Проектная работа

«Приложение для объединения двух мониторов»

Автор работы: Воронов Никита Рустамович

Научный руководитель: Сергиенко Антон Борисович

Описание:

Проект посвящен созданию приложения, реализованном на языке программирования Python, которое способно не просто обмениваться данными, а способное получать, отправлять, обрабатывать их в реальном времени.

Цель проекта:

Создать полностью рабочую версию приложения: реализовать серверную и клиентскую часть. Изучить типы подключения, понять основной принцип обмена данными между двумя устройствами, путем отправки необходимых данных на клиентско-серверную часть.

Введение.

В современном мире проблема обмена информацией и данными уже давно была решена, мы можем хранить свои личные файлы в облаке, общаться на расстоянии, решать проблемы удаленно. Но можете ли вы быть уверены в безопасности своих файлов?

В начале своей работы я поставил для себя цель ─ создать безопасное приложение для обмена данными без вмешательства крупных кампаний, таких как Google, Яндекс, и другие. Конечно, можно использовать их сервисы для наиболее эффективного управления своей информацией, ведь у каждого есть один – два, а может даже больше, аккаунтов на этих платформах. Однако, пока пользователь хранит свои данные у этих корпораций, они могут беспрепятственно просмотреть или скопировать вашу личную информацию, не говоря уже о том, что доступ к вашему аккаунту могут получить посторонние люди. Конечно, вы можете перемещать свою личную и наиболее важную информацию на внешних носителях, но они так же не обеспечивают наибольшую безопасность, к тому же некоторые «похитители информации» уже способны собирать информацию и с них.

Основная задача моего приложения – трансляция экрана пользователя (клиента) на другой компьютер (сервер). Получение данных осуществляется таким образом, что может быть только одно подключение, что защищает от дополнительных ─ внешних подключений.

Описание:

Вся программа написана на языке программирования Python. Выбор языка программирования обусловлен тем, что python, на момент написания, находится на первых местах по популярности среди самостоятельных разработчиков. Python прост в освоении, это очень может помочь тем, кто захочет усовершенствовать код или убедиться в безопасности работы приложения. У данного языка огромное количество готовых решений различного рода задач, некоторые из них даже реализованы на других языках программирования. Более 137 тысяч библиотек для python позволяют не задумываться о необходимости отдельно прописывать все методы по работе как с данными(json для работы с одноименными данными, NumPy для работы с массивами, SQLite для работы с базами данных, так и с другими программами(Excel для сортировки под удобные таблицы, веб-браузер для работы со внешними источниками, и другие).

Для работы в сети python может использовать разные библиотеки, однако если нам требуется работать исключительно с IP-адресами пользователей, то лучший вариант – *socket.* Python socket – это модуль обеспечивает доступ к интерфейсу сокетов BSD (Berkeley Software Distribution). Он доступен на всех современных системах Unix, Windows, MacOS, Linux, и другие. Socket использует IP-адрес и программный порт, принимающий целое число от 0 до 65535, для обмена информацией и данными с системой.

Для визуализации всего программного интерфейса используется одна из популярнейших библиотек по созданию приложений – PyQt5. Это лучший вариант, подходящий для задач, поставленных в проекте.

Клиентский интерфейс представлен в виде двух строк, в которые можно вписать информацию об IP-адресе и порту подключения, информацию о которых должен выдать *сервер.* Серверная часть реализована таким образом, чтобы при подключении, сразу же открывалось новое окно, в котором отображается экран *клиента*, использующий “Клиентский интерфейс”. Обмен данными, а конкретно изображение экрана клиента, осуществляется путем создания соединения между клиентом и сервером по протоколу *AF\_INET*. Клиент, используя библиотеку Pillow для работы с изображениями, делает скрин своего экрана в автоматическом режиме и, не сохраняя его на устройстве, отправляет дальше по коду. После, используя библиотеку io, клиент преобразует полученное изображение в поток байтов. Это сделано для того, чтобы можно было отправить изображение любого качества, размера и формата( PNG, JPG, JPEG, и другие). Далее, благодаря беспроводному подключению, программа отправляет исходный поток байтов на сервер. На сервере используется тот же протокол подключения, что и клиент, отличается только то, что вместо отправки информации, он принимает её. После получения определенной команды о то, что подключение установлено, сервер начинает ждать данные. Когда он получил определенную информацию, то начинается процесс по *“досыланию данных”* . Из-за того, что поток байтов может быть очень большим, он может не вместиться в *“один вагон отправки”*, той информации, которой может отправить клиент за один раз, в таком случае нужно разделить исходный поток на несколько поменьше и отправлять уже их. Проверить полноценность потока можно с помощью обычного условия по проверки строки. После того как поток байт был полностью доставлен, требуется преобразовать его обратно в изображение, для этого вновь используем библиотеку io, но в этот раз уже сервер делает тоже самое, но наоборот – преобразует поток байтов в изображение. Изображение получено, но необходимо его показать. Когда изображение уже полностью преобразовано, сервер запускает новый *поток* для одновременной работы с несколькими подключениями. При этом используется уже встроенная библиотека python – *threading*. Создав новый поток, сервер создает окно, размеры которого определяются по формуле:

Sx, y = x/1,5 \* y/1,5

Где Sx, y  - размер окна в x, y пропорциях, X – ширина экрана, Y – высота экрана. Данные величина высчитываются с помощью функции *size()* из библиотеки pyautogui, которая создана для работы, как раз, с экранами.