

Министерство образования Калининградской области



Государственное бюджетное учреждение
Калининградской области профессиональная
образовательная организация
«Колледж информационных технологий и
строительства»

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

**Разработка приложения по сбору заявок на ремонт различной техники
«Служба поддержки»**

Выполнила: обучающаяся группы ИСп 20-1 Ведель Снежана Петровна

Специальность: 09.02.07 «Информационные системы и программирование»
(код, наименование)

Руководитель ВКР:

Большакова-Стрекалова А.В.
(Ф.И.О. преподавателя)

Выпускная квалификационная
работа допущена к защите
«___»_____20____г.

Рецензент: Тарасевич Илья Владимирович,
директор АНО ДПО "БИРПП"
(Ф.И.О., место работы)

Заместитель директора по
учебно – методической работе
_____/ Павленко Г.Я.
(подпись) (Ф.И.О.)

Председатель ГЭК: Наконечный А.Н.
(Ф.И.О.)

Калининград
2024 г

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ

для выполнения выпускной квалификационной работы

Специальность 09.02.07 «Информационные системы и программирование»

Группа ИСП 20-1

Ф.И.О. обучающейся Ведель Снежана Петровна

Тема ВКР Разработка приложения по сбору заявок на ремонт различной техники «Служба поддержки»

Дата выдачи задания: «22» апреля 2024 г.

Работа должна быть сдана не позднее «15» июня 2024 г.

Перечень вопросов, подлежащих разработке в ВКР

1. Подготовка материала и оформление технического задания и последующего технического проекта по теме дипломной работы.
 - Определение целей и задач проекта.
 - Оформление технического задания, включающего функциональные и нефункциональные требования.
2. Подбор и анализ материалов для определения того нового, что будет разрабатываться в дипломной работе.
 - Изучение существующих аналогов и их недостатков.
 - Определение уникальных аспектов разрабатываемого приложения.
3. Анализ и проектирование структуры и дизайна приложения.
 - Проектирование структуры клиентской части приложения на языке программирования Java с использованием JavaFX.
 - Проектирование пользовательского интерфейса (UI/UX).
 - Проектирование структуры базы данных с помощью СУБД.
 - PostgreSQL для хранения информации о пользователях и заявках.
4. Программная реализация приложения.
 - Разработка клиентской части приложения на языке

программирования Java с использованием JavaFX.

- Создание пользовательского интерфейса.
- Реализация базы данных с помощью СУБД PostgreSQL.

5. Подготовка методического обеспечения.

- Разработка руководства пользователя.
- Разработка руководства системного администратора.

6. Расчет себестоимости разработки.

- Оценка трудозатрат на разработку приложения.
- Расчет общей стоимости проекта.

Графическая часть

1. Структурная схема:

- Структурная схема основных модулей приложения и их взаимодействия.

2. Диаграммы UML:

- ER-диаграмма для представления структуры базы данных.
- Диаграмма активностей для моделирования взаимодействия объектов в рамках основных сценариев использования.

Руководитель ВКР: _____
(подпись)

Большакова-Стрекалова А.В.
(Ф.И.О.)

Задание получил: _____
(подпись)

Ведель С.П.
(Ф.И.О.)

«22» апреля 2024 г.

Содержание

Введение	6
1 Сбор и анализ информации	8
1.1 Анализ и описание предметной области	8
1.1.1 Характеристики и компоненты АИС	8
1.1.2 Описание предметной области.....	10
1.1.3 Роли и их функции	11
1.2 Обзор существующих аналогов программного обеспечения	12
1.2.1 ServiceNow	12
1.2.2 Jira Service Desk	16
1.2.3 Freshdesk.....	19
1.3 Техническое задание.....	23
1.3.1 Описание функциональных требований.....	23
1.3.2 Описание нефункциональных требований.....	24
1.3.3 Описание архитектуры системы	24
2 Используемые технологии и инструменты	26
2.1 Обоснование используемого языка программирования	26
2.2 Выбор системы управления базами данных.....	26
2.3 Обоснование используемых фреймворков и библиотек.....	27
2.4 Выбор редактора кода.....	28
2.5 Выбор графического редактора для создания дизайна.....	29
3 Разработка и реализация системы.....	31
3.1 Проектирование базы данных	31
3.1.1 Определение сущностей и связей	31
3.1.2 Настройка триггеров.....	33
3.2 Диаграмма активностей	34
3.3 Диаграмма вариантов использования	35
3.4 Разработка серверной части	36
3.5 Разработка клиентской части	41
3.5.1 Дизайн	41
3.5.2 Разработка основных функций	52
4 Тестирование приложения.....	55

4.1	Модульное тестирование.....	55
4.2	Интеграционное тестирование	56
4.3	Пользовательское тестирование.....	56
4.4	Ручное тестирование.....	57
5	Методическое обеспечение	58
5.1	Руководство системного администратора	58
5.2	Руководство пользователя	67
6	Экономическая часть	69
	Заключение	73
	Список используемых источников.....	76
	Приложение. Листинг кода.....	78

Введение

Современные предприятия и организации активно используют различную технику для выполнения широкого спектра задач. Важно обеспечить ее бесперебойную работу, так как простои оборудования могут приводить к значительным финансовым потерям и снижению общей эффективности работы. Автоматизация процесса подачи и обработки заявок на ремонт позволяет сократить время простоя техники, повысить эффективность работы обслуживающего персонала и улучшить удовлетворенность пользователей. Несмотря на наличие различных систем для управления заявками на ремонт, многие из них имеют недостатки, такие как сложность интерфейса, недостаточная производительность и ограниченная функциональность. Поэтому создание нового приложения по сбору заявок на ремонт различной техники «Служба поддержки» актуально для повышения эффективности управления заявками на ремонт и удовлетворения потребностей современных организаций.

Проблема заключается в необходимости автоматизации и оптимизации процесса подачи, обработки и выполнения заявок на ремонт различной техники для минимизации простоев оборудования и удовлетворения запросов пользователей. Текущие решения не полностью удовлетворяют потребности пользователей и системных администраторов из-за своей сложности и ограниченной функциональности. Поэтому требуется разработка нового, более эффективного приложения.

Целью проекта является создание приложения по сбору заявок на ремонт различной техники «Служба поддержки» для автоматизации и оптимизации процесса подачи, обработки и выполнения заявок на ремонт различной техники, что позволит минимизировать простои оборудования и удовлетворить запросы пользователей.

Для достижения данной цели необходимо решить следующие задачи:

1. Дать определение и охарактеризовать предметную область.

2. Проанализировать существующие аналоги программного обеспечения для управления заявками на ремонт.
3. Разработать техническое задание, включающее функциональные и нефункциональные требования к системе.
4. Спроектировать и реализовать базу данных для хранения информации о заявках, пользователях и оборудовании.
5. Создать клиентскую часть приложения с удобным и интуитивно понятным пользовательским интерфейсом.

Объектом исследования является процесс подачи, обработки и выполнения заявок на ремонт техники в предприятиях и организациях.

Предметом исследования является разработка и внедрение приложения по сбору заявок на ремонт различной техники «Служба поддержки» для автоматизации и оптимизации процесса подачи, обработки и выполнения заявок на ремонт различной техники.

Ссылка на GitHub - https://github.com/SnezhanaVedel/Support_Service

1 Сбор и анализ информации

1.1 Анализ и описание предметной области

Автоматизированная информационная система (АИС) — это комплекс программного и аппаратного обеспечения, предназначенный для автоматизации процессов обработки информации в организации [1]. Внедрение АИС способствует эффективному управлению ресурсами и помогает принимать обоснованные решения на основе данных.

1.1.1 Характеристики и компоненты АИС

1. Аппаратное обеспечение: компьютеры, серверы, устройства хранения данных и сетевое оборудование для функционирования системы.

2. Программное обеспечение: системное ПО (операционные системы, сетевые протоколы) и прикладное ПО для сбора, обработки и представления данных.

3. База данных: системы управления базами данных (СУБД) с централизованным хранением и доступом к данным [2].

4. Организационные компоненты: сотрудники, процедуры и инструкции, обеспечивающие эффективное использование АИС.

5. Информационная безопасность: механизмы аутентификации, шифрования и мониторинга для защиты данных.

Преимущества использования АИС:

1. Автоматизация процессов: система автоматизирует рутинные операции по сбору, обработке и анализу данных, повышая эффективность и надежность работы.

2. Централизованное управление данными: АИС обеспечивает единое хранилище данных, улучшая их согласованность и снижая вероятность ошибок.

3. Улучшенное принятие решений: инструменты аналитики и отчетности позволяют принимать обоснованные решения на основе актуальной и точной информации.

4. Снижение издержек: оптимизация процессов помогает снизить затраты на выполнение операций, устраняя избыточные этапы и ручные операции.

5. Гибкость и масштабируемость: модульная структура позволяет расширять и изменять функциональность системы в соответствии с нуждами организации.

6. Повышение безопасности данных: инструменты управления доступом, резервного копирования и мониторинга обеспечивают защиту информации от утечек и несанкционированного доступа.

Недостатки использования АИС:

1. Высокие начальные затраты: внедрение АИС требует значительных вложений в оборудование, программное обеспечение, обучение персонала и интеграцию.

2. Сложность внедрения: процесс внедрения может быть длительным и сложным из-за необходимости адаптировать систему к специфическим бизнес-процессам.

3. Зависимость от технологий: организация становится зависимой от определенных технологий и поставщиков, особенно если система сложная и требует специализированного обслуживания.

4. Проблемы безопасности данных: с увеличением объема данных и числа пользователей растет риск утечки или несанкционированного доступа к информации.

5. Необходимость обслуживания и обновления: требуется регулярное техническое обслуживание, обновление программного обеспечения и резервное копирование данных.

6. Изменение структуры: внедрение может потребовать изменений в организационной структуре и рабочих процессах.

7. Угрозы простоя: сбои в системе могут привести к полной остановке работы организации.

8. Сопротивление изменениям: сотрудники могут испытывать трудности с переходом на новую систему из-за непривычных процессов и сопротивляться нововведениям.

Несмотря на эти недостатки, при правильном планировании и внедрении АИС значительно повышают эффективность работы организации, помогая оптимизировать процессы, управлять данными и принимать взвешенные решения на основе актуальной информации.

1.1.2 Описание предметной области

Основная цель приложения «Служба поддержки» — автоматизация и оптимизация процесса подачи, обработки и выполнения заявок на ремонт различной техники, минимизация простоев оборудования и удовлетворение запросов пользователей. Системный администратор и пользователи могут взаимодействовать для эффективного обслуживания оборудования, используя функциональность приложения, написанного на языке Java с использованием базы данных PostgreSQL и работающего по архитектуре «клиент-сервер».

Основные компоненты предметной области включают:

1. Создание заявки: пользователь создаёт заявку на ремонт, предоставляя информацию о проблемах с оборудованием. Заявке автоматически присваивается уникальный идентификатор. Указываются серийный номер оборудования, описание проблемы и текущий статус заявки.

2. Регистрация заявки: системный администратор принимает и регистрирует заявку в системе, добавляя комментарии при необходимости. Информация о заявке и пользователе, её создавшем, сохраняется в базе данных.

3. Обработка заявки: системный администратор анализирует заявку и определяет, какие действия необходимы для её выполнения. При необходимости уточняет детали у пользователя.

4. Исполнение заявки: администратор либо сам выполняет ремонт оборудования, либо организует закупку необходимых запчастей. Если требуются детали, создаётся заказ на их приобретение, который направляется на одобрение директору. После выполнения всех необходимых работ заявка закрывается.

5. Отчётность и информирование: по каждой завершённой заявке системный администратор составляет отчёт, включающий тип проведенного ремонта, затраченное время, использованные ресурсы и суммарные расходы. Отчёт сохраняется в системе.

6. Мониторинг и анализ: системный администратор отслеживает и анализирует статистику по заявкам, оценивая количество, время и ресурсы, затраченные на выполнение каждой заявки. Эта информация используется для улучшения процессов и повышения эффективности работы.

1.1.3 Роли и их функции

Системный администратор:

- Просматривает и закрывает заявки.
- Формирует отчеты в приложении.
- Редактирует таблицу с оборудованием, включая данные о серийных номерах, типе, состоянии и местонахождении техники.
- Редактирует данные о пользователях (ФИО, логин, пароль, контактные данные).
- Формирует запрос на заказ запчастей директору.

Пользователь:

- Отправляет заявки на ремонт техники, предоставляя системному администратору информацию о неисправностях.

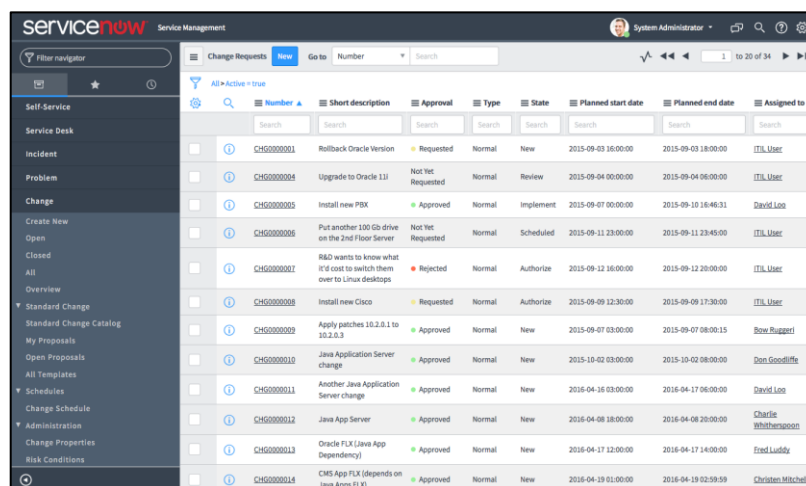
Таким образом, приложение «Служба поддержки» позволяет эффективно организовать процесс ремонта и обслуживания техники, предоставляя пользователям и системному администратору все необходимые инструменты для сбора, обработки и анализа заявок.

1.2 Обзор существующих аналогов программного обеспечения

Обзор существующих аналогов позволяет выявить их сильные и слабые стороны, оценить их функциональные возможности, удобство использования, производительность и надежность. Ниже будет приведено несколько примеров.

1.2.1 ServiceNow

ServiceNow — это облачная платформа, предназначенная для автоматизации и оптимизации различных бизнес-процессов, включая управление заявками на ремонт и обслуживание оборудования. Платформа предоставляет широкий спектр инструментов и функциональных возможностей, которые помогают компаниям и организациям эффективно управлять своими операциями.



The screenshot displays the ServiceNow 'Change Requests' page. On the left is a navigation menu with options like 'Self-Service', 'Service Desk', 'Incident', 'Problem', 'Change', 'Create New', 'Open', 'Closed', 'All', 'Overview', 'Standard Change', 'Standard Change Catalog', 'My Proposals', 'Open Proposals', 'All Templates', 'Schedules', 'Change Schedule', 'Administration', 'Change Properties', and 'Risk Conditions'. The main area shows a table of change requests with columns for 'Number', 'Short description', 'Approval', 'Type', 'State', 'Planned start date', 'Planned end date', and 'Assigned to'. The table contains 14 rows of data, including requests for rolling back Oracle Version, upgrading to Oracle 11i, installing a new PBX, adding a 100 Gb drive, R&D requests, installing new Cisco, applying patches, and Java Application Server changes.

Number	Short description	Approval	Type	State	Planned start date	Planned end date	Assigned to
CHG0000001	Rollback Oracle Version	Requested	Normal	New	2015-09-03 16:00:00	2015-09-03 18:00:00	ITIL User
CHG0000004	Upgrade to Oracle 11i	Not Yet Requested	Normal	Review	2015-09-04 00:00:00	2015-09-04 06:00:00	ITIL User
CHG0000005	Install new PBX	Approved	Normal	Implement	2015-09-07 00:00:00	2015-09-10 16:46:31	David.Loo
CHG0000006	Put another 100 Gb drive on the 2nd floor Server	Not Yet Requested	Normal	Scheduled	2015-09-11 23:00:00	2015-09-11 23:45:00	ITIL User
CHG0000007	R&D wants to know what it'd cost to switch them over to Linux desktops	Rejected	Normal	Authorize	2015-09-12 16:00:00	2015-09-12 20:00:00	ITIL User
CHG0000008	Install new Cisco	Requested	Normal	Authorize	2015-09-09 12:30:00	2015-09-09 17:30:00	ITIL User
CHG0000009	Apply patches 10.2.0.1 to 10.2.0.3	Approved	Normal	New	2015-09-07 03:00:00	2015-09-07 08:00:15	Bow.Ruggieri
CHG0000010	Java Application Server change	Approved	Normal	New	2015-10-02 03:00:00	2015-10-02 08:00:00	Don.Goodliffe
CHG0000011	Another Java Application Server change	Approved	Normal	New	2016-04-16 03:00:00	2016-04-17 06:00:00	David.Loo
CHG0000012	Java App Server	Approved	Normal	New	2016-04-08 18:00:00	2016-04-08 20:00:00	Charlie.Whithamson
CHG0000013	Oracle FLX (Java App Dependency)	Approved	Normal	New	2016-04-17 12:00:00	2016-04-17 14:00:00	Fred.Luddy
CHG0000014	CMS App FLX (depends on Java Apps FLX)	Approved	Normal	New	2016-04-19 01:00:00	2016-04-19 02:59:59	Christen.Mitchell

Рисунок 1 – Интерфейс ServiceNow

Основные функции ServiceNow:

1. Управление заявками на обслуживание:

- Регистрация и отслеживание заявок: пользователи могут создавать заявки на ремонт и обслуживание оборудования, отслеживать их статус и получать уведомления о прогрессе выполнения.

- Автоматизация рабочих процессов: автоматизация обработки заявок с использованием встроенных рабочих процессов, которые могут быть настроены в соответствии с потребностями организации.

- Приоритизация и маршрутизация: заявки могут автоматически приоритизироваться и маршрутизироваться к соответствующим специалистам или группам в зависимости от типа проблемы, приоритета и других критериев.

- Управление инцидентами и проблемами: интеграция управления инцидентами и проблемами позволяет быстрее выявлять и устранять корневые причины неисправностей.

2. Управление конфигурациями и активами:

- Инвентаризация оборудования: ведение базы данных обо всем оборудовании и его конфигурациях, что позволяет точно учитывать все активы и их состояние.

- Отслеживание изменений: автоматический учет всех изменений в конфигурациях оборудования, что помогает избежать непредвиденных сбоев и проблем.

3. Управление изменениями:

- Процессы управления изменениями: поддержка процессов планирования, утверждения и внедрения изменений в инфраструктуру, что помогает минимизировать риски и повышает стабильность работы системы.

- Оценка рисков: анализ рисков, связанных с изменениями, и автоматическое создание плана действий по их минимизации.

4. Управление знаниями:

- База знаний: создание и поддержка базы знаний, содержащей статьи, инструкции и руководства по устранению типичных проблем и выполнению задач.

- Доступ к знаниям: пользователи и специалисты могут быстро находить необходимую информацию для решения проблем и выполнения задач.

5. Управление активами:

- Отслеживание жизненного цикла активов: управление всеми этапами жизненного цикла оборудования от приобретения до утилизации.

- Контроль затрат: отслеживание затрат на оборудование и обслуживание, что позволяет оптимизировать бюджеты и расходы.

6. Отчеты и аналитика:

- Генерация отчетов: создание разнообразных отчетов о заявках, инцидентах, проблемах, изменениях и других аспектах работы системы.

- Анализ данных: аналитические инструменты для выявления тенденций, проблемных областей и возможностей для улучшения.

Преимущества ServiceNow:

- Интеграция и масштабируемость: ServiceNow легко интегрируется с другими системами и приложениями, что позволяет расширять функциональность и адаптироваться под нужды конкретной организации.

- Удобный интерфейс: интуитивно понятный интерфейс облегчает работу пользователей и администраторов.

- Облачная архитектура: облачная модель обеспечивает высокую доступность, безопасность и отказоустойчивость системы.

- Автоматизация: возможность автоматизации многих процессов снижает затраты времени и человеческих ресурсов.

- Поддержка мобильных устройств: доступ к системе с мобильных устройств позволяет пользователям работать с заявками и получать уведомления в любое время и в любом месте.

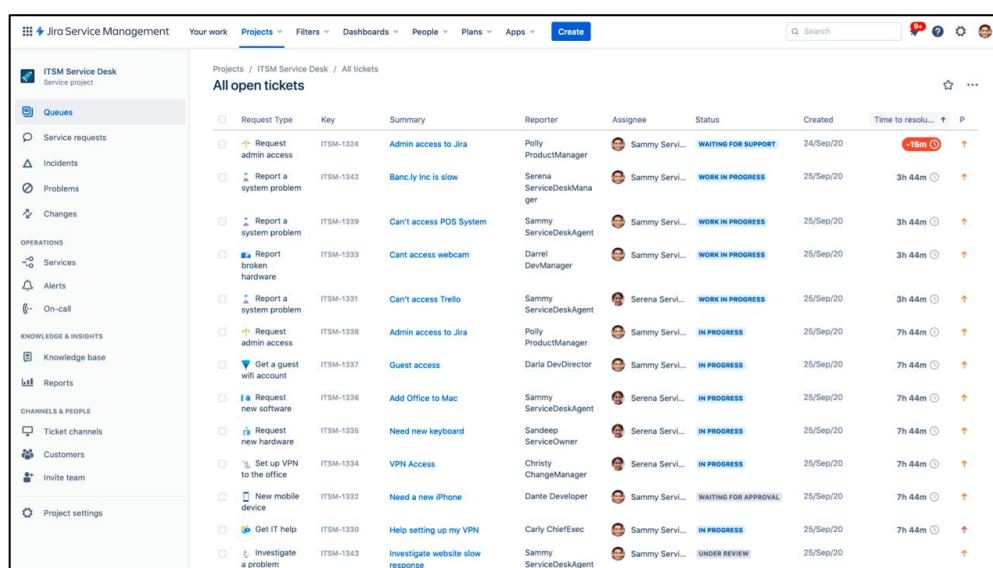
Недостатки ServiceNow:

- Лицензирование: стоимость лицензий на использование ServiceNow может быть достаточно высокой, особенно для небольших компаний или организаций с ограниченным бюджетом.
- Подписка: модель подписки требует регулярных платежей, что может стать значительным финансовым обязательством.
- Сложность конфигурации: настройка и адаптация ServiceNow под конкретные нужды организации могут требовать значительных усилий и специализированных знаний.
- Время внедрения: процесс внедрения может занять много времени, особенно в крупных организациях с комплексной ИТ-инфраструктурой.
- Необходимость обучения: пользователи и администраторы могут столкнуться с необходимостью прохождения обучения для эффективного использования всех возможностей платформы.
- Технические знания: требуются определенные технические знания для настройки и администрирования системы.
- Доступность: так как ServiceNow — облачная платформа, доступ к системе зависит от наличия интернет-соединения. Проблемы с интернетом могут привести к временному недоступности системы.
- Проблемы с задержкой: в зависимости от качества интернет-соединения могут возникать задержки в работе системы.
- Поддержка: хотя ServiceNow предоставляет поддержку, получение помощи может занять время, особенно если требуются сложные технические решения.
- Обновления: регулярные обновления системы могут вызывать временные неудобства и требуют времени для адаптации к новым версиям.
- Ограничения платформы: несмотря на широкие возможности кастомизации, есть определенные ограничения платформы, которые могут не позволить реализовать специфические требования без дополнительных доработок.

– Необходимость в сторонних решениях: для некоторых специфических задач может потребоваться использование дополнительных модулей или сторонних решений, что увеличивает сложность и стоимость системы.

1.2.2 Jira Service Desk

Jira Service Desk — это мощный инструмент для управления заявками и проектами, разработанный компанией Atlassian. Он широко используется в ИТ-индустрии и помогает организациям эффективно управлять запросами на обслуживание, инцидентами, проблемами и изменениями. Jira Service Desk интегрируется с Jira Software, что позволяет объединить управление заявками и разработку программного обеспечения в единой экосистеме.



The screenshot displays the Jira Service Desk interface. On the left is a sidebar with navigation options like 'Queues', 'Service requests', 'Incidents', 'Problems', 'Changes', 'Operations', 'Alerts', 'On-call', 'Knowledge base', 'Reports', 'Channels & People', 'Ticket channels', 'Customers', 'Invite team', and 'Project settings'. The main area shows a list of 'All open tickets' with columns for Request Type, Key, Summary, Reporter, Assignee, Status, Created, and Time to resolve. The table contains 14 rows of ticket data.

Request Type	Key	Summary	Reporter	Assignee	Status	Created	Time to resolve
Request admin access	ITSM-1324	Admin access to Jira	Polly ProductManager	Sammy Servi...	WAITING FOR SUPPORT	24/Sep/20	<15m
Report a system problem	ITSM-1342	Banc.ly Inc is slow	Serena ServiceDeskManager	Sammy Servi...	WORK IN PROGRESS	25/Sep/20	3h 44m
Report a system problem	ITSM-1329	Can't access POS System	Sammy ServiceDeskAgent	Sammy Servi...	WORK IN PROGRESS	25/Sep/20	3h 44m
Report broken hardware	ITSM-1323	Can't access webcam	Darrel DevManager	Sammy Servi...	WORK IN PROGRESS	25/Sep/20	3h 44m
Report a system problem	ITSM-1331	Can't access Trello	Sammy ServiceDeskAgent	Serena Servi...	WORK IN PROGRESS	25/Sep/20	3h 44m
Request admin access	ITSM-1328	Admin access to Jira	Polly ProductManager	Sammy Servi...	IN PROGRESS	25/Sep/20	7h 44m
Get a guest wifi account	ITSM-1327	Guest access	Darla DevDirector	Sammy Servi...	IN PROGRESS	25/Sep/20	7h 44m
Request new software	ITSM-1326	Add Office to Mac	Sammy ServiceDeskAgent	Serena Servi...	IN PROGRESS	25/Sep/20	7h 44m
Request new hardware	ITSM-1325	Need new keyboard	Sandeep ServiceOwner	Serena Servi...	IN PROGRESS	25/Sep/20	7h 44m
Set up VPN to the office	ITSM-1324	VPN Access	Christy ChangeManager	Serena Servi...	IN PROGRESS	25/Sep/20	7h 44m
New mobile device	ITSM-1322	Need a new iPhone	Dante Developer	Sammy Servi...	WAITING FOR APPROVAL	25/Sep/20	7h 44m
Get IT help	ITSM-1320	Help setting up my VPN	Carly ChiefExec	Sammy Servi...	IN PROGRESS	25/Sep/20	7h 44m
Investigate a problem	ITSM-1343	Investigate website slow response	Sammy ServiceDeskAgent	Sammy Servi...	UNDER REVIEW	25/Sep/20	

Рисунок 2 – Интерфейс Jira Service Desk

Основные функции Jira Service Desk:

1. Управление заявками:

- Портал самообслуживания: пользователи могут создавать и отслеживать заявки через удобный портал самообслуживания.
- Каталог услуг: возможность создания каталога услуг, где пользователи могут выбирать необходимые услуги и запрашивать их.

- Шаблоны заявок: настраиваемые шаблоны заявок позволяют стандартизировать процессы подачи заявок и ускорить их обработку.

2. Управление инцидентами:

- Регистрация инцидентов: быстрая регистрация инцидентов с возможностью их классификации и приоритизации.

- Отслеживание инцидентов: возможность отслеживать статус инцидентов и получать уведомления о ходе их решения.

- Эскалация: автоматическая эскалация инцидентов в случае превышения времени отклика или решения.

3. Управление проблемами:

- Идентификация проблем: анализ инцидентов для выявления корневых причин и проблем.

- Регистрация и отслеживание проблем: создание записей о проблемах и отслеживание их статуса до полного разрешения.

- Связь с инцидентами: возможность связывать проблемы с соответствующими инцидентами для комплексного решения.

4. Управление изменениями:

- Запросы на изменение: создание и управление запросами на изменение, включая их оценку и утверждение.

- Процессы управления изменениями: настраиваемые рабочие процессы для управления изменениями и минимизации рисков.

- Планирование изменений: инструменты для планирования и координации изменений в ИТ-инфраструктуре.

5. Управление знаниями:

- База знаний: создание и поддержка базы знаний с полезными статьями и инструкциями.

- Доступ к знаниям: пользователи могут легко находить информацию, необходимую для решения типичных проблем и выполнения задач.

- Автоматизация ответов: автоматическое предложение статей из базы знаний при создании заявок, что помогает пользователям самостоятельно решать проблемы.

6. Отчеты и аналитика:

- Генерация отчетов: создание различных отчетов о заявках, инцидентах, проблемах и изменениях.

- Информационные панели: настраиваемые информационные панели для визуализации ключевых показателей и отслеживания эффективности работы службы поддержки.

- Анализ данных: инструменты для анализа данных и выявления тенденций, что помогает улучшать качество обслуживания.

Преимущества Jira Service Desk:

- Интеграция с Jira Software: плотная интеграция с Jira Software позволяет объединить управление заявками и разработку программного обеспечения в единой экосистеме, что упрощает взаимодействие между командами разработки и поддержки.

- Гибкость и настраиваемость: Jira Service Desk предоставляет широкий спектр возможностей для настройки рабочих процессов, форм заявок и шаблонов, что позволяет адаптировать систему под конкретные нужды организации.

- Удобство использования: интуитивно понятный интерфейс и портал самообслуживания делают работу с системой удобной и эффективной как для пользователей, так и для специалистов службы поддержки.

- Автоматизация процессов: возможность автоматизации рутинных задач и процессов, таких как эскалация инцидентов, уведомления и обработка заявок, что повышает производительность и снижает нагрузку на сотрудников.

- Расширяемость и интеграции: широкий выбор плагинов и интеграций с другими системами и сервисами, что позволяет расширять

функциональность и интегрировать Jira Service Desk в существующую ИТ-инфраструктуру.

Недостатки Jira Service Desk:

- Высокая стоимость: стоимость лицензий на использование Jira Service Desk может быть значительной, особенно для крупных организаций или организаций с большим количеством пользователей.

- Сложность настройки: первоначальная настройка и адаптация системы под специфические потребности организации может потребовать значительных усилий и времени, а также наличие специалистов с соответствующими знаниями.

- Необходимость обучения: для эффективного использования всех возможностей Jira Service Desk может потребоваться обучение пользователей и администраторов, что требует дополнительных ресурсов и времени.

- Производительность: в больших масштабах система может испытывать проблемы с производительностью, особенно при одновременной работе большого количества пользователей и обработке большого объема данных.

1.2.3 Freshdesk

Freshdesk — это облачная система управления заявками на обслуживание и поддержки клиентов, разработанная компанией Freshworks. Она помогает организациям эффективно управлять запросами клиентов, инцидентами и проблемами, обеспечивая высокий уровень обслуживания и удовлетворенности пользователей.

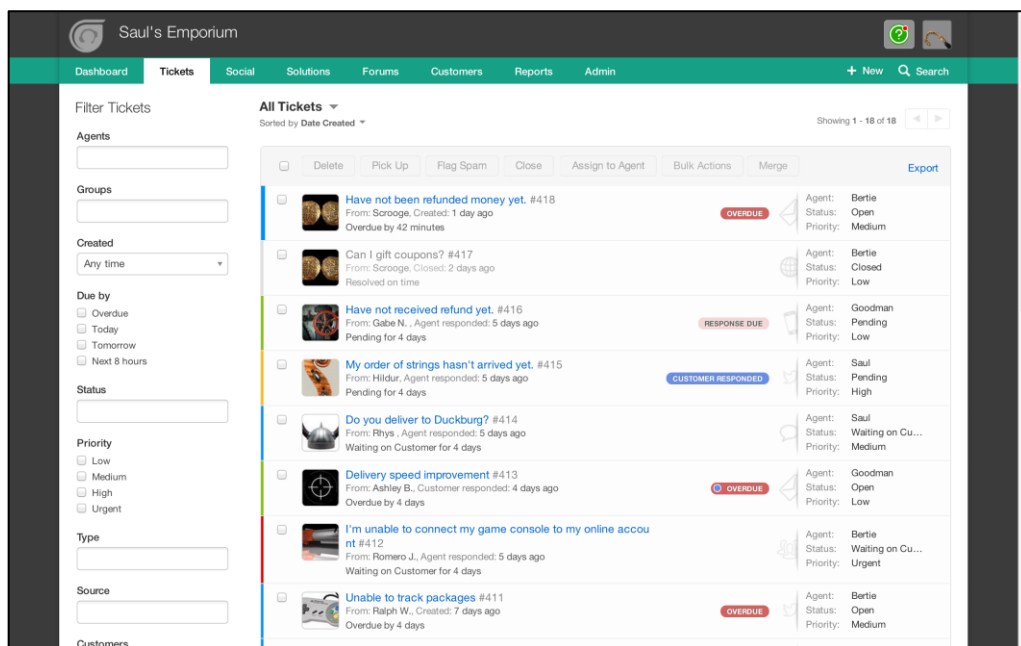


Рисунок 3 – Интерфейс Freshdesk

Основные функции Freshdesk:

1. Управление заявками:

- Централизованное управление заявками: все запросы от клиентов собираются в единую систему, что позволяет легко их отслеживать и управлять ими.
- Автоматизация создания заявок: автоматическое создание заявок из различных источников, таких как электронная почта, веб-формы, телефонные звонки, социальные сети и чаты.
- Приоритизация и категоризация: возможность назначения приоритетов и категорий заявкам для оптимальной их обработки.

2. Автоматизация рабочих процессов:

- Автоматические правила: создание правил автоматической маршрутизации заявок, эскалации и уведомлений.
- Макросы и шаблоны: использование макросов и шаблонов для ускорения обработки типичных запросов.

3. Многофункциональный интерфейс:

- Поддержка различных каналов связи: управление заявками, поступающими через электронную почту, телефон, чат, социальные сети и веб-портал.

- Единый интерфейс: все каналы связи интегрированы в единый интерфейс, что облегчает работу операторов поддержки.

4. Управление знаниями:

- База знаний: создание и поддержка базы знаний с ответами на часто задаваемые вопросы, инструкциями и статьями.

- Самообслуживание: пользователи могут искать информацию и решать свои проблемы самостоятельно, что снижает нагрузку на службу поддержки.

5. Отчеты и аналитика:

- Генерация отчетов: создание различных отчетов по заявкам, производительности команды, времени отклика и другим ключевым показателям.

- Информационные панели: настраиваемые информационные панели для визуализации данных и отслеживания эффективности работы службы поддержки.

6. Интеграции и расширяемость:

- Интеграция с другими системами: поддержка интеграции с различными CRM, ERP, маркетинговыми и бухгалтерскими системами.

- API и приложения: использование API для создания пользовательских интеграций и приложений, доступных в маркете Freshdesk.

Преимущества Freshdesk:

- Удобный и интуитивно понятный интерфейс: простой и понятный интерфейс облегчает работу как для операторов поддержки, так и для конечных пользователей.

- Широкие возможности автоматизации: множество инструментов для автоматизации процессов обработки заявок, что повышает эффективность работы и снижает время отклика.

- Многофункциональная поддержка: поддержка различных каналов связи, включая электронную почту, телефон, чат, социальные сети и веб-портал, что позволяет клиентам выбирать удобный для них способ связи.

- Гибкость и масштабируемость: система легко масштабируется под нужды организаций различного размера, от небольших предприятий до крупных корпораций.

- Интеграции: поддержка множества интеграций с популярными системами и сервисами, что позволяет расширить функциональность Freshdesk и интегрировать его в существующую ИТ-инфраструктуру.

Недостатки Freshdesk:

- Стоимость: хотя Freshdesk предлагает различные тарифные планы, включая бесплатный, для доступа к более продвинутым функциям и интеграциям требуется подписка на платные тарифы, что может быть дорого для некоторых организаций.

- Сложности с кастомизацией: несмотря на наличие множества настроек и возможностей интеграции, некоторые пользователи могут столкнуться с ограничениями при попытке глубокой кастомизации системы под свои специфические нужды.

- Зависимость от интернета: как и любая облачная платформа, Freshdesk требует стабильного интернет-соединения для работы, что может быть проблемой в случае перебоев с интернетом.

- Необходимость обучения: для полного освоения всех возможностей Freshdesk может потребоваться время и обучение, особенно для новых пользователей и администраторов.

На основе анализа систем управления заявками — ServiceNow, Jira Service Desk и Freshdesk — были выявлены их основные недостатки: высокая стоимость, сложность настройки и внедрения, необходимость специализированного обучения, зависимость от стабильного интернет-соединения. В разрабатываемом приложении «Служба поддержки» планирую устранить эти недостатки, предложив более доступную и гибкую

систему лицензирования, простую в настройке и использовании, с возможностью работы без выхода в глобальную сеть. Это позволит организациям любого размера эффективно управлять заявками на ремонт и обслуживание техники.

1.3 Техническое задание

1.3.1 Описание функциональных требований

Основные функции приложения «Служба поддержки» включают:

- Создание и управление заявками на ремонт: пользователи могут создавать заявки на ремонт оборудования, указывая серийный номер и описание проблемы. Система автоматически присваивает каждой заявке уникальный идентификатор и сохраняет информацию о пользователе, который её создал. Заявка включает серийный номер оборудования, описание проблемы и текущий статус.
- Регистрация и аутентификация пользователей: система должна обеспечивать безопасную регистрацию новых пользователей и аутентификацию существующих при входе в систему.
- Обработка заявок: системный администратор имеет возможность просматривать, редактировать и изменять статус заявок. При необходимости он может запрашивать дополнительные сведения у пользователя, создавшего заявку.
- Исполнение заявок: администратор выполняет ремонт оборудования и организует закупку необходимых запчастей. В случае отсутствия деталей создается заказ на их приобретение, который направляется на одобрение директору. После выполнения всех необходимых работ заявка закрывается.
- Отчётность и информирование: по каждой завершённой заявке системный администратор предоставляет отчёт о проделанной работе. Отчёт включает тип ремонта, затраченное время, использованные ресурсы и суммарные расходы. Отчёт сохраняется в системе.

- Мониторинг и анализ: система предоставляет системному администратору возможность отслеживать количество, время и ресурсы, затраченные на выполнение заявок. Информация визуализируется в виде диаграмм и может быть распечатана.

- Управление данными о пользователях и оборудовании: администратор может добавлять, редактировать и удалять записи о пользователях и оборудовании.

1.3.2 Описание нефункциональных требований

Система должна обрабатывать заявки с минимальными задержками и поддерживать масштабируемость при увеличении числа пользователей и объема данных.

Требования к производительности включают:

- Время отклика системы должно быть минимальным для большинства операций.

- Система должна поддерживать обработку заявок в объеме, необходимом для обеспечения бесперебойной работы.

Интерфейс системы должен быть интуитивно понятным и удобным для использования всеми категориями пользователей.

Требования к удобству использования включают:

- Простая и логичная навигация по интерфейсу.
- Наличие подсказок и инструкций для пользователей.

1.3.3 Описание архитектуры системы

Архитектура клиент-сервер: Приложение «Служба поддержки» реализовано по архитектуре клиент-сервер, где серверная часть и база данных находятся в СУБД PostgreSQL.

Компоненты архитектуры:

1. Серверная часть:

- Реализована с использованием функций и триггеров PostgreSQL.

- Обрабатывает бизнес-логику приложения, включая создание, обновление и удаление записей в базе данных.

- Управляет безопасностью и аутентификацией пользователей.

2. Клиентская часть:

- Реализована с использованием JavaFX для обеспечения графического пользовательского интерфейса.

- Обеспечивает пользователям доступ к функциям системы, таким как создание заявок, просмотр статуса заявок, управление данными о пользователях и оборудовании.

- Взаимодействует с серверной частью через SQL-запросы.

3. База данных:

- Используется PostgreSQL для хранения всех данных о заявках, пользователях и оборудовании.

- Содержит таблицы для заявок, пользователей, оборудования, отчетов и заказов.

- Поддерживает функции и триггеры для реализации бизнес-логики на серверной стороне.

Взаимодействие компонентов:

- Клиентское приложение отправляет SQL-запросы к серверной части, реализованной в PostgreSQL.

- Серверная часть обрабатывает запросы, выполняет необходимые операции с данными и возвращает результаты клиентскому приложению.

- База данных хранит всю необходимую информацию и обеспечивает целостность данных через механизмы транзакций и ограничений целостности.

Эта архитектура обеспечивает высокую производительность и надежность системы, а также упрощает процесс разработки и поддержки приложения.

2 Используемые технологии и инструменты

2.1 Обоснование используемого языка программирования

Язык программирования Java был выбран за счет многих преимуществ. Одним из ключевых является кроссплатформенность. Программы могут работать на различных платформах благодаря Java Virtual Machine (JVM), что снижает затраты на разработку и поддержку [3].



Рисунок 4 – Логотип Java

Java также известна своей надежностью и безопасностью. Строгая система типизации предотвращает ошибки на этапе компиляции, а автоматическое управление памятью снижает вероятность утечек. Встроенная система контроля доступа защищает данные [4].

Java предоставляет инструменты для создания масштабируемых приложений, от мобильных до крупных корпоративных систем. Поддержка многопоточности позволяет эффективно использовать ресурсы многопроцессорных систем [5].

Java обладает множеством стандартных библиотек, охватывающих широкий спектр задач, от работы с базами данных до веб-разработки. Это позволяет разработчикам сосредоточиться на бизнес-задачах. Сторонние библиотеки предоставляют дополнительные возможности [6].

2.2 Выбор системы управления базами данных

PostgreSQL была выбрана для этого проекта благодаря её выдающейся производительности, надежности и поддержке стандартов SQL. Эта система

управления базами данных обеспечивает высокую эффективность обработки запросов за счёт оптимизированных алгоритмов, что позволяет быстро и точно выполнять сложные операции [7].



Рисунок 5 – Логотип PostgreSQL

Одним из ключевых преимуществ PostgreSQL является её надёжность. Система поддерживает транзакции и механизмы восстановления данных, что гарантирует целостность и безопасность информации даже в случае сбоев. Это особенно важно для критически важных приложений, где требуется максимальная защита данных.

2.3 Обоснование используемых фреймворков и библиотек

Для разработки данного проекта выбран JavaFX как основной фреймворк для создания пользовательского интерфейса. Он предоставляет богатые возможности для создания интерактивных и визуально привлекательных приложений.



Рисунок 6 – Логотип JavaFX

JavaFX используется для разработки современных и интуитивно понятных пользовательских интерфейсов [8]. Этот фреймворк обеспечивает гибкость и мощные инструменты для создания кроссплатформенных

приложений, которые могут работать на различных операционных системах, включая Windows, macOS и Linux. Это позволяет создавать единый код, который будет функционировать на всех поддерживаемых платформах, что значительно снижает затраты на разработку и тестирование.

Основные преимущества JavaFX:

- Современный интерфейс: позволяет создавать стильные и современные пользовательские интерфейсы, используя FXML для описания UI и CSS для его стилизации. Это позволяет разработчикам разделить логику приложения от его визуального представления, что облегчает разработку и поддержку.

- Поддержка мультимедиа: поддерживает работу с графикой, аудио и видео, что позволяет создавать богатый пользовательский опыт. Вы можете легко интегрировать мультимедийные элементы в свои приложения, что делает их более интерактивными и привлекательными.

- Анимации и эффекты: встроенные возможности JavaFX для создания анимаций и визуальных эффектов позволяют оживить пользовательский интерфейс. Это помогает сделать приложение более динамичным и улучшить взаимодействие с пользователем.

- Интеграция с Java: JavaFX легко интегрируется с остальными компонентами на Java, что позволяет использовать богатый экосистему библиотек и фреймворков, доступных для Java-разработчиков. Это упрощает процесс разработки и расширяет функциональные возможности приложений.

- Поддержка Scene Builder: JavaFX поддерживает инструмент Scene Builder, который позволяет визуальнo разрабатывать пользовательские интерфейсы. Это удобный инструмент для быстрого создания и прототипирования UI без необходимости писать код вручную.

2.4 Выбор редактора кода

Для разработки данного проекта выбрана IntelliJ IDEA благодаря своим мощным инструментам, поддержке различных фреймворков и библиотек,

удобному интерфейсу и возможностям для отладки кода. IntelliJ IDEA поддерживает широкий спектр языков программирования и фреймворков, таких как Java, Kotlin, Scala, Groovy, а также фреймворки, включая Spring и JavaFX. Это позволяет разработчикам работать с различными технологиями в одном редакторе, упрощая интеграцию и управление проектами.



Рисунок 7 – Логотип IntelliJ IDEA

Интеллектуальный редактор кода в IntelliJ IDEA предоставляет функции автодополнения, рефакторинга и подсветки синтаксиса, что значительно ускоряет процесс написания и улучшает качество кода. Интеграция с системами контроля версий, такими как Git и SVN, позволяет эффективно управлять проектами, отслеживать изменения и работать в команде.

Кроме того, IntelliJ IDEA предлагает удобный интерфейс и мощные инструменты для отладки, включая пошаговое выполнение, точки останова и профилировщики производительности. Эти возможности делают IntelliJ IDEA отличным выбором для разработки современных приложений, обеспечивая удобство, эффективность и высокое качество работы.

2.5 Выбор графического редактора для создания дизайна

Figma была выбрана для проектирования пользовательского интерфейса благодаря своим мощным инструментам для прототипирования и дизайна. Она позволяет создавать удобные и интуитивно понятные интерфейсы, улучшая пользовательский опыт. Доступность через веб-интерфейс устраняет необходимость установки ПО и позволяет работать над проектами с любого устройства.



Рисунок 8 – Логотип Figma

Одним из ключевых преимуществ Figma является возможность совместной работы в реальном времени. Несколько дизайнеров могут одновременно вносить изменения в проект, что повышает продуктивность и ускоряет процесс разработки. Функции комментариев и обсуждений прямо в интерфейсе облегчают коммуникацию и обмен идеями.

Кроссплатформенность Figma обеспечивает гибкость, позволяя работать на любой операционной системе. Интеграция с инструментами, такими как Zeplin, Slack и Jira, упрощает рабочий процесс и улучшает взаимодействие между дизайнерами и разработчиками. Возможность экспорта макетов и ресурсов в различные форматы делает Figma удобным инструментом для создания и передачи дизайнов.

3 Разработка и реализация системы

3.1 Проектирование базы данных

3.1.1 Определение сущностей и связей

Проектирование включает создание схемы базы данных, определение таблиц, полей и связей между ними [9]. Основные таблицы: requests, reports, orders, equipment, members.

Описание таблиц:

1. requests: информация о заявках на ремонт:
 - id (PK): уникальный идентификатор заявки;
 - serial_num (FK): серийный номер оборудования;
 - equip_type: тип оборудования;
 - problem_desc: описание проблемы;
 - request_comments: комментарии по заявке;
 - status: статус заявки (в процессе, завершено);
 - date_start: дата начала обработки заявки;
 - member_id (FK): идентификатор пользователя, создавшего заявку.
2. reports: информация об отчетах по заявкам:
 - request_id (PK, FK): уникальный идентификатор заявки;
 - repair_type: тип ремонта;
 - time: время, затраченное на ремонт;
 - cost: стоимость ремонта;
 - resources: использованные ресурсы;
 - reason: причина поломки;
 - help: дополнительная информация.
3. orders: информация о заказах:
 - id (PK): уникальный идентификатор заказа;
 - request_id (FK): идентификатор заявки;

- resource_type: тип ресурса;
 - resource_name: наименование ресурса;
 - cost: стоимость ресурса.
4. equipment: информация об оборудовании:
- serial_num (PK): серийный номер;
 - equip_name: наименование;
 - equip_type: тип оборудования;
 - condition: состояние;
 - details: дополнительные детали;
 - location: местоположение.
5. members: информация о пользователях.
- id (PK): уникальный идентификатор;
 - name: имя;
 - phone: номер телефона;
 - e_mail: электронная почта;
 - login: логин;
 - pass: пароль;
 - role: роль (администратор, техник).

Взаимосвязи между таблицами:

- requests и equipment: таблица requests связана с таблицей equipment через поле serial_num, указывая на оборудование, для которого создается заявка на ремонт;
- requests и members: таблица requests связана с таблицей members через поле member_id, указывая на пользователя, создавшего заявку;
- reports и requests: таблица reports связана с таблицей requests через поле request_id, указывая на отчет, созданный для конкретной заявки;
- orders и requests: таблица orders связана с таблицей requests через поле request_id, указывая на заказ, связанный с заявкой.

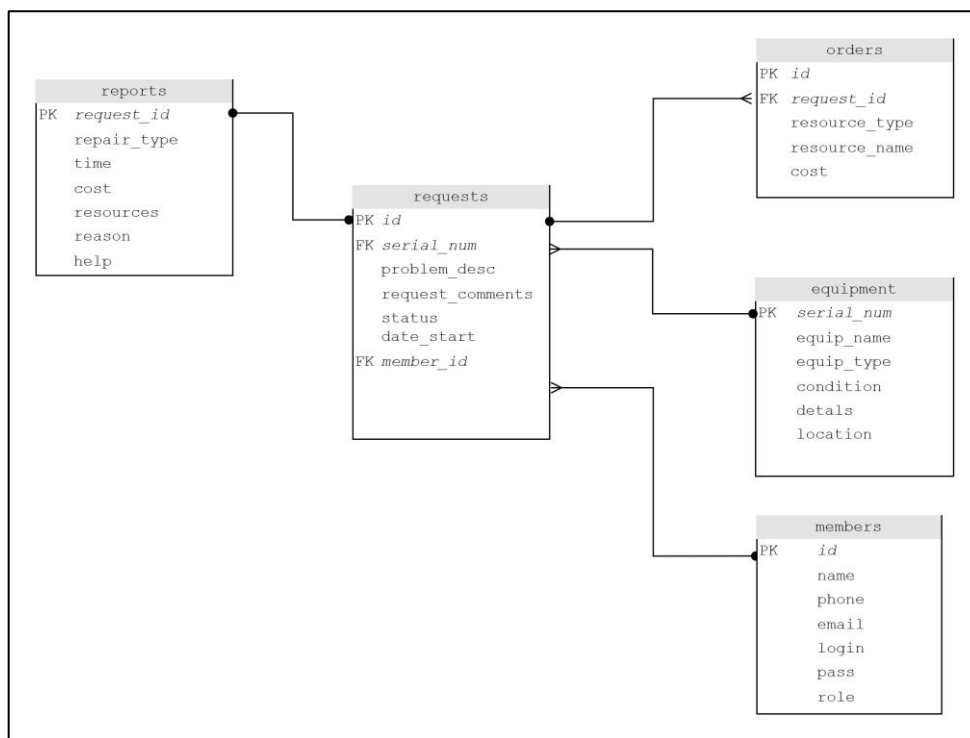


Рисунок 9 – ER диаграмма

3.1.2 Настройка триггеров

В системе управления заявками важно своевременно оповещать пользователей об изменениях в базе данных, таких как создание новых заявок или обновление существующих. Для реализации этой функциональности были разработаны специальные функции и триггеры, которые автоматически отправляют уведомления при выполнении соответствующих операций.

Триггеры представляют собой механизмы, которые автоматически вызывают выполнение определенных действий (в данном случае – функций) в ответ на определенные события в базе данных [10]. В данном случае триггеры настроены на срабатывание после вставки новых строк (для создания заявок) и после обновления существующих строк (для обновления заявок) в таблице заявок. Эти триггеры обеспечивают вызов соответствующих функций, которые отправляют уведомления.

Для уведомления о создании новой заявки используется специальная функция, которая отправляет уведомление в канал уведомлений с

идентификатором новой заявки. Эта функция вызывается специальным триггером, который срабатывает после вставки новой строки в таблицу заявок. Таким образом, каждый раз, когда в таблицу добавляется новая заявка, система автоматически отправляет уведомление, содержащее идентификатор созданной заявки.

Аналогично созданию заявки, для уведомления об обновлении существующей заявки используется другая функция. Эта функция отправляет уведомление в канал уведомлений с идентификатором обновленной заявки. Функция вызывается триггером, который срабатывает после обновления строки в таблице заявок. Каждый раз, когда заявка обновляется, система автоматически отправляет уведомление с идентификатором обновленной заявки.

3.2 Диаграмма активностей

Диаграмма активностей отображает процесс работы через интерфейс приложения. Она включает основные этапы, такие как авторизация, отправка заявок, получение заявок и их обработка. Диаграмма показывает, как пользователь и системный администратор взаимодействуют с приложением, обеспечивая наглядное понимание каждого шага.

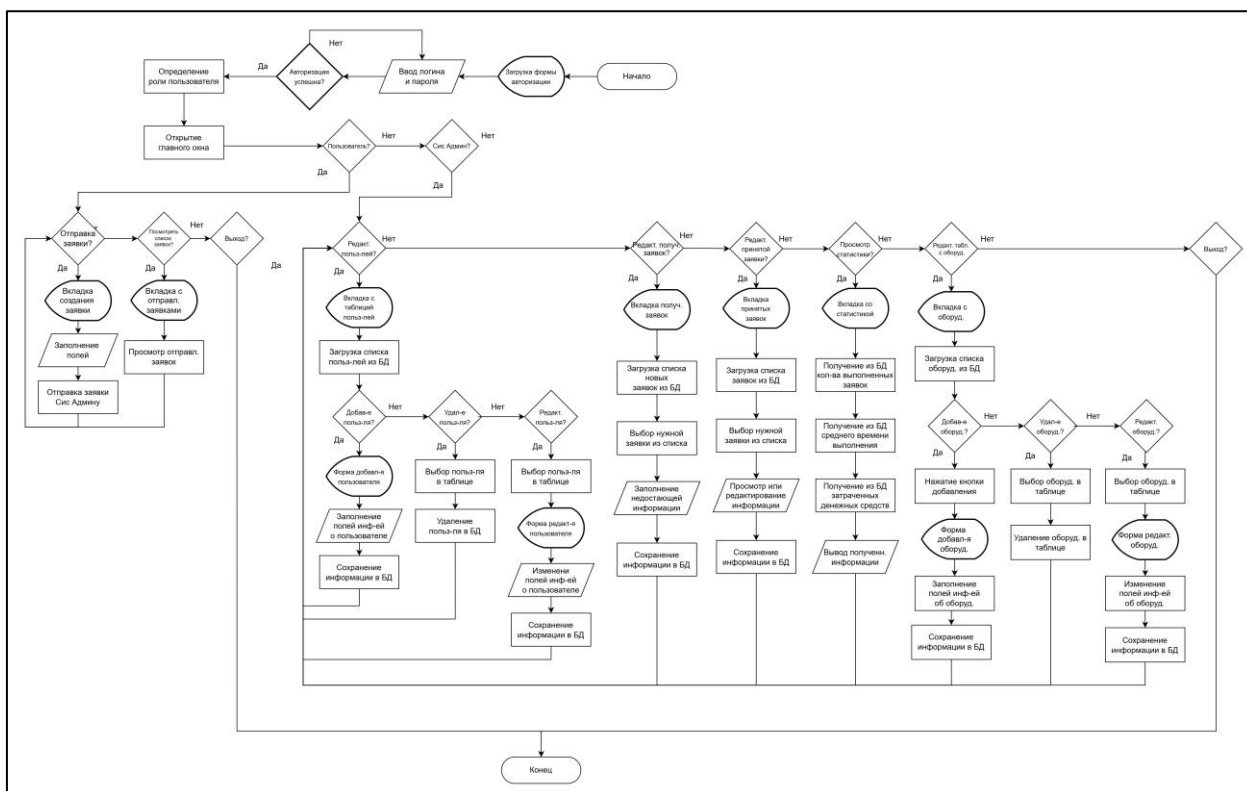


Рисунок 10 – Диаграмма активностей

3.3 Диаграмма вариантов использования

Диаграмма вариантов использования для системы «Служба поддержки» отображает взаимодействие пользователей и системных администраторов с системой. Пользователь может отправлять заявки на ремонт и просматривать их статус. Системный администратор регистрирует и обрабатывает заявки, выполняет ремонт, заказывает запчасти и формирует отчеты. Диаграмма показывает, как каждый участник выполняет свои функции, обеспечивая наглядное понимание процессов службы поддержки.

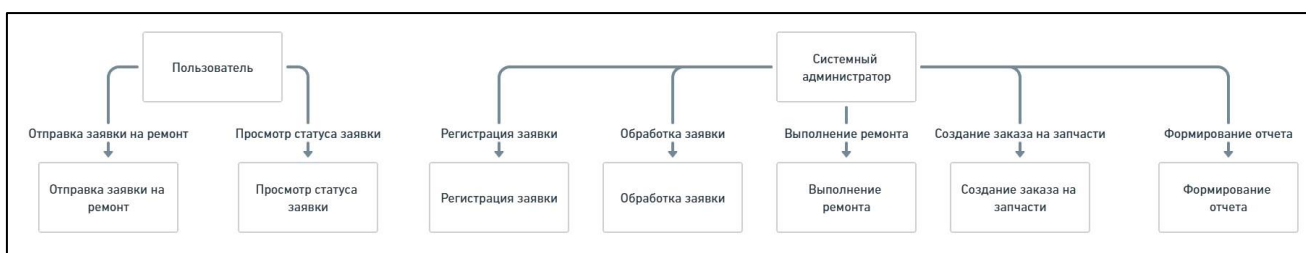


Рисунок 11 – Диаграмма вариантов использования

3.4 Разработка серверной части

Чтобы СУБД была доступна с любого устройства в локальной сети, на компьютере, выступающем в роли сервера для СУБД, нужно выполнить следующие действия.

Открыть проводник. Перейти в папку, куда был установлен PostgreSQL. Открыть файл «pg_hba.conf» с помощью блокнота.

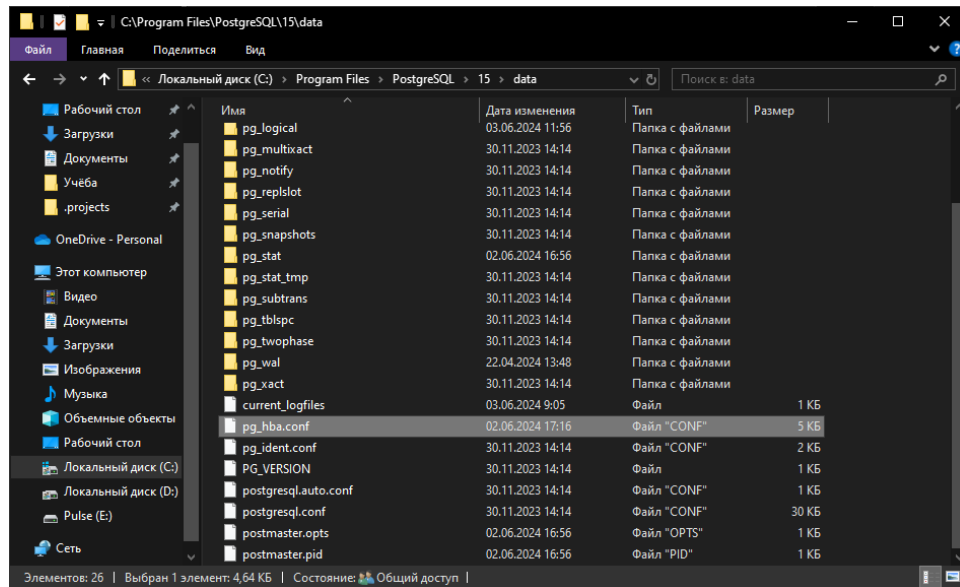


Рисунок 12 – Расположение файла в папке

Добавить запись для разрешения удаленных подключений «host all all 0.0.0.0/0 trust» и сохранить файл.

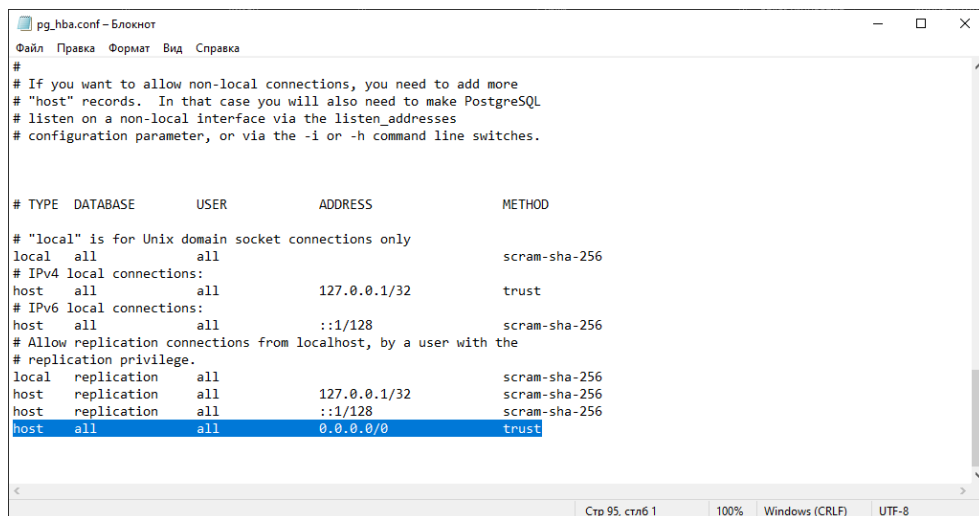


Рисунок 13 – Редактирование файла «pg_hba.conf»

Нажать клавишу «Win + R», ввести «wf.msc» и нажать «Enter».

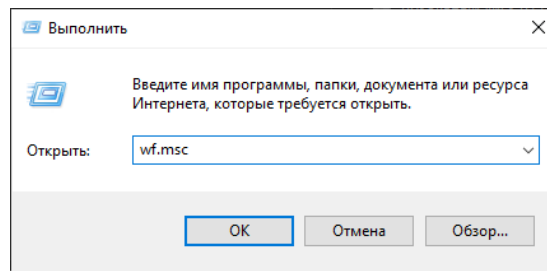


Рисунок 14 – Открытие монитора брандмауэра

В левой части окна нажать на «Правила для входящих подключений», в правой части экрана нажать на «Создать правило».

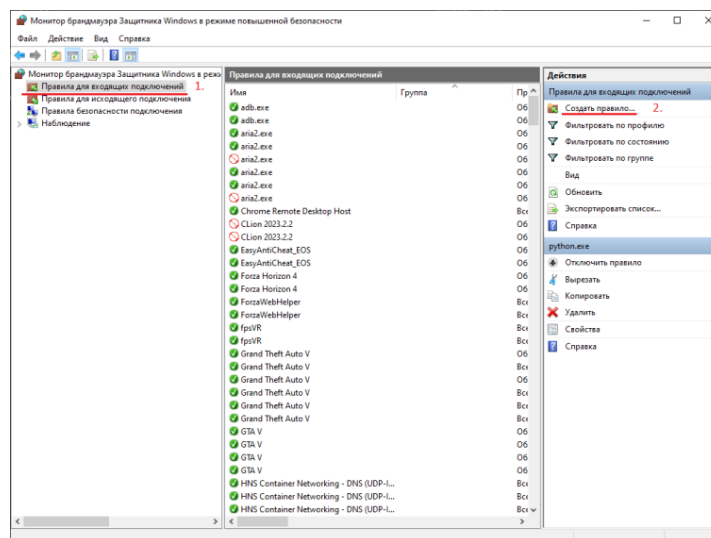


Рисунок 15 – Создание правила для новых входящих подключений

В открывшемся мастере создания правила выбрать «Для порта» и нажать «Далее».

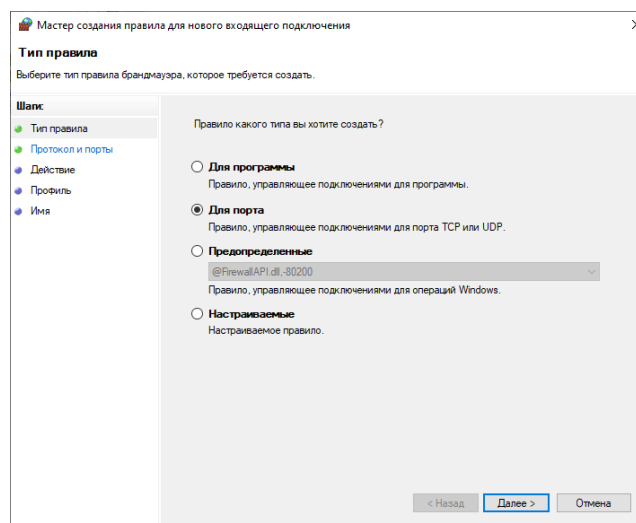


Рисунок 16 – Настройка правила для новых входящих подключений

В протоколе указать «Протокол TCP». В «определенные локальные порты» ввести 8888. Нажать «Далее».

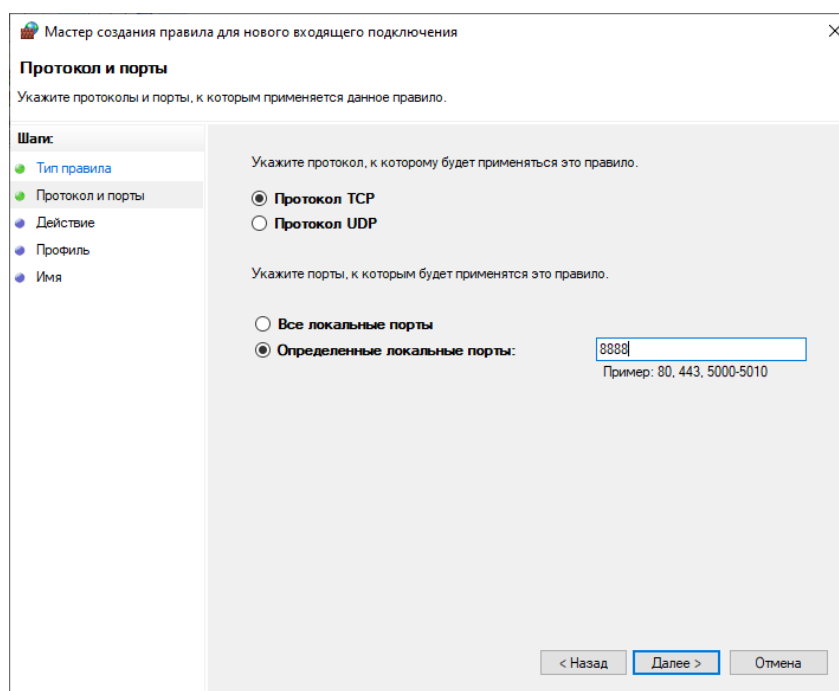


Рисунок 17 – Настройка правила для новых входящих подключений

Выбрать «Разрешить подключение» и нажать «Далее».

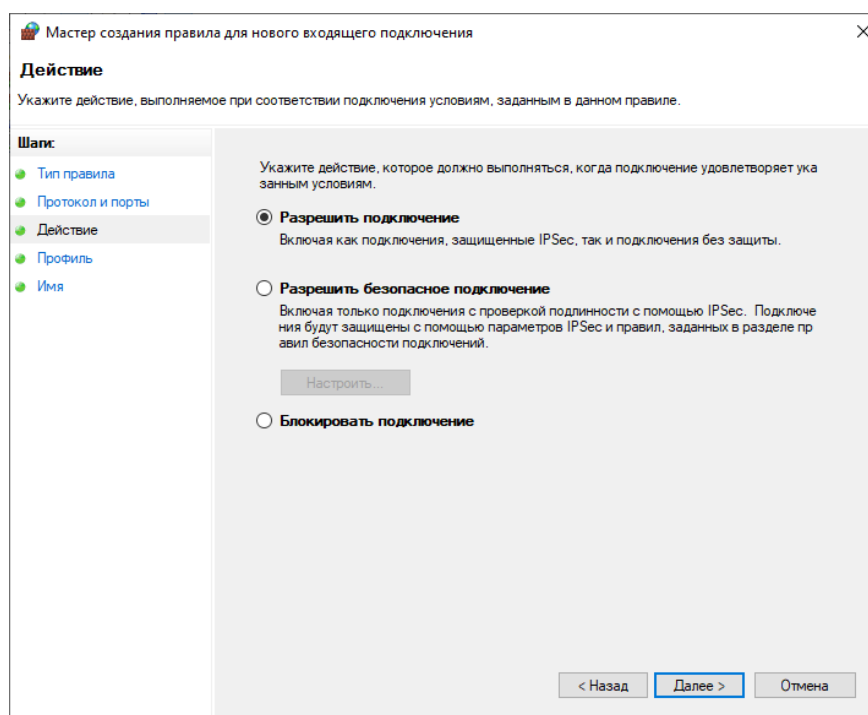


Рисунок 18 – Настройка правила для новых входящих подключений

Выбрать все три профиля («Доменный», «Частный», «Публичный») и нажать «Далее».

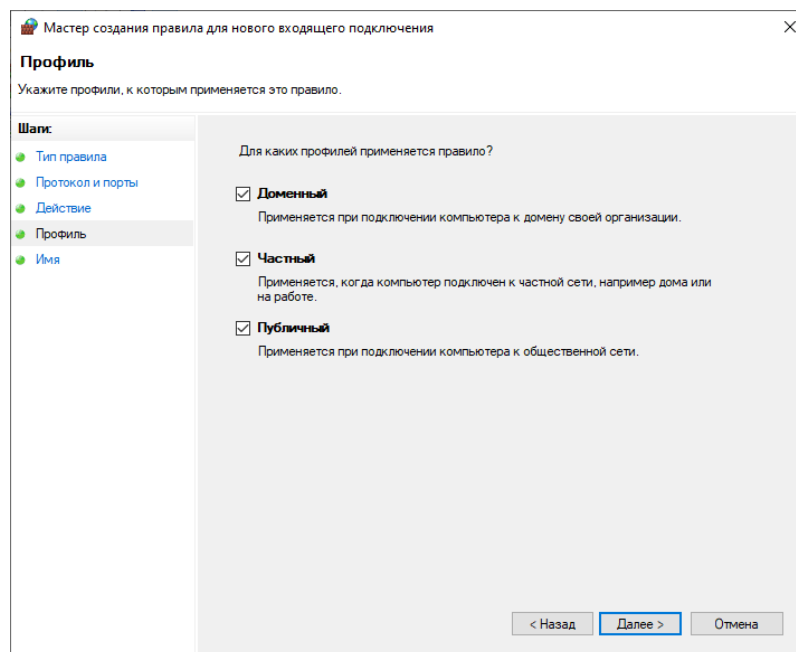


Рисунок 19 – Настройка правила для новых входящих подключений

Задать имя «PostgreSQL», нажать «Готово».

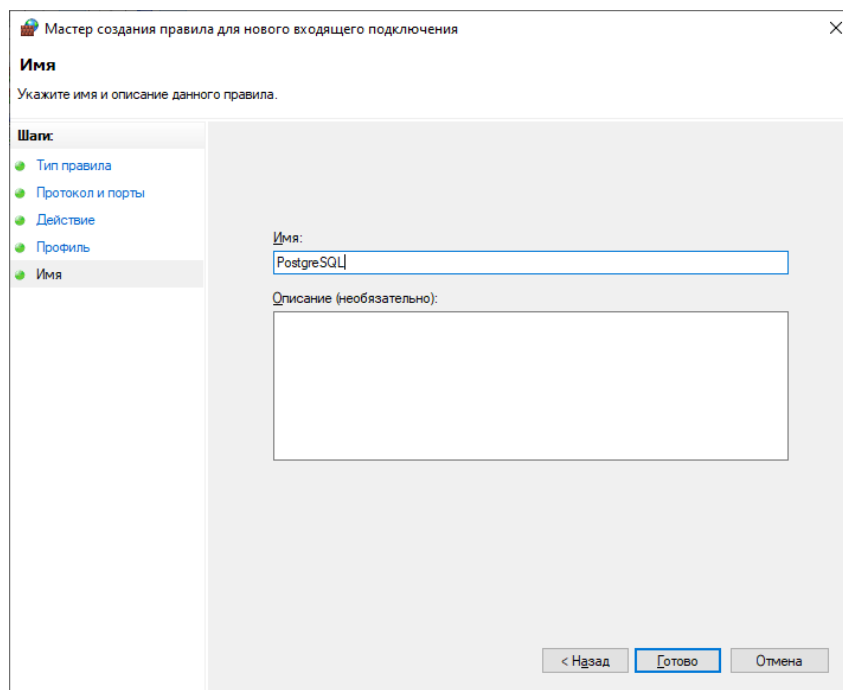


Рисунок 20 – Настройка правила для новых входящих подключений

Нажать «Пуск» правой кнопкой мыши, выбрать «Windows PowerShell (администратор)».

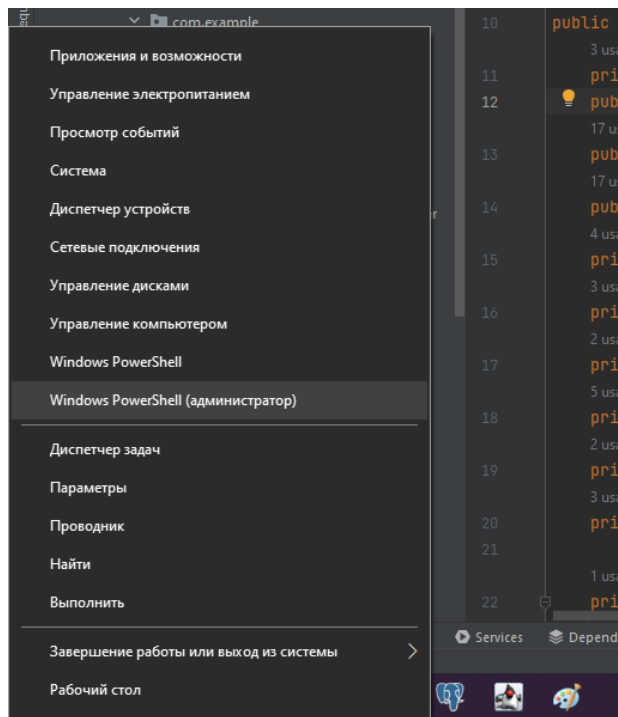


Рисунок 21 – Открытие окна PowerShell

Введение команды «ipconfig» в командную строку и нажать «Enter».

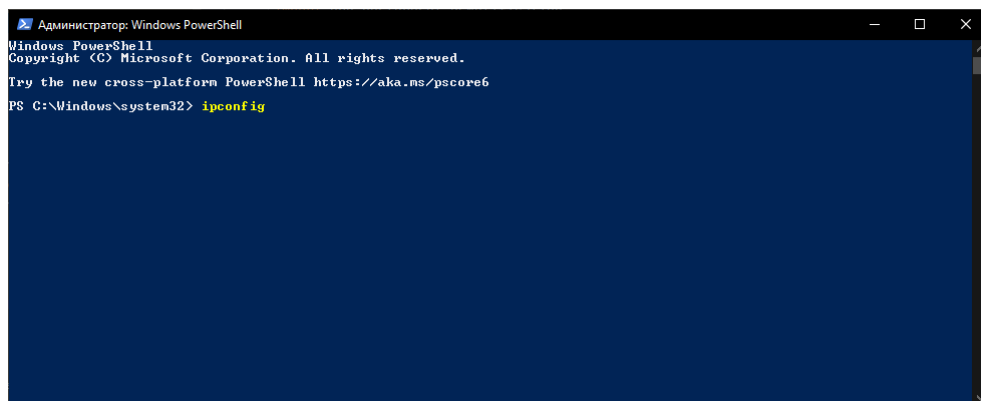


Рисунок 22 –Окно с введенной командой в строку

После выполнения команды, в терминал будут выведены IP адреса. Среди них нужно найти IPv4 адрес используемого сетевого адаптера:

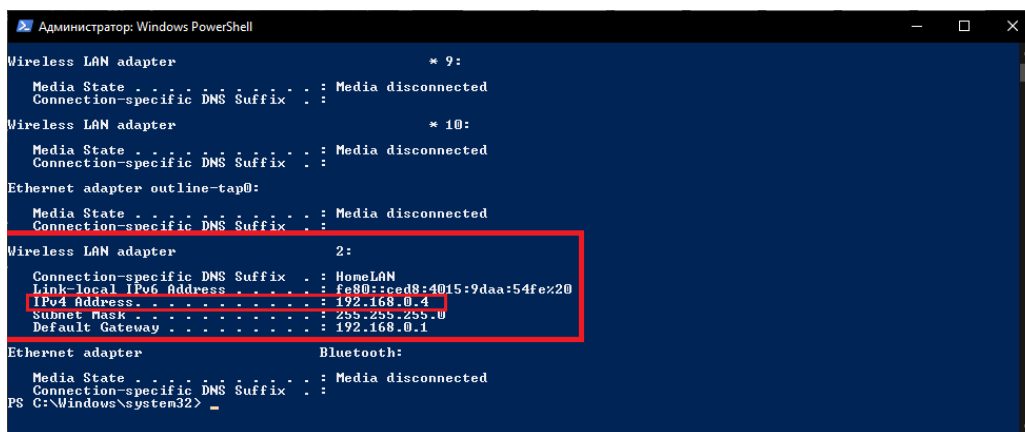


Рисунок 23 – Окно с IP адресами компьютера

После получения IP адреса на сервере, на клиентском устройстве нужно запустить программу и ввести IP адрес сервера:

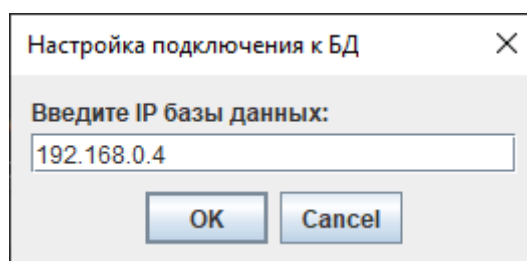


Рисунок 24 – Окно с подключением к базе данных

При успешном подключении выводится диалоговое окно об успешном подключении.

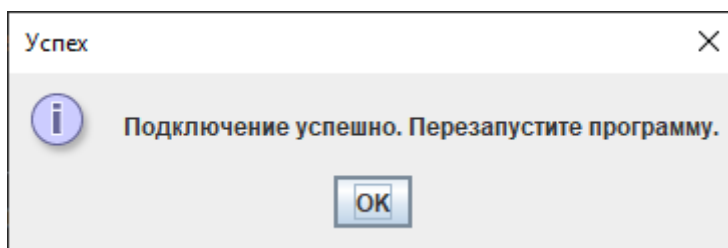


Рисунок 25 – Диалоговое окно

3.5 Разработка клиентской части

3.5.1 Дизайн

На рисунке 26 представлено окно авторизации приложения. Пользователь должен ввести имя пользователя и пароль, чтобы получить доступ к функциональности приложения.

На данном экране есть:

- Поле ввода имени пользователя: в середине окна расположено текстовое поле для ввода логина пользователя.
- Поле ввода пароля: ниже находится текстовое поле для ввода пароля. Введенный пароль отображается в виде точек, что обеспечивает конфиденциальность.
- Кнопка входа: под полями ввода располагается кнопка «Вход». Нажатие на эту кнопку отправляет введенные данные для проверки и предоставляет доступ к основному функционалу приложения.

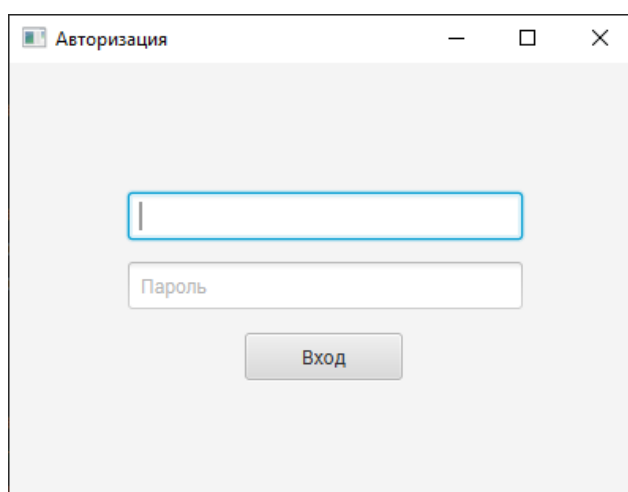


Рисунок 26 – Окно авторизации

На рисунке 27 представлено окно с новыми заявками. Вкладка разделена на несколько основных областей:

- Меню вкладок: в верхней части окна расположены вкладки «Новые заявки», «Все заявки», «Пользователи», «Оборудование», «Заказ запчастей», «Статистика» для администратора, что позволяет быстро переключаться между различными разделами приложения.
- Фильтры: в левой части экрана находится панель фильтров, где пользователь может отфильтровать заявки по статусу, дате создания заявки и номеру заявки. Есть кнопки «Применить» и «Очистить фильтры» для управления фильтрацией. Пользователь может установить нужные

параметры фильтрации и применить их, чтобы отобразить только те заявки, которые соответствуют заданным критериям.

- **Список заявок:** в центральной части экрана отображается список заявок с кратким описанием проблемы и именем клиента. Пользователь может просматривать этот список и выбирать конкретные заявки для дальнейших действий. Краткое описание помогает быстро определить суть проблемы.

- **Детали заявки:** в правой части экрана показаны подробности выбранной заявки. Пользователь может просматривать и редактировать информацию о выбранной заявке. После внесения изменений пользователь может сохранить их, нажав кнопку «Сохранить».

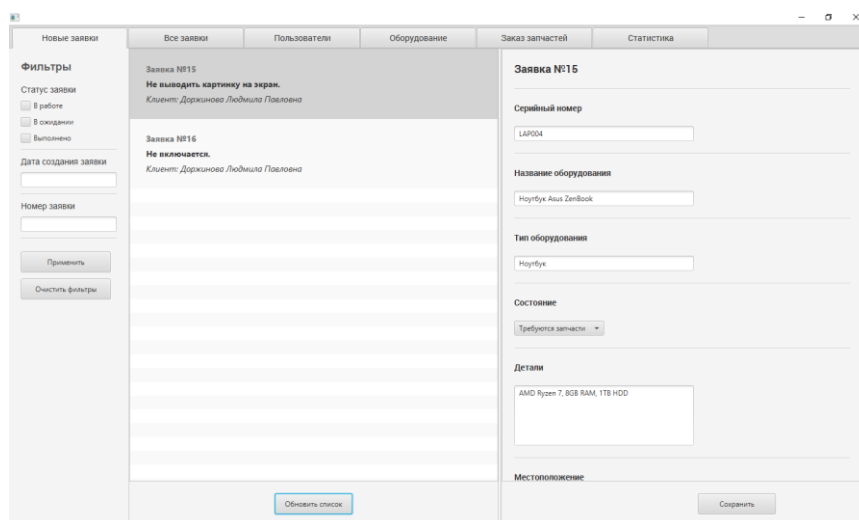


Рисунок 27 – Вкладка с новыми заявками

На рисунке 28 представлено окно приложения, предназначенного для управления заявками на ремонт техники. Интерфейс разделен на несколько основных областей.

Пользователь может взаимодействовать с приложением следующими способами:

- **Меню вкладок:** в верхней части окна расположены вкладки «Новые заявки», «Все заявки», «Пользователи», «Оборудование», «Заказ запчастей», «Статистика», что позволяет быстро переключаться между различными разделами приложения. Пользователь может выбрать нужную вкладку для

просмотра и управления заявками, пользователями, оборудованием и другими аспектами приложения.

- **Фильтры:** в левой части экрана находится панель фильтров, где пользователь может отфильтровать заявки по статусу, дате создания заявки и номеру заявки. Есть кнопки «Применить» и «Очистить фильтры» для управления фильтрацией. Пользователь может установить нужные параметры фильтрации и применить их, чтобы отобразить только те заявки, которые соответствуют заданным критериям.

- **Список заявок:** в центральной части экрана отображается список заявок с кратким описанием проблемы и именем клиента. Пользователь может просматривать этот список и выбирать конкретные заявки для дальнейших действий. Краткое описание помогает быстро определить суть проблемы.

- **Детали заявки:** в правой части экрана показаны подробности выбранной заявки.

- Пользователь может просматривать и редактировать информацию о выбранной заявке. После внесения изменений пользователь может сохранить их, нажав кнопку «Сохранить». Также имеется кнопка «Создать отчет» для формирования отчета по заявке.

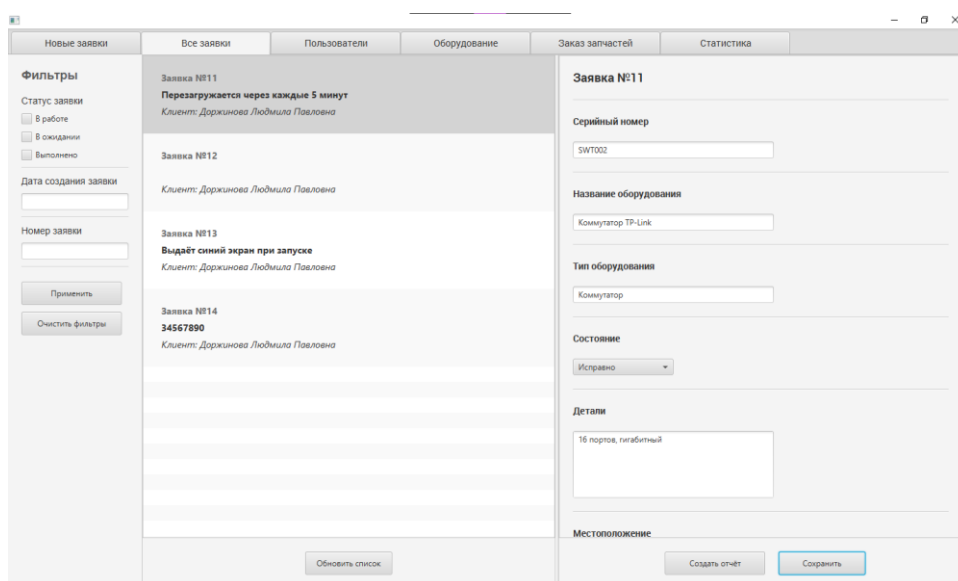


Рисунок 28 – Вкладка со всеми заявками

Пользователь может взаимодействовать с приложением следующими способами:

- Меню вкладок: в верхней части окна расположены вкладки «Новые заявки», «Все заявки», «Пользователи», «Оборудование», «Заказ запчастей», «Статистика», что позволяет быстро переключаться между различными разделами приложения. Пользователь может выбрать вкладку «Пользователи» для просмотра и управления учетными записями.
- Список пользователей: в центральной части экрана отображается таблица со списком пользователей.
- Управление пользователями: в нижней части экрана расположены кнопки «Добавить пользователя» и «Удалить». Пользователь может добавить нового пользователя, заполнив соответствующую форму, или удалить выбранного пользователя из списка.

Новые заявки		Все заявки		Пользователи		Оборудование		Заказ запчастей		Статистика	
ID	ФИО	Номер телефона	E-mail	Логин	Пароль	Роль					
1	Куликов Игорь Геннадьевич	+7 (959) 245-21-47	AlbinaShanskaya581@gmail.com	admin	root	admin					
2	Доржинова Людмила Павловна	+7 (918) 413-19-98	LyudmilaDorzhinova267@gmail.com	user	123	user					

Добавить пользователя

Удалить

Рисунок 29 – Вкладка с пользователями

На рисунке 30 представлено окно приложения, предназначенного для управления оборудованием, связанным с заявками на ремонт техники. Интерфейс разделен на несколько основных областей.

Пользователь может взаимодействовать с приложением следующими способами:

- Меню вкладок: в верхней части окна расположены вкладки «Новые заявки», «Все заявки», «Пользователи», «Оборудование», «Заказ запчастей», «Статистика», что позволяет быстро переключаться между различными разделами приложения. Пользователь может выбрать вкладку «Оборудование» для просмотра и управления списком оборудования
- Список оборудования: в центральной части экрана отображается таблица со списком оборудования.
- Управление оборудованием: в нижней части экрана расположены кнопки «Добавить оборудование» и «Удалить». Пользователь может добавить новое оборудование, заполнив соответствующую форму, или удалить выбранное оборудование из списка.

Серийный номер	Наименование оборудования	Тип оборудования	Состояние	Подробная информация	Местонахождение
LAP001	Ноутбук Apple MacBook Pro	Ноутбук	Новое	Intel i9, 16GB RAM, 1TB SSD	Каб. 301
LAP002	Ноутбук HP Pavilion	Ноутбук	Используется	Intel i5, 8GB RAM, 512GB SSD	Каб. 302
LAP003	Ноутбук Dell Inspiron	Ноутбук	Новое	Intel i7, 16GB RAM, 256GB SSD	Каб. 303
LAP004	Ноутбук Asus ZenBook	Ноутбук	Используется	AMD Ryzen 7, 8GB RAM, 1TB HDD	Каб. 304
LAP005	Ноутбук Lenovo ThinkPad	Ноутбук	Новое	Intel i5, 8GB RAM, 512GB SSD	Каб. 305
MON001	ViewSonic Белый	Монитор	Исправно	1400x900, 75Гц	Каб. 101
MON002	Монитор Samsung	Монитор	Не исправно	LED, 27 дюймов	Каб. 302
MON003	Монитор Dell	Монитор	Новое	IPS, 24 дюйма	Каб. 303
MON004	Монитор Asus	Монитор	Используется	TN, 23 дюйма	Каб. 304
MON005	Монитор Acer	Монитор	Новое	LED, 21 дюйм	Каб. 305
MON006	Монитор LG	Монитор	Новое	LED, 24 дюйма	Каб. 301
PRN001	Принтер HP LaserJet	Принтер	Новое	Лазерный принтер, A4	Каб. 301
PRN002	Принтер Canon P30MA	Принтер	Используется	Струйный принтер, A3	Каб. 302
PRN003	Принтер Epson EcoTank	Принтер	Новое	Струйный принтер, A4	Каб. 303
PRN004	Принтер Brother HL	Принтер	Используется	Лазерный принтер, A4	Каб. 304
PRN005	Принтер Samsung Xpress	Принтер	Новое	Лазерный принтер, A4	Каб. 305
ROU001	Маршрутизатор TP-Link	Маршрутизатор	Новое	AC1200	Каб. 301
ROU002	Маршрутизатор Asus	Маршрутизатор	Используется	AC1750	Каб. 302
ROU003	Маршрутизатор D-Link	Маршрутизатор	Новое	AC750	Каб. 303
ROU004	Маршрутизатор Netgear	Маршрутизатор	Используется	AC1900	Каб. 304
ROU005	Маршрутизатор Mikrotik	Маршрутизатор	Новое	AC2400	Каб. 305
SWT001	Коммутатор Cisco	Коммутатор	Новое	24 порта, гигабитный	Каб. 301
SWT002	Коммутатор TP-Link	Коммутатор	Исправно	16 портов, гигабитный	Каб. 302
SWT003	Коммутатор D-Link	Коммутатор	Новое	8 портов, гигабитный	Каб. 303
SWT004	Коммутатор Netgear	Коммутатор	Используется	24 порта, гигабитный	Каб. 304
SWT005	Коммутатор Zyxel	Коммутатор	Не исправно	16 портов, гигабитный	Каб. 305
SY001	Блок	Системный блок	Исправно	ука	На складе
SY002	Системный блок HP	Системный блок	Не исправно	Intel i5, 8GB RAM, 1TB HDD	Каб. 302
SY003	Системный блок Dell	Системный блок	Новое	AMD Ryzen 5, 16GB RAM, 256GB SSD	Каб. 303
SY004	Системный блок	Системный блок	Исправно	Intel i7, 16GB RAM, 512GB SSD	Каб. 304

Рисунок 30 – Вкладка с оборудованием

На рисунке 31 представлено окно приложения, предназначенного для управления заказами запчастей, связанных с заявками на ремонт техники. Интерфейс разделен на несколько основных областей.

Пользователь может взаимодействовать с приложением следующими способами:

- Меню вкладок: в верхней части окна расположены вкладки «Новые заявки», «Все заявки», «Пользователи», «Оборудование», «Заказ запчастей», «Статистика», что позволяет быстро переключаться между различными разделами приложения. Пользователь может выбрать вкладку «Заказ запчастей» для просмотра и управления списком заказов на запчасти.
- Список заказов: в центральной части экрана отображается таблица со списком заказов на запчасти.
- Управление заказами: в нижней части экрана расположены кнопки «Добавить заказ» и «Удалить». Пользователь может добавить новый заказ на запчасти, заполнив соответствующую форму, или удалить выбранный заказ из списка.

[illegible]

Рисунок 31 – Вкладка с заказами запчастей

На рисунке 32 представлено окно приложения, предназначенного для отображения статистики по заявкам на ремонт техники. Интерфейс разделен на несколько основных областей.

Пользователь может взаимодействовать с приложением следующими способами:

- Меню вкладок: в верхней части окна расположены вкладки «Новые заявки», «Все заявки», «Пользователи», «Оборудование», «Заказ запчастей», «Статистика», что позволяет быстро переключаться между различными

разделами приложения. Пользователь может выбрать вкладку «Статистика» для просмотра аналитической информации по заявкам.

- Статистические данные: в левой части экрана представлены ключевые показатели.

- Графическое представление: в правой части экрана отображена круговая диаграмма, показывающая распределение типов неисправностей.

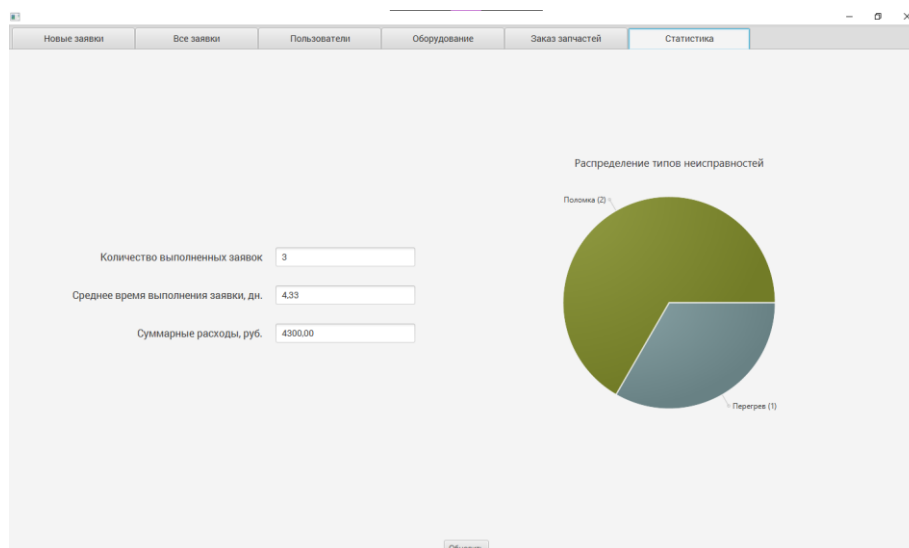


Рисунок 32 – Вкладка статистики

На рисунке 33 представлено окно авторизации приложения. Пользователь должен ввести имя пользователя и пароль, чтобы получить доступ к функциональности приложения.

На данном экране есть:

- Поле ввода имени пользователя: в середине окна расположено текстовое поле для ввода логина пользователя.

- Поле ввода пароля: ниже находится текстовое поле для ввода пароля. Введенный пароль отображается в виде точек, что обеспечивает конфиденциальность.

- Кнопка входа: под полями ввода располагается кнопка «Вход». Нажатие на эту кнопку отправляет введенные данные для проверки и предоставляет доступ к основному функционалу приложения.

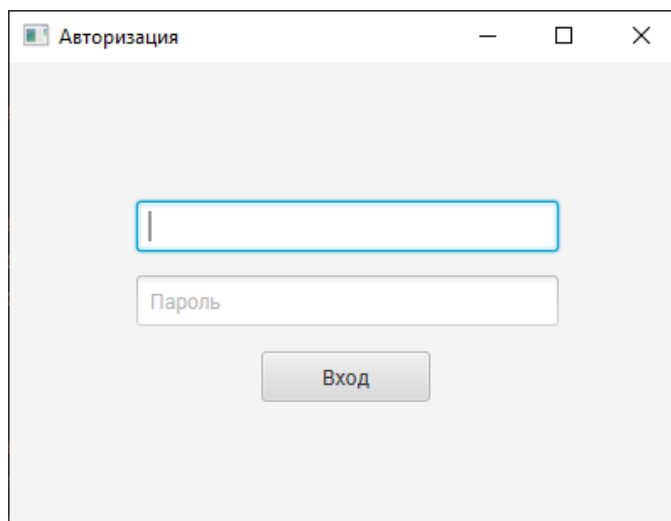


Рисунок 33 – Окно авторизации

На рисунке 34 представлено окно приложения, предназначенного для создания заявки на ремонт техники. Интерфейс разделен на несколько основных областей.

Пользователь может взаимодействовать с приложением следующими способами:

- Меню вкладок: в верхней части окна расположены вкладки «Создать заявку», «Мои заявки», «Профиль» для пользователя, что позволяет быстро переключаться между различными разделами приложения. Пользователь может выбрать вкладку «Создать заявку» для внесения новой заявки на ремонт.
- Форма создания заявки: центральная часть экрана содержит поля для ввода информации о новой заявке.
- Управление заявкой: в нижней части экрана расположены кнопки «Создать заявку» и «Очистить».

Создать заявку Мои заявки Профиль

Серийный номер LAP004

Тип оборудования Ноутбук

Наименования оборудования Ноутбук Asus ZenBook

Описание проблемы Не выводит картинку на экран

Создать заявку Очистить

Рисунок 34 – Вкладка создания заявок у пользователя

На рисунке 35 представлено окно приложения, предназначенного для управления заявками на ремонт техники. Интерфейс разделен на несколько основных областей.

Пользователь может взаимодействовать с приложением следующими способами:

- Меню вкладок: в верхней части окна расположены вкладки «Создать заявку», «Мои заявки», «Профиль», что позволяет быстро переключаться между различными разделами приложения. Пользователь может выбрать вкладку «Мои заявки» для просмотра и управления своими заявками.
- Фильтры: в левой части экрана находится панель фильтров, где пользователь может отфильтровать заявки по статусу, дате создания заявки и номеру заявки. Есть кнопки «Применить» и «Очистить фильтры» для управления фильтрацией. Пользователь может установить нужные параметры фильтрации и применить их, чтобы отобразить только те заявки, которые соответствуют заданным критериям.
- Список заявок: в центральной части экрана отображается список заявок с кратким описанием проблемы и именем клиента. Краткое описание помогает быстро определить суть проблемы.

- Детали заявки: в правой части экрана показаны подробности выбранной заявки.
- Пользователь может просматривать и редактировать информацию о выбранной заявке. После внесения изменений пользователь может сохранить их, нажав кнопку «Сохранить». Также имеется кнопка «Обновить список» для обновления отображаемого списка заявок.

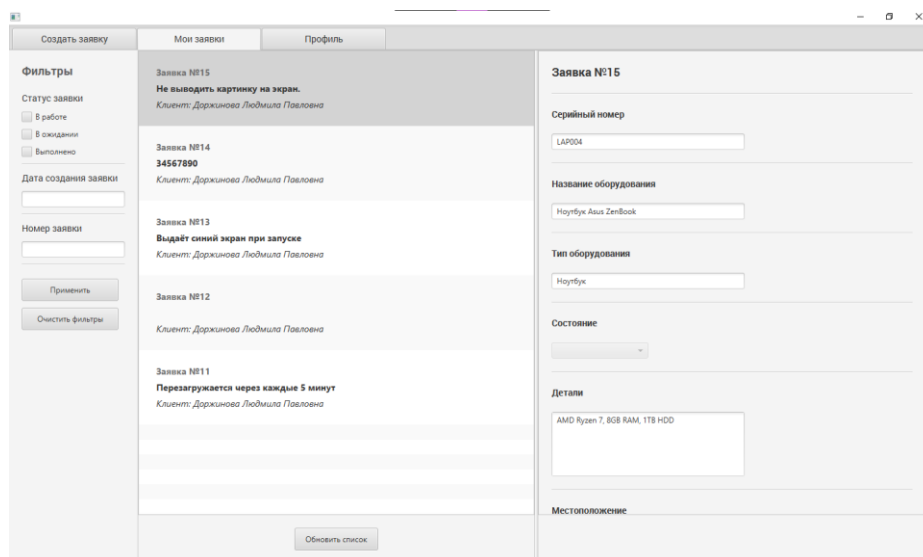


Рисунок 35 – Вкладка отправленных заявок у пользователя

На рисунке 36 представлено окно приложения, предназначенного для управления профилем пользователя. Интерфейс разделен на несколько основных областей.

Пользователь может взаимодействовать с приложением следующими способами:

- Меню вкладок: в верхней части окна расположены вкладки «Создать заявку», «Мои заявки», «Профиль», что позволяет быстро переключаться между различными разделами приложения. Пользователь может выбрать вкладку «Профиль» для просмотра и редактирования своих персональных данных.
- Форма профиля пользователя: центральная часть экрана содержит поля для ввода и редактирования информации профиля.

– Управление профилем: в нижней части экрана расположена кнопка «Сохранить». Пользователь может нажать эту кнопку для сохранения изменений, внесенных в профиль.

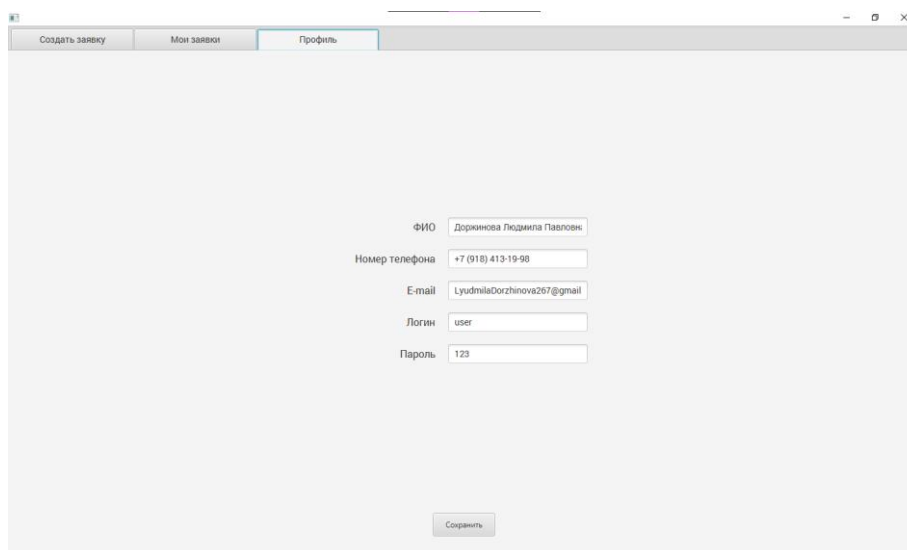


Рисунок 36 – Вкладка профиля у пользователя

3.5.2 Разработка основных функций

1. Авторизация

Реализация:

Процесс разработки функции авторизации начался с создания интерфейса для ввода имени пользователя и пароля. Этот интерфейс был спроектирован таким образом, чтобы быть простым и интуитивно понятным для пользователей. Поля ввода имели встроенные проверки на корректность данных, таких как минимальная длина пароля и проверка формата электронной почты. Введенные данные проверялись по сохраненным учетным данным в базе данных, предоставляя или запрещая доступ к функционалу приложения.

2. Создание заявки

Форма заявки:

Разработка интерфейса для создания новой заявки включала создание формы, содержащей поля для серийного номера, типа оборудования, описания проблемы и других необходимых данных. Пользователь мог легко

заполнить эту форму благодаря продуманному дизайну, который включал всплывающие подсказки и проверки на корректность ввода данных.

Сохранение:

После заполнения формы данные сохранялись в базе данных. Каждой заявке автоматически присваивался уникальный идентификатор, и пользователь получал уведомление о успешной подаче заявки.

3. Редактирование заявки

Интерфейс:

Для редактирования заявок был разработан интерфейс, который позволял пользователям просматривать и изменять уже существующие заявки. Этот экран отображал текущую информацию о заявке и предоставлял возможность редактировать поля, такие как описание проблемы и статус заявки.

Обновление:

Функциональность редактирования включала обновление данных в базе данных. Пользовательские изменения проходили проверки для обеспечения целостности данных и их корректного обновления.

4. Добавление оборудования, пользователя, заказа запчастей

Формы ввода:

Для добавления нового оборудования, пользователей и заказов на запчасти были созданы отдельные интерфейсы. Эти формы включали все необходимые поля для ввода данных и предоставляли удобные инструменты для заполнения информации.

Валидация:

Внедрение проверок на корректность введенных данных включало валидацию на стороне клиента. Это обеспечивало правильность вводимой информации и предотвращало ошибки при сохранении данных в базе.

Сохранение:

После заполнения формы данные сохранялись в соответствующих таблицах базы данных. Пользователю отображалось уведомление о успешном добавлении данных.

5. Создание отчетов

Генерация отчетов:

Реализация функциональности создания отчетов включала разработку системы, которая позволяла пользователям генерировать отчеты по завершенным заявкам. Эти отчеты включали подробности о выполненных работах, затратах и времени, затраченном на выполнение заявки.

6. Формирование статистики

Анализ данных:

Для анализа данных о заявках были разработаны механизмы, которые собирали и анализировали информацию по различным метрикам, таким как количество выполненных заявок, среднее время выполнения и суммарные расходы. Эти данные использовались для улучшения процессов и повышения эффективности работы.

Отображение:

Создание интерфейса для визуализации статистики включало разработку графиков и диаграмм, которые отображали ключевые метрики. Это позволяло пользователям легко интерпретировать данные и принимать обоснованные решения на основе представленной информации.

4 Тестирование приложения

Тестирование является критически важной частью разработки системы управления заявками на ремонт оборудования. Оно включает в себя несколько этапов и методов, направленных на обеспечение качества, надежности и безопасности приложения. В этом разделе описаны подходы и методики, использованные для тестирования основных функций и пользовательского интерфейса.

4.1 Модульное тестирование

Модульное тестирование является первым этапом тестирования, на котором проверяются отдельные модули и компоненты системы. Основная цель модульного тестирования — убедиться в корректной работе каждого модуля в изоляции от других.

- Инструменты: для модульного тестирования использовались инструменты JUnit и Mockito, которые позволяют создавать и выполнять тесты для отдельных методов и классов.

- Тестируемые компоненты: в проекте были протестированы ключевые компоненты, такие как регистрация заявок, обработка заявок и управление оборудованием. Например, проверялась правильность регистрации заявок и корректность работы методов обновления статуса и обработки заявок.

- Методика: создавались тесты для проверки правильности выполнения основных функций, таких как добавление новых заявок, обновление статусов и управление информацией об оборудовании. Тесты проверяли, что заявки корректно сохраняются, обновляются и отображаются.

- Покрытие тестами: обеспечено покрытие кода тестами для ключевых функций, чтобы минимизировать вероятность наличия ошибок.

4.2 Интеграционное тестирование

Интеграционное тестирование направлено на проверку взаимодействия между различными компонентами системы. Цель этого этапа — убедиться в корректной работе системы в целом, когда все модули работают совместно.

Инструменты: использовались инструменты JUnit и Mockito, которые позволяют симулировать взаимодействие компонентов и проверять результаты.

Тестируемые сценарии: тестировались сценарии, включающие полный цикл работы с системой:

- регистрация и обработка новых заявок;
- обновление статусов заявок и управление запасными частями;
- генерация и обработка отчетов.

Проверка взаимодействия: уделялось внимание взаимодействию между различными модулями, такими как модуль управления заявками, модуль отчетности и модуль управления запасными частями.

4.3 Пользовательское тестирование

Пользовательское тестирование направлено на оценку удобства использования (UX) и выявление проблем, с которыми могут столкнуться пользователи при взаимодействии с системой.

Методика: приложение тестировалось в ограниченных условиях, без проведения широкого пользовательского тестирования.

Тестируемые аспекты: оценивались удобство навигации, понятность интерфейса и общая удовлетворенность пользователей.

Сбор обратной связи: обратная связь собиралась непосредственно во время использования приложения.

Итерации: на основе полученной обратной связи были внесены необходимые улучшения в дизайн и функциональность системы.

4.4 Ручное тестирование

Ручное тестирование выполнялось для проверки тех аспектов системы, которые трудно автоматизировать, а также для выполнения регрессионного тестирования после внесения изменений в код.

Методика: тестировались основные функции и сценарии использования, такие как регистрация новых заявок, обновление статусов, обработка запросов на заказ запасных частей и генерация отчетов.

Проверка функциональности: включала проверку всех основных функций, чтобы убедиться в корректной работе системы после внесения изменений.

Регрессионное тестирование: проводилось после каждого крупного изменения в коде, чтобы убедиться, что новые изменения не нарушили существующую функциональность.

Тестирование является ключевым процессом, который проводится на всех этапах разработки и продолжается после выпуска продукта. Этот процесс обеспечивает высокое качество, надежность и безопасность системы управления заявками на ремонт, удовлетворяя ожидания пользователей и соответствуя высоким стандартам.

5 Методическое обеспечение

5.1 Руководство системного администратора

Это руководство предназначено для системного администратора, который будет управлять системой «Служба поддержки». В нем описаны основные задачи и операции, которые необходимо выполнять для эффективного использования системы.

Программные и аппаратные требования к системе

Программные требования:

- Операционная система: Windows 10, macOS 10.13, или современный Linux дистрибутив.

- Java Runtime Environment.

- PostgreSQL 15.

Аппаратные требования

Клиентское оборудование:

- Процессор: двухъядерный процессор с тактовой частотой 2 ГГц.

- Оперативная память: минимум 4 ГБ.

- Место на диске: минимум 500 МБ свободного места.

Серверное оборудование:

- Процессор: двухъядерный процессор с тактовой частотой 2.5 ГГц.

- Оперативная память: минимум 4 ГБ.

- Место на диске: минимум 10 ГБ свободного места.

Чтобы СУБД была доступна с любого устройства в локальной сети, на компьютере, выступающем в роли сервера для СУБД, нужно выполнить следующие действия.

Открыть проводник. Перейти в папку, куда был установлен PostgreSQL. Открыть файл «pg_hba.conf» с помощью блокнота.

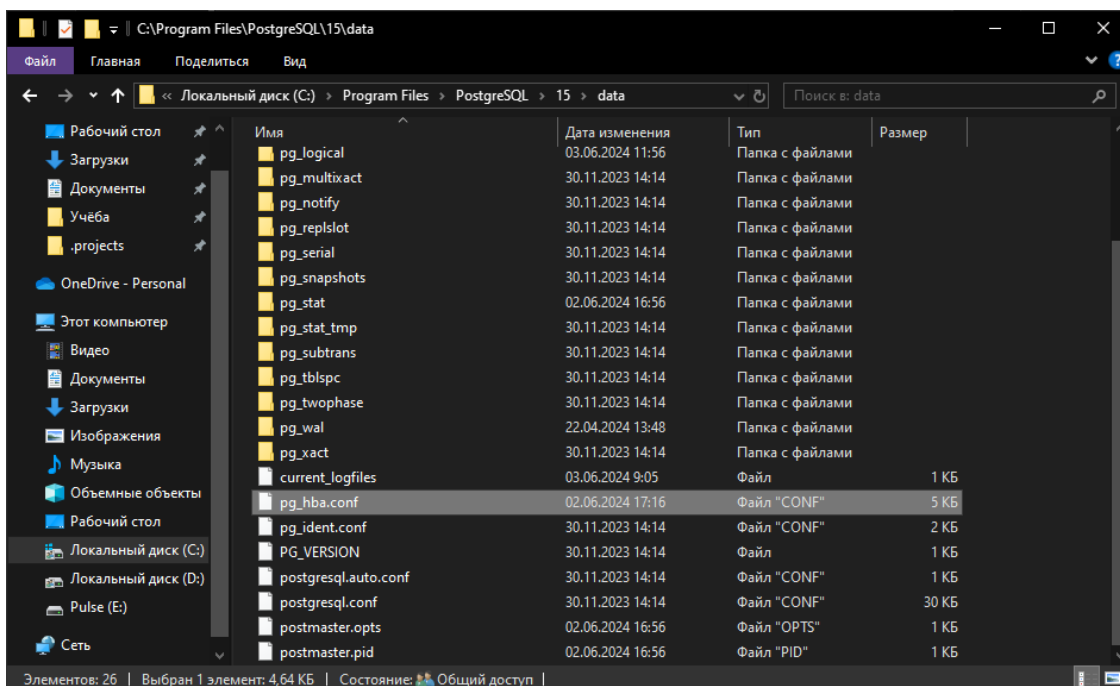


Рисунок 37 – Расположение файла в папке

Добавить запись для разрешения удаленных подключений «host all all 0.0.0.0/0 trust» и сохранить файл.

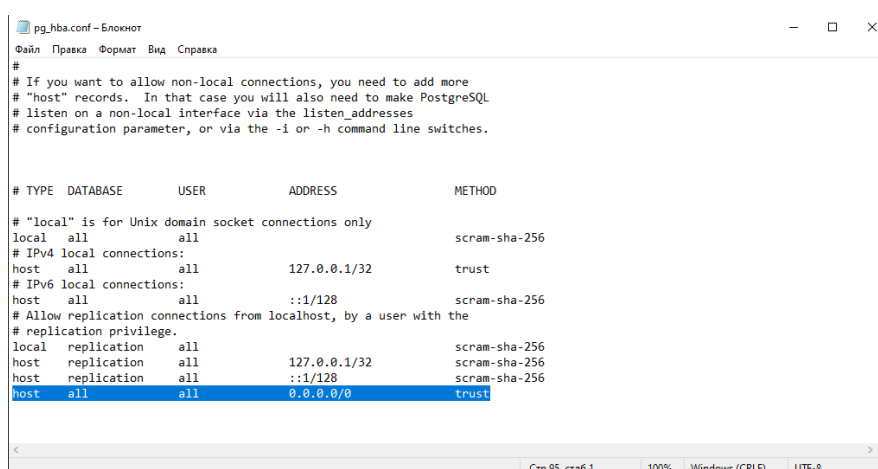


Рисунок 38 – Редактирование файла «pg_hba.conf»

Нажать клавишу «Win + R», ввести «wf.msc» и нажать «Enter».

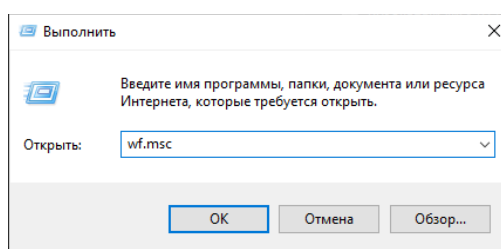


Рисунок 39 – Открытие монитора брандмауэра

В левой части окна нажать на «Правила для входящих подключений», в правой части экрана нажать на «Создать правило».

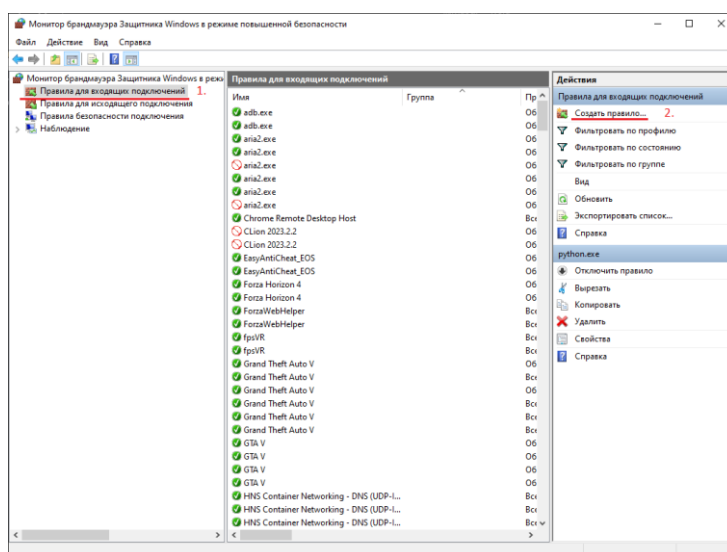


Рисунок 40 – Создание правила для новых входящих подключений

В открывшемся мастере создания правила выбрать «Для порта» и нажать «Далее».

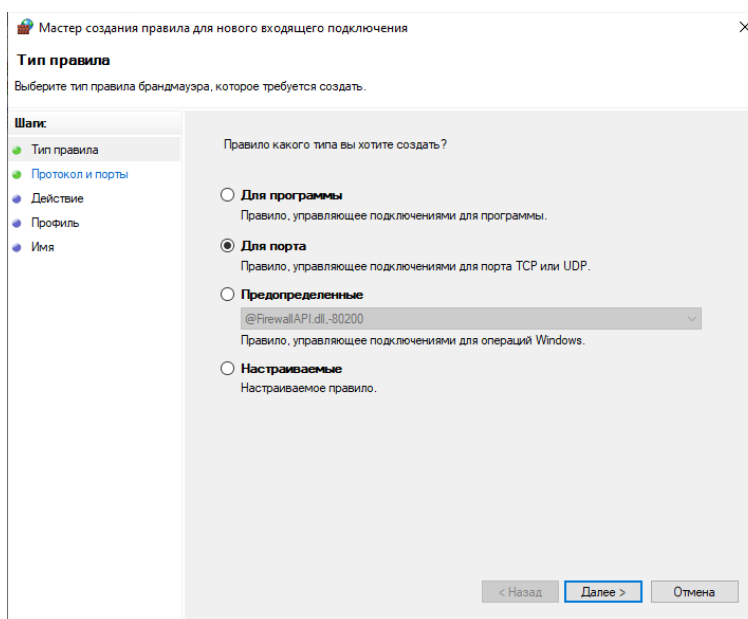


Рисунок 41 – Настройка правила для новых входящих подключений

В протоколе указать «Протокол TCP». В «определенные локальные порты» ввести 8888. Нажать «Далее».

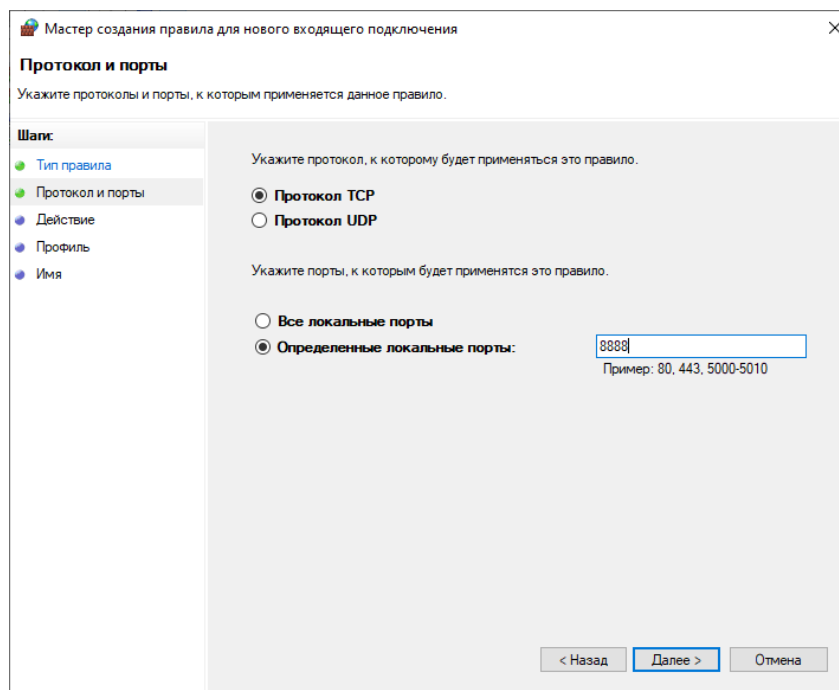


Рисунок 42 – Настройка правила для новых входящих подключений
Выбрать «Разрешить подключение» и нажать «Далее».

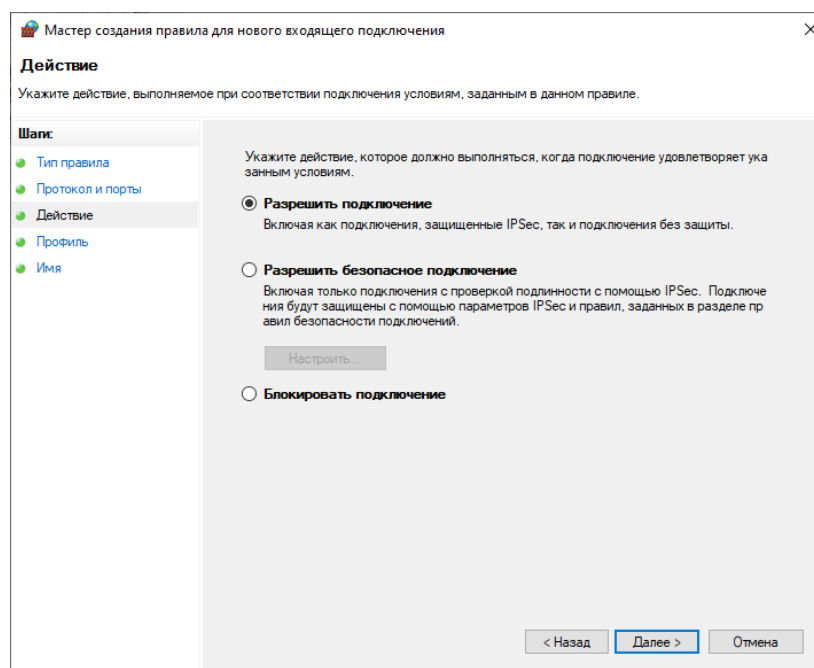


Рисунок 43 – Настройка правила для новых входящих подключений
Выбрать все три профиля («Доменный», «Частный», «Публичный») и нажать «Далее».

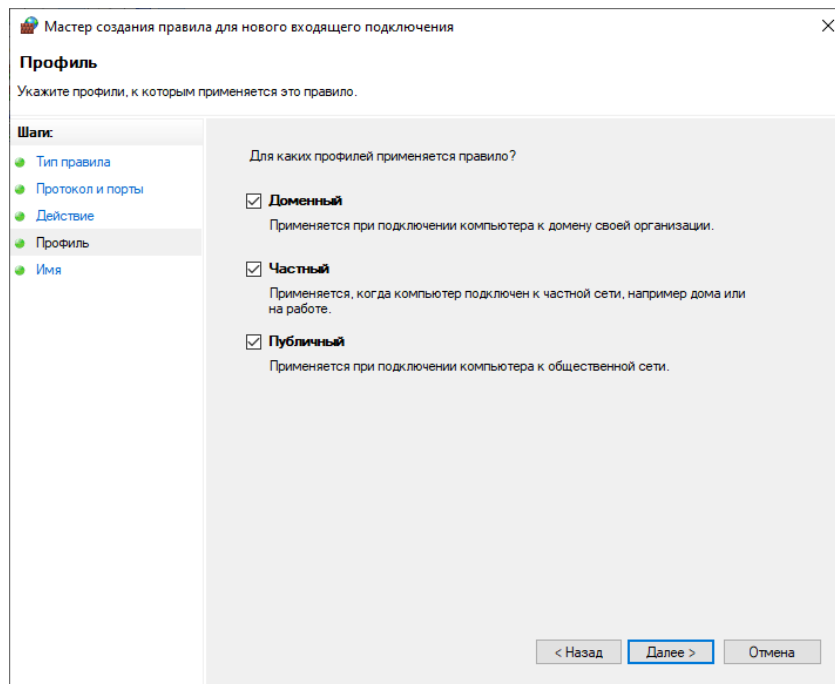


Рисунок 44 – Настройка правила для новых входящих подключений
Задать имя «PostgreSQL», нажать «Готово».

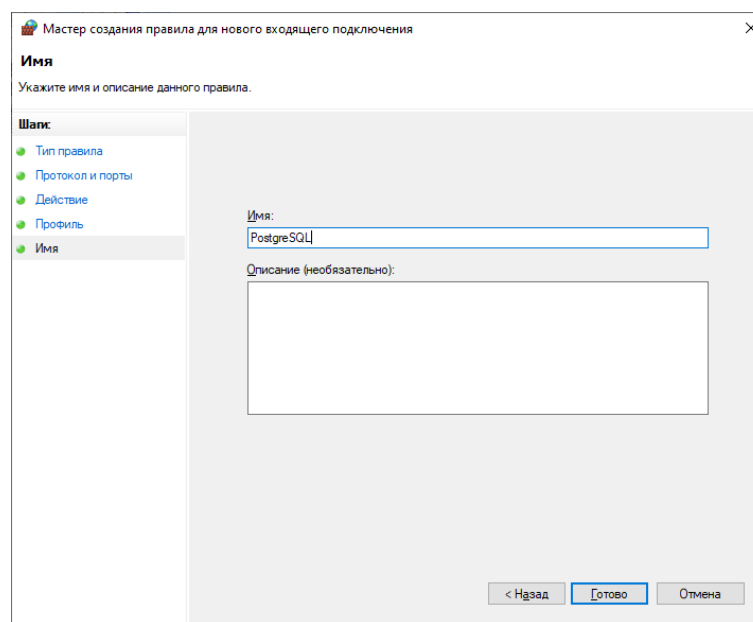


Рисунок 45 – Настройка правила для новых входящих подключений
Нажать «Пуск» правой кнопкой мыши, выбрать «Windows PowerShell (администратор)».

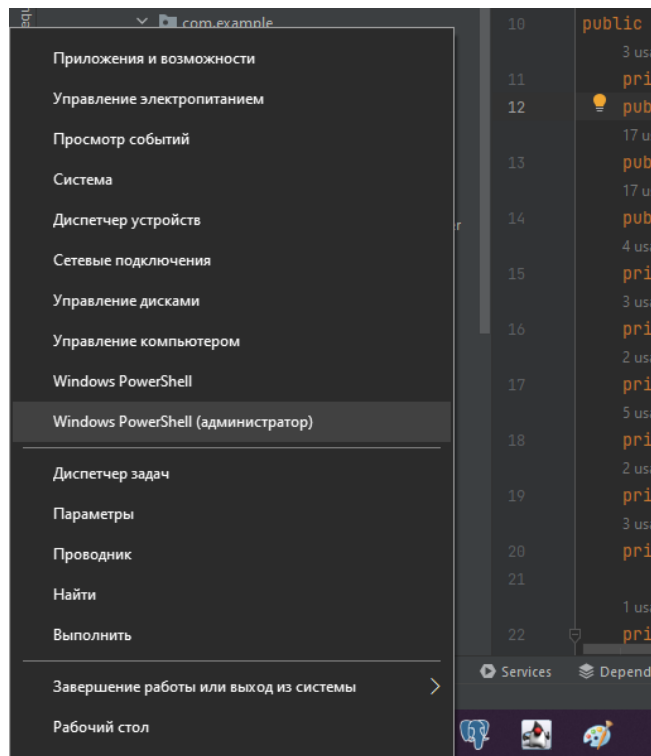


Рисунок 46 – Открытие окна PowerShell

Введение команды «ipconfig» в командную строку и нажать «Enter».

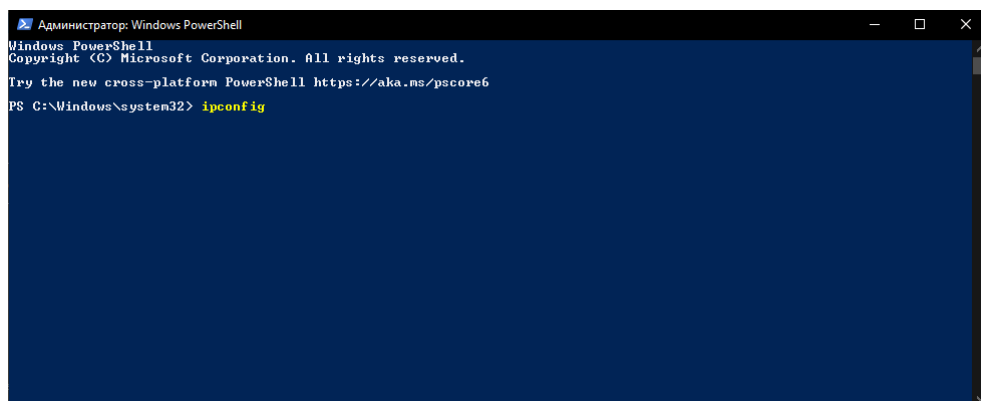


Рисунок 47 –Окно с введенной командой в строку

После выполнения команды, в терминал будут выведены IP адреса. Среди них нужно найти IPv4 адрес используемого сетевого адаптера:

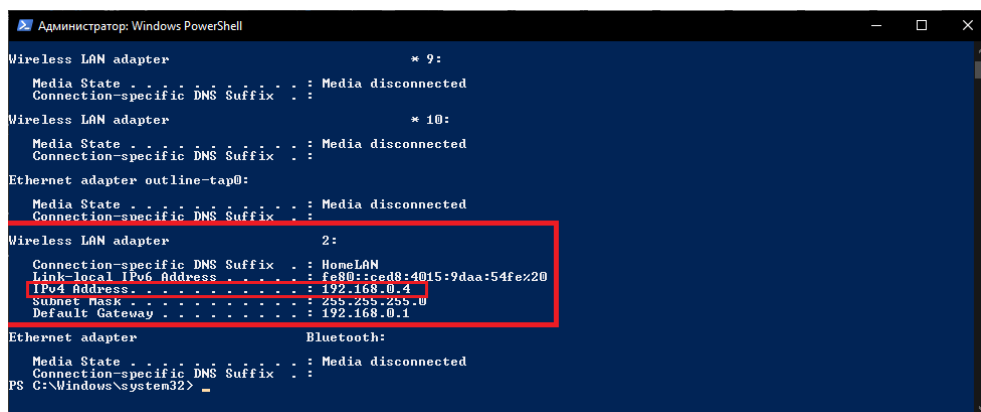


Рисунок 48 – Окно с IP адресами компьютера

После получения IP адреса на сервере, на клиентском устройстве нужно запустить программу и ввести IP адрес сервера:

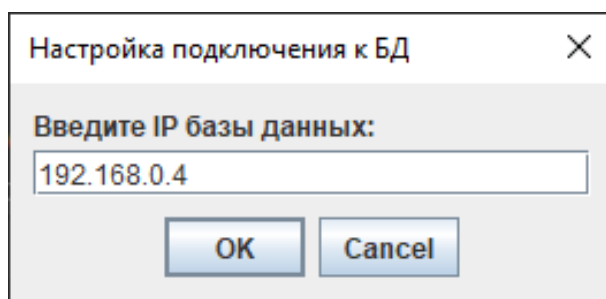


Рисунок 49 – Окно с подключением к базе данных

При успешном подключении выводится диалоговое окно об успешном подключении.

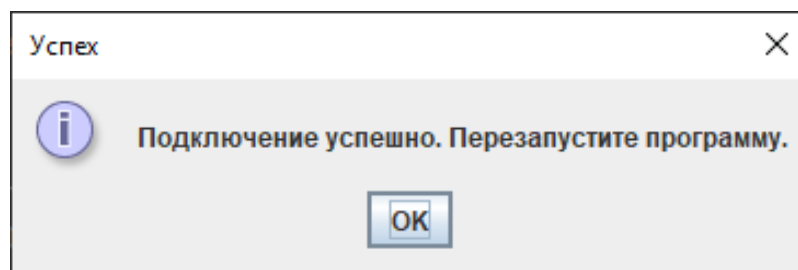


Рисунок 50 – Диалоговое окно

Вход в систему

1. Откройте клиентское приложение «Служба поддержки».
2. Введите свой логин и пароль в соответствующие поля.
3. Нажмите кнопку «Войти».

Регистрация нового пользователя

1. Перейдите во вкладку «Пользователи» в приложении.

2. Нажмите кнопку «Добавить сотрудника».
3. Введите информацию о пользователе: имя, телефон, электронную почту, логин, пароль и роль (например, «Пользователь» или «Системный администратор»).
4. Нажмите кнопку «Добавить».

Редактирование информации о пользователе

1. Перейдите во вкладку «Пользователи».
2. Выберите пользователя из списка двойным кликом.
3. Измените необходимую информацию.
4. Нажмите кнопку «Добавить».

Удаление пользователя

1. Перейдите во вкладку «Пользователи».
2. Выберите пользователя из списка.
3. Нажмите кнопку «Удалить».
4. Подтвердите удаление.

Просмотр заявок

1. Перейдите во вкладку «Все заявки».
2. В списке заявок выберите интересующую вас заявку, чтобы увидеть её детали.

Обработка заявок

1. Перейдите во вкладку «Все заявки».
2. Выберите заявку, которую необходимо обработать.
3. Просмотрите детали заявки.
4. При необходимости запросите дополнительные сведения у пользователя.
5. Измените статус заявки в зависимости от этапа обработки (например, «В работе», «Выполнено» и т.д.).

Исполнение заявок

1. После получения всей необходимой информации выполните ремонт оборудования.

2. Если требуются запчасти, перейдите в раздел «Заказ запчастей» и создайте новый заказ.
3. Направьте заказ на одобрение директору.
4. После выполнения ремонта закройте заявку, изменив её статус на «Выполнено».

Создание отчётов

1. Выберите заявку, по которой нужно создать отчёт.
2. Нажмите кнопку «Создать отчет».
3. Введите информацию для отчета.
4. Нажмите кнопку «Добавить».

Просмотр отчётов

1. Выберите интересующую заявку.
2. Нажмите на кнопку «Посмотреть отчет», после создания отчета.

Мониторинг и статистика

1. Перейдите во вкладку «Статистика».
2. Просматривайте диаграммы и графики для анализа времени и ресурсов, затраченных на выполнение заявок.

Добавление нового оборудования

1. Перейдите во вкладку «Оборудование».
2. Нажмите кнопку «Добавить оборудование».
3. Введите информацию о новом оборудовании.
4. Нажмите кнопку «Добавить».

Редактирование информации об оборудовании

1. Перейдите во вкладку «Оборудование».
2. Выберите оборудование из списка.
3. Выберите оборудование из списка двойным кликом.
4. Измените необходимую информацию.
5. Нажмите кнопку «Добавить».

Удаление оборудования

1. Перейдите во вкладку «Оборудование».

2. Выберите оборудование из списка.
3. Нажмите кнопку «Удалить».
4. Подтвердите удаление.

Добавление нового заказа запчастей

1. Перейдите во вкладку «Заказ запчастей».
2. Нажмите кнопку «Добавить заказ».
3. Введите информацию о новом заказе.
4. Нажмите кнопку «Добавить».

Редактирование информации об заказе запчастей

1. Перейдите во вкладку «Заказ запчастей».
2. Выберите заказ из списка.
3. Выберите заказ из списка двойным кликом.
4. Измените необходимую информацию.
5. Нажмите кнопку «Добавить».

Удаление заказа запчастей

1. Перейдите во вкладку «Заказ запчастей».
2. Выберите заказ из списка.
3. Нажмите кнопку «Удалить».
4. Подтвердите удаление.

5.2 Руководство пользователя

Вход в систему

1. Откройте клиентское приложение «Служба поддержки».
2. Введите свой логин и пароль в соответствующие поля.
3. Нажмите кнопку «Войти».

Создание заявки

1. Перейдите во вкладку «Создать заявку», расположенную в верхней части окна.
2. Заполните поля формы создания заявки, введя необходимую информацию о новой заявке.

3. Нажмите кнопку «Создать заявку» для сохранения новой заявки.
4. Если необходимо очистить все поля формы, нажмите кнопку «Очистить».

Просмотр и управление заявками

1. Перейдите во вкладку «Мои заявки», расположенную в верхней части окна.
2. Используйте панель фильтров в левой части экрана, чтобы отфильтровать заявки по статусу, дате создания заявки и номеру заявки. Установите нужные параметры фильтрации и нажмите кнопку «Применить». Для сброса фильтров нажмите кнопку «Очистить фильтры».
3. В центральной части экрана просмотрите список заявок с кратким описанием проблемы и именем клиента. Выберите интересующую заявку для просмотра ее деталей.
4. В правой части экрана отобразятся подробности выбранной заявки. Внесите необходимые изменения и нажмите кнопку «Сохранить» для сохранения изменений.
5. Для обновления отображаемого списка заявок нажмите кнопку «Обновить список».

Редактирование профиля

1. Перейдите во вкладку «Профиль», расположенную в верхней части окна.
2. В центральной части экрана заполните поля формы профиля, введя или изменив информацию о себе.
3. Нажмите кнопку «Сохранить» для сохранения изменений в профиле.

6 Экономическая часть

Расчет затрат на создание программного продукта

Разработка приложения по сбору заявок на ремонт различной техники «Служба поддержки» разделена на следующие шаги:

- Предпроектное обследование. В данном пункте производится исследование и анализ предметной области, проблематики, а также разработка ТЗ.
- Эскизное проектирование. В данном пункте производится теоретическое проектирование структуры программы, её интерфейс и разрабатывается проектная документация.
- Технорабочий проект. В данном пункте производится реализация программы в соответствии с разработанной структурой в предыдущем шаге, также происходит разработка технической документации.
- Ввод в эксплуатацию. В данном пункте производится тестирование готового программного продукта и его выпуск.

Затраты на разработку проекта (производственные затраты) представляют собой единовременные расходы на всех этапах инновационного процесса: исследование, разработка, тестирование, внедрение. Определение затрат на разработку проекта производится путем составления калькуляции плановой себестоимости. В плановую себестоимость включаются все затраты, связанные с ее выполнением, независимо от источника их финансирования [11].

Смета затрат состоит из прямых и накладных расходов. Расходы на разработку проекта включают в себя следующие статьи:

- прямые материальные затраты;
- основная заработная плата;
- социальные начисления;
- амортизация ЭВМ;
- затраты на программное обеспечение;

- накладные расходы (10% от прямых расходов).

Пункт 1. Прямые материальные затраты

В данном пункте учитываются затраты на материалы, расходуемые на проектирование системы. В случае разработки приложения по сбору заявок на ремонт различной техники «Служба поддержки», все необходимые материалы и оборудование уже были в наличии и приобретение дополнительных ресурсов не потребовалось.

Пункт 2. Основная заработная плата

Основная заработная плата при выполнении работ по разработке программы включает зарплату всех сотрудников, принимающих непосредственное участие в разработке.

Основная заработная плата $Z_{осн}$ рассчитывается по формуле (1):

$$Z_{осн} = \sum_{j=1}^n Z_{средj}, \quad (1)$$

где $Z_{средj}$ - зарплата j-го сотрудника, руб.;

n - количество сотрудников, принимающих непосредственное участие в разработке системы.

В таблице 1 приведены данные, необходимые для расчёта основной заработной платы, а также результаты расчёта.

Таблица 1 - Расчёт стоимости основной заработной платы

№ п/п	Наименование трудового ресурса	Почасовая ставка, руб./ч	Трудозатраты, ч	Суммарная заработная плата, руб.
1	Менеджер проекта / Аналитик по бизнес-процессам	220.00	250	55 000.00
2	UI/UX дизайнер / Технический писатель	180.00	300	54 000.00
3	Full-stack разработчик	300.00	600	180 000.00
4	Тестировщик (QA инженер)	150.00	200	30 000.00
5	Системный администратор	190.00	150	28 500.00
	Итого:			347 500.00 руб.

Таким образом, затраты на трудовые ресурсы составляют 347 500.00 руб.

Пункт 3. Отчисления на социальные нужды

В статью «Отчисления на социальные нужды» включаются сумма страховых взносов, которые составляют 30% от затрат на оплату труда всех работников, занятых разработкой приложения [12]. Отчисления на социальные нужды составят:

$$347\,500.00 * 30\% = 104\,250.00 \text{ руб.}$$

Пункт 4. Амортизация ПЭВМ

В связи отсутствием офисного помещения или студии, организация работы предполагается на удалённой основе на оборудовании сотрудников. По данной причине эта статья затрат не будет рассматриваться.

Пункт 5. Затраты на программное обеспечение

Разработка приложения по сбору заявок на ремонт различной техники «Служба поддержки» производится при помощи бесплатного и платного программного обеспечения. Используемое программное обеспечение и его стоимость на 29.05.2024 приведена в таблице 2.

Таблица 2 - Используемое программное обеспечение и его стоимость

№ п/п	Наименование ПО	Цена одной копии	Количество копий	Суммарная плата, руб.
1	Windows 10 Pro	16 000.00	1	16 000.00
2	Microsoft Office	12 000.00	1	12 000.00
3	IntelliJ IDEA Community Edition	0.00	1	0.00
4	Figma	0.00	1	0.00
5	Microsoft Edge	0.00	1	0.00
Итого:				28 000.00

Таким образом, затраты на ПО составляет 28 000.00 руб.

Пункт 6. Затраты на электроэнергию

Затраты на электроэнергию вычисляются по формуле (2):

$$Z_{\text{ЭН}} = P_{\text{пот}} * T_{\text{вр}} * C_{\text{квт.ч}} \quad (2)$$

где $P_{\text{пот}}$ - потребляемая мощность. Т. к. предполагается работа на удалённой основе, данный показатель равен 0 кВт;

$T_{\text{вр}}$ - фонд времени за период амортизации — 0 д. * 8 ч/д. = 0 ч;

$C_{\text{квт.ч}}$ - стоимость одного киловатта энергии — 5.07 руб.

Таким образом, $Z_{\text{ЭН}} = 0 \text{ кВт} * 0 \text{ ч} * 5.07 \text{ руб.} = 0,00 \text{ руб.}$

Пункт 7. Накладные расходы

Накладные расходы составляют 10% от прямых затрат. В прямые затраты входят затраты по статьям 1-6. Таким образом, накладные расходы составляют:

$$\begin{aligned} & (0.00 + 347\,500.00 + 104\,250.00 + 0.00 + 28\,000.00 + 0.00) * 0.1 = \\ & = 47\,975.00 \text{ руб.} \end{aligned}$$

Общая себестоимость разрабатываемого программного средства составляет:

$$\begin{aligned} & 0.00 + 347\,500.00 + 104\,250.00 + 0.00 + 28\,000.00 + 0.00 + 47\,975.00 \text{ руб.} \\ & = \\ & = 527\,725.00 \text{ руб.} \end{aligned}$$

Заключение

В результате выполнения данной работы было разработано приложение «Служба поддержки» для автоматизации и оптимизации процесса подачи, обработки и выполнения заявок на ремонт техники. Основные выводы по результатам выполненной работы и по каждому разделу отдельно следующие:

- Определение и характеристика предметной области: исследован процесс подачи, обработки и выполнения заявок на ремонт техники в предприятиях и организациях, что позволило выявить ключевые проблемы и требования к системе управления заявками.

- Анализ существующих аналогов: проведен анализ существующих программных решений для управления заявками на ремонт. Выявлены их недостатки, такие как сложность интерфейса и ограниченная функциональность, что позволило сформулировать требования к новому приложению.

- Разработка технического задания: создано техническое задание, включающее функциональные и нефункциональные требования к системе. Это обеспечило четкое понимание целей и задач проекта.

- Проектирование и реализация базы данных: спроектирована и реализована база данных для хранения информации о заявках, пользователях и оборудовании, что обеспечило структурированное хранение данных и возможность их эффективного использования.

- Создание клиентской части приложения: разработана клиентская часть приложения с удобным и интуитивно понятным пользовательским интерфейсом, что позволило пользователям легко взаимодействовать с системой и оперативно подавать заявки на ремонт.

Все поставленные задачи были выполнены в полном объеме, что позволило достичь главной цели проекта - создание приложения «Служба поддержки» для автоматизации и оптимизации процесса подачи, обработки и

выполнения заявок на ремонт различной техники. Приложение позволяет минимизировать простои оборудования и удовлетворить запросы пользователей, что повышает общую эффективность работы организаций.

Рекомендации и исходные данные по конкретному использованию результатов ВКР:

- Личное использование: приложение может использоваться сотрудниками компаний для оперативной подачи заявок на ремонт техники и отслеживания их статуса.

- Корпоративное использование: внедрение в компаниях для улучшения процессов управления заявками на ремонт и повышения эффективности работы обслуживающего персонала.

Предложения по использованию разработанного решения:

- Внедрение в коммерческие проекты: приложение может быть адаптировано и интегрировано в коммерческие системы управления заявками на ремонт.

- Использование в образовательных целях: приложение может использоваться в учебных заведениях для обучения студентов процессам управления заявками и основам работы с информационными системами.

Научная, социальная и экономическая ценность работы:

- Научная ценность: работа расширяет знания в области автоматизации процессов управления заявками на ремонт, предлагая новые подходы к обеспечению удобства и эффективности использования.

- Социальная ценность: приложение обеспечивает пользователям удобный и оперативный способ подачи заявок на ремонт, что способствует поддержанию высокой работоспособности техники.

- Экономическая ценность: внедрение приложения позволяет сократить время простоя оборудования, что ведет к снижению финансовых потерь и повышению общей производительности предприятия.

Перспективы развития предложенного решения:

- Добавление новых функций: внедрение дополнительных функций, таких как прогнозирование необходимых ремонтов на основе анализа данных.

- Оптимизация производительности: улучшение производительности приложения для работы на устройствах с ограниченными ресурсами.

- Поддержка других платформ: адаптация и перенос приложения на другие операционные системы, такие как iOS и Windows, для расширения аудитории пользователей.

Работа имеет значительный потенциал для дальнейшего развития и может стать основой для создания новых продуктов и услуг в области управления заявками на ремонт техники.

Список используемых источников

1. Скляр, А. Я. Системы управления данными: учебное пособие / А. Я. Скляр, А. А. Высоцкая, А. А. Горячев. - Москва: РТУ МИРЭА, 2022. - 163 с. - Текст: электронный // Лань: электронно-библиотечная система. - URL: <https://e.lanbook.com/book/265730> (дата обращения: 06.06.2024).
2. Семенова, И. И. SQL стандарт в современных СУБД: манипулирование данными: учебное пособие / И. И. Семенова, Е. О. Шершнева. - 2-е изд., деривативн., испр. и доп. - Омск: СибАДИ, 2023. - 54 с. - ISBN 978-5-00113-242-4. - Текст: электронный // Лань: электронно-библиотечная система. - URL: <https://e.lanbook.com/book/407393> (дата обращения: 06.06.2024).
3. Хабитуев, Б. В. Программирование на языке Java: практикум : учебное пособие / Б. В. Хабитуев. - Улан-Удэ: БГУ, 2020. - 94 с. - ISBN 978-5-9793-1548-5. - Текст: электронный // Лань: электронно-библиотечная система. - URL: <https://e.lanbook.com/book/171791> (дата обращения: 06.06.2024).
4. Курбатова, И. В. Основы программирования на языке Java: учебное пособие для спо / И. В. Курбатова, А. В. Печкуров. - Санкт-Петербург: Лань, 2024. - 348 с. - ISBN 978-5-507-48516-1. - Текст: электронный // Лань: электронно-библиотечная система. - URL: <https://e.lanbook.com/book/385925> (дата обращения: 06.06.2024).
5. Болбот, О. М. Классы в языке программирования Java: учебно-методическое пособие / О. М. Болбот, В. В. Сидорик; под редакцией В. В. Сидорика. - Минск: БНТУ, 2020. - 76 с. - ISBN 978-985-550-895-4. - Текст: электронный // Лань: электронно-библиотечная система. - URL: <https://e.lanbook.com/book/248009> (дата обращения: 06.06.2024).
6. Пономарчук, Ю. В. Программирование на языке Java: учебное пособие / Ю. В. Пономарчук, И. В. Кузнецов. - Хабаровск: ДВГУПС, 2021. -

103 с. - Текст: электронный // Лань: электронно-библиотечная система. - URL: <https://e.lanbook.com/book/259451> (дата обращения: 06.06.2024).

7. Рогов, Е. В. PostgreSQL 15 изнутри: руководство / Е. В. Рогов. - Москва: ДМК Пресс, 2023. - 662 с. - ISBN 978-5-93700-178-8. - Текст: электронный // Лань: электронно-библиотечная система. - URL: <https://e.lanbook.com/book/348089> (дата обращения: 04.06.2024).

8. Локтев, Д. А. Клиент-серверное приложение на базе JavaFX: учебно-методическое пособие / Д. А. Локтев. - Москва: МГТУ им. Н.Э. Баумана, 2020. - 36 с. - ISBN 978-5-7038-5311-5. - Текст электронный // Лань: электронно-библиотечная система. - URL: <https://e.lanbook.com/book/205283> (дата обращения: 04.06.2024).

9. Токмаков, Г. П. Базы данных: Модели и структуры данных, язык SQL, программирование баз данных: учебное пособие / Г. П. Токмаков. - Ульяновск: УлГТУ, 2021. - 362 с. - ISBN 978-5-9795-2184-8. - Текст: электронный // Лань: электронно-библиотечная система. - URL: <https://e.lanbook.com/book/259706> (дата обращения: 06.06.2024).

10. Шёниг, Г. -. PostgreSQL 11. Мастерство разработки / Г. -. Шёниг; перевод с английского А. А. Слинкина. - Москва: ДМК Пресс, 2020. - 352 с. - ISBN 978-5-97060-671-1. - Текст: электронный // Лань: электронно-библиотечная система. - URL: <https://e.lanbook.com/book/131714> (дата обращения: 04.06.2024).

11. Володина, О. А. Экономика: учебное пособие / О. А. Володина, О. В. Колодяжная. - Новосибирск: СГУВТ, 2023. - 247 с. - ISBN 978-5-8119-0954-4. - Текст: электронный // Лань: электронно-библиотечная система. - URL: <https://e.lanbook.com/book/369899> (дата обращения: 05.06.2024).

12. Шлыкова, Т. Н. Экономика: методические указания / Т. Н. Шлыкова. - Самара: СамГАУ, 2022. - 84 с. - Текст: электронный // Лань: электронно-библиотечная система. - URL: <https://e.lanbook.com/book/244496> (дата обращения: 05.06.2024).

Листинг кода

```

package com.example;

import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.text.Font;
import javafx.stage.Stage;
import javax.swing.*;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.util.Locale;

public class Main extends Application {
    private static Stage primaryStage;
    private static final String DB_URL_FILE = "db_url.txt";
    private static final String ROOT_LOGIN = "postgres";
    private static final String ROOT_PASS = "root";

    static {
        try {
            Class.forName("org.postgresql.Driver");
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        }
    }

    @Override
    public void start(Stage stage) throws Exception {
        primaryStage = stage;
        Font.loadFont(getClass().getResourceAsStream("/fonts/Roboto-Regular.ttf"), 10);
        Locale.setDefault(new Locale("ru"));

        if (new File(DB_URL_FILE).exists()) {
            String dbUrl = new String(Files.readAllBytes(Paths.get(DB_URL_FILE)));
            if (testDbConnection(dbUrl)) {
                showLoginWindow();
            } else {
                JOptionPane.showMessageDialog(null, "Не удалось подключиться к базе данных по  
сохраненному адресу. Пожалуйста, введите новый адрес.", "Ошибка", JOptionPane.ERROR_MESSAGE);
                showStartupWindow();
            }
        } else {
            showStartupWindow();
        }
    }

    private void showStartupWindow() {
        String serverIp = JOptionPane.showInputDialog(null, "Введите IP базы данных:", "Настройка  
подключения к БД", JOptionPane.PLAIN_MESSAGE);
    }
}

```

```

        if (serverIp != null && !serverIp.trim().isEmpty()) {
            serverIp = serverIp.trim();
            String dbUrl = "jdbc:postgresql://" + serverIp + ":8888/postgres";
            if (testDbConnection(dbUrl)) {
                try (FileWriter writer = new FileWriter(DB_URL_FILE)) {
                    writer.write(dbUrl);
                    writer.flush(); // Добавлено чтобы убедиться, что данные записаны
                    JOptionPane.showMessageDialog(null, "Подключение успешно. Перезапустите
программу.", "Успех", JOptionPane.INFORMATION_MESSAGE);
                    System.exit(0); // Закрытие программы
                } catch (IOException e) {
                    e.printStackTrace();
                    JOptionPane.showMessageDialog(null, "Ошибка записи в файл: " + e.getMessage(),
"Ошибка", JOptionPane.ERROR_MESSAGE);
                }
            } else {
                JOptionPane.showMessageDialog(null, "Не удалось подключиться к базе данных. Проверьте
правильность IP адреса.", "Ошибка", JOptionPane.ERROR_MESSAGE);
                showStartupWindow();
            }
        } else {
            JOptionPane.showMessageDialog(null, "IP базы данных не может быть пустым", "Ошибка",
JOptionPane.ERROR_MESSAGE);
            showStartupWindow();
        }
    }

    private boolean testDbConnection(String dbUrl) {
        try (Connection connection = DriverManager.getConnection(dbUrl, ROOT_LOGIN, ROOT_PASS))
        {
            return true;
        } catch (SQLException e) {
            if (e.getMessage().contains("Connection timed out")) {
                JOptionPane.showMessageDialog(null, "Время подключения истекло. Проверьте
правильность IP адреса и доступность базы данных.", "Ошибка", JOptionPane.ERROR_MESSAGE);
            } else {
                JOptionPane.showMessageDialog(null, "Произошла ошибка при подключении к базе
данных.", "Ошибка", JOptionPane.ERROR_MESSAGE);
            }
            e.printStackTrace();
            return false;
        }
    }

    public static void showLoginWindow() throws IOException {
        Parent root = FXMLLoader.load(Main.class.getResource("/view/Login.fxml"));
        Scene scene = new Scene(root);
        scene.getStylesheets().add(Main.class.getResource("/styles.css").toExternalForm());
        primaryStage.setScene(scene);
        primaryStage.setTitle("Авторизация");
        primaryStage.show();
    }

    public static void main(String[] args) {
        launch(args);
    }
}

package com.example.util;
import java.sql.*;

public class Request {
    int id;

```

```

String serial_num;
String equip_name;
String equip_type;
String condition;
String details;
String location;
String problem_desc;
String request_comments;
String status;
String date_start;
int member_id;
String client_name;
String client_phone;
String email;
String login;
String pass;
String role;
Database database;

public Request(int id) {
    this.id = id;
    database = Database.getInstance();
    loadInfoFromDB();
}

public void updateRequestInDB(
    String equip_type,
    String problem_desc,
    String request_comments,
    String status,
    String equip_name,
    String condition,
    String details,
    String location
) {

    //    this.serial_num = equip_num;
    this.problem_desc = problem_desc;
    this.request_comments = request_comments;
    this.status = status;
    this.equip_type = equip_type;
    this.equip_name = equip_name;
    this.condition = condition;
    this.details = details;
    this.location = location;

    database = Database.getInstance();
    // Обновляем запись в таблице requests
    String updateRequestsQuery = String.format("UPDATE requests " +
        "SET problem_desc = '%s', request_comments = '%s', status = '%s' " +
        "WHERE id = %d", problem_desc, request_comments, status, id);
    database.simpleQuery(updateRequestsQuery);

    // Обновляем запись в таблице requests
    String updateEquipQuery = String.format("UPDATE equipment " +
        "SET equip_type = '%s', equip_name = '%s', condition = '%s', details = '%s', location = '%s' " +
        "WHERE serial_num = '%s'", equip_type, equip_name, condition, details, location, serial_num);
    database.simpleQuery(updateEquipQuery);

    MyAlert.showInfoAlert("Информация по заявке обновлена успешно.");
}

public void loadInfoFromDB() {

```



```

        database = Database.getInstance();
        try (Connection connection = DriverManager.getConnection(Database.URL,
Database.ROOT_LOGIN, Database.ROOT_PASS)) {
            String query = "SELECT r.id, r.serial_num, r.problem_desc, r.request_comments, r.status, " +
                "r.date_start, r.member_id, e.equip_name, e.equip_type, e.condition, e.details, e.location, " +
                "m.name, m.phone, m.email, m.login, m.pass, m.role " +
                "FROM requests r " +
                "JOIN equipment e ON r.serial_num = e.serial_num " +
                "JOIN members m ON r.member_id = m.id " +
                "WHERE r.id = ?";

            try (PreparedStatement preparedStatement = connection.prepareStatement(query)) {
                preparedStatement.setInt(1, id);
                try (ResultSet resultSet = preparedStatement.executeQuery()) {
                    if (resultSet.next()) {
                        id = resultSet.getInt("id");
                        serial_num = resultSet.getString("serial_num");
                        problem_desc = resultSet.getString("problem_desc");
                        request_comments = resultSet.getString("request_comments");
                        status = resultSet.getString("status");
                        date_start = resultSet.getString("date_start");
                        member_id = resultSet.getInt("member_id");
                        equip_name = resultSet.getString("equip_name");
                        equip_type = resultSet.getString("equip_type");
                        condition = resultSet.getString("condition");
                        details = resultSet.getString("details");
                        location = resultSet.getString("location");
                        client_name = resultSet.getString("name");
                        client_phone = resultSet.getString("phone");
                        email = resultSet.getString("email");
                        login = resultSet.getString("login");
                        pass = resultSet.getString("pass");
                        role = resultSet.getString("role");
                    }
                }
            } catch (SQLException e) {
                e.printStackTrace();
                MyAlert.showErrorAlert("Ошибка при получении информации о заявке.");
            }
        }
    }

    public int getId() {
        return id;
    }

    public String getEquip_name() {
        return equip_name;
    }

    public String getCondition() {
        return condition;
    }

    public String getDetails() {
        return details;
    }

    public String getLocation() {
        return location;
    }

    public int getMember_id() {
        return member_id;
    }

    public String getEmail() {
        return email;
    }
}

```

```

    public String getLogin() {
        return login;
    }
    public String getPass() {
        return pass;
    }
    public String getRole() {
        return role;
    }
    public Database getDatabase() {
        return database;
    }
    public String getEquip_type() {
        return equip_type;
    }
    public String getProblem_desc() {
        return problem_desc;
    }
    public String getClient_name() {
        return client_name;
    }
    public String getClient_phone() {
        return client_phone;
    }
    public String getSerial_num() {
        return serial_num;
    }
    public String getStatus() {
        return status;
    }
    public String getDate_start() {
        return date_start;
    }
    public String getRequest_comments() {
        return request_comments;
    }
}

package com.example.util;
import org.postgresql.PGConnection;
import org.postgresql.PGNotification;
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.sql.*;
import java.util.ArrayList;
import java.util.List;

public class Database {
    private static Database instance;
    public static String URL;
    public static final String ROOT_LOGIN = "postgres";
    public static final String ROOT_PASS = "root";
    private Connection externalConnection = null;
    private Connection notificationConnection = null;
    private PGConnection pgConnection;
    private Thread notificationThread;
    private volatile boolean listening = true;
    private List<NotificationListener> listeners = new ArrayList<>();

    static {
        try {
            URL = new String(Files.readAllBytes(Paths.get("db_url.txt")));

```

```

    } catch (IOException e) {
        e.printStackTrace();
    }
}

private Database() {
    try {
        notificationConnection = DriverManager.getConnection(URL, ROOT_LOGIN, ROOT_PASS);
        pgConnection = (PGConnection) notificationConnection;
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

public static Database getInstance() {
    if (instance == null) {
        instance = new Database();
    }
    return instance;
}

public boolean accessToDB(String login, String password) {
    try (Connection connection = DriverManager.getConnection(URL, ROOT_LOGIN, ROOT_PASS)) {
        return true;
    } catch (SQLException e) {
        System.out.println(e);
        return false;
    }
}

public Connection getConnection() throws SQLException {
    externalConnection = DriverManager.getConnection(URL, ROOT_LOGIN, ROOT_PASS);
    return externalConnection;
}

public void closeConnection() throws SQLException {
    if (externalConnection != null) {
        externalConnection.close();
    }
}

public void listenForNotifications(String channel) {
    try (Statement stmt = notificationConnection.createStatement()) {
        stmt.execute("LISTEN " + channel);
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

public PGNotification[] getNotifications() {
    try {
        return pgConnection.getNotifications();
    } catch (SQLException e) {
        e.printStackTrace();
        return null;
    }
}

public void startNotificationListener() {
    notificationThread = new Thread(() -> {
        while (listening) {
            PGNotification[] notifications = getNotifications();
            if (notifications != null) {
                for (PGNotification notification : notifications) {
                    handleNotification(notification);
                }
            }
        }
    })
}

```

```

        try {
            Thread.sleep(500);
        } catch (InterruptedException e) {
            Thread.currentThread().interrupt();
        }
    }
});
notificationThread.setDaemon(true);
notificationThread.start();
}
private void handleNotification(PGNotification notification) {
    System.out.println("Received notification: " + notification.getParameter());
    for (NotificationListener listener : listeners) {
        listener.onNotification(notification.getName(), notification.getParameter());
    }
}
public void stopNotificationListener() {
    listening = false;
    if (notificationThread != null) {
        notificationThread.interrupt();
    }
}
public void addNotificationListener(NotificationListener listener) {
    listeners.add(listener);
}
public void removeNotificationListener(NotificationListener listener) {
    listeners.remove(listener);
}
public ResultSet getTable(String selectedTable, String orderBy) {
    try (Connection connection = DriverManager.getConnection(URL, ROOT_LOGIN, ROOT_PASS)) {
        PreparedStatement statement = connection.prepareStatement(
            "SELECT * FROM " + selectedTable + " ORDER BY " + orderBy + " ASC");
        return statement.executeQuery();
    } catch (SQLException e) {
        e.printStackTrace();
        return null;
    }
}
public ArrayList<String> getAllTableColumnNames(String tableName) {
    ArrayList<String> columnNames = new ArrayList<>();
    try (Connection connection = DriverManager.getConnection(Database.URL,
Database.ROOT_LOGIN, Database.ROOT_PASS)) {
        DatabaseMetaData metaData = connection.getMetaData();
        try (ResultSet columns = metaData.getColumns(null, null, tableName, null)) {
            while (columns.next()) {
                String columnName = columns.getString("COLUMN_NAME");
                columnNames.add(columnName);
            }
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return columnNames;
}
public ArrayList<String> stringListQuery(String neededColumn, String table, String where, String
orderBy) {
    ArrayList<String> finalList = new ArrayList<>();
    String finalQuery = "SELECT " + neededColumn + " FROM " + table;
    if (where != null) {
        finalQuery += " WHERE " + where;
    }
    if (orderBy != null) {

```

```

        finalQuery += " ORDER BY " + orderBy;
    }
    try (Connection connection = DriverManager.getConnection(URL, ROOT_LOGIN, ROOT_PASS);
        PreparedStatement statement = connection.prepareStatement(finalQuery.trim());
        ResultSet resultSet = statement.executeQuery()) {
        while (resultSet.next()) {
            finalList.add(resultSet.getString(neededColumn));
        }
    } catch (SQLException e) {
        e.printStackTrace();
        return null;
    }
    return finalList;
}

public ArrayList<String> stringListQuery(String neededColumn, String sql) {
    ArrayList<String> finalList = new ArrayList<>();
    try (Connection connection = DriverManager.getConnection(URL, ROOT_LOGIN, ROOT_PASS);
        PreparedStatement statement = connection.prepareStatement(sql.trim());
        ResultSet resultSet = statement.executeQuery()) {
        while (resultSet.next()) {
            finalList.add(resultSet.getString(neededColumn));
        }
    } catch (SQLException e) {
        e.printStackTrace();
        return null;
    }
    return finalList;
}

public String singleValueQuery(String sql) {
    try (Connection connection = DriverManager.getConnection(URL, ROOT_LOGIN, ROOT_PASS);
        Statement statement = connection.createStatement();
        ResultSet resultSet = statement.executeQuery(sql)) {
        if (resultSet.next()) {
            return resultSet.getString(1);
        }
    } catch (SQLException e) {
        throw new RuntimeException(e);
    }
    return "";
}

public ArrayList<String> executeQueryAndGetColumnValues(String query) {
    ArrayList<String> columnValues = new ArrayList<>();
    try (Connection connection = DriverManager.getConnection(Database.URL,
Database.ROOT_LOGIN, Database.ROOT_PASS);
        Statement statement = connection.createStatement();
        ResultSet resultSet = statement.executeQuery(query)) {
        ResultSetMetaData metaData = resultSet.getMetaData();
        int columnCount = metaData.getColumnCount();
        while (resultSet.next()) {
            for (int i = 1; i <= columnCount; i++) {
                columnValues.add(resultSet.getString(i));
            }
        }
    } catch (SQLException e) {
        throw new RuntimeException("Ошибка выполнения запроса к базе данных", e);
    }
    return columnValues;
}

public boolean updateTable(String selectedTable, String columnChangeName, String newRecord, String
columnSearchName, String columnSearch) {
    try (Connection connection = DriverManager.getConnection(URL, ROOT_LOGIN, ROOT_PASS);
        PreparedStatement statement = connection.prepareStatement(

```

```

        "UPDATE " + selectedTable +
        " SET " + columnChangeName + " = ?" +
        " WHERE " + columnSearchName + " = " + columnSearch)) {
    statement.setObject(1, convertStringToInteger(newRecord));
    return statement.executeUpdate() != -1;
} catch (SQLException e) {
    e.printStackTrace();
    return false;
}
}
private Object convertStringToInteger(String str) {
    try {
        return Integer.parseInt(str);
    } catch (NumberFormatException e) {
        return str;
    }
}
public boolean deleteQuery(String selectedTable, String columnSearchName, String columnSearch) {
    try (Connection connection = DriverManager.getConnection(URL, ROOT_LOGIN, ROOT_PASS);
        PreparedStatement statement = connection.prepareStatement(
            "DELETE FROM " + selectedTable + " WHERE " + columnSearchName + " = " +
columnSearch)) {
        return statement.executeUpdate() != -1;
    } catch (SQLException e) {
        e.printStackTrace();
        return false;
    }
}
public boolean updateQuery(String table, String set, String where) {
    try (Connection connection = DriverManager.getConnection(URL, ROOT_LOGIN, ROOT_PASS);
        PreparedStatement preparedStatement = connection.prepareStatement(
            "UPDATE " + table + " SET " + set + " WHERE " + where)) {
        return preparedStatement.executeUpdate() > 0;
    } catch (SQLException e) {
        e.printStackTrace();
        return false;
    }
}
public boolean simpleQuery(String sql) {
    try (Connection connection = DriverManager.getConnection(URL, ROOT_LOGIN, ROOT_PASS);
        PreparedStatement statement = connection.prepareStatement(sql)) {
        return statement.executeUpdate() != -1;
    } catch (SQLException e) {
        e.printStackTrace();
        return false;
    }
}
public interface NotificationListener {
    void onNotification(String channel, String payload);
}
}

```