Staj projemde benden herhangi bir github hesabını database olarak kullanmam istenmişti. Ben daha önceden github’I aktif olarak kullanmamıştım. Çalışmama ilk olarak github’ı daha iyi kavramaya başladım.

Github’dan verileri çekmek için python dilini kullanmam istenildi. Python dili ile github’ın rest api lerini kullanarak veri çekip ekleme işlemlerini yapmam gerekiyordu. Bende ilk olarak python’ın pygithub kütüphanesini indirdim. Denemelerime onunla başladım.

Pygithub kütüphanesini kullanırken kendi github hesabıma kod seviyesinde girmem gerekiyordu. Bunu aşağıdaki gibi yaptım;

from github import Github

# First create a Github instance:

# using username and password

g = Github("user", "password")

# or using an access token

g = Github("access\_token")

# Github Enterprise with custom hostname

g = Github(base\_url="https://{hostname}/api/v3", login\_or\_token="access\_token")

# Then play with your Github objects:

for repo in g.get\_user().get\_repos():

print(repo.name)

Şekilde görüldüğü gibi üç yol ile github hesabına bağlanılıyor;

* Kullanıcı adı ve şifre
* Github developer ayarlarından üretilen access token
* Github Enterprise özel hostname ile

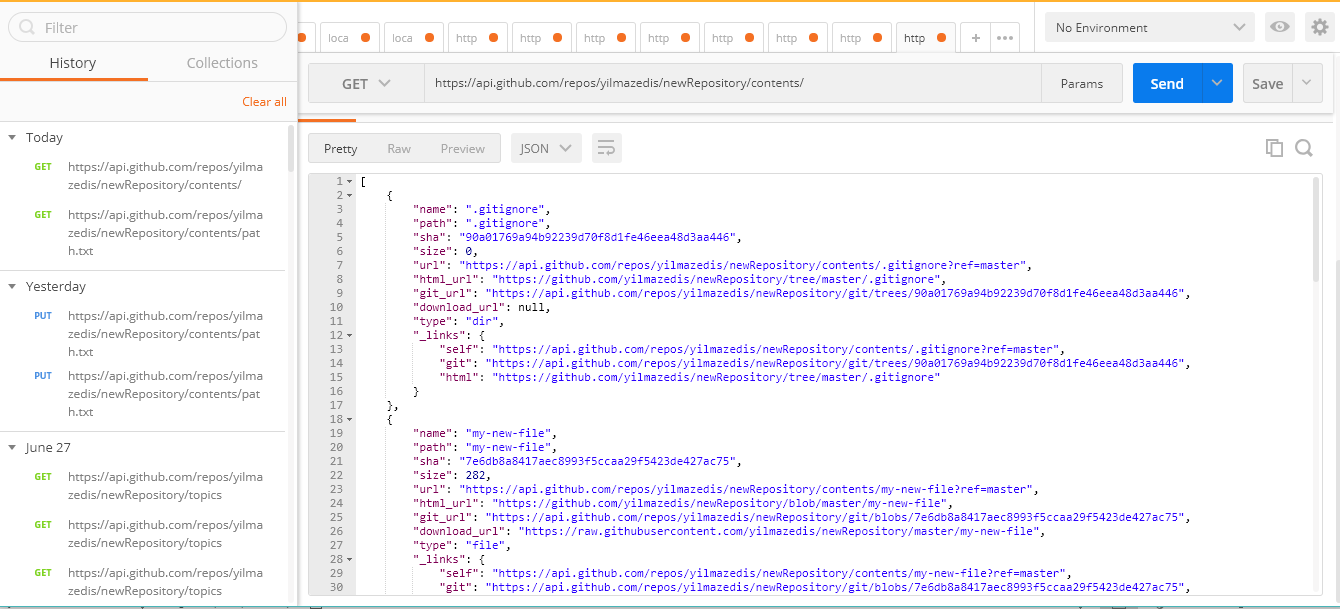
Ben bunlar arasında en çok kullanılan olan access\_token ile bağlanmayı tercih ettim. Kodda da görüldüğü gibi kendi github hesabımdaki repository’leri terminale basıyorum.

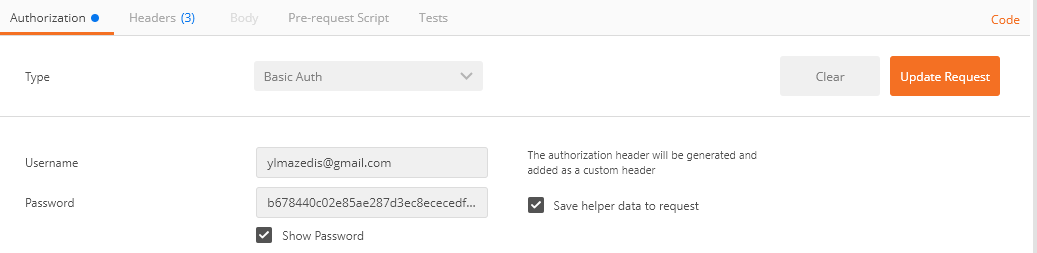
from github import Github  
  
new\_repo = "yeni-repo"  
g = Github("87e6afdd16eca703d8ee871f1d60da172074f8a5")  
user = g.get\_user()

repo = user.create\_repo(new\_repo)  
print(repo)  
  
## İlk argümanda dosyanın tam yolu belirtilmeli  
## Commit belirtilmedi  
## 3. parametre ise dosyanın içeriği  
file = repo.create\_file("/path.txt", "initial commit", "içerik\n2. satır")  
print(file)

Yukarıdaki kodda da görüldüğü gibi daha sonra olarak pygithub ile kendi github hesabıma yeni bir repository ekleme işlemini gerçekleştirdim. Sonra da bu repository içine yeni bir dosya ekledim.

Çalışmalarıma normal olarak Pycharm’da python yazarak devam ediyorum. Fakat dosya oluşturma, dosya doldurma, dosya arama gibi işlemleri hızlı gerçekleştirmem gerekiyordu. Bunu da Postman kullanrak yaptım. Postman bir bakıma kolay bir şekilde http request gönderme uygulamasıdır. Postman’de kullandığım request çeşitleri şu şekildedir;

* Get : Github hesabımdaki repoları ya da dosyaları görüntülemek için kullandım.
* Post : Yeni Repository için kullandım.

Resimde de görüldüğü gibi Get requesti atmak için <https://api.github.com> url’ini kullandım. Ek olarak /repos/:username/:reponame/contents/ şeklinde eklemeler yaptım. Veri çekmek için ise Basic Auth kullandım. Buraya github developerdan ürettiğim access\_token’ı yazdım.

Post, put, yada del yapabilmek için ise input olarak json girilmesi gerekiyor.

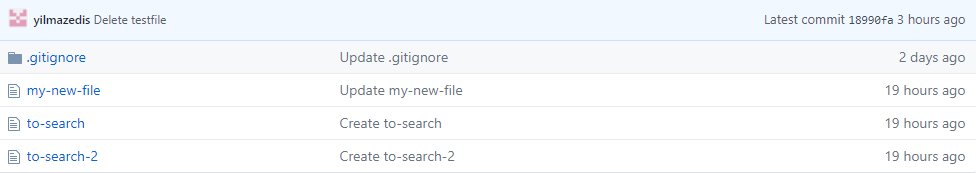
{ "message": "my commit message",

"committer": {"name": "yilmazedis", "email": "ylmazedis@gmail.com"},

"content": "aSBoYXZlIHRyaWVkIHRvIGFkZCBzb21lIGRhdGEgdG8gbXkgZmlsZQ==",

"sha": "56f6613f7c76a005c8b6d163692eba0ebe9db902"}

Yukarıdaki örnek Put işlemini gerçekleştirmek için örnek bir jsondır.

Bu gün Github’a önceki günden yüklediğim dosyalardaki textlerden arama yapmak için python kodu yazmaya başladım. Ilk olarak arama yapacağım repository’i seçtim. Test amaçlı create ettiğim repository newRepository dir. Buna “my-new-file”, “to-search”, “to-search-2” dosyalarını oluşturmuştum. 

Python ile request kütüphanesi ile request.get github daki “newRepository” deki dosyaları çektim. Daha sonra bir boş liste oluşturdum. Bu liste aradığım stringin hangi dosyada olduğunu ve bu dosyaların sha keylerini koymak içindir.

import pybase64  
import requests  
import json  
  
url\_api = 'https://api.github.com/repos/yilmazedis/newRepository/contents/'  
access\_token = 'access\_token b678440c02e85ae287d3ec8ececedf4f4c208920'  
def find\_string\_in\_github(str):

r\_repo = requests.get(url\_api, headers={'Authorization': access\_token})  
 list = []  
 for i in range(len(r\_repo.json())):  
 repo\_name = json.loads(json.dumps(r\_repo.json()[i]))["name"]  
 if repo\_name == ".gitignore":  
 continue  
 print(repo\_name)  
  
 r\_name=requests.get(url\_api+repo\_name,headers={'Authorization': access\_token})  
  
 resp\_dict = json.loads(json.dumps(r\_name.json()))  
  
 resp\_string = pybase64.standard\_b64decode(resp\_dict["content"]).decode("utf-8")  
  
 if resp\_string.find(str) != -1:  
 list.append([[repo\_name], [resp\_dict["sha"]]])  
  
 return list

list = find\_string\_in\_github("units")

print(\*list,sep='\n')

Kodun Çıktıları :

my-new-file

to-search

to-search-2

[['my-new-file'], sha : ['7e6db8a8417aec8993f5ccaa29f5423de427ac75']]

[['to-search'], sha : ['41c8bdd68eb2bd390bfafc66fe8f6b072502b28d']]

Üstteki kodda da görüldüğü gibi “newRepository” repomdaki dosyalarda “ünits” kelimesini arıyorum.Arama sonuçlarımdan repomda üç tane dosya “my-new-file”, “to-search”, “to-search-2” adlarında çıktı. Ve bunlar içinde “ünits” kelimesi “my-new-file”, “to-search” dosyalarında var. Alttaki diğer sonuçlar ise bu doyaların sha keyleridir.

Bugün şu ana kadar yaptığım işlemlere ek olarak http request ile ve python’daki Pygithub kütüphanesi ile herhangi bir dosyaya metin ekleme değiştirme işlemlerini yapmaya başladım.

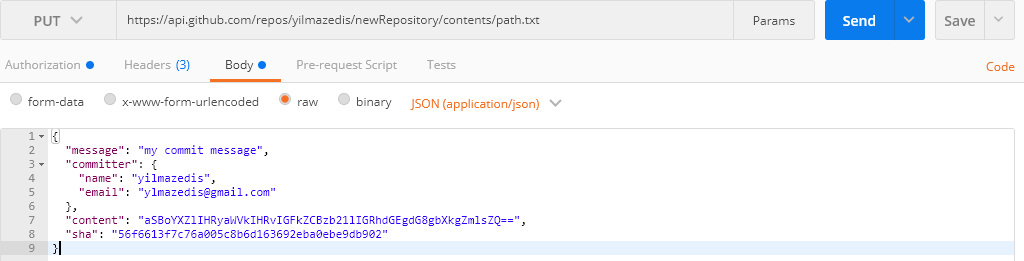
İlk olarak python’daki Pygithub kütüphanesi ile bir dosya oluşturup içine istediğim içeriği yazdım ve gönderdim.

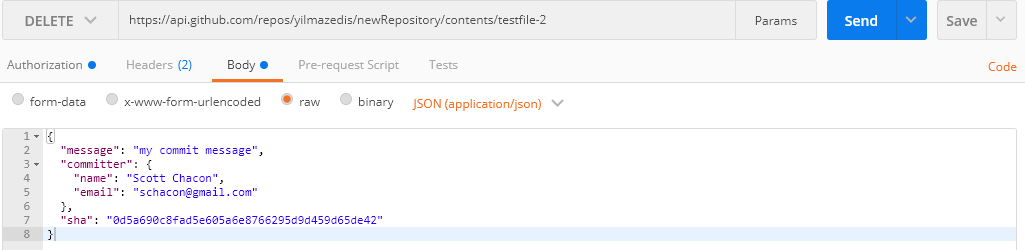
from github import Github  
  
new\_repo = "newRepository"  
g = Github("b678440c02e85ae287d3ec8ececedf4f4c208920")  
user = g.get\_user()  
  
repo = g.get\_user().get\_repo(new\_repo)  
print(repo)  
  
## İlk argümanda dosyanın tam yolu belirtilmeli  
## Commit belirtilmedi  
## 3. parametre ise dosyanın içeriği  
file = repo.create\_file("/path.txt", "initial commit", "içerik\n2. satır")  
print(file)

Kodda da görüldüğü gibi dosyaya “içerik\n2. satır” stringini yazdım. Update işlemleri için kullandığım komut ise şu şekildedir.

repo.update\_file("/path.txt", "your\_commit\_message", file\_content, file.sha)

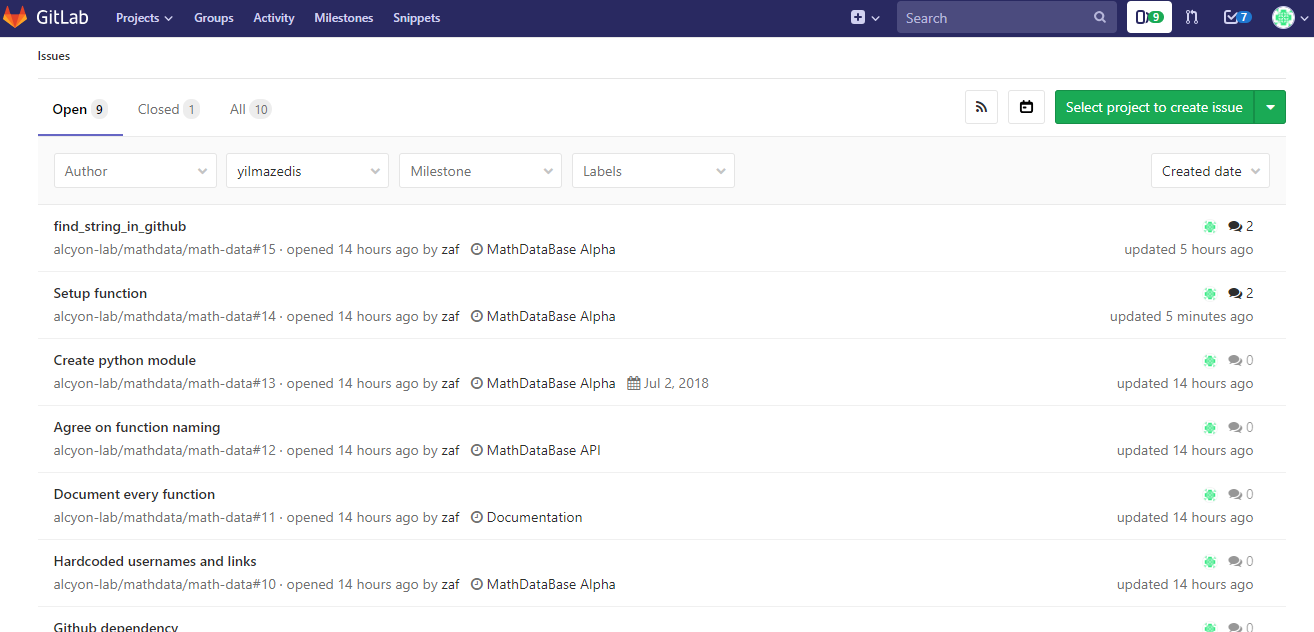
Aynı işlemleri http request ile yapmak için ise Put request işlemlerini kullandım. Kullandığım request çeşitleri şu şekildedir.

* Put : Dosya ekleme işlemi için kullandım.
* Del : Dosya silmek için kullandım.

Del işlemi için ise şu şekilde request attım.

Şekilde de görüldüğü gibi testfile-2 dosyasını silme işlemi için http request atıyorum.

Haftanın son günü olarak proje danışmanımın benden Gitlab’ta başlattığı bazı görevleri yerine getirmeğe başladım. Bu görevler ile çalışmamız daha hızlı ilerletecek. Görevler issues olarak başlatıldı. Yaptığım görevleride TODO kısmından tamamlayarak ilerledim.



Yerine getirmem gereken görevlerden birisi daha önceden gitlab’a koyduğum bazı kodlarda Türkçe yorumların kaldırılmasıydı. Bende ilk olarak ascii karakter olmayan bütün harfleri kaldırdım.

Yapacağımız çalışma cross-platform olduğu için windows ve diğer işletim sistemlerinde bulunan farklı newlinelar (“\n”, “\r”) işlemlerimizde sorun çıkartabileceğini düşündük. Bende bunun hakkında araştırma yapmaya başladım. Daha sonra python dilinde buna karşı olan önlemleri kendi içerisinde hallettiğini öğrendim. Ve bunla ilgili olan issue’i kaldırdım.

def find\_string\_in\_github(str,file\_path,access\_token):

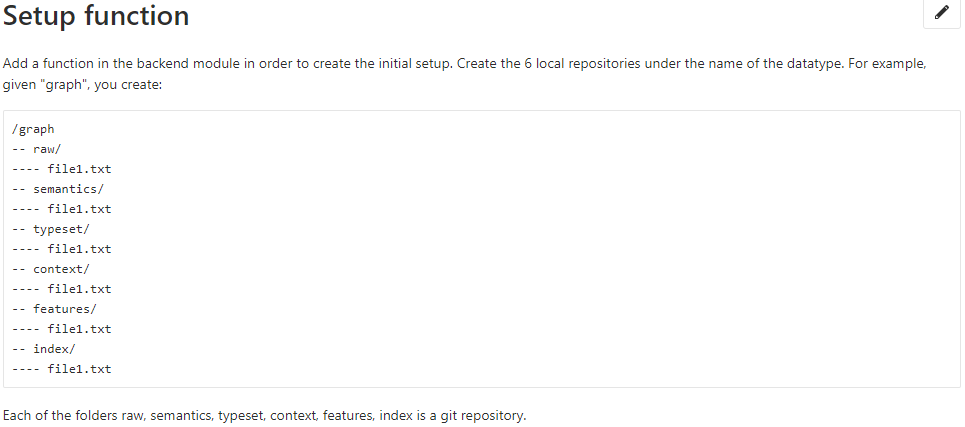
fonksiyonunun parametrelerini attırdım. Daha önceden sadece str parametresini alıyordu ve kod içerisinde belirtilen repository deki dosyalarda arama yapıyordu. Şimdi ise file\_path ve access\_token parametrelerini ekleyerek istediğimiz github hesabındaki istediğimiz repository’den arama yapabiliyoruz.

list = find\_string\_in\_github("units", 'yilmazedis/newRepository', 'b678440c02e85ae287d3ec8ececedf4f4c208920')  
  
print(\*list,sep='\n')

Uygulamamızın verilerinin saklanması için lokalde git kullanılması gerekiyordu. Bende bunu “git init” komutuyla gerçekleştirdim. “git status” ile eklediğim dosyaları yada yaptığım değişiklikleri görebiliyorum. Değişiklikleri kaydetmek için ise “git add .” komutu kullanılıyor. Bunları git günlüğünde raporlamak için ise “git commit” komutunu kullandım. Fakat dil olarak python kullandığım için ise buna uygun python kütüphanesi aradım aradım. Bulduğum kütüphane “pygit” kütüphanesidir. Kullandığım kod aşağıdaki gibidir.

from git import Repo  
import os  
  
path = '.'  
if not os.path.exists(path):  
 os.makedirs(path)  
  
os.chdir(path)  
  
repo = Repo.init(path).git  
  
index = Repo.init(path).index  
  
for x in range (1,10):  
 fname = 'filename' + str(x) + '.txt'  
 f = open(fname, 'w')  
 f.write("test content")  
 f.close()  
 repo.add(fname)  
  
index.commit("initial commit")

Bu kod ile çalışma klasörümde 10 tane dosya oluşturup bunları git’e ekliyorum.

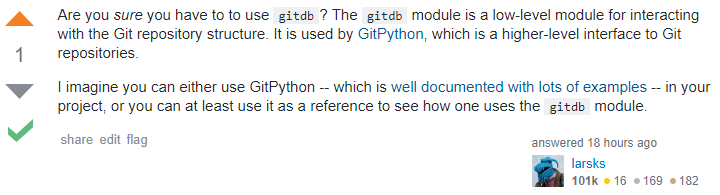
Proje danışmanımın daha önce benden istediği “initial setup” yani veri tabanının temelini oluşturan yapıyı sağladım. Yapıdaki her alt klasör bir git deposu olacak ve ekleme, silme, düzenleme, güncelleme işlemlerini bu yapıdan yapacağız. Bu işlemler yapabilmek için ise “sha” anahtarlarını kullanacağız.

Bu yapıyı oluşturan python kodum şu şekildedir.

import os, errno # this sniped code now whole code.  
def initial\_setup(datatype):  
 try:  
 os.makedirs(datatype)  
 os.chdir(datatype)  
 os.makedirs("raw")  
 os.chdir("raw")  
 os.open("file1.txt", os.O\_CREAT)  
 os.chdir("..")  
 #. . .   
 os.makedirs("index")  
 os.chdir("index")  
 os.open("file1.txt", os.O\_CREAT)  
 except OSError as e:  
 print('Cannot create! ', e.filename, ' is already exist')  
 if e.errno != errno.EEXIST:  
 raise  
initial\_setup("graph")

Projede veri tabanı oluşturmak için gitdb kullanmam istendi. Bende gitdb hakkında dokümantasyon araştırması yapmaya başladım. Araştırmalarımdan elde ettiğime göre sadece resmi sitesinde gitdb nin yapısını anlatan bir pdf buldum. Buda benim nasıl kullanacağım hakkında az bir bilgiydi. Araştırmalarıma hızlı bir şekilde devam ettim fakat kısıtlı bulduğum kaynaklardan dolayı çalışmalarım yavaşlamaya başladı. Bende günün sonuna doğru stackoverflow’da bunun hakkında bir soru sordum. Sorduğum soru aşağıdaki gibidir.



Bu soruma cevap olarak aşağıdaki şekilde bir cevapta bulundular. Bende çalışmalarıma gitpython üzerinden gitmeye karar verdim.

Gitdb alt seviye bir modüldür. Git repository ile etkileşim yapan bir yapıdır. Yani Gitpython kendi içinde gitdb’yi kullanıyor. Bir bakıma gitpython gitdb’nin yüksek seviyede bir arayüzü gibi oluyor. Ben bu araştırma ile gitdb hakkında bilgi sahibi oldum.