{iT}Factory

Întâlnirea 14

Unit testing

Sfaturi generale

- Să tratați cu seriozitate și profesionalism acest nou obiectiv.
- Cei care își ating obiectivele nu sunt întotdeauna cei mai smart, dar întotdeauna vor fi cei mai muncitori!
- Alocați-vă timp pentru studiu. Rutina dă consistență. Consistența dă excelență.
- Să faceți tot posibilul să participați la toate sesiunile live.
- Să vă lăsați comentarii explicative în cod. Notițe pentru voi din viitor.
- Recomand să vizualizați înregistrarea. Să vă notați aspectele importante + întrebări pentru trainer pentru ora următoare.
- Să vă faceți temele și unde nu reușiți singuri, să întrebați pe grup. Trainerul va răspunde și vor beneficia și ceilalți cursanți de răspuns.
- Puteți chiar să faceți un grup doar de studenți și să vă întâlniți o dată pe săptămână să discutați temele împreună. Fiecare va veni cu o perspectivă nouă și în final toți vor avea de câștigat.
- În timpul orelor, să aveți curaj să puneți întrebări când ceva nu e clar.

Reguli curs

- Va exista un sheet de prezență.
- În cadrul acestuia ne vom asuma și noțiunile învățate. Nu trecem mai departe până nu își asumă toți noile concepte.
- Temele se vor adaugă în Folderul grupei, veți face fiecare folder cu numele vostru. Veți primi feedback la aceste teme.
- Temele vor fi împărțite în 2 categorii.
  - Obligatorii (se pot face doar cu noțiunile învățate la clasă)
  - Opționale (acestea vor fi mai advanced și necesită poate extra research). Acest lucru îi va motiva și pe cei care au mai mult timp și le place să se aventureze prin task-uri mai dificile.
- Vă rog să mă întrerupeți oricând aveți întrebări. Doar așa îmi pot da seama unde trebuie să mai insist cu explicații/exemple.
- Vă rog să intrați cu 3 minute mai devreme în caz că apar probleme tehnice. Astfel putem profita la maxim de cele 2 ore alocate.
- Dacă nu puteți intra, sau dacă întârziați, anunțați trainerul pe grup.

Obiective principale

Până la final TOȚI* veți avea:

- Cunoștințe solide despre bazele programării în Python
- Cunoștințe mai avansate și extrem de utile despre programarea bazată pe obiecte.
- Capacitatea să identifice elemente și să scrie test scripts cu ajutorul Selenium
- Un Proiect final de testare automată a aplicațiilor web.
    - Acesta va folosi tendințele actuale: metodologia Behavior Driven Development și Page Object Model Design pattern.
    - Vă avea capacitatea să genereze rapoarte HTML (‘living documentation’)
    - Veți ști de la A la Z acest framework, astfel că veți avea capacitatea să continuați să îl dezvoltați post curs acest proiect (pentru orice website doriți).
- Noțiuni de bază despre API testing. (testarea backend - ce e în spate la un website).

* toți cei care sunt activi, implicați, își fac temele, dedică timp pentru studiu individual și pun întrebări trainerului vor atinge aceste obiective.

Obiective secundare

Nu fac parte din curricula cursului LIVE dar vă punem la dispoziție materiale extra că să aveți un avantaj la interviuri. Sfatul meu e să vă focusați pe ele doar după cursul live. Să nu fiți overwhelmed de new info.

- **Cunoștințe ale bazelor de date relaționale - mySQL (Curs baze de date)**
- **Cunoștințe teoretice despre testarea manuală - acces la o platformă mobilă**
- **Capacitatea de a construi un mic brand personal (Curs Portofoliu Wordpress). Trebuie să ai:**
  - **Website propriu prin care angajatorul să te cunoască pe tine și munca ta**
  - **CV european în eng / sau canva.com**
  - **Profil LinkedIn**
  - **Github public (un loc în cloud unde se pune codul scris de tine)**
  - **Veți primi feedback dacă ne trimiteți un email cu ele la hello@itfactory.ro**

Obiective Întâlnire 14

- Să știm să scriem unit tests (=> repo separat pentru angajator)
- Să avem Postman instalat
- Să cunoaștem cele mai uzuale HTTP response codes
  - 1xx, 2xx, 3xx, 4xx, 5xx
- Să înțelegem ce e un API și să știm cele mai folosite metode
  - Get, post, patch, put, delete
- Să putem să facem câteva request-uri manual în Postman

{iT} Factory

Ce este un Unit Test?

Testarea unitară reprezintă testarea la cel mai low level, practic este o funcție care testează o altă funcție.
Funcția de test apelează funcția testată și se asigură că ea returnează datele corecte, folosind un assert.
De obicei această testare este făcută de devi, dar și un qa o poate face, depinzând de raportul de forțe dev-qa. În companiile cu mai mulți angajați qa această sarcină poate fi trecută în responsabilitatea lor.

```python
# functia ce trebuie testata
def aria_dreptunghiului(self, l, L):
    return l * L
```

```python
# unit test
def test_aria_dreptunghiului():
    assert aria_dreptunghiului(3, 5) == 15, 'Aria nu se calculeaza corect'
```

{iT}Factory

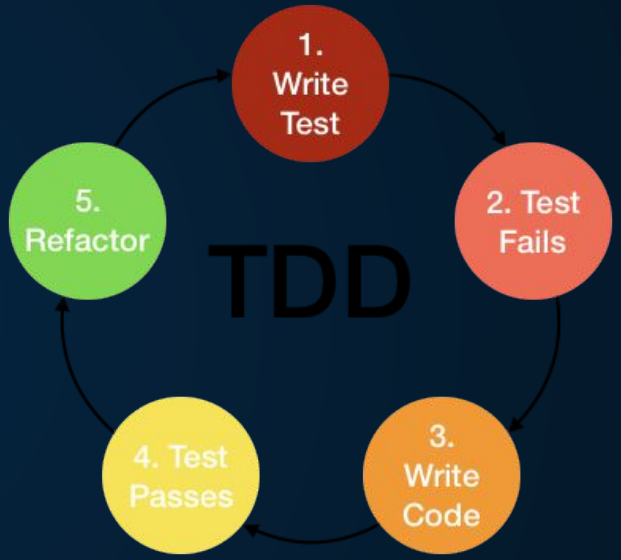
What is TDD?

DD = test driven development
Metodologie care plasează o importanță majoră pe testare.

Se folosește un test first approach.
Tot developmentul începe prin scrierea de teste.

Exercițiu:

1. Instalați pytest (librărie care ne ajuta să rulăm teste)
   pip install pytest
1. Implementați un mini calculator folosind TDD
   a. Clasa MiniCalculator
   b. Atribute: 2 numere a si b
   c. Metode goale de +,-,*,/ folosind ‘pass’
   d. Implementăm testele
   e. Executăm testele (pica)
   f. Implementăm logica
   g. Executăm testele (trec)



Cum se execută testele?
Toate din folder: ‘pytest .\folder_name\’
Un sg. Fisier: ‘pytest .\folder_name\file_name.py’

{iT} Factory

Ce este HTTP/HTTPS?

HTTP/S = Hypertext transfer protocol / secure
Protocol de comunicare între client și server.
Ne ajută să transferăm date prin rețea.

În imagine avem arhitectura standard pe 3 nivele (layers) a unei aplicații.

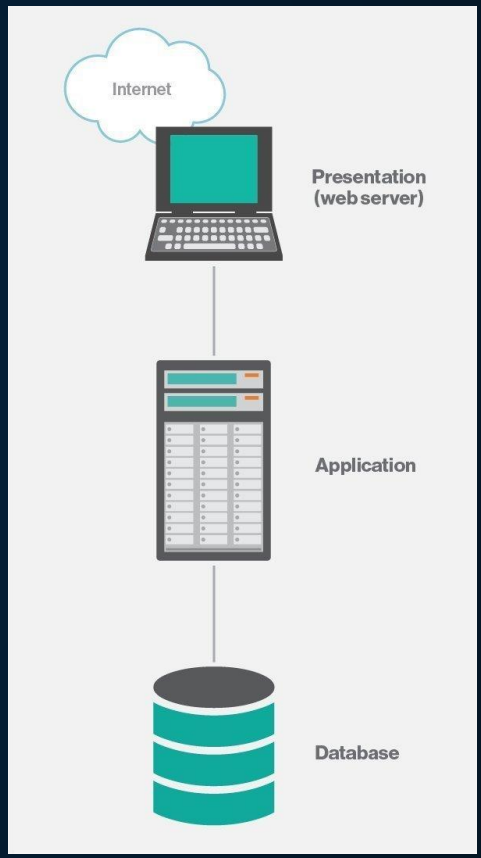
Client <--> HTTPS API requests <--> Server <--> db queries <--> Database

Testing:

UI level (user interface) / Client -> web testing (selenium - știm)

App -> Unit Testing (știm)

API level (comunicarea dintre Client și Server) -> API testing (învățăm)



{iT} Factory

Raspunsuri HTTP

https://developer.mozilla.org/en-US/docs/Web/HTTP/Status

Cele mai folosite:

- 200 OK - succes - de obicei când se cer date de la server
- 201 Created - success - când se pun date în server
- 204 No Content - success - de obicei când ștergem ceva

- 400 Bad request - ceva nu a mers bine, probabil valori invalide pentru parametri
- 401 Unauthorized - nu suntem logați în app
- 403 Forbidden - suntem logați dar nu avem drepturi de edit de exemplu
- 404 Not Found - nu găsește endpoint - probabil
- 408 Request Timeout - a durat prea mult până să ajungă la server requestul

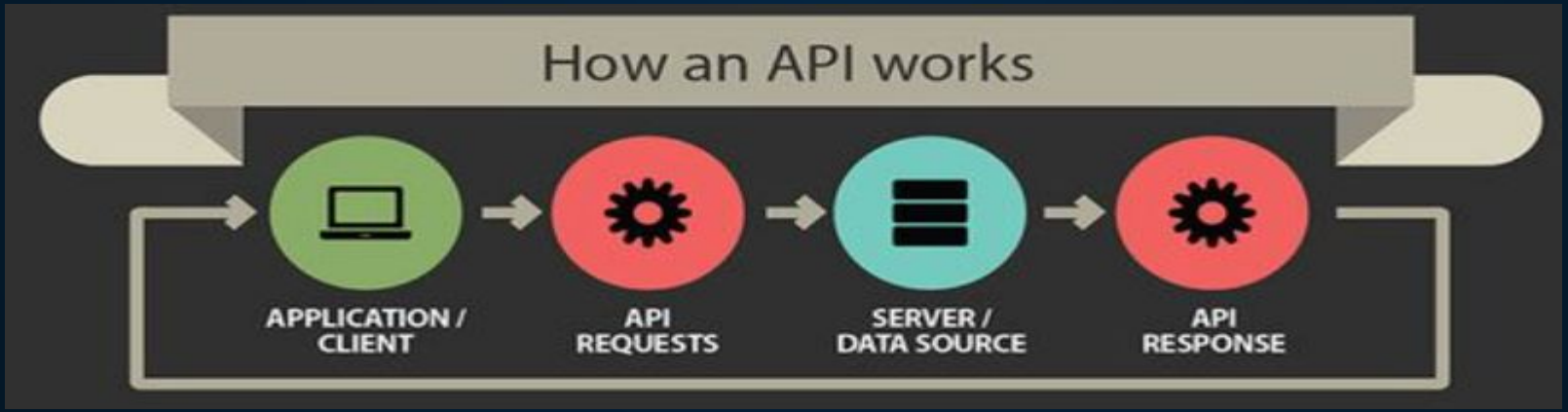
- 500 Internal Server Error - requestul ajunge la server dar cel mai probabil este un bug
- 503 Service Unavailable - serverul e oprit pentru mentenanță de exemplu

Ce este API?

API - application programming interface (‘Interfața de programare a aplicațiilor’)

Niște comenzi bine documentate care ajută programatorul sau interafața aplicației să comunice cu logica din spate.
Clientul trimite un request către un endpoint
Dacă este pornit, serverul ascultă non stop și este pregătit să primească și să proceseze aceste cerințe
După ce serverul procesează informațiile primite/cerute oferă un răspuns clientului
Clientul interpretează răspunsul și îl afișează într-un mod user-friendly utilizatorului (avantaj: un singur API poate fi folosit de orice tip de client - iOS, Android, web etc și de oricâte device-uri)

Schimbul de date se face sub forma unui JSON. Acesta are o structură asemănătoare unui dictionar din Python (cheie:valoare)



{iT}Factory

Metode API

Acronimul CRUD vine de la create, read, update, delete - acțiunile principale când lucrăm cu date în software development

Cele mai folosite metode API sunt:

GET - cerem date de la server - în general 200

POST - trimitem date la server - în general 201

PATCH - updatăm date prin updatarea anumitor atribute ale obiectului (ex: din user doar prenumele) - în general 200 sau 201

PUT - updatăm date prin suprascrierea întregului obiect (ex: tot userul - nume, prenume, adresă) - în general 200 sau 201

DELETE - ștergem date - în general 204

Exerciții:
Le vizualizăm pe toate într-un website folosind developer tools (F12 -> network)
Le implementăm în Postman
Folosim API:
https://documenter.getpostman.com/view/4012288/TzK2bEa8#abe537df-fccc-4ee6-90d2-7513e3024d6b
Implementăm un get, post, patch, put, delete
Studiem JSON-urile într-un parser: http://json.parser.online.fr/

{iT}Factory

Întrebări de interviu:

- De la ce vine TDD? Ce este particular în această metodologie de dezvoltare?
- Ce este un unit test?
- Cu ce cifră încep request-urile HTTP de succes?
- Care e diferența dintre response code 4xx și 5xx?
- De la ce vine API?
- Care sunt metodele principale API? (5)
- Care e diferența dintre patch și put?

Întrebări & curiozități?