



UNIVERZITET U NOVOM SADU
PRIRODNO-MATEMATIČKI
FAKULTET
DEPARTMAN ZA MATEMATIKU
I INFORMATIKU

Smerač

- seminarski rad iz predmeta Skript jezici -

Aleksa Siriški, 159/22
Novi Sad, 2022.

Sadržaj

Sadržaj	2
1 Uvod	3
1.1 Discord	3
1.2 Python	3
1.3 Ideja	3
1.4 Asinhronizacija	3
1.5 Smerovi	3
1.6 Kalendar	4
2 Opis programa	5
2.1 Nagli prekid usled greške	5
2.2 Promenljive okruženja	5
2.3 Logging	6
2.4 Globalne promenljive	6
2.5 Slash komande	7
2.5.1 Kreiranje slash komande	7
2.5.2 Dodeljivanje funkcionalnosti slash komandi	8
2.6 Kalendar	9
2.6.1 Instanciranje niti	9
2.6.2 Učitavanje kalendara	10
2.6.3 Generisanje sedmice	11
2.6.4 Parsiranje sedmice	12
2.7 Kreiranje grafa	13
3 Zaključak	14
Literatura	15

1 Uvod

1.1 Discord

Discord[1], kao jedna od najpopularnijih društvenih mreža za programere i entuzijaste računarskih tehnologija, je logičan izbor za razmenu informacija i raspoređivanje časova IT i RN smerova Univerziteta u Novom Sadu. Uprkos jednostavnosti Discord-ovog grafičkog interfejsa nije lako omogućiti studentima da sami sebi određuju smer a direktna integracija sa Google kalendarom je apsolutno neizvodljiva. Na sreću, Discord tim je osposobio trećim licima da kreiraju 'Bot' naloge i uz pomoć njihovog REST API će nastati Smerač - Discord Bot stvoren da omogući studentima izbor sopstvenog smer a kao i integraciju Google kalendara.

1.2 Python

Smerač je Python[2] skripta napisana u svega 300 linija koda. Računajući da korisnicima nije bitno da li će im smer biti dodeljen za 1ms ili 1s Python je bio prikladan izbor. Najpre zbog mnogobrojnih ugrađenih biblioteka i modula, kao što su `interactions-py`[3] i `asyncio`[4], ali i zbog jednostavne sintakse koja je omogućila eksponencijalan razvoj ovog programa.

1.3 Ideja

Discord bot koji mora biti dovoljno jednostavan ali isto tako i univerzalno programiran da se može lako primeniti na više različitih okruženja (Discord servera). Takva prilagodljivost se postiže pametnim planiranjem toka rada programa i korišćenjem promenljivih okruženja (eng. *environment variables*). Takođe je jako bitna licenca koja će se koristiti, da ne bi došlo do krađe autorskih prava i zloupotrebe softvera. Tačno iz tih razloga odabrana licenca za Smerač će biti GNU General Public License[5].

1.4 Asinhronizacija

Kako bi program mogao da istovremeno čita poruke od različitih korisnika potrebno je kreiranje dodatnih niti. Pomoću biblioteke `asyncio` Smerač će imati mogućnost paralelnog parsiranja kalendara i dodeljivanja smer a svakom studentu koji to zatraži. Pored bržeg izvođenja programa asinhronne funkcije služe da oduzmu čekanje odgovora od servera, tj. ako zahtev prvog korisnika traje duže da se obradi drugi korisnici ne moraju da čekaju nego se i njihovi zahtevi obrađuju u istom trenutku.

1.5 Smerovi

Najbitnija i najjednostavnija funkcionalnost Smerača će biti dodeljivanje smer a studentima. Program će izvršavati korisničke komande oblika:

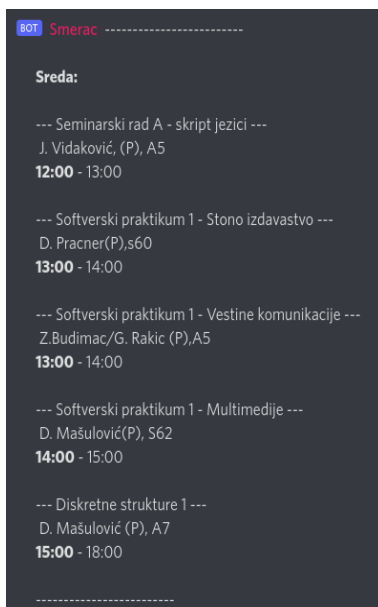
```
/smer <naziv_smera>
```

gde je <naziv_smera> IT/RN/PM i dodeljivati odgovarajući smer tom studentu.

1.6 Kalendar

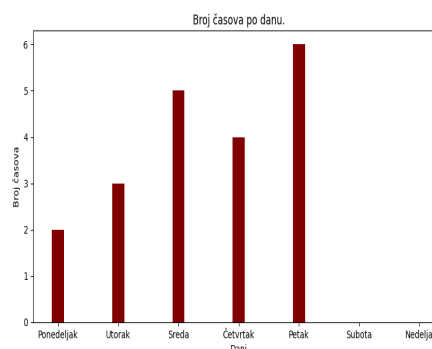
Učitavanje podataka iz Google-ovog kalendara će biti omogućeno zahvaljujući postojanju JSON objekta proizvoljnog kalendara koji je nalik ugrađenim rečnicima u Python-u. Uprkos lakom učitavanju potrebno je kompleksnije parsiranje kalendara, naime jedno string polje 'summary' sačinjava naziv predmeta, profesor, tip časa kao i mesto odvijanja. Smerač će to parsiranje izvesti na što efikasniji način sa minimalnim brojem petlji i korišćenjem ugrađenih i optimalnih funkcija za rad sa stringovima.

Nakon uspešno parsiranih podataka potrebno je generisati raspored prilagođen za čitanje na računarima i telefonima. Smerač će spojiti istoimene predmete ali će zapisati odvojene termine časova u zavisnosti od profesora koji ga drži kao i mesta odvijanja nastave. Primer ispisa se može videti na Slici a. Pored tekstualnog ispisa Smerač će od istih podataka generisati stubičasti grafik koji omogućava lakši prikaz broja časova na dnevnom nivou, primer toga se nalazi na Slici b.



Sreda:	
--- Seminarski rad A - skript jezici ---	
J. Vidaković, (P), A5	
12:00 - 13:00	
--- Softverski praktikum 1 - Stono izdavanje ---	
D. Pracner(P),s60	
13:00 - 14:00	
--- Softverski praktikum 1 - Vestine komunikacije ---	
Z.Budimac/G. Rakic (P),A5	
13:00 - 14:00	
--- Softverski praktikum 1 - Multimedije ---	
D. Mašulović(P), S62	
14:00 - 15:00	
--- Diskretne strukture 1 ---	
D. Mašulović (P), A7	
15:00 - 18:00	

(a) Parsiran i ureden raspored časova za sredu, IT smer



(b) Grafik sa dnevnim prikazom časova, IT smer

2 Opis programa

2.1 Nagli prekid usled greške

Deo koda prikazan na Listingu 1 služi da prekine tok programa u potpunosti, najčešće se koristi prilikom učitavanja promenljivih okruženja tj. u slučaju da nisu postavljene.

```
1 def fail(msg):  
2     print(msg)  
3     exit(1)
```

Listing 1: Nagli prekid usled greške

2.2 Promenljive okruženja

Da bi Smerač mogao učitati potrebne postavke, koristi se biblioteka *OS*. U kodu prikazanom na Listingu 2 se takođe može videti upotreba naglog prekida kao i korišćenje nekih podrazumevanih vrednosti.

```
1 def setup_config():  
2     config = dict()  
3  
4     ...  
5  
6     COMMAND = os.getenv("COMMAND")  
7     if COMMAND == None:  
8         COMMAND = "smer"  
9     config["command"] = COMMAND  
10  
11    ...  
12  
13    DISCORD_TOKEN = os.getenv("DISCORD_TOKEN")  
14    if DISCORD_TOKEN == None:  
15        fail("DISCORD_TOKEN env isn't set")  
16    config["discord_token"] = DISCORD_TOKEN  
17  
18    NUMBER_OF_ROLES = int(os.getenv("NUMBER_OF_ROLES"))  
19    if NUMBER_OF_ROLES == None:  
20        fail("NUMBER_OF_ROLES env isn't set")  
21  
22    ...  
23  
24    return config
```

Listing 2: Učitavanje promenljivih okruženja

2.3 Logging

Jedna od ključnih funkcionalnosti svakog programa je mogućnost zapisivanja šta se dešava prilikom izvršavanja. To je najpotrebnija alatka prilikom debugovanja koda, bilo to starog ili novog koda neke naknadno dodate funkcionalnosti. Podešavanje fajla za logovanje je prikazano na Listingu 3 a opcije loga na Listingu 4.

```
1 def setup_config():
2
3     ...
4
5     LOG_FILE = os.getenv("LOG_FILE")
6     if LOG_FILE == None:
7         LOG_FILE = "/config/logs/%s.log"
8                     %(datetime.today().strftime("%Y-%m-%d-%H-%M-%S"))
9     config["log_file"] = LOG_FILE
10
11     ...
```

Listing 3: Podešavanje fajla za logovanje

```
1 def setup_logger(config):
2     if os.getenv("DEBUG") == None:
3         logging_level = logging.INFO
4     else:
5         logging_level = logging.DEBUG
6
7     logging.basicConfig(
8         filename=config["log_file"],
9         level=logging_level,
10        format=
11            "%(asctime)s - %(name)s[%(process)s] - %(levelname)s - %(message)s",
12    )
```

Listing 4: Opcije loga

2.4 Globalne promenljive

Globalne promenljive su potrebne isključivo kada sve funkcije, iz celog programa, zahtevaju čitanje i pisanje zajedničkih podataka. Odličan primer ovoga bi bila konfiguracija sa svim postavkama potrebnim za rad programa prikazana na Listingu 5. Pored toga, koristi se i promenljiva `log` i `bot` koja služi da interaguje sa `interactions-py` API.

```
1 log = logging.getLogger("smerac")
2 config = setup_config()
3 bot = interactions.Client(token=config["discord_token"])
```

Listing 5: Globalne promenljive

2.5 Slash komande

Slash komande se zapisuju u obliku:

`/<naziv_komande> [opcija_1] [opcija_2] ...`

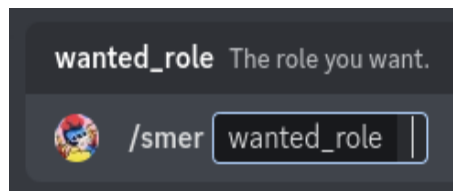
Takve komande su standardizovan način pozivanja funkcija proizvoljnog Discord bota.

2.5.1 Kreiranje slash komande

Pomoću biblioteke `interactions-py` moguće je kreirati slash komandu sa nazivom `smer` i obaveznom opcijom `traženi_smer` što je prikazano na Listingu 6. Na Slici a vidljiv je izgled komande korisnicima na Discordu.

```
1 @bot.command(  
2     name="smer",  
3     description="Choose your role!",  
4     options = [  
5         interactions.Option(  
6             name="wanted_role",  
7             description="The role you want.",  
8             type=interactions.OptionType.STRING,  
9             required=True,  
10        ),  
11    ],  
12 )
```

Listing 6: Kreiranje slash komande



(a) Slash komanda na Discordu

2.5.2 Dodeljivanje funkcionalnosti slash komandi

Da bi kreirana slash komanda izvršavala neki kod potrebno je odmah ispod njene kreacije deklarirati asinhronu funkciju koja će izvršavati željeni kod.

```
async def choose_role(ctx: interactions.CommandContext, wanted_role: str):
    author = ctx.author

    log.debug(f"Started choosing role for {author.nick}")

    guild = await ctx.get_guild()
    discord_roles = await guild.get_all_roles()

    for author_role_id in author.roles:
        author_role = await guild.get_role(author_role_id)
        for role in config["roles"]:
            if role.upper() == author_role.name.upper():
                await author.remove_role(author_role_id)
                log.debug(f"Removed {author.nick} from {role}")
            else:
                log.debug(f"Role {role} isn't {author_role.name}")

    if wanted_role != "none":
        for role in config["roles"]:
            if wanted_role.upper() == role.upper():
                for discord_role in discord_roles:
                    if wanted_role.upper() == discord_role.name.upper():
                        await author.add_role(discord_role.id)
                        break
                break
        log.info(f"Added {author.nick} to {wanted_role}.")
        await ctx.send(f"Successfully added {author.nick} to {wanted_role.upper()}!",
                       ephemeral=True)
    else:
        log.info(f"Removed {author.nick} from all roles.")
        await ctx.send(f"Successfully removed {author.nick} from all roles!",
                       ephemeral=True)
```

Opcija *ephemeral* znači da se poruka šalje samo autoru komande, tj. korisniku koji je pokrenuo komandu za izmenu smeru.

2.6 Kalendar

2.6.1 Instanciranje niti

Kako bi se kalendar za svaki od smerova asinhrono učitavao, odvojena je funkcija koja pronalazi sve odgovarajuće kanale od funkcije za učitavanje, parsiranje i generisanje kalendara.

```
async def calendar(delay):
    log.debug("Calendar")

    for guild in bot.guilds:
        channels = await guild.get_all_channels()
        for category in channels:
            if category.type ==
                interactions.ChannelType.GUILD_CATEGORY and
                category.name.upper() == "calendar":
                for role in config["roles"]:
                    for channel in channels:
                        if channel.parent_id == category.id and
                            role.upper() == channel.name.upper():
                            CALENDAR_URL = os.getenv("CALENDAR_URL_" + role)
                            if CALENDAR_URL == None:
                                log.info("CALENDAR_URL_%s env isn't set,
                                    skipping."%(role))
                            else:
                                asyncio.create_task(updateCalendar(channel,
                                    CALENDAR_URL, delay))
                                break
                    break
            break
```

U slučaju da ne postoji URL za kalendar proizvoljnog smeru, program prijavi to logu i nastavlja normalno da se izvršava.

2.6.2 Učitavanje kalendara

Glavna funkcija koja se bavi pozivom svih pomoćnih za rad sa kalendarom. Učitava JSON objekat sa URL-a i potom generiše narednu sedmicu i parsira je za podatke. Na kraju vrši ispis parsirane sedmice i takođe šalje graf sa brojem časova po danu.

```

async def updateCalendar(channel, calendar_url, delay):
    log.debug("Updating calendar for " + channel.name)
    spacer = "-----"
    week_old = dict()
    while True:
        calendar = json.loads(requests.get(calendar_url).text)
        week_data = await generateWeek(calendar["items"])
        week = await parseWeek(week_data)
        if week_old != week:
            log.debug(str(week_old))
            log.debug("----- week_old != week -----")
            log.debug(str(week))
            week_old = week
            await channel.purge(8, check=check_pinned)
            for weekday in week:
                if week[weekday] != []:
                    day_output = spacer
                    day_output += "\n\n**" + dayInWeekSrpski(weekday) +
                                   "：**\n\n"
                    for event in week[weekday]:
                        day_output += "--- " + event["name"] + " ---\n"
                        for info in event["info"]:
                            day_output += info["name"] + "\n"

                            for i in range(len(info["start_times"])):
                                day_output += "**" + info["start_times"][i]
                                    + "** - " + info["end_times"][i] + "\n"
                            day_output += "\n"
                    day_output += spacer
                    await channel.send(content=day_output)
            await channel.send(files=await classesPerDayGraph(channel.name,
                                                                week))
    await asyncio.sleep(delay)

```

2.6.3 Generisanje sedmice

Prolazi kroz raspored za celu godinu i uzima sve časove koji se odvijaju u narednih 7 dana a potom ih svrstava po danima i sortira po početnom vremenu odvijanja.

```
async def generateWeek(events):
    log.debug("Generating week")

    current_date = date.today()
    week = {"mon": [], "tue": [], "wed": [], "thu": [], "fri": [], "sat": [], "sun": []}

    for event in events:
        start_date=datetime.fromisoformat(event["start"]["dateTime"]).date()
        if (start_date >= current_date) and (start_date < current_date +
                                            timedelta(days=7)):
            if dayInWeek(start_date.weekday()) == "mon":
                week["mon"].append(event)
            elif dayInWeek(start_date.weekday()) == "tue":
                week["tue"].append(event)
            elif dayInWeek(start_date.weekday()) == "wed":
                week["wed"].append(event)
            elif dayInWeek(start_date.weekday()) == "thu":
                week["thu"].append(event)
            elif dayInWeek(start_date.weekday()) == "fri":
                week["fri"].append(event)
            elif dayInWeek(start_date.weekday()) == "sat":
                week["sat"].append(event)
            elif dayInWeek(start_date.weekday()) == "sun":
                week["sun"].append(event)

    for weekday in week:
        week[weekday].sort(key = lambda event :
                           datetime.fromisoformat(event["start"]["dateTime"]))

    return week
```

2.6.4 Parsiranje sedmice

Prolazi kroz sedmicu i spaja duplikate časova, ali razlikuje ako isti čas drže različiti proferosi ili se odvija na drugom mestu.

```

async def parseWeek(week_data):
    log.debug("Parsing week")
    week = {"mon": [], "tue": [], "wed": [], "thu": [], "fri": [], "sat": [], "sun": []}
    for weekday in week_data:
        if week_data[weekday] != []:
            for event in week_data[weekday]:
                name, info = event["summary"].split(", ", 1)
                start_time = datetime.fromisoformat(
                    event["start"]["dateTime"]).strftime("%H:%M")
                end_time = datetime.fromisoformat(
                    event["end"]["dateTime"]).strftime("%H:%M")
                foundEvent = False
                if week[weekday] != []:
                    for event_old in week[weekday]:
                        if name.upper().replace(" ", "") ==
                            event_old["name"].upper().replace(" ", ""):
                            foundInfo = False
                            for info_old in event_old["info"]:
                                if info.upper().replace(" ", "") ==
                                    info_old["name"].upper().replace(" ", ""):
                                    info_old["start_times"].append(start_time)
                                    info_old["end_times"].append(end_time)
                                    foundInfo = True
                            if not foundInfo:
                                new_info = {"name": info, "start_times": [],
                                             "end_times": []}
                                new_info["start_times"].append(start_time)
                                new_info["end_times"].append(end_time)
                                event_old["info"].append(new_info)
                            foundEvent = True
                if not foundEvent:
                    new_event = {"name": name, "info": []}
                    new_info = {"name": info, "start_times": [], "end_times": []}
                    new_info["start_times"].append(start_time)
                    new_info["end_times"].append(end_time)
                    new_event["info"].append(new_info)
                    week[weekday].append(new_event)

    return week

```

2.7 Kreiranje grafa

Prolazi kroz parsiranu sedmicu i broji koliko ima časova po danu, nakon toga generiše graf uz pomoć matplotlib[6] biblioteke. Povratna vrednost je lokacija sačuvane slike grafa.

```
async def classesPerDayGraph(channel_name, week):
    log.debug("Plotting graph for " + channel_name)

    filename = config["saved_plots"] + "/" + channel_name + ".png"
    data = dict()

    for weekday in week:
        data[dayInWeekSrpski(weekday)] = len(week[weekday])

    keys = list(data.keys())
    values = list(data.values())

    fig = plt.figure(figsize = (10, 5))

    plt.bar(keys, values, color = 'maroon', width = 0.2)

    plt.xlabel("Dani")
    plt.ylabel("Broj časova")
    plt.title("Broj časova po danu.")

    plt.savefig(filename)
    plt.close()

    return interactions.File(filename=filename)
```

3 Zaključak

test

Literatura

- [1] “Discord.” <https://discord.com>.
- [2] “Python.” <https://www.python.org>.
- [3] “interactions-py.” <https://github.com/interactions-py/interactions.py>.
- [4] “Asyncio.” <https://docs.python.org/3/library/asyncio.html>.
- [5] “Gpl.” <https://www.gnu.org/licenses/gpl-3.0.en.html>.
- [6] “Matplotlib.” <https://matplotlib.org/stable/tutorials/introductory/pyplot.html>.