



- PROIECT FINAL -

HARABULA ANDREEA

10 APRILIE 2024

PARTEA I – NOTIUNI TEORETICE:

Cerintele de business reprezinta o descriere detaliata a necesitatilor si asteptarilor unei organizatii sau unei afaceri in ceea ce priveste un produs. Aceste cerinte sunt esentiale pentru a asigura succesul si satisfactia clientilor sau a utilizatorilor finali. Scopul acestora este de a defini clar obiectivele proiectului si de a ii oferi o directie corecta. Cerintele de business pot proveni de la diverse surse precum clienti, manager de produs, parteneri de afaceri etc.

TEST CONDITION VS TEST CASE

Test Condition reprezinta o conditie sau un criteriu specific care trebuie verificat in timpul procesului de testare pentru a valida functionalitatea sau comportamentul unui sistem.

Test case reprezinta un scenariu sau un set de pasi specifici care trebuie urmati pentru a efectua un test sub o anumita conditie de testare.

ETAPELE PROCESULUI DE TESTARE:

- 1. Planificare a testelor (Test Planning):** Definirea obiectivelor și a scopului testelor, identificarea resurselor necesare pentru testare (oameni, hardware, software), stabilirea strategiei de testare și a planurilor de testare, crearea programului de testare.
- 2. Analiza cerințelor (Requirement Analysis):** înțelegerea și revizuirea cerințelor software-ului, identificarea și clarificarea cerințelor testabile, dezvoltarea matricei de urmărire a testelor pentru a asigura acoperirea cerințelor.
- 3. Proiectarea testelor (Test Design):** crearea specificațiilor de testare, inclusiv scenarii de testare și cazuri de testare, dezvoltarea datelor de testare, crearea planurilor de testare și a procedurilor de testare.
- 4. Implementare a testelor (Test Implementation):** configurarea mediului de testare, implementarea cazurilor de testare, dezvoltarea și pregătirea datelor de testare, crearea procedurilor de testare și automatizarea testelor, dacă este necesar.
- 5. Execuția testelor (Test Execution):** rularea cazurilor de testare conform planului de testare, monitorizarea și înregistrarea rezultatelor testelor, identificarea și raportarea defecțiunilor (bugs).
- 6. Analiza rezultatelor și raportare (Test Analysis and Reporting):** analizarea rezultatelor testelor pentru a evalua performanța și calitatea software-ului, raportarea rezultatelor și evidențierea problemelor identificate, documentarea și raportarea defecțiunilor într-un mod structurat.
- 7. Urmărire și gestionare (Test Monitoring and Management):** monitorizarea progresului testelor în raport cu planurile de testare, actualizarea planurilor de testare în funcție de descoperirile din timpul testelor, gestionarea defecțiunilor și asigurarea rezolvării acestora.
- 8. Încheierea testelor (Test Closure):** evaluarea testelor în raport cu obiectivele stabilite, documentarea experiențelor și învățămintele din procesul de testare, finalizarea documentației de testare.

RETESTING VS REGRESSION TESTING:

- **Retesting** – se concentreaza pe repetarea testelor ce au identificat initial defectele si are loc dupa ce au fost remediate defectele gasite initial;
- **Regression Testing** – are ca scop asigurarea ca modificarile aduse asupra produsului (cum ar fi corectii de bug-uri, imbunatatiri, noi functionalitati) nu au afectat functionalitatii existente. Acest tip de testare este efectuat de obicei dupa ce s-au adus schimbari majore sau dupa ce au fost introduce noi functionalitati.

FUNCTIONAL TESTING VS NON- FUNCTIONAL TESTING:

- **Functional testing** – asa cum ii spune si denumirea, acest tip de testare vizeaza functionalitatile individuale ale unui sistem pentru a se asigura ca acesta indeplineste cerintele specifice.
- **Non-functional testing** – are ca scop verificarea atributelor ce descriu cat de bine isi indeplineste sistemul functiile;

BLACK-BOX TESTING VS WHITE- BOX TESTING:

- **Blackbox testing** – se concentreaza pe functionalitatea sistemului si nu pe structura interna a codului sursa iar testerul nu are cunostinte in prealabil despre structura interna a sistemului;
- **Whitebox testing** –se concentreaza pe structura interna a codului sursa si pe logica de implementare a sistemului. In acest caz persoana ce testeaza cunoaste detalii despre codul sursa si arhitectura sistemului;

GRUPAREA TEHNICILOR DE TESTARE:

Whitebox:

- **Statement Coverage;**
- **Decision Coverage;**

Blackbox:

- **Equivalence Partitioning;**
- **Boundary Value Analysis;**
- **State Transitioning;**
- **Decisional Table;**

Experience testing:

- **Ad-hoc Testing;**
- **Exploratory Testing;**
- **Error guessing;**

VERIFICATION VS VALIDATION

- Verification – este o metoda realizată după finalizarea dezvoltării, confirmând că produsul îndeplinește nevoile și așteptările utilizatorilor. Aceasta are ca scop confirmarea corectitudinii implementării în raport cu specificațiile tehnice.
- Validation - este realizată după finalizarea dezvoltării, confirmând că produsul îndeplinește nevoile și așteptările utilizatorilor.

POSITIVE TESTING VS NEGATIVE TESTING:

- **Positive testing** - este o tehnică de testare în care se validează că sistemul funcționează conform așteptărilor în condiții normale sau conforme. Se testează cazurile în care input-urile sau acțiunile ar trebui să producă rezultate valide și corecte conform specificațiilor.
- ✓ **Exemplu:** luam situația în care avem o aplicație de autentificare, un test pozitiv ar implica introducerea unor date valide (un nume de utilizator și o parolă corectă) și verificarea dacă utilizatorul este autentificat cu succes.
- **Negative testing:** - este o tehnică de testare care se concentrează pe evaluarea comportamentului sistemului în condiții anormale sau non-conforme. Scopul este să detecteze și să identifice erorile și vulnerabilitățile care ar putea apărea în situații neașteptate.
- ✓ **Exemplu:** Pentru aceeași aplicație de autentificare, un test negativ ar implica introducerea unor date incorecte (un nume de utilizator sau o parolă greșită) și verificarea modului în care sistemul gestionează aceste erori, cum ar fi afișarea unui mesaj de eroare sau blocarea temporară a contului.

NIVELURI DE TESTARE:

Testarea unitară (Unit Testing):

are ca scop verificarea corectitudinii funcțiilor individuale, metodelor sau claselor la nivel de cod.

Testarea Integrată (Integration Testing):

are ca scop verificarea interacțiunii dintre modulele sau componente ale sistemului.

Testarea Sistemului (System Testing):

are ca scop verificarea întregului sistem în ansamblul său, asigurându-se că funcționalitățile integrate funcționează corespunzător.

Testarea de acceptanta (User Acceptance Testing - UAT):

are ca scop verificarea dacă sistemul îndeplinește cerințele și așteptările utilizatorilor.

PARTEA II – VARIANTA 3: SQL

Pentru partea practica am ales cea de a treia varianta si anume crearea unei baze de date. Tema aleasa a fost “Contracte firme si facturi” si am inceput prin a crea o baza de date numita “Contracte_colaboare” utilizand in aplicatia My SQL instructiunea “create database”.

1. Instrucțiuni DDL:

✓ **CREATE** – cu ajutorul acestei instructiuni DDL am creat cele 3 tabele.

```
1 • create database contracte_colaborare;  
2 • use contracte_colaborare;  
3 • create table firme_contractate(  
4   nr_crt int primary key auto_increment not null,  
5   nume_societate Varchar (25) not null,  
6   oras Varchar (25) not null,  
7   anul_infiintarii int  
8 );
```

nr_crt	nume_societate	oras	anul_infiintarii
3	Sagrod SRL	Darabani	1999
4	Mega Image	Vaslui	2022
5	Premium Dental Center	Iasi	2023
6	Kosarom	Pascani	2019
*	NULL	NULL	NULL

```
16 • create table contracte(  
17   nr_crt int primary key auto_increment not null,  
18   tip_contract Varchar (25) not null,  
19   persoana_contact Varchar (25) not null,  
20   valoare_contract int  
21 );
```

nume_firma	cod_fiscal	valoare_contract
Mega Image	707317	2000
La doi pasi	800800	3000
Sagrod SRL	992034	7000

nr_crt	tip_contract	persoana_contact	valoare_contract
2	sponsorizare	Ion Alexandru	2000
3	publicitate	Maria Andrei	5000
4	vanzare	Robu Cristian	1000
5	determinat	Ion Ion	200
6	nedeterminat	Mihai Cristea	200
*	NULL	NULL	NULL

✓ **ALTER** – folosind aceasta instructiune am modificat tabelul firme_contractate si am adaugat coloana data_facturare.

```
ALTER TABLE firme_contractate ADD COLUMN data_facturare int;
```

nr_crt	nume_societate	oras	anul_infiintarii	data_facturare
1	Mega Image	Vaslui	2022	NULL
2	La doi pasi	Iasi	2005	NULL
3	Sagrod SRL	Darabani	1999	NULL
4	Mega Image	Vaslui	2022	NULL
5	Premium Dental Center	Iasi	2023	NULL
6	Kosarom	Pascani	2019	NULL
*	NULL	NULL	NULL	NULL

- ✓ **TRUNCATE** – utilizand aceasta instructiune am sters inregistrarile din tabel, pastrand insa structura acestuia:

```
41 • TRUNCATE TABLE taxe_firme;
```

```
32 • create table taxe_firme(  
33     nume_firma Varchar (25) not null,  
34     cod_fiscal int,  
35     valoare_contract int  
36 );  
37 • select * from taxe_firme;  
38 • insert into taxe_firme (nume_firma, cod_fiscal, valoare_contract) values ('Mega Image', '20231')
```

result Grid | Filter Rows: | Export: | Wrap Cell Content: |

nume_firma	cod_fiscal	valoare_contract
------------	------------	------------------

- ✓ **DROP** – cu ajutorul acestei comenzi am sters tabel:

```
42 • DROP TABLE taxe_firme;
```

```
52 20:32:14 select * from taxe_firme LIMIT 0, 1000
```

Error Code: 1146. Table 'contracte_colaborare.taxa_firme' doesn't exist

2. Instrucțiuni de DML:

- ✓ **INSERT** – folosind aceasta instructiune am inserat informatiile necesare in table.

```
11 • insert into firme_contractate (nume_societate, oras, anul_infiintarii) values ('Sagrod SRL', 'Darabani', '1999');  
12 • insert into firme_contractate (nume_societate, oras, anul_infiintarii) values ('La doi pasi', 'Iasi', '2005');  
13 • insert into firme_contractate (nume_societate, oras, anul_infiintarii) values ('Mega Image', 'Vaslui', '2022');  
14 • insert into firme_contractate (nume_societate, oras, anul_infiintarii) values ('Premium Dental Center', 'Iasi', '2023');  
15 • insert into firme_contractate (nume_societate, oras, anul_infiintarii) values ('Kosarom', 'Pascani', '2019');
```

- ✓ **DELETE** – comanda Delete a fost folosita pentru a sterge inregistrarile dintr-o tabela:

```
• DELETE FROM contracte  
WHERE nr_crt = '1';
```

	nr_crt	tip_contract	persoana_contact	valoare_contract
▶	2	sponsorizare	Ion Alexandru	2000

- ✓ **UPDATE** – cu ajutorul acestei comenzi am modificat informatiile din tabel

```
• UPDATE contracte  
SET valoare_contract = 1000;  
• use contracte colaborare;
```

	nr_crt	tip_contract	persoana_contact	valoare_contract
▶	2	sponsorizare	Ion Alexandru	1000
	3	publicitate	Maria Andrei	1000
	4	vanzare	Robu Cristian	1000
	5	determinat	Ion Ion	1000
	6	nedeterminat	Mihai Cristea	1000

3. INSTRUCTIUNI DQL:

- ✓ **SELECT ALL**– folosind aceasta instructiune putem sa selectam toate coloanele dintr-o tabela, fara a utiliza un filtru specific.

```
22 • select * from contracte;
```

	nr_crt	tip_contract	persoana_contact	valoare_contract
▶	2	sponsorizare	Ion Alexandru	1000
	3	publicitate	Maria Andrei	1000
	4	vanzare	Robu Cristian	1000
	5	determinat	Ion Ion	1000
	6	nedeterminat	Mihai Cristea	1000

- ✓ **SELECT** - În SQL, filtrarea rezultatelor unei interogări SELECT se face folosind clauza WHERE precum in exemplul de mai jos:

```
51 • SELECT nume_societate
52 FROM firme_contractate
53 WHERE nume_societate = 'Premium Dental Center' OR anul infiintarii > 2005;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Contents: |

nume_societate
Mega Image
Mega Image
Premium Dental Center
Kosarom

- ✓ **FILTRARE CU WHERE** – cu ajutorul acestei instructiuni se afiseaza doar informatiile trecute ca si conditie WHERE.

```
• DELETE FROM contracte
WHERE nr_crt = '1';
```

- ✓ **FILTRARE CU LIKE** – folosind acest filtru se poate obtine o portiune de text dintr-o coloana.

```
47 • SELECT oras FROM firme_contractate WHERE oras LIKE '%iasa%';
```

Result Grid | Filter Rows: | Export: | Wrap Cell Contents: |

oras
Iasi
Iasi

- ✓ **FILTRARE CU AND** – acest tip de filtrare se utilizeaza pentru a combina mai multe conditii intr-o singura clauza WHERE.

```
48 • SELECT valoare_contract
49 FROM contracte
50 WHERE persoana_contact = 'Ion Ion' AND valoare_contract > 200;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

valoare_contract
1000

- ✓ **FILTRARE CU OR** – filtrarea cu OR se utilizeaza pentru a permite ca rezultatul sa contina inregistrari care indeplinesc cel putin una din conditiile specificate.

```
51 • SELECT nume_societate
52 FROM firme_contractate
53 WHERE nume_societate = 'Premium Dental Center' OR anul infiintarii > 2005;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

nume_societate
Mega Image
Mega Image
Premium Dental Center
Kosarom

- ✓ **FUNCTII AGREGATE** – in SQL functiile agregate sunt functii care opereaza pe un set de valori si intorc un rezultat agregat. Pentru partea practica am ales urmatoarele functii:

MIN () – gaseste valoarea minima intr-o coloana

```
54 • SELECT MIN(anul_infiintarii) FROM firme_contractate;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

MIN(anul_infiintarii)
1999

MAX() – gaseste valoarea maxima intr-o coloana

```
55 • SELECT MAX(anul_infiintarii) FROM firme_contractate;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

MAX(anul_infiintarii)
2023

✓ JOINURI:

INNER JOIN – aceasta instructiune returneaza doar randurile care au corespondenta in ambele tabele implicate.

```
78 * SELECT anul_infiintarii, anul_deschiderii
79 FROM contracte
80 INNER JOIN firme_contractate ON anul_infiintarii = anul_deschiderii;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

anul_infiintarii	anul_deschiderii
2022	2022
2005	2005
1999	1999
2022	2022
2023	2023
2019	2019

RIGHT JOIN - aceasta operatiune returneaza toate randurile din tabela din dreapta si randurile corespunzatoare tabelii din stanga. In cazul de fata, deoarece tabelele nu au corespondenta intre ele, valorile sunt nule.

```
81 * SELECT nume_societate, anul_deschiderii
82 FROM contracte
83 RIGHT JOIN firme_contractate ON nume_societate = anul_deschiderii;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

nume_societate	anul_deschiderii
Mega Image	NULL
La doi pasi	NULL
Sagrod SRL	NULL
Mega Image	NULL
Premium Dental Center	NULL
Kosarom	NULL

LEFT JOIN - aceasta operatiune returneaza toate randurile din tabela din stanga si randurile corespunzatoare tabelii din dreapta. In cazul de fata, deoarece tabelele nu au corespondenta intre ele, valorile sunt nule.

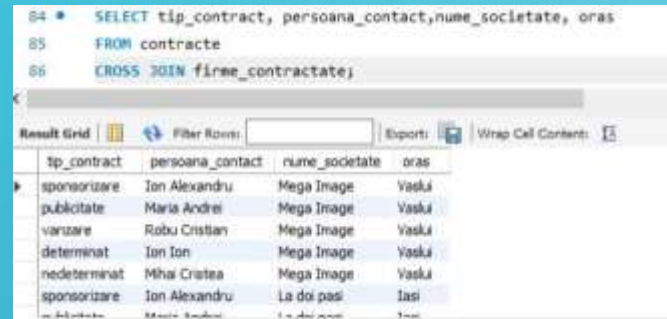
```
81 * SELECT nume_societate, anul_deschiderii
82 FROM contracte
83 Left JOIN firme_contractate ON nume_societate = anul_deschiderii;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

nume_societate	anul_deschiderii
NULL	1999
NULL	2005
NULL	2022
NULL	2023
NULL	2019

- ✓ **CROSS JOIN** – aceasta instructiune returneaza toate combinatiile posibile de randuri dintre doua tabele.

```
64 * SELECT tip_contract, persoana_contact, nume_societate, oras
65 FROM contracte
66 CROSS JOIN firme_contractate;
```



tip_contract	persoana_contact	nume_societate	oras
sponsorizare	Ion Alexandru	Mega Image	Vaslui
publicitate	Maria Andrei	Mega Image	Vaslui
vanzare	Robu Cristian	Mega Image	Vaslui
determinat	Ion Ion	Mega Image	Vaslui
nedeterminat	Mihai Cristea	Mega Image	Vaslui
sponsorizare	Ion Alexandru	La doi pasi	Iasi

- ✓ **LIMITE** – limitarea rezultatelor interogarii poate fi realizata folosind clauza LIMIT.

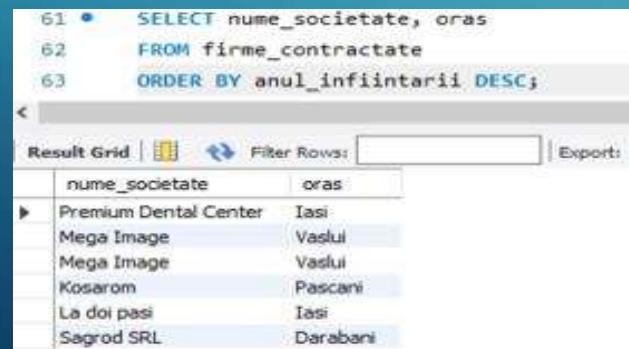
```
58 * SELECT tip_contract, persoana_contact
59 FROM contracte
60 LIMIT 3;
```



tip_contract	persoana_contact
sponsorizare	Ion Alexandru
publicitate	Maria Andrei
vanzare	Robu Cristian

- ✓ **ORDER BY** – operatiunea aceasta este utilizata pentru a sorta rezultatele unei interogari in functie de una sau mai multe coloane.

```
61 * SELECT nume_societate, oras
62 FROM firme_contractate
63 ORDER BY anul_infiintarii DESC;
```



nume_societate	oras
Premium Dental Center	Iasi
Mega Image	Vaslui
Mega Image	Vaslui
Kosarom	Pascani
La doi pasi	Iasi
Sagrod SRL	Darabani

✓ **CHEI PRIMARE** – cu ajutor acestei instructiuni se identifica in mod unic o intregistrare intr-o tabela.

```
3 • create table firme_contractate(  
4     nr_crt int primary key auto_increment not null,  
5     nume_societate Varchar (25) not null,  
6     oras Varchar (25) not null,  
7     anul_infiintarii int  
8 );
```

	nr_crt	tip_contract	persoana_contact	valoare_contract
▶	2	sponsorizare	Ion Alexandru	2000
	3	publicitate	Maria Andrei	5000
	4	vanzare	Robu Cristian	1000
	5	determinat	Ion Ion	200
	6	nedeterminat	Mihai Cristea	200
	NULL	NULL	NULL	NULL

✓ **CHEIE SECUNDARA** – folosind aceasta instructiune se stabileste o relatie intre cele doua tabele si asigura integritatea referentiala.

```
87 • create table taxe_impozite(  
88     nr_inregistrare int primary key,  
89     suma_restanta int,  
90     an_incasari int,  
91     nr_crt int,  
92     foreign key (nr_crt) REFERENCES firme_contractate (nr_crt)  
93 );
```

Result Grid

nr_inregistrare	suma_restanta	an_incasari	nr_crt
103	4000	2021	NULL
109	233	2023	NULL
204	1000	2024	NULL
* NULL	NULL	NULL	NULL

MULTUMESC!

[HTTPS://GITHUB.COM/ANDREEAHARABULA/PROIECT-TESTARE-MANUALA](https://github.com/andreeaharabula/proiect-testare-manuala)