МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ЕЛЕКТРОНІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

Кваліфікаційна робота магістра

ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНА ТЕХНОЛОГІЯ КЕРУВАННЯ ВІДДАЛЕНИМ КОНСОЛЬНИМ СЕРВЕРОМ ЧЕРЕЗ МОБІЛЬНИЙ ДОДАТОК

В.о. завідувача кафедри
кандидат технічних наук, доцент,
доцент кафедри комп'ютерних наук Ігор ШЕЛЕХОВ

Науковий керівник:
старший викладач кафедри
комп'ютерних наук, к.ф.-м.н. Дмитро ВЕЛИКОДНИЙ

Здобувач вищої освіти гр. ІН.м-13 Дар'я ГРОМЕНКО

СУМИ 2022

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

Факультет ЕлІТ Кафедра Комп'ютерних наук

Спеціальність 122 Комп'ютерні науки

ЗАТВЕРДЖУЮ:

в.о. зав. кафедри _____

«_____»_____ 20_____р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Дар'ї Валеріївни Громенко

1. Тема роботи *Інформаційно-комунікаційна технологія керування віддаленим консольним сервером через мобільний додаток*
затверджую наказом по СумДУ від «____» _____ 20 ___ р. № _____
2. Термін здачі здобувачем кваліфікаційної роботи _____
3. Вхідні дані до кваліфікаційної роботи _____
_____
4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) *1) Вступ; 2) Інформаційний огляд; 3) Постановка завдання; 4) Вибір методу рішення завдання; 5) Програмна реалізація; 6)Висновок; 7) Список літератури; 8) Додаток.*
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) _____
_____
6. Консультанти до проекту (роботи), із значенням розділів проекту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання «____» _____ 20 ___ р. Керівник _____

Завдання прийняв до виконання _____

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів дипломного проекту (роботи)	Термін виконання	Примітка
1	*Аналіз проблеми та постановка задачі*		
2	*Вибір методу рішення*		
3	*Програмна реалізація*		
4	*Оформлення кваліфікаційної магістерської роботи*		

Здобувач вищої освіти _____

Науковий керівник _____

РЕФЕРАТ

Записка: 70 ст., 21рис., 1 табл., 1 додаток, 17 джерел.

Об’єкт дослідження — процес інформаційного аналізу і розробки мобільного додатку для віддаленого керування консольним сервером.

Мета роботи – розробка інформаційно-комунікаційної технології керування віддаленим консольним сервером за допомогою мобільного додатку.

Методи дослідження – методи проєктування інформаційних систем, об’єктно-орієнтована парадигма програмування.

Результати – розроблений власний мобільний додаток за допомогою об’єктно-орієнтованої мови програмування Java на операційній системі Android для керування віддаленим консольним сервером.

КОНСОЛЬНИЙ СЕРВЕР, ВІДДАЛЕНИЙ ДОСТУП, SSH, TELNET, МОБІЛЬНИЙ ДОДАТОК, JAVA, ANDROID.

ЗМІСТ

ВСТУП

IT-інфраструктура сьогодні має вирішальне значення для багатьох глобальних корпорацій. Простої мережі можуть призвести до величезних фінансових втрат у короткостроковій і довгостроковій перспективі, коли компанії втрачають здатність працювати, спілкуватися або обробляти платежі. У крайніх випадках деякі підприємства ніколи не відновлюються після незапланованих простоїв.

Щоб забезпечити безвідмовну роботу і знизити потенційні втрати, підприємствам потрібна можливість якомога швидше усувати перебої в мережевому обладнанні. Невеликі організації з однією серверною кімнатою поблизу можуть досить легко вирішувати проблеми, використовуючи персонал на місці. Однак для великих корпорацій із сотнями або тисячами офісів утримання IT-персоналу в кожному офісі непомірно дороге. Відправка когось, коли є проблема, є дороговартісною з точки зору поїздок, а також тривалого простою. Віддалене управління через термінальні або консольні сервери може бути ефективним рішенням, що поєднує в собі швидкість і економію коштів.

Через підвищену обчислювальну потужність мобільні пристрої Android, такі як телефони і планшети, стали заміною комп'ютерів. Таким чином, цілком можливо керувати серверами та виконувати функції системного адміністратора за допомогою планшета чи телефону. Фактично деякі телефони та планшети Android мають режими робочого столу. Поєднання бездротової клавіатури та миші дає вам чудову альтернативу ноутбуку. Таким чином, встановлення подібного мобільного додатка для Android дозволяє віддалено виконувати потрібні команди для налаштування мережі.

Головними перевагами мобільного додатка є: зручність - ви можете отримати доступ до своїх вузлів SSH з будь-якого місця, використовуючи пристрій у вашій кишені; підключення до хоста SSH за допомогою програми для Android так само

безпечно, як і підключення за допомогою більш традиційного клієнта SSH для настільних комп'ютерів. Більшість клієнтських програм Android SSH також повнофункціональні, як і їх настільні побратими (наприклад, PuTTy).

В рамках даної роботи буде описано створення мобільного додатку для віддаленого підключення та роботи з консольним сервером.

1 ІНФОРМАЦІЙНИЙ ОГЛЯД

1.1 Віддалений доступ

Віддалений доступ - це можливість доступу уповноваженої особи до комп'ютера або мережі з географічної відстані через мережеве з'єднання.

Віддалений доступ дозволяє користувачам підключатися до потрібних їм систем, коли вони знаходяться фізично далеко. Це особливо важливо для співробітників, які працюють у філіях, перебувають у відрядженнях або на дистанційній роботі.

Віддалений доступ дозволяє користувачам у будь-який час отримати доступ до файлів та інших системних ресурсів на будь-яких пристроях або серверах, які підключені до мережі. Це підвищує продуктивність праці співробітників і дозволяє їм краще співпрацювати з колегами по всьому світу.

Стратегія віддаленого доступу дає організаціям гнучкість для найму найкращих талантів незалежно від місця розташування, усунення ізоляції та сприяння співпраці між командами, офісами та місцезнаходженням.

Фахівці технічної підтримки можуть використовувати віддалений доступ для підключення до комп'ютерів користувачів з віддалених місць, щоб допомогти їм вирішити проблеми з їх системами або програмним забезпеченням.

Віддалений доступ здійснюється за допомогою комбінації програмного, апаратного забезпечення та мережевого підключення.

Наприклад, традиційний віддалений доступ до широкої доступності підключення до Інтернету здійснювався за допомогою програмного забезпечення для емуляції терміналу, яке керувало доступом через апаратний модем, підключений до телефонної мережі[1].

Сьогодні віддалений доступ частіше здійснюється за допомогою:

1. Програмне забезпечення: за допомогою захищених програмних рішень, таких як, наприклад, VPN.

2. Апаратне забезпечення: шляхом підключення хостів через дротовий мережевий інтерфейс або мережевий інтерфейс Wi-Fi.

3. Мережа: шляхом підключення через Інтернет.

Віддалений доступ VPN з'єднує окремих користувачів з приватними мережами. При використанні VPN з віддаленим доступом кожному користувачеві потрібен VPN-клієнт, здатний підключатися до VPN-сервера приватної мережі.

Коли користувач підключається до мережі через VPN-клієнт, програмне забезпечення шифрує трафік, перш ніж передавати його через Інтернет. Сервер VPN, або шлюз, розташований на межі цільової мережі, розшифровує дані та надсилає їх на відповідний хост всередині приватної мережі.

Комп'ютер повинен мати програмне забезпечення, яке дозволяє йому підключатися та спілкуватися з системою або ресурсом, розміщеним у службі віддаленого доступу організації. Після підключення комп'ютера користувача до віддаленого хосту на ньому може з'явитися вікно з робочим столом цільового комп'ютера.

Підприємства можуть використовувати віддалені робочі столи, щоб дозволити користувачам віддалено підключатися до своїх додатків і мереж. Віддалені робочі столи використовують прикладне програмне забезпечення - іноді включене в операційну систему (ОС) віддаленого хоста - яке дозволяє програмам працювати віддалено на мережевому сервері і в той же час відображатися локально.

Користувачі можуть отримати безпечний доступ до локальних і хмарних додатків і серверів з будь-якого місця, з будь-якого пристрою за допомогою різних методів автентифікації, включаючи віддалений єдиний вхід, який надає користувачам простий і безпечний доступ до потрібних їм додатків без налаштування VPN або зміни політик брандмауера.

Крім того, організації можуть використовувати багатофакторну автентифікацію для перевірки особи користувача шляхом поєднання декількох облікових даних, унікальних для однієї особи.

Традиційно підприємства використовували модеми та технології комутованого доступу, щоб дозволити співробітникам підключатися до офісних мереж через телефонні мережі, підключені до серверів віддаленого доступу. Пристрої, підключені до комутованих мереж, використовують аналогові модеми для дзвінків на призначені телефонні номери для встановлення з'єднань і відправлення або отримання повідомлень[2].

Широкосмуговий зв'язок надає віддаленим користувачам можливість високошвидкісного підключення до бізнес-мереж та Інтернету.

Кабельний широкосмуговий зв'язок розподіляє пропускну здатність між багатьма користувачами, і, як наслідок, швидкість передачі даних може бути низькою в години інтенсивного використання в районах з великою кількістю абонентів.

Широкосмуговий зв'язок DSL (цифрова абонентська лінія) забезпечує високошвидкісний зв'язок через телефонну мережу з використанням технології широкосмугового модему. Однак DSL працює лише на обмеженій фізичній відстані і може бути недоступною в деяких районах, якщо місцева телефонна інфраструктура не підтримує технологію DSL.

До послуг стільникового Інтернету можна отримати доступ за допомогою мобільних пристроїв через бездротове з'єднання з будь-якого місця, де доступна мережа стільникового зв'язку.

Послуги супутникового Інтернету використовують телекомунікаційні супутники для надання користувачам доступу до Інтернету в районах, де наземний доступ до Інтернету недоступний, а також для тимчасових мобільних установок.

Волоконно-оптична широкосмугова технологія дозволяє користувачам швидко і безперешкодно передавати великі обсяги даних[3].

До поширених протоколів віддаленого доступу відносяться наступні:

– Протокол "точка-точка" (Point-to-Point Protocol, PPP) дозволяє хостам встановлювати пряме з'єднання між двома кінцевими точками.

– IPsec - Internet Protocol Security - набір протоколів безпеки, що використовуються для забезпечення аутентифікації та шифрування для захисту передачі IP-пакетів через Інтернет.

– Point-to-Point Tunneling (PPTP) - один з найстаріших протоколів для реалізації VPN. Однак з роками він виявився вразливим до багатьох видів атак. Хоча PPTP не є безпечним, в деяких випадках він зберігає свою актуальність.

– Протокол тунелювання другого рівня (L2TP) - це протокол VPN, який не пропонує шифрування або криптографічну аутентифікацію для трафіку, який проходить через з'єднання. Як наслідок, він зазвичай працює в парі з IPsec, який надає ці послуги.

– Remote Authentication Dial-In User Service (RADIUS) - протокол, розроблений в 1991 році і опублікований як специфікація треку Internet Standard в 2000 році, щоб дозволити серверам віддаленого доступу зв'язуватися з центральним сервером для аутентифікації користувачів, що входять в мережу, і авторизації їх доступу до запитуваної системи або служби.

– Система контролю доступу контролера термінального доступу (TACACS) - це протокол віддаленої автентифікації, який спочатку був поширений в мережах Unix, що дозволяє серверу віддаленого доступу пересилати пароль користувача на сервер автентифікації, щоб визначити, чи повинен бути дозволений доступ до даної системи. TACACS+ є окремим протоколом, призначеним для обробки аутентифікації та авторизації, а також для обліку доступу адміністраторів до мережевих пристроїв, таких як маршрутизатори та комутатори[4].

1.2 Консольний сервер

Для підвищення доступності мережі та можливостей віддаленого усунення збоїв корпорації часто доповнюють програми NMS рішеннями для зовнішнього управління. Консольні сервери надають системним адміністраторам доступ до пристрою, що вийшов з ладу, через зовнішнє з'єднання (OOB). Доступність OOB дає вам змогу отримувати доступ до ваших мережевих пристроїв, навіть якщо ваша

мережа не працює, через Інтернет, стільниковий зв'язок 4G або POTS - стару просту телефонну службу. У цьому рішенні використовуються стандартні програми, такі як SSH/Telnet і HTTPS.

Консольний сервер - це апаратний пристрій, який під'єднується через послідовні порти RS-232 до вашого комутатора, маршрутизатора, брандмауера, сервера, PBX, UPS or PDU на одному кінці та до порту Ethernet на іншому кінці. Один консольний сервер може мати до 48 послідовних портів. Консольні сервери часто під'єднані до іншої мережі, відмінної від вашої мережі передавання даних.

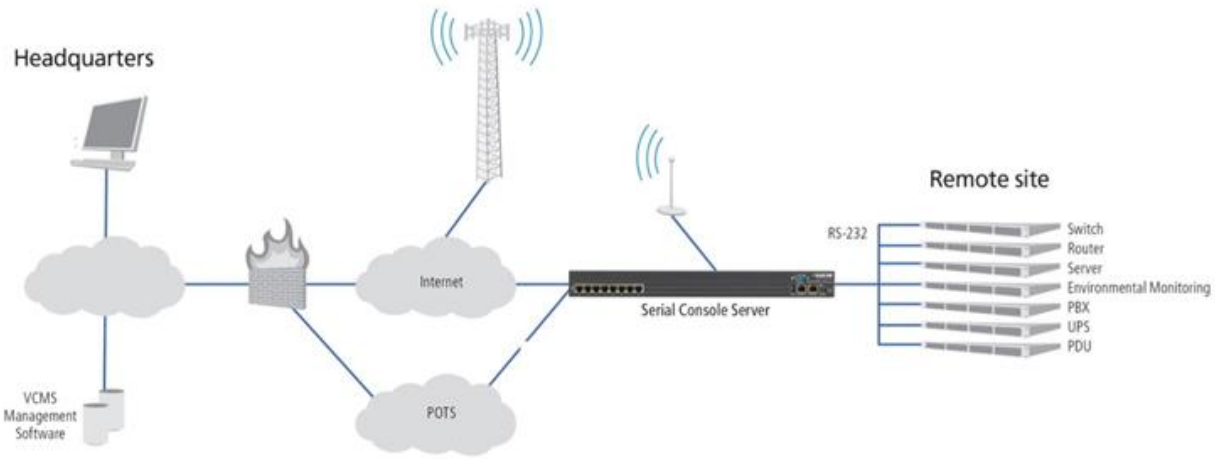

Рисунок 1.1 – Out-of-Band Management з використанням консольного серверу

Консольні сервери надають низку послідовних портів, які підключаються до консольних портів активних мережевих пристроїв. Консольні сервери забезпечують віддалений доступ до критично важливої IT-інфраструктури, включно з серверами, комутаторами та маршрутизаторами в центрах обробки даних і розподілених середовищах. Зазвичай вони включають доступ Secure Shell (SSH), який є мережевим протоколом для безпечної роботи мережевих служб у незахищеній мережі. Деякі консольні сервери мають багатофункціональні можливості, як-от реєстрація всіх даних послідовних портів, запуск подій,

дотримання інтелектуальних правил, надсилання сповіщень електронною поштою та резервні порти Ethernet для приєднання до кількох мереж.

Консольні сервери можуть заощадити ваш час і гроші, забезпечивши високу доступність ваших IT-систем і забезпечивши безперервність вашого бізнесу. Віддалене керування та усунення несправностей забезпечують швидке аварійне відновлення після перебоїв у подачі електроенергії, а також позбавляють від необхідності відправляти персонал на місце для усунення проблем, що економить ваш час і додаткові витрати на поїздки. З метою безпеки зовнішнє управління інтегрується з VPN і системами аутентифікації та поширює корпоративну політику безпеки на розподілені активи. Консольні сервери також є масштабованим рішенням. Незалежно від розміру вашої мережі ви можете отримувати доступ до пристроїв, відстежувати стан і усувати збої з будь-якого місця через єдиний портал VCMS[5].

1.3 Telnet vs SSH

Як було сказано у минулому розділі, існує декілька популярних протоколів для підключення віддаленого доступу до мережі, а саме Telnet та SSH. Нижче приведено їх порівняння.

Telnet - це стандартний протокол TCP/IP для служби віртуального терміналу. Це дає вам змогу встановити з'єднання з віддаленою системою таким чином, щоб вона відображалася як локальна система. Повна форма TELNET - термінальна мережа.

Протокол Telnet здебільшого використовується мережевим адміністратором для віддаленого доступу та управління мережевими пристроями. Він допомагає їм отримати доступ до пристрою, підключившись по телнету до IP-адреси або імені хоста віддаленого пристрою. Він дає змогу користувачам отримувати доступ до будь-якої програми на комп'ютері віддалено. Він допомагає їм встановити з'єднання з віддаленою системою[6].

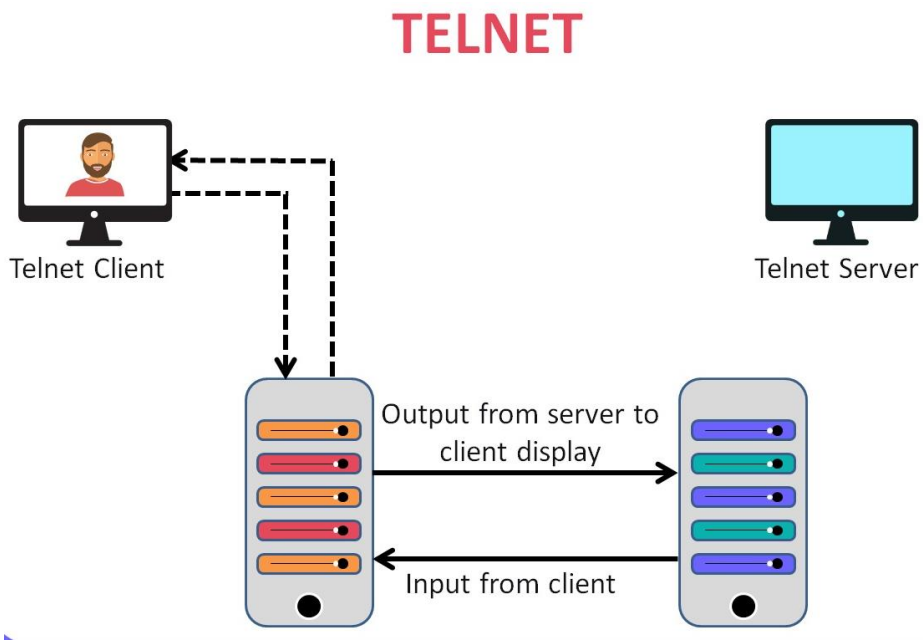

Рисунок 1.2 – Приклад використання протоколу Telnet

SSH - це мережевий протокол, який використовується для віддаленого доступу до пристрою та керування ним. Повна форма SSH - Secure Shell - це основний протокол для доступу до мережевих пристроїв і серверів через Інтернет.

Він допомагає вам увійти в інший комп'ютер через мережу і дає змогу виконувати команди на віддаленій машині. Ви можете переміщати файли з одного комп'ютера на інший. Протокол SSH шифрує трафік в обох напрямках, що допомагає запобігти торгівлі, прослуховуванню та крадіжці паролів[7].

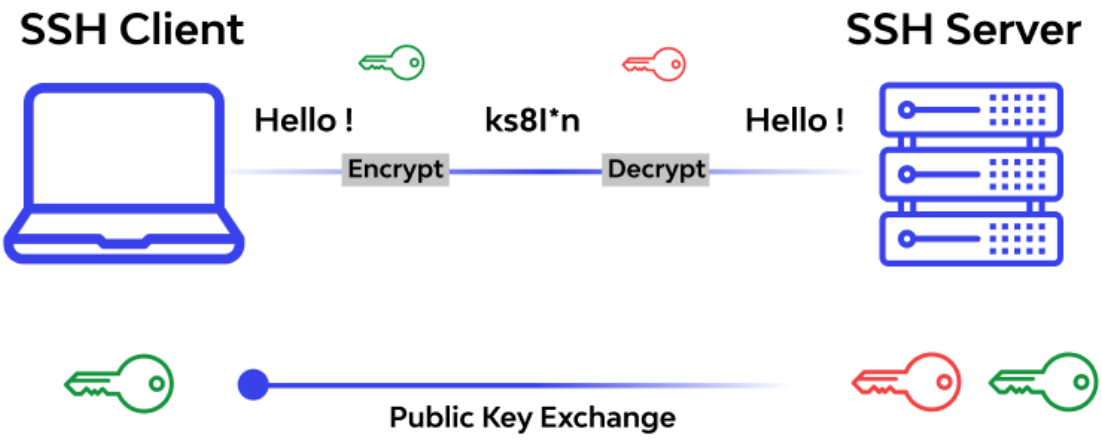

Рисунок 1.3 – Приклад роботи протоколу SSH

Порівняємо ці два протоколи нижче у Таблиці 1.

Таблиця 1 – Порівняльна характеристика Telnet та SSH

Telnet	SSH
Telnet використовує порт 23, який був розроблений спеціально для локальних мереж	SSH за замовчуванням працює на порту 22, який ви можете змінити.
Жодних привілеїв для аутентифікації користувача не передбачено.	SSH - безпечніший протокол, тому для автентифікації він використовує шифрування з відкритим ключем.
Підходить для приватних мереж	Підходить для публічних мереж
Telnet передає дані у вигляді простого тексту.	Для надсилання даних слід використовувати зашифрований формат, а також використовувати безпечний канал.
Telnet вразливий для атак безпеки.	SSH допоможе вирішити багато проблем безпеки Telnet.
Потрібно низьке використання смуги пропускання.	Потрібна висока пропускна здатність.

Дані, надіслані з використанням цього протоколу, не можуть бути легко інтерпретовані хакерами.	Імена користувачів і паролі можуть бути схильні до шкідливих атак.
Використовується в операційних системах Linux і Windows.	Використовується усіма популярними операційними системами.

Отже, можно зробити висновки про недоліки та переваги даних протоків.

Переваги телнета:

– Цей протокол може використовуватися для надсилання та отримання інформації.

– Підтримує аутентифікацію користувачів.

– Можлива спільна робота кількох користувачів.

– Цей віддалений вхід заощадив години досліджень.

Переваги SSH:

– Він безкоштовний для некомерційного використання.

– Версія з відкритим вихідним кодом зазнала поліпшень, таких як виправлення помилок, та пропонує багато додаткових функцій.

– SSH може пропонувати кілька послуг, використовуючи одне й те саме з'єднання.

– SSH допомагає безпечно тунелювати небезпечні додатки, такі як SMTP, IMAP, POP3 і CVS.

– Тунелювання портів ефективно працює для простих VPN.

– Він пропонує надійну автентифікацію та безпечний зв'язок незахищеними каналами.

– SSH дозволяє користувачам безпечно входити на інший комп'ютер через небезпечну мережу.

– Забезпечте конфіденційність ваших даних за допомогою надійного шифрування.

– Цілісність зв'язку виконана таким чином, що її не можна змінити.

– Аутентифікуйте документи, що засвідчують особу відправників та одержувачів.

– Дозволяє повертати, пересилати чи шифрувати інші сеанси на основі TCP/IP.

– Дозволяє користувачеві переглядати вміст каталогів, редагувати файли та віддалено отримувати доступ до користувацьких додатків бази даних.

Недоліки Telnet:

– Telnet не є ідеальним протоколом для передачі рухів курсору або інформації GUI.

– Не підтримує транспортування зашифрованих даних. Замість цього він підтримує тільки відомий номер порту.

– Динамічний порт не підтримується.

– Telnet не шифрує жодних даних, що надсилаються через з'єднання.

– Номери портів можуть бути використані.

– Відображати тільки текст і цифри, без графіки та кольору.

Недоліки SSH:

– Він не призначений для передачі переміщень курсору або інформації про переміщення графічного інтерфейсу.

– SSH не вирішує всі проблеми TCP, оскільки TCP працює нижче SSH.

– SSH не може захистити користувачів від атак через інші протоколи.

– Цей протокол не захищає від троянських коней або вірусів[8].

1.4 Способи підключення віддаленого сервера до мобільного додатку

Відсоток мобільних пристроїв, які використовуються для доступу до інформаційних систем, зростає з кожним роком порівняно з настільними. Це означає, що обсяг мобільного трафіку вже перевищив трафік настільних систем.

Отже, які ж варіанти зв'язку між вашим мобільним пристроєм і віддаленим сервером?

1. Пряме з'єднання TCP/IP

Тут мобільний додаток безпосередньо підключається до бази даних сервера для отримання відповідей. Для забезпечення конфіденційності можна використовувати SSH або інший тип шифрування. У цьому випадку мобільний застосунок використовує сокети TCP/IP (або UDP, якщо необхідно), підключені до сервера.

Особливості:

– Пряме підключення з використанням стандартного TCP/IP telnet/SSH;

– Верифікація користувача за допомогою простого логіна/пароля;

– Команди користувацького інтерфейсу перетворюються на команди сервера.

Переваги:

- Простота;
- Середні витрати на розробку back-end додатків;
- Відсутність додаткових витрат на обслуговування додаткового обладнання.

Недоліки:

- Складна масштабованість;
- Деякі порти TCP/IP можуть бути закриті або відключені, що призведе до припинення роботи мобільного додатка;
- Тільки одна функціональність для стандартного протоколу telnet/SSH.

2. Веб-додаток

Мобільні додатки можуть бути побудовані як веб-додатки і можуть використовувати різні фреймворки, такі як Electron і NW.js.

Особливості:

– Використання веб-сервісів через спеціалізовані API;

– Верифікація користувача за допомогою простого/безпечного логіна/пароля, як у настільному веб-додатку;

– Команди користувацького інтерфейсу перетворюються на API-запити.

Переваги:

- REST API;
- Простота розробки на стороні клієнта;
- Можливість використання зовнішніх ресурсів (серверів, баз даних) з підключенням з боку сервера;
- Відсутність додаткових витрат на утримання додаткового обладнання на back-end;
- Бекенд може бути розміщений на хмарному або виділеному сервері;
- Легко масштабується за допомогою хмарних технологій.

Недоліки:

- Складніший цикл розробки;
- У деяких випадках може бути порівняно повільним через користувацький інтерфейс HTML/JS, наприклад, 2d і 3d графіка.

3. Хмарна інтеграція

За допомогою хмарної інтеграції можна дуже легко підключити ваш мобільний додаток до back-end. Завдяки сучасним хмарним системам, можна створити масштабовану, надійну і безпечну внутрішню частину для будь-якого мобільного застосунку. Це рішення також буде економічно ефективним для будь-якої кількості мобільних користувачів.

Особливості:

– Підключення до хмари з використанням її сервісів (API);

– Верифікація користувача за допомогою простого/безпечного логіна/пароля;

– Легка масштабованість API, як вгору, так і вниз;

– Версійність API;

– Команди користувацького інтерфейсу перетворюються на команди хмарного API.

Переваги:

- REST API;
- Простота розробки на стороні клієнта;
- Можливість використовувати зовнішні ресурси (сервери, бази даних) з хмарними сервісами;
- Доступ до всіх хмарних сервісів;
- Отримання доступу до кожного комп'ютера в хмарі відповідно до прав користувача;
- Висока масштабованість;
- Виняткова надійність і міцність.

Недоліки:

- Вимагає багато часу на налаштування;
- Нелегко змінити хмарного провайдера за необхідності;
- Back-end розробники мають бути кваліфікованими;

4. GraphQL

GraphQL забезпечує новий підхід до розробки API. Його можна порівняти і протиставити REST та іншим архітектурам веб-сервісів. GraphQL дає змогу вам визначити структуру необхідних даних проєкту. Та сама структура буде повернута сервером без необхідності зміни API, що запобігає передачі надлишкових даних.

Функції:

– Надсилання та отримання нових даних без зміни структури API;

– Сервери GraphQL доступні приблизно на 10 мовах програмування.

Переваги:

- GraphQL підтримує читання і запис (зміну) даних;
- Можливість підписки на зміни даних у режимі реального часу.

Недоліки:

- Необхідно вивчати відносно нову технологію;
- Зниження ефективності веб-кешування результатів запитів.
- Для інтеграції GraphQL необхідно звернутися до документації з GraphQL[9].

1.5 Постановка задачі

Мета кваліфікаційної роботи магістра – розробка інформаційно-комунікаційної технології керування віддаленим консольним сервером за допомогою мобільного додатку. Для досягнення поставленої мети необхідно виконати такі завдання:

1) Розробити інформаційну модель віддаленого доступу до консольного серверу.
2) Обрати засоби програмної реалізації.
3) Виконати програмну реалізацію мобільного додатку для керування віддаленим консольним сервером.
4) Протестувати коректність роботи мобільного додатку.

2 ВИБІР МЕТОДІВ РІШЕННЯ ЗАВДАННЯ

2.1 Основні переваги операційної системи Android

Зі зростанням використання смартфонів мобільні додатки стали невід'ємною частиною успіху будь-якого бізнесу, оскільки вони допомагають організаціям залучати більше клієнтів через мобільні пристрої. Кількість людей, які використовують смартфони, зростає, і майже всі вважають за краще користуватися смартфонами, чи то покупки в Інтернеті, чи то читання погоди, чи то пошук місцевих послуг, чи то читання блогів, чи то що-небудь ще.

Смартфони зробили нову революцію в цифровому світі, оскільки вони допомогли підприємствам швидко і більш ефективно збільшити продажі у сфері електронної комерції. Якщо говорити про мобільні додатки, то Android займає близько 73% мобільного ринку в усьому світі, що робить його найбільш використовуваною мобільною операційною системою у світі.

Наразі в магазині Google Play Store налічується понад 3,3 мільйона застосунків, і це число продовжує зростати, що призвело до значного зростання розробки мобільних застосунків для Android. У результаті багато компаній почали переводити свої додатки на Android, щоб ефективно охопити ширшу аудиторію.

Будучи платформою з відкритим вихідним кодом, Android надає розробникам свій комплект засобів розробки програмного забезпечення (SDK) і дає змогу їм створювати, тестувати й розгортати застосунок за мінімальний час і зусилля, тим самим скорочуючи витрати на розробку і час розробників. Це дає змогу розробникам швидко впроваджувати конструктивні особливості та отримувати готові до запуску мобільні додатки, які легше задовольняють потреби клієнтів.

На додачу до цього, Android постачається з настроюваними функціями та можливостями, які роблять процес розробки застосунків для Android швидшим, більш плавним і економічно ефективнішим. Процес розробки додатків для Android можна розділити на чотири етапи:

– дизайн додатка
– розробка програми
– тестування застосунку
– Розгортання застосунку

Android має численні можливості та вбудовані потужні інструменти, які дають змогу розробникам застосунків для Android швидко реалізувати функції, додати нові функціональні можливості, спроєктувати застосунок, протестувати функції та швидко зробити застосунок готовим до реального розгортання, скоротивши водночас вартість і час розроблення, порівнюючи з платформою iOS. Android пропонує економічно ефективні та надійні рішення з розробки додатків.

Google Play Store постачається з простим у реалізації, зручним для людей і низьким бар'єром для швидкого надсилання додатка на його платформу. На платформі Android досить легко отримати схвалення вашого додатка в Google Play Store і зробити його готовим до використання користувачами, оскільки він слідує філософії Google.

І навпаки, коли йдеться про платформу Apple, деякі зі встановлених правил несумісні з тими, що діють на iOS. Іноді затвердження займає набагато більше часу, ніж очікувалося, що призводить до затримок у запуску додатків і подальших збитків для бізнесу.

Розробники можуть швидко інтегрувати або оновити широкий спектр інструментів і функцій для зручного управління даними, а потім пов'язати їх з модифікаціями[10].

Безпека - одна з найважливіших і найбільших проблем. Якщо говорити про платформу застосунків Android, то вона надає вбудовані потужні функції безпеки та регулярні оновлення від Google, щоб зробити застосунок безпечнішим і масштабованішим.

Порівняно з платформою iOS, Android надає різні функції безпеки та оновлення, щоб застосунок працював гладко і без проблем. Таким чином, можна

сказати, що безпека - це ще одна обов'язкова перевага розробки додатків для Android.

Android має найсучаснішу та найоновленішу систему, завдяки якій жодна шкідлива програма або загроза безпеці не може навіть виявити структуру даних. Крім того, він постійно надсилає push-повідомлення, щоб допомогти користувачам постійно оновлювати свої системи або додатки.

Використання останніх технологічних досягнень, доступних розробникам Android, допоможе вам отримати максимальну віддачу від вашого додатка. Платформа Android дає змогу розробляти висококласні додатки з використанням сучасних технологій, як-от AR і VR.

Завдяки мові Java та широкій популярності цієї мови, Android надзвичайно інтуїтивно зрозумілий у використанні. Крім того, Java має хороший набір бібліотек і набагато краще підходить для створення висококласних додатків. Таким чином, можна сказати, що розробникам не потрібно бути експертами в декількох складних мовах, щоб створювати додатки Android[11].

2.2 Вибір мови програмування

Java - це об'єктно-орієнтована мова програмування з відкритим вихідним кодом, яка працює на будь-якій операційній системі. Сьогодні Java перетворилася на цілу екосистему, що складається з трьох основних компонентів, які можна записати один раз і запустити будь-де.

Java Development Kit. Встановивши додаток Notebook на свій комп'ютер, ви можете використовувати цей набір для написання коду для будь-якого рішення.

Java Runtime Environment. Це засіб розповсюдження програмного забезпечення, що складається з бібліотеки класів Java, віртуальної машини Java та засобу конфігурації.

Integrated Development Environment. Це набір інструментів для редагування, компіляції та запуску коду.

Такий універсальний стек технологій, який справедливо зараховують до сильних сторін Java, робить її зручною мовою для використання в багатьох типах проектів.

Типи проектів, в яких використовується Java

– Відеоігри. Багато ігрових рушіїв використовують C# та C++. Однак для новачків, які хочуть навчитися розробляти ігри та графіку з нуля, Java є кращим вибором. Особливо, якщо згадати численні фреймворки і бібліотеки (OpenGL, LibGDX та інші), які пропонує Java саме для цього.

– Настільні додатки. Інша група фреймворків на основі Java (Griffon, JavaFX, Swing, AWT) настільки досконалі, що розробка користувальницького інтерфейсу (який має вирішальне значення для деревовидних додатків або додатків з 3D-графікою) є простою справою. AcrobatReader та ThinkFree є гарними прикладами такого використання Java.

– Додатки та веб-сайти. Створення цих програмних продуктів здійснюється за допомогою Struts, Hibernate, Spring, Servlets, JSF та інших фреймворків на основі Java. Не дивно, що Amazon, LinkedIn, AliExpress, Wayfair та інші сайти вищої ліги використовують Java для їх створення.

– Мобільні додатки. Будучи офіційною мовою програмування для Android, Java використовується у майже 50% мобільних додатків, що працюють на цій операційній системі, включаючи такі великі проєкти, як Netflix, Uber, Google Earth та Tinder. Крім того, спеціалізований крос-платформний фреймворк J2ME може бути використаний для створення відповідних продуктів, які функціонують на всіх типах смартфонів з підтримкою Java.

– Корпоративне програмне забезпечення. Переваги Java, що гарантують високу продуктивність і масштабованість продуктів, побудованих на ній, роблять її другим найкращим вибором для побудови ERP і CRM систем, а також різноманітних професійних рішень у фінансовій, банківській, роздрібній торгівлі, логістиці та інших галузях.

– Big Data. Хоча традиційно технологія великих даних базується на Python, вона також широко використовує Java, особливо її інструменти Hadoop і Deeplearning4j.

– Вбудовані системи. SIM-карти, приводи Blue-Ray, а також різні процесори та мікросхеми, що використовуються в інтегрованих електромеханічних системах, підтримуються прошивкою на основі Java.

– Internet of Things. Оскільки Java не залежить від платформи, вона ідеально підходить для розробки IoT. Коли ви напишете в ньому код, то зможете запускати його на багатьох гаджетах, пов'язаних з середовищем IoT[12].

Існує кілька основних і фундаментальних переваг Java:

1. Простота

Java проста у використанні, а також легша у використанні, написанні, компіляції, налагодженні та вивченні, ніж альтернативні мови програмування. Java має менш складні концепції в порівнянні з C++, в результаті чого в Java також використовується автоматичне виділення пам'яті і збір сміття.

2. Об'єктно-орієнтований

Java є повністю об'єктно-орієнтованою мовою "out of the box". Це дозволяє створювати стандартні програми та код багаторазового використання.

3. Незалежний від платформи

Код Java може виконуватися на будь-якій машині, яка не потребує встановлення спеціального програмного забезпечення, але на машині має бути присутня JVM.

4. Розподілені обчислення

Розподілені обчислення включають в себе можливість запуску і виконання завдань на декількох комп'ютерах в мережі, що працюють спільно. Вона допомагає створювати додатки, які працюють в розподілених мережах і можуть сприяти підвищенню функціональності як даних, так і додатків.

5. Безпека

У інших мовах програмуання явні вказівники, що зберігають в пам'яті адреси значень, надають хакеру готовий шлях для доступу до рішення. Ці модулі відсутні в Java, що захищає систему від спроб злому. Крім того, правила доступу можуть бути визначені для кожного класу в додатку за допомогою Java-специфічних менеджерів безпеки.

6. Розподіл пам'яті

У мові Java концепція управління пам'яттю полягає в тому, що вона поділяється на дві частини, одна з яких є купою, а інша - стеком. Кожного разу, коли ми оголошуємо змінну, JVM виділяє пам'ять для змінних та масивів для програм зі стеку або простору купи. Це допомагає впорядковувати та зберігати інформацію, а також легко її відновлювати.

7. Багатопоточність

При написанні коду програмісти розглядають потоки як мінімальні одиниці обробки. Перемикання між кількома з них займає певний час, оскільки вони використовують один і той самий об'єм пам'яті. За допомогою Java ви можете запускати кілька потоків одночасно і швидше просуватися по вашому проекту.

8. Різноманітні API

Java здатна запропонувати різні API для розробки додатків. Java API (Application Programming Interface) - це не що інше, як багатий набір команд або методів, які визначають спосіб взаємодії між різними видами діяльності, такими як підключення до бази даних, робота в мережі, введення/виведення, розбір XML, утиліти та багато іншого.

9. Надійність

Java є однією з найбільш надійних мов програмування для розробників та програмістів. Це робить Java більш надійною. Компілятори Java здатні виявляти будь-які помилки кодування. Існує також багато інших наборів функцій, таких як обробка виключень і збір сміття, що робить Java більш надійною[13].

2.3 Вибір середи розробки

Android Studio допомагає створювати додатки для Android та керувати ними за допомогою комп'ютера. Вибір правильного інструменту для створення додатків для Android зараз має вирішальне значення. А створювати додатки ще ніколи не було так просто, все завдяки Android Studio.

Android Studio відома як IDE, також відома як інтегроване середовище розробки. Ця IDE діє як майстерня, що містить інструменти, які допомогають створити Android -додаток. Android Studio показує огляд дизайну вашого додатку в реальному часі. Також перевіряє наявність помилок у вашому машинному коді і допомагає виправити їх, надаючи необхідні пропозиції. Ви можете побачити розробку вашого додатку за допомогою Android Studio, змоделювавши додаток на мобільному або комп'ютерному пристрої[14].

Чотири ключові елементи складають Android Studio, і кожен з них працює як єдине ціле, допомагаючи вам розробляти програми для Android. Наведений нижче короткий огляд цих ключових елементів.

1) Набори для розробки програмного забезпечення (SDK).

В Android Studio елемент SDK включає в себе включення основних інструментів, таких як документація та фреймворки. Цей елемент також включає в себе емулятор, який допоможе вам перевірити прогрес у розробці.

2) Інтегроване середовище розробки (IDE)

IDE допомагає розробнику писати код, необхідний для створення Android-додатків. IDE допомагає комбінувати набори для розробки та керувати ними відповідно до ваших потреб за допомогою плагінів.

3) Мова програмування

Android Studio також включає в себе Java та Kotlin, дві офіційні мови програмування. Ви також можете використовувати C і C ++ за допомогою стороннього інструменту.

4) Бібліотеки

Бібліотеки допомагають швидко писати код за допомогою автоматичної системи написання коду. Він дозволяє виявляти і переписувати коди помилок. Його система API розуміє шаблон кодування[15].

Android Studio має інтелектуальну систему редагування коду, яка допомагає швидше писати код. Ця система може легко виявити будь-які помилки у вашому коді, позначити їх та запропонувати зміни. Він допомагає вам бути більш ефективними, автоматично завершуючи ваш код. Він зчитує попередні шаблони написання коду і видає коди завершення відповідно до раніше написаного коду.

Android Studio включає в себе емулятор Android, який допоможе вам протестувати ваш додаток під час розробки. Емулятор - це інструмент, який допомагає запустити Android-додаток з комп'ютера. За допомогою емулятора ви можете перевірити апаратну та програмну сумісність вашого додатку з будь-якого пристрою.

Функція "Шаблони коду" є чудовим доповненням до Android Studio як для нових, так і для старих розробників. Ці шаблони коду містять різні типи шаблонів коду. Він діє як навігаційна система і допомагає легко писати складний код.

Ще одним корисним доповненням до Android Studio є перевірка помилок. Ця функція допомагає контролювати структурну цілісність коду. Він допомагає виявити слабкі місця у ваших кодах і допомагає їх покращити.

Android Studio також включає в себе інструменти тестування, такі як JUnit4 і Functional-UI, які допоможуть вам перевірити структуру вашого коду. Вони зберігають написаний вами код і створює тест для цього коду в інтерфейсі емулятора. Це допомагає визначити подальшу цілісність програми.

Android Studio пропонує гнучку систему інтерфейсу, яка дає вам повну свободу налаштувань. Ви можете налаштовувати свої проекти та включати різні інтеграції із зовнішніх джерел. І працювати над ними так, як вважаєте за потрібне[16].

Інструмент з графічним інтерфейсом допомагає зробити Android Studio простішою у використанні і допомагає позбутися завдань, над якими працювати менш цікаво і нудно, таких як дизайн макетів, аналіз APK, перекладачі і т.д.

2.4 Вибір бібліотек для реалізацій підключення до мережі

JSch - це чиста Java реалізація SSH2. JSch дозволяє підключатися до sshd-сервера і використовувати переадресацію портів, переадресацію X11, передачу файлів і т.д., а також ви можете інтегрувати його функціонал в свої власні Java-програми. JSch поширюється за ліцензією в стилі BSD.

SSH забезпечує підтримку безпечного віддаленого входу в систему, безпечну передачу файлів, а також безпечну переадресацію TCP/IP і X11. Він може автоматично шифрувати, аутентифікувати та стискати дані, що передаються. Протокол SSH існує у двох несумісних різновидах: SSH1 і SSH2. SSH2 був винайдений для того, щоб уникнути патентних проблем, пов'язаних з RSA (термін дії патенту на RSA закінчився), а також для усунення деяких проблем з цілісністю даних, які були притаманні SSH1, і з ряду інших технічних причин. Протокол SSH2 був стандартизований на робочій групі IETF Secure Shell, а проекти, пов'язані з протоколом SSH2, доступні в Інтернеті.

Java Secure Channel - основний пакет. Цей пакет містить всі типи додатків, що використовують бібліотеку. Нижче наведено огляд про найважливіші з них:

JSch - відправна точка, що використовується для створення сесій та управління ідентичностями.

Сеанс - підключення до SSH-сервера. Використовується для налаштування параметрів, переадресації портів і відкриття каналів.

Клас Channel та його підкласи ChannelExec, ChannelShell, ChannelSubsystem для віддаленого виконання команд.

ChannelSftp використовується для безпечної передачі файлів. Також може знадобитися SftpATTRS та реалізація SftpProgressMonitor.

UserInfo та UIKeyboardInteractive мають бути реалізовані для забезпечення зворотного зв'язку з користувачем та/або парольної чи клавіатурно-інтерактивної автентифікації.

Більшість інших класів призначені або для внутрішнього використання, або для спеціального використання, і не є необхідними для звичайної програми.

Короткий алгоритм дії:

1. Отримуємо від користувача налаштування підключення (як параметр виклику або діалоговим вікном).
2. Створюємо сесію і встановлюємо з'єднання.
3. Створюємо Channel (у цьому випадку для інтерактивного виконання команд, хоча є й інші варіанти).
4. "Загортаємо" системні потоки введення і виведення в отриманий канал.

2.5 VirtualBox

VirtualBox – це програмне забезпечення, що імітує комп'ютерну систему і здатне запускати програми так, наче це був би справжній комп'ютер. Зазвичай це програмне забезпечення визначається як "ефективна та ізольована копія фізичної машини". Наразі цей термін також включає в себе віртуальні машини, які не мають прямого еквівалента фізичному обладнанню.

Такі віртуальні машини можуть запускати процеси, обмежені наданими їм ресурсами та абстракціями. Жоден із процесів не може в будь-який момент покинути середовище цього "віртуального комп'ютера".

Один із найпоширеніших методів створення віртуальних машин - запуск операційних систем для їх "тестування". Це дає нам змогу тестувати операційну систему, таку як Ubuntu, з нашого комп'ютера з операційною системою Windows, без необхідності встановлення її безпосередньо на комп'ютер і без будь-якого впливу на конфігурацію основної операційної системи.

VirtualBox є безкоштовним і відкритим вихідним кодом, який може працювати практично на будь-якій хост-операційній системі.

VirtualBox може працювати як у 32-бітних, так і в 64-бітних операційних системах на базі процесорів Intel x86-64. Він працює однаково на всіх хост-системах. Крім того, оскільки одні й ті самі формати файлів і образів можуть використовуватися в декількох операційних системах вузла, віртуальна машина, створена на одному вузлі, може просто працювати на іншому, а гостьові віртуальні машини можна імпортувати й експортувати з використанням формату відкритої віртуалізації (OVF).

Oracle VM VirtualBox Enterprise дає змогу корпоративним IT-фахівцям визначати й підтримувати платформу хоста за замовчуванням для різних бізнес-підрозділів, ролей і вимог, у комплекті з усіма необхідними елементами керування й оновленнями безпеки, у той час як кожен працівник може визначати різні віртуальні машини на різних платформах залежно від їхнього робочого дня та сьогоднішніх потреб.

Така функціональність, як групи віртуальних машин, дає змогу користувачам організовувати віртуальні машини як індивідуально, так і колективно. Запуск, закриття, зупинка, скидання, збереження стану, завершення роботи, вимкнення живлення та інші операції можуть застосовуватися до груп віртуальних машин так само, як і до окремих віртуальних машин.

VirtualBox підтримує гостьовий SMP, USB-пристрої, повну підтримку ACPI, роздільну здатність декількох екранів, вбудовану сумісність з iSCSI і завантаження по мережі PXE.

Virtualbox дає змогу користувачам використовувати віртуалізацію на своїх домашніх комп'ютерах із меншими витратами та з більшою швидкістю. Це допомагає користувачеві створити уявлення операційної системи, знижуючи витрати на обладнання при одночасному підвищенні продуктивності та ефективності[17].

Користувачі одночасно можуть запускати безліч операційних систем на своєму наявному комп'ютері, включно з Microsoft Windows, Mac OS X, Linux і

Oracle Solaris. Користувач може створювати різні платформи або збирати їх на одному сервері для тестування і розробки.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Інформаційна модель

Для розуміння того, як працює додаток я створила діаграму діяльності. Для її створення я використовувала безкоштовний ресурс draw.io.

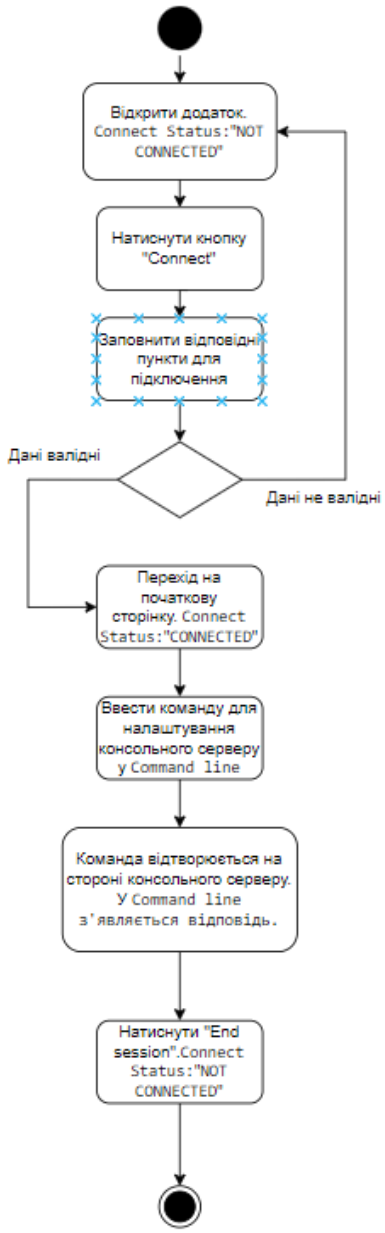

Рисунок 3.1 – UML-діаграма діяльності

На даній схемі представлений механізм роботи мобільного додатку. Для початку роботи необхідно відкрити додаток, після чого користувач потрапляє на головну сторінку, де можна побачити кнопки для підключення та завершення сесії, вікно для вводу команд та статус підключення (на цьому етапі він «NOT CONNECTED»).

Далі для того, щоб здійснити підключення до консольного серверу, користувач повинен натиснути кнопку «Connect», після чого він потрапляє до вікна з текстовими полями для вводу відповідних даних для віддаленого підключення: «Username», «Host», «Port», «Password». Для підключення користувач повинен заповнити їх валідними значеннями на натиснути кнопку «Connect».

Якщо значення були введені невірно, користувач повертається до головної сторінки, статус підключення залишається незмінним, тобто «NOT CONNECTED».

Якщо значення введено вірно, користувач потрапляє до головної сторінки, де бачить статус підключення «CONNECTED». Це значить, що було здійснено успішне підключення до консольного серверу за допомогою протоколу SSH, і тепер користувач має доступ до налаштування відповідного обладнання зі свого смартфону, ввівши потрібні команди у «Command line».

Щоб завершити сеанс та здійснити відключення від консольного серверу необхідно натиснути кнопку «End session». Після успішного від’єднання користувач побачить відповідний статус підключення «NOT CONNECTED».

3.2 Опис класів

Код програми наведений в додатку даної кваліфікаційної магістерської роботи. Для кращого розуміння проекту у даному розділі описане призначення кожного класу проекту. Для початку розглянемо основний функціонал додатку.

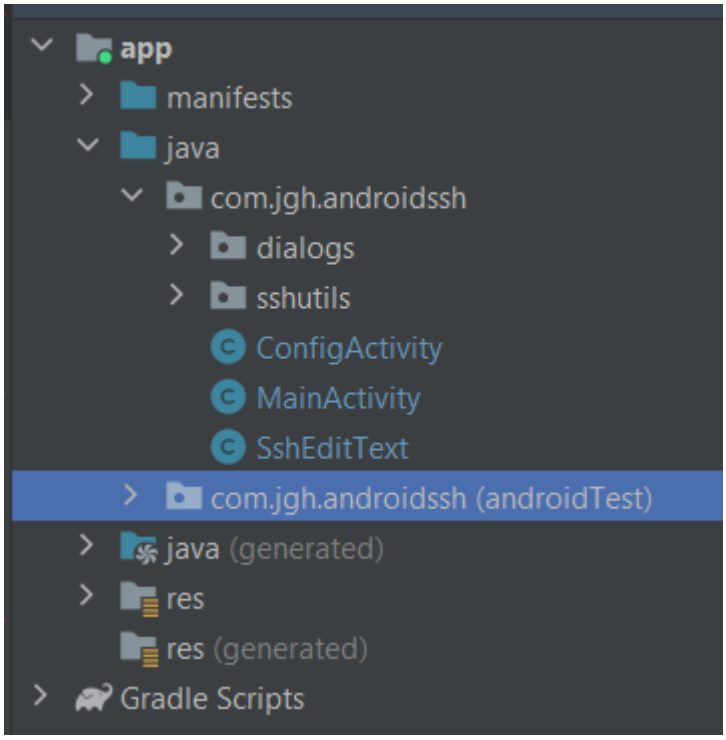

Рисунок 3.2 – Класи, що описують основний функціонал додатку

Клас Main activity відповідає за підключення до SSH-сервера та запуску командної оболонки.

Клас EditText створений для моделювання терміналу оболонки SSH. Основною особливостю цього класу, у порівнянні з базовим EditText, є неможливість видалення (backspace) символів з рядків вище нижнього рядка, як у терміналі.

Класс ConfigActivity – клас, що містить в собі методи для роботи з Menu.

Далі хочу розглянути класи, що слугують SSH-утілітами.

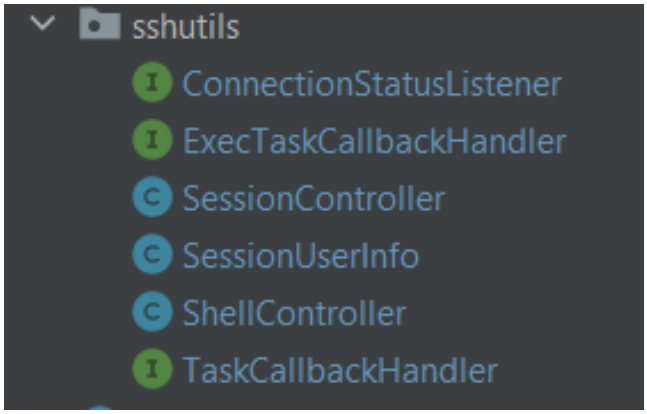

Рисунок 3.3 – SSH Utils

Interface ConnectionStatusListener обробляє процеси відключення та підключення сеансу.

Class SessionController - контролер для сеансів Jsch SSH. Усі SSH-підключення виконуються через цей клас.

Class SessionUserInfo - клас для зберігання інформації, яка використовується для встановлення з'єднань сеансу JSch. Реалізує інтерфейс UserInfo бібліотеки JSch. Використовується для відкриття сесії, підключення за допомогою імені користувача, пароля та інформації про адресу хоста.

Class ShellController - контролер для SSH shell-подібного процесу між локальним пристроєм і віддаленим сервером SSH. Підтримує відкритий канал до віддаленого сервера та передає дані між локальними пристроями і дистанційними.

Interface TaskCallbackHandler - інтерфейс для списків віддаленого каталогу. Містить зворотні виклики до реалізації функціональності для запуску та очікування віддаленого завершення процес створення списку.

3.3 Результат роботи мобільного додатку

Для демонстрації роботи мобільного додатку, створеного у ході виконання кваліфікаційної магістерської роботи, я використала емулятор смартфону в Android Studio та Virtual Box з операційною системою Ubuntu в якості віддаленого консольного серверу.

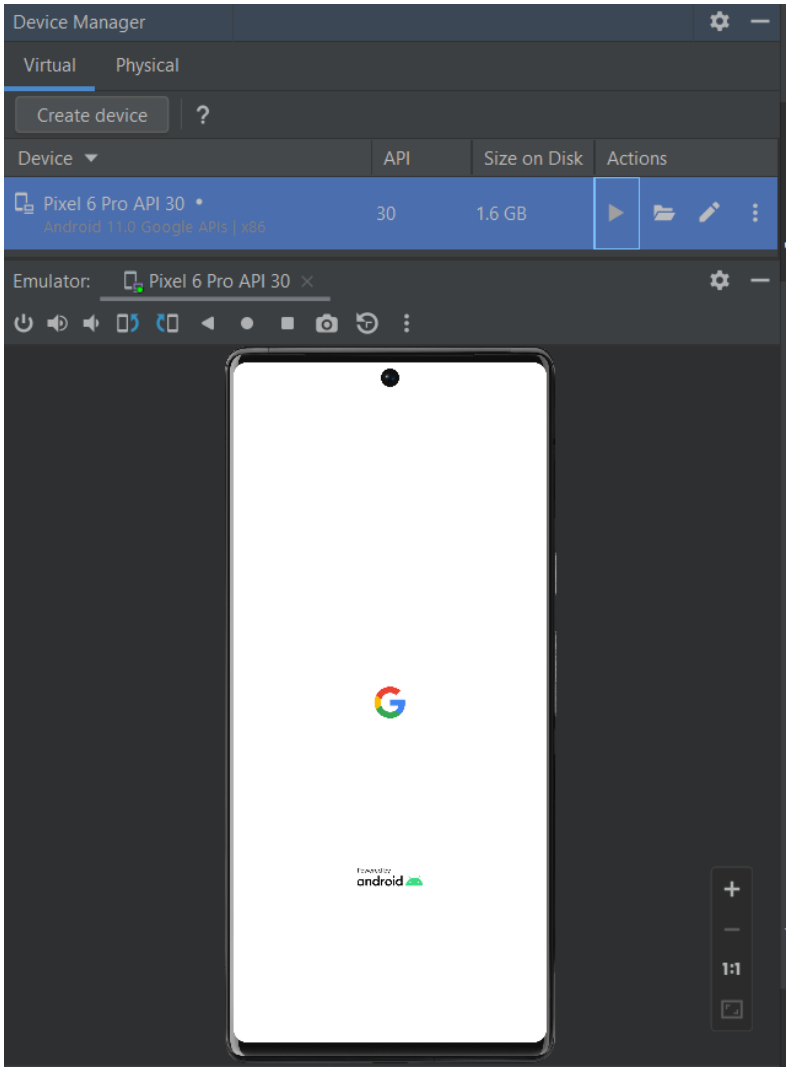

Рисунок 1.4 – Емулятор смартфону з ОС Android

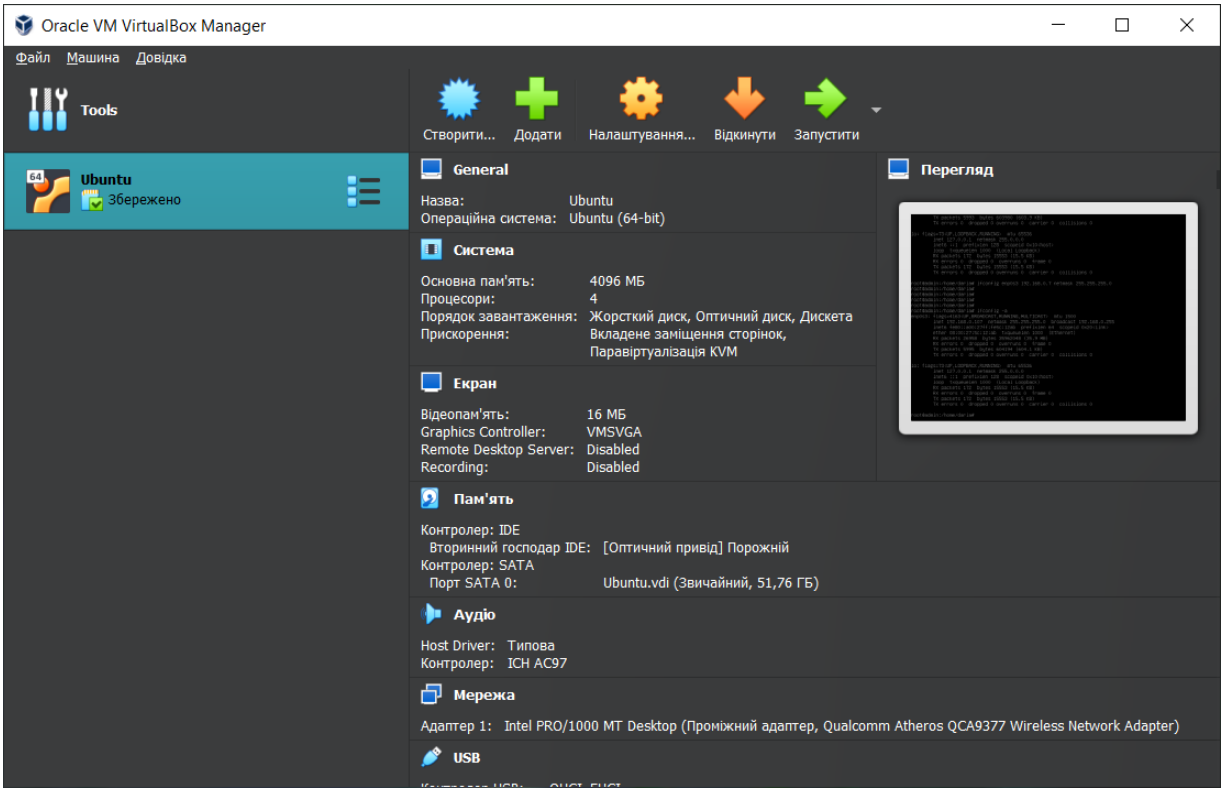

Рисунок 3.5 – Virtual Box з ОС Ubuntu

Відкривши додаток користувач потрапляє на головну сторінку, де бачить статус підключення, вікно для вводу команд, кнопки для встановлення з`єднання та для завершення сесії.

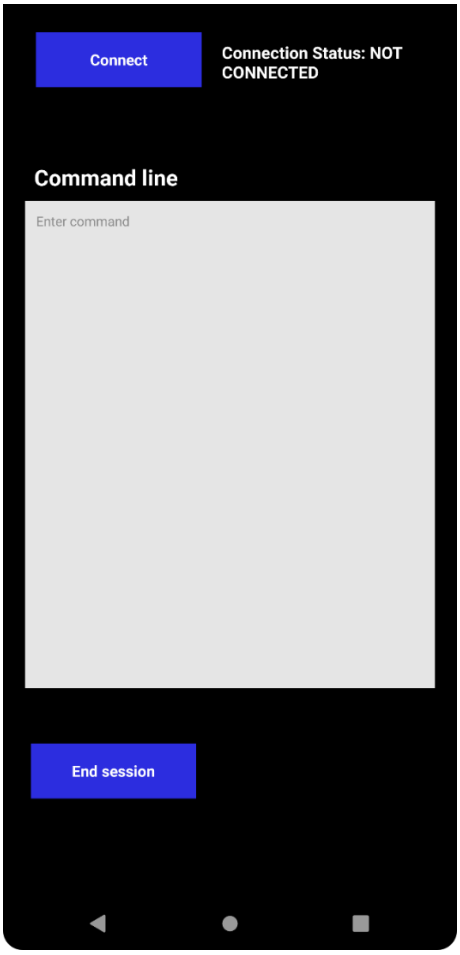

Рисунок 3.6 – Головна сторінка додатку

Щоб підключитися до потрібного обладнання, у моєму випадку до Virtual Box, необхідно натиснути кнопку «Connect», після чого користувач потрапляє на сторінку, де бачить поля для вводу «Username», «Hostname», «Port», «Password», які необхідні для встановлення з’єднання нашого смартфону з віддаленим сервером.

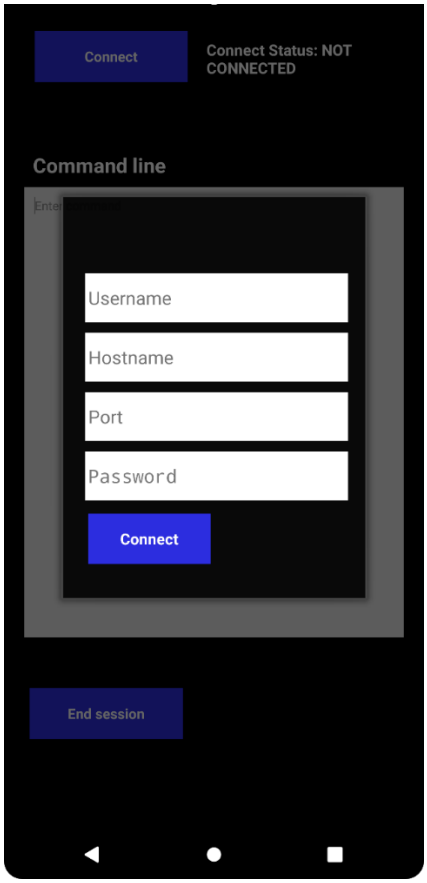

Рисунок 3.7 – Вікно для вводу даних

Для того, щоб дізнатися IP-адресу віртуальної машини, я виконала наступну команду:

```
root@admin:/home/daria# ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.0.106  netmask 255.255.255.0  broadcast 192.168.0.255
        inet6 fe80::a00:27ff:fe5c:12ab  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:5c:12:ab  txqueuelen 1000  (Ethernet)
        RX packets 26810  bytes 35950508 (35.9 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 5886  bytes 585046 (585.0 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 172  bytes 15553 (15.5 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 172  bytes 15553 (15.5 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Рисунок 3.8 – Конфігурація Virtual Box

Після заповнення всіх полів користувач має натиснути кнопку «Connect», щоб відбулося підключення смартфону до консольного сервера.

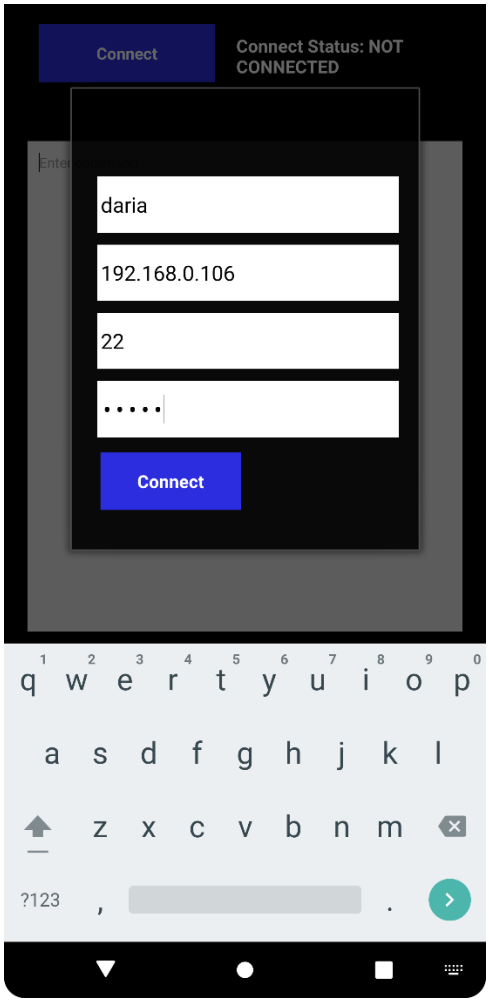

Рисунок 3.9 – Заповнення даних для встановлення з'єднання

Якщо всі дані валідні, користувач опиняється на головній сторінці додатку та може побачити, що «Connection Status: CONNECTED». Це означає, що процес підключення до віддаленого обладнання пройшов успішно і він тепер має можливість для його налаштування, прописуючи необхідні команди у Command Line.

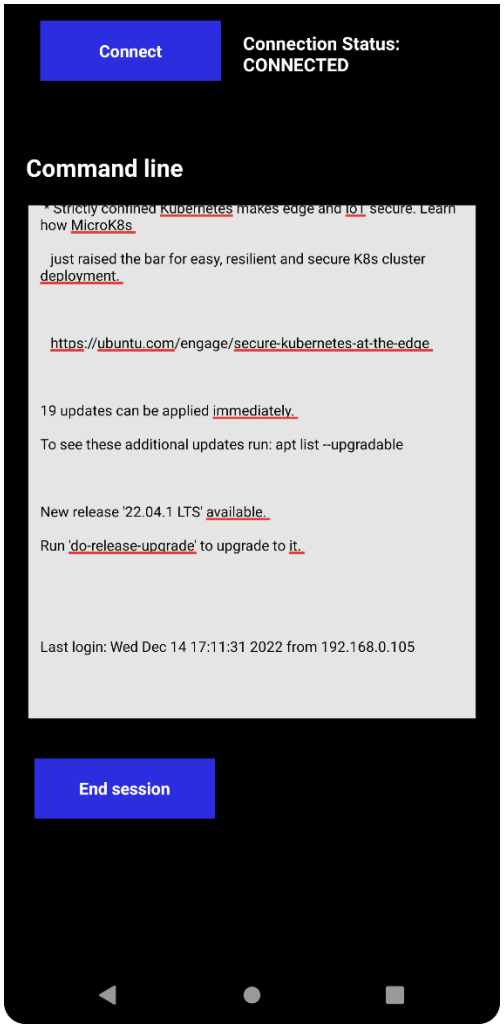

Рисунок 3.10 – Головна сторінка після успішного підключення

Для тестування того, що за допомогою додатку можна віддалено налаштовувати та керувати консольним серсером, я вирішила змінити статичну IP-адресу одного з інтерфейсів віртуальної машини, виконавши наступні команди у Command line нашого мобільного додатку:

Ifconfig -a //команда, що відображає інформацію про всі інтерфейси в системі

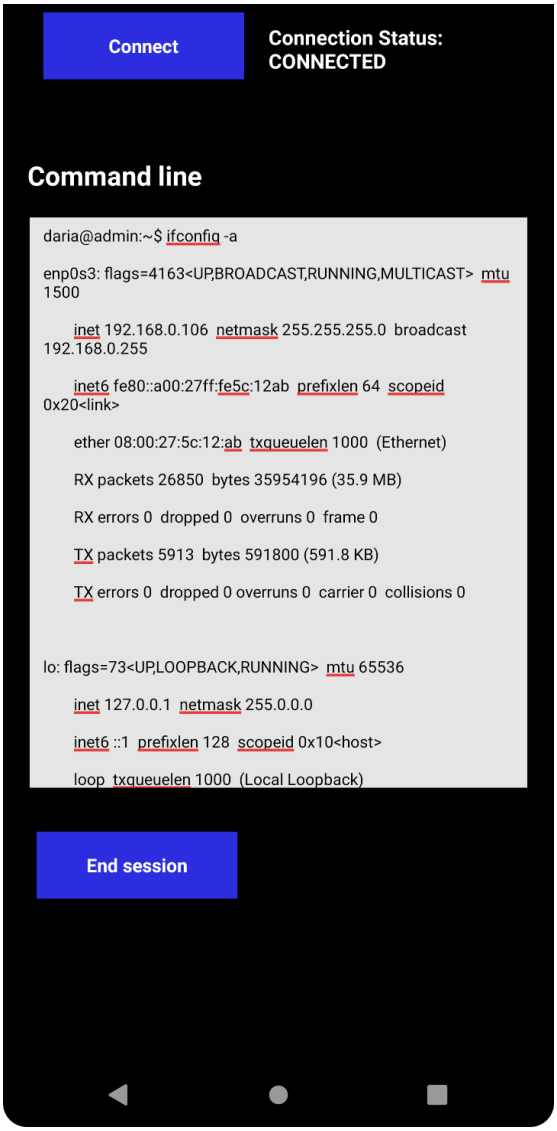

Рисунок 3.11 – перевірка даних інтерфейсу

Після чого виконала команду «sudo su», яка дозволяє нам використовувати обліковий запис і пароль для виконання системних команд із привілеями root, та ввела password.

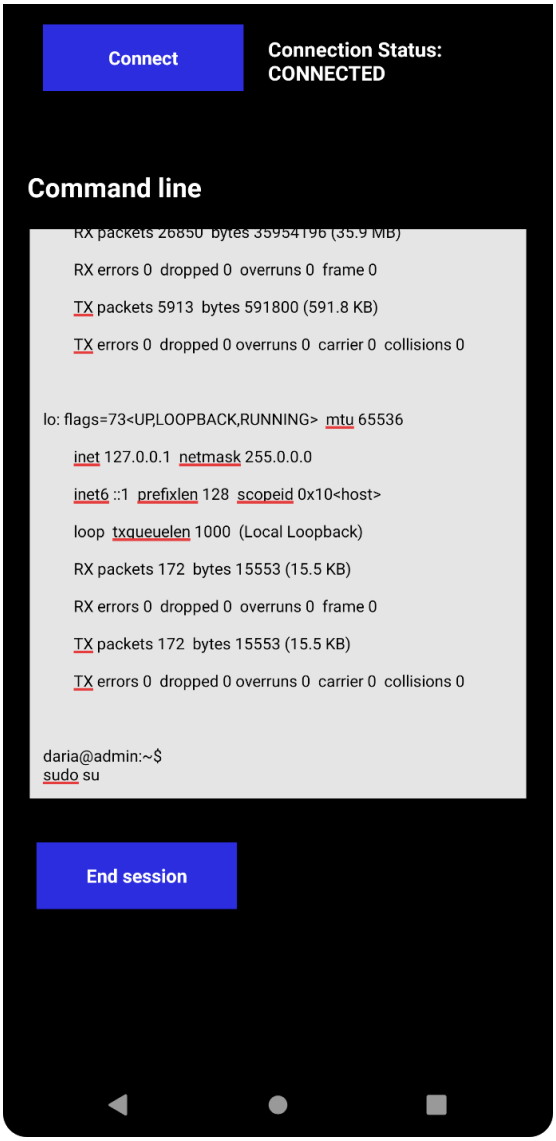

Рисунок 3.12 – Команда для входу до привілейованого режиму

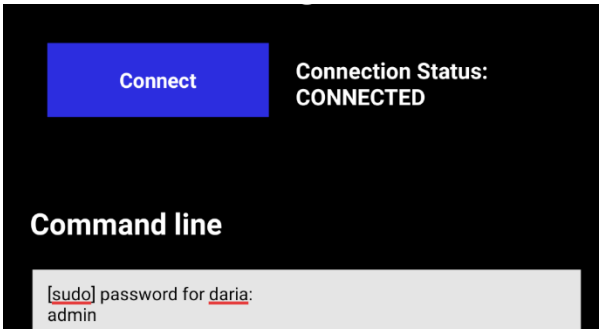

Рисунок 3.13 – Ввод паролю для входу в режим root

Далі, успішно перейшовши до привілейованого режиму, змінила IP-адресу за домогою команди: Ifconfig enp0s3 192.168.0.107 netmask 255.255.255.0. Тут Ifconfig служить для призначення інтерфейсу адреси, команда enp0s3 є скороченим позначенням інтерфейсу: «en» означає Ethernet, «p0» — номер шини карти Ethernet, а «s3» — номер слота. За допомогою netmask призначаємо маску мережі.

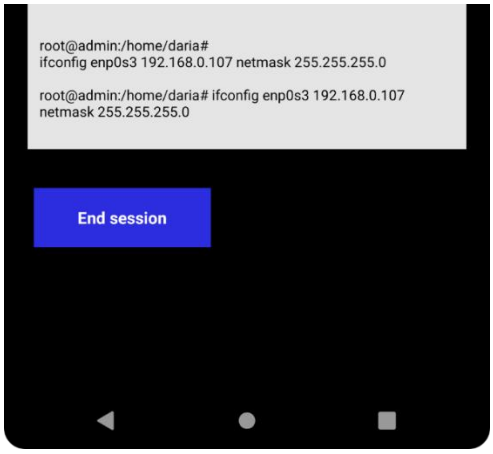

Рисунок 3.14 – Зміна IP-адреси інтерфейсу

Потім перевірила чи справді змінилася Ip-адреса на віртуальній машині, знову виконавши команду Ifconfig -a.

```
root@admin:/home/daria# ifconfig -a
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.0.107  netmask 255.255.255.0  broadcast 192.168.0.255
        inet6 fe80::a00:27ff:fe5c:12ab  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:5c:12:ab  txqueuelen 1000  (Ethernet)
        RX packets 26958  bytes 35962048 (35.9 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 5995  bytes 604194 (604.1 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 172  bytes 15553 (15.5 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 172  bytes 15553 (15.5 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Рисунок 3.15 – Перевірка правильності зміни IP-адреси

Як бачимо, у результаті IP-адреса змінилася з початкової 192.168.0.106 на 192.168.0.107.

Для того, щоб продовжити налаштування, необхідно виконати підключення вже з новою адресою. Щоб від’єднатися від консольного серверу, потрібно натиснувши кнопку «End session». Після цього можна побачити, що «Connection Status: NOT CONNECTED», а Command line порожня, отже ми успішно від’єдналися від серверу.

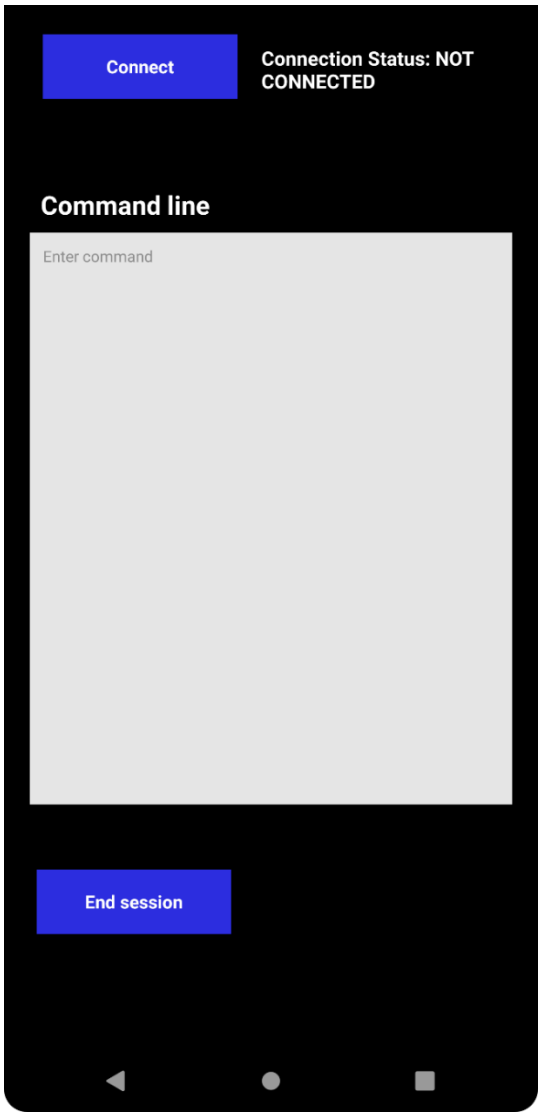

Рисунок 2 – Результат завершення попереднього сеансу

Під’єднуюсь вже з новою IP-адресою.

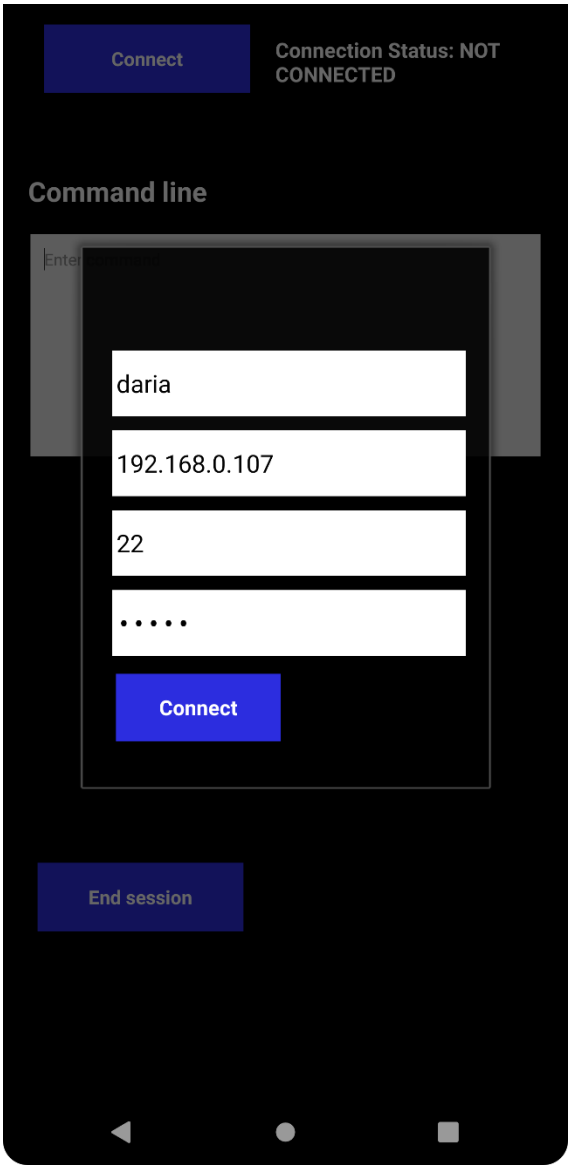

Рисунок 3.17 – Підключення до консольного серверу з новою адресою

Як бачимо з Рис.3.18 з’єднання пройшло успішно.

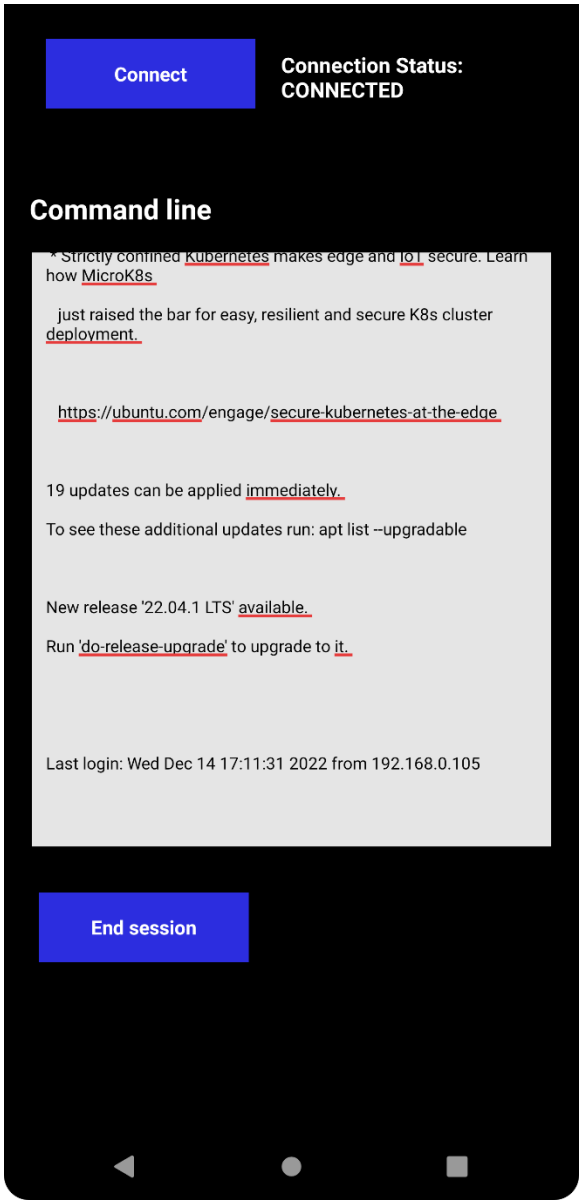

Рисунок 3 – Результат вдалого підключенння до консольного серверу з новою IP-адресою

Отже, вдалося виконати налаштування віддаленого консольного серверу з мобільного додатку.

ВИСНОВКИ

Під час виконання кваліфікаційної роботи магістра було здійснено інформаційний огляд проблематики теми, обрано методи рішення та розроблено мобільний додаток для керування віддаленим консольним сервером. Даний застосунок дає доступ до консольного серверу, можливість керувати й налаштовувати його знаходячись на відстані через мережевий протокол SSH.

В проєкті я обрала Android, як операційну систему, на якій працюватиме мобільний додаток. Основною мовою програмування була обрана Java з використанням бібліотеки Jsch. Для емуляції роботи консольного серверу я використала Virtual Box з операційною системою Ubuntu. Як результат - наданий готовий проєкт, що відповідає меті даної кваліфікаційної роботи магістра.

СПИСОК ЛІТЕРАТУРИ

1. What is remote access [Електронний ресурс] – Режим доступу: https://www.techtarget.com/searchsecurity/definition/remote-access

2. A. Woland, V. Santuka, J. Sanbower, C. Mitchell. Integrated Security Technologies and Solutions. – Indianapolis, Indiana: Cisco Press, 2019. – 136 с.

3. Doug Lowe. Networking All-in-One For Dummies. – Hoboken, New Jersey: Published by: John Wiley & Sons, Inc., 2018. – 56с.

4. M. Meyers. CompTIA Network+ Certification. – London: McGraw-Hill Education, 2018. – 523 с.

5. Out-of-Band Management with Console Servers [Електронний ресурс] – Режим доступу: https://www.blackboxas.no/no-no/page/27149/Ressurser/Tekniske-Ressurser/black-box-forklarer/lan/What-are-Terminal-Servers-Console-Servers

6. Telnet - Terminal Network [Електронний ресурс] – Режим доступу: https://www.javatpoint.com/computer-network-telnet

7. What is SSH and how do I use it? [Електронний ресурс] – Режим доступу: https://www.ucl.ac.uk/isd/what-ssh-and-how-do-i-use-it

8. Adithya Srivastava. SSH Protocol Architecture. – Michigan: Independently Published, 2018. - 25 с.

9. Connect a Remote Server to Your Mobile App [Електронний ресурс] – Режим доступу: https://svitla.com/blog/4-ways-to-connect-a-remote-server-to-your-mobile-app

10. John Horton. Android Programming for Beginners – Third Edition. – Birmingham: Published by Packt Publishing Ltd., 2018. – 591с.

11. Introduction to Android Development [Електронний ресурс] – Режим доступу: https://www.geeksforgeeks.org/introduction-to-android-development/

12. Cay S. Horstmann. Core Java Volume I-Fundamentals: Fundamentals. – Hoboken, New Jersey: Prentice Hall, 2018. – 574с.

13. Herbert Schildt. Java: The Complete Reference, Twelfth Edition. – United States: McGraw Hill, 2021. – 238c.

14. Neil Smyth. Android Studio 3.0 Development Essentials – Android 8 Edition (English Edition). – North Carolina: Payload Media, Inc, 2018, - 191 c.

15. J. Paul Cardle. Android App Development in Android Studio: Java+Android Edition for Beginners. – Manchester: Manchester Academic Publishers, 2018. – 128c.

16. Android Studio Tutorial [Електронний ресурс] – Режим доступу: https://www.javatpoint.com/android-tutorial

17. Robin Catling. The VirtualBox Networking Primer. – London: Proactivity Press, 2020, – 56c.

ДОДАТОК

Код файлу MainActivity.java:

```
package com.jgh.androidssh;
import android.app.Activity;
import android.app.Fragment;
import android.app.FragmentTransaction;
import android.os.Bundle;
import android.os.Handler;
import android.text.Editable;
import android.text.TextWatcher;
import android.util.Log;
import android.view.KeyEvent;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.Window;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;
import com.jgh.androidssh.dialogs.SshConnectFragmentDialog;
import com.jgh.androidssh.sshutils.ConnectionStatusListener;
import com.jgh.androidssh.sshutils.ExecTaskCallbackHandler;
import com.jgh.androidssh.sshutils.SessionController;
import java.util.regex.Pattern;
public class MainActivity extends Activity implements OnClickListener {
    private static final String TAG = "MainActivity";
    private final int REQ_SCP = 1;
    private TextView connectedStatus;
    private boolean isConnected = false;
    private SshEditText commandEdit;
    private Button connectButton, endSessionBtn;
```

```
private Handler handler;
private Handler tvHandler;
private String forLastLine;
@Override
public void onCreate(Bundle bundle) {
    super.onCreate(bundle);
    requestWindowFeature(Window.FEATURE_NO_TITLE);
    setContentView(R.layout.activity_main);
    connectButton = (Button) findViewById(R.id.enterbutton);
    endSessionBtn = (Button) findViewById(R.id.endsessionbutton);
    commandEdit = (SshEditText) findViewById(R.id.command);
    connectedStatus = (TextView) findViewById(R.id.connectstatus);
    connectButton.setOnClickListener(this);
    endSessionBtn.setOnClickListener(this);
    connectedStatus.setText("Connect Status: NOT CONNECTED");
    handler = new Handler();
    tvHandler = new Handler();
    commandEdit.addTextChangedListener(new TextWatcher() {
        @Override
        public void beforeTextChanged(CharSequence charSequence, int i, int i2, int i3) {
        }
        @Override
        public void onTextChanged(CharSequence charSequence, int i, int i2, int i3) {
        }
        @Override
        public void afterTextChanged(Editable editable) {
            String[] sr = editable.toString().split("\r\n");
            String s = sr[sr.length - 1];
            forLastLine = s;
        }
    });
    commandEdit.setOnEditorActionListener(
            new TextView.OnEditorActionListener() {
```

```
            @Override
            public boolean onEditorAction(TextView v, int actionId, KeyEvent event) {
               if (isEditTextEmpty(commandEdit)) {
                  return false;
               }
               else {
                  if (event == null || event.getAction() != KeyEvent.ACTION_DOWN) {
                     return false;
                  }
                  String command = getLastLine();
                  ExecTaskCallbackHandler t = new ExecTaskCallbackHandler() {
                     @Override
                     public void onFail() {
                        makeToast(R.string.taskfail);
                     }
                     @Override
                     public void onComplete(String completeString) {
                     }
                  };
                  commandEdit.AddLastInput(command);
                  new Thread(new Runnable() {
                     @Override
                     public void run() {SessionController.getSessionController().executeCommand(handler, commandEdit, t, command);
                     }
                  }).start();
                  return false;
               }
            }
         }
      );
   }
   private void makeToast(int text) {
```

```
    Toast.makeText(this, getResources().getString(text), Toast.LENGTH_SHORT).show();
}
private String getLastLine() {
    int index = commandEdit.getText().toString().lastIndexOf("\n");
    if (index == -1) {
        return commandEdit.getText().toString().trim();
    }
    if(forLastLine == null){
        Toast.makeText(this, "no text to process", Toast.LENGTH_LONG).show();
        return "";
    }
    String[] lines = forLastLine.split(Pattern.quote(commandEdit.getPrompt()));
    String lastLine = forLastLine.replace(commandEdit.getPrompt().trim(), "");
    Log.d(TAG, "command is " + lastLine + ", prompt is  " + commandEdit.getPrompt());
    return lastLine.trim();
}
private String getSecondLastLine() {
    String[] lines = commandEdit.getText().toString().split("\n");
    if (lines == null || lines.length < 2) return commandEdit.getText().toString().trim();
    else {
        int len = lines.length;
        String ln = lines[len - 2];
        return ln.trim();
    }
}
private boolean isEditTextEmpty(EditText editText) {
    if (editText.getText() == null || editText.getText().toString().equalsIgnoreCase("")) {
        return true;
    }
    return false;
}
public void onClick(View v) {
    if (v == connectButton) {
```

```
            showDialog();
        } else if (v == this.endSessionBtn) {
            try {
                if (SessionController.isConnected()) {
                    SessionController.getSessionController().disconnect();
                    commandEdit.setText(null);
                }
            } catch (Throwable t) { //catch everything!
                Log.e(TAG, "Disconnect exception " + t.getMessage());
            }
        }
    }
    void showDialog() {
        FragmentTransaction ft = getFragmentManager().beginTransaction();
        Fragment prev = getFragmentManager().findFragmentByTag("dialog");
        ft.addToBackStack(null);
        SshConnectFragmentDialog newFragment = SshConnectFragmentDialog.newInstance();
        newFragment.setListener(new ConnectionStatusListener() {
            @Override
            public void onDisconnected() {
                isConnected = false;
                tvHandler.post(new Runnable() {
                    @Override
                    public void run() {
                        connectedStatus.setText("Connection Status: NOT CONNECTED");
                    }
                });
            }
            @Override
            public void onConnected() {
                if (!isConnected) {
                    isConnected = true;
                    new Thread(new Runnable() {
```

```
                @Override
                public void run() {
                    SessionController.getSessionController().openShell(handler, commandEdit);
                }
            }).start();
        }
        isConnected = true;
        tvHandler.post(new Runnable() {
            @Override
            public void run() {
                connectedStatus.setText("Connection Status: CONNECTED");
            }
        });
    }
});
newFragment.show(ft, "dialog");
  }
}
```

Код файлу SshEditText.java:

```
package com.jgh.androidssh;
import android.content.Context;
import android.text.InputType;
import android.text.Layout;
import android.text.Selection;
import android.util.AttributeSet;
import android.view.KeyEvent;
import android.view.inputmethod.EditorInfo;
import android.view.inputmethod.InputConnection;
import android.view.inputmethod.InputConnectionWrapper;
import android.widget.EditText;
public class SshEditText extends EditText {
  private String lastInput;
```

```
private String nPrompt = "";
public SshEditText(Context context) {
    super(context);
    setup();
}
public SshEditText(Context context, AttributeSet attrs) {
    super(context, attrs);
    setup();
}
public SshEditText(Context context, AttributeSet attrs, int defStyle) {
    super(context, attrs, defStyle);
    setup();
}
public void setup(){
    this.setRawInputType(InputType.TYPE_CLASS_TEXT);
    this.setImeOptions(EditorInfo.IME_ACTION_GO);
    this.setTextSize(12f);
}
@Override
protected void onSelectionChanged(int s, int e) {
    setSelection(this.length());
}
public String getLastInput() {
    synchronized (this) {
        String rez = lastInput;
        lastInput = null;
        return rez;
    }
}
public String peekLastInput() {
    synchronized (this) {
        return new String(lastInput);
    }
```

```
}
public void AddLastInput(String s) {
    synchronized (this) {
        if (lastInput == null) {
            lastInput = "";
        }
        lastInput = s;
    }
}
public int getCurrentCursorLine() {
    int selectionStart = Selection.getSelectionStart(this.getText());
    Layout layout = this.getLayout();
    if (!(selectionStart == -1)) {
        return layout.getLineForOffset(selectionStart);
    }
    return -1;
}
public boolean isNewLine() {
    int i = this.getText().toString().toCharArray().length;
    if(i == 0)
        return true;
    char s = this.getText().toString().toCharArray()[i - 1];
    if (s == '\n' || s == '\r') return true;
    return false;
}
public synchronized void setPrompt(String prompt){
    nPrompt = prompt;
}
public synchronized String getPrompt(){
    return nPrompt;
}
@Override
public InputConnection onCreateInputConnection(EditorInfo outAttrs) {
```

```
        return new SshConnectionWrapper(super.onCreateInputConnection(outAttrs),
                true);
    }
    private class SshConnectionWrapper extends InputConnectionWrapper{
        public SshConnectionWrapper(InputConnection target, boolean mutable) {
            super(target, mutable);
        }
        @Override
        public boolean sendKeyEvent(KeyEvent event) {
            if (event.getKeyCode() == KeyEvent.KEYCODE_DEL) {
                if(isNewLine()) {
                    return false;
                }
                else if(getCurrentCursorLine() < getLineCount() - 1){
                    return false;
                }
            }
            return super.sendKeyEvent(event);
        }
        @Override
        public boolean deleteSurroundingText (int beforeLength, int afterLength){
            if(isNewLine()) {
                return false;
            }
            else if(getCurrentCursorLine() < getLineCount() - 1){
                return false;
            }
            else {
                return super.deleteSurroundingText(beforeLength, afterLength);
            }
        }
    }
}
```

Код файлу ConfigActivity.java:

```
package com.jgh.androidssh;
import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
public class ConfigActivity extends Activity {
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_config);
   }
   @Override
   public boolean onCreateOptionsMenu(Menu menu) {
      getMenuInflater().inflate(R.menu.menu_config, menu);
      return true;
   }
   @Override
   public boolean onOptionsItemSelected(MenuItem item) {
      int id = item.getItemId();
      //noinspection SimplifiableIfStatement
      if (id == R.id.action_settings) {
         return true;
      }
      return super.onOptionsItemSelected(item);
   }
}
```

Код файлу ConnectionStatusListener.java:

```
package com.jgh.androidssh.sshutils;
public interface ConnectionStatusListener {
  public void onDisconnected();
```

```
   public void onConnected();
}
```

Код файлу ExecTaskCallbackHandler.java:

```
package com.jgh.androidssh.sshutils;
public interface ExecTaskCallbackHandler {
   void onFail();
   void onComplete(String completeString);
}
```

Код файлу SessionController.java:

```
package com.jgh.androidssh.sshutils;
import android.os.Handler;
import android.util.Log;
import android.widget.EditText;
import java.lang.InterruptedException;
import com.jcraft.jsch.JSch;
import com.jcraft.jsch.JSchException;
import com.jcraft.jsch.Session;
import java.util.Properties;
public class SessionController {
   private static final String TAG = "SessionController";
   private Session mSession;
   private SessionUserInfo mSessionUserInfo;
   private Thread mThread;
   private ShellController mShellController;
   private ConnectionStatusListener mConnectStatusListener;
   private static SessionController sSessionController;
   private SessionController() {
   }
   public static SessionController getSessionController() {
      if (sSessionController == null) {
```

```
            sSessionController = new SessionController();
        }
        return sSessionController;
    }
    public Session getSession() {
        return mSession;
    }
    private SessionController(SessionUserInfo sessionUserInfo) {
        mSessionUserInfo = sessionUserInfo;
        connect();
    }
    public static boolean exists() {
        return sSessionController != null;
    }
    public static boolean isConnected() {
        Log.v(TAG, "session controller exists... " + exists());
        if (exists()) {
            if (sSessionController.getSession() == null) {
                return false;
            }
            Log.v(TAG, "disconnecting");
            if (sSessionController.getSession().isConnected())
                return true;
        }
        return false;
    }
    public void setUserInfo(SessionUserInfo sessionUserInfo) {
        mSessionUserInfo = sessionUserInfo;
    }
    public SessionUserInfo getSessionUserInfo() {
        return mSessionUserInfo;
    }
    public void connect() {
```

```
        if (mSession == null) {
            mThread = new Thread(new SshRunnable());
            mThread.start();
        } else if (!mSession.isConnected()) {
            mThread = new Thread(new SshRunnable());
            mThread.start();
        }
    }
    public void setConnectionStatusListener(ConnectionStatusListener csl) {
        mConnectStatusListener = csl;
    }
    public void openShell(Handler handler, EditText editText) {
        if (mShellController == null) {
            mShellController = new ShellController();
            try {
                mShellController.openShell(getSession(), handler, editText);
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }
    public boolean executeCommand(Handler handler, EditText editText, ExecTaskCallbackHandler
callback, String command) {
        if (mSession == null || !mSession.isConnected()) {
            return false;
        } else {
            openShell(handler, editText);
            synchronized (mShellController) {
                mShellController.writeToOutput(command);
            }
        }
        return true;
    }
```

```
public void disconnect() {
}
public class SshRunnable implements Runnable {
    public void run() {
        JSch jsch = new JSch();
        mSession = null;
        try {
            mSession = jsch.getSession(mSessionUserInfo.getUser(), mSessionUserInfo.getHost(),
                    mSessionUserInfo.getPort()); // port 22
            mSession.setUserInfo(mSessionUserInfo);
            Properties properties = new Properties();
            properties.setProperty("StrictHostKeyChecking", "no");
            mSession.setConfig(properties);
            mSession.connect();
        } catch (JSchException jex) {
            Log.e(TAG, "JschException: " + jex.getMessage() +
                    ", Fail to get session " + mSessionUserInfo.getUser() +
                    ", " + mSessionUserInfo.getHost());
        } catch (Exception ex) {
            Log.e(TAG, "Exception:" + ex.getMessage());
        }
        Log.d("SessionController", "Session connected? " + mSession.isConnected());
        new Thread(new Runnable() {
            @Override
            public void run() {
                while (true) {
                    try {
                        Thread.sleep(2000);
                        if (mConnectStatusListener != null) {
                            if (mSession.isConnected()) {
                                mConnectStatusListener.onConnected();
                            } else mConnectStatusListener.onDisconnected();
                        }
```

```
                    } catch (InterruptedException e) {
                    }
                }
            }
        }).start();
    }
  }
}
```

Код файлу SessionUserInfo.java:

```
package com.jgh.androidssh.sshutils;
import com.jcraft.jsch.UserInfo;
public class SessionUserInfo implements UserInfo {
   private final String mPassword;
   private final String mUser;
   private final String mHost;
   private final int mPort;
   public SessionUserInfo(String user, String host, String password, int port) {
      mUser = user;
      mHost = host;
      mPassword = password;
      mPort = port;
   }
   public String getPassphrase() {
      return null;
   }
   public String getUser() {
      return mUser;
   }
   public String getHost() {
      return mHost;
   }
   public String getPassword() {
```

```
        return mPassword;
    }
    public int getPort(){ return mPort;}
    public boolean promptPassphrase(String arg0) {
        return true;
    }
    public boolean promptPassword(String arg0) {
        return true;
    }
    public boolean promptYesNo(String arg0) {
        return true;
    }
    public void showMessage(String arg0) {
    }
}
```

Код файлу ShellController.java:

```
package com.jgh.androidssh.sshutils;
import android.os.Handler;
import android.util.Log;
import android.widget.EditText;
import com.jcraft.jsch.Channel;
import com.jcraft.jsch.JSchException;
import com.jcraft.jsch.Session;
import com.jgh.androidssh.SshEditText;
import java.io.BufferedReader;
import java.io.DataOutputStream;
import java.io.IOException;
import java.io.InputStreamReader;
public class ShellController {
    public static final String TAG = "ShellController";
    private BufferedReader mBufferedReader;
    private DataOutputStream mDataOutputStream;
```

```
    private Channel mChannel;
    private String mSshText = null;
    public ShellController() {
    }
    public DataOutputStream getDataOutputStream() {
        return mDataOutputStream;
    }
    public synchronized void disconnect() throws IOException {
        try {
            Log.v(TAG, "close shell channel");
            if (mChannel != null)
                mChannel.disconnect();
            Log.v(TAG, "close streams");
            mDataOutputStream.flush();
            mDataOutputStream.close();
            mBufferedReader.close();
        }catch(Throwable t){
            Log.e(TAG, "Exception: "+t.getMessage());
        }
    }
    public void writeToOutput(String command) {
        if (mDataOutputStream != null) {
            try {
                mDataOutputStream.writeBytes(command + "\r\n");
                mDataOutputStream.flush();
            } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }
    }
    public void openShell(Session session, Handler handler, EditText editText) throws JSchException,
IOException {
```

```
if (session == null) throw new NullPointerException("Session cannot be null!");
if (!session.isConnected()) throw new IllegalStateException("Session must be connected.");
final Handler myHandler = handler;
final EditText myEditText = editText;
mChannel = session.openChannel("shell");
mChannel.connect();
mBufferedReader = new BufferedReader(new InputStreamReader(mChannel.getInputStream()));
mDataOutputStream = new DataOutputStream(mChannel.getOutputStream());
new Thread(new Runnable() {
   @Override
   public void run() {
      try {
         String line;
         while (true) {
            while ((line = mBufferedReader.readLine()) != null) {
               final String result = line;
               if (mSshText == null) mSshText = result;
               myHandler.post(new Runnable() {
                  public void run() {
                     synchronized (myEditText) {
                        ((SshEditText)myEditText).setPrompt(result); //set the prompt to be the current line, so eventually it will be the last line.
                        myEditText.setText(myEditText.getText().toString() + "\r\n" + result + "\r\n"+fetchPrompt(result));
                        Log.d(TAG, "LINE : " + result);
                     }
                  }
               });
            }
         }
      } catch (Exception e) {
         Log.e(TAG, " Exception " + e.getMessage() + "." + e.getCause() + "," + e.getClass().toString());
```

```
            }
        }
    }).start();
}
public static String fetchPrompt(String returnedString){
    return "";
}
public static String removePrompt(String command){
    if(command != null && command.trim().split("\\$").length > 1){
        String[] split = command.trim().split("\\$");
        String s = "";
        for(int i = 1; i< split.length; i++){
            s += split[i];
        }
        return s;
    }
    return command;
}
}
```

Код файлу TaskCallbackHandler.java:

```
package com.jgh.androidssh.sshutils;
import com.jcraft.jsch.ChannelSftp;
import java.util.Vector;
public interface TaskCallbackHandler {
    public void OnBegin();
    public void onFail();
    public void onTaskFinished(Vector<ChannelSftp.LsEntry> lsEntries);
}
```