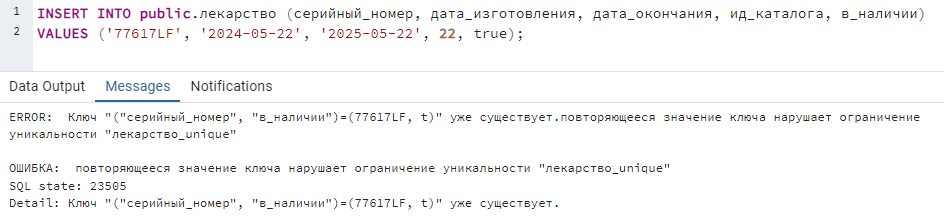


Рисунок 5.1.2 - Ограничение уникальности в таблице «лекарство». В наличии не может быть несколько лекарств с одинаковым серийным номером.

Тест для таблицы «персона» (Рисунок 5.1.3).

Ограничения:

- проверка уникальности составного ключа (ФИО, дата рождения, телефон).

Желаемый результат:

- запись не добавлена, возникает ошибка уникальности.

Входные параметры:

- фамилия (уже существующей записи): «Иванов»;

- имя (уже существующей записи): «Иван»;

- отчество (уже существующей записи): «Иванович»;

- дата рождения (уже существующей записи): «1980-01-01»;

- телефон (уже существующей записи): «+79991234567».

Выходные параметры:

- ошибка уникальности для комбинации ФИО, даты рождения и телефона

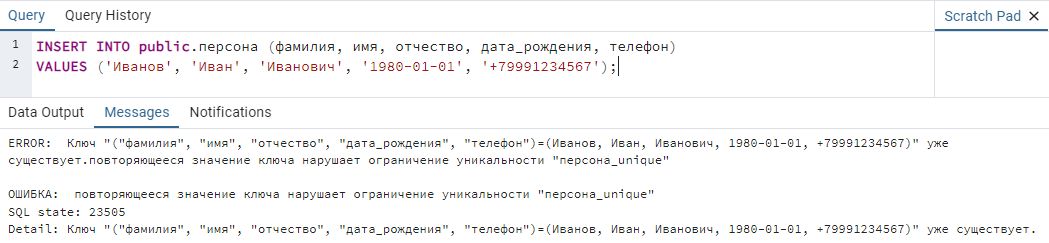


Рисунок 5.1.3 - Ограничение уникальности в таблице «персона». В системе не может быть несколько персон с одинаковыми ФИО, датой рождения и телефоном.

На рисунках 5.1.4, 5.1.5, 5.1.6 демонстрируется ограничение внешнего ключа для некоторых таблиц.

Тест для таблицы «каталог» (Рисунок 5.1.4).

Ограничения:

- проверка внешнего ключа на удаление записи из таблицы «каталог», когда на нее ссылается запись в таблице «лекарство».

Желаемый результат:

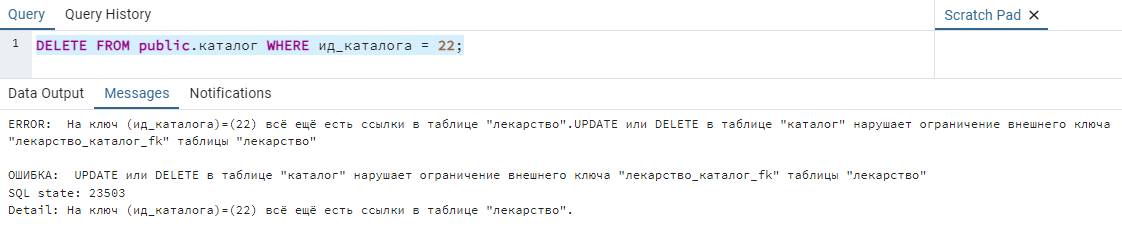
- запись не удалена, возникает ошибка нарушения внешнего ключа.

Входные параметры:

- идентификатор записи в таблице «каталог», на которую ссылаются записи в таблице «лекарство».

Выходные параметры:

- ошибка удаления записи на которую есть ссылка.

Рисунок 5.1.4 - Ограничение внешнего ключа в таблице «каталог». Нельзя удалить запись из каталога, пока на нее есть ссылка в таблице «лекарство».

Тест для таблицы «каталог» (Рисунок 5.1.4).

Ограничения:

- проверка внешнего ключа на добавление записи в таблицу «поставка» со ссылкой на несуществующее лекарство.

Желаемый результат:

- запись не добавлена, возникает ошибка нарушения внешнего ключа.

Входные параметры:

- идентификатор лекарства, которого нет в таблице «лекарство».

Выходные параметры:

- ошибка добавления записи: «Ошибка: Нельзя добавить запись, ссылающуюся на несуществующее лекарство»

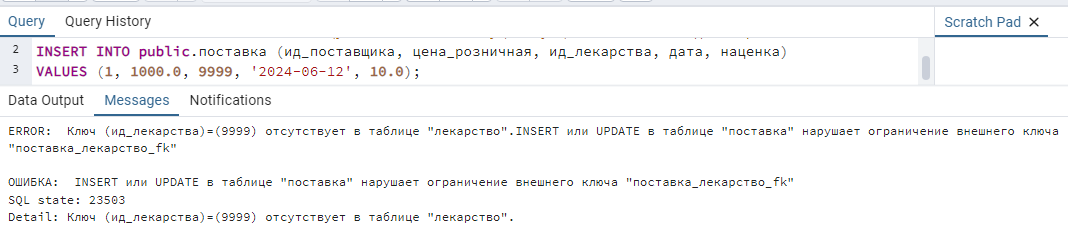


Рисунок 5.1.5 - Ограничение внешнего ключа в таблице «поставка». В данной таблице не может быть ссылки на несуществующее лекарство.

Тест для таблицы «сотрудник» (Рисунок 5.1.6).

Ограничение:

- проверка внешнего ключа на добавление записи в таблицу «сотрудник» со ссылкой на несуществующую персону.

Желаемый результат:

- запись не добавлена, возникает ошибка нарушения внешнего ключа.

Входные параметры:

- идентификатор персоны, которой нет в таблице «персона».

Выходные параметры:

- ошибка добавления записи ссылающуюся на несуществующий ключ.

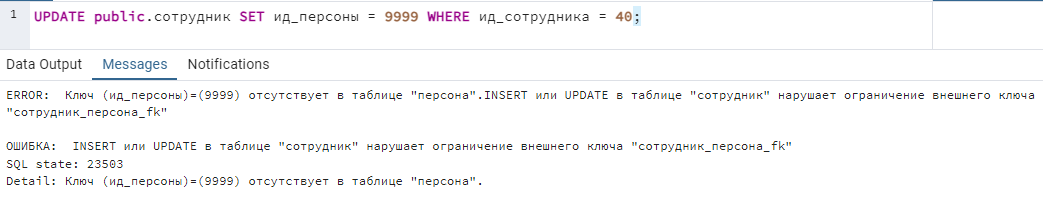


Рисунок 5.1.6 - Ограничение внешнего ключа в таблице «сотрудник». В данной таблице не может быть ссылки на несуществующую персону.

## 5.2 Тестирование ограничений бизнес-логики

Рассмотрим случай, когда необходимо удалить сотрудника из системы. В случае, если данный сотрудник не является клиентом системы, то тогда запись об этом сотруднике должна удалиться только из таблицы «сотрудник», но при этом остаться в таблицах «персона» и «покупатель». Это справедливо и для удаления клиента.

Тест для удаления сотрудника, который не является клиентом (Рисунок 5.2.1).

Ограничения:

‑ проверка удаления записи из таблицы «сотрудник» при условии, что сотрудник не является клиентом системы, и записи остаются в таблицах «персона» и «покупатель».

Желаемый результат:

- запись успешно удалена из таблицы «сотрудник», записи остаются в таблицах «персона» и «покупатель».

Входные параметры:

- идентификатор сотрудника, который не является клиентом.

Выходные параметры:

- записи в таблице «сотрудник» с заданным идентификатором не существует;

- записи с заданным идентификатором остаются в таблицах «персона» и «покупатель».

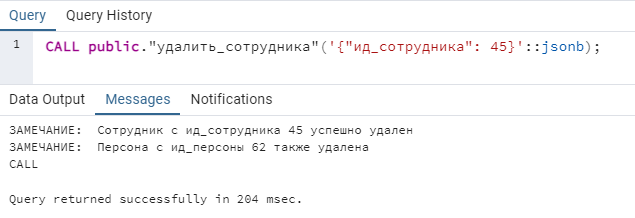


Рисунок 5.2.1 - Удаление сотрудника, который не является клиентом

Тест для удаления сотрудника, который является клиентом (Рисунок 7.2.2)

Ограничения:

- проверка удаления записи из таблицы «сотрудник» при условии, что сотрудник также является клиентом системы, и записи остаются в таблицах «персона» и «покупатель».

Желаемый результат:

- запись успешно удалена из таблицы «сотрудник», записи остаются в таблицах «персона» и «покупатель».

Входные параметры:

- идентификатор сотрудника, который также является клиентом.

Выходные параметры:

- запись в таблице «сотрудник» с заданным идентификатором отсутствует;

- запись с заданным идентификатором остается в таблице «персона».;

- запись с соответствующим идентификатором остается в таблице «покупатель».

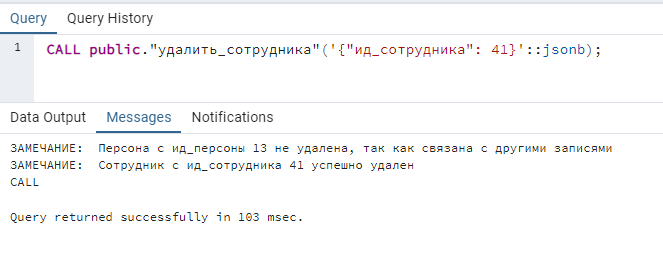


Рисунок 5.2.2 - Удаление сотрудника, который является клиентом

В данной работе использовались несколько ограничений доменов. Как показано на рисунке 5.2.3 они препятствуют вводу данных, которые могут нарушить целостность БД.

Тест для ограничений доменов:

Ограничения:

- проверка того, что в таблице «продажа» сумма является положительным числом с плавающей точкой.

Желаемый результат:

- запись отклонена, если значение не является положительным числом с плавающей точкой.

Входные параметры:

- запись в таблице «продажи» содержащая отрицательное значение в поле «сумма».

Выходные параметры:

- ошибка ограничения домена «pos\_float».

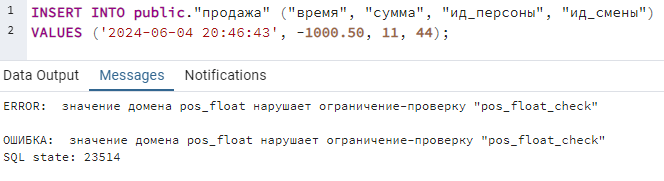


Рисунок 5.2.3 - Ограничение-проверка домена «pos\_float»

# 6 Информационная безопасность

PostgreSQL так же предоставляет функционал создания пользователей и выдачи привилегий на уровне базы данных. Если выполнить скрипт, приведённый на рисунке 6.1, можно создать пользователя, в данном случае кассира, и дать ему доступ к каким-то конкретным операциям над какими-то конкретными таблицами, так же можно ограничить возможность вызова функций и процедур для данного пользователя.

CREATE USER cashier;

ALTER USER cashier WITH PASSWORD 'password';

GRANT EXECUTE ON ALL FUNCTIONS IN SCHEMA public TO cashier;

GRANT EXECUTE ON ALL PROCEDURES IN SCHEMA public TO cashier;

REVOKE ALL PRIVILEGES ON ALL TABLES IN SCHEMA public FROM cashier;

GRANT SELECT, INSERT, UPDATE, DELETE ON TABLE public.«продажа» TO cashier;

GRANT SELECT, INSERT, UPDATE, DELETE ON TABLE public.«продажа\_лекарство» TO cashier;

GRANT SELECT, INSERT, UPDATE, DELETE ON TABLE public.«лекарство» TO cashier;

GRANT SELECT, INSERT, UPDATE, DELETE ON TABLE public.«смена» TO cashier;

GRANT SELECT ON TABLE public.«каталог» TO cashier;

GRANT SELECT ON TABLE public.«каталог\_классификация» TO cashier;

GRANT SELECT ON TABLE public.«классификация» TO cashier;

Рисунок 6.1 - Создание пользователя «cashier» и выдача ему «грантов» на операции над таблицами

С помощью этого скрипта можно определить, какие «гранты» есть у пользователя выбранного, смотрите рисунок 6.2.

SELECT

grantee,

table\_schema,

table\_name,

privilege\_type

FROM

information\_schema.role\_table\_grants

WHERE

grantee = 'cashier';

Рисунок 6.2 - Скрипт, позволяющей увидеть все «гранты» указанного пользователя

Результат выполнения запроса приведен на рисунке 6.3.

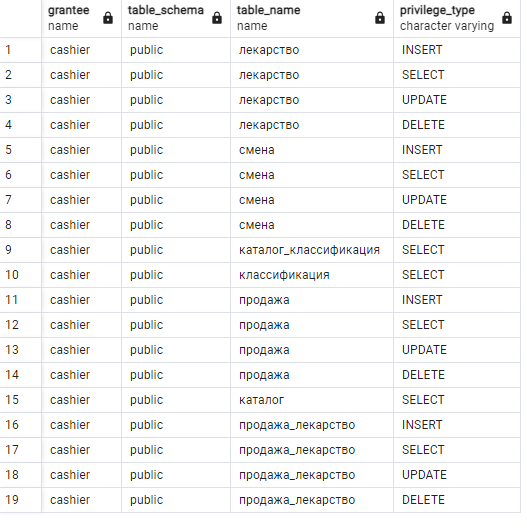


Рисунок 6.3 - Все «гранты» пользователя «cashier»

Данный функционал PostgreSQL не особо гибкий, так как в конфигурации Spring при запуске сервера мы можем указать только какого-то одного пользователя, под которым будут выполняться все операции. Как правило указывается супер-пользователь «postgres». Поэтому в данной работе предусмотрен базовый механизм авторизации, который с помощью персональных кодов сотрудников и уровней доступа регулирует доступ сотрудников к тому или иному функционалу системы.

Процесс авторизации пользователя происходит в момент, когда он пытается зайти на какую-либо страницу в системе, не считая главной. Как только он перейдет на страницу, то увидит модальное окно, представленное на рисунке 6.4.

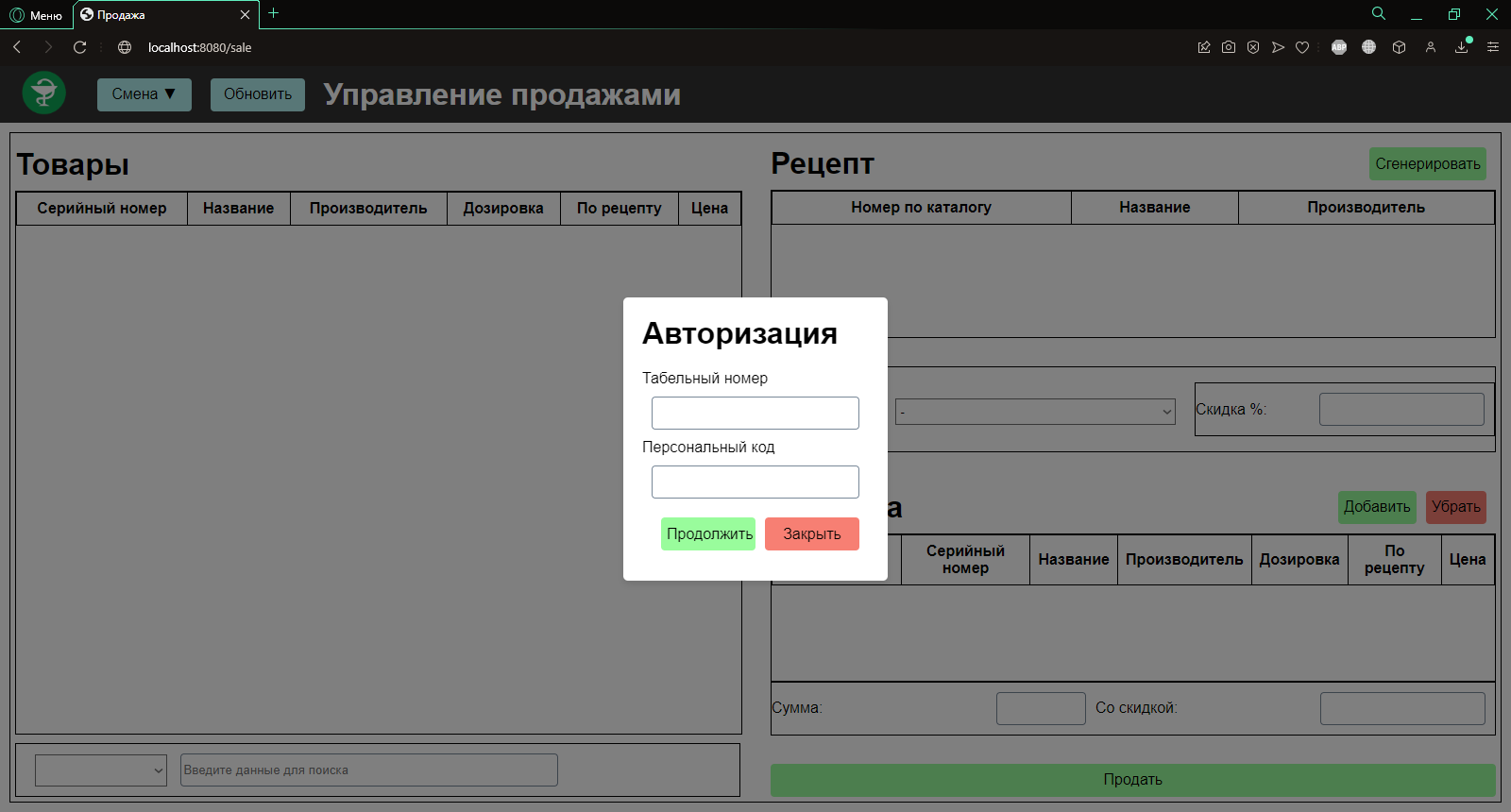


Рисунок 6.4 - Модальное окно авторизации

В случае, если пользователь введет сочетание табельного номера и персонального кода, запустится механизм, который сравнит эти значения со значениями, хранящимися в БД, а также проверит уровень доступа сотрудника с введенным табельным номером. Если комбинация табельного номера и персонального кода окажется правильной, а также уровень доступа сотрудника позволит ему пользоваться функционалом данной страницы, то модальное окно закроется, а пользователь сможет продолжить работу.

На уровне приложений данный функционал реализуется с использованием функций JavaScript и процедуры «проверить\_персональный\_код» в базе данных. Функция «ModalCreatePersonalCodeCheck» создает модальное окно, в которое пользователь вводит табельный номер и персональный код. Затем вызывается функция «CheckPersonalCode», которая отправляет данные на сервер, где они обрабатываются с помощью процедуры «проверить\_персональный\_код». Если сервер возвращает положительный статус и уровень доступа, соответствующий правам на просмотр данной страницы, вызывается функция «LoadPage», которая загружает соответствующую страницу.

# 7 Установка и эксплуатация

Для установки системы автоматизированного аптечного учета необходимо выполнить несколько шагов:

1) установка postgresql:

- скачайте и установите postgresql с официального сайта postgresql;

- создайте новую базу данных для системы.

2) установка java и spring:

- убедитесь, что на вашем компьютере установлена версия openjdk-21.0.2 java development kit (jdk);

- скачайте и установите spring boot.

3) запуск проекта:

- откройте терминал и перейдите в корневую директорию проекта;

- выполните команду ./mvnw spring-boot:run для запуска сервера;

- убедитесь, что сервер успешно запустился и слушает заданный порт.

4) инициализация базы данных:

- разверните дамп базы данных с помощью команды pg\_dump -U postgres -h localhost -p 5432 -d Apteka > Apteka \_dump.sql;

- заполните базу данных начальными данными, если это необходимо.

Теперь рассмотрим эксплуатацию системы. На рисунке 7.1 изображена главная страница, доступная каждому пользователю, взаимодействующему с системой. Здесь можно выбрать нужный функционал системы и получить разнообразные отчеты.

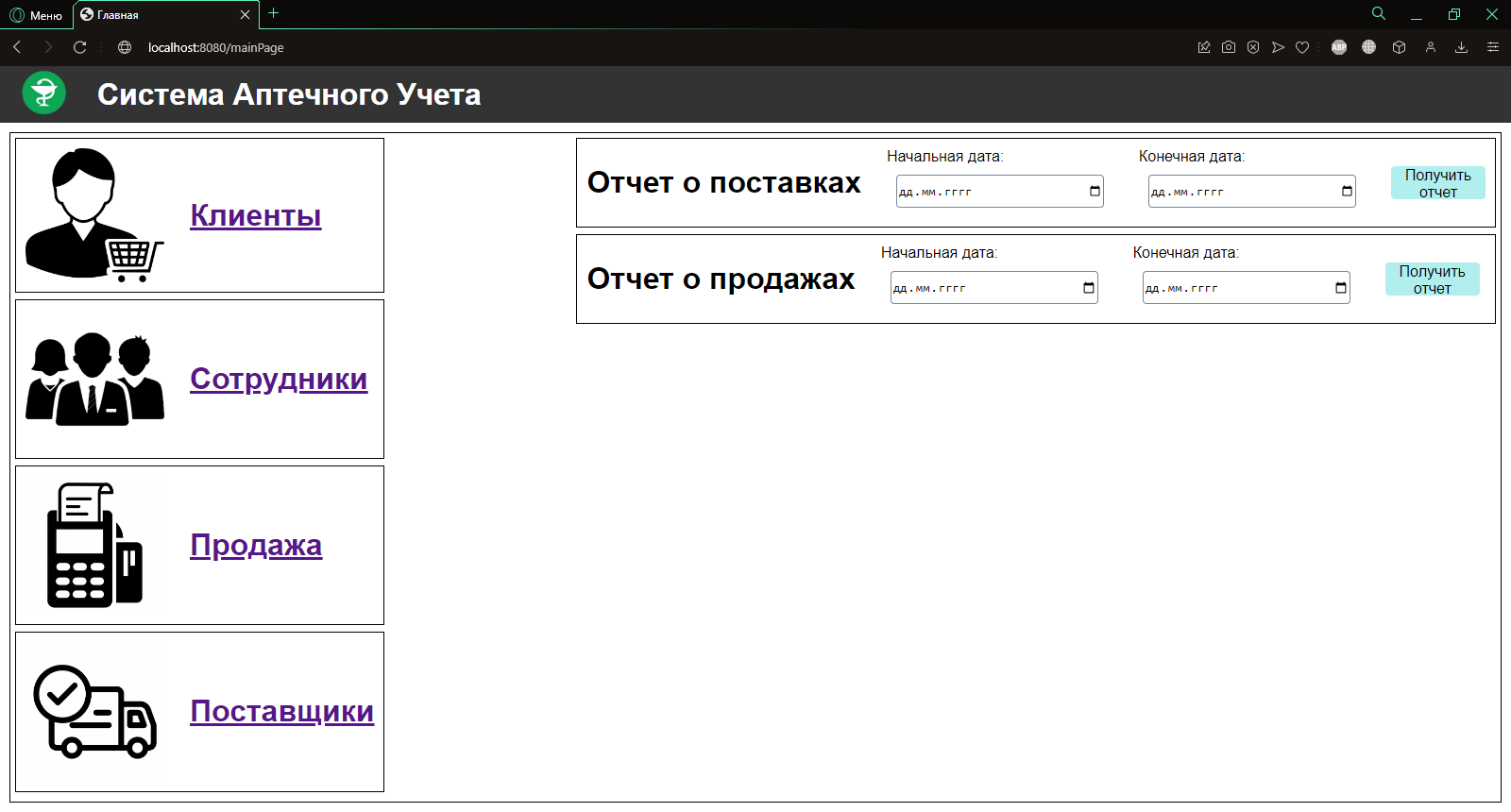


Рисунок 7.1 - Главная страница системы

Перейдем к странице «Продажа», отвечающей за реализацию функции продажи лекарств. На данной странице, как показано на рисунке 7.2, выделены четыре основных блока: «Товары», «Рецепт», Клиент» и «Продажа».

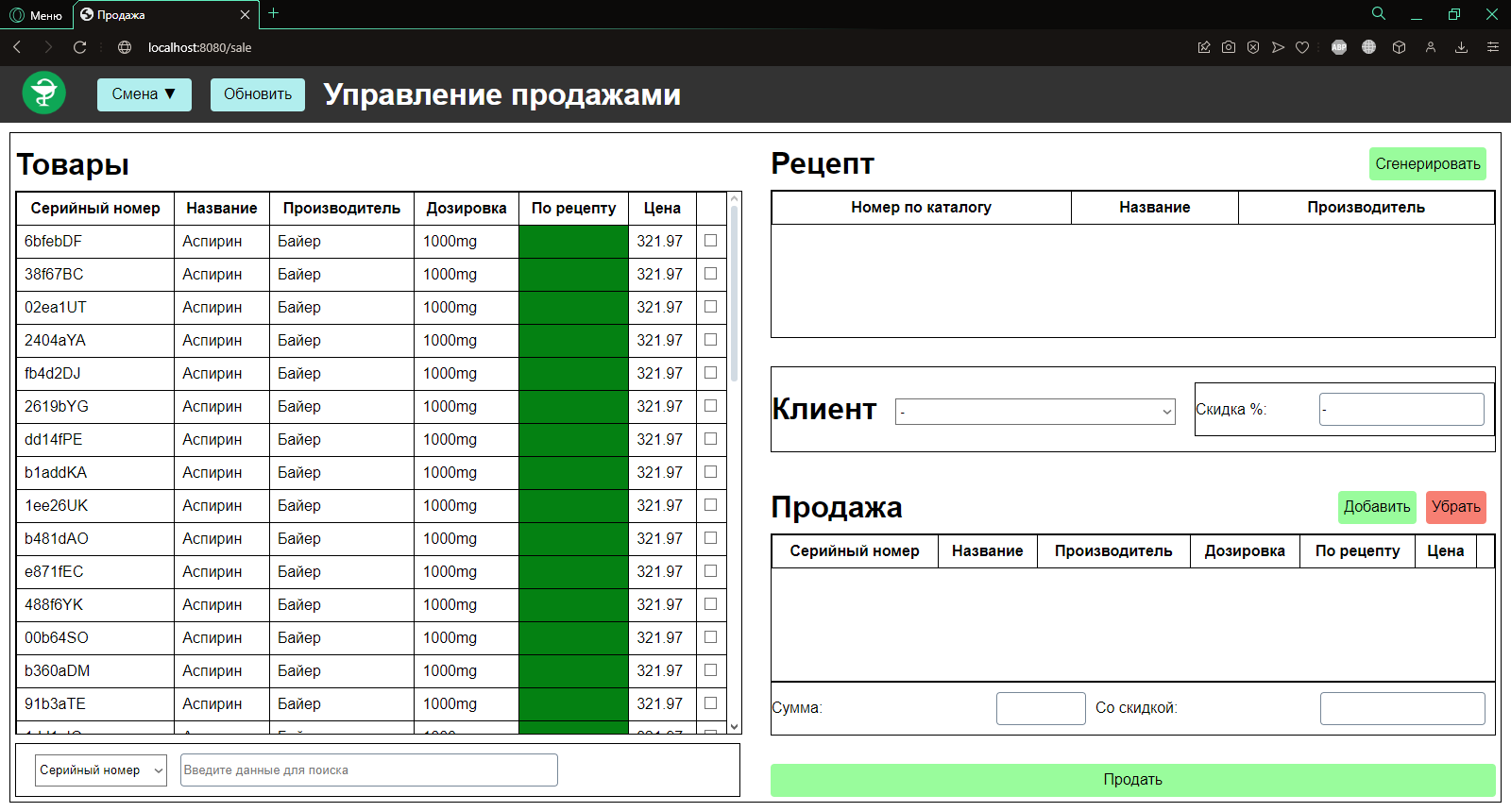


Рисунок 7.2 - Страница продаж

В блоке «Товары» представлены все препараты, имеющиеся в данный момент на складе. В нем можно увидеть, какие лекарства отпускаются по рецепту или без него, а также узнать их стоимость. В столбце «По рецепту» ячейки окрашиваются в красный или зеленый цвет, тем самым символизируют о том, что данное лекарство отпускается без рецепта или по нему.

Раздел «Рецепт» необходим для того, чтобы продемонстрировать работу с рецептами в системе, если клиенту необходимо купить лекарства по назначению врача. Современные аптеки взаимодействуют с системой ЕГИСЗ, которая позволяет продавцу ввести серию и номер рецепта для определения, какие лекарства можно продать по данному рецепту. Однако в рамках данной курсовой работы не реализовано реальное взаимодействие с ЕГИСЗ. Вместо этого предусмотрена генерация случайного рецепта для демонстрации функционала программы без необходимости подключения к внешним системам. На рисунке 7.3 рассмотрим среднестатистическую ситуацию продажи.

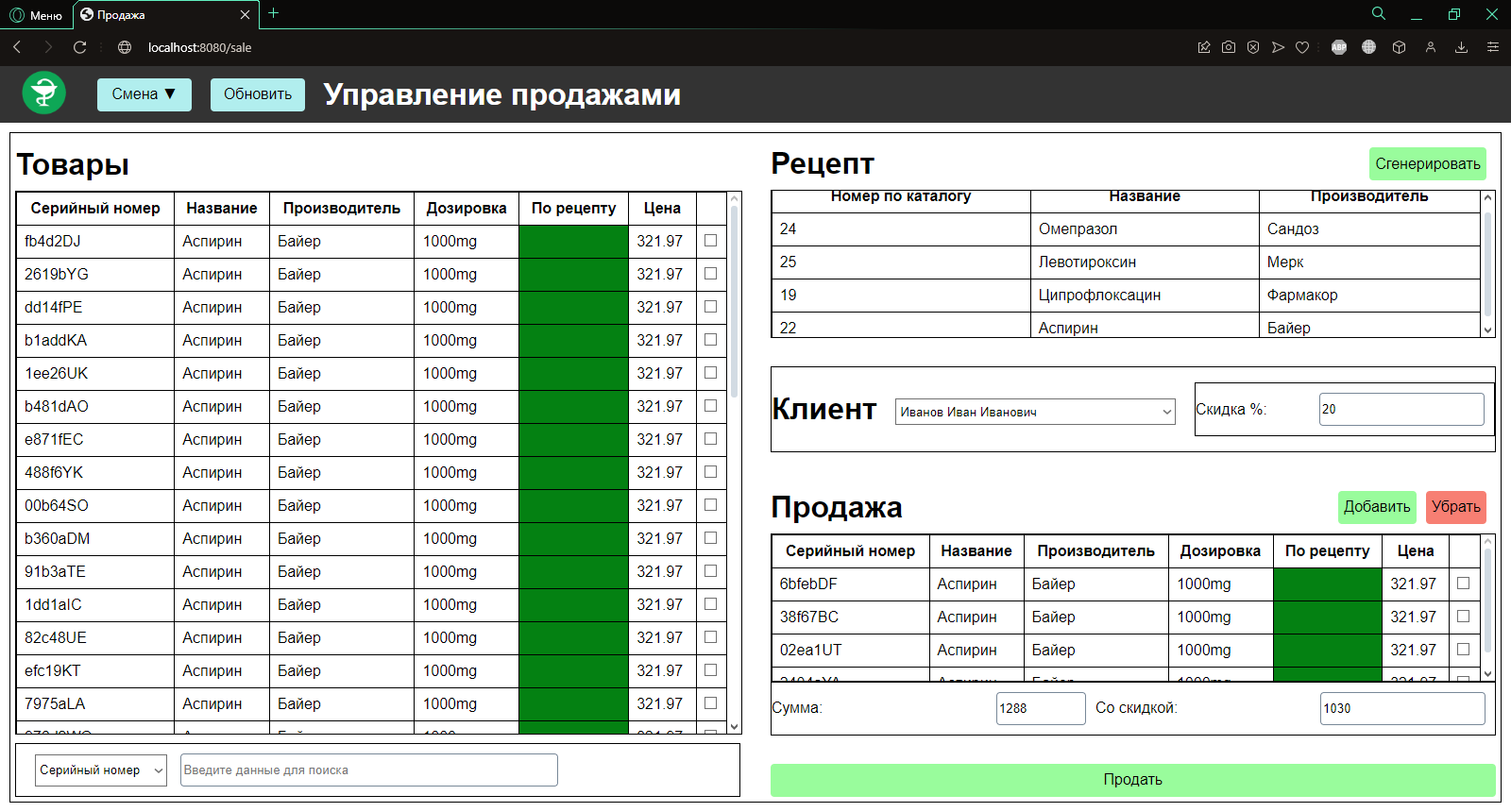


Рисунок 7.3 - Пример осуществления продажи

Следующий блок предназначен для работы с клиентами. Если клиент использует свою бонусную карту, система идентифицирует его и применяет индивидуальную скидку к итоговой стоимости покупки.

Блок «Продажа» отображает, какие лекарства из таблицы «Товары» были выбраны для продажи. С помощью кнопок «Добавить» и «Убрать» можно перемещать отмеченные чекбоксом лекарства из таблицы «Товары» в таблицу «Продажа». В этом блоке также показывается итоговая цена покупки и цена с учетом скидки. При нажатии кнопки «Продать» происходит завершение продажи товара.

Далее рассмотрим страницу «Поставщики», изображенную на рисунке 7.4. На ней представлены все поставщики и их товары, которые можно заказать. Для работы с поставщиками существует множество различных систем, поэтому в рамках данной работы для демонстрации функционала представлен определённый набор поставщиков с условным наличием конкретного ассортимента товаров.

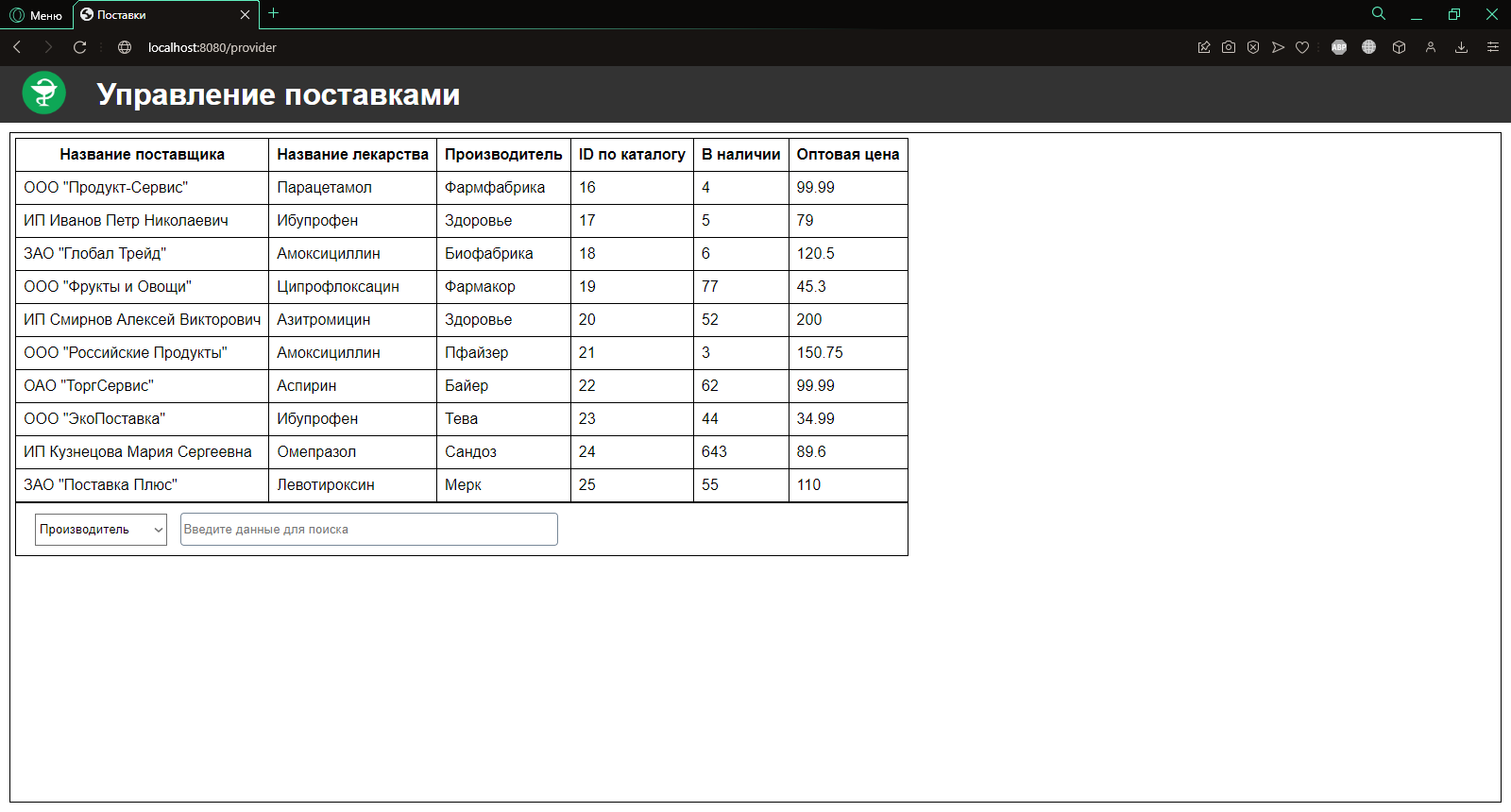


Рисунок 7.4 - Страница поставщиков

Чтобы система могла эффективно взаимодействовать с поставщиками, их каталог должен совпадать с тем, что находится в системе, и его необходимо регулярно обновлять. Для этого также существует большое количество специализированных программ.

Рассмотрим процесс создания поставки. Как видно на рисунке 7.5, открывается форма, в которой некоторые поля автоматически заполняются в соответствии с выбором поставщика. Наша задача указать наценку, дату и количество закупаемого товара. Далее товар поступает на склад, где он регистрируется, а затем поступает в продажу.

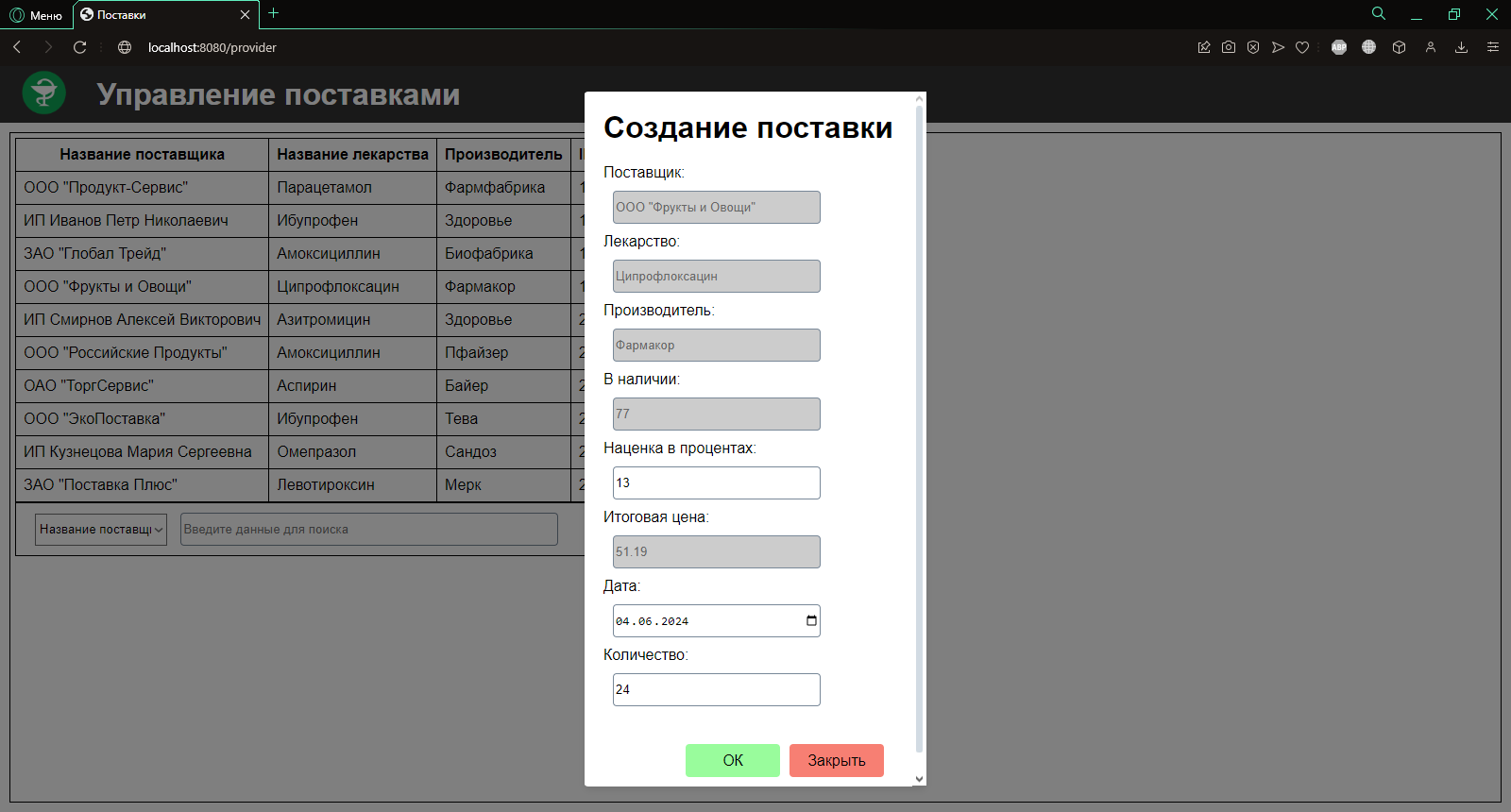


Рисунок 7.5 - Организация поставки

На странице «Клиенты» осуществляется создание и удаление клиентов, так же там можно увидеть размер скидки каждого клиента, как показано на рисунке 7.6.

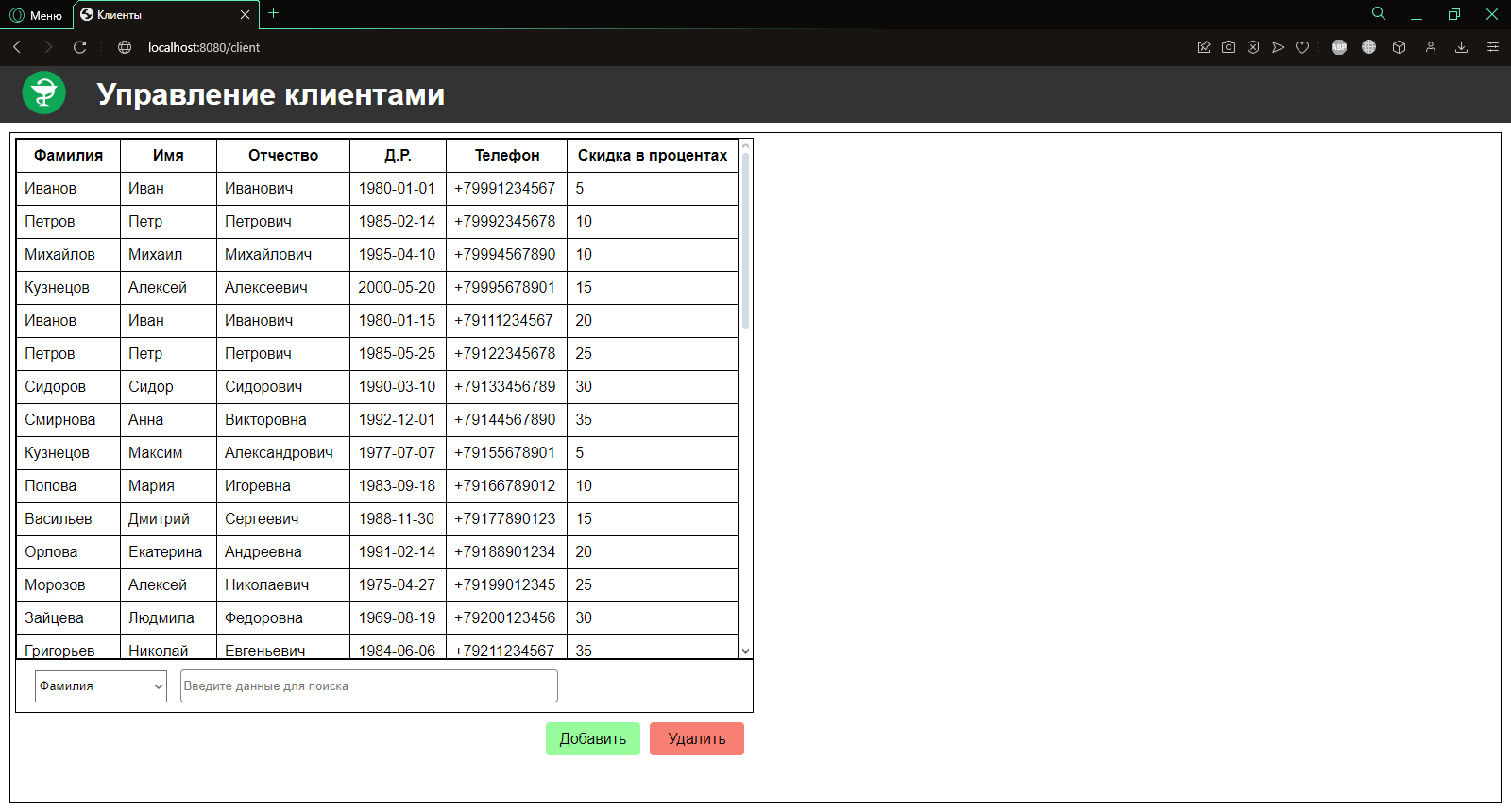


Рисунок 7.6 - Страница управления клиентами

Для того чтобы зарегистрировать нового клиента необходимо заполнить форму, представленную на рисунке 7.7.

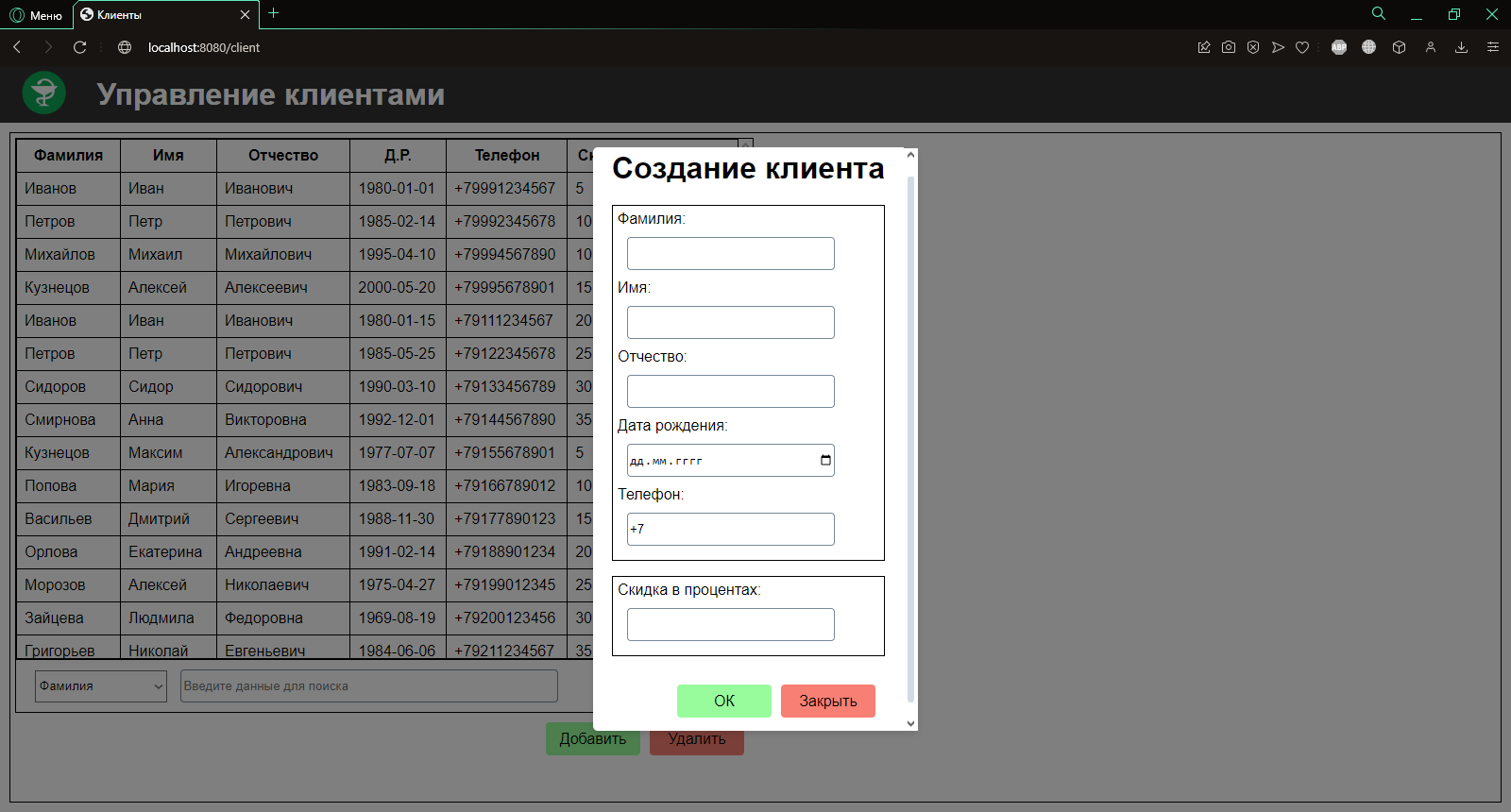


Рисунок 7.7 - Регистрация клиента

Управление сотрудниками осуществляется схожим образом, отличие лишь в том, что для регистрации нового сотрудника необходимо указать его паспортные данные, должность и персональный код, как показано на рисунке 7.8.

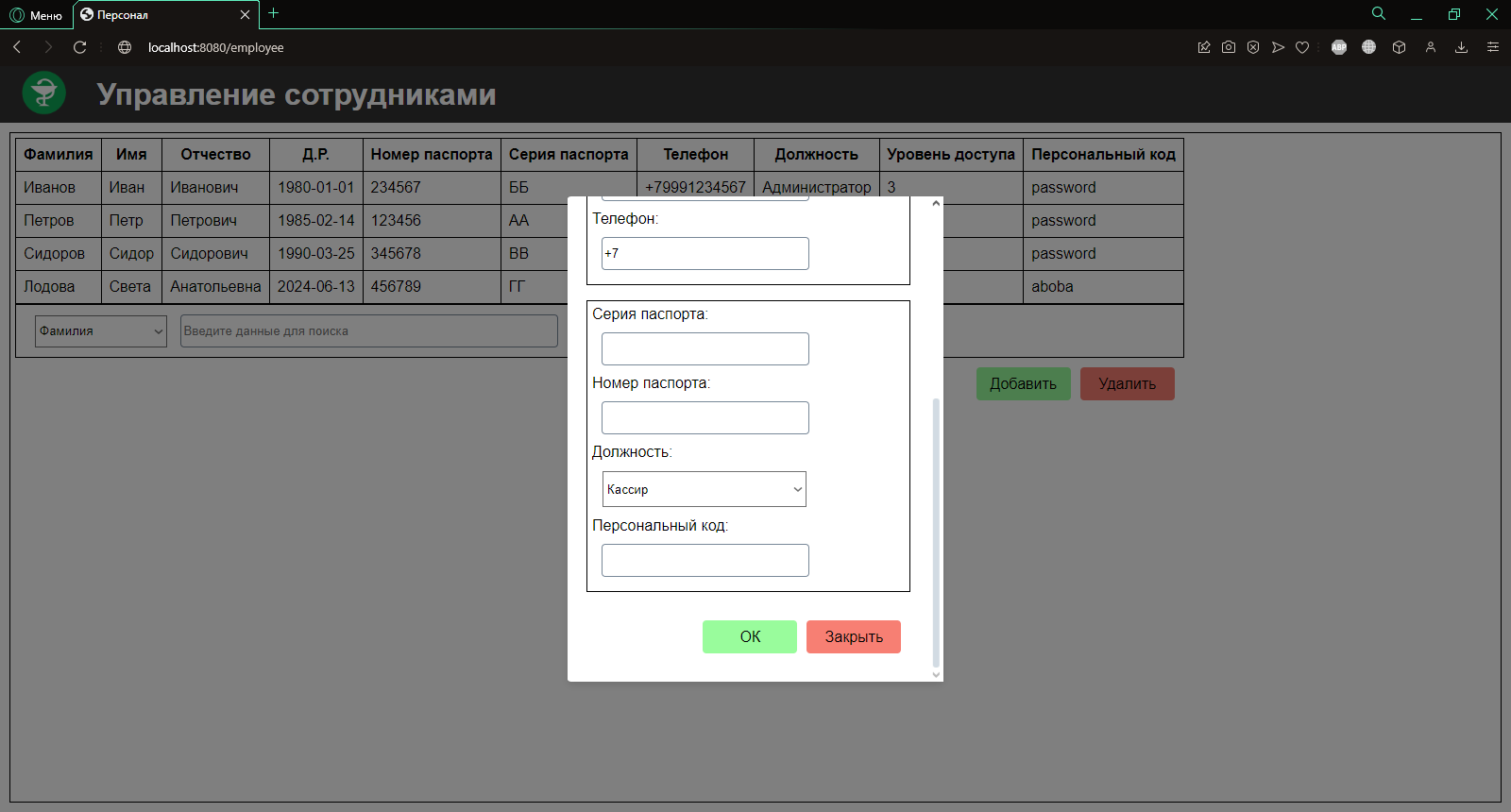


Рисунок 7.8 - Регистрация сотрудника

На этом основной функционал системы рассмотрен, осталось обсудить только создание отчетов. Для этого на главной странице необходимо в правой части меню выбрать интересующий нас отчет, указать период и нажать кнопку «Получить отчет», после чего браузер скачает отчет в виде xlsx документа, как показано на рисунке 7.9.

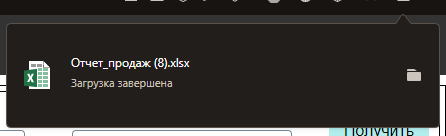


Рисунок 7.9 - Создание отчета

# Заключение

В рамках данной курсовой работы была разработан проект автоматизированной системы аптечного учёта, направленный на оптимизацию и повышение эффективности работы аптек и аптечных сетей. В процессе работы были достигнуты все поставленные цели и выполнены ключевые задачи, что позволило создать комплексную и функциональную систему.

Основные достижения проекта включают:

- формирование и анализ требований к базе данных, что позволило глубже понять специфические потребности отрасли и заложить основу для дальнейшего проектирования и реализации системы;

- проведение концептуального и физического проектирования базы данных, что обеспечило надёжность, целостность и масштабируемость системы данных;

- разработку клиент-серверного приложения с использованием современных технологий, таких как PostgreSQL и Java Spring, что позволило создать высокопроизводительное и безопасное веб-приложение;

- реализацию основных функционалов системы, включая управление товарным запасом, кадровый учёт, учёт клиентов, продажу и транзакции, закупки и поставки, а также создание отчетов. Эти компоненты были интегрированы в единое решение, способное удовлетворить потребности различных пользователей системы;

- тестирование системы на всех этапах разработки, которое помогло выявить и устранить потенциальные ошибки, повышая надёжность и стабильность работы приложения;

- проектирование механизмов защиты базы данных от несанкционированного доступа, что обеспечило безопасность и конфиденциальность данных.

Все этапы разработки системы автоматизированного аптечного учёта были выполнены с учётом современных подходов и технологий. Созданная система отвечает требованиям НФБК что исключает аномалии данных и обеспечивает высокую степень целостности.

Таким образом, разработанная система автоматизированного аптечного учёта представляет собой инновационное и эффективное решение, которое способно значительно повысить производительность и качество работы аптек и аптечных сетей. В перспективе существует возможность расширения функционала системы и её интеграции с другими системами здравоохранения для дальнейшего улучшения автоматизации процессов и повышения уровня обслуживания клиентов.

# Список использованных источников

1. Байдаров И. PostgreSQL. Оперативное руководство. - М.: ДМК Пресс, 2021. - 320 с.

2. Постановление Правительства РФ от 9 февраля 2022 года № 140 «О Единой государственной информационной системе в сфере здравоохранения (ЕГИСЗ)» // СЗ РФ. - 2022. - № 7.

3. Приказ Министерства здравоохранения Российской Федерации от 20 декабря 2020 года № 1010н «Об утверждении Порядка осуществления внутреннего контроля качества и безопасности медицинской деятельности» // Бюллетень нормативных актов федеральных органов исполнительной власти. - 2021. - № 5.

4. Нахмансон Е. Автоматизация в фармацевтической отрасли // Бизнес и технологии. - 2023. - № 3. - С. 45-52.

5. Официальная документация PostgreSQL [Электронный ресурс]. - Режим доступа: https://www.postgresql.org/docs/. - Дата обращения: 05.06.2024.

6. Официальная документация Spring [Электронный ресурс]. - Режим доступа: https://spring.io/projects/spring-framework. - Дата обращения: 05.06.2024.

7. Учебные материалы по JavaScript [Электронный ресурс]. - Режим доступа: https://developer.mozilla.org/ru/docs/Web/JavaScript. - Дата обращения: 05.06.2024.

Приложение А  
(обязательное)

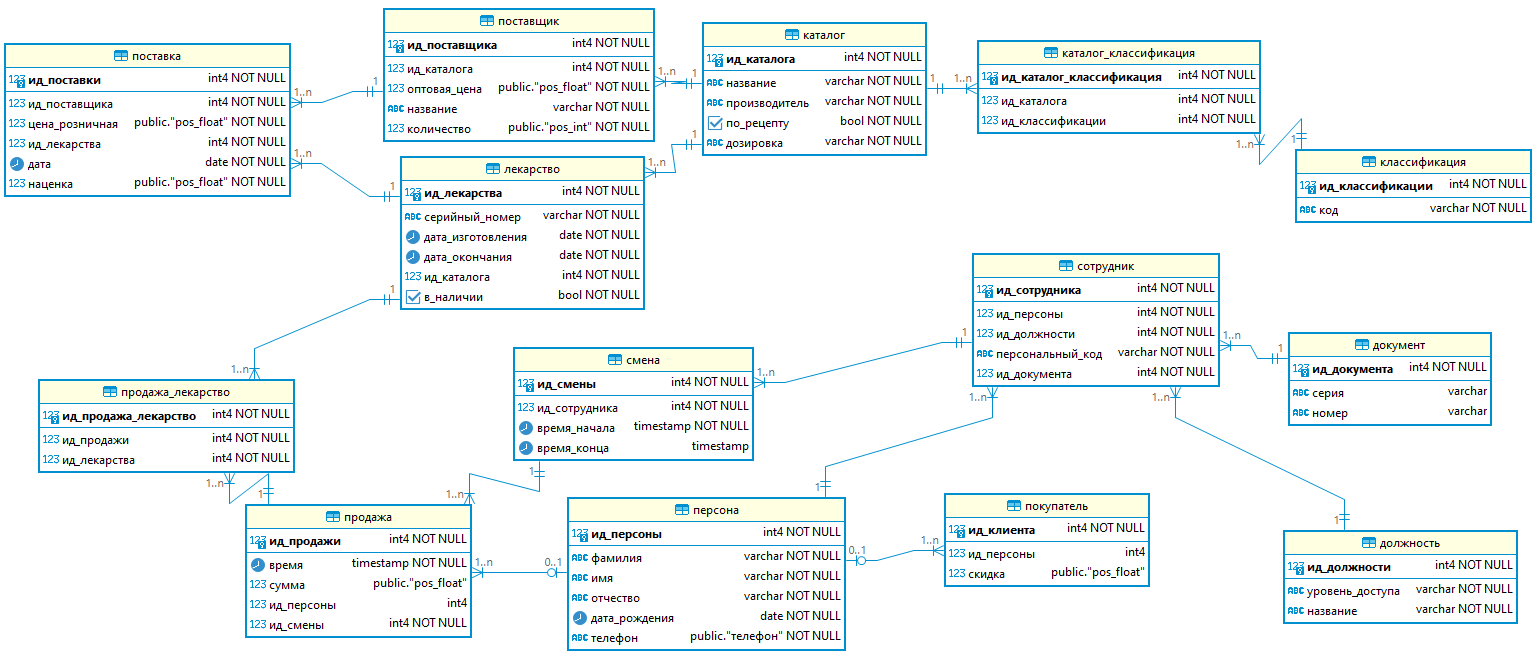


Рисунок А.1 - Схема базы данных