|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Министерство науки и высшего образования Российской Федерации  Федеральное государственное бюджетное образовательное учреждение высшего образования  «Московский государственный технический университет имени Н.Э. Баумана»  (национальный исследовательский университет)»  Московский техникум космического приборостроения  УТВЕРЖДАЮ  Заместитель директора по УР  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Т.Н. Михайлова  (подпись, дата)   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | **ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**   |  | | --- | | **РАЗРАБОТКА ПРОГРАММНОЙ СИСТЕМЫ УДАЛЕННОГО** | | **УПРАВЛЕНИЯ ФАЙЛАМИ И ОПЕРАЦИОННЫМИ СИСТЕМАМИ** |   Пояснительная записка  Группа ТИП-83 | | | | | Председатель предметной (цикловой) комиссии |  |  | Е.А. Митрошенкова | | (подпись, дата) |  | (ФИО) | | Руководитель разработки  от техникума |  |  | О.Ю. Малыхина | | (подпись, дата) |  | (ФИО) | | Рецензент |  |  | Ш.Р. Кесамиров | | (подпись, дата) |  | (ФИО) | | Руководитель разработки  от предприятия |  |  | А.В. Шулькин | | (подпись, дата) |  | (ФИО) | | Консультант по  экономической части |  |  | С.А. Ковалев | | (подпись, дата) |  | (ФИО) | | Разработчик |  |  | А.Р. Петрунин | | (подпись, дата) |  | (ФИО) |   Москва, 2023 |

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Московский государственный технический университет имени Н.Э. Баумана»

(национальный исследовательский университет)»

Московский техникум космического приборостроения

Срок окончания работы « » июня 2023 г.

Зав. отделением О.В. Коротченко Дата « » 20 г.

(подпись)

ЗАДАНИЕ

на выполнение дипломного проекта

Студенту Петрунин Артем Романович

(фамилия, имя, отчество)

Тема работы Разработка программной системы удаленного управления файлами и операционными системами

Введение

1. Теоретическое обоснование разрабатываемого программного продукта
   1. Описание предметной области
   2. Сравнительный анализ программ-аналогов
   3. Моделирование проектируемой системы
   4. Разработка функциональных требований к программе
   5. Обоснование выбора средств реализации программной системы
   6. Вывод по разделу
2. Разработка программного продукта
   1. Разработка архитектура программной системы
   2. Разработка структуры данных
   3. Конструирование пользовательского интерфейса
   4. Схемы алгоритма программы и подпрограмм
   5. Отладка и тестирование программы
   6. Руководство пользователя
   7. Вывод по разделу
3. Охрана труда
4. Технико-экономическая часть

Заключение

Список использованных источников

Приложения

Презентация

Дата выдачи задания « » 20 г.

Руководитель дипломного проекта от техникума Малыхина О.Ю.

(подпись, дата) (ФИО)

Руководитель работы от предприятия Шулькин А.В.

(подпись, дата) (ФИО)

Срок сдачи обучающимся готовой работы: г.

Задание принял к исполнению:

# СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 4](#_Toc138160555)

[1 Теоретическое обоснование разрабатываемого программного продукта 8](#_Toc138160556)

[1.1 Описание предметной области 8](#_Toc138160557)

[1.2 Сравнительный анализ программ-аналогов 10](#_Toc138160558)

[1.3 Моделирование проектируемой системы 13](#_Toc138160559)

[1.4 Разработка функциональных требований к программной системе 14](#_Toc138160560)

[1.4.1 Назначение программы 14](#_Toc138160561)

[1.4.2 Требования к функциональным характеристикам 14](#_Toc138160562)

[1.4.3 Требования к надежности 14](#_Toc138160563)

[1.4.4 Требования к составу и параметрам технических средств 15](#_Toc138160564)

[1.4.5 Требования к информационной и программной совместимости 15](#_Toc138160565)

[1.4.6 Требования к программной документации 15](#_Toc138160566)

[1.5 Обоснование выбора средств реализации программной системы 17](#_Toc138160567)

[1.6 Вывод по разделу 18](#_Toc138160568)

[2 Разработка программного продукта 19](#_Toc138160569)

[2.1 Разработка архитектуры программной системы 19](#_Toc138160570)

[2.2 Разработка структуры данных 21](#_Toc138160571)

[2.3 Конструирование пользовательского интерфейса 23](#_Toc138160572)

[2.4 Схемы алгоритма программы и подпрограмм 26](#_Toc138160573)

[2.4.1 Схема алгоритма функции authorization 26](#_Toc138160574)

[2.4.2 Схема алгоритма функции registration 27](#_Toc138160575)

[2.4.3 Схема алгоритма функции createGroup 28](#_Toc138160576)

[2.4.4 Схема алгоритма функции createConnection 29](#_Toc138160577)

[2.5 Отладка и тестирование программы 30](#_Toc138160578)

[2.5.1 Отладка программы 30](#_Toc138160579)

[2.5.2 Тестирование программы 31](#_Toc138160580)

[2.5.2.1 Тестирование программы в нормальных условиях 32](#_Toc138160581)

[2.5.2.2 Тестирование программы в экстремальных условиях 35](#_Toc138160582)

[2.5.2.3 Тестирование программы в исключительных ситуациях 37](#_Toc138160583)

[2.5.2.4 Сообщения 39](#_Toc138160584)

[2.6 Руководство пользователя 40](#_Toc138160585)

[2.6.1 Назначение программы 40](#_Toc138160586)

[2.6.2 Условия работы программы 40](#_Toc138160587)

[2.6.3 Входные данные 41](#_Toc138160588)

[2.6.4 Выполнение программы 42](#_Toc138160589)

[2.7 Вывод по разделу 46](#_Toc138160590)

[3 Охрана труда 47](#_Toc138160591)

[3.1 Техника безопасности при работе на персональном компьютере 47](#_Toc138160592)

[3.2 Требования к помещению 49](#_Toc138160593)

[3.3 Требования к пожарной безопасности 50](#_Toc138160594)

[4 Техником-экономическая часть 51](#_Toc138160595)

[4.1 Технико-экономические расчеты экономической целесообразности программы 51](#_Toc138160596)

[4.2 Расчетная часть 54](#_Toc138160597)

[4.2.1 Расчет трудоемкости разработанной программы 54](#_Toc138160598)

[4.2.2 Расчет себестоимости разработанной программы 57](#_Toc138160599)

[4.2.3 Анализ возможных путей снижения себестоимости 62](#_Toc138160600)

[4.3 Графическая часть 64](#_Toc138160601)

[ЗАКЛЮЧЕНИЕ 67](#_Toc138160602)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 68](#_Toc138160603)

[ПРИЛОЖЕНИЯ 69](#_Toc138160604)

[ПРИЛОЖЕНИЕ А 70](#_Toc138160605)

[ПРИЛОЖЕНИЕ Б 100](#_Toc138160606)

# ВВЕДЕНИЕ

Тема: Разработка системы удаленного управления и операционными системами. В современном мире удаленная работа и управление компьютерными системами становятся неотъемлемой частью работы многих компаний и организаций. Благодаря быстрому развитию технологий и широкому доступу к интернету, удаленный доступ к файлам и возможность управления операционными системами стали необходимостью для обеспечения эффективности и безопасности рабочих процессов. Это особенно актуально для компаний с географически распределенными командами, филиалами или сотрудниками, работающими в удаленном режиме.

Однако, при осуществлении удаленной работы возникают ряд проблем, таких как безопасность данных, удобство доступа и управления системами, а также совместная работа над проектами. Для решения этих проблем и обеспечения эффективного удаленного рабочего процесса необходимы специализированные инструменты и системы управления файлами и операционными системами.

Системы удаленного управления позволяют пользователям получать доступ к компьютерам и серверам из любой точки мира через интернет. Они предоставляют возможность удаленного управления и контроля над операционной системой, файлами, приложениями и другими ресурсами компьютерной системы. Такие системы могут быть основаны на различных протоколах и технологиях, таких как Remote Desktop Protocol (RDP), Virtual Network Computing (VNC) и Secure Shell (SSH), и обеспечивают шифрование и аутентификацию для защиты конфиденциальности данных.

Системы управления файлами позволяют эффективно организовывать и обмениваться файлами между удаленными пользователями. Они обеспечивают возможность загружать, скачивать, редактировать и совместно работать над файлами, а также устанавливать права доступа и контролировать версии файлов. Такие системы часто включают в себя функции облачного хранения, синхронизации и резервного копирования данных, обеспечивая надежность и доступность файлов в любое время и из любого места.

В целом, разработка систем удаленного управления и операционными системами является актуальной задачей, которая позволяет компаниям и организациям обеспечить безопасность, эффективность и гибкость в рамках удаленной работы. Такие системы способствуют повышению производительности и сокращению затрат, а также обеспечивают комфортные условия для сотрудников, позволяя им эффективно совмещать работу и личные обязанности.

С развитием технологий удаленного управления и операционными системами, компании и организации получают ряд преимуществ. Во-первых, удаленный доступ к файлам и операционным системам обеспечивает гибкость и мобильность. Сотрудники могут работать из любой точки мира, не привязываясь к конкретному рабочему месту или офису. Это особенно полезно для компаний с дистанционными или международными командами, а также для сотрудников, работающих в командировках или на удаленных объектах.

Во-вторых, удаленное управление и операционные системы позволяют быстро реагировать на проблемы и проводить техническую поддержку удаленных пользователей. Администраторы систем могут удаленно настраивать и обновлять программное обеспечение, решать проблемы сети и диагностировать ошибки без необходимости физического присутствия на месте.

Кроме того, системы удаленного управления и операционными системами способствуют повышению безопасности данных. Они обеспечивают шифрование и аутентификацию, защищая информацию от несанкционированного доступа и внешних угроз. Благодаря возможностям удаленного удаления данных в случае утери или кражи устройства, компании могут снизить риск утечки конфиденциальной информации.

Кроме того, удаленное управление операционными системами способствует более эффективному использованию ресурсов. Администраторы могут мониторить и оптимизировать производительность систем, распределять вычислительные мощности и управлять доступом к ресурсам в режиме реального времени. Это позволяет снизить издержки и повысить эффективность работы компьютерных систем.

Однако, несмотря на все преимущества удаленного управления и операционными системами, существуют и некоторые вызовы и риски. Важно обеспечить надежность и защиту системы от возможных угроз, таких как хакерские атаки или вирусы. Также следует учесть вопросы приватности и соблюдения законодательства о защите данных при удаленной работе и доступе к конфиденциальной информации.

В целом, разработка систем удаленного управления и операционными системами становится все более важной и актуальной. Компании и организации, осознавая преимущества таких систем, стремятся использовать современные инструменты и технологии для обеспечения безопасности, эффективности и гибкости рабочих процессов в условиях удаленной работы.

Целью дипломной работы является разработка системы удаленного управления файлами и операционными системами для обеспечения эффективной и безопасной удаленной работы.

Для создания такой системы нужно решить следующие задачи:

1. Произвести анализ предметной области;
2. Осуществить проектирование программного продукта с построением диаграмм потоков данных, функциональной диаграммы;
3. Осуществить проектирование БД;
4. Разработать интерфейс;
5. Определить экономическую эффективность жизненного цикла системы;
6. Разработать руководство по эксплуатации системы.

Объектом исследования данного проекта является компания ООО «СМАРТ СИСТЕМС».

Предметом исследования является процесс технической поддержки сотрудников в компании ООО «СМАРТ СИСТЕМС».

ООО «СМАРТ СИСТЕМС» - это IT компания, специализирующаяся на разработке и предоставлении инновационных решений в области информационных технологий. Компания занимается широким спектром деятельности, охватывающим различные аспекты IT сферы. Вот некоторые из ключевых направлений работы компании:

Разработка программного обеспечения: ООО «СМАРТ СИСТЕМС» разрабатывает пользовательские приложения, веб-приложения, мобильные приложения и другое программное обеспечение для различных платформ и операционных систем. Компания имеет опыт в создании индивидуальных решений под конкретные потребности клиентов.

Интеграция и внедрение систем: Компания предлагает услуги по интеграции различных информационных систем и платформ, а также их внедрению в организации. Это включает в себя настройку и адаптацию системы под требования клиента, обучение пользователей и поддержку в процессе внедрения.

Системная инфраструктура и сетевые решения: ООО «СМАРТ СИСТЕМС» специализируется на проектировании, развертывании и поддержке корпоративной сетевой инфраструктуры. Компания предлагает услуги по настройке серверов, сетевого оборудования, виртуализации, облачных решений, а также обеспечивает безопасность сети и защиту от угроз.

Консалтинг и IT-стратегия: Компания оказывает консультационные услуги по вопросам информационных технологий и разработке IT-стратегии для организаций. Это включает в себя анализ текущей IT инфраструктуры, определение потребностей и целей клиента, разработку плана действий и рекомендаций по оптимизации использования технологий.

Безопасность и защита данных: ООО «СМАРТ СИСТЕМС» предлагает решения и услуги по обеспечению безопасности информации и защите данных. Компания разрабатывает стратегии безопасности, реализует системы защиты от внешних угроз, проводит аудит безопасности, внедряет системы резервного копирования и восстановления данных.

Техническая поддержка и обслуживание: Компания предоставляет техническую поддержку и обслуживание IT инфраструктуры клиентов. Это включает в себя решение текущих технических проблем, обновление и мониторинг систем, а также предоставление консультаций и помощи пользователям.

ООО «СМАРТ СИСТЕМС» стремится предоставлять клиентам современные и инновационные решения, помогая им оптимизировать бизнес-процессы, повысить эффективность работы и обеспечить безопасность информации. Компания ориентирована на индивидуальный подход к каждому клиенту, адаптируя свои услуги под конкретные требования и потребности заказчика.

Разработка системы удаленного управления файлами и операционными системами позволит сотрудникам компании эффективно и безопасно работать удаленно. С помощью этой системы сотрудники смогут получить доступ к нужным файлам и приложениям с любого устройства и из любой точки мира, что позволит им быть более мобильными и продуктивными. Также система обеспечит высокий уровень безопасности и защиты данных, что важно для компаний, работающих с конфиденциальной информацией.

# Теоретическое обоснование разрабатываемого программного продукта

## Описание предметной области

Система удаленного управления файлами и операционными системами - область IT и компьютерных сетей, позволяющая удаленно управлять операционными система-ми и файлами на удаленных компьютерах. Разработка такой системы включает в себя проектирование архитектуры, разработку ПО, настройку безопасности, тестирование и оптимизацию производительности. Она используется для управления компьютерными парками, удаленного обучения, виртуализации рабочих мест и т.д.

В текущее время, если у сотрудников компании появляются проблемы технического плана, они приходят в отдел техподдержки лично.

Требуется разработать ИС которая предоставит функционал для удаленного подключения к операционной системе.

Данная ИС будет представлен в качестве приложения.

Для более наглядного представления на рисунке 1.1 приведена диаграмма IDEF0, модель «AS-IS».

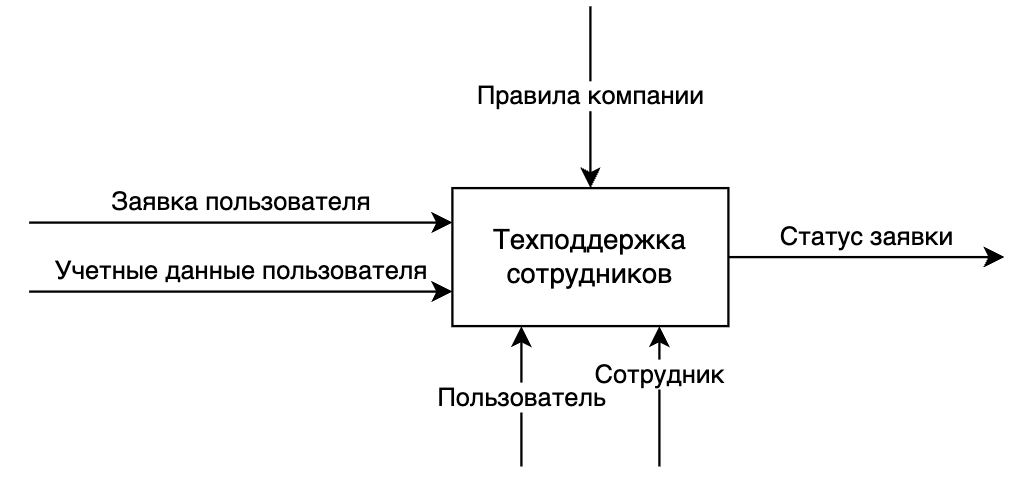


Рисунок 1.1 ‒ Диаграмма IDEF0, модель «AS-IS» для процесса «Техподдержка сотрудников»

Результат построения декомпозиции диаграммы IDEF0, модель «AS-IS» приведен на рисунке 1.2.

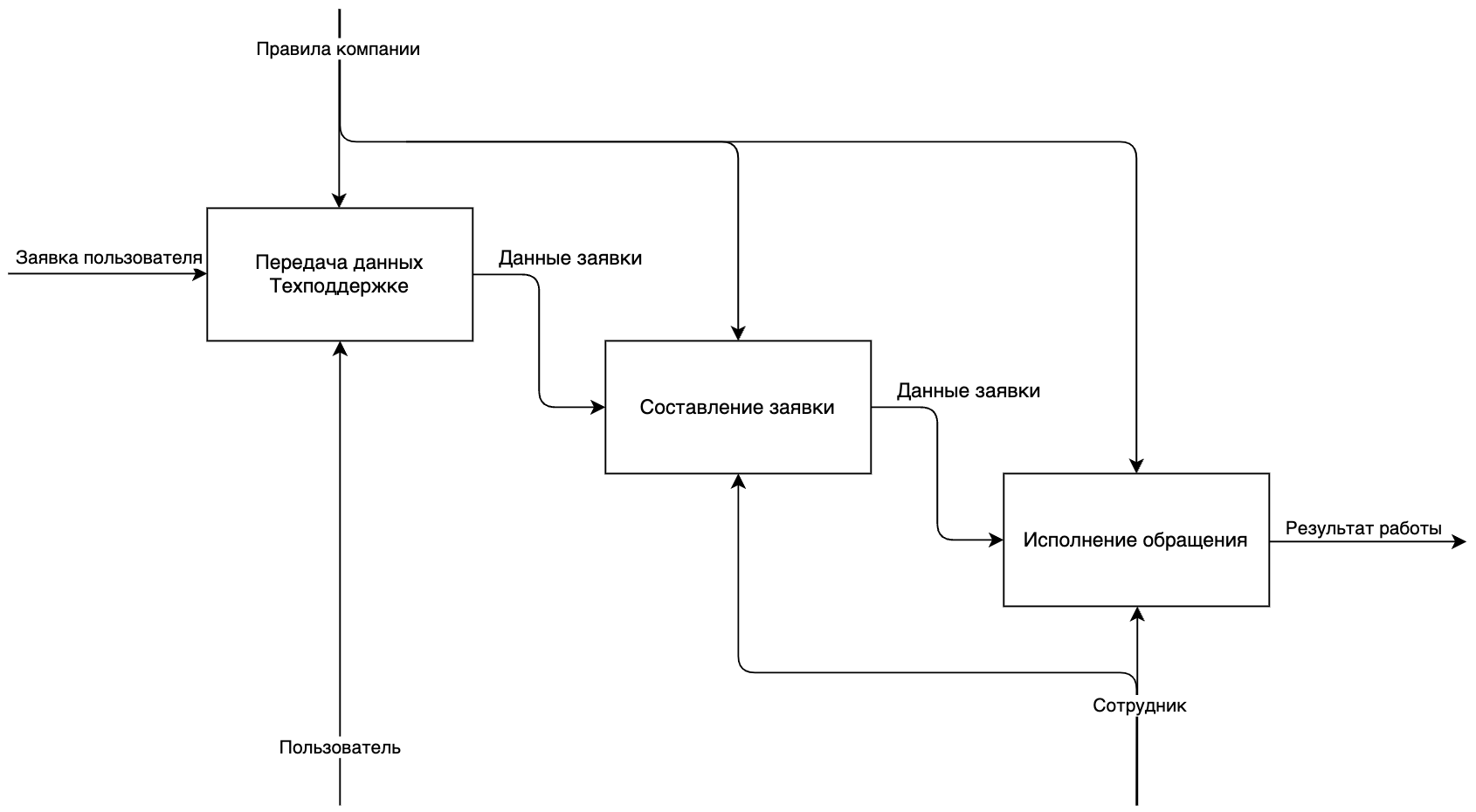


Рисунок 1.2 ‒ Декомпозиция диаграммы IDEF0, модель «AS-IS»

## Сравнительный анализ программ-аналогов

Примерами программ-аналогов разрабатываемой системы являются следующие решения:

* «Termius»;
* «Putty».

Рассмотрим каждую из приведенных прогамм.

«Termius» эта программа является одной из наиболее распространенных систем для удаленного управления файлами и операционными системами. Основные функции программы:

* создание соединений;
* возможность подключения по протоколу SFTP и SSH.

Для более наглядного представления на рисунке 1.3 приведен интерфейс программы:

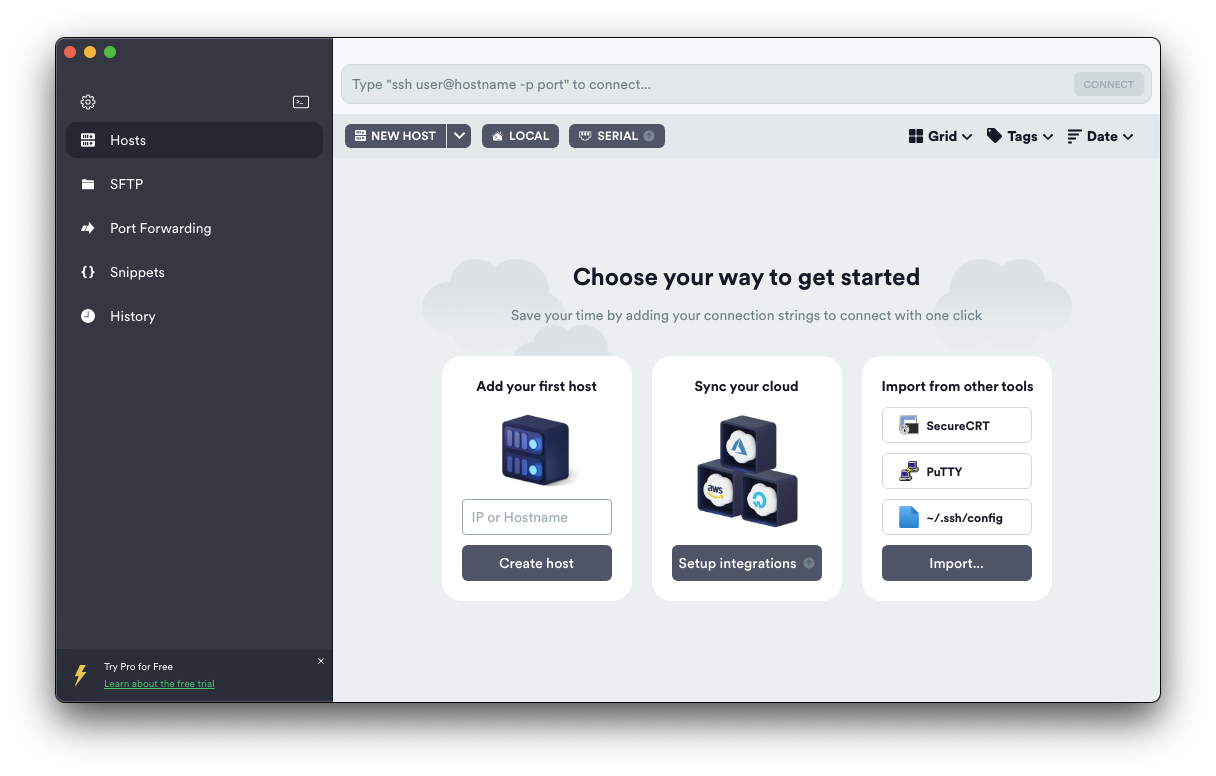


Рисунок 1.3 ‒ Интерфейс программы «Termius»

Достоинства программы:

* позволяет быстро выполнять подключение и манипулирование файлами;
* красивый интерфейс.

Недостатки программы:

* высокая стоимость;
* требует наличия определнных навыков для ее настройки и администрирования.

«Putty» - эта программа является бесплатной и открытой системой для удаленного управления файлами и операционными системами. Основные функции программы:

* возможность удаленного подключения через различные протоколы.

Для более наглядного представления на рисунке 1.4 приведен интерфейс программы:

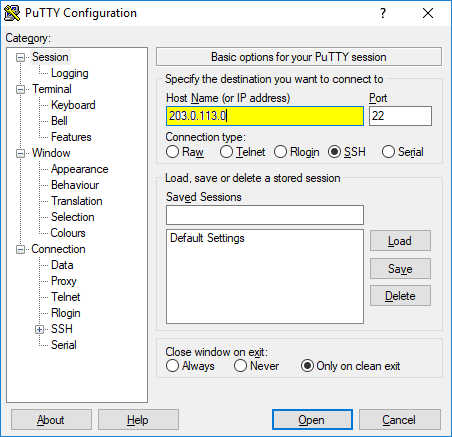


Рисунок 1.4 ‒ Интерфейс программы «Putty»

Достоинства программы:

* бесплатная.

Недостатки программы:

* устаревший функционал;
* не продуманный интерфейс.

Ниже представлена таблица 1.1, в которой приведен анализ программ-аналогов

Таблица 1.1 — «Анализ программ-аналогов»

|  |  |  |  |
| --- | --- | --- | --- |
| Характеристика | Termius | Putty | Дипломный проект |
| Цена | Высокая | Бесплатная | Бесплатная |
| Установка и настройка | Требуются навыки | Требуются навыки | Простая |
| Скорость работы | Высокая | Средняя | Высокая |
| Масштабируемость | Средняя | Средняя | Средняя |

Результат сравнительного анализа показал, что каждая из программ-аналогов имеет свои преимущества и недостатки, и для разработки приложения было принято решение использовать собственное решение с учетом конкретных потребностей.

## Моделирование проектируемой системы

На рисунке 1.6 приведена диаграмма IDEF0, модель «TO-BE».

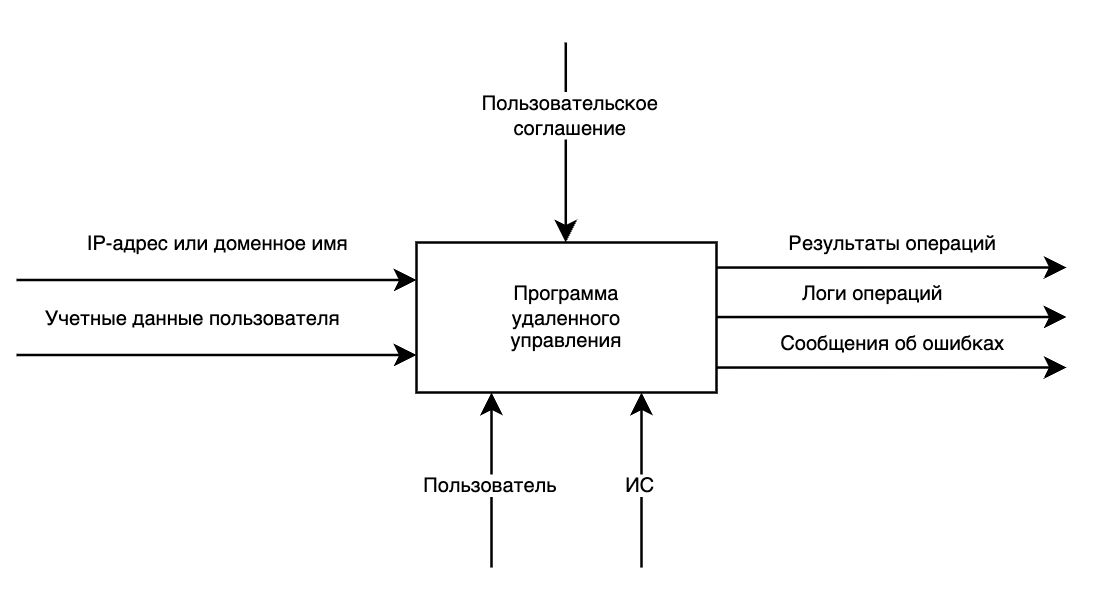


Рисунок 1.6 ‒ Диаграмма IDEF0, модель «TO-BE» для процесса

«Техподдержка сотрудников»

Управление процессом включает в себя политику пользовательского соглашения, которые необходимы для обеспечения правил использования.

Процесс начинается с добавления пользователем данных для подключения, после чего ИС проверяет доступность использую текущие данные.

После проверки доступности, выполняется команда, которую ввел пользователь и ее результат. Результат построения декомпозиции диаграммы IDEF0, модель «TO-BE» приведен на рисунке 1.7.

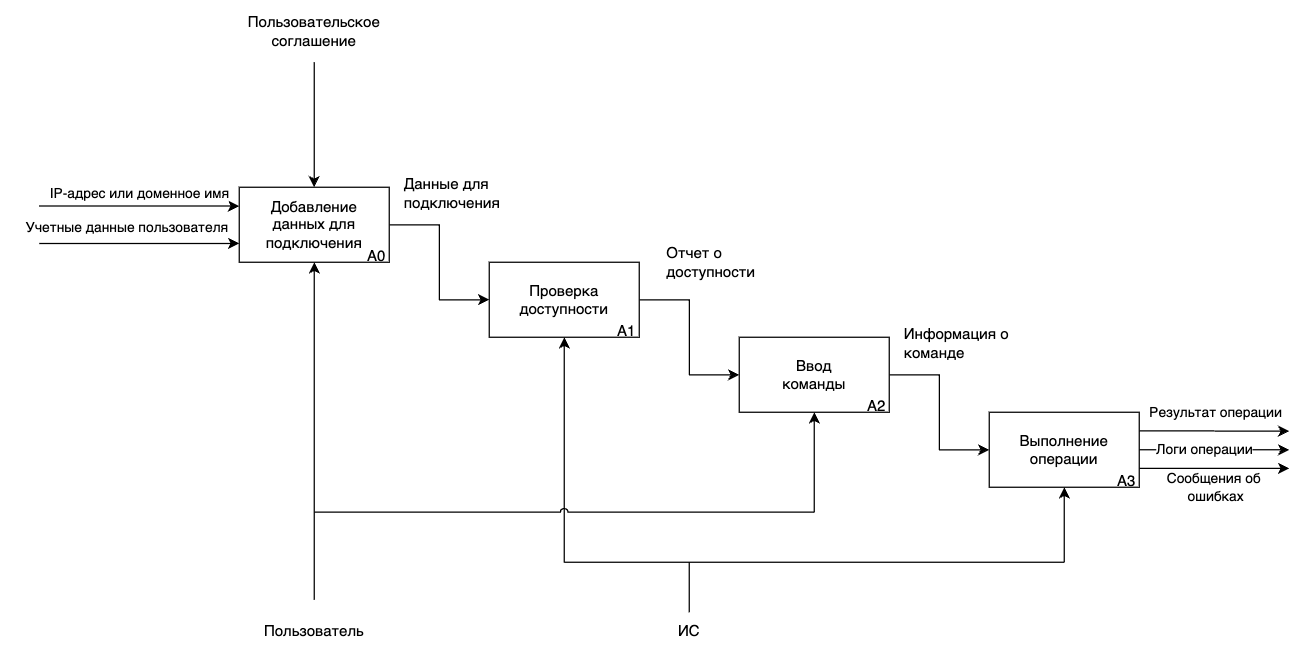


Рисунок 1.7 ‒ Декомпозиция диаграммы IDEF0, модель «TO-BE»

## Разработка функциональных требований к программной системе

### Назначение программы

Назначением программы разработки системы удаленного управления файлами и операционными системами является обеспечение эффективной и безопасной удаленной работы сотрудников компании, что позволит повысить их производительность и мобильность, а также обеспечить высокий уровень защиты данных. Программа позволит сотрудникам получать доступ к необходимым файлам и приложениям с любого устройства и из любой точки мира, а также обеспечит быстрый и удобный доступ к технической поддержке. Кроме того, система удаленного управления позволит оптимизировать работу IT-отдела компании, ускорив процессы технической поддержки и повысив качество обслуживания.

### Требования к функциональным характеристикам

Программа должна обеспечивать возможность выполнения следующих функций:

* + управление фалйами: система должна обеспечивать возможность удаленного управления файлами, включая загрузку и выгрузку, а так же перемещение между удаленным и локальным компьютером;
  + Удаленный доступ: система должна обеспечивать безопасный удаленный доступ к удаленному компьютеру с помощью протоколов, таких как SSH;

### Требования к надежности

Надежность технических средств обеспечивается использованием сертифицированных средств вычислительной техники и их комплектующих. При разработке ПО важным критерием является наделжность. Это означает, что программа должна работать стабильно и без сбоев, а также обеспечивать безопасность и конфиденциальность данных. Для обеспечения надежности программы следует учитывать следующие требования:

* предусмотреть обработку исключений, связанных с вводом пользователем неправильных данных;
* стабильность работы: программа должна работать стабильно и без сбоев;
* резервное копирование: программа должна иметь механизмы резервного копирования данных, чтобы в случае сбоя или потери данных можно было быстро восстановить работоспособность системы.

### Требования к составу и параметрам технических средств

Минимальные системные требования:

* + процессор с тактовой частотой не менее 2,0 ГГц;
  + объем оперативного запоминающего устройства – 4 Гб и более;
  + объем постоянного запоминающего устройства – 16 Гб и более.

### Требования к информационной и программной совместимости

* + Приложение должно быть разработано на языке программирования TypeScript.
  + Данные пользователей и заказов должны храниться в базе данных PostgreSQL.
  + Приложение должно поддерживать передачу данных через протокол HTTPS и использовать механизмы шифрования для защиты конфиденциальной информации.
  + Приложение должно быть совместимо с операционными системами Windows и Linux.

### Требования к программной документации

Программа должна быть интуитивно понятной и расширяемой при необходимости. Разработанные программные модули (программа) должны быть самодокументированы, т.е. листинг программы должен содержать все необходимые комментарии.

В состав сопровождающей документации должны входить:

1. Пояснительная записка, содержащая описание разработки:
   * схемы алгоритма программы;
   * отладка программы;
   * иестирование программы;
   * руководство пользователя.
2. Приложения:
   * листинг программы;
   * результаты выполнения программы.

Разрабатываемая программа должна включать подсказки пользователю для более удобной работы.

## Обоснование выбора средств реализации программной системы

Для реализации серверной части был выбран язык программирования TypeScript и хостинг firstvds.ru. Для реализации клиентской части - библиотека React. Данный выбор обусловлен рядом преимуществ, которые позволят эффективно и быстро создать новое ПО.

В первую очередь, TypeScript был выбран из-за его типизации, которая позволя-ет уменьшить количество ошибок в коде и сделать его более понятным и легко под-держиваемым. TypeScript также является совместимым с JavaScript, что позволит ис-пользовать существующий код и библиотеки.

Вторым выбранным инструментом является хостинг firstvds.ru. Он был выбран благодаря его надежности и доступности. Регистрация и настройка домена производит-ся просто и быстро, а цены являются конкурентными на рынке.

Библиотека React, который позволяет быстро создавать интерактивные пользо-вательские интерфейсы. React также обладает хорошей производительностью и мас-штабируемостью, что позволит удовлетворить потребности кондитерского предприятия и его клиентов.

В итоге, выбранный инструментарий является оптимальным для реализации программной системы для кондитерского предприятия. TypeScript и React позволят со-здать качественный и интерактивный пользовательский интерфейс, а хостинг firstvds.ru обеспечит доступность и надежность системы.

## Вывод по разделу

В разделе 1.1 была рассмотрена предметная область разработки приложения для разработки системы удаленного управления файлами и операционными системами. Были представлены диаграммы IDEF0, которые показывают процесс обработки данных в информационной системе и политику управления процессом.

В разделе 1.2 был проведен анализ существующих программ-аналогов и определены их основные преимущества и недостатки.

В разделе 1.3 были приведены диаграммы:

* Последовательности;
* Деятельности;
* USE CASE.

В разделе 1.4 были разработаны функциональные требования к программной системе. Была определена функциональность каждого модуля, а также определены требования к интерфейсу приложения.

В разделе 1.5 был обоснован выбор средств реализации программной системы. Были рассмотрены различные технологии и языки программирования, которые могут быть использованы для разработки приложения. Также были определены требования к хостингу и базе данных.

Таким образом, проектирование и разработка является актуальной задачей, которая позволит упростить и оптимизировать работу. Реализация данного проекта требует тщательного анализа и выбора средств реализации, а также разработки функциональных требований к программной системе и моделирования проектируемой системы.

# Разработка программного продукта

## Разработка архитектуры программной системы

Структура программы представлена на рисунке 2.1.



Рисунок 2.1 ‒ Структура программы

В таблице 2.1 представлены методы, используемые в программе и их назначение.

Таблица 2.1 — Методы, используемые в программе

|  |  |
| --- | --- |
| Метод | Назначение |
| register | Используется для регистрации пользователя |
| login | Используется для входа пользователя |
| getUser | Используется для получения пользователя |
| updateUser | Используется для обновления данных пользователя |
| deleteUser | Используется для удаления пользователя |
| createConnection | Используется для создания соединения |
| deleteConnection | Используется для удаления соединения |
| getConnections | Используется для получения соединения |

Продолжение таблицы 2.1— Методы, используемые в программе

|  |  |
| --- | --- |
| updateConnection | Используется для обновления соединения |
| createGroup | Используется для создания группы |
| deleteGroup | Используется для удаления группы |
| getGroups | Используется для получения групп |
| updateGroup | Используется для обновления группы |

Формы, используемые в основной программе, приведены в таблице 2.2.

Таблица 2.2 — Формы, используемые в основной программе

|  |  |
| --- | --- |
| Форма | Назначение |
| Auth | Экран «Авторизация» с возможностью авторизации |
| Register | Экран «Регистрациии» с возможностью регистрации |
| Main | Экран «Главная» со списком всех групп и подключений |
| Console | Экран «Консоли» для управления операционными системами |
| SFTP | Экран «SFTP» для управления файлами |
| CreateGroup | Экран «Добавления группы» с возможностью добавления группы |
| CreateConnection | Экран «Добавления соединения» с возможностью добавления соединения |

## Разработка структуры данных

Схема базы данных, указанная на рисунке 2.2, содержит информацию о пользователях, продуктах, заказах, корзинах, статистике продаж и сотрудниках.

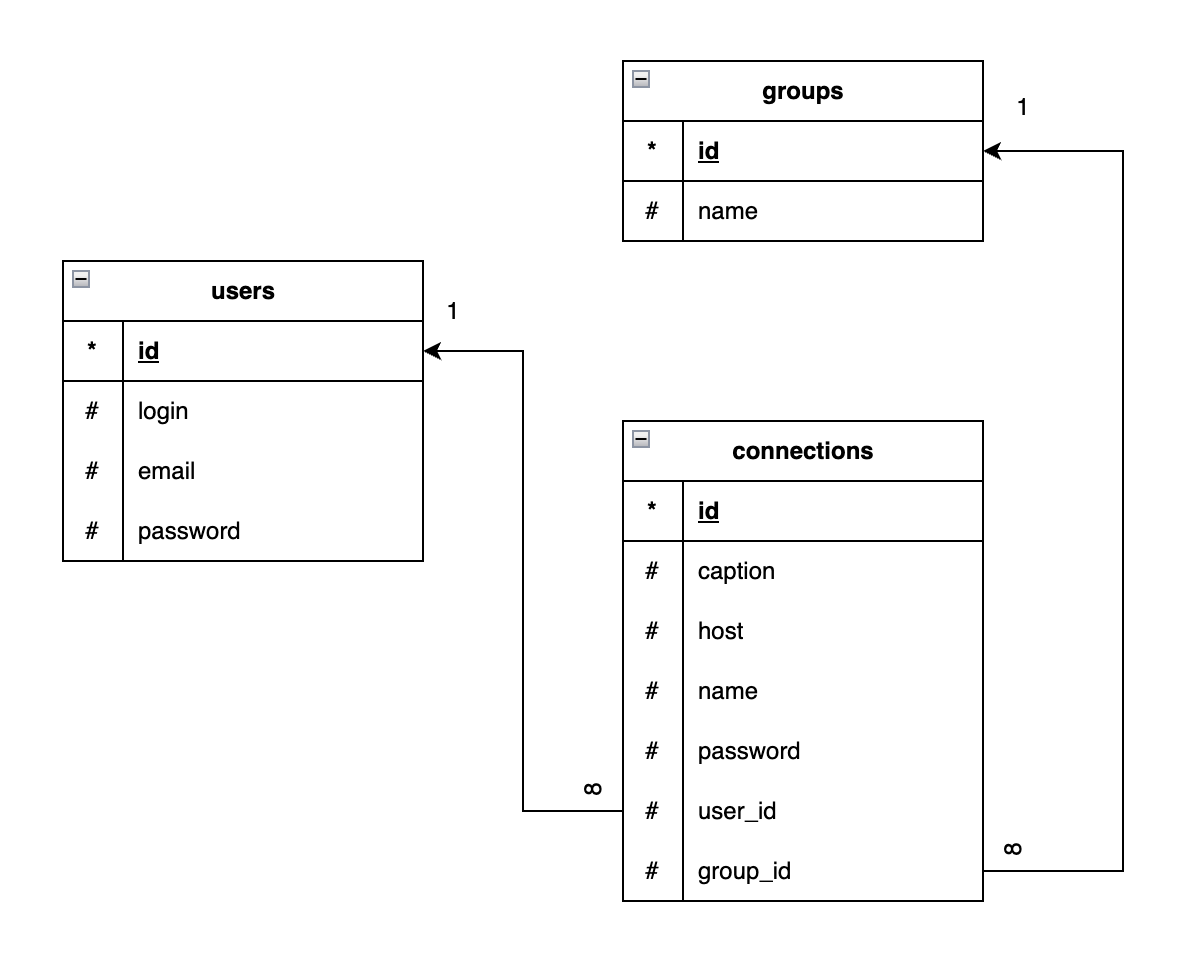


Рисунок 2.2 — Логическая схема базы данных

В таблицах 2.3 - 2.5 представлены структуры таблиц разрабатываемой базы данных it\_support.

Таблица 2.3 — «Пользователи» (Users)

|  |  |  |  |
| --- | --- | --- | --- |
| Идентификатор | Описание столбца | Тип данных | Размер |
| id | Код пользователя | INT | — |
| login | Логин | VARCHAR | 50 |
| email | Почта | VARCHAR | 100 |
| password | Пароль | VARCHAR | 50 |

Таблица 2.4 — «Группы» (Groups)

|  |  |  |  |
| --- | --- | --- | --- |
| Идентификатор | Описание столбца | Тип данных | Размер |
| id | Код группы | INT | — |
| name | Название группы | VARCHAR | 255 |

Таблица 2.5 — «Соединения» (Connections)

|  |  |  |  |
| --- | --- | --- | --- |
| Идентификатор | Описание столбца | Тип данных | Размер |
| id | Код соединения | INT | — |
| caption | Название | UUID | — |
| host | Хост | VARCHAR | 50 |
| login | Логин | VARCHAR | 100 |
| password | Пароль | VARCHAR | 50 |
| user\_id | Код пользователя | INT | — |
| group\_id | Код группы | INT | — |

## Конструирование пользовательского интерфейса

Интерфейс является важной частью любого приложения, и его создание должно основываться на удобстве и простоте использования. Чтобы пользователь мог легко получить нужный результат, необходимо, чтобы интерфейс был интуитивно понятен.

На рисунке 2.3 представлена форма «Авторизация», которая предоставляет пользователю возможность зарегистрироваться или войти в свой аккаунт и перейти к главному меню.



Рисунок 2.3 — Экран «Авторизация»

На рисунке 2.4 представлен экран «Регистрация», где пользователь может вернуться на экран авторизации или заполнить необходимую информацию, чтобы включилась синхронизация между аккаунтами.



Рисунок 2.4 — Экран«Регистрация»

На рисунке 2.5 представлен экран «Главная». С помощью левой панели навигации пользователю предоставляется возможность перейти на экраны: «Главная», «SFTP», увидеть открытые соединения с серверами, а так же историю подключений. В верхней панели вкладок пользователь может произвести, а так же добавить новое соединение и группу.

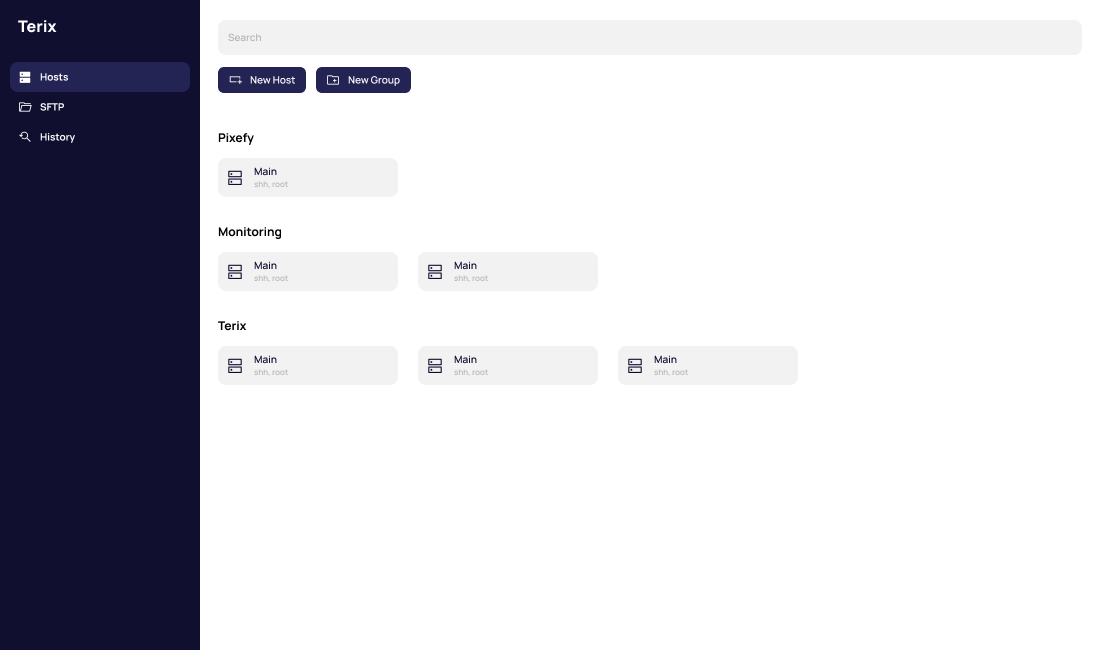


Рисунок 2.5 — Экран «Главная»

На рисунке 2.6 представлен экран «SFTP». На данном экране пользователь может присоединиться к уже созданному соединию для удаленного управления файлами.

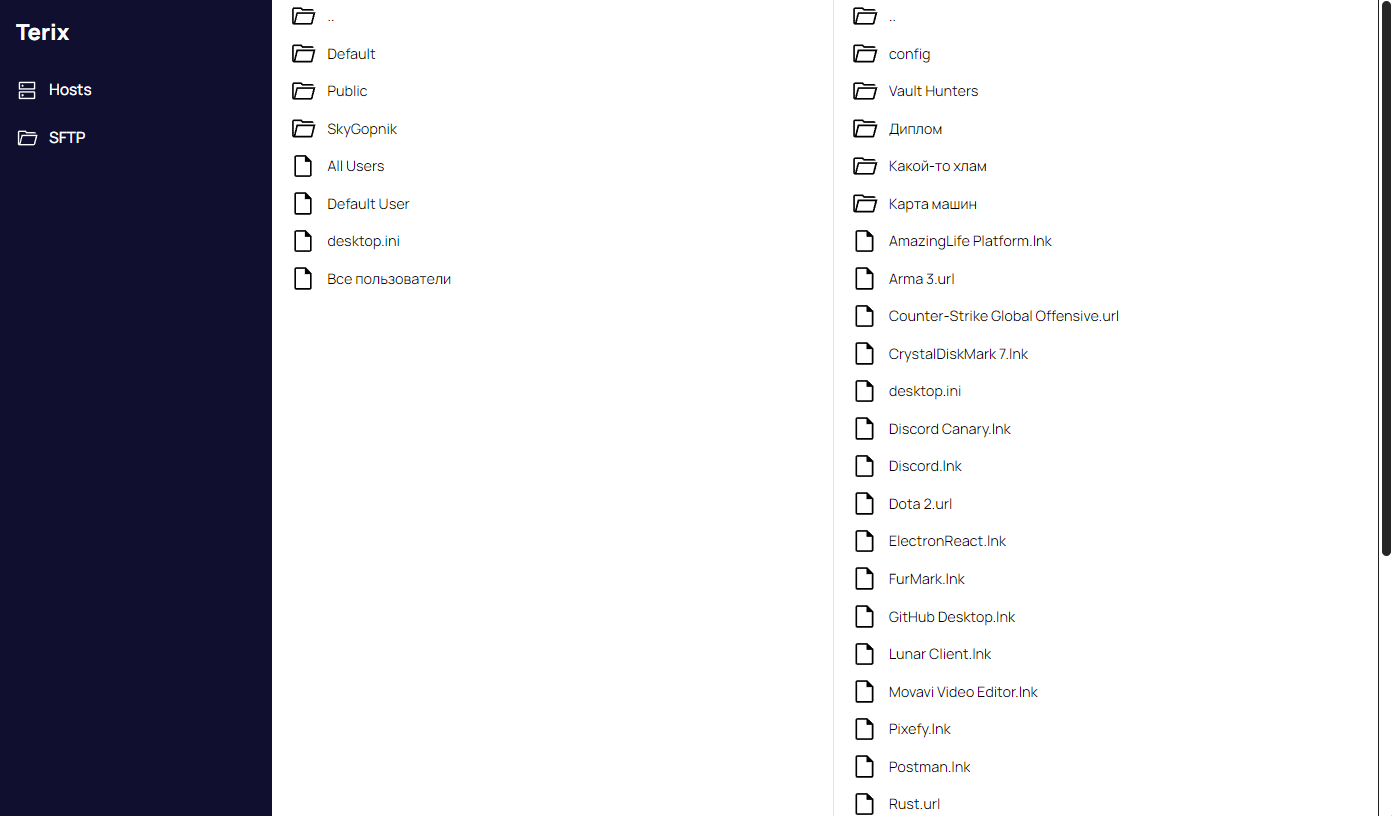


Рисунок 2.6 — Экран «SFTP»

На рисунке 2.7 представлен экран «Консоль». На данном экране пользователь может войти в консоль для удаленного управления операционной системой.

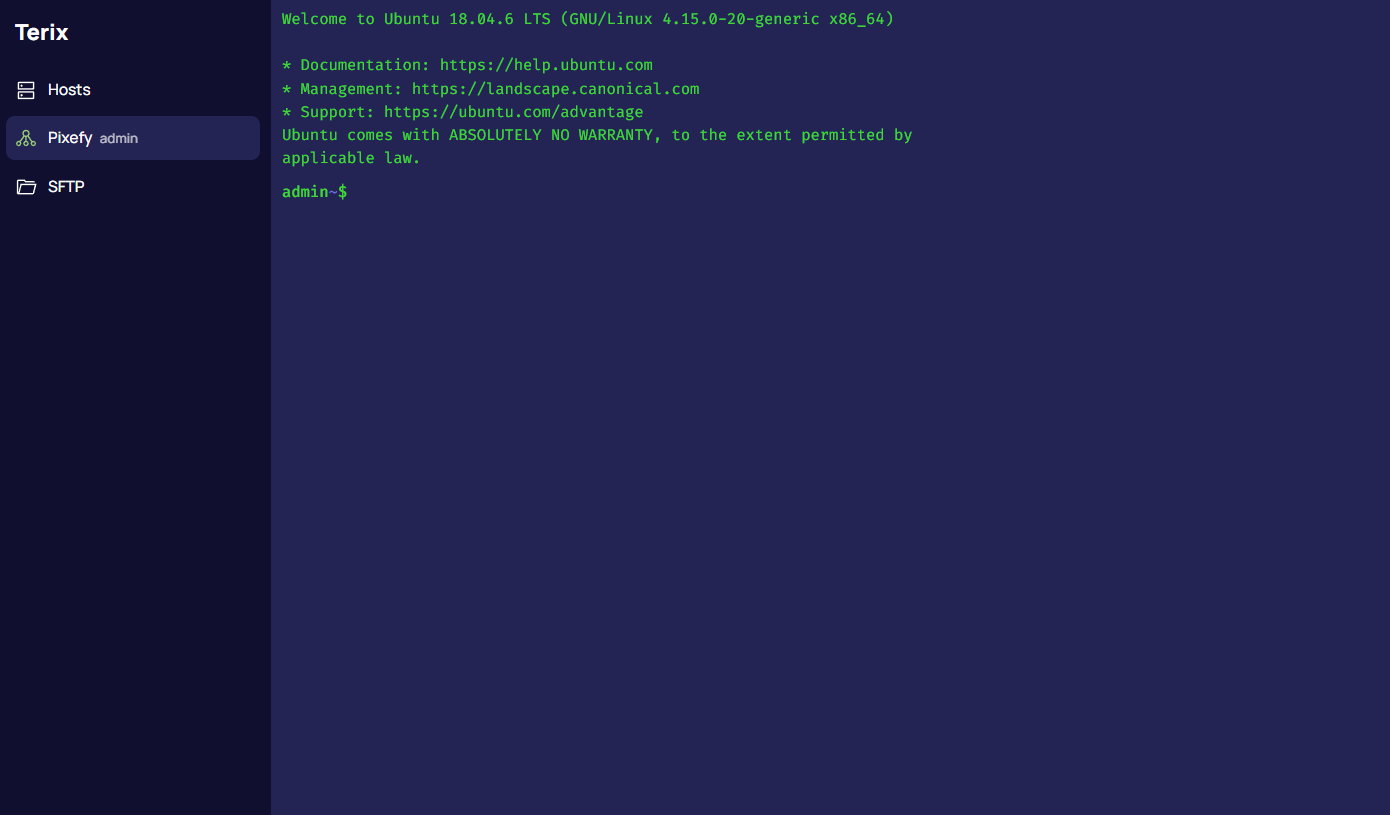
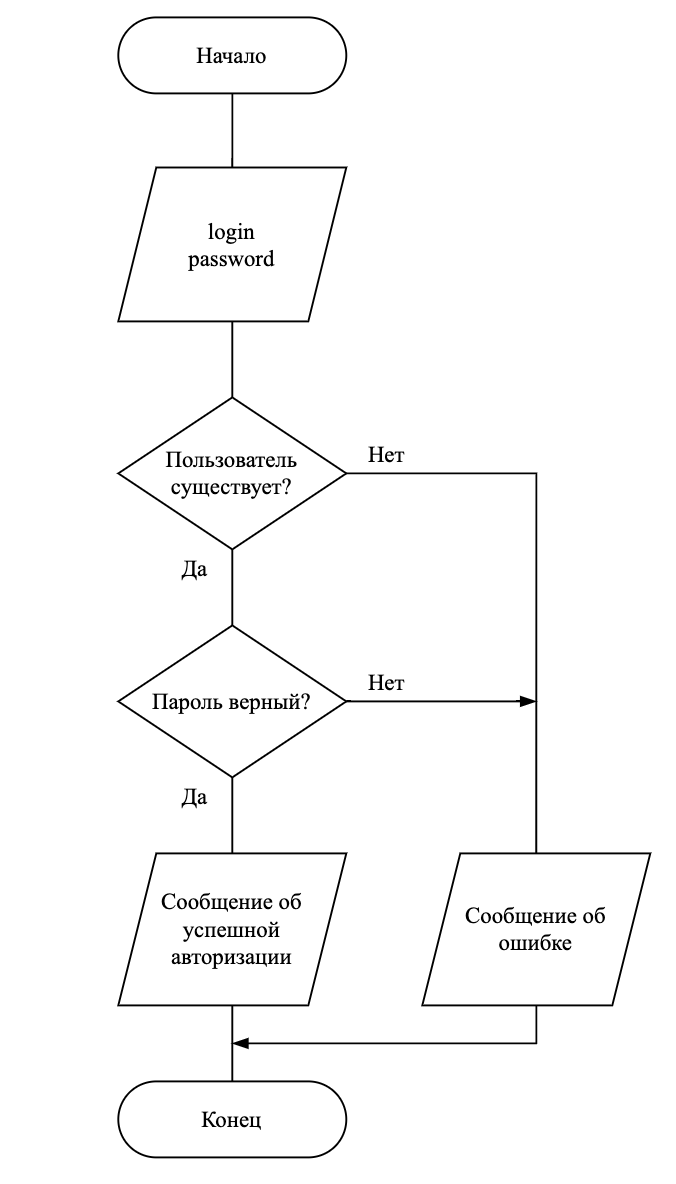


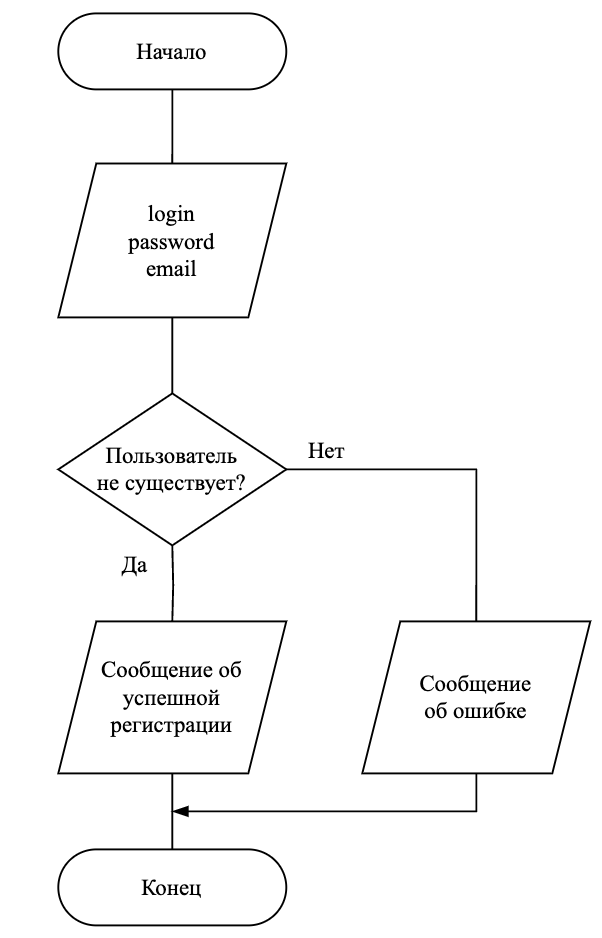
Рисунок 2.7 — Экран «Консоль»

## Схемы алгоритма программы и подпрограмм

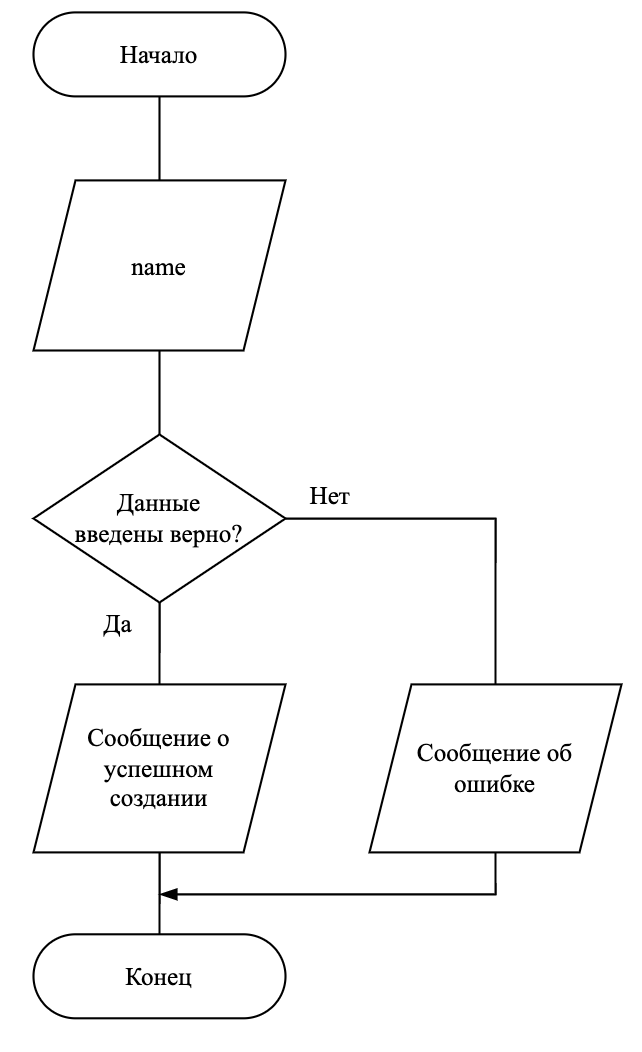
### Схема алгоритма функции authorization



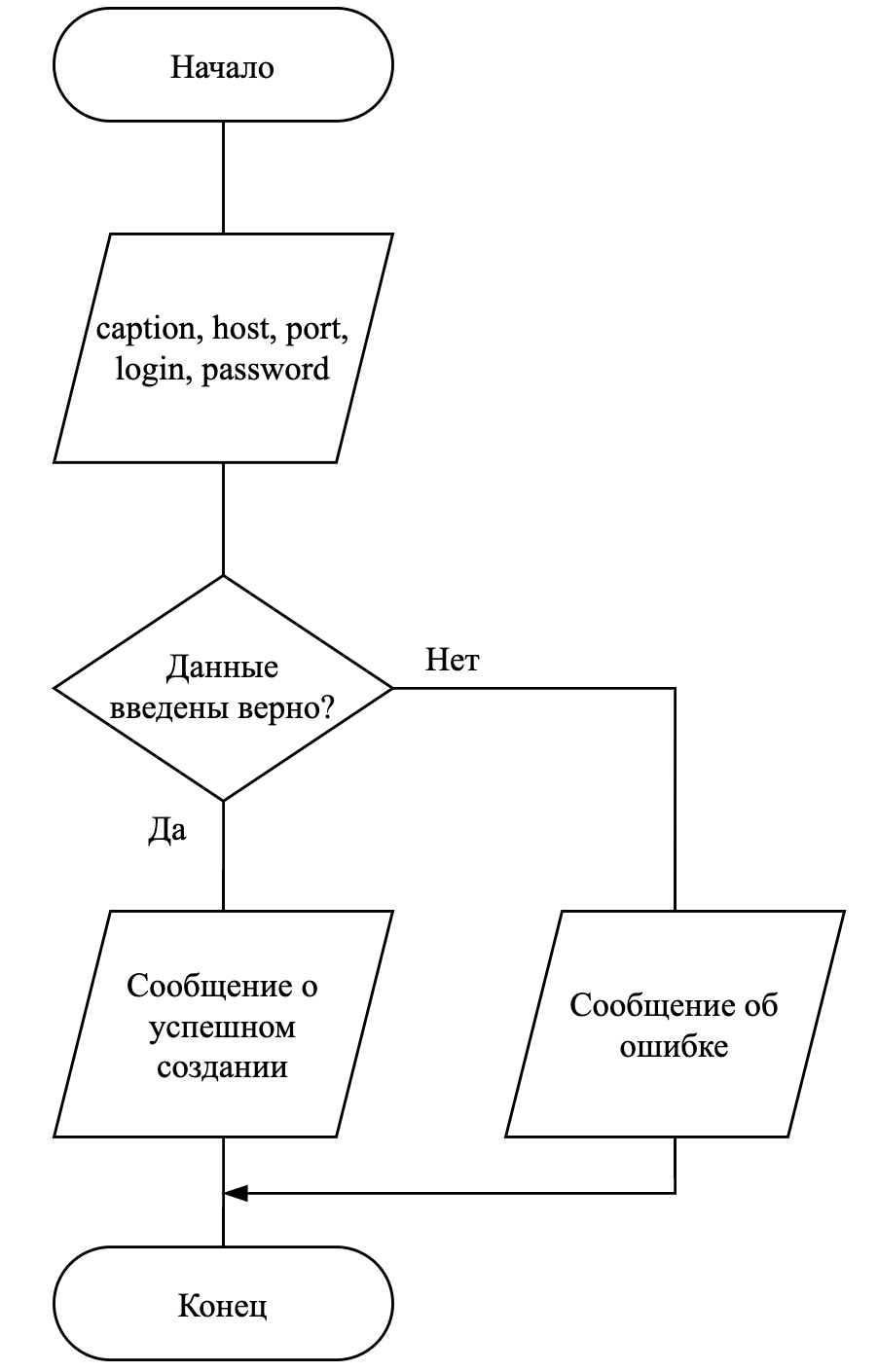
### Схема алгоритма функции registration



### Схема алгоритма функции createGroup



### Схема алгоритма функции createConnection



## Отладка и тестирование программы

### Отладка программы

О Отладка программы - это процесс обнаружения и исправления ошибок в программном коде с целью устранения несоответствий и неправильных действий. Этот этап разработки обычно занимает значительное время и может быть даже более трудоемким, чем написание самой программы. Почти каждая программа содержит ошибки до начала процесса отладки.

Синтаксические ошибки - это ошибки, которые компилятор (транслятор, интерпретатор) обнаруживает при анализе синтаксиса программы и, в некоторой степени, её семантики. Они возникают, когда исходный код не соответствует синтаксису выбранного языка программирования.

Ошибки компоновки - это ошибки, которые возникают при объединении модулей программы компоновщиком (редактором связей). Они происходят, когда связи между модулями и подпрограммами не совпадают или при обращении к ним возникают проблемы.

Ошибки выполнения - это ошибки, которые возникают во время выполнения программы и часто приводят к её преждевременному завершению. Наиболее распространённым примером такой ошибки является деление на ноль.

Ошибки логики - это ошибки, которые препятствуют программе выполнять ожидаемые действия. Код может быть скомпилирован и выполнен без ошибок, но результат операций может быть непредсказуемым и неверным.

Во время отладки данной программы были допущены следующие ошибки:

1. синтаксические ошибки:

* ошибки в названиях переменных
* неправильное имя оператора

Пример ошибки:

d = SELECT \* FROM Users;

Исправленный вариант:

d = "SELECT \* FROM Users";

1. ошибки компоновки:

* неверный список перечисляемых параметров.

Пример ошибки:

let string = sql("INSERT INTO ? VALUES (?)", dto);

Исправленный вариант:

let string = sql("INSERT INTO ? VALUES (?)", "Users", dtos);

1. ошибки выполнения:

* несовпадение полученных результатов с ожидаемыми;
* неверное использование неполного типа;
* неправильно организованный цикл с параметром.

Пример ошибки:

// Несовпадение полученных результатов с ожидаемыми

function addNumbers(a: number, b: number): number {

return a - b; // Неправильно вычитаем вместо сложения

}

const result = addNumbers(2, 3);

console.log(result); // Ожидалось 5, но получили -1

Исправленный вариант:

function addNumbers(a: number, b: number): number {

return a + b; // Исправлено на сложение

}

const result = addNumbers(2, 3);

console.log(result); // Выводится 5, как ожидалось

Все ошибки были исправлены. Для того, чтобы убедиться в правильности работы программы, необходимо провести её тестирование.

### Тестирование программы

Тестирование программы — это процесс проверки программного кода на соответствие заданным требованиям и выявление ошибок в работе программы. Тестирование является неотъемлемой частью процесса разработки программного обеспечения и проводится на всех этапах разработки.

Тестирование программы является необходимой частью процесса разработки программного обеспечения и позволяет выявить ошибки в работе программы на ранних этапах разработки. Проведение качественного тестирования помогает создать надежную и стабильную программную систему, что в свою очередь повышает удовлетворенность пользователей и уменьшает затраты на поддержку программы.

### Тестирование программы в нормальных условиях

Тестирование в нормальных условиях было проведено со всеми формами. Были протестированы переходы от экрана к экрану, предоставление возможности входа в систему и регистрации, автоматическое обновление списков. Тестирование процедуры добавления, удаления и обновления соединения представлено на рисунках 2.8, 2.9, 2.10.

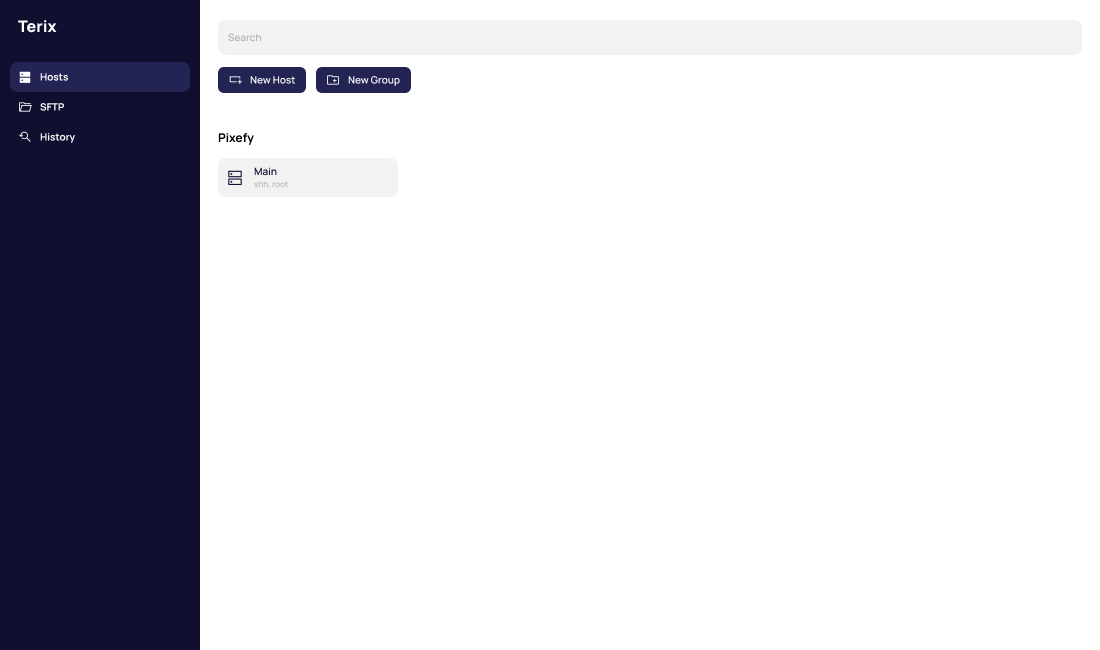


Рисунок 2.8 — Добавление нового соединения

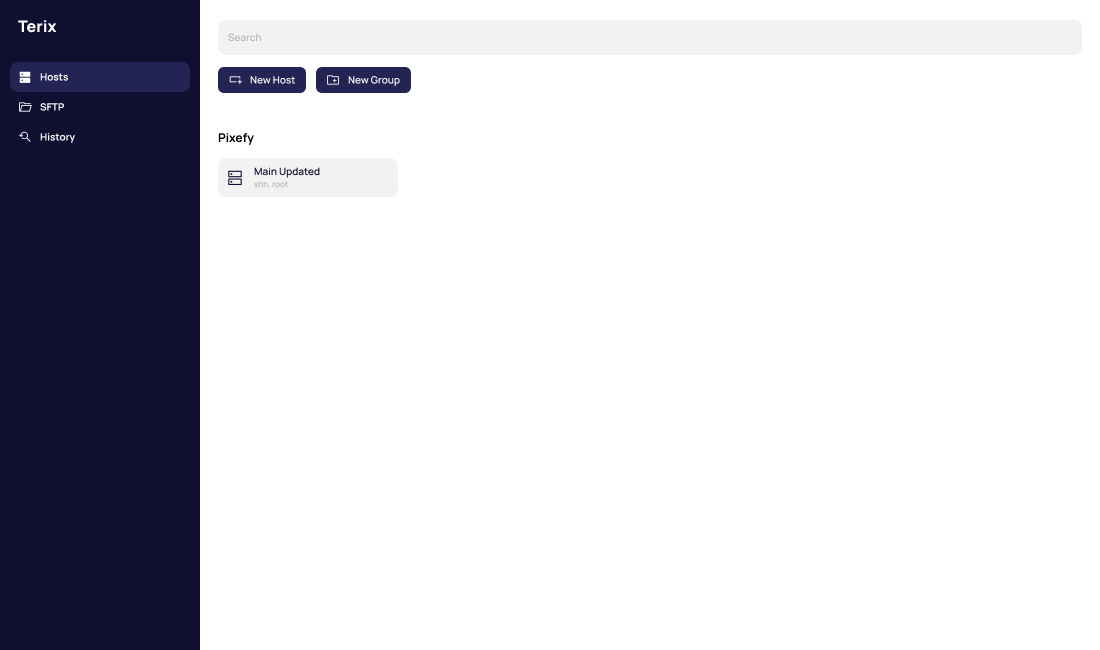


Рисунок 2.9 — Обновление соединения

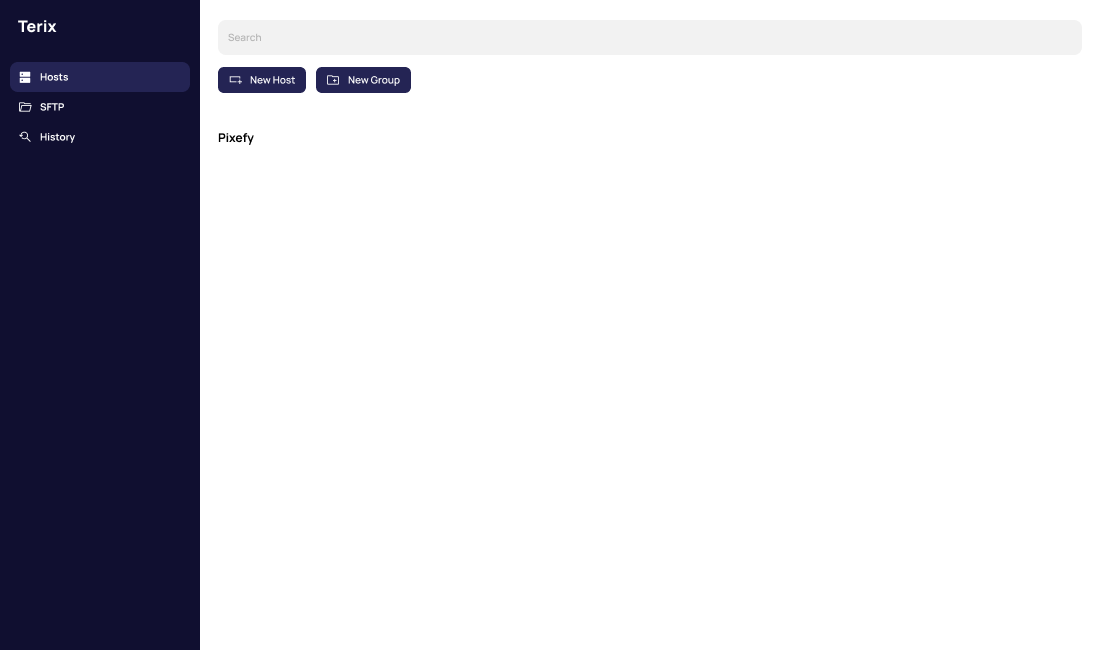


Рисунок 2.10 — Удаление соединения

Тестирование процедуры проверки занятости логина при регистрации новых пользователей на рисунке 2.11.

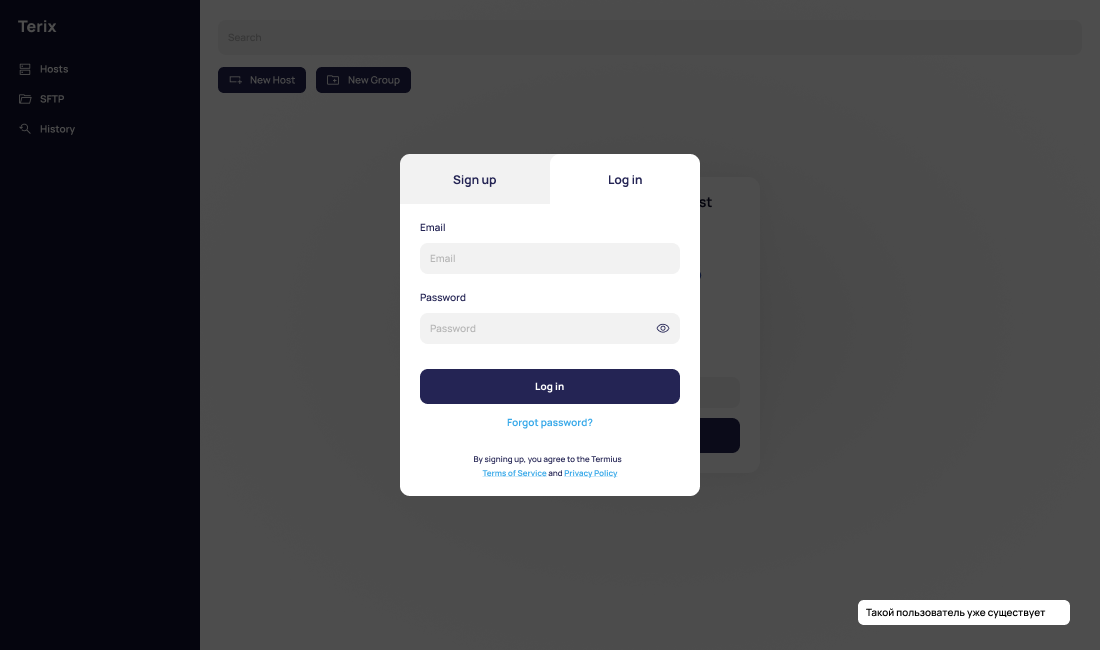


Рисунок 2.11 — Сообщение о том что пользователь уже существует

После тестирования в нормальных условиях можно сделать вывод что программа работает правильно.

### Тестирование программы в экстремальных условиях

На рисунках 2.12 и 2.13 видно, что при регистрации если пользователь не введет данные, то программа выдаст ошибку «обязательно для заполнения.

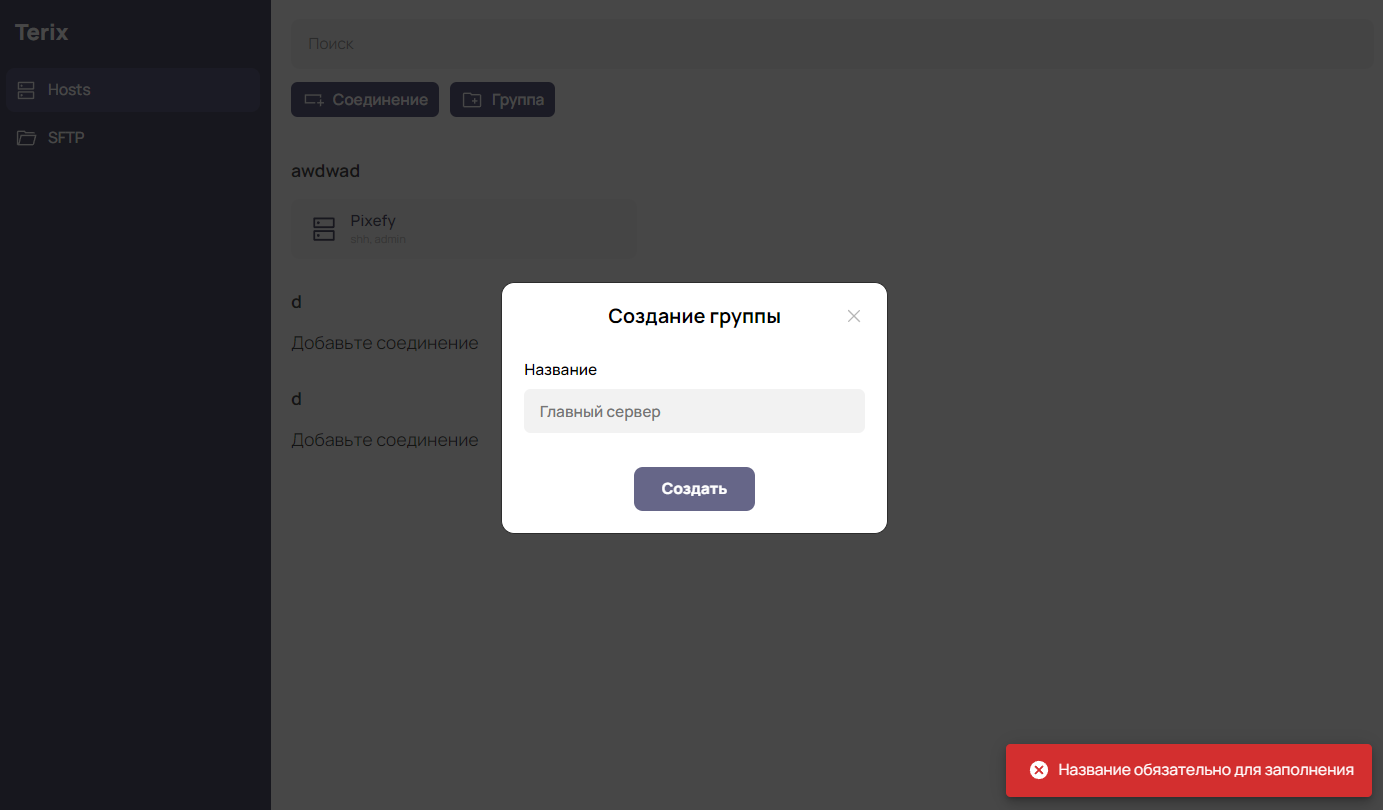


Рисунок 2.12 — Сообщение «Обязательно для заполнения»

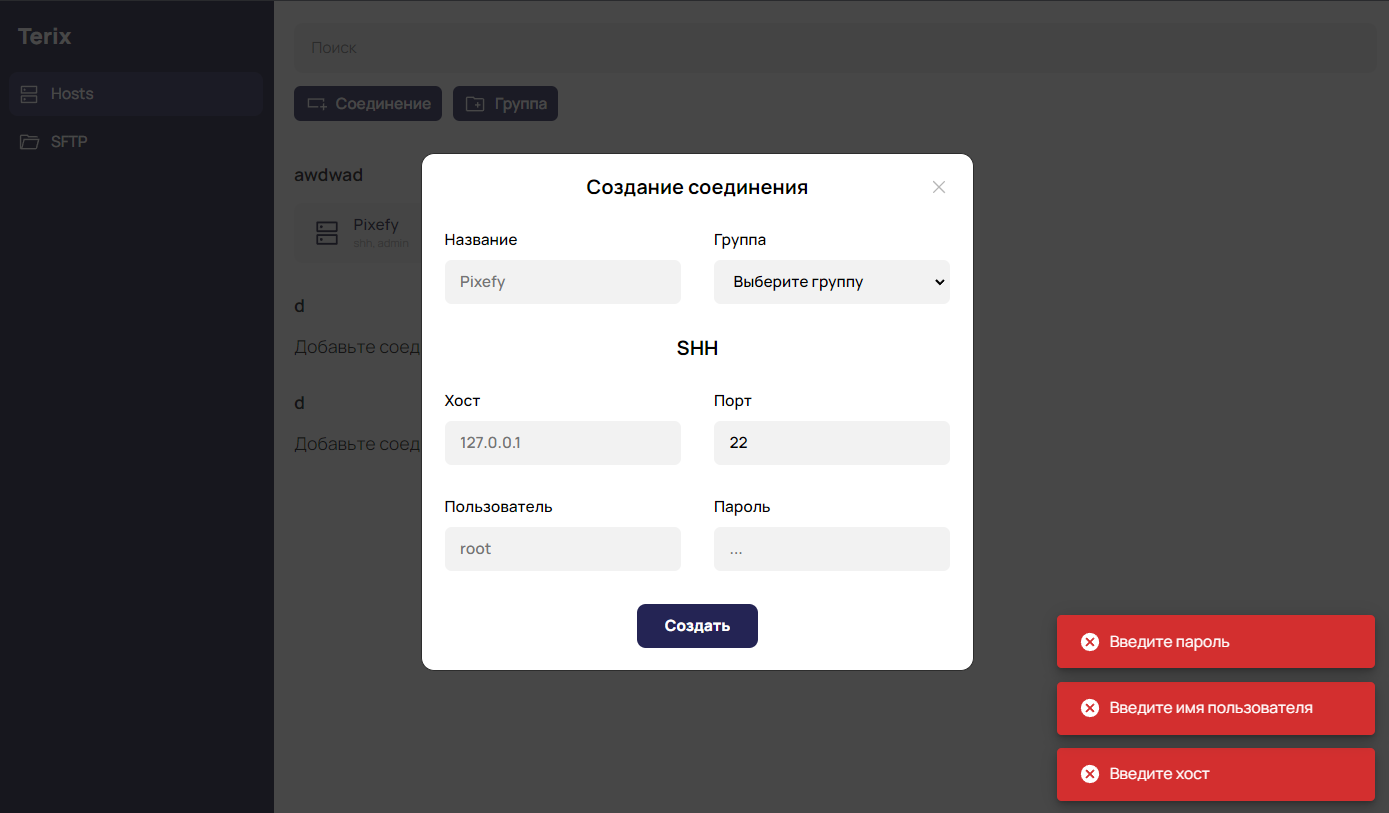


Рисунок 2.13 — Сообщение «Обязательно для заполнения»

После тестирования в экстремальных условиях можно сделать вывод что программа работает правильно.

### Тестирование программы в исключительных ситуациях

На рисуне 2.14 видно, что при создании группы если пользователь введет уже название группы, которое уже существует, то программа выдаст сообщение об ошибки. Если пользователь введет некоректные данные для подключения, то приложение так же выдаст ошибку (рисунок 2.15).

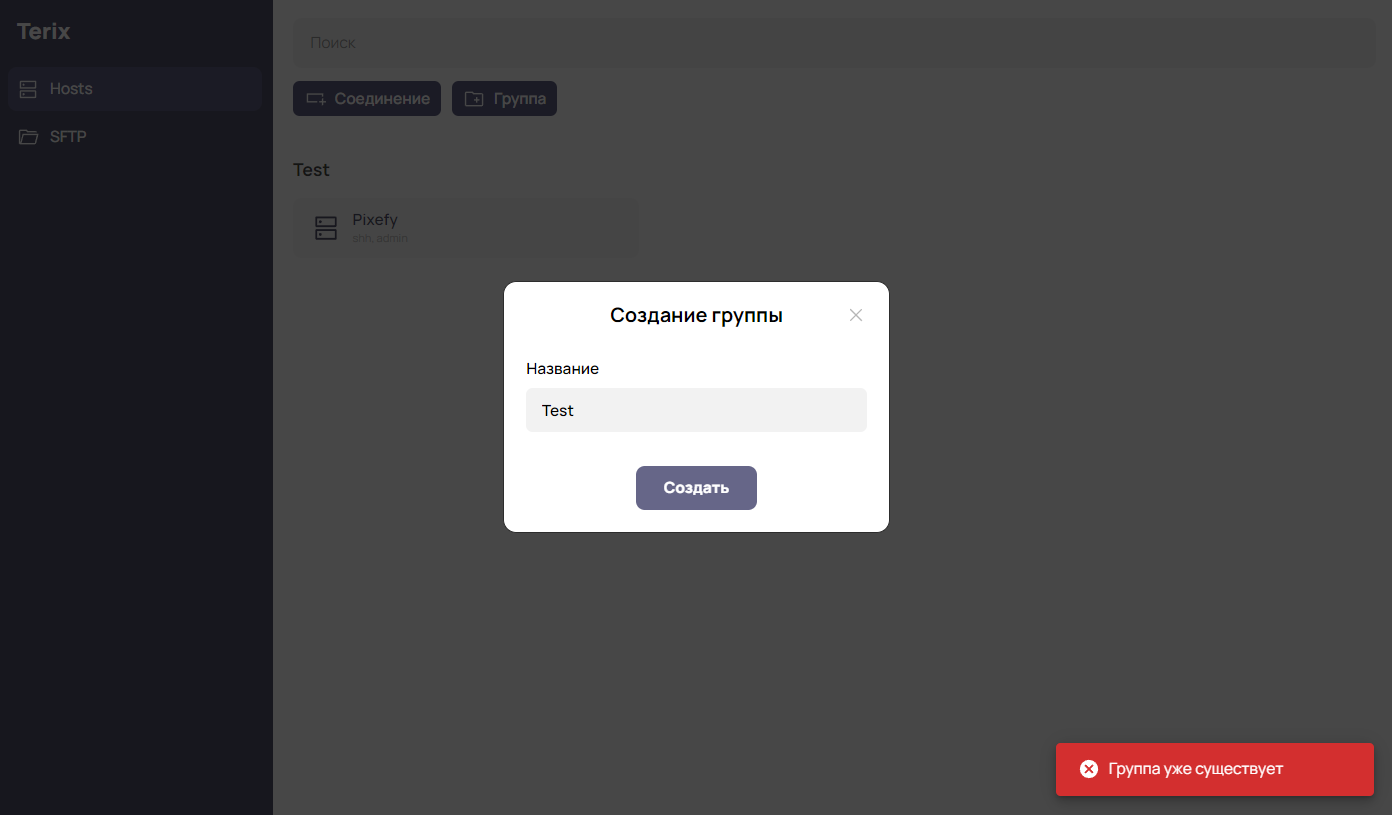


Рисунок 2.14 — Сообщение «Группа уже существует»

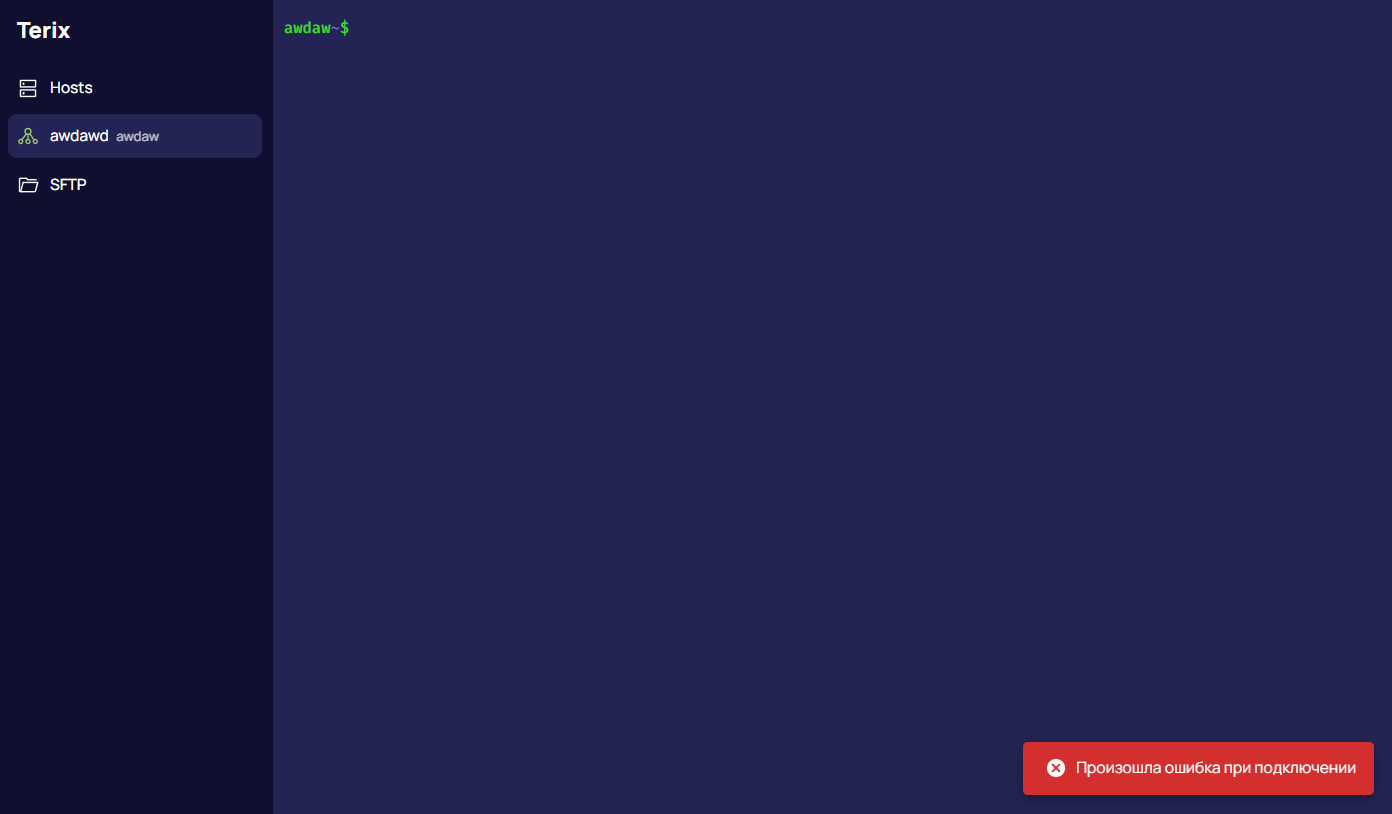


Рисунок 2.15 — Сообщение «Произошла ошибка при подключении»

После тестирования в исключительных ситуациях можно сделать вывод что программа работает правильно.

Тестирование в нормальных, экстремальных условиях и исключительных ситуациях показало отсутствие ошибок, можно сделать вывод, что программа работает правильно.

### Сообщения

В таблице 2.6 представлены все сообщения с их описанием.

Таблица 2.6 — Сообщения

|  |  |  |
| --- | --- | --- |
| Текст сообщения | Описание | Действия для исправления |
| Произоша ошибка при подключении | Данные для подключения к удаленной операционной системе неверны | Исправить данные для подключения |
| Группа уже существует | Введённое при создании группы название уже существует | Ввести другое название |
| Поле обязательно для заполнения | Необходимые поля формы не заполнены | Заполнить все необходимые поля |

## Руководство пользователя

### Назначение программы

Данная программа предназначена для системы удаленного управления файлами и операционными системами. Она позволяет зарегистрированным пользователям авторизоваться в системе и получить доступ к функциям удаленного управления файлами и операционными системами.

Система обеспечивает возможность просмотра, редактирования, перемещения и удаления файлов на удаленных компьютерах или серверах. Создавать группы для удобного хранения подключений, а так же получать доступ к созданным подключениям на другом устройстве, за счет синхронизации данных между ними.

### Условия работы программы

Минимальные системные требования:

* + объем оперативного запоминающего устройства – 2 Гб и более;
  + объем постоянного запоминающего устройства – 16 Гб и более;
  + система android 6 и выше;

Требования к программному обеспечению:

* + создание подключения;
  + создание группы;
  + восстановление пароля;
  + синхронизация данных;
  + управление опреационной системой;
  + управление файлами;
  + регистрация новых пользователей;
  + возможность входа в систему.

### Входные данные

Ограничения на входные данные представлены в таблице 2.7

Таблица 2.7 — Ограничения на входные данные

|  |  |  |
| --- | --- | --- |
| Таблица | Поле | Количество символов |
| Пользователи | Логин | 50 |
| Пароль | 50 |
| Почта | 100 |
| Соединения | Название | 80 |
| Хост | 100 |
| Логин | 50 |
| Пароль | 50 |
| Группа | Название | 80 |

### Выполнение программы

Для использования программы необходимо выполнить следующие действия:

1. Для запуска программы, необходимо нажать на рабочем столе на приложение «Terix».
2. После запуска откроется экран «Главная» (рисунок 2.16), в котором можно перейти на экран «SFTP» и войти в приложение для синхронизации данных.

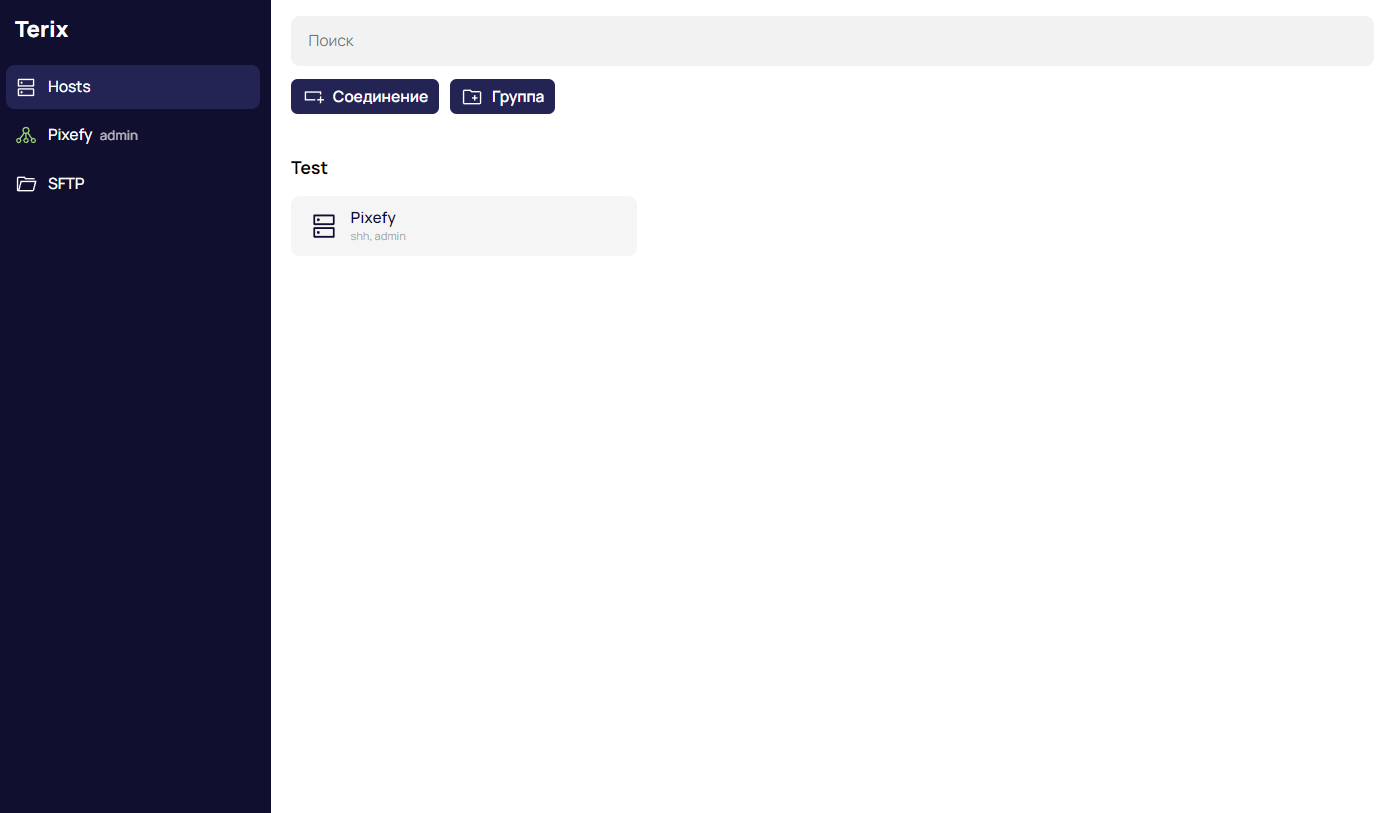


Рисунок 2.16 — Экран «Главная»

1. На экране «Главная» можно добавить новую группу или соединения, а так же изменить или удалить уже существующие (рисунок 2.17) и (рисунок 2.18).

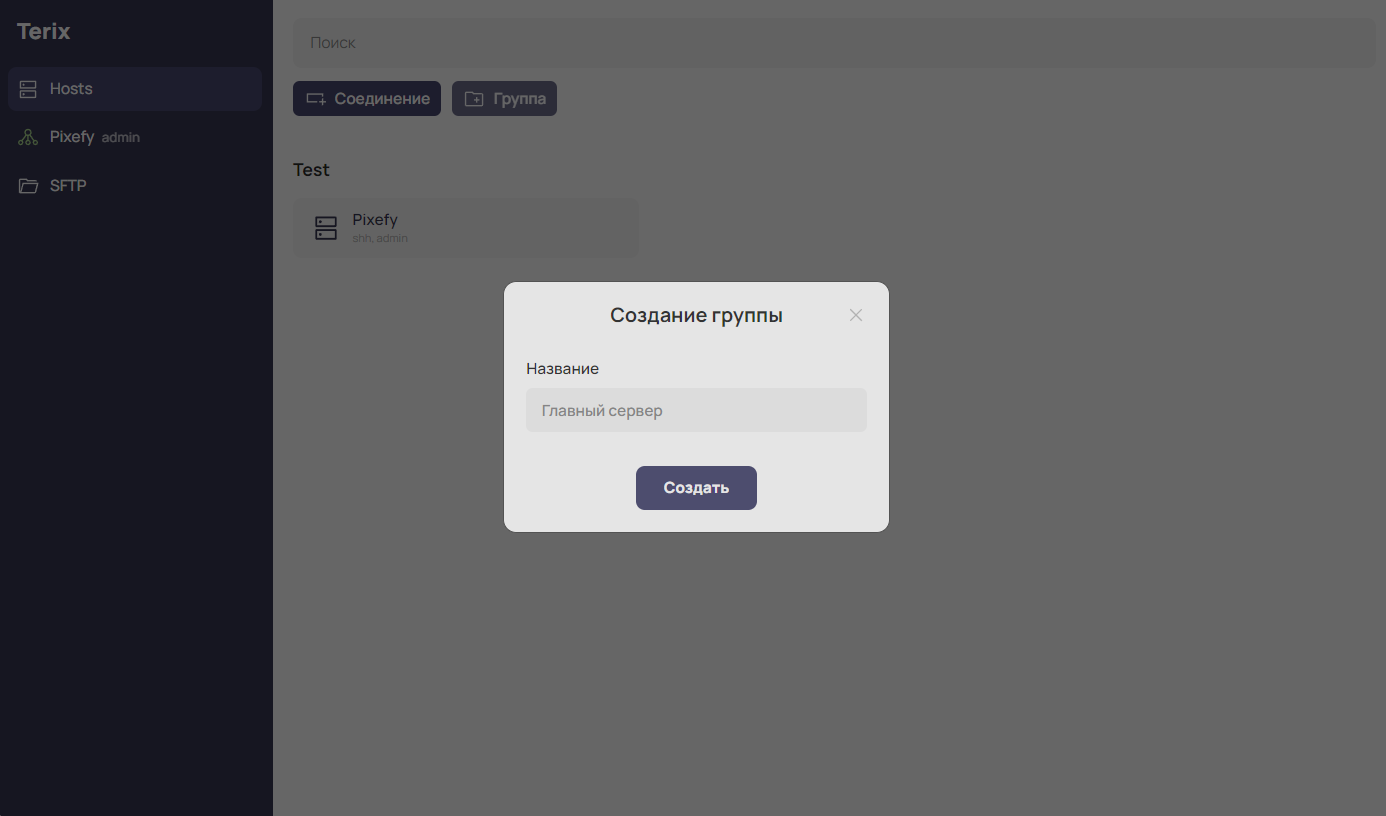


Рисунок 2.17 —Экран «Создание группы»

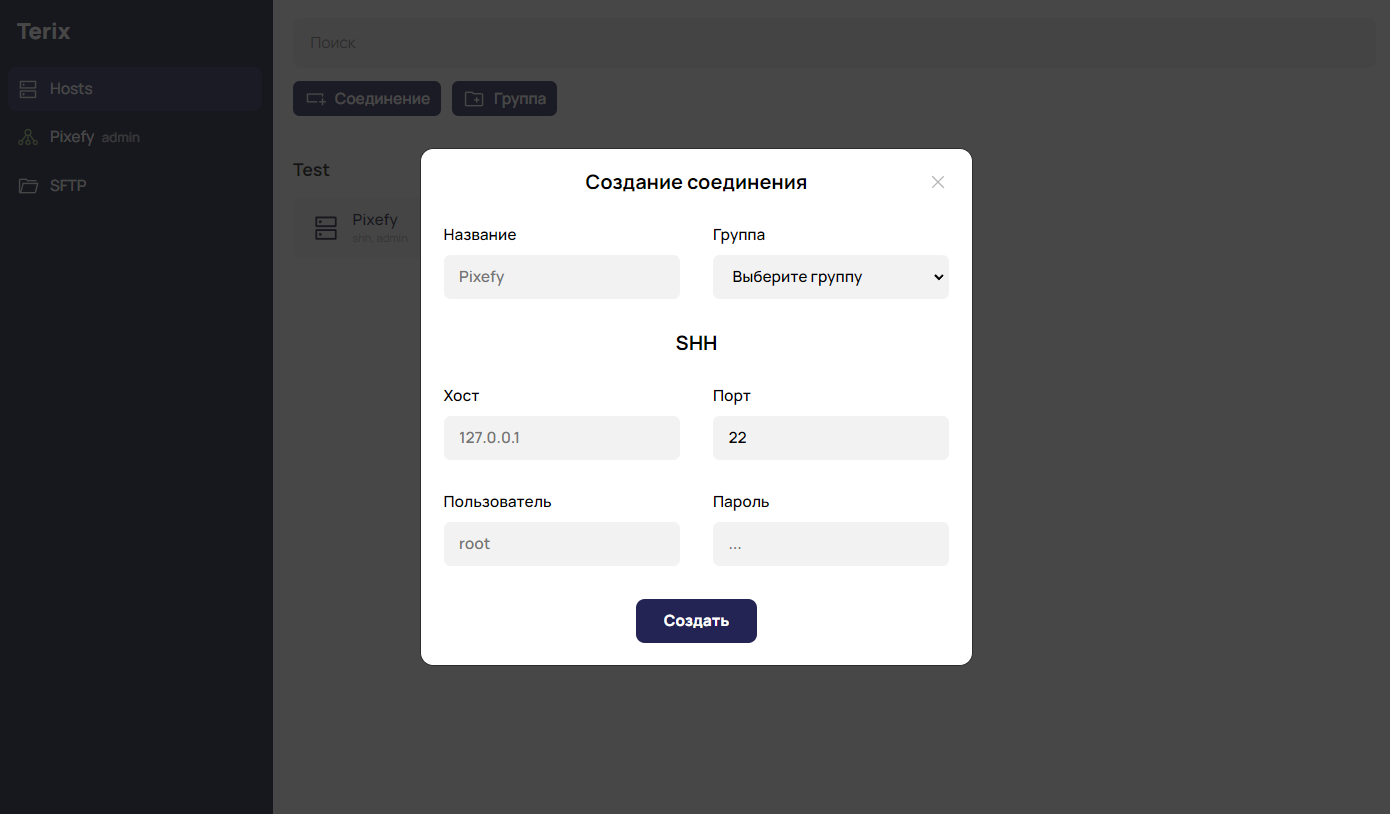


Рисунок 2.18 —Экран «Создание подключения»

1. Так же на экране «Главный» (рисунок 2.16) можно войти в аккаунт (рисунок 2.19) или создать ноый (рисунок 2.20), для синхронизации данных между устройствами.



Рисунок 2.19 — Экран «Авторизация»



Рисунок 2.20 — Экран«Регистрация»

1. На экране «Главная» (рисунок 2.16) можно перейти на экран «SFTP» (рисунок 2.21) он используется для удаленного управления файлами и при нажатии на соедиения перейти на экран «Консоль» (рисунок 2.22), она используется для удаленного управления операционной системой.

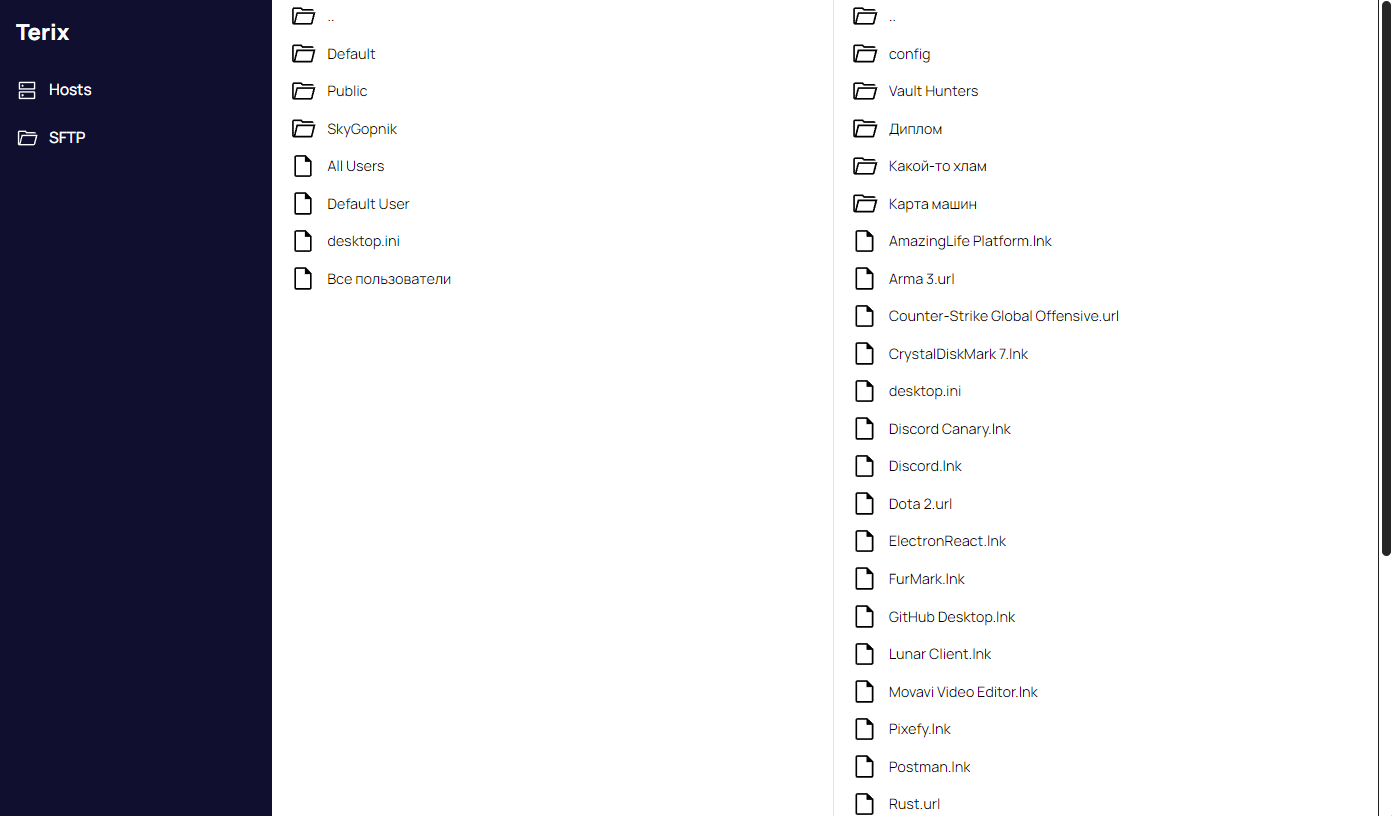


Рисунок 2.21 — Экран «Главная»

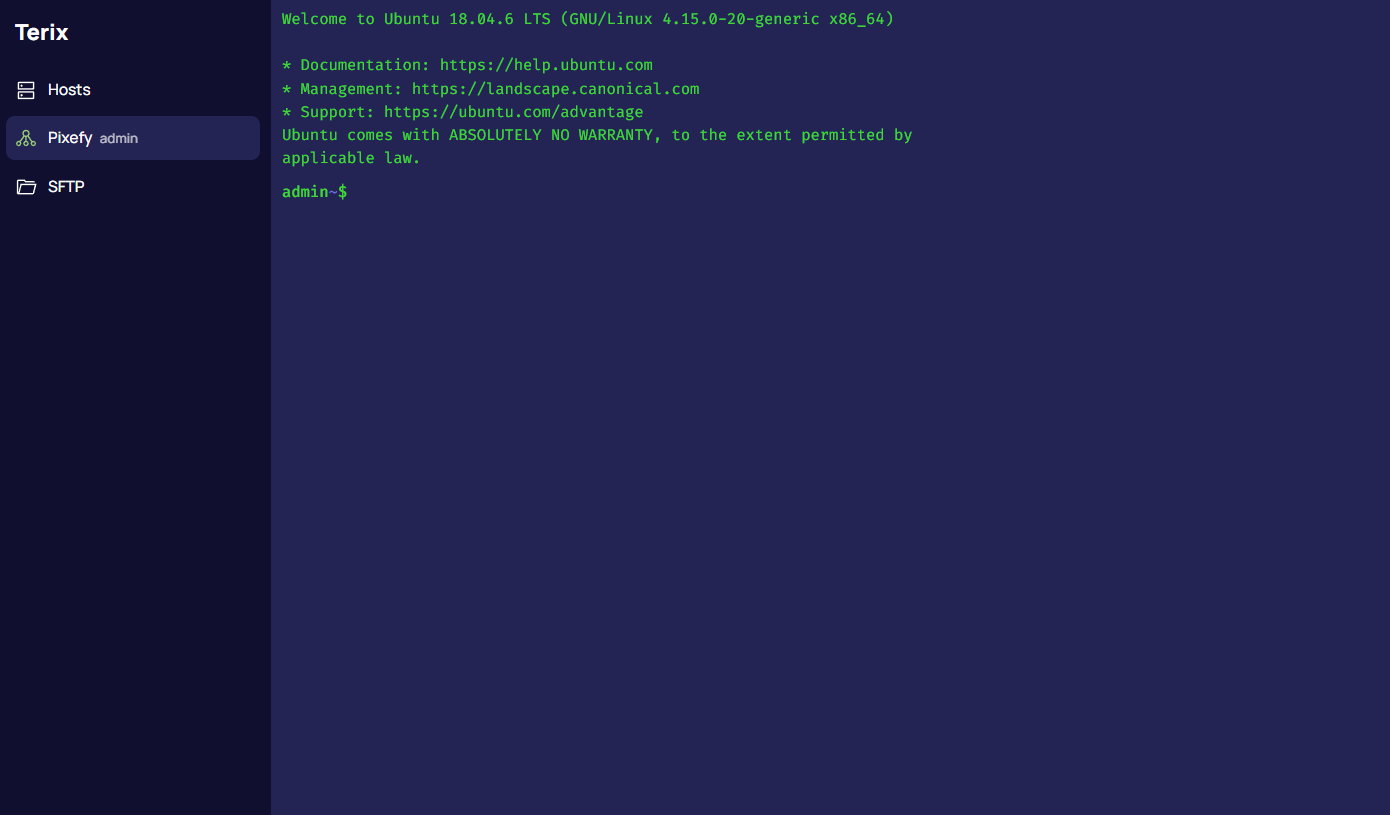


Рисунок 2.22 — Экран «Консоль»

### Вывод по разделу

В данном разделе была разработана архитектура программной системы, определены основные компоненты и их взаимодействие. Затем была выполнена разработка структуры данных, которая позволила организовать хранение и обработку информации в системе.

Одном из важных этапов стало конструирование пользовательского интерфейса. Был разработали интерфейс, который учитывает потребности пользователей и обеспечивает удобство взаимодействия с системой.

Для функций программы были разработаны схемы алгоритмов, что позволило четко определить последовательность действий и логику работы программы.

Так же были выполнены процессы отладки и тестирования программы. Проведена отладка программы, исправлены выявленные ошибки и проверена ее работа в различных условиях. Тестирование программы включало проверку ее работы в нормальных условиях, экстремальных ситуациях и исключительных случаях. Были предусмотрены соответствующие сообщения для информирования пользователей о результатах выполнения программы.

В заключительном пункте раздела было составлено руководство пользователя. Определены назначение программы, условия работы, входные данные и последовательность выполнения программы. Это позволит пользователям эффективно использовать разработанную программную систему.

# Охрана труда

### Техника безопасности при работе на персональном компьютере

Техника безопасности при работе с компьютером на предприятии предусматривает наличие общедоступной инструкции, в которой указаны обязательные требования к обустройству рабочего места и процессу использования техники. При работе с компьютером существуют определенные риски для здоровья и безопасности. Чтобы предотвратить возможные проблемы, следует принимать меры по обеспечению безопасности при работе на компьютере.

Рабочее место должно быть настроено таким образом, чтобы обеспечить комфортное положение тела и минимизировать риск возникновения напряжения в мышцах и суставах:

* высоту рабочей поверхности стола отрегулировать в пределах 680 - 800 мм, при отсутствии регулировки высота рабочей поверхности стола должна составлять 725 мм;
* рабочий стол должен иметь пространство для ног высотой не менее 600 мм, шириной не менее 500 мм, глубиной на уровне колен - не менее 450 мм и для вытянутых ног не менее 650 мм;
* клавиатуру расположить на поверхности стола на расстоянии 100 - 300 мм от края, обращенного к пользователю, или на специальной регулируемой по высоте рабочей, отделенной от основной, столешницы.
* уровень глаз при вертикально расположенном экране должен приходится на центр или 2/3 высоты экрана, линия взора должна быть перпендикулярна центру экрана и оптимальное ее отклонение от перпендикуляра, проходящего через центр экрана в вертикальной плоскости, не должно превышать ± 5°, допустимое - ± 10°.

Длительное время, проведенное за компьютером, может приводить к усталости глаз и напряжению. Нужно выполнять простые упражнения для глаз и расслабляющие упражнения для мышц шеи и спины, чтобы снять напряжение и улучшить кровообращение.

Работник при работе с персональным компьютером обязан:

* осмотреть и привести в порядок рабочее место;
* отрегулировать освещенность на рабочем месте, убедиться в достаточности освещенности, отсутствии отражений на экране;
* проверить правильность подключения оборудования к электросети;
* проверить исправность токопроводящих проводов и отсутствие оголенных участков проводов;
* убедиться в наличии защитного заземления;
* протереть салфеткой поверхность экрана и защитного фильтра.

### Требования к помещению

Помещения ВЦ, их размеры (площадь, объем) должны в первую очередь соответствовать количеству работающих и размещаемому в них комплекту технических средств. Должны быть предусмотрены соответствующие параметры температуры, освещения, чистоты воздуха, производственных шумов и др.

Помещение должно быть должным образом освещено, чтобы сотрудники могли выполнять свои задачи без напряжения глаз. Естественное освещение предпочтительно, но если его недостаточно, следует обеспечить достаточное и равномерное искусственное освещение.

Снижение шума, создаваемого на рабочих местах внутренними источниками, а также шума, проникающего извне, является очень важной задачей. Снижение шума в источнике излучения можно обеспечить применением упругих прокладок между основанием машины, прибора и опорной поверхностью.

Рабочее пространство должно быть достаточно просторным для комфортного выполнения задач и свободного передвижения сотрудников. Мебель, включая столы, стулья и полки, должна быть эргономичной, чтобы предотвратить возможные проблемы с позвоночником, мышцами и суставами.

В помещении должны быть предусмотрены пожарные выходы и пути эвакуации, которые должны быть ясно обозначены. Помещение должно быть чистым и свободным от препятствий, таких как мусор, провода или скользкие поверхности.

Помещение, в котором работают сотрудники, должно быть хорошо проветриваемым, чтобы обеспечить свежий воздух и предотвратить скопление вредных веществ, таких как пыль или химические испарения.

Кондиционирование воздуха должно обеспечивать автоматическое поддержание параметров микроклимата в необходимых пределах в течение всех сезонов года, очистку воздуха от пыли и вредных веществ, создание небольшого избыточного давления в чистых помещениях для исключения поступления неочищенного воздуха. Температура воздуха, подаваемого в помещения ВЦ, должна быть не ниже 19 градусов.

### Требования к пожарной безопасности

Пожарная безопасность предусматривает обеспечение безопасности людей и сохранения материальных ценностей предприятия на всех стадиях его жизненного цикла (научная разработка, проектирование, строительство и эксплуатация).

Опасными факторами пожара, воздействующими на людей и материальные ценности, помимо открытого пламени, повышенной температуры корпусов оборудования и окружающей среды, являются также токсические продукты горения и термического разложения, пониженная концентрация кислорода в воздухе рабочей зоны и вызванные описанными факторами их вторичные проявления.

В помещении должны быть понятные и доступные пути эвакуации, которые должны быть освещены и обозначены соответствующими знаками. Сотрудники должны быть обучены процедурам эвакуации и знать, как правильно действовать в случае пожара.

Организация должна установить и регулярно проверять пожарную сигнализацию, автоматические пожаротушители и другие системы пожаротушения, чтобы обеспечить своевременное обнаружение пожара и его быстрое тушение.

Рабочее пространство должно быть свободно от потенциальных источников возгорания, таких как легковоспламеняющиеся жидкости или материалы. Сотрудникам следует предоставлять необходимое оборудование, такое как огнетушители, и обучать их его правильному использованию.

# Техником-экономическая часть

### Технико-экономические расчеты экономической целесообразности программы

Задача состоит в разработке системы программной системы удаленного управления файлами и операционными системами.

Программа программа разработана с целью улучшения эффективности и качества обслуживания сотрудников организации, используя графический интерфейс. В ней встроены подсказки и механизмы, предотвращающие возможные ошибки.

Для написания программы использовался язык программирования TypeScript.

Программа должна быть разработана для компьютеров со следующей конфигурацией:

* Процессор с тактовой частотой не менее 2,0 ГГц;
* Объем оперативного запоминающего устройства – 4 Гб и более;
* Объем постоянного запоминающего устройства – 16 Гб и более.

Для того, чтобы говорить об экономической эффективности разработанного проекта, необходимо рассчитать его себестоимость.

Себестоимость – это затраты предприятий, связанные с производством и реализацией продукции. Калькуляция себестоимости выпускаемой продукции включает в себя затраты на сырье и материалы, оплату труда, страховые взносы, амортизацию основных фондов и прочие расходы. Расчет себестоимости показан на формуле (4.1).

(4.1)

С = М + КИ(ПФ) + Тр-з + Зосн + Здоп + Св + Нцех + Нзав + Впр,

где С – себестоимость, руб.;

М – стоимость материалов, основы для полуфабрикатов, разделенных на вспомогательные (не входящие в состав готовой продукции) и основные, руб.;

КИ(ПФ) – стоимость комплектующих материалов, готовых элементов конечного продукта, используемых для сборки, ремонта и упаковки, руб.;

Тр-з – транспортно-заготовительные расходы, связанные с заготовкой, доставкой и хранением материальных ценностей (товаров, сырья, материалов, инструментов), руб.;

Зосн – заработная плата основная основных производственных рабочих, определяется в зависимости от трудоемкости, сложности и действующей формы оплаты труда, руб.;

Здоп – заработная плата дополнительная основных производственных рабочих, расходуется на оплату отпусков, отдельных видов премирования, пособия для уходящим в армию, выплат государственных и общественных обязанностей, руб.;

Св – страховые взносы основных производственных рабочих, расходуется на пенсии и медицину, руб.;

Нцех – накладные расходы цеховые, расходуются на амортизацию оборудования цеха, заработную плату всех работников цеха (исключая основных производственных рабочих), ЖКХ, текущий и капитальный ремонт помещений цеха, руб.;

Нзав – накладные расходы завода, расходуются на амортизацию общезаводского оборудования, заработную плату всех работников завода (исключая работников цехов), ЖКХ, текущий и капитальный ремонт здания завода, аренду, отчисления вышестоящим организациям, руб.;

Впр – внепроизводственные расходы, расходуются рекламу, упаковку, командировки, презентации новых разработок, руб.

Ввиду специфики работы программиста и отсутствия материальной части формула приобретает вид (4.2).

(4.2)

С = Зосн + Здоп + Св + Нцех + Нзав + Впр,

где Зосн – заработная плата основная программистов, определяется в зависимости от трудоемкости задачи и оклада, руб.;

Здоп – заработная плата дополнительная программистов, расходуется на оплату отпусков, отдельных видов премирования, пособия для уходящим в армию, выплат государственных и общественных обязанностей, руб.;

Св – страховые взносы программистов, расходуется на пенсии и медицину, руб.;

Нцех – накладные расходы отдела, расходуются на амортизацию компьютеров и другого оборудования, заработную плату всех работников отдела (исключая программистов), ЖКХ, текущий и капитальный ремонт помещений отдела, руб.;

Нзав – накладные расходы фирмы, расходуются на амортизацию общефирменного оборудования, заработную плату всех работников фирмы (исключая работников отделов), ЖКХ, текущий и капитальный ремонт здания фирмы, аренду, отчисления вышестоящим организациям, руб.;

Впр – внепроизводственные расходы, расходуются на антивирусные программы, оплату интернета, покупку нового оборудования, сопровождение программы во время ее выполнения, руб.

В связи с высокой стоимостью оборудования для программиста, необходимостью быстрой амортизации и большим объемом потребляемой электроэнергии формула приобретает вид (4.3).

(4.3)

С = Зосн + Здоп + Св + А + Э + Нцех + Нзав + Впр,

где А – сумма амортизации оборудования, руб.;

Э – плата за потребленную электроэнергию, руб.

### Расчетная часть

### Расчет трудоемкости разработанной программы

Для определения себестоимости решения задачи необходимо, прежде всего, найти трудоемкость решения задачи.

Трудоемкость – это сумма затрат труда (по времени), необходимых для изготовления единицы продукции, которая предназначена для решения разработки программы в соответствии с содержанием задания.

Трудоемкость рассматривается как сумма затрат времени на разных этапах решения задачи. Общая трудоемкость рассчитывается по формуле (4.4).

(4.4)

То = Ти + Та + Тбс + Тп + Тот + Тд + Тмр + Тэвм,

где То – трудоемкость общая, час;

Ти – затраты труда на изучение материала, описание задачи, час;

Та – затраты труда на разработку алгоритмов решения задачи, час;

Тбс – затраты труда на разработку блок-схем алгоритма программы, час;

Тп – затраты труда на программирование, час;

Тот – затраты труда на отладку программы, час;

Тд – затраты на оформление документации, час;

Тмр – затраты труда на машинно-ручные работы, час;

Тэвм – время машинного счета на ЭВМ, час.

Слагаемые трудоемкости определяются через количество программных команд данной стадии разработки.

Затраты труда на изучение и описание задачи определяются по формуле (4.5).

(4.5)

где Q – предполагаемое число форм данной стадии разработки;

β – коэффициент, учитывающий качество описания задачи, равный 1,3 по данным предприятия;

В – скорость программиста, количество команд в час;

Ккв – коэффициент квалификации исполнителя, равный 0,8 по данным предприятия.

**Расчет трудоемкости на этапе изучения и описания задачи**

Число форм данной программы равно 7. По формуле (4.5) трудоемкость изучения материала и описания задачи.

B = 1;

β = 1,3;

Ккв = 0,8.

**Расчет затрат труда на этапе разработки алгоритма решения задачи**

Величина Ta находится по формуле, идентичной Tи.

B = 0,7;

β = 1,3;

Ккв = 0,8.

**Расчет затрат труда на этапе разработки схем алгоритма программы**

Величина Tбс находится по формуле, идентичной Tи.

B = 0,8;

β = 1,3;

Ккв = 0,8.

**Расчет затрат труда на этапе программирования**

Величина Tп находится по формуле, идентичной Ти.

B = 1,2;

β = 1,3;

Ккв = 0,8.

**Расчет затрат труда на этапе отладки**

Величина Tо находится по формуле, идентичной Tи.

B = 0,6;

β = 1,3;

Ккв = 0,8.

**Расчет затрат труда на этапе документации**

Величина Tд находится по формуле, идентичной Tи.

B = 0,8;

β = 1,3;

Ккв = 0,8.

**Расчет затрат труда на этапе машинно-ручных операций**

Трудоемкость на этапе машинно-ручных операций рассчитывается по формуле (4.6).

(4.6)

Где t – время ввода информации, сек.

Время ввода информации находится по формуле (4.7).

(4.7)

где С – объем программы в символах.

По формуле (4.7) необходимо найти время ввода информации при С = 150472.

Зная время ввода информации, можно найти трудоемкость на этапе машинно- ручных операций по формуле (4.6).

**Расчет затрат труда на этапе машинного времени**

Время машинного счета на ЭВМ вычисляется по формуле (4.8).

(4.8)

где tвв – время ввода, час;

tвыв – время вывода, час;

tсч – время счета, час.

По статистическим данным Тэвм = 0,02 часа.

**Расчет общей трудоемкости**

Зная все затраты труда на всех этапах решения задачи, можно вычислить общую трудоемкость решаемой задачи по формуле (4.4).

Трудоемкость решения задачи на ЭВМ 100,19 часа.

### Расчет себестоимости разработанной программы

Для нахождения себестоимости разработки программы при решении задачи на ЭВМ необходимо использовать формулу (4.9).

(4.9)

**Расчет основной заработной платы**

Заработная плата – это часть национального дохода, переданного в личное распоряжение работника в соответствии с количеством и качеством затраченного им труда. Заработная плата состоит из основной, дополнительной заработной платы и страховых взносов.

Основная заработная плата зависит от степени квалификации работника, уровня сложности, количества и качества выполняемой работы, а также условий, в которых выполняется работа и определяется по формуле (4.10).

(4.10)

где – часовая тарифная ставка, руб.;

– общая трудоемкость решения задачи, час

Часовая тарифная ставка определяется по формуле (4.11).

(4.11)

где Оклад – месячный штатный оклад программиста по данным предприятия, равен 80000 рублей;

22,8 – среднее количество рабочих дней в месяц;

8 – количество рабочих часов в смену.

Найдем часовую тарифную ставку по формуле (4.11).

Для нахождения основной заработной платы необходимо использовать формулу (4.10).

**Расчет дополнительной заработной платы**

Далее, необходимо найти дополнительную заработную плату по формуле (4.12), которая равна 80% от основной заработной платы по данным предприятия.

(4.12)

**Расчет страховых взносов**

Зная размер основной и заработной платы, можно вычислить сумму страховых взносов, которая составляет 30,2% от размера фонда оплаты труда (ФОТ) и является обязательным государственным налогом. ФОТ рассчитывается по формуле (4.13).

(4.13)

Зная ФОТ можно рассчитать страховые взносы по формуле (4.14).

(4.14)

**Расчет амортизационных отчислений**

Амортизация — это ежемесячное денежное отчисление для возмещения износа оборудования. Амортизация рассматриваемой программы вычисляется в три этапа. Сначала рассчитывается годовая амортизация по формуле (4.15).

(4.15)

где – первоначальная себестоимость оборудования, использованного для разработки программы, по данным предприятия 170 000 руб;

4 – срок окупаемости оборудования, год.

Далее рассчитывается амортизация дневная по формуле (4.16).

(4.16)

где 247 – количество рабочих дней в 2023 году.

Чтобы найти амортизацию программы необходимо вычислить амортизацию часовую, исходя из формулы (4.17).

(4.17)

где 8 – количество рабочих часов в смену.

Итого на каждый час работы программиста приходится 21,51 рубля амортизации. Вычислим амортизацию рассматриваемой программы по формуле (4.18).

(4.18)

**Расчет затрат на электроэнергию**

Затраты на электроэнергию рассчитываются по формуле (4.19).

(4.19)

где Тариф – действующий тариф на электроэнергию в Москве, по данным предприятия 6,67руб/кВт-час.

**Расчет накладных цеховых расходов**

Накладные расходы отдела определяются в процентном отношении от основной зарплаты, составляют 140% по данным предприятия. В состав накладных расходов отдела включаются такие затраты как заработная плата аппарата управления отдела (начальника отдела, заместителя начальника, системного администратора), амортизационные отчисления на текущий и капитальный ремонт заданий и оборудования, сооружений, на охрану труда в данном отделе и на непроизводительные затраты. Рассчитывается по формуле (4.20).

(4.20)

**Расчет цеховой себестоймости**

Себестоимость отдела – это показатель затратности производства, его анализ позволяет выявить фактор отклонения от процентного объема расходов и сравнить эффективность работы разных отделов. Рассчитывается по формуле (4.21).

(4.21)

**Расчет накладных расходов фирмы**

Накладные расходы фирмы определяются в процентном отношении от основной зарплаты и составляют 100% по данным предприятия. Накладные расходы фирмы – это расходы по управлению фирмой, содержание общефирменного персонала с отчислением на страховые взносы, расходы по командировкам, амортизационные отчисления общефирменного оборудования, на текущий и капитальный ремонт зданий, отчисление вышестоящим организациям. Рассчитываются по формуле (4.22).

(4.22)

**Расчет производственной себестоймости**

Производственная себестоимость определяется путем суммирования общезаводских и целевых расходов с себестоимостью цеховой. Она включает производственные затраты всех отделов, занятых производством продукции или услуг, и расходы по общему управлению предприятием. Себестоимость производственная рассчитывается по формуле (4.23).

**Расчет внепроизводственных расходов**

(4.23)

Внепроизводственные расходы являются неотъемлемой частью затрат предприятия и могут включать в себя рекламу, сопровождение программы во время ее исполнения на предприятии, оплату интернета, первоначальную настройку оборудования и сетей, поддержание работоспособности системы ЭВМ. По данным предприятия они составляют 10% от стоимости заводской и рассчитываются по формуле (4.24).

(4.24)

**Расчет полной себестоимости решения задачи**

Таким образом, зная все необходимые величины, можно вычислить полную себестоимость. Она включает затраты организации не только на выпуск продукции и организацию производственного процесса, но и на ее реализацию, иначе – на ее поставку на рынок конечного товара и услуг. Она учитывается при формировании цены реализации этой продукции и служит показателем для определения суммы прибыли, получаемой от ее продажи. Вычисляется по формуле (4.25).

(4.25)

### Анализ возможных путей снижения себестоимости

Исходя из формулы 4.3 возможными путями снижения себестоимости могу стать:

* Заработная плата основная и дополнительная;
* Cтраховые взносы основных производственных рабочих;
* Накладные расходы цеховые;
* Накладные расходы завода;
* Внепроизводственные расходы;
* Затраты на электроэнергию.

1. Снижение заработной платы может быть обеспечено при пересмотре тарифов за выполнение работ. Однако использование данного пути представляется нецелесообразным направлением, так как в полном объеме связано с трудоемкостью выполняемой работы и мотивацией труда. Снижение оплаты труда негативно отражается на заинтересованности работника, снижается качество работы и его лояльность к организации.
2. Страховые взносы рассчитываются в процентном соотношении от величины заработной платы, в связи с этим повлиять на данный параметр, не затрагивая заработную плату невозможно.
3. Значительная часть накладных расходов, цеховых и завода, также пропорционально связаны с оплатой труда, в связи с этим существенно снизить их не представляется возможным.
   * Сокращение накладных расходов фирмы может быть достигнуто за счет исключения расходов на командировки, так как разрабатываемый продукт не требует согласований с внешними субъектами. Также, поскольку разработка осуществляется в головной организации, то из накладных цеховых расходов можно исключить отчисления вышестоящим организациям.
   * Сокращение накладных расходов цеховых расходов может быть обеспечено сокращением административно-управленческого персонала фирмы, в частности, в связи с вводом разрабатываемого автоматизированного приложения, могут быть высвобождены не менее двух человек менеджеров низового (технического) уровня, или три единицы служащих (секретари, помощники, работники канцелярии), при этом, ха счет оптимизации в распределении выполняемых функций обеспечивается высвобождение фонда заработной платы в размере от 1,5 до 2%, что позволяет обеспечить снижение себестоимости на 0,5 -1%
4. Наиболее вероятными направлениями снижения себестоимости могут стать минимизация внепроизводственных расходов и затраты на электроэнергию.
   * По первому направлению (минимизация внепроизводственных расходов) результат может быть получен в связи с отказом от: затрат на рекламу (так как разрабатываемый продукт не рассчитан на внешнего потребителя и не нуждается в рекламе); Затрат на первоначальную настройку оборудования и сетей, так как продукт создается с учетом имеющихся настроек. Также отсутствуют затраты на интернет, так как продукт может использоваться с применением внутрифирменной локальной сети.
   * Снижение затрат на электроэнергию может быть обеспечено за счет замен ламп накаливания и ламп с большей энергоемкостью более экономичными или оснащение подсобных помещений датчиками движения.

### Графическая часть

В таблице 4.1 представлена производительность труда программиста.

Таблица 4.1 – Производительность труда программиста

|  |  |
| --- | --- |
| **Характер работы** | **Производительность, количество форм/час** |
| Изучение описания задачи | 1 |
| Разработка алгоритмов решения | 0,7 |
| Разработка схем алгоритма | 0,8 |
| Программирование по готовой схеме алгоритма с использованием алгоритмического языка | 1,2 |
| Автономная отладка программы | 0,6 |
| Оформление документации | 0,8 |

В таблице 4.2 представлена трудоемкость при решении задачи на ЭВМ.

Таблица 4.2 – Трудоемкость при решении задачи на ЭВМ

|  |  |  |
| --- | --- | --- |
| **Наименование затрат** | **Единица измерения** | **Трудоемкость в часах** |
| Трудоемкость решения задачи на ЭВМ | час | 100,19 |
| В том числе: | | |
| Время изучения описания задачи | час | 11,38 |
| Время на разработку алгоритма задачи | час | 16,25 |
| Время на разработку схемы алгоритма | час | 14,22 |
| Время на программирование | час | 9,48 |
| Время на отладку программы | час | 18,96 |

Продолжение таблицы 4.2

|  |  |  |
| --- | --- | --- |
| Время на оформление документации | час | 14,22 |
| Время на машинно-ручные работы | час | 16,67 |
| Время машинного счета | час | 0,02 |

В таблице 4.3 представлена калькуляция разработки программы.

Таблица 4.3 – Таблица калькуляции

|  |  |  |  |
| --- | --- | --- | --- |
| **Наименование статьи калькуляции** | **Сумма в руб** | **Процент к итогу** | **Процент к основной зарплате** |
| Прямые затраты: | | | |
| Основная заработная плата | 43944,81 | 18,91 | 100,00 |
| Дополнительная заработная плата | 35155,85 | 15,13 | 80,00 |
| Страховые взносы | 23888,40 | 10,28 | 54,36 |
| Амортизация оборудования | 2154,99 | 0,93 | 4,90 |
| Затраты на электроэнергию | 668,30 | 0,29 | 1,52 |
| Косвенные затраты: | | | |
| Накладные цеховые расходы | 61522,73 | 26,47 | 140,00 |
| Себестоимость цеховая | 167336,07 | 72,00 | 380,78 |
| Накладные заводские расходы | 43944,81 | 18,91 | 100,00 |
| Себестоимость производственная | 211279,88 | 90,91 | 480,78 |
| Внепроизводственные расходы | 21127,99 | 9,09 | 48,08 |
| Себестоимость полная | 232407,87 | 100,00 | 528,86 |

В таблице 4.4 представлены технико-экономические показатели.

Таблица 4.4 – Технико-экономические показатели

|  |  |  |  |
| --- | --- | --- | --- |
| **Технико- экономический показатель** | **Единица измерения** | **Формула** | **Результат** |
| Трудоёмкость решения задачи на ЭВМ | час | То = Ти + Та + Тбс +  +Тп + Тот + Тд + Тмр +  +Тэвм | 100,19 |
| Себестоимость решения задачи на ЭВМ | руб. | С = Зосн + Здоп + Св + А +  +Э + Нцех + +Нзав + Впр | 232407,87 |

# ЗАКЛЮЧЕНИЕ

В заключение выпускной квалификационной работы можно отметить значимость данного исследования в современном информационном обществе. Развитие технологий и расширение возможностей удаленного доступа к файлам и операционным системам ставят перед разработчиками задачу создания эффективных и безопасных инструментов удаленного управления.

В рамках данной работы была предложена и реализована программная система, которая позволяет осуществлять удаленное управление файлами и операционными системами, обеспечивая пользователям гибкость, мобильность и безопасность в работе с данными. Система предоставляет возможность удаленного доступа, управления и обмена файлами, а также выполнения операций на удаленных операционных системах.

В процессе разработки были рассмотрены современные подходы и методы удаленного управления, проведен анализ требований и функциональности системы. Были выбраны и применены соответствующие технологии и инструменты для разработки, а также разработаны соответствующие алгоритмы и протоколы взаимодействия.

Полученные результаты демонстрируют успешную реализацию программной системы, которая удовлетворяет поставленным требованиям и достигает поставленных целей. Проведенное тестирование подтвердило работоспособность и надежность системы, а также ее эффективность в реальных сценариях использования.

В итоге, разработанная программная система удаленного управления файлами и операционными системами представляет собой важный шаг в области удаленных технологий, предоставляя пользователям удобный и безопасный способ взаимодействия с удаленными ресурсами. Результаты данной работы могут быть использованы для дальнейшего совершенствования и развития подобных систем, а также внедрены в различные области, где требуется удаленное управление файлами и операционными системами.

В результате внедрения приложения, организация получила следующие преимущества:

* Ускорение и безопасноть при работе с удаленными операционными системами.
* Удобство и доступность для всех разработчиков.

Внедрение приложения приведет к оптимизации процессов и повышению эффективности работы, что положительно отразится на всей организации.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Документация по языку программирования TypeScript для Веб-приложений. [Электронный ресурс]. Доступно: https://www.typescriptlang.org/.
2. Документация по библиотеке React для построения UI и логики. [Электронный ресурс]. Доступно: https://react.dev/.
3. Документация по фреймворку Electron. [Электронный ресурс]. Доступно: https://www.electronjs.org/.
4. Документация по HTML и CSS. [Электронный ресурс]. Доступно: https://www.w3schools.com/.

# ПРИЛОЖЕНИЯ

# ПРИЛОЖЕНИЕ А

(обязательное)

Листинг программы

/\*Выпускная квалификационная работа по теме: «Разработка программной системы удаленного управления файлами и операционными системами»

\*Название: Terix

\*Разработал: Петрунин Артем Романович

\*Группа: ТИП-83

\*Дата: 20.06.2023 v.1.0.0

\*Язык: TypeScript

\*Краткое описание: данная программа является информационной системой для технического обслуживания сотрудников компании

\*Задание:

* 1. создание базы данных для информационной системы «Terix»;
  2. создание форм для просмотра, создания, удаления групп и соедиений, с помощью использования графического интерфейса;
  3. реализация возможности регистрация новых пользователей в системе и их авторизации.

\*Используемые формы:

Main - «Главная»; Login - «Авторизация»; Register - «Регистрация»; SFTP - «Управление файлами»; Console - «Управление операционными системами»; CreateGroup - «Создание группы»; CreateConnection - «Создание подключения»;

import axios from "axios";

import { SnackbarProvider } from "notistack";

import { createRoot } from "react-dom/client";

import config from "renderer/config";

import { SkeletonTheme } from "react-loading-skeleton";

import { Providers } from "renderer/components/Providers";

import App from "./App";

import "react-tooltip/dist/react-tooltip.css";

import "react-loading-skeleton/dist/skeleton.css";

axios.defaults.baseURL = config.api.baseUrl;

axios.defaults.headers.Authorization = "Bearer aZzCtpkQHeO4KxsU3NmP9mZYS3DC0KEf";

const container = document.getElementById("root") as HTMLElement;

const root = createRoot(container);

root.render(

<Providers>

<SnackbarProvider

anchorOrigin={{

vertical: 'bottom',

horizontal: 'right',

}}

>

<SkeletonTheme

baseColor="#262626"

highlightColor="#343434"

>

<App />

</SkeletonTheme>

</SnackbarProvider>

</Providers>

);

import { useEffect } from "react";

import { MemoryRouter as Router, Route, Routes } from "react-router-dom";

import Sidebar from "renderer/components/Sidebar";

import MainPage from "renderer/pages/Main";

import SftpPage from "renderer/pages/Sftp";

import ConsolePage from "renderer/pages/Console";

import { useAppDispatch, useAppSelector } from "renderer/hooks/redux";

import { useFirstRender } from "renderer/hooks/useFirstRender";

import { groupsSlice } from "renderer/store/reducers/groups/slice";

import { connectionsSlice } from "renderer/store/reducers/connections/slice";

import "./App.scss";

export default function App() {

const { groups } = useAppSelector((state) => state.groupsReducer);

const { connections } = useAppSelector((state) => state.connectionsReducer);

const firstRender = useFirstRender();

const dispatch = useAppDispatch();

useEffect(() => {

if (firstRender) {

const stateString = localStorage.getItem("state");

if (!stateString) {

return;

}

const { groups, connections } = JSON.parse(stateString) || {};

dispatch(groupsSlice.actions.set(groups));

dispatch(connectionsSlice.actions.set(connections));

return;

}

localStorage.setItem("state", JSON.stringify({

groups,

connections

}));

}, [groups, connections]);

return (

<Router>

<Routes>

<Route path="/" element={(<Sidebar><MainPage /></Sidebar>)} />

<Route path="/sftp" element={(<Sidebar><SftpPage /></Sidebar>)} />

<Route path="/console" element={(<Sidebar><ConsolePage /></Sidebar>)} />

</Routes>

</Router>

);

}

@import "renderer/style/reset";

@import "renderer/style/fonts";

@import "renderer/style/colors";

@import "renderer/style/selection";

@import "renderer/style/scrollbar";

body {

width: 100vw;

height: 100vh;

font-family: 'Manrope', sans-serif;

overflow-x: hidden;

background: #FFFFFF;

}

@import url('https://fonts.googleapis.com/css2?family=Manrope:wght@200;300;400;500;600;700;800&display=swap');

@import url('https://fonts.googleapis.com/css2?family=Fira+Code:wght@400;500;600&display=swap');

/\* Box sizing rules \*/

\*,

\*::before,

\*::after {

box-sizing: border-box;

}

a {

text-decoration: none;

&:focus {

outline: unset;

}

}

input,

button {

all: unset;

box-sizing: border-box;

}

/\* Remove default margin \*/

body,

h1,

h2,

h3,

h4,

p,

figure,

blockquote,

dl,

dd {

margin: 0;

}

/\* Remove list styles on ul, ol elements with a list role, which suggests default styling will be removed \*/

ul[role="list"],

ol[role="list"] {

list-style: none;

}

/\* Set core root defaults \*/

html:focus-within {

scroll-behavior: smooth;

}

/\* Set core body defaults \*/

body {

min-height: 100vh;

text-rendering: optimizeSpeed;

line-height: 1.5;

}

/\* A elements that don't have a class get default styles \*/

a:not([class]) {

text-decoration-skip-ink: auto;

}

/\* Make images easier to work with \*/

img,

picture {

max-width: 100%;

display: block;

}

/\* Inherit fonts for inputs and buttons \*/

input,

button,

textarea,

select {

font: inherit;

}

/\* Remove all animations, transitions and smooth scroll for people that prefer not to see them \*/

@media (prefers-reduced-motion: reduce) {

html:focus-within {

scroll-behavior: auto;

}

\*,

\*::before,

\*::after {

animation-duration: 0.01ms !important;

animation-iteration-count: 1 !important;

transition-duration: 0.01ms !important;

scroll-behavior: auto !important;

}

}

import React, { useState } from "react";

import { useNavigate } from "react-router-dom";

import HostIcon from "@icons/host.svg";

import GroupIcon from "@icons/group.svg";

import HostItemIcon from "@icons/host\_item.svg";

import EditIcon from "@icons/edit.svg";

import CreatehostPlaceholder from "@images/createhost\_placeholder.png";

import CreateGroup from "renderer/modals/CreateGroup";

import CreateConnection from "renderer/modals/CreateConnection";

import { useAppDispatch, useAppSelector } from "renderer/hooks/redux";

import style from "./index.module.scss";

import { ConnectionI } from "renderer/types/connection";

import { connectionSlice } from "renderer/store/reducers/connection/slice";

import { GroupI } from "renderer/types/groups";

export default function MainPage() {

const { history } = useAppSelector((state) => state.connectionReducer);

const { groups } = useAppSelector((state) => state.groupsReducer);

const { connections } = useAppSelector((state) => state.connectionsReducer);

const navigate = useNavigate();

const dispatch = useAppDispatch();

const [editGroupData, setEditGroupData] = useState<GroupI>();

const [createGroupVisible, setCreateGroupVisible] = useState(false);

const [editConnectionData, setEditConnectionData] = useState<ConnectionI>();

const [createConnectionVisible, setCreateConnectionVisible] = useState(false);

const connectSSH = async (connection: ConnectionI) => {

await Promise.all([

dispatch(connectionSlice.actions.addConnection({

...connection,

messages: "",

commands: []

})),

dispatch(connectionSlice.actions.changeActiveConnection(history.length))

]);

await navigate("/console");

};

const editConnection = async (e: React.MouseEvent<HTMLDivElement>, connection: ConnectionI) => {

e.stopPropagation();

await setEditConnectionData(connection);

setCreateConnectionVisible(true);

};

const editGroup = async (e: React.MouseEvent<HTMLDivElement>, group: GroupI) => {

e.stopPropagation();

await setEditGroupData(group);

setCreateGroupVisible(true);

};

const getConnections = (group: GroupI) => {

return connections.filter((item) => item.groupId === group.id);

};

return (

<>

<div className={style.page}>

<input

className={style.search}

type="text"

placeholder="Поиск"

/>

<div className={style.actions}>

{groups.length !== 0 && (

<button

className={style.actions\_\_item}

onClick={async () => {

await setEditConnectionData(undefined);

setCreateConnectionVisible(true);

}}

>

<img src={HostIcon} alt="" />

<span>Соединение</span>

</button>

)}

<button

className={style.actions\_\_item}

onClick={async () => {

await setEditGroupData(undefined);

setCreateGroupVisible(true);

}}

>

<img src={GroupIcon} alt="" />

<span>Группа</span>

</button>

</div>

<div className={style.groups}>

{groups.length !== 0 ? (

groups.map((group, index) => (

<div className={style.groups\_\_item} key={group.name + index}>

<div className={style.groups\_\_header}>

<div className={style.groups\_\_caption}>{group.name}</div>

<div className={style.groups\_\_actions}>

<div

className={style.groups\_\_action}

onClick={(e) => editGroup(e, group)}

>

<img src={EditIcon} alt="" />

</div>

</div>

</div>

<div className={style.groups\_\_list}>

{getConnections(group).length !== 0 ? (

getConnections(group).map((connection) => (

<div

className={style.group\_\_item}

key={group.name + connection.label}

onClick={() => connectSSH(connection)}

>

<img className={style.groupItem\_\_icon} src={HostItemIcon} alt="" />

<div className={style.groupItem\_\_info}>

<div className={style.groupItem\_\_name}>{connection.label}</div>

<div className={style.groupItem\_\_description}>shh, {connection.login}</div>

</div>

<div

className={style.groupItem\_\_action}

onClick={(e) => editConnection(e, connection)}

>

<img src={EditIcon} alt="" />

</div>

</div>

))

) : (

<div>Добавьте соединение</div>

)}

</div>

</div>

))

) : (

<div className={style.placeholder}>

<div className={style.placeholder\_\_modal}>

<h2 className={style.placeholder\_\_title}>Добавьте свое первое соединение</h2>

<img className={style.placeholder\_\_image} src={CreatehostPlaceholder} alt="" />

<button

className={style.placeholder\_\_button}

onClick={async () => {

await setEditGroupData(undefined);

setCreateGroupVisible(true);

}}

>

Создать

</button>

</div>

</div>

)}

</div>

</div>

<CreateGroup

data={editGroupData}

isVisible={createGroupVisible}

onClose={() => setCreateGroupVisible(false)}

/>

<CreateConnection

data={editConnectionData}

isVisible={createConnectionVisible}

onClose={() => setCreateConnectionVisible(false)}

/>

</>

);

}

import { enqueueSnackbar } from "notistack";

import React, { useEffect, useRef, useState } from "react";

import { useAppDispatch, useAppSelector } from "renderer/hooks/redux";

import { useAsyncEffect } from "renderer/hooks/useAsyncEffect";

import { connectionSlice } from "renderer/store/reducers/connection/slice";

import style from "./index.module.scss";

export default function ConsolePage() {

const { activeConnection, history } = useAppSelector((state) => state.connectionReducer);

const [command, setCommand] = useState("");

const consoleRef = useRef<HTMLDivElement>(null);

const dispatch = useAppDispatch();

useAsyncEffect(async () => {

if (activeConnection === undefined) {

return;

}

if (history[activeConnection].messages) {

return;

}

const { host, port, login, password } = history[activeConnection];

try {

await window.electron.app.connectSSH(host, port, login, password);

const data = await window.electron.app.sshExecute("run-parts /etc/update-motd.d/");

dispatch(connectionSlice.actions.addMessage({ index: activeConnection, data }))

} catch (e) {

console.log(e);

enqueueSnackbar({

message: "Произошла ошибка при подключении",

variant: "error"

});

}

}, []);

useEffect(() => {

if (!consoleRef?.current) {

return;

}

consoleRef.current.scrollTop = consoleRef.current.scrollHeight;

}, [history[activeConnection!]]);

const onKeyDown = async (e: React.KeyboardEvent<HTMLInputElement>) => {

if (e.key !== "Enter") {

return;

}

const data = await window.electron.app.sshExecute(command);

dispatch(connectionSlice.actions.addMessage({ index: activeConnection!, data }));

setCommand("");

};

if (activeConnection === undefined) {

return null;

}

const connection = history[activeConnection];

return (

<div className={style.console}>

<div className={style.consoleMessage} ref={consoleRef}>

{connection.messages}

</div>

<label className={style.consoleWritebar}>

<span className={style.writebar\_\_placeholder}>{connection.login}<span>~</span>$</span>

<input

className={style.writebar\_\_input}

value={command}

type="text"

onChange={(e) => setCommand(e.currentTarget.value)}

onKeyDown={onKeyDown}

/>

</label>

</div>

);

}

import { useState } from "react";

import { useAsyncEffect } from "renderer/hooks/useAsyncEffect";

import FileIcon from "@icons/file.svg";

import FolderIcon from "@icons/folder.svg";

import style from "./index.module.scss";

import { classNames } from "renderer/utils/classNames";

interface ElementItem {

name: string,

isDirectory: boolean

}

function Test() {

const [path, setPath] = useState<string>();

const [disks, setDisks] = useState<Array<string>>();

const [elements, setElements] = useState<Array<ElementItem>>();

useAsyncEffect(async () => {

const data = await window.electron.app.getDisks();

setDisks(data);

}, []);

const getElements = async (data: Array<string>, path: string) => {

let elements: Array<ElementItem> = [];

await Promise.all(

data.map(async (folder) => {

try {

const isDirectory = await window.electron.app.isDirectory(path + folder);

elements.push({

name: folder,

isDirectory

});

} catch (e) {

console.log(e);

}

})

);

return elements.sort((a, b) => +b.isDirectory - +a.isDirectory);

};

const goBack = async () => {

const splitPath = path?.split("\\");

if (!splitPath) {

return;

}

splitPath.length -= 2;

const prevPath = splitPath.join("\\") + "\\";

console.log("prevPath", prevPath);

if (prevPath === "\\") {

setPath(undefined);

return;

}

const data = await window.electron.app.getDirectoriesAndFiles(prevPath);

const elements = await getElements(data, prevPath);

setElements(elements);

setPath(prevPath);

};

const goPath = async (folder: string) => {

const directory = (path ? (path + folder) : folder) + "\\";

const data = await window.electron.app.getDirectoriesAndFiles(directory);

const elements = await getElements(data, directory);

setElements(elements);

setPath(directory);

};

const renderFolder = (name: string) => (

<div

className={style.elements\_\_item}

key={name}

onDoubleClick={() => goPath(name)}

>

<img className={style.item\_\_icon} src={FolderIcon} alt="" />

<span className={style.item\_\_text}>{name}</span>

</div>

);

const renderIcon = (name: string) => (

<div

className={style.elements\_\_item}

key={name}

>

<img className={style.item\_\_icon} src={FileIcon} alt="" />

<span className={style.item\_\_text}>{name}</span>

</div>

);

return (

<div className={classNames(style.elements, style.elementsLocal)}>

{path && (

<div className={style.elements\_\_item} onDoubleClick={goBack}>

<img className={style.item\_\_icon} src={FolderIcon} alt="" />

<span>..</span>

</div>

)}

{path ? (

elements?.map(({ name, isDirectory }) => isDirectory ? renderFolder(name) : renderIcon(name))

) : (

disks?.map((disk) => renderFolder(disk))

)}

</div>

);

}

export default function SftpPage() {

return (

<div className={style.sftp}>

<Test />

<Test />

</div>

);

}

import { findIndex } from "lodash";

import { v4 as randomUUID } from "uuid";

import { enqueueSnackbar } from "notistack";

import React, { useEffect, useState } from "react";

import ModalBase from "renderer/modals/Base";

import Button from "renderer/ui/Button";

import Input from "renderer/ui/Input";

import { useAppDispatch, useAppSelector } from "renderer/hooks/redux";

import { groupsSlice } from "renderer/store/reducers/groups/slice";

import { GroupI } from "renderer/types/groups";

import style from "./index.module.scss";

interface IProps {

data?: GroupI,

isVisible: boolean;

onClose(): void

}

export default function CreateGroup(props: IProps) {

const { isVisible, onClose } = props;

const { groups } = useAppSelector((state) => state.groupsReducer);

const dispatch = useAppDispatch();

const [name, setName] = useState("");

const isUpdate = !!props.data;

useEffect(() => {

if (!props.data) {

setName("");

return;

}

setName(props.data.name);

}, [props.data]);

const createGroup = () => {

if (name.length === 0) {

enqueueSnackbar({

message: "Название обязательно для заполнения",

variant: "error"

});

return;

}

if (isUpdate) {

const index = findIndex(groups, { id: props.data?.id });

dispatch(groupsSlice.actions.edit({

index,

data: {

...props.data!,

name

}

}));

} else {

const groupIndex = findIndex(groups, { name });

if (groupIndex !== -1) {

enqueueSnackbar({

message: "Группа уже существует",

variant: "error"

});

return;

}

dispatch(groupsSlice.actions.add({

id: randomUUID(),

name

}));

}

onClose();

};

const removeGroup = () => {

const index = findIndex(groups, { id: props.data?.id });

dispatch(groupsSlice.actions.remove(index));

onClose();

};

return (

<ModalBase

className={style.createGroup}

title={`${isUpdate ? "Изменение" : "Создание"} группы`}

isVisible={isVisible}

onClose={onClose}

>

<Input

name="name"

caption="Название"

value={name}

placeholder="Главный сервер"

onChange={(e) => setName(e.currentTarget.value)}

/>

<div className={style.createGroup\_\_action}>

{isUpdate && (

<Button appearance="destructive" onClick={removeGroup}>Удалить</Button>

)}

<Button onClick={createGroup}>{isUpdate ? "Изменить" : "Создать"}</Button>

</div>

</ModalBase>

);

}

.createGroup {

&\_\_action {

margin-top: 30px;

display: flex;

align-items: center;

justify-content: center;

gap: 10px;

}

}

import { z } from "zod";

import { findIndex } from "lodash";

import { v4 as randomUUID } from "uuid";

import React, { useEffect } from "react";

import { enqueueSnackbar } from "notistack";

import ModalBase from "renderer/modals/Base";

import Select from "renderer/ui/Select";

import Button from "renderer/ui/Button";

import Input from "renderer/ui/Input";

import { useZodForm } from "renderer/hooks/zod";

import { useAppDispatch, useAppSelector } from "renderer/hooks/redux";

import { connectionsSlice } from "renderer/store/reducers/connections/slice";

import { ConnectionI } from "renderer/types/connection";

import style from "./index.module.scss";

interface IProps {

data?: ConnectionI,

isVisible: boolean;

onClose(): void

}

const schema = z.object({

label: z.string()

.nonempty("Введите название"),

groupId: z.string()

.nonempty("Выберите группу"),

host: z.string()

.nonempty("Введите хост"),

port: z.number(),

login: z.string()

.nonempty("Введите имя пользователя"),

password: z.string()

.nonempty("Введите пароль")

});

export default function CreateConnection(props: IProps) {

const { isVisible, onClose } = props;

const { groups } = useAppSelector((state) => state.groupsReducer);

const { connections } = useAppSelector((state) => state.connectionsReducer);

const isUpdate = !!props.data;

const {

setFormData,

formData,

validateForm,

clearForm,

onInputChange,

onSelectChange,

onInputBlur

} = useZodForm(schema, {

label: "",

groupId: "",

host: "",

port: 22,

login: "",

password: ""

});

const dispatch = useAppDispatch();

useEffect(() => {

if (!props.data) {

clearForm();

return;

}

setFormData({

...props.data

});

}, [props.data]);

const createConnection = () => {

const validate = validateForm();

if (!validate.isValid) {

const errors = validate.errors!;

(Object.keys(errors) as Array<keyof typeof errors>).map((key) => {

if (!errors[key]) {

return;

}

enqueueSnackbar({

message: errors[key],

variant: "error"

});

});

return;

}

if (isUpdate) {

const index = findIndex(connections, { id: props.data?.id });

dispatch(connectionsSlice.actions.edit({

index,

data: {

...props.data!,

...formData

}

}));

} else {

dispatch(connectionsSlice.actions.add({

id: randomUUID(),

...formData

}));

clearForm();

}

onClose();

};

const removeConnection = () => {

const index = findIndex(connections, { id: props.data?.id });

dispatch(connectionsSlice.actions.remove(index));

onClose();

};

return (

<ModalBase

className={style.createGroup}

title={`${isUpdate ? "Изменение" : "Создание"} соединения`}

isVisible={isVisible}

onClose={onClose}

>

<div className={style.createGroup\_\_grid}>

<Input

caption="Название"

name="label"

value={formData.label}

placeholder="Pixefy"

onChange={onInputChange}

onBlur={onInputBlur}

/>

<Select

caption="Группа"

name="groupId"

value={formData.groupId}

onChange={onSelectChange}

>

<option value="" disabled>Выберите группу</option>

{groups.map((item) => (

<option key={item.id} value={item.id}>{item.name}</option>

))}

</Select>

</div>

<div className={style.createGroup\_\_title}>SHH</div>

<div className={style.createGroup\_\_grid}>

<Input

caption="Хост"

name="host"

value={formData.host}

placeholder="127.0.0.1"

onChange={onInputChange}

onBlur={onInputBlur}

/>

<Input

caption="Порт"

name="port"

value={formData.port}

placeholder="25565"

onChange={onInputChange}

onBlur={onInputBlur}

/>

<Input

caption="Пользователь"

name="login"

value={formData.login}

placeholder="root"

onChange={onInputChange}

onBlur={onInputBlur}

/>

<Input

caption="Пароль"

type="password"

name="password"

value={formData.password}

placeholder="..."

onChange={onInputChange}

onBlur={onInputBlur}

/>

</div>

<div className={style.createGroup\_\_action}>

{isUpdate && (

<Button appearance="destructive" onClick={removeConnection}>Удалить</Button>

)}

<Button onClick={createConnection}>{isUpdate ? "Изменить" : "Создать"}</Button>

</div>

</ModalBase>

);

}

import React, { useEffect } from "react";

import { DivProps } from "react-html-props";

import { classNames } from "renderer/utils/classNames";

import CloseIcon from "@icons/close.svg";

import style from "./index.module.scss";

export interface ModalProps extends DivProps {

title: string;

isVisible: boolean;

onClose(): void

}

export default function ModalBase(props: ModalProps) {

const {

title,

isVisible,

children,

onClose

} = props;

useEffect(() => {

const closeListener = (event: KeyboardEvent) => {

const { code } = event;

if (code === "Escape") {

onClose();

event.stopPropagation();

}

};

document.addEventListener("keydown", closeListener);

return () => {

document.removeEventListener("keydown", closeListener);

};

}, []);

return (

<div

className={classNames(

style.wrapper,

!isVisible && style.wrapperHidden

)}

>

<div

className={classNames(

style.baseModal,

!isVisible && style.baseModalHidden,

props.className

)}

>

<div className={style.header}>

<h2 className={style.header\_\_title}>

{title}

</h2>

<div className={style.closeActions}>

<img

className={style.closeActions\_\_icon}

src={CloseIcon}

alt=""

onClick={onClose}

/>

</div>

</div>

<div className={style.content}>

{children}

</div>

</div>

</div>

);

}

export function useModalClose(

cleanUp: () => void,

onClose: () => void

) {

return () => {

onClose();

setTimeout(cleanUp, 300);

};

}

import { TypedUseSelectorHook, useDispatch, useSelector } from "react-redux";

import { AppDispatch, RootState } from "renderer/store";

export const useAppDispatch = () => useDispatch<AppDispatch>();

export const useAppSelector: TypedUseSelectorHook<RootState> = useSelector;

import { useEffect } from "react";

export function useAsyncEffect(

effect: () => Promise<void>,

deps: ReadonlyArray<unknown>

) {

useEffect(() => {

effect();

}, deps);

}

import { useRef, useEffect } from "react";

export function useFirstRender() {

const firstRender = useRef(true);

useEffect(() => {

firstRender.current = false;

}, []);

return firstRender.current;

}

import React, { ChangeEvent, useState } from "react";

import { z, ZodType } from "zod";

import mapValues from "lodash/mapValues";

import pick from "lodash/pick";

import omit from "lodash/omit";

export type ZodFormData<TSchema extends ZodType> = z.infer<TSchema>;

export type ZodFormErrors<TSchema extends ZodType> = {

[key in keyof z.infer<TSchema>]?: string;

};

export function useZodForm<TSchema extends ZodType>(

schema: TSchema,

initialState: ZodFormData<TSchema>

) {

const [formData, setFormData] = useState(initialState);

const [formErrors, setFormErrors] = useState<ZodFormErrors<TSchema>>({});

const setValue = (key: keyof typeof formData, value: any) => {

setFormData({

...formData,

[key]: value

});

};

const clearForm = () => {

setFormData(initialState);

setFormErrors({});

};

const validateForm = useZodValidation(

schema, formData, formErrors, setFormErrors

);

const onInputChange = (event: ChangeEvent<HTMLInputElement>) => {

const { target } = event;

const { name, value } = target;

setFormData({

...formData,

[name]: value

});

// eslint-disable-next-line @typescript-eslint/ban-ts-comment

// @ts-ignore

setFormErrors({

...omit(formErrors, name)

});

};

const onSelectChange = (event: ChangeEvent<HTMLSelectElement>) => {

const { target } = event;

const { name, value } = target;

setFormData({

...formData,

[name]: value

});

}

const onInputBlur = (event: React.FocusEvent<HTMLInputElement>) => {

validateForm(event.target.name);

};

return {

formData,

setFormData,

formErrors,

setFormErrors,

setValue,

clearForm,

validateForm,

onInputChange,

onSelectChange,

onInputBlur

};

}

export function useZodValidation<TSchema extends ZodType>(

schema: TSchema,

formData: ZodFormData<TSchema>,

formErrors: ZodFormErrors<TSchema>,

setFormErrors: (errors: ZodFormErrors<TSchema>) => void

) {

return (...fieldNames: string[]): {

isValid: boolean,

errors?: ZodFormErrors<TSchema>

} => {

const result = schema.safeParse(formData);

if (result.success) {

return {

isValid: true

};

}

const zodErrors = result.error.flatten().fieldErrors;

const mappedErrors = mapValues(zodErrors, (it) => it?.at(0));

const anyField = fieldNames.length === 0;

if (anyField) {

setFormErrors(mappedErrors);

} else {

setFormErrors({

...formErrors,

...pick(mappedErrors, fieldNames)

});

}

return {

isValid: false,

errors: mappedErrors

};

};

}

import React, { ReactNode } from "react";

import { Link, useLocation, useNavigate } from "react-router-dom";

import { useAppDispatch, useAppSelector } from "renderer/hooks/redux";

import { connectionSlice } from "renderer/store/reducers/connection/slice";

import { classNames } from "renderer/utils/classNames";

import HostsIcon from "@icons/hosts.svg";

import SftpIcon from "@icons/sftp.svg";

import HistoryIcon from "@icons/history.svg";

import ActiveHostIcon from "@icons/active\_host.svg";

import RemoveIcon from "@icons/close.svg";

import style from "./index.module.scss";

export default function Sidebar({ children }: {

children: ReactNode

}) {

const { history, activeConnection } = useAppSelector((state) => state.connectionReducer);

const navigate = useNavigate();

const { pathname } = useLocation();

const dispatch = useAppDispatch();

const connectSSH = async (index: number) => {

await dispatch(connectionSlice.actions.changeActiveConnection(index));

await navigate("/console");

};

const onRemoveHistoryClick = async (e: React.MouseEvent<HTMLDivElement>, index: number) => {

e.stopPropagation();

await navigate("/");

dispatch(connectionSlice.actions.removeConnection(index));

};

const checkActive = (url: string) => {

if (url === "/" && pathname === "/") {

return true;

}

return url !== "/" && pathname?.includes(url);

};

return (

<div className={style.layout}>

<div className={style.sidebar}>

<div className={style.sidebar\_\_logo}>

Terix

</div>

<div className={style.sidebar\_\_menu}>

<Link

className={classNames(

style.menu\_\_item,

checkActive("/") && style.menu\_\_itemActve

)}

to="/"

>

<img src={HostsIcon} alt="" />

<span>Hosts</span>

</Link>

{history.map((item, index) => (

<div

className={classNames(

style.menu\_\_item,

(checkActive("/console") && index === activeConnection) && style.menu\_\_itemActve

)}

key={item.label + index}

onClick={() => connectSSH(index)}

>

<img src={ActiveHostIcon} alt="" />

<span>{item.label} <span className={style.menuItem\_\_small}>{item.login}</span></span>

<img

className={style.menuItem\_\_remove}

src={RemoveIcon} alt=""

onClick={(e) => onRemoveHistoryClick(e, index)}

/>

</div>

))}

<Link

className={classNames(

style.menu\_\_item,

checkActive("/ssh") && style.menu\_\_itemActve

)}

to="/sftp"

>

<img src={SftpIcon} alt="" />

<span>SFTP</span>

</Link>

{/\*<div className={style.menu\_\_item}>\*/}

{/\* <img src={HistoryIcon} alt="" />\*/}

{/\* <span>History</span>\*/}

{/\*</div>\*/}

</div>

</div>

<div className={style.content}>

{children}

</div>

</div>

);

}

import { ReactNode } from "react";

import { Provider } from "react-redux";

import { store } from "renderer/store";

export function Providers({ children }: {

children: ReactNode

}) {

return (

<Provider store={store}>

{children}

</Provider>

);

}

// Disable no-unused-vars, broken for spread args

/\* eslint no-unused-vars: off \*/

import { contextBridge, ipcRenderer } from "electron";

import { IpcChannels } from "common/ipc";

const electronHandler = {

app: {

openLink: (url: string) => {

ipcRenderer.send(IpcChannels.app.openLink, url);

},

connectSSH(host: string, port: number, login: string, password: string) {

return ipcRenderer.invoke(IpcChannels.app.connectSSH, { host, port, login, password });

},

sshExecute(command: string): Promise<string> {

return ipcRenderer.invoke(IpcChannels.app.sshExecute, command);

},

getDirectoriesAndFiles(source: string): Promise<Array<string>> {

return ipcRenderer.invoke(IpcChannels.app.getDirectoriesAndFiles, source);

},

getDisks(): Promise<Array<string>> {

return ipcRenderer.invoke(IpcChannels.app.getDisks);

},

isDirectory(path: string): Promise<boolean> {

return ipcRenderer.invoke(IpcChannels.app.isDirectory, path);

}

}

};

contextBridge.exposeInMainWorld("electron", electronHandler);

export type ElectronHandler = typeof electronHandler;

import { ipcMain, shell } from "electron";

import { NodeSSH } from "node-ssh";

import { readdir, lstat } from 'fs/promises';

const nodeDiskInfo = require('node-disk-info');

const ssh = new NodeSSH();

ipcMain.on("app:open\_link", (\_, link) => {

shell.openExternal(link)

.catch((e) => {

console.log("Unable to open link", e);

});

});

ipcMain.handle("ssh:connect", async (\_, { host, port, login, password }) => {

await ssh.connect({

host,

port,

username: login,

password

});

});

ipcMain.handle("ssh:execute", async (\_, command) => {

const result = await ssh.execCommand(command);

return result.stdout || result.stderr;

});

ipcMain.handle("pc:get\_directories\_and\_files", async (\_, source) => {

return (await readdir(source, { withFileTypes: true }))

.map(dirent => dirent.name);

});

ipcMain.handle("pc:get\_disks", async (\_) => {

return (await nodeDiskInfo.getDiskInfo())

.map((disk: any) => disk.mounted);

});

ipcMain.handle("pc:is\_directory", async (\_, path) => {

return (await lstat(path)).isDirectory();

});

export const IpcChannels = {

app: {

openLink: "app:open\_link",

connectSSH: "ssh:connect",

sshExecute: "ssh:execute",

getDirectoriesAndFiles: "pc:get\_directories\_and\_files",

getDisks: "pc:get\_disks",

isDirectory: "pc:is\_directory",

}

};

# ПРИЛОЖЕНИЕ Б

# (обязательное)

# Результаты выполнения программы

На рисунке Б.1 продемонстрирован интерфейс экрана «Авторизация».



Рисунок Б.1 — Экран «Авторизация»

На рисунке Б.2 представлен первый экран «Регистрация».



Рисунок Б.2 — Экран «Регистрация»

На рисунке Б.3 изображен экран «Главная».

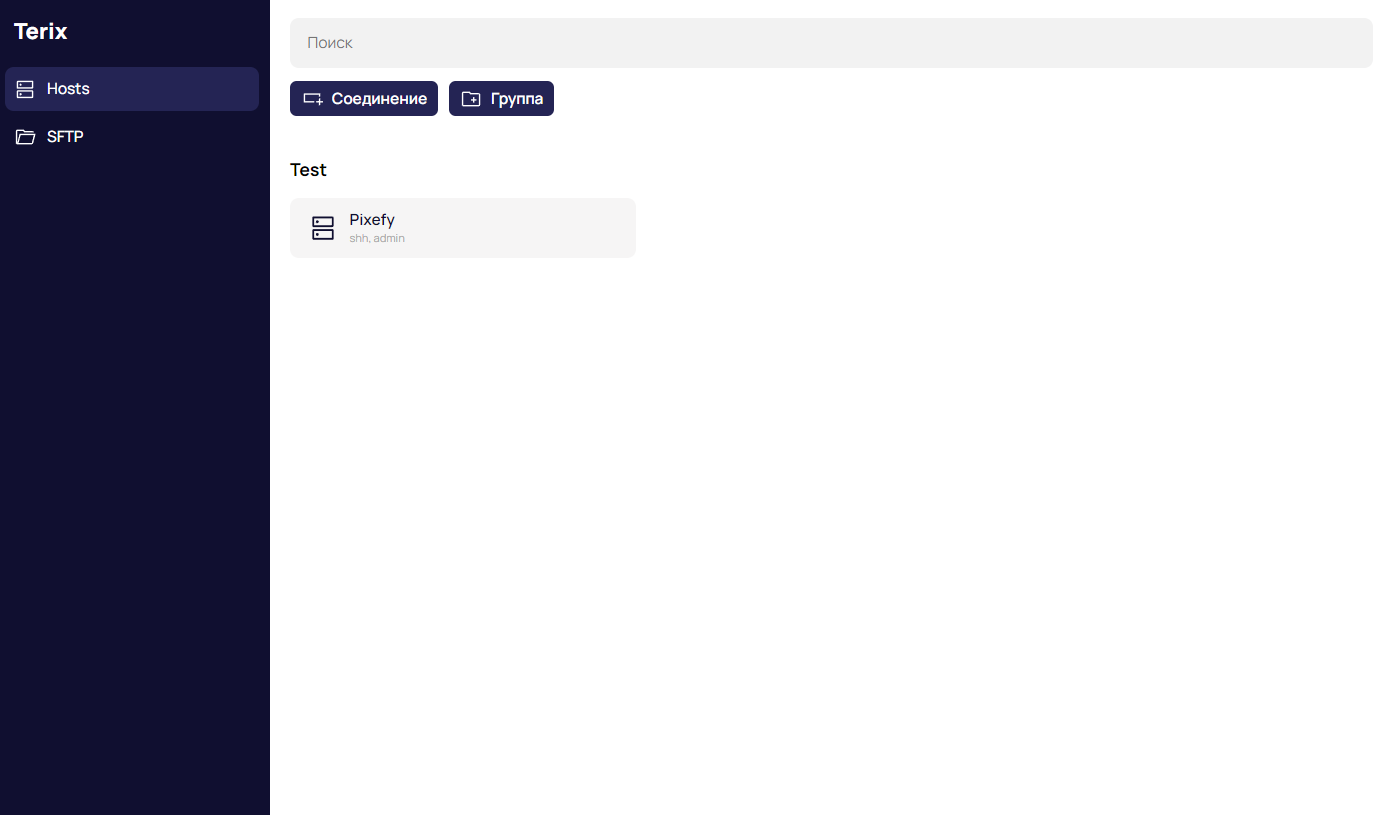


Рисунок Б.3 — Экран «Главная»

На рисунке Б.4 изображен экран «Консоль».

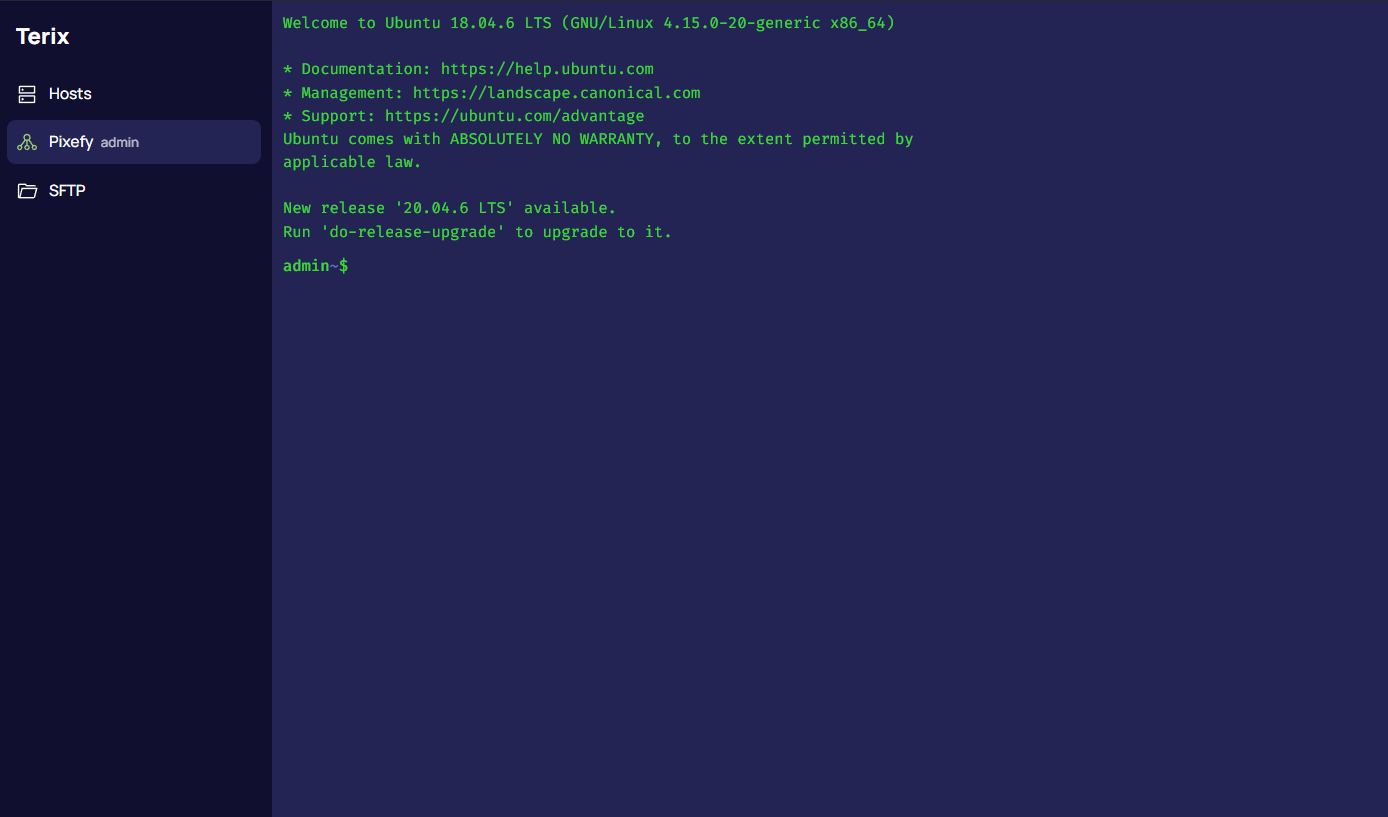


Рисунок Б.4 — Экран «Консоль»

На рисунке Б.5 изображено окно «SFTP».

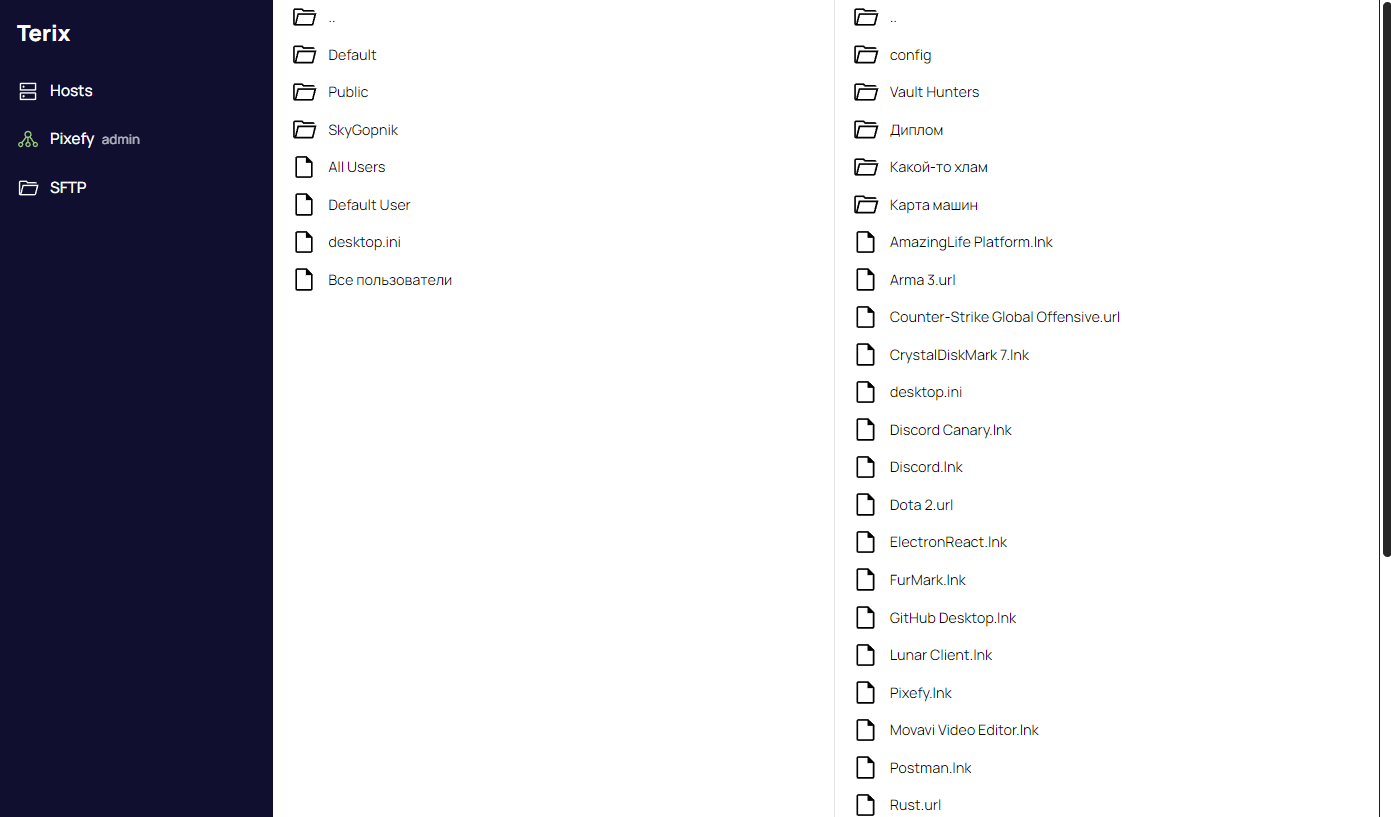


Рисунок Б.5 — Окно «SFTP»

На рисунке Б.6 изображено окно «Создания соединения».

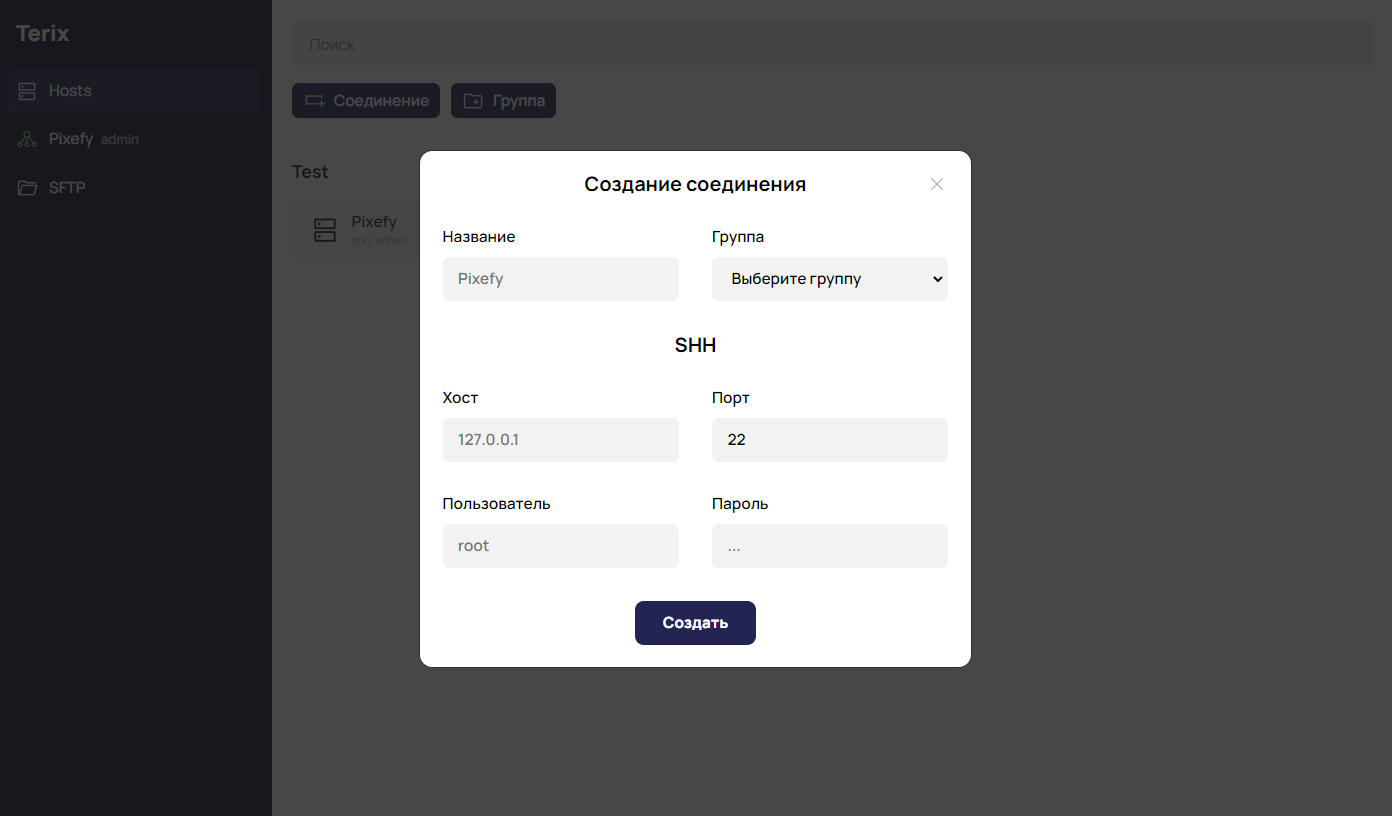


Рисунок Б.6 — Окно «Создание соединения»

На рисунке Б.7 изображен экран «Создания группы».

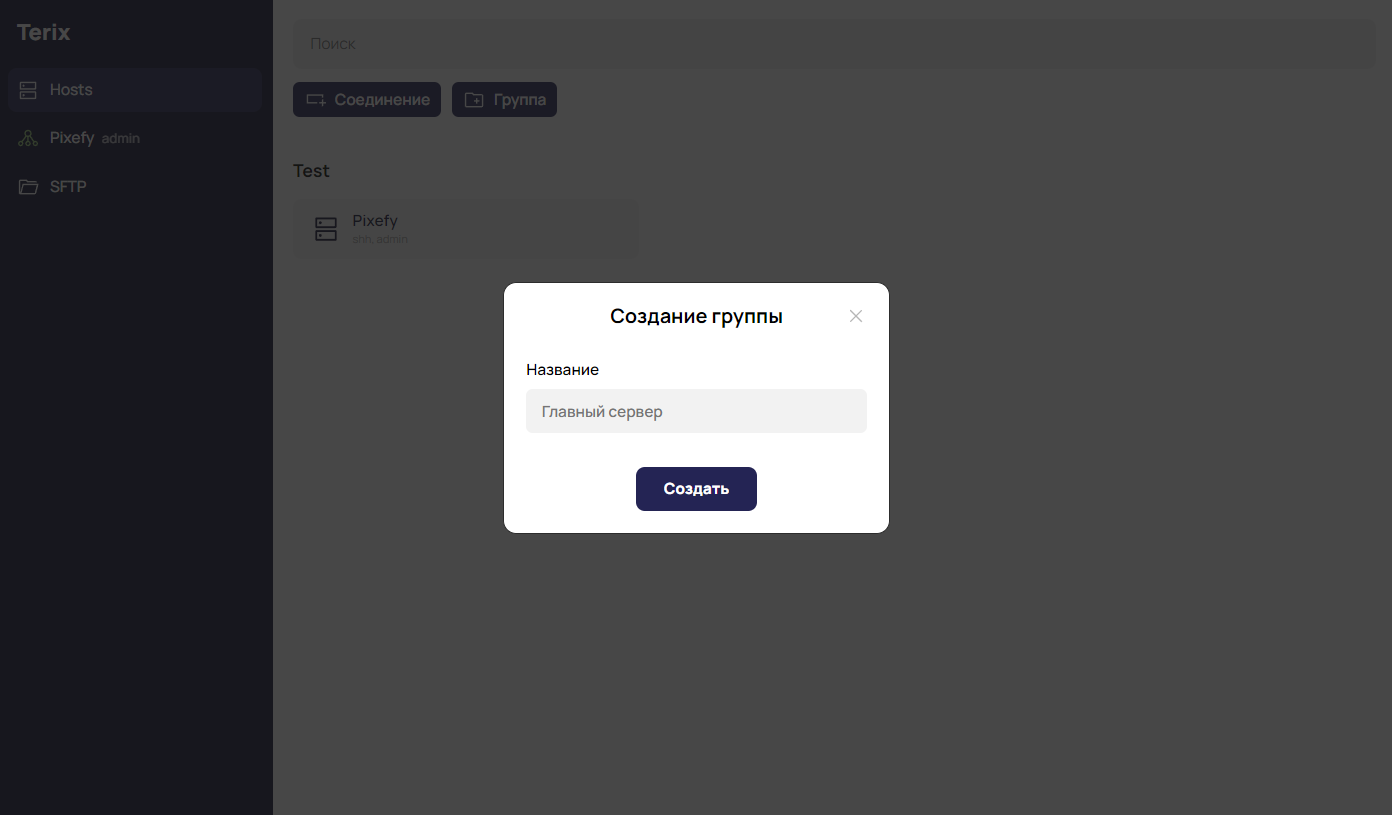


Рисунок Б.7 — Экран «Создание группы»