Содержание

ВВЕДЕНИЕ 3

ГЛАВА I. Теоретическая часть 5

1.1 Анализ предметной области 5

1.1.2 Существующие стандарты и регламенты для аптечного дела 6

1.2 Цели и задачи 8

1.3 Основные конструкции для разработки базы данных 10

1.3.1 Назначение базы данных 11

1.3.2 Выполняемые базой данных функции 12

1.4 Разработка web-приложения 14

1.4.1 Принципы проектирования web-приложения 16

1.4.2 Механизмы аутентификации и авторизации для пользователей 18

1.5 Тестирование системы 20

1.5.1 Методология тестирования 21

1.5.2 Проведение функционального тестирования 22

1.5.3 Обработка ошибок и тестирование безопасности 23

ГЛАВА II. Практическая часть 25

2.1 Архитектура программного обеспечения 25

2.1.1 Общая структура системы 25

2.1.2 Технологический стек 26

2.1.3 Взаимодействие компонентов 27

2.2 Проектирование базы данных 27

2.2.1 Принципы проектирования 28

2.2.2 Проектирование логической базы данных 28

2.2.3 Связи между таблицами 29

2.2.4 Создание физической базы данных 30

2.3 Пользовательский интерфейс 33

2.3.1 Основные страницы 33

2.3.2 Адаптивность 34

2.4 Система аутентификации и авторизации 34

2.4.1 Механизм аутентификации 35

2.4.2 Система ролей 35

2.5 Реализация API 36

2.5.1 Основные маршруты API и их назначение 37

2.5.2 Формат запросов 37

2.6 Обработка ошибок и логирование 37

2.6.1 Типы ошибок 38

2.6.2 Механизм логирования 39

ЗАКЛЮЧЕНИЕ 40

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 42

ВВЕДЕНИЕ

Современные технологии активно интегрируются во все сферы нашей жизни, включая здравоохранение, что открывает новые горизонты для улучшения качества медицинских услуг. В условиях постоянного роста потребностей населения в медицинских услугах и препаратах аптечные пункты сталкиваются с рядом вызовов, связанных с управлением своими ресурсами. Эффективное управление аптечными пунктами становится одной из ключевых задач, направленных на обеспечение доступности и качества медицинской помощи, что, в свою очередь, закладывает основу для здорового и полноценного общества. В частности, оптимизация процессов управления складскими запасами представляется важным аспектом, обладающим значительным потенциалом, который не только позволяет сократить затраты, но и значительно повышает уровень обслуживания клиентов.

Неэффективное управление запасами может привести к различным проблемам, что в свою очередь может негативно сказаться на здоровье пациентов и усложнить работу аптечных сотрудников. Например, дефицит необходимых лекарственных средств может вызвать большие трудности в лечении, а невозможность своевременного реагирования на потребности пациентов может усугубить их состояние и ухудшить качество жизни. С другой стороны, избыток товара также создает сложности, включая увеличенные затраты на хранение и риск устаревания продукции, что в конечном итоге приводит к финансовым потерям. Таким образом, наладка систем учета и контроля запасов становится не просто важной, а необходимой мерой для обеспечения стабильного и высококачественного обслуживания клиентов, что является приоритетом для современных аптечных пунктов.

В связи с вышеописанными проблемами целью данной дипломной работы является разработка веб-приложения для аптечного пункта, обладающего функционалом для эффективного управления складскими запасами. Создание интуитивно понятного и эффективного инструмента, который позволит фармацевтам и управляющим аптекой в реальном времени отслеживать наличие лекарственных средств, контролировать сроки годности, а также планировать закупки, является основным направлением данного исследования. Это приложение не только упростит ведение учета, но и станет важным помощником в ежедневной работе.

Ключевыми задачами данной работы являются анализ существующих решений, исследование потребностей пользователей и проектирование интерфейса, что поможет создать поистине инновационный продукт, отвечающий всем современным требованиям. По мере продвижения в разработке следует учитывать разнообразные аспекты, такие как удобство использования, безопасность данных и интеграция с другими системами, что в итоге обеспечит комплексный подход к решению поставленной задачи. Ожидается, что внедрение разработанного веб-приложения окажет значительное влияние на процессы управления в аптечном пункте. Оно должно способствовать повышению эффективности работы сотрудников, снижению издержек, улучшению обслуживания клиентов и, в конечном итоге, повышению качества медицинской помощи. Кроме того, создание такого инструмента будет способствовать более эффективному распределению ресурсов и оптимизации логистики в сфере аптечного обслуживания, что неизменно скажется на общей эффективности системы здравоохранения.

ГЛАВА I. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

1.1 Анализ предметной области

Предметной областью данной дипломной работы является разработка Web-приложения для аптечного пункта с возможностью управления складскими запасами. Аптечный пункт - это предприятие, занимающееся продажей лекарственных средств и других медицинских препаратов населению. Эффективное управление аптечным пунктом требует точного учета и контроля складских запасов, чтобы обеспечить наличие необходимых лекарств и минимизировать затраты на хранение и списание устаревшей продукции.

В настоящее время, многие аптечные пункты сталкиваются с рядом проблем, связанных с управлением складскими запасами. Эти проблемы включают в себя сложности и трудности, которые возникают при попытке эффективно управлять запасами лекарственных средств. Отсутствие автоматизированных систем учета и контроля запасов является одной из основных проблем, которая приводит к ручному учету и ошибкам. Это может привести к неактуальным данным о состоянии запасов, что может привести к ошибкам в управлении запасами и, как следствие, к нехватке или превышению определенных лекарственных средств.

Трудности в прогнозировании потребности в лекарственных средствах и планировании закупок также являются серьезной проблемой. Это может привести к ситуациям, когда аптечный пункт закупает слишком много или слишком мало определенных лекарств, что может привести к финансовым потерям и снижению качества обслуживания населения. Кроме того, отсутствие возможности отслеживать сроки годности лекарств и списывать устаревшую продукцию может привести к серьезным проблемам с качеством лекарств и безопасностью пациентов.

Управление многочисленными поставщиками и контроль качества поступающей продукции также является значительной проблемой. Аптечные пункты часто работают с несколькими поставщиками, и обеспечение качества лекарств и других медицинских препаратов, поступающих от этих поставщиков, является крайне важным. Однако, это может быть сложной задачей, особенно если аптечный пункт не имеет эффективной системы управления поставщиками и контроля качества.

Разработка Web-приложения для аптечного пункта с возможностью управления складскими запасами может помочь решить эти проблемы и улучшить эффективность управления аптечным пунктом. Это приложение может обеспечить автоматизированный учет и контроль запасов, прогнозирование потребности в лекарственных средствах и планирование закупок, отслеживание сроков годности лекарств и списание устаревшей продукции, а также управление поставщиками и контроль качества поступающей продукции. Таким образом, разработка такого приложения может иметь положительное влияние на качество обслуживания населения и финансовую эффективность аптечного пункта.

1.1.2 Существующие стандарты и регламенты для аптечного дела

Существующие стандарты и регламенты для аптечного дела играют важную роль в обеспечении качества и безопасности фармацевтических препаратов, а также в регулировании деятельности аптек и фармацевтических организаций. Эти стандарты и регламенты разработаны для защиты здоровья и интересов пациентов, а также для поддержания высокого уровня профессиональной компетенции среди фармацевтов и других специалистов аптечного дела.

Одним из ключевых аспетов стандартов и регламентов для аптечного дела является обеспечение качества фармацевтических препаратов. Это включает в себя соблюдение строгих требований к производству, хранению и транспортировке препаратов, а также к их маркировке и документации. Кроме того, стандарты и регламенты устанавливают требования к качеству и чистоте используемых материалов, а также к оборудованию и технологиям, применяемым в аптеках и фармацевтических организациях.

Другим важным аспектом стандартов и регламентов для аптечного дела является обеспечение безопасности пациентов. Это включает в себя разработку и реализацию мер по предотвращению ошибок при назначении и выдаче препаратов, а также по мониторингу побочных эффектов и других реакций на препараты. Кроме того, стандарты и регламенты устанавливают требования к документации и учету фармацевтических препаратов, а также к их отслеживанию и контролю.

Стандарты и регламенты для аптечного дела также регулируют деятельность аптек и фармацевтических организаций. Это включает в себя требования к лицензированию и аккредитации аптек, а также к квалификации и компетенции фармацевтов и других специалистов. Кроме того, стандарты и регламенты устанавливают требования к оборудованию и технологиям, применяемым в аптеках, а также к их обслуживанию и техническому поддержанию.

В России стандарты и регламенты для аптечного дела установлены федеральными законами и нормативными актами, такими как закон "О лекарственных средствах" и приказы Министерства здравоохранения Российской Федерации. Эти акты устанавливают требования к качеству и безопасности фармацевтических препаратов, а также к деятельности аптек и фармацевтических организаций.

Для обеспечения соблюдения стандартов и регламентов для аптечного дела, аптеки и фармацевтические организации должны иметь систему управления качеством, которая включает в себя разработку и реализацию политики качества, а также обучение персонала и проведение внутренних аудитов. Кроме того, аптеки и фармацевтические организации должны участвовать в системе внешнего контроля качества, которая включает в себя регулярные инспекции и аудиты со стороны регулирующих органов.

В целом, существующие стандарты и регламенты для аптечного дела играют важную роль в обеспечении качества и безопасности фармацевтических препаратов, а также в регулировании деятельности аптек и фармацевтических организаций. Соблюдение этих стандартов и регламентов является обязательным для всех аптек и фармацевтических организаций, и их несоблюдение может привести к серьезным последствиям, включая штрафы и приостановку деятельности.

1.2 Цели и задачи

Целью данной дипломной работы является разработка Web-приложения для аптечного пункта, которое позволит эффективно управлять складскими запасами и решить вышеуказанные проблемы. Задачами дипломной работы является проведение комплексного анализа существующих систем управления складскими запасами и их недостатков, чтобы понять, какие функции и возможности необходимы для эффективного управления аптечным пунктом.

Другой задачей дипломной работы является разработка концепции и архитектуры Web-приложения для аптечного пункта, которая будет учитывать все необходимые функции и возможности для управления складскими запасами. Это требует глубокого понимания предметной области и опыта в области разработки Web-приложений. Разработка концепции и архитектуры приложения включает в себя определение требований к функциональности, безопасности и масштабируемости приложения, а также выбор технологий и фреймворков для его реализации.

Проектирование интерфейса и функционала приложения для управления складскими запасами является еще одной важной задачей дипломной работы. Интерфейс приложения должен быть интуитивно понятным и удобным в использовании, чтобы сотрудники аптечного пункта могли легко управлять запасами и получать необходимую информацию. Функционал приложения должен включать в себя учет и контроль запасов, прогнозирование потребности в лекарственных средствах и планирование закупок, отслеживание сроков годности лекарств и списание устаревшей продукции, управление поставщиками и контроль качества поступающей продукции.

Реализация приложения с использованием современных технологий и фреймворков является следующей задачей дипломной работы. Для реализации приложения необходимо выбрать подходящие технологии и фреймворки, язык программирования JavaScript, фреймворк Boostrap, базы данных SQLiteStudio, и интерфейсы HTML, CSS, JavaScript. Реализация приложения должна быть проведена с учетом требований к безопасности и масштабируемости.

Тестирование и отладка приложения для обеспечения его стабильности и безопасности являются важными задачами дипломной работы. Тестирование должно быть проведено на различных этапах разработки приложения, чтобы выявить и исправить ошибки и недостатки. Отладка приложения должна быть проведена для обеспечения его стабильной работы и безопасности.

Оценка эффективности приложения и его влияния на работу аптечного пункта является окончательной задачей дипломной работы. Эффективность приложения должна быть оценена по нескольким критериям, включая его функциональность, удобство использования, безопасность и масштабируемость. Влияние приложения на работу аптечного пункта должно быть оценено по критериям, таким как улучшение управления запасами, снижение затрат и увеличение качества обслуживания населения.

Функциональные требования к Web–приложению для аптечного пункта включают в себя учет и контроль запасов, прогнозирование потребности в лекарственных средствах и планирование закупок, отслеживание сроков годности лекарств и списание устаревшей продукции, управление поставщиками и контроль качества поступающей продукции, генерация отчетов и аналитических данных для оценки эффективности работы аптечного пункта. Учет и контроль запасов являются одними из наиболее важных функций приложения, поскольку они позволяют сотрудникам аптечного пункта получать актуальную информацию о состоянии запасов и управлять ими эффективно.

Прогнозирование потребности в лекарственных средствах и планирование закупок являются еще одной важной функцией приложения. Эта функция позволяет сотрудникам аптечного пункта прогнозировать потребность в определенных лекарственных средствах и планировать закупки соответствующим образом. Отслеживание сроков годности лекарств и списание устаревшей продукции являются важными функциями приложения, поскольку они позволяют сотрудникам аптечного пункта обеспечить качество лекарств и избежать финансовых потерь.

Управление поставщиками и контроль качества поступающей продукции являются важными функциями приложения, поскольку они позволяют сотрудникам аптечного пункта управлять поставщиками и контролировать качество поступающей продукции. Генерация отчетов и аналитических данных для оценки эффективности работы аптечного пункта является еще одной важной функцией приложения, поскольку она позволяет сотрудникам аптечного пункта оценивать эффективность работы и принимать обоснованные решения.

1.3 Основные конструкции для разработки базы данных

База данных – это организованная структура, предназначенная для хранения информации. Систему управления базой данных (СУБД) можно определить, как комплекс программных средств, предназначенных для создания структуры базы данных, заполнения её содержимым, редактирования содержимого.

– Таблицы.В базе данных информация хранится в виде двумерных таблиц. Можно так же импортировать и связывать таблицы из других СУБД или систем управления электронными таблицами. Одновременно могут быть открыты 1024 таблицы.

– Запросы. При помощи запросов можно произвести выборку данных по какому-нибудь критерию из разных таблиц. В запрос можно включать до 255 полей.

– Формы. Формы позволяют отображать данные из таблиц и запросов в более удобном для восприятия виде. С помощью форм можно добавлять и изменять данные, содержащиеся в таблицах. В формы позволяют включать модули.

– Отчёты. Отчёты предназначены для печати данных, содержащихся в таблицах и запросах, в красиво оформленном виде. Отчёты так же позволяют включать модули.

1.3.1 Назначение базы данных

База данных "Аптечный пункт" разрабатывается для эффективного управления информацией о лекарственных препаратах, поставщиках, клиентах, продажах и других аспектах, связанных с деятельностью аптеки. Целью этой базы данных является обеспечение надежного хранения данных, их быстрого доступа и обработки для оптимизации бизнес-процессов и обслуживания клиентов.

– Управление лекарственными препаратами: Возможность регистрации новых препаратов, хранение информации о них (наименование, состав, дозировка, указания и т.д.), поиск и обновление информации о препаратах.

– Управление поставщиками: Хранение информации о поставщиках, их контактных данных, условиях поставки и прочих деталях. Возможность добавления новых поставщиков, их изменения и удаления

– Управление клиентами: Возможность регистрации новых клиентов, хранение информации о них (контактные данные, медицинские данные, история покупок и т.д.), поиск и обновление информации о клиентах.

–....Управление продажами: Система учета продаж, возможность оформления заказов, хранение информации о продажах и заказах клиентов.

–....Финансовый учет: Учет финансовых операций, включая оплаты клиентов, счета, выставление счетов и отчетность.

–....Аналитика и отчетность: Генерация отчетов о продажах, клиентах, популярности препаратов, финансовых потоках и других аспектах деятельности аптеки.

1.3.2 Выполняемые базой данных функции

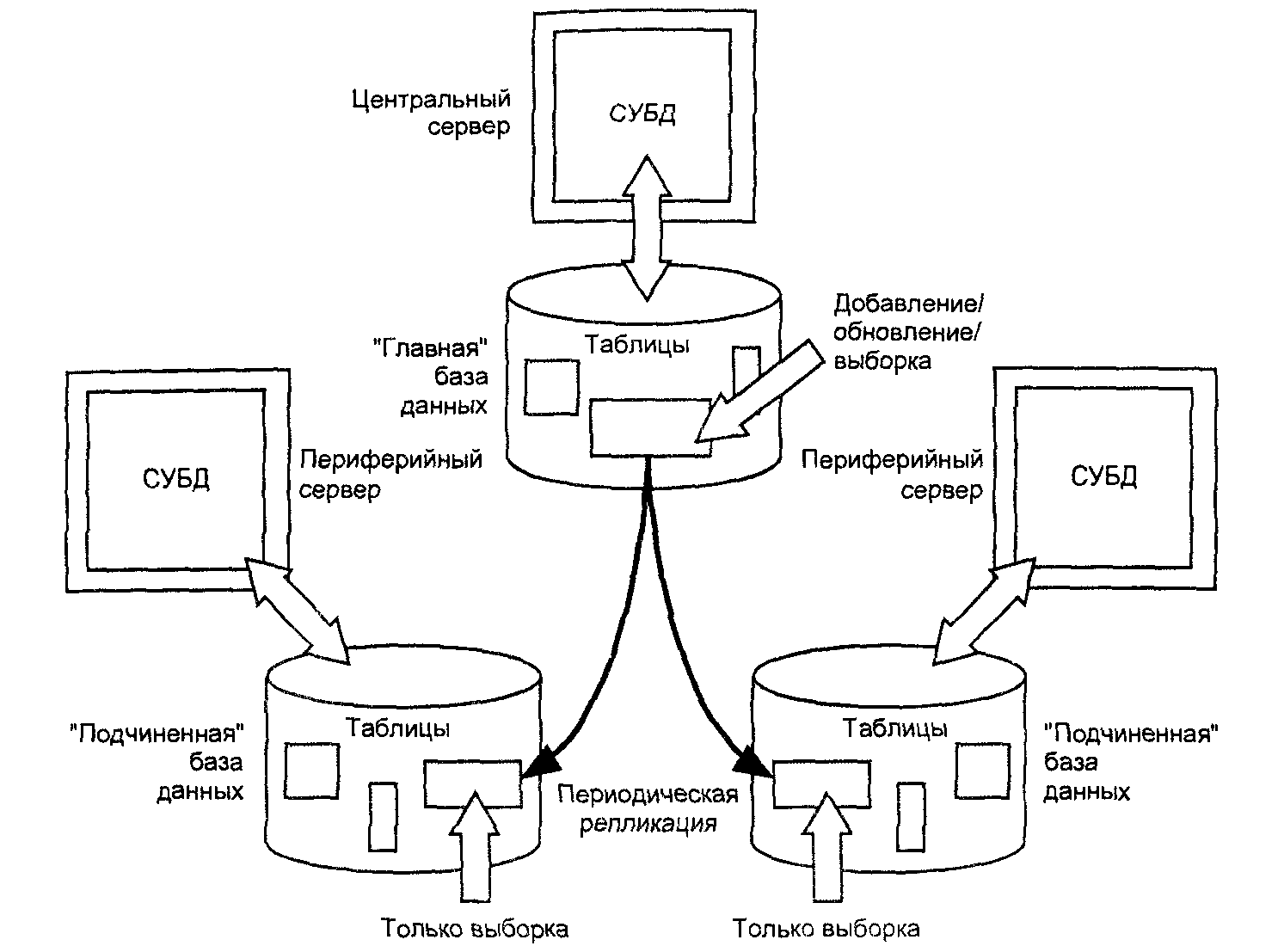


Рисунок 1 Схема описывающая работу СУДБ

Базы данных используются для хранения информации в различных приложениях, начиная от простых веб-сайтов и блогов, до сложных систем управления предприятием. Они позволяют эффективно управлять данными, обеспечивая возможность поиска, сортировки, фильтрации и анализа информации. Соответственно, они применяются в сферах, где требуются подобные функции. Яркий пример – базы данных интернет-магазинов, позволяющие быстро узнавать, есть ли товар в наличии, сколько он стоит и так далее.

Базы данных выполняют множество функций, которые помогают организациям и индивидуальным пользователям управлять данными.

База данных выполняет следующие функции:

– Организация и хранение данных, чувствительных к некорректным изменениям. Механизмы БД позволяют обеспечить правильность и согласованность информации во всех необходимых частях базы. Для защиты от случайных правок или замены используются разнообразные ограничения, триггеры и другие механизмы для проверки вводимых данных и поддержания их целостности. Примером подобного использование может служить организация банковской транзакции. Для банка критически важно, чтобы при переводе средств деньги ушли с одного счета и пришли на другой. Целостность данных обеспечивается механизмами БД, которые просто не записывают изменения, если они не произошли в полном объеме.

–  Хранение информации. БД предоставляет структурированное хранилище для хранения больших объемов данных. Она может содержать информацию о клиентах, продуктах, заказах, транзакциях и многом другом.

– Управление данными. БД позволяет эффективно управлять данными, включая добавление, удаление и изменение информации. Она предоставляет механизмы для поиска, сортировки и фильтрации данных, а также для выполнения сложных запросов.

– Поддержка многопользовательского доступа. БД дает возможность нескольким пользователям одновременно работать с данными. Она обеспечивает контроль доступа таким образом, чтобы предотвратить конфликты при одновременном изменении информации несколькими пользователями. Также БД может ограничивать доступ к данным пользователям разного уровня и настраивать доступные им операции.

– Формирование отчетов для аналитики. Современная БД может предоставлять инструменты для анализа и получения информации.

1.4 Разработка web-приложения

Разработка web-приложения для аптечного пункта с возможностью управления складскими запасами является сложным и детальным процессом, который требует глубокого понимания требований и задач системы. Для начала, необходимо определить основные функции и возможности, которые будут реализованы в системе.

Одной из наиболее важных функций системы является возможность ввода и редактирования информации о лекарствах и складских запасах. Для этого необходимо разработать удобный и интуитивно понятный интерфейс, который позволит пользователям вносить и редактировать данные о лекарствах, включая их название, описание, цену и количество в запасе. Кроме того, система должна предоставлять возможность отслеживать движения складских запасов, включая поставки, продажи и другие операции.

Другой важной функцией системы является возможность управления заказами и поставками. Для этого необходимо разработать модуль, который позволит пользователям создавать и редактировать заказы, включая информацию о лекарствах, количестве и поставщике. Кроме того, система должна предоставлять возможность отслеживать статус заказов и получать уведомления о поставках.

Система также должна предоставлять возможность создавать отчеты и аналитические данные о складских запасах и заказах. Для этого необходимо разработать модуль, который позволит пользователям создавать отчеты по различным критериям, включая даты, лекарства и поставщиков. Кроме того, система должна предоставлять возможность экспортировать отчеты в различные форматы, включая Excel и PDF.

Для обеспечения безопасности и защиты данных, система должна реализовывать механизмы аутентификации и авторизации. Для этого необходимо разработать систему, которая будет проверять идентификацию пользователей и определять их права доступа к различным функциям и данным системы. Кроме того, система должна предоставлять возможность назначать роли и права доступа пользователям, в зависимости от их функций и обязанностей.

В процессе разработки системы, необходимо также учитывать вопросы масштабируемости и гибкости. Система должна быть способна обрабатывать большое количество данных и пользователей, без значительного снижения производительности. Кроме того, система должна быть гибкой и адаптивной, чтобы она могла быть легко модифицирована и расширена, в зависимости от возникающих потребностей.

Для реализации системы, можно использовать различные технологии и инструменты, включая языки программирования, базы данных и фреймворки. Например, можно использовать язык программирования JavaScript в сочетании с базой данных SQLiteStudio Кроме того, можно использовать фреймворк, такой как Boostrap для ускорения и упрощения процесса разработки.

В целом, разработка web-приложения для аптечного пункта с возможностью управления складскими запасами является сложным и детальным процессом, который требует глубокого понимания требований и задач системы. Для успешной реализации системы, необходимо учитывать вопросы безопасности, масштабируемости и гибкости, а также использовать различные технологии и инструменты для ускорения и упрощения процесса разработки.

При разработке системы, необходимо также учитывать вопросы пользовательского интерфейса и пользовательского опыта. Система должна быть удобной и интуитивно понятной, чтобы пользователи могли легко ориентироваться в ней и выполнять свои задачи. Для этого необходимо разработать систему, которая будет предоставлять четкие и понятные инструкции, а также будет реализовывать механизмы обратной связи, чтобы пользователи могли получить помощь и поддержку, когда это необходимо.

Кроме того, система должна быть способна обрабатывать различного рода ошибки и исключения, которые могут возникнуть в процессе ее работы. Для этого необходимо разработать систему, которая будет обнаруживать и обрабатывать ошибки, а также будет предоставлять пользователям информацию о том, что произошло и как это исправить.

В процессе разработки системы, необходимо также учитывать вопросы тестирования и отладки. Система должна быть тщательно протестирована, чтобы убедиться, что она работает корректно и выполняет все свои функции. Для этого необходимо разработать план тестирования, который будет включать различные сценарии и тестовые случаи, а также будет использоваться для выявления и исправления ошибок.

В целом, разработка web-приложения для аптечного пункта с возможностью управления складскими запасами является сложным и детальным процессом, который требует глубокого понимания требований и задач системы. Для успешной реализации системы, необходимо учитывать вопросы безопасности, масштабируемости, гибкости, пользовательского интерфейса и пользовательского опыта, а также использовать различные технологии и инструменты для ускорения и упрощения процесса разработки.

1.4.1 Принципы проектирования web-приложения

При проектировании web-приложения важно учитывать простоту и ясность дизайна. Дизайн приложения должен быть простым и понятным для пользователей, чтобы они могли легко ориентироваться в нем и находить необходимую информацию. Интерфейс должен быть интуитивным и не перегруженным лишней информацией, чтобы пользователи могли сосредоточиться на своих задачах и не отвлекаться на ненужные элементы.

Адаптивность является еще одним важным фактором при проектировании web-приложения. Приложение должно быть адаптировано для различных устройств и размеров экрана, включая смартфоны, планшеты и компьютеры. Это позволит пользователям доступ к приложению с любого устройства, независимо от его размера и типа.

Безопасность является критически важным аспектом web-приложения. Приложение должно обеспечивать безопасность пользовательских данных и защиты от различных угроз, таких как хакерские атаки и вирусы. Для этого необходимо реализовать эффективные механизмы аутентификации и авторизации, а также использовать шифрование данных и другие средства защиты.

Масштабируемость является еще одним важным фактором при проектировании web-приложения. Приложение должно быть способно обрабатывать увеличение трафика и пользовательской нагрузки, не теряя своей производительности. Для этого необходимо использовать эффективные алгоритмы и структуры данных, а также использовать распределенные системы и облака.

Тестирование и отладка являются важными этапами при проектировании web-приложения. Приложение должно быть подвергнуто тщательному тестированию и отладке, чтобы выявить и исправить ошибки и недочеты. Это позволит обеспечить стабильность и надежность приложения и предотвратить появление ошибок и сбоев в работе.

Поддержка и обновление являются важными аспектами web-приложения. Приложение должно быть поддержано и обновляться регулярно, чтобы обеспечить его стабильность и актуальность. Это позволит добавлять новые функции и возможности, исправлять ошибки и недочеты, а также улучшать производительность и безопасность приложения.

Доступность является важным фактором при проектировании web-приложения. Приложение должно быть доступно для людей с ограниченными возможностями, включая пользователей с нарушениями зрения, слуха и движения. Для этого необходимо использовать специальные технологии и средства, такие как экранные считыватели, клавиатуры Брайля и другие устройства.

Эффективность является еще одним важным фактором при проектировании web-приложения. Приложение должно быть эффективным в решении задач пользователей и обеспечивать максимальную продуктивность. Для этого необходимо использовать эффективные алгоритмы и структуры данных, а также использовать распределенные системы и облака.

Соответствие требованиям является важным аспектом web-приложения. Приложение должно соответствовать всем необходимым требованиям и стандартам, включая требования к безопасности, конфиденциальности и доступности. Для этого необходимо использовать специальные технологии и средства, такие как шифрование данных, аутентификация и авторизация, а также использовать распределенные системы и облака.

При проектировании web-приложения необходимо учитывать головоломный подход, который предполагает разделение сложных задач на простые и понятные этапы. Это позволит пользователям легко ориентироваться в приложении и находить необходимую информацию. Кроме того, необходимо учитывать пользовательский опыт, интерактивный дизайн, контент-стратегию и технологическую нейтральность, чтобы создать эффективное и удобное приложение.

В целом, проектирование web-приложения требует тщательного планирования, анализа и реализации. Необходимо учитывать различные факторы, такие как простота и ясность дизайна, навигация и поиск, адаптивность, безопасность, масштабируемость, тестирование и отладка, поддержка и обновление, доступность, эффективность и соответствие требованиям, чтобы создать эффективное и удобное приложение.

1.4.2 Механизмы аутентификации и авторизации для пользователей

Для обеспечения безопасности и защиты данных в системе управления складскими запасами аптечного пункта необходимо реализовать механизмы аутентификации и авторизации для пользователей. Эти механизмы будут нести ответственность за проверку идентификации пользователей и определение их прав доступа к различным функциям и данным системы.

Аутентификация представляет собой процесс проверки идентификации пользователя перед предоставлением ему доступа к системе. В нашей системе аутентификация будет осуществляться путем ввода логина и пароля. Создание таблицы пользователей является первым шагом в реализации аутентификации, где будет храниться информация о пользователях, включая логин, пароль и другие данные. После этого, реализация формы входа будет необходима, которая будет запрашивать у пользователя логин и пароль.

После ввода логина и пароля, система будет проверять их корректность путем сравнения с данными в таблице пользователей. Если логин и пароль корректны, система будет создавать сессию для пользователя, которая будет хранить информацию о его идентификации и правах доступа. Эта сессия будет использоваться для проверки прав доступа пользователя к различным функциям и данным системы.

Авторизация представляет собой процесс определения прав доступа пользователя к различным функциям и данным системы. В нашей системе авторизация будет осуществляться на основе ролей пользователей. Роли будут определять, какие функции и данные системы доступны пользователю. Для реализации авторизации необходимо создать набор ролей, которые будут определять права доступа пользователей к системе.

Назначение ролей пользователям будет осуществляться в зависимости от их функций и обязанностей. Для каждой роли необходимо определить, какие функции и данные системы доступны пользователям с этой ролью. Перед предоставлением доступа к какой-либо функции или данным, система будет проверять, имеет ли пользователь соответствующую роль и права доступа.

Реализация авторизации на основе ролей будет включать создание таблицы ролей, которая будет хранить информацию о ролях и их правах доступа. Кроме того, будет создана таблица связей между пользователями и ролями, которая будет хранить информацию о связях между пользователями и ролями. Назначение ролей пользователям будет осуществляться в зависимости от их функций и обязанностей, и система будет проверять права доступа пользователя перед предоставлением доступа к какой-либо функции или данным.

Эта система авторизации на основе ролей будет обеспечивать безопасность и защиту данных в системе управления складскими запасами аптечного пункта. Она будет гарантировать, что пользователи имеют доступ только к тем функциям и данным, которые необходимы для выполнения их задач и обязанностей. Кроме того, эта система будет гибкой и сможет быть легко модифицирована и расширена, если это необходимо.

В целом, реализация механизмов аутентификации и авторизации для пользователей является важным шагом в обеспечении безопасности и защиты данных в системе управления складскими запасами аптечного пункта. Она будет гарантировать, что пользователи имеют доступ только к тем функциям и данным, которые необходимы для выполнения их задач и обязанностей, и будет предотвращать несанкционированный доступ к системе и ее данным. Это позволит повысить уровень безопасности и защиты данных в системе, а также обеспечить эффективное и результативное управление складскими запасами аптечного пункта.

1.5 Тестирование системы

Тестирование системы является важным этапом в разработке и эксплуатации информационных систем для управления аптечным делом. Это включает в себя проверку того, что система работает в соответствии с требованиями и стандартами, и что все функции и компоненты системы работают правильно и выполняют свои задачи.

Тестирование системы проводится для того, чтобы убедиться, что система соответствует требованиям и стандартам, и что она может выполнять все необходимые функции и операции. Это включает в себя проверку всех компонентов системы, включая интерфейс пользователя, базу данных, и программное обеспечение.

Подход к тестированию системы должен быть тщательным и детальным, с учетом всех возможных сценариев и потенциальных ошибок. Это требует использования специализированных методов и инструментов, таких как тестовые сценарии и тестовые данные.

Тестирование системы может проводиться на различных уровнях, начиная от тестирования отдельных компонентов системы и заканчивая тестированием системы в целом. На каждом уровне тестирования имеются свои собственные цели и задачи, но общей целью является обеспечение того, что система работает правильно и выполняет свои задачи.

Тестирование системы позволяет выявить потенциальные ошибки и уязвимости в системе, и принять меры для их исправления. Это также позволяет улучшить общую качество и надежность системы, и убедиться, что она соответствует требованиям и стандартам.

Анализ результатов тестирования системы является важным этапом в тестировании. Он позволяет выявить потенциальные ошибки и уязвимости, и принять меры для их исправления. Это также позволяет улучшить общую качество и надежность системы, и убедиться, что она соответствует требованиям и стандартам.

1.5.1 Методология тестирования

Методология тестирования представляет собой совокупность принципов, методов и инструментов, используемых для проверки качества и функциональности информационных систем. Она включает в себя планирование, разработку и проведение тестирования, а также анализ и интерпретацию результатов.

Методология тестирования основывается на принципах проверки соответствия системы требованиям и стандартам, а также на принципах обнаружения и исправления ошибок. Она предполагает использование специализированных инструментов и методов, таких как тестовые сценарии и тестовые данные, для проверки различных аспектов системы, включая ее функциональность, производительность и безопасность.

Одной из ключевых составляющих методологии тестирования является тестовый процесс, который включает в себя планирование, разработку, проведение и анализ тестирования. Тестовый процесс должен быть тщательно спланирован и организован, чтобы обеспечить эффективное и качественное тестирование системы.

Методология тестирования также включает в себя использование различных тестовых методов, таких как функциональное тестирование, нефункциональное тестирование и приемочное тестирование. Каждый из этих методов имеет свои собственные цели и задачи, но все они направлены на проверку качества и функциональности системы.

В целом, методология тестирования является важным аспектом разработки и эксплуатации информационных систем. Она позволяет убедиться, что система работает правильно и выполняет свои задачи, и что она соответствует требованиям и стандартам. Это также позволяет улучшить общую качество и надежность системы, и убедиться, что она может выполнять все необходимые функции и операции.

1.5.2 Проведение функционального тестирования

Проведение функционального тестирования является важным этапом в разработке и эксплуатации информационных систем для управления аптечным делом. Это включает в себя проверку того, что система работает в соответствии с требованиями и стандартами, и что все функции и компоненты системы работают правильно и выполняют свои задачи.

Функциональное тестирование проводится для того, чтобы убедиться, что система соответствует требованиям и стандартам, и что она может выполнять все необходимые функции и операции. Это включает в себя проверку всех компонентов системы, включая интерфейс пользователя, базу данных, и программное обеспечение.

Проведение функционального тестирования требует использования специализированных методов и инструментов, таких как тестовые сценарии и тестовые данные. Тестовые сценарии представляют собой набор шагов, которые необходимо выполнить, чтобы проверить функцию или компонент системы, в то время как тестовые данные представляют собой фиктивные данные, которые используются для проверки системы.

Функциональное тестирование может проводиться на различных уровнях, включая тестирование отдельных компонентов системы, тестирование интеграции компонентов, и тестирование системы в целом. На каждом уровне тестирования имеются свои собственные цели и задачи, но общей целью является обеспечение того, что система работает правильно и выполняет свои задачи.

Проведение функционального тестирования позволяет выявить потенциальные ошибки и уязвимости в системе, и принять меры для их исправления. Это также позволяет улучшить общую качество и надежность системы, и убедиться, что она соответствует требованиям и стандартам.

В целом, проведение функционального тестирования является важным этапом в разработке и эксплуатации информационных систем для управления аптечным делом, и требует использования специализированных методов и инструментов. Оно позволяет убедиться, что система работает правильно и выполняет свои задачи, и что она соответствует требованиям и стандартам.

Проведение функционального тестирования также включает в себя анализ результатов тестирования и принятие мер для их улучшения. Это может включать в себя исправление ошибок и уязвимостей, улучшение производительности и скорости системы, и обеспечение того, что система соответствует требованиям и стандартам.

1.5.3 Обработка ошибок и тестирование безопасности

Обработка ошибок и тестирование безопасности является важным аспектом разработки и эксплуатации информационных систем для управления аптечным делом. Это включает в себя выявление и исправление ошибок и недостатков в программном обеспечении, а также проверку безопасности системы на соответствие требованиям и стандартам.

Для эффективной обработки ошибок и тестирования безопасности необходимо использовать специализированные методы и инструменты, такие как тестирование сценариев, анализ кода и отладка. Эти методы и инструменты позволяют выявить потенциальные ошибки и уязвимости в программном обеспечении, а также проверить безопасность системы на соответствие требованиям и стандартам.

Обработка ошибок включает в себя выявление и исправление ошибок и недостатков в программном обеспечении, а также анализ причин их возникновения. Это позволяет предотвратить повторное возникновение подобных ошибок и улучшить общую надежность и безопасность системы.

Тестирование безопасности включает в себя проверку безопасности системы на соответствие требованиям и стандартам, а также выявление потенциальных уязвимостей и ошибок. Это позволяет выявить потенциальные риски и принять меры для их минимизации, а также улучшить общую безопасность системы.

Для эффективного тестирования безопасности необходимо использовать специализированные инструменты и методы, такие как сканирование уязвимостей, анализ кода и пентестинг. Эти инструменты и методы позволяют выявить потенциальные уязвимости и ошибки в программном обеспечении, а также проверить безопасность системы на соответствие требованиям и стандартам.

Кроме того, обработка ошибок и тестирование безопасности должно быть осуществлено на регулярной основе, чтобы обеспечить постоянный контроль и улучшение безопасности системы.

ГЛАВА II. Практическая часть

2.1 Архитектура программного обеспечения

Архитектура программного обеспечения представляет собой фундаментальную организацию системы. Она включает структурные компоненты такие как модули, классы и сервисы, их взаимосвязи через интерфейсы и протоколы взаимодействия, а также принципы проектирования и развития системы.

Архитектура служит стратегическим каркасом, который определяет распределение функциональности между компонентами, способы их взаимодействия и глобальные решения, влияющие на всю систему в целом. Это набор ключевых архитектурных решений, которые сложно изменить после реализации, оказывают широкое воздействие на систему и определяют ее качественные характеристики, включая масштабируемость и надежность.

Архитектура рассматривается на разных уровнях - уровне всей системы, уровне отдельных компонентов и уровне организации исходного кода. Она характеризуется абстракцией ключевых элементов системы, согласованностью принимаемых решений, балансом между требованиями и ограничениями, а также долговечностью и способностью к эволюции.

По своей сути архитектура определяет структуру системы и взаимодействие ее компонентов, но не детали реализации. Она выступает в роли связующего звена между требованиями к системе и ее конечной реализацией, обеспечивая целостный подход к проектированию программного обеспечения.

2.1.1 Общая структура системы

Разработанная система основывается на классической трехуровневой архитектуре, которая давно зарекомендовала себя как надежное и понятное решение для построения современных приложений. Эта архитектурная модель предусматривает четкое разделение ответственности между различными компонентами системы, что позволяет обеспечить более удобное сопровождение и развитие проекта в будущем. В клиентской части, выполненной с использованием современных веб-технологий, реализована визуальная составляющая, которая отвечает за отображение данных и взаимодействие с пользователем. Такой подход делает интерфейс интуитивно понятным и приятным для восприятия, что способствует повышению удобства и эффективности работы пользователя. Серверный компонент, созданный на базе популярного движка Node.js в сочетании с фреймворком Express.js, выполняет функцию обработки бизнес-логики и управления данными, обеспечивая при этом высокую скорость отклика и масштабируемость системы. В качестве системы хранения данных выбрана SQLite — легкая, надежная и производительная база данных, которая отлично подходит для задач данного проекта благодаря своей простоте в настройке и эксплуатации, что также способствует снижению затрат времени и ресурсов на поддержку системы.

2.1.2 Технологический стек

Выбор технологий для реализации системы базировался на тщательном анализе требований к производительности, безопасности и удобству разработки. В результате был сделан акцент на использовании Node.js как платформы, которая славится своей высокой эффективностью при обработке многочисленных запросов и возможностью асинхронной работы, что в совокупности обеспечивает отзывчивость и стабильность системы. Использование фреймворка Express.js значительно упростило организацию маршрутизации и обработку HTTP-запросов, что позволило быстрее реализовать необходимые функции и сделать архитектуру более понятной и модульной. Для работы с данными выбрана SQLite — легкая база данных, которая не требует установки сложных серверных решений и обеспечивает надежное хранение информации. В области безопасности реализована аутентификация с помощью express-session, что гарантирует надежное управление пользовательскими сессиями и предотвращает несанкционированный доступ к системе. В целом, выбранный стек технологий позволяет добиться баланса между высокой производительностью, безопасностью и удобством поддержки, что важно для успешной реализации проекта.

2.1.3 Взаимодействие компонентов

Обеспечение взаимодействия между клиентской и серверной частями системы построено на использовании REST API — архитектурного стиля, который обеспечивает гибкость, масштабируемость и возможность дальнейшего расширения системы. Клиентская часть, реализованная с применением современных технологий, использует fetch-запросы для обмена данными с сервером. Такой подход позволяет обновлять содержимое интерфейса в реальном времени без необходимости перезагружать страницу, что существенно повышает удобство и скорость работы системы. Передаваемые данные оформляются в универсальном формате JSON, что делает их легко обрабатываемыми как на стороне сервера, так и на стороне клиента. Такой обмен данными способствует созданию системы, которая легко расширяется и адаптируется под новые требования, а также обеспечивает стабильную работу и высокую отзывчивость.

2.2 Проектирование базы данных

Проектирование базы данных — это процесс создания структуры для хранения и организации данных, который обеспечивает эффективную работу информационной системы. Оно включает анализ требований, определение сущностей и их взаимосвязей, а также разработку схемы базы данных.

Основная цель проектирования — создать базу данных, которая будет соответствовать потребностям системы, обеспечивать целостность данных и высокую производительность при обработке запросов. Процесс начинается с концептуального проектирования, где определяются основные сущности и их атрибуты без привязки к конкретной СУБД. Затем выполняется логическое проектирование, в ходе которого создается детализированная схема с учетом правил нормализации для минимизации избыточности данных. На этапе физического проектирования выбираются конкретные типы данных, индексы и оптимизируется структура для выбранной системы управления базами данных.

Важной частью проектирования является обеспечение целостности данных через механизмы первичных и внешних ключей, а также настройка ограничений для проверки корректности информации. Результатом становится схема базы данных, которая поддерживает все необходимые операции — добавление, изменение, удаление и поиск информации — с требуемой скоростью и надежностью.

Проектирование учитывает как текущие требования системы, так и возможные изменения в будущем, что позволяет создавать гибкие и масштабируемые решения.

2.2.1 Принципы проектирования

При разработке структуры базы данных особое внимание уделялось принципам нормализации, которые помогают организовать данные так, чтобы минимизировать дублирование и обеспечить их целостность. В результате этого подхода система становится более управляемой, а данные — более точными и надежными. Важным аспектом также стало создание гибкой и расширяемой схемы, которая позволяет вносить новые функции и добавлять дополнительные таблицы без существенных изменений в уже существующей структуре. Такой подход обеспечивает долговременную работоспособность системы, а также дает возможность легко адаптировать ее к новым задачам и требованиям, возникающим в процессе эксплуатации.

2.2.2 Проектирование логической базы данных

Для эффективного управления процессами аптечного пункта в разрабатываемом Web-приложении была спроектирована логическая модель базы данных. Она включает основные сущности, отражающие реальные объекты бизнес-процессов, и их атрибуты.

Пользователи(Users) – Хранит информацию о сотрудниках системы (администраторы, кассиры, фармацевты). Включает логин, зашифрованный пароль, роль и дату создания.

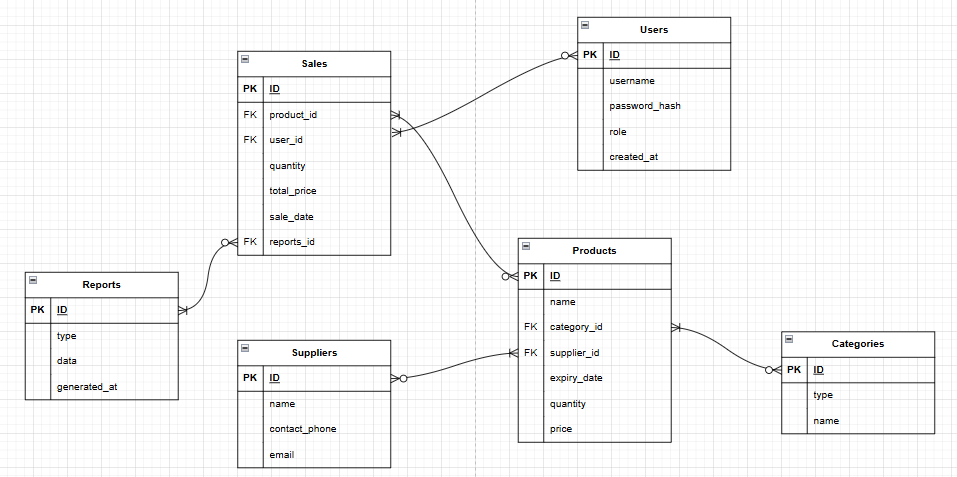
Категории лекарств (Categories) – Используются для классификации медикаментов по типам (например, Витамины, Анальгетики, Антибиотики). Содержит тип и уникальное имя категории.

Поставщики (Suppliers) – Содержит сведения о поставщиках медикаментов: наименование, контактные данные, адрес электронной почты.

Товары / Лекарства (Products) – Определяет ассортимент препаратов, доступных в аптеке. Каждый товар связан с категорией и поставщиком, имеет срок годности, цену и количество на складе.

Продажи (Sales) – Фиксирует факты продаж: проданный товар, пользователь, осуществивший продажу, количество, итоговую цену и дату продажи. Поддерживает расчет total\_price.

Отчёты (Reports) – Служит для хранения агрегированной информации: статистика продаж, остатки товаров, просроченные препараты и т.п. Хранит тип отчета, его данные в формате JSON и дату генерации.

Между сущностями установлены связи с использованием первичных и внешних ключей, что обеспечивает целостность данных и поддерживает бизнес-логику работы приложения. Например, пользователи привязаны к определённой роли, а продажи могут содержать множество товарных позиций.

Картинка 2 Схема логической базы данных

Такое проектирование логической модели позволяет гибко расширять функциональность приложения без нарушения структуры данных и обеспечивает высокую производительность операций чтения и записи.

2.2.3 Связи между таблицами

Организация связей между таблицами осуществляется при помощи механизмов первичных и внешних ключей, что обеспечивает поддержание целостности данных при выполнении различных операций — добавлении, изменении или удалении информации. Такой подход помогает избежать ошибок и противоречий, создавая устойчивую и надежную структуру базы данных. Благодаря правильно организованным связям, система становится более стабильной и готовой к масштабированию, а управление данными — более простым и понятным. Это важный аспект, который обеспечивает надежную работу всей системы в целом и дает возможность легко внедрять новые функции и расширять существующий функционал.

2.2.4 Создание физической базы данных

Физическая модель базы данных разработана на основе логической модели и реализована с помощью языка структурированных запросов SQL. Для реализации выбрана реляционная СУБД SQLite, которая обеспечивает надёжность, масштабируемость и высокую скорость обработки запросов.

Таблица 1 - «Users»

|  |  |  |
| --- | --- | --- |
| №  п/п | Атрибуты | Тип данных |
| 1 | ID | INT(PK) |
| 2 | Username | Text |
| 3 | Password\_hash | Text |
| 4 | email | Text |
| 5 | role | Text |
| 6 | created\_at | Datetime |

Таблица Users хранит данные сотрудников аптеки с разграничением прав доступа и связана с сущностью Sales по атрибуту id. Ключевые атрибуты включают username для входа в систему, password\_hash для хранения пароля и role определяющий уровень доступа.

Таблица 2 - «Categories»

|  |  |  |
| --- | --- | --- |
| №  п/п | Атрибуты | Тип данных |
| 1 | ID | INT(PK) |
| 2 | Name | Text |
| 3 | Type | Text |

Таблица Categories классифицирует лекарственные препараты по типу отпуска и связана с сущностью Products по атрибуту id, содержа такие важные атрибуты как type для разделения на рецептурные и безрецептурные препараты и name с названием категории.

Таблица 3 - «Suppliers»

|  |  |  |
| --- | --- | --- |
| №  п/п | Атрибуты | Тип данных |
| 1 | ID | INT(PK) |
| 2 | Name | Text |
| 3 | Contact\_phone | Text |
| 4 | Email | Text |

Таблица Suppliers учитывает организации-поставщиков лекарственных средств и связана с сущностью Products по атрибуту id, где основными атрибутами являются contact\_phone для связи с поставщиком и email.

Таблица 4 - «Product»

|  |  |  |
| --- | --- | --- |
| №  п/п | Атрибуты | Тип данных |
| 1 | ID | INT(PK) |
| 2 | Name | Text |
| 3 | Category\_id | INT(PK) |
| 4 | Expiry\_date | Date |
| 5 | Price | Real |
| 6 | Quantity | INT |
| 7 | Supplier\_id | INT(FK) |
| 8 | Created\_at | Datetime |

Таблица Products ведет учет лекарственных препаратов на складе аптеки, будучи связанной с сущностями Categories, Suppliers и Sales. Важнейшие атрибуты включают expiry\_date для контроля сроков годности, quantity для отслеживания остатков на складе и price указывающий стоимость препарата.

Таблица 5 - «Sales»

|  |  |  |
| --- | --- | --- |
| №  п/п | Атрибуты | Тип данных |
| 1 | ID | INT(PK) |
| 2 | Product\_id | INT(FK) |
| 3 | Contact\_phone | Text |
| 4 | Email | Text |
| 5 | User\_id | INT(FK) |
| 6 | Sale\_date | Datetime |

Таблица Sales фиксирует факты продаж лекарственных препаратов и связана с сущностями Products и Users, содержа ключевые атрибуты total\_price с автоматически рассчитываемой суммой продажи и sale\_date с датой совершения операции.

Таблица 6 - «Reports»

|  |  |  |
| --- | --- | --- |
| №  п/п | Атрибуты | Тип данных |
| 1 | ID | INT(PK) |
| 2 | Type | INT(FK) |
| 3 | Data | Text(JSON) |
| 4 | Generated\_at | Datetime |

Таблица Reports формирует аналитические отчеты о работе аптеки, используя данные из таблиц Sales и Products. Основные атрибуты включают type определяющий тип отчета, data содержащий результаты в формате JSON и generated\_at с датой формирования отчета.

Все таблицы связаны между собой через систему внешних ключей что обеспечивает целостность данных, а для оптимизации производительности созданы индексы на часто используемые поля.

Реализация физической модели обеспечивает целостность данных, минимизирует избыточность и повышает производительность системы. Структура базы данных ориентирована на расширяемость, что позволяет в дальнейшем дополнять приложение новыми функциями без необходимости серьёзных изменений в архитектуре данных.

2.3 Пользовательский интерфейс

Пользовательский интерфейс — это часть системы, через которую происходит взаимодействие человека с программой или устройством. Он включает все элементы, позволяющие пользователю вводить данные, управлять системой и получать результаты работы. Интерфейс строится на основе анализа потребностей и поведения пользователей, чтобы сделать взаимодействие интуитивно понятным и эффективным.

Хороший интерфейс учитывает принципы удобства использования, обеспечивая логичную структуру, последовательность действий и быстрый доступ к основным функциям. Визуальные компоненты располагаются в соответствии с приоритетами задач, а элементы управления делаются узнаваемыми и предсказуемыми в работе. Современные интерфейсы адаптируются под разные устройства и условия использования, сохраняя функциональность на всех платформах.

Качество интерфейса напрямую влияет на продуктивность пользователей, сокращая время на обучение и уменьшая количество ошибок при работе с системой. Он служит посредником между технической сложностью программы и простотой её повседневного использования, скрывая ненужные детали реализации и выделяя важные для пользователя функции.

2.3.1 Основные страницы

Интерфейс системы включает в себя ряд страниц, каждая из которых предназначена для выполнения своих задач и обеспечивает комфортное взаимодействие пользователя с системой. Страница авторизации служит воротами в систему, предоставляя безопасный и надежный механизм входа, что помогает защитить данные и функционал от несанкционированного доступа. После успешной авторизации пользователь попадает на главную панель управления, содержание которой динамически адаптируется в зависимости от роли и прав доступа, что обеспечивает персональный подход и повышает эффективность работы. В системе также предусмотрены разделы для просмотра и редактирования данных, выполненные в удобном и интуитивно понятном стиле, что значительно облегчает работу с системой даже для менее опытных пользователей. Такой подход способствует быстрому освоению интерфейса и снижает время обучения.

2.3.2 Адаптивность

Особое значение при создании пользовательского интерфейса имело его адаптивность — способность корректно и удобно функционировать на различных устройствах и экранах. В процессе разработки использовались современные методы верстки, такие как отзывчивый дизайн, который позволяет интерфейсу автоматически подстраиваться под размеры экрана и тип устройства. В результате, система хорошо отображается как на больших настольных мониторах, так и на мобильных телефонах и планшетах. Это обеспечивает пользователю возможность работать с системой в любом месте и в любое время без ущерба для удобства или функциональности. Такой подход делает систему универсальной и доступной, что особенно важно в современном мире, где использование различных устройств становится нормой.

2.4 Система аутентификации и авторизации

Система аутентификации и авторизации обеспечивает безопасный доступ к защищенным ресурсам. Аутентификация отвечает за подтверждение личности пользователя, проверяя предоставленные учетные данные, такие как логин и пароль, биометрические данные или аппаратные ключи. После успешной аутентификации система авторизации определяет уровень доступа пользователя, устанавливая какие действия и ресурсы ему разрешены в соответствии с назначенными правами.

Эти механизмы работают совместно для защиты данных и функциональности системы от несанкционированного доступа. Современные реализации используют шифрование для безопасного хранения и передачи учетных данных, а также поддерживают многофакторную аутентификацию для усиления защиты. Система ведет журнал событий входа для отслеживания активности пользователей и выявления подозрительных действий.

Гибкость настроек позволяет адаптировать уровни доступа под различные роли пользователей, обеспечивая принцип минимальных привилегий, когда каждый получает ровно те права, которые необходимы для его работы. Интеграция с другими компонентами системы гарантирует согласованную работу всех защитных механизмов при сохранении удобства для легитимных пользователей.

2.4.1 Механизм аутентификации

Механизм аутентификации в системе построен на базе библиотеки express-session, которая обеспечивает надежное управление пользовательскими сессиями и защиту данных. Важной частью является безопасное хранение паролей — они сохраняются в базе данных в виде обработанных с использованием современных алгоритмов хеширования, что значительно усложняет возможность их получения злоумышленниками. Такой подход способствует повышению уровня защиты системы и предотвращает возможные угрозы, связанные с утечкой данных. В целом, эта система обеспечивает надежную идентификацию пользователей, позволяя им безопасно входить в систему и получать только разрешенный доступ, что является важной основой для дальнейших механизмов безопасности.

2.4.2 Система ролей

В системе реализована гибкая и продуманная модель разграничения прав доступа, основанная на ролях пользователей. Каждая роль определяет набор прав и возможностей, доступных конкретному пользователю, что помогает организовать рабочий процесс так, чтобы он был максимально эффективным и безопасным. Например, пользователи с ролью поставщика могут добавлять новые товары, расширяя базу данных и управляя запасами, в то время как фармацевты имеют доступ к просмотру остатков и работе с ними, что соответствует их профессиональной деятельности. Такое разделение прав способствует предотвращению несанкционированных действий и обеспечению безопасности данных, а также помогает организовать работу так, чтобы каждый пользователь имел доступ только к тому функционалу, который ему необходим для выполнения своих задач. Это создает условия для эффективного и безопасного использования системы в рамках установленных ролей.

2.5 Реализация API

Реализация API представляет собой процесс создания программного интерфейса, который позволяет различным компонентам системы или отдельным приложениям взаимодействовать между собой. API определяет набор правил и протоколов для обмена данными, устанавливая четкие форматы запросов и ответов.

В процессе разработки API проектируются конечные точки (endpoints), которые соответствуют конкретным бизнес-процессам или операциям с данными. Каждая конечная точка обрабатывает определенный тип запроса, выполняет необходимые действия с данными и возвращает результат в согласованном формате, обычно JSON или XML.

API реализует механизмы аутентификации и авторизации для контроля доступа, а также обработку ошибок с понятными кодами состояний. Важной частью является документирование всех доступных методов, их параметров и примеров использования.

Современные API часто строятся по REST-архитектуре, которая использует стандартные HTTP-методы и обеспечивает масштабируемость системы. Качественно реализованный API обеспечивает надежное и безопасное взаимодействие между различными программными компонентами, сервисами или платформами.

2.5.1 Основные маршруты API и их назначение

Серверная часть системы организована вокруг набора API-интерфейсов, которые реализованы по принципам REST – современного, широко используемого подхода к созданию веб-сервисов. Среди них особое место занимает маршрут /api/auth, который отвечает за процессы аутентификации и управление сессиями пользователей, обеспечивая безопасный вход и выход из системы. Для работы с данными о лекарствах предназначен endpoint /api/medicines, который позволяет выполнять операции по добавлению, редактированию и получению информации о медикаментах, что значительно облегчает управление лекарственными средствами. Управление складскими запасами осуществляется через маршрут /api/inventory, обеспечивая своевременный учет и контроль остатков товаров. Такой набор API делает систему не только гибкой и расширяемой, но и легко интегрируемой с другими системами или внешними сервисами, что важно для поддержки и масштабирования проекта.

2.5.2 Формат запросов

Все взаимодействия с API осуществляются с использованием стандартных методов HTTP, таких как GET, POST, PUT и DELETE, что соответствует современным практикам веб-разработки. Передача данных происходит в формате JSON – универсальном и широко поддерживаемом формате, который легко обрабатывается как на серверной стороне, так и на клиентской. Такой подход обеспечивает высокую совместимость и возможность быстрого расширения функциональности системы, а также интеграции с внешними системами и сервисами. В результате, система становится более гибкой и адаптируемой, что важно для обеспечения долгосрочной эффективности и удобства эксплуатации.

2.6 Обработка ошибок и логирование

Обработка ошибок и логирование представляют собой важные механизмы обеспечения надежности и отказоустойчивости программных систем. Обработка ошибок включает выявление, классификацию и корректное реагирование на исключительные ситуации, возникающие при работе программы. Этот процесс предотвращает неожиданные завершения работы и обеспечивает стабильное функционирование системы даже при возникновении проблем.

Логирование дополняет обработку ошибок, фиксируя информацию о работе системы в хронологическом порядке. В журналы событий записываются как критические ошибки, так и штатные операции, что позволяет анализировать поведение системы в различных условиях. Собранные данные используются для диагностики проблем, аудита действий пользователей и анализа производительности.

Эффективная реализация этих механизмов предполагает многоуровневый подход, где учитывается важность событий и степень их влияния на работу системы. Техническая информация об ошибках структурируется и дополняется контекстными данными для упрощения анализа. Современные системы логирования поддерживают фильтрацию, ротацию журналов и интеграцию с инструментами мониторинга.

Грамотно организованные обработка ошибок и логирование сокращают время на поиск и устранение неисправностей, повышают стабильность работы системы и обеспечивают прозрачность происходящих процессов. Эти механизмы особенно важны в распределенных системах и сложных бизнес-приложениях, где необходимо оперативно реагировать на возникающие проблемы.

2.6.1 Типы ошибок

В системе реализован продвинутый механизм обработки ошибок, который охватывает широкий спектр возможных ситуаций. Важной частью этой системы является тщательная валидация входных данных, которая позволяет своевременно выявлять и предотвращать потенциальные проблемы еще на этапе их возникновения. Обработка ошибок доступа организована так, чтобы обеспечивать безопасность и предотвращать несанкционированные действия, что является важным аспектом защиты системы. В случае возникновения серверных ошибок, таких как сбои в работе или внутренние неисправности, они фиксируются и обрабатываются специальными модулями, что повышает стабильность и надежность системы. Такой комплексный подход помогает минимизировать влияние проблем на работу пользователей и обеспечивает возможность быстрого реагирования и устранения неисправностей.

2.6.2 Механизм логирования

Для повышения уровня надежности и обеспечения прозрачности работы системы внедрен механизм логирования, который регистрирует все важные события и действия. В специальные журналы записывается информация о каждом авторизационном событии, что позволяет проводить аудит активности пользователей и выявлять возможные нарушения или подозрительные действия. Кроме того, вся критическая информация о ошибках фиксируется с максимальной детализацией, что значительно облегчает диагностику и устранение неисправностей. Ведение учетных записей о действиях пользователей помогает не только обеспечить безопасность, но и анализировать эффективность работы системы, выявлять узкие места и реализовывать меры по их устранению. Такой подход способствует созданию надежной системы, которая способна не только эффективно функционировать, но и своевременно реагировать на возникающие проблемы, обеспечивая стабильную работу и безопасность для всех пользователей.

ЗАКЛЮЧЕНИЕ

В процессе выполнения дипломной работы была достигнута основная цель — разработано полнофункциональное web-приложение для аптечного пункта, обеспечивающее управление складскими запасами, контроль поступлений, учет продаж и реализацию роли пользователей с разграничением прав доступа. Система построена с использованием современных и надежных технологий, что позволяет ей быть адаптивной, масштабируемой и безопасной.

На первом этапе был проведен анализ предметной области, выявлены ключевые проблемы, с которыми сталкиваются сотрудники аптек при ручном учете и недостаточной автоматизации бизнес-процессов. Также были рассмотрены действующие нормативные акты и стандарты, регламентирующие хранение, реализацию и учет лекарственных средств, что позволило сформировать корректные требования к информационной системе.

В теоретической части работы были определены цели и задачи, охватывающие как архитектурные, так и функциональные аспекты. На основании требований был спроектирован интерфейс и разработана структура базы данных, отвечающая принципам нормализации и логической связности. При проектировании базы данных особое внимание было уделено учету партий медикаментов, сроков годности и взаимодействию с поставщиками, что критично для поддержания качества медицинской продукции.

На этапе проектирования реализована трехуровневая архитектура системы, включающая клиентскую часть (JavaScript), серверную логику (Node.js + Express.js) и уровень хранения данных (SQLite).

Система имеет модульную структуру и предоставляет гибкий REST API для взаимодействия компонентов. Это обеспечивает простоту поддержки, расширяемость и возможность интеграции с другими системами. Для отладки и сопровождения реализовано централизованное логирование, а также предусмотрены механизмы обработки ошибок с выдачей корректных HTTP-кодов.

Проведенное тестирование подтвердило работоспособность системы, ее устойчивость к типовым ошибкам, а также соответствие требованиям, сформулированным в начальных этапах разработки. Web-приложение обеспечивает удобный и интуитивно понятный интерфейс, сокращает временные затраты сотрудников на учет и контроль остатков, минимизирует человеческий фактор и повышает общую эффективность аптечного пункта.

Таким образом, разработанная система может быть внедрена в малые и средние аптечные организации и может стать основой для дальнейшего масштабирования — например, в рамках сети аптек, с возможностью интеграции с онлайн-кассами, сканерами штрих-кодов, автоматической отчетностью перед контролирующими органами и модулем финансовой аналитики.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Абрамова Н. А. Информационные технологии в управлении. — М.: Форум, 2020. — 384 с.
2. Босова Л. Л., Босова А. Ю. Информационные технологии. — М.: Бином, 2021. — 256 с.
3. Кузнецов С. В. Проектирование баз данных: Учебник. — СПб.: Питер, 2022. — 432 с.
4. Date C. J. Введение в системы баз данных. — М.: Вильямс, 2019. — 944 с.
5. Фролов С. И. Web-программирование на JavaScript и Node.js. — М.: ДМК Пресс, 2022. — 320 с.
6. Сивков С. Ю. Web-технологии. HTML, CSS, JavaScript. — М.: ДМК Пресс, 2021. — 384 с.
7. Войтович И. В. Node.js для профессионалов. — СПб.: БХВ-Петербург, 2021. — 416 с.
8. Радченко Д. С. REST API. Разработка веб-сервисов. — М.: ДМК Пресс, 2022. — 336 с.
9. Павлов А. Н. Технологии клиент-серверных приложений. — М.: Горячая линия – Телеком, 2019. — 280 с.
10. Краснов Н. Н. Проектирование интерфейсов: UX и UI в веб-приложениях. — М.: Эксмо, 2021. — 240 с.
11. Глушаков А. В. Современные базы данных: проектирование и оптимизация. — М.: Инфра-М, 2021. — 288 с.
12. Соловьёв В. Д. Безопасность информационных систем. — М.: Академия, 2020. — 400 с.
13. Кормушин А. И. Основы информационной безопасности. — М.: Солон-Пресс, 2020. — 288 с.
14. Емельянов М. В. Программная инженерия: основы и принципы. — СПб.: Питер, 2020. — 352 с.
15. Барабанов А. А. Информационные системы в здравоохранении. — М.: ГЭОТАР-Медиа, 2021. — 368 с.
16. Федеральный закон № 61-ФЗ «Об обращении лекарственных средств» от 12.04.2010 г.
17. Приказ Минздрава РФ от 31.08.2016 № 647н «Об утверждении правил хранения лекарственных препаратов».
18. ECMAScript® 2021 Language Specification. [Электронный ресурс] — Режим доступа: https://tc39.es/ecma262/
19. Node.js Documentation. [Электронный ресурс] — Режим доступа: https://nodejs.org/en/docs/
20. Open Web Application Security Project (OWASP). [Электронный ресурс] — Режим доступа: https://owasp.org
21. GitHub. Платформа для совместной разработки ПО. [Электронный ресурс] — Режим доступа: https://github.com/

22. Электронный портал «ГАРАНТ». Правовые акты в области фармацевтики. [Электронный ресурс] — Режим доступа: https://www.garant.ru/