Министерство образования Республики Беларусь
Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей
Кафедра информатики
Дисциплина «Операционные системы и сети»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовому проекту
на тему:
«УДАЛЕННОЕ АДМИНИСТРИРОВАНИЕ»

БГУИР КП 1-40 04 01 016 ПЗ

Выполнил студент группы 053504
Борисенок Елизавета Андреевна

_____
(подпись студента)
Курсовой проект представлен на проверку ___.____.2023

_____
(подпись студента)

Минск 2023

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Конечные пользователи и компании по всему миру используют современные технологии для выполнения своих задач и достижения своих целей. В связи с этим, управление системами и сетями становится все более важным и требует от команды администраторов многих усилий. Одним из способов снижения нагрузки на администраторов и повышения эффективности управления сетями и системами является удаленное администрирование.

Удаленное администрирование – это процесс управления системами, который осуществляется удаленно с помощью специальных инструментов и технологий. Этот процесс позволяет администраторам удаленно мониторить, настраивать и управлять удаленными компьютерами, серверами, устройствами и сетями.

Целью данной курсовой работы является рассмотрение принципов и методов удаленного администрирования, а также оценка его эффективности и безопасности. В работе будут рассмотрены различные технологии удаленного администрирования, такие как *RDP*, *SSH*, *VPN*, а также методы обеспечения безопасности удаленного доступа к системам и сетям, такие как двухфакторная аутентификация, шифрование и многоуровневая защита. Также будет составлена собственная реализация программы удаленного администрирования.

В результате данной работы будет представлено обширное исследование удаленного администрирования, которое поможет администраторам и *IT*-специалистам лучше понимать этот процесс и использовать его в своей работе.

В настоящее время удаленное администрирование является неотъемлемой частью работы *IT*-специалистов и администраторов. Это связано с тем, что современные технологии позволяют работать удаленно из любой точки мира, что увеличивает гибкость и мобильность работы. Более того, удаленное администрирование позволяет управлять большим количеством систем и сетей одновременно, что повышает производительность и снижает затраты на обслуживание.

Однако, удаленное администрирование также представляет определенные риски и вызывает опасения в плане безопасности. Возможность удаленного доступа к системам и сетям может быть использована злоумышленниками для нанесения ущерба или кражи конфиденциальных данных. Поэтому, важно использовать надежные и безопасные методы удаленного администрирования и защиты данных.

В данной курсовой работе также будут рассмотрены примеры использования удаленного администрирования в различных сферах, таких как образование, медицина, банковское дело и т.д. Это поможет понять, как удаленное администрирование может быть применено для оптимизации процессов в разных отраслях и областях.

1 ТЕОРЕТИЧЕСКИЕ ОСНОВЫ УДАЛЕННОГО АДМИНИСТРИРОВАНИЯ

1.1 Введение в удаленное администрирование

Удаленное администрирование – это процесс управления системами и сетями из любой точки мира с использованием технологий удаленного доступа. Оно позволяет администраторам и IT-специалистам удаленно управлять серверами, настраивать программное обеспечение, выполнять диагностику и ремонт оборудования, мониторить работу сетей и систем. Визуальная схема представлена на рисунке 1.1.

Рисунок 1.1 – Визуальная схема удаленного администрирования

Системное администрирование – управление компьютерными системами, в том числе: операционными, графическими, базами данных, а также создание, настройка и обеспечение работоспособности локальных сетей. Локальная вычислительная сеть – компьютерная сеть, покрывающая обычно относительно небольшую территорию или небольшую группу зданий (дом, офис, фирму, институт). Сетевая операционная система – операционная система со встроенными возможностями для работы в компьютерных сетях.

К таким возможностям можно отнести:

1) поддержку сетевого оборудования
2) поддержку сетевых протоколов
3) поддержку протоколов маршрутизации
4) поддержку фильтрации сетевого трафика
5) поддержку доступа к удалённым ресурсам
6) поддержку сетевых протоколов авторизации
7) наличие в системе сетевых служб

Примеры сетевых операционных систем:

1) *Novell NetWare*

2) *Microsoft Windows* (95, *NT* и более поздние)

3) Различные *UNIX* системы, такие как *Solaris, FreeBSD*

4) Различные *GNU/Linux* системы

Главными задачами являются разделение ресурсов сети (например, дисковые пространства) и администрирование сети. С помощью сетевых функций системный администратор определяет разделяемые ресурсы, задаёт пароли, определяет права доступа для каждого пользователя или группы пользователей. Существуют специальные сетевые ОС, которым приданы функции обычных систем обычные ОС, которым приданы сетевые функции. Сегодня практически все современные ОС имеют встроенные сетевые функции [1].

1.2 Основные понятия, технологии и инструменты

Основными технологиями, используемыми при удаленном администрировании, являются:

1) *Remote Desktop Protocol (RDP)* – протокол удаленного рабочего стола, позволяющий получить удаленный доступ к рабочему столу операционной системы;

2)*Virtual Private Network (VPN)* – виртуальная частная сеть, обеспечивающая безопасное соединение между удаленными компьютерами через общедоступную сеть (интернет);

3) *Secure Shell (SSH)* – протокол удаленного управления операционными системами *UNIX* и *Linux*;

4) *Web-based Remote Access* – удаленный доступ через веб-браузер.

Для удаленного администрирования могут использоваться различные инструменты и программы, такие как *TeamViewer*, *Remote Desktop Connection*, *VNC, LogMeIn* и другие.

Одним из главных преимуществ удаленного администрирования является гибкость и мобильность. Администраторы могут работать из любой точки мира, что позволяет быстро реагировать на проблемы и устранять их. Однако, удаленное администрирование также имеет недостатки, связанные с безопасностью и скоростью передачи данных, которые будут рассмотрены в соответствующих разделах работы [2].

1.3 Преимущества и недостатки удаленного администрирования

Удаленное администрирование имеет свои преимущества и недостатки, которые необходимо учитывать при использовании этой технологии. Среди главных преимуществ можно выделить:

1) гибкость и мобильность: удаленное администрирование позволяет администраторам работать из любой точки мира, что делает процесс

управления системами более гибким и мобильным;

2) экономия времени и денег: удаленное администрирование позволяет сократить затраты на командировки и уменьшить время на решение проблем;

3) быстрое реагирование на проблемы: удаленное администрирование позволяет быстро реагировать на проблемы и устранять их, что позволяет уменьшить время простоя систем.

Однако, удаленное администрирование также имеет некоторые недостатки, которые могут вызвать проблемы в процессе работы. Среди них:

1) проблемы с безопасностью: удаленный доступ к системам может создавать угрозу для безопасности данных, что требует дополнительных мер по защите информации;

2) проблемы со скоростью передачи данных: удаленный доступ может быть замедленным из-за ограниченной пропускной способности сети, что может привести к снижению производительности и эффективности работы;

3) проблемы с оборудованием и программным обеспечением: удаленное администрирование требует использования специального оборудования и программного обеспечения, что может вызвать дополнительные затраты на его приобретение и настройку.

В целом, удаленное администрирование является эффективным инструментом управления системами и сетями, но при его использовании необходимо учитывать и преимущества, и недостатки, которые могут возникнуть в процессе работы [3].

1.4 Инструменты удаленного администрирования

Для удаленного администрирования существует множество инструментов, каждый из которых имеет свои преимущества и недостатки. Среди наиболее распространенных инструментов можно выделить:

1) *TeamViewer*: простой и удобный инструмент, позволяющий удаленно управлять компьютером, осуществлять перенос файлов и работать с удаленным рабочим столом. *TeamViewer* поддерживает множество операционных систем и может использоваться для удаленной поддержки пользователей и администрирования серверов. Интерфейс программы изображен на рисунке 1.2;

2) *Remote Desktop Connection*: встроенный инструмент операционной системы Windows, который позволяет удаленно управлять компьютером через сеть. *Remote Desktop Connection* поддерживает множество функций, включая работу с файлами и устройствами, а также позволяет подключаться к удаленному рабочему столу. Интерфейс программы изображен на рисунке 1.3;

3) *VNC (Virtual Network Computing)*: инструмент, который позволяет управлять компьютером через интернет с помощью удаленного доступа к его экрану и клавиатуре. *VNC* поддерживает множество операционных систем и является открытым и бесплатным инструментом. Интерфейс программы изображен на рисунке 1.4;

4) *PowerShell Remoting*: инструмент для удаленного управления компьютерами на базе *Windows* с помощью *PowerShell*. *PowerShell Remoting* позволяет выполнять команды и скрипты на удаленных компьютерах, а также управлять настройками системы. Пример работы представлен на рисунке 1.5;

5) *SSH (Secure Shell)*: инструмент для удаленного управления компьютерами на базе *Unix* и *Linux*. *SSH* позволяет работать с удаленной командной строкой, переносить файлы и управлять настройками системы. Схема взаимодействия с помощью *Secure Shell* представлена на рисунке 1.6;

Каждый из этих инструментов имеет свои преимущества и недостатки, которые необходимо учитывать при выборе оптимального инструмента для удаленного администрирования. Например, инструменты, такие как *TeamViewer* и *Remote Desktop Connection*, хорошо подходят для работы с *Windows*-системами, но могут не быть подходящими для *Unix* и *Linux*-систем. Однако, *SSH* и *PowerShell Remoting* являются универсальными инструментами, которые могут использоваться для удаленного управления большинством типов систем [4].

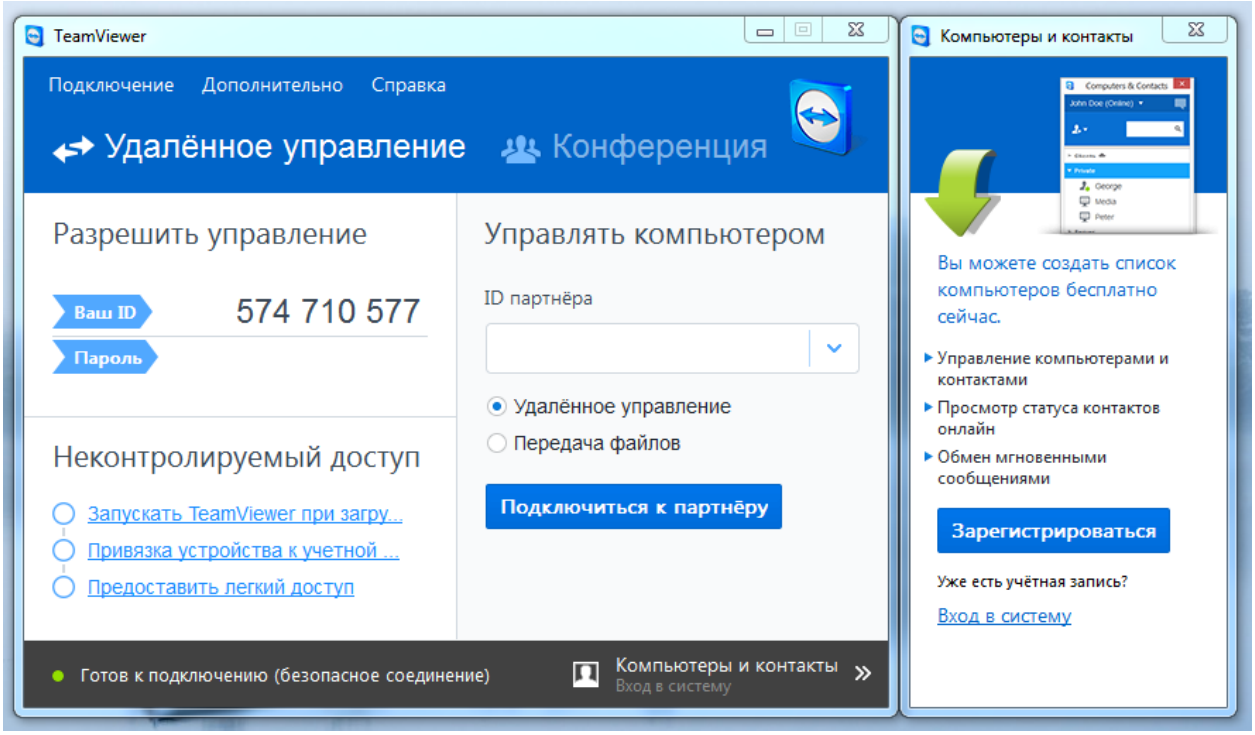

Рисунок 1.2 – Интерфейс программы *TeamViewer*

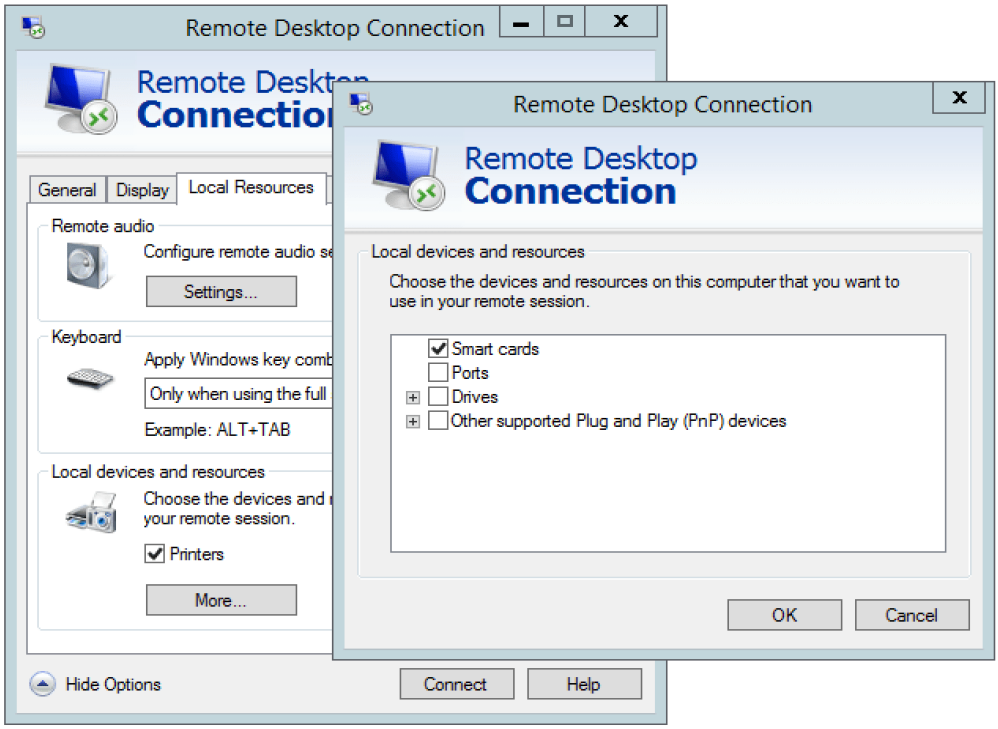

Рисунок 1.3 – Интерфейс программы *Remote Desktop Connection*

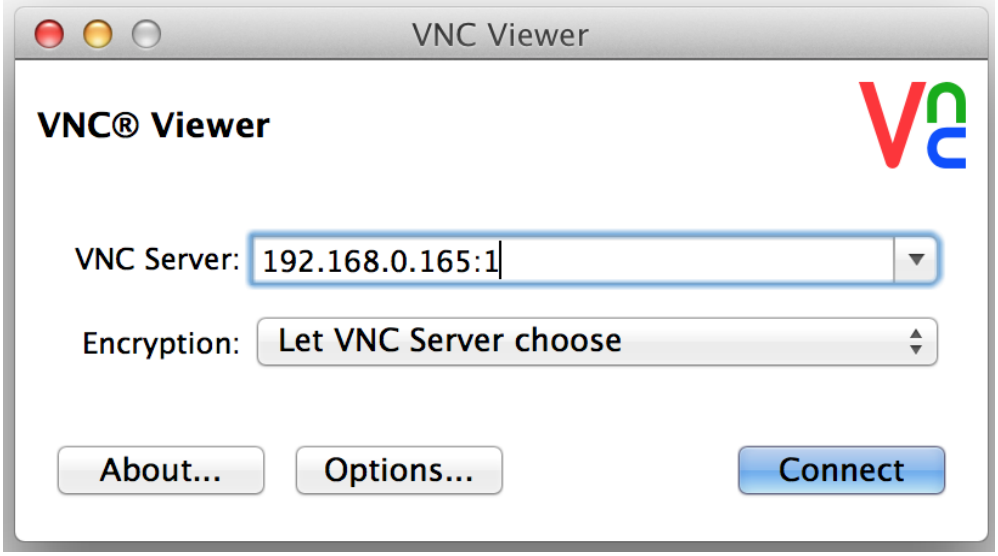

Рисунок 1.4 – Интерфейс программы *Virtual Network Computing*

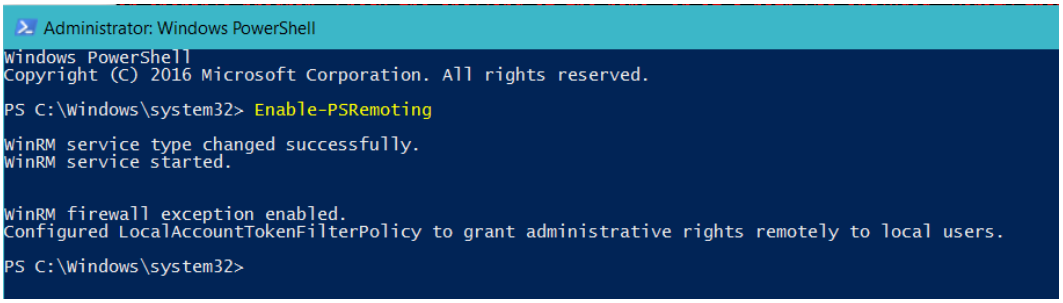

Рисунок 1.5 – Пример работы с *PowerShell Remoting*

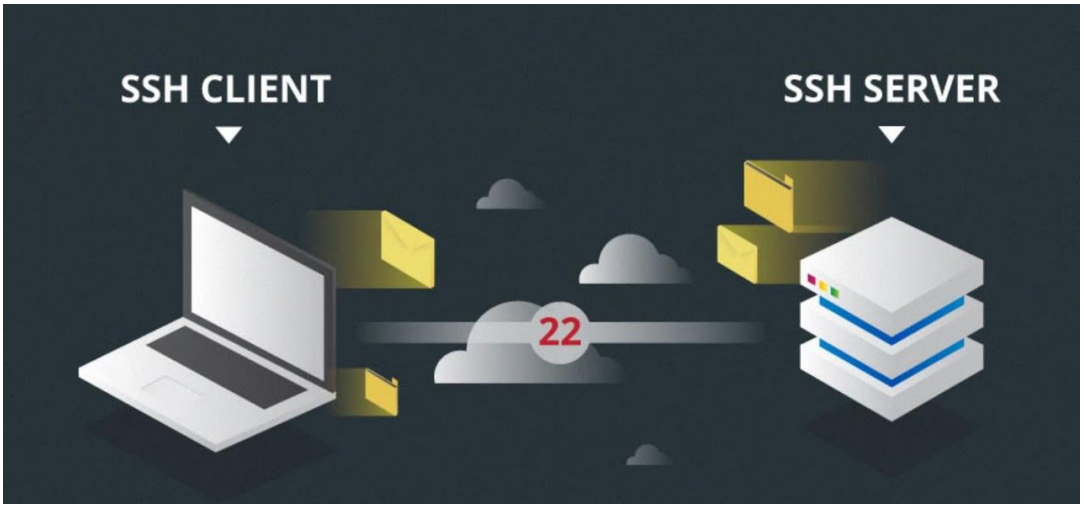

Рисунок 1.6 – Схема взаимодействия с помощью *Secure Shell*

1.5 Риски и способы их минимизации

При удаленном администрировании существуют риски, связанные с безопасностью и доступностью системы. Некоторые из основных рисков удаленного администрирования включают:

1) уязвимости безопасности: удаленный доступ к системе может стать дырой в безопасности, позволяя злоумышленникам получить доступ к конфиденциальным данным и настройкам системы. Для минимизации этого риска необходимо использовать защищенные протоколы и соединения, а также правильно настроить и защитить систему;

2) сбои сети: удаленное администрирование может быть затруднено при сбоях в сети, что может привести к потере данных или недоступности системы. Для минимизации этого риска необходимо использовать надежные соединения и инструменты, а также резервные копии данных;

3) нехватка ресурсов: удаленный доступ к системе может потребовать большого объема ресурсов, таких как скорость интернет-соединения и мощность процессора. Для минимизации этого риска необходимо правильно настроить систему и использовать оптимизированные инструменты.

Для минимизации рисков удаленного администрирования необходимо

использовать надежные инструменты и соединения, а также правильно настроить и защитить систему. Некоторые из способов минимизации рисков удаленного администрирования включают:

1) использование защищенных протоколов и соединений, таких как *SSL* или *SSH*;

2) настройка брандмауэров и других систем защиты для предотвращения несанкционированного доступа к системе;

3) создание резервных копий данных для восстановления системы в случае сбоя;

4) оптимизация настроек системы и инструментов для минимизации использования ресурсов.

В целом, удаленное администрирование может быть безопасным и эффективным инструментом при правильном использовании и настройке инструментов и систем.

1.6 Примеры практического применения

Удаленное администрирование может использоваться в различных сферах, включая бизнес, образование и здравоохранение. Вот несколько примеров практического применения удаленного администрирования:

1) удаленное управление устройствами: удаленное управление устройствами позволяет администраторам управлять компьютерами, ноутбуками, планшетами и смартфонами, которые находятся вне офиса или в другой стране. Это может быть полезно для компаний, которые имеют мобильных сотрудников или филиалы в разных городах;

2) удаленное обучение: удаленное администрирование также может использоваться для обучения студентов и сотрудников. Виртуальные классы и обучающие программы позволяют проводить уроки и семинары удаленно, что увеличивает доступность обучения и позволяет экономить время и деньги;

3) Удаленное здравоохранение: удаленное администрирование может использоваться в медицинских учреждениях для удаленного консультирования пациентов, управления медицинским оборудованием и мониторинга пациентов из дома. Это позволяет улучшить качество медицинского обслуживания и уменьшить затраты на здравоохранение.

Это лишь некоторые примеры того, как удаленное администрирование может быть применено на практике. С развитием технологий и увеличением числа удаленных работников, удаленное администрирование становится все более важным инструментом для управления бизнесом и обеспечения эффективной работы [5].

2 ПЛАТФОРМА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

2.1 Характеристика программного обеспечения

Платформа – это комплекс аппаратных и программных средств, на котором функционирует программное обеспечение пользователя ЭВМ. Основа аппаратной платформы *(hardware*-платформы*)* – процессор. Тип процессора определяет архитектуру аппаратных средств – аппаратную платформу, т. е. тип и характеристики компьютера.

Системное программное обеспечение – это «программная оболочка» аппаратных средств, предназначенная для отделения остальных программ от непосредственного взаимодействия с оборудованием и организации процесса обработки информации в компьютере. Прикладное программное обеспечение предназначено для решения определенных задач пользователя. К системному программному обеспечению относятся такие типы программ, как операционные системы, различные сервисные средства, функционально дополняющие возможности операционных систем, инструментальные средства (системы управления базами данных, программирования, оболочки экспертных систем).

Основная компонента системного программного обеспечения – операционная система выполняет следующие функции:

1) организация работы компьютера, при которой возможно одновременное выполнение нескольких программ пользователя;

2) организация хранения программ и данных пользователя на носителях информации и, возможно, санкционирование доступа к этой информации;

3) обеспечение взаимодействия с пользователем на основе графического интерфейса;

4) обеспечение сетевых возможностей, т. е. возможности доступа к информации, хранимой в памяти другого компьютера локальной или глобальной сети.

В курсовом проекте платформой программного обеспечения будет использоваться *Linux.*

2.2 Операционная система Windows

Операционная система *Windows*:

1) является 32-разрядной;

2) поддерживает вытесняющую многозадачность;

3) работает на разных аппаратных архитектурах и обладает способностью к сравнительно легкому переносу на новые аппаратные архитектуры;

4) поддерживает работу с виртуальной памятью;

5) является полностью реентерабельной;

Компьютерная программа в целом или её отдельная процедура

называется реентерабельной (от англ. *reentrant* – повторно входимый), если она разработана таким образом, что одна и та же копия инструкций программы в памяти может быть совместно использована несколькими пользователями или процессами. При этом второй пользователь может вызвать реентерабельный код до того, как с ним завершит работу первый пользователь и это как минимум не должно привести к ошибке, а в лучшем случае не должно вызвать потери вычислений (то есть не должно появиться необходимости выполнять уже выполненные фрагменты кода).

Реентерабельный код имеет следующие преимущества:

1) хорошо масштабируется с мультипроцессорной обработкой;
2) является распределенной вычислительной платформой;
3) защищена как от внутренних сбоев, так и от внешних;
4) пользовательский интерфейс совместим с предыдущими версиями;
5) обладает высокой производительностью независимо от платформы;
6) обеспечивает простоту адаптации за счет поддержки *Unicode*;
7) поддерживает многопоточность и объектную модель. [9]

2.3 Общее описание структуры системы

Архитектура ОС *Windows* претерпела ряд изменений в процессе эволюции. Первые версии системы имели микроядерный дизайн, основанный на микроядре *Mach*, которое было разработано в университете Карнеги-Меллона. Архитектура более поздних версий системы микроядерной уже не является. Причина заключается в постепенном преодолении основного недостатка микроядерных архитектур – дополнительных накладных расходов, связанных с передачей сообщений.

По мнению специалистов Microsoft, чисто микроядерный дизайн коммерчески невыгоден, поскольку неэффективен.

Поэтому большой объем системного кода, в первую очередь управление системными вызовами и экранная графика, был перемещен из адресного пространства пользователя в пространство ядра и работает в привилегированном режиме.

В результате в ядре ОС *Windows* переплетены элементы микроядерной архитектуры и элементы монолитного ядра (комбинированная система).

Сегодня микроядро ОС *Windows* слишком велико (более 1 Мб), чтобы носить приставку «микро».

Основные компоненты ядра *Windows NT* располагаются в вытесняемой памяти и взаимодействуют друг с другом путем передачи сообщений, как и положено в микроядерных операционных системах.

В тоже время все компоненты ядра работают в одном адресном пространстве и активно используют общие структуры данных, что свойственно операционным системам с монолитным ядром.

Высокая модульность и гибкость первых версий *Windows NT* позволила успешно перенести систему на такие отличные от *Intel* платформы, как *Alpha*

(корпорация *DEC*), *Power PC* (*IBM*) и *MIPS* (*Silicon Graphic*). Более поздние версии ограничиваются поддержкой архитектуры *Intel x86*.

Упрощенная схема архитектуры, ориентированная на выполнение *Win32*-приложений, показана на рисунке 2.1.

Рисунок 2.1 – Упрощенная архитектурная схема ОС *Windows*

ОС *Windows* состоит из компонентов, работающих в режиме ядра и пользователя. Несмотря на миграцию системы в сторону монолитного ядра она сохранила некоторую структуру. В схеме, представленной на рисунке 1.10, отчетливо просматриваются несколько функциональных уровней, каждый из которых пользуется сервисами более низкого уровня.

Задача уровня абстрагирования от оборудования (*hardware abstraction layer, HAL*) – скрыть аппаратные различия аппаратных архитектур для потенциального переноса системы с одной платформы на другую. *HAL* предоставляет выше лежащим уровням аппаратные устройства в абстрактном виде, свободном от индивидуальных особенностей. Это позволяет изолировать ядро, драйверы и исполнительную систему ОС *Windows* от специфики оборудования (например, от различий между материнскими платами).

Ядром обычно называют все компоненты ОС, работающие в привилегированном режиме работы процессора или в режиме ядра. Корпорация *Microsoft* называет ядром (*kernel*) компонент, находящийся в невыгружаемой памяти и содержащий низкоуровневые функции операционной системы, такие, как диспетчеризация прерываний и исключений, планирование потоков и другие. Оно также предоставляет набор процедур и базовых объектов, применяемых компонентами высших уровней.

Ядро и *HAL* являются аппаратно-зависимыми и написаны на языках Си и ассемблера.

Основные достоинства приложения *UWP*.

Безопасность. Приложения *UWP* объявляют, к каким ресурсам устройства и данным они осуществляют доступ. Пользователь должен разрешить такой доступ.

Возможность использовать общий *API* на всех устройствах под управлением *Windows*.

Возможность использования возможностей отдельных устройств и адаптации пользовательского интерфейса к разным размерам экранов, разрешениям и плотностям точек.

Доступность в *Microsoft Store*, на всех устройствах (или только тех, которых вы укажете), работающих под управлением *Windows 10* или *Windows 11*. В *Microsoft Store* предусмотрено несколько способов монетизировать ваше приложение.

Возможность устанавливаться и удаляться без риска для компьютера или «деградации» ПО.

Увлекательность: возможность использовать живые плитки, *push*-уведомления и пользовательские действия, взаимодействующие с временной шкалой *Windows* и функцией «Продолжить с места остановки» Кортаны, для поддержания интереса пользователей к приложению.

Программируемый на *C#, C++, Visual Basic* и *JavaScript*. Для пользовательского интерфейса можно использовать *WinUI, XAML, HTML* или *DirectX* [6].

2.4 Язык программирования Python

Python – это высокоуровневый интерпретируемый язык программирования, который был создан в конце 1980-х годов в Нидерландах Гвидо ван Россумом. Он имеет простой и понятный синтаксис, что делает его очень популярным среди начинающих программистов и обеспечивает быстрое и легкое создание программ.

Python используется для различных задач, таких как научные исследования, веб-разработка, автоматизация и многие другие. Он также имеет множество сторонних библиотек, таких как *NumPy, Pandas, TensorFlow, Django, Flask* и многие другие, что делает его очень гибким и мощным инструментом для разработки программного обеспечения [7].

2.5 Основные особенности языка Python

Интерпретируемый язык: *Python* – это интерпретируемый язык, что означает, что он не компилируется в машинный код, а выполняется интерпретатором.

Кросс-платформенность: *Python* поддерживает большое количество операционных систем, таких как *Windows, Linux, macOS* и многие другие.

Простой синтаксис: *Python* имеет простой и читаемый синтаксис, что

делает его легко изучаемым для начинающих программистов.

Обширная стандартная библиотека: *Python* поставляется с обширной стандартной библиотекой, которая содержит множество модулей, что позволяет программистам решать широкий спектр задач.

Динамическая типизация: В *Python* не нужно объявлять типы данных, что облегчает и ускоряет процесс написания кода.

Объектно-ориентированное программирование: *Python* поддерживает объектно-ориентированное программирование, что позволяет создавать классы и объекты, что упрощает разработку крупных проектов [8].

Основные возможности языка представлены на рисунке 2.2.

Рисунок 2.2 – Основные возможности использования *Python*

2.6 Среда разработки PyCharm

PyCharm – это интегрированная среда разработки (*IDE*) для языка программирования *Python*, разработанная компанией *JetBrains*. Эта среда предназначена для облегчения процесса разработки, тестирования и отладки приложений *Python*. Интерфейс приложения представлен на рисунке 2.3.

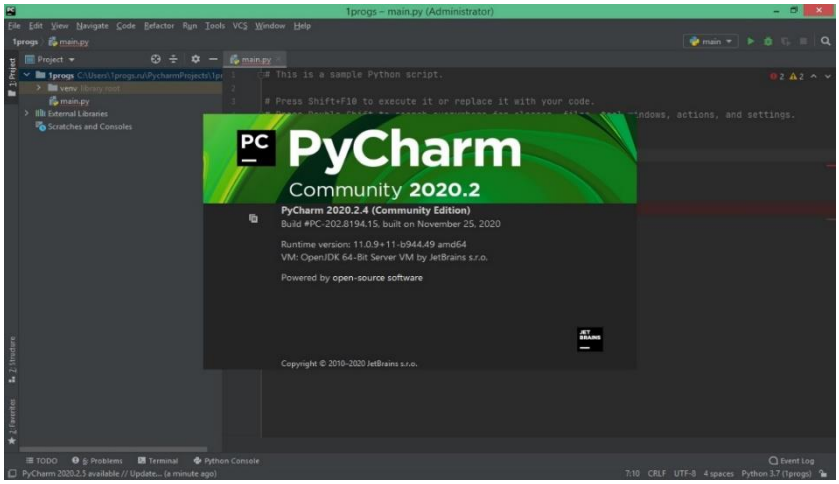

Рисунок 2.3 – Интерфейс *IDE PyCharm*

PyCharm предоставляет множество функций и инструментов, таких как автодополнение кода, проверка синтаксиса, управление версиями, отладка, поддержка виртуальных сред, интеграция с *Git* и многими другими. Кроме того, *PyCharm* имеет встроенную поддержку для популярных фреймворков, таких как *Django*, *Flask* и *Pyramid*.

Среда разработки *PyCharm* доступна в двух версиях: *Community Edition* (бесплатная версия) и *Professional Edition* (платная версия). *Professional Edition* предоставляет расширенные возможности, такие как поддержка различных языков программирования, инструменты для разработки веб-приложений, анализ кода и многие другие.

PyCharm поддерживает операционные системы *Windows*, *macOS* и *Linux*, и может быть загружен с официального сайта *JetBrains* [9].

2.7 Описание Kivy Framework

Kivy – это бесплатный и открытый кроссплатформенный фреймворк для разработки мультимедийных приложений на языке *Python*, который используется для создания графического интерфейса пользователя (*GUI*) и мультимедийных приложений. Архитектура *Kivy* приведена на рисунке 2.4.

Ниже приведены основные возможности *Kivy*.

Графический интерфейс пользователя. *Kivy* предоставляет готовые элементы интерфейса, такие как кнопки, текстовые поля, изображения и другие, которые могут быть легко настроены и адаптированы для вашего приложения.

Анимации и переходы. *Kivy* поддерживает мощную систему анимации и переходов, что позволяет создавать плавные и эффектные переходы между различными экранами и элементами вашего приложения.

Графика. *Kivy* использует библиотеку *OpenGL ES* для обеспечения быстрой и эффективной графики ваших приложений. Это позволяет создавать сложные и интерактивные графические приложения, такие как игры и

визуализации.

Мультимедиа. *Kivy* поддерживает воспроизведение аудио и видео, а также обработку графических форматов, таких как *JPEG* и *PNG*.

Мульти-тач. *Kivy* поддерживает мульти-тач на устройствах с сенсорными экранами, что позволяет создавать интерактивные приложения, такие как игры и мультимедиа-приложения.

Язык программирования *Python*. *Kivy* написан на языке программирования *Python*, который является популярным и легким в использовании языком. Это делает *Kivy* привлекательным для начинающих разработчиков, которые уже знакомы с *Python*.

Открытый и бесплатный. *Kivy* является открытым и бесплатным фреймворком с лицензией *MIT*, что позволяет использовать его в коммерческих и некоммерческих проектах.

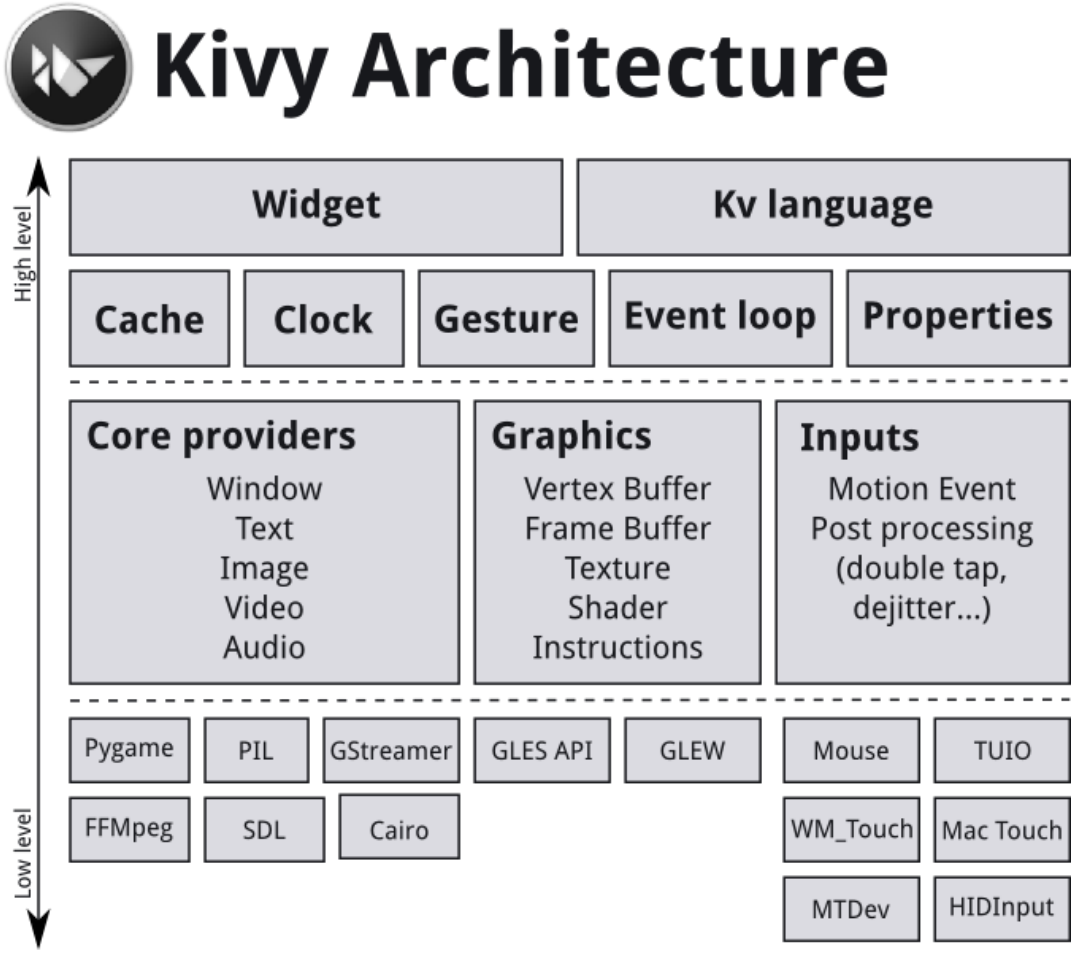

Рисунок 2.4 – Архитектура фреймворка *Kivy*

В целом, *Kivy* – это мощный и гибкий фреймворк для создания мультимедийных приложений на *Python* с возможностью создания красивого и интерактивного пользовательского интерфейса.

3 ТЕОРЕТИЧЕСКОЕ ОБОСНОВАНИЕ РАЗРАБОТКИ

3.1 Обоснование разработки программного продукта

Разработка приложения для удаленного администрирования может быть обоснована несколькими факторами, включая удобство, безопасность и эффективность управления удаленными устройствами и серверами.

Во-первых, удаленное администрирование позволяет управлять серверами и устройствами из любой точки мира, что обеспечивает гибкость и удобство в работе. Это особенно полезно для компаний, которые имеют филиалы в разных регионах и странах, так как это позволяет быстро реагировать на проблемы и устранять их.

Во-вторых, разработка приложения для удаленного администрирования может повысить уровень безопасности, так как администраторы могут управлять удаленными серверами и устройствами без необходимости физического доступа к ним. Это уменьшает вероятность кражи или повреждения оборудования, а также позволяет быстро реагировать на угрозы безопасности.

Наконец, удаленное администрирование позволяет эффективно использовать ресурсы и сократить затраты на обслуживание оборудования и инфраструктуры. Администраторы могут быстро реагировать на проблемы и устранять их, что позволяет снизить время простоя серверов и устройств.

Таким образом, разработка приложения для удаленного администрирования является важным шагом в повышении удобства, безопасности и эффективности управления удаленными серверами и устройствами.

3.2 Актуальность программ удаленного администрирования

Удаленное администрирование остается актуальным в современном мире, поскольку он обеспечивает несколько преимуществ в сравнении с традиционными методами администрирования.

Географический разброс. Удаленное администрирование позволяет администраторам управлять компьютерами и серверами, находящимися в разных местах, не выходя из своего офиса или дома. Это существенно упрощает процесс управления сетью или парка серверов и сокращает затраты на командировки.

Гибкость. Удаленное администрирование дает возможность администраторам быстро реагировать на изменения в сети или на сервере, не тратя время на поездки и не нарушая работу предприятия.

Безопасность. Удаленное администрирование обеспечивает высокий уровень безопасности данных, поскольку администраторы могут работать на серверах и компьютерах, не оставляя следов на местных дисках.

Экономичность. Удаленное администрирование позволяет экономить на затратах на транспорт, а также на содержании помещений и оборудования, что особенно важно для небольших предприятий.

Непрерывность работы. Удаленное администрирование позволяет администраторам быстро реагировать на проблемы и устранять их до того, как они приведут к остановке работы сервера или сети.

Таким образом, удаленное администрирование остается актуальным и эффективным методом управления серверами и сетями в настоящее время.

3.3 Протокол TCP и плюсы его использования

Протокол *TCP* (*Transmission Control Protocol*) – это основной протокол, который обеспечивает надежную передачу данных в сетях *TCP/IP*. Он работает на уровне транспортного протокола (*Transport Layer*) в модели *OSI* и обеспечивает установление, поддержание и завершение соединения между двумя приложениями, которые обмениваются данными через Интернет.

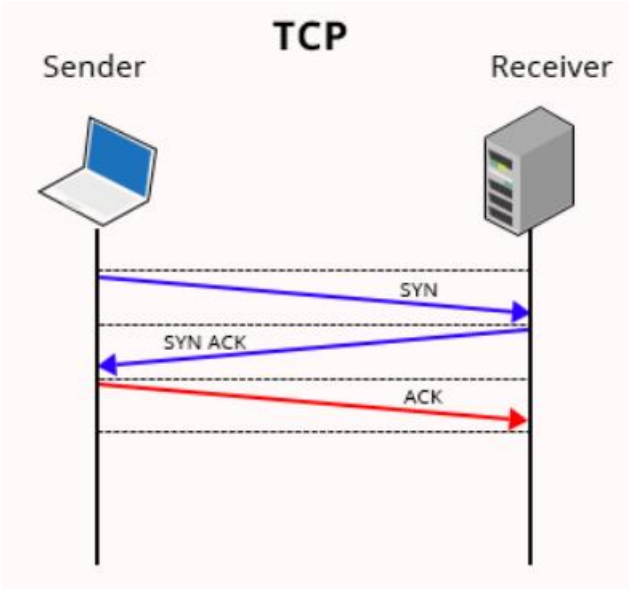

Рисунок 3.1 – Схема протокола *TCP*

Протокол *TCP* используется для передачи больших объемов данных и обеспечивает гарантированную доставку данных. Данные, передаваемые через протокол TCP, делятся на сегменты и каждый сегмент передается по отдельности. При передаче данных TCP использует механизм контроля ошибок, который гарантирует, что все сегменты будут доставлены в том порядке, в котором они были отправлены, и без потерь или ошибок.

Его основными преимуществами являются:

1) Гарантированная доставка. *TCP* гарантирует, что данные будут доставлены в том порядке, в котором они были отправлены, и что они будут доставлены без ошибок. Если какие-либо данные не будут доставлены, *TCP*

автоматически повторит попытку передачи;

2) управление потоком. *TCP* автоматически управляет скоростью передачи данных, чтобы избежать перегрузки сети. Это позволяет избежать потери данных в случае, если скорость передачи данных превышает возможности получателя;

3) установление соединения. *TCP* использует процедуру "трех рукопожатий" для установления соединения между отправителем и получателем. Это гарантирует, что обе стороны готовы к передаче данных, и что данные будут передаваться только между ними;

4) надежность: *TCP* обеспечивает надежность передачи данных, благодаря механизму контроля ошибок и повторной передачи данных в случае их потери;

5) мультиплексирование: *TCP* позволяет одному устройству устанавливать несколько соединений и передавать данные по каждому из них.

В целом, протокол *TCP* обеспечивает высокую степень надежности и управляемости при передаче данных в Интернете, что делает его основным протоколом для большинства приложений и сервисов [10].

4 ПРОЕКТИРОВАНИЕ ФУНКЦИОНАЛЬНЫХ ВОЗМОЖНОСТЕЙ ПРОГРАММЫ

4.1 Описание структуры программы

Программный продукт содержит следующие директории, изображенные на рисунке 4.1:

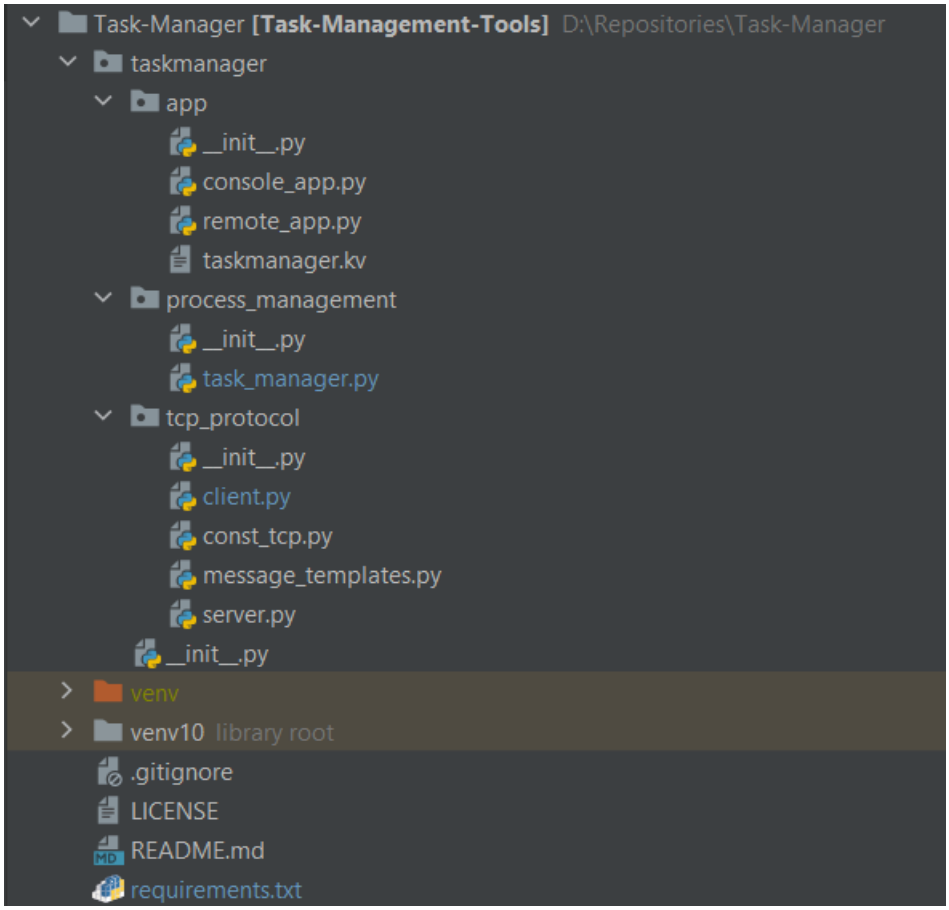

Рисунок 4.1 – Архитектура программного продукта

Папка *app* содержит файлы для запуска. *console_app.py* запускается на компьютере, который будет выполнять запросы от удаленного пользователя. *remote_app.py* запускается на компьютере, который будет производить управление над другим устройством. Также в директории находится файл *taskmanager.kv*, в котором прописан интерфейс приложения.

Подключение происходит при помощи протокола *TCP*, реализация которого выполнена в директории *tcp_protocol*. Там находятся стандартные файлы, свойственные данному типу подключения, а именно client и server, а также файлы, содержащие константные значения, такие как выводимые в консоль сообщения и *IP*, *PORT*.

4.2 Принцип работы

Для начала работы, следует запустить в соответствующем порядке следующие файлы из директории *app*: *console_app.py*, *remote_app.py*.

Первый файл запустить консоль, где впоследствии будут выводиться данные о производимых на удаленном устройстве операциях. Помимо этого, сами операции будут выполнятся на экран.

Второй файл следует запускать на другом устройстве. При наличии только одного устройства, для работы программы можно просто запустить два файла поочередно. После запуска *remote_app.py* появится оконное приложение, показанное на рисунке 4.2.

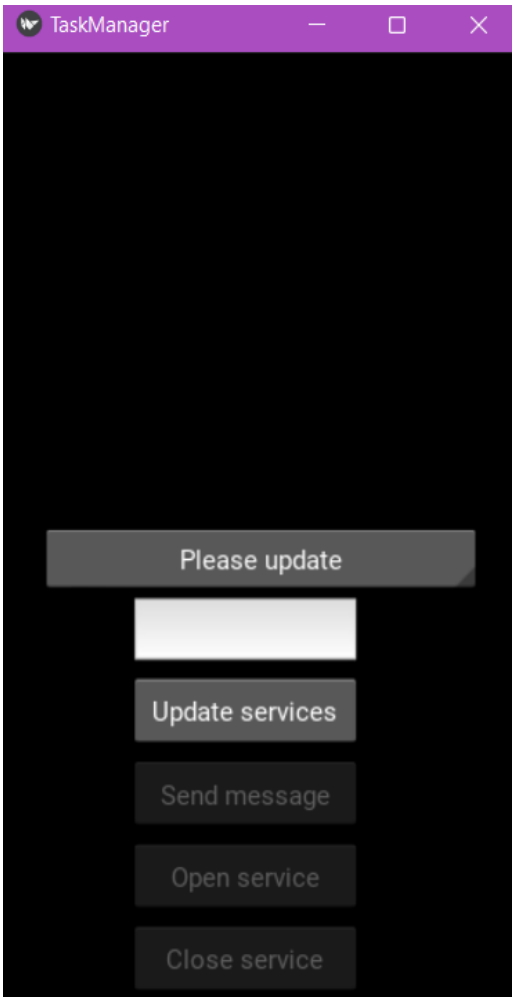

Рисунок 4.2 – Интерфейс программного продукта

При запуске программа, прежде чем выполнить любое действие из возможных, потребует обновить сервисы. Это нужно для определения доступных программ, имеющихся на удаленном компьютере, которые можно запустить/закрыть.

После обновления появится список доступных на компьютере программ. Данный список изображен на рисунке 4.3. Можно заметить, что стали активны ранее закрытые действия.

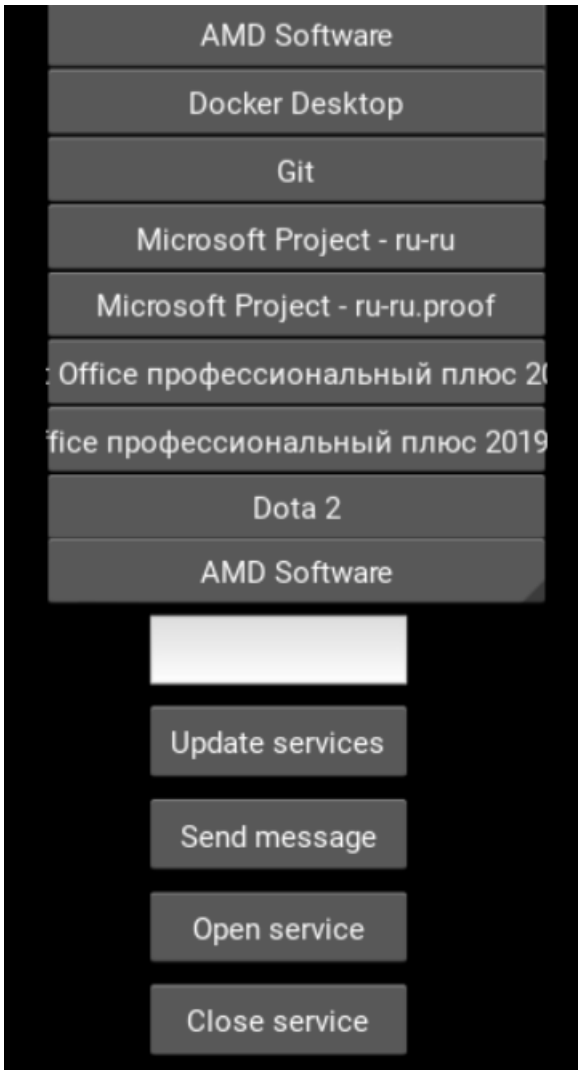

Рисунок 4.3 – Список доступных программ

Выполнение рассматривается на примере *Docker Desktop*. Запустим программу с другого устройства с помощью команды *open service*. Результат работы изображен на рисунке 4.4. Помимо открытия самой программы, в консоль выводятся сведения о выполнении команды, о чем было упомянуто выше. Чтобы закрыть приложение, следует выполнить команду *close service*.

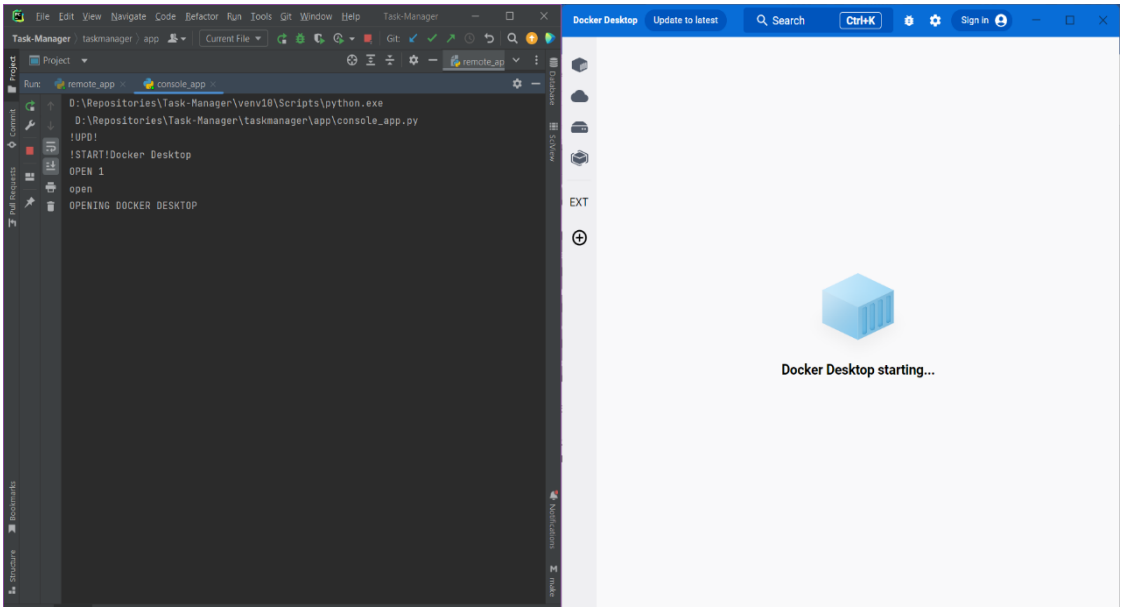

Рисунок 4.4 – Запуск программы

Программа имеет возможность отправки сообщения на удаленный компьютер. Результат выполнения данной операции изображен на рисунке 4.5.

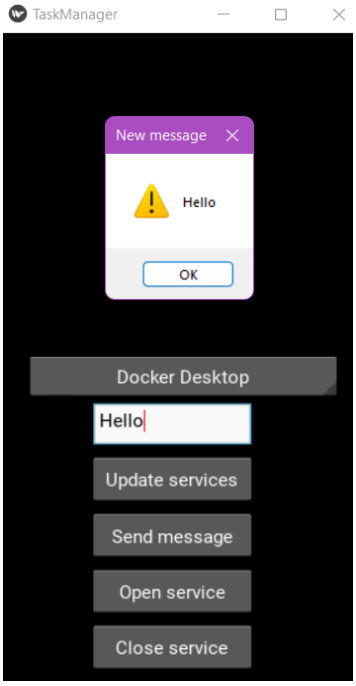

Рисунок 4.5 – Отправка сообщения

Для отправки сообщения следует написать текст в пустом окошке и нажать кнопку *send message*. У пользователя, над устройством которого ведется управление, появится всплывающее окошко *new message* с соответствующим сообщением.

ЗАКЛЮЧЕНИЕ

Курсовая работа на тему "Удаленное администрирование" позволила рассмотреть основные аспекты удаленного управления IT-системами и рассмотреть примеры его практического применения.

Удаленное администрирование – это эффективный и удобный инструмент для управления IT-системами. Оно позволяет администраторам мониторить и управлять сетевой инфраструктурой, серверами, безопасностью и доступностью данных, а также обучать и поддерживать пользователей. Однако, важно учитывать возможные угрозы безопасности при удаленном администрировании и использовать соответствующие меры защиты. В целом, удаленное администрирование является незаменимым инструментом в современной IT-отрасли, который помогает повысить эффективность работы и сократить расходы на обслуживание IT-инфраструктуры.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Удаленное администрирование [Электронный ресурс]. – Режим доступа: https://uznayvse.ru/voprosyi/chto-takoe-udalennoe-administrirovanie-72336.html. - Дата доступа: 04.03.2023.

[2] Основное преимущество удаленного администрирования [Электронный ресурс]. – Режим доступа: https://shindler.ru/uslugi/radmin. – Дата доступа: 07.03.2023.

[3] Программы удаленного администрирования [Электронный ресурс]. – Режим доступа: https://www.datasystem.ru/udalennoe-administrirovanie/. – Дата доступа: 10.03.2023.

[4] Что такое удаленное администрирование [Электронный ресурс]. – Режим доступа: https://www.globalit.ru/services/it-outsourcing/remote-administration-computers-and-servers.html. – Дата доступа: 10.03.2023.

[5] Удаленное администрирование компьютеров, сетей и серверов [Электронный ресурс]. – Режим доступа: https://systech.ru/it-autsorsing/udalennoe-administrirovanie. – Дата доступа: 15.03.2023.

[6] Структура Windows [Электронный ресурс]. – Режим доступа: https://studopedia.su/1381317struktura-os-Windows.html. – Дата доступа: 16.03.2023.

[7] Язык программирования Python [Электронный ресурс]. – Режим доступа: https://pythonworld.ru/. – Дата доступа: 16.03.2023.

[8] Лутц М. Изучаем Python / Марк Лутц. – Вильямс, 2019. – 720 с.

[9] Общие сведения о PyCharm [Электронный ресурс]. – Режим доступа: https://www.jetbrains.com/ru-ru/pycharm/features/. – Дата доступа: 20.03.2023.

[10] Протокол TCP [Электронный ресурс]. – Режим доступа: https://www.ibm.com/docs/ru/aix/7.1?topic=management-transmission-control-protocolinternet-protocol. – Дата доступа: 10.04.2023.

ПРИЛОЖЕНИЕ А
(обязательное)
Исходный код программы

Исходный код программы находится в открытом доступе в репозитории. Ссылка: https://github.com/lizzzon/Task-Manager.