```
# ----------------------Immports---------------
from .. import loader, utils
from ..inline.types import InlineCall

from typing import Union, Optional
import re
import asyncio
from asyncio.exceptions import TimeoutError
from datetime import datetime, timedelta
import logging
import json as Json
from random import choice
import pytz

from telethon.tl.types import (
    Message,
    User,
    MessageEntityPhone,
    MessageEntityMentionName,
    MessageEntityTextUrl,
    MessageEntityMention,
    Channel,
)

from telethon.tl.functions.channels import GetParticipantsRequest
from telethon.tl.types import ChannelParticipantsSearch

MEP = MessageEntityPhone
MEMN = MessageEntityMentionName
METU = MessageEntityTextUrl
MENT = MessageEntityMention
logger = logging.getLogger('BioWars Tools')
re._MAXCACHE = 3000

# ---------------------------Module--------------------


@loader.tds
class BioWars(loader.Module):
    """В разработке \nБудущий конкурент юб Лапика и Кнопки"""

    strings = {
        "name": "BioWars Tools",
```

```
        "link_id": "tg://openmessage?user_id=",
        "link_username": "https://t.me/",

        "commands": {
            "z": "[args] [reply] ",
            "id": "[arg/reply] -",
            "ids": "[args] [reply] - Чекает айди по реплаю",
            "dov": "Показывает информацию по доверке",
            'zz': 'Аналог команды .б из био',
            'nik': '[id] [имя] - запись человека',
            'pref': '[id] [префикс] - записывает префикс дова'
        },
        # Зарлист
        'zar.search':
            "🔍 Жертва {} приносит:\n"
            "☣️ <b>+{} био-опыта</b>\n"
            "📆 <b>Дата:</b> <i>{}</i> \n"
            '📆 <b>Заражение до:</b> {} ',

        'zar.save':
            "🦠Я записал <b><code>{}</code></b> хозяин.\n"
            "<b>☣️<s>{}</s> +{} био-опыта.</b>  \n"
            "📆 <b>Заражение до:</b> {} ",

        'z.nf': '🔍 Жертва <code>{}</code> не найдена в зарлисте.',

        'edit_nik': '<b>Юзер <code>@{0}</code> сохранен как</b> <a href =
"tg://openmessage?user_id={0}">{1}</a>',
        'edit_pref': '<b>Префикс <code>{}</code> сохранен для <code>@{}</code></b>',
        # Руководства по модулю
        "bio.commands": "<b>🚀 Boy, ты наверное удивился увидев что команд в    модуле нет,
но к твоему счастью они все-же есть. 🧙Магия какая-то...   \n\n"
        "📁 Доступные команды:</b> \n"
        "{1} \n"
        "<b>😉 Оказывается это  еще не все, ниже можешь посмотреть еще  руководства по
модулю. Приятного использования</b> \n"
        "<code>{0}biotools инфо</code> - <b>небольшая информация о вас</b> \n"
        "<code>{0}biotools зарлист</code> - <b>помощь по зарлисту</b> \n"
        "<code>{0}biotools доверка</code> - <b>помощь по доверке</b>",
        'bio.info':
            '📊 <b>Небольшая информация:</b> \n\n'
            '☢️ <b>Жертв в зарлисте:</b> <code>{}</code> \n'
            '☢️ <b>Суммарный опыт с жертв:</b> {} \n'
```

```
            '🔍<b>Известных:</b> <code>{}</code> \n'
            '🔰 <b>Доверенных пользователей:</b> <code>{}</code>',

        "bio.zar": "",
        "bio.dov": "Возможности доверки",
        'bio.dov.levels':
            '<b>📊 Информация об уровнях доверки: \n'
            'Существует 4 уровня доверки \n\n'
            '📗 1 уровень: \n'
            '  🔒 <i>Возможности</i>: Доступ к заражениям | Вакцина | Калькулятор | Краткая
лаба | Просмотр жертв в зарлисте \n'
            '📒 2 уровень: \n'
            '  🔏 <i>Возможности</i>: Управление зарлистом | Просмотр жертв \n'
            '📙 3 уровень: \n'
            '  🔐 <i>Возможности</i>:  Просмотр болезней | Просмотр мешка | Чек навыков \n'
            '📕 4 уровень: \n'
            '  🔓 <i>Возможности</i>: Фулл лаба | Смена имени патогена(лабы) | Возможность
ставить +вирусы | Прокачка навыков \n\n'
            '🩹 Примечание: \n'
            'Всем овнерам автоматически ставится 4 уровень доверки \n'
            'При желании это можно изменить</b>',
        # Все что относится к доверке
        "dov": "<b>⚙️ Информация по доверке</b> \n"
        "🕹️ <code>{0}dov dovs</code> - список доверенных пользователей \n"
        "🕹️ <code>{0}dov prefs</code> - список доверенных вам пользователей \n\n"
        '  🕹️ <code>{0}dov set</code> [айди/реплай] - Добавить|Удалить саппорта \n'
        "  🕹️ <code>{0}dov set</code> [айди/реплай] |уровень| -- Добавить|
Повысить/Понизить уровень доверки сапорта\n\n"

        "🕹️ <code>{0}dov nik</code> [ник] -- <b>Установить ник</b> \n"
        "  🔱 Ваш ник: <code>{1}</code> \n\n"
        "🕹️ <code>{0}dov st</code> -- <b>Включение/Выключение доверки</b> \n"
        "  {2} Статус доверки: <b>{3}</b> \n\n"
        "<b>🗃️ Подробнее о доверке можно почитать командой:</b> \n"
        "<code>{0}biotools доверка</code> \n"
        "<b>🗃️ Подробнее об уровнях доверки можно почитать командой:</b> \n"
        "<code>{0}biotools доверка -уровни</code>",

        'dov.users': '🚀 <b>Список доверенных пользователей:</b> \n' \
        '{}',
        "dov.users.chat": '🔰 Список доверенных пользователей в чате: \n' \
        '{}',
        "dov.prefs": '🚀 <b>Список ваших доверок:</b> \n'
```

```
        '{}',
        "dov.prefs.chat": '🔰 Список доверевших пользователей в чате: \n' \
        '{}',

        # Команды доверки
        "dov.rem": "⚠️ @{} <b>удален из списка доверенных пользователей!</b>",
        "dov.add":
            "🔰 @{} <b>добавлен в список доверенных пользователей! \n"
            "🔐 <b>Уровень доверки:</b> {}",
        'dov.edit_level':
            '🚀 Вы изменили уровень доверки у <code>@{}</code>! \n'
            '🔰 <b> <s>{}</s> ⇨ {}</b>',
        "nick.rename": "ℹ️ Вы изменили ник! \n" \
        "🔰 <s>{0}</s> ⇨ <b>{1}</b>",
        "dov.status.True": "✅ <b>Доверка запущена</b>",
        "dov.status.False": "❎ <b>Доверка приостановлена</b>",
        # Ошибки
        "no.reply": "🙄 Отсуствует реплай",
        "no.args": "🙄 Отсуствуют нужные аргументы",
        "no.args_and_reply": "🙄 Отсуствует реплай и аргументы",
        "args_error": '🙄 Аргументы введены неправильно',
        "len_error": "📘 <b>Превышено допустимое количество символов. Лимит 8
символов</b>",
        "hueta": "😶 Тебе не кажется что тут что-то не так?",
        # просто слова
        "messages.biotop": [
            '🙄 Самое время посмотреть биотоп',
            '👽 Интересный биотоп',
            '🤧 Скучный биотоп'
        ],
        'messages.misc': [
            "🤔 Что же на этот раз?",
            "😰 Куда катится мир...",
            "🥶 Видимо сейчас будет фарм",
            "🫡 Вперед по новой",
        ],
        "get_user": "🚀Пользователь: \n"
        "<b>🥷</b> <a href='tg://openmessage?user_id={}'>{}</a> \n"
        "<b>📃Юзернейм:</b> @{} \n"
        "<b>🆔Айди:</b> <code>@{}</code>",
        'calc_formul': {
            'zar': 2.5,
            'imun': 2.45,
```

```
            'sb': 2.1,
            'kvala': 2.6,
            'pat': 2,
            'letal': 1.95
        }


    }
# ----------------------Functions------------------

    def __init__(self):
        self.config = loader.ModuleConfig(
            loader.ConfigValue(
                "Вкл/Выкл доверки",
                False,
                "Статус доверки",
                validator=loader.validators.Boolean(),
            ),
            loader.ConfigValue(
                "Автозапись жертв",
                False,
                "Автозапись жертв(БЕТА) \nМожет работать некоректно",
                validator=loader.validators.Boolean(),
            ),
        )

    async def client_ready(self, client, db):
        # Nummod + BioWars
        self.client = client
        self.db = db

        # NumMod
        if not self.db.get("NumMod", "numfilter"):
            # Добовление овнеров юб в список доверевшихся людей
            # Айди аккаунта там тоже присуствуе
            owners = list(getattr(self.client.dispatcher.security, "owner"))

            # У овнеров автоматически будет 4 уровень доверки
            # 1) Заражения\вакцина\зарлист\калькулятор\краткая лаба\inline .б
            # 2) возможность записывать жертв в зарлист\чек жертв
            # 3)Чек болезней\чек мешка\чек навыков через вир лаб {навык}
            # 4) Фулл лаба\смена имени патогена(лабы)\возможность ставить
+вирусы\прокачка навыков
```

```
            self.db.set(
                "NumMod",
                "numfilter",
                {"users": owners, "filter": None, "status": False},
            )
        # infList
        if not self.db.get("NumMod", "infList"):
            self.db.set("NumMod", "infList", {})

        if not self.db.get("BioWars", "DovUsers"):
            owners = list(getattr(self.client.dispatcher.security, "owner"))
            users = {}
            for i in self.db.get("NumMod", "numfilter")["users"]:
                users[str(i)] = 1
            for i in owners:
                users[str(i)] = 4
            self.db.set("BioWars", "DovUsers", users)

        if not self.db.get('BioWars', 'FamousUsers'):
            self.db.set('BioWars', 'FamousUsers', {})
            # {id : username }
        if not self.db.get('BioWars', 'LastInfect'):
            self.db.set('BioWars', 'LastInfect', None)

        if not self.db.get('BioWars', 'InfectionBefore'):
            self.db.set('BioWars', 'InfectionBefore', {})

        if not self.db.get('BioWars', 'YourLetal'):
            self.db.get('BioWars', 'YourLetal', 1)
            # Ваш летал, будет использоватся при записи жертв в зарлист
        if not self.db.get('BioWars', 'UsersNik'):
            self.db.set('BioWars', 'UsersNik', {})
            # user_id : желаемый ник
        if not self.db.get('BioWars', 'FamousPrefs'):
            self.db.set('BioWars', 'FamousPrefs', {})
            # user_id : pref
        # При заражении ставится True
        # Статус заражения
        self.db.set("BioWars", "infStatus", False)
        # Интервал между заражениями
        self.db.set("BioWars", "infInterval", 4)

    async def send(self, text: str, message: Message) -> None:
```

```
        """Если возникает ошибка при отправке сообщения с инлайн клавиатурой,   то
отправляется обычное сообщение с таким же текстом"""
        # from telethon.errors.rpcerrorlist import BotResponseTimeoutError
        try:
            await self.inline.form(
                text,
                reply_markup={
                    "text": " 🔻 Закрыть",
                    "callback": self.inline__close,
                },
                message=message,
                disable_security=False,
            )
        except:
            await utils.answer(message, text)

    async def inline__close(self, call) -> None:
        await call.delete()

    async def return_user(self, username: str) -> int:
        if username not in self.db.get('BioWars', 'FamousUsers').values():
            await self._write_user(username=username)

        famous_users = self.db.get('BioWars', 'FamousUsers')
        for k, v in famous_users.items():
            if v == username:
                user_id = k
                return user_id

    async def save_nik(self, user_id: int, nik: str) -> None:
        users_nik = self.db.get('BioWars', 'UsersNik')
        users_nik[str(user_id)] = nik
        self.db.set('BioWars', 'UsersNik', users_nik)

    async def save_pref(self, user_id: int, nik: str) -> None:
        users_nik = self.db.get('BioWars', 'FamousPrefs')
        users_nik[str(user_id)] = nik
        self.db.set('BioWars', 'FamousPrefs', users_nik)

    async def _write_user(self, username: Optional[str] = None, user_id: Optional[int] = None) ->
None:
        famous_users = self.db.get('BioWars', 'FamousUsers')
        if (username in famous_users) or (user_id in famous_users):
```

```
            return

        if username and user_id:
            famous_users[user_id] = username
        # Если есть юзер айди, вытаскиваем юзернейм и сохраняем его
        if not username:
            if user_id not in famous_users.keys():
                try:
                    user = await self.client.get_entity(user_id)
                    username = user.username if user.username else None
                    famous_users[user_id] = username
                except:
                    return None
        if not user_id:
            # Если есть юзернейм, то вытаскиваем с помощью него юзер айди
            if username not in famous_users.values():
                try:
                    user = await self.client.get_entity(username)
                    user_id = user.id
                    famous_users[user_id] = username
                except:
                    return None
        # Сохраняем все
        self.db.set('BioWars', 'FamousUsers', famous_users)

    async def save_last_infect(self, user: Optional[str]) -> None:
        if user:
            user = user.replace('@', '')
            if not user.isdigit():
                if user.startswith('https://t.me/'):
                    user = user.replace('https://t.me/', '')
                user = await self.return_user(username=user)

        save = user if user else None
        self.db.set('BioWars', 'LastInfect',
                    save)

    # Нужен класс чата, а не айди чата
    async def get_members_chat(self, chat: Channel) -> Union[list, str]:
        offset_user = 0    # номер участника, с которого начинается считывание
        limit_user = 50   # максимальное число записей, передаваемых за один раз

        users = []   # список всех участников канала
```

```
    filter_user = ChannelParticipantsSearch('')
    try:
        while True:
            participants = await self.client(GetParticipantsRequest(
                chat,
                filter_user,
                offset_user,
                limit_user,
                hash=0))
            if not participants.users:
                break

            users.extend(participants.users)
            offset_user += len(participants.users)
        ids = [i.id for i in users]
        return ids

    except TypeError:
        return 'NotChat'

async def _handler_link(self, link) -> Optional[str]:
    if link.startswith(self.strings("link_id")):
        return "@" + link.replace(self.strings("link_id"), "")
    elif link.startswith(self.strings("link_username")):
        return "@" + link.replace(self.strings("link_username"), "")
    else:
        return None

async def number_convert(self, number: int) -> str:
    if number >= 1000000000:
        return f"{number / 1000000000:.1f}B"
    elif number >= 1000000:
        return f"{number / 1000000:.1f}M"
    elif number >= 1000:
        return f"{number / 1000:.1f}k"
    else:
        return str(number)

async def get_pref(self) -> str:
    return self.db.get("hikka.main", "command_prefix", ".")

async def _generator_links(self, reply, args: str) -> Union[list, str]:
```

```
    list_args, lis = [], []
    for i in args.split(" "):
        if "-" in i:
            ot_do = i.split("-")
            try:
                list_args.extend(
                    str(x) for x in range(int(ot_do[0]), int(ot_do[1]) + 1)
                )
            except Exception:
                return "wrong_ot-do"

        else:
            list_args.append(i)

    a = reply.text
    entity = reply.get_entities_text()
    users = []
    # validate_text = await self.validate_text(text)

    for e in entity:
        if isinstance(e[0], MENT):
            url = e[1]
            # if not url.startswith('@'):
            # continue

            users.append(url)

        elif isinstance(e[0], METU):
            url = await self._handler_link(e[0].url)
            users.append(url)

    for arg in list_args:
        lis.append(users[int(arg)-1])

    return lis

async def _o_generator_links(self, reply: Message) -> Union[list, str]:
    lis = []
    json = Json.loads(reply.to_json())
    try:
        for i in range(len(reply.entities)):
            try:
                link = json["entities"][i]["url"]
```

```
                if link.startswith("tg"):
                    users = "@" + link.split("=")[1]
                    lis.append(users)
                elif link.startswith("https://t.me"):
                    a = "@" + str(link.split("/")[3])
                    lis.append(a)
                else:
                    return "hueta"
            except Exception:
                blayt = reply.raw_text[
                    json["entities"][i]["offset"]: json["entities"][i]["offset"]
                    + json["entities"][i]["length"]
                ]
                lis.append(blayt)
        return lis
    except TypeError:
        return "hueta"

async def get_top_zhertv(self, message: Message, num_list: int) -> None:
    import operator
    # Сортировка зарлиста
    infList = self.db.get('NumMod', 'infList')
    a = {}
    zhertvs = []
    for k, v in infList.items():
        a[k] = int(float((v[0]))) if not 'k' in v[0] else int(
            float(v[0][:-1].replace(',', '.')) * 1000)

    sort = sorted(a.items(), key=operator.itemgetter(1), reverse=True)

    sort_dict = dict(sort)

    users = list(sort_dict.keys())

    for i in range(0, len(users), 50):
        e_c = users[i: 50 + i]

        if len(e_c) < 50:
            e_c = e_c + [None for y in range(50 - len(e_c))]
        zhertvs.append(e_c)

    if num_list > len(zhertvs):
        await utils.answer(message, 'Такого номера вкладки нет')
```

```
            return
        # ---------------------------------------------
        # Генерация текста с жертвами

        infectBefore = self.db.get(
            'BioWars', 'InfectionBefore')
        niks = self.db.get('BioWars', 'UsersNik')
        all_exps = int(sum([eval(i[0].replace(",", ".").replace(
            'k', '*1000')) for i in list(infList.values())]))
        bio_exp = await self.number_convert(all_exps)
        all_exps = '{:,}'.format(all_exps).replace(',', ' ')

        sms = f'Топ ваших жертв({num_list}/{len(zhertvs)}): \n'
        count = 1

        for i in zhertvs[num_list-1]:
            if not i:
                continue
            user = infList[i]
            zar_do = infectBefore[i] if i in infectBefore.keys(
            ) else '<b>неизвестная дата</b>'
            if i[1:] in niks.keys():
                nik = niks[str(i[1:])]
                usr = f'<a href="tg://openmessage?user_id={i[1:]}">{nik}</a>'
            else:
                usr = i
            sms += f'{count}. {usr} | +{user[0]} | заражение до {zar_do} \n'
            count += 1

        sms += f'\n📊 Итого: {len(infList)} заражённых и {bio_exp} био-опыта \n'
        sms += f'🧬 Ежедневная премия: {all_exps} био-ресурса'
        await self.send(sms, message)

    async def get_zhertv(self, message: Message, num_list: int) -> None:
        infList = self.db.get('NumMod', 'infList')
        users = list(reversed(infList.keys()))
        zhertvs = []

        for i in range(0, len(users), 50):
            e_c = users[i: 50 + i]

            if len(e_c) < 50:
                e_c = e_c + [None for y in range(50 - len(e_c))]
```

```
            zhertvs.append(e_c)

        # генерация сообщения

        if num_list > len(zhertvs):
            await utils.answer(message, 'Такого номера вкладки нет')
            return

        infectBefore = self.db.get('BioWars', 'InfectionBefore')
        niks = self.db.get('BioWars', 'UsersNik')
        all_exps = int(sum([eval(i[0].replace(",", ".").replace(
            'k', '*1000')) for i in list(infList.values())]))
        bio_exp = await self.number_convert(all_exps)
        all_exps = '{:,}'.format(all_exps).replace(',', ' ')

        sms = f'Ваши жертвы({num_list}/{len(zhertvs)}): \n'
        count = 1

        for i in zhertvs[num_list-1]:
            if not i:
                continue
            user = infList[i]
            zar_do = infectBefore[i] if i in infectBefore.keys(
            ) else '<b>неизвестная дата</b>'
            if i[1:] in niks.keys():
                nik = niks[str(i[1:])]
                usr = f'<a href="tg://openmessage?user_id={i[1:]}">{nik}</a>'
            else:
                usr = i
            sms += f'{count}. {usr} | +{user[0]} | заражение до {zar_do} \n'
            count += 1

        sms += f'\n📊 Итого: {len(infList)} заражённых и {bio_exp} био-опыта \n'
        sms += f'🧬 Ежедневная премия: {all_exps} био-ресурса'
        await self.send(sms, message)

    async def bio(self, reply: Message, me: User) -> None:

        infList = self.db.get("NumMod", "infList")
        b = reply.raw_text.splitlines()

        niks = self.db.get('BioWars', 'UsersNik')
        chat_flag = True if 'Биотоп чмоней' in b[0] or '🏢 УЧАСТНИКИ КОРПОРАЦИИ' in b[0] else
```

```
False
        b.pop(0)
        sms = ''
        exps = []
        # Add exp
        for i in b:
            try:
                a = i.split('|')
                if not chat_flag:
                    continue
                exps.append(a[-2])
            except:
                pass

        json = Json.loads(reply.to_json())

        if len(exps) == 0:

            entity = reply.get_entities_text()
            users = []

            for e in entity:
                if isinstance(e[0], MENT):
                    url = e[1]
                    users.append(url)
                elif isinstance(e[0], METU):
                    url = await self._handler_link(e[0].url)
                    users.append(url)
            count = 1
            for i in users:
                if str(i[1:]) == str(me.id):
                    name = me.first_name
                    sms += f'{str(count)}. 🔆 <a href=
"tg://openmessage?user_id={me.id}">{name}</a>\n'
                    count += 1
                    continue

                if str(i[1:]) in niks:
                    nik = niks[str(i[1:])]
                    name = f"<a href='tg://openmessage?user_id={i[1:]}'>{nik}</a>"
                else:
                    name = i
```

```
            exp = infList[i][0] if i in infList else None
            exp = f'☢ {exp} опыта' if exp else '🆕 Новая жертва'
            sms += f'{count}. {name} | {exp} \n'
            count += 1
        return sms

    else:
        count = 1
        for i in range(0, len(b)):
            try:
                exp = exps[i].replace(",", ".")
                s = exp.find(' опыт')
                exp = exp[1:s].replace(' ', '')
                if 'k' in exp:
                    exp_count = float(exp[:-1])
                    if exp_count < 10.0:
                        exp = int(round(exp_count * 100, 0))

                    else:
                        exp_count = float(exp[:-1])
                        exp_count = int(exp_count)
                        exp = str(exp_count / 10) + 'k'

                else:
                    exp_count = int(exp)
                    exp = exp_count // 10

            except:
                exp = None

            link = json["entities"][i]["url"]
            bla = []
            if link.startswith('tg'):
                for i in link.split('='):
                    bla.append(i)

                if str(bla[1]) == str(me.id):
                    name = me.first_name
                    sms += f'{str(count)}. 🔆 <a href=
"tg://openmessage?user_id={me.id}">{name}</a> | {exp} опыта \n'
                    count += 1
                    continue
```

```
user_id = bla[1]
if '@' + str(user_id) in infList:
    if chat_flag:
        user = infList['@' + str(user_id)]
        usr_exp = user[0].replace(',', '.')
        exp_count = str(exp)

        if usr_exp[-1] == 'k':
            usr_exp = float(usr_exp[:-1]) * 1000

        if exp_count[-1] == 'k':
            exp_count = float(exp_count[:-1]) * 1000

        result = int(float(exp_count) - float(usr_exp))

        # abc.append(str(result))

        if result > 0:
            if result < 1000:
                result = f'✅ [+{str(result)}]'
            else:
                result = f'✅ [+{str(round(float(result) / 1000, 1))}k]'
        elif result == 0:
            result = f' 🟰 [{str(result)}]'

        else:
            if result > -1000:
                result = f'❌ [{str(result)}]'
            else:
                result = f'❌ [{str(round(float(result) / 1000, 1))}k]'
        zh = f"({user[0]}) | <b>{result}</b>"
    else:
        zh = f"☢ (+{infList['@' + str(user_id)][0]})"
else:
    if chat_flag:  # если это чат и жертвы нет в зарлисте
        exp_count1 = str(exp)
        if exp_count1[-1] == 'k':
            exp_count1 = float(exp_count1[:-1]) * 1000

        if round(float(exp_count1), 1) < 10000.0:  # +{}к
            zh = f'| 🆕 <b>[+{round(float(exp_count1) / 1000,1)}]</b>'
        else:  # + {}
            zh = f'| 🆕 <b>[+{round(float(exp_count1)/ 1000,1)}k]</b>'
```

```
                else:
                    zh = ''

            try:
                if str(bla[1]) in niks:
                    nik = niks[str(bla[1])]
                    name = f"<a href='tg://openmessage?user_id={bla[1]}'>{nik}</a>"
                else:
                    name = '@' + str(bla[1])

                exp = f'| {exp}'
                sms += f'{str(count)}. {name} {zh} {exp} опыта \n'

            except:
                if str(bla[1]) in niks:
                    nik = niks[str(bla[1])]
                    name = f"<a href='tg://openmessage?user_id={bla[1]}'>{nik}</a>"
                else:
                    name = '@' + str(bla[1])

                exp = f'| {exp}'
                sms += f'{str(count)}. {name} {zh} {exp} опыта \n'
        count += 1
    return sms

async def message_q(  # отправляет сообщение боту и возращает ответ
    self,
    text: str,
    bot_id: int = 5443619563,
    mark_read: bool = True,
    delete: bool = True,
) -> str:
    """Отправляет сообщение и возращает ответ"""
    async with self.client.conversation(bot_id, exclusive=False) as conv:
        try:
            msg = await conv.send_message(text)
            response = await conv.get_response()
            if mark_read:
                await conv.mark_read()
            if delete:
                await msg.delete()
                await response.delete()
            return response.text
```

```
            except TimeoutError:
                return "Timeout"


# ----------------------Commands in watcher----------


    async def z_command(self, message: Message, args_raw: str, text: str, reply: Message) ->
None:
        if self.db.get("BioWars", "infStatus"):
            await message.reply('❎ Заражения еще не завершены')
            return

        if not args_raw and not reply:  # .z - аргументов нет
            text = self.strings("no.args_and_reply")
            await utils.answer(message, text)
            return

        if (reply and not args_raw):
            entities = reply.get_entities_text()
            if re.search(r"👨‍🔬 Была проведена операция заражения", reply.text):

                infect = await self._handler_link(entities[1][0].url)
                self.db.set("BioWars", "infStatus", True)
                await message.reply(f"биоеб {infect}")

                await self.save_last_infect(str(infect))
                self.db.set("BioWars", "infStatus", False)
                return

            elif len(entities) == 1 and len(entities[0]) == 2:
                user_id = entities[0][1]

                if not user_id.startswith('@'):
                    return

                self.db.set("BioWars", "infStatus", True)
                await message.reply(f"биоеб {user_id}",)

                await self.save_last_infect(str(user_id))
                self.db.set("BioWars", "infStatus", False)

            else:
```

```
            user_id = reply.sender_id

            self.db.set("BioWars", "infStatus", True)
            await message.reply(f"биоеб @{user_id}",)

            await self.save_last_infect(str(user_id))
            self.db.set("BioWars", "infStatus", False)

        return
    if reply and args_raw == 'o':

        ids = await self._o_generator_links(reply)

        self.db.set("BioWars", "infStatus", True)

        for i in ids:
            interval = self.db.get("BioWars", "infInterval", 4)
            await message.client.send_message(
                message.peer_id, f"биоеб {i}", reply_to=reply
            )
            await asyncio.sleep(interval)

        else:
            await message.reply('✅ Заражения окончены!')

        self.db.set("BioWars", "infStatus", False)
        return

    if reply and args_raw:
        users = await self._generator_links(reply, args_raw)

        if len(users) == 1:
            self.db.set("BioWars", "infStatus", True)
            await message.reply(f"биоеб {users[0]}")

            await self.save_last_infect(users[0])
            self.db.set("BioWars", "infStatus", False)
        else:
            self.db.set("BioWars", "infStatus", True)
            interval = self.db.get("BioWars", "infInterval", 4)
            for infect in users:
                if self.db.get("BioWars", "infStatus"):
                    await message.reply(f"биоеб {infect}",)
```

```
                await asyncio.sleep(interval)
            else:
                return
        else:
            await message.reply('✅ Заражения окончены!')

        self.db.set("BioWars", "infStatus", False)
        return

async def id_command(self, message: Message, args: str, reply) -> None:
    if not args and not reply:
        user = await self.client.get_me()

    elif reply:
        user_id = reply.sender_id
        user = await message.client.get_entity(user_id)

    elif args.startswith("@"):
        if args[1:].isdigit():
            user_id = int(args[1:])
        else:
            user_id = args[1:]

        user = await message.client.get_entity(user_id)
    else:
        return

    username = user.username if user.username else "Отсуствует"
    await self.write_user(username, user.id)
    await self.client.send_message(
        message.chat_id,
        self.strings("get_user").format(
            user.id, user.first_name, username, user.id
        ),
        reply_to=reply,
    )

async def ids_command(self, message: Message, args_raw: str, reply) -> None:
    if not reply:
        await utils.answer(message, self.strings("no.reply"))
        return
    ids = (
        await self._generator_links(reply, args_raw)
```

```
            if args_raw
            else await self._o_generator_links(reply)
        )
        for i in ids:
            await message.client.send_message(
                message.peer_id, f".ид {i}", reply_to=reply
            )
            await asyncio.sleep(3.5)
        else:
            await message.respond("<b>Все айди прочеканы!</b>")

    async def dov_command(
        self, message: Message, args_list: list, args_raw: str, reply
    ) -> None:

        numfilter = self.db.get("NumMod", "numfilter")
        biowars_dovs = self.db.get("BioWars", "DovUsers")
        pref = await self.get_pref()

        if not args_raw and not reply:
            status_emj = "▶️" if self.config["Вкл/Выкл доверки"] else "⏸️"
            status = "Включено" if self.config["Вкл/Выкл доверки"] else "Выключено"
            nik = numfilter["filter"] if numfilter["filter"] else "Не   установлен"

            text_message = self.strings("dov").format(
                pref, nik, status_emj, status
            )
            await self.send(text_message, message)
            return

        if args_list[0].lower() == "set":
            # Если 2 аргумента то ставим первый уровень, если 3 аргументы и 3 типа инт ставим уровень указанный в нем

            level = None
            data = args_list[1:]
            logging.info(f'{data}')
            if reply:
                user_id = str(reply.sender_id)
                if data:
                    level = int(data[0]) if data[0].isdigit() else None
                if not level:
                    level = None
```

```
        elif re.fullmatch(r"@\d+", data[0]):
            user_id = data[0].replace('@', '')
            # try:
            if len(data) >= 2:
                level = int(data[1]) if data[1].isdigit() else None
            if not level:
                level = None
            # except Exception:
                # await utils.answer(message, self.strings('args_error'))
                # return
        else:
            await utils.answer(message, self.strings('args_error'))
            return
        # Я знаю что in распостраняется только на user_id

        if level and user_id in biowars_dovs.keys():
            old_level = biowars_dovs[user_id]
            biowars_dovs[user_id] = level
            self.db.set("BioWars", "DovUsers", biowars_dovs)
            await utils.answer(message, self.strings('dov.edit_level').format(user_id, old_level,
level))

            return

        elif str(user_id) in biowars_dovs.keys():
            numfilter["users"].remove(str(user_id))
            biowars_dovs.pop(user_id)
            self.db.set("BioWars", "DovUsers", biowars_dovs)
            await utils.answer(message, self.strings("dov.rem").format(user_id))
            return
        else:
            logging.info(
                f'{user_id} - {level}')
            level = level if level else 1
            numfilter["users"].append(user_id)
            biowars_dovs[user_id] = level
            text_message = self.strings("dov.add").format(user_id, level)
            self.db.set("BioWars", "DovUsers", biowars_dovs)

            await utils.answer(message, text_message)
            return
```

```
elif args_list[0].lower() == "nik":
    if args_list[1]:
        if len(args_list[1]) > 8 or len(args_list) >= 3:
            await utils.answer(message, self.strings("len_error"))
            return

        old_nik = numfilter["filter"] if numfilter["filter"] else "Отсуствует"
        nik = args_list[1]
        numfilter['filter'] = nik
        self.db.set("NumMod", "numfilter", numfilter)
        await utils.answer(message,
                           self.strings("nick.rename").format(
                               old_nik, nik)
                           )

    else:
        await utils.aswer('Какой ник будем ставить?')
        return

elif args_list[0].lower() == "dovs":
    niks = self.db.get('BioWars', 'UsersNik')

    dovs_users = ''

    if len(args_list) > 1 and args_list[1].lower() == 'chat':
        r = await self.get_members_chat(message.chat)
        if r == 'NotChat':
            await utils.answer(message, 'Это не чат')
            return
        else:
            users = r
        i = 1
        for user in users:
            if str(user) in biowars_dovs.keys():
                level = biowars_dovs[str(user)]

                level = '4ур <b>(🔐 Полный Доступ)</b>' if level == 4 else f'{level} ур'

                if str(user) in niks.keys():
                    nik = niks[str(user)]
                    usr = f'<a href="tg://openmessage?user_id={user}">{nik}</a>'
                else:
                    usr = f'<code>@{user}</code>'
```

```
                dovs_users += f'<b>{i})</b> {usr} - {level} \n'
                i += 1
        dovs_users = dovs_users if dovs_users else 'В этом чате никого нет'

        await self.send(self.strings('dov.users.chat').format(dovs_users), message)
        return
    for i, (user_id, level) in enumerate(biowars_dovs.items(), start=1):

        level = '4ур <b>(🔐 Полный Доступ)</b>' if level == 4 else f'{level} ур'

        if str(user_id) in niks.keys():
            nik = niks[str(user_id)]
            usr = f'<a href="tg://openmessage?user_id={user_id[1:]}">{nik}</a>'
        else:
            usr = f'<code>@{user_id}</code>'

        dovs_users += f'<b>{i})</b> {usr} - {level} \n'
    await self.send(self.strings('dov.users').format(dovs_users), message)
    return

elif args_list[0].lower() == "prefs":
    prefs_users = self.db.get('BioWars', 'FamousPrefs')
    niks = self.db.get('BioWars', 'UsersNik')
    prefs = ''
    if len(args_list) > 1 and args_list[1].lower() == 'chat':
        r = await self.get_members_chat(message.chat)
        if r == 'NotChat':
            await utils.answer(message, 'Это не чат')
            return
        else:
            users = r
        i = 1
        for user in users:
            if str(user) in prefs_users.keys():
                pref = prefs_users[str(user)]

                if str(user) in niks.keys():
                    nik = niks[str(user)]
                    usr = f'<a href="tg://openmessage?user_id={user}">{nik}</a>'
                else:
                    usr = f'<code>@{user}</code>'
```

```
                prefs += f'<b>{i})</b> {usr} | {pref} \n'
                i += 1
            prefs = prefs if prefs else 'В этом чате никого нет'

            await self.send(self.strings('dov.prefs.chat').format(prefs), message)
            return

        for i, (user_id, pref) in enumerate(prefs_users.items(), start=1):
            if str(user_id) in niks.keys():
                nik = niks[str(user_id)]
                usr = f'<a href="tg://openmessage?user_id={user_id[1:]}">{nik}</a>'
            else:
                usr = f'<code>@{user_id}</code>'
            prefs += f'<b>{i})</b> {usr} | {pref} \n'

        prefs = prefs if prefs else 'Тут никого нет'

        await self.send(self.strings('dov.prefs').format(prefs), message)
        return

    elif args_list[0].lower() == "st":
        status = self.config["Вкл/Выкл доверки"]
        if status:
            self.config["Вкл/Выкл доверки"] = False
            await utils.answer(message, self.strings("dov.status.   False"))
        else:
            self.config["Вкл/Выкл доверки"] = True
            await utils.answer(message, self.strings("dov.status.True"))

async def bio_command(self, message: Message, reply: Message, me) -> None:
    if reply.text.startswith('Биотоп чмоней:'):
        sms = choice(self.strings('messages.biotop')) + '\n'
    else:
        sms = choice(self.strings('messages.misc')) + '\n'

    sms += await self.bio(reply, me)

    await self.send(sms, message)

async def nik_command(self, message: Message, args_list: list, args_raw: str) -> None:

    user_id = args_list[0].replace('@', '')
    user_nikname = ' '.join(args_list[1:])
```

```
        await self.save_nik(user_id, user_nikname)
        await utils.answer(message, self.strings('edit_nik').format(user_id, user_nikname))

    async def pref_command(self, message: Message, args_list: list) -> None:
        user_id = args_list[0].replace('@', '')
        user_pref = ' '.join(args_list[1:])
        await self.save_pref(user_id, user_pref)
        await utils.answer(message, self.strings('edit_pref').format(user_pref, user_id))

# ----------------------Commands------------------

    async def biotoolscmd(self, message: Message) -> None:
        """Помощь по модулю"""
        args_raw = utils.get_args_raw(message)
        infList = self.db.get("NumMod", "infList")
        famous_users = self.db.get('BioWars', 'FamousUsers')
        dov_users = self.db.get("BioWars", "DovUsers")
        if not args_raw:
            pref = await self.get_pref()
            commands = ""
            comm = self.strings("commands")
            for com, desc in comm.items():
                commands += f"▫ <code>{pref}{com}</code> {desc} \n"
                text = self.strings("bio.commands").format(
                    pref, commands)

        elif args_raw.lower() == "зарлист":
            text = self.strings("bio.zar").format()
        elif args_raw.lower() == "доверка":
            text = self.strings("bio.dov").format()
        elif args_raw.lower() == 'доверка -уровни':
            text = self.strings("bio.dov.levels")
        elif args_raw.lower() == 'инфо':
            exps = int(sum([eval(i[0].replace(",", ".").replace(
                'k', '*1000')) for i in list(infList.values())]))
            text = self.strings("bio.info").format(
                len(infList.keys()),
                '{:,}'.format(exps).replace(',', ' '),
                len(famous_users.keys()),
                len(dov_users.keys())
            )

        else:
```

```
            await utils.answer(message, "Что то явно не так")
            return
        await self.send(text, message)
        return

    # @loader.watcher("only_pm", "only_messages")
    async def watcher(self, message: Message):
        if not isinstance(message, Message):
            return

        text = message.text
        reply = await message.get_reply_message()
        sndr_id = message.sender_id
        me = await self.client.get_me()
        pref = await self.get_pref()
        args_list, args_raw = utils.get_args(
            message), utils.get_args_raw(message)
        infList = self.db.get("NumMod", "infList")
        msg_splitlines_1 = message.raw_text.splitlines()[0] if text else ''

        # Относится к автозаписе жертв
        # Берем айди/юзернейм из собщения пользователя и сохраняем в бд (в бд будет
лежать айди зараженного)
        if mes := re.fullmatch(r'(биоеб|биоеб)
(?P<lvl>[1-9]?[0]?\s)?((https?://)?t\.me/|@)([0-9a-z_A-Z]+)', msg_splitlines_1.lower()):
            if not self.config["Автозапись жертв"]:
                return
            if str(me.id) != str(sndr_id):
                return
            user = mes.group(5)
            await self.save_last_infect(user)
            return

        if re.search(r'(биоеб|биоеб) (?P<lvl>[1-9]?[0]?\s)?(равного|слабее|сильнее|р|=|-|\+)',
msg_splitlines_1.lower()):
            if not self.config["Автозапись жертв"]:
                return
            if str(me.id) != str(sndr_id):
                return

            user = None

            await self.save_last_infect(user)
```

```
            return

        # Автозапись жертв
        # Если ссылки на сообщение нету берем ее из бд
        if mes := re.search(r'🦠 <a
href="(?P<link>(?:https?://)?t\.me/[0-9a-z_]+|tg://openmessage\?user_id=(?P<id>[0-9]+))">.{1,2
55}</a> подвер.{1,2} заражению', text):
            if not self.config["Автозапись жертв"]:
                return

            mes = mes.groupdict()
            vremya = datetime.now(pytz.timezone(
                "Europe/Moscow"))
            msg_text = text
            split_text = text.splitlines()
            irises_id = [
                5443619563,
                707693258,
                5226378684,
                5137994780,
                5434504334,
                1136703023,
                1120322272

            ]
            if sndr_id not in irises_id:
                return
            attempts = "🗓 Отчёт об операции заражения объекта:"
            podverg = split_text[0] if attempts not in msg_text else split_text[3]
            retur = 0

            if mes['id'] == str(me.id):
                retur = 1

            elif me.username:
                if mes['link'] == 'https://t.me/' + me.username.lower():
                    retur = 1

                else:
                    return
            else:
                return
```

```
        if not retur:
            return

        reg = r"""🤒 Заражение на (\d+) дн[яей]{,2}
☣️ +(.*) био-опыта"""

        s = re.compile(reg)
        info = s.search(msg_text)

        letal = int(info.group(1))
        count = info.group(2).replace('+', '')

        try:
            x = msg_text.index('user?id=') + 8
            user = msg_text[x:].split('"', maxsplit=1)[0]
            self.db.set('BioWars', 'LastInfect', None)

        except ValueError:  # Если нет ссылки на жертву то берем ее из бд
            # Если в заражение от бота есть реплай, то берем айди из реплая, иначе из бд
            if reply:
                t = reply.raw_text.splitlines()[0]
                if '@' in t:
                    s = t.find('@')
                    user = t[s:].replace('@', '')
                else:
                    s = t.find('https://t.me/')
                    user = t[s:].replace('https://t.me/', '')

                if not user.isdigit():
                    user = await self.return_user(username=user)

                self.db.set('BioWars', 'LastInfect', None)

            else:
                # Берем данные о последнем зараженном
                # Если статус Тру(тоесть еще не заражли его)
                # То берем его айди и записываем его в дб
                user = self.db.get('BioWars', 'LastInfect')
                # self.db.set('BioWars', 'LastInfect',
                #             {'user_id': user['user_id'],
                #             'status': False})
                self.db.set('BioWars', 'LastInfect', None)
```

```
            # user = user['user_id']
        if not user:
            return
        letal_in_db = self.db.get('BioWars', 'YourLetal')
        user = '@' + str(user)
        if letal != letal_in_db:
            self.db.set('BioWars', 'YourLetal', letal)

        vremya1 = vremya.strftime("%d.%m")
        vremya_do = vremya.strftime("%d.%m") if letal == 1 else (vremya +
                                                timedelta(days=int(letal))).strftime("%d.%m.%Y")

        # Хранит данные до какого числа заражение
        # Используется для того чтобы не портить структуры зарлиста наммода
        infectBefore = self.db.get('BioWars', 'InfectionBefore')
        infectBefore[user] = vremya_do

        self.db.set('Biowars', 'InfectionBefore', infectBefore)
        old_count = ' ' + str(infList[user][0]) if user in infList else ''
        if user in infList:
            del infList[user]

        infList[user] = [str(count), vremya1]

        self.db.set("NumMod", "infList", infList)

        if message.chat_id != -1316297204:
            await message.reply(self.strings('zar.save').format(user, old_count, count,
vremya_do))

        return
    # Чат айди локдауна
    if message.chat_id == -1316297204:
        return

    if re.fullmatch(r"жд\s@\d{3,12}.{,10}", text, flags=re.ASCII):
        if str(sndr_id) != (me.id):
            return

    elif re.fullmatch(r"жл\s@\d{3,12}", text, flags=re.ASCII):
        if str(sndr_id) != str(me.id):
            return
```

```
        # ----------------------Commands------------------
        owners = list(getattr(self.client.dispatcher.security, "owner"))

        if text.startswith(pref) and sndr_id in owners:
            command = text.replace(pref, "").split()[0].lower()
            if command not in self.strings("commands"):
                return

            if command == "z":
                await self.z_command(message, args_raw, text, reply)
                return
            elif command == "id":
                await self.id_command(message, args_raw, reply)
                return
            elif command == "ids":
                await self.ids_command(message, args_raw, reply)
                return
            elif command == "dov":
                await self.dov_command(message, args_list, args_raw, reply)
                return
            elif command == 'zz':
                await self.bio_command(reply, me)
                return
            elif command == 'nik':
                await self.nik_command(message, args_list, args_raw)
                return
            elif command == 'pref':
                await self.pref_command(message, args_list)
                return

        numfilter = self.db.get("NumMod", "numfilter")
        if self.config["Вкл/Выкл доверки"] and str(sndr_id) in self.db.get("BioWars",
"DovUsers").keys() and numfilter['filter']:
            nik = numfilter['filter'].lower()

            if not text.lower().startswith(nik):
                return

            dov_users = self.db.get("BioWars", "DovUsers")

            level = dov_users[str(sndr_id)]
            # убираем из текста имя доверки
```

```
# text = text.replace(
#     f"{nik} ", '', 1).replace(f'{nik}', '', 1)

# Сделано из-за небольших проблем к командой replace
# Оно может случайно и удалить часть вводимой команды
# Пример: вир +вирусы
# Убирало вир и убирало вир из +вирусы, в итоге осталовалось +усы

text = text[len(
    nik)+1:] if f'{nik} ' in text.lower() else text[len(nik):]
text_low = text.lower()
text_norm = text

args_raw = text
args_list = text.split(' ')

if level >= 1:
    if re.fullmatch('з', text_norm) and reply:
        rtext = reply.raw_text

        if '@' in rtext:
            s = rtext.find('@')
            a = rtext[s:].split('\n')
            a = ' '.join(a).split(' ')
            user_id = a[0]

        else:
            user_id = '@' + str(reply.sender_id)

        if not user_id[1:].isdigit():
            user_id = '@' + str(await self.return_user(username=user_id))

        if user_id in infList:
            user = infList[user_id]
            infectBefore = self.db.get(
                'BioWars', 'InfectionBefore')

            zar_do = infectBefore[user_id] if user_id in infectBefore else 'Неизвестно'
            niks = self.db.get('BioWars', 'UsersNik')
            if str(user_id[1:]) in niks.keys():
                nik = niks[str(user_id[1:])]
                usr = f'<a href="tg://openmessage?user_id={user_id[1:]}">{nik}</a>'
            else:
```

```
                usr = f'<code>{user_id}</code>'

            await message.reply(self.strings('zar.search').format(usr, user[0], user[1],
zar_do))
        else:
            await message.reply(self.strings('z.nf').format(user_id))
        return

    elif send_mesa := re.search(r"з\s", text):
        en = message.entities[0]
        link = message.raw_text[en.offset:en.offset+en.length]
        user_id = await self._handler_link(link) if '@' not in link else link
        if not user_id[1:].isdigit():
            user_id = '@' + str(await self.return_user(username=user_id))

        if user_id in infList:
            user = infList[user_id]
            infectBefore = self.db.get(
                'BioWars', 'InfectionBefore')

            zar_do = infectBefore[user_id] if user_id in infectBefore else 'Неизвестно'

            niks = self.db.get('BioWars', 'UsersNik')
            if str(user_id[1:]) in niks.keys():
                nik = niks[str(user_id[1:])]
                usr = f'<a href="tg://openmessage?user_id={user_id[1:]}">{nik}</a>'
            else:
                usr = f'<code>{user_id}</code>'

            await message.reply(self.strings('zar.search').format(usr, user[0], user[1],
zar_do))
        else:
            await message.reply(self.strings('z.nf').format(user_id))

    elif mes := re.fullmatch(r'(калькулятор|к|калк) (\w+) (\d+(-\d+)?)', text_low):

        skill = mes.group(2)
        n = mes.group(3).split('-')

        if re.search(r"зз|зараз[уkаность]{,5}", text_low,  flags=re.ASCII):
            n1, n2 = int(n[0]), int(n[1])
            step = self.strings('calc_formul')['zar']
            total = 0
```

```
                for i in range(n1+1, n2+1):
                    total += int(i ** step)
                total = '{:,}'.format(total).replace(',', ' ')

                text_msg = f'🦠 Для улучшение навыка «заразность» с {n1} до {n2} уровня потребуется {total} био-ресурсов 🧬'
                await message.reply(text_msg)
                return

            elif re.search(r"(?P<let>летал[укаьность]{,5})", text_low, flags=re.ASCII):
                n1, n2 = int(n[0]), int(n[1])
                step = self.strings('calc_formul')['letal']
                total = 0
                for i in range(n1+1, n2+1):
                    total += int(i ** step)
                total = '{:,}'.format(total).replace(',', ' ')

                text_msg = f'☠️ Для улучшение навыка «летальность» {n1} до {n2} уровня потребует {total} био-ресурсов 🧬'
                await message.reply(text_msg)
                return

            elif re.search(r"(?P<pat>пат[огены]{,5})", text_low, flags=re.ASCII):
                n1, n2 = int(n[0]), int(n[1])
                step = self.strings('calc_formul')['pat']
                total = 0
                for i in range(n1+1, n2+1):
                    total += int(i ** step)
                total = '{:,}'.format(total).replace(',', ' ')

                text_msg = f'☠️ Для улучшение навыка «количество патогенов» с {n1} до {n2} потребует {total} био-ресурсов 🧬'
                await message.reply(text_msg)
                return

            elif re.search(r"(?P<kvala>квал[улаификация]{,8}|разраб[откау]{,4})", text_low, flags=re.ASCII):
                n1, n2 = int(n[0]), int(n[1])
                step = self.strings('calc_formul')['kvala']
                total = 0
                for i in range(n1+1, n2+1):
                    total += int(i ** step)
                total = '{:,}'.format(total).replace(',', ' ')
```

```
                text_msg = f'👨‍🔬 Для улучшение навыка «квалификация» {n1} до {n2} уровня потребуется {total} био-ресурсов 🧬'
                await message.reply(text_msg)
                return

            elif re.search(r"(?P<imun>иммун[уеитетка]{,4}|имун[уеитетка]{,4})", text_low, flags=re.ASCII):
                n1, n2 = int(n[0]), int(n[1])
                step = self.strings('calc_formul')['imun']
                total = 0
                for i in range(n1+1, n2+1):
                    total += int(i ** step)
                total = '{:,}'.format(total).replace(',', ' ')

                text_msg = f'🛡️ Для улучшение навыка «иммунитет» с {n1} до {n2} уровня потребуется {total} био-ресурсов 🧬'
                await message.reply(text_msg)
                return

            elif re.search(r'(?P<sb>сб|безопасно[сть]{,3}|служб[ау]{,2})', text_low):
                n1, n2 = int(n[0]), int(n[1])
                step = self.strings('calc_formul')['sb']
                total = 0
                for i in range(n1+1, n2+1):
                    total += int(i ** step)
                total = '{:,}'.format(total).replace(',', ' ')

                text_msg = f'🕵️ Для улучшение навыка «служба безопасности» с {n1} до {n2} уровня потребуется {total} био-ресурсов 🧬'

                await message.reply(text_msg)
                return
            else:
                return

        elif inf := re.search(

r"(бей{,3}|кус[ьайни]{,3}|зарази[тьть]{,3}|еб[ниажшь]{,3}|уеб[иаошть]{,3}|опуст[и]{,3}|организуй горячку{,3})",
            text_low, flags=re.ASCII
        ):
            inf = inf.group(1)
```

```
            text = text.replace(
                f"{inf} ", '').replace(inf, '')

            args_raw = text
            args_list = args_raw.split(' ')

            if args_raw.lower() == 'стоп':
                status = self.db.get("BioWars", "infStatus")
                if status:
                    self.db.set("BioWars", "infStatus", False)
                    await utils.answer(message, '✅ Заражения остановлены')
                    return
                else:
                    await utils.answer(message, '❌ Заражения не запущены!')
                    return

            if args_list[0] == 'интервал':
                if args_list[1] and args_list[1].isdigit():
                    time = float(args_list[1].replace(',', '.'))
                    self.db.set("BioWars", "infInterval", time)
                    await utils.answer(message, f'✅ Установлен интервал между
заражениями: {time} с')
                    return
                else:
                    await utils.answer(message, f'❌ Укажите интервал!')
                    return
            if send_mesa :=
re.search(r"(?P<lvl>[1-9]?[0]?\s)?(?P<link>@[0-9a-zA-Z_]+|(?:https?://)?t\.me/[0-9a-zA-Z_]+|tg://
openmessage\?user_id=(?P<id>[0-9]+))", text):
                if self.db.get("BioWars", "infStatus"):
                    await message.reply('❌ Заражения еще не завершены')
                    return

                send_mesa = send_mesa.groupdict()

                send_mesa['link'], send_mesa['id'] = '@' + \
                    send_mesa['id'] if send_mesa['id'] else send_mesa['link'], ''
                send_mesa['lvl'] = send_mesa['lvl'] or ''
                mes = ''.join(send_mesa.values())

                user = send_mesa['id'] if send_mesa['id'] else send_mesa['link']
```

```
                    user = user.replace(
                        '@', '').replace('https://t.me/', '')

                    await self.save_last_infect(user)
                    self.db.set("BioWars", "infStatus", True)
                    await message.reply(f'биоеб {mes}')
                    self.db.set("BioWars", "infStatus", False)
                    return

                await self.z_command(message, args_raw, text, reply)
                return

            elif re.search(r"вак[цинау]{,3}|леч[ись]{,2}|хи[лльсяйинг]{,2}|лек[арство]{,2}",
text_low, flags=re.ASCII):
                await message.reply('хил')
                return

            elif re.fullmatch(r"лаб[ау]{,2}", text, flags=re.ASCII):  # регулярка
                lab_raw = await self.message_q(  # отправляет сообщение ботуи
возвращает текст
                    f"биолаб",
                    5443619563,
                    mark_read=True,
                    delete=True,
                )
                if lab_raw == 'Timeout':
                    await message.respond('Время ожидание ответа от ириса истекло')
                    return

                lab_lines = lab_raw.splitlines()  # текст с лабой, разбитый     на строки
                if "🦠 Информация о вирусе" not in lab_lines[0]:
                    return
                sms = ""
                for i in lab_lines:  # цикл for по всем строкам в тексте лабы
                    if "🧪 Патогенов:" in i:
                        sms += f"{i}\n"
                    if "⏱ Новый патоген:" in i:
                        sms += f"{i}\n"
                    if "☣️ Био-опыт:" in i:
                        sms += f"{i}\n"
                    if "🧬 Био-ресурс:" in i:
                        sms += f"{i}\n"
                    if " ❗ Руководитель в состоянии горячки, вызванной  болезнью" in i:
```

```
            s = i.replace(
                " ❗ Руководитель в состоянии горячки,     вызванной болезнью ", "")
            sms += f"{s}\n"
        if " ❗ Руководитель в состоянии горячки ещё" in i:
            s = i.replace(
                " ❗ Руководитель в состоянии горячки ещё  ", "")
            sms += f"{s}\n"
    await message.reply(sms)  # ответ
    return
elif args_raw.lower() == 'зз' or args_raw.lower() == 'био':
    if not reply:
        await utils.answer(message, self.strings('no.reply'))
        return
    await self.bio_command(message, reply, me)
    return

# Не доделано
elif args_raw == 'сб':

    if not reply:
        await utils.answer(message, self.strings('no.reply'))
        return

    if re.search(r"👨‍🔬 Была проведена операция заражения", reply.text):
        entities = reply.get_entities_text()

        infect = await self._handler_link(entities[1][0].url)
        infect = infect.replace("@", "")

        # except:
        # infect = await self._handler_link(entities[0][0].url).replace("@","")

        if not infect.isdigit():
            username = infect
            infect = await self.return_user(username)
        if not infect:
            return

        if '@' + str(infect) in infList:
            user = infList['@' + str(infect)]
            infectBefore = self.db.get(
                'BioWars', 'InfectionBefore')
```

```
                    zar_do = infectBefore['@' + str(
                        infect)] if infect in infectBefore.keys() else 'Неизвестно'
                    niks = self.db.get('BioWars', 'UsersNik')
                    if str(infect) in niks.keys():
                        nik = niks[str(infect)]
                        usr = f'<a href="tg://openmessage?user_id={infect}">{nik}</a>'
                    else:
                        usr = f'<code>@{infect}</code>'

                    await message.reply(self.strings('zar.search').format(usr, user[0], user[1],
zar_do))
                else:
                    # await message.reply(f'{self.strings("z.nf").format('@' + str(infect))}'
                    await message.reply(self.strings('z.nf').format("@" + str(infect)))
                return
            # чек жертв с помощью дова
            # Пример: вир з @777000
        #    elif re.search(r"(?P<zarlist>з\s(?P<link>@[0-9a-z_]+|(?:https?#
://)?t\.me/[0-9a-z_]+|tg://openmessage\?user_id=(?P<id>[0-9]+))", # text, flags=re.ASCII):
                # pass
        if level >= 2:
            # Запись жертв с помощью дова
            # Пример: вир жд @777000
            if re.search(r"жд\s@\d{3,12}.{,10}", text_low, flags=re.ASCII):
                pass
            # Чек ежедневки

            elif re.fullmatch(r"еж[ау]{,2}", text_low, flags=re.ASCII):
                await message.reply('биоежа')

            elif send_mesa := re.fullmatch(r"(топ жертв[ыау]{,2} )(?P<list>[0-9]{,10})?", text_low,
flags=re.ASCII) or re.fullmatch(r"(топ жертв[ыау]{,2})", text_low, flags=re.ASCII):

                try:
                    send_mesa = send_mesa.groupdict()
                    n = int(send_mesa['list'])
                except:
                    n = 1

                await self.get_top_zhertv(message=message, num_list=n)
                return

            elif send_mesa := re.fullmatch(r"(жертв[ыау]{,2} )(?P<list>[0-9]{,10})?", text_low,
```

```
flags=re.ASCII) or re.fullmatch(r"(жертв[ыау]{,2})", text_low, flags=re.ASCII):
                try:
                    send_mesa = send_mesa.groupdict()
                    n = int(send_mesa['list'])
                except:
                    n = 1

                await self.get_zhertv(message=message, num_list=n)
                return
            if level >= 3:
                # Чек болезней
                if re.fullmatch(r"болезни|бол", text_low, flags=re.ASCII):
                    await message.reply('биоболь')

                # Просмотр мешка
                elif re.search(r'биомешок', text_low):
                    await message.respond('биомешок')

                if send_mesa := re.search(r"биолаб[ау]{,2}(?P<args>(\s(\w{1,12})){1,})", text_low,
flags=re.ASCII):

                    send_mesa = send_mesa.groupdict()
                    lab_args = send_mesa['args'].split()
                    lab_raw = await self.message_q(  # отправляет сообщение боту и возвращает
текст
                        f"биолаб",
                        5443619563,
                        mark_read=True,
                        delete=True,
                    )
                    if lab_raw == 'Timeout':
                        await message.respond('Время ожидание ответа от ириса истекло')

                    lab_lines = lab_raw.splitlines()  # текст с лабой, разбитый на строки
                    if "🦠 Информация о вирусе" not in lab_lines[0]:
                        return
                    sms = ""

                    args = ['d', 's', 'c', 'n', 'p', 'q', 'np', 'inf', 'imm',
                            'm', 'ss', 'be', 'br', 'so', 'prev', 'i', 'dis', 'f']
                    for arg in lab_args:
                        if arg in args:
                            for i in lab_lines:  # цикл for по всем строкам в тексте лабы
```

```
                if "🦠 Информация о вирусе" in i and arg == 'd':
                    sms += f"{i}\n"
                if "Руководитель" in i and arg == 's':
                    sms += f"{i}\n"
                if "В составе Корпорации" in i and arg == 'c':
                    sms += f"{i}\n"

                if "🏷️ Имя патогена:" in i and arg == 'n':
                    sms += f"{i}\n"
                if "🧪 Патогенов:" in i and arg == 'p':
                    sms += f"{i}\n"
                if "👨‍🔬 Разработка:" in i and arg == 'np':
                    sms += f"{i}\n"

                if "🦠 Заразность:" in i and arg == 'inf':
                    sms += f"{i}\n"
                if "🛡️ Иммунитет:" in i and arg == 'imm':
                    sms += f"{i}\n"
                if "☠️ Летальность:" in i and arg == 'm':
                    sms += f"{i}\n"
                if "🕵️ Служба безопасности:" in i and arg == 'ss':
                    sms += f"{i}\n"

                if "☣️ Био-опыт:" in i and arg == 'be':
                    sms += f"{i}\n"
                if "🧬 Био-ресурс:" in i and arg == 'br':
                    sms += f"{i}\n"

                if "😷 Спецопераций:" in i and arg == 'so':
                    sms += f"{i}\n"
                if "🥽 Предотвращены:" in i and arg == 'prev':
                    sms += f"{i}\n"
                if "🤒 Заражённых:" in i and arg == 'i':
                    sms += f"{i}\n"
                if "🤒 Своих болезней:" in i and arg == 'dis':
                    sms += f"{i}\n"

                if " ❗ Руководитель в состоянии горячки, вызванной болезнью" in i and
arg == 'f':
                    s = i.replace(
                        " ❗ Руководитель в состоянии горячки, вызванной болезнью
", "")
                    sms += f"🤒 Горячка от {s}\n"
```

```
                    if " ❗ Руководитель в состоянии горячки ещё" in i and arg == 'f':
                        s = i.replace(
                            " ❗ Руководитель в состоянии горячки ещё ", "")
                        sms += f"🤒 Горячка на {s}\n"
                else:
                    print(arg)
                    sms += f'Неизвестный аргумент: <code>{arg}</code> \n'
            await message.reply(sms)
            return

        if level == 4:
            # Прокачка навыков
            if send_mes := re.search(r"(?P<ch>зараз[куаность]{,5}
чек[нутьиай]{,4}\s|чек[айниуть]{,4} зараз[куаность]{,5}\s)(?P<lvl>[0-5]+)", text_low,
flags=re.ASCII):
                send_mes = send_mes.groupdict()
                send_mes['ch'] = '+заразность '
                send_mes['lvl'] = send_mes['lvl'] or ''
                mes = ''.join(send_mes.values())
                await message.reply(mes)

            elif send_mes := re.search(r"(?P<pat>пат[огены]{,5} чек[айниуть]\s|чек[айниуть]{,4}
пат[огены]{,5}\s)(?P<lvl>[0-5]+)", text_low, flags=re.ASCII):
                send_mes = send_mes.groupdict()
                send_mes['pat'] = '+патоген '
                send_mes['lvl'] = send_mes['lvl'] or ''
                mes = ''.join(send_mes.values())
                await message.reply(mes)
            elif send_mes := re.search(r"(?P<let>летал[каьностьу]{,5}
чек[айниуть]{,4}\s|чек[айниуть]{,4} летал[каьностьу]{,5}\s)(?P<lvl>[1-5]+)", text_low,
flags=re.ASCII):
                send_mes = send_mes.groupdict()
                send_mes['let'] = '+летальность '
                send_mes['lvl'] = send_mes['lvl'] or ''
                mes = ''.join(send_mes.values())
                await message.reply(mes)
            elif send_mes := re.search(r"(?P<kvala>квал[лаификацияу]{,8}
чек[айниуть]{,4}\s|разраб[откау]{,4} чек[айниуть]{,4}\s|чек[айниуть]{,4}
разраб[откау]{,4}\s|чек[айниуть]{,4} квал[улаификация]{,8}\s)(?P<lvl>[0-5]+)", text_low,
flags=re.ASCII):
                send_mes = send_mes.groupdict()
                send_mes['kvala'] = '+квалификация '
                send_mes['lvl'] = send_mes['lvl'] or ''
```

```
                mes = ''.join(send_mes.values())
                await message.reply(mes)
            elif send_mes := re.search(r"(?P<imun>чек[айниуть]{,4}
иммун[еитеткау]{,4}\s|чек[айниуть]{,4} имун[еитеткау]{,4}\s|имун[еитеткау]{,4}
чек[айниуть]{,4}\s|иммун[еитеткау]{,4} чек[айниуть]{,4}\s)(?P<lvl>[0-5]+)", text_low,
flags=re.ASCII):
                send_mes = send_mes.groupdict()
                send_mes['imun'] = '+иммунитет '
                send_mes['lvl'] = send_mes['lvl'] or ''
                mes = ''.join(send_mes.values())
                await message.reply(mes)
            elif send_mes := re.search(r"(?P<sb>сб чек[айниуть]{,4}\s|безопасно[сть]{,3}
чек[айниуть]{,4}\s|служб[ау]{,2} чек[айниуть]{,4}\s|чек[айниуть]{,4}
служб[ау]{,2}\s|чек[айниуть]{,4} безопасно[сть]{,3}\s|чек[айниуть]{,4} сб\s)(?P<lvl>[0-5]+)",
text_low, flags=re.ASCII):
                send_mes = send_mes.groupdict()
                send_mes['sb'] = '+безопасность '
                send_mes['lvl'] = send_mes['lvl'] or ''
                mes = ''.join(send_mes.values())
                await message.reply(mes)
# кач    алки
            elif send_mes := re.search(r"(?P<zar>зараз[уканость]{,5}\s)(?P<lvl>[0-5]+)", text_low,
flags=re.ASCII):
                send_mes = send_mes.groupdict()
                send_mes['zar'] = '++заразность '
                send_mes['lvl'] = send_mes['lvl'] or ''
                mes = ''.join(send_mes.values())
                await message.reply(mes)
            elif send_mes := re.search(r"(?P<pat>пат[огены]{,5}\s)(?P<lvl>[0-5]+)", text_low,
flags=re.ASCII):
                send_mes = send_mes.groupdict()
                send_mes['pat'] = '++патоген '
                send_mes['lvl'] = send_mes['lvl'] or ''
                mes = ''.join(send_mes.values())
                await message.reply(mes)
            elif send_mes := re.search(r"(?P<let>летал[укаьность]{,5}\s)(?P<lvl>[1-5]+)",
text_low, flags=re.ASCII):
                send_mes = send_mes.groupdict()
                send_mes['let'] = '++летальность '
                send_mes['lvl'] = send_mes['lvl'] or ''
                mes = ''.join(send_mes.values())
                await message.reply(mes)
            elif send_mes :=
```

```
re.search(r"(?P<kvala>квал[улаификация]{,8}\s|разраб[откау]{,4}\s)(?P<lvl>[0-5]+)", text_low,
flags=re.ASCII):
                send_mes = send_mes.groupdict()
                send_mes['kvala'] = '++квалификация '
                send_mes['lvl'] = send_mes['lvl'] or ''
                mes = ''.join(send_mes.values())
                await message.reply(mes)
            elif send_mes :=
re.search(r"(?P<imun>иммун[уеитетка]{,4}|имун[уеитетка]{,4}\s)(?P<lvl>[0-5]+)", text_low,
flags=re.ASCII):
                send_mes = send_mes.groupdict()
                send_mes['imun'] = '++иммунитет '
                send_mes['lvl'] = send_mes['lvl'] or ''
                mes = ''.join(send_mes.values())
                await message.reply(mes)
            elif send_mes :=
re.search(r"(?P<sb>сб\s|безопасно[сть]{,3}\s|служб[ау]{,2}\s)(?P<lvl>[0-5]+)", text_low,
flags=re.ASCII):
                send_mes = send_mes.groupdict()
                send_mes['sb'] = '++безопасность '
                send_mes['lvl'] = send_mes['lvl'] or ''
                mes = ''.join(send_mes.values())
                await message.reply(mes)

        # управление вирусами
            elif re.search(r'\+вирус[аы]{,2}|увед[ыомления]', text_low):
                await message.reply('+вирусы')

            elif re.search(r'-вирус[аы]{,2}', text_low):
                await message.reply('-вирусы')

            # Смена имени лабы и пата

            elif send_mesa := re.search(r'\+пат[оген]{,4}(?P<pat>(\s(\w{1,12})){1,})', text_norm):

                send_mesa = send_mesa.groupdict()
                pat = ' '.join(send_mesa['pat'].split())

                await message.reply(f'+вирус {pat}')
                return

            elif send_mesa := re.search(r'\+биолаб[а]{,1}(?P<lab>(\s(\w{1,12})){1,})', text_norm):
```

```
        send_mesa = send_mesa.groupdict()
        pat = ' '.join(send_mesa['lab'].split())

        await message.reply(f'+имя Лаборатории {pat}')
        return

    elif send_mesa := re.search(r'-пат[оген]{,4}', text_low):
        await message.reply(f'-вирус')
        return

    elif send_mesa := re.search(r'-лаб[а]{,1}', text_low):
        await message.reply(f'-Имя Лаборатории')
        return

# Чек фулл лабы
    if re.fullmatch(r'лаборатория', text):
        await message.respond('биолаб')
```