Министерство образования и науки Краснодарского края

Государственное автономное профессиональное

образовательное учреждение Краснодарского края

«Лабинский аграрный техникум»

|  |  |
| --- | --- |
|  | УТВЕРЖДАЮ  Заместитель директора по  учебной работе    \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_О.А. Мезенцева  «\_\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_г. |

**Дипломный проект**

по специальности 09.02.07 «Информационные системы и программирование»

Разработка информационной системы по учету работы отдела АСУ в ГБУЗ "Лабинская центральная районная больница"

Руководитель\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_С.Д. Петров

подпись дата должность, ученая степень

Выпускник \_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_ А. Ю. Федоренко

подпись дата

|  |  |
| --- | --- |
| ДОПУЩЕН К ЗАЩИТЕ  Председатель УМО преподавателей компьютерных специальностей  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ И.П. Ефентьева  «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_20\_\_\_г. | НОРМОКОНТРОЛЬ  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_С.С. Сурков  «\_\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_г. |

Дата защиты «\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_г. Оценка \_\_\_\_ «\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_»

Протокол ГЭК № \_\_\_\_\_\_\_\_ от «\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_г.

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc760427236)

[Глава 1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ И РАЗРАБОТКА ТРЕБОВАНИЙ 6](#_Toc460362146)

[1.1 Общая характеристика ГБУЗ «Лабинская центральная районная больница» 6](#_Toc616130578)

[1.2 Описание функциональной структуры отдела АСУ 7](#_Toc1278096128)

[1.3 Анализ проблемных ситуаций и информационных потоков 9](#_Toc1813458764)

[1.4 Постановка задачи автоматизации учёта работы отдела АСУ 11](#_Toc1768526546)

[1.5 Формулирование требований к информационной системе 12](#_Toc1440023127)

[Выводы по главе 14](#_Toc1592139735)

[Глава 2. ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ 16](#_Toc1638065815)

[2.1 Выбор и обоснование технологий и программного обеспечения 16](#_Toc1259838928)

[2.2 Проектирование функциональной структуры системы 21](#_Toc1087538058)

[2.3 Разработка концептуальной и логической моделей базы данных 28](#_Toc246985923)

[Выводы по главе 34](#_Toc480568968)

[Глава 3. РЕАЛИЗАЦИЯ ИНФОРМАЦИОННОЙ СИСТЕМЫ 35](#_Toc1228368159)

[3.1 Реализация базы данных и SQL запросов 35](#_Toc581259559)

[3.2 Реализация клиентского приложения 39](#_Toc682512191)

[3.3 Руководство пользователя 46](#_Toc11441177)

[Выводы по главе 49](#_Toc686959944)

[ЗАКЛЮЧЕНИЕ 51](#_Toc1108894189)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ: 53](#_Toc1140152908)

**ВВЕДЕНИЕ**

Современные медучреждения нуждаются в грамотном управлении информационными ресурсами, чтобы отвечать растущим запросам к качеству и оперативности медицинской помощи. Автоматизация управленческих процессов в отделе автоматизированных систем управления (АСУ) ГБУЗ "Лабинская ЦРБ" даст возможность не только улучшить внутренние процессы, но и увеличит общую эффективность работы больницы, что особенно актуально в условиях ограниченных ресурсов.

Проведённый анализ текущей ситуации выявил, что существующая система учёта работы отдела АСУ построена преимущественно на ручных операциях. Сотрудники вынуждены вести параллельный учёт в электронных таблицах Excel и бумажных журналах, что неизбежно приводит к целому ряду системных проблем. Во-первых, значительная часть рабочего времени тратится на дублирование информации и составление отчётов вручную. Во-вторых, человеческий фактор при ручном вводе данных становится источником регулярных ошибок и неточностей. В-третьих, отсутствие единого информационного пространства затрудняет контроль выполнения задач и анализ эффективности работы подразделения.

Особенно сильно эти проблемы проявляются в следующих аспектах деятельности отдела:

* учёт и распределение поступающих заявок осуществляется без единой системы;
* отсутствует автоматизированный контроль сроков выполнения задач;
* формирование отчётности требует ручного сбора и обработки данных из различных источников.

Эти проблемы негативно влияет на общую эффективность работы не только самого отдела АСУ, но и учреждения в целом. Задержки в обработке заявок, трудности с планированием ресурсов - все это создает дополнительные организационные сложности в условиях и без того высокой нагрузки на медицинский персонал.

Разработка специализированной информационной системы для автоматизации учёта работы отдела АСУ позволит кардинально изменить текущее положение дел. Внедрение такого решения обеспечит:

* создание единого информационного пространства для управления задачами;
* автоматизацию процессов документооборота и отчётности;
* повышение прозрачности рабочих процессов;
* снижение временных затрат на рутинные операции;
* улучшение качества управленческих решений за счёт достоверной аналитик.

Целью дипломного проекта является разработка и внедрение информационной системы автоматизированного учёта работы отдела АСУ в ГБУЗ "Лабинская центральная районная больница", которая упростит управление задачами и ресурсами.

Задачи дипломного проекта:

1. провести анализ организационной структуры и бизнес-процессов отдела АСУ;
2. определить функциональные и нефункциональные требования к системе;
3. разработать концептуальную, логическую и физическую модели базы данных;
4. выбрать подходящие технологии для реализации системы;
5. реализовать программный продукт с учётом требований информационной безопасности;
6. подготовить инструкции для пользователей.

Объект исследования – процессы управления и учёта деятельности отдела автоматизированных систем управления в медицинском учреждении.

Предмет исследования – методы и средства автоматизации учёта рабочего времени, планирования задач и контроля выполнения работ в отделе АСУ.

Методами исследования являются системный анализ предметной области, методы проектирования информационных систем, а также современные технологии разработки программного обеспечения

Реализация разрабатываемой информационной системы позволит не только устранить существующие недостатки в работе отдела АСУ, но и создаст основу для дальнейшего развития цифровой инфраструктуры больницы. Автоматизация учёта задач сократит временные затраты на рутинные операции, минимизирует количество ошибок, вызванных человеческим фактором, и обеспечит руководителей достоверными аналитическими данными для принятия обоснованных управленческих решений.

**Глава 1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ И РАЗРАБОТКА ТРЕБОВАНИЙ**

1. **Общая характеристика ГБУЗ «Лабинская центральная районная больница»**

Государственное бюджетное учреждение здравоохранения «Лабинская центральная районная больница» (ГБУЗ «Лабинская ЦРБ» МЗ КК) является главным медицинским учреждением Лабинского района Краснодарского края. Больница была основана в 1989 году в результате реорганизации сети учреждений здравоохранения города Лабинска и района. С момента создания учреждение прошло несколько этапов преобразований, включая изменение организационно-правовой формы и наименования, и в 2018 году было переведено в государственную собственность Краснодарского края [7].

ГБУЗ «Лабинская ЦРБ» располагается по адресу: 352501, Краснодарский край, г. Лабинск, ул. Пирогова, д. 1. Медицинское учреждение является некоммерческой организацией, находящейся в ведении министерства здравоохранения Краснодарского края. Основной целью деятельности больницы является оказание квалифицированной медицинской помощи населению в соответствии с территориальной программой государственных гарантий.

Основные направления деятельности:

* оказание стационарной и амбулаторной медицинской помощи;
* проведение диагностических и лечебных процедур;
* фармацевтическая деятельность;
* использование источников ионизирующего излучения для медицинских целей.

Структура учреждения включает в себя множество отделений, включая терапевтическое, хирургическое, педиатрическое, а также специализированные службы, такие как отделение анестезиологии и реанимации, клинико-диагностическая лаборатория и другие. Важным структурным подразделением является отдел автоматизированных систем управления, который отвечает за внедрение, сопровождение и развитие информационных технологий в учреждении.

Больница оснащена современным медицинским оборудованием и информационными системами, которые позволяют эффективно осуществлять учёт пациентов, управлять медицинскими данными и оптимизировать административные процессы. Однако существующие системы учёта работы отдела АСУ требуют модернизации для повышения прозрачности, скорости обработки данных и минимизации ручного труда.

Деятельность учреждения финансируется за счёт средств краевого бюджета и обязательного медицинского страхования. Управление осуществляется главным врачом, который назначается министерством здравоохранения Краснодарского края.

Вследствие этого, ГБУЗ «Лабинская ЦРБ» представляет собой многопрофильное медицинское учреждение, деятельность которого требует эффективного управления информационными ресурсами, что делает актуальной разработку специализированной информационной системы для отдела АСУ.

1. **Описание функциональной структуры отдела АСУ**

Отдел АСУ играет важную роль в обеспечении эффективной работы медицинского учреждения, отвечая за внедрение, сопровождение и постоянное совершенствование информационных технологий. Основной целью деятельности отдела является автоматизация главных процессов больницы, включая медицинские услуги, административное управление и документооборот.

В сфере разработки и внедрения ИТ-продуктов специалисты отдела занимаются созданием и адаптацией программного обеспечения, предназначенного для учёта пациентов, управления медицинскими услугами, кадрового делопроизводства и финансовой отчётности. Важной частью работы является интеграция внутренних систем с внешними платформами, такими как ЕГИСЗ и системы электронного документооборота, что позволяет обеспечить согласованность данных и упростить обмен информацией.

Техническая поддержка и обслуживание включают в себя обеспечение бесперебойной работы серверного оборудования, рабочих станций и периферийных устройств. Сотрудники отдела настраивают и администрируют локальную сеть, следят за информационной безопасностью и оперативно устраняют возникающие неполадки в программном обеспечении и технике.

Управление базами данных является ещё одним важным направлением. Отдел занимается разработкой, оптимизацией и сопровождением баз данных медицинской информационной системы, а также обеспечивает регулярное резервное копирование и восстановление информации для предотвращения потери данных.

Автоматизация документооборота позволяет ускорить обработку внутренней и внешней документации за счёт внедрения специализированных систем электронного документооборота. Дополнительно отдел разрабатывает отчётные формы для контролирующих органов, что упрощает процесс предоставления необходимой документации.

Важной составляющей работы отдела является обучение и консультации персонала. Сотрудники проводят инструктажи по работе с новыми программными продуктами, а также разрабатывают методические материалы и инструкции для удобства пользователей.

Несмотря на активное развитие информационных технологий, в работе отдела сохраняются некоторые сложности. Одной из основных проблем является ручной учет задач, что приводит к дублированию информации и потере времени. Отсутствие единой системы учёта работ снижает эффективность управления процессами.

Ещё одной трудностью является децентрализованное хранение документов. Часть отчётности до сих пор ведётся в Excel, что усложняет контроль, анализ и поиск необходимых данных.

Кроме того, многие процессы остаются недостаточно автоматизированными, что вынуждает сотрудников вручную вводить значительные объёмы информации. Это увеличивает вероятность ошибок и замедляет выполнение задач.

Для повышения эффективности работы отдела необходимо дальнейшее внедрение современных ИТ-продуктов, которые позволят оптимизировать учёт задач, централизовать хранение документов и сократить долю ручного труда в повседневных операциях.

1. **Анализ проблемных ситуаций и информационных потоков**

Проведённое исследование деятельности отдела автоматизированных систем управления выявило ряд системных проблем, существенно снижающих эффективность работы подразделения. Сложившаяся ситуация требует детального рассмотрения и скорейшего принятия мер по оптимизации ключевых процессов.

Основные проблемные области:

* в сфере учёта выполняемых работ наблюдается серьёзный организационный пробел. Отсутствие единой платформы для фиксации заявок и контроля выполнения задач приводит к дублированию информации в разрозненных системах - электронных таблицах Excel и бумажных журналах. Такая практика не только увеличивает временные затраты сотрудников, но и создаёт сложности при оценке реальной загрузки специалистов и планировании работ;
* особую озабоченность вызывает состояние документооборота в отделе. Несмотря на частичную автоматизацию некоторых процессов, система согласования документов остаётся несовершенной. Сотрудники регулярно сталкиваются с проблемами поиска актуальных версий документов, а отсутствие чёткого механизма контроля исполнения поручений приводит к сбоям в рабочих процессах;
* серьёзным риском для организации является децентрализованный подход к хранению данных. Критически важная информация распределена по локальным компьютерам сотрудников без создания единого защищённого репозитория. Такая практика не только затрудняет доступ к необходимым данным, но и создаёт реальную угрозу их потери при технических сбоях;
* процесс формирования отчётности также требует доработки. Необходимость ручного сбора и объединения данных из различных источников приводит к значительным временным затратам и повышает вероятность ошибок при подготовке регулярных отчётов.

В работе отдела можно выделить три основных направления информационных потоков. Входящие потоки включают запросы на техническую поддержку, требования по доработке систем, нормативные документы и данные оборудования. Внутренние процессы охватывают распределение задач, обмен технической документацией, мониторинг выполнения работ и учёт возникающих инцидентов. Исходящие потоки представляют отчётную документацию, технические инструкции и аналитические материалы.

Основные проблемы в организации информационных потоков связаны с преобладанием неформальных каналов коммуникации, отсутствием единых регламентов работы с данными, необходимостью многократного ввода одинаковой информации в разные системы и задержками при передаче сведений между подразделениями.

Выявленные проблемы имеют комплексный негативный эффект. На организационном уровне это проявляется в низкой прозрачности процессов, сложностях управления нагрузкой персонала и задержках выполнения важных задач. Экономические последствия включают потери рабочего времени, необходимость содержания избыточного штата и финансовые риски из-за ошибок учёта. Технологические ограничения выражаются в трудностях получения актуальной информации, масштабирования решений и интеграции новых систем.

Проведённый анализ однозначно свидетельствует о необходимости внедрения специализированной информационной системы. Такое решение позволит формализовать ключевые бизнес-процессы, централизовать управление информационными потоками, повысить эффективность работы сотрудников и улучшить качество управленческих решений. Реализация этого проекта должна стать приоритетным направлением в развитии ИТ-инфраструктуры учреждения.

1. **Постановка задачи автоматизации учёта работы отдела АСУ**

Анализ текущих сложностей и информационных процессов отдела АСУ ГБУЗ "Лабинская ЦРБ" выявил необходимость внедрения специализированного программного решения для систематизации учёта деятельности подразделения.

Основные цели внедрения:

1. формирование централизованной среды для координации задач отдела АСУ;
2. улучшение процедур обработки обращений и работы с документами;
3. повышение открытости и управляемости операционных процессов;
4. сокращение временных затрат на выполнение стандартных операций на 30–40%.

Функциональные требования к автоматизированной системе управления:

1. Модуль управление заявками:

Система должна обеспечивать комплексное управление заявками, включая их регистрацию с обязательной классификацией по приоритетам и статусам выполнения. Для каждой заявки необходимо предусмотреть назначение ответственного исполнителя, контроль сроков выполнения (от момента создания до закрытия), а также возможность ведения обсуждений через систему комментариев.

1. Модуль управления пользователями:

Требуется реализовать гибкую систему разграничения прав доступа на основе ролей. Система должна позволять управлять учётными записями сотрудников и их доступом, обеспечивая дифференцированный доступ к функционалу в зависимости от должностных обязанностей.

1. Отчётный модуль:

Автоматизированный механизм формирования отчётности должен включать генерацию отчётов с возможностью их экспорта в форматы офисных документов и вывода на печать.

**Базовые элементы системы:**

* учётные записи сотрудников и их роли;
* заявки с параметрами (стадии выполнения, важность);
* инструменты комментирования и отчётности.

1. Административный модуль:

Предназначен для управления системой. Он позволяет регистрировать новых пользователей, настраивать их права доступа, управлять структурными подразделениями и настраивать системные параметры. Администратор может изменять перечень возможных статусов заявок, настраивать приоритеты и категории задач.

1. **Формулирование требований к информационной системе**

По результатам изучения текущих процессов и задач автоматизации определены ключевые требования к создаваемому программному решению.

Основные функциональные возможности:

Система должна предоставлять инструменты для управления заявками и задачами, включая регистрацию новых заявок с обязательным указанием названия, описания, статуса и приоритета. Должна быть реализована возможность назначения исполнителей, контроля сроков выполнения, а также изменения статусов заявок (например, "в работе", "выполнено" и другие). Для удобства пользователей система должна поддерживать поиск заявок и ведение истории обсуждений через комментарии.

Управление пользователями и правами доступа предполагает разграничение ролей: администратор, сотрудник отдела АСУ и обычный пользователь. Каждый зарегистрированный пользователь получает личный профиль, в котором отображается история связанных с ним заявок.

Для оценки эффективности работы система должна автоматически генерировать отчётные формы с возможностью выгрузки в документы формата Word.

Технические аспекты:

Решение должно стабильно функционировать при нагрузке до 50+ одновременных пользователей. Время обработки запросов — не более 2 секунд, а суточный объем новых обращений — до 100. Обязательна многоуровневая проверка подлинности пользователей при входе.

Требования к пользовательскому интерфейсу:

Интерфейс системы должен быть интуитивно понятным и полностью локализованным на русский язык. Основные разделы включают:

* перечень заявок с возможностью поиска;
* форма добавления и корректировки заявок;
* персональный раздел с историей выполненных заявок;
* блок отчётности;
* панель администрирования для настройки системы.

Надёжность и безопасность:

Система должна быть отказоустойчивой: в случае сбоев необходимо автоматическое сохранение данных. Также требуется обеспечить защиту конфиденциальной информации за счёт разграничения прав доступа.

Требования к сопровождению:

Для упрощения администрирования в систему интегрируется управляющая панель с гибкими настройками прав. Дополнительно разрабатывается подробное руководство с описанием функций и пошаговыми инструкциями.

**Выводы по главе**

Анализ текущей деятельности отдела АСУ ГБУЗ "Лабинская ЦРБ" выявил ряд системных проблем, негативно влияющих на операционную эффективность подразделения. Наиболее существенные трудности обусловлены фрагментированным учётом рабочих задач, распределенным хранением информации, необходимостью постоянного ручного ввода данных и недостаточным уровнем автоматизации документооборота. Такая ситуация неизбежно ведёт к снижению управляемости процессов, неоправданным временным потерям и возрастанию количества ошибок.

По результатам проведённого исследования определены ключевые направления автоматизации, среди которых:

* формирование унифицированной информационной среды;
* совершенствование механизмов обработки заявок;
* усиление мониторинга рабочих процессов;
* сокращение объёма рутинных операций.

Разработанный перечень требований охватывает функциональные характеристики системы, технические параметры, особенности пользовательского интерфейса, а также аспекты надёжности.

Внедрение предлагаемого решения обеспечит:

* единый центр управления задачами и запросами;
* значительное сокращение ручных операций;
* ускорение обработки служебной информации;
* повышение качества контроля исполнения поручений;
* гарантированную защиту данных с дифференцированным доступом.

Реализация автоматизированной системы учёта представляет собой стратегически важную инициативу, направленную на совершенствование управления ИТ-инфраструктурой медицинского учреждения. Сформулированные требования создают прочный фундамент для разработки программного обеспечения, которое принципиально изменит подход к организации рабочих процессов и существенно повысит качество сервисного обслуживания.

**Глава 2. ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ**

1. **Выбор и обоснование технологий и программного обеспечения**

При разработке информационной системы учёта работы отдела АСУ для медицинского учреждения важное значение играет правильный выбор технологического стека. Этот выбор определяет не только текущую функциональность системы, но и возможности её дальнейшего развития, простоту поддержки и надёжность работы. В условиях медицинского учреждения особенно важны стабильность работы, безопасность данных и простота использования, так как система будет эксплуатироваться персоналом без глубоких технических знаний.

Технологический стек должен обеспечивать выполнение всех функциональных требований, предъявляемых к системе, включая управление заявками, формирование отчётов и разграничение прав доступа. При этом важно учитывать существующую ИТ-инфраструктуру учреждения, которая в большинстве медицинских организаций построена на решениях Microsoft. Дополнительными факторами выбора являются доступность специалистов для поддержки системы и соответствие бюджетным ограничениям проекта.

Основой для клиентской части системы был выбран язык программирования C# в сочетании с Windows Forms. C# — это кроссплатформенный язык общего назначения, который делает разработчиков продуктивным при написании высокопроизводительного кода. С миллионами разработчиков C# является самым популярным языком .NET. C# имеет широкую поддержку в экосистеме и для всех рабочих нагрузок .NET. Основанный на объектно-ориентированных принципах, он включает множество функций из других парадигм, включая функциональное программирование. Низкоуровневые функции поддерживают сценарии высокой эффективности без написания небезопасного кода. Большая часть среды выполнения и библиотек .NET написана на C#, и прогресс в C# часто используется для всех разработчиков .NET [26].

Windows Forms была выбрана в качестве технологии для пользовательского интерфейса благодаря своей простоте разработки и естественной интеграции с операционной системой Windows. Windows Forms — это платформа пользовательского интерфейса для создания классических приложений Windows. Она обеспечивает один из самых продуктивных способов создания настольных приложений на основе визуального конструктора, предоставленного в Visual Studio. Функциональность, такая как размещение визуальных элементов управления методом перетаскивания, упрощает создание настольных приложений [27].

В качестве основной среды разработки выбрана Microsoft Visual Studio Community 2022 (рис. 2.1). Visual Studio — это мощное средство разработчика, которое можно использовать для выполнения всего цикла разработки в одном месте. Это комплексная интегрированная среда разработки (IDE), которую можно использовать для записи, редактирования, отладки и сборки кода. Затем разверните приложение. Visual Studio включает компиляторы, средства завершения кода, управление версиями, расширения и многие другие функции для улучшения каждого этапа процесса разработки программного обеспечения [28].

Её преимущества для данного проекта включают:

* полноценную поддержку разработки на C# и Windows Forms;
* встроенные инструменты для работы с базами данных;
* интегрированную систему контроля версий Git;
* богатый набор средств отладки и профилирования;
* возможность использования расширений через NuGet;
* бесплатную лицензию для образовательных и некоммерческих проектов.

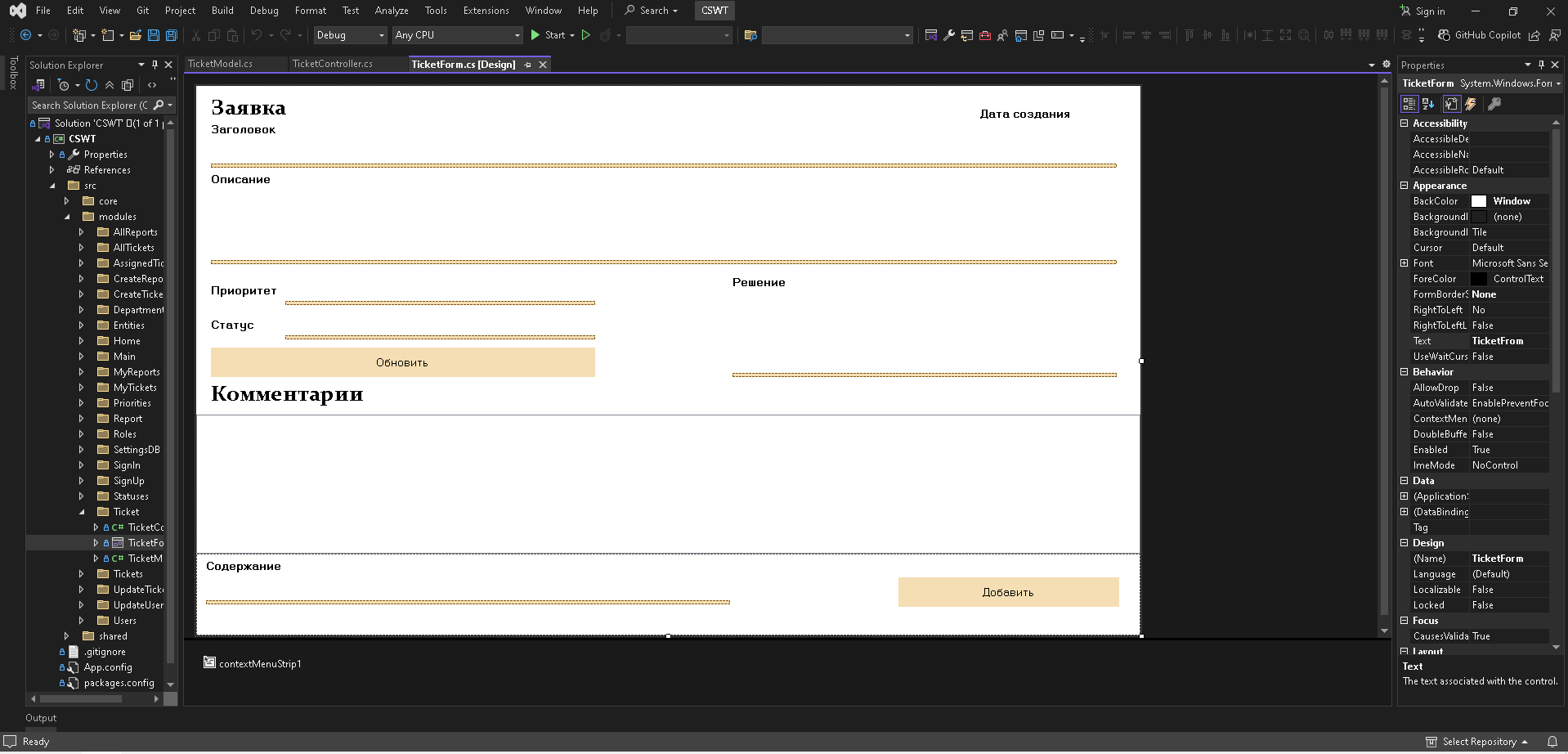


Рис. 2.1 - Интерфейс Visual Studio

Для хранения данных была выбрана система управления базами данных PostgreSQL 15.

База данных (БД) представляет собой совокупность специальным образом организованных данных, хранимых в памяти вычислительной системы и отображающих состояние объектов, и их взаимосвязей в рассматриваемой предметной области [23].

Система управления базами данных (СУБД) — это комплекс языковых и программных средств, предназначенный для создания, ведения и совместного использования БД многими пользователями. Обычно СУБД различают по используемой модели данных. Так, СУБД, основанные на использовании реляционной модели данных, называют реляционными СУБД [23].

PostgreSQL - это объектно-реляционная система управления базами данных (ORDBMS), основанная на Postgres, версия 4.2, разработанная в Калифорнийском университете на факультете компьютерных наук Беркли. Postgres впервые зарегистрировал многие концепции, которые стали доступны только в некоторых коммерческих системах баз данных намного позже. Эта СУБД сочетает в себе надёжность коммерческих решений с открытостью исходного кода [24]. PostgreSQL полностью поддерживает ACID-транзакции, что критически важно для системы учёта заявок, где необходимо гарантировать целостность данных. Встроенные механизмы репликации обеспечивают отказоустойчивость системы, а поддержка JSON позволяет гибко хранить структурированные данные отчётов. Для работы с СУБД используется pgAdmin4 инструмент с графическим интерфейсом для управления базами данных PostgreSQL (рис. 2.2).

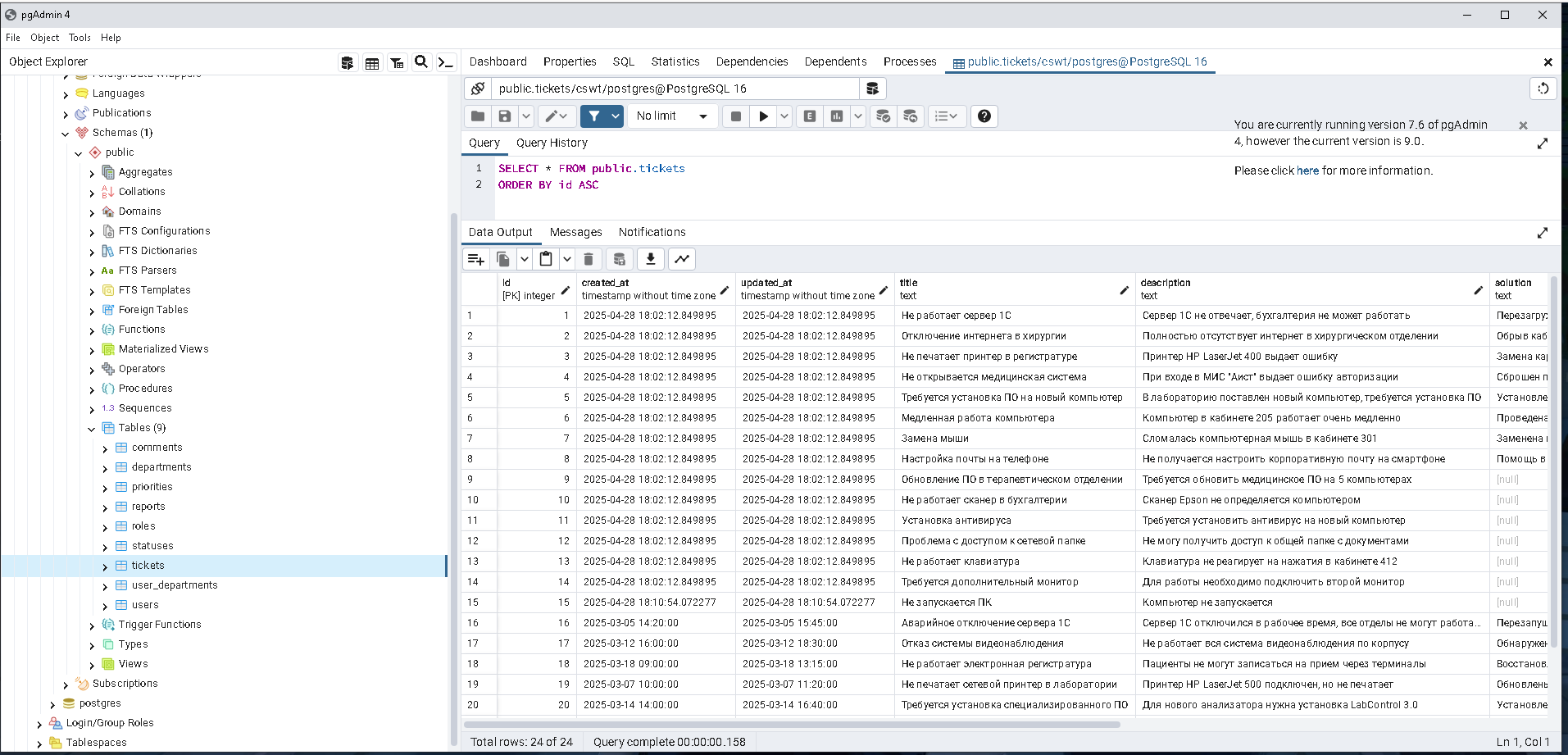


Рис. 2.2 - Интерфейс pgAdmin4

Взаимодействие между клиентским приложением и базой данных осуществляется через драйвер Npgsql. Npgsql является поставщиком данных с открытым исходным кодом ADO.NET для PostgreSQL, он позволяет программам, написанным в C#, Visual Basic, F#, чтобы получить доступ к серверу баз данных PostGRESQL. Он реализован в 100% C# код, бесплатный и является открытым исходным кодом [29]. Этот драйвер обеспечивает высокую производительность и поддерживает все возможности PostgreSQL, включая асинхронные операции, что важно для обеспечения отзывчивости интерфейса при работе с большими объёмами данных.

Система построена по классической трехуровневой архитектуре, что позволяет чётко разделить ответственность между компонентами (рис. 2.3). Уровень представления реализован на Windows Forms и отвечает за взаимодействие с пользователем. Бизнес-логика системы вынесена в отдельный слой, что упрощает её тестирование и модификацию. Доступ к данным осуществляется через слой репозиториев, который абстрагирует работу с базой данных.

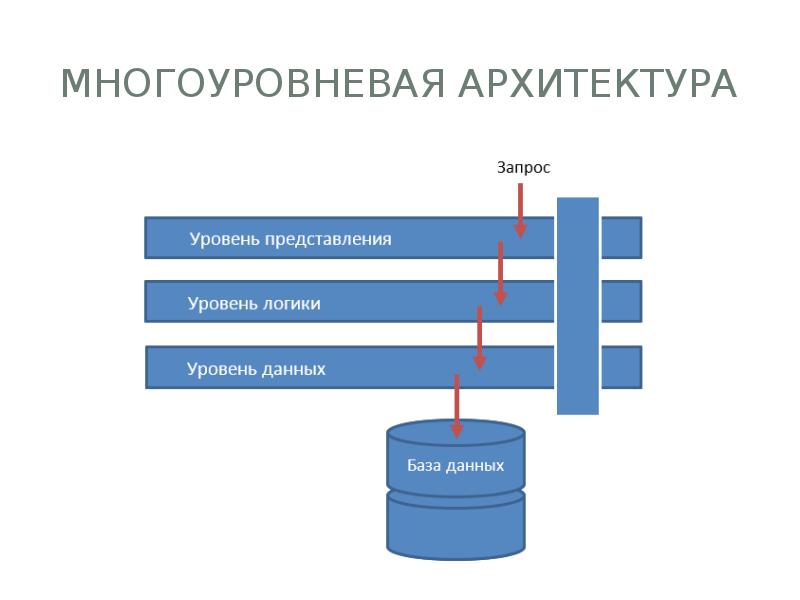


Рис. 2.3 - Многоуровневая архитектура

Такой подход обеспечивает гибкость архитектуры и позволяет в будущем модифицировать отдельные компоненты системы без необходимости её полной переработки. Например, при необходимости добавления веб-интерфейса можно будет использовать существующий слой бизнес-логики, заменив только уровень представления.

Серверная часть системы развёртывается на выделенном оборудовании с учетом требований к производительности и надёжности. Минимальная конфигурация сервера включает многоядерный процессор, 8 ГБ оперативной памяти и SSD-накопитель для системы. Данные рекомендуется хранить на отдельном жестком диске с регулярным резервным копированием.

В качестве операционной системы сервера может использоваться как Windows, так и Linux, так как PostgreSQL поддерживает обе платформы. Для медицинских учреждений, где преобладает Windows-инфраструктура, часто предпочтительнее первый вариант из-за простоты администрирования.

Клиентские рабочие станции должны соответствовать современным требованиям для комфортной работы с приложением. Особое внимание уделяется стабильности сетевого соединения между клиентами и сервером базы данных, так как это напрямую влияет на скорость работы системы.

Выбранный технологический стек позволяет легко расширять функциональность системы в будущем. Одним из направлений развития может стать добавление веб-интерфейса, что позволит предоставить доступ к системе через браузер. Другим перспективным направлением является разработка мобильного приложения для оперативного управления заявками.

Выбранные технологии обеспечивают баланс между надежностью, производительностью и стоимостью владения, что особенно важно для бюджетных медицинских учреждений. Гибкость архитектуры позволяет адаптировать систему к изменяющимся требованиям без необходимости полной замены технологического стека.

1. **Проектирование функциональной структуры системы**

На основании проведённого анализа требований к системе учёта работы отдела АСУ была разработана комплексная функциональная архитектура, построенная по трехуровневому принципу. Такая архитектура обеспечивает чёткое разделение ответственности между компонентами системы и упрощает её дальнейшее развитие и поддержку.

Уровень представления реализован в виде Windows Forms приложения, которое обеспечивает взаимодействие с конечными пользователями системы. Этот уровень отвечает за отображение информации, сбор пользовательского ввода и обеспечение удобного интерфейса для работы с системой. Выбор Windows Forms обусловлен необходимостью создания привычного для медицинского персонала интерфейса с минимальным временем освоения.

Уровень бизнес-логики разработан на языке C# и содержит все основные алгоритмы обработки данных. Этот уровень включает классы, отвечающие за обработку заявок, управление пользователями, генерацию отчётов и другие ключевые функции системы. Бизнес-логика полностью отделена от уровня представления, что позволяет изменять интерфейсные решения без переработки основной функциональности.

Уровень данных построен на базе PostgreSQL 15 - современной реляционной СУБД с открытым исходным кодом. Этот уровень отвечает за надёжное хранение и эффективный поиск информации. PostgreSQL была выбрана благодаря её стабильности, производительности и соответствию требованиям проекта по обработке транзакций.

Система включает четыре ключевых функциональных модуля, каждый из которых решает определённый круг задач:

1. Модуль авторизации и регистрации обеспечивает безопасный доступ к системе. Он реализует ролевую модель управления правами, где каждому пользователю назначается определённая роль (администратор, сотрудник отдела АСУ, клиент).
2. Модуль управления заявками является центральным компонентом системы. Он позволяет создавать новые заявки, назначать их конкретным исполнителям, изменять статусы выполнения и добавлять комментарии по ходу работы. Каждая заявка содержит информацию о приоритете выполнения и сроке обработки. На рисунке 2.4 представлена схема жизненного цикла заявки, начиная от момента её создания до окончательного закрытия.

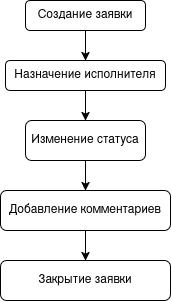


Рис. 2.4 - Жизненный цикл заявки

1. Модуль отчётности предоставляет инструменты для анализа работы отдела АСУ. Он позволяет генерировать стандартные отчёты. Особенностью модуля является возможность экспорта отчётов в формат DOCX для их дальнейшего использования в официальной документации. Отчёты формируются на основе актуальных данных и могут быть настроены в соответствии с потребностями руководства.
2. Административный модуль предназначен для управления системой. Он позволяет регистрировать новых пользователей, настраивать их права доступа, управлять структурными подразделениями и настраивать системные параметры. Администратор может изменять перечень возможных статусов заявок, настраивать приоритеты и категории задач.

Функционал работы с заявками включает полный цикл управления задачами:

* создание новых заявок с указанием всех необходимых параметров;
* редактирование существующих заявок;
* закрытие выполненных заявок с указанием решения;
* назначение приоритетов (низкий, средний, высокий);
* изменение статусов выполнения (новая, в работе, на проверке, выполнена);

Функционал управления пользователями обеспечивает:

* регистрацию новых пользователей с обязательной проверкой уникальности логина;
* назначение ролей и прав доступа в соответствии с должностными обязанностями;
* привязку пользователей к структурным подразделениям;
* возможность сброса паролей администратором системы.

Пользовательский интерфейс системы включает несколько основных экранов, каждый из которых решает конкретные задачи:

1. экран авторизации - точка входа в систему, где пользователь вводит свои учетные данные. Интерфейс минималистичен и содержит только необходимые элементы для входа;
2. главное меню - центральный узел навигации, откуда пользователь может перейти к любому функционалу в соответствии со своими правами доступа. Меню адаптируется под роль пользователя, скрывая недоступные функции;
3. список заявок - основной рабочий экран системы, где отображается перечень задач с возможностью поиска. Интерфейс предоставляет быстрый доступ к основным операциям с заявками;
4. форма создания/редактирования заявки содержит все необходимые поля для ввода информации о задаче. Поля динамически проверяют вводимые данные и предупреждают об ошибках;
5. профиль пользователя позволяет просматривать и редактировать личные данные, изменять пароль;
6. административная панель предоставляет расширенные инструменты управления системой, доступные только пользователям с соответствующими правами.

При проектировании интерфейса были применены следующие принципы UX/UI:

* единый стиль оформления всех элементов интерфейса;
* интуитивно понятная навигация с минимальным количеством переходов;
* оптимизация рабочих процессов для сокращения количества действий;
* чёткая визуальная иерархия информации;
* контекстные подсказки и сообщения об ошибках.

Работа с базой данных организована с использованием драйвера Npgsql, который обеспечивает эффективное взаимодействие между клиентским приложением и PostgreSQL. Все запросы к базе данных оптимизированы для минимизации времени отклика. Критически важные операции, такие как обновление статусов заявок или изменение прав пользователей, выполняются в рамках транзакций для обеспечения целостности данных.

Бизнес-логика системы включает многоуровневую валидацию вводимых данных. Каждый уровень (клиентский и уровень базы данных) выполняет свою часть проверок. Обработка ошибок реализована через централизованную систему исключений, которая гарантирует, что ни одна ошибка не останется без внимания, а пользователь получит понятное сообщение о проблеме.

Для поддержания стабильной и бесперебойной работы системы внедрён комплексный подход к обеспечению надёжности, охватывающий как обработку возможных сбоёв, так и оптимизацию производительности.

В области обработки ошибок система оснащена многоуровневым механизмом защиты. Центральным элементом является глобальный обработчик исключений, который перехватывает и классифицирует все непредвиденные ситуации в процессе работы. Пользователи получают понятные сообщения об ошибках на естественном языке, что позволяет быстро понять суть проблемы без технических подробностей. Все критические события и ошибки фиксируются в системном журнале с детализацией времени возникновения, условий и контекста выполнения.

Вопросы производительности решаются комплексно на нескольких уровнях. Все SQL-запросы проходят тщательный анализ и оптимизацию для обеспечения максимальной скорости выполнения. Для операций, требующих значительного времени обработки, применяются асинхронные методы выполнения, что предотвращает блокировку интерфейса.

Реализовано 4 уровня доступа пользователей:

1. уровень гостя. Он имеет возможность зарегистрироваться, войти в систему и настроить подключение к БД;
2. уровень клиента доступен сразу после регистрации. Он имеет доступ к созданию заявки и управлению своими заявками;
3. уровень сотрудника АСУ, который имеет доступ к новым заявкам, возможности принятия заявок в работу, редактированию заявок и генерации и просмотра отчётов;
4. уровень администратора. Он имеет доступ ко всем сущностями системы (роли, статусы, приоритеты, отделы, пользователи, заявки) и управлению ими (рис. 2.5).

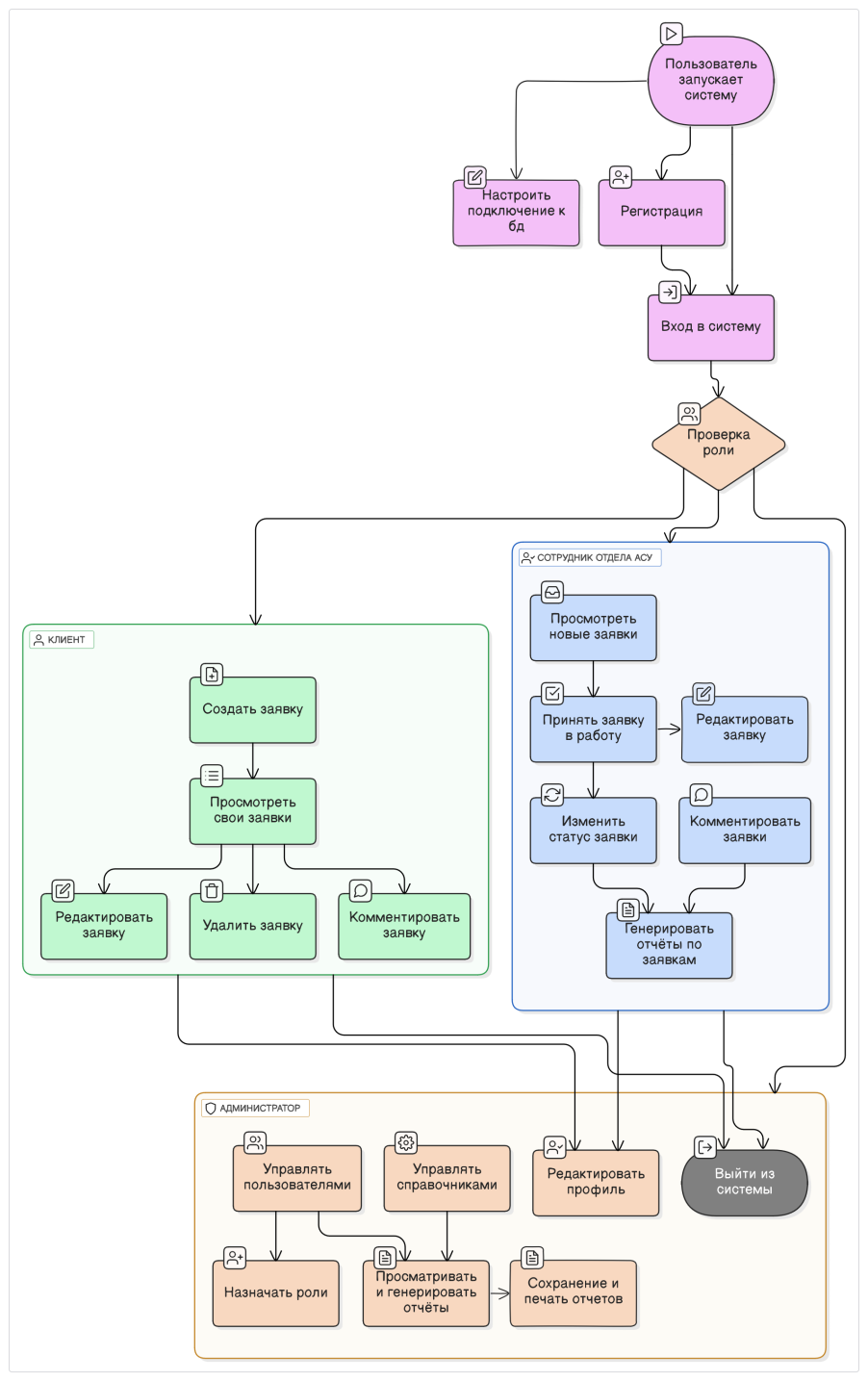


Рис. 2.5 - Use case диаграмма

На рисунке 2.6 представлена схема базы данных, отражающая все таблицы и связи между ними. Схема разработана с соблюдением принципов нормализации до третьей нормальной формы, что обеспечивает эффективное хранение данных без избыточности.

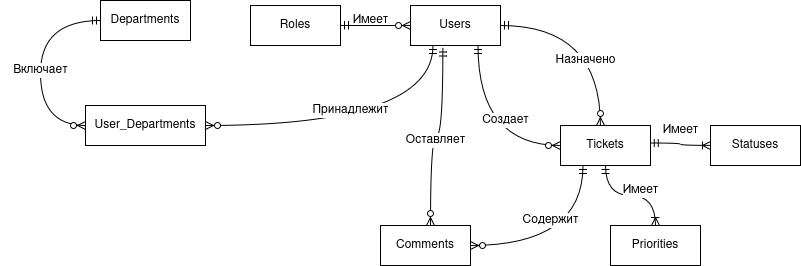


Рис. 2.6 - Схема базы данных

1. **Разработка концептуальной и логической моделей базы данных**

На этапе проектирования была разработана детальная концептуальная модель данных, представленная в виде ER-диаграммы (рис. 2.7). Эта модель отражает ключевые сущности системы и их взаимосвязи, формируя целостное представление о предметной области. Модель включает такие основные сущности как пользователи, роли, заявки, статусы, приоритеты, отделы и комментарии, каждая из которых обладает уникальным набором атрибутов.

Взаимосвязи были реализованы через первичный ключ (PK) и вторичный ключ (FK). Вторичный ключ из одной таблицы указывает на первичный ключ из другой таблицы, формирую связь один ко многим. Во вторичном ключе хранится значение первичного ключа другой таблицы.

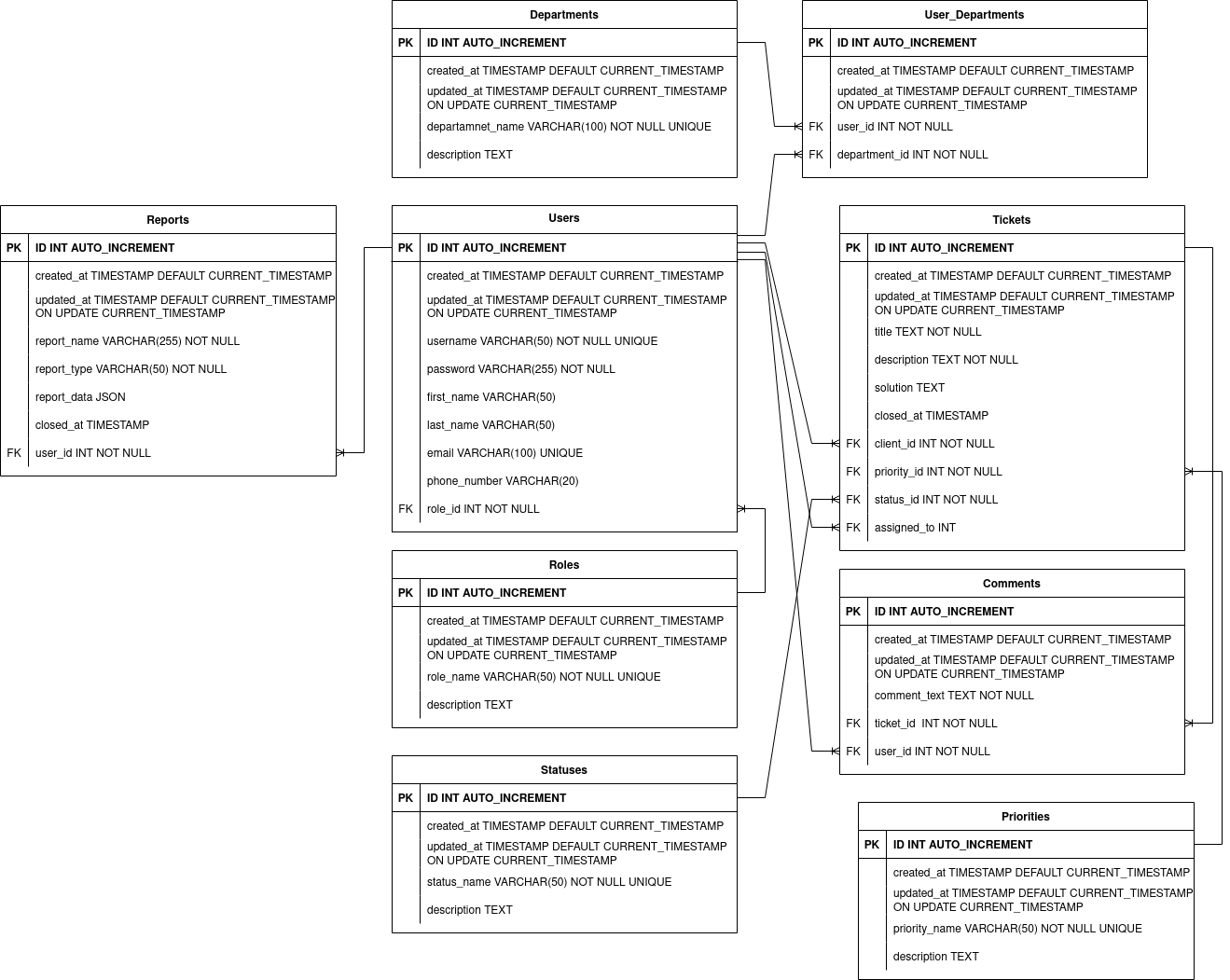


Рис. 2.7 - ER-диаграмма

На основе концептуальной модели была разработана логическая структура базы данных, реализованная в PostgreSQL.

1. Таблица Roles хранит информацию о ролевой модели системы:

CREATE TABLE Roles (

ID SERIAL PRIMARY KEY,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

role\_name VARCHAR(50) NOT NULL UNIQUE,

description TEXT

);

Эта таблица определяет различные роли пользователей (администратор, исполнитель, пользователь) и их описание. Автоматическое заполнение временных меток позволяет отслеживать историю изменений.

1. Таблица Users содержит учетные данные и персональную информацию пользователей:

CREATE TABLE Users (

ID SERIAL PRIMARY KEY,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

username VARCHAR(50) NOT NULL UNIQUE,

password VARCHAR(255) NOT NULL,

first\_name VARCHAR(50),

last\_name VARCHAR(50),

email VARCHAR(100) UNIQUE,

phone\_number VARCHAR(20),

role\_id INT,

FOREIGN KEY (role\_id) REFERENCES Roles(ID)

);

В таблице хранятся все данные о пользователях, а уникальные ограничения на логин и email предотвращают дублирование учётных записей.

1. Таблица Statuses определяет жизненный цикл заявок:

CREATE TABLE Statuses (

ID SERIAL PRIMARY KEY,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

status\_name VARCHAR(50) NOT NULL UNIQUE,

description TEXT

);

Каждый статус (открыта, в работе, выполнена) имеет чёткое описание, что упрощает работу с системой для новых пользователей.

1. Таблица Priorities устанавливает уровни важности задач:

CREATE TABLE Priorities (

ID SERIAL PRIMARY KEY,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

priority\_name VARCHAR(50) NOT NULL UNIQUE,

description TEXT

);

Система поддерживает стандартные приоритеты (низкий, средний, высокий), но структура таблицы позволяет легко добавлять новые уровни при необходимости.

1. Таблица Departments отражает организационную структуру учреждения:

CREATE TABLE Departments (

ID SERIAL PRIMARY KEY,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

department\_name VARCHAR(100) NOT NULL UNIQUE,

description TEXT

);

1. Таблица User\_Departments реализует связь между пользователями и подразделениями:

CREATE TABLE User\_Departments (

ID SERIAL PRIMARY KEY,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

user\_id INT,

department\_id INT,

FOREIGN KEY (user\_id) REFERENCES Users(ID),

FOREIGN KEY (department\_id) REFERENCES Departments(ID),

UNIQUE (user\_id, department\_id)

);

Уникальный индекс на паре user\_id и department\_id гарантирует, что один пользователь не будет привязан к одному подразделению несколько раз.

1. Таблица Tickets - центральная сущность системы:

CREATE TABLE Tickets (

ID SERIAL PRIMARY KEY,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

title TEXT NOT NULL,

description TEXT NOT NULL,

solution TEXT,

closed\_at TIMESTAMP,

client\_id INT,

priority\_id INT,

status\_id INT,

assigned\_to INT,

FOREIGN KEY (client\_id) REFERENCES Users(ID),

FOREIGN KEY (priority\_id) REFERENCES Priorities(ID),

FOREIGN KEY (status\_id) REFERENCES Statuses(ID),

FOREIGN KEY (assigned\_to) REFERENCES Users(ID)

);

Каждая заявка содержит полную информацию о проблеме, её решении и участниках процесса. Внешние ключи обеспечивают целостность данных.

1. Таблица Comments хранит историю обсуждений:

CREATE TABLE Comments (

ID SERIAL PRIMARY KEY,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

comment\_text TEXT NOT NULL,

ticket\_id INT,

user\_id INT,

FOREIGN KEY (ticket\_id) REFERENCES Tickets(ID),

FOREIGN KEY (user\_id) REFERENCES Users(ID)

);

1. Таблица Reports предназначена для хранения отчетов:

CREATE TABLE Reports (

ID SERIAL PRIMARY KEY,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

report\_name VARCHAR(255) NOT NULL,

report\_type VARCHAR(50) NOT NULL,

report\_data JSONB,

user\_id INT,

FOREIGN KEY (user\_id) REFERENCES Users(ID)

);

Использование типа JSONB для report\_data позволяет гибко хранить различные форматы отчётов.

Разработанная структура базы данных соответствует принципам нормализации до третьей нормальной формы (3NF). Это означает, что все атрибуты не содержат повторяющихся групп и отсутствуют частичные зависимости от составного первичного ключа.

Для обеспечения целостности данных реализованы:

* автоматическое обновление полей created\_at и updated\_at;
* внешние ключи для всех связей между таблицами;
* уникальные ограничения на критически важные поля;
* проверочные ограничения на допустимые значения.

Оптимизация производительности достигается за счёт использования транзакций для критических операций и оптимизированных SQL-запросов

Разработанная модель обеспечивает эффективное хранение данных, быстрый доступ к информации и простоту дальнейшего расширения функциональности системы. Гибкость структуры позволяет адаптировать базу данных к изменяющимся требованиям без кардинальных переработок.

**Выводы по главе**

В ходе проектирования информационной системы учета работы отдела АСУ для медицинского учреждения была проделана комплексная работа по формированию архитектуры, выбору технологий и разработке структуры базы данных. Основой системы стал технологический стек, включающий C# и Windows Forms для клиентской части, PostgreSQL в качестве СУБД и трехуровневую архитектуру, обеспечивающую чёткое разделение ответственности между компонентами. Такой подход гарантирует надёжность, безопасность и простоту эксплуатации системы, что особенно важно в условиях медицинского учреждения.

Ключевое внимание было уделено проектированию функциональной структуры, включающей модули авторизации, управления заявками, отчётности и администрирования. Каждый модуль разработан с учётом специфики работы медицинского персонала, обеспечивая интуитивно понятный интерфейс и оптимизированные рабочие процессы.

Разработанная концептуальная и логическая модель базы данных отражает все сущности предметной области и их взаимосвязи. Структура базы данных соответствует принципам нормализации до третьей нормальной формы, что минимизирует избыточность данных и обеспечивает целостность информации. Использование внешних ключей, уникальных ограничений и транзакций гарантирует корректность работы системы даже при высокой нагрузке.

Гибкость выбранных решений позволяет легко адаптировать систему к изменяющимся требованиям, добавлять новые функции и масштабировать её в будущем. В целом, проведённое проектирование создаёт прочную основу для разработки информационной системы, отвечающей всем поставленным задачам и требованиям заказчика.

**Глава 3. РЕАЛИЗАЦИЯ ИНФОРМАЦИОННОЙ СИСТЕМЫ**

1. **Реализация базы данных и SQL запросов**

Процесс реализации базы данных для информационной системы учета работы отдела АСУ включает несколько ключевых этапов: установку и настройку СУБД, проектирование структуры, наполнение тестовыми данными и обеспечение безопасности. В качестве системы управления базами данных была выбрана PostgreSQL 15, установленная на операционную систему Windows 10. Выбор данной СУБД обусловлен её надёжностью, производительностью и соответствием требованиям проекта.

Установка PostgreSQL выполнялась через официальный инсталлятор EnterpriseDB с использованием графического интерфейса. В процессе установки были заданы следующие параметры:

* пароль для суперпользователя postgres;
* порт подключения по умолчанию: 5432;
* локализация: Russian\_Russia.1251;
* компоненты: сервер, pgAdmin, командные строки.

После установки была выполнена первоначальная настройка через утилиту pgAdmin 4 (рис. 3.1), которая предоставляет удобный графический интерфейс для администрирования PostgreSQL.

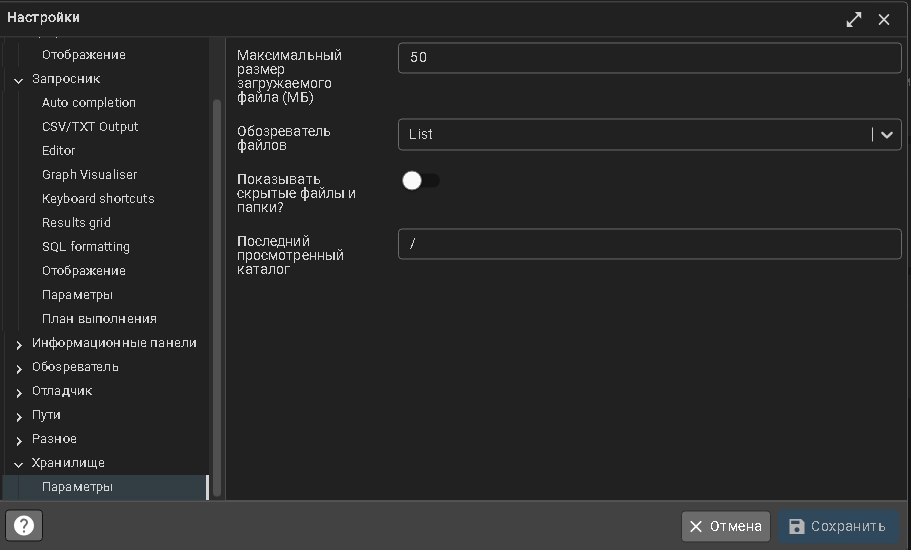


Рис. 3.1 - Настройка pgAdmin 4

Структура базы данных была реализована в соответствии с логической моделью, разработанной на этапе проектирования. Для создания таблиц использовались SQL-запросы, выполненные через интерфейс pgAdmin. Пример создания основных таблиц:

CREATE TABLE Roles (

ID SERIAL PRIMARY KEY,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

role\_name VARCHAR(50) NOT NULL UNIQUE,

description TEXT

);

CREATE TABLE Users (

ID SERIAL PRIMARY KEY,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

username VARCHAR(50) NOT NULL UNIQUE,

password VARCHAR(255) NOT NULL,

first\_name VARCHAR(50),

last\_name VARCHAR(50),

email VARCHAR(100) UNIQUE,

phone\_number VARCHAR(20),

role\_id INT,

FOREIGN KEY (role\_id) REFERENCES Roles(ID)

);

Полная схема базы данных, отображающая все таблицы и связи между ними, представлена на рисунке 2.6.

Для тестирования функциональности системы база данных была заполнена тестовыми данными. Пример добавления тестовых записей:

INSERT INTO Roles (role\_name, description) VALUES

('Admin', 'Полный доступ ко всем функциям системы'),

('ASU\_staff', 'Доступ к заявкам и отчетам'),

('Client', 'Может создавать заявки и просматривать свои заявки');

INSERT INTO users (username, password, first\_name, last\_name, email, role\_id) VALUES

('admin',’ admin\_password', 'Иван', 'Иванов', 'admin@example.com', 1),

(‘staff'', ‘staff\_password', 'Петр', 'Петров', staff@example.com', 2),

('client;', ‘client\_password', 'Сергей', 'Сергеев', client@example.com', 3);

Пример тестовых данных для таблицы заявок:

INSERT INTO tickets (title, description, client\_id, priority\_id, status\_id, assigned\_to) VALUES

('Не работает принтер', 'В кабинете 205 не печатает принтер HP LaserJet', 3, 2, 1, 2),

('Ошибка в программе', 'При попытке открыть историю болезни выдает ошибку', 3, 1, 2, 2);

Для защиты данных были реализована настройка ролевого доступа в PostgreSQL и регулярно создаются резервные копии.

Пример резервного копирования:

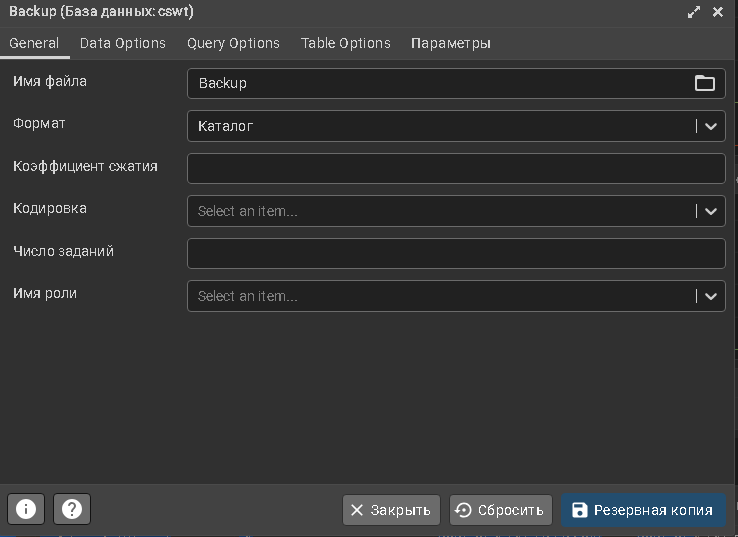


Рис. 3.2 - Резервная копия

Реализованная база данных полностью соответствует требованиям проекта и обеспечивает:

* Надёжное хранение данных;
* Быстрый доступ к информации;
* Целостность данных через систему ограничений;
* Простоту масштабирования и расширения.

Для интеграции базы данных с клиентским приложением использовался драйвер Npgsql. Пример подключения приведен на рисунке 3.3.

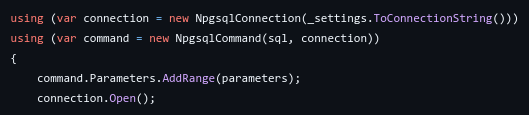


Рис. 3.3 - Подключение к БД

На рисунке 3.3 из \_settings получаем строку подключения и открываем подключение.

Для работы с БД используется набор sql запросов для таблиц на создание, удаление, обновление и чтение данных таблиц. Это набор операций называется CRUD.

CRUD — это аббревиатура, обозначающая четыре основные операции управления данными: создание, чтение, обновление и удаление.

Создание (Create) - это запрос на добавление данных в таблицу. Он состоит из sql команды INSERT INTO (названия полей) VALUES (данные для полей).

Чтение (Read) - это запрос на чтение данных из таблицы. Под чтение подразумевается получение данных. Он состоит из команды SELECT \* FROM (название таблицы). Он также может дополнятся разными параметрами. Например после названия таблицы можно написать WHERE ID = 1. Этот запрос выберет все данные с ID равному 1. Также часто используется параметр JOIN. Он нужен для запроса в несколько таблиц. Например есть таблица Users и в ней есть поле role\_id. По role\_id можно с помощью JOIN сделать запрос в таблицу Roles и достать имя роли. ORDER BY позволяет сортировать данные по убыванию или возрастанию.

Обновление (Update) является запросом на обновление данных. Он включает в себя sql команда UPDATE (название таблицы) SET название поля = новые данные. Он также может включать параметр WHERE.

Удаление (Delete) - это запрос на удаление данных. Он состоит из команды DELETE FROM (название таблицы). Если не добавлять параметр WHRE то удалятся все данные из таблицы, поэтому он необходим.

1. **Реализация клиентского приложения**

Клиентское приложение информационной системы учёта работы отдела АСУ было разработано с использованием языка программирования C#, Windows Forms и платформы .NET Framework 4.7.2. В качестве среды разработки применялась Microsoft Visual Studio Community 2022, как было сказано в главе 2.1.

Архитектура приложения следует принципам трехуровневой архитектуры с чётким разделением ответственности между компонентами, что обеспечивает высокую сопровождаемость и возможность дальнейшего расширения системы.

Архитектура состоит из 3 слоёв:

1. Уровень представления представляет собой формы Windows Forms которые составляют основу пользовательского интерфейса. Каждая форма отвечает за отображение данных и сбор пользовательского ввода, делегируя обработку бизнес-логике (рис. 3.4). Основные формы системы: вход, регистрация, создание заявки, управление заявками, редактирование заявок в зависимости от роли пользователя и отчеты.

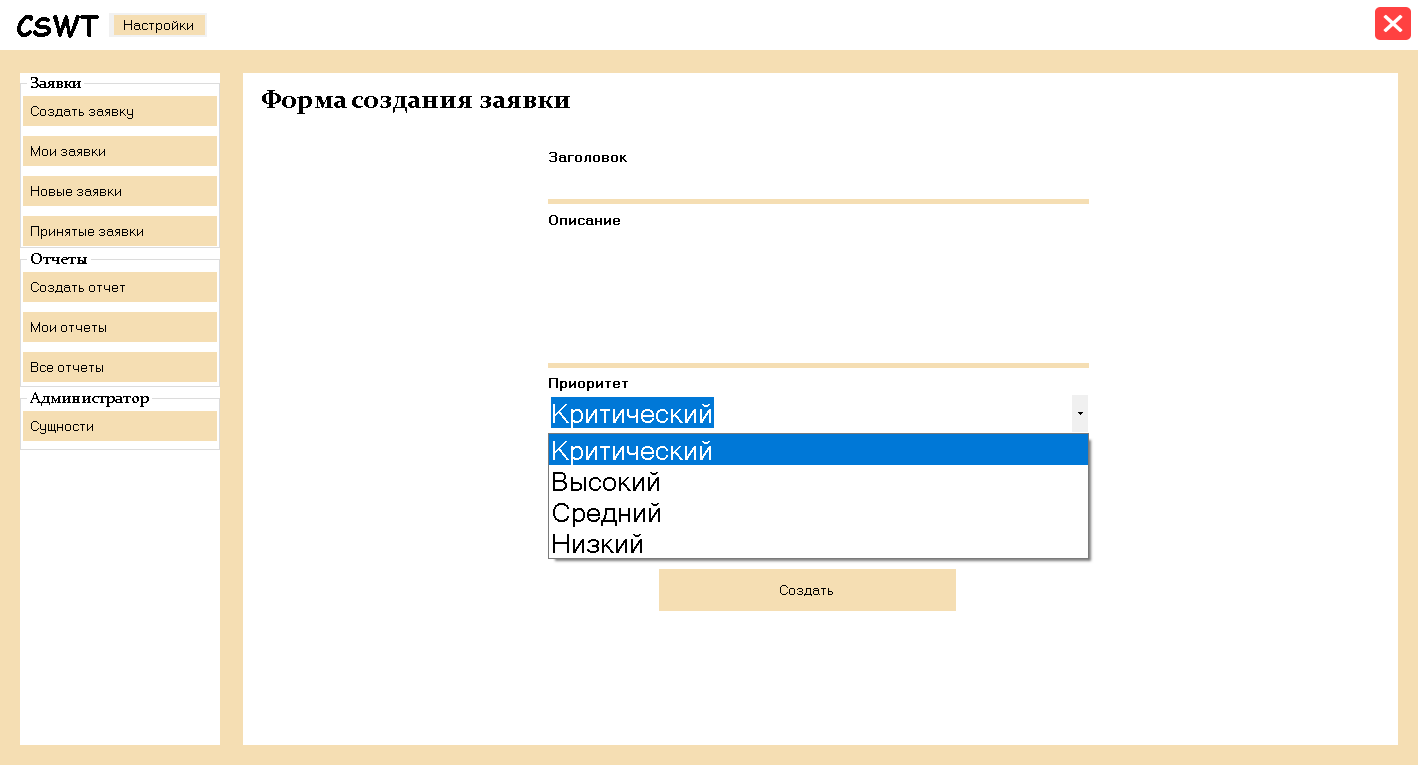


Рис. 3.4 - Уровень представления

1. Уровень бизнес-логики реализован через систему контроллеров, отвечающих за обработку и передачи данных в модели (рис. 3.5). У каждой формы есть свой сервис.



Рис. 3.5 - Уровень бизнес-логики

1. Уровень доступа к данным включает репозиторий, отвечающий за управлением соединения, отправку запросов и вывода ошибок. Также туда входят модели и сервисы, которые создают и отправляют запросы в репозиторий.

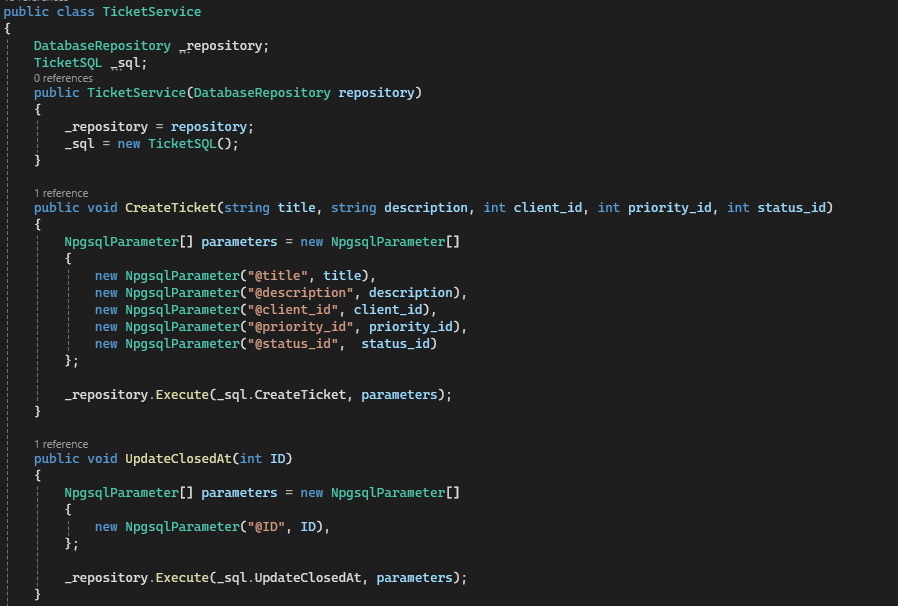


Рис. 3.6 - Уровень данных

**Авторизации состоит из 2 модулей: вход и регистрация** . При входе в систему пользователь проходит многоэтапную проверку: сначала проверяется наличие учетной записи с указанным логином, затем корректность введённого пароля. При регистрации проводится проверка на наличия пользователя с таким же паролем и почтой, и в случае существования такого пользователя будет ошибка. После успешной регистрации всплывёт соответствующее сообщение (рис. 3.7). После успешной аутентификации система устанавливает соответствующие права доступа в зависимости от роли пользователя.

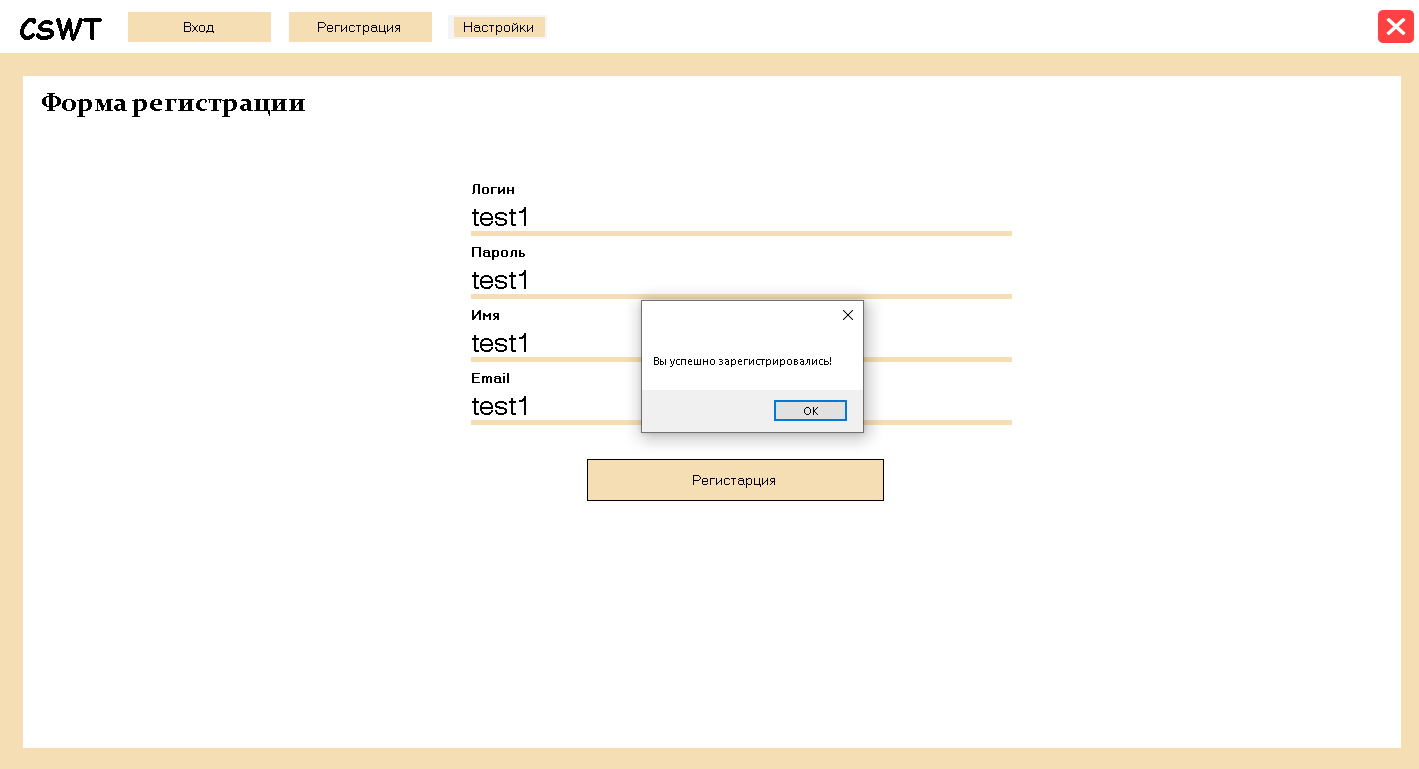


Рис. 3.7 - Модуль регистрации

Модули управления заявками представляет собой центральные компоненты системы, обеспечивающие полный жизненный цикл обработки заявок - от создания до закрытия. При создании новой заявки система автоматически присваивает ей уникальный идентификатор, фиксирует дату и время создания, а также связывает заявку с учётной записью создателя. Для каждой заявки предусмотрены обязательные поля: название проблемы, описание проблемы, статус и приоритет. Система позволяет назначать исполнителей, изменять статусы выполнения, добавлять комментарии.

Пользователь с уровнем доступа клиента может создавать заявку, просмотреть заявки, удалить, редактировать и писать комментарии к своим заявкам.

Пользователь с доступом сотрудника АСУ имеет доступ к новым заявкам с возможность принятия ее, к своим принятым заявкам, к комментариям, к редактированию и выполнению заявки (рис. 3.8)

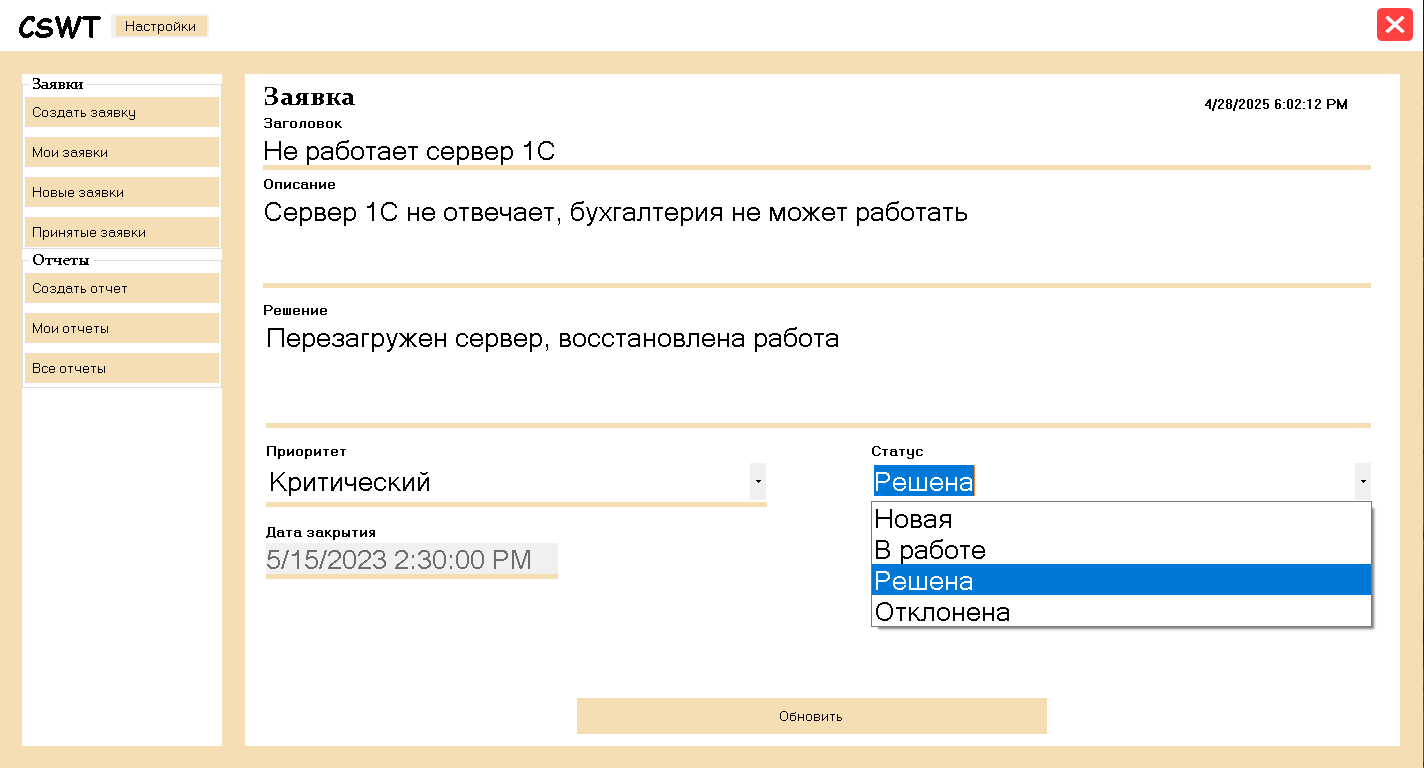


Рис. 3.8 - Модуль заявки

Модуль отчетности предоставляет широкие возможности для анализа работы отдела АСУ. Система поддерживает генерацию отчетов различных типов: статистические отчеты по всем заявкам, отчеты заявкам за последний месяц, отчеты по сотрудникам. Для каждого отчета предусмотрен различные формат вывода - DOCX или печать. Модуль отчетности состоит из генерации отчета, просмотра своих и всех отчетов и формы просмотра отчета.

Отчет состоит из названия, типа и данных (рис. 3.9). Названия и тип генерируются автоматически, а данные собираются из БД.

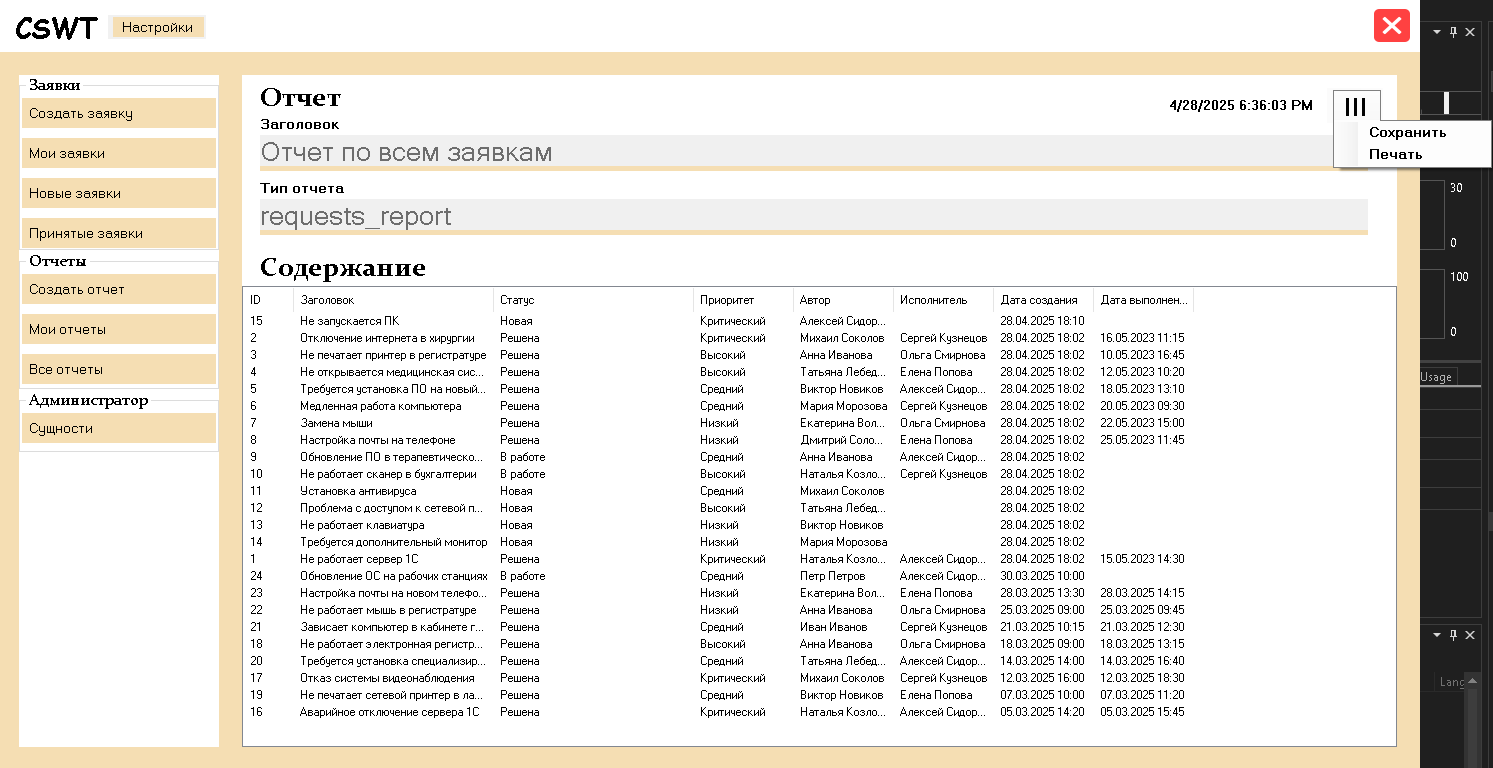


Рис. 3.9 - Модуль отчета

Административный модуль предназначен для управления системными параметрами и настройками. Администраторы могут управлять всеми данными системы - статусами, приоритетами, структурными подразделениями, заявками, пользователями, ролями (Рис. 3.10). Для каждого справочника предусмотрены функции добавления и удаления записей с обязательной проверкой на использование в существующих заявках.

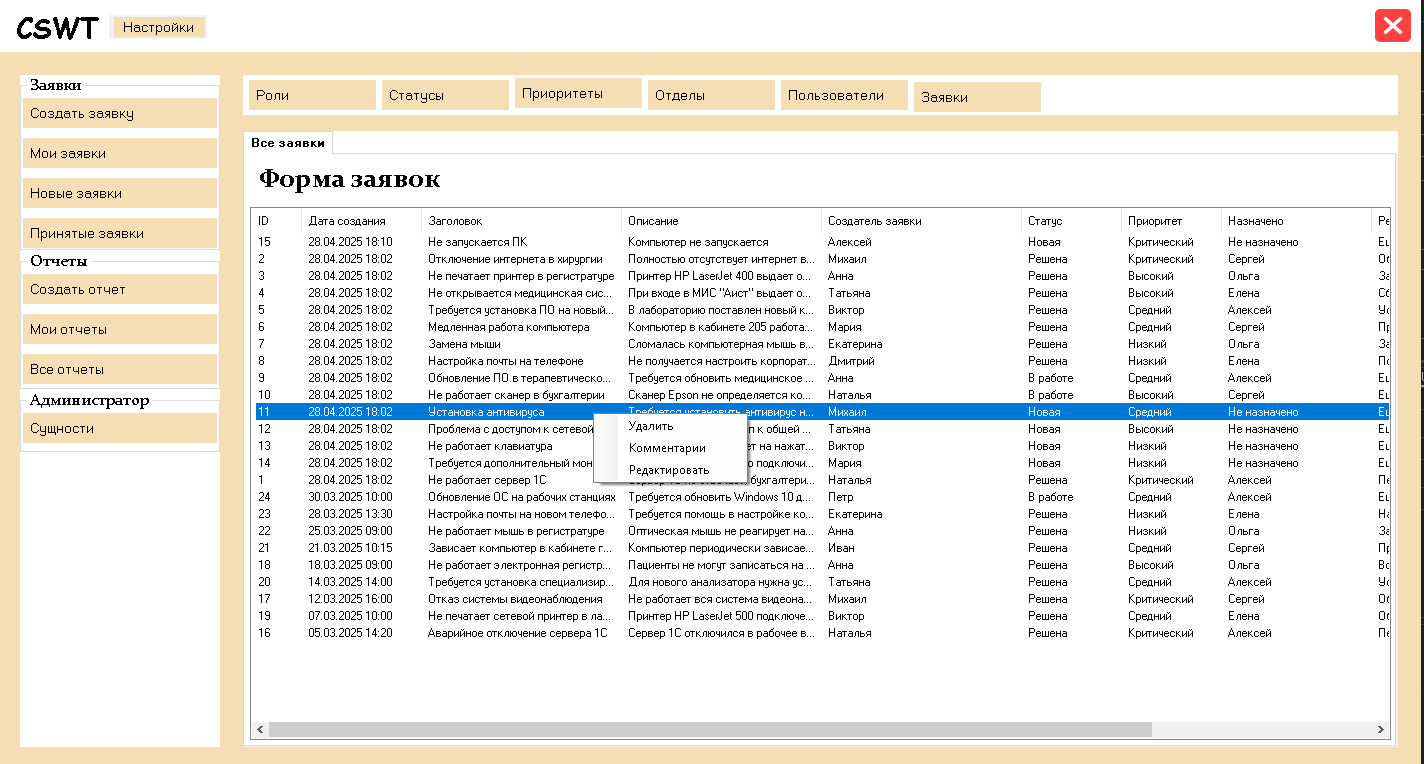


Рис. 3.10 - Управление заявками администратором

Также был разработан модуль, отвечающий за работы с подключением к БД.

Он содержит данные о БД, нужные для подключения. Данные хранятся в отдельном файле и при запросе достаются из этого файла. Сохранить настройки можно при помощи формы настроек БД (рис. 3.11)

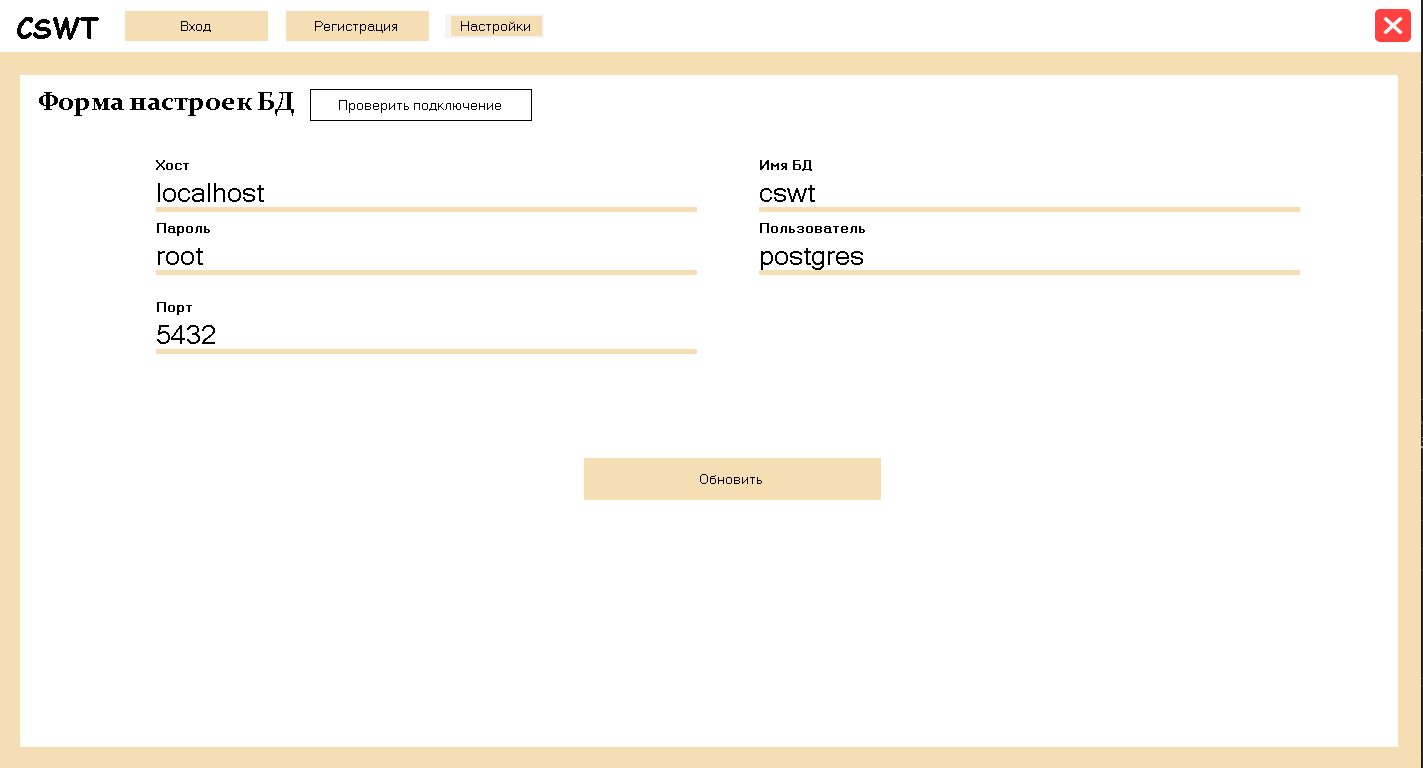


Рис. 3.11 - Подключение к БД

Модуль профиля позволяет обновить свой профиль (Рис. 3.12). Можно изменить все данные кроме роли, она доступны только администраторам.



Рис. 3.12 - Профиль

Каждый модуль системы был реализован с учётом требований надёжности и отказоустойчивости. Все критические операции выполняются в рамках транзакций, что гарантирует целостность данных даже в случае сбоев.

Интерфейсы всех модулей были разработаны с учётом эргономических требований и особенностей работы медицинского персонала. Особое внимание уделено простоте освоения - даже новые пользователи могут быстро начать работу с системой благодаря интуитивно понятному интерфейсу и встроенным подсказкам. Все элементы интерфейса локализованы на русский язык и соответствуют принятой в организации терминологии.

Техническая реализация модулей основана на современных технологиях и подходах к разработке программного обеспечения. Код системы хорошо структурирован, содержит подробные комментарии и соответствует принятым стандартам именования. Это существенно упрощает дальнейшее развитие и сопровождение системы, позволяя новым разработчикам быстро вникать в особенности реализации.

1. **Руководство пользователя**

Руководство предназначено для всех категорий пользователей информационной системы учёта работы отдела автоматизированных систем управления ГБУЗ "Лабинская центральная районная больница". Разработанная система представляет собой современное программное решение, позволяющее оптимизировать процессы обработки заявок, контроля выполнения задач и формирования аналитической отчётности.

Руководство содержит подробные инструкции по работе с системой, начиная от процедуры установки и заканчивая решением типовых проблем. Особое внимание уделено описанию функциональных возможностей системы для различных категорий пользователей - обычных сотрудников, специалистов отдела АСУ и администраторов системы.

Перед первым запуском системы необходимо выполнить её установку на рабочую станцию. Для корректной работы приложения требуется компьютер с операционной системой Windows 10 или более новой версией. В случае работы в сетевом режиме необходимо предварительно настроить подключение к серверу базы данных PostgreSQL.

Процедура входа в систему начинается с запуска клиентского приложения. На экране авторизации пользователю необходимо ввести свои учётные данные. После проверки введённых данных система предоставляет доступ к функционалу в соответствии с назначенной ролью пользователя. В случае неверного ввода данных система отображает соответствующее сообщение об ошибке (рис. 3.13).

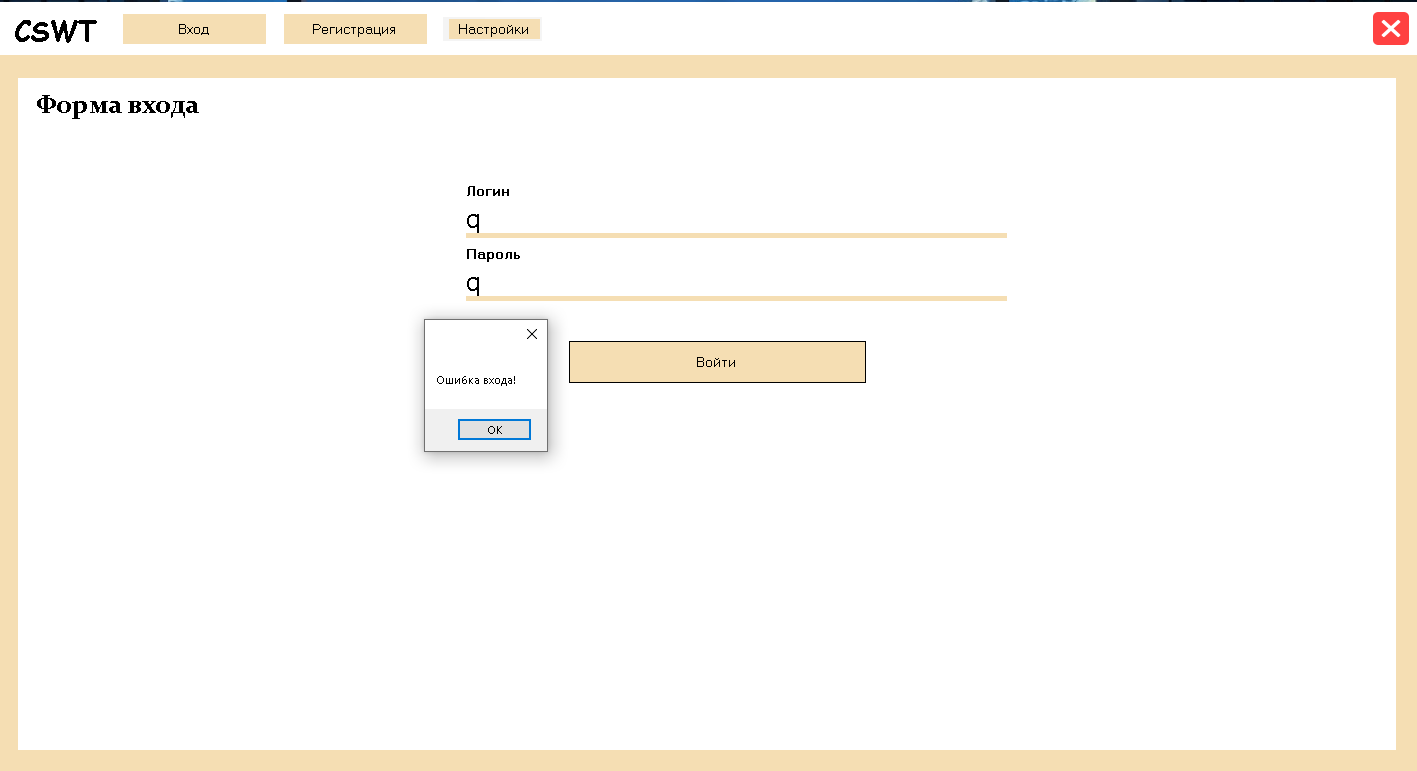


Рис. 3.13 - Ошибка входа в систему

Для восстановления доступа при утрате пароля необходимо обратится к администратору.

Работа с заявками является ключевой функцией системы. Для создания новой заявки пользователю необходимо перейти в соответствующий раздел меню и заполнить обязательные поля формы: название проблемы, её описание и уровень приоритета. После сохранения заявка поступает в обработку и становится доступной для просмотра и редактирования.

В разделе "Мои заявки" пользователь может просматривать список всех созданных им обращений. Для каждой заявки предусмотрена возможность редактирования (изменения описания и названия) и удаления. Особое внимание уделено механизму комментирования - пользователи могут добавлять пояснения и уточнения по ходу обработки заявки, что значительно улучшает коммуникацию между сотрудниками (рис. 3.14).

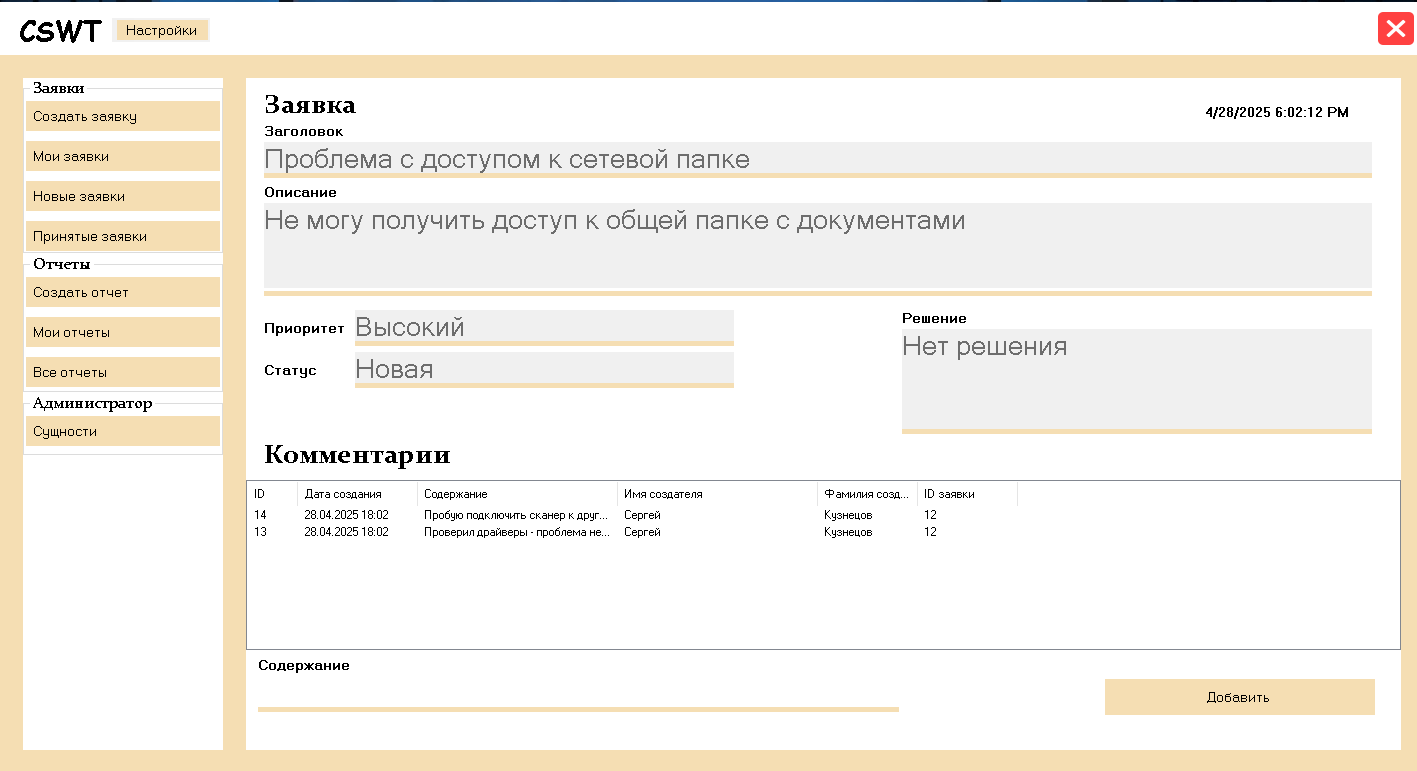


Рис. 3.14 - Комментирование заявки

Для специалистов отдела АСУ система предоставляет расширенный функционал управления задачами. В разделе "Новые заявки" отображаются все поступившие обращения, которые можно принять в работу. После принятия заявки сотрудник может изменять её статус в соответствии с этапами обработки: "В работе", "Отклонено" или "Выполнено". Это позволяет чётко отслеживать прогресс решения каждой проблемы.

Формирование отчётов - важная аналитическая функция системы. Пользователи могут генерировать различные типы отчётов: статистику по заявкам за выбранный период, анализ выполненных задач за месяц или отчёт по пользователям. Сформированные отчёты можно экспортировать в формат DOCX для дальнейшего редактирования или сразу выводить на печать.

Администраторам системы доступен расширенный функционал управления. В их обязанности входит регистрация новых пользователей, редактирование их учётных данных, удаление учётных записей при необходимости. Также администраторы могут настраивать системные данные: добавлять новые статусы заявок, виды приоритетов, структурные подразделения организации.

В процессе работы с системой пользователи могут столкнуться с некоторыми трудностями. Наиболее распространённой проблемой является невозможность входа в систему. В этом случае рекомендуется проверить правильность ввода логина и пароля, обратив внимание на состояние клавиши Caps Lock. Если проблема сохраняется, следует обратиться к администратору системы.

Другая частая ситуация - невозможность сохранения созданной заявки. В этом случае необходимо проверить заполнение всех обязательных полей формы (название и описание проблемы), а также убедиться в наличии активного подключения к серверу базы данных.

Для изменения персональных данных пользователю необходимо перейти в раздел "Профиль" и выбрать пункт меню "Редактировать данные". Здесь можно изменить контактную информацию и другие параметры учетной записи (за исключением роли, которые изменяются только администратором).

В случае возникновения технических проблем, не описанных в данном руководстве, пользователи могут обратиться в службу технической поддержки. Контактная информация для обращения: электронная почта [support@labinsk-crb.ru](https://mailto:support@labinsk-crb.ru" \t "/home/Shau/Documents\x/_blank). Специалисты поддержки готовы помочь в решении любых вопросов, связанных с работой системы.

**Выводы по главе**

В этой главе была реализована информационная система для учёта работы отдела АСУ. Разработка велась в два основных этапа: сначала была создана база данных, а затем - клиентское приложение.

Для базы данных был выбран PostgreSQL 15, потому что это надёжная и современная система управления базами данных. Была проведена настройка СУБД, настроены права доступа и резервное копирование. Были создана все необходимые таблицы (пользователи, роли, заявки), прописаны связи между ними и заполнены тестовыми данными для проверки работы.

Клиентскую часть написана на C# с использованием Windows Forms. Это оказалось удобным решением, так как интерфейс получился интуитивно понятным даже для неподготовленных пользователей. Трёхуровневая архитектура обеспечила гибкость и удобство поддержки. Код был разделён на три слоя (интерфейс, логика и работа с данными), что в будущем упростит доработку системы.

Особое внимание уделено основным модулям:

* Система входа и регистрации с проверкой прав доступа;
* Удобное создание и редактирование заявок;
* Генерация отчётов;
* Административная панель для управления системой.

Когда основные функции были готовы, было написано подробное руководство пользователя. В нем пошагово объясняется, как работать с системой - от установки до решения типовых проблем.

В результате получилась готовая к внедрению система, которая:

* Позволяет удобно вести учёт заявок;
* Автоматизирует создание отчётов;
* Надёжно защищает данные;
* При необходимости легко дополняется новыми функциями.

**ЗАКЛЮЧЕНИЕ**

В ходе выполнения дипломного проекта была разработана информационная система для автоматизации учёта работы отдела автоматизированных систем управления ГБУЗ «Лабинская центральная районная больница». Проект решает ключевые проблемы, связанные с ручным учётом задач, дублированием данных и отсутствием единого информационного пространства, что существенно снижало эффективность работы подразделения.

На этапе анализа предметной области были изучены организационная структура больницы, бизнес-процессы отдела АСУ и выявлены основные недостатки существующей системы учета. На основании проведенного исследования сформулированы функциональные и технические требования к разрабатываемой системе, включая модули управления заявками, пользователями, отчётностью и администрирования.

Проектирование системы осуществлялось с применением современных технологий: в качестве клиентской части выбрана платформа .NET Framework с языком C# и Windows Forms, а для хранения данных — СУБД PostgreSQL. Трёхуровневая архитектура (уровень представления, бизнес-логики и данных) обеспечила гибкость, надёжность и масштабируемость решения.

Реализация системы включала создание базы данных с соблюдением принципов нормализации, разработку интуитивно понятного пользовательского интерфейса и интеграцию между клиентским приложением и сервером. Особое внимание уделено безопасности данных, отказоустойчивости и удобству эксплуатации.

Внедрение данной системы позволит:

* централизовать управление заявками и задачами отдела АСУ;
* сократить время на рутинные операции за счёт автоматизации процессов;
* повысить прозрачность и контролируемость выполнения задач;
* улучшить качество отчётности за счёт автоматической генерации документов;
* обеспечить безопасное хранение данных и разграничение прав доступа.

Разработанная информационная система соответствует всем поставленным требованиям и готова к внедрению в работу отдела АСУ ГБУЗ «Лабинская ЦРБ». В перспективе возможна дальнейшая доработка системы, включая добавление веб-интерфейса, мобильного приложения и интеграцию с другими медицинскими информационными системами.

Вследствие этого, выполненный дипломный проект подтверждает возможность эффективной автоматизации учёта работы ИТ-подразделений в медицинских учреждениях и демонстрирует практическую значимость внедрения специализированных решений для повышения качества управления процессами.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ:**

1. Федеральный закон от 29.07.2018 № 250-ФЗ «О цифровых технологиях в здравоохранении».
2. ГОСТ 2.105-95. ЕСКД. Общие требования к текстовым документам.
3. ГОСТ 7.32-2003. Отчет о научно-исследовательской работе. Структура и порядок оформления.
4. ГОСТ 34.601-90. Автоматизированные системы. Стадии создания.
5. ГОСТ Р 55079-2012. Информационные технологии в здравоохранении.
6. ГОСТ Р ИСО/МЭК 12207-2010. Процессы жизненного цикла программных средств.
7. Устав ГБУЗ "Лабинская ЦРБ".
8. Методические рекомендации по дипломному проекту 2025 ИСиП.
9. Агальцов В.П. Базы данных. — М.: ИНФРА-М, 2019.
10. Бейтс Б., Сьерра К. Изучаем C#. — СПб.: Питер, 2021. — 720 с.
11. Гохберг Г.С., Зафиевский А.В., Короткин А.В. Информационные технологии. — М.: Издательский центр «Академия», 2020. — 208 с.
12. Гребенюк Е.И., Гребенюк Н.А. Технические средства информатизации. — М.: Издательский центр «Академия», 2020. — 272 с.
13. Келим Ю.М. Вычислительная техника. — М.: Издательский центр «Академия», 2021. — 384 с.
14. Коннолли Т., Бегг К. Базы данных. Проектирование, реализация и сопровождение. — М.: Вильямс, 2021. — 896 с.
15. Михеева Е.В. Информационные технологии в профессиональной деятельности. — М.: Издательский центр «Академия», 2021. — 384 с.
16. Михеева Е.В. Практикум по информационным технологиям в профессиональной деятельности. — М.: Издательский центр «Академия», 2021. — 256 с.
17. Рудаков А.В. Технология разработки программных продуктов. — М.: Издательский центр «Академия», 2010. — 208 с.
18. Рихтер Дж. CLR via C#. — М.: Питер, 2020. — 896 с.
19. Семакин И.Г., Шестаков А.П. Основы программирования. — М.: Издательский центр «Академия», 2011. — 432 с.
20. Троелсен Э. Язык программирования C# 9.0 и платформа .NET 5. — М.: Вильямс, 2022. — 1328 с.
21. Фаулер М. Архитектура корпоративных программных приложений. — М.: Вильямс, 2019. — 544 с.
22. Хомоненко А.Д. Базы данных. — СПб.: Корона-Век, 2020. — 736 с.
23. Хомоненко А.Д., Цыганков В.М., Мальцев М.Г. Базы данных. — СПб.: Корона, 2019. — 736 с.
24. PostgreSQL Documentation. URL: https://www.postgresql.org/docs/
25. Microsoft .NET Framework Documentation. URL: https://docs.microsoft.com/en-us/dotnet/
26. C# Programming Guide. URL: https://learn.microsoft.com/ru-ru/dotnet/csharp/tour-of-csharp/
27. Windows Forms Documentation. URL: https://docs.microsoft.com/ru/dotnet/desktop/winforms/
28. Visual Studio Documentation. URL: https://docs.microsoft.com/ru/visualstudio/
29. Npgsql Documentation. URL: http://www.npgsql.org/doc/