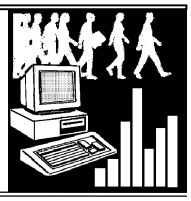
КОМПЬЮТЕРНЫЕ ТЕХНОЛОГИИ



УДК 004.55, 004.584, 004.623 DOI: https://doi.org/10.34680/2076-8052.2022.3(128).120-125

ЧАТ-БОТ ВКОНТАКТЕ РАСПИСАНИЯ УЧЕБНЫХ ЗАНЯТИЙ УНИВЕРСИТЕТА

А.А.Лихач, П.В.Татаренко*

UNIVERSITY SCHEDULE CHATBOT FOR VKONTAKTE

A.A.Likhach, P.V.Tatarenko*

Новгородский государственный университет имени Ярослава Мудрого, s244527@std.novsu.ru
**Санкт-Петербургский горный университет, Tatarenko_PV@pers.spmi.ru*

Предложена концепция интеллектуального чат-бота, совмещающего в себе возможность интеграции с популярными мессенджерами и обновляемую на основании потребностей пользователей базу знаний. Рассмотрена процедура исследования и разработки бота расписания занятий Новгородского государственного университета имени Ярослава Мудрого для социальной сети ВКонтакте на языке программирования Python. Задачей разработки являлось улучшение качества обслуживания студентов и преподавателей университета за счет внедрения чат-бота расписания. Чат-бот отображает пользователю расписание занятий по имени группы или имени преподавателя. Описаны этапы разработки от сбора данных с сайта университета до развёртывания бота на хостинге. Кроме того, рассматривается API, как мост между сайтом университета и ботом.

Ключевые слова: чат-бот, вконтакте, база расписания занятий, Python, API, web-scrapping, Django

Для цитирования: Лихач А.А., Татаренко П.В. Чат-бот ВКонтакте расписания учебных занятий университета // Вестник НовГУ. Сер.: Технические науки. 2022. №3(128). С.120–125. DOI: https://doi.org/10.34680/2076-8052.2022.3(128).120-125

The concept of an intelligent chatbot is proposed, which combines the ability to integrate with popular instant messengers and a knowledge base updated based on user needs. The research and development of the class schedule bot of Novgorod State University for the social network VKontakte in the Python programming language was carry out. The task of the development was to improve the quality of service for students and teachers of the university through the introduction of a schedule chat bot. The chatbot displays the class schedule to the user by the name of the group or the name of the teacher. The development stages are described from collecting data from the university website to deploying a bot on a hosting. The API is also considered as a bridge between the university website and the bot.

Keywords: chatbot, vkontakte, class schedule database, python, API, web-scrapping, Django

For citation: Likhach A.A., Tatarenko P.V. University schedule chatbot for VKontakte // Vestnik NovSU. Issue: Engineering Sciences. 2022. №3(128). P.120–125. DOI: https://doi.org/10.34680/2076-8052.2022.3(128).120-125

Введение

Необходимость автоматизации информационных процессов обусловлена увеличением объема данных в информационной системе организации, необходимостью ускорения и использования более сложных методов их обработки. Сегодня большинство людей регулярно пользуются социальными сетями, мессенджерами и другими приложениями, как для общения, так и для организации деятельности.

Одной из важнейших проблем качественной организации учебного процесса в высшем учебном заведении является задача формирования учебного расписания. Несмотря на то, что на сайте университета есть раздел с учебным расписанием, дополнительный функционал может упростить и улучшить взаимодействие с пользователем [1]. Этот функционал можно реализовать разными способами и один из них — чат-бот, искусственный собеседник в мессенджере, способный почти мгновенно отвечать на сообщения пользователя [2]. В случае бота расписания выбор пал на социальную сеть ВКонтакте, она популярна и почти у каждого студента на телефон скачано мобильное приложение. Главный плюс чат-бота заключается в том, что пользователю не нужно скачивать отдельное мобильное приложение с расписанием, тем самым расходуя память устройства.

Сам чат-бот реализуется как компьютерная программа, которая получает запросы, обрабатывает их и отправляет ответы. Для создания бота [3] нужны:

— сообщество, от имени которого бот будет общаться с пользователями,

— сервер, который будет принимать уведомления о событиях;

— логика самого бота — скрипт, определяющий то, как бот будет реагировать на события.

```python
import requests

URL = 'https://www.novsu.ru/univer/timetable/ochn/i.1103357/ ... &year=2019'
page = requests.get(URL)
```

Рис.1. Запрос через метод requests.get

```python
from selenium import webdriver


# start web browser
browser=webdriver.Chrome()

# get source code
browser.get("https://www.novsu.ru/univer/timetable/ochn/ ... &year=2019")
html = browser.page_source
```

Рис.2. Запрос с использованием web-driver из библиотеки Selenium

Разработка чат-бота

Функционал бота должен выдавать расписание на нужный день по запросу, высылать уведомления о предстоящих занятиях в выбранное время и об изменениях расписания. Кроме того, нам необходимо организовать автономный сбор данных расписания и развернуть его на отдельном API сервере.

Исходя из задач разработки бота, был выбран язык программирования Python [4], так как там есть готовые библиотеки, которые упростят разработку:

- vk_api — удобная обёртка для API ВКонтакте [5];
- Requests — библиотека для web-запросов;
- beautifulsoup — библиотека для анализа и обработки html;
- Django — фреймворк для создания веб-приложений.

Сбор данных

Важным вопросом при создании бота расписания является, то, как это расписание вообще получать. На сайте университета нет API для предоставления расписания в нужном формате, поэтому было решено брать расписание напрямую с сайта университета. Каждая веб-страница — это html-документ, который описывает разметку и структуру страницы и отображается браузером. Код страницы можно посмотреть и проанализировать в панели разработчика (в Google Chrome она вызывается клавишей F12), сам код находится во вкладке elements. Код таблицы с расписанием находится внутри элемента <table … class=”schedule”>. Таблица состоит из строк: есть строки с названием дня, и строки с расписанием одного занятия.

При попытках получения html кода страницы через обычный метод библиотеки Requests requests.get программа (рис.1) выполнялась без ошибок, но в полученном html-документе отсутствовало расписание занятий, и был лишь шаблон страницы, хотя браузер показывал правильный html-документ с расписанием.

На сайте расписание генерируется внутри html-дерева при помощи JavaScript, исполняемого в браузере, и для получения нужного html-кода в программе необходимо так же выполнять JavaScript с веб-страницы. Для этого можно воспользоваться библиотекой Selenium. Инстанцию браузера создаём через web-driver (можно использовать Chrome Driver), теперь get-запрос доставляет html-код страницы со всем расписанием, и уже с этим можно работать и извлекать нужные данные (рис.2).

Для хранения расписания одной группы была выбрана структура, изображённая на рис.3.

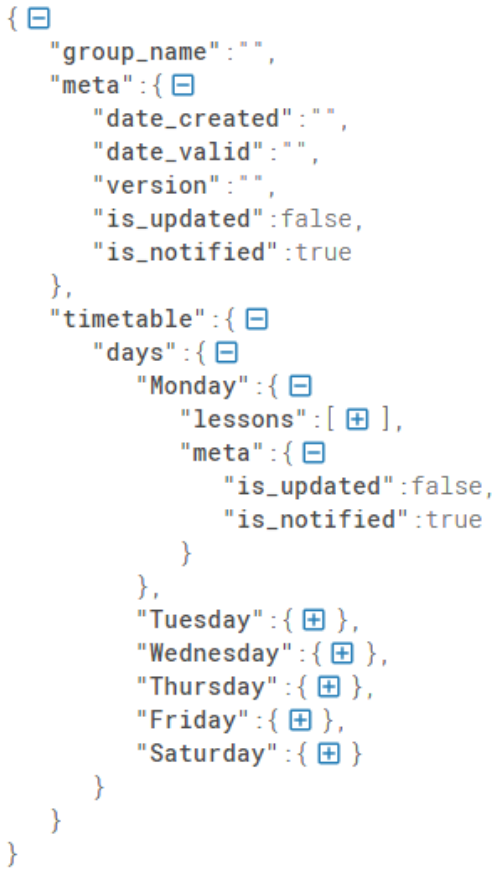


Рис.3. Структура объекта расписания группы

В поле group_name находится имя группы, в поле timetable находится поле days, которое содержит 6 полей для каждого дня недели (за исключением воскресенья, так как в этот день нет занятий) Monday … Saturday. Внутри каждого дня находится массив lessons, содержащий объекты занятий (их структура будет рассмотрена позже) и два логических поля для фиксации изменений одного дня. Помимо этого, у объекта расписания группы есть поле meta, которое содержит метаданные: время создания, дату действительности, а также два логических поля для фиксации изменений уже всей недели.

Структура объекта занятия выглядит аналогично. По содержанию поля объекта соответствуют именам колонок таблицы расписания с сайта.


```
"group_list":{
   "current_year":"2020",
   "meta":{
      "date":"10/12/2020 21:21:36",
      "version":"1.2"
   },
   "institutes":[
      {
         "institute_name":"IEIS_PI",
         "groups":[
            {
               "id":"0711",
               "type":"ДО",
               "year":"2020",
               "course":1,
               "name":"0711ДО"
            },
            { }
         ]
      },
      { },
      { },
      { },
      { },
      { }
   ]
}
```


Рис.4. Структура списка групп

Чтобы собрать полное расписание, необходим список всех групп. Для его получения написан скрипт, который аналогичным методом собирает данные в формате, показанном на рис.4, с сайта университета по ссылке https://www.novsu.ru/univer/timetable/ochn/.

Уже на основании списка групп можно собрать полное расписание. Формат ссылки для расписания каждой группы показан на рис.5.

При сборке расписания важно отследить изменения, для этого достаточно сравнить новое расписание с предыдущим, в Python это сделать несложно. Поскольку во время работы скрипта данные хранятся в виде словарей, то можно сравнить их простым оператором сравнения, так как оба объекта содержат одинаковые поля.

Разработка API

Чтобы оптимизировать нагрузку на бота, для хранения расписания нужно использовать отдельный API сервер, который будет выступать в роли моста между сайтом университета и ботом (рис.6).

Для создания API подойдет библиотека Django, так как там уже есть готовый набор инструментов для разработки нужного интерфейса. Важно заранее спланировать операции и маршруты, с которыми будет работать API:

• /timetable/{group_name}/{type} — возвращает расписание на всю неделю;

• /timetable/{group_name}/{type}/{day}/ — возвращает расписание для выбранного дня;

• /check-for-updates/not-notified-groups/ — возвращает список групп, у которых обновилось расписание;

• /group-list/{institute}/{course}/{type}/ — возвращает список групп, подходящих под нужные параметры.

В результате на каждый запрос API должен возвращать ответ в формате .json. Маршруты настраиваются в файле urls.py.

Сначала был написан класс Parser, который собирает необходимые данные расписания с сайта и сохраняет их в формате .json. Для взаимодействия с полученной базой данных был написан класс Explorer, который инициализируется .json файлом с расписанием и реализует следующие методы:

• search_week(self, group_number, _type) — возвращает расписание на всю неделю;

• search_day(self, group_number, _type, day) — возвращает расписание для выбранного дня;

```
f"https://www.novsu.ru/univer/timetable/ochn/i.1103357/?page=EditViewGroup&
instId={institute_id}&name={group_name}&type={type_}&year={year}"
```

Рис.5. Формат ссылки расписания группы

Рис.6. Схема взаимодействия внутри проекта

```python
import vk_api
from vk_api.bot_longpoll import VkBotLongPoll, VkBotEventType

vk_session = vk_api.VkApi(token='token')
longpoll = VkBotLongPoll(vk_session, group_id='group_id')

for event in longpoll.listen():
    if event.type == VkBotEventType.MESSAGE_NEW:
        message_text = event.object.message['text']
        print(message_text)
```

Рис.7. Установка соединения через vk-api

• get_not_notified_group_list(self) — возвращает список групп, у которых обновилось расписание.

В файле handler.py создаётся инстанция класса Explorer для работы с базой расписания и там же объявляются методы, обрабатывающие маршруты, затем handler.py импортируется в urls.py. Так же был написан класс Collector, который обновляет базу расписания: скрипт с обновлением должен выполняться каждый день в фиксированное время и делать это параллельно с работающим API.

Развертывание API

Для развертывания был выбран сервис PythonAnywhere, так как он удобен и прост при начальной установке, а также обладает инструментами мониторинга для отслеживания ошибок. Для развертывания необходимо выполнить следующие шаги [6]:

— загрузить код на платформу PythonAnywhere. Это делается через bash-терминал клонированием репозитория из GitHub;

— настроить виртуальную среду, установить Django и сопутствующие библиотеки. Это делается при помощи команды mkvirtualenv и стандартного менеджера пакетов pip. Сам Python установлен по умолчанию;

— настроить веб-приложение на платформе PythonAnywhere и отредактировать файл wsgi.ru.

Теперь нужно настроить скрипт с обновлением. Во вкладкеTasks выбираем пункт Scheduledtasks, выбираем нужный скрипт и устанавливаем частоту и время выполнения. Осталось протестировать все методы, и API готов к постоянной работе.

Разработка бота

Для работы бота необходимо создать сообщество ВКонтакте, от лица которого бот будет принимать и отправлять сообщения. После создания группы первым делом нужно разрешить сообщения сообщества, а также разрешить добавлять сообщество в беседы, это делается во вкладке «*Сообщения»* в настройках сообщества. Во вкладке «*Работа с API»* нужно сгенерировать ключ доступа, он будет использоваться в программе для инициализации соединения.

Простейший скрипт бота на языке Python с использованием библиотеки vk-api [5] показан на рис.7. Импортируем нужные библиотеки и инициализируем сессию, используя ключ доступа, устанавливаем соединение LongPoll и через итератор обрабатываем каждое событие.

Из объекта event можно получить много полезной информации:

• type — тип события. Нужные нам события имеют тип MESSAGE_NEW;

• user_id/chat_id — id пользователя ВКонтакте, или же относительный id беседы (нумерация начинается с 1);

• from_user/from_chat — определяют, от кого получено сообщение.

Пользователю нужно указать данные своей группы, чтобы бот узнал какое расписание нужно высылать. Исходя из того, что пользователя можно определить только по user_id/chat_id, бот должен запоминать пользователя и данные, введённые им. Для этих целей нужно вести базу данных пользователей. Модель пользователя показана на рис.8. Поле notifications определяет то, получает ли пользователь рассылку изменений расписания. Поля today_mail и tomorrow_mail хранят время рассылки расписания на сегодня и на завтра или же –1, если рассылка не требуется. Поле mail_error_counter — это счётчик ошибок отправки сообщения, они могут возникать, если пользователь запретил сообщения сообщества, тогда нужно удалить его из базы данных, дабы не возникало исключений.

```
user = {
    "vk_type": 'user' or 'chat',
    "chat_id": -1 if empty,
    "user_id": -1 if empty,
    "institute": institute_name,
    "_type": 'ДО' or 'ВО' or 'ВУ',
    "course": n,
    "group_id": group_id,
    "notifications": True or False,
    "stage": ... ,
    "today_mail": ... ,
    "tomorrow_mail": ... ,
    "mail_error_counter": ...
}
```

Рис.8. Модель пользователя

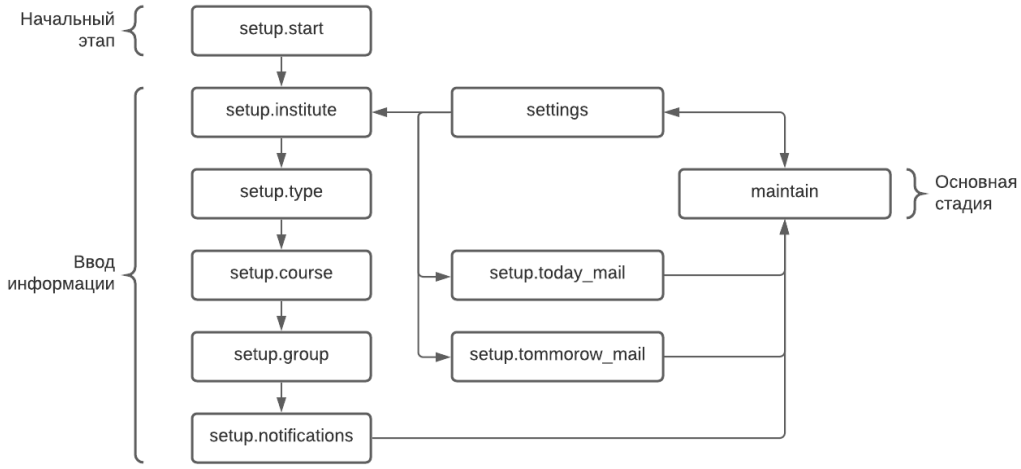


Рис.9. Карта этапов

Для работы с ботом были выделены три главных этапа. Текущий этап хранится в поле stage и никак не показывается пользователю, дабы сохранить инкапсуляцию. Этапы:

• setup — стадия, на которой пользователь вводит данные группы, а также настраивает уведомления;

• maintain — это стадия, на которой бот уже знает всю информацию и работает в обычном режиме, т. е. выдает расписание на выбранный день и отправляет уведомления;

• settings— стадия для перенастройки бота: изменить время рассылки, заново выбрать группу.

На рис.9 показана подробная карта этапов. У каждого этапа нужно определить:

— сообщение, которое будет отправлять пользователю. К примеру, на этапе setup.institute бот должен отправить список институтов;

— шаблон правильного сообщения, при котором происходит переход на новый этап. На этапе setup.institute правильным сообщением будет аббревиатура названия института (ИЭИС, ИПТ и т.д.);

— данные пользователя, которые нужно изменить при правильном сообщении. При правильном сообщении на этапе setup.institute нужно изменить поле institute на значение, которое указал пользователь, а также перейти на новый этап, присвоив полю stage значение setup.type.

Перейдем к программной реализации. Для базы данных используется легковесная библиотека TinyDB [7], которая реализует функционал документно-ориентированной системы управления базами данных. На её основе написан класс Users DB, который включает в себя методы поиска по базе данных на основе разных параметров.

Для обращения к API написан класс APICollector. Он содержит методы, которые при помощи библиотеки Requests выполняют get-запросы к API и возвращают полученные данные в формате объекта словаря. Для каждого этапа (setup, maintain, settings) написан отдельный Handler-класс, который определяет сообщения, отправляемые пользователю, а также шаблоны корректных сообщений, полученных от пользователя.

У чатов ВКонтакте есть специальный функционал — это клавиатуры. Он позволяет упростить взаимодействие с ботом, заменив символьную клавиатуру на набор кнопок. Для генерации клавиатур используется класс KeyboardBuilder. Также был написан класс Timetable Message для генерации строковых сообщений из объекта расписания. Теперь осталось написать функции рассылки расписания. Это нужно выполнять параллельно с основным циклом, что делается при помощи встроенного модуля threading и библиотеки Schedule [8]. Код показан на рис.10.

```
def do_schedule():
    schedule.every().day.at('19:40').do(send_timetable_updates)

    # sending today and tommorow timetables
    for i in range(24):
        plan_time = get_time(i)
        schedule.every().day.at(plan_time).do(send_timetable_today, i)
        schedule.every().day.at(plan_time).do(send_timetable_tommorow, i)

    while True:
        schedule.run_pending()
        time.sleep(1)

thread = Thread(target=do_schedule, daemon=True)
thread.start()
```

Рис.10. Рассылка расписания

Для развертывания также был выбран сервис PythonAnywhere. Нужно клонировать GitHub репозиторий, настроить виртуальную среду и запустить основной скрипт бота как постоянную задачу. Это делается во вкладке Tasks раздел Always-ontasks. Теперь скрипт всегда запущен и бот постоянно активен.

Заключение

Бурное развитие коммуникационных технологий обеспечило доминирование мобильных устройств над персональными компьютерами. В то же время массовое развитие и доступность информационных и коммуникационных услуг сделали социальные сети, интерактивные чат-боты и мобильные системы обмена сообщениями (мессенджеры) основными технологиями поиска и обмена информацией. Идея чат-бота — виртуального собеседника — в целом не нова. История чат-ботов восходит к 1966 г., когда Вейзенбаум написал компьютерную программу под названием ELIZA. Она имела всего 200 строк кода [9]. На сегодняшний день чат-боты прошли эволюцию от достаточно примитивных программ с минимальной функциональностью, основанных исключительно на заложенных в них шаблонах и ограниченном малом наборе правил, до продвинутых программ, не только общающихся с пользователем, но и автоматизирующих выполнение рутинных задач. Сегодня многие пользователи активно используют различных ботов, которых считают полезными и удобными.

Статья посвящена разработке бота ВКонтакте, решающего проблему наличия расписания в учебном заведении [10]. Проект реализован с использованием Python. В результате мы получили простой и в то же время полезный бот ВКонтакте. Некоторые студенты полностью перешли на использование ботов в качестве основного источника информации о расписании тренировок.

Практическая значимость заключается в том, что этот чат-бот может использоваться в любой компании, у которой есть бизнес-процесс обработки запросов клиентов в чате [11].

Сегодня боты стали полноценной альтернативой мобильной версии сайта. Они предоставляют не только более адаптированный к мобильным устройствам сервис, но и дополнительные функции в виде рассылки расписания, которую на мобильной версии не сделать.

1. Дейнеко Т.А. Переход к автоматизированному составлению расписания учебного процесса в ОмГУ // Математическое и компьютерное моделирование: Сб. мат. VI Междунар. науч. конф. Омск, 23 ноября 2018. Омск: ОмГУ, 2018. С.170–171.
2. Yan M., Castro P., Cheng P., Ishakian V. Building a Chatbot with Serverless Computing // Proceedings of the 1st International Workshop on Mashups of Things and APIs (MOTA'16). 2016. Article number: 5. DOI: https://doi.org/10.1145/3007203.3007217
3. API for chat bots VK [Электронный ресурс]. URL: https://vk.com/dev/bots_docs (дата обращения: 9.06.2022).
4. Косарев О.В. Онлайн курс «Основы программирования в Python» сетевой академии Cisco как инструмент для самостоятельной работы студентов // Современные образовательные технологии в подготовке специалистов для минерально-сырьевого комплекса: Сб. науч. тр. II Всерос. науч. конф. Санкт-Петербург, 27–28 сентября 2018 г. СПб.: Санкт-Петербургский горный университет, 2018. С.194–200.
5. Documentation vk_api [Электронный ресурс]. URL: https://vk-api.readthedocs.io/en/latest/ (дата обращения: 9.06.2022).
6. Deploying Django project on PythonAnywhere. URL: https://help.pythonanywhere.com/pages/DeployExistingDjangoProject (дата обращения: 9.06.2022).
7. Tiny DB documentation [Электронный ресурс]. URL: https://tinydb.readthedocs.io/en/latest/ (дата обращения: 9.06.2022).
8. Schedule documentation [Электронный ресурс]. URL: https://schedule.readthedocs.io/en/stable/ (дата обращения: 9.06.2022)
9. Weizenbaum J. Computer power and human reason: from judgment to calculation. San Francisco: W.H.Freeman and Company., 1976. 300 p
10. Катунцов, Е. В. Способы мотивации обучения студентов вуза в области информационных технологий // Современные образовательные технологии в преподавании естественно-научных и гуманитарных дисциплин: Сб. науч. тр. II Междунар. науч.-метод. конф., Санкт-Петербург, 09–10 апреля 2015 г. .СПб.: Национальный минерально-сырьевой университет «Горный», 2015. С.368–372.
11. Minakov V.F., Lobanov O.S., Makarchuk T.A. et al. Dynamic management model of innovations generations // Proceedings of 2017 20th IEEE International Conference on Soft Computing and Measurements. 2017. Article number: 7970743. P.849–852. DOI: https://doi.org/10.1109/SCM.2017.7970743

References

1. Deyneko T.A. Perekhod k avtomatizirovannomu sostavleniyu raspisaniya uchebnogo protsessa v OmGU [Transition to automated scheduling of the educational process in Omsk State University]. Matematicheskoye i komp'yuternoye modelirovaniye: Sb. mat. VI Mezhdunar. nauch. konf. [Mathematical and computer modeling: Proc. of the VI Intern. scientific conf.]. Omsk, 2018, pp. 170–171.
2. Yan M., Castro P., Cheng P., Ishakian V. Building a Chatbot with Serverless Computing. Proceedings of the 1st International Workshop on Mashups of Things and APIs (MOTA '16), 2016, art. no. 5. doi: https://doi.org/10.1145/3007203.3007217
3. API for chat bots VK. Available at: https://vk.com/dev/bots_docs (accessed: 09.06.2022).
4. Kosarev O.V. Onlayn kurs «Osnovy programmirovaniya v Python» setevoy akademii Cisco kak instrument dlya samostoyatel'noy raboty studentov [Online course "Fundamentals of Python Programming" of the Cisco Network Academy as a tool for independent work of students]. Sovremennyye obra-zovatel'nyye tekhnologii v podgotovke spetsialistov dlya mineral'no-syr'yevogo kompleksa: Sb. nauch. tr. II Vse-ros. nauch. konf. [Modern educational technologies in training specialists for the mineral resource complex: Proc. of the II All-russian scientific conf.]. Saint-Petersburg, 2018, pp. 194–200.
5. Documentation vk_api. Available at: https://vk-api.readthedocs.io/en/latest/ (accessed: 09.06.2022).
6. Deploying Django project on PythonAnywhere. Available at: https://help.pythonanywhere.com/pages/DeployExistingDjango Project (accessed: 09.06.2022).
7. Tiny DB documentation. Available at: https://tinydb.readthedocs.io/en/latest/ (accessed: 09.06.2022).
8. Schedule documentation. Available at: https://schedule.readthedocs.io/en/stable/ (accessed: 09.06.2022)
9. Weizenbaum J. Computer power and human reason: from judgment to calculation. San Francisco: W.H.Freeman and Company Publ., 1976. 300 p
10. Katuntsov Ye.V. Sposoby motivatsii obucheniya studentov vuza v oblasti informatsionnykh tekhnologiy [Ways of motivating the education of university students in the field of information technology]. Sovremennyye obrazovatel'nyye tekhnologii v prepodavanii yestestvenno-nauchnykh i gumanitarnykh distsiplin: Sb. nauch. tr. II Mezhdunar. nauch.-metod. konf. [Modern educational technologies in teaching natural sciences and the humanities: Proc. of the II International scientific methodological conference]. Saint-Petersburg, 2015, pp. 368-372.
11. Minakov V.F., Lobanov O.S., Makarchuk T.A., et al. Dynamic management model of innovations generations. Proc. of the 20th IEEE International Conference on Soft Computing and Measurements. 2017, art. no. 7970743, pp. 849–852. doi: https://doi.org/10.1109/SCM.2017.7970743