FAKULTA MECHATRONIKY,
INFORMATIKY A MEZIOBOROVÝCH
STUDIÍ TUL

Bakalářská práce

Tvorba a využití botu pro výuku matematiky na platformě Discord

Studijní program: B0613A140005 Informační technologie
Studijní obor: Aplikovaná informatika

Autor práce: **Radek Mocek**
Vedoucí práce: Ing. Igor Kopetschke
Ústav nových technologií a aplikované informatiky

Liberec 2024

FAKULTA MECHATRONIKY,
INFORMATIKY A MEZIOBOROVÝCH
STUDIÍ TUL

Zadání bakalářské práce

Tvorba a využití botu pro výuku matematiky na platformě Discord

Jméno a příjmení:	**Radek Mocek**
Osobní číslo:	M21000127
Studijní program:	B0613A140005 Informační technologie
Specializace:	Aplikovaná informatika
Zadávající katedra:	Ústav nových technologií a aplikované informatiky
Akademický rok:	2023/2024

Zásady pro vypracování:

1. Vypracujte rešerši sociálních platforem, které umožňují integraci botů.
2. Analyzujte vybranou skupinu existujících botů na platformě Discord a knihoven pro jejich tvorbu.
3. Navrhněte bot zaměřený na výklad a příklady z lineární algebry při využití specifických funkcí Discordu včetně administrace a interaktivních zpráv.
4. Navržené řešení implementujte a nasaďte do testovacího provozu pro vybranou skupinu uživatelů.
5. Vyhodnoďte zpětnou vazbu od uživatelů a navrhněte případné úpravy a vylepšení.

Rozsah grafických prací:	dle potřeby dokumentace
Rozsah pracovní zprávy:	30 – 40 stran
Forma zpracování práce:	tištěná/elektronická
Jazyk práce:	čeština

Seznam odborné literatury:

[1] COULOURIS, George F. Distributed systems: concepts and design. 5th ed. Boston: Addison-Wesley, c2012. ISBN 978-01-3214-301-1.

[2] BEČVÁŘ, Jindřich. Lineární algebra. Vydání páté. Praha: Matfyzpress, 2019. ISBN 978-80-7378-378-5.

[3] MORRIS, Tee. Discord For Dummies. New jersey: John Wiley & Sons, 2020. ISBN 9781119688037.

Vedoucí práce:	Ing. Igor Kopetschke Ústav nových technologií a aplikované informatiky
Datum zadání práce:	12. října 2023
Předpokládaný termín odevzdání:	14. května 2024

L.S.

prof. Ing. Zdeněk Plíva, Ph.D.
děkan

doc. Ing. Josef Chaloupka, Ph.D.
garant studijního programu

V Liberci dne 19. října 2023

Prohlášení

Prohlašuji, že svou bakalářskou práci jsem vypracoval samostatně jako původní dílo s použitím uvedené literatury a na základě konzultací s vedoucím mé bakalářské práce a konzultantem.

Jsem si vědom toho, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu Technické univerzity v Liberci.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti Technickou univerzitu v Liberci; v tomto případě má Technická univerzita v Liberci právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Současně čestně prohlašuji, že text elektronické podoby práce vložený do IS/STAG se shoduje s textem tištěné podoby práce.

Beru na vědomí, že má bakalářská práce bude zveřejněna Technickou univerzitou v Liberci v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů.

Jsem si vědom následků, které podle zákona o vysokých školách mohou vyplývat z porušení tohoto prohlášení.

21. dubna 2024 Radek Mocek

Poděkování

Děkuji vedoucímu své práce Ing. Igoru Kopetschke za poskytnutou podporu. Mé díky také patří všem testerům za jejich zpětnou vazbu.

Tvorba a využití botu pro výuku matematiky na platformě Discord

Abstrakt

Tato bakalářská práce přibližuje na problematiku tvorby Discord botů ve smyslu automatizovaných uživatelů. Jejím výsledkem je bot, který se zaměřuje na výklad a příklady z lineární algebry. Pro jeho implementaci je použita knihovna discord.py, která umožňuje tvorbu botů v jazyce Python. Výklad probíhá pomocí odesílání zpráv s teoretickými materiály do textového kanálu. Matematické příklady jsou pak náhodně generovány. Bot navíc demonstruje využití specifických vlastností platformy Discord, jako je systém rolí a interaktivní zprávy. Důraz je kladen na snadnou rozšiřitelnost o nové teoretické materiály a nové kategorie generovaných příkladů. Po implementaci je bot nasazen do testovacího provozu pro vybranou skupinu uživatelů, jejichž zpětná vazba je následně vyhodnocena.

Klíčová slova: Discord bot, Python, discord.py, výuka matematiky, lineární algebra

Creating and using a bot for teaching mathematics on the Discord platform

Abstract

This bachelor's thesis introduces the topic of creating automated users called bots on the Discord social platform. It results in a bot with a focus on theory explanation and mathematical problems from linear algebra. The bot is implemented in Python and uses the discord.py library. The theory explanation is achieved by sending messages containing theoretical sources to a text channel. Mathematical problems are generated randomly. Additionally, the bot demonstrates specific features of the Discord platform, such as roles and message components. Emphasis is placed on easy extensibility with new theoretical sources and mathematical problem categories. After the implementation, the bot is deployed for testing to a selected group of users, whose feedback is then evaluated.

Keywords: Discord bot, Python, discord.py, teaching mathematics, linear algebra

Obsah

Seznam obrázků

Seznam tabulek

Seznam zkratek

API	application programming interface, rozhraní pro programování aplikací
DMs	direct messages, přímé zprávy
FOSS	free and open-source software, svobodný a otevřený software
HTML	HyperText Markup Language, značkovací jazyk
HTTPS	HyperText Transfer Protocol Secure, protokol pro zabezpečenou komunikaci po počítačové síti
ID	identifikace, identifikátor
JDA	Java Discord API, knihovna pro tvorbu Discord botů v jazyce Java
JSON	JavaScript Object Notation, formát pro výměnu dat
NoSQL	not only SQL, typ databází
REST	Representational State Transfer, architektura pro výměnu dat
SDK	software development kit, sada nástrojů pro vývoj softwaru
SQL	Structured Query Language, dotazovací jazyk pro relační databáze
TUL	Technická univerzita v Liberci
UI	user interface, uživatelské rozhraní
VoIP	Voice over Internet Protocol, telefonie přes internetový protokol

1 Úvod

Platforma Discord, která původně vznikla za účelem poskytnutí online komunikace hráčům videoher, se dnes řadí mezi nejpoužívanější nástroje pro okamžité zasílání zpráv a internetovou telefonii. Kromě běžných uživatelů zde existuje ekosystém tzv. botů. Bot je speciální typ uživatele, jehož chování je určené programem. Velké množství existujících botů se zaměřuje na veřejné Discord servery, kde usnadňují jejich správu, poskytují statistiky, nebo přidávají nové sociální funkce. Jelikož je součástí komunity na platformě Discord mnoho mladších uživatelů – tedy i mnoho studentů – a zároveň se jedná o validní aplikaci pro provozování online výuky, nabízí se otázka, zdali by pro výukové účely na této platformě nemohl být vytvořen specializovaný bot.

Cílem rešeršní části této práce je provést analýzu sociálních platforem, které oficiálně podporují integraci botů. Následuje analýza Discord botů, kteří se zaměřují na matematiku, a knihoven, které tvorbu těchto botů umožňují. Cílem praktické části je pak vytvoření matematického Discord bota zaměřeného na výklad teorie a generování příkladů z lineární algebry, který by zároveň měl demonstrovat specifické vlastnosti zvolené platformy. Po implementaci bota následuje jeho nasazení do testovacího provozu, vyhodnocení zpětné vazby od uživatelů a navržení případných vylepšení.

Textová zpráva v kapitole 2 popisuje, jak jsou v kontextu této práce myšleny výrazy sociální platforma a bot. Následně je provedena zmíněná rešerše sociálních platforem. Nejvíce pozornosti je věnováno platformě Discord, která je vybrána pro implementaci bota. Kapitola 3 krátce popisuje, jaké postavení má téma matematiky na platformě Discord a poté shrnuje dva základní typy matematických botů, které se zde vyskytují. V kapitole 4 je provedena rešerše komunitních knihoven pro tvorbu Discord botů.

Kapitola 5 se věnuje praktické části práce. V sekci 5.1, která se zaměřuje na návrh, je bot popsán „zvenčí“. Je zde tedy představeno a zdůvodněno, jak vypadají jeho jednotlivé funkce z pohledu koncového uživatele. Implementační sekce 5.2 již popisuje, jaké kroky musely být provedeny, aby byl tento návrh realizován. Poslední sekce 5.3 obsahuje vyhodnocení zpětné vazby od uživatelů, návrhy na případná vylepšení a shrnutí problematiky vývoje pro zvolenou platformu.

2 Boti na sociálních platformách

Tato kapitola představuje Discord a některé další sociální platformy, které přímo podporují tvorbu a integraci botů. Nejdříve je ale nutné upřesnit, jak jsou v tomto kontextu myšleny výrazy *sociální platforma* a *bot*.

Discord je vcelku obtížné zařadit do jedné konkrétní kategorie softwaru. Dokud se nově zaregistrovaný uživatel nepřipojí na žádný server, pak se Discord chová jako aplikace poskytující okamžité zasílání zpráv a VoIP, kde lze komunikovat pouze s lidmi, které si uživatel přidá do přátel. Po připojení na nějaký veřejný Discord server se ale aplikace přibližuje ke kategorii sociálních médií, kdy uživatel může sdílet a konzumovat obsah v rámci moderované komunity. Pojem sociální platforma je zde tedy myšlen spíše jako zastřešující termín pro Discord a jemu podobné aplikace, u nichž o zařazení do této kapitoly především rozhodovalo, zdali nějakým způsobem podporují integraci botů. Společnou vlastností níže zmíněných platforem je také to, že všechny mají k dispozici aplikace pro web, desktop a mobilní zařízení.

Uživatelé sociálních médií často vnímají pojem bot negativně, jelikož se mezi boty mimo jiné řadí i programy generující spam nebo umělou návštěvnost za účelem zkreslení statistik [1]. Na platformách zkoumaných v této kapitole je ale bot speciální typ uživatele, jehož chování je automatizované a určené programem. Tento bot je zpravidla přidán do skupinové konverzace, kde pak za pomoci volání API dané platformy dokáže provádět stejné akce jako běžný uživatel. Jedná se například o odesílání zpráv a čtení jejich obsahu, moderaci členů serveru, nebo i připojení do hlasového hovoru. Nemusí se jednat o chatbota, který se snaží imitovat lidské chování. Bot obvykle reaguje na předem danou sadu příkazů vykonáním určité činnosti.

Aby byl bot online a funkční, musí být program, který definuje jeho chování, neustále spuštěný. O to se zkoumané platformy zpravidla nestarají, a proto je nutné mít program spuštěný na vlastním serveru nebo využít hosting třetí strany. Tento program pak komunikuje s danou platformou např. pomocí protokolu WebSocket. Komunikace může být velmi komplexní, proto vznikají komunitní i oficiální knihovny, které tvorbu bota značně usnadňují.

Účet pro bota může být založen v rozhraní pro vývojáře (např. *Discord Developer Portal*), kde je následně vygenerován jeho ověřovací token. Ten se pak používá při komunikaci programu s platformou a měl by zůstat tajný, jinak může dojít k tzv. odcizení bota. To znamená, že někdo jiný použije svůj vlastní program společně s uniklým tokenem pro předefinování chování bota, k jehož účtu daný token náleží.

Tabulka 2.1: Přehled sociálních platforem

Název	Vznik	Návštěv*	Bezplatná verze	Zaměření
Discord	2015	864 100 000	Ano	Obecné
Teams	2017	320 690 000	Omezená	Firemní
Slack	2013	97 000 000	Omezená	Firemní
Guilded	2017	1 260 000	Ano	Herní
Matrix/Element	2014/2016	494 420	FOSS	Obecné
Flock	2014	161 300	Omezená	Firemní
Revolt	2021	136 970	FOSS	Obecné

Název	Čeština	Oficiální SDK	Markdown**	TeX
Discord	Ano	Ne	5	Ne
Teams	Ano	Ano	3	Ne
Slack	Ne	Ano	2	Ne
Guilded	Ne	Ne	4	Ne
Matrix/Element	Ano	Ano	4	Ne
Flock	Ne	Ano	1	Ne
Revolt	Ano	Ano	5	Ano

* počet návštěv hlavní webové stránky za únor 2024 podle společnosti Semrush

** 0 až 5 bodů za: kurzíva, seznamy, víceúrovňové nadpisy, odkazy a zdrojový kód se zvýrazněním klíčových slov

2.1 Discord

Platforma Discord vznikla v roce 2015 jako reakce na nedostatky tehdejších služeb pro online komunikaci mezi hráči videoher [1]. S nabývající popularitou se ale na Discordu začaly objevovat veřejné komunity zaměřené i na jiná témata, než ta herní. Často se jedná o oficiálně vytvořené servery patřící k určitému produktu, které slouží jako diskuzní fórum a propagace zároveň. Momentálně je podle počtu členů největší komunitou server Midjourney, který patří ke stejnojmennému nástroji pro tvorbu obrázků pomocí umělé inteligence. Uživatelé zde mohou o nástroji diskutovat, hlásit chyby, sdílet vytvořené obrázky, nebo je nechat generovat použitím Midjourney bota.

Základní verze Discordu je zdarma s možností měsíčního předplatného *Discord Nitro*. Předplatitelé mají navíc k dispozici kosmetické prvky (např. animovaný avatar a vlastní emotikony) a některé základní funkce jsou pro ně vylepšeny (např. nahrávání větších souborů a streamování videa ve vyšším rozlišení).

Program disponuje obvyklými funkcemi pro okamžité zasílání zpráv a VoIP. V tzv. přímých zprávách si mezi sebou uživatelé mohou posílat textové zprávy a soubory. Textové zprávy je možné formátovat v jazyce Markdown např. pro psaní tučného textu, nadpisů, seznamů a zdrojových kódů se zvýrazněním klíčových slov. Dále je k dispozici hlasový hovor s případným sdílením obrazovky a videa z webkamery.

Tyto akce lze provádět i v soukromých skupinách, které ale mají omezený počet deseti uživatelů.

Discord server je izolovaná kolekce uživatelů a kanálů [2]. Textové a hlasové kanály funkčně odpovídají komunikaci v soukromé skupině. Rozdíl je ten, že komunikace v kanálu se mohou účastnit všichni členové daného serveru, kterých může být v základu až 500 000. Členové k tomu ale musí mít dostatečná oprávnění. Lze jim totiž přiřazovat tzv. role, u kterých se pak tato oprávnění nastavují. Např. všichni uživatelé s rolí x mohou číst historii zpráv v kanále y.

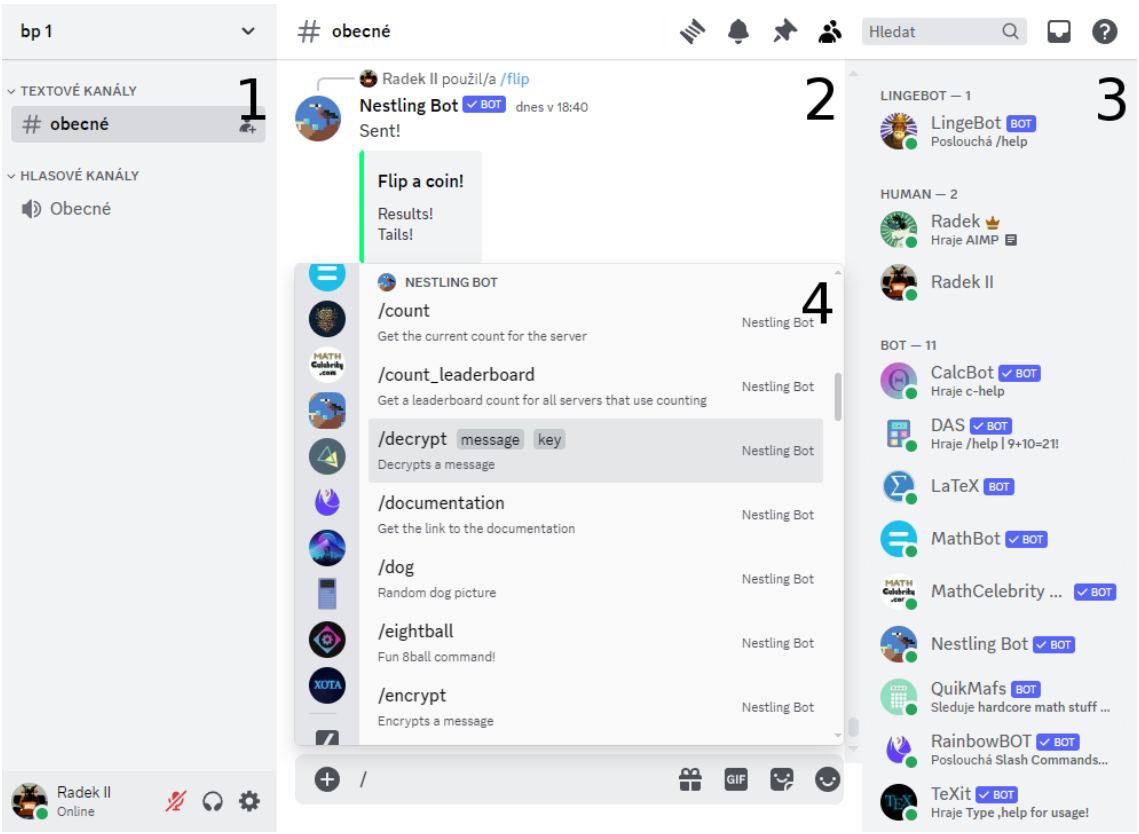
2.1.1 Aplikace a boti

Jako Discord aplikace se označuje vše, co oficiálně komunikuje s Discord API. Bot je pak automatizovaný uživatel, který může být k aplikaci přidán. Od roku 2021 jsou preferovanou metodou ovládání bota příkazy začínající lomítkem. Seznam těchto tzv. podpůrných příkazů (*slash commands*) se synchronizuje s Discordem a díky tomu je pak k dispozici jejich našeptávání. Namísto příkazu umí bot reagovat i na události jako *uživatel klikl na tlačítko* a *uživatel se připojil na server*. Botem odeslané zprávy pak mohou obsahovat interaktivní prvky jako tlačítko a výběrový seznam. Lze také vyvolat vyskakovací okno obsahující tyto prvky. Kromě práce se zprávami může bot upravovat nastavení serveru a připojovat se do hlasových hovorů. [3]

Boti tedy operují především na serverech[1] a existuje několik možností, jak je na vybraný server přidat. Na konci roku 2022 byl do Discordu přidán tzv. adresář aplikací (*App Directory*), který momentálně obsahuje seznam více než 6000 dostupných botů, kde každý z nich má svou stránku s podrobnějším popisem a tlačítkem pro přidání na server. Aby byl bot v tomto adresáři obsažen, musí nejprve projít schvalovacím procesem, a proto se jich zde nachází pouze zlomek ze všech existujících. Každý bot pak má svůj speciální zvací odkaz (*bot invite link*) a existuje několik neoficiálních databází jako top.gg, které obsahují seznam botů s jejich popisem a zvacím odkazem. Poslední možností jsou boti s otevřeným zdrojovým kódem, které si uživatel stáhne a spustí na svém stroji. Jedná se o provoz vlastní instance bota. Tento způsob tedy vyžaduje botu založit účet a vygenerovat token.

Discord boti nabízející nástroje pro usnadnění moderace zejména na velkých serverech patří mezi ty nejpoužívanější. Obvykle mají svou stránku s administrací, kde si lze např. zobrazit statistiky serveru, nastavit automatické přivítání a přiřazení rolí novým členům, vyhození za spam apod. Oblíbenou kategorií byli také hudební boti, kteří v hlasovém hovoru pouštěli vybrané skladby. Jejich popularita klesla roku 2021, kdy Google zakázal provoz botů přehrávajících hudbu z webu YouTube. Další častá kategorie botů jsou hry a zábava. Jedná se buď o boty poskytující jednoduché hry v prostředí textového chatu, nebo propojující nějakou existující videohru s Discordem.

[1] Boty lze ale používat i v přímých zprávách. Bot se sice tváří jako uživatel, do přátel ho ale přidat nelze a pro zahájení přímé konverzace je nutné s ním nejprve mít nějaký společný server. Bota nelze přidat do soukromé skupiny.



1. Kanály na vybraném serveru
2. Obsah vybraného textového kanálu
3. Členové a boti s přístupem do vybraného kanálu
4. Dostupné podpůrné příkazy vybraného bota

Obrázek 2.1: Uživatelské rozhraní platformy Discord

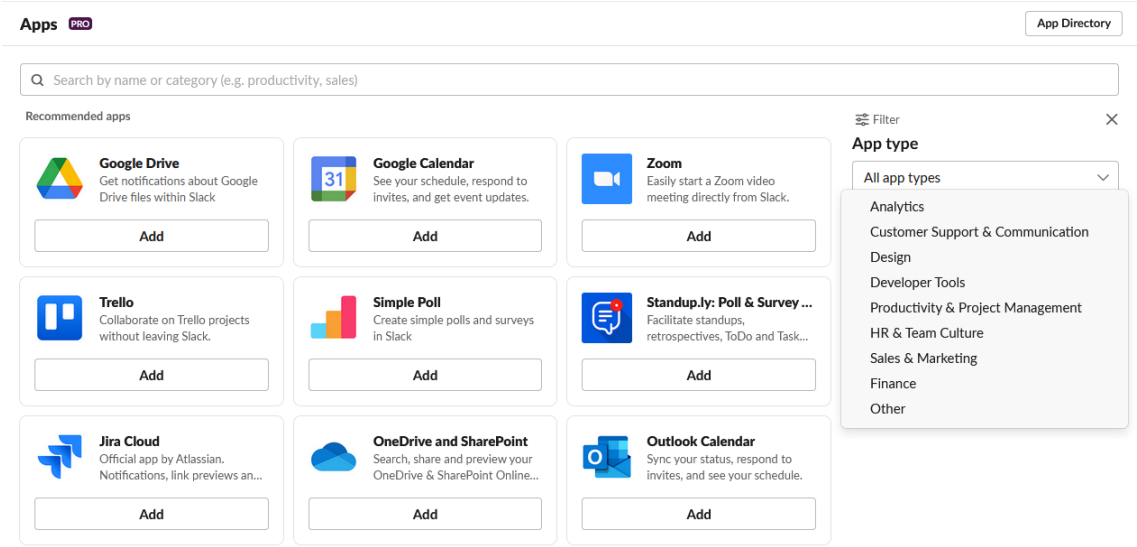
2.2 Slack

Z populárních platforem se Discordu vzhledově i funkčně nejvíce podobá program Slack. Ten byl spuštěn již v roce 2013 a od té doby si vybudoval reputaci jako standard pro komunikaci v technologických společnostech. [1]

Slack se prezentuje jako software usnadňující interní komunikaci v organizacích a dal by se nazvat jako „Discord pro firmy“. Namísto serverů jsou zde tzv. workspaces. Workspace má své členy a textové kanály, ve kterých lze zahájit i hlasový hovor. Členové jednoho workspace si mezi sebou mohou posílat přímé zprávy a pomocí funkce *Slack Connect* je možné komunikovat i s lidmi z jiného workspace. Na rozdíl od Discordu, kde jsou přímé zprávy kompletně oddělené od ekosystému serverů, se zde všechny prováděné akce dějí uvnitř nějakého workspace.

Slack má k dispozici několik úrovní měsíčního předplatného a verze zdarma je značně omezena. V bezplatné verzi nelze např. zobrazit zprávy starší než 90 dnů a provozovat hlasový hovor ve více než dvou lidech. Omezené je i použití tzv. aplikací.

Zdejší aplikace (*Apps*) se podobají zkoumaným botům a jejich výběr odpovídá cílové skupině Slacku. Zaměřují se především na zvýšení produktivity, teambuilding, nebo integraci již existující služby třetí strany do prostředí Slacku. Stejně jako uživatelé mohou Slack aplikace spravovat konverzace a odesílat do nich textové zprávy. Ovládání aplikací probíhá buď pomocí příkazů začínajících lomítkem, nebo má každá aplikace svou domovskou stránku a může také generovat vyskakovací okna. Aplikací odeslané zprávy, její domovská stránka a vyskakovací okna mohou obsahovat širokou škálu interaktivních prvků jako tlačítko, zaškrtávací pole, výběr data a času apod.



Obrázek 2.2: Doporučené Slack aplikace

Zorientovat se v procesu tvorby Slack aplikací nemusí být jednoduché. Aplikace se totiž dělí na klasické (*classic*) a nové (zvané *modern* nebo *next-generation*, představené v roce 2021). API těchto dvou druhů aplikací se liší. Ani slovo bot není ve Slack ekosystému jednoznačné. U klasických aplikací se jedná o jejich speciální typ, se kterým je komunikace vedena běžnou řečí namísto příkazů nebo UI (chatbot). Tento bot je označen jako *legacy* a v budoucnu pravděpodobně nebude podporován. U nového typu aplikací se z pohledu uživatele slovo bot nikde nevyskytuje (maximálně v názvu nějaké aplikace), z dokumentace je ale zjevné, že se stejně jako u Discordu jedná o speciálního uživatele ovládaného danou aplikací. Tato informace však není nikde explicitně zmíněna. Při práci s dokumentací nebo neoficiálními návody je tedy nutné dávat pozor na to, s jakou technologií daný manuál pracuje, protože může dojít ke snadné záměně výše zmíněných pojmů.

Klasické Slack aplikace lze vyvíjet v řadě komunitních knihoven. *Next-generation apps* pak mají připravené oficiální TypeScript SDK a o hosting se stará přímo Slack, pro jejich vývoj a následné nasazení je ale nutné mít placenou verzi Slacku.

2.3 Guilded a Revolt

Guilded a Revolt jsou platformy, které jsou přímo inspirované Discordem a podobnost mezi nimi je nepřehlédnutelná. Ekosystém uživatelů, přímých zpráv, serverů a botů je u těchto tří platforem takřka identický. Guilded ani Revolt ale nemají ekvivalent k podpůrným příkazům. Boti tedy musí kontrolovat obsah každé odeslané zprávy na serveru, zdali neobsahuje některý z jejich příkazů. Každý bot má obvykle svůj prefix jako např. vykřičník. Problém nastává, pokud má více botů stejný prefix a příkaz se stejným názvem, pak např. po napsání „!help“ do chatu odešlou svou nápovědu všichni takoví boti[2]. Obě platformy mají oproti Discordu znatelně menší uživatelskou základnu a počet jejich veřejných botů se pohybuje v řádu desítek.

Guilded (2017) lze považovat za konkurenci Discordu, která se zaměřuje na jeho původní účel. Tím je komunikace pro hráče videoher. Guilded přidává funkce užitečné pro pořádání pravidelných komunitních událostí. Uživatel zde také může propojit svůj účet s vybranými herními tituly. [4]

Program je zdarma, ale správci mohou na svém serveru zapnout dobrovolné měsíční předplatné, ze kterého je část odváděna Guilded. Pro tvorbu botů lze využít komunitních knihoven.

Revolt (2021) je Discord alternativa s otevřeným zdrojovým kódem. Oproti konkurenci nemá Revolt žádné speciální funkce, díky tomu je ale méně komplikovaný. Kromě řady komunitních knihoven pro tvorbu botů je k dispozici i oficiální revolt.js. Tato knihovna byla vytvořena tak, aby do ní šel snadno migrovat bot napsaný v knihovně discord.js.

2.4 Další platformy

Do této sekce byly zařazeny další sociální platformy, které sice podporují implementaci botů ve smyslu automatizovaných uživatelů, ale oproti výše zmíněným platformám mají buď znatelně odlišnou strukturu a uživatelské rozhraní, nebo menší uživatelskou základnu, která se pak projevuje i v malém počtu existujících botů a malé komunitě kolem knihoven pro jejich tvorbu.

2.4.1 Matrix

Matrix je otevřený protokol pro šifrovanou a decentralizovanou komunikaci po síti. Existuje pro něj několik klientských aplikací jako Element a poskytovatelů jako matrix.org. Protokol je také možné provozovat na vlastním serveru. Komunikace zde probíhá v tzv. místnostech. Místnost je jeden textový kanál s možností zahájení hlasového hovoru, který může být soukromý i veřejný. Pak jsou zde tzv. prostory,

[2] Tímto způsobem se před představením podpůrných příkazů ovládali i boti na Discordu. Problém stejných příkazů se řešil buď víceznakovými prefixy (menší šance na kolizi), nebo nastavitelným prefixem, to už vyžaduje úložiště uchovávající dvojice *ID serveru* a *prefix*. Tento starý typ příkazů lze stále používat např. pokud je vývojář chce skrýt před běžnými uživateli, protože se neobjevují v našeptávači.

které jsou v podstatě jen spravované seznamy různých místností. Podle dokumentace jsou prostory podobné Discord serverům nebo Slack workspaces, avšak zde připojení do prostoru uživatele automaticky nepřipojí do všech místností (uživatel si může vybrat, kam se připojí) a místnosti nejsou na prostorech závislé (místnost může existovat bez prostoru, může ale být i součástí více prostorů). Matrix podporuje komunikaci uživatelů s různými poskytovateli tohoto protokolu ve stejné místnosti.

Bot zde není speciálním typem uživatele, ale je pro něj zaregistrován běžný uživatelský účet. Program s logikou bota tedy může ovládat jakýkoliv účet, pokud zná jeho přístupový token[3]. Pro tvorbu takovýchto botů je připraveno oficiální TypeScript SDK. Veřejných botů pro Matrix není mnoho, mezi populární boty ale patří i boti modulární, do nichž mohou ostatní vývojáři přidávat vlastní funkce.

Specifickou vlastností protokolu Matrix jsou mosty (*bridges*). Ty umožňují propojit místnost s jinou platformou včetně Discordu a Slacku. Chat z platformy třetí strany by se pak měl zrcadlit s vybranou místností. Uživatelé z tohoto chatu mají v Matrix místnosti svou reprezentaci zvanou *ghost*. Pokud to daná platforma podporuje, tak v jejím ekvivalentu místnosti budou uživatelé ze strany Matrixu reprezentováni jako tzv. *puppets*. [5]

2.4.2 Microsoft Teams

Microsoft Teams je, stejně jako Slack, platforma pro firemní komunikaci. Každý účet na Teams náleží jedné organizaci a obvykle se do něj přihlašuje přes firemní e-mail. Organizace se pak dělí na týmy, které se mohou dělit na kanály. Tým je kolekce uživatelů, která by měla reprezentovat určitý celek v dané organizaci (např. lidé pracující na konkrétním projektu). Kanály jsou sekce svého týmu, ve kterých probíhá komunikace ve formě příspěvků a komentářů. Klasický textový chat s případným hlasovým hovorem probíhá v samostatné sekci a umožňuje se spojit i s uživateli z jiné organizace.

Podobně jako u Slacku se zdejší aplikace zaměřují především na zvýšení produktivity a propojení Teams s dalšími službami. Bot je v Teams terminologii aplikace vykonávající repetitivní úkony, která se může účastnit konverzace v chatu nebo kanálu. Boti se tvoří v oficiálním SDK, které je dostupné pro jazyky C#, JavaScript a Python.

Na konci roku 2023 bylo představeno Microsoft Copilot Studio, které mimo jiné podporuje tvorbu Teams botů pomocí grafického rozhraní bez psaní zdrojového kódu. Pro tento typ botů není potřeba vlastní hosting.

Platforma Teams je součástí předplatného Microsoft 365. Existuje i bezplatná verze, ta má ale omezené funkce a nedovoluje integraci aplikací včetně botů. Kromě Teams pro firmy byla později představena verze pro domácnosti. Ta by měla sloužit pro komunikaci s rodinou a přáteli. Většina funkcí je v této verzi vypnuta. Místo týmů jsou zde tzv. komunity, které se chovají totožně, pro jejich správu je ale nutné použít mobilní aplikaci.

[3] V terminologii Discordu se běžné uživatelské účty řízené programem nazývají *self-bots* a jejich provoz porušuje podmínky používání.

2.4.3 Flock

Flock je platforma pro firemní komunikaci, která podle svého webu nabízí oproti konkurenčnímu Slacku a Teams více funkcí za méně peněz. Tento web obsahuje výčet více jak sta vlastností, ve kterých je Flock údajně lepší než konkurence. Po bližším průzkumu je ale zjevné, že některá uvedená tvrzení jsou nepravdivá nebo zavádějící.

Např. je zde uvedeno, že do Slacku se nelze přihlásit pomocí účtu Google, soukromý Slack kanál nejde přepnout na veřejný a ve Slacku není k dispozici UI pro spravování seznamu úkolů. Během testování platformy Slack byla tato tři tvrzení prověřena a žádné z nich není pravdivé. Platforma Flock byla i přesto do rešerše zahrnuta, protože podporuje tvorbu botů a objevuje se v několika webových článcích typu *nejlepší platformy pro komunikaci*.

Struktura Flocku je podobná jako u ostatních platforem: Jednotlivé servery se nazývají týmy a obsahují textové kanály, ve kterých je možné zahájit hlasový hovor. Zdejší aplikace komunikují s Flock API a boti jsou jejich uživatelské reprezentace. Výběr aplikací je podobný jako u platforem Slack a Teams.

Platforma pro vývoj Flock aplikací se nazývá FlockOS a má slogan „World's First Chat Operating System“, s operačním systémem ale nemá nic společného. Boty lze programovat v oficiálním SDK pro jazyky Java, JavaScript a Python, jehož poslední verze ale vyšla v roce 2017.

2.4.4 Mattermost, Rocket.Chat a Zulip

Mattermost, Rocket.Chat a Zulip jsou komunikační platformy s otevřeným zdrojovým kódem. Díky tomu je možné provozovat tyto platformy na vlastních strojích (self-hosting), což je vhodné pro firmy, které díky tomu mají všechna data z chatů uložena u sebe. Všechny tři platformy ale nabízí i svůj hosting v několika úrovních měsíčního předplatného. Struktura těchto programů se pak podobá Slacku.

Mattermost nemá žádnou knihovnu pro tvorbu botů, veškerou logiku komunikace s jejich API tedy musí zajistit vývojář bota. Pro Rocket.Chat je možné vyvíjet boty v oficiálním JavaScript SDK, nebo je vytvořit v grafickém rozhraní na platformě Botpress. Zulip má pro vývoj botů k dispozici oficiální Python knihovnu.

3 Matematika na platformě Discord

Ze zkoumaných sociálních platforem má Discord nejkomplexnější a nejlépe zdokumentovaný ekosystém botů. Kolem tvorby těchto botů vznikla početná komunita a existuje pro ni tedy i nejvíce knihoven. Discord nemá žádné specifické zaměření (jako firemní nebo herní) a díky tomu pokrývá širokou škálu uživatelů. Proto byla pro tvorbu matematického bota vybrána právě tato platforma.

Ačkoliv nebyl Discord vytvořen pro účely výuky a nemá žádné speciální matematické nástroje, *vzdělávání* je jednou z pěti hlavních kategorií veřejných Discord serverů a patří do ní i servery specializované na matematiku. *Mathematics* je nejpopulárnější server s tímto zaměřením, na kterém mohou uživatelé diskutovat a řešit matematické problémy různých úrovní složitosti.

Na serveru je také automatický systém přiřazování textových kanálů uživatelům s konkrétním matematickým problémem. Ostatní uživatelé se v daném kanále mohou pokusit zadaný problém vyřešit. Jakmile je zadavatelova otázka zodpovězena, textový kanál se může použít pro problém dalšího uživatele. O logiku přiřazování kanálů se stará místní bot.

Matematicky zaměřených Discord botů není mnoho. V oficiálním adresáři aplikací nebyl během rešerše nalezen žádný. V neoficiálních databázích pak sice bylo nalezeno několik desítek botů, kteří se podle popisu alespoň částečně zabývají matematikou, velká část z nich ale byla offline a neměla k dispozici zdrojový kód. Otestováno bylo tedy nakonec 11 botů, které lze zařadit do dvou kategorií.

3.1 Bot vykreslující matematické výrazy

Jelikož Discord nemá žádnou oficiální podporu pro vykreslování matematických výrazů, což je v textové konverzaci ohledně matematiky velmi užitečná funkce, existuje hned několik botů, kteří touto funkcí disponují. Konkrétně vykreslení umožňuje 5 z 11 testovaných botů. K popisu výrazu se ve všech případech používá TeX syntax, ve které je zadán vstupní parametr vykreslovacího příkazu. Bot pak podle parametru vygeneruje obrázek s matematickým výrazem a odešle ho do textového kanálu, kde byl příkaz zadán.

Dva zkoumaní boti mají možnost zapnout automatické vykreslování: Jakmile jakákoliv odeslaná zpráva na serveru obsahuje dvakrát symbol dolaru, bot celou tuto zprávu vykreslí do obrázku a text mezi dolary nahradí matematickým zápisem (obrázek 3.1). Tři boti pak umožňují změnit barevné schéma generovaných obrázků. Je vhodné, aby tyto obrázky neměly průhledné pozadí, jinak by např. uživatelé,

kteří používají Discord ve světlém motivu, nemohli přečíst text v obrázcích s bílou barvou písma na průhledném pozadí.

Obrázek 3.1: Automatické vykreslení matematického výrazu

U botů psaných v jazyce JavaScript lze pro implementaci této funkce využít knihoven KaTeX a MathJax. V jazyce Python to pak umožňuje balík Matplotlib, který ale sám o sobě neumí vykreslovat složitější prvky jako např. matice. K tomu je potřeba, aby na stroji, kde je bot spuštěn, byla nainstalována TeX distribuce, která se pak použije namísto výchozího Matplotlib vykreslování. Další možností, která je nezávislá na použitém jazyku a knihovně, je využít pro vykreslení nějakou externí službu.

3.2 Bot jako kalkulačka

Celkem 9 z 11 otestovaných botů disponuje příkazy pro výpočet jednoduchých matematických příkladů. V lepším případě se jedná o příkaz typu *vypočti*, kterému se jako vstupní parametr zadá výraz s různými operacemi včetně závorek, goniometrických funkcí apod. Výstupem takového příkazu je výsledek celého výrazu. Méně praktickou variantou jsou pak boti se sadou příkazů pro různé operace typu *sečti*, *vynásob*, *vypočti sinus* apod. Vstupem těchto příkazů je buď jedno, nebo více čísel.

Kromě výpočtu numerických příkladů se také objevují příkazy např. pro úpravu výrazu a výpočet derivace. Sedm botů pak dokáže vygenerovat obrázek s jednoduchým grafem dle zadaného předpisu. Dva boti umí zadat uživatelův vstup do služby Wolfram Alpha a do chatu odeslat její výstup.

Jeden bot dokáže generovat náhodné příklady, jedná se ale pouze o čtyři základní binární operace. Tedy např. $x+y$, kde x a y jsou v rozmezí 1 až 99. Bot postupně generuje sérii deseti příkladů a uživatel má u každého deset vteřin na zadání výsledku. Po dokončení série příkladů je pak uživateli zobrazeno jeho skóre.

4 Prostředky pro tvorbu Discord botů

Discord API, se kterým program ovládající bota komunikuje, se dělí na dvě hlavní vrstvy: Pro obecné operace slouží HTTPS REST API a pro odesílání a přihlašování k událostem v reálném čase se používají trvalá zabezpečená připojení pomocí protokolu WebSocket. Samotná komunikace pak probíhá ve formátu JSON a pro unikátní identifikátory všech objektů se využívá formát Twitter Snowflake ID. Discord operuje na tak velkém měřítku, že zajistit úplnou konzistenci jeho dat by bylo nemožné, a proto je většina operací v Discord API eventuálně konzistentní. [2]

Eventuální konzistence znamená, že distribuovaná a replikovaná databáze povoluje relativně velké množství nekonzistentních dat. Pokud ale po nějaký čas neproběhne operace zápisu, tak se data na všech replikách postupně „srovnají“. [6]

Tvorba aplikací pracujících nad eventuálně konzistentními daty může být náročná. Události z Discord API navíc nemusí klientovi vždy přijít právě jen jednou, a proto je na ně nutné reagovat idempotentně. Výsledek několikrát provedené idempotentní operace by měl být stejný, jako kdyby byla provedena jen jednou [7].

Z důvodu usnadnění tvorby Discord botů začaly pro tento účel vznikat komunitní knihovny. Oficiální dokumentace jich zmiňuje celkem 21 pro 11 různých programovacích jazyků. Tato kapitola blíže představuje některé z těchto knihoven, ve kterých byl v rámci rešerše implementován jednoduchý příkaz pojmenovaný *pow*. Ten přijímá dva celočíselné parametry, se kterými provede umocňování. Ke zprávě s výsledkem je ještě přidáno tlačítko odkazující na webovou stránku (viz příloha B).

Tabulka 4.1: Nejpopulárnější knihovny pro tvorbu Discord botů

Název	**Jazyk**	**Vznik**	**Oficiální tutoriály**	**Popularita***
discord.js	JavaScript	2015	Ano	24 452
discord.py	Python	2015	Ne	14 024
DiscordGo	Go	2015	Ne	4 563
serenity	Rust	2016	Ne	4 259
JDA	Java	2015	Ano	4 038
Discord.Net	C#	2015	Ano	3 165
Pycord	Python	2021	Ano	2 619
Discord4J	Java	2015	Ano	1 697
Eris	JavaScript	2016	Ne	1 465
DSharpPlus	C#	2016	Ano	1 179

* počet GitHub hvězd ke dni 15. 2. 2024

4.1 discord.py

Discord.py je knihovna pro tvorbu Discord botů v jazyce Python, která je distribuována v podobě balíku instalovaného programem pip. Pro definici všech funkcí bota se využívá klíčových slov `async` a `await`. Bot je zde instancí předdefinované třídy s metodami, které se spouští při určité události a jejichž tělo lze dodefinovat. Pro přiřazení metod ke konkrétní události se často používají dekorátory, což je syntax umožňující upravit chování metod bez zásahu do jejich těla.

Pomocí tzv. cogs lze rozdělit logiku bota do více tříd, které dědí z třídy *Cog* definované knihovnou. Každá taková třída je v discord.py terminologii zvaná cog a slouží jako kolekce souvisejících metod, které obsluhují příkazy a reagují na události. Zvykem je, aby se každý cog nacházel ve zvláštním souboru a aby se každý z těchto souborů nacházel v adresáři pojmenovaném *cogs*. Instance cog tříd jsou pak do bota nahrány nikoliv přes importování, ale pomocí mechanismu rozšíření (*extensions*), který knihovna přidává. Za běhu programu je možné rozšíření přidávat a odebírat bez nutnosti restartování (*hot reload*).

Výhodou discord.py je jeho velká komunita, od které lze získat podporu např. na webu Stack Overflow nebo na oficiálním Discord serveru této knihovny. Díky použití jazyka Python a dekorátorů je psaný kód čitelný a zároveň má vysokou hustotu. Pro definici příkazů lze navíc využít anotací a komentářů u příslušných metod. Knihovna má k dispozici referenční příručku, která stručně popisuje všechny její funkce. Podrobnější popisy nebo příklady použití se zde ale vyskytují jen zřídka.

Největším problémem discord.py je podle hlasování uživatelů na GitHub Issues absence oficiálních tutoriálů. Referenční příručka je dobrým nástrojem hlavně pro programátory, kteří už obecnou tvorbu botů v této knihovně ovládají, a neobsahuje např. informace o tom, jak strukturovat větší projekt, nebo co znamenají jednotlivé funkce knihovny z pohledu koncového uživatele.

V roce 2021, kdy byly mimo jiné představeny podpůrné příkazy, vyšla nová verze Discord API, která přinesla zásadní změny v jeho používání. To vedlo k frustraci autorů některých knihoven a vývoj discord.py byl pozastaven. Po šesti měsících se repozitář opět otevřel a vývoj mohl pokračovat, mezitím ale vzniklo několik nových knihoven založených na discord.py (konkrétně Pycord, Nextcord a disnake). Zároveň začaly vycházet balíky, které do existující knihovny přidávaly podporu pro novou verzi Discord API. Zdrojové kódy a způsoby použití těchto knihoven jsou si velmi podobné, a proto je při hledání informací nutné dávat pozor na to, s jakou knihovnou daný zdroj pracuje.

4.2 discord.js

Discord.js je knihovna pro běhové prostředí Node.js umožňující tvorbu Discord botů v jazycích JavaScript a TypeScript. Logika knihovny je založena na objektech `Promise` a při psaní kódu se využívá tzv. arrow funkcí. Režie pro zprovoznění podpůrných příkazů a modularity (zde ve stylu *co soubor, to příkaz*) je o něco složitější než u discord.py.

Velkou výhodou této knihovny je existence oficiálních tutoriálů, které programátora provedou krok za krokem od úvodní instalace npm balíčku až po tvorbu pokročilých příkazů. Kromě popisu postupů a zdrojových kódů se díky tomuto návodu programátor rychle dozví, co všechno do svého bota vlastně může implementovat. Nechybí ani referenční příručka a početná komunita jako u discord.py.

Knihovna nevyužívá dekorátorů ani komentářů pro definování příkazů. Výsledný kód obsahuje mnoho vnořených nebo zřetězených funkcí, kvůli kterým může být delší a hůře čitelný. Knihovna sice podporuje TypeScript, oficiálních ani komunitních návodů pro jeho korektní použití ale není mnoho.

4.3 Discord.Net

Discord.Net je knihovna pro tvorbu Discord botů na platformě .NET, kterou lze do projektu přidat přes systém NuGet. Dokumentace je psaná pro jazyk C#, ve kterém je tato knihovna implementována. Pro tvorbu bota je ale možné použít i jiné jazyky z této platformy jako F# a Visual Basic. Knihovna má k dispozici referenční příručku i tutoriály, její nevýhodou je menší komunita.

Pro tvorbu složitějších botů se doporučuje použít vkládání závislostí[1]. Oproti předchozím knihovnám Discord.Net u podpůrných příkazů odděluje logiku jejich synchronizace a obsluhy. Kód zde tedy není strukturován na metody odpovídající jednotlivým příkazům, ale všechny příkazy jsou obsluhovány jedinou metodou, ve které je kód dále rozvětven[2].

4.4 JDA

JDA je knihovna pro tvorbu botů v jazycích Java a Kotlin. Do projektu je přidána buď pomocí nástroje Maven, nebo Gradle. Stejně jako u Discord.Net je zde oddělená logika synchronizace podpůrných příkazů s Discordem od jejich samotné obsluhy.

Referenční příručka JDA je vygenerována přes Javadoc, což může být vhodné pro programátory zvyklé na jazyk Java, na první pohled ale působí složitě a nepřehledně. K dispozici jsou také tutoriály včetně ukázkové implementace bota přehrávajícího hudbu v hlasovém hovoru. Ze čtyř zkoumaných knihoven se jedná o jedinou, která by takto v návodu obsahovala krok za krokem kompletní proces tvorby bota s konkrétním zaměřením.

[1] Vkládání závislostí (*dependency injection*) je návrhový vzor, kde se třetí strana stará o vztahy mezi komponentami a jejich závislostmi [8]. Konkrétně jde o to, aby byla za běhu programu do vybraných atributů dosazena správná instance.

[2] Podpůrné příkazy musí být synchronizovány, aby je mohli uživatelé používat. V discord.py a discord.js jsou všechny potřebné informace pro synchronizaci (jméno, popis a vstupní parametry) součástí definice obslužné funkce daného příkazu. V Discord.Net jsou tyto informace uvedeny ve speciální metodě pro synchronizaci. Druhá speciální metoda pak obsluhuje všechny příkazy. Obvykle je uvnitř druhé metody rozvětvení typu *switch* podle jména příkazu.

5 Tvorba matematického Discord bota

Tato kapitola popisuje tvorbu matematického Discord bota, který byl pojmenován *LingeBot*. Zaměřuje se na výklad teorie a generování příkladů z lineární algebry. K výkladu této látky lze přistupovat s různou mírou obecnosti: Některé zdroje prezentují lineární algebru jako sadu postupů pro řešení určitých matematických úloh, jiné ji vykládají jako teorii vektorových prostorů a lineárních zobrazení [9]. Bot byl vytvořen spíše pro prezentování informací z druhé zmíněné strany spektra, avšak jeho návrh obecně podporuje jakýkoliv typ matematických materiálů. Generované příklady se pak soustředí na základní výpočty s maticemi.

Pro tvorbu bota byla zvolena knihovna discord.py zejména kvůli její početné komunitě a čitelnosti výsledného zdrojového kódu. Při volbě také hrála roli existence matematických knihoven pro jazyk Python (konkrétně Matplotlib, NumPy, SciPy a SymPy), které jsou zavedené a dobře zdokumentované.

5.1 Návrh

5.1.1 Vykreslování matematických výrazů

Jak již bylo zmíněno v sekci 3.1, pro textovou komunikaci ohledně matematiky je vhodné používat správné matematické symboly. Příkladem je situace, kdy chce jeden uživatel odeslat a popsat druhému uživateli vzorec obsahující mocniny (např. x^2). Pokud si uživatel nepamatuje specifický alt kód, pak symbol „^{2}“ není možné na klávesnici napsat. Ne příliš praktickým řešením je najít a zkopírovat daný symbol buď z internetu, nebo programu jako je *mapa znaků*. Další možností je zavést alternativní značení, tedy např. místo x^2 psát $x\text{\^{}}2$ nebo $x**2$. Tato značení ale nejsou nijak normalizovaná a v textové konverzaci zejména více uživatelů mohou vést k nedorozuměním. Obě možnosti navíc selhávají ve chvíli, kdy je potřeba zapsat buď velké množství různých symbolů, nebo složitější výrazy obsahující zlomky, matice atd. Některá textová prostředí podporují vykreslování HTML: Pro horní a dolní index lze využít značek `sup` a `sub`, řecká písmena lze pak psát jako `&alpha;`. Pro jakýkoliv složitější matematický výraz je ale jediným zavedeným způsobem program TeX a jeho syntax.

Protože Discord vykreslování TeX výrazů nepodporuje, jedná se o první funkci navrhovaného bota. Konkrétně jde o příkaz `/render` (*vykresli*), jehož parametrem je matematický výraz zapsaný v TeX (resp. Matplotlib Mathtext) syntaxi, který má být vykreslen. Po spuštění příkazu je do chatu odeslán obrázek obsahující

vykreslení požadovaného výrazu. Pod tímto obrázkem se nachází trojice tlačítek *Potvrdit*, *Upravit* a *Smazat*. Tlačítko pro smazání kompletně odstraní botem odeslanou zprávu. Tlačítkem pro úpravu se vyvolá vyskakovací okno s textovým polem, ve kterém je předvyplněn uživatelův původní vstup. Po opravě nebo rozšíření výrazu je daný obrázek překreslen. Pokud se ve vstupu od uživatele nachází chyba, pak zpráva místo obrázku obsahuje výpis z chybového hlášení, díky kterému může uživatel chybu snadněji detekovat. Tlačítko potvrzení pouze nechá trojici tlačítek zmizet a tedy znemožní výraz dále upravovat nebo ho smazat. Tato akce se po určitém časovém intervalu provede automaticky, aby tlačítka zbytečně neznepřehledňovala textovou konverzaci.

Obrázek 5.1: Příkaz `/render` – vykreslený matematický výraz a vyskakovací okno

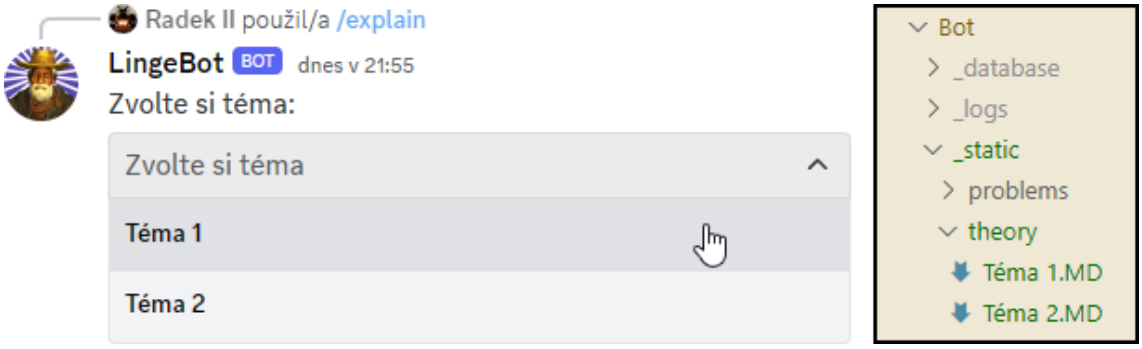
Vykreslování matematických výrazů je stěžejní funkcí navrhovaného bota. Kromě příkazu `/render` totiž může být použito i během výkladu teorie nebo při zobrazování vygenerovaných příkladů.

5.1.2 Výklad teorie

Výklad teorie funguje na principu odesílání zpráv obsahujících předem připravené výpisky z určitého tématu. Je dostupný pod příkazem `/explain` (*vysvětli*), po jehož zadání je uživateli zobrazen výběrový seznam všech dostupných témat. Každé téma odpovídá jednomu textovému souboru. Díky tomu nejsou výpisky součástí zdrojového kódu a snadněji se upravují. Obsah souborů je psaný v jazyce Markdown, který Discord v textových zprávách podporuje a tedy umožňuje jejich formátování.

Jazyk Markdown nemá žádný oficiální styl zápisu pro označení matematických výrazů, zvykem ale je výraz uzavřít z obou stran řetězcem tvořeným dvěma symboly dolaru („$$“, dále jen „dvoudolar“). Pokud se tedy ve výpiscích nachází text ohraničený dvoudolary, pak je tato část do textového kanálu odeslána ve formě obrázku podobně jako u příkazu `/render`. Maximální délka odeslané zprávy na Discordu je 2000 znaků, a proto je bot schopen odesílané výpisky rozdělit do více zpráv.

Každé teoretické téma se dělí na několik podtémat, jejichž názvy a obsah jsou určeny podle nadpisů druhé úrovně v Markdown souboru. Po zvolení tématu



Obrázek 5.2: Příkaz `/explain` – výběr témat

z výběrového seznamu je do chatu odeslán seznam příslušných podtémat, mezi kterými lze přepínat pomocí tlačítek *Další* a *Předchozí*. V textovém kanálu se tedy vždy nachází zprávy patřící k aktuálně vybranému podtématu, které jsou po stisknutí jednoho z tlačítek smazány a nahrazeny novými. K flexibilnějšímu přepínání mezi podtématy byl přidán další výběrový seznam, z něhož lze vybrat jakékoliv podtéma bez nutnosti procházet je popořadě jako při použití tlačítek. Dalšími tlačítky u rozhraní výkladu teorie jsou *Odeslat do DMs* (přepošle uživateli zprávy k danému podtématu do přímých zpráv), *Ukončit* (nechá zmizet tlačítka, odeslané zprávy zůstanou) a *Ukončit a smazat* (tlačítka i zprávy jsou odstraněny). Pokud konverzace s botem již probíhá v přímých zprávách (namísto serveru), pak tlačítko *Odeslat do DMs* není dostupné.

Během návrhu se původně uvažovalo o možnosti nahrávání nových témat uživateli. Discord umožňuje jako parametr příkazu předat soubor a uživatel by takto mohl do bota nahrát vlastní Markdown zápisky. Tento návrh ale otevírá několik otázek: Budou si moct nahrané zápisky zobrazit všichni uživatelé, nebo jen členové stejného serveru? Má být nastaven limit maximálního počtu nahraných souborů na konkrétního uživatele nebo na server? Jak se stavět k nevhodnému obsahu? Navíc by bylo potřeba rozhraní, kde by uživatel mohl své nahrané soubory spravovat. To by v prostředí textového chatu nebylo příliš praktické, vhodnější by bylo k botu ještě vytvořit webové rozhraní.

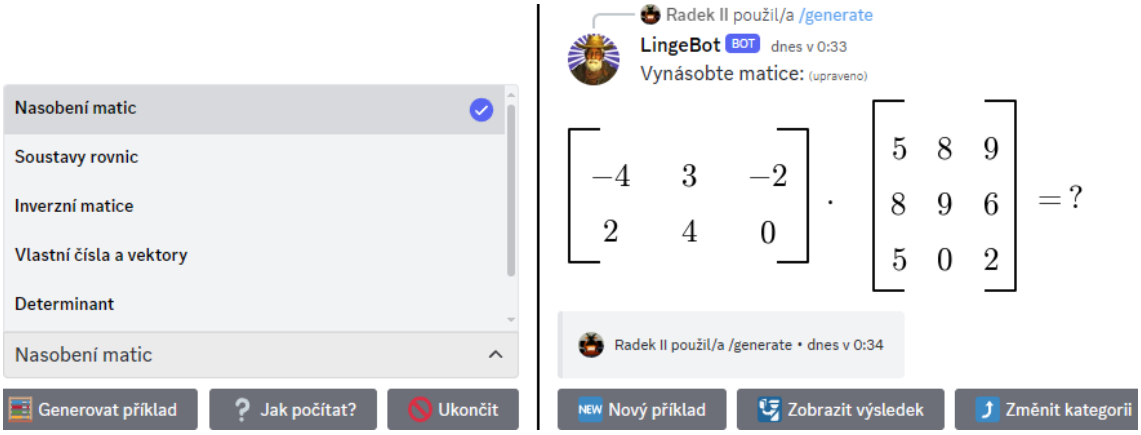
Nakonec byl zvolen přístup, kde si každý uživatel může provozovat vlastní instanci navrhovaného bota. Soubory s tématy jsou pak nakopírovány do vyhrazeného adresáře, odkud mohou být snadno spravovány.

5.1.3 Generace příkladů

Příklady lze generovat použitím příkazu `/generate`, po jehož zadání je do textového kanálu opět odeslán výběrový seznam, který tentokrát obsahuje dostupné kategorie příkladů. Po volbě některé z kategorií se objeví trojice tlačítek *Generovat příklad*, *Jak počítat?* a *Změnit kategorii / Ukončit*. Po vygenerování příkladu jej bot bez výsledku odešle do chatu. Uživatel si může zkusit příklad vypočítat, a poté si nově objeveným tlačítkem nechat zobrazit výsledek. Tento proces lze opakovat, dokud není rozhraní ukončeno.

Navrhovaný bot neobsahuje mechanismus pro kontrolu uživatelem zadaných výsledků. Pokud by takovým výsledkem byla např. matice většího rozměru, její zadávání by bylo zdlouhavé a náchylné na překlepy. Muselo by také mít nastavená jasná pravidla zápisu, nebo by uživatelský vstup musel projít komplexnější syntaktickou analýzou. Pokud se uživatel dozví, že jeho výsledek je nesprávný, dalším krokem by mělo v každém případě být nalezení chyby a poučení se z ní. Znalost výsledku sice může hledání chyby trivializovat, avšak frustrace z nesprávných výsledků u uživatelů pravděpodobně stejně vede k okamžitému zobrazení těch správných.

Po stisknutí tlačítka *Jak počítat?* jsou do textového kanálu odeslány zprávy s návodem pro výpočet příkladů z dané kategorie. Logika odesílání těchto zpráv je stejná jako u výkladu teorie. Obsah je tedy čerpán z Markdown souborů, které lze snadno upravovat, a nechybí tlačítko pro přeposlání návodu do přímých zpráv. Každá kategorie příkladů nemusí nutně obsahovat návod, pak je tlačítko pro jeho zobrazení zašedlé a nelze s ním interagovat. Opačný stav, kdy existuje pouze návod a nelze generovat příklady, je také možný.



Obrázek 5.3: Příkaz `/generate` – výběr kategorie a vygenerovaný příklad

5.1.4 Možnosti moderace

Aby byly demonstrovány specifické vlastnosti platformy Discord, byly do bota přidány funkce pro moderaci (administraci) pomocí sady příkazů `/setup`. Administrátoři serveru mohou u každého příkazu nastavit, kdo má oprávnění jej používat. Lze vybrat ze tří kategorií: pouze administrátoři, administrátoři a moderátoři bota, nebo kdokoliv (výchozí stav). Tato nastavení dávají smysl pouze při používání bota na serverech a v přímé konverzaci nemají význam.

Administrátor je typ oprávnění, které je součástí Discordu a uživatelům ho lze přidělit v rámci role. Pokud není taková role nikomu přidělena, pak je majitel serveru jediným uživatelem s těmito právy a zároveň jediným, kdo může tuto roli někomu dalšímu přidělit.

Za „moderátora bota“ je považován uživatel, kterému byla administrátorem přidělena speciální role s názvem *LingeMod*. Tato role je vytvořena navrhovaným botem

po jeho připojení na nový server. Díky moderátorské roli lze uživatele bez administrátorských práv rozdělit na dvě skupiny a pouze jedné z nich povolit používání určitých příkazů.

Nastavení může vypadat např. následovně: Kdokoliv může vykreslovat matematické výrazy pomocí `/render`, pouze moderátoři bota (a tedy i administrátoři, kteří jim jsou nadřazeni) mohou generovat příklady pomocí `/generate` a pouze administrátoři mohou používat výklad teorie příkazem `/explain`.

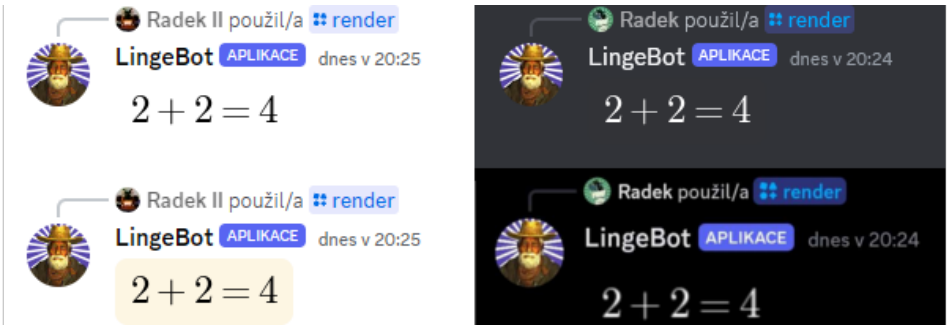
Podobná oprávnění lze nastavit i u tlačítek a výběrových seznamů, které jsou součástí botem odeslaných zpráv. Ve výchozím stavu s nimi může interagovat pouze uživatel, na něhož je zpráva s interaktivními prvky mířena, tedy zadavatel daného příkazu. Interakci s „cizími“ ovládacími prvky je ale možné povolit jakémukoliv administrátorovi nebo i moderátorovi bota. Tlačítko pro přeposlání podtématu do přímých zpráv je vždy dostupné všem.

5.1.5 Další funkce

Navrhovaný bot obsahuje příkaz `/help` pro zobrazení stručné nápovědy. Zvykem při tvorbě Discord botů je také přidání příkazu `/ping`, pomocí kterého lze otestovat, zdali je bot online. Příkaz ping je inspirován stejnojmenným programem a odpověď na něj obvykle vypisuje dobu odezvy (latenci) daného bota. V tomto případě byly do výpisu přidány i další informace jako doba provozu (uptime) a počet serverů, na kterých se navrhovaný bot nachází.

Dalším přidaným příkazem, který byl inspirován prostředím příkazové řádky, je `/clear`. Po jeho zadání bot do chatu odešle zprávu tvořenou neviditelnými znaky, která je tak dlouhá, aby se všechny předchozí zprávy již nevešly na obrazovku. To může být užitečné pro zpřehlednění konverzace v daném textovém kanálu.

U bota je možné změnit barevné schéma vykreslovaných matematických výrazů. Konkrétně jsou k dispozici čtyři kombinace barvy písma a pozadí u odeslaných obrázků. Barevné schéma je vázané na server a může být přenastaveno jeho administrátorem. Všechny botem odeslané obrázky na určitý server budou respektovat jeho nastavené schéma nezávisle na uživateli příkazu. Uživatel si totiž může nastavit barevné schéma pro konverzaci s botem v přímých zprávách. To je použito u zpráv reagujících na příkazy spuštěné v přímé konverzaci a zároveň u zpráv přeposlaných z jakéhokoliv serveru pomocí tlačítka *Odeslat do DMs*.



Obrázek 5.4: Příkaz `/render` – dostupná barevná schémata

5.2 Implementace

5.2.1 Struktura projektu

Před implementací jednotlivých funkčních částí výsledného bota bylo nutné rozhodnout, jak bude projekt strukturován. U knihovny discord.py je toto rozhodnutí ponecháno především na vývojáři. Dokumentace knihovny neobsahuje doporučený způsob, jak kód bota rozdělovat do více souborů, ani ukázku většího projektu, kterým by se dalo inspirovat.

Zvyk ohledně cog souborů (sekce 4.1) byl během implementace bota dodržen. Logika vykonání příkazů ale byla ve většině případů oddělena do samostatných souborů, které se v pomyslné hierarchii projektu nachází na jiné úrovni. Cogs je pak importují a pouze volají jejich metody. Podle toho, zdali jsou tyto soubory závislé na knihovně discord.py, je lze rozdělit do dvou skupin.

Soubory, jež Discord knihovnu importují, ale nejedná se o cogs, se nachází v adresáři *d_modules*. Je v nich implementována logika, která je společná pro více příkazů, jako je kontrola oprávnění pro spuštění příkazu a práce s textovými zprávami. Zároveň se zde nachází veškerá logika pro obsluhu interaktivních prvků jako tlačítek a vyskakovacích oken, ale i modul obsahující definici samotného bota.

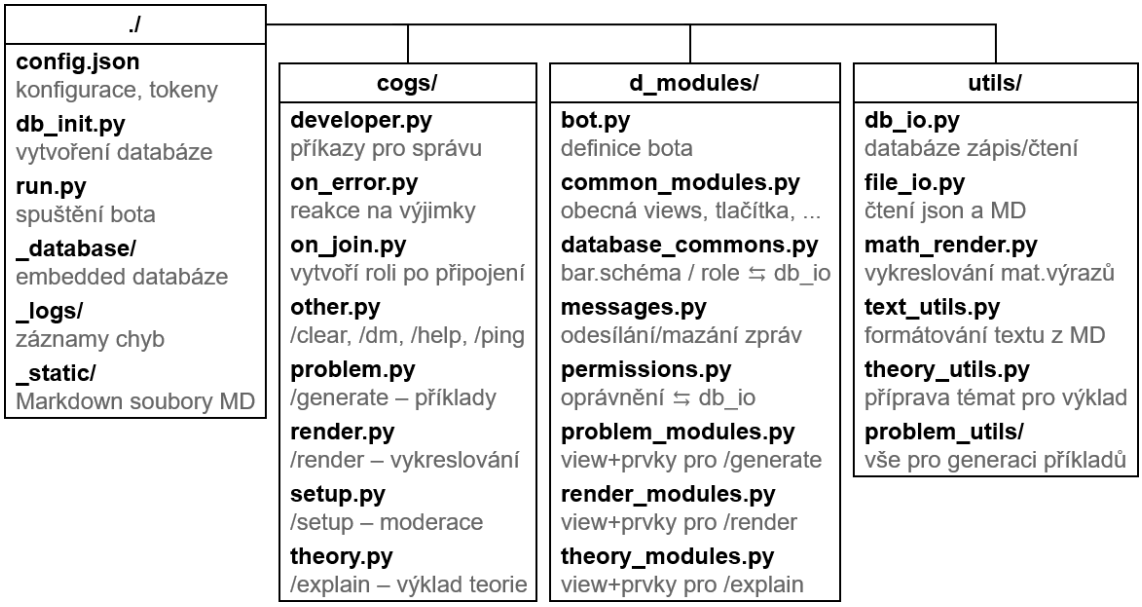
Druhá skupina souborů, které knihovnu discord.py neimportují, se nachází v adresáři *utils*. Metody těchto modulů jsou používány především v *d_modules* pro společné funkce, které jsou nezávislé na Discordu. Tyto moduly naopak importují knihovny jako sqlite, Matplotlib a SymPy. Starají se tedy o práci s databází, vykreslování matematických výrazů a generování příkladů.

Vedle těchto adresářů se ještě nachází soubory *config.json* (uložení tokenu bota), *db_init.py* (vytvoření databáze), *run.py* (spuštění bota) a adresáře *_database*, *_logs* a *_static*. Adresář *_static* obsahuje Markdown soubory s výpisky z teorie a návody pro počítání příkladů.

5.2.2 Implementace vykreslování matematických výrazů

První verze implementace příkazu `/render` byla prostá. Pomocí balíku Matplotlib je vytvořen prázdný obrázek *figure*, jehož barva odpovídá barvě pozadí Discord klienta ve výchozím nastavení. Parametr příkazu, který obsahuje předpis matematického výrazu, je do obrázku vložen jako text se zapnutým parametrem *render_math*. Balík Matplotlib se postará o správné vykreslení výrazu a obrázek je uložen do souboru, který je odeslán do textového kanálu. Později bylo ukládání do souboru nahrazeno proudem *io.BytesIO*, který souborové rozhraní pouze simuluje.

Výklad lineární algebry se neobejde bez vykreslování matic, ty ale Matplotlib Mathtext nepodporuje. Balík sice umožňuje nastavit, aby vykreslování používalo místní TeX distribuci, během implementace ale nebylo jasné, kde bude výsledný bot nasazen a jestli zde bude možné TeX nainstalovat. Vykreslování tímto způsobem je navíc pomalejší. Proto byla pro účely bota napsána vlastní logika vykreslování matic.



Obrázek 5.5: Struktura projektu

U textu vkládaného do Matplotlib obrázku lze nastavit jeho souřadnice x a y. Matice na vstupu vykreslovací funkce je rozdělena na své jednotlivé prvky, které jsou vykresleny zvlášť na odpovídajících souřadnicích. Pomocí úseček jsou pak vytvořeny hranaté závorky matice. Problém může nastat při určování souřadnic jednotlivých prvků v matici.

Pokud jsou všechny prvky matice jednociferná čísla, pak jejich souřadnice x odpovídá číslu sloupce a y číslu řádku v matici. Tyto souřadnice jsou ještě vynásobeny konstantou, která určuje velikost mezery mezi jednotlivými prvky[1]. Jakmile by ale byl některý z prvků matice širší (např. trojciferné číslo), začal by se překrývat se svými sousedními prvky. Proto je potřeba přidat funkci, která dokáže zjistit šířku prvků v matici. Odsazení je pak nastaveno podle nejširšího nalezeného prvku.

Šířku prvku lze určit jeho dočasným vykreslením do samostatného obrázku, u kterého jsou následně zjištěny jeho rozměry. Tento obrázek je opět uložen do proudu BytesIO, který je po zjištění šířky zahozen. Takovýto způsob není nejefektivnější zejména u vykreslování větších matic, a proto byla implementována rychlejší funkce, která šířku prvků pouze aproximuje.

Cílem funkce je pro zadaný matematický výraz v TeX syntaxi vrátit číslo, které odpovídá jeho délce. Jednotkou délky je počet vykreslených číslic. Písmo použité u vykreslovaných matematických výrazů je proporcionální a některé znaky tedy mohou být širší než jiné. Číslice jsou ale jako jediné neproporcionální, a proto byly zvoleny jako jednotka délky. Protože se jedná pouze o aproximaci, je důležité, aby funkce délku výrazu přeceňovala. Podcenění by mohlo vést k překrývání sousedních prvků matice.

[1]Konstanta u souřadnic y by měla být záporná, protože řádky matice jsou obvykle číslovány odshora, zatímco souřadnice v Matplotlib obrázku mají počátek vlevo dole. V této implementaci je to ale vyřešeno tak, že se jednotlivé řádky matice při vykreslování procházejí pozpátku.

```
Funkce __approx_tex_len(text: str, nahradit_zlomky: bool) vrací int:
Krok 1:
        Pokud nahradit_zlomky:
                Nahraď všechny výrazy v text ve tvaru "\frac{Č}{J}"
                za výstup funkce __approx_tex_frac_replace(Č, J).
Krok 2:
        Nahraď všechny podřetězce v text,
        které začínají na "\" a končí libovolným nepísmenným znakem,
        za tříznakový řetězec "000".
Krok 3:
        Nahraď všechny znaky v text,
        které se zároveň nachází v "+-=ABCDEFGHKMNOPQRUVWXYZmw",
        za dvouznakový řetězec "00".
Krok 4:
        Odstraň všechny znaky z text,
        které se zároveň nachází v " _^{}[]".
Krok 5:
        Vrať délku upraveného řetězce text.

Funkce __approx_tex_frac_replace(čitatel, jmenovatel: str) vrací str:
        délka_čitatele = __approx_tex_len(čitatel, false)
        délka_jmenovatele = __approx_tex_len(jmenovatel, false)

        Vrať řetězec tvořený znaky "0"
        o délce maximum(délka_čitatele, délka_jmenovatele).
```

Obrázek 5.6: Pseudokód aproximace délky vykresleného výrazu

Aproximovaná délka vykresleného výrazu odpovídá počtu znaků vstupního řetězce po tom, co je na něm provedeno několik úprav. Kroky 2 a 3 nahrazují určité části textu řetězcem s dvěma (třemi) nulami. To, že je nahrazujícím znakem nula, není podstatné. Důležitá je délka těchto řetězců. Následující krok odebere řídící znaky a mezery, které se nevykreslují. Poté už je vrácena délka řetězce.

Krok 2 pokrývá příkazy jako `\sqrt` a `\approx`. Ne všechny tyto symboly jsou dlouhé jako tři číslice. Extrémním případem je příkaz `\iota`, který je dokonce kratší než jediná číslice. Jak již ale bylo zmíněno, délku je lepší přecenit nežli podcenit. Krok 3 nahrazuje znaky, u kterých bylo zjištěno, že jsou delší než jedna číslice. Jedná se především o velká písmena, ale patří sem i znaky jako plus a minus.

Při určování délky jsou problémové zlomky psané jako `\frac{Č}{J}`. V nejhorším případě, pokud je délka čitatele $\check{C}$ i jmenovatele J stejně dlouhá, by byla šířka zlomku vyhodnocena jako dvojnásobek té skutečné. Proto byl přidán krok 1, ve kterém se funkce rekurzivně volá na všechny zlomky ve vstupním řetězci. Ty jsou pak nahrazeny řetězcem o aproximované délce jejich čitatele, nebo jmenovatele. Vybrán je ten delší z nich.

Stejný problém jako zlomky také způsobují kombinační čísla `\binom{N}{K}`, ta se ale běžně v maticích nenachází. Krok 2 původně nahrazoval pouze dvouznakovým řetězcem, to ale bylo málo např. pro příkazy `\sin` a `\cos`, které lze použít pro popis matice rotace. Proto byla délka zvýšena na tři, to je ale stále málo pro příkaz `\sinh`.

Tabulka 5.1: Příklady aproximace délky vykresleného výrazu

Krok 1	`1+\sqrt{x}` `1+\sqrt{x}`	`1+\sqrt[N]{x}` `1+\sqrt[N]{x}`	`x^2\approx\frac{123}{4}` `x^2\approx000`
Krok 2	`1+\sqrt{x}` `1+000{x}`	`1+\sqrt[N]{x}` `1+000[N]{x}`	`x^2\approx000` `x^2000000`
Krok 3	`1+000{x}` `100000{x}`	`1+000[N]{x}` `100000[00]{x}`	`x^2000000` `x^2000000`
Krok 4	`100000{x}` `100000x`	`100000[00]{x}` `10000000x`	`x^2000000` `x2000000`
Krok 5	7	9	8
Porovnání	$1+\sqrt{x}$ 1234567	$1+\sqrt[N]{x}$ 123456789	$x^2 \approx \frac{123}{4}$ 12345678

Popisovaná funkce by tedy stále šla vylepšovat pro mnoho dalších případů, které by mohly nastat. Vzhledem k zaměření bota by ale již všechny časté scénáře měly být pokryté.

Jelikož je pro vykreslování matic použita vlastní funkce, bylo možné zvolit styl jejich zápisu. Pro svou přehlednost, snadné strojové zpracování a rychlost zápisu uživatelem byla zvolena syntax podobná maticím v prostředí MATLAB: Do hranatých závorek jsou postupně vypsány prvky matice oddělené čárkou a pro oddělení jednotlivých řádků se místo čárky píše středník.

Po implementaci funkcí pro vykreslení základních matematických výrazů a matic bylo potřeba tyto funkce zkombinovat, aby bylo možné vykreslovat rovnice obsahující více matic. Uživatelský vstup je nejprve nutné rozdělit na maticové a nematicové části (podvýrazy), pro jejichž vykreslení je pak zvolena příslušná funkce. První podvýraz je vykreslen na nulové souřadnici x, která se postupně zvětšuje o aproximovanou délku podvýrazu. Při rozdělování na podvýrazy je nutné ignorovat hranaté závorky u odmocnin, maticový podvýraz je totiž detekován podle hranatých závorek. Aby byl celý výraz správně vertikálně zarovnán, je potřeba předem zjistit počet řádků té nejvyšší matice a podle ní určovat souřadnici y zbylých podvýrazů.

Posledním krokem bylo přidání podpory pro víceřádkové výrazy zarovnané podle určitého znaku. Obvykle se zarovnává podle rovnítka. Pomocí dvou zpětných lomítek je v uživatelském vstupu označen nový řádek. Na každém řádku lze zapsat ampersand, který není vykreslen ve výsledném obrázku a značí, že následující symbol je ten zarovnávací. Každý řádek je podle ampersandu rozdělen na levou a pravou část. Obě tyto části jsou vykresleny na nulové souřadnici x, u té levé je ale nastaveno Matplotlib zarovnání vpravo pomocí parametru *horizontalalignment*.

Vlastní funkce pro vykreslování matic ale nepočítá s přenastavováním zarovnání na úrovni balíku Matplotlib. Pokud se tedy ve zpracovávaném řádku nachází matice, pak je jeho levá část zarovnána pomocí funkce pro aproximaci délky výrazu: Matplotlib zarovnání je ponecháno na výchozí hodnotě (vlevo) a souřadnice x je rovna záporné aproximované délce. V opačném případě, kdy se v řádku žádné

matice nevyskytují, je přeskočen krok s rozdělováním na podvýrazy a rovnou se sází Matplotlib Mathtext.

5.2.3 View a interaktivní prvky

K odeslané zprávě s vykresleným matematickým výrazem je přidána trojice tlačítek *Potvrdit*, *Upravit* a *Smazat*. Aby mohl být ke zprávě přidán jakýkoliv interaktivní prvek, musí být nejprve vytvořeno tzv. view. To slouží jako kontejner pro všechny interaktivní prvky a ke zprávě je přidáno přes stejnojmenný parametr. Pro definici složitější view logiky je možné vytvořit třídu, která dědí z obecného view. Do této třídy se pak přidají atributy pro uchování určitého stavu a metody pro změnu tohoto stavu. Každý interaktivní prvek má referenci na své view, díky tomu v něm lze např. po stisknutí tlačítka volat určitou metodu.

Pro případy bota byla vytvořena třída *LingeBotView*, která rozšiřuje obecné view. Třída uchovává referenci na zprávu, ke které je přiřazena, a na uživatele, který zadal příkaz vyvolávající tuto zprávu. Odkaz na uživatele se využívá pro kontrolu oprávnění. Odkaz na původní zprávu v tomto případě není potřeba, z *LingeBotView* ale dědí další třídy, které se používají pro uživatelské rozhraní výkladu teorie a generace příkladů. Tyto třídy již odkaz na původní zprávu využívají. Protože se obecné view nikde jinde nepoužívá, jsou tyto třídy pro zjednodušení dále v textu nazývány pouze jako view.

Jednotlivá tlačítka také dědí z podobné obecné třídy, u které primárně přepisují metodu *callback* volanou po jejich stisknutí. Tlačítko *Smazat* nevyžaduje po svém view odkaz na rodičovskou zprávu, protože si ho může získat z objektu interakce, který je metodě *callback* automaticky předáván jako první parametr. Tlačítko pro úpravu matematického výrazu v sobě musí mít uložen původní uživatelský vstup, který je předvyplněn v textovém poli vyvolaného vyskakovacího okna, a barevné schéma, které bylo pro dané vykreslení použito. Vyskakovací okno po svém potvrzení volá stejnou funkci jako příkaz `/render`, jen místo nové zprávy upraví tu původní. Zvolené barevné schéma se uchovává v proměnné, aby se nemuselo při každé úpravě výrazu znovu zjišťovat z databáze.

5.2.4 Zpracování chyb

Pokud je za běhu bota v nějaké části kódu vyvolána výjimka, neznamená to ukončení programu. Knihovna discord.py výjimky v podstatě ignoruje. Ve výchozím nastavení jsou pouze vypsány do standardního chybového výstupu. Průběh funkce, ve které výjimka vznikla, sice není dokončen a Discord uživateli zobrazí hlášku „interakce se nezdařila“, bota by však tímto způsobem nemělo být možné vyřadit z provozu.

Do implementovaného bota byla přidána funkce logování chyb do textového souboru. Zároveň byl vytvořen cog *on_error*, který hlásí chyby uživatelům. Výjimka, která byla vyvolána z důvodu nedostatečného oprávnění uživatele pro vykonání určité akce, může při používání bota na serveru nastávat běžně. Proto není součástí logování a má svou připravenou odpověď. Tuto výjimku vyvolává dekorátor přidaný knihovnou, který kontroluje, zdali je uživatel administrátor. Vyvolání stejné

výjimky bylo implementováno i do vlastní kontrolní funkce, která bere v potaz roli moderátora a nastavení oprávnění na daném serveru.

U ostatních výjimek je do příslušného textového kanálu odeslán poslední řádek jejich výpisu. Tento přístup není vždy žádoucí, protože může případným útočníkům pomoci lépe pochopit vnitřní implementaci programu. Zdrojové kódy tohoto bota jsou ale dohledatelné, jelikož je tvořen s možností provozu vlastní instance a případného rozšiřování jeho obsahu. Díky výpisu chyb do chatu mohou uživatelé tyto chyby snadněji nahlašovat.

5.2.5 Implementace výkladu teorie

Před implementací logiky výkladu teorie byly vytvořeny samotné Markdown soubory. Ty ve formě převážně odrážkových výpisků pokrývají určitá témata z lineární algebry. Souborů je celkem 13 a zaměřují se především na vektorové prostory a lineární zobrazení mezi nimi.

Pro výběr teoretického tématu se využívá výběrový seznam. Názvy jednotlivých položek v seznamu odpovídají názvům Markdown souborů, ze kterých se čerpají výpisky pro výklad teorie. Podobně jako u tlačítka lze u výběrového seznamu využít metody *callback*, která se volá po zvolení některé z položek. Název zvolené položky je uložen do atributu, který je již předán do view obstarávajícího zobrazení výpisků.

Po výběru určitého tématu se tedy odešle zpráva obsahující view pro teorii, které si z Markdown souboru načte výpisky ke zvolenému tématu do interní proměnné. Z toho důvodu se změna obsahu v souborech s výpisky projeví až při dalším použití příkazu `/explain`. Text výpisků je podle nadpisů druhé úrovně rozdělen na jednotlivá podtémata, jejichž seznam je jediným obsahem úvodní zprávy. Tlačítka *Další*, *Předchozí* a nový výběrový seznam využívají metody ve view pro přepínání právě zobrazeného podtématu.

Metoda pro změnu podtémat nejprve zpracuje text patřící k aktuálně vybranému podtématu a rozdělí ho na jednotlivé zprávy, které jsou následně odeslány do textového kanálu. Na odeslané zprávy si view drží referenci. Jakmile je podtéma opět změněno a odešlou se zprávy nové, ty staré se smažou a reference nyní ukazuje na ty nové. Před smazáním starých zpráv se jako nová rodičovská zpráva daného view nastaví ta poslední odeslaná. Díky tomu zůstanou ovládací prvky vždy dole.

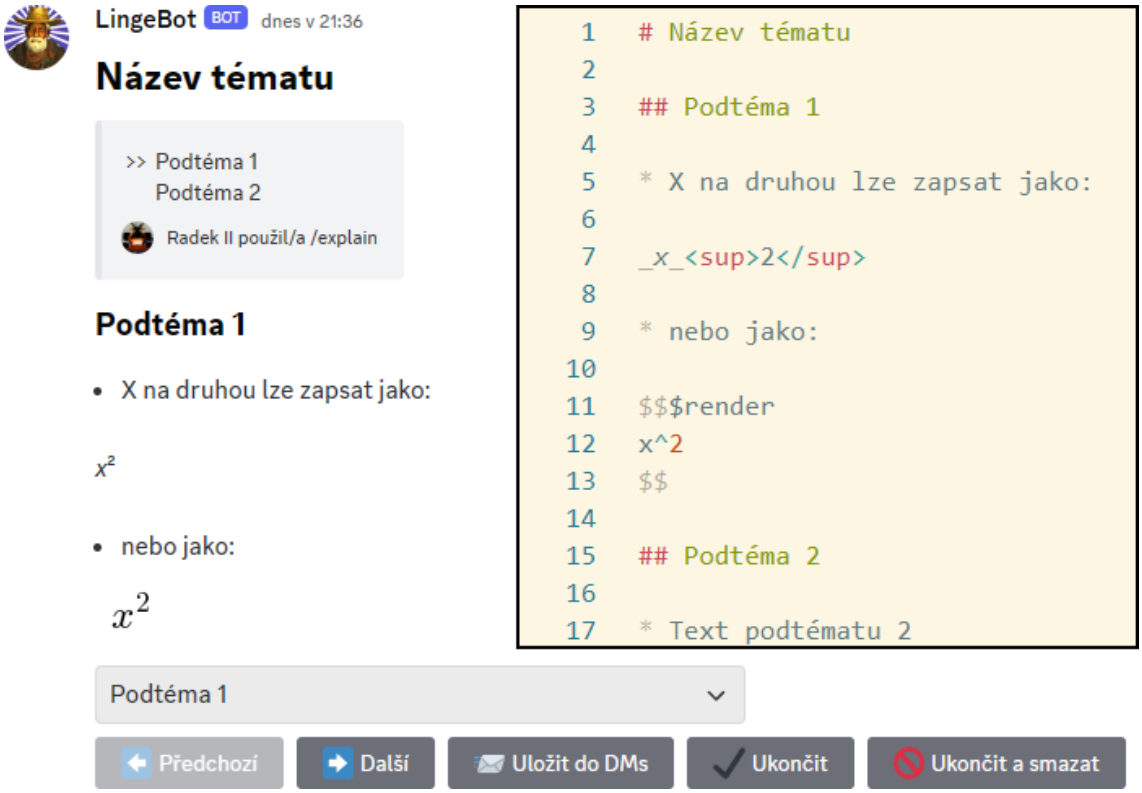
Odesílání a mazání více zpráv není okamžité. Discord API obsahuje endpoint pro instantní smazání většího množství zpráv, který je implementován v knihovně discord.py. Tato funkce ale během testování nefungovala spolehlivě a vybrané zprávy nebyly ve zdánlivě náhodných případech smazány. Zprávy navíc nebylo možné po použití této funkce smazat běžným způsobem jednu po druhé. Ekvivalentní endpoint pro odeslání více zpráv neexistuje, proto jsou zprávy odesílány i mazány po jednom. Pokud je zpráv více jak pět, byla mezi jednotlivé akce přidána krátká prodleva, aby se předešlo omezením proti spamu ze strany Discordu (*rate limiting*).

5.2.6 Zpracování Markdown souborů

Při tvorbě materiálů pro výuku matematiky je často potřeba použít určitý matematický symbol nebo výraz jako součást věty. Discord ale neumožňuje horizontálně vkládat obrázky mezi text, a proto je jedinou možností použít znaky Unicode. Pro usnadnění je možné v Markdown souborech využít HTML značek `sup` a `sub` pro psaní horních a dolních indexů. Protože Discord tyto značky nepodporuje, je jejich obsah převeden na Unicode znaky při rozdělování textu do jednotlivých podtémat.

Pro převod se používá Python balík *UnicodeIt*. Vytvořené zápisky pro lineární algebru často používají dolní indexy. Počet znaků, které existují v Unicode ve variantě dolního indexu, je ale omezený. Pokud kvůli neexistujícímu znaku nedokáže UnicodeIt obsah značky `sub` převést, je místo toho daný text vložen mezi zpětné uvozovky. Discord pak tuto část textu reprezentující dolní index zformátuje jako jednořádkový zdrojový kód: Písmo je neproporcionální, má menší velikost a tmavší pozadí. Text se díky tomu více odlišuje od zbytku a tváří se jako pomocný symbol.

Pro nahrazení textu umístěného mezi dvoudolary za obrázek s vykresleným matematickým výrazem je nejprve celé podtéma podle dvoudolarů rozděleno na části. Aby nebylo nutné detekovat, které části jsou určené pro vykreslení a které jsou obyčejný text, bylo přidáno pravidlo, že zprávy pro vykreslení musí začínat řetězcem `$render`. V takovém případě je řetězec `$render` zahozen a zbytek textu je vstupem vykreslovací funkce. Pravidla pro zápis jsou tedy stejná jako u příkazu `/render`.



Obrázek 5.7: Příkaz `/explain` – převod Markdown souboru do textového kanálu

Discord sice umožňuje, aby jediná zpráva najednou obsahovala text i obrázky, problémem ale je, že jsou všechny obrázky umístěny na konci zprávy pod veškerým textem. Proto je rozdělování na více zpráv jedinou možností, jak zajistit přesné umístění obrázků v odeslaném textu. Pokud zpráva obsahuje internetový odkaz, který míří na určitý obrázek, pak ho Discord automaticky zobrazí. Vepsáním takového odkazu mezi dvoudolary se opět zajistí správné umístění tohoto obrázku z internetu.

Bot automaticky rozděluje zprávy, které jsou delší než 2000 znaků, kvůli omezení ze strany Discordu. Využívá se Python modulu *textwrap*, díky kterému se rozdělení zprávy může nacházet pouze na místě mezery nebo nového řádku. Pokud by automatické rozdělení nebylo vhodně zvoleno, je možné opět použít dvoudolar, který značí, kde bude zpráva „manuálně“ rozdělena.

5.2.7 Implementace generace příkladů

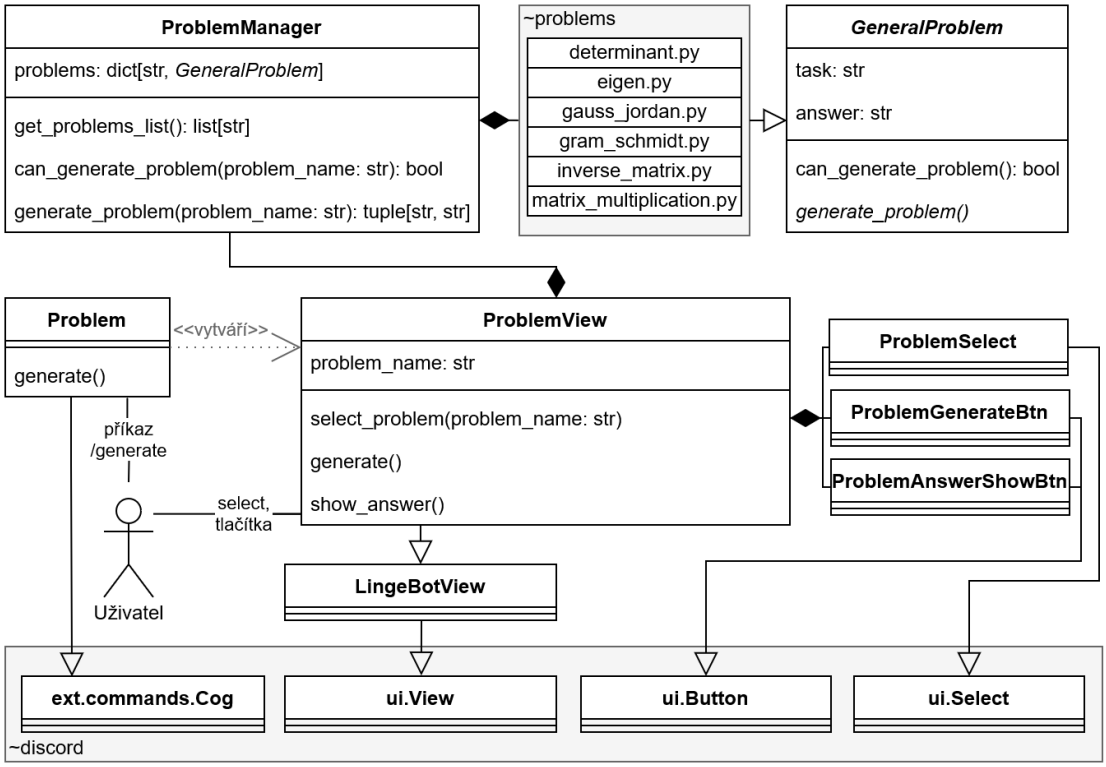
Implementace uživatelského rozhraní generátoru příkladů také využívá komplexnějšího view. Logika odesílání zpráv s návodem po stisknutí tlačítka *Jak počítat?* používá stejné funkce jako výklad teorie. Logika zobrazování příkladů a jejich výsledků se už liší.

Dokud se nevyužije možnosti zobrazení návodu, který může být tvořen více odeslanými zprávami, tak je rodičovská zpráva daného view stále stejná a její obsah je modifikován na základě stisknutých tlačítek. Tato zpráva obsahuje text a maximálně jeden obrázek. View má v sobě uložený řetězec se zadáním a řešením aktuálního příkladu. Po generaci nového příkladu je obsahem zprávy text zadání a po kliknutí na *Zobrazit výsledek* je obsah změněn na text řešení.

Obrázek obsažený v rodičovské zprávě je vykreslením matematického výrazu, jehož předpis je součástí řetězců se zadáním a řešením. Řetězec obsahuje nejprve textovou část zprávy a pokud jsou následně zapsány tři znaky dolaru, vše za nimi je považováno jako předpis výrazu pro vykreslení. Styl zápisu byl oproti Markdown souborům takto zjednodušen, protože zde není potřeba obsah rozdělovat do více zpráv a vykreslovat více obrázků.

Pro snadné implementování nových kategorií příkladů byla navržena struktura tříd, kterou popisuje obrázek 5.8. Abstraktní třída *GeneralProblem* obsahuje řetězce pro zadání a řešení příkladu. Pro jejich nastavení slouží metoda, která je definována v potomcích této třídy. Každý potomek reprezentuje jednu kategorii příkladů a nachází se ve vlastním souboru. Soubory jsou součástí balíku *problems* kvůli snadnějšímu importování. O poskytnutí řetězců k vybrané kategorii se stará *ProblemManager*, který obsahuje všechny potomky *GeneralProblem*. Tato vrchní trojice bloků je nezávislá na platformě Discord a společně s logikou pro vykreslování matematických výrazů by mohla být použita v jakékoliv Python aplikaci.

Spodní část diagramu je závislá na knihovně discord.py. Cog po zadání příkazu odešle zprávu s view, které obsahuje instanci *ProblemManager*. Z ní si view získá seznam dostupných kategorií pro naplnění výběrového seznamu a řetězce se zadáním a řešením, pokud je stisknuto příslušné tlačítko. View se také stará o načtení Markdown souboru s návodem pro výpočet, pokud nějaký existuje.



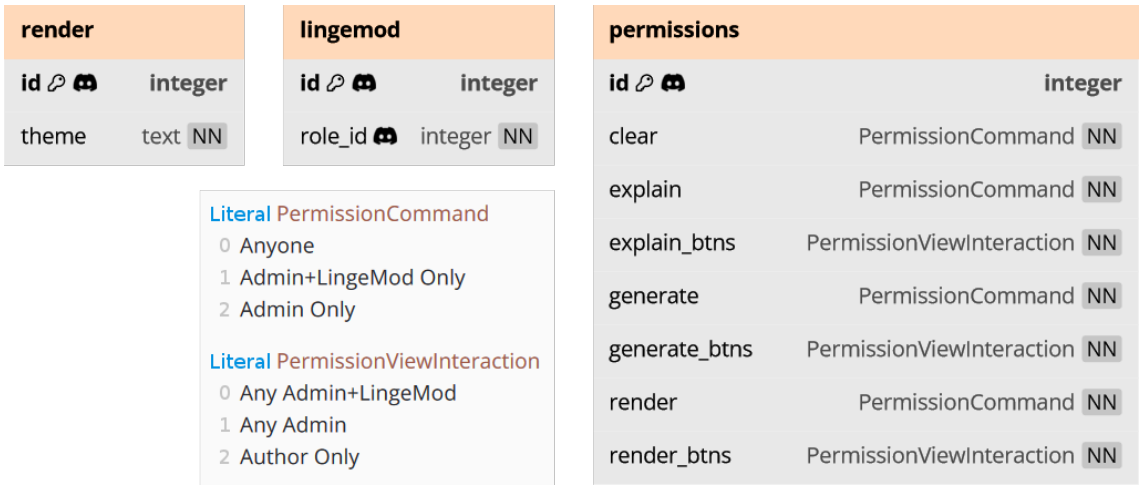
Obrázek 5.8: Diagram tříd generátoru příkladů

Do bota byly přidány generátory příkladů pro výpočet násobení matic, soustav rovnic, inverzních matic, vlastních vektorů a determinantu. Příklady pro výpočet soustavy rovnic obsahují tři neznámé a vedou na Gaussovu–Jordanovu eliminační metodu. Příklad tedy začíná maticí s rozměrem 3×3, která musí být převedena na jednotkovou, a tříprvkovým vektorem s výsledky. Pokud by byla počáteční matice vygenerována úplně náhodně, výsledek by pak pravděpodobně obsahoval desetinná čísla, což není pro výukové materiály vhodné. Proto je nejdříve vytvořena jednotková matice, na které je provedeno několik náhodných gaussovských operací, které nemění její lineární obal. Pro snazší kontrolu obsahuje řetězec s řešením kromě výsledku i mezikrok výpočtu, ve kterém je matice v horním stupňovitém tvaru. Pro tento účel je použita metoda LU rozkladu bez zlomků z balíku SymPy. Po použití této metody je ještě každý řádek matice vydělen jeho největším společným dělitelem. Tato podoba matice se totiž nejvíce podobá té, ke které se uživatel dopracuje při ručním výpočtu Gaussovy eliminační metody.

Další kategorií příkladů, kde není vhodné generovat zadání úplně náhodně, je výpočet inverzní matice. V tomto případě je potřeba vygenerovat celočíselnou matici, jejíž inverze je také tvořena celými čísly. Tento požadavek splňují matice, které mají determinant roven ±1. Determinant matice v horním (nebo dolním) stupňovitém tvaru je roven součinu jejích diagonálních prvků a determinant součinu dvou čtvercových matic stejné velikosti je roven součinu determinantů těchto dvou matic [9]. Náhodnou matici, která má determinant roven jedné, tedy lze vytvořit součinem

dvou matic, kde jedna je v horním a druhá v dolním stupňovitém tvaru. Obě matice mají na diagonále samé jedničky, aby byl jejich determinant roven jedné, a zbylé prvky jsou zvoleny náhodně.

5.2.8 Databáze



Obrázek 5.9: Schéma databáze

Požadavky bota na databázi nejsou vysoké. Je potřeba ukládat nastavení barevného schématu pro vykreslování matematických výrazů, pokud jej uživatel nebo administrátor změní. Dále je pro každý server, na kterém se bot nachází, ukládán identifikátor moderátorské role *LingeMod*. Pokud by bot detekoval roli pomocí jejího názvu a role by byla přejmenována, přestala by pak moderace fungovat. Poslední tabulka uchovává nastavení oprávnění pro jednotlivé servery.

Pro takovýto účel byla zvolena databáze SQLite. Mnoho neoficiálních tutoriálů pro tvorbu Discord botů používá jako databázi přímý zápis a čtení ze souborů JSON. Tento způsob je nevhodný, protože není zaručena žádná bezpečnost transakcí a veškerá režie je zanechána na programátorovi. Pokus o implementaci vlastní režie by vedl k naprogramování vlastního databázového systému, proto je vhodnější obrátit se na již hotová a ověřená řešení. SQLite je embedded databáze, která je pro potřeby tohoto bota dostačující a kterou jazyk Python nativně podporuje.

Identifikátory serverů, rolí a uživatelů jsou spravovány na straně Discordu. Každá funkce v knihovně discord.py, která reaguje např. na událost zadání příkazu uživatelem, má jako první vstupní parametr objekt typu interakce (*discord.Interaction*), ze kterého lze zjistit ID daného uživatele a serveru, kde byl příkaz zadán.

Pokud např. uživatel zadá příkaz pro změnu barevného schématu, tak je ID daného serveru získáno z objektu interakce a v tabulce je k němu přiřazen název zvoleného schématu. Pokud je příkaz zadán v přímých zprávách, pak je místo ID serveru, které v tomto kontextu neexistuje, použito ID uživatele. Všechny Discord identifikátory by měly být unikátní, a proto nejsou potřeba dvě tabulky pro schémata serveru a schémata uživatelů. Při použití příkazu `/render` je pak z interakce opět

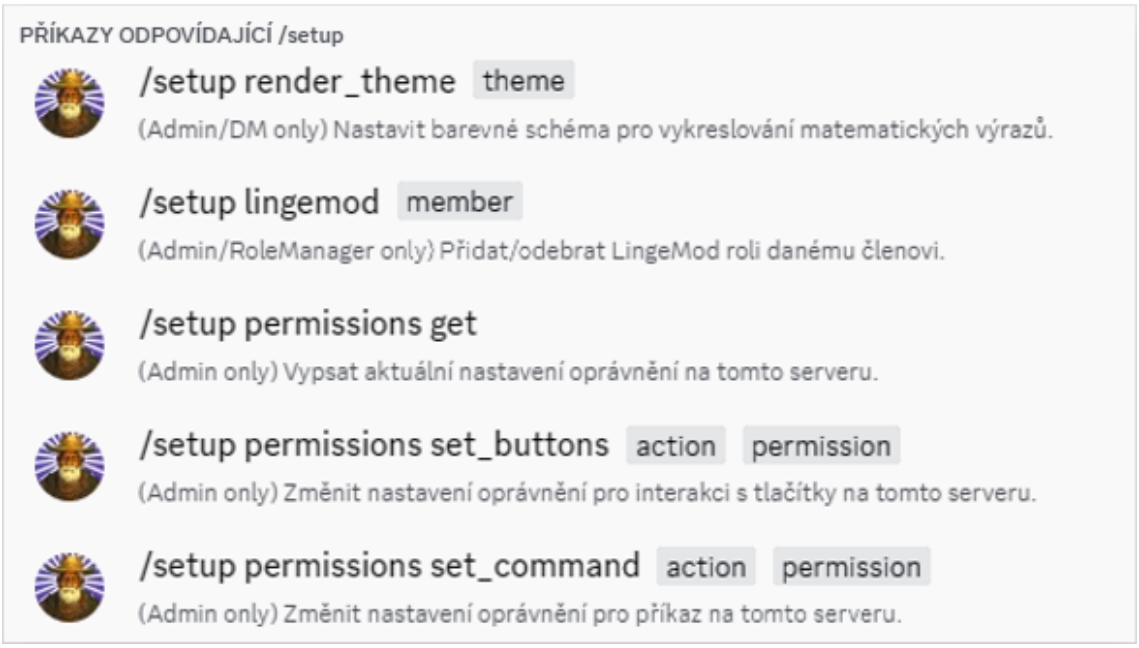
zjištěno ID, pomocí něhož se v databázi dohledá nastavené barevné schéma, které se použije pro vykreslení.

Identifikátory jsou tedy podle potřeby získávány z Discord API. V případě tohoto bota proto může být databáze jednoduchá a tabulky mezi sebou nemusí mít žádné vazby. Definice databáze ze strany SQL obsahuje pouze názvy tabulek a jejich sloupců. Ostatní logika se odehrává na straně programu v jazyce Python.

Pro reprezentaci hodnot uchovávajících barevné schéma a oprávnění se používá typ *Literal* z modulu *typing*, který může nabývat jen předem definovaných hodnot. Literal je podporován knihovnou discord.py a pokud je nastaven jako datový typ parametru příkazu, pak Discord uživateli povolí příkaz odeslat jen tehdy, když je zadaný parametr roven některé z předem definovaných hodnot. K těmto hodnotám je navíc dostupné našeptávání.

Typ *ThemeLiteral* může nabývat čtyř různých řetězců, které reprezentují jednotlivá barevná schémata. Tyto řetězce jsou přímo ukládány do databáze k příslušnému ID. U nastavení oprávnění existují dva typy: jeden pro zadání příkazu a druhý pro interakci s tlačítky a výběrovými seznamy. Tyto typy jsou před uložením do databáze převedeny na čísla 0, 1, nebo 2 (viz obrázek 5.9). SQLite nepodporuje výčtové typy, a proto se tato konverze opět odehrává na straně Python programu.

5.2.9 Implementace moderace



Obrázek 5.10: Seznam příkazů pro nastavení

Příkazy `/setup` jsou součástí tzv. skupiny podpůrných příkazů. Běžný příkaz je zapsán pomocí lomítka, po němž následuje název příkazu, který musí být jednoslovný. Příkazy ze skupiny se zadávají ve tvaru `/<název_skupiny> <název_příkazu>`. Díky tomu mohou mít podobné příkazy stejný začátek a seznam příkazů je tak

přehlednější. Skupiny příkazů je možné vnořovat, toho využívají příkazy `/setup permissions`.

Spuštění jednoho z příkazů nastavujících oprávnění změní v tabulce *permissions* na řádku daného serveru příslušný atribut. Záznam v tabulce není přidán hned po připojení bota na server, ale až po prvním spuštění nastavujícího příkazu. Dokud záznam neexistuje, využívá se výchozích hodnot definovaných ve zdrojovém kódu. První volání příkazu nejprve přidá řádek s těmito výchozími hodnotami do tabulky a pak změní vybraný atribut.

Při každém pokusu o spuštění příkazu nebo interakci s view jako první proběhne kontrola oprávnění. Funkce obstarávající tuto kontrolu přistupuje do databáze a má k dispozici objekt interakce, ze kterého lze zjistit údaje o daném uživateli. Pokud má uživatel dostatečná oprávnění, funkce proběhne bez efektu a obsluha jeho interakce může pokračovat. Při nedostatečných právech je vyhozena výjimka, na kterou reaguje cog *on_error*. V tomto případě je do chatu odeslána zpráva informující o nedostatečných právech.

Implementace funkce kontroly minimalizuje počet přístupů do databáze. Administrátorům je např. vždy povoleno použít jakýkoliv příkaz nehledě na nastavení oprávnění daného serveru. Nastavení je z databáze získáno až poté, co je z objektu interakce zjištěno, že uživatel administrátorem není. Identifikátor role *LingeMod* je z databáze získán až v posledním případě, kdy je zjištěno, že příkaz není dostupný všem.

5.3 Nasazení a zpětná vazba

Po implementaci všech navrhovaných funkcí byl bot nasazen do testovacího provozu a zpřístupněn skupině uživatelů. Vybranou skupinou byly studenti TUL, kteří měli v té době zapsaný předmět zaměřený na lineární algebru. Součástí testovacího provozu bylo dotazníkové šetření, kterého se zúčastnilo celkem devět respondentů. Výsledek dotazníku se nachází v příloze C.

Všichni respondenti již měli na platformě Discord založený účet a někteří z nich měli zkušenosti i s platformami Slack a Teams. Šest respondentů v minulosti již použilo nějakého Discord bota s tím, že nejčastější kategorií jsou boti přehrávající hudbu v hlasovém hovoru. Žádnému z respondentů se nestalo, že by měl problém s pochopením ovládání bota, avšak čtyřem z nich nepřišlo ovládání příliš intuitivní. Kvalita vytvořených teoretických materiálů byla ohodnocena jako velmi dobrá. Většina respondentů by tento typ obsahu raději konzumovala jiným způsobem, avšak názor, že prostředí platformy Discord s sebou nese určité výhody, je také většinový.

5.3.1 Návrhy na vylepšení

Pro efektivnější procházení teoretických výpisků u příkazu `/explain` by bylo vhodné přidat funkci vyhledávání. Výběrový seznam teoretických témat obsahuje pouze jejich názvy a nemusí být jasné, co vše je v nich obsaženo. Vyhledávání by šlo implementovat buď pomocí textového pole, nebo nového příkazu, kde by jediným

parametrem byl řetězec pro vyhledání. Interaktivní prvek textového pole nemůže být součástí běžného view a lze použít pouze ve vyskakovacím okně. To by se vyvolalo buď příkazem, nebo reakcí na stisk tlačítka. Po zadání hledaného výrazu by byla uživateli vypsána všechna (pod)témata, kde se daný výraz nachází. Zobrazení výňatků z výpisků se zvýrazněním hledaných slov by už bylo složitější, protože podpora barevného písma na Discordu je velmi omezená a formátování např. na tučné písmo by mohlo rozbít již přítomný Markdown zápis. Také by bylo vhodné přidat intuitivní způsob, kterým by si uživatel mohl rychle zobrazit vybrané podtéma ze seznamu výsledků vyhledávání. Výběrový seznam se jeví jako nejvhodnější, protože může mít proměnný počet položek.

U příkazu `/generate` lze vždy generovat příklady pouze z jedné kategorie. Pro změnu je potřeba stisknout příslušné tlačítko a následně zvolit jinou kategorii z výběrového seznamu. Pro vylepšení by šla přidat funkce, která by vygenerovala sadu příkladů napříč více kategoriemi. Příklady by se uživateli ukazovaly buď postupně, nebo všechny najednou. Případně by mohl být přidán časový limit, během kterého by se uživatel musel pokusit stihnout všechny příklady vypočítat.

Při tvorbě teoretických výpisků a návodů pro počítání příkladů je možné používat pouze obrázky z internetu. Pro flexibilnější tvorbu by bylo vhodné přidat podporu vkládání obrázků z disku. Pravidla pro zápis vložení obrázku by ideálně měla dodržovat syntaxe jazyka Markdown, aby při tvorbě zápisků v libovolném editoru fungoval náhled i s obrázky. Cesty k obrázkům, které jsou relativní k příslušným Markdown souborům, by případně musely být převedeny na cesty relativní ke kořenu projektu, aby je mohla použít knihovna discord.py.

Z pohledu zdrojového kódu je nedostatkem způsob práce s databází. Potřeby bota pro databázi jsou minimální a z důvodu záměru zachování této jednoduchosti nebylo žádoucí, aby kvůli ní musela běžet další aplikace. Vzhledem ke struktuře ukládaných dat by se k tomuto projektu více hodila dokumentová NoSQL databáze. Žádné zavedené embedded NoSQL řešení ale nebylo nalezeno, a proto byla zvolena databáze SQLite. S tou se zde pracuje přímo pomocí Python modulu *sqlite3* a její schéma je tvořeno „ručně“ příkazy typu `CREATE TABLE`. Vhodnější by bylo použít balík *SQLAlchemy*, který by usnadnil modifikaci stávajícího schématu a případný přechod na alternativní databázový systém.

Jak je zmíněno v 5.2.2, funkce pro aproximaci délky vykreslených matematických výrazů podporuje pouze podmnožinu všech možných vstupů. Pro její vylepšení by se musel druhý krok rozšířit o rozvětvení, které by explicitně reagovalo na určité TeX příkazy. Např. `\sin` by nahradil řetězcem o délce tří a `\sinh` o délce čtyř znaků.

5.3.2 Vhodnost platformy Discord

Discord je platforma s početnou komunitou uživatelů, která disponuje komplexním API pro tvorbu botů. Většina populárních botů se zaměřuje na servery s velkým počtem členů a nabízí usnadnění moderace, nebo přidává nové sociální funkce. Díky možnosti vytvoření soukromých serverů a komunikace s botem v přímých zprávách lze ale na platformě Discord nasadit řešení pro jakoukoliv cílovou skupinu.

Vzdělávání i matematika mají na Discordu své místo, botů zaměřených na výuku je ale málo. Jedním z důvodů jsou pravděpodobně přirozená omezení vycházející z prostředí textového chatu. Kromě hlasového hovoru je odeslaná zpráva jediný způsob, jakým může bot uživateli předat určitou informaci. K dispozici jsou sice interaktivní prvky jako tlačítka, která lze ke zprávě přidat, oproti webovým a desktopovým aplikacím jsou ale možnosti pro tvorbu rozhraní značně omezené. Jednou z výzev při tvorbě bota v této práci tedy bylo vymyšlení co nejpřirozenějšího rozhraní, které za pomoci převážně odesílání, upravování a mazání zpráv dokáže uživateli zprostředkovat dané matematické materiály.

V tomto ohledu není Discord nejvhodnějším prostředím pro nasazení takovéto aplikace. Na druhou stranu se jedná o platformu populární mezi studenty a i všichni oslovení testeři zde již měli založený účet. Díky systému podpůrných příkazů a prostředkům pro tvorbu rozhraní, které jsou sice omezené, ale jednotné, by pro uživatele mělo být snazší pochopit ovládání libovolného bota.

Discord může sloužit i jako platforma pro online výuku ve školách. Vyučující by v tomto případě do bota vložil své teoretické materiály, ze kterých by pak studenti mohli čerpat a při neporozumění určitému tématu začít diskuzi v textovém kanálu. K lepšímu vyjádření pak slouží příkaz `/render`, díky kterému není nutné pro potřebu odeslání matematického výrazu opustit okno chatu a použít externí aplikaci. Vyučující by také mohl chtít bota použít jako doprovod ke svému výkladu v hlasovém hovoru, pak by se např. nastavilo oprávnění, že příkaz `/explain` mohou používat pouze administrátoři.

Výhodou Discordu je jeho dostupnost na všech hlavních desktopových i mobilních operačních systémech a také ve webové aplikaci. Uživatelovy konverzace jsou okamžitě synchronizovány na všech jeho zařízeních. Pro uživatele, kteří Discord nepoužívají, může být odrazující nutnost zakládat si kvůli použití bota účet.

Nevýhodou Discord botů je jejich závislost na této platformě. Bot, který využívá specifických funkcí Discordu, nepůjde snadno přesunout na jinou platformu. Discord je proprietární a nelze provozovat jeho vlastní instanci. Pokud je platforma z důvodu výpadku nedostupná, pak je bot také nedostupný.

Dva měsíce po nasazení bota přestal Discord interpretovat text ohraničený podtržítky jako kurzívu, pokud se přímo za ním nacházela otevírající závorka. Zpráva napsaná jako `_f_(_x_)` se tedy místo „$f(x)$“ nyní zobrazuje jako „_f_(x)“. Kvůli tomu musela být všechna podtržítka v teoretických materiálech u těchto případů nahrazena hvězdičkami. Pro Discord vychází aktualizace minimálně jednou za měsíc a instalují se automaticky. Takovéto změny mohou nečekaně změnit chování bota.

6 Závěr

V rámci této bakalářské práce byla provedena rešerše sociálních platforem, které oficiálně podporují integraci botů. Následně byla zanalyzována skupina existujících Discord botů, kteří se alespoň částečně zabývají tématem matematiky. Poslední část rešerše zanalyzovala knihovny, které usnadňují tvorbu botů pro platformu Discord.

V praktické části práce byl vytvořen matematický bot zvaný *LingeBot*. Ten byl implementován v jazyce Python za pomoci knihovny discord.py. Jeho třemi hlavními funkcemi jsou vykreslování matematických výrazů, výklad teorie a generování příkladů z lineární algebry. Veškeré funkce bota probíhají v prostředí textového chatu a lze je spouštět pomocí tzv. podpůrných příkazů, které začínají lomítkem.

Příkaz `/render` vloží do textového kanálu obrázek s vykresleným matematickým výrazem, který odpovídá uživatelskému vstupu. Formát vstupního řetězce je podobný jazyku TeX a pro vykreslení se využívá balík Matplotlib, jehož funkce jsou ale v tomto ohledu omezené. Bot byl proto rozšířen o vlastní funkce pro vykreslování matic libovolných rozměrů. Matice mohou být součástí delších rovnic a rovnice lze rozdělit na více řádků s případným nastavením zarovnání.

Příkaz `/explain` poskytuje rozhraní výkladu teorie. Pro účely bota byly vytvořeny materiály z lineární algebry pokrývající především témata vektorových prostorů a lineárních zobrazení mezi nimi, obecně je ale podporován jakýkoliv typ matematických materiálů. Jednotlivá teoretická témata jsou psaná v jazyce Markdown a uložená v samostatných souborech. Bot pak formátovaný text z těchto souborů, který může obsahovat i vykreslené matematické výrazy, odesílá do textového kanálu. Soubory lze upravovat bez nutnosti restartování bota.

Příkaz `/generate` poskytuje rozhraní pro generaci příkladů. Uživatel si může nechat vygenerovat příklad z vybrané kategorie a následně si zobrazit výsledek s případným postupem řešení. Při implementaci byl kladen důraz na snadnou rozšiřitelnost mechanismu o nové kategorie příkladů.

Pro demonstraci specifických funkcí platformy Discord byly přidány možnosti moderace. Bot po připojení na server vytvoří moderátorskou roli zvanou *LingeMod*, díky které může administrátor rozdělit členy serveru na dvě skupiny. Pro jednotlivé příkazy pak lze nastavit, jaká oprávnění musí uživatel mít, aby je mohl na daném serveru použít. Rozhraní výkladu teorie a generátoru příkladů pak pro své ovládání využívá interaktivních prvků jako tlačítek a výběrových seznamů.

Bot byl nasazen do testovacího provozu pro vybranou skupinu uživatelů, kteří se zúčastnili dotazníkového šetření. Všichni respondenti již měli na platformě Discord založený účet. Nikdo neměl problém s pochopením ovládání bota, avšak v některých případech respondenti nepovažovali uživatelské rozhraní v prostředí textového chatu

za příliš intuitivní. I přesto, že by většina respondentů preferovala pro tento typ obsahu jiné médium, většinový byl také názor, že výuka na platformě Discord má své výhody. Kvalita ukázkových teoretických materiálů byla ohodnocena jako velmi dobrá.

Funkce výkladu teorie by šla vylepšit přidáním fulltextového vyhledávání. Při tvorbě teoretických materiálů v podobě Markdown souborů by byla vhodná možnost vkládání obrázků z disku. U generování příkladů by šla přidat funkce, která by vygenerovala sadu příkazů z různých kategorií najednou. Z pohledu zdrojového kódu by pak bylo vhodné použít pro práci s databází balík *SQLAlchemy*, který by usnadnil případné rozšíření databáze nebo přechod na jiný databázový systém.

Při vývoji bota pro platformu Discord je výzvou poskytnutí intuitivního uživatelského rozhraní pouze prostřednictvím interaktivních zpráv. Zároveň je dostupnost bota závislá na dostupnosti platformy. Na druhou stranu má Discord velkou komunitu uživatelů, kterou lze pomocí bota oslovit. Rozhraní pro ovládání všech botů je omezené, ale jednotné a tím pádem snadněji uchopitelné. Výhodou platformy je také její dostupnost na všech hlavních desktopových i mobilních operačních systémech a ve webové aplikaci.

Tato práce tedy nejprve v rámci rešerše představila obecný koncept botů na Discordu a dalších sociálních platformách. Praktická část pak přiblížila konkrétní problematiku tvorby botů pro platformu Discord od návrhu matematického bota až po jeho nasazení. Následovalo vyhodnocení zpětné vazby od uživatelů a navržení případných vylepšení. Nakonec byly shrnuty výhody a nevýhody provozu řešení na této platformě.

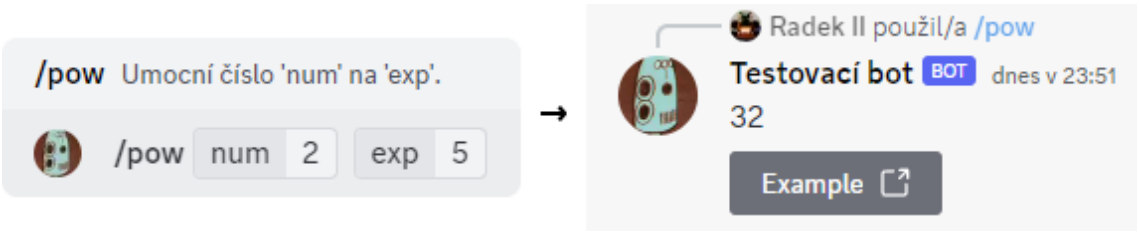
Seznam použité literatury

1. MORRIS, Tee. *Discord For Dummies*. Hoboken: Wiley, 2020. ISBN 978-1-119--68803-7.
2. DISCORD INC. *Discord Official API Documentation* [online]. [2016]. [cit. 2024-02-01]. Dostupné z: https://discord.com/developers/docs.
3. DISCORD INC. *Developing on Discord: Starter Packet* [online]. [2022]. [cit. 2024-02-02]. Dostupné z: https://assets-global.website-files.com/6257adef93867e50d84d30e2/6527e5dee6b6cd26313396d4_Developing%20on%20Discord_Starter%20Pack.pdf.
4. MATNEY, Lucas. *Roblox acquires Discord competitor Guilded* [online]. 2021-08. [cit. 2024-02-05]. Dostupné z: https://techcrunch.com/2021/08/16/roblox-acquires-discord-competitor-guilded/.
5. MATRIX.ORG FOUNDATION. *Matrix.org - Documentation* [online]. [2015]. [cit. 2024-02-08]. Dostupné z: https://matrix.org/docs.
6. VAN STEEN, Maarten a TANENBAUM, Andrew Stuart. *Distributed Systems*. Fourth Edition. van Steen, 2023. ISBN 978-90-815406-4-3.
7. VITILLO, Roberto. *Understanding Distributed Systems*. Vitillo, 2021. ISBN 978--1838430207.
8. COULOURIS, George; DOLLIMORE, Jean; KINDBERG, Tim a BLAIR, Gordon. *Distributed Systems: Concepts and Design*. Fifth Edition. Boston: Pearson, 2011. ISBN 978-0-13-214301-1.
9. BEČVÁŘ, Jindřich. *Lineární algebra*. Vydání páté. Praha: MatfyzPress, 2019. ISBN 978-80-7378-378-5.

A Odkazy na zdrojové kódy a manuál

- Zdrojové kódy bota a této zprávy jsou dostupné na:
  https://github.com/RadekMocek/BP
- Manuál pro použití bota je dostupný na:
  https://radekmocek.github.io/PDO
- Hlavní instanci bota lze pozvat na server pomocí odkazu:
  https://discord.com/oauth2/authorize
  ?client_id=1160906076179415070
  &permissions=2416045056
  &scope=bot+applications.commands

B Implementace ukázkového příkazu



Obrázek B.1: Ukázkový příkaz – vstup a výstup

```python
import discord

@discord.app_commands.command()
async def pow(self,
              itx: discord.Interaction,
              num: int,
              exp: int) -> None:
    """Umocní číslo 'num' na 'exp'."""
    view = discord.ui.View()
    view.add_item(discord.ui.Button(label="Example",
                                    url="https://example.com"))
    await itx.response.send_message(content=num ** exp,
                                    view=view)
```

Obrázek B.2: Ukázkový příkaz – implementace v discord.py

```javascript
const {
    ActionRowBuilder, ButtonBuilder, ButtonStyle, SlashCommandBuilder
} = require("discord.js");

module.exports = {
    data: new SlashCommandBuilder()
        .setName("pow")
        .setDescription("Umocní číslo 'num' na 'exp'.")
        .addIntegerOption(op => op.setName("num").setRequired(true))
        .addIntegerOption(op => op.setName("exp").setRequired(true)),
    async execute(interaction) {
        const num = interaction.options.getInteger("num");
        const exp = interaction.options.getInteger("exp");
        const button = new ButtonBuilder()
            .setLabel("Example")
            .setURL("https://example.com")
            .setStyle(ButtonStyle.Link);
        const row = new ActionRowBuilder().addComponents(button);
        await interaction.reply({
            content: "" + num ** exp, components: [row]
        });
    },
};
```

Obrázek B.3: Ukázkový příkaz – implementace v discord.js

```
using Discord;
using Discord.WebSocket;

private async Task SlashCommandHandler(SocketSlashCommand command) {
    if (command.Data.Name == "pow") {
        var num = (Int64)command.Data.Options.ElementAt(0).Value;
        var exp = (Int64)command.Data.Options.ElementAt(1).Value;
        var builder = new ComponentBuilder().WithButton("Example", ↩
            style: ButtonStyle.Link, url: "https://example.com");
        await command.RespondAsync("" + Math.Pow(num, exp), ↩
            components: builder.Build());
    }
}
```

Obrázek B.4: Ukázkový příkaz – implementace v Discord.Net

```
import net.dv8tion.jda.api.events.interaction.command.↩
    SlashCommandInteractionEvent;
import net.dv8tion.jda.api.hooks.ListenerAdapter;
import net.dv8tion.jda.api.interactions.components.buttons.Button;

public void onSlashCommandInteraction(SlashCommandInteractionEvent ↩
    event) {
    if (event.getName().equals("pow")) {
        int num = event.getOption("num").getAsInt();
        int exp = event.getOption("exp").getAsInt();
        event.reply("" + Math.pow(num, exp)).addActionRow(Button.link↩
            ("https://example.com", "Example")).queue();
    }
}
```

Obrázek B.5: Ukázkový příkaz – implementace v JDA

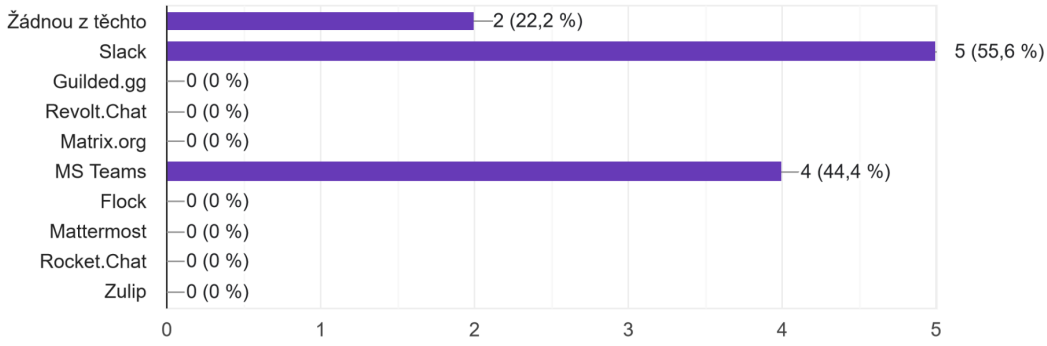
C Výsledky dotazníkového šetření

Jak často používáte Discord?

9 odpovědí

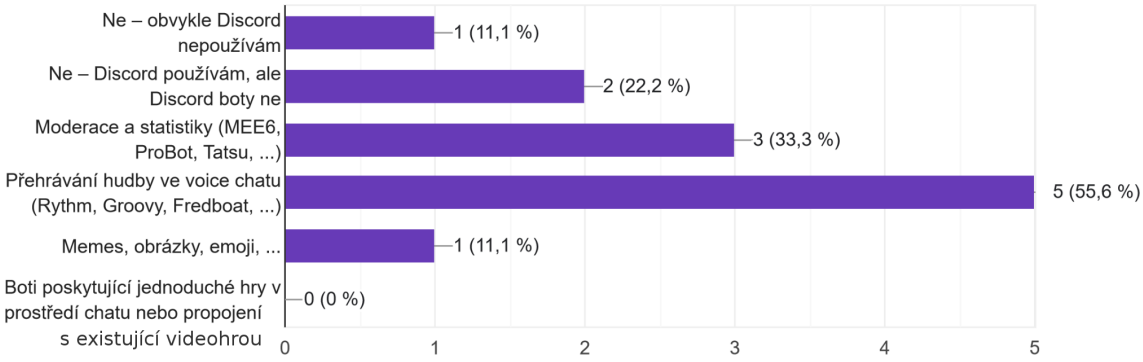
Používáte nějakou podobnou sociální platformu (alespoň příležitostně)?

9 odpovědí



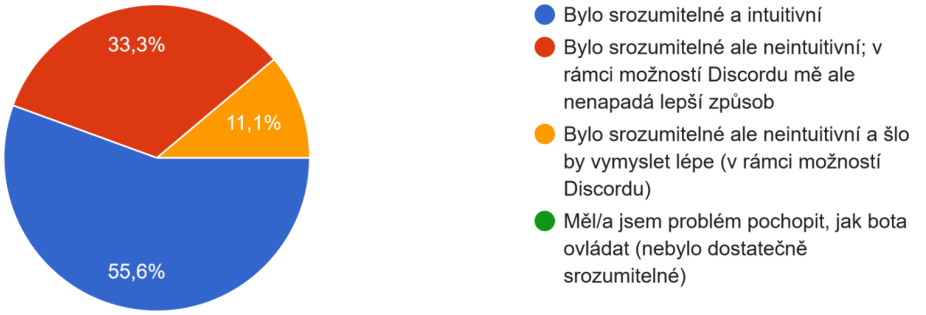
Používal/a jste v minulosti nějakého jiného Discord bota?

9 odpovědí



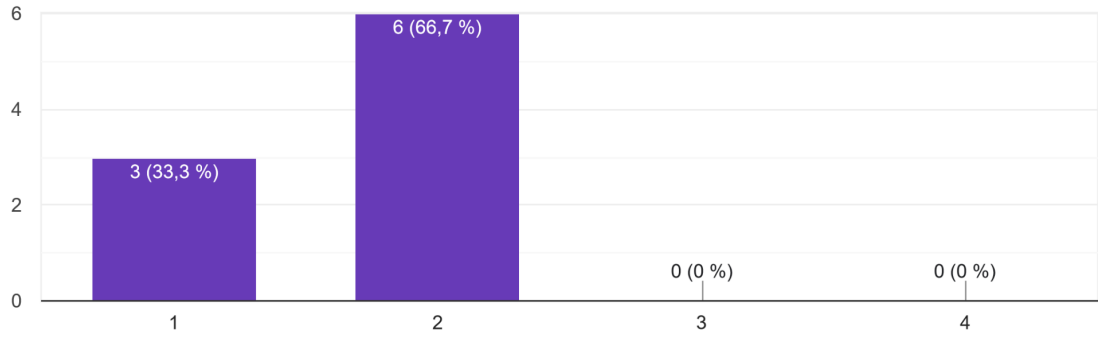
Bylo pro vás ovládání bota (konkrétní implementace příkazů a tlačítek) intuitivní/srozumitelné?

9 odpovědí



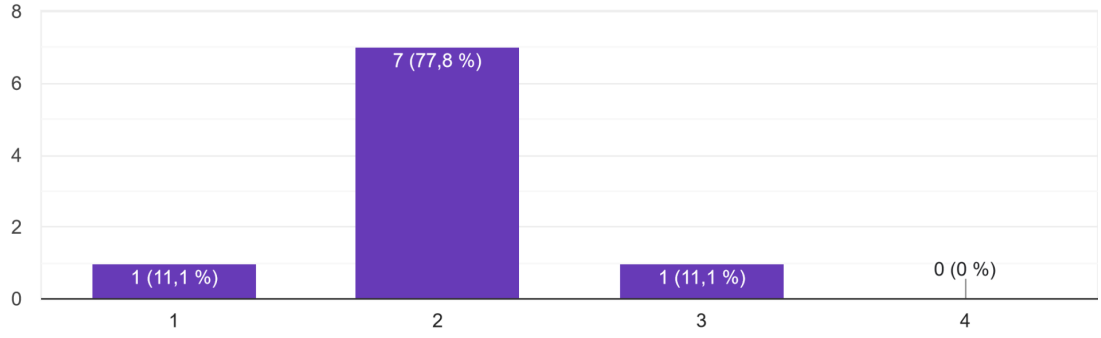
Jaká byla podle vás celková kvalita obsahu teoretických materiálů

9 odpovědí



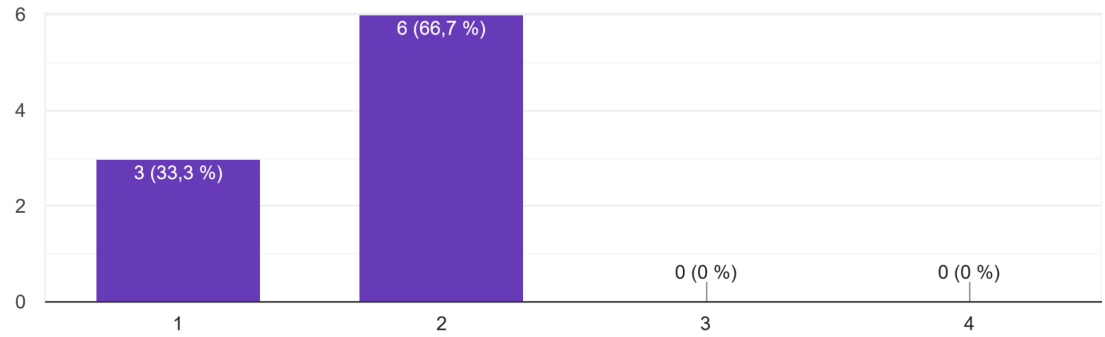
Teorie konkrétně: Robustnost výpisků (jak dostatečně byla daná témata pokryta)

9 odpovědí



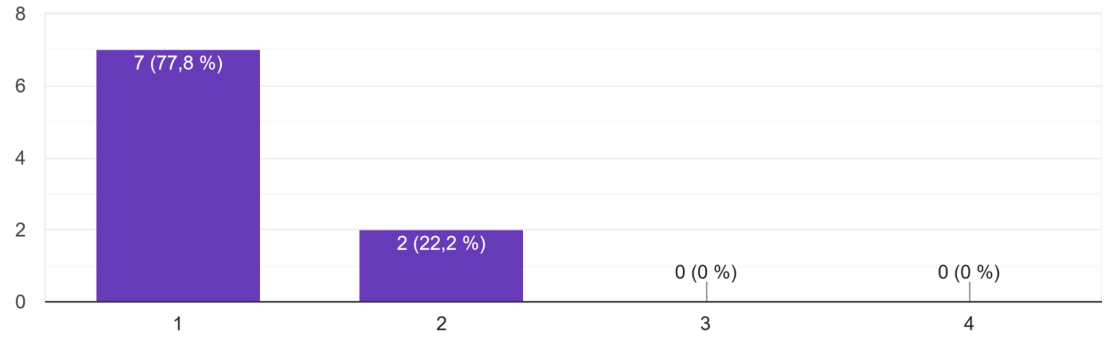
Teorie konkrétně: Srozumitelnost výpisků (jak srozumitelný byl obsah výpisků)

9 odpovědí



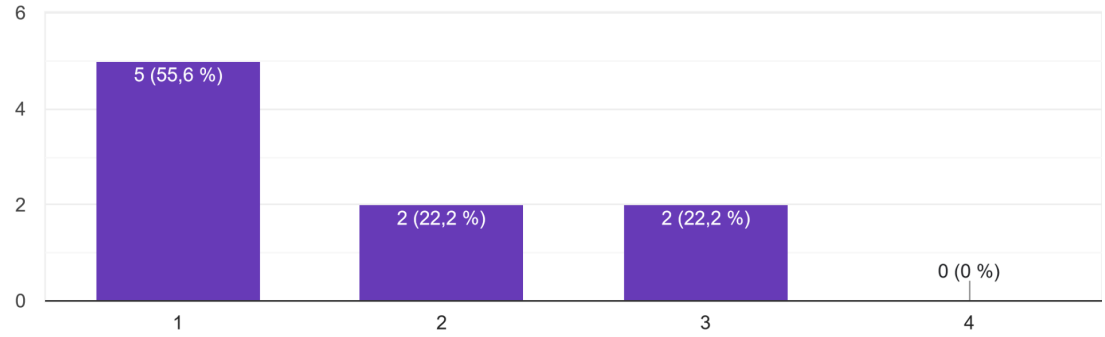
Teorie konkrétně: Prezentace výpisků (pravopis, gramatika, typografie, ...)

9 odpovědí



Teorie konkrétně: Čitelnost výpisků z pohledu Discordu (jak dobře se vám obsah konzumoval z prostředí textového chatu)

9 odpovědí



Vidíte nějakou výhodu v takovéto výuce přes Discord bota?

9 odpovědí