ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБЩЕОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ШКОЛА № 358 МОСКОВСКОГО РАЙОНА САНКТ-ПЕТЕРБУРГА

Проект

«Разработка telegram бота на python для одновременного управления множеством компьютеров с помощью интуитивно понятных элементов управления»

Работу выполнил:

Ефременко Симеон,

учащийся 10 «Б» класса ГБОУ СОШ №358

Московского района г. Санкт-Петербурга

Руководитель:

Шутей К. А.

Санкт-Петербург

2023-2024 г.

**Содержание**

[Введение 3](#_Toc163436324)

[Актуальность 3](#_Toc163436325)

[Цель 3](#_Toc163436326)

[Задачи 3](#_Toc163436327)

[Практическая значимость 3](#_Toc163436328)

[Целевая аудитория 3](#_Toc163436329)

[1. Технологии, примененные в ходе выполнения проекта 4](#_Toc163436330)

[1.1. Telegram бот 4](#_Toc163436331)

[1.2. Технологии связи компьютеров 5](#_Toc163436332)

[1.2.1. Fastapi 5](#_Toc163436333)

[1.2.2. Websocket 6](#_Toc163436334)

[2. Структура 7](#_Toc163436335)

[2.1. Описание 7](#_Toc163436336)

[2.2. Клиентская программа 7](#_Toc163436337)

[2.3. Серверная программа 8](#_Toc163436338)

[2.4. Пользовательская программа 8](#_Toc163436339)

[2.4.1. База данных 8](#_Toc163436340)

[2.4.2. Модуль общения с сервером 9](#_Toc163436341)

[2.4.3. Взаимодействие с пользователем 9](#_Toc163436342)

[Заключение 11](#_Toc163436343)

[Список литературы 12](#_Toc163436344)

[Приложения 13](#_Toc163436345)

# Введение

## Актуальность

В контексте образования, школьные компьютерные классы играют ключевую роль в процессе обучения информатике и другим предметам. Однако, несмотря на их значимость, эффективное управление большим количеством компьютеров может представлять собой сложную задачу для преподавателей и администраторов.

Традиционные методы управления, такие как ручная настройка каждого компьютера, могут быть крайне трудоемкими и неэффективными. В этом контексте разработка telegram бота для автоматизации управления компьютерами представляется как важный шаг в совершенствовании образовательных процессов. Позволяя управлять компьютерами через мессенджер telegram, мы создаем удобный и доступный инструмент для преподавателей и администраторов, который значительно упрощает и оптимизирует их повседневную работу.

Такой подход не только снижает временные затраты на административные процессы, но и позволяет преподавателям и администраторам сосредоточиться на более важных аспектах образовательного процесса, таких как разработка курсов, организация уроков и поддержка учащихся в освоении информационных технологий. Таким образом, разработка telegram бота для управления компьютерами имеет большой потенциал в совершенствовании образовательной среды и повышении ее эффективности.

## Цель

Создать telegram бота на python для одновременного управления множеством компьютеров с помощью интуитивно понятных элементов управления.

## Задачи

1. Собрать информацию для создания telegram бота на python для одновременного управления множеством компьютеров с помощью интуитивно понятных элементов управления
2. Проанализировать полученную информацию
3. Создать и отладить клиентскую, серверную и пользовательскую программы

## Практическая значимость

Автоматизация массового управления компьютерами, что экономит множество сил и времени.

## Целевая аудитория

Продвинутые пользователи персональных компьютеров, педагоги образовательных учреждений.

# Технологии, примененные в ходе выполнения проекта

## Telegram бот

Часто в наше время программисты пишут свои *консольные утилиты*[[1]](#footnote-1), с помощью которых они могут сделать буквально что угодно, узнать прогноз погоды, узнать курс рубля, вести свой личный бюджет, все это легко можно сделать на python. Но если программист захочет сделать продукт массовым, то ему нужно добавить *gui*[[2]](#footnote-2), что является очень непростой задачей. И тут в силу вступают чат боты, облегчающие эту задачу, которые есть в большинстве мессенджеров, но я остановлюсь на telegram, ведь это один из самых популярных мессенджеров в мире.

Но что же такое эти ваши чат боты? Ну так вот чат бот – это ­роботизированный аккаунт в мессенджере, который запрограммирован на автоматическое совершение действий: продажа товаров, развлечение пользователей или поиск полезной информации. Также общение с ним происходит в основном в виде переписки, вы отправляйте ему сообщение, а он отвечает на это определенным образов, в зависимости от того, как вы его запрограммировали. Также в сообщении можно отправлять фото, видео, документы, геолокацию, стикеры, и многое другое.

Вообще чат боты имеют множество преимуществ перед приложениями с gui, и не только для разработчика, но и для пользователя, вот некоторые из них:

* Простота использования, все чат боты в telegram имеют одинаковый интерфейс, я имею ввиду, что используя чат ботов они всегда работают таким образом, что вы отправляйте сообщение, и бот на него отвечает (в основном).
* Красивый и простой интерфейс, telegram был создан в очень красивом и минималистичном стиле, и еще вы можете вручную его кастомизировать, например поставить на фон любимого котика, вместо стандартного варианта, или изменить стандартный синий цвет на белый
* Нет необходимости скачивать приложение и хранить его на устройстве
* Также все данные, которыми оперирует бот, находятся на отдельном сервере, что также позволяет вам использовать 1 бота со множества устройств, как с компьютеров, так и с телефонов. И еще ваши данные не привязаны к устройству, то есть если у вас есть chatgpt, но в качестве чат бота в telegram, то вы можете с легкостью продолжить диалог на компьютере, даже если вы начали его на телефоне

И в своем проекте для одновременного управления компьютерами я решил использовать чат бота telegram для упрощения разработки и для улучшения пользовательского опыта, так-как разработать красивый и интуитивно понятный интерфейс для приложения на телефоне, крайне сложная задача. И еще у меня в любом случае должен работать сервер для связывания всех компьютеров воедино, и для упрощения и централизации приложения чат бот, практически идеальный выбор. Также telegram боты можно программировать на множестве языков, на python, rust, go, js[[3]](#footnote-3), java и это далеко не все. По факту на любом языке, на котором можно выйти в сеть есть возможность написать telegram бота по средствам взаимодействия с telegram api. но напрямую отправлять запросы на сервера telegram, это очень сложно, поэтому для этого создали библиотеки[[4]](#footnote-4), на которых можно в разы проще реализовать сложную логику взаимодействия с telegram api. 1 из таких библиотек для python является aiogram, на нем можно писать быстрых, и эффективных ботов, буквально за десяток строк кода, заместо тысяч без. Aiogram не стал для мня чем-то новым, я знал его и до создания данного проекта, и также говоря про aiogram, я хотел бы рассказать что значит aio? Эта приставка означает, что эта библиотека написана асинхронно, что делает ее на порядок эффективнее множества аналогов. Сначала вам нужно дать определение асинхронного программирования, так вот асинхронное программирование — концепция программирования, которая заключается в том, что результат выполнения функции доступен не сразу, а через некоторое время в виде некоторого асинхронного (нарушающего обычный порядок выполнения) вызова. Теперь скажу своими словами, представьте, что у вас идет огромная нагрузка на вашего чат бота, им пользуются сотни человек одновременно, и он на каждое сообщение делает запрос в базу данных, чтобы что-то прочитать, что-то записать, и все это занимает какое-то время, тогда пользователи будут находится в некой “очереди”, на ответ.

## Технологии связи компьютеров

Первоначально самым сложным для меня было придумать, как можно связать множество компьютеров, чтобы все они могли синхронно выполнять какие-либо команды, для этого мне пришлось изучить 2 вещи, это фреймворк fastapi на python и технология websocket. Обо всем по порядку:

### Fastapi

Начнем с того, что такое фреймворк? Фреймворк – программная платформа, определяющая структуру программной системы; программное обеспечение, облегчающее разработку и объединение разных компонентов большого программного проекта. Но если говорить более простым языком, то это уже написанный кем-то код, для упрощения разработки, например с telegram ботами можно взаимодействовать через прямые запросы к серверу telegram, но никто так не делает, все используют фреймворк aiogram для написания ботов на python, но о нем позже. Fastapi, это фреймворк для создания сайтов, как aiogram фреймворк для создания сайтов. Это крайне легковесное решение, в сравнении с тем же django fastapi был разработан для создания api[[5]](#footnote-5) сайтов или маленьких, легковесных сайтов, не требующих очень сложной архитектуры. Также этот фреймворк крайне быстр по сравнению со своими аналогами, ниже я представлю результаты теста, проведенного мной, где можно увидеть скорость работы того или иного фреймворка, тестирование я проводил на своем личном компьютере. Также ради интереса я тестировал не только фреймворки на python, также были затронуты и другие языки, например rust и golang. А и суть теста в том, что я написал на 21 разном фреймворке сайт, который отправляет ip пользователя, и программа для тестирования засекает время, за которое сайт сможет отправить ip 1000 раз, и эта программа была написана в 2 вариантах, на языке python и на языке rust, суть в том, что rust быстрее python, но у него крайне сложный синтаксис, и также если вы хотите поподробнее ознакомится с результатами моего тестирования, то вот ссылка на мой репозиторий GitHub, где есть полная таблица результатов со всеми фреймворками, которые я тестировал <https://github.com/lixelv/framework-test>, и так как мы рассматриваем язык python для создания сайта, ввиду его простоты, я укажу результаты только для фреймворков на python в приложении 1.

Как мы можем заметить из самых популярных фреймворков (fastapi, flask, django) fastapi является самым быстрым, но также существуют фреймворки pyramid, bottle и tornado. Но они сложнее fastapi, и скорость fastapi меня более чем устраивает. Дальше мы поговорим о технологиях websocket и short polling

### Websocket

Websocket — протокол связи поверх TCP-соединения, предназначенный для обмена сообщениями между браузером и веб-сервером, используя постоянное соединение. Для чего мне собственно нужна эта технология? В общем я, создав альфа-версию моего проекта, по сути, еле работающий концепт, использовал другую технологию для связи клиентской части с серверной, short pooling. Давайте я вам объясню в чем тут собственно разница, и с какой проблемой я столкнулся, применяя short polling. Сначала дадим определение. Short polling – поддержание соединения браузером и веб-сервером за счет частой отправки http или https запросов на сервер, для получения ответа. В данном случае браузером я называю автоматизированную программу, которая имеет доступ к отправке запросов в сеть Интернет. За частую short polling используется для связи программы, которая только отправляет запросы, и сервера, который может принимать новые события для программы. Эту технологию очень просто реализовать, просто отправляй запрос на сервер каждые n секунд. Также эта технология встроена в фреймворке aiogram.

Но у нее есть 2 важных недостатка в моем случае:

* На серверную программу идет слишком большая нагрузка, потому что каждый раз создавать и разрывать подключение с клиентской программой – это очень ресурс затратно
* Чтобы уменьшить нагрузку на сервер все еще используя short polling нужно увеличить задержку, то есть компьютеры будут в разное время выполнять команду, что может сильно путать, и не так эффектно, как одновременное, практически мгновенное выполнение команды.

Поэтому я в поисках решения этих проблем я нашел технологию websocket. Ее преимущества в том, что, подключаясь по протоколу TCP компьютеры создают постоянное подключение к основному серверу, и как только нужно будет им выполнить какую-либо команду, они сразу же смогут ее принять и выполнить. Но у нее есть 1 сложность. Нужно создать дополнительную систему учета подключенных, компьютеров, чтобы сервер знал, куда отправлять команды. Но зато теперь можно было легко поадресно отправлять команды (то есть выключить только 1 компьютер, вместо все)

# Структура

## Описание

IT продукты в основном имеют в своем наборе больше 1 программы, например если я создам VPN сервис для телефонов, то у меня будет целых 3 программы, это:

1. Программа для пользователя, то есть приложение в Google Play
2. Программа для главного сервера, который находит для меня свободный прокси сервер по выбранному региону и соединяет меня с ним
3. Программа для прокси серверов

В моем же продукте я могу выделить 3 программы, это:

1. Клиентская, запускается на компьютерах, которыми управляет главный серверная программа
2. Серверная, это, по сути, сайт, с помощью которого я связываю все клиентские программы во едино, чтобы, отправив команду ее можно было перенаправить на все компьютеры, или на какой-то конкретный, также с помощью этой программы можно отслеживать какие компьютеры подключены, а какие нет
3. Пользовательская, это уже сам бот, с помощью которого можно будет перенаправлять команды на серверную программу. Тут должна быть реализована система интуитивно понятного ввода команд, но об этом позже.

## Клиентская программа

Компьютеры должны запускать какую-то программу, с помощью которой у меня будет доступ к выполнению команд и мне нужно как-то научится обрабатывать эти команды, и что вообще это будут за команды, как они будут интерпретироваться. Для этого я создал свой маленький командный интерпретатор, он принимает на вход от серверной программы строку, в которой написана команда, которую нужно выполнить. Вот пример записи команды, которая открывает YouTube: **c:/link/msedge.lnk,** [**https://youtube.com**](https://youtube.com)*.* Теперь я вам объясню, как работает этот интерпретатор. Изначально в него вшиты следующие команды, которые принимают какие-либо аргументы, перечисленные через “, ”:

* press, нажимает нужную клавишу пример: **press, a**
* hotkey, нажимает сочетание клавиш, пример: **hotkey, ctrl, a**
* click, нажимает клавишу мыши, пример: **click, left**
* write, печатает текст, заданный в аргументе,   
  пример: **write, Hello world**
* sleep, ожидает сколько-то секунд, **sleep, 4**
* exec, выполняет код написанный на python, заданный в аргументе, пример: **exec, print(“Hello, world”)**

Если же вы написали другую команду, то она будет передана в командную строку, например в ней можно указать путь до приложения, или файла, и тогда он откроется, также .exe файл может принимать дополнительные аргументы.

Еще была реализована система выполнения нескольких действий за раз, за счет символа &, пример: **c:/link/msedge.lnk,** [**https://youtube.com**](https://youtube.com) **& sleep, 2 & press, f11**.

Дополнительно для упрощения процесса тестирования я добавил логирование (описание всех происходящих событий в текстовый файл), и результат выполнения команды (успех, ошибка) отправляется с каждого компьютера на серверную программу, где записывается в нужном файле, который можно получить, написав боту **/log**.

## Серверная программа

Как-то нужно связать все компьютеры, это можно сделать, написав маленький сайт на библиотеке fastapi, к которому будут подключатся все клиентские программы. Здесь я также добавил некоторые “страницы” на сайт, с помощью которых бот (пользовательская программа) сможет отправлять команды от пользователя к серверной программе, и на клиентские программы, расположенные на компьютерах. Также за счет этих “страниц” бот может получать информацию о подключенных пользователях, и отправлять это пользователю.

## Пользовательская программа

Теперь мы переходим к самой сложной части моего проекта, это пользовательская программа, то есть telegram бот. Здесь мне нужно было написать множество его частей, таких как база данных, модуль, отвечающий за общение с сервером, модуль, отвечающий за общение с telegram api (основной) и скомпоновать все это в единую запускаемую программу, ну я пройдусь по каждому пункту отдельно:

### База данных

Для того чтобы бот работал ему нужно сохранять информацию о пользователях, о их id, о том имеют ли они право управлять компьютером, когда они начали использовать бота, и так далее. К каждому пользователю привязывается свой id, и он уникален, по нему можно идентифицировать пользователя, который ему пишет. В базе данных существует 3 таблицы, это таблица пользователей, таблица команд пользователей и таблица компьютеров (соотносит mac адрес к ip). О 2 я расскажу в части про взаимодействие с ботом. Для тех, кто знает, что такое ООП[[6]](#footnote-6), я написал отдельный класс для взаимодействия с базой данных, где создал функции, которые читают или записывают данные в базу данных. Это крайне удобный подход, так как множество раз переписывать одну и ту же строчку SQL кода (язык, который используется для взаимодействия с базой данных) крайне неэффективно, из-за того, что:

1. Код смешанного python и SQL, выглядит ужасно, громоздко и не интуитивно.
2. Если ты 1 раз допустил ошибку в SQL коде, то тебе нужно будет искать все места, где ты допустил эту ошибку, и исправлять ее.

Обобщая мой подход лучше классического по многим аспектам, которые я перечислил выше.

### Модуль общения с сервером

Для общения с серверной частью я создал отдельный файл, где написал множество функций, отвечающих за работу с сервером. Для этого я использовал классическую библиотеку python: requests. С ее помощью я отправлял запросы на сервер, с помощью который я доставал информацию о подключенных компьютерах и мог отправлять команды от пользователя на сервер, откуда они перенаправляются на компьютеры.

Также тут я реализовал команду start, с ее помощью можно

### Взаимодействие с пользователем

И теперь я расскажу вам о самой сложной части моего бота – это логика работы бота, на aiogram, и того, как бот взаимодействует с пользователем. Сначала бот определяет имеет ли пользователь права администратора, и если их нет, то он отвечает на любое сообщение, кроме **/start** (добавляет пользователя в базу данных), сообщением о том, что нет прав доступа.

Теперь я расскажу вам о концепции интуитивных команд. Суть в том, что каждый раз писать боту команду, это слишком сложно, так как для этого нужно уметь использовать мой “язык” программирования команд, что требует слишком большого количества усилий и времени. Заместо этого я решил использовать inline клавиатуру, встроенный инструмент от telegram. В чем их суть? Это очень классная штука, сейчас объясню почему. Смотрите, ваш бот отправил какое-то сообщение, и вам нужно иметь возможность, например удалить его с помощью нажатия на клавишу под сообщением, так вот набор клавиш под сообщением и есть inline клавиатура. В примере с удалением сообщения, у вас есть клавиша под сообщением, на которой написано, “Удалить” или “❌”. И нажимая на эту кнопку происходит отправка запроса на удаление сообщения к боту, где он обрабатывает полученный запрос. Также с помощью inline клавиш можно прикреплять ссылки.

Ну так вот, что-же я с их помощью реализовал? Изначально вы отправляете ему команду **/program код команды @.@ название команды,** также **/program** можно заменить на **/p**, после чего он записывает вашу команду в базе данных, и она будет доступна только вам, также тут я реализовал такую штуку, как аргументы команды, она обозначается через **@arg**, приведу пример, нужно вам создать команду, открывающая нужную ссылку в браузере, так вот, скорее всего при каждом вызове команды вам нужно будет вводить разные сайты, для этого я и ввел аргументы команд. Сейчас я вам покажу сообщение, добавляющее в бота команду открытия нужной ссылки, и после 2 секунд открывающее полноэкранный режим, клавишей f11:

**/program c:/link/msedge.lnk, @arg & sleep, 2 & press, f11 @.@ browser**

Это 1-ое, 2-ое: я добавил возможность отправлять запрос на только 1 компьютер, чтобы, например выключить компьютер играющему ученику. Также при вызове команды с аргументом создается новая команда, которая невидима. Еще я добавил команду удаления команд, это **/delete** или **/d**. И теперь самая главная команда, которую я использовал чаще всего – это **/activate** или **/a**.

Эта команда используется для отправки команд, и вот как она работает. Сначала написав эту команду боту, он отправляет вам сообщение со списком ваших в виде inline клавиатуры. Потом если вы выбрали команду с дополнительным аргументом, тогда бот вас попросит ввести нужный аргумент, просто отправьте ему сообщение с аргументом типа: [**https://youtube.com**](https://youtube.com), а если у вас нет аргументов в команде, то у вас не будет запрашивать аргумент. После чего вам отправляют список из компьютеров (их mac адрес) в виде inline клавиатуры, и еще 1 клавиши “Все”, а если ни один компьютер не подключен, при выборе вашей команды, бот отправит вам сообщение о том, что нет подключенных компьютеров. После чего выбранная вами команда выполняется на выбранном компьютере или компьютерах, тем самым автоматизировав рутину настройки каждого из компьютеров по отдельности. Также примеры того, как выглядит интерфейс бота, вы можете увидеть в приложении 2.

# Заключение

Обобщая все сказанное выше, я смог найти нужные новые технологии и библиотеки для создания моего продукта, также мне удалось их освоить на более простых примере создания массового тестирования современных web фреймворков. Я достиг поставленной цели, и смог решить все задачи в полном объеме.

Разработка таких продуктов может являться крайне сложным процессов, и

В заключении хочу отметить, что мною была проделана огромная работа, и также я много времени уделил отладке моего продукта, ведь даже после его создания он нуждался в множественной доработке. Также забыл упомянуть количество строк кода в моем проекте, оно равняет 1095 с учетом пустых строк кода и 850 без учета.

# Список литературы

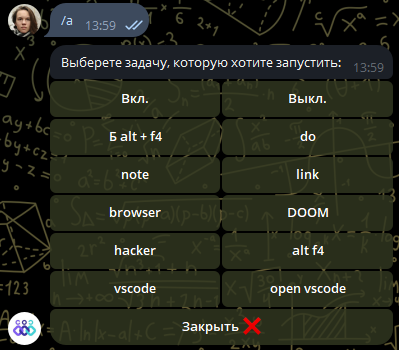
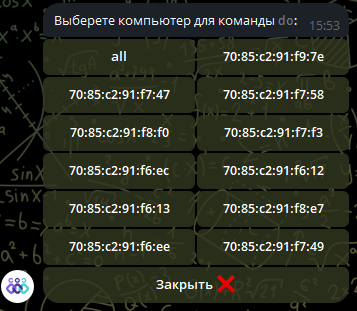
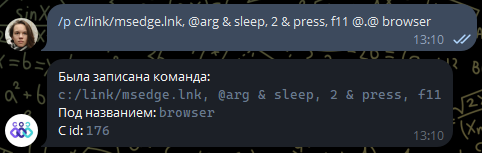
* Хабаров С. П. Построение распределенных систем на базе WebSocket, Учебное пособие для СПО, 2021, 216 с.
* Гэддис Т. Начинаем программировать на Python, 2021, 875 с.
* Фаулер М. Asyncio и конкурентное программирование на Python, 2023, 398 с.
* Документация python: <https://docs.python.org/3/>
* Документация aiogram: <https://docs.aiogram.dev/en/latest/>
* Документация fastapi: <https://fastapi.tiangolo.com/ru/>
* Документация mysql: <https://dev.mysql.com/doc/>

# Приложения

**Приложение 1**



**Приложение 2**

1. *консольные утилиты* – программы, работающие только в командной строке [↑](#footnote-ref-1)
2. *gui* – графический интерфейс пользователя [↑](#footnote-ref-2)
3. *Js* – Java Script [↑](#footnote-ref-3)
4. *Библиотеки* – уже написанный код, который ты можешь использовать [↑](#footnote-ref-4)
5. *api* – это способ, с помощью которого различные программы и сайты могут взаимодействовать между собой, обмениваться информацией и использовать функционал других систем, не зная их внутренней структуры. [↑](#footnote-ref-5)
6. *ООП* − объектно-ориентированное программирование [↑](#footnote-ref-6)