

Ministerul Educației al Republicii Moldova  
Universitatea de Stat din Moldova  
Facultatea de Matematică și Informatică  
Departamentul Informatică

**RAPORT**  
**PRACTICA DE PRODUCȚIE**

Elaborat de: Vlasî Anișoara, IA1901  
Coordonator: Siminel Victoria, lector universitar

CHIȘINĂU, 2022

## CUPRINS:

<b>INTRODUCERE</b> .....	3
<b>I ALLIED TESTING – DESCRIERE</b> .....	4
<b>II. CONSIDERAȚII TEORETICE</b> .....	5
<b>2.1. Limbaje De Programare</b> .....	6
<b>2.2. Framework-uri</b> .....	6
<b>2.3. Tool-uri</b> .....	8
<b>IV. SARCINI PRACTICE</b> .....	10
<b>4.1. Sarcina 1: Test Cases Și Program Java</b> .....	10
<b>4.2. Sarcina 2: Performance Testing În Jmeter</b> .....	24
<b>4.3. Sarcina 3: Script-uri în Powershell</b> .....	34
<b>CONCLUZII</b> .....	40
<b>BIBLIOGRAFIE</b> .....	41

## INTRODUCERE

Practica de producție are drept scop familiarizarea studenților cu viitoarea specialitate, însușind deprinderi pentru formarea profesională. În cadrul practicii, studentul este implicat în activitatea întreprinderii și dobândește experiența necesară, pentru a evolua în continuare. În plus, acesta își arată capacitățile și competențele dobândite pe parcursul anilor de studiu.

Deci, obiectivele principale ale practicii de producție sunt următoarele:

- Acumularea în practică a cunoștințelor acumulate în procesul de studiu;
- Argumentarea metodelor de proiectare a produselor program în unitatea economică;
- Adaptarea proiectelor software la condițiile dinamice ale unității economice;
- Analiza fezabilității aplicării sistemelor software în practică.

La finalizarea practicii de producție, studentul trebuie să demonstreze abilitățile cognitive în domeniul Informațional, precum cunoașterea și înțelegerea sarcinilor expuse de către coordonatorul practicii, aplicarea în practică a programelor/soft-urilor realizate, analizarea celor studiate și nu în ultimul rând sinteza și evaluarea celor efectuate pe parcurs.

Practica de producție are ca scop nu doar punerea în evidență a competențelor profesionale, ci și demonstrarea faptului că studentul se poate ușor integra într-o echipă, demonstrând și abilitățile de comunicare și interacțiune, fiind un punct foarte important în orice companie.

Pentru a realiza stagiul de practică, am depus CV-ul pentru a trece un internship în compania Allied Testing. În urma acestuia, am fost angajată oficial, demonstrând competențele dobândite pe parcurs. Deci, raportul de practică va conține descrierea entității Allied Testing, produsele soft și pachetele de programe însușite, problemele și sarcinile înaintate, precum și metodele de soluționare aplicate asupra acestora.

## I. ALLIED TESTING – DESCRIERE

” Allied Testing, a company founded in US in 2000, is a leading QA and testing specialist firm with the main focus on the capital markets, trading and finance industry. Over the years we have earned a reputation of premiere application quality management specialists. We work with CIOs, VPs of Development and Quality Directors to improve stability and resilience of their systems, develop or fine-tune QA and sourcing strategies, and take on non-trivial testing tasks, while simultaneously driving down costs of testing.

Allied offers its clients a full range of QA capabilities, from process audit and strategy consulting to implementation and ongoing service delivery, including both onsite and offshore/nearshore.

We have a strong experience acting as QA Center of Excellence in global, distributed, multi-vendor development environments. Our flexible engagement model allows our clients to manage peaks and valleys in their staffing requirements.

Deep domain expertise, global footprint and superb technical capabilities of our staff have enabled Allied to become a Quality Assurance partner of choice to such companies as Thomson Reuters, Raymond James, TD Ameritrade, Citibank, Fitch Solutions, Cetera, and others.”  
[<https://www.alliedtesting.com/about/what-we-do/>]

## II. CONSIDERAȚII TEORETICE

*Testarea software* reprezintă o investigație empirică realizată cu scopul de a oferi părților interesate informații referitoare la calitatea produsului sau serviciului supus testării, luând în considerație contextul operațional în care acesta din urmă va fi folosit. Testarea software pune la dispoziție o viziune obiectivă și independentă asupra produsului în dezvoltare, oferind astfel businessului posibilitatea de a înțelege și evalua riscurile asociate cu implementarea produsului soft. Tehnicile de testare includ, dar nu sunt limitate la, procesul de execuție a programului sau aplicației în scopul identificării defectelor/erorilor de software. Testarea software mai poate fi definită ca un proces de validare și verificare a faptului că un program/aplicație/produs software corespunde business cerințelor și cerințelor tehnice care au ghidat proiectarea și implementarea lui; și rulează și se comportă corespunzător așteptărilor.[1]

*Testarea automată* este o tehnică de testare a software-ului care funcționează folosind instrumente software speciale de testare automată pentru a executa o suită de cazuri de testare.

Automatizarea testelor este cea mai bună modalitate de a crește eficacitatea, acoperirea testului și viteza de execuție în testarea software-ului. Testarea automată a software-ului este importantă din următoarele motive[2]:

- Testarea manuală a tuturor fluxurilor de lucru, a tuturor domeniilor, a tuturor scenariilor negative necesită timp și bani
- Este dificil să testați manual site-urile multilingve
- Testare Automată în nu necesită intervenție umană. Puteți rula testul automat nesupravegheat (peste noapte)
- Testarea Automată mărește viteza de execuție a testului
- Automatizarea ajută la creșterea acoperirii testelor
- Testarea manuală poate deveni plictisitoare și, prin urmare, predispusă la erori.

Sunt mai multe tipuri de testare automatizată, precum:

- Unit Testing
- Smoke Testing
- Integration Testing
- Regression Testing
- API Testing
- Security Testing
- Performance Testing

- Acceptance Testing
- Web Services Testing
- UI Testing etc.

Principalele tipuri de testare utilizate pe parcursul practicii de producție sunt: UI Testing, Web Services Testing, Performance Testing, Regression Testing și Sanity Testing.

Pe parcursul practicii de producție am studiat mai multe tool-uri și programe utilizate pentru testarea software automatizată. Principalul limbaj de programare utilizat este JAVA, împreună cu framework-uri, precum Cucumber și TestNG. Alte tool-uri și limbaje de programare studiate și puse în practică sunt Git, JMeter, Maven, Powershell, Fiddler și Jira. În continuare voi face o mică descriere a acestora.

## 2.1. Limbaje De Programare

### 1. Java

*Java* este un limbaj de programare OOP sau orientat-obiect, dezvoltat de James Gosling la Sun Microsystems (acum filială Oracle), la începutul anilor '90 și lansat în 1995. Limbajul împrumută o mare parte din sintaxa C și C++, dar are un model al obiectelor mai simplu.[3]

Fie că vorbim de numărul de utilizatori, de locurile de muncă disponibile sau de numărul de programatori, Java este mereu pe primele locuri.[3]

### 2. Powershell

*PowerShell* este un program de automatizare a sarcinilor și de gestionare a configurației de la Microsoft, constând dintr-un shell de linie de comandă și limbajul de scriptare asociat.

În PowerShell, sarcinile administrative sunt în general efectuate prin cmdlet-uri (pronunțate *command-lets*), care sunt clase specializate .NET care implementează o anumită operație. Acestea funcționează prin accesarea datelor din diferite depozite de date, cum ar fi sistemul de fișiere sau Registrul Windows, care sunt puse la dispoziția PowerShell prin intermediul furnizorilor. Dezvoltatorii terți pot adăuga cmdlet-uri și furnizori la PowerShell. Cmdleturile pot fi folosite de scripturi, care la rândul lor pot fi ambalate în module. Cmdleturile funcționează în tandem cu .NET API.[4]

## 2.2. Framework-uri

### 1. Cucumber

*Cucumber* este un instrument software care sprijină dezvoltarea bazată pe comportament. Esențial pentru abordarea Cucumber este analizatorul său obișnuit, numit Gherkin, ce permite ca

comportamentele software așteptate să fie specificate într-un limbaj logic pe care clienții îl pot înțelege. Este adesea folosit pentru testarea altor software-uri. Rulează teste de acceptare automate scrise într-un stil de dezvoltare determinată de comportament.[5]

*Gherkin* este limbajul pe care Cucumber îl folosește pentru a defini cazurile de testare. Este conceput pentru a fi non-tehnic și ușor de citit și descrie în mod colectiv cazuri de utilizare referitoare la un sistem software. Scopul din spatele sintaxei lui *Gherkin* este de a promova practici de dezvoltare bazate pe comportament într-o întreagă echipă de dezvoltare, inclusiv analiști de afaceri și manageri. Acesta urmărește să impună cerințe ferme, fără ambiguitate, începând din fazele inițiale ale definirii cerințelor de către managementul afacerii și în alte etape ale ciclului de viață al dezvoltării.[5]

Pe lângă furnizarea unui script pentru testarea automată, sintaxa limbajului natural a lui *Gherkin* este concepută pentru a oferi o documentare simplă a codului testat.[5]

## 2. TestNG

*TestNG* este un cadru de testare pentru limbajul de programare Java creat de Cédric Beust și inspirat de JUnit și NUnit. Scopul de proiectare al *TestNG* este să acopere o gamă mai largă de categorii de teste: unitar, funcțional, end-to-end, integrare etc., cu funcționalități mai puternice și mai ușor de utilizat.[6]

Principala adnotare în *TestNG* este `@Test`, care poate fi utilizată pentru personalizarea testelor în diferite căi. Cele mai utile setări care pot fi utilizate cu adnotarea `@Test` sunt:

- `dataProvider` – specifică numele furnizorului de date utilizat
- `dataProviderClass` – specifică numele clasei unde este localizat `dataProvider`-ul
- `expectedExceptions` – specifică lista de excepții care pot fi aruncate de codul testului
- `invocationCount` – specifică de câte ori va fi executat testul
- `successPercentage` – specifică procentajul testelor ce trebuie să fie executate cu succes. Dacă numărul de teste executate cu succes este mai mare decât acest număr, atunci se consideră testul executat cu succes, chiar dacă unele teste au statusul failure.
- `testName` – util pentru a da un nume mai semnificativ testului
- `timeOut` – timpul maxim în care trebuie să fie executat testul, altfel, acesta va fi considerat failed.

O altă adnotare importantă este `@DataProvider`, asociat unei metode, care oferă o serie de valori reale variate pentru metodele de testare dependente.

Furnizorul de date va returna un obiect `Object[][]` – matrice de obiecte în care dimensiunea primei dimensiuni este de câte ori va fi invocată metoda de testare și a doua dimensiune conține o matrice de obiecte care trebuie să fie compatibile cu tipurile de parametri ai testului[6].

### 2.3. Tool-uri

#### 1. Git

*Git* este un sistem de control al versiunilor conceput pentru a gestiona munca în echipă efectuată pe un proiect. *Git* îi ajută pe colaboratori să urmărească modificările în fișiere sau proiecte și să accelereze procesul general.[7]

*Git* are un depozit la distanță care este stocat pe un server și un depozit local care este stocat în computerul fiecărui dezvoltator. Acest lucru înseamnă că codul nu este stocat doar pe un server central, dar copia completă a codului este prezentă și în toate computerele dezvoltatorilor. Deoarece fiecare nod are o copie locală, aproape toate operațiunile de pe *Git* sunt locale (excepțiile fiind comanda *Pull* și *Push*). Ceea ce înseamnă că nu trebuie să fii conectat tot timpul la depozitul de la distanță pentru a-ți face treaba.[8]

#### 2. Maven

*Maven* este un sistem de build și administrare a proiectelor, scris în Java. Face parte din proiectele găzduite de Apache Software Foundation. Funcționalitățile sale principale sunt descrierea procesului de build al software-ului și descrierea dependențelor acestuia. Proiectele sunt descrise printr-unul sau mai multe fișier XML, denumite POM-uri (Project Object Model), dar au o structură implicită, ceea ce încurajează structurarea similară a proiectelor. POM-ul principal conține informații despre module, precum și despre dependențele proiectului (alte proiecte). Ordinea operațiunilor de build este definită prin declararea unor plugin-uri folosite, din cadrul cărora unele goal-uri sunt plasate și configurate în diferitele faze predefinite din ciclul de viață al unui build. *Maven* descarcă dinamic bibliotecile Java și plugin-urile, din unul sau mai multe repository-uri.[9]

#### 3. JMeter

Aplicația Apache *JMeter*<sup>TM</sup> este un software open source, o aplicație Java 100% pură, concepută pentru a testa comportamentul funcțional și pentru a măsura performanța. A fost conceput inițial pentru testarea aplicațiilor web, dar de atunci s-a extins la alte funcții de testare.[10]

Apache *JMeter* poate fi folosit pentru a testa performanța atât pe resurse statice, cât și pe cele dinamice. Poate fi folosit pentru a simula o sarcină grea pe un server, un grup de servere, o

rețea sau un obiect pentru a-i testa puterea sau pentru a analiza performanța generală la diferite tipuri de încărcare.[10]

#### 4. Fiddler

*Fiddler* este un instrument de server proxy de depanare folosit pentru a înregistra, inspecta și modifica traficul HTTP și HTTPS între un computer și un server web sau servere.

Fiddler captează traficul HTTP și HTTPS și îl înregistrează pentru ca utilizatorul să îl examineze. Înregistrarea se realizează prin implementarea interceptării „man-in-the-middle” folosind certificate autosemnate. Fiddler poate fi folosit pentru a edita sesiuni de rețea prin setarea punctelor de întrerupere pentru a întrerupe procesarea cererilor și permițând modificarea cererii și/sau a răspunsului. [11]

#### 5. Jira

*Jira* este un produs proprietar de urmărire a problemelor dezvoltat de Atlassian, care permite urmărirea erorilor și gestionarea agilă a proiectelor.[12]

Este utilizat pe scară largă ca instrument de urmărire a problemelor pentru toate tipurile de testare.[13]

#### IV. SARCINI PRACTICE

Partea practică constă din mai multe task-uri primite. În continuare voi descrie câteva dintre ele, dar voi face câteva schimbări în denumirea site-urilor și aplicațiilor testate, acestea fiind confidențiale.

##### 4.1. Sarcina 1: Test Cases Și Program Java

Este dat un site web XXX, care constă din mai multe pagini, printre care și pagina de înregistrare. Pe baza cerințelor din documentul primit de la QA Team Lead, este necesar de scris test case-uri care ar acoperi aceste cerințe.

REG-001:

User should be able to register a new user. Username must be at least 4 characters long and contain at least one upper case character, one lower case character and one numeric character. Password must be at least 6 characters long and contain at least one upper case character, one lower case character and one numeric character.

If username or password is incorrect, after clicking on Register button a message "Registration failed! Check your username or password." should be displayed. User should not be able to register two users with identical user names.

Rezolvare:

**Tabelul 4.1 Test Case „User Registration with valid data”**

Test Name:	User registration with valid data.	
Test Description:.	Verify of the user is able to register with valid data	
Step Name	Step Description	Step Expected Results
1.	Open the XXX web site	Registration page opened
2.	Enter a valid data in "Username" text field, containing 4 characters with upper case, lower case and a number (e.g. Aq8%)	"Username" text field is populated
3.	Enter a valid data in "Password" text field, containing 6 characters with upper case, lower case and a number (e.g. sT&2_w)	"Password" text field is populated
4.	Click on [Register] button	User is registered and the "CRA Service" main page is displayed
5.	Click on "Log out" link at the upper right corner of the page	The registration page is displayed
6.	Enter a valid data in "Username" text field, containing more than 100 characters with upper case, lower case and a number	"Username" text field is populated
7.	Enter a valid data in "Password" text field,	"Password" text field is populated

**Tabelul 4.2 Test Case „User registration with invalid username.”**

Test Name:	User registration with invalid username.	
Test Description:	Check if the “Registration failed! Check your username or password.” message is displayed after entering invalid username	
Step Name	Step Description	Step Expected Results
1.	Open the XXX web site	Registration page opened
2.	Enter an invalid value (less than 4 characters) in the “Username” text field, like “An5”. Enter a valid value in “Password” text field and click on [Register] button	“Registration failed! Check your username or password.” is displayed
3.	Enter an invalid value (without uppercase) in the “Username” text field, like “asn5”. Enter a valid value in “Password” text field and click on [Register] button	“Registration failed! Check your username or password.” is displayed
4.	Enter an invalid value (without lowercase) in the “Username” text field, like “MAN5”. Enter a valid value in “Password” text field and click on [Register] button	“Registration failed! Check your username or password.” is displayed
5.	Enter an invalid value (without digits) in the “Username” text field, like “MonN”. Enter a valid value in “Password” text field and click on [Register] button	“Registration failed! Check your username or password.” is displayed
6.	Leave the “Username” text field empty. Enter a valid value in “Password” text field and click on [Register] button	“Registration failed! Check your username or password.” is displayed

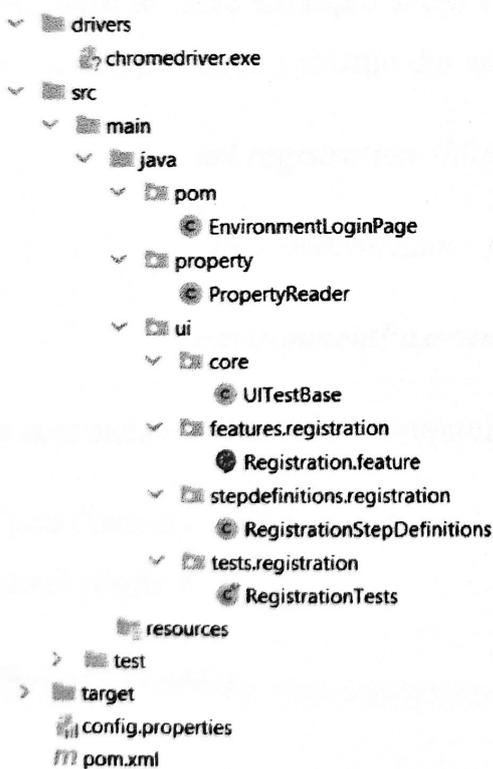
**Tabelul 4.3 Test Case „User registration with invalid password”**

Test Name:	User registration with invalid password.	
Test Description:	Check if the “Registration failed! Check your username or password.” message is displayed after entering invalid password	
Step Name	Step Description	Step Expected Results
1.	Open the XXX web site	Registration page opened
2.	Enter an invalid value (less than 6 characters) in the “Password” text field, like “AnhP5”. Enter a valid value in “Username” text field and click on [Register] button	“Registration failed! Check your username or password.” is displayed
3.	Enter an invalid value (without uppercase) in the “Password” text field, like “asnrr5”. Enter a valid value in “Username” text field and click on [Register] button	“Registration failed! Check your username or password.” is displayed
4.	Enter an invalid value (without lowercase) in the “Password” text field, like “MAN555”. Enter a valid value in “Username” text field and click on [Register] button	“Registration failed! Check your username or password.” is displayed
5.	Enter an invalid value (without digits) in the “Password” text field, like “MonNbv”.	“Registration failed! Check your username or password.” is displayed

**Tabelul 4.4 Test Case „User registration with existing username”**

Test Name:	User registration with existing username.	
Test Description:	Check if the “Registration failed! Check your username or password.” message is displayed after entering an existing username.	
Step Name	Step Description	Step Expected Results
1.	Open the XXX web site	Registration page opened
2.	Enter an existing username in "Username" text field, e.g. username used in the "User registration with valid data positive" test case, that is "Aq8%"	"Username" text field is populated
3.	Enter a valid data in "Password" text field, containing 6 characters with upper case, lower case and a number (e.g. sTt28w)	"Password" text field is populated
4.	Click on [Register] button	“Registration failed! Check your username or password.” is displayed

Pe baza test case-urilor create, se creează un proiect Java, utilizând framework-ul Cucumber, care va acoperi aceste test case-uri. Deci vom avea următoarea structură a proiectului:



**Figura 4.1 Structura proiectului**

În fișierul pom.xml sunt inserate toate dependențele din Maven. Iată câteva dintre ele:

### 4.3. Sarcina 3: Script-uri în Powershell

Sarcina constă în pregătirea unor fișiere pentru testarea performanței de generare a unor rapoarte. Pentru aceasta am utilizat Powershell. În continuare voi descrie pașii de pregătire a fișierelor, și voi adăuga codul respectiv. Denumirile și unele caracteristici vor fi schimbate din considerente de confidențialitate.

Avem două locații, A și B. Trebuie de scris un script care va copia fișiere din locația A în locația B, și în același timp, în locația B trebuie să fie create alte 2 foldere, în dependență de denumirea fișierelor, și să se sorteze aceste fișiere în folderele date.

Avem următorul script:

```
function Copy-Files{  
  
    param([string]$sourcepath, [string]$destpath)  
  
    Get-ChildItem ($sourcepath + "\*")|ForEach-Object{  
  
        $filename = $_.Name  
  
        switch -Wildcard ($filename){  
  
            "*NAME1*" {$foldername = "NAME1"}  
  
            "*NAME2*" {$foldername = "NAME2"}  
  
            default {$foldername = NULL}  
  
        }  
  
        if($foldername){  
  
            if(!(Test-Path -Path ($destpath + '\' + $foldername))){  
  
                New-Item -Path ($destpath + '\') -Name $foldername -ItemType "directory"  
  
                Copy-Item -Path ($sourcepath + '\' + $filename) -Destination ($destpath + '\' +  
$foldername)  
  
            }  
  
        }  
  
    }  
  
}
```

În rezultat avem fișierele redenumite pentru ambele foldere, NAME1 și NAME2:

This PC > New Volume (D:) > exemple > B > NAME1

Name	Date modified
FILE1	3/31/2022 6:58 PM
FILE2	3/31/2022 6:58 PM

**Figura 4.38** Conținutul Fișierului NAME1 după Modificări

This PC > New Volume (D:) > exemple > B > NAME2

Name	Date modified
FILE1	3/31/2022 6:58 PM
FILE2	3/31/2022 6:58 PM
FILE3	3/31/2022 6:59 PM

**Figura 4.39** Conținutul Fișierului NAME2 după Modificări

Practica de producție reprezintă o oportunitate de a-ți valorifica cunoștințele și abilitățile acumulate pe parcursul anilor de studiu. În cadrul acesteia, studentul este implicat în diverse activități de cercetare și de soluționare a unor probleme reale din cadrul unei entități economice.

Domeniul IT oferă un spectru larg de activități și profesii ce pot fi însușite pe parcursul practicii de producție. Domeniul în care am activat pe parcursul practicii este legat de testarea software, atât manuală cât și automatizată. Consider că acesta este un început foarte bun pentru a-ți dezvolta capacitățile de observare, gândire critică, curiozitate, disciplină, acuratețe, flexibilitate, analiză, time management, și nu în ultimul rând de comunicare și interacțiune atât cu membrii de echipă cât și cu clienții.

Pe parcursul practicii am avut ocazia să descopăr multe lucruri interesante și să învăț de la cei mai buni specialiști din acest domeniu. Am studiat diferite instrumente de testare și m-am aprofundat în limbaje de programare precum Java, care am început să-l studiez la universitate. Datorită sarcinilor primite și a problemelor întâlnite pe parcurs, am reușit să-mi formez un bagaj de cunoștință și deprinderi care cu siguranță mă vor ajuta în dezvoltarea profesională. Acum, când navighez prin diverse aplicații și pagini web, observ fiecare cusur ce nu ar trebui să persiste. De asemenea, înțeleg mult mai multe lucruri, care înainte îmi păreau străine. Anume aceste competențe îmi vor fi de folos atunci când voi fi dispusă să-mi creez propriile aplicații, astfel, voi ști exact care sunt punctele slabe în procesul dezvoltării acestor aplicații. Un alt moment foarte important sunt oamenii care m-au ghidat și cu care am interacționat pe parcurs. Mereu au fost disponibili și deschiși spre colaborare și atunci când aveam nevoie primeam cu drag ajutor.

Toate cele menționate mai sus mi-au dat curaj, dorință și motivație de a continua să studiez aplicații, instrumente și limbaje de programare noi, pentru a excela în domeniul IT și pentru a obține rezultate cât mai frumoase.

## BIBLIOGRAFIE

1. [https://ro.wikipedia.org/wiki/Testare\\_software](https://ro.wikipedia.org/wiki/Testare_software)
2. <https://www.guru99.com/automation-testing.html>
3. <https://codecool.com/ro/blog/ghid-java-incepatori/>
4. <https://en.wikipedia.org/wiki/PowerShell>
5. [https://en.wikipedia.org/wiki/Cucumber\\_\(software\)](https://en.wikipedia.org/wiki/Cucumber_(software))
6. <https://en.wikipedia.org/wiki/TestNG>
7. <https://www.toolsqa.com/git/git-tutorial/>
8. <https://www.toolsqa.com/git/what-is-git/>
9. [https://ro.wikipedia.org/wiki/Apache\\_Maven](https://ro.wikipedia.org/wiki/Apache_Maven)
10. [https://jmeter.apache.org/usermanual/test\\_plan.html](https://jmeter.apache.org/usermanual/test_plan.html)
11. [https://en.wikipedia.org/wiki/Fiddler\\_\(software\)](https://en.wikipedia.org/wiki/Fiddler_(software))
12. [https://en.wikipedia.org/wiki/Jira\\_\(software\)](https://en.wikipedia.org/wiki/Jira_(software))
13. <https://www.tutorialspoint.com/jira/index.htm>