|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **Министерство науки и высшего образования Российской Федерации**  Калужский филиал  федерального государственного бюджетного  образовательного учреждения высшего образования  ***«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»***  ***(КФ МГТУ им. Н.Э. Баумана)*** |

|  |  |
| --- | --- |
| **ФАКУЛЬТЕТ** | **ИУК «Информатика и управление»** |
| **КАФЕДРА** | **ИУК4 «Программное обеспечение ЭВМ,** |
| **информационные технологии»** | |

**Лабораторная работа №4**

**«Технологии машинного обучения в облачной**

**системе Yandex Cloud»**

**ДИСЦИПЛИНА: «Облачные технологии»**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Выполнил: студент гр. ИУК4-82Б | |  |  | ( | Сафронов Н.С. | ) |
|  |  |  | (подпись) |  | (Ф.И.О.) |  |
| Проверил: | |  |  | ( | Амеличев Г.Э. | ) |
|  |  |  | (подпись) |  | (Ф.И.О.) |  |

|  |  |
| --- | --- |
| Дата сдачи (защиты):  Результаты сдачи (защиты): | |
|  | - Балльная оценка:  - Оценка: |

Калуга, 2024

**Цель работы:** приобретение практических навыков по работе с сервисом Yandex Cloud.

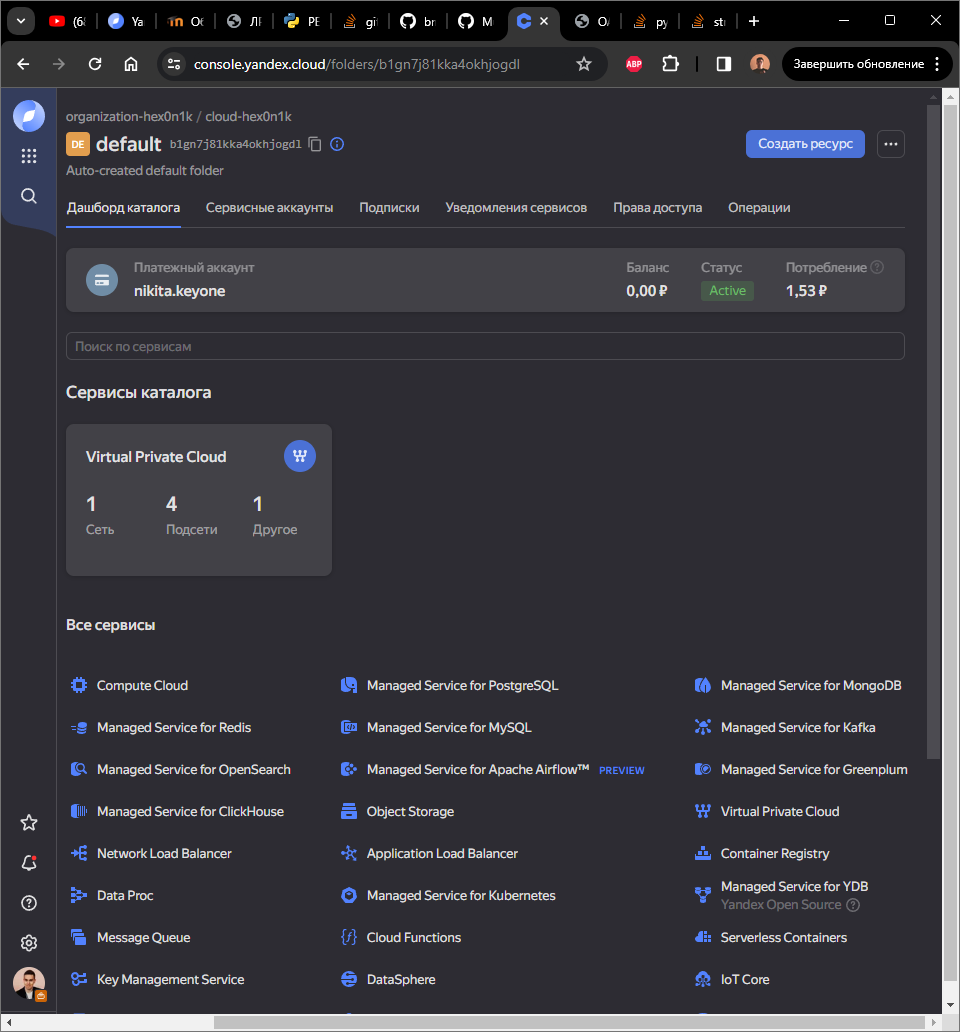
**Постановка задачи**

Решите задачу согласно варианту с использованием платформы Yandex Cloud.

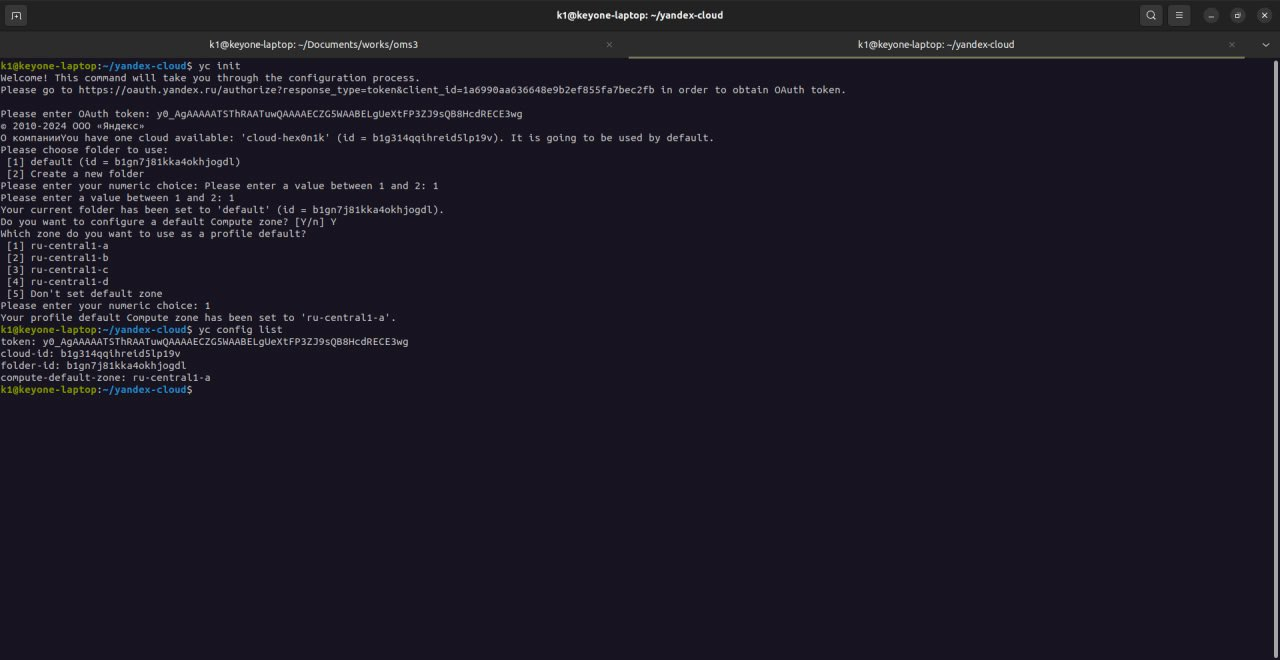
**Вариант 4**

С помощью сервиса Yandex Cloud Translate перевести на русский язык машинно-сгенерированный текст на английском языке.

**Результаты выполнения работы**



**Рисунок 1 –** Созданная учётная запись



**Рисунок 2 –** Создание профиля

**Листинг программы**

**Файл main.py:**

from wrappers.cli import CliWrapper

from services.translate import TranslationService

from services.gpt import GptService

import sys

if \_\_name\_\_ == '\_\_main\_\_':

cli\_wrapper = CliWrapper()

config = cli\_wrapper.parse\_config(sys.argv[1:])

gpt\_service = GptService(

token=config['token'],

folder\_id=config['folder\_id'],

api\_url=config['gpt\_api\_url']

)

print(f'[GPT] Started generation for prompt {config["prompt"]}...')

text = gpt\_service.get\_generated\_text(config['prompt'])

print('[GPT] Text was successfully generated:', text)

translations\_service = TranslationService(

token=config['token'],

target\_language=config['target\_language'],

folder\_id=config['folder\_id'],

api\_url=config['translations\_api\_url']

)

print(

f'[Translation] Started translation to {config["target\_language"]}',

'for generated text...'

)

translation = translations\_service.get\_translation(text)

print('[Translation] Text was successfully translated:', translation)

**Файл models/base.py:**

import typing as t

ModelType = t.TypeVar('ModelType')

def load\_model(cls: t.Type[ModelType], data: t.Mapping) -> ModelType:

required = cls.\_\_dict\_\_.get('\_\_required\_keys\_\_', frozenset())

optional = cls.\_\_dict\_\_.get('\_\_optional\_keys\_\_', frozenset())

model = cls(

\*\*{key: data[key] for key in required},

\*\*{key: data[key] for key in optional if key in data},

)

return model

**Файл models/config.py:**

"""."""

import os

import typing as t

class Config(t.TypedDict):

"""Конфигурация для сервиса перевода Yandex Cloud"""

token: str

folder\_id: str

target\_language: str

translations\_api\_url: str

gpt\_api\_url: str

prompt: str

default\_config = Config(

token=os.environ.get('YC\_TOKEN', ''),

folder\_id=os.environ.get('YC\_FOLDER\_ID', ''),

target\_language=os.environ.get('YC\_FOLDER\_ID', ''),

translations\_api\_url='https://translate.api.cloud.yandex.net/translate/v2/'

'translate',

gpt\_api\_url='https://llm.api.cloud.yandex.net/foundationModels/v1/'

'completion',

prompt=os.environ.get('YC\_GPT\_PROMPT', ''),

)

Файл models/translation.py:

"""."""

import typing as t

class Translation(t.TypedDict):

"""."""

text: str

detectedLanguageCode: str

**Файл services/gpt.py:**

import requests

import logging

from models.translation import Translation

from models.base import load\_model

class GptService:

"""."""

def \_\_init\_\_(

self,

token: str,

folder\_id: str,

api\_url: str,

):

self.\_token = token

self.\_folder\_id = folder\_id

self.\_api\_url = api\_url

self.\_logger = logging.Logger('GPT')

def \_request\_generation(self, prompt: str) -> dict:

body = {

"modelUri": f"gpt://{self.\_folder\_id}/yandexgpt-lite/latest",

"completionOptions": {

"maxTokens": 500,

},

"messages": [

{

"role": "user",

"text": prompt,

},

]

}

headers = {

'Content-Type': 'application/json',

'Authorization': f'Bearer {self.\_token}',

}

response = requests.post(

url=self.\_api\_url,

json=body,

headers=headers,

timeout=5,

)

if not response.ok:

self.\_logger.error(

f'Unable to generate text, code={response.status\_code}'

)

raise RuntimeError('Unable to generate')

return response.json()

def get\_generated\_text(self, prompt: str) -> dict:

response = self.\_request\_generation(prompt)

single\_result = response.get('result').get('alternatives')[0]

return single\_result.get('message').get('text')

**Файл services/translate.py:**

import requests

import logging

from models.translation import Translation

from models.base import load\_model

class TranslationService:

"""."""

def \_\_init\_\_(

self,

token: str,

folder\_id: str,

target\_language: str,

api\_url: str,

):

self.\_token = token

self.\_folder\_id = folder\_id

self.\_target\_language = target\_language

self.\_api\_url = api\_url

self.\_logger = logging.Logger('Translate')

def \_request\_translate(self, text: str) -> dict:

body = {

'targetLanguageCode': self.\_target\_language,

'folderId': self.\_folder\_id,

'texts': text,

}

headers = {

'Content-Type': 'application/json',

'Authorization': f'Bearer {self.\_token}',

}

response = requests.post(

url=self.\_api\_url,

json=body,

headers=headers,

timeout=5,

)

if not response.ok:

self.\_logger.error(

f'Unable to generate text, code={response.status\_code}'

)

raise RuntimeError('Unable to translate')

return response.json()

def get\_translations(self, text: str) -> list[Translation]:

raw\_translations = self.\_request\_translate(text).get('translations')

return [load\_model(Translation, x) for x in raw\_translations]

def get\_translation(self, text: str) -> Translation:

return self.get\_translations(text)[0].get('text')

**Файл wrappers/cli.py:**

import argparse

import copy

from models.config import default\_config, Config

class CliWrapper:

def \_init\_parser(self) -> argparse.ArgumentParser:

parser = argparse.ArgumentParser(

description='Yandex Cloud Translations API'

)

parser.add\_argument('prompt', type=str)

parser.add\_argument('--token', type=str, required=False)

parser.add\_argument('--folder\_id', type=str, required=False)

parser.add\_argument('--target\_language', type=str, required=False)

return parser

def \_\_init\_\_(self):

self.\_parser = self.\_init\_parser()

def parse\_config(self, data: str | list[str]) -> Config:

raw\_config = vars(self.\_parser.parse\_args(data))

config = copy.deepcopy(default\_config)

if token := raw\_config.get('token'):

config['token'] = token

if folder\_id := raw\_config.get('folder\_id'):

config['folder\_id'] = folder\_id

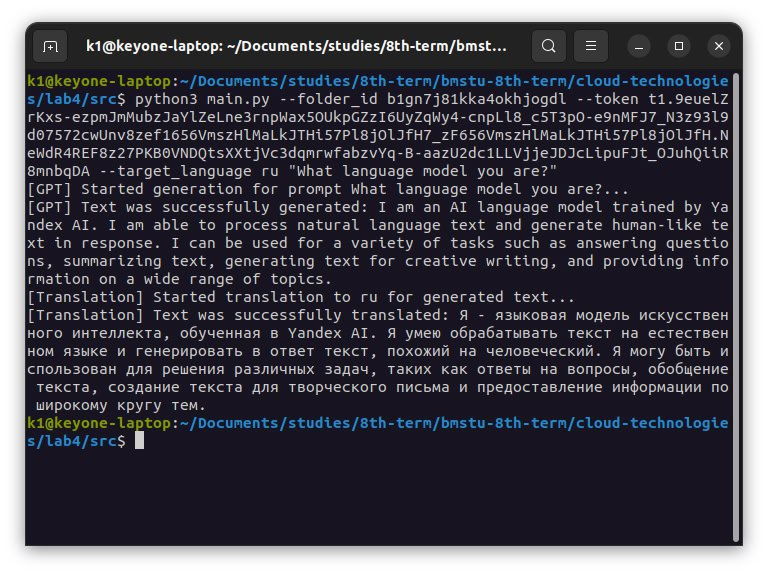
if target\_language := raw\_config.get('target\_language'):

config['target\_language'] = target\_language

config['prompt'] = raw\_config.get('prompt')

return config

**Результат выполнения программы**



**Рисунок 3 –** Результат выполнения программы

**Вывод:** в ходе выполнения лабораторной работы были получены практические навыки по работе с сервисом Yandex Cloud.