

# VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

## Projekt ISA - Programování síťové služby Discord bot

# Obsah

<b>1</b>	<b>Uvedenie do problematiky</b>	<b>2</b>
1.1	SSL . . . . .	2
1.2	HTTPS . . . . .	2
1.3	OAuth2 . . . . .	2
1.4	Bot token . . . . .	2
1.5	Komunikácia s Discord API . . . . .	2
<b>2</b>	<b>Návrh a implementácia aplikácie</b>	<b>3</b>
2.1	Štruktúra programu . . . . .	3
2.2	Priebeh programu . . . . .	3
2.2.1	Argumenty . . . . .	3
2.2.2	Nadviazanie spojenia s Discord serverom pre komunikáciu s API . . . . .	3
2.2.3	Komunikácia s API a spracovanie requestov . . . . .	3
<b>3</b>	<b>Základné informácie o programe</b>	<b>4</b>
3.1	Použité knižnice . . . . .	4
3.2	Súbory . . . . .	4
<b>4</b>	<b>Návod na použitie</b>	<b>4</b>
<b>5</b>	<b>Záver</b>	<b>4</b>
<b>6</b>	<b>Literatura</b>	<b>5</b>

# 1 Uvedenie do problematiky

Úlohou je vytvoriť program isabot, ktorý bude pôsobiť ako bot na komunikačnej službe Discord. Bot sa pripojí na kanál #isabot a bude zachytávať a opakovať všetky správy odoslané do kanálu užívateľmi, ktorí nemajú v mene podreťazec "bot" vo formáte: "echo: <username> - <message>".

Na komunikáciu bota so serverom sa používa *HTTPS REST API*<sup>1</sup> Discordu, na ktoré sa pripája pomocou HTTPS protokolu s využitím SSL protokolu.

## 1.1 SSL

**SSL (Secure Socket Layer)** je protokol, resp. vrstva vložená medzi vrstvu transportnú (TCP/IP) a aplikačnú (HTTP). Zabezpečuje *šifrovanú* dátovú komunikáciu medzi komunikujúcimi stranami. Najčastejšie sa využíva pri komunikácii s webovými servermi pomocou *HTTPS*. V tomto prípade je použitá na komunikáciu s Discord API.

## 1.2 HTTPS

**HTTPS (Hypertext Transfer Protocol Secure)** je protokol umožňujúci zabezpečenú komunikáciu v počítačovej sieti. HTTPS využíva protokol HTTP spolu s protokolom SSL alebo TLS. Zaisťuje autentizáciu, dôvernosc a integritu prenášaných dát.

## 1.3 OAuth2

Pre overenie Discord bota s Discord API sa používa služba **OAuth2**, pomocou OAuth2 API, ktoré prijíma OAuth2 bearer token. V tomto tokene sú nastavené oprávnenia bota, t.j. akcie, ktoré môže bot vykonávať na serveri.

## 1.4 Bot token

Pre autorizáciu Discord bota sa používa **bot token**, ktorý je špecifický pre každého jedného bota.

## 1.5 Komunikácia s Discord API

S Discord API sa komunikuje cez HTTPS protokol, ktorý prijíma na URL: *https://discord.com/api/v6* GET a POST requesty. V tomto projekte sú použité:

### Pre získanie dostupných serverov pre bota:

GET /api/v6/users/@me/guilds HTTP/1.1

Host: discord.com

Authorization: Bot <Bot token>

### Pre získanie ID channelu:

GET /api/v6/guilds/<GuildId>/channels HTTP/1.1

Host: discord.com

Authorization: Bot <Bot token>

### Pre odoslanie opakovanej správy:

POST /api/v6/channels/<ChannelId>/messages HTTP/1.1

Host: discord.com

Authorization: Bot <Bot token>

Content-Type: application/json

Content-Length: <Length of content>

### Pre získanie správ v Discord kanáli:

GET /api/v6/channels/<ChannelId>/messages HTTP/1.1

Host: discord.com

Authorization: Bot <Bot token>

```
{"content": "echo: <Username> - <Message>"}
```

---

<sup>1</sup>Discord reference documantation

## 2 Návrh a implementácia aplikácie

### 2.1 Štruktúra programu

**Trieda Message** - popisuje jednu správu v Discord chate.

**string username** - meno používateľa, ktorý odoslal správu.

**string content** - obsah odoslanej správy.

**string timestamp** - čas odoslania správy.

**string msgID** - ID správy.

**bool areIdentical(Message b)** - funkcia, ktorá porovnáva dve správy. Ak majú identické parametre, vráti *true*, v opačnom prípade vráti *false*.

#### Funckie:

**int main(int argc, char \*argv[])** - spracuje argumenty, nadviaže spojenie s Discord serverom.

**void printhelp()** - vypíše nápovedu na štandardný výstup.

**int SendPacket(const char \*req)** - odošle request na Discord API, ktorý je obsiahnutý v premennej *req*

**int RecvPacket(int sw)** - spracuje odpoveď GET requestu. Parameter sw (switch) určuje, ktorý GET request sa spracováva.

**vector<string> retrieveMessages(string s)** - rozdelí históriu chatu z odpovede GET requestu na jednotlivé správy.

**void parseMessages(vector<string> vs)** - spracuje jednotlivé správy, nové správy odošle pomocou POST requestu do Discord chatu.

### 2.2 Priebeh programu

#### 2.2.1 Argumenty

Na začiatku programu sa spracujú argumenty pomocou knižnice `< getopt.h >`.

#### 2.2.2 Nadviazanie spojenia s Discord serverom pre komunikáciu s API

Vytvorí sa socket, ktorý sa pripojí na IP adresu Discord serveru.

Ak je vytvorenie socketu úspešné, inicializuje sa SSL spojenie.

#### 2.2.3 Komunikácia s API a spracovanie requestov

Ak bolo nadviazanie SSL spojenia úspešné, pošlú sa pomocou funkcie *int SendPacket(const char req)* GET requesty pre získanie dostupných serverov pre bota (z odpovede sa vyfiltruje pomocou regexu *GuildID* vo funkcii *int RecvPacket(int sw)* (kde *sw == 1*) ) a pre získanie ID channelu #isabot (z odpovede sa vyfiltruje pomocou regexu *ChannelID* vo funkcii *int RecvPacket(int sw)* (kde *sw == 2*) ).

Program vstúpi do nekonečného while cyklu, ktorý posiela GET request pre získanie správ v channeli #isabot. Odpoveď, ktorá mu príde obsahuje históriu správ chatu, ktorú funkcia *vector<string> retrieveMessages(string s)* rozdelí na jednotlivé správy, ktoré uloží do vektora *vector<string> str\_vect*. Následne sa z nich vo funkcii *void parseMessages(vector<string> vs)* pomocou regexu vyfiltruje *Username*, *Content*, *Timestamp* a *MessageID*. V prvom cykle sa vytvoria zo všetkých správ *message* objekty a vložia sa do vektora *msg\_vect*, ktorý obsahuje správy, ktoré boli poslané do chatu pred spustením bota a všetky správy, ktoré boli spracované a bot ich nemá opakovať späť do chatu.

V ďalších cykloch bot z GET requestu pre získanie správ v channeli #isabot porovnáva *Username*, *Content*, *Timestamp* a *MessageID* správ, so správami ktoré má uložené vo vektore *msg\_vect* a tie, ktoré sa tam nevyskytujú pomocou POST requestu opakuje do chatu vo formáte **"echo: <username> - <message>"**.

Danú správu si taktiež pridá do vektora *msg\_vect*. Ak bol zadán pri spustení programu argument *[-v|-verbose]* vypíše opakovanú správu na štandardný výstup vo formáte **"<channel> - <username>: <message>"**.

## 3 Základné informácie o programe

### 3.1 Použité knižnice

<arpa/inet.h> <iostream> <ctype.h> <stdio.h> <stdlib.h> <unistd.h> <string> <getopt.h>  
<string.h> <sys/socket.h> <openssl/ssl.h> <openssl/err.h> <regex> <vector> <netinet/in.h>

### 3.2 Súbor

Celý program je obsiahnutý v súbore `isabot.cpp`.

Preklad je zabezpečený *Makefile*-om.

Pomocou príkazu *make* sa súbor `isabot.cpp` preloží, a vytvorí sa binárny súbor je nazvaný *isabot*.

## 4 Návod na použitie

Pred spustením je potrebné program preložiť príkazom *make*.

Po preložení sa program spustí príkazom `./isabot[-h|-help] [-v|-verbose] -t <bot_access_token>`

Spustenie programu bez argumentov zobrazí nápovedu.

Argumenty:

**[-h|-help]** - zobrazí nápovedu. (Nepovinný argument)

**[-v|-verbose]** - bude vypisovať botom opakovanú správu na štandardný výstup vo formáte

<channel> - <username>: <message>. (Nepovinný argument)

**-t <bot\_access\_token>** - za <bot\_access\_token> treba vložiť token používaného bota. (Povinný argument)

## 5 Záver

Celkovo projekt hodnotím ako pozitívnu a zaujímavú skúsenosť. Dal mi praktické znalosti v komunikácii cez HTTPS protokol pomocou SSL knižnice, posielaní a spracovanie odpovedí GET a POST requestov a komunikácii s Discord API. Oprášil moje znalosti regexu a pri písaní dokumentácie znalosti  $\LaTeX$ u.

## 6 Literatura

### Reference

- [1] Discord API dokumentácia. [online], [vid. 2020-11-03].  
URL <https://discord.com/developers/docs/reference>
- [2] HTTPS Wikipedia. [online], [vid. 2020-11-03].  
URL <https://cs.wikipedia.org/wiki/HTTPS>
- [3] OpenSSL dokumentácia. [online], [vid. 2020-11-03].  
URL <https://www.openssl.org/docs>
- [4] SSL Wikipedia. [online], [vid. 2020-11-03].  
URL [https://cs.wikipedia.org/wiki/Secure\\_Sockets\\_Layer](https://cs.wikipedia.org/wiki/Secure_Sockets_Layer)