**СЛАЙД 1**

Тема: Система віддаленої взаємодії із віртуальними машинами

Виконав: студент 4 курсу Борсук Андрій Юрійович

Науковий керівник: Валь Олександр Олександрович

**СЛАЙД 2**

Актуальність

Багато девопсів стикаються з тим, що проекти, над якими вони працюють, потребують прямого втручання, проте люди не можуть постійно перебувати на робочому місці. Багатьом із них стало б пригоді рішення, яке надавало б можливість взаємодіяти зі своїми машинами віддалено.

**СЛАЙД 3**

Мета

Метою розроблюваного застосунку є надання можливості девопсам віддалено, з будь-якої точки світу, виправляти проблеми, що виникають або просто слідкувати за станом їх віртуальних машин.

**СЛАЙД 4**

Завдання проекту

* Розробити можливість авторизації та реєстрації
* Розробити клієнт SSH-підключення
* Розробити клієнт SFTP-підключення
* Розробити клієнт збору метрик
* Локалізувати систему
* Реалізувати можливість додання віртуальних машин
* Розробити модуль маніпуляції акаунтами та машинами
* Розробити захищений API
* Інтегрувати в проект Telegram Bot Api

**СЛАЙД 5**

Об’єкт дослідження

Реалізація системи віддаленої взаємодії із віртуальними машинами.

**СЛАЙД 6**

Основною мовою додатку є мова програмування C#. Головними її перевагами, які стали визначальними для проекту є об’єктно-орієнтованість, високий рівень безпеки, LINQ, широкі можливості бібліотек та фреймворк ASP.NET Core, який дозволяє легко будувати веб застосунки.

Мовою, якою буде написаний клієнт, що збиратиме метрики, був обраний Python.

Основною базою даних проєкту буде Microsoft SQL Server, в ній буде зберігатися інформація про користувачів та віртуальні машини.

Для кешування сесійних даних користувача буде використана база даних Redis.

Entity Framework Core – ORM-фреймворк для взаємодії з базою даних. Він дозволяє легко та зручно під’єднуватися до БД та взаємодіяти з її сутностями.

SSH.NET – бібліотека, яка надає функціонал, для створення SSH- та SFTP-з’єднань з віртуальними машинами.

Telegram.Bot – найпопулярніший .Net клієнт для взаємодії в Telegram Bot Api.

**СЛАЙД 7**

Архітектура

Архітектурою додатка була обрана N-tier архітектура.

Для Api ж була використана архітектура REST Api.

Як видні із діаграми, користувач взаємодіє зі ботом, який отримує повідомлення від користувача, розпізнає в них команди за допомогою ключових слів. Далі бот в виконує відповідний обробник команди. Обробник ж за потреби спілкується із Web Api за допомогою захищених http-запитів. Web Api отримують запити на ендпоїнти контролера та передають їх у відповідні сервіси. Сервіси ж у свою чергу за необхідності взаємодіють із базою даних або ж віртуальною машиною. Як видно з графіку, то в середині застосунку вся взаємодія відбувається за допомогою http- та tcp-запитів.

**СЛАЙД 8**

Реалізація.

Кінцевий застосунок було розроблено у вигляді 2 окремих частин.

**СЛАЙД 9**

Перша частина проекту відповідає за облік даних у базі даних та створення каналу взаємодії між користувачем та машиною.

Всі дані, що зберігаються в базі даних захищені, всі паролі зашифровані. Це означає, що всі критичні дані, які передаються http-запитами захищені. І навіть за умови їх перехоплення зловмисником, він не отримає ніякого доступу до даних.

У користувача є можливість створювати декілька акаунтів та прив’язувати до них різні віртуальні машини. Також користувачу надається інструментарій для адміністрування його акаунтів та віртуальних машин.

**СЛАЙД 10**

Система вміє створювати ізольований та захищений шлюз взаємодії між клієнтом та віртуальною машиною. Цей шлях забезпечує повноту виконання команди, які передає користувач та отримання відповіді від машини, яка вже потім передається користувачу.

Також за допомогою цього шлюзу можна не тільки виконувати команди, а й маніпулювати файловою системою машини.

Система дозволяє користувачам отримувати графіки, які описують картину стану машини.

Цей інструментарій забезпечує користувача повним доступом до машини, що надає змогу девопсам спробувати виправити неполадки, що виникли, в будь-який момент.

**СЛАЙД 11**

Друга ж частина додатку відповідає за графічний інтерфейс та отримання команд від користувача.

Спілкування між користувачем та додатком відбувається за допомогою повідомлень.

Як було зазначено вище система розпізнає команди користувача за ключовими словами, та виконується відповідний обробник команди.

**СЛАЙД 12**

Система на повну використовує такий функціонал Telegram Bot Api, як клавіатури. Це дозволяє забезпечити сценарність взаємодії користувача з ботом та надання йому функціоналу відповідно від його стану.

Вся взаємодія в додатку захищена за допомогою JWT токенів. Якщо користувач пробує виконати команду не авторизованим, то система просить його авторизуватися.

**СЛАЙД 13**

Також система забезпечує зручний обмін файлами між користувачем та віртуальною машиною та зручний вивід відповіді від віртуальної машини.

**СЛАЙД 14**

Як було сказано вище система надає інструментарій для адміністрування акаунтів та віртуальних машин, що надає користувачу зможу змінювати дані віртуальної машини, та навіть видаляти їх.

**СЛАЙД 15**

Якщо в користувача є якісь питання, то він завжди може звернутися за допомогою до мовної моделі OpenAI – ChatGPT.

Також користувач може користуватися системою тією мовою, якою зручно.

**СЛАЙД 16**

Метрики

**СЛАЙД 17**

1. Було сформульовано проблему та визначено необхідний функціонал
2. Використано технології: C#, Python, MS SQL Server, Redis, Entity Framework Core, SSH.NET та Telegram.Bot.
3. Було розроблено серверну частину, яка повністю виконує всі поставленні завдання
4. В проект було інтегровано Telegram Bot Api, що дозволило створити зручний інтерфейс для будь-якого пристрою
5. Систему було протестовано. Всі знайдені баги були виправлені

Реалізований проект, складається з 2 частин. В поєднанні вони створюють зручний кишеньковий додаток, який вирішує поставлену проблему

**СЛАЙД 18**

Q&A

1. **БІЛЬШЕ ПРО АРХІТЕКТУРУ: ТЕКСТ ЗІ СЛАЙДІВ СКАЗАТИ, А ЗАМІСТЬ НЬОГО ПОСТАВИТИ ГРАФІКИ**
2. **ПОДУМАТИ ЯК СТИСНУТИ ДЕКІЛЬКА СЛАЙДІВ З ОДИНИМ РЕЧЕННЯМ В ОДИН БІЛЬШ ЗАГАЛЬНИЙ**
3. **СЛАЙДИ З UI РОБИТИ ПО 2 СКРІНИ НА СЛАЙД І ТЕКСТ, ЯКИЙ ЇХ ОПИСУЄ, ГОВОРИТИ ПРО ОБИДВА СЛАЙДИ**
4. **ДОДАТИ ПРИКЛАДИ ГРАФІКІВ МЕТРИК**
5. **ПЕРЕРОБИТИ ВИСНОВИК, ЇХ МОЖНА РОЗДІЛИТИ НА ДЕКІЛЬКА СЛАЙДІВ І 22 КЕГЛЬ**
6. **ЧАС, ТРІШКИ ЗАДОВГО ГОВОРИВ**
7. **СЛАЙДІВ МАКСИМУМ 18**