

Софийски университет
Факултет по Математика и Информатика

ПРОЕКТ

за курса

**Линукс и езиците за програмиране в
биоинформатиката**

Преподавател: Д. Василев, И. Михайлов

RESTFULL API опериращо върху данни от Ensembl

Студент: Анжелина Жан Георгиева, фак. Номер 26408

Описание на проекта

Проектът реализира RESTfull API на Python, което дава възможност да се извлича информация от базата данни на геномния браузър Ensembl (<https://www.ensembl.org/>). За реализацията са използвани Flask и BioPython.

Към API се подават заявки (GET) с идентификационния номер (ID) на съответната секвенция.

Функционалности

1. `/v1/sequence/gene/id/` - приема id на ген и да връща като резултат цялата секвенция на гена и всички екзони като отделни свойства
2. `/v1/sequence/gene/id/?gc_content=true&swap=A:T` - приема id на ген и да връща като резултат секвенцията с пресметнато процентното GC съдържание и разменени бази аденин с тимин в конкретния пример
3. `/v1/sequence/id/?content-type=fasta` - приема ID и връща секвенция в специфичен формат. Поддържаните формати са: fasta, x-fasta

Функцията **get_seq(id)** създава заявки към API-то по подадено ID на секвенцията, прави проверка дали се иска **gc_content** и/или **swap** и връща съответния резултат, ако не се искат тези параметри функцията връща секвенцията и екзоните и (за което използва предварително създадената функция **get_exons**).

Функцията **get_fasta(id)** създава заявка към API-то по подадено ID на секвенцията, прави проверка какъв тип, формат е подаден на content-type от потребителя и връща резултат във указания формат, ако подадения тип не е от зададените ни (fasta, x-fasta) връща съобщение, че не може да бъде показан в такъв формат.

Под формата на коментари в кода съм обяснила подробно изпълнението на функциите и заявките.

```
from flask import Flask, request
from Bio.SeqUtils import GC
import requests, json

app=Flask(__name__)
server = "https://rest.ensembl.org"
```

```

#функция връщаща екзоните
#Функцията връща речник от двете полета seq и object_resultE (резултата в json
формат за екзоните на секвенцията)

def get_exons(seq, object_resultE):
#Създава списък за екзоните, който се попълва с данните от резултата
(object_resultE) за избраните полета start, end, id и цялата секвенция
    exons=[]
    for e in object_resultE["Exon"]:
        exons.append({"start": e["start"],
                      "end": e["end"],
                      "id": e["id"],
                      "seq": seq[e["start"]-object_resultE["start"]:e["end"]-
object_resultE["start"]]])

    return exons #връща резултат за екзоните

#показва секвенцията и екзоните
@app.route('/v1/sequence/gene/<id>/') #метода по подразбиране е GET

#id е идентификационния номер на гена
def get_seq(id):

#id е идентификационния номер на секвенцията
#extS е частта от URL-а на заявката, която ще върне резултат за секвенцията
    extS= "/sequence/id/"+id+"?content-type=application/json"

#extE е частта от URL-а на заявката, която ще върне резултат за екзоните на
секвенцията
    extE = "/lookup/id/"+id+"?content-type=application/json&expand=1"

    gcContent=request.args.get("gc_content") #проверява за опция gc_content
    swap=request.args.get("swap") #проверява за опция swap

#Сглобява URL-а и изпраща заявките
    s = requests.get(server+extS)
    e = requests.get(server+extE)

#Взема текстовата форма на върнатия резултат
    sresult=s.text
    erezult=e.text

    object_resultS = json.loads(sresult) #json резултат за секвенцията
    object_resultE = json.loads(erezult) #json резултат за екзоните

#На променливата seq се присвоява стойността на секвенцията
    seq = object_resultS["seq"]

```

Резултат

```
else:
#Ако не се иска GC content или swar връща цялата секвенция и екзоните
return { "seq": seq, "exons": get_exons(seq, object_resultE)}
```

Резултат

127.0.0.1:5000/v1/sequence/gene/ENST00000645032/

```

{
  "end": 38291484,
  "id": "ENSE00001025723",
  "seq": "TGTAAGCTACTGTAATCAAGACTGAGGTATTGGCAAAAAGATAGACAAAGATTAATGAAACAGAATAGAGTCTACAATTAGACCCATATG",
  "start": 38291393
},
{
  "end": 38291024,
  "id": "ENSE00003474832",
  "seq": "CTTTGTAATAATAGAGATTATACTTTGGCAACAGTAAATTAGGGTTTTTAAATGTCAAT",
  "start": 38290959
},
{
  "end": 38288041,
  "id": "ENSE00003499935",
  "seq": "TTACATGAGAACTGGAGGTTCTTGGAAAACTGGACCAAGAAAAATCCAATTGACTGATAGAGACTCTCCCATGGATAGCCTGATGATGTTAATGAGGAAACGAAATACCATGAGGTGTTAACTAGAA",
  "start": 38287861
},
{
  "end": 38287245,
  "id": "ENSE00001477761",
  "seq": "GTAGTTGATCTCCGGAGTTTCGCCATGCGGAACTTGGGGGCTTTCGCGGCCGCGTCGGTGCGGAGTAGCTGCTTTAGCCCCGACCAACCGTCTCTACAGCCTCTGGCCCCCTGATGCCCGGTGAGTTGCGCCCGCGCCGTTGGCGGAAGGCCGGGAGGGAGGGTCCGCGGGCGGCCGGGCCCGCCTCCGGATCGGCCCTGTAGGGGGACAGGCCCAAGGCTTGAGGGCAGAGGACCCGGACGGGAAGAACAGGATGGGGGAGAACGGGAAAGAGGGTGCTGCTTTTCGTATGCTTTTAAATAAGCTTTGATGAGCTTTTGAAGGCAGGGACACTTGGAGTGCAATGGCGCAATCTCGGCTCACCAGCAACCTCCGCATCCCGGGTTCAGGCGATTCTCCTGCCTTAGCCTCCGAGTAGCTGCGATTACAGGCATGCGCCACCACGCCCGGGCTGATCTCGAACTCCCGACCTCAGGTGATCAGCCACCTCGGCCTCCCAAATTG+2139 more,
  "start": 38284408
},
{
  "seq": "GTAGTTGATCTCCGGAGTTTCGCCATGCGGAACTTGGGGGCTTTCGCGGCCGCGTCGGTGCGGAGTAGCTGCTTTAGCCCCGACCAACCGTCTCTACAGCCTCTGGCCCCGGCGGCCCGGTGAGTTGCGCCCGCGCCGTTGGCGGAAGGCCGGGAGGGAGGGTCCGCGGGCGGCCGGGCCCGCCTCCGGATCGGCCCTGTAGGGGGACAGGCCCAAGGCTTGAGGGCAGAGGACCCGGACGGGAAGAACAGGATGGGGGAGAACGGGAAAGAGGGTGCTGCTTTTCGTATGCTTTTAAATAAGCTTTGATGAGCTTTTGAAGGCAGGGACACTTGGAGTGCAATGGCGCAATCTCGGCTCACCAGCAACCTCCGCATCCCGGGTTCAGGCGATTCTCCTGCCTTAGCCTCCGAGTAGCTGCGATTACAGGCATGCGCCACCACGCCCGGGCTGATCTCGAACTCCCGACCTCAGGTGATCAGCCACCTCGGCCTCCCAAATTG+42404 more
}

```

#Секвенция в специфичен формат

@app.route('/v1/sequence/<id>/') #метода по подразбиране е GET

def get_fasta(id):

#fastaType приема стойността подадена за content-type

fastaType=request.args.get("content-type")

extS= "/sequence/id/"+id+"?content-type=application/json"

s = requests.get(server+extS)

sresult=s.text

object_resultS = json.loads(sresult)

seq = object_resultS["seq"]

#проверка за въведената стойност и връща съответния резултат

if (fastaType=="fasta"):

return {"seq": seq, "id": ">"+id}

elif (fastaType=="x-fasta"):

return{"seq": seq, "id": ">"+ id + " " + object_resultS["desc"]}

else:

return "Could not parse " + fastaType

Резултати:

```

{
  "id": ">ENST00000645032",
  "seq":
    "GTAGTTGATCTCCGGAGTTTCGCCATGCGGAAC TTGGGGCTTTCGCGGCCCGCGTCGGTGCGGAGTAGCTGCTTTAGCCCCGACCAA
    CCGGTGAGTTGCGCCCGCGCCGGTGGCGGAAGGCCGGGAGGGAGGGTCCGCGGGCGGCCGGGCCCCGCCTCCGGATCGGCCCTGTAGGG
    GGACGGGAAGAACAGGATGGGGGAGAACGGGAAAGAGGGTGCCTGTCTTTCGTATGCTTTTTAAATAAGCTCTTGATGAGCTTCTTGAA
    CGGCTCACC GCAACCTCCGCATCCCGGGTTCAGGCGATTCTCCTGCCTTAGCCTCCGAGTAGCTGCGATTACAGGCATGCGCCACCACG
    GATCAGCCCACCTCGGCCTCCCAAATTG+42404 more
}

```

```

{
  "id": ">ENST00000645032 chromosome:GRCh38:X:38284408:38327509:-1",
  "seq":
    "GTAGTTGATCTCCGGAGTTTCGCCATGCGGAAC TTGGGGCTTTCGCGGCCCGCGTCGGTGCGGAGTAGCTGCTTTAGCCCCGACCAAAC
    CCGGTGAGTTGCGCCCGCGCCGGTGGCGGAAGGCCGGGAGGGAGGGTCCGCGGGCGGCCGGGCCCCGCCTCCGGATCGGCCCTGTAGGGGG
    GGACGGGAAGAACAGGATGGGGGAGAACGGGAAAGAGGGTGCCTGTCTTTCGTATGCTTTTTAAATAAGCTCTTGATGAGCTTCTTGAAGG
    CGGCTCACC GCAACCTCCGCATCCCGGGTTCAGGCGATTCTCCTGCCTTAGCCTCCGAGTAGCTGCGATTACAGGCATGCGCCACCACGCC
    GATCAGCCCACCTCGGCCTCCCAAATTG+42404 more
}

```

Could not parse text

Графичен интерфейс

За създаването на графичен интерфейс е използван tkinter.

Интерфейсът съдържа поле за въвеждане на ID и бутони за връщане на резултат от създадените заявки във файла Project.py

```

import tkinter as tk
import requests

window = tk.Tk()
window.title("Sequence view")

localhost="http://127.0.0.1:5000"

def get_seq():
    id=geneID.get()
    response=requests.get(localhost+"/v1/sequence/gene/"+id)
    output.insert(tk.END,response.json())

```

```

def gcContent():
    id=geneID.get()
    response=requests.get(localHost+"/v1/sequence/gene/"+id + "?gc_content=true")
    output.insert(tk.END,response.json())

def swap():
    localHost="http://127.0.0.1:5000"
    id=geneID.get()
    response=requests.get(localHost+"/v1/sequence/gene/"+id + "?swap=A:T")
    output.insert(tk.END,response.json())

def fasta():
    id=geneID.get()
    response=requests.get(localHost+"/v1/sequence/"+id + "?content-type=fasta")
    output.insert(tk.END,response.json())

labelGene = tk.Label(window, text = "Gene ID", font=("Arial", 12))
labelGene.grid(row = 0, column = 0)
geneID = tk.Entry(window)
geneID.grid(row = 0, column = 1)

output=tk.Text(window,height=30, width=100)
output.grid(row=3, column=0, columnspan=4)

btnShow = tk.Button(window, text = "Show sequence and exons",font=("Arial", 12), command =get_seq)
btnShow.grid(row = 1, column = 0)

btnShowGC = tk.Button(window, text = "Show GC content",font=("Arial", 12), command=gcContent)
btnShowGC.grid(row = 1, column = 1 )

btnShowSwap = tk.Button(window, text = "Swap",font=("Arial", 12), command=swap)
btnShowSwap.grid(row = 1, column =2)

btnShowFasta = tk.Button(window, text = "Show in Fasta",font=("Arial", 12), command=fasta)
btnShowFasta.grid(row = 1, column = 3)

#button Clear
btnClear=tk.Button(window, text="Clear", font=("Arial", 12),command=lambda:output.delete("1.0",tk.END))
btnClear.grid(row=4, column=5)

```

```
#button QUIT
btnQuit = tk.Button(window, text="QUIT", fg="red", font=("Arial", 12), command=
window.destroy)
btnQuit.grid(row=5, column=5)

window.mainloop()
```

Резултат от графичния интерфейс.

