
{iT}Factory

# Intalnirea 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!.
- Alocă-ți 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 prezenta.
- In cadrul acestuia ne vom asuma si notiunile invatate. Nu trecem mai departe pana nu isi asuma toti noile concepte.
- Temele se vor adauga in Folderul grupei, veti face fiecare folder cu numele voastru. Veti primi feedback la aceste teme.
- Temele vor fi impartite in 2 categorii.
  - Obligatorii (se pot face doar cu notiunile invatate la clasa)
  - Optionale (acestea vor fi mai advanced si necesita poate extra research). Acest lucru ii va motiva si pe cei care au mai mult timp si le place sa se aventureze prin task-uri mai dificile.
- Va rog sa ma intrerupeti oricand aveti intrebari. Doar asa imi pot da seama unde trebuie sa mai insist cu explicatii/exemple.
- Va rog sa intrati cu 3 minute mai devreme in caz ca apar probleme tehnice. Astfel putem profita la maxim de cele 2 ore alocate
- Daca nu puteti intra, sau daca intarziati, anuntati trainerul pe grup

# Obiective principale

Pana la final TOTI* veti avea:

- Cunostinte solide despre bazele programarii in Python
- Cunostinte mai avansate si extrem de utile despre programarea bazata pe obiecte.
- Capacitatea sa identifice elemente si sa scrie test scripts cu ajutorul Selenium
- Un Proiect final de testare automata a aplicatiilor web.
    - Acesta va folosi tendintele actuale: metodologia Behavior Driven Development si Page Object Model Design pattern.
    - Va avea capacitatea sa genereze rapoarte HTML (‘living documentation’)
    - Veti sti de la A la Z acest framework, astfel ca veti avea capacitatea sa continuati sa il dezvoltati post curs acest proiect (pentru orice website doriti).
- Notiuni de baza despre API testing. (testarea backend - ce e in spate la un website).

* toti cei care sunt activi, implicati, isi fac temele, dedica timp pentru studiu individual si pun intrebari trainerului vor atinge aceste obiective.

# Obiective secundare

**Nu fac parte din curricula cursului LIVE dar va punem la dispozitie materiale extra ca sa aveti un avantaj la interviuri. Sfatul meu e sa va focusati pe ele doar dupa cursul live. Sa nu fiti overwhelmed de new info.**

- **Cunostinte ale bazelor de date relationale - mySQL (Curs baze de date)**
- **Cunostinte teoretice despre testarea manuala - acces la o platforma mobila**
- **Capacitatea de a construi un mic brand personal (Curs Portofoliu Wordpress). Trebuie sa ai:**
    - **Website propriu prin care angajatorul sa te cunoasca pe tine si munca ta**
    - **CV european in eng / sau canva.com**
    - **Profil LinkedIn**
    - **Github public (un loc in cloud unde se pune codul scris de tine)**
    - **Veti primi feedback daca ne trimiteti un email cu ele la hello@itfactory.ro**

# Obiective Intalnire 14

- Sa stim sa scriem unit tests (=> repo separat pt. angajator)
- Sa avem Postman instalat
- Sa cunoastem cele mai uzuale HTTP response codes
  - 1xx, 2xx, 3xx, 4xx, 5xx
- Sa intelegem ce e un API si sa stim cele mai folosite metode
  - Get, post, patch, put, delete
- Sa putem sa facem cateva request-uri manual in Postman

{iT} Factory

# Ce este un Unit Test?

**Testarea unitara reprezinta testarea la cel mai low level, practic este o functie care testeaza o alta functie.**
**Functia de test apeleaza functia testata si se asigura ca ea returneaza datele corecte, folosind un assert.**
**De obicei aceasta testare este facuta de devi dar si un qa o poate face, depinzand de raportul de forte dev-qa. In companiile cu mai multi angajati qa aceasta sarcina poate fi trecuta in responsabilitatea lor.**

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

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

{iT}Factory

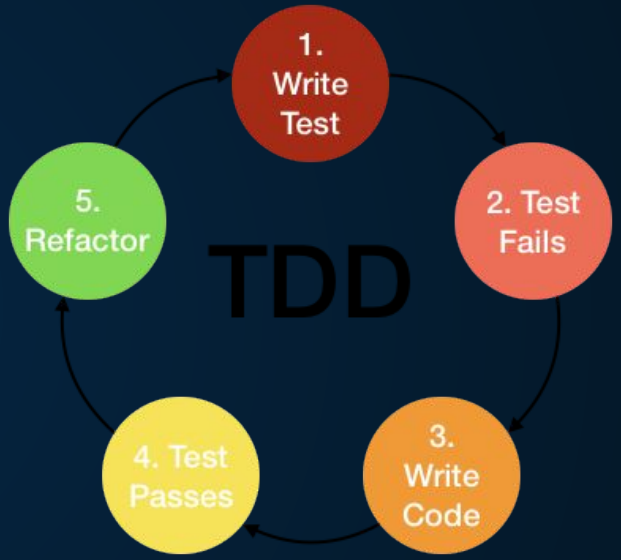
# What is TDD?

TDD = test driven development
Metodologie care plaseaza o importanta majora pe testare.

Se foloseste un test first approach.
Tot developmentul incepe prin scrierea de teste.

Exercitiu:

1. Instalati pytest (librarie care ne ajuta sa rulam teste)
   pip install pytest
1. Implementati un mini calculator folosind TDD
   a. Clasa MiniCalculator
   b. Atribute: 2 numere a si b
   c. Metode goale de +,-,*,/ folosind ‘pass’
   d. Implementam testele
   e. Executam testele (pica)
   f. Implementam logica
   g. Executam testele (trec)




Cum se executa 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 intre client si server.
Ne ajuta sa transferam date prin retea.

In imagine avem arhitectura standard pe 3 nivele (layers) a unei aplicatii.

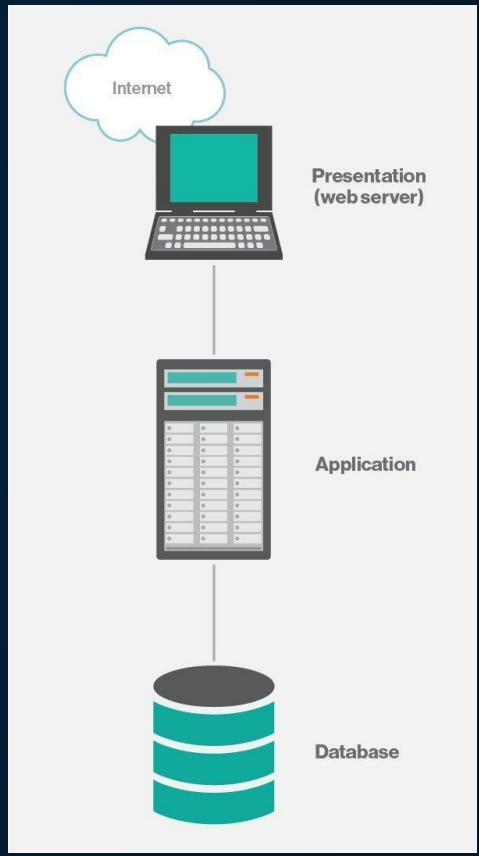
Client <--> HTTPS API requests <--> Server <--> db queries <--> Database

Testing:

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

App -> Unit Testing (stim)

API level (comunicarea dintre Client si Server) -> API testing (invatam)



# HTTP/HTTPS Protocols

{iT}Factory

Potocol = un standard sau o conventie asupra modului in care se desfasoara un anumit lucru.

Un Protocol permite comunicarea intre 2 device-uri pe una sau mai multe retele.

HTTP este un protocol de tip request-response, si ii ofera utilizatorilor o modalitate de a comunica cu resursele web.

HTTP (Hypertext Transfer Protocol) vs HTTPS (Hypertext Transfer Protocol Secure)

- Scopul principal este sa transfere date de la un server catre browserul tau.
- HTTPS, spre deosebire de HTTP foloseste un encrypted connexion ca sa comunice intre server si browser.
- HTTPS foloseste un certificat (SSL - secure sockets layer) care protejeaza datele transmise de furt.

# Raspunsuri HTTP

{iT}Factory

**1xx:** Informational
Request recieved/processing

**2xx** Success
Successfully Received, understood and accepted

**3xx** Redirect
Further action must be taken/redirect

**4xx**:
Client Error
Request doesn't have what it needs

**5xx:** Server ErrorServer failed to fullfill an apparent valid request

{iT}Factory

# Raspunsuri HTTP

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

Cele mai folosite:

- 200 OK - succes - de obicei cand se cer date de la server)
- 201 Created - success - cand se pun date in server
- 204 No Content - success - de obicei cand stergem ceva

- 301 - Moved to new URL

- 400 Bad request - ceva nu a mers bine, probabil valori invalide pt params
- 401 Unauthorized - nu suntem logati in app
- 403 Forbidden - suntem logati dar nu avem drepturi de edit de ex
- 404 Not Found - nu gaseste endpoint - probabil
- 408 Request Timeout - a durat prea mult pana sa ajunga la server requestul

- 500 Internal Server Error - req ajunge la server dar cel mai probabil este un bug
- 503 Service Unavailable - serverul e oprit pt. mentenanta de ex

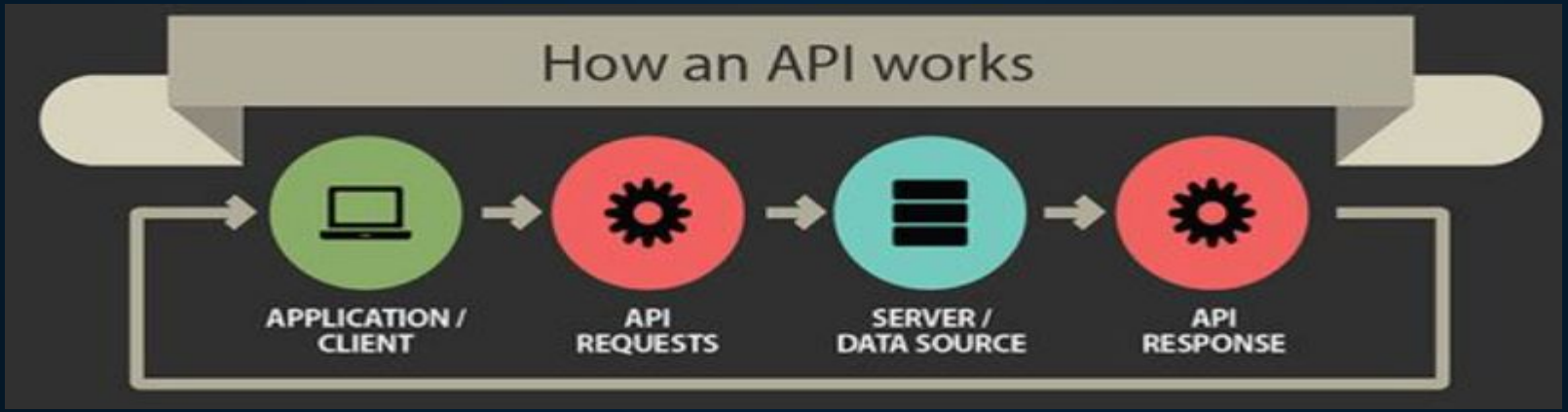
{iT}Factory

# Ce este API?

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

Niste comenzi bine documentate care ajuta programatorul sau interafata aplicatiei sa comunice cu logica din spate.
Clientul trimite un request catre un endpoint
Daca este pornit, serverul asculta non stop si este pregatit sa primeasca si sa proceseze aceste cerinte
Dupa ce serverul proceseaza informatiile primite/cerute ofera un raspuns clientului
Clientul interpreteaza raspunsul si il afiseaza intr-un mod user-friendly utilizatorului (avantaj: un sg API poate fi folosit de orice tip de client - ios, android, web etc si de oricate device-uri)

Schimbul de date se face sub forma unui JSON. Acesta are o structura asemanatoare unui dict din Python (cheie:valoare)



{iT}Factory

# REST API

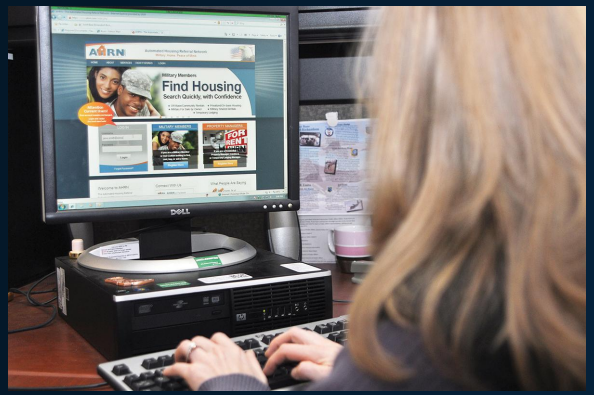
**REST** - Respresentational state transfer
- Este un stil arhitectural si un approach in scopul comunicarii care este foarte des folosit in dezvoltarea de web service-uri.
- Este complet bazat pe resurse

**API** - permite unei bucati de software sa comunice cu alta

**REST API** - face un request de la client catre server si aduce date folosind https protocol
- Are documentatie. Nu poti testa REST API fara documentatie
- Da mesaje de logging si mesaje de eroare exacte

**CUVINTE CHEIE:**
- Client - software care ruleaza pe device-ul utilizatorului si initiaza o comunicare
- Server - ofera un API ca o cale de acces la datele si feature-urile sale
- Resource - orice bucata de informatie pe care serverul poate sa o dea clientului: text file, video, audio

# How it works?




**Client**

Request

GET/POST/PUT/DELETE

Response

XML/JSON

**Server**

La nivelul serverului, REST API creeaza un obiect si trimite ca raspuns starea obiectului respectiv

JSON (Javascript object notation) - este formatul in care datele sunt trimise de la server catre client.

- Este derivat din java script pentru prezentarea simpla a datelor sub forma de cheie : valoare

# Request

Un request este format din:

**Endpoint** - contine un URI (Uniform Resource Identifier) care indica unde si cum sa gaseasca resursa pe internet.
Cel mai comun tip de URI este URL(Unique Resource Location), care serverste ca o adresa web completa.

**Headers -** stocheaza informatii relevante despre client si server.

- Date de autentificare (API key, numele/IP-ul computerului unde este instalat serverul si informatii despre formatul raspunsului)

**Body -** contine date aditionale pe care vrem sa le trimitem catre server

{iT}Factory

# Metode HTTP

Acronimul CRUD vine de la create, read, update, delete - actiunile principale cand lucram cu date in software dev

Cele mai folosite metode API sunt:

GET - cerem date de la server - in general 200

POST - trimitem date la server - in general 201

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

PUT - updatam date prin suprascrierea intregului obiect (ex: tot userul - nume, prenume, adresa) - in general 200 sau 201

DELETE - stergem date - in general 204

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

{iT}Factory

# Metode HTTP

| | | |
|---|---|---|
| **C -> CREATE** | **POST** | Creates a resource |
| **R -> READ** | **GET** | Fetches data from server |
| **U -> UPDATE** | **PUT** | Updates a resource |
| **D -> DELETE** | **DELETE** | Deletes a resource |

In REST terms we call these methods **verbs**

## SAFE METHODS

It doesn't bring any change

**GET**

## UNSAFE METHODS

It changes something

**POST**
**PUT**
**DELETE**

# Metode idempotente

Idempotent = este proprietatea anumitor operații care pot fi aplicate de mai multe ori, fără a schimba rezultatul dincolo de aplicația inițială.

Chiar daca le repetam de mai multe ori, nu afecteaza datele din server

**Metode idempotente**

**PUT**
**DELETE**

**POST**

POST creates a new resource whenever it’s used

# Intrebari interviu

- De la ce vine TDD? Ce este particular in aceasta metodologie de dezvoltare?
- Ce este un unit test?
- Cu ce cifra incep response codes HTTP de succes?
- Care e diferenta dintre response code 4xx si 5xx?
- De la ce vine API?
- Care sunt metodele principale HTTP? (5)
- Care e diferenta dintre patch si put?