Katedra informatiky
Přírodovědecká fakulta
Univerzita Palackého v Olomouci

BAKALÁŘSKÁ PRÁCE

Bot na platformě Discord

2023

Lukáš Netřeba

Vedoucí práce:
Mgr. Roman Vyjídáček

Studijní program: Informatika, prezenční forma

Bibliografické údaje

Autor:	Lukáš Netřeba
Název práce:	Bot na platformě Discord
Typ práce:	bakalářská práce
Pracoviště:	Katedra informatiky, Přírodovědecká fakulta, Univerzita Palackého v Olomouci
Rok obhajoby:	2023
Studijní program:	Informatika, prezenční forma
Vedoucí práce:	Mgr. Roman Vyjídáček
Počet stran:	27
Přílohy:	1 CD/DVD
Jazyk práce:	český

Bibliographic info

Author:	Lukáš Netřeba
Title:	Bot on Discord platform
Thesis type:	bachelor thesis
Department:	Department of Computer Science, Faculty of Science, Palacký University Olomouc
Year of defense:	2023
Study program:	Computer Science, full-time form
Supervisor:	Mgr. Roman Vyjídáček
Page count:	27
Supplements:	1 CD/DVD
Thesis language:	Czech

Anotace

Obsahem práce je vytvoření Bota pro uživatele a komunity nacházející se na komunikační platformě Discord. Tato aplikace umožní bota používat pro správu komunit, nastavením jejich pravidel, pravomocí a moderace chatu. Bot dále umožní vytvářet a organizovat události, na které se uživatelé z komunit budou moci přihlásit a být upozorněni před jejich začátkem. Dále bude zprostředkovávat audio přehrávač skrze hlasové kanály na serverech.

Synopsis

Content of this theses aims to create a Bot for users and communities on Discord platform. Application will provide management for those communities with their own set of rules, permissions and chat moderation. The bot will take care of creating and organizing scheduled events for community users that can participate in and be notified just before the event starts. The bot will also mediate an audio player in voice channels on the servers.

Klíčová slova: Discord, nástroj pro správu, Python, Bot, audio přehrávač, události

Keywords: Discord, administration tool, Python, Bot, audio player, scheduled events

Chtěl bych poděkovat svému vedoucímu práce, panu Mgr. Romanu Vyjídáčkovi, za to, že mi umožnil zpracovat toto téma a za jeho odbornou pomoc při jejím vypracování. Dále bych chtěl poděkovat svým blízkým a kolegům, kteří byli se mnou trpěliví a motivovali mě k vypracování práce.

Místopřísežně prohlašuji, že jsem celou práci včetně příloh vypracoval/a samostatně a za použití pouze zdrojů citovaných v textu práce a uvedených v seznamu literatury.

datum odevzdání práce podpis autora

Obsah

1 Úvod **8**

2 Platforma Discord **8**
2.1 Definice a terminologie . . . 9
2.2 Servery a komunity . . . 10
2.3 Kanály hlasové a textové . . . 10
2.4 Uživatelé, role a pravomoce . . . 11
2.5 Limitace a předplatné . . . 11

3 Discord Bot **13**
3.1 Ovládání a práce s botem . . . 14
3.1.1 Prefixované zprávy . . . 14
3.1.2 Lomítkové příkazy . . . 15
3.2 Technologie pro vývoj a implementaci . . . 16
3.2.1 Discord API . . . 16
3.2.1.1 WebSocket API . . . 16
3.2.1.2 REST API . . . 17
3.2.2 Asyncio . . . 17
3.2.3 FFmpeg . . . 18
3.3 Knihovny pro práci s Discord API . . . 19
3.4 Knihovna discord.py . . . 19
3.5 Možnosti hostování aplikace . . . 20

4 Návrh funkcionalit **20**
4.1 Automatické vlastnosti . . . 20
4.1.1 Pravidla nově příchozím . . . 20
4.1.2 Přiřazení pravomocí nově příchozím . . . 20
4.1.3 Kontrola nově příchozích . . . 20
4.1.4 Kontrola odesílaných textových zpráv . . . 20
4.2 Manuální vlastnosti . . . 20
4.2.1 Správa uživatelů . . . 20
4.2.2 Správa událostí . . . 20
4.2.3 Přehrávání audio stop . . . 20

5 Implementace **21**
5.1 Souborová struktura . . . 21
5.2 Spouštěcí soubor . . . 21
5.3 Vytvoření aplikace a účtu jako bot . . . 21
5.4 Ukládání dat . . . 21
5.5 Spouštění příkazů . . . 21
5.6 Struktura příkazů v kódu . . . 21
5.7 Pravidla komunit . . . 21
5.8 Kontrola zpráv . . . 21

5.9 Kontrola uživatelů 21
5.10 Události a jejich spravování 21
5.10.1 Události integrované v Discordu 21
5.10.2 Události zapouzdřené ve zprávě 21
5.11 Přehrávač 21
5.12 Nahrání bota na hosting 21
5.13 Testování bota 21

Závěr **22**

Conclusions **23**

A Obsah přiloženého datového média **24**

Seznam zkratek **26**

Literatura **27**

Seznam obrázků

1 Systémový bot platformy Discord 14
2 Příkaz vyvolaný prefixovanou zprávou 15
3 Kontextové menu lomítkových příkazů 15
4 Způsob komunikace bota a platformy [7] 16
5 Asynchronní programování s jedním procesem [12] 18

Seznam tabulek

1 Limity pro nepředplacené uživatele a komunity 12
2 Úrovně limitů pro předplaticelské účty 12
3 Úrovně limitů pro předplacené servery 13
4 Knihovny pro práci s boty [11] 19

1 Úvod

Tato práce se zabývá vytvořením bota pomocí knihovny discord.py v programovacím jazyce Python pro komunikační platformu Discord, který bude sloužit jako nástroj pro komunity ke zlepšení správy nad danou komunitou, jejími pravidly, nastavením pravomocí pro nově připojené uživatele a jejich následnou kontrolou vůči prohřeškům z jiných komunit, kde bot mohl problémového uživatele již spatřit a případně tak zavčas varovat moderátory. Bot umožní vytvoření události s názvem, popiskem a datem konání a evidencí uživatelů přihlášených (odmítnutých, nerozhodných) na tuto událost a těsně před započetím události je upozorní. Dále zprostředkuje možnost posílat audio stopu z YouTube přímo do hlasových kanálů na serveru. V práci jsou také rozebráno jaké možnosti máme, kde můžeme takovou aplikaci nasadit, vlastnosti každé z možností a implementace jedné z možností.

V teoretické části jsou popsány požadavky k pochopení a vypracování práce. Tedy způsob, kterým platforma Discord funguje po uživatelské a technické stránce, funkcí vlastních aplikací v kontextu s touto platformou a technologiemi používanými při jejich vývoji.

Praktická část se zabývá efektivním návrhem a následnou implementací bota a jeho otestováním.

Výsledkem práce je funkční aplikace realizující bota, který může být nasazen pro jakoukoliv komunitu na Discordu.

2 Platforma Discord

Discord je VoIP a sociální komunikační platforma. Její uživatelé zde mají možnost komunikovat skrze audiohovor, videohovor, média, soubory a zprávy a to buď prostřednictvým soukromých chatů a nebo jako součástí komunit (veřejné, soukromé), kterým se také někdy říká „servery“ nebo „guildy“. Discord je multiplatformní a běží na operačních systémech Windows, macOS, Linux, Android, iOS a ve webových prohlížečích. V roce 2021 bylo na Discordu registrováno 350 milionů uživatelů a evidováno 150 milionů aktivních uživatelů denně.

Platforma vznikla jako nápad dvou studentů pro odvětví herního průmyslu, kteří cítili nedostatky v komunikaci již existujících VoIP software se spoluhráči v taktických hrách jako je Final Fantasy XIV. [1] První release Discordu tak přišel začátkem roku 2015, dnes po letech vývoje se používá nejen v odvětí herního průmyslu ačkoliv ten stále dominuje. [2]

Discord řadíme mezi platformy jako jsou např. Microsoft Teams, Microsoft Skype, Slack, Teamspeak, Reddit, Facebook Groups, které všechny patří ke komunikačním platformám s různými rozdíly podle uživatelů na které jsou cíleny.

2.1 Definice a terminologie

Discord používá svoji vlastní terminologii, kterou se budeme řídit v celém textu práce. Následující seznam nás seznámí s nejpodstatnějšími termíny.

`Server`
: Server neboli komunitní server, někdy také označován anglicky slovem *guild* v kontextu kódu, je v uživatelském rozhraní platformy uskupení sdružující uživatele, hlasové a textové kanály.

`Kanál`
: Kanál je místo, skrze které mohou uživatelé serveru komunikovat podle jeho typu. Existují pouze dva typy kanálů a to *hlasové*, nebo *textové*. Textové přenášejí zprávy, média a soubory, zatímco hlasové přenášejí navíc audio a video.

`Kategorie`
: Kategorie je pojmenované uskupení kanálů, které jsou v uživatelském rozhraní platformy zobrazeny jako složky. Kategorie mohou obsahovat další kategorie a kanály.

`Role`
: Role je pojmenovaná skupina uživatelů, které mohou být v uživatelském rozhraní platformy zobrazeny jako složky. Každá role s sebou nese svoji množinu systémových oprávnění a barvu k zobrazení. Role jsou postaveny v lineární hierarchii, role tak bývá nadřazena/podřazena jiné roli.

`Emoji`
: Discord emoji je spojením Unicode emoji jako jsou smajlíci a personalizovanými čtvercovými obrázky definované v rámci konkrétního discord serveru.

`Reakce`
: Reakce je discord emoji, které pomocí interakcí s tlačítkem u libovolné zprávy mohl uživatel nebo bot přidat.

`Uživatel`
: Uživatel je osoba, která má vytvořený účet na platformě a může se tak připojit k libovolnému serveru. Uživatelé se dělí na dva druhy: *člověk* nebo *bot*. Když říkáme slovo uživatel, myslíme tím druh účtu, který není automatizovaný a stojí za ním skutečný člověk.

`Bot`
: Bot je speciální druh uživatele, jehož účet je řízený programem, který komunikuje s Discord API. Má přístup k více funkcím než obyčejný uživatel, které může používat k různým úkonům. Navíc bývá méně limitován než běžný uživatel co se do počtu zaslaných požadavků do API týče.

Příkaz

Příkaz je textová zpráva napsaná v uživatelském rozhraní v poli pro odesílání zpráv. Příkazy se dělí na dva typy, a to prefixové nebo lomítkové. Prefixované mají serverem určený speciální symbol, který rozlišuje normální zprávu od příkazové k vyhodnocení botem. Tato zpráva je odeslána do chatu jako běžná zpráva. Lomítkový příkaz se neodesílá do chatu jako prefixový příkaz, ale je odchycen v Discord API a předaný botu k obsloužení.

2.2 Servery a komunity

Server na platformě je v uživatelském rozhraní jako kulatá ikonka v jeho levé části. Není zde myšleno, že server je výkonný počítač, který obsluhuje požadavky. Discord server, synonymem komunita či guilda, je nám poskytován a hostován přímo službou discordu jako takovou, která však běží na fyzických strojích společnosti, které obsluhují několik instancí discord komunit zároveň.

Na server se může uživatel dostat pouze pomocí pozvánky, která může být krátký textový řetězec připomínající hash sloužící jako identifikátor pozvánky. Popřípadě URL adresa, kde je identifikátor obsažen v parametru. Pozvánka může být implicitně parametrizována navíc o dobu expirace popřípadě limitem počtu užití. Neomezené pozvánky se zpravidla používají pro veřejné servery.

Při vytváření vlastního discord serveru můžeme v průvodci najít i šablony pro různé účely, které nám pomohou vytvořit a základně nastavit discord server podle našich specifických potřeb komunity.

Na serveru jsme schopni najít kategorie, textové a hlasové kanály a samotné uživatele serveru a případné integrace aplikací, které mají programem řízený účet odlišený za jménem pomocí štítku s nápisem „BOT“.

2.3 Kanály hlasové a textové

Kanály obecně mají společnou vlastnost, kterou je nastavení oprávnění podle rolí nebo „per user“.

Textové kanály mohou být dle oprávnění pro uživatele neviditelné, zamčené pro psaní, zakázané v jiných směrech např. posílání odkazů, přidávání reakcí, možnost smazat vlastní nebo cizí zprávu, nebo nastavené na určitý časový limit pro odesílání zpráv.

Hlasové kanály opět dle oprávnění mohou být neviditelné, zaheslované, nastavené maximálním limitem současně připojených uživatelů, nastavené kvality přenosu audia a videa.

Speciálním případem je komunikace uživatel s jiným uživatelem mimo server přes soukromý chat nebo hovor. Zde neplatí žádné oprávnění a uživatelé nejsou limitováni oprávněními, které lze aplikovat na role. Jedinou výjimkou jsou limity nastavené samotnou platformou, mezi které patří mazání zpráv druhého uživatele, komunikovat s druhým uživatelem mimo seznam přátel. Komunikace

mimo alespoň jeden server, který spolu sdílí, a uživatel druhý, příjemce, nevypl základní bezpečnostní funkci v prostředí svého účtu.

2.4 Uživatelé, role a pravomoce

Uživatel je kdokoliv kdo provedl registraci na platformě Discord a zavazuje se k dodržení podmínek služby.

Role je skupina oprávnění vztahující se pouze na serveru na němž byla vytvořena a může být přiřazena pouze uživatelům tohoto serveru.

K základní roli, kterou má každý server je role *everyone* s výchozím nastavením oprávnění, které se zpravidla ponechává a vytvářejí se role nové. Pro zpřísnění oprávnění než je role *everyone* bývá zvykem vytvořit novou roli, kterou některý z botů na serveru nastaví nově příchozím. Tato základní role však slouží v chatové zprávě pro hromadné označení všech členů serveru pomocí *@everyone*.

Existuje jedna role implicitní a tou je *server owner*, který se dá poznat, že v seznamu uživatelů serveru má uživatel za svým jménem ikonku královské koruny. Role není zobrazena v seznamu rolí serveru a není možné s ní nijak zacházet, má nejvyšší administrátorské oprávnění nade všemi.

Role na discord serverech jsou vytvářeny jako pořadový seznam v nastavení serveru, toto pořadí určuje jejich lineární hierarchii. Což znamená, že role nadřazena může dělat úkony na roli podřazené.

Pravomoce jsou elementární úkony, které mohou být nastaveny konkrétni roli. Mezi takové úkony patří: právo smazat zprávu, vytvořit pozvánku, vyhodit uživatele, upravit název kanálu, upravit nastavení serveru, vytvořit nové role. Všechny můžeme vidět v uživatelském rozhraní serveru pro správu rolí.

2.5 Limitace a předplatné

Myšleny jsou limity, které jsou kladeny na uživatele a komunitní servery na platformě s možností jejich výběru na základě předplatného, limitem aplikací pak rozumíme omezení, které jsou kladeny na účet bota a jeho přístup k informacím platformy.

Některé limity je schopen si uživatel nebo komunitní server navýšit díky předplatitelům. Pro uživatele se druh předplatného nazývá *Nitro Basic* a vylepšená varianta *Nitro*. Předplacený server se nazývá dle její úrovně, které jsou *Tier 1*, *Tier 2* a *Tier 3*.

Limity bez předplacení, podle tabulky 1, platné pro všechny uživatele a komunitní servery.

Limity s předplatným, se v případě předplatného pro uživatele řadí do dvou úrovní, pro komunitu se předplatné řadí do úrovní tří. Tabulka 2 ukazuje úrovně předplatného pro uživatele, tabulka 3 ukazuje předplatné úrovně pro servery.

K limitům patří také „rate limit“, což je maximální počet požadavků, které uživatelský účet nebo účet bota může platformě zasílat, bot má tento limit vyšší. Požadavek je např. odeslání zprávy, vytvoření pozvánky serveru, probíhající audio nebo videohovor. Při rychlém vyčerpání přiděleného maximálního počtu požadavků pro určitý časový úsek požadavky nemají žádný efekt dokud neuplyne určitý čas po kterém se předešlé požadavky automaticky pokusí sami zopakovat. Nadmírným a úmyslným čerpáním tohoto maximální počtu požadavků je proti ToS a může dojít k pozastavení nebo odstranění účtu jedince nebo komunitního serveru.

Typ	Limit
Počet uživatelů na serveru	250,000
Počet online uživatelů na serveru	5,000
Počet serverů uživatel smí být členem	100
Počet kategorií na serveru	50
Počet kanálů na serveru	500
Počet rolí na serveru	250
Počet neanimovaných emoji na serveru	50
Počet anomovaných emoji na serveru	50

Tabulka 1: Limity pro nepředplacené uživatele a komunity

Typ	Úroveň 1	Úroveň 2
Vlastní emoji kdekoliv	Ano	Ano
Vlastní samolepky kdekoliv	Ano	Ano
Velikost souborů	50MB	500MB
Vysoké rozlišení videa	Ne	Ano, až 4k@60
Personalizovaný profil účtu	Ne	Ano
Personalizovaný profil „per server“	Ne	Ano
Odznak předplatitele	Ano	Ano
Vlastní pozadí ve videohovorů	Ano	Ano
Počet serverů uživatel smí být členem	100	200
Delší zprávy	do 2000 znaků	do 4000 znaků

Tabulka 2: Úrovně limitů pro předplaticelské účty

Typ	Úroveň 1	Úroveň 2	Úroveň 3
Vlastních emoji serveru	až 100	až 150	až 250
Audio kvalita	až 128Kbps	až 256Kbps	až 384Kbps
Vlastní pozadí v pozvánce	Ano	Ano	Ano
Vlastních samolepek serveru	až 15	až 30	až 60
Animovaná ikona serveru	Ano	Ano	Ano
Kvalita 1080p@30 videa všem	Ano	Ano	Ano
Vlastní ikony rolí	Ne	Ano	Ano
Vlastní banner serveru	Ne	Ano	Ano
Kvalita 1080@60 videa všem	Ne	Ano	Ano
Až 50MB velikost souborů všem	Ne	Ano	Ano
Až 100MB velikost souborů všem	Ne	Ne	Ano
Animovaný vlastní banner serveru	Ne	Ne	Ano
Vlastní odkaz pozvánky	Ne	Ne	Ano

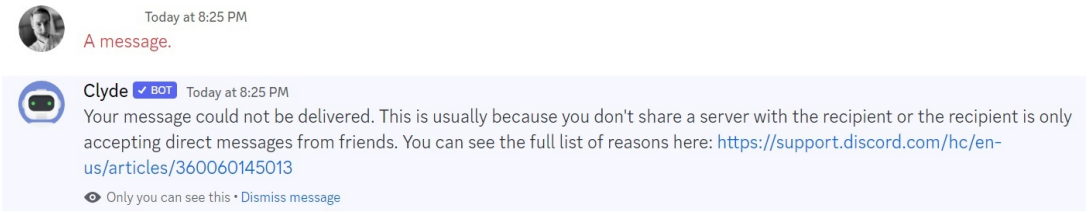
Tabulka 3: Úrovně limitů pro předplacené servery

3 Discord Bot

Discord bot je uživatelský účet, který je kompletně automatizovaný programem, tento program pak komunikuje prostřednictvým Discord API [5] s platformou na které provádí automatické úkony nebo úkony vyvolané uživatelem či prostředím. Platforma komunikuje s botem skrze WebSocket API [6] a předává tak v „realtime“ informace bez nutnosti navazovat neustále nová spojení, bot si tyto odpovědi ukládá do mezipaměti, jedná se tak o rychlý přenos informace mezi platformou a botem. Druhým směrem bot odpovídá platformě skrze její REST API [7], kde místo odesílání celých objektů odesílá pouze identifikátory objektů, které má ve své mezipaměti.

Bota si může zřídit jakýkoliv registrovaný uživatel discordu, což může učinit na oficiálních webových stránkách platformy, kde svoji aplikaci pojmenuje a vytvoří. Vytvořením dostane autorizační token, který poté následně použije ve zdrojovém kódu podle patřičné implementace Discord API knihovny pro zvolený programovací jazyk.

Clyde bot je bot jenž nepatří žádnému registrovanému uživateli, ale jedná se o systémového bota platformy, který je zde pro vylepšení a používání této platformy. Můžeme ho spatřit na obr. 1, když se pokusíme poslat zprávu uživateli, který nás zablokoval.



Obrázek 1: Systémový bot platformy Discord

3.1 Ovládání a práce s botem

Ovládání bota je buď automatické, nebo manuální. Automatické fungují tak, že přes websocket dostává bot informaci o eventech, které se na platformě stali, třeba uživatel odeslal zprávu a botu přijde event typu **on-message** pro který ve zdrojovém kódu bude vytvořena funkce, která funguje jako *callback* a bude spuštěna při tomto eventu z websocketu. Tyto eventy jsou předdefinované a je na programátorovi, které funkce jako callbacky vytvoří podle toho, kterým eventům chce jeho aplikace naslouchat. Manuální jsou funkce, které definují příkaz prostřednictvým uživatelského rozhraní v chatovacím okně dvěmi způsoby: prefixovanou zprávou, nebo příkaz za lomítkem.

Pokud bot obsahuje funkce, které jsou v příslušném jazyce knihovny Discord API označeny jako event a dodržují definované pojmenování, pak budou automaticky skrze knihovnu spuštěny jako callback.

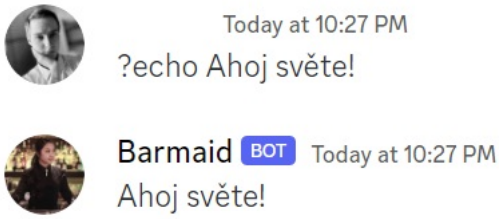
Uživatel bota nemusí v případě úkonů, které reagují na websocketem zaslané eventy, nic dělat. Pro úkony, které vyžadují zásah nebo doplnění informací k jejich provedení, musí uživatel tyto úkony vyvolat manuálně sám přes příkazy prefixové, nebo lomítkové.

3.1.1 Prefixované zprávy

Je způsob komunikace mezi uživatelem a botem, kdy uživatel napíše zprávu do chatu jako obvykle s rozdílem, že před začátek zprávy dá speciální symbol, který bot poté rozpozná, která zpráva obsahuje příkaz a která zpráva byla mezi probíhající konverzací na komunitním serveru. Mezi speciální symboly, které se často používají jsou: ?, !, &, *. Tyto symboly jsou vždy na začátku zprávy a následuje za nimi řetezec znaků, který je definován jako příkaz. Příkaz je pak rozdělen na jeden či více argumentů pomocí mezer, před první mezerou je tak symbolické jméno příkazu.

Na obr. 2 můžeme vidět příkaz *echo* s jedním argumentem následující za první mezerou po slově echo. Při implementaci funkce příkazu je vždy možnost pro poslední argument příkazu posbírat celý řetězec včetně mezer a chápat jej jako jeden argument jako tomu je v tomto případě.

Pokud máme příkaz s více argumenty, kdy do jednoho argumentu potřebujeme dát řetězec obsahující i mezery je potřeba tento řetězec obalit do uvozovek.

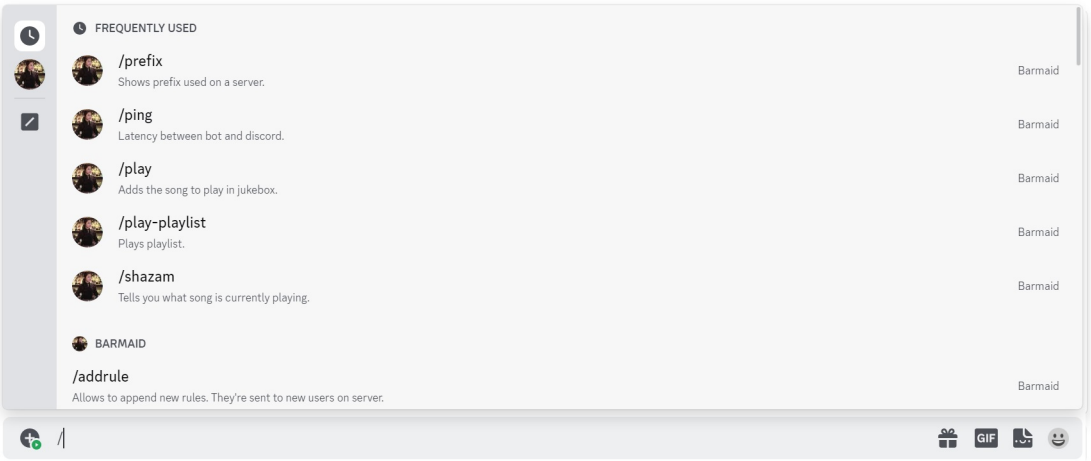


Obrázek 2: Příkaz vyvolaný prefixovanou zprávou

3.1.2 Lomítkové příkazy

Je novějším způsobem komunikace uživatel a bot, kdy uživatel předepíše do textového pole pro odeslání nové zprávy lomítko.

V uživatelském rozhraní se rozbalí menu s možnostmi příkazů, které jsou definovány v kódu bota. Uživatel vybere příkaz a je mu našeptáváno jaké argumenty musí použít a co do nich patří pro úspěšné vykonání příkazu. Poprvé byly lomítkové příkazy přidány do uživatelského prostředí a do odpovídajích API koncem roku 2020, v roce 2022 se stali jako povinností pro boty, kteří musejí být verifikováni za účelem nakládání s citlivými daty uživatelů platformy. Dokud bot nepřesáhne jeho počet 100 komunitních serverů ke kterým se připojil není potřeba žádného ověření.



Obrázek 3: Kontextové menu lomítkových příkazů

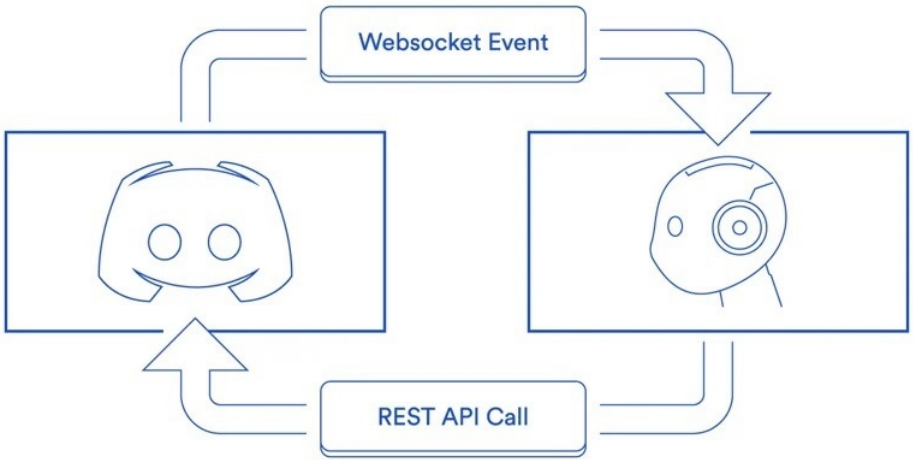
Na obr. 3 vpravo vidíme sloupec ve kterém nalezneme všechny boty na daném komunitním serveru, uprostřed nabídku se všemi příkazy kde shora jako první se zobrazují nejčastěji používané příkazy a na konci řádku příkazu vidíme název bota, kterému příkaz patří. Více botů může mít stejně pojmenovaný příkaz a uživatel je schopen je rozlišit podle profilové fotky bota, nebo jeho jména.

3.2 Technologie pro vývoj a implementaci

Pro komunikaci s Discordem má platforma své vlastní API, k počátku roku 2023 se datuje už *10.* verze. Přičemž se dá používat ještě verze předešlá, verze *6-8* jsou nyní zastaralé a verze *1-5* jsou již zrušené [5]. Komunikovat s tímto API napřímo je náchylné na chyby programátora při implementaci, které mohou způsobit nechtěné přetěžování API, které může vyústit k zablokování této komunikace platformou nebo dokonce účtu bota či samotného uživatele vlastnícího tohoto bota. Od těchto strastí s verzí a správnou komunikací nás abstrahují pryč knihovny pro práci s API platformy Discord. Knihovny jsou napsány jako „wrapper“ v patřičném programovacím jazyce usnadnující práci s API a ošetřují jeho nebezpečného zacházení. Další knihovny pro vybraný programovací jazyk mohou pomoci při komunaci bota s webovými stránkami na internetu, databází a dalšími službami pro specifickou činnost bota.

3.2.1 Discord API

Je preferovaný způsob komunikace mezi botem a platformou Discord, umožňuje snadné, rychlé a bezpečné vytvoření bota pro komunikaci s platformou přes dva základní prvky: **WebSocket API** a **REST API**. Na obr. 4 převzatého z článku [7], je vidět průběh komunikace kdy platforma vlevo komunikuje ve směru bota pomocí WebSocket API, a bot vpravo ve směru platformy pomocí volání REST API.



Obrázek 4: Způsob komunikace bota a platformy [7]

3.2.1.1 WebSocket API

WebSocket API je používán při přijímání eventů ze serverových strojů platformy Discord. Komunikace je navázána mezi platformou a botem skrze websocket, který umožňuje vytvořit přes informace v „realtime“, eliminuje se tím potřeba

pro každý přenos informace vytvořit nové spojení a vzniká tak velmi rychlý způsob přenosu informací.

Mezi eventy, které se přenáší z platformy botu jsou například: uživatel zaslal zprávu, uživatel upravil zprávu, uživatel smazal zprávu, uživatel se připojil ke komunitě, uživatel připl zprávu, uživatel vytvořil vlákno ve zprávě, uživatel se připojil k hovoru, uživatel si ztlumil mikrofon v hovoru.

Během přenosu informace o jakém eventu se stal je spolu s ním předávány objekty *zpráv*, *uživatele*, *místnosti*, *komunitního serveru* obsahující úplné informace. Po úspěšném přenosu informace o tom, že nastal event si bot ukládá všechny informace o objektech do **cache** se kterými je poté schopen zpracovat pro úkony co bot nad nimi provede a odesílá odpovědi s jeho reakcí do REST API.

3.2.1.2 REST API

Akce, kterých je bot schopen provést musí provést přes volání REST API, bot je také schopen si skrze REST API vyžádat získání informací, ale to není zpravidla kvůli výkonu používáno, neboť většinu těchto informací již dostal z websocketu a má je uložené ve své cache.

Bot z této cache vytáhne objekty se kterými potřebuje pracovat a při provedení změn nad nimi odesílá do REST API pouze **identifikátor** objektu a změnu, kterou provedl a neodesílá tak znovu celý objekt z důvodu rychlosti a případnému zahlcení API.

Identifikátor objektu je velké celé přirozené číslo, pod kterým si platforma je schopna najít objekt, kterému patří. V případě bota si i bot je schopen pomocí identifikátoru dohledat objekt kterému patří avšak s omezením, že objekt se musí nacházet v jeho cache. Nemůže tak například získat objekt komunitního serveru jenž bot není členem i kdyby znal jeho identifikátor, protože takový objekt mu websocket s tímto identikátorem nikdy nezaslal.

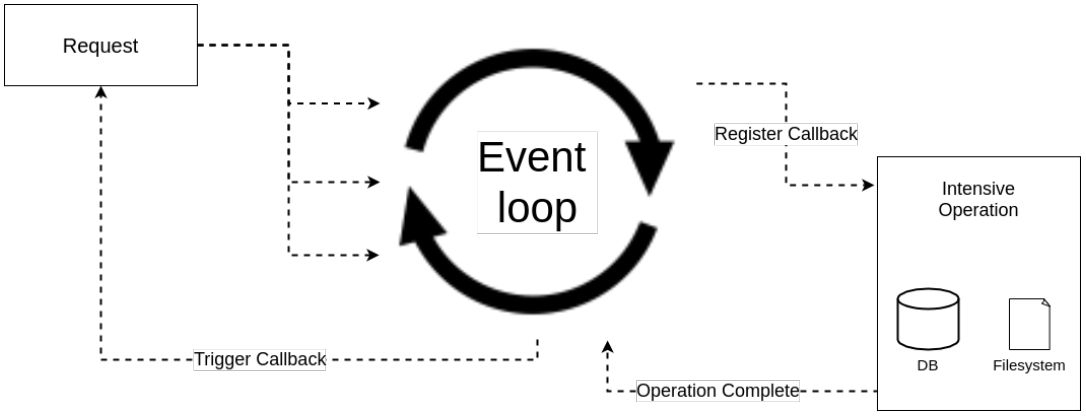
3.2.2 Asyncio

Asynchronní programování používané při komunaci klient-server trvá nějaký čas než se informace přenese a dostaví se odpovědi, pro využití procesorového času efektivně namísto čekání se vykonává něco jiného zatímco se čeká než odpověď dorazí. Asyncio, přesněji asyncIO je zkratka za asynchronous input/output, technologie která se stará o efektivní hospodaření s procesorovým časem při operacích, jenž mají delší efekt jako právě komunikace po síti.

Tato technologie dává možnost tzv. „single threaded“ procesům zdánlivě vykonávat více operací současně, jako jsou například samotné webové prohlížeče s *JavaScriptem*. Dlouho trvající operace nejprve všechny naráz jednu po druhé spustí a po vyčkání vrací v žádaném pořadí odpovědi po provedení operace, než aby se sekvenčně spustila jedna operace a vyčkalo se na dokončení než se spustí v sekvenci další dlouhotrvající operace.

Program má pouze jedno vlákno označované též jako **main thread** a tedy jeden **execution stack**, kde může vykonávat pouze jednu operaci. Při asynchron-

ním programování požadavek jako je získání dat z webové stránky by blokoval execution stack po celou dobu čekání odpovědi namísto toho předává tento svůj požadavek do **event loopu** a z execution stacku se tím funkce odstraní, ukládá se místo kam se po callbacku navrátí, a přenechá to na event loopu. Event loop se postaraná o obsloužení tohoto požadavku a spustí jej, po dokončení požadavku se skrze callback vrací odpovědi požadavku zpátky do execution stacku main threadu.



Obrázek 5: Asynchronní programování s jedním procesem [12]

Na obr. 5 vlevo vidíme request z main threadu programu předaný event loop smyčce starající se o časově náročné input/output operace vpravo, po jejich dokončení se vrací zpět přes callback do main threadu programu.

3.2.3 FFmpeg

Je open-source software CLI nástroj pro práci s audiem a videem a dalšími multimediálními soubory a datovými proudy. [13] [14] **Fast Forward moving picture experts group**, zkráceně FFmpeg byl vytvořen již v roce 2000 a dnes je používán ve spoustě aplikací jako je *Google Chrome*, *blender* a platforem jako *YouTube*, *Discord* a *Vimeo*. Nástroj je schopen dekódovat, enkódovat, transkódovat, multiplexovat, demultiplexovat, streamovat, filtrovat a přehrávat jakýkoliv multimediální soubor na světě. Podporuje přes 100 video kodeků.

FFmpeg funguje tak, že vezme jakýkoliv multimediální soubor, který pomocí demultiplexeru rozdělí do několika audio a video stop jako separátní data pakety. Pakety jsou dále dekódovány do jednotlivých snímků, které poté mohou být zprocesované či filtrované. Zprocesováním jako například úpravou jasu, kontrastu, přidáním titulků. Poté tyto snímky jsou zpátky zakódovány a multiplexerem složené nazpět do cíleného formátu.

Součástí nástroje je také nástroj *ffplay* pro přehrávání multimediálního souboru a *ffprobe* pro získání metadat ze souboru.

3.3 Knihovny pro práci s Discord API

K funkcionalitám platformy Discord sice jde přistupovat napřímo pomocí API v různých verzích a použitím HTTP protokolu a zasíláním *requestů*, ale knihovny mohou tuto činnost podstatně zjednodušit a usnadnit a použít ji je výhodnější. Každá knihovna zapouzdřuje všechna volání do Discord API a stará se o limity a chybové stavy.

Limit o který se stará je ratelimit a chrání uživatele knihovny před jeho překročením, pokud kód provádí spoustu požadavků rychle tak předchází zahlcením API pozdržejím požadavků překračující určitý limit a odesílá je postupně v pořadí co nejdříve je to možné.

Zamezuje odeslání informací, které by vedli k chybovým stavům API a tyto špatné informace propaguje knihovna svými chybovými stavy v příslušném programovacím jazyce.

Discord sám některé z knihoven třetích stran [11] schválil a označil, že splňují požadavky pro jejich API, výrazně také doporučil použití takových knihoven pro komunikaci s platformou než přístupem napřímo. Nejčastěji používané knihovny vidíme v tabulce 4 níže:

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

Tabulka 4: Knihovny pro práci s boty [11]

3.4 Knihovna discord.py

Jedná se o knihovnu pro Discord API v programovacím jazyce Python, vytvořenou jako open-source projekt se zdrojovým kódem na platformě Github. [15] Team, který knihovnu vytvořil ji udržuje aktuální, funkční a snaží se pokrýt vše, co nabízí celé Discord API a vytvořit tak jednoduchý, rychlý a bezpečný nástroj při vývoji a implementaci botů. Uživatele knihovny zcela abstrahuje od volání do zmíněného API, který v kódu pracuje pouze objektově orientovaným přístupem. Tento způsob dělá kód jednodušší, čitelnější a přenositelnějším.

Nově přidané funkce platformou Discord a Discord API bývají implementované do knihovny v řádu dnů, neboť se na jejím vývoji podílí přes 300 kontributorů.

Knihovna je ve dvou verzích, jedna bez podpory audia, druhá s podporou audia. Při její instalaci je tak nutné specifikovat, kterou nainstalovat, bez explicitního uvedení o audio verzi je nainstalována základní obsahující eventy (zasílané websocketem) a podpora pro vytváření příkazů.

3.5 Možnosti hostování aplikace

Spuštění může být realizováno na jakémkoliv hardware běžící s libovolným operačním systémem a nainstalovanými potřebnými programy na osobním počítači, osobním mikropočítači, nebo ve službě cloudu na pronajatém vzdáleném počítači.

Použití osobního počítače je způsob při vývoji a testování aplikace bota a to z několika důvodů. Je vhodné a rychlé vyzkoušet si všechny implementované změny ihned bez nutnosti přenosu souborů na jiný hardware. Avšak bývá nežádoucí ponechávat neustále zapnutý a připojený k síti osobní počítač, proto se na ostrý provoz vybírá plnohodnotný hosting.

Použitím osobního mikropočítače jako je Raspberry Pi [9] je plnohodnotná varianta k hostování aplikace bota. Raspberry Pi navíc dává velikou flexibilitu v operačních systémech, hardwarových specifikacích osobního mikropočítače a jeho provozování je levné.

Cloudová služba AWS Cloud Computing [10] dává možnost se u jejich služby zaregistrovat a pronajmout si vzdálený stroj na kterém bot následně poběží. Nabídka vzdálených strojů a jejich hardwarovou konfigurací je veliká, pro nenáročné aplikace postačující základní výpočetní výkon jsou tyto stroje i bezplatné.

4 Návrh funkcionalit

4.1 Automatické vlastnosti

4.1.1 Pravidla nově příchozím

4.1.2 Přiřazení pravomocí nově příchozím

4.1.3 Kontrola nově příchozích

4.1.4 Kontrola odesílaných textových zpráv

4.2 Manuální vlastnosti

4.2.1 Správa uživatelů

4.2.2 Správa událostí

4.2.3 Přehrávání audio stop

5 Implementace

5.1 Souborová struktura

5.2 Spouštěcí soubor

5.3 Vytvoření aplikace a účtu jako bot

5.4 Ukládání dat

5.5 Spouštění příkazů

5.6 Struktura příkazů v kódu

5.7 Pravidla komunit

5.8 Kontrola zpráv

5.9 Kontrola uživatelů

5.10 Události a jejich spravování

5.10.1 Události integrované v Discordu

5.10.2 Události zapouzdřené ve zprávě

5.11 Přehrávač

5.12 Nahrání bota na hosting

5.13 Testování bota

Závěr

tady budu muset napsat něco o tom že integrovaný discord eventy ještě nebyly když jsem měl téma bakalářky, ale během vývoje to tam přidali neboť je to fast growing app af

že se povedlo napojit na YouTube API a používat ho, dalo by se napojit i na Spotify ale je otázkou ToS sdílením placené hudby (bot by měl premium spotify acc)

a nakonec jako rozšíření by bylo supr zakomponovat do toho voice recognition když jsou uživatelé ve voice channels aby příkazy šli vyvolávat i hlasem a nejen přes psaný příkazy

ne vše je v discord api a museli jsme sáhnout do rest api sami a to pro integrované discord eventy které v discord.py ve verzi 2.0.0 ještě nejsou přidané, to byl problém avšak se to povedlo

Conclusions

Thesis conclusions in “English”.

A Obsah přiloženého datového média

Na samotném konci textu práce je uveden stručný popis obsahu přiloženého datového média (CD/DVD, flash disk apod.), tj. jeho závazné adresářové struktury, důležitých souborů apod.

bin/
: Instalátor Instalator programu, popř. program Program, spustitelné přímo z média. / Kompletní adresářová struktura webové aplikace Webovka (v ZIP archivu) pro zkopírování na webový server. Adresář obsahuje i všechny runtime knihovny a další soubory potřebné pro bezproblémový běh instalátoru a programu z média / pro bezproblémový provoz webové aplikace na webovém serveru.

doc/
: Text práce ve formátu PDF, vytvořený s použitím závazného stylu KI PřF UP v Olomouci pro závěrečné práce, včetně všech příloh, a všechny soubory potřebné pro bezproblémové vygenerování PDF dokumentu textu (v ZIP archivu), tj. zdrojový text textu, vložené obrázky, apod.

src/
: Kompletní zdrojové texty programu Program / webové aplikace Webovka se všemi potřebnými (příp. převzatými) zdrojovými texty, knihovnami a dalšími soubory potřebnými pro bezproblémové vytvoření spustitelných verzí programu / adresářové struktury pro zkopírování na webový server.

readme.txt
: Instrukce pro instalaci a spuštění programu Program, včetně všech požadavků pro jeho bezproblémový provoz. / Instrukce pro nasazení webové aplikace Webovka na webový server, včetně všech požadavků pro její bezproblémový provoz, a webová adresa, na které je aplikace nasazena pro účel testování při tvorbě posudků práce a pro účel obhajoby práce.

Navíc médium obsahuje:

data/
: Ukázková a testovací data použitá v práci a pro potřeby testování práce při tvorbě posudků a obhajoby práce.

install/
: Instalátory aplikací, runtime knihoven a jiných souborů potřebných pro provoz programu Program / webové aplikace Webovka, které nejsou standardní součástí operačního systému určeného pro běh programu / provoz webové aplikace.

literature/
: Vybrané položky bibliografie, příp. jiná užitečná literatura vztahující se k práci.

U veškerých cizích převzatých materiálů obsažených na médiu jejich zahrnutí dovolují podmínky pro jejich šíření nebo přiložený souhlas držitele copyrightu. Pro všechny použité (a citované) materiály, u kterých toto není splněno a nejsou tak obsaženy na médiu, je uveden jejich zdroj (např. webová adresa) v bibliografii nebo textu práce nebo v souboru `readme.txt`.

Literatura

[1] Wikipedia. *Final Fantasy XIV.* [online]. 2013. [cit. 2023-16-03]. Dostupné z: https://en.wikipedia.org/wiki/Final_Fantasy_XIV

[2] Wikipedia. *Discord.* [online]. 2014. [cit. 2023-16-03]. Dostupné z: https://en.wikipedia.org/wiki/Final_Fantasy_XIV

[3] Chanty. *Slack vs Discord – Gaming, Working or Both?* [online]. 2022. [cit. 2023-16-03]. Dostupné z: https://www.chanty.com/blog/discord-vs-slack/

[4] Discord. *Rate Limits.* [online]. 2015. [cit. 2023-17-03]. Dostupné z: https://discord.com/developers/docs/topics/rate-limits

[5] Discord. *Discord API.* [online]. 2015. [cit. 2023-17-03]. Dostupné z: https://discord.com/developers/docs/reference#api-reference

[6] Discord. *Gateway.* [online]. 2015. [cit. 2023-17-03]. Dostupné z: https://discord.com/developers/docs/topics/gateway

[7] Toptal. *How to Make a Discord Bot: an Overview and Tutorial.* [online]. 2010-2023. [cit. 2023-17-03]. Dostupné z: https://www.toptal.com/chatbot/how-to-make-a-discord-bot

[8] Caleb Hattingh. *Using Asyncio in Python.* O'Reilly Media, Incorporated, 2020. ISBN 978-149-2075-332

[9] Raspberry. *Computing for everybody.* [online]. 2012. [cit. 2023-18-03]. Dostupné z: https://www.raspberrypi.com/

[10] Amazon. *What is cloud computing?* [online]. 2006. [cit. 2023-18-03]. Dostupné z: https://aws.amazon.com/what-is-cloud-computing/

[11] Discord. *Community Resources.* [online]. 2015. [cit. 2023-18-03]. Dostupné z: https://discord.com/developers/docs/topics/community-resources#libraries

[12] Interactive Bees Blog. *Asyncio - vylepšené asynchronní paradigma.* [online]. 2022. [cit. 2023-18-03]. Dostupné z: https://blog.neonkid.xyz/283

[13] Wikipedia. *FFmpeg.* [online]. 2007. [cit. 2023-18-03]. Dostupné z: https://cs.wikipedia.org/wiki/FFmpeg

[14] FFmpeg. *Ffmpeg.* [online]. 2023. [cit. 2023-18-03]. Dostupné z: https://ffmpeg.org/

[15] Github. *discord.py.* [online]. 2023. [cit. 2023-18-03]. Dostupné z: https://github.com/Rapptz/discord.py