

Multifunkční bot pro aplikaci Discord

A Multifunction Bot for Discord

Michal Hřešil

Bakalářská práce
2022



Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav informatiky a umělé inteligence

Akademický rok: 2021/2022

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení: Michal Hřešil
Osobní číslo: A19036
Studijní program: B3902 Inženýrská informatika
Studijní obor: Softwarové inženýrství
Forma studia: Prezenční
Téma práce: Multifunkční bot pro aplikaci Discord
Téma práce anglicky: A Multifunction Bot for Discord

Zásady pro vypracování

1. Nastudujte a popište problematiku spojenou s tématem práce.
2. Pochopte a zvolte technologie spojené s implementací botů v kontextu s platformou Discord.
3. Vyberte vhodné vlastnosti multifunkčního bota pro aplikaci Discord a proveďte návrh.
4. Vámi navrženého bota implementujte.
5. Implementaci vhodně otestujte a vyhodnoťte.
6. Všechny výsledky vhodně reprezentujte.

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

1. NGUYEN, Don. Node.js Okamžitě. Přeložil Ondřej BAŠE. Brno: Computer Press, 2016. ISBN 9788025148204.
2. ŠKULTÉTY, Rastislav. JavaScript: programujeme internetové aplikace. Praha: Computer Press, 2001. Pro každého uživatele. ISBN 8072264575.
3. SUEHRING, Steve. JavaScript: krok za krokem. Brno: Computer Press, 2008. Krok za krokem (Computer Press). ISBN 9788025122419.
4. ČERNÁ, Alena. Kyberšikana: průvodce novým fenoménem. Praha: Grada, 2013, 150 s. Psyché. ISBN 9788021063747.

Vedoucí bakalářské práce:

Ing. Petr Žáček, Ph.D.

Ústav informatiky a umělé inteligence

Datum zadání bakalářské práce: **3. prosince 2021**

Termín odevzdání bakalářské práce: **23. května 2022**



doc. Mgr. Milan Adámek, Ph.D. v.r.
děkan

prof. Mgr. Roman Jašek, Ph.D., DBA v.r.
ředitel ústavu

Ve Zlíně dne 24. ledna 2022

Prohlašuji, že

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – bakalářskou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne 20.05.2022

Michal Hřešil v.r.
podpis studenta

ABSTRAKT

Cílem práce je vytvořit veřejný rejstřík problémových uživatelů platformy Discord a bota pro jeho automatickou správu. Bot se postará o jejich detekci na serverech, aby došlo ke snížení počtu uživatelů platformy působících škodu, rozšiřujících nenávist a porušujících pravidla serverů, na které se připojí. Aplikace umožňuje správcům serverů a jimi pověřeným uživatelům dostávat upozornění o připojení problémového jedince na server a umožní jim zobrazit si jejich záznam v rejstříku obsahující seznam serverů kde došlo k zablokování přístupu danému uživateli a důvod, proč se tak stalo. Bot se také snaží detekovat autonomní uživatele rozesílající nevyžádané zprávy a závadné odkazy napříč servery nebo uživatele šířící nenávistné ideologie. Díky tomu, že po jejich zablokování skrze bota dojde k evidenci jejich veřejně dostupných údajů je bot schopen tyto uživatele detekovat ihned po připojení na jiný server. Bot má tudíž možnost upozornit na zmíněné problémové uživatele a případně je zablokovat dříve, než se pokusí napáchat škodu.

Klíčová slova: Discord, kyberšikana, JavaScript, NodeJs, Discord.js, MongoDB, Bot

ABSTRACT

This work aims to create a public register of problem users of the Discord platform and a bot for its automatic administration. The bot will take care of their detection on the servers to reduce the number of platform users who cause harm, spread hatred and violate the rules of the servers to which they connect. The application allows server administrators and their authorized users to receive notifications about a problem individual's connection to the server and allows them to view their registry entry containing a list of servers where access to the user was blocked and the reason why. The bot also tries to detect autonomous users sending unsolicited messages and malicious links across servers or users spreading hate ideologies. Because after their blocking through the bot, their publicly available data is registered, the bot can detect these users immediately after connecting to another server. The bot, therefore, has the opportunity to alert these problematic users and possibly block them before attempting to do damage.

Keywords: Discord, cyberbullying, JavaScript, NodeJs, Discord.js, MongoDB, Bot

Chtěl bych moc poděkovat svému vedoucímu práce, panu Ing. Petru Žáčkovi Ph.D. za to, že mi umožnil zpracovat tohle téma, za pedagogickou a odbornou pomoc při vypracovávání bakalářské práce.

Prohlašuji, že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

OBSAH

ÚVOD.....	9
I TEORETICKÁ ČÁST.....	10
1 KOMUNIKAČNÍ PLATFORMA DISCORD	11
1.1 SERVERY	11
1.2 UŽIVATELÉ, ROLE A OPRÁVNĚNÍ	12
1.3 MODERACE NA PLATFORMĚ DISCORD	12
2 PROGRAMEM ŘÍZENÝ ÚČET – DISCORD BOT	13
2.1 OVLÁDÁNÍ A KOMUNIKACE S BOTE M	14
2.1.1 Příkazy typu „legacy“	14
2.1.2 Příkazy typu „slash“	15
2.2 TECHNOLOGIE PRO VÝVOJ A IMPLEMENTACI BOTŮ	17
2.2.1 Discord API.....	17
2.2.1.1 WebSocket API.....	18
2.2.1.2 REST API	18
2.2.2 Knihovny třetích stran pro práci s Discord API.....	19
2.2.3 Knihovna Discord.js.....	20
2.2.3.1 JavaScript.....	20
2.2.3.2 NodeJs.....	21
3 TERMINOLOGIE S PROBLEMATIKOU KYBERŠIKANY	23
3.1 KYBERŠIKANA.....	23
3.2 PLATFORMA DISCORD A KYBERŠIKANA	24
II PRAKTICKÁ ČÁST	27
4 NÁVRH BOTA PRO USNADNĚNÍ MODERACE	28
4.1 AUTOMATICKÉ VLASTNOSTI BOTA	28
4.1.1 Spouštěč příkazů	29
4.1.2 Pozdrav nově příchozích členů	30
4.1.3 Automatická kontrola nově příchozích členů	30
4.2 MANUÁLNÍ VLASTNOSTI BOTA	31
4.2.1 Odebrání uživatele ze serveru.	31
4.2.2 Zablokování uživatele na serveru.....	32
4.2.3 Vyhledání uživatele ve sdílené databázi	33
4.2.4 Příkaz pro zobrazení nápovědy	34
4.2.5 Příkaz pro prvotní nastavení.....	34
4.2.6 Příkaz pro nahlášení uživatele.....	35
5 IMPLEMENTACE BOTA	37

5.1	SOUBOROVÁ STRUKTURA	37
5.2	Hlavní spouštěcí soubor	39
5.3	Vytvoření aplikace a přidání účtu typu BOT	40
5.4	Spouštěč příkazů	42
5.5	Struktura příkazu z hlediska kódu	44
5.6	Automatická kontrola uživatelů a pozdravy	45
5.7	Příkaz pro odebrání uživatele ze serveru	49
5.8	Příkaz pro zablokování uživatele	51
5.9	Příkaz pro ověření uživatele	52
5.10	Příkaz pro nahlášení uživatelů	54
5.11	Příkaz pro prvotní nastavení	57
5.12	Další příkazy	59
6	TESTOVÁNÍ BOTA	60
	ZÁVĚR	64
	SEZNAM POUŽITÉ LITERATURY	65
	SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK	68
	SEZNAM OBRÁZKŮ	69
	SEZNAM TABULEK	71
	SEZNAM PŘÍLOH	72

ÚVOD

Tato bakalářská práce se zabývá vytvářením bota pro komunikační platformu Discord, který bude sloužit jako nástroj pro komunitu ke zlepšení moderace jednotlivých serverů a evidování uživatelů napříč servery, což bude mít za následek zviditelnění uživatelů opakovaně porušujících pravidla a následnou detekci potenciálně nechtěného nebo nebezpečného uživatele.

Teoretická část je zaměřena na požadavky nutné k pochopení pro správně vypracování práce, tedy jakým způsobem funguje platforma Discord po uživatelské a technické stránce, jakým způsobem fungují bota v kontextu platformy Discord, jaké jsou technologie pro jejich vývoj a co všechno bot dovede. Také je třeba seznámit se s pojmy jako moderace nebo kyberšikana, tedy o co přesně se jedná a kde se s nimi lze setkat. Tím se zabývá teoretická část této bakalářské práce.

V praktické části se práce zabývá návrhem pro efektivního bota a následnou implementací, která je řádně otestována.

Výsledkem práce bude bot, který je funkční a připraven k nasazení na jakýkoliv server.

I. TEORETICKÁ ČÁST

1 KOMUNIKAČNÍ PLATFORMA DISCORD

Komunikační platforma Discord je aplikace, kterou používá po celém světě desítky milionů lidí ke komunikaci hlasové, textové nebo ke sdílení své obrazovky pro více uživatelů najednou ve videokonferenci, a to jak v soukromě, tak i pracovně. Aplikace umožňuje vytvářet uživatelům vlastní servery a v nich vytvářet specializované místnosti k textové či hlasové komunikaci, kde v právě zmíněných hlasových kanálech umožňuje přímý přenos obrazovky pro ostatní posluchače. Platforma Discord také umožňuje vytvářet role které slouží k hromadnému oddělení určitých uživatelů od ostatních, a to jak po vizuální stránce, například barevné jméno nebo zobrazení odděleně v seznamu uživatelů do vlastní kategorie s názvem role, tak po stránce oprávnění, kdy každá role může mít nastaveny určité funkce, například blokování uživatelů, mazání cizích zpráv, přístupy do místností určeny jen danou roli apod.

Servery se dělí na dvě hlavní kategorie, veřejné a soukromé. Veřejný server je takový, pro který se pozvánka nachází volně na internetu, není nijak omezena a může se tak připojit kdokoli. Takovéto servery pak mohou tak sloužit pro jakoukoliv komunitu uživatelů od herní, přes filmovou, knižní až po studium či práci. Soukromé servery slouží pro uzavřenější komunitu, jako jsou například pracovní kolegové, spolužáci či přátelé. Takový server pak nemá pozvánku platnou neomezeně. Ta se většinou vytváří individuálně pro každého uživatele zvlášť a po připojení uživatele nebo uplynutí času se stává neplatnou. Platforma také podporuje funkce, jaké běžná sociální síť jako například přímé soukromé zprávy mezi uživateli, přímé soukromé hovory, přidávání uživatelů do seznamu přátel nebo sdílení souborů. Aplikace je dostupná pro všechny platformy jak v provedení webové aplikace, tak i desktopové nebo mobilní aplikace. [1]

1.1 Servery

Server na platformě Discord je místo vytvořené specificky pro určitou komunitu. Každý uživatel má možnost vytvořit si vlastní, ať už soukromý nebo veřejný server. Soukromý server vyžaduje speciální pozvánkou formou odkazu, která platí jen po omezenou dobu a poté se deaktivuje. K veřejnému serveru se používá pozvánka trvalá, která nikdy neztratí platnost a je možno ji uveřejnit a internet. Organizace serveru je prováděna pomocí místností, kdy každá místnost může být buď textová, nebo hlasová. Místnosti je také možno třídit do kategorií. Textová místnost je chat sloužící typicky k určitému tématu, které pak nese i název dané místnosti, kde mohou uživatelé psát, posílat soubory či emotikony. Hlasové místnosti slouží pro propojení uživatelů do hovoru. Odpojením všech uživatelů místnost nezaniká. [1]

1.2 Uživatelé, role a oprávnění

Uživatel je kdokoliv, kdo si vytvoří uživatelský účet na platformě Discord, a které má následný přístup k jejím funkcím a zavazuje se tím k dodržování obecných pravidel platformy. Za jednotlivé servery však neodpovídá platforma, ale uživatelé, kterým daný server patří.

Po připojení na server je každému uživateli přidělena sada oprávnění. Platforma Discord řídí uživatelská oprávnění pomocí rolí. Při vytvoření serveru je automaticky vytvořena role s názvem „@everyone“ obsahující základní oprávnění běžných uživatelů jako například posílat zprávy, zobrazit si uživatele na serveru, upravovat nebo mazat své vlastní zprávy.

Dále je při vytváření serveru automaticky vytvořena také interní role, které není viditelná ani upravovatelná a slouží pro udělení absolutního oprávnění pro vlastníka serveru. Další role jsou již vytvářena vlastníkem serveru k oddělení uživatelů na základě vizuálního nebo po stránce oprávnění. Například vlastník může vytvořit roli s názvem „vlastníkovi oblíbenci“, která bude mít růžovou barvu, a role „@everyone“ bude mít barvu bílou. Výsledkem bude, že uživatelé s vytvořenou rolí budou mít růžové jméno a v seznamu uživatelů budou odděleni od ostatních ve své vlastní kategorii nesoucí název jejich role.

Po stránce oprávnění se vytváří role, které mají vyšší oprávnění než běžné uživatelské role, jako například vyhazovat uživatele ze serveru, blokovat jim přístup k serveru, mazat jejich zprávy nebo mají přístup do určité části nastavení serveru, kde běžní uživatelé nemohou. Takovým uživatelům se říká moderátoři a jejich úděl je dohlížet na komunitu serveru, vymáhat dodržování pravidel a asistovat s organizačními záležitostmi majiteli serveru, nebo ho zastoupit v jeho nepřítomnosti. [2] [3] [4]

1.3 Moderace na platformě Discord

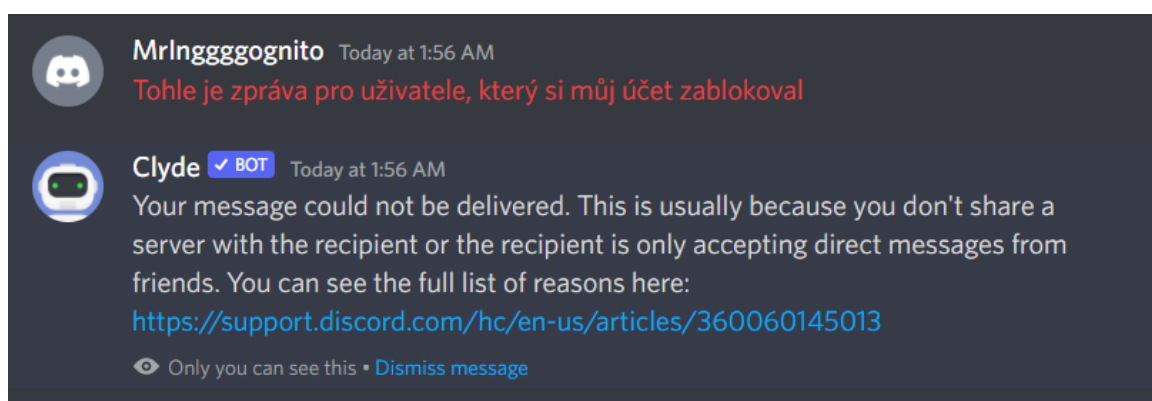
Moderátor je termín, který se v komunitě uživatelů platformy Discord používá k označení uživatele, který má vyšší oprávnění na serveru než ostatní a byl vlastníkem pověřen o dohlížení na komunitu, má právo posuzovat, zda došlo k porušení pravidel serveru či nikoliv a následně vyměřuje tresty za porušení pravidel. K usnadnění práce používají moderátoři takzvané Discord boty, kteří obsahují nástroje a funkce, jež vývojáři vytvořili přesně pro moderátory a jejich potřeby. [5]

2 PROGRAMEM ŘÍZENÝ ÚČET – DISCORD BOT

Bot v kontextu platformy Discord je aplikace obsahující programem řízený účet, určený pro automatizování funkcí serveru. Bot komunikuje přímo s Discord API a má přístup k všem funkcím, jaké má standardní uživatel platformy Discord, ale také ke všem funkcím programovacího nebo skriptovacího jazyka, ve kterém je psán hlavní program, jímž se bot řídí. V současné době se používají knihovny na práci s Discord API pro jazyky Javascript a Python, tedy knihovny Discord.js a Discord.py, nejsou to však jediné možnosti vývoje viz kapitola 2.2.2. Přidání bota na server vyžaduje oprávnění stanovené autorem bota. Po přidání bota je automaticky vytvořena neupravitelná role s oprávněním. [6] [7] [8]

Bota si může vytvořit jakýkoliv uživatel, který tak učiní na stránkách Discord platformy, kde vytvoří aplikaci, té následně vytvoří bota a pak s pomocí jedné ze zmíněných technologií připojí svůj zdrojový kód. Podpora knihovny Discord.py je momentálně ukončena, tudíž není aktuální a neimplementuje tak nejnovější možnosti které Discord platforma nabízí. Oproti tomu je knihovna Discord.js pravidelně aktualizována, upravována a její dokumentace obohacována o nové funkce, které umožňují, aby měl bot přístup k nejaktuálnějším funkcím platformy Discord.

Bot však nemusí být tvořen pouze uživatelem, poněvadž samotná aplikace Discord používá svého interního bota pro systémovou komunikaci přímo s uživatelem. [6] [7] [8]



Obrázek 1 – pokus o kontaktování uživatele, který si daný účet zablokoval

Na obr. 1 lze vidět systémového bota který informuje uživatele s přezdívkou „MrInggggognito“, že jeho zpráva jinému uživateli nebyla doručena a uvádí možné důvody, proč k tomu došlo včetně odkazu na článek podpory platformy Discord s podrobnějším vysvětlením.

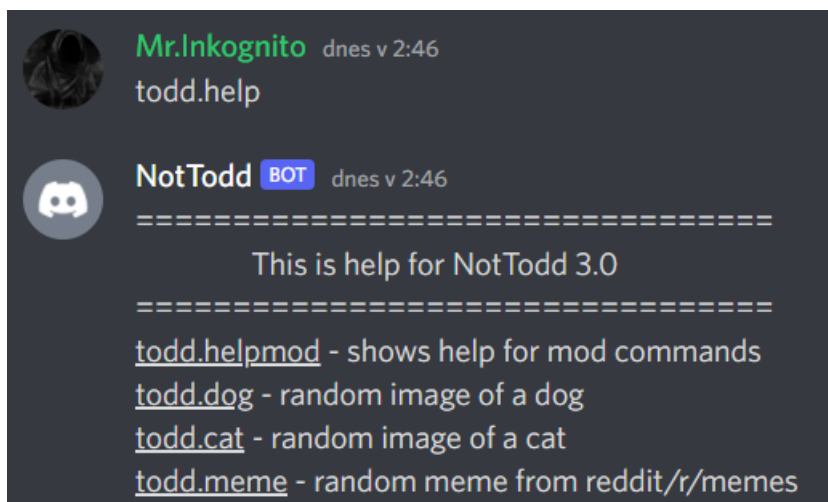
2.1 Ovládání a komunikace s botem

Bot může obsahovat funkce, které prování automaticky bez zásahu jakýmkoliv uživatelem, jako například pozdrav nově přichozích uživatelů označením v textové místnosti následované uvítací zprávou.

Pokud bot obsahuje funkce, které vykonávají uživatelské instrukce, k jejich spouštění slouží specializované příkazy skládající se vždy z hlavního příkazu a příkazových parametrů. Hlavní příkaz volá funkcionalitu bota, zatímco parametry dále upřesňují. Například pro příkaz k zablokování by bylo kromě hlavního příkazu „ban“ potřeba ještě přidat jako parametr příkazu uživatele, kterého chceme zablokovat. [9]

2.1.1 Příkazy typu „legacy“

V dřívějších verzích Discord API a knihoven byly příkazy tvořeny za pomoci standardní zprávy do textového kanálu, kdy odlišení standardní zprávy od příkazu spočívalo v tzv. předponě příkazu neboli „prefix“. [9]

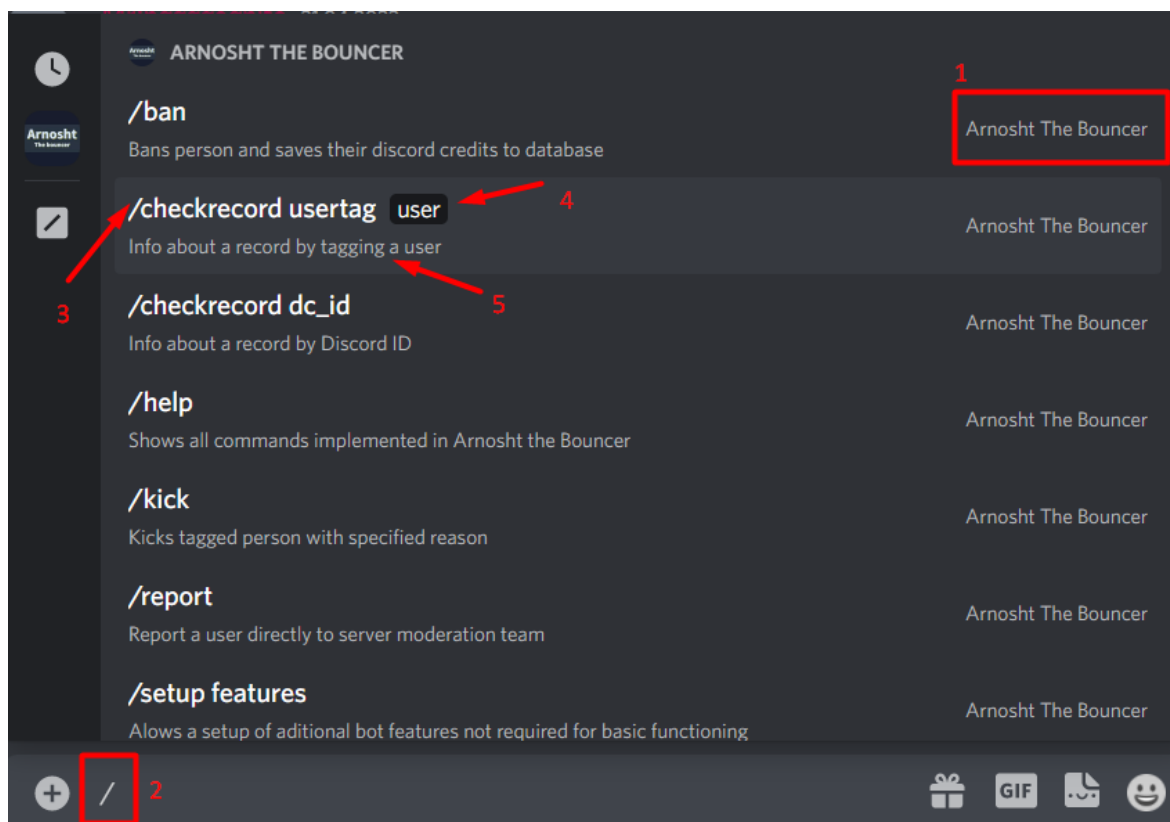


Obrázek 2 – ukázka spouštění příkazu typu „legacy“

Na obr. 2 je příkaz příkazu, který se skládá ze dvou základních částí, a to předpona, díky které bot pozná, že text vložený za ní bude příkaz pro něj, v tomto případě se jedná o předponu „todd.“. Následuje pak název příkazu, tedy „help“, pomocí kterého bot vykoná instrukci naprogramovanou pro daný příkaz, v případě obr. 2 se jedná o výpis příkazu, jež je bot schopen vykonat. V případě parametrů pak bylo naprogramovat funkcionalitu, která určité části řetězce za příkazem brala jako parametry. Hlavní nevýhodou pro uživatele bylo, že se musel příkaz provést do textové místnosti a viděli jej i ostatní uživatelé. Dále pak musel uživatel vědět, jak přesně má příkaz vypadat, nebo hádat či studovat návod pro bota. [9]

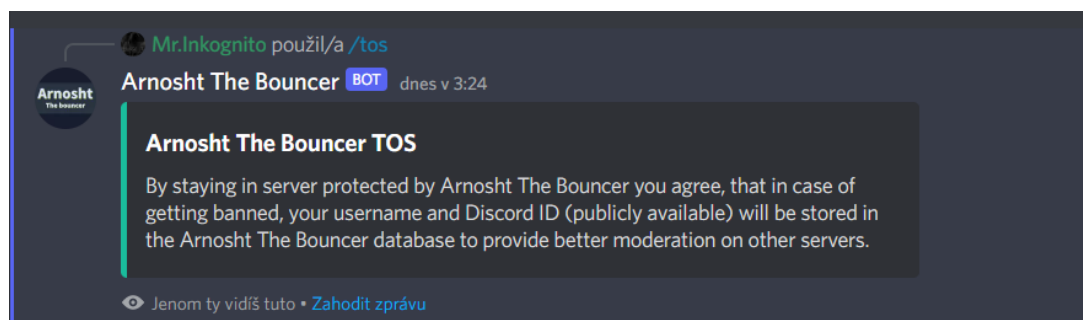
2.1.2 Příkazy typu „slash“

Na konci roku Discord představil nový způsob, jakým budou aplikace a boti ovládány. Jedná se o tzv. „slash commands“, neboli příkazy, které jsou volané univerzálně pomocí znaku lomítka. Jedná se o způsob, jakým Discord chce, aby bylo s boty komunikováno do budoucna, protože poskytují celou řadu výhod. Hlavní výhodou je, že se odeslaný příkaz nezobrazí jako zpráva v textovém kanále pro ostatní uživatele, nýbrž jako skrytá zpráva v chatu, jež vidí pouze bot, kterému patří, ten na danou zprávu provede odpověď s výsledkem příkazu. Další velkou výhodou je, že odpověď bota se dá nastavit tak, že se bude zobrazovat pouze uživateli, který příkaz použil, nikoliv ostatním uživatelům. Díky tomu nedochází k zahlcování textových místností příkazy a odpadá tak nutnost tvořit specializované místnosti jen pro příkazy. Pro uživatele je také výhoda, že při vložení znaku lomítka do pole pro odesílání zprávy jsou vidět všechny příkazy, které bot dovede, jejich popis a parametry potřebné ke správné funkci příkazu. Příkaz může mít parametry které jsou volitelné nebo vyžadované a každý typ parametru má svoji vlastní funkcionalitu, například parametr pro vložení uživatele dovoluje vložit pouze označení na uživatele, nikoliv číslo nebo text nebo parametr pro vložení čísla dovoluje jen číslo, na vše ostatní upozorní a nedovolí uživateli odeslat příkaz. Pro uživatele se tak jedná o pohodlnější a intuitivnější způsob ovládání botů. Nově zavedený systém také velmi usnadňuje vývojářům botů jejich práci. [9]



Obrázek 3 – výpis příkazů po vložení lomítka do pole pro odesílání zpráv

Obr. 3 ukazuje seznam příkazů, který se uživateli zobrazí po vložení znaku lomítka do pole pro odesílání zpráv (viz možnost 2 v obr. 3). U každého příkazu v seznamu je vidět, kterému botu patří (viz možnost 1 v obr. 3), jeho jméno (viz možnost 3 v obr. 3), název parametru, jež daný příkaz vyžaduje (viz možnost 4 v obr. 3) a stručný popis, co daný příkaz dělá (viz možnost 5 v obr. 3)



Obrázek 4 – odeslaný příkaz a výsledek příkazu jako odpověď na něj

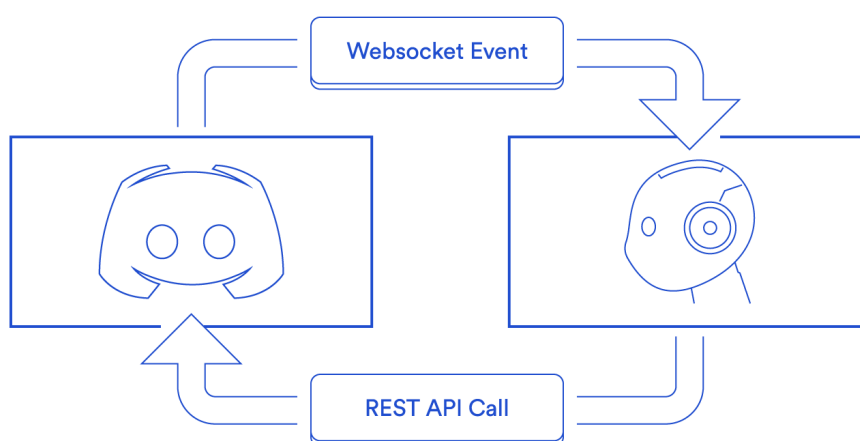
Obr. 4 udává příklad použití příkazu „slash“. Odeslaný příkaz i interakce bota jsou v případě obr. 4 vidět pouze uživatelem spouštějícím příkaz, oproti obr. 2, kde je odeslaný příkaz a interakce bota vidět všemi na serveru.

2.2 Technologie pro vývoj a implementaci botů

Hlavní komunikaci mezi botem a platformou Discord zajišťuje služba Discord API. Ta ovšem není používána přímo, z důvodu časové náročnosti a náchylnosti na chyby v implementaci a používání, které, pokud způsobují přetěžování samotné API nebo s ní nevhodně zachází, mohou vyústit až v zablokování účtu na platformě Discord z důvodu vážného porušení pravidel. Z toho důvodu jsou používány knihovny třetích stran, tzv. „wrappery“, které nejsou spravovány platformou Discord, ale jsou jí ověřeny a schváleny. Knihovny poskytují jednodušší interakci s Discord API a ošetření chybového a nebezpečného zacházení řeší za uživatele. Knihovny třetích stran jsou psány v různých programovacích jazycích, mají své vlastní dokumentace a jejich podpora a aktuálnost vzhledem k verzi Discord API závisí na autoru dané knihovny. [10]

2.2.1 Discord API

Discord API je základní prvek platformy Discord a umožňuje funkci botů. Skládá se ze dvou hlavních částí, WebSocket API, která slouží k získání událostí a odpovědí v reálném čase ze serveru platformy, a REST API sloužící k provádění akcí na platformě. Obr. 5 zobrazuje příklad, jak probíhá komunikace mezi botem (vpravo) a platformou Discord (vlevo) za pomoci volání REST API a následného přijímání odpovědi ze serveru pomocí události WebSocket API. [10]



Obrázek 5 – ukázka komunikace mezi botem a platformou pomocí API [10]

2.2.1.1 WebSocket API

WebSocket API je používána pro přijímání událostí ze serveru platformy Discord, jako jsou například informace, že uživatel poslal zprávu do textové místnosti, uživatel zprávu odstranil, byly aktualizovány role a oprávnění server nebo například že došlo k zablokování uživatele na serveru. Události s sebou nesou také objekty související s jejich voláním, kdy například odpověď na událost vytvoření nové textové zprávy obsahuje také informace o uživateli, který zprávu poslal a informace o serveru, kde k události došlo. Nejedná se však zcela o všechny informace, proto je v případě potřeby těchto informací třeba použít REST API. [10]

2.2.1.2 REST API

Většina akcí, kterých je bot schopen je prováděna skrze REST API, jako například změna oprávnění, blokování uživatelům přístup na server, vytváření místností nebo posílání zpráv do textových místností. REST API se dá také použít k získání informací ze serveru, ale většinou se pro tyto účely používají údaje, které jsou obsaženy v odpovědi WebSocket API, zmíněné v kapitole 2.2.1.1, a jsou následně obohaceny o údaje dočasné pomocné paměti bota. [10]

Tzv. paměť „cache“ je v kontextu s boty platformy Discord paměť, do které bot ukládá stále se opakující informace, které nejsou potřeba předávat REST API a to z důvodu zrychlení zpracování požadavku a omezení velikosti volání REST API aby nedocházelo k přetěžování a neporušila se maximální velikost. Lze tak například ukládat objekt uživatele odesílající zprávy, kdy se na server odešle pouze uživatelský identifikátor místo celého objektu, který je uložen v paměti, a při následné odpovědi je možno při práci s danou odpovědí přistoupit k uživatelskému objektu pomocí dané paměti. [10]

2.2.2 Knihovny třetích stran pro práci s Discord API

Ačkoliv jde k funkcím a voláním Discord API přistupovat přímo, daleko výhodnější a bezpečnější je použít knihovnu, která specifickým způsobem závislým na použitém programovacím jazyku implementuje všechny volání Discord API do funkcí a zároveň zavádí limitace pro dané funkce, aby nebyl překročen „rate limit“, což je speciální limitace, po jíž překročení dojde k zablokování dalších dotazů bota na onu službu. Platforma Discord v současnosti žádnou vlastní knihovnou nedisponuje, je tedy nutné použít knihovny třetích stran, které byly vytvořeny speciálně za účelem usnadnění tvorby nových botů vývojářům. Platforma Discord některé z knihoven třetích stran schválila a potvrdila, že splňují „rate limit“ a další požadavky pro Discord API, a tudíž doporučuje jejich použití místo vlastních operací přímo se zmíněnou API. [10] [11] [12]

Jedny z hlavních knihoven jsou:

Název knihovny	Programovací jazyk
Discord.Net	C#
Discord4J	Java
discord.js	JavaScript
Eris	JavaScript
discord.py	Python
Discordia	Lua

Tabulka 1 – hlavní knihovny pro tvorbu botů [11]

V současné době je nejstahovanější a nejpoužívanější knihovna pro tvorbu botů knihovna Discord.js, jež je tvořena jazykem JavaScript. Dříve se také velmi často používala knihovna Discord.py, psána v jazyce Python. Knihovna se v současné době neaktualizuje a autor opustil její vývoj a údržbu, takže neobsahuje nejnovější funkce, jež jsou podporovány Discord API. [8] [13]

2.2.3 Knihovna Discord.js

Jedná o knihovnu psanou v jazyce Javascript zabalenou jako modul pro spouštěcí prostředí NodeJs, které uživateli dovolí provádět interakce s Discord API velmi jednoduše a zároveň za uživatele řeší případně limitace volání na onu API. Na rozdíl od jiných knihoven postavených na jazyce JavaScript, knihovna Discord.js je tvořena převážně jako objektově orientovaná, čímž dělá výsledný zdrojový kód jednodušší na čtení, pochopení a práci s ním výrazně usnadňuje.

Knihovna je vytvářena jako „open source“ a je veřejně dostupná přes správce verzí GitHub. Při jejím vytváření byl kladen důraz hlavně na použitelnost, konzistenci, celkový výkon knihovny a téměř stoprocentní pokrytí celé Discord API. Tým zodpovědný za vývoj a údržbu také aktualizuje a implementuje nové funkce do knihovny téměř hned co jsou nové funkce implementovány platformou Discord do Discord API. [13] [14]

2.2.3.1 JavaScript

Jedná se o objektově orientovaný skriptovací jazyk určený primárně pro tvorbu dynamických webů. Zdrojový kód je překládán na straně uživatele přímo jádrem prohlížeče a umožňuje tak, aby pokročilé a dynamické uživatelské rozhraní mohlo být generováno v reálném čase na straně klienta. JavaScript také umožňuje propojení aplikace se službami třetích stran, databázovými API nebo dynamickou kontrolu formulářů.

JavaScript je netypový jazyk, díky kterému nepoužívá proměnné pevně daného datového typu jako typové skriptovací jazyky, například TypeScript, odvozený z JavaScriptu.

V současné době je JavaScript využíván primárně pomocí frameworků jako React, Angular nebo Vue na straně klienta a tvoří „front end“ webových stránek. S příchodem spouštěcího prostředí NodeJs a jemu podobným je možné JavaScript používat i pro „back end“ vývoj na straně serveru. [15] [16]

2.2.3.2 NodeJs

NodeJs je běhové prostředí, které umožňuje spouštět JavaScript na straně serveru. Kromě klasického psaní skriptů do webových stránek a tvorby „front end“, je možné pomocí něj vytvořit i „back end“ webových aplikací. [17]

NodeJs pracuje jako „single-thread“ aplikaci neboli že aplikace pracuje pouze na jednom vlákně. Z toho důvodu využívá principu asynchronního programování neboli že na sebe navzájem jednotlivé výpočetní úkony nečekají, čímž se zrychlí práce daného prostředí. [18]



Obrázek 6 – ukázka zpracovávání požadavků prostředím NodeJs [18]

Obr. 6 zobrazuje schéma fungování serveru v prostředí NodeJs. Klient zašle požadavek na server, ten následně požadavek odešle do smyčky událostí, smyčka přidělí požadavek příslušné operaci, která jej zpracuje. Během toho server přijímá další požadavky, které následně odesílá na zpracování. Díky tomu není server blokován náročným požadavkem a nemusí čekat na jeho dokončení. Když je požadavek zpracován, dojde k opětovnému zavolání a následnému odeslání výsledku operace zpět jako odpověď na klientský požadavek. [18]

Moduly, výhody a nevýhody NodeJs

NodeJs umožňuje rozšiřovat funkcionalitu pomocí modulů, instalujících se pomocí NPM neboli správce balíčků prostředí NodeJs. Modul je knihovna, kterou si může uživatel přidat do svého projektu a tím zásadně rozšířit jeho funkcionalitu. Jedním ze stažitelných modulů je i knihovna Discord.js zmíněná v kapitole 2.2.3. Při instalaci modulů je potřeba klást důraz na verzi daného modulu, závislosti, které má daný modul na modulech jiných a bez kterých by nefungoval, jak tvůrce modulu zamýšlel, nebo nefungoval vůbec. Při vybírání modulu je také potřeba obezřetnost, jelikož většina modulů je vytvářena a spravována třetí stranou, podobně jako mobilní aplikace na platformě Google Play, jsou tak neoficiální a NodeJs nemůže zaručit neexistenci škodlivého kódu nebo efektivnost daného modulu. [17] [18]

Mezi hlavní výhody patří, že se jazyk JavaScript dá využít jak pro „front end“ tak i pro „back end“, tudíž není v případě NodeJs učení nových principů. Dále díky svému asynchronnímu přístupu dokáže zpracovat více žádostí najednou nezávisle na sobě a tím zajišťuje vyšší rychlost zpracování požadavků od uživatelů. V neposlední řadě je velká výhoda i aktivní komunita, která vytváří, aktualizuje a rozšiřuje moduly, které usnadňují práci programátorům pracujícím v NodeJs. [17] [18]

S aktivní komunitou se pojí i nevýhody, například nestabilita API, rozhraní modulů nebývají zpětně kompatibilní, takže pokud dojde k aktualizaci NodeJs, musí se změnit kód modulů, aby zajistil bezproblémovou funkčnost. Také pokud dojde k zásadní změně v modulu, na kterém závisí moduly jiné, je třeba dané moduly také aktualizovat, jinak dojde k jejich nestabilitě nebo nefunkčnosti a tím pádem je nutné aktualizovat i program využívající dané moduly přizpůsobit funkcionalitu nové verzi. [17] [18]

3 TERMINOLOGIE S PROBLEMATIKOU KYBERŠIKANY

Šikana je termín, který je v moderní společnosti velmi dobře známý. Týkat se může kohokoliv a kdykoliv. Nejčastěji se však týká studentů základních a středních škol. Jedná se o opakované slovní, fyzické nebo sociální napadání. Agresor zneužívá svého postavení nebo vlastností, jako je fyzická síla, popularita, nebo vlastnictví citlivých informací k fyzickému ubližování, psychickému týrání, napadení sociálního statutu nebo vyhrožování násilím na šikanovaném jedinci, který se většinou cítí bezbranný a působí tak na agresora jako vhodný cíl. Mezi fyzické projevy šikany se řadí fyzický útok či opakované napadání jedince, poplívání jedince, strkání nebo ničení osobního majetku jedince za účelem ublížení. Jako slovní šikana se označuje opakované vulgární označování, provokování, nevhodné sexuální nářky a vyhrožování. Sociální neboli společenská šikana má za cíl narušit společenské postavení jedince pomocí veřejného zesměšňování, cíleného vyčleňování nebo šíření nepravdivých pomluv s cílem narušit společenský statut jedince a ovlivnit tak názor cizích lidí na něj. [19]

3.1 Kyberšikana

Kyberšikana je termín označující druh šikany, který probíhá přes informační technologií. S kyberšikanou se lze nejčastěji setkat na sociálních sítích, kde pod pocitem anonymity šíří uživatelé platformou nenávistné komentáře cílené na uživatele nebo skupinu, vyhrožování násilím, fotografie a videa soukromého charakteru poškozující šikanovaného nebo odcizí účet jiného uživatele a ten pak používají k porušování pravidel a pošpinění jména onoho uživatele. Mezi velmi časnými projevy kyberšikany patří také zasílání výhrůžných emailů a soukromých zpráv, pořizování videozáznamů nebo záznamů zvukových a jejich následné uveřejnění s cílem poškodit osobu, tvorba internetových stránek a blogů přímo určena k ponižování osob, vydírání, odhalování tajemství nebo opakovaný pokus o kontaktování či pronásledování osob. [20] [21]

Mezi velmi časnými a velmi nebezpečnými kategoriemi kyberšikany patří sexting, kybergrooming a kyberstalking. Sexting znamená rozesílání textových zpráv, obrázků nebo videí se sexuálním obsahem, většinou se jedná o obrázky a videa šikanované osoby, které agresor rozesílá veřejně po internetu a tím poškozuje šikanovanou osobu, nebo naopak vyhrožuje rozesláním citlivých materiálů za účelem vydírání šikanované osoby. Kybergrooming označuje situaci, kdy se agresor snaží přimět pomocí sofistikované psychologické manipulace k důvěře a následnému osobnímu setkání, kde pak může dojít k sexuálnímu obtěžování nebo zneužití. [20]

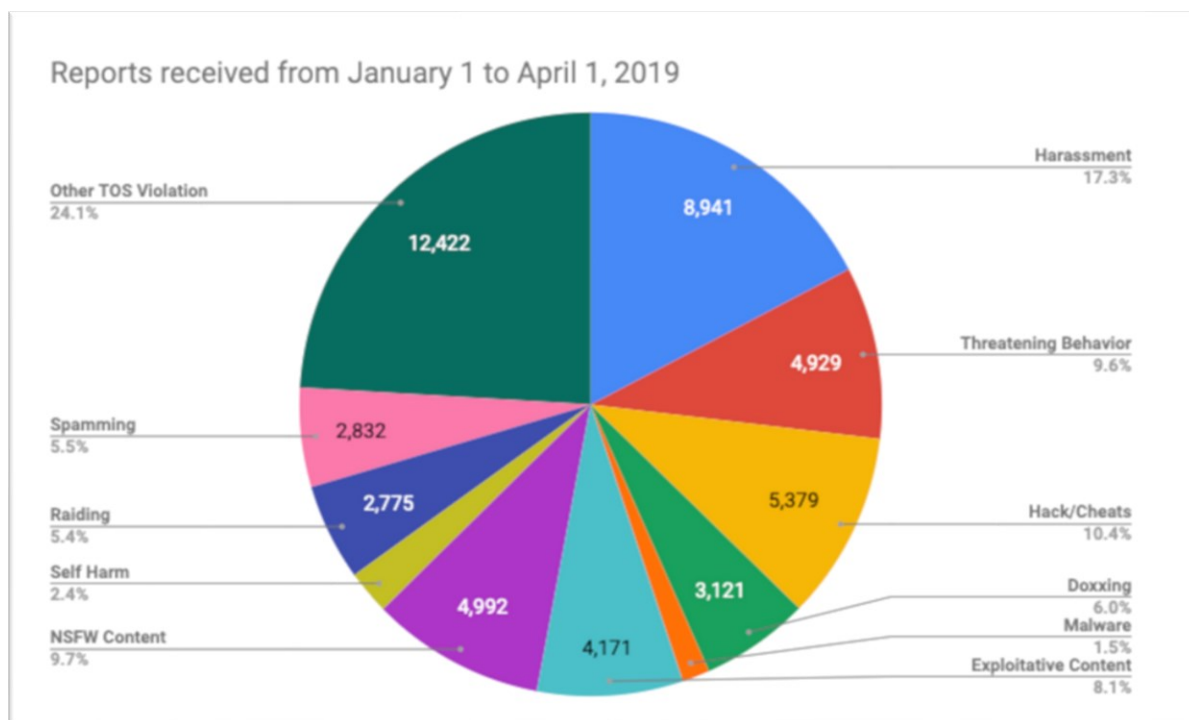
Velmi častý cíl kybergroomingu bývají nezletilé děti, které si agresor vyhlédne pomocí sociálních sítí nebo komunity počítačových her, díky čemuž je snadnější s dítětem manipulovat. Kyberstalking a doxxing jsou další velmi časté příklady kyberšikany, ke kterým může docházet na internetu. Kyberstalking spočívá v pronásledování osoby na sociálních sítích, sledování každé aktivity, kterou daná osoba zveřejní a může přerůst až v posednutí. Doxxing znamená zveřejnění uživatelských osobních údajů nebo jiných citlivých informací přímo související s jeho skutečnou identitou na internetu. [20] [22] [23]

Kyberšikana má větší dopad na psychické zdraví, hlavně u dětí, a to z důvodu pocitu, že před šikanou není úniku. Zatímco klasická šikana nastává jen za určitých situací nebo na určitých místech, před kyberšikanou není unik, dokonce ale ve vlastním domově. Dalším faktorem je, že šikana online přetrvává, i když oběť změní prostředí. U obětí kyberšikany se pak můžou vyskytovat deprese, strach a úzkosti kolem lidí, protože si oběť není jistá, zda někdo například neviděl fotku, kterou šíří agresor, a v extrémních případech může vézt i k sebevraždě oběti. Dalším důsledkem, hlavně u studentů je vynechávání školní docházky a zhoršování známek, z důvodu ponižování, které má díky sociální sítí mnohem větší dosah. [21] [23]

3.2 Platforma Discord a kyberšikana

Ačkoliv platforma Discord má ve svých podmínkách, že nebude za žádných okolností tolerovat například násilné projevy, projevy rasistické či obtěžování, tak na uživatele serverů dohlíží hlavně majitelé serverů, případně moderátoři. Discord dal k dispozici velké množství nástrojů a možností, jak hlídat a moderovat vlastní servery, možnosti nastavení soukromí svého účtu jako například viditelnosti účtu na platformě nebo možnost, že daného uživatele mohou kontaktovat pouze uživatelé, kteří byli přidáni do seznamu přátel. Obecně však za události a obsah jednotlivých serverů odpovídá jejich majitel. Platforma dala k dispozici nástroj pro tvorbu sady vlastních pravidel, která se následně zobrazují v oddělené místnosti. Stává se tedy, že uživatelé opakovaně porušující pravidla serveru, jsou následně zablokováni serverovým moderátorem a mají zamezen přístup k onomu serveru, mohou napadnout server patřící jinému vlastníkovu a opakovaně porušovat pravidla a šířit nenávisť, dokud nejsou opět zablokováni a nepřesunou se na další server. Pokud se chce komunita zablokovat uživatele z celé platformy Discord, je potřeba kontaktovat hlavní podporu. [24]

Kontaktování hlavní podpory je ale zdoluhavý proces, který má ne vždy požadované výsledky a není jisté, zda se k problému podpora dostane včas, a tak nejlepší možná obrana proti útokům na servery je jen blokáce problémových jedinců. [24]



Obrázek 7 – nahlášení uživatelů od 1. ledna do 1. dubna 2019 [25]

Na prvním místě grafu nahlášení (obr. 1) se nachází kategorie „jiné porušení TOS“, kam se řadí nahlášení nejrozumnějšího typu. Hned druhé v pořadí je obtěžování, což je jeden z nejčastějších problémů na serverech platformy Discord. Společně s vyhrožováním tvoří značnou část kyberšikany odehrávající se na platformě. Statistika je tvořena pouze nahlášeními, které se dostanou až k hlavnímu moderačnímu týmu platformy Discord, bohužel nemalá část nahlášení zůstává neevidována a nevyřešena. [20] [25]

Hlavním problémem moderace na serverech platformy Discord je, že i když poskytuje funkce, které mohou moderátoři používat k vymáhání dodržování pravidel a následným trestům za porušení, neposkytuje možnost odhalit o uživateli zjistit informace o porušení pravidel jiných serverů. Uživatelé proto problém řeší sdílením obrázků a uživatelských jmen uživatelů narušujících komunitu pomocí jiných sociálních sítí, jako je například Twitter. [3] [25] [26]



Obrázek 8 – uživatel informující přes Twitter komunitu o závadném chování [27]

Na obr. 8 je uživatel, sdílející na svůj profil sociální sítě Twitter uživatelské jméno a ukázkou konverzace odehrávající se na platformě Discord. Uživatel po uživatelským jménem caoimheinnit informuje komunitu o sexuálním obtěžování a rasových projevech uživatele s uživatelským jménem Vickey a prosí o sdílení, aby se tato informace dostala k co nejvíce lidem. Ve varování nebyl poskytnut identifikátor účtu, pouze uživatelské jméno, z čehož plyne že po změně uživatelského jména se může agresor připojit na jiný server a páchat škodu.

II. PRAKTICKÁ ČÁST

4 NÁVRH BOTA PRO USNADNĚNÍ MODERACE

Tato kapitola se zabývá návrhem a výběrem vhodných vlastností pro bota.

Cílem bylo navrhnout bota, který bude disponovat základními moderačními funkcemi stejně, jako má k dispozici běžný moderátor. Bot rozšiřuje příkazy, které jsou integrovány do aplikace o své vlastní, které rozšiřují jejich běžnou funkcionalitu. Hlavní důvod, proč je potřeba rozšířit funkcionalitu příkazů je, že kdykoliv provede moderátor nebo vlastník serveru zablokování uživatele na serveru jím ovládaném, uživatel je sice zablokován, ale není nikde evidován mimo zablokovaný server.

Pokud dojde k zablokování uživatele pomocí bota, bot automaticky uloží veřejně dostupné informace o zablokovaném do své databáze, která se společná napříč všemi servery, které obsahují onoho bota. Zatím, co uživatelské jméno si může daný uživatel změnit, jeho identifikátor dostupný přes aplikaci Discord změnit nelze a je unikátní pouze jeden na účet.

Evidování zablokovaných uživatelů pro, například, rasistické nářázky, obtěžování nebo zasílání nevhodných obrázků, pomůže při odhalení zmíněných problémových uživatelů na jiných serverech, a to kontrolou hned jak se uživatel připojí. O potenciálním problémovém uživateli se tak majitel serveru a moderátoři mohou dozvědět dříve, než stihne napáchat škody.

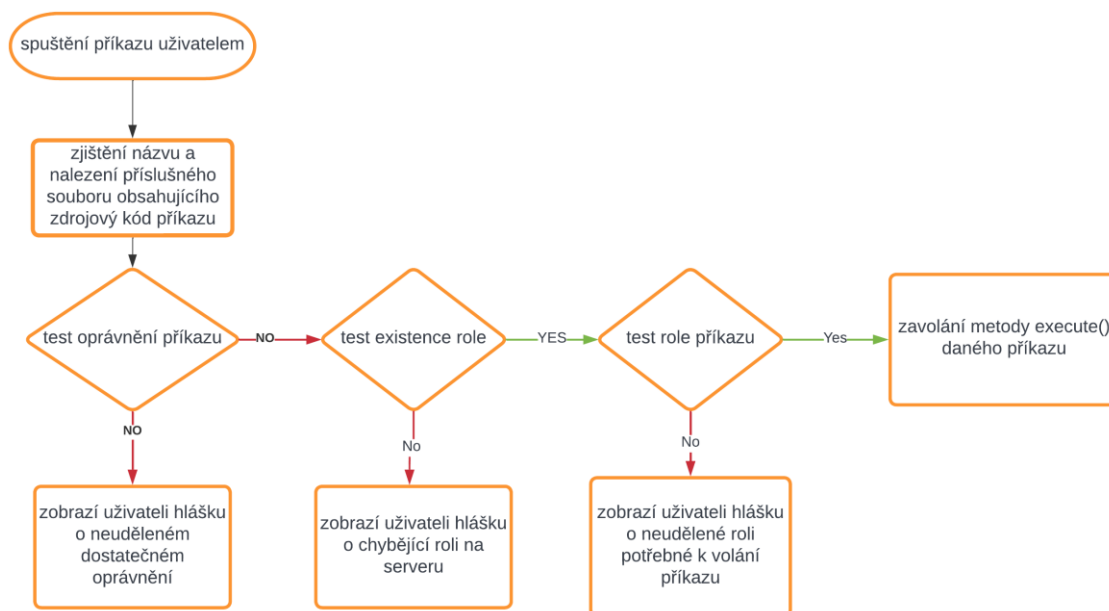
Bot umí rozlišovat uživatele podle oprávnění k nim aktuálně přiděleným a povoluje provést akce pomocí něj pouze uživatelům autorizovaným provádět požadované akce. Pokud nemá uživatel spouštějící příkaz roli, která disponuje oprávněním k blokaci uživatelů na serveru, bot odpoví uživateli chybovou hláškou o chybějícím oprávnění.

4.1 Automatické vlastnosti bota

Jedná se o soupis akcí a vlastností, které bot provádí automaticky bez nutnosti opakovaného spouštění uživatelem. Vlastnosti, jako je například struktura s názvem „spouštěč příkazů“, která automaticky kontroluje oprávnění a role uživatelů používajících příkaz a také provádí kontrolu, zda uživatel příkaz nespouští příliš často po sobě, je-li tato funkce vyžadována daným příkazem

4.1.1 Spouštěč příkazů

Z důvodu usnadnění práce s příkazy a přehlednosti kódu, jsou příkazy rozděleny do jednotlivých souborů. Spouštěč příkazů je struktura schopná identifikovat požadovaný soubor příkazu dle názvu a spustit zdrojový kód, jenž daný soubor obsahuje.



Obrázek 9 – návrhové schéma spouštěče příkazů

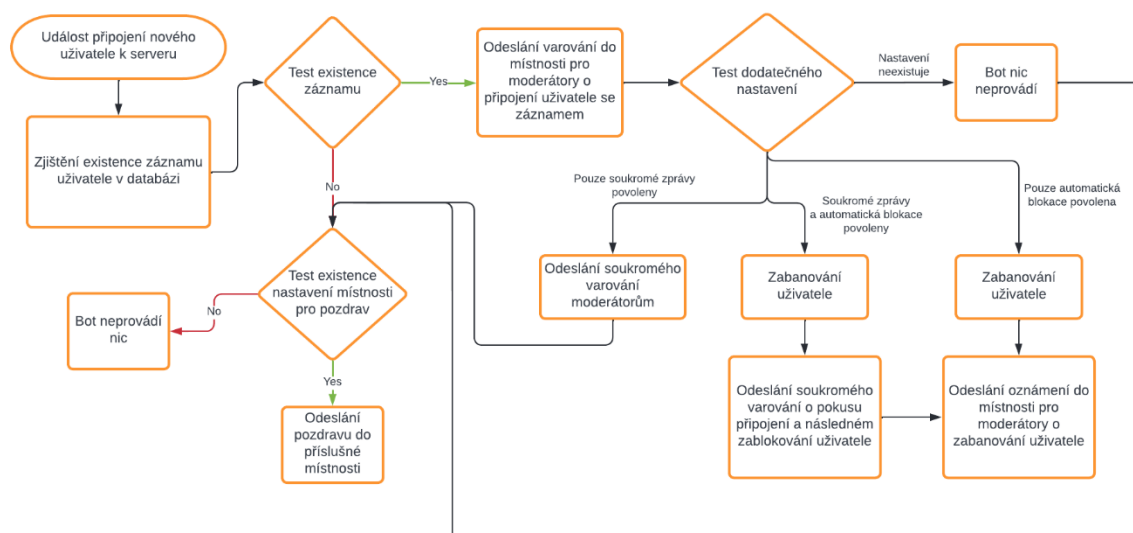
Obr. 9 obsahuje návrhové schéma spouštěče příkazů. Spouštěč převezme příkaz od uživatele, zkontroluje, zda má uživatel nastaveno oprávnění vyžadováno příkazem, následně ověří, že server obsahuje specializovanou roli vytvořenou botem pro moderační operace a pokud ano, zda je tato role přidělena uživateli. Pokud je uživatel oprávněn příkaz použít, dojde ke spuštění zdrojového kódu uvnitř souboru s příkazem. Pokud nastane situace, že daná role na severu neexistuje, je uživatel vyzván k jejímu vytvoření, má-li oprávnění tak učinit. Pokud ne, musí uživatel kontaktovat moderátorský tým nebo přímo majitele serveru o skutečnosti, že nebylo provedeno správné prvotní nastavení systému.

4.1.2 Pozdrav nově příchozích členů

Majitelé serveru mohou zapnout funkce automatického pozdravu nově příchozích uživatelů tím, že pomocí příkazu nastaví textovou místnost jako místnost pro pozdravy. Následně bot odešle do nastavené místnosti pozdrav každému nově příchozímu uživateli, seznámí jej se svou přítomností a účelem na serveru a poučí jej o následcích porušování pravidel a následné blokaci při kterém dochází k vytváření záznamu.

4.1.3 Automatická kontrola nově příchozích členů

Bot kontroluje každého uživatele, který se připojí na server. Dojde-li k zjištění existence záznamu, bot zašle upozornění do specializované textové místnosti a také soukromou zprávu moderátorům, je-li tato rozšiřující možnost nastavena majitelem serveru.



Obrázek 10 – návrhové schéma pro kontrolu uživatelů po připojení

Obr. 10 popisuje postup který nastane, když se na server připojí uživatel. Po připojení dojde ke prohledání databáze zablokovaných uživatelů, zda má nově příchozí záznam. Pokud se v databázi záznam nenachází, bot zkontroluje, zda je nastavena místnost pro pozdravy, pokud není, nic neprovádí. Když místnost nastavena je, bot odešle pozdrav a označí nově příchozího uživatele. Pokud uživatel záznam má, bot odešle varovnou správu o přítomnosti uživatele se záznamem na serveru do příslušné místnosti s názvem „arnosht-join-warning“. Bot dále zjistí, zda existuje dodatečné nastavení kontroly nově příchozích. Neexistuje-li nastavení, pak bot dále nic neprovádí a přejde rovnou k pozdravu.

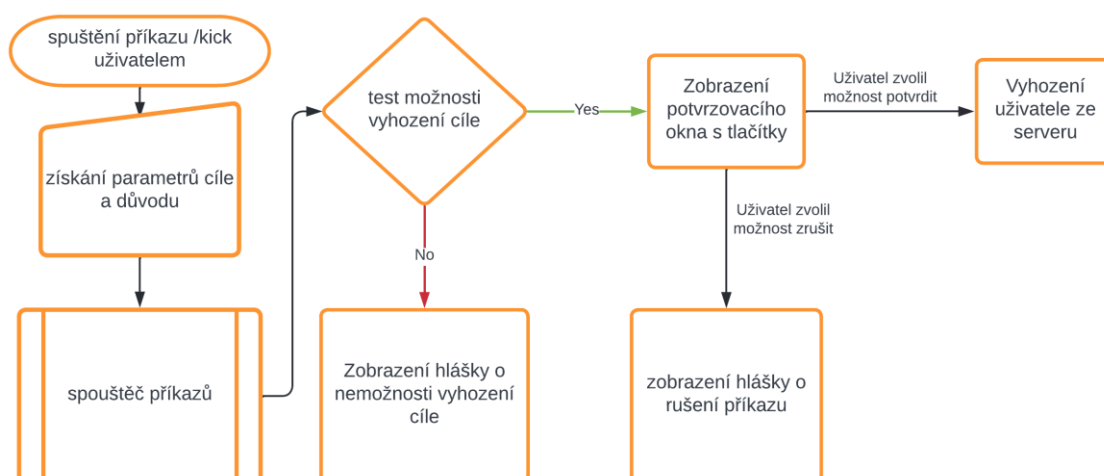
Pokud nastavení existuje, bot dodatečně provede akce, které byly nastaveny. Mezi tyto dodatečné akce patří odeslání soukromé zprávy všem uživatelům, kteří mají moderátorskou roli poskytovanou botem a automatické zablokování uživatelů se záznamem s dodatečným odesláním informace o zablokování do specializované místnosti a soukromé zprávy, je-li možnost soukromých zpráv povolena.

4.2 Manuální vlastnosti bota

Jedná se o soupis manuálních akcí, které je bot schopen vykonat po vyžádání uživatelem. Jako příkazový systém bot používá příkazy typu „slash“, neboli příkazy vyvolávané pomocí znaku lomítka následované příkazem a parametry viz. kapitola 2.1.2. Každý příkaz zobrazující se v nápovědě má svůj vlastní popis, aby měl uživatel spouštějící příkaz přehled o tom, co daný příkaz prování bez nutnosti vyhledávání v nápovědě obsažené v botu viz obr. 5 možnost číslo pět kapitoly 2.1.2. Parametry mají vlastní datové typy a omezení, které uživateli dovolí odeslat příkaz pouze, pokud jsou data validní.

4.2.1 Odebrání uživatele ze serveru.

Použitím příkazu odebere bot specifikovanou osobu ze serveru a spolu s důvodem který byl zadán ji uloží do protokolu událostí serveru. Uživatel je i nadále schopen opětovného připojení k serveru, má-li k dispozici pozvánku.

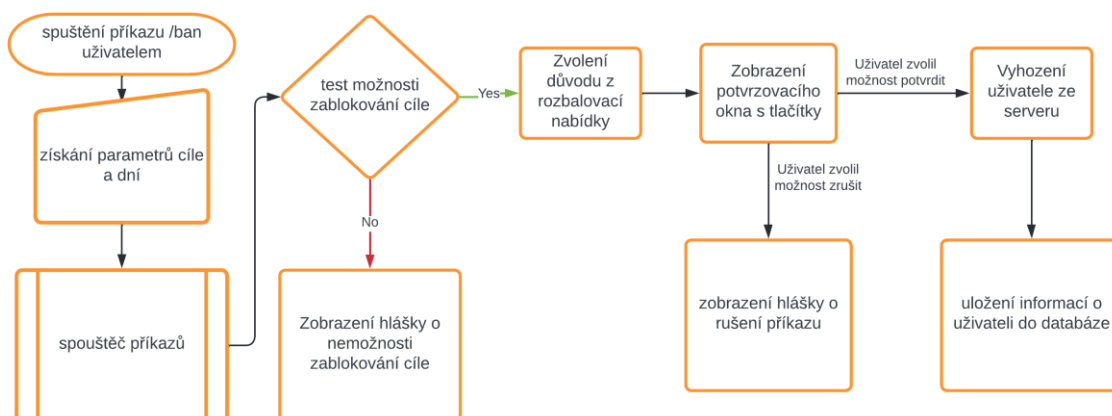


Obrázek 11 – návrhové schéma příkazu pro odebrání uživatelů

Obr. 11 popisuje schéma příkazu pro odebrání uživatelů ze serveru. Příkaz přebírá dva parametry, jimiž jsou cíl neboli uživatel, který má být odebrán a důvod, který se následně uvede do protokolu událostí serveru jako důvod odebrání uživatele. Pokud má uživatel spouštějící příkaz oprávnění k jeho použití, ale pokusil se ho použít na uživatele, který je v hierarchii serveru na vyšší úrovni, například majitel serveru nebo při pokusu odebrat sám sebe, dojde k zobrazení chybové hlášky o skutečnosti, že daného uživatele nelze ze serveru odebrat. Pokud uživatele odebrat lze, zobrazí se uživateli potvrzovací dialog, který ho vyzve k potvrzení volby odebrání uživatele. Pokud uživatel zvolí tlačítko zrušit, dojde ke zrušení akce a příkaz se sám ukončí, následovaný informativní hláškou, že byl příkaz ukončen. Když dojde ke zvolení možnosti potvrdit, dojde k odebrání cíleného uživatele a zapsání události včetně důvodu do protokolu událostí na serveru.

4.2.2 Zablokování uživatele na serveru

Použitím příkazu pro zablokování, bot danou osobu odebere ze serveru a zamezí jí přístup ze stejného uživatelského účtu na daný server, a to až do manuálního odblokování pověřeným uživatelem. Osobu společně s důvodem zablokování uloží do protokolu událostí serveru a následně vytvoří záznam, který uloží do své databáze. Odblokováním uživatele na serveru nedojde k vymazání záznamu v databázi, obdobně jako nedojde k vymazání záznamu z trestního rejstříku osoby po spáchání zločinu.

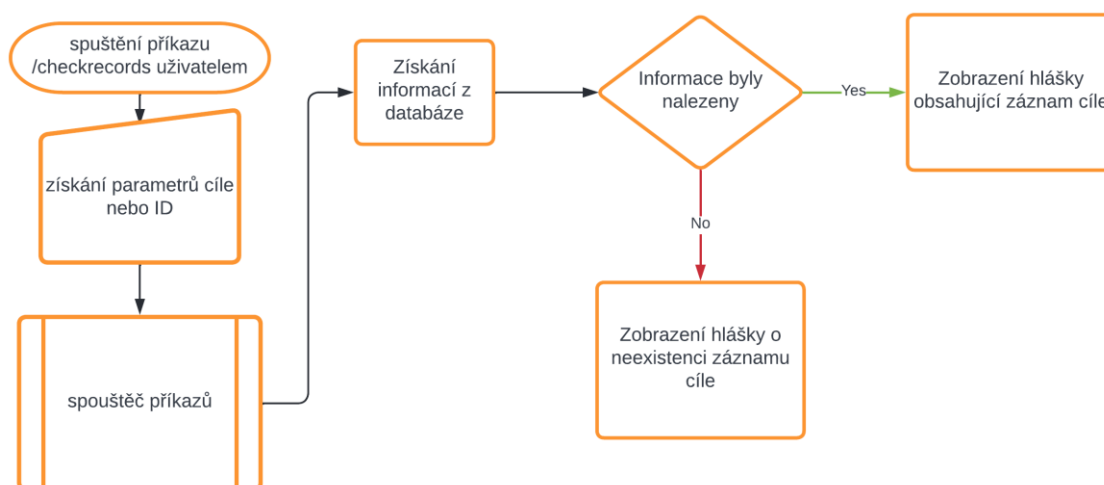


Obrázek 12 – návrhové schéma příkazu pro blokování uživatelů

Obr. 12 zobrazuje schéma návrhu pro příkaz sloužící k zablokování přístupu na server a vytváření záznamů pro zablokované uživatele. Po zadání příkazu a zvolení parametrů, tedy cíleného uživatele k zablokování a čísla udávajícího počet, za kolik posledních dní mají být uživateli zprávy odstraněny. Následně spouštěč příkazů zkontroluje příslušné oprávnění a spustí příkaz. Nastane-li situace, že se uživatel snaží zablokovat cíl na vyšší pozici v hierarchii serveru, je zobrazena chybová hláška o nemožnosti zablokování cíle. Pokud má uživatel oprávnění zablokovat cíl, zobrazí se nabídka předem definovaných důvodů blokování. Po zvolení důvodu je zobrazen potvrzovací dialog obsahující souhrn vybraných možností. Při zvolení možnosti zrušit, dojde ke zrušení akce a zobrazení hlášky o zrušení. Dojde-li ke zvolení možnosti potvrdit, cílený uživatel je zablokován, informace a důvod zapsán do protokolu událostí serveru a informace o zablokovaném jsou odeslány do databáze.

4.2.3 Vyhledání uživatele ve sdílené databázi

Použitím příkazu bot údaje poskytnuté o cíleném uživateli vyhledá v databázi a zobrazí výsledek hledání. Když cílený uživatel žádný záznam nemá, zobrazí bot zprávu, že nenalezl žádnou shodu. Pokud cílený uživatel záznam v databázi má, pak zobrazí, kolikrát byl uživatel zablokován, jaký byl nejčastější důvod, na kolika serverech byl zablokován a nejčastější přezdívky používané daným uživatelem.



Obrázek 13 – návrhové schéma příkazu prověřujícího uživatele

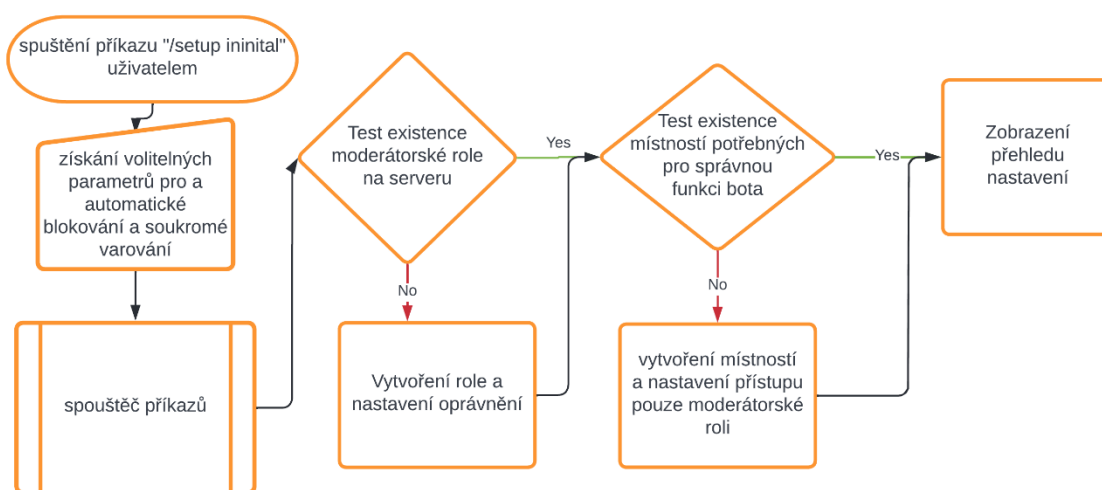
Obr. 13 popisuje schéma příkazu pro prověření uživatele, zda má záznam v databázi. Příkaz je možné spustit ve dvou variantách. První varianta přijímá jako parametr přímo vybrání uživatele ze seznamu uživatelů na serveru, kde je podmínka, že musí být daný cílený uživatel členem serveru. Druhá varianta spočívá ve vložení identifikátoru uživatelského účtu neboli Discord ID, díky kterému lze zobrazit záznam uživatele, který momentálně

4.2.4 Příkaz pro zobrazení nápovědy

Příkaz, k jehož spuštění je oprávněn každý uživatel serveru, zobrazuje rozsáhlý seznam všech příkazů, které momentálně bot dovede provést. Zobrazuje také návod, jak provést prvotní nastavení serveru, aby všechny funkce bota fungovaly korektně.

4.2.5 Příkaz pro prvotní nastavení

Příkaz umožní majiteli serveru provést prvotní a nejdůležitější nastavení serveru. Bez tohoto nastavení nelze zaručit správnou funkci bota a nelze vyloučit neočekávané chování.



Obrázek 14 – návrhové schéma příkazu prvotního nastavení

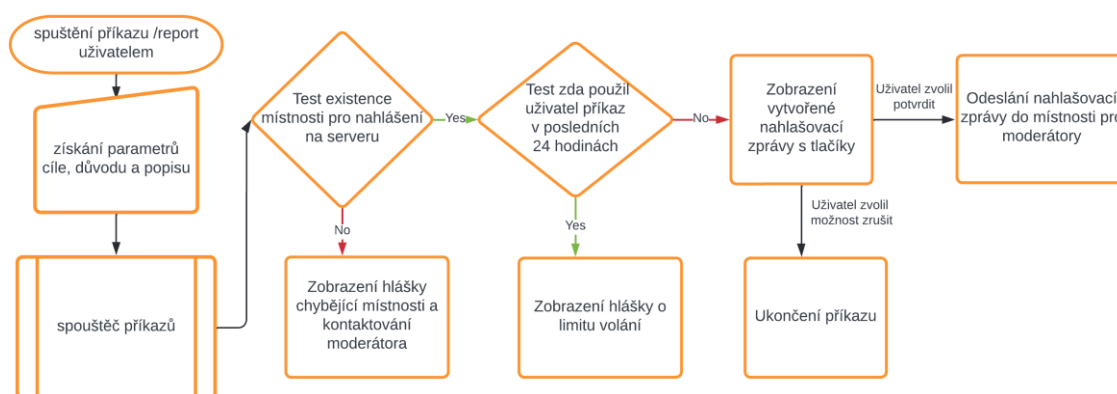
Na obr. 13 lze vidět schéma příkazu pro prvotní nastavení serveru. Po spuštění majitelem provede test, zdali se na serveru nachází speciální moderátorská role, opravňující moderátory k určitým příkazům bota, nazývaní se „Arnosht_moderator“. Pokud se zde role nachází, nastavení je přeskočeno, pokud na serveru role neexistuje, je automaticky vytvořena a jsou jí přidělena potřebná oprávnění. Dále se provedou kontroly pro jednotlivé místnosti nutné pro správnou funkci bota, zamčené před běžnými uživateli, a tudíž pro ně neviditelné.

Seznam místností obsahuje textový kanál s názvem „arnosht-report-log“, do kterého jsou zasílány uživatelská nahlášení porušení pravidel. Do textového kanálu „arnosht-join-warnings“ jsou posílány oznámení o uživateli se záznamem, kteří se připojili na server. Textový kanál „arnosht-mod-text“ a hlasový kanál „arnosht-mod-talk“ slouží pro textovou a hlasovou komunikaci mezi moderátorským týmem nebo mezi týmem a majitelem serveru, který má do místností přístup.

Příkazem pro dodatečné nastavení lze povolit nebo zakázat možnost, aby se kromě varování o připojeném uživateli se záznamem posíláném do specializované místnosti, posílaly oznámení do soukromých přímých zpráv každého s rolí „arnosht_moderator“. Lze také nastavit možnost automatické blokace, kdy jsou nově příchozí se záznamem automaticky blokováni.

4.2.6 Příkaz pro nahlášení uživatele

Příkaz, na jenž má právo každý uživatel, umožní nahlásit jiného uživatele serveru moderátorskému týmu, diskrétně, jednoduše a uceleně.



Obrázek 15 – návrhové schéma pro nahlášovací příkaz

Obr. 14 popisuje návrhové schéma příkazu pro nahlásování jiných uživatelů. Příkaz obsahuje parametr pro cíleného uživatele, který bude nahlášen, důvod pro nahlášení vybraný z předem definovaných důvodů a detailního popisu, ve kterém nahlášující uživatel popíše důvod, proč se rozhodl cíleného uživatele nahlásit. Po použití se provede kontrola, zda se na serveru nachází místnost s názvem „arnosht-report-log“, do které se nahlášení posílají. Pokud role není přítomna, uživateli je zobrazena chybová hláška o chybějící místnosti a nutnosti provést první nastavení. Pokud je server nastavený v pořádku, provede se kontrola, zda uživatel spouštějící příkaz onen příkaz nepoužil za posledních dvacet čtyři hodin. Jedná se o

prevenci opakovaného zasílání nahlášení téhož uživatele za sebou. Před odesláním nahlášení je zobrazen potvrzovací dialog zobrazující kompletní nahlašující zprávu, po stisknutí tlačítka potvrdit se zpráva diskrétně bez povědomí ostatních uživatelů server odešle do soukromého kanálu dostupnému pouze moderátorskému týmu, který může předmět nahlášení začít řešit.

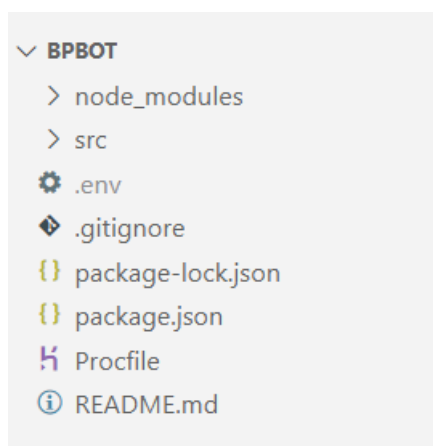
Účel příkazu je možnost odeslat problém celému moderátorskému týmu najednou a diskrétně, bez povědomí ostatních uživatelů, oproti běžnému označení moderátorské role v textové místnosti a popsání důvodu, případně psaní soukromé zprávy individuálně členům týmu.

5 IMPLEMENTACE BOTA

Tato kapitola popisuje, jakým způsobem je bot implementován.

Pro implementaci bota byla zvolena knihovna Discord.js společně s běhovým prostředím NodeJs a jako vývojové prostředí bylo zvoleno vývojové prostředí Visual Studio Code od firmy Microsoft, a to z důvodu možnosti instalace doplňků pro práci s knihovnou Discord.js. Pro ukládání dat byla zvolena databáze jménem MongoDB. Jedná se o „NoSQL“ databázi, tedy databázi ukládající svá data do formátu „JSON“. Pro správu verzí a zálohy byl použit GitHub. Jako jazyk, kterým bude bot komunikovat s uživatelem byla zvolena angličtina, a to z důvodu použitelnosti bota i na zahraničních serverech a univerzálnosti.

5.1 Souborová struktura

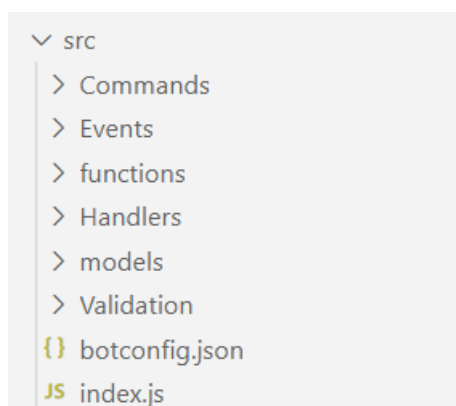


Obrázek 16 – souborová struktura bota

Na obr. 16 lze vidět, jakým způsobem je dělena základní souborová struktura.

- Složka „node_modules“ obsahuje soubory všech modulů a knihoven, které bot využívá.
- Složka „src“ obsahuje zdrojové soubory bota, jako jsou příkazy nebo hlavní spouštěcí soubor

- Soubory „.gitignore“ je soubor obsahující názvy souborů, které nemají být odesílány na GitHub. Nachází se v něm záznam pro soubor „.env“ sloužící jako proměnná prostředí pro vývoj obsahující přístupový token k botu a přístupové údaje k databázi. Proměnné prostředí byly využity při vývoji a pro hotovou aplikaci fungující na serveru se vytváří na straně serveru, kde jsou zabezpečeny.
- Soubory „package.json“ a „package-lock.json“ jsou generovány prostředím NodeJs a používány pro správu nainstalovaných modulů a informací o aplikaci bota.
- Soubor „Procfile“ vyžaduje server na kterém je bot nahrán k zapínání a vypínání bota.
- Soubor „README.md“ je automaticky vygenerován správcem verzí GitHub



Obrázek 17 – obsah složky „src“ v souborech bota

Obr. 17 popisuje obsah složky „src“ ve které se nachází:

- Složka „Commands“ obsahuje složky a soubory jednotlivých příkazů
- Složka „Events“ obsahuje soubory jednotlivých událostí použitých při implementaci
- Složka „functions“ obsahuje spouštěč příkazů viz kapitola 4.1.1
- Složka „Handlers“ obsahuje spouštěč událostí. Všechny vytvořené další spouštěče automatizovaných akcí budou přidávány do této složky
- Složka „models“ obsahuje modely ve formátu JSON pro databázi
- Složka „Validation“ obsahuje dva soubory, jeden se všemi názvy událostí knihovny Discord.js a druhý se všemi jmény oprávnění. Oba soubory jsou využity pro validaci při tvorbě jednotlivých příkazů.

- Soubor „botconfig.json“ obsahuje nastavení pro jméno moderátorské role, na kterou je odkazováno v kódu jako proměnnou, a to z důvodu, aby případná změna názvu role nevyústila v přepisování celé struktury.
- Soubor „index.js“ je hlavní soubor bota, po jehož spuštění dojde k uvedení bota do režimu online

5.2 Hlavní spouštěcí soubor

Hlavní spouštěcí soubor „index.js“ slouží k uvedení bota do provozu.

```
src > JS index.js > ...
1  //imports
2  const Discord = require("discord.js");
3  const fs = require('fs');
4  const config = require("./botconfig.json");
5  require('dotenv').config();
6
7  //declarations
8  const client = new Discord.Client({ intents: 32767 });
9  client.commands = new Discord.Collection();
10
11 const functions = fs.readdirSync('./src/functions').filter(file => file.endsWith(".js"));
12
13 //event handler
14 require("./functions/Events")(client);
15
16 //commands handler
17 const commandFolders = fs.readdirSync('./src/commands');
18
19
20 //bot start
21 (async () => {
22
23     for(file of functions){
24         require(`./functions/${file}`)(client);
25     }
26
27     client.handleCommands(commandFolders, "./src/commands");
28
29     client.login(process.env.TOKEN);
30 })();
31
```

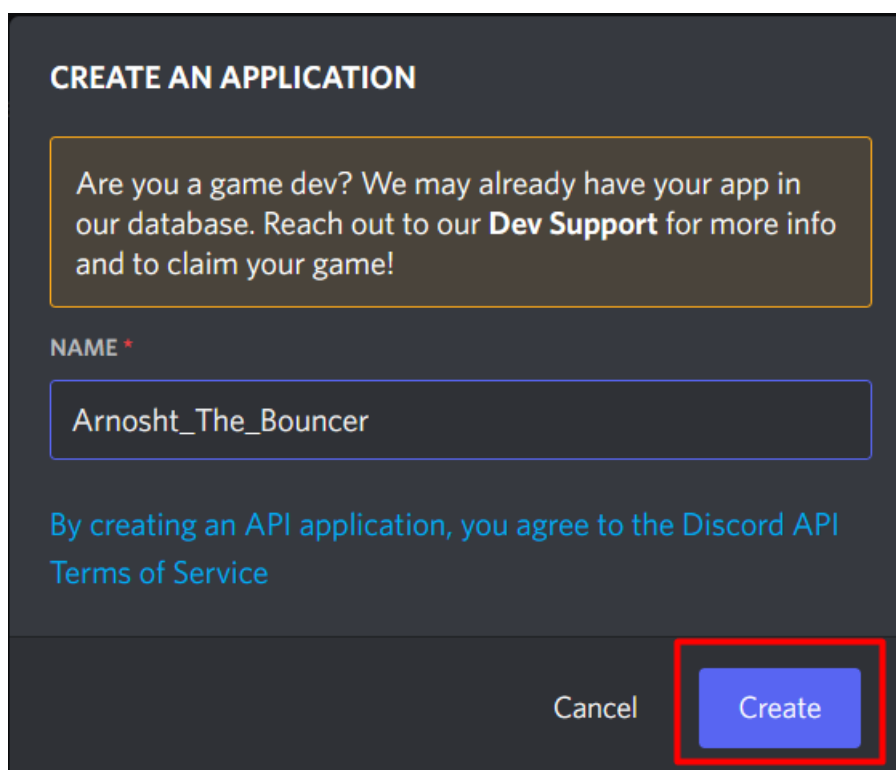
Obrázek 18 – hlavní kód pro spuštění bota

Na obr. 18 lze vidět hlavní spouštěcí část, tedy soubor „index.js“. Na řádcích 2 až 5 probíhá import knihoven a souborů potřebných pro start bota. Na řádku 8 se vytváří nová instance bota, tedy proměnná „client“ vyžadující oprávnění správce, tedy nejvyšší možné. Následně probíhá vytváření instancí pro spouštěč příkazů a událostí, až po řádek 29, na kterém se nachází uvedení bota do stavu online přihlášením zdrojového kódu k účtu typu bot.

Do metody „login“ se předává proměnná prostředí „TOKEN“, která obsahuje unikátní identifikátor pomocí kterého je možné propojit zdrojový kód s aplikací.

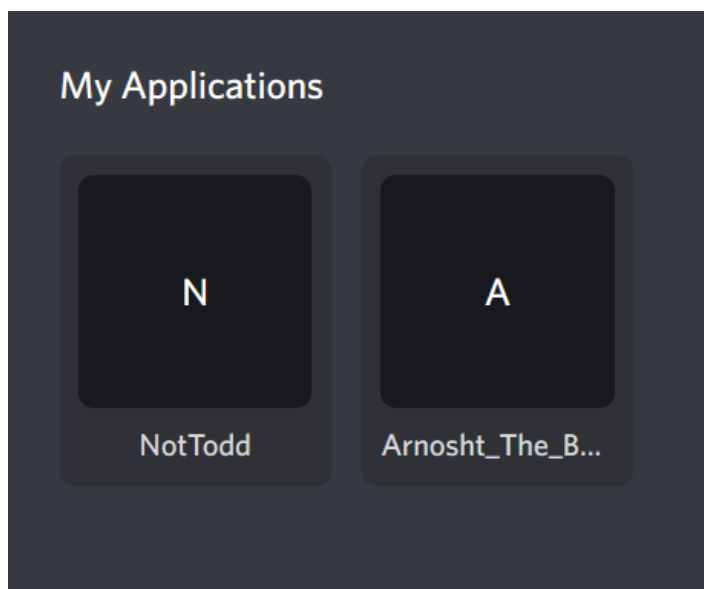
5.3 Vytvoření aplikace a přidání účtu typu bot

K vytvoření bota je nejdříve nutné vytvořit si aplikaci na stránkách platformy Discord v sekci „developer“. Následně je potřeba ve vytvořené aplikaci přidat účet typu bot.



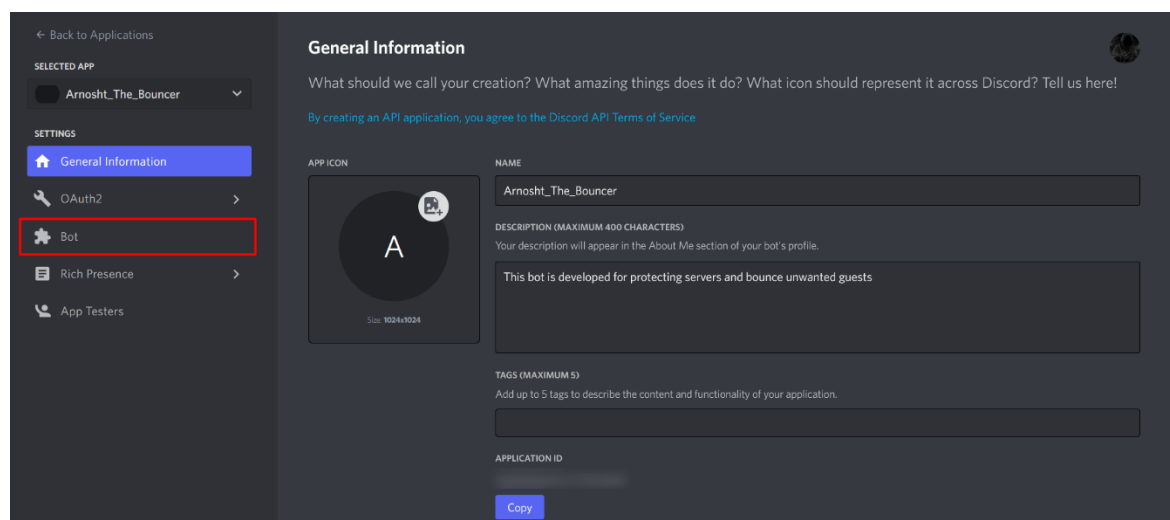
Obrázek 19 – vytvoření nové Discord API aplikace

Po vyplnění jména aplikace a potvrzení tlačítkem „Create“ zvýrazněným na obr. 19, dojde k vytvoření nové aplikace.



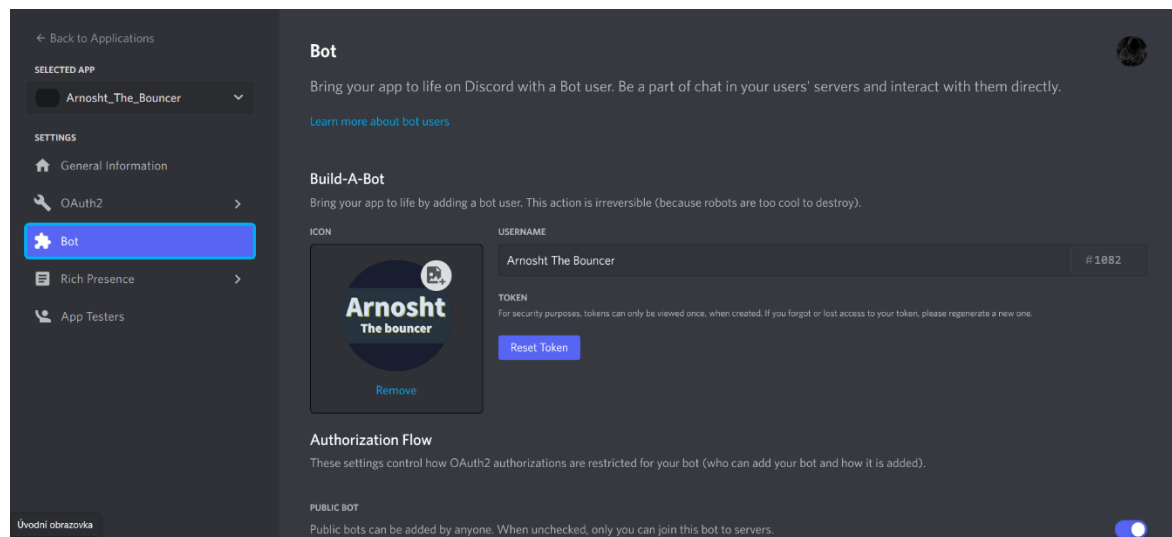
Obrázek 20 – seznam vytvořených aplikací

Na obr. 20 je vidět aplikace se jménem „NotTodd“. Jedná se o původní prototyp moderačního bota navrženého v roce 2020 používající příkazovou strukturu „legacy“ a knihovnu Discord.js verze 11, která se stala s příchodem aktuální verze 13 neboli verze, na které pracuje moderační bot „Arnosht“. Aktualizace knihoven byla důvodem k vytvoření nové aplikace od začátku, jelikož se při přechodu z verze 11 na verzi 12 změnily principy fungování aplikace, změnil se zápis funkcí a verze 12 a později 13 přinesla nové vylepšené funkce, které byly využity při vývoji moderačního bota s názvem „Arnosht“



Obrázek 21 – detail aplikace „Arnosht_The_Bouncer“

Na obr. 21 je vidět detail vytvořené aplikace. Je potřeba přidat bot účet pro vytvoření bota a přidání na Discord server. To se provádí v sekci „bot“ vyznačené červeně na obr. 21



Obrázek 22 – detail bot účtu aplikace „Arnosht_The_Bouncer“

Na obr. 22 je vidět vytvořený bot účet, kde je možné přidat profilový obrázek, jméno bot účtu a zjistit přístupový token, který je použit v souboru „index.js“ pro propojení serveru NodeJs a účtu bot.

5.4 Spouštěč příkazů

Jedná se o strukturu, jejíž kód se nachází v souboru `src/functions/handleCommands.js`. V první části struktura získá všechny soubory s příponou „.js“ a následně vytvoří pole obsahující názvy všech příkazů, jejichž soubory byly nazeleny. V druhé části se zavolá REST API (viz kapitola 2.2.1.1), které je pole příkazů předáno a následně odesláno na servery discord, kde se provede aktualizace příkazů ve všech instancích bota napříč servery.

```
(async () => {  
  try {  
    console.log(table.toString());  
    console.log('Slash commands refresh started.');  
    await rest.put(  
      Routes.applicationCommands(clientId),  
      {  
        body: client.commandArray  
      },  
    );  
  
    console.log('Slash commands refreshed successfully.');  } catch (error) {  
    console.error(error);  
  }  
})();
```

Obrázek 23 – ukázka volání REST API a aktualizace příkazů

Při spuštění bota dojde k vypsání tabulky do konzole obsahující seznam příkazů, které struktura načetla a které budou odeslány k aktualizaci.

5.5 Struktura příkazu z hlediska kódu

```
1 const { SlashCommandBuilder } = require('@discordjs/builders');
2
3 module.exports = {
4   permission: "ADMINISTRATOR", //BAN_MEMBERS, SEND_MESSAGES, MANAGE_MESSAGES...
5   role: false, // true
6   cooldown: "15s", //12h, 10s, 15h...
7   data: new SlashCommandBuilder()
8     .setName('testcommand')
9     .setDescription('This is a test command'),
10   async execute(interaction){
11     interaction.followUp({
12       content: "this is a response",
13       ephemeral: true
14     })
15   }
16 }
```

The image shows a code editor with line numbers 1 to 26. The code is a JavaScript object representing a Discord slash command. It is annotated with seven numbered boxes: Box 1 (red) highlights the import of `SlashCommandBuilder` on line 1. Box 2 (pink) highlights the `permission` property on line 4. Box 3 (green) highlights the `role` property on line 5. Box 4 (orange) highlights the `cooldown` property on line 6. Box 5 (dark green) highlights the `data` property and its configuration on lines 7-9. Box 6 (purple) highlights the `execute` method on line 10. Box 7 (blue) highlights the `followUp` call inside the `execute` method on lines 11-13.

Obrázek 24 – popis zdrojového kódu příkazu

Obr 24 je rozdělen do sekcí. V sekci 1 je vytvořena konstanta na strukturu obsaženou v knihovně `Discord.js` která pomáhá na základě předaných parametrů generovat objekty příkazů, místo ručního vytváření zmíněných objektů. V sekci 2 se nachází parametr pro spouštěč příkazů, který udává název oprávnění, které musí uživatel mít ke spuštění příkazu, v tomto případě se jedná o nejvyšší možné. Sekce 3 zobrazuje taktéž parametr pro spouštěč příkazů a udává, zda je k použití příkazu nutná moderátorská role. V sekci 4 je zvolen čas, po který nebude moct uživatel příkaz opakovat. Jedná se taktéž o parametr spouštěče příkazů. Parametry v sekci 2-4 jsou nepovinné a při jejich neuvedení bude příkaz fungovat v základním nastavení, tedy bez kontroly oprávnění, bez nutnosti role a s libovolným opakováním příkazu po sobě. V sekci 5 je vytvářen objekt příkazu, kterému je potřeba dát jméno (řádek 12 obr. 24), kterým se pak příklad spouští, v případě obr. 24 by se jednalo o „/testcommand“. Popis příkazu (řádek 13 obr. 24) je popis, které se v seznamu příkazu zobrazuje pod příkazem. V sekci 6 se nachází metoda „execute“, která obsahuje kód provádějící se po spuštění příkazu. Sekce 7 zobrazuje kód, který se provede, tedy odeslání objektu s odpovědí obsahující řetězec a proměnnou uvádějící, zda odpověď uvidí všichni nebo jen uživatel.

5.6 Automatická kontrola uživatelů a pozdravy

Pokud dojde k připojení nového uživatele k serveru, bot zjistí uživatelské Discord ID, které je v objektu nového uživatele, následně provede dotaz na databázi, která vrátí objekt obsahující všechny záznamy, kde se Discord ID záznamu shoduje s uživatelským.

```
//warning
banSchema.find({
  bannedMemberID: member.user.id
}, (banErr, banrecords) => {
```

Obrázek 25 – dotaz na databázi zablokovaných uživatelů

Pokud je proměnná „banrecords“ prázdná, nebyl nalezen žádný záznam a bot přejde rovnou k pozdravu.

Pokud uživatel záznam má, dojde ke spočítání všech záznamů a zjištění všech unikátních uživatelských jmen.

```
var altnames = [];

banrecords.forEach(rec => {
  if (!altnames.includes(rec.bannedMemberName)) {
    altnames.push(rec.bannedMemberName);
  }
})
```

Obrázek 26 – získání alternativních jmen uživatele

Ve výsledném varování jsou pak vidět všechny přezdívky, pod kterými byl uživatel zablokován. Následně dojde k sestavení varování pomocí struktury, kterou platforma Discord využívá pro zobrazování oznámení, zvanou „embed“.

```
const embed = new MessageEmbed()
  .setColor('RED')
  .setTitle('WARNING - user with record joined')
  .setDescription(`User that just joined has a database record`)
  .addFields({
    name: "User",
    value: `${member}`
  })
  .addFields({
    name: "Server",
    value: `${member.guild.name}`
  })
  .addFields({
    name: "Known alternate names",
    value: `${altnames || "No alternate names"}`
  })
  .setFooter({
    text: `You got this message because you have a role ${botconfig.roleName}`
  })
```

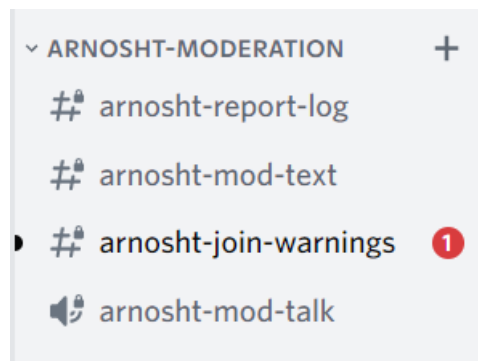
Obrázek 27 – sestavení struktury obsahující varování o uživateli se záznamem

Následně se vytvořené varování odešle do textové místnosti s názvem „arnosht-join-warnings“

```
member.guild.channels.cache.find(ch => ch.name === "arnosht-join-warnings").send({
  content: `${botRole}`,
  embeds: [embed]
});
```

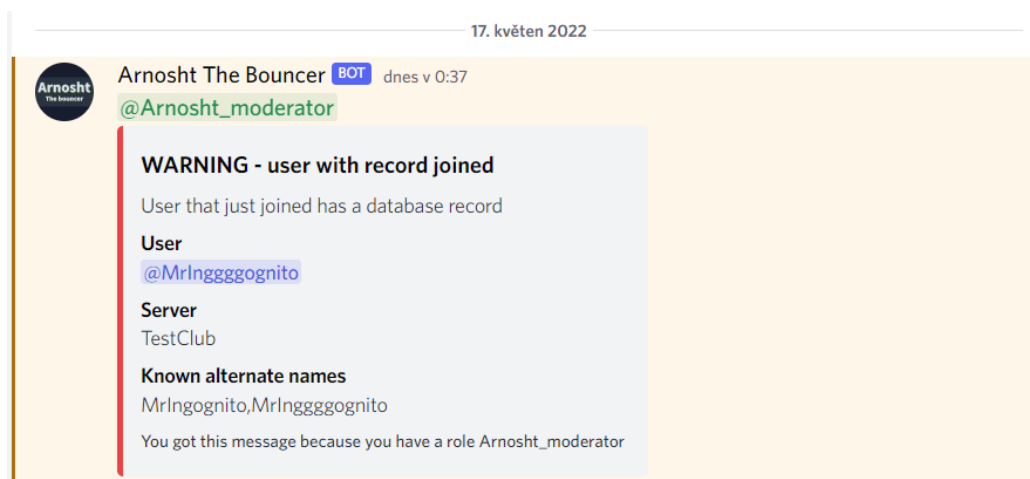
Obrázek 28 – odeslání varování do textové místnosti

Výsledné varování pak vypadá následovně:



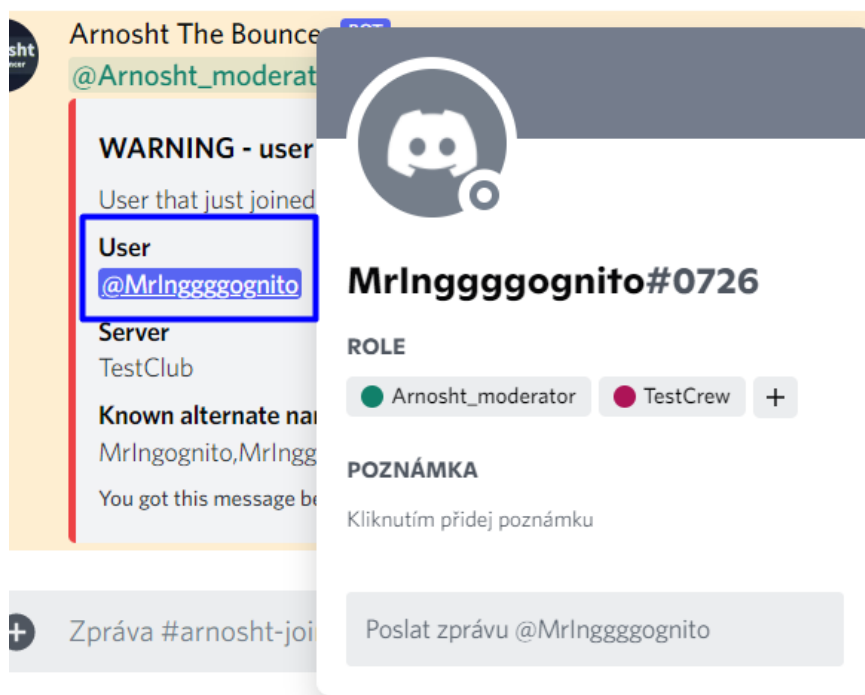
Obrázek 29 – zobrazení upozornění v seznamu místností

Na obr. 29 je vidět označení moderátorské role, tudíž každý moderátor dostane upozornění, že byli v místnosti označeni.



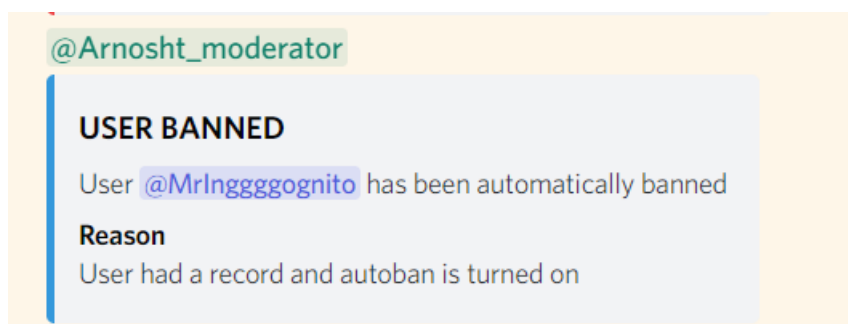
Obrázek 30 – varování o existenci záznamu nového uživatele

Obr. 30 zobrazuje zprávu, kterou bot odešle v případě, že našel záznam u uživatele, který se právě připojil. Zpráva začíná označením role moderátorů, aby každý uživatel disponující zmíněnou rolí dostal upozornění. Následně je odeslána struktura „embed“ obsahující informace o varování, o uživateli, který se připojil, a to s možností prokliku přímo na daného uživatele a jeho profil, server, na které se uživatel připojil a alternativní přezdívky, se kterými byl uživatel zablokovan na jiných serverech.



Obrázek 31 – zobrazení profilu uživatele z varování

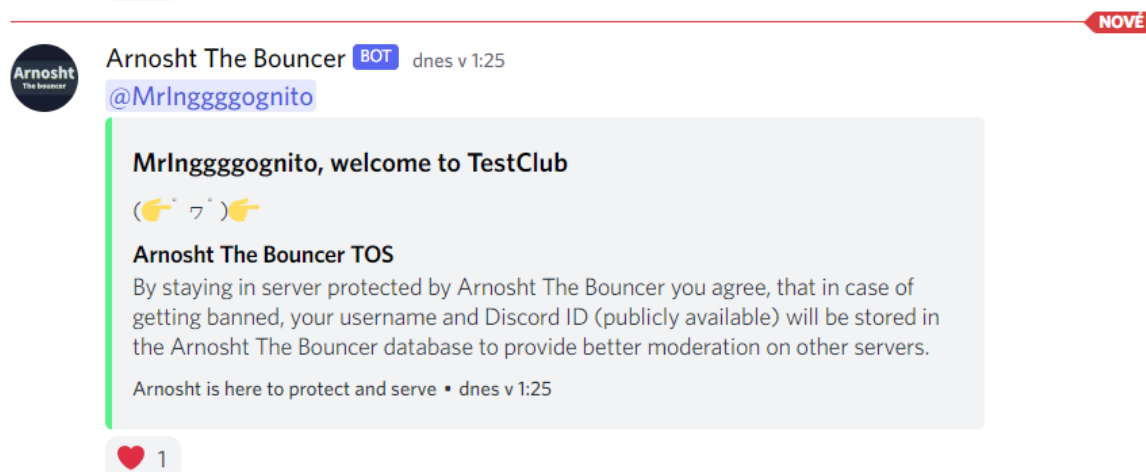
Pokud je nastavena možnost automatického blokování, za varováním je odeslána informace o zablokování uživatele.



Obrázek 32 – oznámení o automatickém zablokování uživatele

Pokud je nastavena možnost varování pomocí soukromých zpráv, identická varování se odešlou kromě specializovaných místností také všem moderátorům do soukromé zprávy.

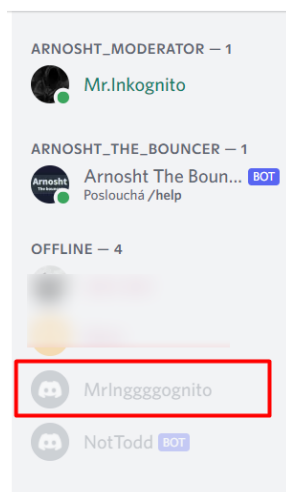
Pokud nedojde k zablokování uživatele, a majitel serveru nastavil místnost pro odesílání pozdravů, bot pozdraví nově příchodícího uživatele.



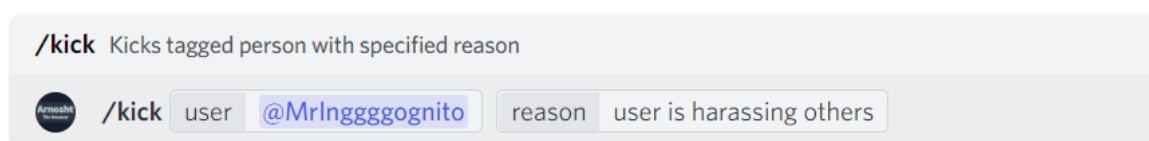
Obrázek 33 – pozdravení nového uživatele a seznámení s botem

5.7 Příkaz pro odebrání uživatele ze serveru

Pro použití příkazu je třeba zadat dva parametry, a to konkrétně vybrat uživatele ze seznamu připojených uživatelů a důvod, proč bude daná osoba odebrána. Celá událost se zapíše do protokolu událostí serveru, ve kterém ji může majitel a moderátoři snadno dohledat i s důvodem.

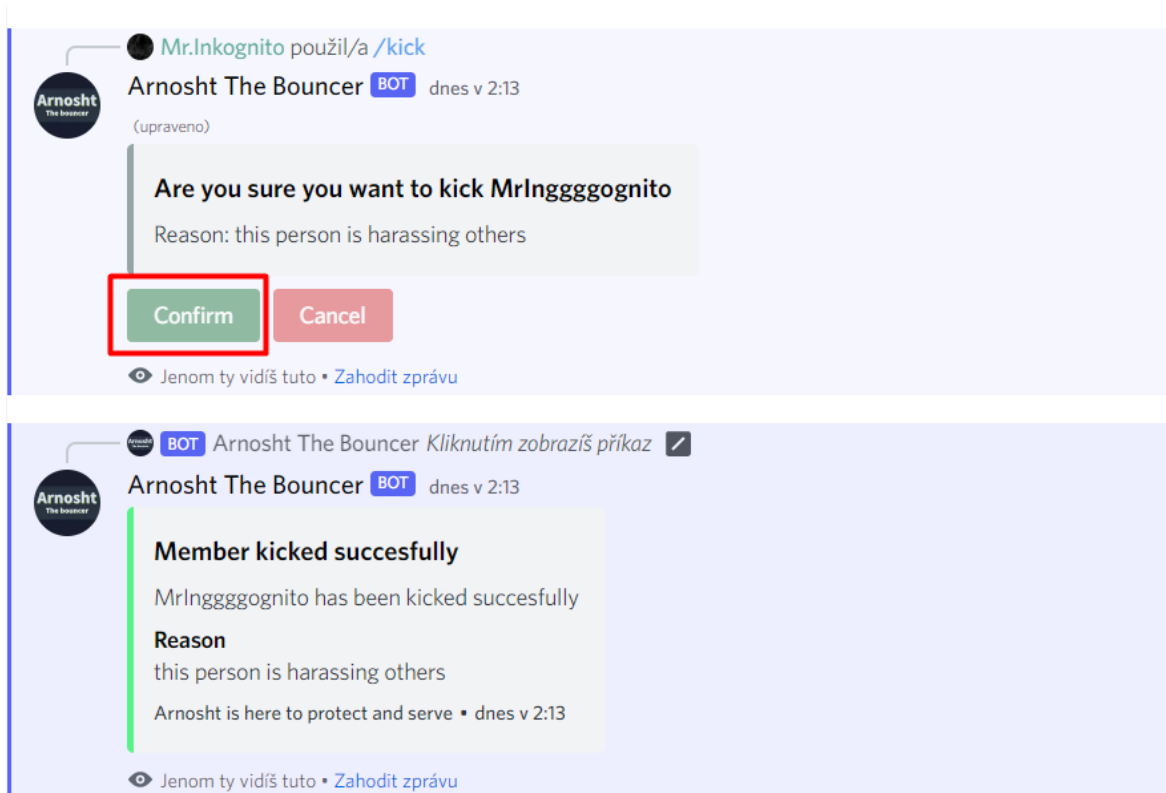


Obrázek 34 – seznam členů před odebráním uživatele



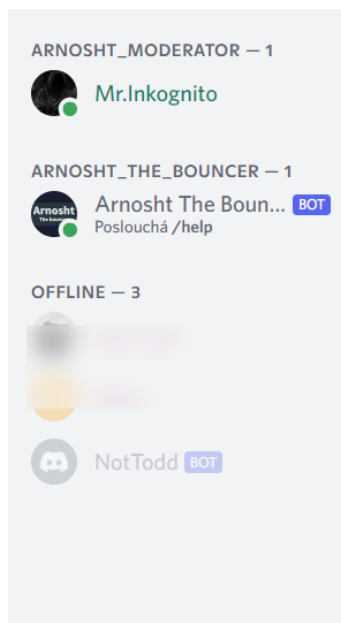
Obrázek 35 – použití příkazu /kick

Po odeslání příkazu dojde k zobrazení potvrzovacího dialogu.



Obrázek 36 – potvrzení odebrání uživatele

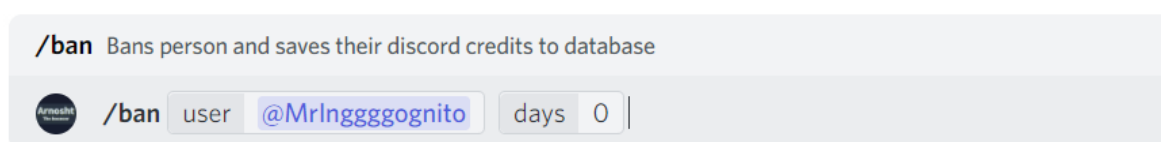
Obr. 36 zobrazuje možnost, kdy bylo odebrání potvrzeno. Následuje informace o odebrání uživatele a zamknutí potvrzovacích tlačítek, aby se zamezilo dalším reakcím na neplatnou událost.



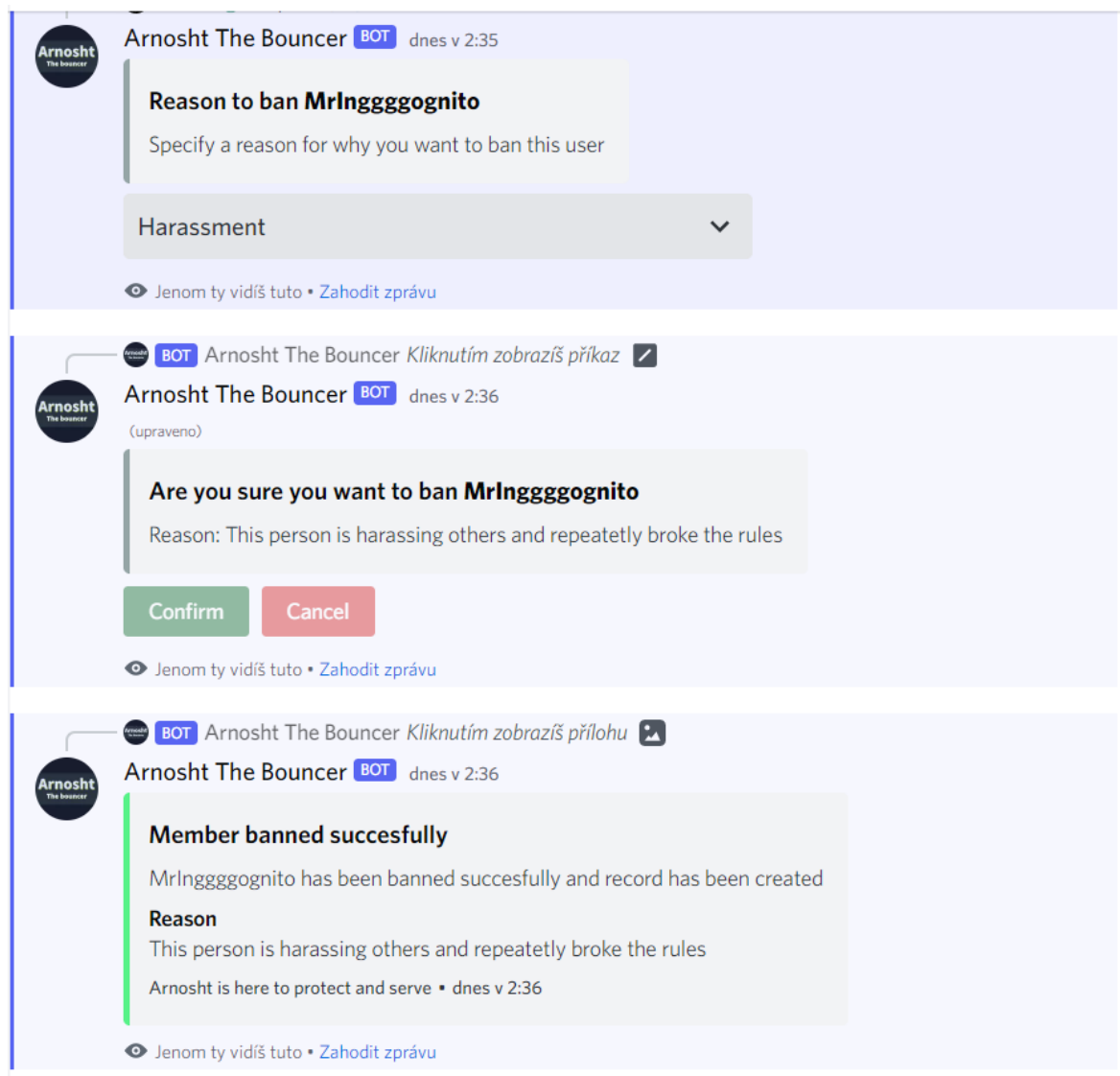
Obrázek 37 – seznam serveru po odebrání uživatele

5.8 Příkaz pro zablokování uživatele

Pro odeslání příkazu je potřeba zadat parametr cíleného uživatele k zablokování a čísla udávajícího počet dní za které se mají smazat zprávy, které poslal zablokovaný uživatel. Následně bot zobrazí rozbalovací menu pro udání důvodu zablokování z předem definovaného seznamu.



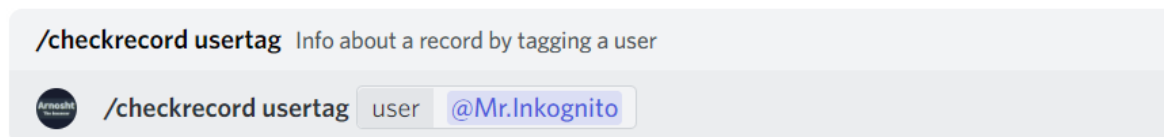
Obrázek 38 – zablokuje uživatele a nesmaže žádné jeho zprávy



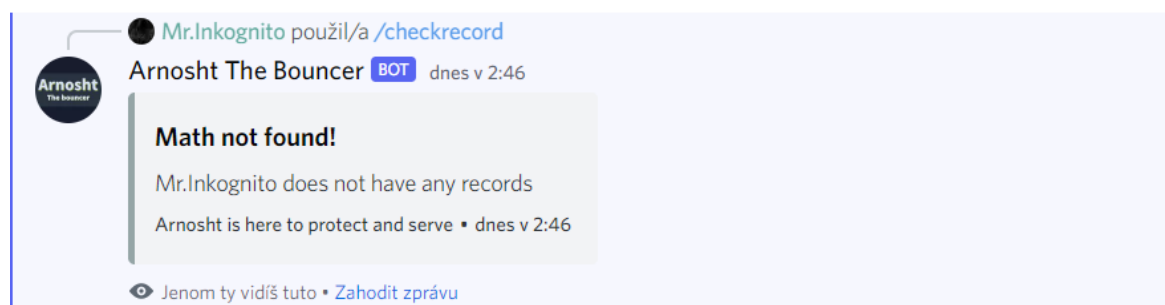
Obrázek 39 – provedení příkazu zablokování

5.9 Příkaz prověření uživatele

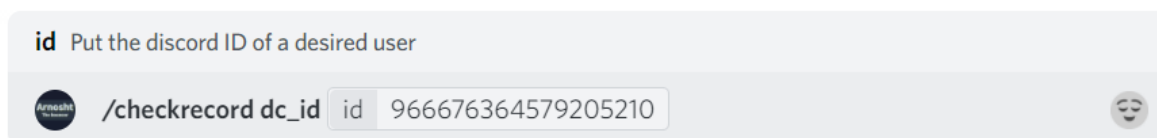
Příkaz má dvě varianty použití. První varianta vyhledává za pomoci vybraného uživatele ze seznamu a druhá vyhledává pomocí Discord ID, unikátního identifikátoru účtu. Bot po odeslání příkazu prohledá databázi a zjistí, zdali má vyhledávaná osoba záznam, pokud ano, zobrazí jméno, na které je záznam veden, celkový počet zablokování vedených k uživateli a všechny alternativní přezdívky, které uživatel používal v době zablokování. Pokud vyhledávaná osoba záznam nemá, bot odešle oznámení o neexistenci záznamu pro daného uživatele.



Obrázek 40 – prověřování uživatele vybráním ze seznamu uživatelů



Obrázek 41 – při prověřování uživatele nenalezen záznam



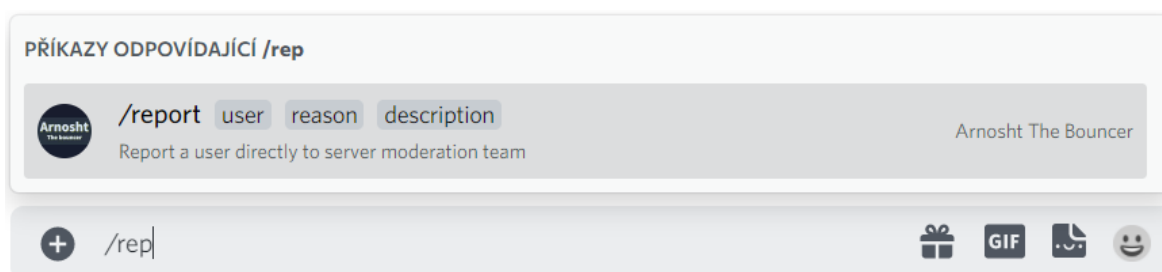
Obrázek 42 - prověřování uživatele pomocí Discord ID



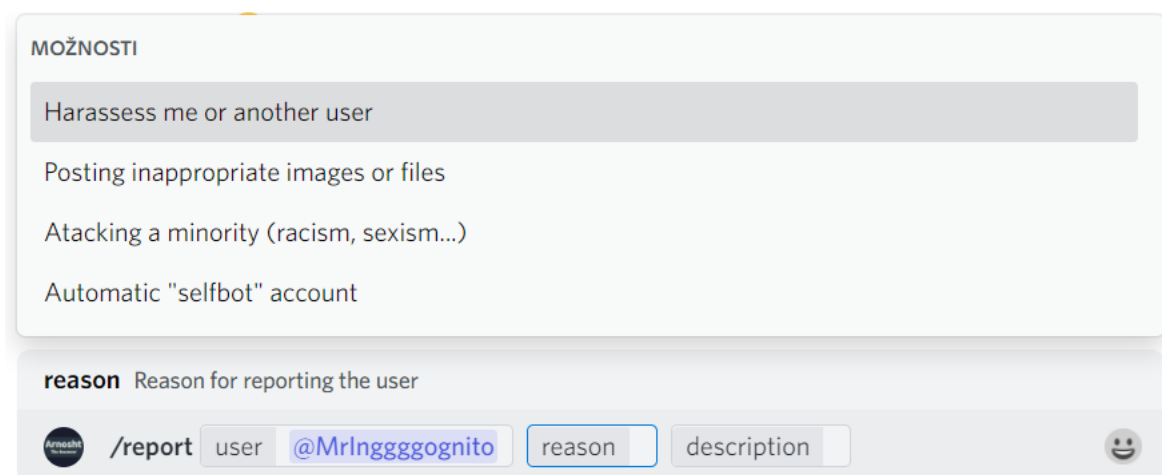
Obrázek 43 – uživatel má záznam v databázi

5.10 Příkaz pro nahlášení uživatelů

Příkaz dává uživateli rychlou a jednoduchou možnost, jak kontaktovat moderátorský tým, bez nutnosti soukromé zprávy jednomu nebo více členům nebo zprávy do veřejné místnosti a označení role moderátorského týmu. Příkazu je potřeba přidat parametr uživatele, který se provádí vybráním ze seznamu uživatelů na serveru, důvodu, kde je třeba vybrat jeden z předem definovaných důvodů a popis, obsahující detailní popis situace a důvodu, proč je označený uživatel nahlašován. Následně proběhne rekapitulace a potvrzení nahlašovací zprávy a po potvrzení autorem příkazu je nahlášení odesláno do specializované místnosti vytvořené botem.



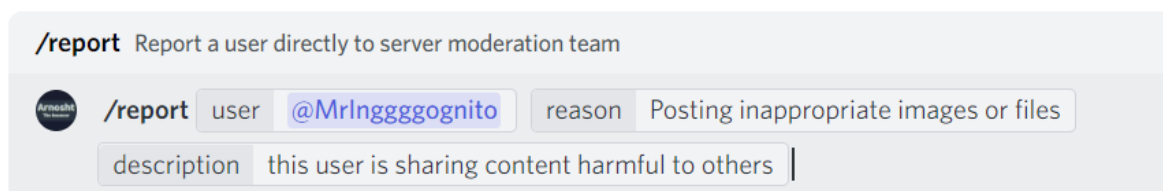
Obrázek 44 – nápověda k příkazu /report



Obrázek 45 – seznam důvodů nahlášení

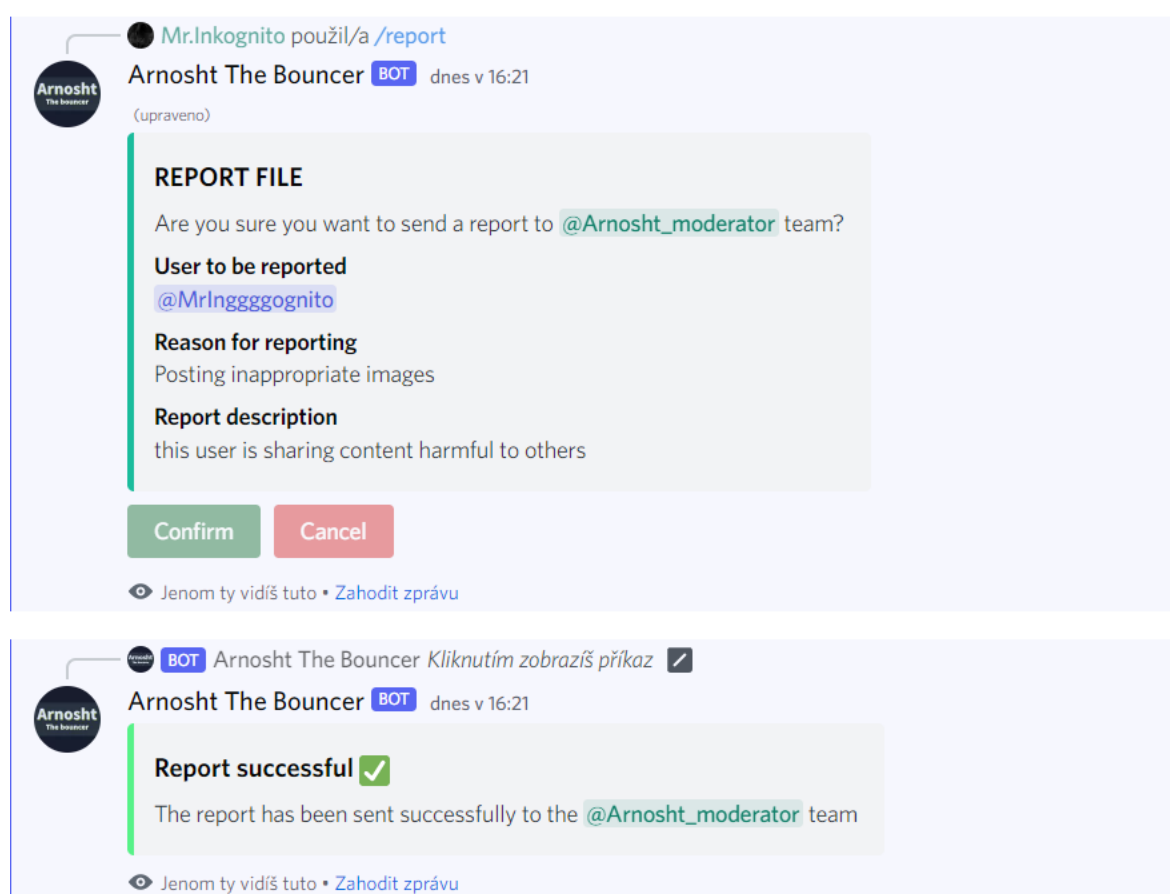
Na obr. 44 je vidět celá nápověda k příkazu, zobrazující všechny povinné parametry a krátký popis příkazu. Obr. 45 zobrazuje seznam, který je možno vložit do parametru s názvem „reason“, neboli důvod. Předem definované možnosti jsou zde z důvodu ucelenosti a přehlednosti daných nahlašovacích zpráv. Parametr popis neboli „description“ dává uživateli

nahlašujícímu jiného prostor vyjádřit se a upřesnit důvod nahlášení, který bude mít vliv na následné zpracování moderátorem.



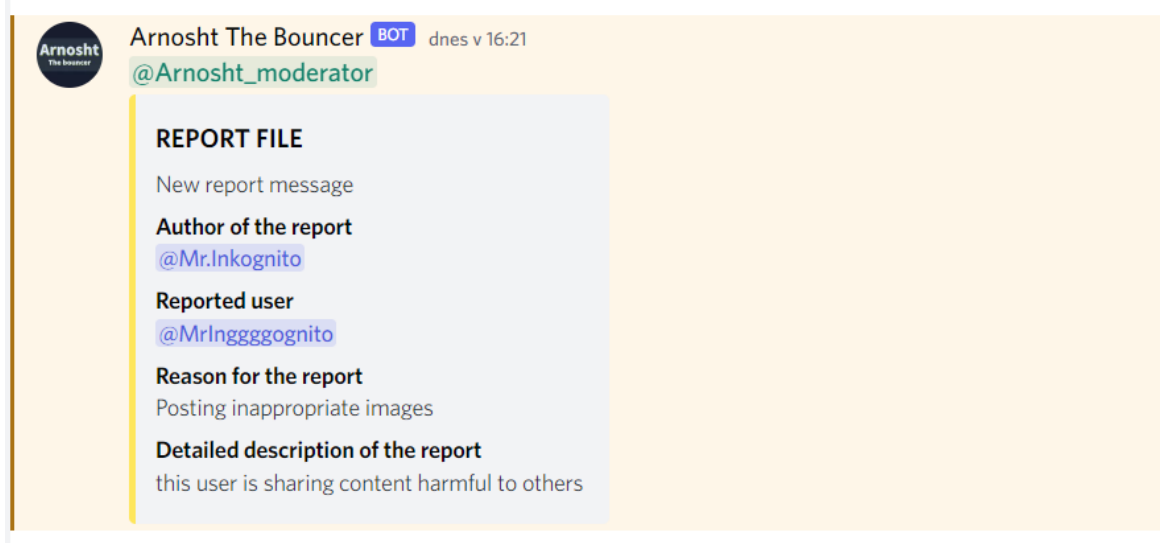
The screenshot shows a chat window with a header that says "/report Report a user directly to server moderation team". Below this, there is a form with three input fields: "user" containing "@MrIngggognito", "reason" containing "Posting inappropriate images or files", and "description" containing "this user is sharing content harmful to others". The form is ready to be submitted.

Obrázek 46 – hotový příkaz /report připravený k odeslání



Obrázek 47 – rekapitulace příkazu /report a následné potvrzení odeslání

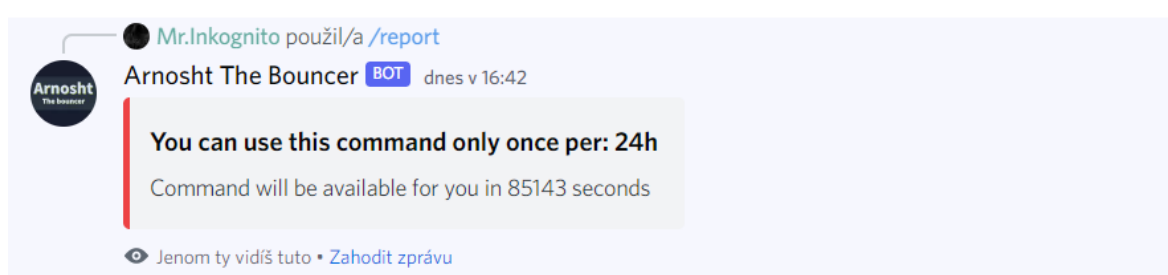
Obr. 47 popisuje rekapitulaci, která se uživateli zobrazí po odeslání příkazu zobrazeného na obr. 46. Souhlasí-li uživatel, klikne na zelené tlačítko „potvrdit“ a bot oznámí, že bylo nahlášení odesláno úspěšně.



Obrázek 48 – záznam nahlášení, který přišel moderátorům

Obr. 48 zobrazuje označení role moderátorů ve speciální místnosti „arnosht-report-log“ a následně zobrazí nahlašovací zprávu, obsahující uživatele, který ji odeslal, uživatele, kterého se týká, důvod, za jakým bylo nahlášení odesláno a detailní popis, proč k tomu došlo.

Aby se předešlo zneužívání funkce a časnému opakovanému nahlášení je v příkazu implementována speciální kontrola nazvaná „anti-spam cooldown“. Tato funkce zajistí, aby každý uživatel mohl zavolat příkaz pouze jednou za den, pokud se pokusí odeslat druhé nahlášení, objeví se chybová hláška, viz obrázek níže.

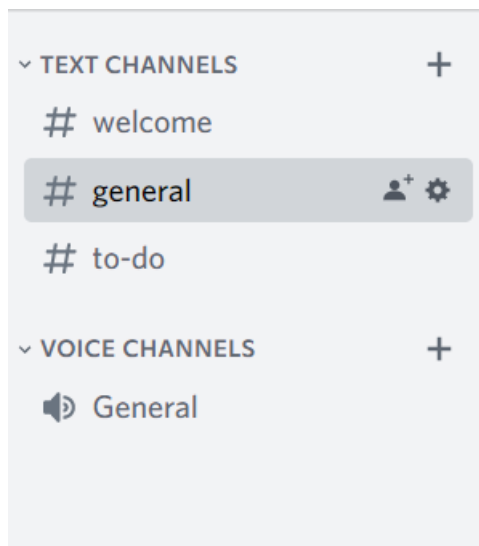


Obrázek 49 – informace o nutnosti vyčkat před opětovným použitím příkazu

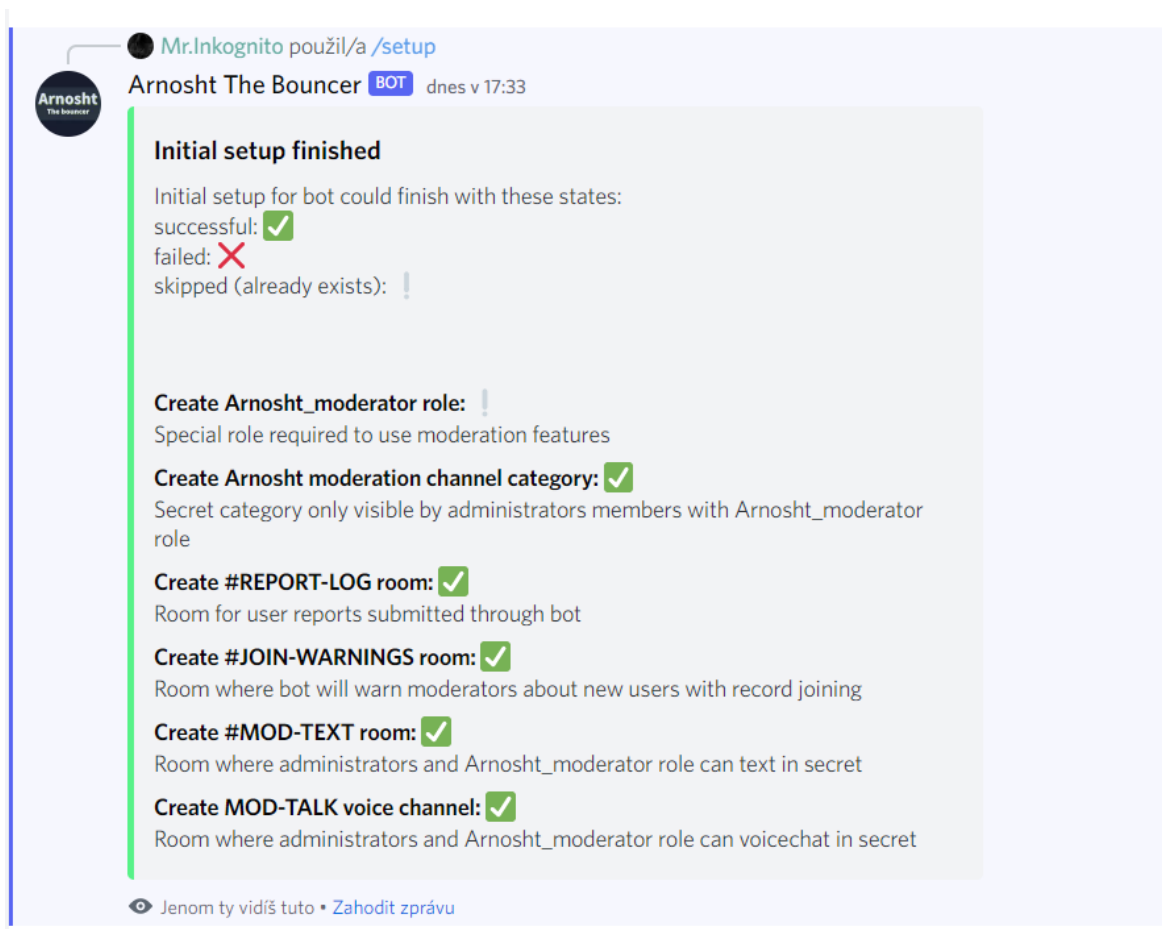
Kontrola funkce se provádí přímo ve spouštěči příkazů, takže je možno funkci „anti-spam cooldown“ použít na jakýkoliv příkaz.

5.11 Příkaz pro prvotní nastavení

Po spuštění příkazu „/setup“ s možností spuštění „/setup initial“ se provede veškeré základní nastavení nutné pro správnou funkci bota.

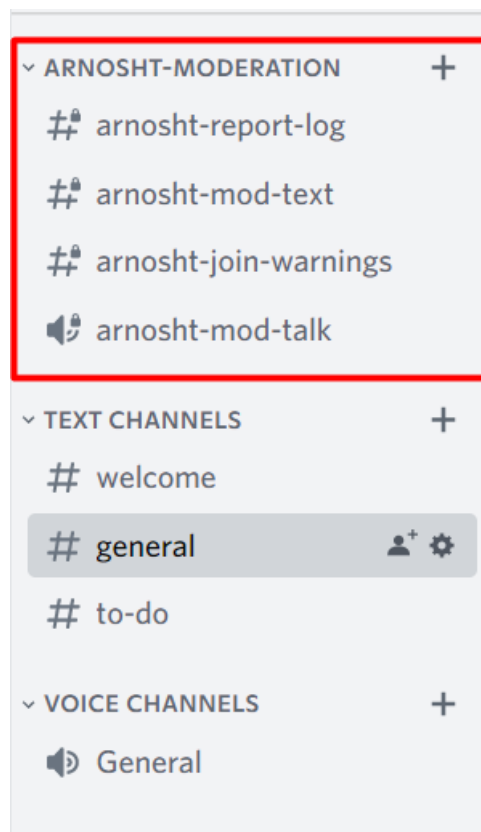


Obrázek 50 – server před spuštěním „/setup initial“



Obrázek 51 – výsledek příkazu „/setup initial“

Obr. 51 zobrazuje souhrn akcí, které příkaz provedl a jejich stav. V případě obr. 15, bot přeskočil vytváření role moderátora, jelikož zjistil její přítomnost na serveru. Jelikož nezjistil přítomnost žádné ze speciálních místností nutných ke správnému fungování viz obr. 50, vytvořil dané místnosti a nastavil přístupová oprávnění pouze pro moderátorskou roli. Z pohledu běžného uživatele se tedy server nezmění a zůstává jako na obr. 50. Z pohledu moderátora byly přidány nové místnosti, viz obrázky níže.



Obrázek 52 – vytvořené místnosti pro správnou funkci bota

Pomocí další možnosti spuštění příkazu „/setup“, tedy „/setup features“, který umožňuje majiteli serveru povolit a zakazovat dodatečné funkce jako automatické blokování uživatel se záznamem po připojení na server nebo varování moderátorů pomocí soukromé zprávy.

5.12 Další příkazy

Mezi další příkazy, které neplní moderační účel a jsou obsaženy pouze pro zkvalitnění uživatelské interakce jsou následující:

- `/help` – příkaz zobrazující uživateli seznam všech příkazů které bot obsahuje a jejich detailní popis. Obsahuje také upozornění pro nutnost prvotního nastavení.
- `/tos` – příkaz seznámí uživatele se svou existencí na serveru a vysvětlí, k čemu dojde, je-li uživatel zablokován pomocí bota.
- `/welcomechannel` – příkaz obsahuje tři hlavní možnosti spuštění. Možnost `info`, která zobrazí kanál, jež je nastaven pro odesílání pozdravů botem nebo jeho případnou neexistenci a nutnosti nastavení. Možnost `„set“` umožní majiteli serveru nastavit kanál pro pozdravy a možnost `„uset“` odstraní kanál z nastavení bota.

6 TESTOVÁNÍ BOTA

Testování bota probíhalo v prvotních fázích na soukromém testovacím serveru za pomoci vytvořeného alternativního účtu platformy Discord. Následně po odladění přešel na dva menší soukromé servery, jejichž majitelé souhlasili se zavedením experimentálního bota na jejich server. V současné době se bot nenachází na žádném veřejném komunitním serveru.

Testování na soukromém testovacím serveru probíhalo ve verzi, kdy byl bot spuštěn na lokálním prostředí a struktura příkazů byla navržena tak, aby se aktualizovala pouze pro jeden testovací server, a to za pomoci specifikování Discord ID serveru, na kterém se mají příkazy aktualizovat a Discord ID bota, jehož příkazy se mají aktualizovat.

```
const clientId = '960898693177942097';  
const guildId = '794654979284926494';  
...
```

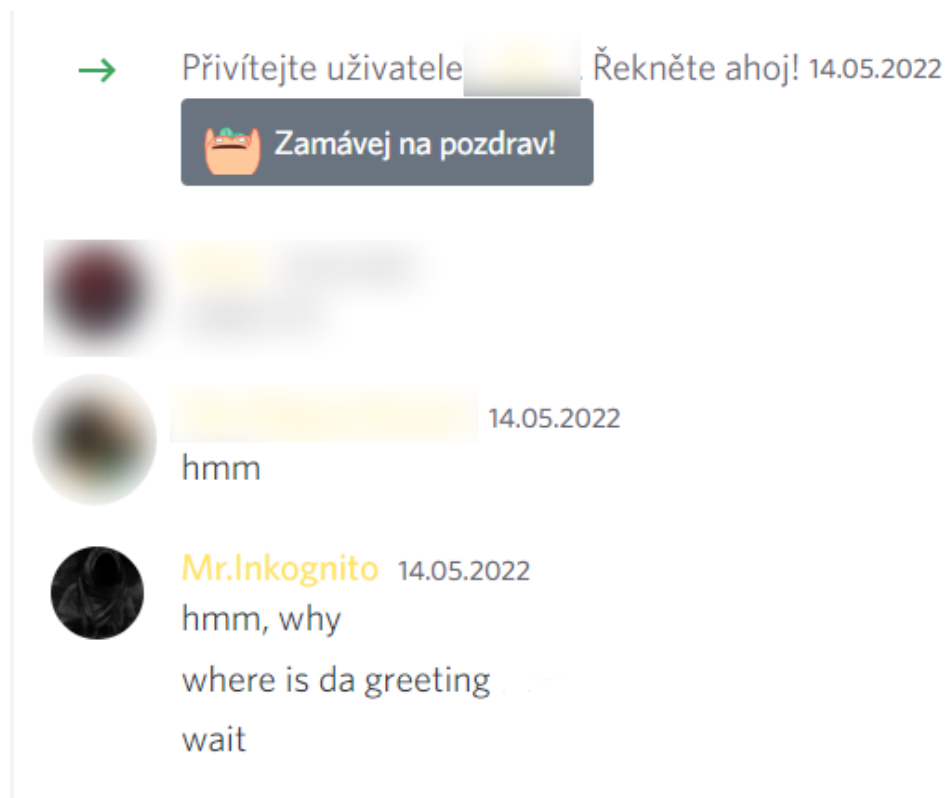
Obrázek 53 – vytvoření proměnných pro aktualizaci příkazů

```
await rest.put(  
  Routes.applicationGuildCommands(clientId,guildId),  
  {  
    body: client.commandArray  
  },  
);
```

Obrázek 54 – ukázka aktualizace příkazů pro jeden server

Testování díky tomu probíhalo rychleji, jelikož aktualizace globálních příkazů na všech serverech může trvat až hodinu.

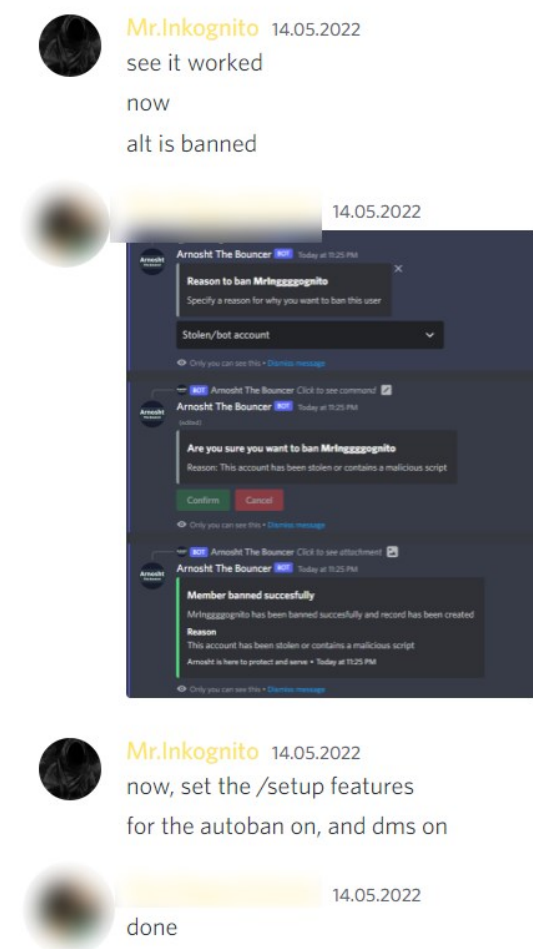
Při testování na domluveném soukromém zahraničním server bylo odhaleno několik drobných chyb a jedna velká. Velká chyba spočívala v tom, že v jistých případech bot uživatele po připojení nepozdravil, i když měl nastavený kanál a funkci pro pozdravy zapnutou.



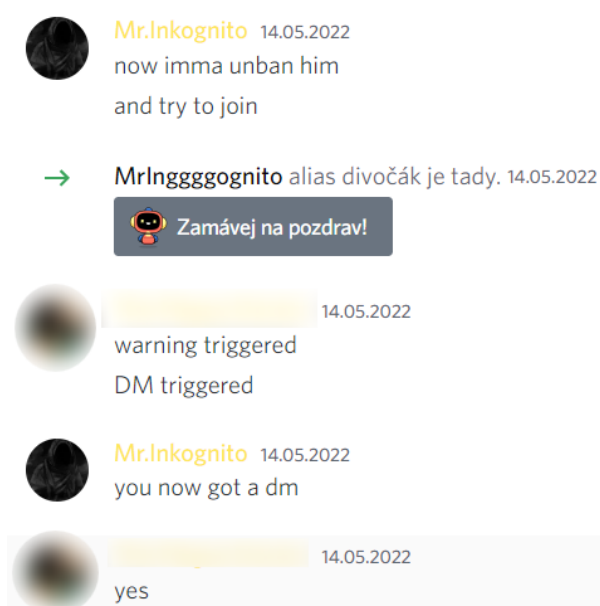
Obrázek 55 – ukázka z testování předem domluvenými testery

Chyba spočívala ve špatné struktuře kódu pro kontrolu nově připojených uživatelů. Pokud neměl uživatel záznam v databázi, bot jej nepozdravil, pokud ano, pozdravy fungovaly v pořádku. Tento problém nebylo možno odhalit na testovacím serveru, jelikož testovací alternativní účet vždy záznam měl. Chyba byla úspěšně odstraněna a funkce pro pozdravy funguje již v pořádku viz kapitola 5.6 obrázek 33.

Další hlavní část testování spočívala v přehlednosti a intuitivnosti příkazů a nastavení.



Obrázek 56 – testování příkazu /ban



Obrázek 57 – testování soukromé varovné zprávy

Funkcionalita upozorňování do specializované místnosti a soukromých zpráv, otestována a prohlášena za funkční, je detailně popsána v kapitole 5.6, která se zabývá detailním popisem funkcionality kontroly uživatelů.

Po řádném testování funkcí se provedla korekce odpovědí bota, aby byly gramaticky správně. Korekci prováděli anglicky mluvící testéři včetně jednoho, jehož hlavním jazykem je právě angličtina.

```
guildMembersAdd.js:
166   reason: "This user was automatically banned by bot because they had record" - needs an 'a' between had and record

ban.js:
193   .setTitle(`There has been an error while searching through database`) - between trough and database should be a 'the'
198   .setTitle(`There has been an error while saving to database`) - same here, between to and database should be a 'the'
```

Obrázek 58 – ukázka protokolu pro korekci zasláního testery

Po řádném testování a ověření, že všechny funkce pracují správně byl bot nasazen na server a všechny příkazy se aktualizovaly pro jakýkoliv server, na kterých se bot nacházel viz Obr. 23.

ZÁVĚR

Cílem práce bylo vytvořit bota pro asistenci moderátorům a uživatelům na jednotlivých serverech komunikační platformy Discord.

Teoretická část obsahuje seznámení s platformou Discord a jejími funkcemi. Dále se část zabývá problematikou samotné práce, tedy klasifikací kyberšikany na platformě a způsoby, jakými lze omezit. Také popisuje automatizované boty v kontextu s platformou Discord, jejich dovednosti, funkce a princip vývoje.

V praktické části je popsán návrh obsahující vlastnosti, které má bot umět k plnění své funkcionality a následná implementace, pro kterou byla zvolena knihovna Discord.js v běhovém prostředí NodeJs a jako databáze je použita MongoDB, která se řadí mezi NoSQL databáze, tedy databáze ukládající data do formátu JSON, místo klasických relačních tabulek.

Výstupem z práce je bot, schopný ukládat uživatele po jejich zablokování na serveru a varovat majitele a moderátory jiného server, pokud dojde k připojení daného uživatele. Dále je bot schopen přijímat nahlášení nevhodného chování a odesílat jej přímo moderátorskému týmu bez povšimnutí jiných uživatelů serveru. Bot je také schopen provést prvotní nastavení pro svou funkcionalitu, tedy vytvoření rolí a místností pro správnou funkci, a to pomocí jednoho příkazu. V intuitivnosti a ovladatelnosti nebyl testery nalezen žádný problém. Bot je v současnosti domluven na větší veřejný server, kde budou sbírány připomínky a návrhy k úpravám či novým funkcím. Uživatelé, kteří s botem pracují budou mít tak možnost přímo ovlivnit jeho další vývoj.

SEZNAM POUŽITÉ LITERATURY

- [1] What is discord?. In: *Discord.com* [online]. 2022 [cit. 2022-05-16]. Dostupné z: <https://discord.com/safety/360044149331-What-is-Discord>
- [2] Permissions. In: *Discord.com* [online]. 2022 [cit. 2022-05-16]. Dostupné z: <https://discord.com/developers/docs/topics/permissions>
- [3] Roles and Permissions. In: *Anidiots.guide* [online]. 2022 [cit. 2022-05-16]. Dostupné z: <https://anidiots.guide/understanding/roles/>
- [4] Role Management 101. In: *Discord.com* [online]. 2022 [cit. 2022-05-16]. Dostupné z: <https://support.discord.com/hc/en-us/articles/214836687-Role-Management-101>
- [5] Role of administrators and moderators on Discord. In: *Discord.com* [online]. 2022 [cit. 2022-05-16]. Dostupné z: <https://discord.com/safety/360044103531-Role-of-administrators-and-moderators-on-Discord>
- [6] MCCALL, Vivian. How to add a bot to Discord to help you run and organize your chatroom. In: *Businessinsider.com* [online]. 2020 [cit. 2022-05-16]. Dostupné z: <https://www.businessinsider.com/how-to-add-a-bot-to-discord>
- [7] SANTORA, Jacinda. 22 Discord Bots That Will Keep Your Server Hopping. In: *Influencermarketinghub.com* [online]. 2021 [cit. 2022-05-16]. Dostupné z: <https://influencermarketinghub.com/discord-bots/>
- [8] Discord.js VS discord.py. In: *Libhunt.com* [online]. 2022 [cit. 2022-05-16]. Dostupné z: <https://www.libhunt.com/compare-discord.js-vs-discord.py>
- [9] Permission to Slash, Granted: Introducing Slash Command Permissions. In: *Discord.com* [online]. 2022 [cit. 2022-05-16]. Dostupné z: <https://discord.com/blog/slash-commands-permissions-discord-apps-bots>
- [10] SCHMIDT, Randall. How to Make a Discord Bot: an Overview and Tutorial. In: *Toptal.com* [online]. c2010-2022 [cit. 2022-05-16]. Dostupné z: <https://www.toptal.com/chatbot/how-to-make-a-discord-bot>

- [11] CHIARELLI, Andrea. Community Resources. In: *Discord.com* [online]. 2022 [cit. 2022-05-16]. Dostupné z: <https://discord.com/developers/docs/topics/community-resources#libraries>
- [12] Rate Limits. In: *Discord.com* [online]. 2022 [cit. 2022-05-16]. Dostupné z: <https://discord.com/developers/docs/topics/rate-limits>
- [13] 10 Best JavaScript Discord API Libraries. In: *Openbase.com* [online]. 2022 [cit. 2022-05-16]. Dostupné z: <https://openbase.com/categories/js/best-javascript-discord-api-libraries>
- [14] Imagine a bot. In: *Discord.js* [online]. 2022 [cit. 2022-05-16]. Dostupné z: <https://discord.js.org/#/>
- [15] ŠKULTÉTY, Rastislav. *JavaScript: programujeme internetové aplikace*. Praha: Computer Press, 2001. Pro každého uživatele. ISBN 80-722-6457-5.
- [16] KOĐOUSKOVÁ, Barbora. JavaScript pro začátečníky: co to je a jak funguje. In: *Rascasone.com* [online]. 2022 [cit. 2022-05-16]. Dostupné z: <https://www.rascasone.com/cs/blog/co-je-javascript-pro-zacatecniky>
- [17] NGUYEN, Don, Ondřej BAŠE. *Node.js Okamžitě*. Brno: Computer Press, 2016. ISBN 978-802-5148-204.
- [18] KOĐOUSKOVÁ, Barbora. Proč k vývoji webových aplikací použít technologii NodeJS?. In: *Rascasone.com* [online]. 2021 [cit. 2022-05-16]. Dostupné z: <https://www.rascasone.com/cs/blog/node-js-architektura-moduly-npm>
- [19] What Is Bullying. In: *Stopbullying.gov* [online]. 2021 [cit. 2022-05-14]. Dostupné z: <https://www.stopbullying.gov/bullying/what-is-bullying>
- [20] BURÝŠKOVÁ, Lenka. Víte co je KYBERŠIKANA?. In: *Policie.cz* [online]. 2009 [cit. 2022-05-16]. Dostupné z: <https://www.policie.cz/clanek/vite-co-je-kybersikana.aspx>
- [21] Cyberbullying: What is it and how to stop it. In: *Unicef.org* [online]. 2022 [cit. 2022-05-14]. Dostupné z: <https://www.unicef.org/end-violence/how-to-stop-cyberbullying>

- [22] What is Doxing – Definition and Explanation. In: *Kaspersky.com* [online]. 2022 [cit. 2022-05-16]. Dostupné z: <https://www.kaspersky.com/resource-center/definitions/what-is-doxing>
- [23] KAUFMAN, Aaron a Brianna RAUENZAHN. Cyberbullying and the Limits of Free Speech. In: *Theregreview.org* [online]. 2020 [cit. 2022-05-14]. Dostupné z: <https://www.theregreview.org/2020/12/19/saturday-seminar-cyberbullying-free-speech-limits/>
- [24] HINDUJA, Sameer. Discord: A Chat App Not Just For Gamers. In: *Cyberbullying.org* [online]. 2018 [cit. 2022-05-16]. Dostupné z: <https://cyberbullying.org/discord-chat-app-gamers>
- [25] GRAYSON, Nathan. Discord Explains How It Handles Harassment, Doxxing, And Threatening Behavior. In: *Kotaku.com* [online]. [cit. 2022-05-16]. Dostupné z: <https://kotaku.com/discord-explains-how-it-handles-harassment-doxxing-an-1837220883>
- [26] How We're Fighting Spammers on Discord. In: *Discord.com* [online]. 2022 [cit. 2022-05-16]. Dostupné z: <https://discord.com/blog/how-discord-is-fighting-spam>
- [27] CAOIMHEINNIT. This guy has repeatedly harassed me. In: *Twitter* [online]. 2021 [cit. 2022-05-14].
- [28] Introduction. In: *Discord.js Guide* [online]. 2021 [cit. 2022-05-17]. Dostupné z: <https://discordjs.guide/>

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

API application programming interface

NPM node package manager

REST representational state transfer

TOS terms of service

SEZNAM OBRÁZKŮ

Obrázek 1 – pokus o kontaktování uživatele, který si daný účet zablokoval	13
Obrázek 2 – ukázka spouštění příkazu typu „legacy“	14
Obrázek 3 – výpis příkazů po vložení lomítka do pole pro odesílání zpráv	16
Obrázek 4 – odeslaný příkaz a výsledek příkazu jako odpověď na něj	16
Obrázek 5 – ukázka komunikace mezi botem a platformou pomocí API [10]	17
Obrázek 6 – ukázka zpracovávání požadavků prostředím NodeJs [18]	21
Obrázek 7 – nahlášení uživatelů od 1. ledna do 1. dubna 2019 [25]	25
Obrázek 8 – uživatel informující přes Twitter komunitu o závadném chování [27] ..	26
Obrázek 9 – návrhové schéma spouštěče příkazů	29
Obrázek 10 – návrhové schéma pro kontrolu uživatelů po připojení	30
Obrázek 11 – návrhové schéma příkazu pro odebrání uživatelů	31
Obrázek 12 – návrhové schéma příkazu pro blokování uživatelů	32
Obrázek 13 – návrhové schéma příkazu prověřujícího uživatele	33
Obrázek 14 – návrhové schéma příkazu prvotního nastavení	34
Obrázek 15 – návrhové schéma pro nahlašovací příkaz	35
Obrázek 16 – souborová struktura bota	37
Obrázek 17 – obsah složky „src“ v souborech bota	38
Obrázek 18 – hlavní kód pro spuštění bota	39
Obrázek 19 – vytvoření nové Discord API aplikace	40
Obrázek 20 – seznam vytvořených aplikací	41
Obrázek 21 – detail aplikace „Arnosht_The_Bouncer“	41
Obrázek 22 – detail bot účtu aplikace „Arnosth_The_Bouncer“	42
Obrázek 23 – ukázka volání REST API a aktualizace příkazů	43
Obrázek 24 – popis zdrojového kódu příkazu	44
Obrázek 25 – dotaz na databázi zablokovaných uživatelů	45
Obrázek 26 – získání alternativních jmen uživatele	45
Obrázek 27 – sestavení struktury obsahující varování o uživateli se záznamem	46
Obrázek 28 – odeslání varování do textové místnosti	46
Obrázek 29 – zobrazení upozornění v seznamu místností	46
Obrázek 30 – varování o existenci záznamu nového uživatele	47
Obrázek 31 – zobrazení profilu uživatele z varování	48
Obrázek 32 – oznámení o automatickém zablokování uživatele	48

Obrázek 33 – pozdravení nového uživatele a seznámení s botem.....	49
Obrázek 34 – seznam členů před odebráním uživatele.....	49
Obrázek 35 – použití příkazu /kick.....	49
Obrázek 36 – potvrzení odebrání uživatele	50
Obrázek 37 – seznam serveru po odebrání uživatele.....	51
Obrázek 38 – zablokuje uživatele a nesmaže žádné jeho zprávy	51
Obrázek 39 – provedení příkazu zablokování	52
Obrázek 40 – prověřování uživatele vybráním ze seznamu uživatelů.....	53
Obrázek 41 – při prověřování uživatele nenalezen záznam	53
Obrázek 42 - prověřování uživatele pomocí Discord ID	53
Obrázek 43 – uživatel má záznam v databázi.....	53
Obrázek 44 – nápověda k příkazu /report	54
Obrázek 45 – seznam důvodů nahlášení.....	54
Obrázek 46 – hotový příkaz /report připravený k odeslání	55
Obrázek 47 – rekapitulace příkazu /report a následné potvrzení odeslání	55
Obrázek 48 – záznam nahlášení, který přišel moderátorům	56
Obrázek 49 – informace o nutnosti vyčkat před opětovným použitím příkazu.....	56
Obrázek 50 – server před spuštěním „/setup initial“	57
Obrázek 51 – výsledek příkazu „/setup initial“	57
Obrázek 52 – vytvořené místnosti pro správnou funkci bota	58
Obrázek 53 – vytvoření proměnných pro aktualizaci příkazů	60
Obrázek 54 – ukázka aktualizace příkazů pro jeden server.....	60
Obrázek 55 – ukázka z testování předem domluvenými testery	61
Obrázek 56 – testování příkazu /ban.....	62
Obrázek 57 – testování soukromé varovné zprávy	62
Obrázek 58 – ukázka protokolu pro korekci zasláního testery	63

SEZNAM TABULEK

Tabulka 1 – hlavní knihovny pro tvorbu botů [11].....	19
---	----

SEZNAM PŘÍLOH

PŘÍLOHA P I: Obsah CD

PŘÍLOHA P I: OBSAH CD

Struktura obsahu přiloženého CD:

- fulltext.pdf – Bakalářská práce v elektronické podobě
- Adresář *BP_Hresil* obsahující zdrojové soubory vytvářeného bota
 - Adresář *node_modules* obsahující noubory prostředí NodeJs a použitých modulů
 - Adresář *src* obsahující jednotlivé příkazy a hlavní spouštěcí soubor *index.js*
 - Soubory potřebné pro funkci bota