Facultatea Calculatoare, Informatica si Microelectronica

Universitatea Tehnica a Moldovei

Programarea in retea

Lucrarea de laborator#1

# Project Setup

*Autor:*
Alexandru Stamatin

*asistent universitar:*
Alexandru Gavrisco

# Lucrarea de laborator #1

## 1 Scopul lucrarii de laborator

Crearea unei structuri initiale a proiectului

## 2 Obiective

- Configurarea proiectului
- Studierea guideline-urilor
- Adaugarea informatiei utile in README(instructiuni de rulare a proiectului

# 3 Efectuarea lucrarii de laborator

## 3.1 Sarcinile propuse

- Crearea unui repozitoriu pe GitHub/BitBucket/GitLab
- Initializarea proiectului(cu structura corespunzatoare si .gitignore relevant)
- Adaugarea informatiei utile in README(instruciuni de rulare a proiectului pentru inceput)
- Adaugarea a cateva dummy teste
- Integrarea proiectului cu continous integration

## 3.2 Analiza lucrarii de laborator

Pentru realizarea lucrarilor de laborator la cursul "Programarea in retea" a fost ales limbajul python. Dupa studierea guideline-urilor a fost creat un repozitoriu structurat pe GitHub care a fost apoi integrat cu Continous Integration

- Crearea unui repozitoriu pe GitHub/BitBucket/GitLab

  A fost creat un repozitoriu cu denumirea "PR-Labs" pe GitHub utilizand linia de comanda prin intermediul git init. In cadrul acestuia urmeaza sa fie realizate sarcinile lucrarilor de laborator.

- Initializarea proiectului(cu structura corespunzatoare si .gitignore relevant)

  Repozitoriul a fost structurat conform guideline-urilor pentru limbajul ales si contine urmatoarele elemente:

  - setup.py
    Fisierul contine metadate despre proiect, faciliteaza instalarea modulelor si pachetelor python
  - requirements.txt
    Fisierul specifica dependentele necesare pentru a contribui la proiect.
  - PR/__init__.py
    Directorul PR contine codul proiectului. Fisierul __init__.py este inclus pentru ca Python sa trateze directorul ca pe un pachet.
  - PR/core.py
  - PR/helpers.py
  - docs/conf.py
  - tests/test.py
    Contine o suita de teste pentru modulele proiectului
  - .gitignore
    Specifica fisierele ce trebuie ignorate de catre Git

- Adaugarea informatiei utile in README(instructiuni de rulare a proiectului pentru inceput)

  In fisierul README a fost plasata o descriere scurta a proiectului si un Status Image ce arata starea curenta a proiectului obtinut prin intermediul Travis CI

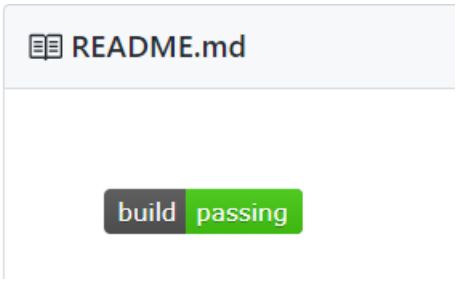


Figure 1: Status Image

- Adaugarea a cateva dummy teste

  In fisierul test.py au fost definite 2 cazuri de test pentru functiile definite in modulul de baza

```
class dummyTest(unittest.TestCase):
    def test_reverse(self):
        self.assertEqual(PR.core.reverse("Hey"),"yeH")
    def test_sqr(self):
        self.assertFalse(PR.core.square("Num"))

if __name__ == '__main__':
    unittest.main()
```

- Integrarea proiectului cu continous integration

  Proiectul a fost integrat cu Travis CI. In acest scop la repozitoriu a fost adaugat fisierul .travis.yml care contine informatie cu privire la limbajul de programare si actiunile ce trebuie executate de Travis.

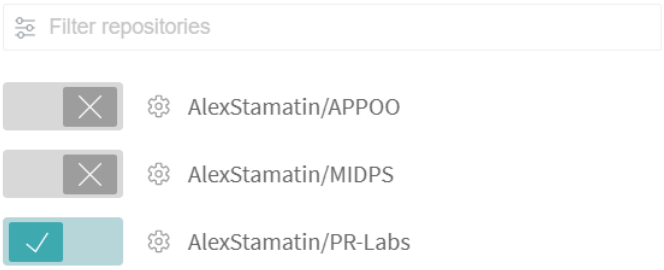


Figure 2: Selectarea repozitoriului pentru integrarea cu CI

# Concluzie

In cadrul acestei lucrari de laborator am creat un repozitoriu structurat n conformitate cu guideline-ruile pentru limbajul python, am studiat utilizarea GitHub prin intermediul liniei de comanda si modul de utilizare a CI. Dupa structurarea proiectului au fost create cateva dummy teste pentru functiile definite in modulul de baza. Integrarea cu Travis CI permite testarea automata proiectului dupa fiecare modificare operata.
In urma efectuarii lucrarii am obtinut experiente practice de utilizare a sistemului de control a versiunilor Git si de structurare a unui proiect in dependenta de limbajul de programare ales

# References

[1] The Hitchhikers Guide to Python! http://docs.python-guide.org/en/latest/writing/structure/