

# Proiect Final

Constantin Davidica  
-28.03.2024-

# PARTEA I - TEORIE

**Variabilele** sunt locații de memorie unde putem înregistra, la un moment dat, o singură valoare de același tip.

**Constantele** sunt pur și simplu valori care sunt... constante, cu alte cuvinte, valori care nu se modifică. Constantele sunt opusele variabilelor, deoarece valoarea unei variabile se poate modifica pe durata execuției unui program.

**În Python există următoarele tipuri de date:**

- Numar intreg (int) - ex : `an_constructive = 1985`
- Booleene (adevarat/fals) - ex : `constructive_finalizata = True`
- Float (numar zecimal) - ex : `suprafata = 85.73`
- String (sir de caractere) - ex : `tip_constructive = 'Apartament'`

**Si următoarele structure de date:**

- Lista - ex : `list1 = ["andrei", 21, 54, "mere"]`
- Dictionar - ex : `dictionar1 = {  
    "marca" : "Seat"  
    "model" : "Leon"  
}`
- Set - ex : `culori = {'roz' , 'portocaliu', 'alb'}`
- Tuple - ex : `fructi = ( "mere", "pere", "banane")`

# PARTEA I - TEORIE

Functia **IF** permite rularea unui bloc de cod numai daca o conditie data este adevarata.  
In situatia in care este necesar a se verifica mai multe conditii se foloseste structura **IF-ELIF-ELSE**.  
Conditii **ELIF** putem avea oricate avem nevoie.

Exemplu structura IF-ELIF-ELSE:

```
If conditie1:  
    executa codul 1  
elif conditie 2:  
    executa codul 2  
elif conditie 3:  
    executa codul 3  
else:  
    executa codul 4
```

Pentru secventa din exemplu la rulare se parcurg urmasorii pasi:

- Se verifica conditie 1. Daca aceasta este adevarata se executa codul 1 si se iese din operatorul conditional
- Daca conditie 1 este falsa, se va trece la verificarea conditie 2. Daca aceasta este adevarata se executa codul 2 si se iese din operatorul conditional
- Daca si aceasta este falsa, se va trece la verificarea conditie 3. Daca aceasta este adevarata se executa codul 3 si se iese din operatorul conditional
- Daca toate conditiile de mai sus sunt false, se va executa codul aferent instructiunii **else** (codul 4)

# PARTEA I - TEORIE

**Funcțiile** sunt secvențe de cod care sunt definite pentru a fi apoi apelate în diverse secțiuni ale programelor.

Acestea asigură o structurare mult mai bună a aplicațiilor. În general, o funcție primește un set de argumente, efectuează un număr de operații și returnează o valoare.

Un **parametru** este o variabilă care este inițializată atunci când funcția este apelată.

**Clasa** este o machetă/ blueprint al unui obiect din lumea reală.

**Obiectul** reprezintă o instanțiere a unei clase.

Diferența între **clasa** și **obiect** este importantă. Dacă o clasă poate fi asimilată unui tip de date, un obiect poate fi echivalat cu o variabilă sau cu o valoare având un anumit tip de date.

Practic, o clasă este o "fabrică" de obiecte, care produce obiecte cu aceeași structură, având proprietăți și metode identice.

**Selectorul** reprezintă un șir de caractere prin intermediul căruia se poate identifica unul sau mai multe elemente dintr-o pagină web.

Acesta se utilizează pentru a interacționa cu pagina mai departe în procesul de testare automată.

Tipuri de selectori : **ID, Class, Name, XPATH, CSS Selector, Link Text, Partial Link Text**

# PARTEA I - TEORIE

**Dezvoltare bazată pe teste (TDD)** este o abordare de dezvoltare software în care sunt dezvoltate cazuri de testare pentru a specifica și valida ceea ce va face codul.

Avantajele TDD constau în faptul că se asigură o acoperire mai amplă a aplicației prin teste. Se poate crea un minim necesar de cod pentru implementarea unei funcționalități. TDD este implementat direct de echipa de dezvoltare, prin ceea ce se numește testare unitară.

**Testarea unitară** reprezintă modalitatea prin care fiecare componentă a codului este testată în mod izolat.

**Testare BDD (Behavior-driven development)** este o tehnică de dezvoltare software agilă și este ca o extensie a TDD, adică, Test Driven Development. În BDD, cazurile de testare sunt scrise într-un limbaj natural pe care chiar și non-programatorii îl pot citi.

Principalele avantaje ale BDD sunt:

- Toți membrii echipei au aceeași înțelegere a modului în care ar trebui să funcționeze aplicația dezvoltată.
- Actualizarea constantă a documentației proiectului.
- Demonstrarea faptului că implementarea funcționează corect.

**Gherkin** folosește o sintaxă simplă foarte asemănătoare cu limba engleză, cu posibilitatea de a utiliza variabile. Fiecare afirmație din Gherkin este precedată de cuvintele-cheie Given-When-Then.

# PARTEA I - TEORIE

**API** înseamnă „interfață de programare a aplicațiilor”. Un API este, în esență, un set de reguli care dictează modul în care două programe comunică între ele.

Un API facilitează comunicarea între programele software.

Metodele principale de HTTP sunt : **GET, POST, PUT, PATCH, DELETE**

**GET** - Prin intermediul acestei metode se poate solicita extragerea de informatii din baza de date.

**POST** - Prin intermediul acestei metode se poate solicita scrierea de informatii din baza de date.

**PUT** - Prin intermediul acestei metode se poate solicita modificarea de informatii din baza de date pin modificarea completa a unui obiect.

**PATCH** - Prin intermediul acestei metode se poate solicita modificarea de informatii din baza de date pin modificarea partiala a unui obiect.

**DELETE** - Prin intermediul acestei metode se poate solicita stergerea de informatii din baza de date prin rescrierea intregului obiect.



# PARTEA II — FRAMEWORK DE TESTARE AUTOMATA IN SELENIUM

Prin proiectul prezentat in urmatoarele slide-uri imi propun sa testez cateva functionalitati fundamentale ale site-ului [www.tiriacaauto.ro/](http://www.tiriacaauto.ro/)

Proiectul poate fi accesat la urmatorul link:

<https://github.com/Cosmin2890/Proiect-Final.git>

Printre functionalitatile testate se regasesc urmatoarele:

1. Verificarea mesajului de eroare atunci cand se introduce o adresa de email cu un format invalid.

```
@reccparola
Scenario: Check validation error message when email is invalid format
  When ai_uitat_parola: I fill in my email "my_email@test"
  When ai_uitat_parola: I click on the Trimite button
  Then ai_uitat_parola: I verify the invalid email validation message "Nu exista user cu adresa de email"
```

2. Verificarea mesajului de eroare atunci cand nu se introduce adresa de email.

```
@reccparola2
Scenario: Check validation error message when email era is empty
  When ai_uitat_parola: I make sure the email input is cleared
  When ai_uitat_parola: I click on the Trimite button
  Then ai_uitat_parola: I verify the invalid email validation message "Campul 'Email' trebuie completat."
```

3. Verificarea mesajului de eroare atunci user-ul si/sau parola sunt introduse gresit.

```
@login_test_1
Scenario: Enter wrong credentials and check the error
  When login: I fill in an email "email@mail.com"
  When login: I fill in a password "pass"
  When login: I click the login button
  Then login: It shown an error message "Datele de identificare nu pot fi confirmate."
```

# PARTEA II — FRAMEWORK DE TESTARE AUTOMATA IN SELENIUM

4. Verificarea daca dupa ce s-a introdus user si parola valide se ajunge pe pagina dorita.

```
@login_test_2
    Scenario: Enter good credentials and verify expected page
        When login: I fill in an email "myrbestphotos@gmail.com"
        When login: I fill in a password "testare1234"
        When login: I click the login button
        Then login: Verify expected pages "https://www.tiriacauto.ro/account"
```

5. Verificarea daca dupa ce s-a introdus user si parola valide se ajunge pe pagina dorita, apoi deconectare de la cont si verificare daca ajunge pe pagina dorita

```
@login_logout_test
    Scenario: Enter good credentials, verify expected page, logout and verify again expected page
        When login: I fill in an email "myrbestphotos@gmail.com"
        When login: I fill in a password "testare1234"
        When login: I click the login button
        Then login: Verify expected pages "https://www.tiriacauto.ro/account"
        Then logout: I click the logout button
        Then login: Verify expected pages "https://www.tiriacauto.ro/"
```

6. Din postura de client logat se va testa schimbarea a numelui din campul "Nume"

```
@account_test_1
    Scenario: I click Informatii personale and I change the lastname in Nume field
        When account: I click the informatii personale button
        Then account: I Change the lastname in "Popescu"
        Then account: I click the Trimite button
```



# PARTEA II — FRAMEWORK DE TESTARE AUTOMATA IN SELENIUM

7. Din postura de client logat se va testa schimbarea a prenumelui din campul "Prenume"

```
@account_test_2
```

```
Scenario: I click Informatii personale and I change the firstname in Prenume field
```

```
When account: I click the informatii personale button
```

```
Then account: I Change the firstname in "Adrian"
```

```
Then account: I click the Trimite button
```

8. Din postura de client logat se va verifica daca la rubrica "Masinile mele" se regaseste mesajul "Momentan nu ai nicio masina adaugata." in situatia in care nu s-a inregistrat nicio masina

```
@account_test_3
```

```
Scenario: I click Masinile mele and verify if I haven't car registered
```

```
When account: I click the Masinile mele button
```

```
Then account: I verify the message if I have the car registered "Momentan nu ai nicio masina adaugata."
```

# PARTEA II — FRAMEWORK DE TESTARE AUTOMATA IN SELENIUM

Pentru clonarea proiectului din github se parcurg urmatoorii pasi:

1. In folderul in care se doreste crearea clonei se da click dreapta—Open Git Bash here
2. Se introduce comanda “git clone <https://github.com/Cosmin2890/Proiect-Final.git>”
3. Din folderul in care s-a creat clona proiectului se sterge folderul “.git”

Tehnologia de testare este BDD si pentru a rula testele in Pycharm este necesar a fi instalate urmatoarele librarii prin introducerea urmatoarelor comenzi in Terminal:

- Selenium – “pip install selenium”
- Web driver manager – “pip install webdriver-manager”
- Libraria behave - “pip install behave”
- Rapoarte BDD - “pip install behave-html-formater”

# PARTEA II — FRAMEWORK DE TESTARE AUTOMATA IN SELENIUM

In cadrul proiectului se regasesc urmatoarele directoare:

1. Directorul features . Este directorul in care se regasete programarea in limbaj Gherkin.

```
@login_test
Scenario: Enter wrong credentials and check the error
  When login: I fill in an email "email@mail.com"
  When login: I fill in a password "pass"
  When login: I click the login button
  Then login: It shown an error message "Datele de identificare nu pot fi confirmate."
```

2. Directorul pages . In acest director se regasesc functiile necesare testarii fiecarei pagini.

```
def field_name_change(self, nume):
    self.driver.find_element(*self.NUME_FIELD).clear()
    self.driver.find_element(*self.NUME_FIELD).send_keys(nume)
```

3. Directorul steps . Aici codul face legatura intre limbajul Gherkin si functiile de testare.

```
@then('acount: I Change the firstname in "{prenume}"')
def step_impl(context, prenume):
    context.acount_page.field_firstname_change(prenume)
    time.sleep(2)
```

# PARTEA II — FRAMEWORK DE TESTARE AUTOMATA IN SELENIUM

Rularea testelor se face din terminal pentru fiecare feature prin introducerea comenzii:

```
behave -f html -o behave-report.html --tags="nume test"
```

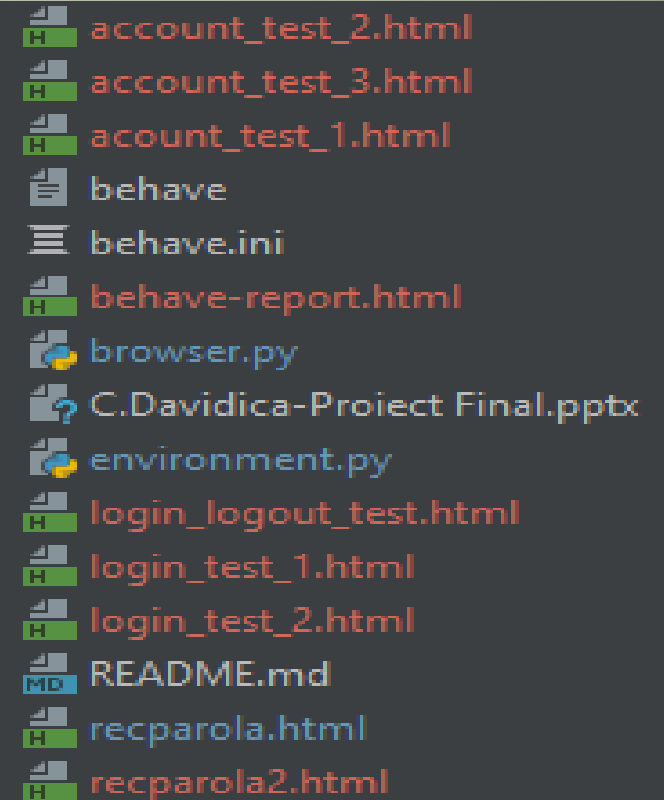
Urmatoarele rapoarte se regasesc si in cadrul proiectului in format .html("denumire\_test".html).

Orice rulare a unui test va genera un nou raport cu denumirea behave-report.html.

Din lista alaturata fisierele colorate cu albastru semnifica faptul ca la momentul realizarii capturii de ecran acesta au suferit modificari fata de varianta urcata pe github iar cele cu rosu sunt fisiere noi care necesita a fi urcate pe pltforma.

Pentru update-ul proiectului sa introduc urmatoarele comenzi:

1. In folderul in care se afla proiectul se da click dreapta—Open Git Bash here
  2. Se introduce comanda "git add ."
  3. Se introduce comanda "git commit -m "update 1."  
In locul textului"update 1" se poate scrie orice doresti
  4. Se introduce comanda "git push."
- In acest moment repo-ul esta updatat.



- account\_test\_2.html
- account\_test\_3.html
- account\_test\_1.html
- behave
- behave.ini
- behave-report.html
- browser.py
- C.Davidica-Proiect Final.pptx
- environment.py
- login\_logout\_test.html
- login\_test\_1.html
- login\_test\_2.html
- README.md
- recparola.html
- recparola2.html

# PARTEA II — FRAMEWORK DE TESTARE AUTOMATA IN SELENIUM

## Feature: Reseteaza parola

### Background:

@recparola

Scenario: Check validation error message when email is invalid format

Given login: I am a user on the login page (5.137s)

Given login: I accept all cookies (2.137s)

When login: I click on the Ai uitat parola link (0.890s)

When ai\_uitat\_parola: I fill in my email "my\_email@test" (10.178s)

When ai\_uitat\_parola: I click on the Trimite button (13.148s)

Then ai\_uitat\_parola: I verify the invalid email validation message "Nu exista user cu adresa de email specificata." (10.057s)

@recparola2

Scenario: Check validation error message when email era is empty

## Feature: Check the Login functionality

### Background:

@login\_test\_1

Scenario: Enter wrong credentials and check the error

@login\_test\_2

Scenario: Enter good credentials and verify expected page

@login\_logout\_test

Scenario: Enter good credentials, verify expected page, logout and verify again expected page

Given login: I am a user on the login page (4.979s)

Given login: I accept all cookies (2.154s)

When login: I fill in an email "myrbestphotos@gmail.com" (0.193s)

When login: I fill in a password "testare1234" (0.126s)

When login: I click the login button (3.931s)

Then login: Verify expected pages "https://www.tiriacauto.ro/account" (0.020s)

Then logout: I click the logout button (10.106s)

Then login: Verify expected pages "https://www.tiriacauto.ro/" (0.013s)

## Feature: Reseteaza parola

### Background:

@recparola

Scenario: Check validation error message when email is invalid format

@recparola2

Scenario: Check validation error message when email era is empty

Given login: I am a user on the login page (4.711s)

Given login: I accept all cookies (2.140s)

When login: I click on the Ai uitat parola link (0.771s)

When ai\_uitat\_parola: I make sure the email input is cleared (0.106s)

When ai\_uitat\_parola: I click on the Trimite button (13.065s)

Then ai\_uitat\_parola: I verify the invalid email validation message "Campul 'Email' trebuie completat." (10.068s)

## Feature: Check the Login functionality

### Background:

@login\_test\_1

Scenario: Enter wrong credentials and check the error

Given login: I am a user on the login page (4.950s)

Given login: I accept all cookies (2.156s)

When login: I fill in an email "email@mail.com" (0.173s)

When login: I fill in a password "pass" (0.098s)

When login: I click the login button (1.362s)

Then login: It shown an error message "Datele de identificare nu pot fi confirmate." (0.056s)

@login\_test\_2

Scenario: Enter good credentials and verify expected page

@login\_logout\_test

Scenario: Enter good credentials, verify expected page, logout and verify again expected page



# PARTEA II — FRAMEWORK DE TESTARE AUTOMATA IN SELENIUM

## Feature: Check the Login functionality

### Background:

#### @login\_test 1

Scenario: Enter wrong credentials and check the error

#### @login\_test 2

Scenario: Enter good credentials and verify expected page

Given login: I am a user on the login page (2.465s)

Given login: I accept all cookies (4.373s)

When login: I fill in an email "myrbestphotos@gmail.com" (0.205s)

When login: I fill in a password "testare1234" (0.129s)

When login: I click the login button (4.082s)

Then login: Verify expected pages "https://www.tiriacauto.ro/account" (0.013s)

#### @login\_logout\_test

Scenario: Enter good credentials, verify expected page, logout and verify again expected page

## Feature: Check the My account functionality

### Background:

#### @account\_test 1

Scenario: I click Informatii personale and I change the lastname in Nume field

#### @account\_test 2

Scenario: I click Informatii personale and I change the firstname in Prenume field

Given login: I am a user on the login page (4.844s)

Given login: I accept all cookies (2.115s)

Given account: I fill in an email "myrbestphotos@gmail.com" (2.242s)

Given account: I fill in a password "testare1234" (2.188s)

Given account: I click the login button (6.792s)

Given account: Verify expected pages "https://www.tiriacauto.ro/account" (0.018s)

When account: I click the informatii personale button (2.157s)

Then account: I Change the firstname in "Adrian" (2.231s)

Then account: I click the Trimite button (0.173s)

#### @account\_test 3

Scenario: I click Masinile mele and verify if I haven't car registered

## Feature: Check the My account functionality

### Background:

#### @account\_test 1

Scenario: I click Informatii personale and I change the lastname in Nume field

Given login: I am a user on the login page (2.445s)

Given login: I accept all cookies (4.599s)

Given account: I fill in an email "myrbestphotos@gmail.com" (2.613s)

Given account: I fill in a password "testare1234" (2.204s)

Given account: I click the login button (6.690s)

Given account: Verify expected pages "https://www.tiriacauto.ro/account" (0.028s)

When account: I click the informatii personale button (2.249s)

Then account: I Change the lastname in "Popescu" (2.219s)

Then account: I click the Trimite button (0.155s)

#### @account\_test 2

Scenario: I click Informatii personale and I change the firstname in Prenume field

#### @account\_test 3

Scenario: I click Masinile mele and verify if I haven't car registered

## Feature: Check the My account functionality

### Background:

#### @account\_test 1

Scenario: I click Informatii personale and I change the lastname in Nume field

#### @account\_test 2

Scenario: I click Informatii personale and I change the firstname in Prenume field

#### @account\_test 3

Scenario: I click Masinile mele and verify if I haven't car registered

Given login: I am a user on the login page (4.783s)

Given login: I accept all cookies (2.135s)

Given account: I fill in an email "myrbestphotos@gmail.com" (2.389s)

Given account: I fill in a password "testare1234" (2.134s)

Given account: I click the login button (6.778s)

Given account: Verify expected pages "https://www.tiriacauto.ro/account" (0.017s)

When account: I click the Masinile mele button (5.192s)

Then account: I verify the message if I have the car registered "Momentan nu ai nicio masina adaugata." (10.063s)



The background is a light green gradient. In the corners, there are white line art illustrations of circuit boards. The top-left and bottom-left corners feature more complex, branching circuit patterns. The top-right and bottom-right corners have simpler, more linear circuit patterns.

# Multumesc!