МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

КОМПЛЕКСНА КВАЛІФІКАЦІЙНА

МАГІСТЕРСЬКА РОБОТА

на тему:

«Інформаційна технологія віддаленого керування інформаційно-телекомунікаційною системою. Серверна частина.»

Завідувач
випускаючої кафедри **Довбиш А.С.**

Керівник роботи **Петров С.О.**

Студента групи ІН.м – 91н **Панченка С.В.**

СУМИ 2021

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
Кафедра комп'ютерних наук

Затверджую_____
Зав. кафедрою Довбиш А.С.
"_____" _____2021 г.

ЗАВДАННЯ

до комплексної дипломної роботи

Студента другого курсу, групи ІН.м-91н спеціальності "Інформатика" денної форми навчання Панченка Станіслава Валерійовича.

Тема: " **Інформаційна технология віддаленого керування інформаційно-телекомунікаційною системою. Серверна частина.**"

Затверджена наказом по СумДУ
№_____ от _____2021 г.

Зміст пояснювальної записки: 1) інформаційний огляд; 2) постановка завдання; 3) вибір програмних засобів для рішення поставленої задачі; 4) розробка серверної частини системи; 5) аналіз готової системи.

Дата видачі завдання "_____"_____2021 г.
Керівник випускної роботи _____ Петров С.О.
Завдання прийняв до виконання _____ Панченко С.В.

РЕФЕРАТ

Записка: 43 стор., 3 рис., 2 табл., 3 додатків, 16 джерел.

Об’єкт дослідження — системи віддаленого керування, методи та інструменти їх реалізації.

Мета роботи — інтеграція інструментів віддаленого керування до Telegram, дослідження можливостей подібних інтеграцій та потенціал використання API.

Методи дослідження — реалізація серверної частини системи віддаленого керування комп’ютером з використанням Telegram-бота та PHP.

Результати — розроблено серверну частину Telegram-бота для віддаленого виконання команд cmd на комп’ютері. Зроблено висновки щодо потенціалу використання API та інтеграцій різних систем.

СИСТЕМА ВІДДАЛЕНОГО КЕРУВАННЯ, TELEGRAM-БОТ,
ХОСТИНГ, ВЕБХУК, PHP, API.

ЗМІСТ

ВСТУП

Актуальність. У зв’язку з пандемією COVID-19 гостро постало питання віддаленого доступу, роботі на віддалені, сервісів доставки та багато інших змін, що стосуються як повсякденного життя, так і нестандартних завдань. Адаптивність, як невід’ємна складова еволюції, допомогла людству зреагувати на складні умови та швидко пристосуватися до них. Інформаційні технології посіли особливе місце у процесі адаптації, швидко провели необхідні вдосконалення та забезпечили плавний перехід до нових умов. У аспекті інформаційних систем складена ситуація змусила знову та по-новому поглянути на питання зв’язку, віддаленого керування, безконтактні системи, а також використання сучасних технологій у медицині. Враховуючи усі ці фактори було вирішено дослідити системи віддаленого керування, їх походження, розвиток, сучасні досягнення, розглянути та проаналізувати наявні технології, систематизувати та порівняти отримані дані та зробити висновки для подальшого покращення життя як розробників, так і користувачів в майбутньому.

Частиною комплексної роботи є реалізація серверної частини системи віддаленого керування.

Мета цієї роботи – дослідити рівень розвинутості систем віддаленого керування, систематизувати отримані дані та на їх основі розробити зручний сервіс для віддаленого керування персональним комп’ютером використовуючи сучасні тенденції.

Завдання:

- Ознайомитися із історією віддаленого керування;
- Оглянути сучасні системи віддаленого керування;
- Ознайомитися із готовими рішеннями поставленої проблеми;
- Проаналізувати їх проблеми та переваги;
- Обрати необхідні для розробки ресурси;
- Розробити серверну частину системи.

1. ІНФОРМАЦІЙНИЙ ОГЛЯД

1.1 Історія систем віддаленого керування

Першою спробу дистанційного керування вважається робота сербського фізика Ніколи Тесли. У 1898 році вчений продемонстрував підводний човен, що керувався дистанційно за допомогою радіохвиль.

Продовжуючи праці Тесли, Леонардо Торез Куведо у 1903 році продемонстрував роботу його робота, що виконував прості команди на віддаленні за допомогою радіохвиль.

Як відомо, більшість винаходів людства стали можливі завдяки військовим розробкам. Дистанційне керування не вийняток, тож із 30-х років XX століття почався активний розвиток систем дистанційного керування для військових цілей. У 1932 році була створена модель аероплана на радіокеруванні.

У другій половині XX століття дистанційне керування почало активно з’являтися у повсякденному житті людей у найрізноманітніших його сферах. Технології завжди намагалися зроби життя людей простішим, а бажання виконувати якомога більше дій, не підіймаючись з місця не лишало розумів як користувачів, так і розробників. Основним прикладом можна вважати пульт дистанційного керування до мультимедійних систем. За період розвитку технологій було винайдено багато рішень для дистанційного керування. Серед них були системи на дротах, з використанням радіохвиль, світлових сигналів, ультрафіолетового та інфрачервоного світла, звуку, ультразвуку і т.д.

1.2 Сучасні системи віддаленого керування

На сьогоднішній день людство не може уявити собі життя без сучасних досягнень науки та техніки. Інформаційні технології стали невід’ємною частиною людського побуту. Щодня з’являються нові рішення, що роблять наше життя простіше та економлять час. Головним поштовхом до розвитку систем дистанційного розвитку став факт розвитку та поширення систем

комунікації, таких як інтернет, мобільний інтернет, Bluetooth, Wi-Fi та інші, знайомі кожному користувачу системи. Таким чином системи віддаленого керування отримали змогу до майже безмежного розширення сфер свого застосування.

Основні приклади застосування дистанційного керування:

- Космічна техніка;
- Зв’язок та комунікація;
- Охоронні системи;
- Комп’ютерна техніка;
- Мультимедійна техніка;
- Військова техніка;
- Громадський транспорт;
- Під час Будівництва;
- Електроенергетика;
- Медицина;
- Лабораторне обладнання.

Можливості використання систем віддаленого керування не обмежуються вищевказаним списком, тому що кожного дня покращуються існуючі та створюються нові. На сьогоднішній день з використанням сучасних технологій можливо як регулювати штори за допомогою голосу, так і безпечно працювати з радіоактивними відходами з допомогою дистанційно керуємих роботів.

Як один з найцікавіших прикладів сучасного дистанційного керування можна привести набираючу популярність систему «розумного будинку». Як і більшість сучасних систем, «розумний будинок» має двосторонній зв’язок, тобто явним чином передає користувачу інформацію у відгук. Система має центральний контролер, що керує усіма частинами, системи моніторингу (різноманітні датчики температури, вологості, мікрофони, фотоелементи і т.д.), системи виконання (акустичні системи, «розумні розетки», побутова

техніка з відповідними функціями, електромеханічні системи регулювання і т.д.) та органи керування. Таким чином користувач має можливість не тільки полегшити собі життя знаходячись у будинку, а й керувати ним за його межами. Отже проблема невимкненої праски може назавжди відійти у минуле, а шанс крадіжки значно зменшиться.

1.3 Системи віддаленого керування персональним комп'ютером

Ідея віддаленого керування персональним комп'ютером почала зароджуватися разом із створенням персонального комп'ютера.

Однією з перших вдалих спроб віддаленого керування була програма Carbon Copy, розроблена Meridian Technology. Carbon Copy надавала віддалений сумісний доступ до екрану, передачу файлів та чат-вікна.

Інший приклад – програма pcAnywhere, що дозволяла керувати персональним комп'ютером з встановленим хостом pcAnywhere та із відомим паролем.

SSH (Secure Shell) – Мережевий протокол для віддаленого керування операційною системою. Реалізує безпечну передачу даних на відміну від схожих протоколів Telnet та rlogin. Для встановлення такого зв'язку між девайсом та комп'ютером необхідні SSH-клієнт та SSH-сервер.

Такий метод віддаленого підключення перевірений часом, дозволяє виконати більшу частину дій на віддаленому комп'ютері, але вимагає від користувача знання роботи із консольними оболонками системи.

Операційні системи створюють власні системи віддаленого керування. Windows віддалений робочий стіл або віддалене керування MacOS – основні приклади таких систем.

З часом на ринку програмних продуктів з'являлися й інші програми, що реалізували віддалене керування комп'ютером. Наведемо порівняння деяких з них.

Таблиця 1.1 Порівняння існуючих програм віддаленого керування

Застосунок	**Показ екрану**	**Віддалений доступ**	**Обмін повідомлення**	**Спільний доступ**	**Відео-зв'язок**	**Передача файлів**	**Платформа/ ОС**
Ammyy Admin	Так	Так	Так	Так	Ні	Так	Windows
BeAnywhere	Так	Так	Так	Так	Ні	Так	Windows, Java, Mac, Android
Chrome Remote Desktop	Так	Так	Так	Так	Так	Ні	Chrome OS, Linux (beta), OS X, iOS, Windows, Android
Glance	Так	Ні	Так	Так	Так	Ні	(No download) Windows, Mac, Linux, iPhone, Android
Goverlan Reach	Так	Так	Так	Так	Так	Так	Windows, Mac, Linux, iPhone, Android
CloudBerry Remote Assistant от	Так	Так	Так	Так	Так	Ні	Windows

CloudBerry Lab							
IBM Lotus Sametime	Так	Так	Так	Так	Так	Так	Windows, Linux, Mac
LogMeIn	Так	Так	Ні	Ні	Ні	Так	Windows, Mac, iPhone, iPad, Android
Mikogo	Так	Так	Так	Так	Так	Так	Windows, Mac, iPhone, iPad, Android
Nefsis	Так	Так	Так	Так	Так	Так	Windows
Netviewer	Так	Так	Так	Так	Так	Так	Windows, Mac
NoMachine	Так	Так	Так	Ні	Ні	Так	Windows, Mac, Linux, Raspberry, iOS, Android
RAdmin	Так	Так	Так	Ні	Ні	Так	Windows
RealVNC	Так	Так	Так	Так	Ні	Так	Windows, Mac, Linux, Raspberry Pi, iOS, Android, Chrome

							OS, Solaris, HP-UX, AIX
Skype	Так	Так	Так	Ні	Так	Так	Windows, Mac, (в Linux — без можливості показу)
Splashtop	Так	Так	Ні	Так	Ні	Так	Windows, Mac, iPhone, iPad, Android, Linux, Chrome OS, Chrome browser, FireTV, FireTV stick
TeamViewer	Так	Так	Так	Так	Так	Так	Windows, Mac, Linux, iPhone, Android
Techinline	Так	Так	Так	Так	Ні	Так	Windows
TigerVNC	Так	Так	Ні	Так	Ні	Так	Windows, Linux, Mac
TightVNC	Так	Так	Ні	Так	Ні	Так	Windows

WebEx	Так	Так	Так	Так	Так	Так	Windows, Linux, Mac, Unix, Solaris, iPhone
Wire	Так	Ні	Так	Ні	Так	Так	Windows, Linux, Mac, Unix, iPhone, Android
Zoom	Так	Так	Так	Так	Так	Так	Android, iOS, Windows, Linux, Mac

Найближче до теми роботи знаходиться Віддалений робочий стіл Chrome.

Основною його перевагою є простота у реалізації підключення. Вище було згадано, що тендецією розвитку є синхронізація девайсів. GoogleLLC. активно реалізують таку синхронізацію, тому створення можливості керування девайсами було лише питанням часу.

Для роботи з Chrome Remote Desktop необхідно виконати наступні кроки:

- Встановити Chrome Remote Desktop на комп’ютер, до якого необхідно підключитися
- У «Сервісах» обрати «Віддалений робочий стіл Chrome»
- Дозволити віддалені підключення
- Ввести PIN-код
- Виконати вхід до аканту Google з іншого девайсу
- Підключитися до віддаленого робочого столу використовуючи розширення браузера(для персональних комп’ютерів) або додатку(для смартфонів) ввівши відповідний PIN-код.

Враховуючи ситуацію у світі, пов’язану із пандемією COVID-19 багато компаній перейшли на віддалений режим. Таким чином програми віддаленого керування дали змогу швидко та ефективно перейти на новий режим. Для збереження зв’язку при неможливості знаходження співробітників поруч можливість демонстрації робочого столу та виконання дій дистанційно значно спростило роботу як роботодавцям, так і співробітникам. Служби підтримки мають змогу віддалено надавати технічну допомогу клієнтам.

Вивчаючи питання різноманітних систем, що функціонують у ситуації пандемії, враховуючи системи віддаленого керування і не тільки виявилось, що значну роль під час розробки будь-якого програмного продукту має API.

1.4 API та його застосування

API (Application Programming Interface) – інтерфейс програмування, набір классів, функцій, процедур, структур та констант для зовнішнього виклику іншими програмами. Іншими словами, API – стороння програма, функціонал якої може використовувати розробник у своїх цілях, не створюючи його власноруч з початку. До того ж не потрібно знати як працює система, API якої використовується.

API використовується маже усюди, де є інформаційні технології. Наприклад, інтернет, що складається з 7 рівнів мережевої моделі OSI, технології верхніх рівнів використовують API нижніх для конвертації протоколів під час передачі будь-яких пакетів даних.

Більшість користувачів не здогадуються про використання API у додатку. Найяскравішим прикладом під час пандемії COVID-19 можна привести мобільні додатки доставки. Під час замовлення користувач може провести оплату через API банківських систем, а потім обрати місце доставки за допомогою API карт (в першу чергу – API Google Maps). Таким чином користувачеві не потрібно окремо встановлювати додаткових додатків, витрачати зайвий час на пошук та вибір способу розрахунку, повідомлення коректної адреси і т.д. У свою чергу розробникам не потрібно додатково

створювати свою платіжну систему, систему надання адреси та інші, це значно спрощує процес розробки.

1.4 Вибір мессенджера

Так як у ситуації пандемії COVID-19 більшість діяльності перейшла на дистанційний режим, то можна вважати, що мессенджери – додатки для обміну повідомленнями – стали навід'ємною частиною людської діяльності. За даними сайту statista.com на 2021 рік, WhatsApp та Facebook Messenger займають перше да друге місця за кількістю користувачів у світі, далі йдуть Weixin/WeChat та QQ, що орієнтовані на китайську аудиторію та не користуються популярністю в Україні, наступним є Telegram.

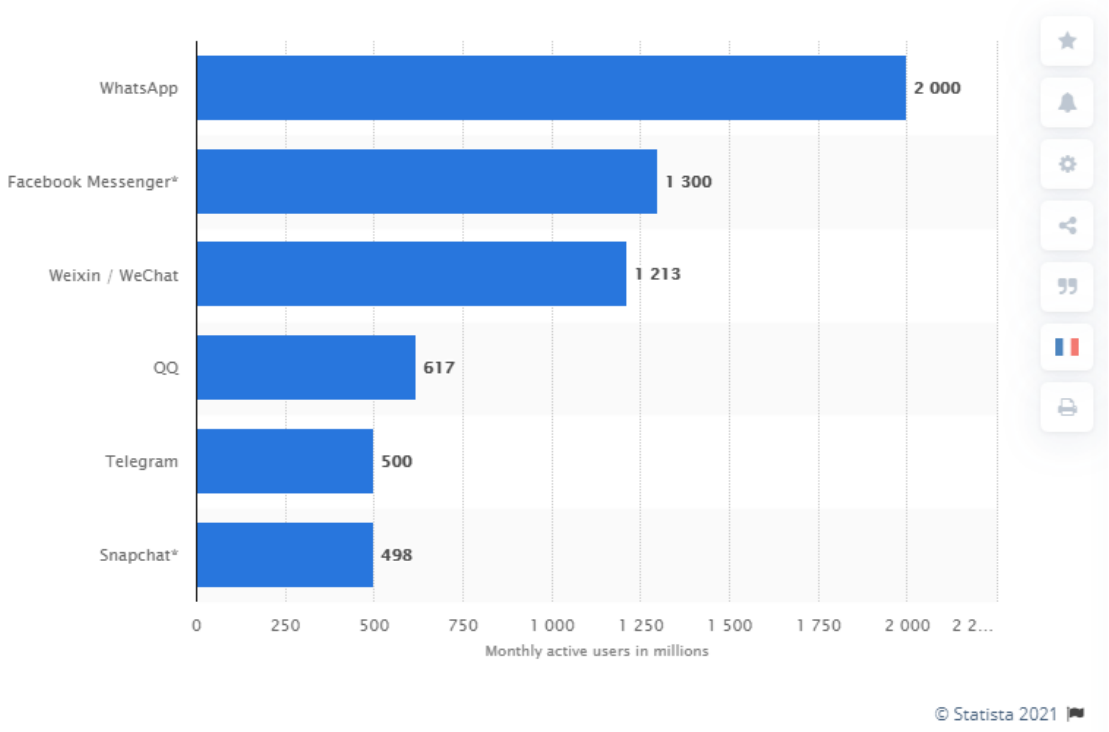

Рисунок 1.1 Популярність мессенджерів у світі

Щодо українських користувачів, за даними компанії Kantar на червень 2020 року найпопулярнішим мессенджером є Viber, але популярність набирають Facebook Messenger та Telegram.

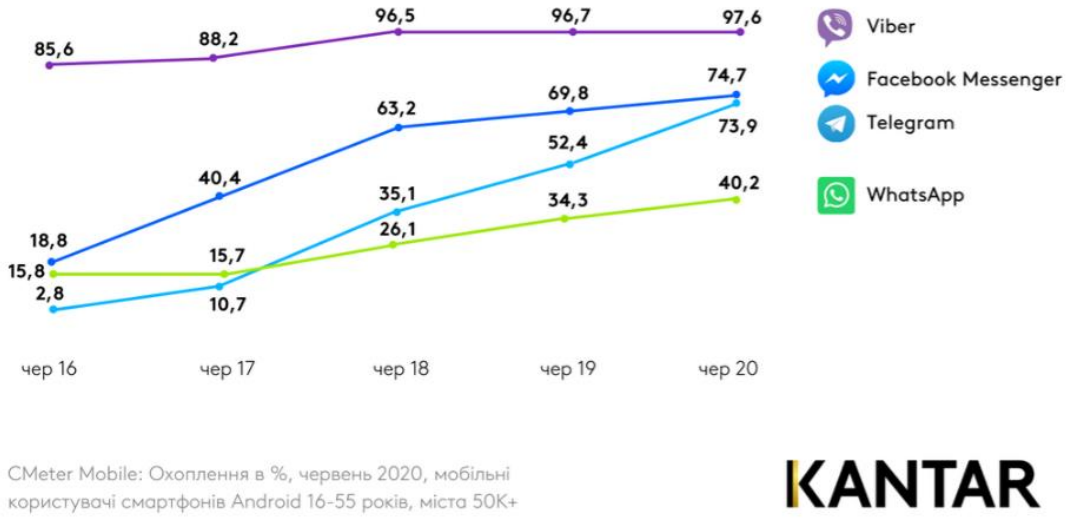

Рисунок 1.2 Динаміка кількості користувачів мессенджерів в Україні

Розглянувши та порівнявши можливості створення власних додатків-ботів з використанням вищезгаданих мессенджерів було обрано Telegram.

1.6 Постановка задачі

Враховуючи вищезгадане та сучасні тенденції було прийняте рішення створити сервіс віддаленого керування комп’ютером, що потребує від користувача найменшу кількість необхідних операцій.

Для серверної частини обрано мову програмування PHP для встановлення зв’язку між Telegram-ботом та комп’ютером користувача.

Від серверної частини системи вимагається працювати з інформацією, отриманою від комп’ютера, Telegram-бота, обробці отриманих даних та відправлення відповідей боту та комп’ютеру.

Таким чином маємо задачі:

- Створити Telegram-бот
- Реалізувати роботу команд бота в мережі
- Встановити зв’язок між комп’ютером та ботом

2 ТЕХНІЧНІ ЗАСОБИ РЕАЛІЗАЦІЇ

2.1 Telegram-бот. BotFather

Telegram – кроссплатформний мессенджер, розроблений у 2013 році Павлом Дуровим. Станом на 12 січня 2021 року Telegram має 500 мільйонів активних користувачів.

Бот – користувач, що керується програмою та слугує імітацією користувача. За час існування інформаційних систем було створено велику кількість ботів, як наприклад:

- Віртуальний співбесідник
- Пошуковий робот
- Автовідповідач
- Бот ведення логів
- Довідковий бот
- Бот комп’ютерної гри
- і т.д.

На жаль, створюються боти, що можуть заважати користувачам. Таким чином створювалися спам-боти, боти для голосувань або фальшивого додавання контенту, фішингові боти і т.д. Потужний інструмент ботів, як і будь-який інший може принести дуже велику користь або шкоду, в залежності від того, у чию владу потрапив цей інструмент.

Для теми роботи необхідний чат-бот, що буде відстежувати команди користувача та реагувати на них.

Для створення ботів в Telegram є бот BotFather. За допомогою нього створюються власні боти, що мають декілька особливостей, що відрізняють його від живого користувача. По-перше, це статус бота – “bot”, замість «Онлайн» та «Не в мережі». По-друге, повідомлення ботів видалаються через відведений час, для коректної роботи серверу (навідміну від діалогів живих користувачів, історія повідомлень зберігається до того часу, доки користувач не вирішить видалити її самостійно). По-третє, боти у Telegram не починають

діалог самостійно, а лише відповідають на повідомлення, що виключає можливіть спам-ботів. Також ім’я бота повинно закінчуватися на «bot» (наприклад @AnswersBot), а у групових чатах бот не отримує повідомлень, що не адресовані баспосередньо йому.

У роботі використовується API сервісу BotFather, що дозволяє швидко та зручно створити власного бота, що працює із достатньою результативністю та мінімальними затратами. Таким чином реалізується одна з основних переваг API – полегшити роботу розробникам. До того ж користувачі, що звикли до процесу роботи із ботами Telegram мають низький поріг входження до користування ботом, створеним у даній роботі.

Telegram-боти мають достатньо широкий фунціонал, включаючи кастомізовані клавіатури, команди за замовчуванням, додаткові інтерфейси і т.д.

На даний момент створено велику кількість корисних Telegram-ботів, серед яких боти відомих та великих ресурсі, таких як:

- GitHub (@GitHubBot)
- Wolfram Alpha (@WolframBot)
- Dr.Web (@DrWebBot)
- Wikipedia (@wikipedia_voice_bot)
- Netflix (@netflixnewsbot)
- «Нова Пошта», «Делівері», «Міст-Експресс», «Укрпошта» (@QTrackerBot)
- «Приват Банк» (@PrivatBankBot)
- AliExpress (@alisearchbot)
- Ebay (@ebayglobalbot)
- Amazon (@amazonglobalbot)

2.2 Мережеві протоколи

Для забезпечення передачі даних через мережу існують протоколи різних рівнів та різного призначення, наприклад: Ethernet, IP, TCP, FTP і т.д.

Модель OSI – мережева модель, у якій кожен рівень відповідає за свою частину передачі даних. Відповідно до цієї модель прикладний рівень сьомий, найвищий.

Для розробки додатків в першу чергу мають місце протоколи прикладного рівня, так як робота з протоколами нижчого рівня реалізується платформами, що використовуються для розробки.

Деякі приклади протоколів мережевого рівня:

- MIME
- LDP
- FTP
- SSH
- VoIP
- TELNET
- SMTP
- BitTorrent
- SNMP
- DHCP
- HTTP
- і т.д.

У даній роботі використовується протокол HTTP.

HTTP (Hyper Text Trnasfer Protocol) – мережевий протокол прикладного рівня, що належить до сьомого рівня моделі OSI та орієнтований на передачу веб-сторінок, хоча може використовуватись і для іншої інформації. При роботі з протоколом HTTP необхідно вказувати метод, що визначає які дані будуть передаватися та у якому вигляді.

HTTP протокол працює за логікою запит-відповідь. Зазвичай використовуються два основні методи HTTP запиту – GET та POST. Наведемо основні відмінності між ними.

Таблиця 2.1 Порівняння GET та POST методів

GET	POST
Можуть кешуватися	Ніколи не кешуються
Залишаються в історії браузера	Ніколи не залишаються в історії браузера
Дані для відправлення знаходяться в URL-адресі (обмежена довжина)	Дані для відправлення знаходяться в тілі запиту (необмежена довжина)
Використовується для отримання даних від ресурсу	Використовується для відправлення даних до ресурсу для обробки

Після отримання сервером HTTP-запиту та обробки необхідних даних на клієнт повертається HTTP-відповідь із необхідними даними. Також із відповіддю відправляється код стану, що повідомляє клієнту яка реакція була на його запит. Код стану має формат тризначного числа, перше з яких відповідає групі кодів. Всього груп п’ять:

- 1— інформаційні (повідомлення про обробку протоколів).
- 2— успіх (запит оброблено коректно)
- 3— перенаправлення (для успішного запиту необхідно виконати наступні дії)
- 4— помилка клієнта (невірний запит)
- 5— помилка сервера (неможливість сервера обробити запит)

Найчастіше зустрічаються наступні коди стану:

- 200 OK (запит вдалий)
- 304 Not Modified (сторінка не змінена)

- 400 Bad Request (синтаксична помилка запиту)
- 403 Forbidden (необхідний ресурс закритий для доступу)
- 404 Not Found (необхідний ресурс не знайдено)
- 405 Method Not Allowed (недоступний метод запиту)
- 408 Request Timeout (відсутні запити від клієнта за відведений час)
- 500 Internal Server Error (внутрішні помилки сервера)
- 502 Bad Gateway (невірний шлюз серверу)
- 503 Service Unavailable (сервер тимчасово не має можливості обробити запит)

Для забезпечення безпеки передачі інформації через мережу використовується схема HTTPS, що має відмінний від HTTP порт за замовчуванням та додатковий механізм шифрування.

2.3 Сервер, PHP

PHP (PHP: HypertextPreprocessor) – інтерпретована скриптова мова програмування, основною функцією якого є виконання дій на сервері.

У загальному випадку PHP може використовуватись для трьох цілей:

- Створення скриптів для виконання на сервері. Для цього необхідні інтерпретатор PHP, веб-сервер та браузер
- Створення скриптів для виконання в командному рядку. Для цього необхідний парсер PHP
- Створення віконних додатків, що виконуються на стороні клієнта на базі PHP-GTK

PHP дозволяє обирати між процедурним та об’єктно-орієнтованим підходом до написання програм, або їх комбінування.

Також PHP охоплює широкий спектр підримуваних надлашувань, таких як різні бази даних, робота з зображеннями, PDF-документами та flash-анімаціями, XML-файлами, регулярними виразами Pearl, користувацьким інтерфейсом і т.д.

У даній роботі PHP використовується для зв’язку Telegram-боту та комп’ютерної частини. Для написання використовується середовище PHPStorm.

PHPStorm – кроссплатформне середовище розробки, розроблене JetBrains для програмування на PHP та суміжних мовах. До переваг PHPStorm також належать:

- RHPCodeSniffer (перевірка коду в реальному часі)
- Автодоповнення
- Рефакторинги
- Підтримка SQLта баз даних
- Інтеграція систем керування версіями (Git)
- Локальна історія
- PHPUML (Діаграми класів)

Схема роботи, що буде детально описана далі базується на ідеї зв’язку Клієнта-комп’ютера та Telegram-клієнта із спільним сервером, що містить Telegram Bot API. Таким чином клієнт-комп’ютер передає дані на сервер із PHP-скриптами та отримує вказівки на виконання команд, а Telegram-клієнт повідомляє команди на сервер та отримує відповідний зворотній зв’язок.

Для виконання поставленої задачі будуть використані такі елементи PHP:

a) Робота з HTTP-запитами
b) Робота зі строковим типом даних
c) Конструкція switch/case/default для реакції на команди
d) Інструменти роботи з файлами:
    1. fopen
    2. fgets
    3. fwrite
    4. file_get_contents
    5. file_put_contents

e) base64_decode для конвертації зображень до зручного вигляду для його обробки

2.4 Хостинг та вебхуки

Хостинг – розміщення інформації на сервері з метою неперервного доступу до них у мережі.

Класифікація хостингів:

a) Повнофункціональні хостинги
    1. Віртуальний хостинг
    2. Віртуальний виділений хостинг
    3. Виділений сервер
    4. Колокація
    5. Хмарний хостинг
    6. Реселлерхостинг
b) Обмежений хостинг
    1. Веб хостинг
    2. E-mail хостинг
    3. DNS хостинг
    4. Ігровий хостинг

Для реалізації поставленої задачі необхідний веб-хостинг.

Веб-хостинг – це послуга з розміщення веб-сайтів на спеціалізованих ресурсах для отримання можливості доступу до них через мережу Інтернет. Ресурси, що надають послуги веб-хостингу називають хостинг-провайдерами. Існують достатньо безкоштовних простих хостингів для власних невеликих проектів, але для великих комерційних продуктів послуга має бути платною. Одним з основних критеріїв оцінки хостингу є аптайм.

Аптайм хостинга – відсоток часу перебування серверу в робочому стані.

Також під час створення власного домена з використанням веб-хостингу встановлюється ширина інтернет-каналу, що визначає можливий об’єм

передачі даних за одиницю часу. Ширина каналу вимірюється у бітах за секунду (біт/с).

000webhost– безкоштовний хостинг, що надає домен та директорію розміщення власних файлів з підтримкою PHP та MySQL.

Для виконання поставленої задачі необхідно мати реєстрацію на ресурсі 000webhost та створити домен для системи.

Вебхук – метод розробки веб-систем при якому сервер очікує подію та відправляє відповідний користувацький запит на вказану користувачем адресу. На даний момент багато сервісів використовують вебхуки у ситуації, коли необхідно мати зв’язок із сторонніми ресурсами. Вебхуки спрацьовують на події, тому немає необхідності постійних запитів на момент змін.

2.5 Створення структури системи

Для роботи системи необхідно об’єднати між собою наступні компоненти:

- TelegramBot-клієнт
- Telegram Bot API
- Хостинг сервіс
- C# програму на комп’ютері

Робота системи виконується наступним чином:

- ПК-клієнт зберігає логін та пароль користувача, відправляє дані користувача на хостинг та перевіряє наявність команд кожні 4с
- Telegram-бот очікує команд користувача, відправляє запит на сервер TelegramBotAPI, який в свою чергу реалізує вебхук до хостингу
- Хостинг отримує дані від комп’ютера та запити від Telegram-боту через вебхук та відповідає на них

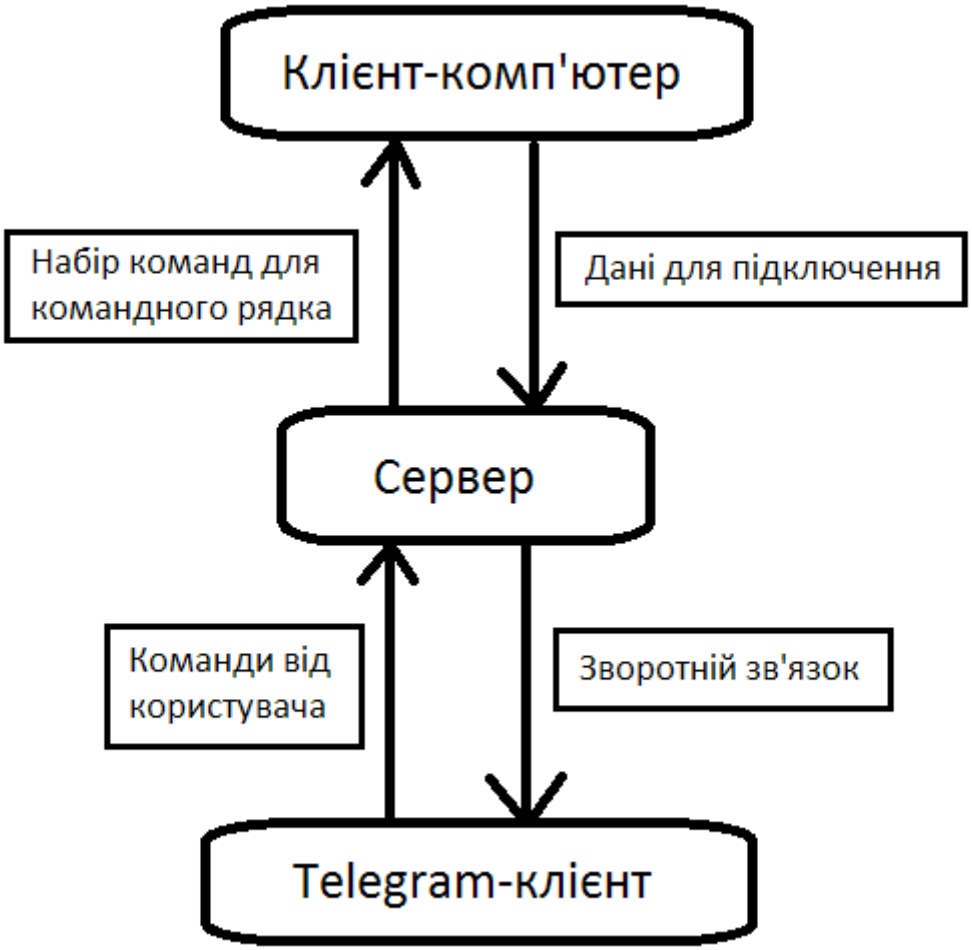

Рисунок 2.1 DF-діаграма проекту

На сервері знаходяться файли користувачів, назва яких Username.dat, де Username – логін комп’ютера користувача, що встановлюється останнім через форму ПК-клієнта. У тій самі формі потрібно задати пароль, що буде знаходитись у вищезгаданому файлі.

Також на сервері можуть знаходитися файли із командами для комп’ютера у форматі Username.cmd.dat, де Username – логін ПК-клієнта

3. РЕАЛІЗАЦІЯ СИСТЕМИ. СЕРВЕРНА ЧАСТИНА

3.1 Створення бота. BotFather

Використовуючи BotFather у Telegram бот створюється наступним чином:

Вводимо команду /newbot

- Додаємо ім’я (name) бота для контактів та чатів
- Додаємо Username – ім’я для використання у посиланнях (латиницею)
- Отримуємо токен для подальшого зв’язку з сервером

Бот має наступны налаштування:

- /setname – ім’я бота
- /setdescription –опис бота
- /setadouttext – короткий опис бота
- /setuserpic – встановлення зображення профілю бота
- /setjoingroups – можливість додавати бота в групи
- /setprivacy – режим приватності
- /deletebot – видалення бота

Таким чином маємо створеного Telegram-бота, що зв’язаний із сервером Telegram Bot API.

Для роботи з ботом у чаті користувач набирає повідомлення, що має вигляд:

/command login password

де command – необхідна команда,

login – логін ПК-клієнта,

password – пароль ПК-клієнта.

command має мати одне з трьох значень:

- /new для створення нової групи
- /add для додавання групи
- /save для збереження

- /cmd для введення команд
- /delete для видалення групи

3.2 Серверна реалізація. PHP

Telegram Bot API реалізує вебхук для перенаправлення запитів від Telegram-боту через сервер Telegram на необхідний розробнику сервер. Для виконання роботи обрано хостинг-сервіс 000webhost. Сервіс дозволяє створити власний домен для подальшого досягнення від ПК-клієнту. Після створення домену на сервер завантажуються необхідні файли.

Реалізація серверної частини на PHP розділена на 2 файли: index.php, що містить код для реагування на запити з Telegram-клієнту та image.php, що слугує для отримання даних від ПК-клієнтів та збереження їх у потрібному форматі.

Розглянемо зміст index.php.

Для початку необхідно описати адресу бота за токеном, що був отриманий від BotFather. Ініціалізуємо змінну $access_token відповідним значенням та змінну $api адресою Telegram Bot API з використанням збереженого токена.

```
$access_token = '765888819:AAHOgkWevzJg2-lju3A1k5Pku9_gqXOZaNA';
$api = 'https://api.telegram.org/bot' . $access_token;
```

Далі описуємо список змінних, що характеризують інформацію, отриману від користувача через бота:

```
$chat_id = $output['message']['chat']['id'];
$first_name = $output['message']['chat']['first_name'];
$message = $output['message']['text'];
$message = explode(' ', $message);
```

Також отримуємо дані про користувача з файлу data.dat

```
$data = file_get_contents('data.dat');
```

Наступна змінна необхідна для збереження ім’я групи, базове значення говорить про те, що група не створюється в даний момент.

```
$process = '/';
```

Далі використовуємо оператор switch для перевірки отриманої команди від бота. case відповідають за коректні варіанти команд, default – у разі помилки.

На початку необхідний файл у форматі логін.група.dat та відкрити його для опису. Змінна $fileName містить назву файла, а $file – безпосередньо файл, використовуючи назву, та аргумент «w+», що сповіщає, що данні буду дописані в кінець файлу.

```
switch ($message[0]) {
    $fileName = $first_name . '.' . $process . '.dat';
    $file = fopen($fileName, 'w+');
```

У першому case перевіряємо команду new. Перевіряємо змінну $process, якщо вона була змінена, то група вже створюється, використовуємо метод invalidComand() для повідомлення про помилку, що буде описано нижче. Інакше перевіряємо на наявність групи з такою назвою і в позитивному результаті повідомляємо користувача відповідним повідомленням за допомогою методу sendMessage(), що також буде описана нижче. В іншому випадку зберігаємо назву групи у змінній $process і також повідомляємо клієнта про успішність операції. Використовуємо ключове слово break для переходу до наступного case.

```
case 'new' :
        if ($process != '/') {
            invalidCommand($chat_id);
        }
        else if ($file) {
```

```
                sendMessage($chat_id, 'Group already
exist');
            }
            else {
                $process = $message[1];
                sendMessage($chat_id, 'Now add
clients');
            }
            break;
```

Наступний case для команди add. Перевіряємо, що група створена, в інакшому разі повідомляємо користувачу про помилку за допомогою методу invalidComand() аналогічно минулому case. Далі виконуємо перевірку авторизації за допомогою методу authCheck(), що буде описана нижче. Далі виконуємо дозапис в файл конструкції «логін/» та даємо зворотній зв’язок користувачу.

```
    case 'add' :
            if ($process == '/') {
                invalidCommand($chat_id);
            }
            else if (authCheck($chat_id, $message)) {
                $temp = fgets($file);
                file_put_contents($fileName, "");
                fwrite($file, $temp . $message[1] .
'/');
                sendMessage($chat_id, 'Added');
            }
            break;
```

Наступним перевіряється команда save. Знову перевіряємо логічність процесу з відповідними повідомленнями та, в іншому разі, повертаємо змінну

$process до початкового стану з відповідним зворотнім зв’язком до користувача.

```
    case 'save' :
            if ($process == '/') {
                invalidCommand($chat_id);
            }
            else {
                $process = '/';
                sendMessage($chat_id,              'Group
successfully added');
            }
            break;
```

Далі перевіряємо команду cmd. Знову перевіряємо змінну $process, в разі помилки повідомляємо користувача. Відкриваємо файл з групою та перевіряємо чи існує така група, в негативному разі інформуючи користувача методом sendMessage() та виходячи з операції за допомогою ключового слова break. У змінну $groups отримуємо данні із файлу $groupFile та ризбиваємо на масив методом explode() за роздільником «/». Ініціюємо змінну $string для строкових значень команд. Далі за допомогою оператора циклу for зберігаємо усі складові вхідної команди. Наступним кроком за допомогою циклу foreach записуємо усі непорожні складові команди для всіх логінів у файли з форматом назви «логін.cmd.dat» та закриваємо відповідні файли. Після цього закриваємо файл $groupFile та завершуємо виконання даного case.

```
    case 'cmd' :
            if ($process != '/') {
                invalidCommand($chat_id);
            }
            else {
```

```
                $groupFile = fopen($first_name . '.' .
$message[1] . '.dat', 'w+');

                if (!$groupFile) {
                    sendMessage($chat_id,   'No   group
with such name');
                    break;
                }

                $groups = fgets($groupFile);
                $groups = explode('/', $groups);

                $string = "";

                for ($i = 2; $i < count($message); $i++)
{
                    $string .= $message[$i] . ' ';
                }

                foreach ($groups as $key => $value) {
                    if ($value != '') {
                        $userFile   =   fopen($value   .
'.cmd.dat', 'w');

                        fwrite($userFile, $string);
                        fclose($userFile);
                    }
                }
```

```
                fclose($groupFile);
            }
            break;
```

Останній case – delete. Відкриваємо файл $groupFile. Виконуємо аналогічні перевірки що і у попередніх кроках: перевіряємо процес додавання, перевіряємо наявність групи, супроводжуємо відповідними повідомленнями, в іншому разі використовуємо метод unlink() для видалення файлу, повідомляємо користувача про успішне виконання, заеріваємо файл та завершуємо case.

```
    case 'delete' :
            $groupFile = fopen($first_name . '.' .
$message[1] . '.dat', 'w+');

            if ($process != '/') {
                invalidCommand($chat_id);
            }
            else if (!$groupFile) {
                sendMessage($chat_id, "No group with
such name");
            }
            else {
                unlink($groupFile);
                sendMessage($chat_id, "Group
sucessfully deleted");
            }

            fclose($groupFile);
            break;
```

У третьому case, а саме у випадку команди /image, знаходяться ті самі перевірки логіну та паролю. У разі їх співпадіння викликається sendImage() (що також описана нижче), що відправляє зображення з серверу, у якому міститься screenshot екрану користувача на Telegram-клієнт.

У випадку коли команда від користувача не знайшла відповідність жодному case виконається блок default, у якому на Telegram-клієнт надсилається відповідне повідомлення.

```
        default:
            sendMessage($chat_id, 'Invalid command');
            break;
    }
```

Після виконання вищеперерахованих дій необхідно повернути HTTP-код 200 (ОК) та закрити запит.

```
    http_response_code(200);
    fastcgi_finish_request();
```

Далі реалізується метод sendMessage(), що реалізує відправлення повідомлення на Telegram-клієнт.

```
    function sendMessage($id, $mess) {
        file_get_contents($GLOBALS['api'] . '/sendMessage?chat_id=' . $id . '&text=' . urlencode($mess));
    }
```

Наступним реалізується метод invalidCommand().

```
    function invalidCommand($id) {
        sendMessage($id, 'Invalid command');
    }
```

Останнім реалізований метод authCheck(); Відкриваємо файл логіну для читання. Створюємо змінну $flag логічного типу, що містить факт успішної авторизації. Якщо файл з логіном не знайдено – повідомляємо користувача та

ставимо $flag на хибний. Інакше відкриваємо файл логіну, використовуємо метод explode() для знаходження пароля та перевіряємо із введеним користувачем паролем. Якщо паролі не співпадають – ставимо $flag на хибне та відправляємо користувачу повідомлення. Закриваємо файл та повертаємо результат у вигляді змінної $flag.

```
    function authCheck($id, $mess) {
        $auth = fopen($mess[1] . '.dat', 'r+');
        $flag = true;

        if(!$auth) {
            sendMessage($id, 'No computer with such
login');
            $flag = false;
        }
        else {
            $pass = fgets($auth);
            $pass = explode(' ', $pass);
            if($pass[0] != $mess[2])
            {
                sendMessage($id, 'Incorrect password');
                $flag = false;
            }
        }

        fclose($auth);
        return $flag;
    }
```

Розглянемо зміст файла client.php. Отримуємо дані від клієнта-комп’ютера. Створюємо файл логін.dat та зберігаємо у ньому пароль.

Закриваємо файл відправляємо повідомлення про успішну передачу, закінчуємо зв’язок, закінчуємо метод.

```
if (count($_POST)) {
    $login = $_POST['login'] . '.dat';
    $file = fopen($login, 'w+');
    file_put_contents($login, "");
    fwrite($file, $_POST['pass']);
    fclose($file);

    http_response_code(200);
    fastcgi_finish_request();
    exit();
}
```

Останній файл deleteCmd.php. Зміст аналогічний попередньому, видаляємо файл із командами користувача.

```
if (count($_POST)) {
    $file = $_POST['login'] . '.cmd.dat';
    unlink($file);

    http_response_code(200);
    fastcgi_finish_request();
    exit();
}
```

ВИСНОВКИ

Під час виконання роботи було реалізовано серверну частину системи віддаленого керування комп’ютером за допомогою Telegram-боту, що повноцінно функціонує разом з клієнтською частиною, реалізованою на C#/.NET з використанням Windows Forms та WebClient.

Особливістю роботи є зв’язування різних ресурсів та API. Було використано інструменти створення Telegram-боту, API Telegram-ботів з вебхуками, хостинги, PHP технології.

Метою даної роботи було вивчення гнучкості сучасних систем до умов, що змінюються. У ході роботи було виявлено, що сучасний світ технологій дозволяє швидко та зручно адаптуватися та дозволяє людям продовжувати свою активність. Було проаналізовано такий важливий компонент розробки, як API, його особливості та необхідність його використання для розробників та користувачів.

Система виконує поставлену задачу, а саме: пов’язує комп’ютер користувача із Telegram-ботом будь-якого зручного для користувача девайсу та дозволяє виконувати дії у командному рядку.

Розробивши систему та проаналізувавши її було знайдено декілька недоліків, а саме:

- Паролі зберігаються у незашифрованому вигляді
- Валідність роботи тільки на системі Windows
- Відсутність будь-якої валідації на стороні ПК-клієнта

Враховуючи вищесказане можна сформувати план подальшого покращення системи:

- Виправити знайдені недоліки
- Реалізувати підтримку систем Linux
- Провести широке відкрите тестування

СПИСОК ЛІТЕРАТУРИ

1. Програмування мовою C# 6.0 / Коноваленко І.В. – Тернопіль: ТНТУ, 2016. – 228 с.
2. C#. Посібник для 10-11 класів інформаційно-технологічного профілю. Частина 1 / Зеленський О.С.- 2010. – 59 с.
3. Кузнецов Максим Валерьевич. Самоучитель PHP 7 / Кузнецов М.В., Симдянов И.В. – Санкт-Петербург: БХВ-Петербург, 2018. – 450 с.
4. Симдянов Игорь Вьячеславович. PHP 7. В подлиннике / Симдянов И.В., Котеров Д.В. – Санкт-Петербург: БХВ-Петербург, 2016. - 1073 с.
5. Си Шарп: Создание приложений для Windows / Лабор В.В. – Мн.: Харвест, 2003. – 384 с.
6. Upgrading to PHP 7 / Davey Shafik – O'Reilly Media, Inc. ISBN:9781492048374, 2015. – 84 с.
7. Мкртчян В.С., Фіногеєв О.Г., Финогеєв Є.А., Губанов Н.Н.. Облачный мониторинг сети инженерных коммуникаций системы городского теплоснабжения в скользящем режиме // Інтернет-журнал «Отходы и ресурсы» Том 3, №1 (2016) http://resources.today/PDF/07RRO116.pdf
8. Марк Сейфер. Никола Тесла. Повелитель Вселенной – Liters, ISBN 5457239942, 2020. – 4307 с.
9. API Design Patterns / JJ Geewax – MEAP , ISBN 9781617295850, 2019. - 487 с.
10. Непрерывное развитие API. Правильные решения в изменчивом технологическом ландшафте / Мехди Меджуи, Эрик Уайлд,Ронни Митра, Майк Амундсен [переклад Григорьєва А.] - O'Reilly Media, Inc., ISBN 9785446112326, 2020. – 272 с.
11. REST API Design Rulebook: Designing Consistent RESTful Web Service Interfaces / Mark Masse - O'Reilly Media, Inc., ISBN 9781449310509, 2012. – 93 с.

12. Dickson B. How to prevent your IoT devices from being forced into botnet bondage [Електронний ресурс] / Dickson. – 2015. – Режим доступу до ресурсу: https://techcrunch.com/2016/08/16/how-to-prevent-your-iot-devices-from-beingforced-into-botnet-slavery/.
13. Pethuru R. The Internet of Things / Raj Pethuru. – Munich: Appress, 2017. – 365 с.
14. Бланк С. Стартап: Настільна книга засновника / С. Бланк, Б. Дорф; [пер. з англ. Т. Гутман, І. Окунькова, О. Бакушева] — Москва: Альпіна Паблішер, 2013. — 485 с.
15. Telegram. Основные критерии и возможности / Мариус С. – Москва ЛитРес Самиздат ISBN 9785532999176, 2020. – 39 с.
16. Designing Bots: Creating Conversational Experiences / Amir Shevat – O'Reilly Media, Inc., ISBN 9781491974827, 2017. – 348 с.

ДОДАТКИ

Додаток А

Текст файлу index.php

```
    $access_token = '765888819:AAHOgkWevzJg2-
lju3A1k5Pku9_gqXOZaNA';
    $api = 'https://api.telegram.org/bot' .
$access_token;

    $chat_id = $output['message']['chat']['id'];
    $first_name =
$output['message']['chat']['first_name'];
    $message = $output['message']['text'];
    $message = explode(' ', $message);
    $data = file_get_contents('data.dat');
    $process = '/';

    switch ($message[0]) {
        $fileName = $first_name . '.' . $process .
'.dat';
        $file = fopen($fileName, 'w+');

        case 'new' :
            if ($process != '/') {
                invalidCommand($chat_id);
            }
            else if ($file) {
                sendMessage($chat_id, 'Group already
exist');
            }
```

```
            else {
                $process = $message[1];
                sendMessage($chat_id, 'Now add
clients');
            }
            break;
        case 'add' :
            if ($process == '/') {
                invalidCommand($chat_id);
            }
            else if (authCheck($chat_id, $message)) {
                $temp = fgets($file);
                file_put_contents($fileName, "");
                fwrite($file, $temp . $message[1] .
'/');
                sendMessage($chat_id, 'Added');
            }
            break;
        case 'save' :
            if ($process == '/') {
                invalidCommand($chat_id);
            }
            else {
                $process = '/';
                sendMessage($chat_id, 'Group
successfully added');
            }
            break;
        case 'cmd' :
```

```
            if ($process != '/') {
                invalidCommand($chat_id);
            }
            else {
                $groupFile = fopen($first_name . '.' .
$message[1] . '.dat', 'w+');

                if (!$groupFile) {
                    sendMessage($chat_id, 'No group
with such name');
                    break;
                }

                $groups = fgets($groupFile);
                $groups = explode('/', $groups);

                $string = "";

                for ($i = 2; $i < count($message);
$i++) {
                    $string .= $message[$i] . ' ';
                }

                foreach ($groups as $key => $value) {
                    if ($value != '') {
                        $userFile = fopen($value .
'.cmd.dat', 'w');

                        fwrite($userFile, $string);
```

```
                        fclose($userFile);
                    }
                }

                fclose($groupFile);
            }
            break;
        case 'delete' :
            if ($process != '/') {
                invalidCommand($chat_id);
            }
            else {
                unlink($file);
            }
            break;
        default:
            invalidCommand($chat_id);
            break;

        fclose($file);
    }

    http_response_code(200);
    fastcgi_finish_request();

    function sendMessage($id, $mess) {
        file_get_contents($GLOBALS['api'] .
'/sendMessage?chat_id=' . $id . '&text=' .
urlencode($mess));
```

```
    }

    function invalidCommand($id) {
        sendMessage($id, 'Invalid command');
    }

    function authCheck($id, $mess) {
        $auth = fopen($mess[1] . '.dat', 'r+');
        $flag = true;

        if(!$auth) {
            sendMessage($id, 'No computer with such
login');
            $flag = false;
        }
        else {
            $pass = fgets($auth);
            $pass = explode(' ', $pass);
            if($pass[0] != $mess[2])
            {
                sendMessage($id, 'Incorrect
password');
                $flag = false;
            }
        }

        fclose($auth);
        return $flag;
    }
```

Додаток Б

Вміст файлу client.php

```
if (count($_POST)) {
    $login = $_POST['login'] . '.dat';
    $file = fopen($login, 'w+');
    file_put_contents($login, "");
    fwrite($file, $_POST['pass']);
    fclose($file);

    http_response_code(200);
    fastcgi_finish_request();
    exit();
}
```

Додаток В

Вміст файлу deleteCmd.php

```
if (count($_POST)) {
    $file = $_POST['login'] . '.cmd.dat';
    unlink($file);

    http_response_code(200);
    fastcgi_finish_request();
    exit();
}
```