VYSOKÉ UČENÍ TECHNICKÉ v BRNĚ

Fakulta informačních technologií

Dokumentácia projektu v predmete ISA.

Varianta: Discord bot

17.11.2020

Filip Bali

Obsah

1 Úvod

Táto dokumentácia popisuje vypracovanie projektu do predmetu „Síťové aplikace a správa sítí“ (skratka ISA), kde bolo cieľom projektu navrhnúť a implementovať bota pre komunikačnú službu Discord.

Hlavnou úlohou programu je počúvanie správ na kanály s názvom „#isa-bot“ a ich následné zopakovanie v zadanom formáte. Správa je zopakovaná iba v prípade, ak názov používateľa, ktorý správu odoslal neobsahuje v jeho názve podreťazec „bot“, nakoľko program takto špecifikuje, že danú správu poslal bot (zadanie vyžaduje, aby neboli zopakované správy, ktoré pochádzajú od iného bota). Zároveň, sú zopakované len správy, ktoré sú napísané na kanál „#isa-bot“ až od momentu spustenia programu, čo znamená, že história správ, ktorú už kanál obsahoval pred spustením programu nebude opakovaná a bude programom ignorovaná.

Komunikácia so službou Discord je naviazaná pomocou SSL spojenia. Pri komunikácií je využívaná Discord API, konkrétne REST požiadavkami. **Pre správnu funkčnosť programu musí mať bot v rámci servera nastavené minimálne práva:**

1) Musí môcť vidieť kanály na danom servery
2) Musí môcť vedieť čítať históriu správ na danom kanály
3) Správy s vloženými linkami (tzv. „embed links“)

2 Internetový bot (Bot)

Je počítačový program, ktorý opakovane vykonáva rutinnú činnosť na internete.

2.1 Discord

Je komunikačná platforma určená pre komunity podporujúca rôzne formy komunikácie(textovú, hlasovú).

2.2 Representational state transfer (REST)

Je softwarová architektúra rozhrania pre distribuované systémy komunikujúce pomocou siete. Viac informácií je dostupných v dokumente RFC 6690.

3 Vytvorenie a správa bota

- Vytvorenie a pridanie bota je možné podľa postupu ako na webovej adrese: https://discordpy.readthedocs.io/en/latest/discord.html

- Možnosť správy/administrácie bota je možné využiť a iný Discord aplikácii je možné na webovej adrese: https://discord.com/developers/applications

4 Návrh a implementácia

K implementácií programu bol použitý programovací jazyk C++. Nakoľko cieľom návrhu bol program ktorý využije objekto-orientovaného paradigmu a zároveň možností vyššej abstrakcie a využitia typov, ktorých správa dynamickej alokácie je automatická aby bolo znížené riziko chýb spojené so zlou alokáciou pamäte programátorom.

Funkcionalita programu bola zapuzdrená do niekoľko tried, ktoré obsahujú príslušné metódy. Každý názov triedy začína s veľkým začiatočným písmenom „C“ a pokračuje s názvom triedy samotnej.

- **Triedy**
  - **CParams**
    - **Popis triedy**

      Obsluhuje argumenty programu. Zisti či je program spustený s vhodnými parametrami, vykoná ich validáciu a uloží potrebné hodnoty do premenných inštancie triedy.
    - **Metódy triedy**
      - Privátne
        - void getParams(int argc, char *argv[])
        - void printHelpMsg()
        - void checkArgument(int *ArgNum ,int argc, char *argv[])
      - Verejné
        - bool getParamBool(const string &param)
        - string getAccessToken()
  - **CJSON**
    - **Popis triedy**

      Obsahuje metódy ktoré uľahčujú prácu s objektovou notáciou JavaScript-u (ďalej len JSON). Overuje prítomnosť potrebných identifikátorov v JSON a uchová v premenných inštancie triedy ich hodnotu.
    - **Metódy triedy**
      - Verejné
        - void split_json_messages_into_vector(std::string json)
        - std::string JSON_filter(std::string regex, std::string string)
        - std::vector<std::vector<std::string>> load_values_from_json()
  - **CConnection**
    - **Popis triedy**

      Obsahuje metódy, ktorých sa vykonáva hlavná časť programu. Sprostredkováva komunikáciu so servermi Discord.
    - **Metódy triedy**
      - Privátne
        - void MainProcess()
        - void Connect()
        - std::string send_GET_request(std::string api_call, std::string snowflake = "")
        - std::string send_POST_request(std::string api_call, std::string content)
        - std::string JSON_filter(std::string regex, std::string string
      - Verejné
        - explicit CConnection(CParams &Params)

5 Parametre programu a spustenie

Program podporuje niekoľko parametrov a ich kombinácií. Parametre možno navzájom kombinovať, avšak pre správnu funkcionalitu môže byť každý parameter uvedený iba raz.

5.1 Slovný popis parametrov

1. „**-h**“ alebo „**--help**“
   - **<u>Voliteľný parameter:</u>**
   - Vypíše na štandardný výstup pomocnú správu o funcionalite programu a návod ako program spustiť a použiť.
   - Ak je tento parameter zadaný, program sa po vypísaní správy **ukončí**.
2. „**-v**“ alebo „**--verbose**“
   - **<u>Voliteľný parameter.</u>**
   - Ak je tento parameter zadaný, program bude vypisovať správy na štandardný výstup vo formáte "<channel> - <username>: <message>" pri každej reakcií na správu.
3. „**-t** <bot_access_token>“
   - **<u>Povinný parameter.</u>**
   - Parameter slúži pre zadanie tokenu ktorý špecifikuje konkrétneho bota.
   - Znaky „<“ a „>“ sa nepíšu. Slúžia na predstavu kde je očakávané umiestnenie tokenu pri zadaní parametra.

5.2 Spustenie

Parametre ktoré sú v hranatých zátvorkách sú voliteľné. Pri príkladoch pre spustenie sa predpokladá, že všetky súbory z adresára xbalif00.tar sú rozbalené na rovnakej ceste v ktorej sa práve nachádza terminál. Zároveň sa predpokladá, že nie je problém z absenciou práv prihláseného užívateľa k súborom a teda vie čítať/vytvárať/meniť súbory v danom adresári. Zároveň sa predpokladá, že systém je v rovnakej konfigurácií a má nainštalované rovnaký software ako referenčné stroje Merlin(merlin.fit.vutbr.cz) a Eva(eva.fit.vutbr.cz).

Pre preklad programu (Úspešný preklad sa vyžaduje pri každom ďalšom spustení):
$ make

Vyvolanie pomocnej správy:
$ ./isabot -h **alebo** $ ./isabot –help

Zapnutie bota bez výpisov na štandardný výstup:
$ ./isabot -t <access_bot_token>
Reálný príklad:
$ ./isabot -t NzYxNTc5NDkyODkxNTU3OTA5.X3cqLA.kdYlkNnS8t9awuGT-eGj4-ztw8h

Zapnutie bota s výpismi na štandardný výstup:
$ ./isabot -t <access_bot_token> -v **alebo** $ ./isabot -t <access_bot_token> --verbose
Reálný príklad:
$ ./isabot -t NzYxNTc5NDkyODkxNTU3OTA5.X3cqLA.kdYlkNnS8t9awuGT-eGj4-ztw8h --verbose

6 Popis dôležitých metód programu:

a. Trieda CConnection

- **std::string JSON_filter(std::string regex, std::string string)**
  - Metóda JSON_filter má dva argumenty, reťazec regex a reťazec string a poskytuje programu vyextrahovanie podreťazca ktorý sa nachádza v reťazci, ktorý predstavuje výstupnú hodnotu metódy. Parameter regex predstavuje regulárny výraz podľa ktorého prebehne vyextrahovanie a parameter string predstavuje reťazec z ktorého bude extrakcia vykonaná.

- **std::string send_GET_request(std::string api_call, std::string snowflake = "")**
  - Má povinný argument **api_call**, ktorý predstavuje reťazec ktorý je štandardizovaný v rámci Discord API. Špecifikovaním **api_call** je možné poslať GET požiadavku s konkrétnou požiadavkou voči Discord API. Ďalší parameter predstavuje reťazec **snowflake**, ktorý je avšak nepovinný (ak nie je zadaný potom tento parameter je nahradený za prázdny reťazec) a predstavuje unikátnu hodnotu snowflake, ktorá je špecifikovaná službou Discord. Snowflake predstavuje niekoľkomiestne číslo ktoré predstavuje časové razítko, napríklad správy a služba Discord túto hodnotu využíva napríklad ako ID pre konkrétnu správu na danom servery. **Hlavná úloha metódy je odoslanie GET požiadavky na Discord server a následné získanie odpovede v podobe JSON formátu.** JSON formát je následne zo správy vyextrahovaný a vrátený ako reťazec funkcie. Metóda zároveň kontroluje možné chyby komunikácie (HTTP ERROR), ktoré vyhodnocuje a vtedy je ako výstupný reťazec funkcie **FALSE_ERROR_JSON**. Metóda je zároveň rozšírená o spracovanie http chyby a reakciu na ňu. Reakcie sa líšia vyhodnotením o akú http ide (Potrebné spomalenie komunikácie medzi programom a Discord serverom, vnútorná chyba serveru Discord a iné).

- **std::string send_POST_request(std::string api_call, std::string content)**
  - Ma povinný argument **api_call**, ktorý predstavuje reťazec ktorý je štandardizovaný v rámci Discord API. Špecifikovaním **api_call** je možné poslať POST požiadavku s konkrétnou požiadavkou voči Discord API. Ďalší parameter predstavuje reťazec **content** pri ktorom sa očakáva že bude mať JSON formát, nakoľko služba Discord podporuje zasielanie nových správ prostredníctvom ich API výhradne vo formáte JSON. Metodá zároveň spracuje odpoveď od služby Discord ktorá je vo formáte JSON. Následne ju uloží do reťazca a predá ako výstupný parameter z metódy.

b. Trieda CJSON

- **void split_json_messages_into_vector(std::string json)**
  - Z parametru reťazca **json** ktory je očakávaný vo formáte JSON načíta JSON správu zo služby Discord, ktorá obsahuje všetky správy v jednom reťazci. Tento reťazec následne rozdelí pomocou oddeľovača „}, {“, ktorý zabezpečí, že reťazec rozdelí do poľa po jednotlivých správach, z ktorých je následne tento oddeľovač odstránený.

- **std::vector<std::vector<std::string>> load_values_from_json()**
  - S využitím metódy **„std::string JSON_filter(...)“** dokáže získať potrebné dáta z jednej konkrétnej správy ktorá je vo formáte JSON. Predovšetkým ID správy, obsah správy, prezývka odosielateľa a TTS. Tieto údaje sú uložené do dvojrozmerného poľa ktorá je výstupom metódy, kde stĺpcový index predstavuje jednu konkrétnu správu a riadkový index dáta tejto správy, v poradí ako sú napísané vyššie.

7 Priebeh programu pri spustení

Program pri svojom spustení najskôr vytvorí inštanciu triedy **CParams** a vloží do konštruktoru triedy počet argumentov a pole argumentov. Pri vytváraní objektu z triedy **CParams** vyvolá konštruktor aj metódy pre spracovanie argumentov programu a výsledok je uložený do premenných inštancie. Vytvorená inštancia je uložená do objektovej premennej s názvom **Params**.

Následne je vytvorená inštancia triedy **CConnection** a objekt inštancie je uložený do objektovej premennej **Connection**. Pri vytváraní inštancie je parametrom konštruktoru vyžadovaná inštancia triedy **CParams**. Zároveň konštruktor volá metódu triedy „**void Connect()“**, ktorá predstavuje hlavné jadro programu. Táto metóda následne volá ďalšiu metódu triedy a to „**void init_ssl_resources()“**, ktorá inicializuje všetky potrebné premenné a vytvára nové SSL spojenie so serverom služby Discord. V prípade akéhokoľvek zlyhania program informuje o nezdare s chybovou hláškou o probléme na štandardný výstup. Ak prebehla inicializácia a nadviazanie SSL spojenia úspešné, program pokračuje niekoľkými volaniami metód triedy „**std::string send_GET_request(...)“** a **„std::string JSON_filter(...)“**. S ich volaniami program získa informácie o botovi, na akých serveroch sa nachádza, aké majú servery kanály a tieto dáta spracuje. Program podľa zadania predpokladá, že je umiestnený iba jednom servery a že daný server obsahuje kanál **#isa-bot**, ktorých ID spracuje a uloží. Následne program na kanály **#isa-bot** v príslušnom servery počúva pre nové správy.

Ak program je práve spustený a pýta sa na správy na kanály prvý krát, potom zisti ktorá správa je posledná, uloží si ID tejto správy a následne žiada o ďalšie správy po tejto správe na ktoré už bude reagovať.

Ak program už pozná ID poslednej správy na kanály potom sa program nachádza v nekonečnom cykle v ktorom sústavne kontroluje či sa na kanály **#isa-bot** vyskytli nové správy s vyšším ID ako má správa ktorú program považuje za poslednú(táto správa nemôže byť od bota, tj. ID správy ktoré je prehlásené za posledné je vždy od reálneho užívateľa). Po tom čo program získa všetky správy spojené v jednom JSON formáte, vytvorí inštanciu triedy **CJSON** a vykoná ich spracovanie v metóde „**void split_json_messages_into_vector(...)“** z inštancie triedy **CJSON** a to tak že správa v JSON formáte je rozdelená do poľa, ktoré je výstupom metódy a ktorého každý index predstavuje práve jednu správu kde správy sú uložené od najstaršej po najnovšiu. Následne je zavolaná metóda „**std::vector<std::vector<std::string>> load_values_from_json()“** z inštancie triedy **CJSON,** ktorá využíva predchádzajúce pole kde sú správy uložené po indexoch a výstupom je dvojrozmerné pole, ktoré predpokladá, že každý stĺpcoví index je práve jedna správa a každý stĺpec má 4 riadky ktoré podľa indexov predstavujú:

0. ID správy
1. Obsah správy
2. Prezývka odosielateľa
3. TTS správy

Následne takto spracované správy sú poslané späť na Discord ako správy pomocou metódy „**std::string send_POST_request(...)“** ak prezývka ktorú správu odoslal nemá v sebe podreťazec „bot“, pretože tieto správy sú programom označené ako správy od iných botov a na tieto správy nie je podľa zadania odpovedane.

Následne kolobeh počúvania pre nové správy, ich spracovanie a zopakovanie na server je vykonávaný pokým nie je program ukončený zaslaním signálu SIGINT.

8 Literatúra

- https://tools.ietf.org/html/rfc7235
- https://tools.ietf.org/html/rfc7231
- https://tools.ietf.org/html/rfc8259
- https://www.ietf.org/rfc/rfc6690.txt
- https://discord.com/developers/docs/intro
- https://stackoverflow.com/questions/14265581/parse-split-a-string-in-c-using-string-delimiter-standard-c
- https://stackoverflow.com/questions/11627440/regex-c-extract-substring
- https://en.cppreference.com/w/
- https://en.cppreference.com/w/cpp/regex
- Prednášky predmetu ISA (VUT FIT)
- https://www.openssl.org/docs/man1.1.1/man1/