

**UNIVERSITATEA „TIBISCUS” DIN TIMIȘOARA**  
**FACULTATEA DE CALCULATOARE**  
**ȘI INFORMATICĂ APLICATĂ**

**Style Definition:** TOC 2: Indent: Left: 0,42 cm, Hanging: 1,55 cm, Space Before: 3 pt, Tab stops: 1,55 cm, Left

**Style Definition:** TOC 3: Indent: Left: 0,85 cm, Hanging: 2,33 cm, Tab stops: 2,33 cm, Left

**Style Definition:** Caption: Centered, Space Before: 3 pt, Keep with next

# **LUCRARE DE LICENȚĂ**

**CONDUCĂTOR ȘTIINȚIFIC:**

**Lect. univ.dr. Florentina Anica Pinte**

**CANDIDAT:**

**Țăran Constantin**

**TIMIȘOARA**  
**2016**

**UNIVERSITATEA „TIBISCUS” DIN TIMIȘOARA**  
**FACULTATEA DE CALCULATOARE**  
**ȘI INFORMATICĂ APLICATĂ**



# **LUCRARE DE LICENȚĂ**

**CONDUCĂTOR ȘTIINȚIFIC:**

**Lect. univ.dr. Florentina Anica Pinte**

**CANDIDAT:**

**Țăran Constantin**

**TIMIȘOARA**

**2016**

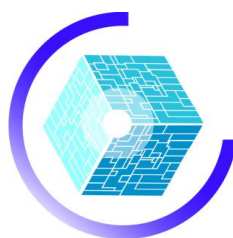
**~~Lect. univ.dr. Florentina Anica Pinte~~**

**~~CANDIDAT:~~**

**~~Țăran Constantin~~**

**TIMIȘOARA  
2016**

**Țăran Constantin**



**CALCULATOARE  
ȘI INFORMATICĂ  
APLICATĂ**

# Titlu lucrare licență

*Universitatea „Tibiscus” din Timișoara  
Facultatea de Calculatoare și Informatică Aplicată  
Departamentul de Informatică*

## REFERAT

*asupra lucrării de licență cu titlul*

---

*elaborată de candidatul* \_\_\_\_\_

*Subsemnata/ul \_\_\_\_\_ de la  
departamentul de Informatică al Facultății de Calculatoare și Informatică  
Aplicată, în calitate de coordonator științific, fac următoarele precizări:*

- lucrarea este structurată pe \_\_\_\_ capitole (\_\_\_\_ pagini), în care se tratează tema propusă;*
- lucrarea este îngrijit redactată și tehnoredactată;*

- lucrarea corespunde din punct de vedere științific, dovedind cunoașterea și aprofundarea de către candidat a domeniului abordat;

- bibliografia consultată este bine aleasă și de actualitate;

- \_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_

*În concluzie, sunt de acord cu acceptarea lucrării în vederea susținerii publice în fața comisiei de licență.*

*Propun, în calitate de coordonator științific, nota \_\_\_\_\_.*

*Timișoara,*

\_\_\_\_/\_\_\_\_/\_\_\_\_

\_\_\_\_\_

|

## Cuprins

<b>1</b>	<b>PARTEA TEORETICĂ A LUCRĂRII.....</b>	<b>5</b>
1.1	INTRODUCERE.....	5
1.2	NOTIUNI TEORETICE ȘI PRACTICE DESPRE CONTABILITATEA PRIMARĂ ÎN PARTIDĂ SIMPLĂ.....	6
1.2.1	NOTIUNI TEORETICE DESPRE CONTABILITATE.....	6
1.2.2	NOTIUNI TEORETICE DESPRE CONTABILITATEA PRIMARĂ PFA.....	11
1.2.3	NOTIUNI PRACTICE ÎN CONTABILITATEA PRIMARĂ PFA.....	13
1.3	PREZENTAREA MEDIULUI DE DEZVOLTARE A APLICAȚIEI.....	15
1.3.1	SCURTĂ PREZENTARE A PLATFORMEI .NET.....	16
1.3.2	COMPILAREA PROGRAMELOR ÎN ARHITECTURA .NET.....	17
1.3.3	SCURTĂ PREZENTARE MEDIULUI DE PROGRAMARE MICROSOFT VISUAL STUDIO.....	18
1.3.4	LIMBAJUL C#.....	22
<b>2</b>	<b>PARTEA PRACTICĂ A LUCRĂRII.....</b>	<b>32</b>
2.1	COMPONENTELE APLICAȚIEI „CONTAPFA”.....	32
2.1.1	ARHITECTURA BAZEI DE DATE.....	32
2.1.2	ARHITECTURA APLICAȚIEI CONTAPFA.....	40
2.2	FUNCȚIONAREA APLICAȚIEI.....	48
	<b>Concluzii.....</b>	<b>55</b>
	<b>Bibliografie.....</b>	<b>56</b>
<b>1</b>	<b>PARTEA TEORETICĂ A LUCRĂRII.....</b>	<b>6</b>
1.1	INTRODUCERE.....	6
1.2	NOTIUNI TEORETICE ȘI PRACTICE DESPRE CONTABILITATEA PRIMARĂ ÎN PARTIDĂ SIMPLĂ.....	7
1.2.1	NOTIUNI TEORETICE DESPRE CONTABILITATE.....	7
1.2.2	NOTIUNI TEORETICE DESPRE CONTABILITATEA PRIMARĂ PFA.....	11
1.2.3	NOTIUNI PRACTICE ÎN CONTABILITATEA PRIMARĂ PFA.....	12
1.3	PREZENTAREA MEDIULUI DE DEZVOLTARE A APLICAȚIEI.....	15
1.3.1	SCURTĂ PREZENTARE A PLATFORMEI .NET.....	15
1.3.2	COMPILAREA PROGRAMELOR ÎN ARHITECTURA .NET.....	16
1.3.3	SCURTĂ PREZENTARE MEDIULUI DE PROGRAMARE MICROSOFT VISUAL STUDIO.....	17
1.3.4	LIMBAJUL C#.....	20
<b>2</b>	<b>PARTEA PRACTICĂ A LUCRĂRII.....</b>	<b>29</b>
2.1	COMPONENTELE APLICAȚIEI „CONTAPFA”.....	29
2.1.1	ARHITECTURA BAZEI DE DATE.....	29
2.1.2	ARHITECTURA APLICAȚIEI CONTAPFA.....	35
2.2	FUNCȚIONAREA APLICAȚIEI.....	41
	<b>Concluzii.....</b>	<b>46</b>
	<b>Bibliografie.....</b>	<b>47</b>

|

4

Formatted Table

# 1 PARTEA TEORETICĂ A LUCRĂRII

## 1.1 INTRODUCERE

Prezenta lucrare poartă numele de “**Aplicație Contabilitate Primară PFA**” și a fost concepută pentru a ușura munca unui PFA în partidă simplă care desfășoară activități independente.

Formatted: Indent: First line: 1,27 cm

Persoana fizică autorizată, adică PFA, se definește ca persoană autorizată să desfășoare orice activitate economică permisă de lege, folosind în principal forța sa de muncă. Activitățile economice pe care o persoană fizică le poate desfășura ca PFA sunt cele prevăzute de codul CAEN.

PFA-ul este cea mai simplă formă de organizare. Această formă de organizare a fost gândită pentru prestări de servicii de mică anvergură. În acest caz contabilitatea poate fi ținută, teoretic și de proprietar deoarece numărul de documente obligatorii este mai mic decât în cazul unei societăți comerciale. Persoana fizică are libertatea de a alege locul și modul de desfășurare a activității, programului de lucru și poate să își desfășoare activitatea pentru mai mulți clienți.

Pornind de la cele enunțate mai sus aplicația a fost concepută pentru a ajuta o persoană fizică autorizată care își ține singură contabilitatea și are scopul de a ține la un loc și în mod organizat evidența tuturor documentelor contabile precum și a lucrărilor, contractelor și proceselor verbale.

Totodată pe măsura continuării dezvoltării acestei aplicații, se are în vedere scoaterea de rapoarte în format \*pdf, pentru a completa cu o mai mare ușurință tabelele și registrele electronice și cele în format analogic. Conceperea acestei aplicații este foarte importantă pentru un PFA deoarece simplifică mult timpul petrecut la calculator pentru a fi la zi cu toate registrele obligatorii dar și cu evidența clienților, a facturilor, a contractelor, mai pe scurt cu evidența încasărilor și plăților.

Prezenta lucrare începe cu o scurtă prezentare a ceea ce se va găsi în aceasta-, pentru ca mai apoi să fie descrisă mai pe larg teoria care a stat la baza conceperii aplicației.

În capitolul 1 se explică ce înseamnă contabilitate primară în partidă simplă, ce înseamnă un PFA. Aici vom explica ce registre sunt obligatorii de completat și cum trebuie completate. Pentru a înțelege importanța acestei aplicații trebuie să înțelegem ce înseamnă persoană fizică autorizată și cum își justifică prin diverse tabele și rapoarte câștigurile. Aici sunt menționate și obligațiile unui PFA din punct de vedere legal.

Doar după ce s-a înțeles ce înseamnă activitatea unui PFA se poate folosi această aplicație. Teoria care a stat la baza conceperii ei este descrisă în capitolul 2, iar modul de funcționare a aplicației este descris în partea practică.



## 1.2 NOȚIUNI TEORETICE ȘI PRACTICE DESPRE CONTABILITATEA PRIMARĂ ÎN PARTIDĂ SIMPLĂ

### 1.2.1 NOȚIUNI TEORETICE DESPRE CONTABILITATE

Contabilitatea are ca obiect evidența, calculul și controlul stării și mișcării patrimoniului, cu indicarea destinației și sursei lor de proveniență, a mijloacelor economice, a proceselor economice, precum și a rezultatelor economice.

Ea se clasifică în: contabilitate curentă, care cuprinde lucrările ce se efectuează zi de zi, pe parcursul perioadei de gestiune și contabilitatea periodică, care se realizează la sfârșitul perioadei de gestiune.

Contabilitatea poate fi în partidă simplă sau în partidă dublă și anume:

- *Contabilitatea în partidă simplă* presupune înregistrarea unei operații economice într-un singur cont. De exemplu, intrarea unei sume de bani în contul bancar se înregistrează la partida „Banca”. Pentru aceasta se folosește un simplu jurnal de încasări și plăți, dar nu se reflectă angajamentele (creanțe, datorii), ci numai transferurile de numerar.
- *Contabilitatea în partidă dublă* presupune reflectarea unei operații economice prin înregistrarea în două partide, concomitent, una indicând originea și cealaltă destinația. În exemplul inițial, dacă suma de bani a fost vărsată în contul bancar din casieria agentului economic, se vor folosi două partide: „Banca” (indicând destinația sumei) și „Casa” (indicând originea sumei). Contabilitatea în partidă dublă folosește o multitudine de conturi deschise pentru toate elementele patrimoniale care înregistrează mișcări.

După modul de organizare a conturilor, există sisteme de contabilitate cu un singur circuit și cu două circuite.

- *Sistemul de contabilitate cu un singur circuit (monist)* organizează conturile într-un flux unic al operațiilor economice, fără să facă distincție între operațiile care fac obiectul contabilității financiare și cele care privesc gestiunea internă.
- *Sistemul de contabilitate cu dublu circuit (dualist)* organizează conturile într-un flux dublu, delimitând operațiile cu terții, precum și rezultatele financiare (financial accounting), de circuitul gestiunii interne (managerial accounting) privind costurile, producția și rentabilitatea.

În România, la ora actuală, contabilitatea este organizată în sistem dualist, informația fiind clasificată astfel [...bibliografie]:

**Formatted:** Bulleted + Level: 1 +  
Aligned at: 0,63 cm + Indent at: 1,27  
cm, Tab stops: 1 cm, Left

**Formatted:** Bulleted + Level: 1 +  
Aligned at: 0,63 cm + Indent at: 1,27  
cm, Tab stops: 1 cm, Left

- *Informație contabilă financiară (publică sau generală)*, destinată utilizatorilor externi (acționari, investitori, salariați, creditori, administrația financiară, publicul larg). Contabilitatea financiară sau generală are caracter obligatoriu pentru toate unitățile patrimoniale.
- *Informație contabilă de gestiune destinată utilizatorilor interni (conducerea întreprinderii)*. Contabilitatea internă de gestiune se organizează de fiecare unitate patrimonială în funcție de specificul activității ei și de necesitățile proprii. Informația contabilă financiară este cuprinsă în situațiile financiare de sinteză: bilanț, cont de profit și pierdere, situația modificărilor capitalului propriu, situația fluxurilor de trezorerie, politici contabile și note explicative. Informația contabilă de gestiune este nestandardizată, uneori și nemonetară (în etalon natural-metri cubi, kilograme, metri lineari), și cuprinde informații despre costul unitar al produselor, profitul pe produs sau tendințele pe care le au costurile în funcție de volumul activității, având ca scop calcularea unor indicatori de eficiență, de productivitate, de rotație a stocurilor, foarte utili managerilor atunci când iau decizii.

**Formatted:** Font: Times New Roman, 12 pt

**Formatted:** List Paragraph, Indent: Left: 0,75 cm, Hanging: 1 cm, Bulleted + Level: 2 + Aligned at: 1,9 cm + Indent at: 2,54 cm, Tab stops: 1,25 cm, Left

Principalii utilizatori ai informației contabile sunt:

- — *Conducerea întreprinderii* care utilizează datele contabile în mod operativ pentru luarea unor decizii juste și pentru întocmirea de planuri, prognoze și bugete;
- — *Investitorii*, care sunt interesați de nivelul dividendelor care pot fi obținute sau de pierderile probabile;
- — *Salariații* sau candidații potențiali, care sunt interesați de dinamica locurilor de muncă sau de oportunități de promovare în viitor;
- — *Clienții*, care urmăresc continuarea relațiilor comerciale cu respectiva companie;
- — *Creditorii comerciali*, respectiv furnizorii de bunuri și servicii, care sunt preocupați de lichiditatea pe termen scurt a întreprinderii, deoarece doresc să își recupereze creanțele cât mai rapid;
- — *Creditorii financiari (băncile)*, care sunt interesați de capacitatea întreprinderii de a rambursa creditele și dobânzile;
- — *Auditorii*, care sunt chemați să își exprime o opinie independentă și obiectivă asupra fidelității datelor cuprinse în situațiile financiare. Tipurile de opinie favorabilă aferente auditului financiar se exprimă printr-una din formulele: «situațiile financiare dau o imagine fidelă...» sau «situațiile financiare prezintă în mod sincer în toate aspectele lor semnificative...».
- — *Ministerul de Finanțe*, care întocmește prognoze referitoare la veniturile statului, pe baza informațiilor din contabilitatea financiară furnizate de unitățile patrimoniale;
- — *Comisia Națională de Statistică*, care centralizează raportările statistice ale întreprinderilor având ca scop calcularea indicatorilor macroeconomici și elaborarea conturilor naționale;
- — *Publicul*, care este interesat să obțină informații privitoare la impactul activității întreprinderilor asupra mediului de afaceri, asupra mediului înconjurător ori asupra comunităților locale.

**Formatted:** Font: Times New Roman, 12 pt

**Formatted:** List Paragraph, Indent: Left: 1,12 cm, Bulleted + Level: 2 + Aligned at: 1,9 cm + Indent at: 2,54 cm, Tab stops: 1,5 cm, Left

Au mai fost elaborate:

- Regulamentul de aplicare a Legii Contabilității;
- Planul de Conturi General;
- Normele metodologice pentru utilizarea conturilor;

**Formatted:** Font: Times New Roman, 12 pt

**Formatted:** List Paragraph, Bulleted + Level: 2 + Aligned at: 1,9 cm + Indent at: 2,54 cm

- Modelele registrelor contabile și ale situațiilor de sinteză;
- Formularele comune privind activitatea financiar-contabilă și normele de întocmire și utilizare a acestora.

Pe plan național, cadrul juridic al contabilității se bazează pe reguli ierarhizate și definite astfel [... de unde te-ai inspirat ca pe plan național ] [4]:

- *Textele legislative (legi și ordonanțe)*. Legile sunt adoptate de Parlament și se referă la probleme fundamentale, în limitele acordate de Constituție. Ordonanțele se adoptă de către executiv, de regulă, în situații de urgență, urmând să treacă prin Parlament pentru validare.
- *Texte de reglementare (decrete și hotărâri)* sunt emise de guvern pentru situații specifice;
- *Elemente de jurisprudență (deciziile tribunalelor)* au menirea de a stabili dacă dispozițiile legale au fost corect aplicate în diverse spețe.
- *Izvoare de doctrină (răspunsuri ministeriale, circulare administrative, alte surse)* se constituie din ansamblul interpretărilor și avizelor referitoare la situații asupra cărora textele legislative și de reglementare nu au făcut precizări. După 1989, procesul de tranziție spre economia de piață a impus trecerea la un sistem nou, modern, adaptat la necesitățile de evidență și raportare actuale.

Deși marea majoritate a unităților patrimoniale din România organizează contabilitatea în partidă dublă, sunt situații în care legea permite ținerea ei în partidă simplă. Astfel, consiliile locale comunale, unitățile de cult, asociațiile de proprietari, asociațiile și persoanele fizice autorizate să desfășoare activități independente pot ține contabilitatea în partidă simplă (cu ajutorul unui jurnal de încasări și plăți).

Contabilii își pot exercita profesia în calitate de salariați ai unei întreprinderi (în baza unor contracte de muncă) sau ca experți independenți (în baza unor contracte de prestări de servicii cu clienții lor).

Experții contabili și contabilii autorizați (persoane fizice sau juridice) pot ține contabilitatea unei întreprinderi, pe baza unui contract de prestări de servicii, fiind responsabili pentru ținerea contabilității în conformitate cu prevederile legii. În situația în care managerii au nevoie de consultanță în ceea ce privește calculul impozitelor și taxelor, ei pot apela la consiliere de specialitate, din partea consultanților fiscali, membri ai Camerei Consultanților Fiscali. Uneori, managerii pot apela la servicii de evaluare (a întreprinderilor, a proprietăților imobiliare, a bunurilor mobile, a activelor financiare), pentru care apelează la experți autorizați, membri ai C.E.C.C.A.R. sau A.N.E.V.A.R. (Asociația Națională a Evaluatorilor din România).

Procedeele metodei contabilității se clasifică în:

- a. — procedee comune tuturor științelor;
- b. — procedee specifice metodei contabilității;
- c. — procedee ale metodei contabilității, utilizate și de alte discipline economice.

a) *Procedee comune tuturor științelor:*

- *Observația* este prima etapă în cercetare, în orice știință. În contabilitate ea servește la extragerea din documentele justificative a fenomenelor și proceselor economice care au avut loc în timp.

**Formatted:** Font: Times New Roman, 12 pt

**Formatted:** List Paragraph, Bulleted + Level: 1 + Aligned at: 0,63 cm + Indent at: 1,27 cm, Tab stops: 1 cm, Left

**Formatted:** Indent: First line: 1,27 cm

**Formatted:** Font: Times New Roman, 12 pt

**Formatted:** List Paragraph, Numbered + Level: 2 + Numbering Style: a, b, c, ... + Start at: 1 + Alignment: Left + Aligned at: 1,9 cm + Indent at: 2,54 cm

**Formatted:** Indent: Left: 0,12 cm, Bulleted + Level: 1 + Aligned at: 0,63 cm + Indent at: 1,27 cm, Tab stops: 0,5 cm, Left

- *Raționamentul* folosește judecățile logice pentru a ajunge la anumite concluzii, pornind de la operațiile economice reflectate în contabilitate.
- *Comparația* constă în alăturarea a două sau mai multe elemente înregistrate în contabilitate și sesizarea asemănarilor sau deosebirilor dintre ele, servind astfel la stabilirea unor concluzii și decizii pentru viitor.
- *Clasificarea* este operația de sortare și repartizare pe clase sau într-o anumită ordine, a operațiilor economice și financiare.
- *Analiza* presupune cercetarea în adâncime a unui fenomen, cu studierea sistematică a fiecărui element component în parte.
- *Sinteza* este procedeul prin care se realizează trecerea de la particular la general, de la simplu la complex. Sinteza folosește la centralizarea datelor contabile în vederea obținerii unor concluzii corecte.

b) *Procedee specifice metodei contabilității:*

- „*Bilanțul* este un procedeu reprezentativ al metodei contabilității”<sup>1</sup> ...aici dacă ai pus ghilimele trebuie să ai specificați de unde ai luat citatul [...] așa sau cu nota de subsol, procedeu al metodei contabilității, servind la centralizarea datelor din contabilitatea curentă și reflectând în expresie valorică patrimoniul, sub dublu aspect (activ/pasiv) la un anumit moment dat, conform principiului dublei reprezentări. În forma completă, bilanțul este documentul de sinteză care oferă o imagine de ansamblu a activității întreprinderii, incluzând informații privitoare la rezultate (profit/pierdere), cifra de afaceri, capitalul propriu, fluxurile de trezorerie, relațiile cu alți agenți economici, cu Bugetul de Stat, cu băncile etc.
- *Contul* este acel procedeu care servește la reflectarea fiecărui element patrimonial în parte, în mod analitic, și mișcările pe care le înregistrează într-o perioadă determinată de timp. Datorită principiului dublei înregistrări, sistemul de conturi realizează în permanență un perfect echilibru, prin care se poate verifica exactitatea datelor înregistrate în conturi.
- *Balanța* de verificare este un alt procedeu specific metodei contabilității, care realizează legătura între informațiile înregistrate în conturi și bilanț. Informațiile cuprinse în balanța de verificare se referă la volumul modificărilor care au avut loc în structura patrimoniului, atât în perioada curentă cât și cumulată.

**Formatted:** Font: Times New Roman, 12 pt

**Formatted:** List Paragraph, Bulleted + Level: 1 + Aligned at: 0,63 cm + Indent at: 1,27 cm, Tab stops: 1 cm, Left

c) *Procedee ale metodei contabilității, utilizate și de alte discipline economice:*

- *Documentația* presupune folosirea pentru fiecare operație economică și financiară, a unui document justificativ, care face dovada înfăptuirii lor, și prin care se poate verifica justetea lor și respectarea disciplinei financiare și contractuale.
- *Evaluarea* presupune transformarea fenomenelor și proceselor economice, cuantificabile în etalon natural în date cuantificabile în unități monetare. Prin aducerea lor la un numitor comun, se pot efectua analize financiare de cea mai mare importanță pentru manageri.
- *Calculația* constă în efectuarea de operații aritmetice în documentele primare, în calculul rulajelor, a totalului sumelor și a soldurilor finale. Calculația se folosește și la inventarierea patrimoniului, la reflectarea cheltuielilor, veniturilor, a rezultatelor finale etc.

**Formatted:** Font: Times New Roman, 12 pt

**Formatted:** List Paragraph, Bulleted + Level: 1 + Aligned at: 0,63 cm + Indent at: 1,27 cm, Tab stops: 1 cm, Left

<sup>1</sup>

- **Inventarierea patrimoniului** reprezintă un ansamblu de operații prin care se constată existența elementelor de activ și pasiv, cantitativ și valoric, sau numai valoric, aflate în patrimoniul întreprinderii la un moment dat.

Unul dintre procedeele metodei contabilității este acela al documentației, care presupune ca toate operațiunile economico-financiare care afectează patrimoniul trebuie fundamentate și justificate prin acte scrise (documente justificative) înainte de înregistrarea în conturi.

*Documentele justificative* stau la dispoziția acționarilor, cenzorilor, organelor fiscale și altor organe de control pentru a se verifica modul de desfășurare a activității, a respectării dispozițiilor legale referitoare la calculul și plata impozitelor și taxelor.

Structura documentelor de evidență este particularizată în funcție de tipul operațiunii în cauză, dar există câteva *elemente obligatorii*, și anume:

- denumirea documentului (exemple: factură, extras de cont, nota de intrare-recepție, stat de plată a salariilor, bon de consum etc.);
- antetul care cuprinde denumirea și adresa agentului economic care a emis documentul; denumirea părților care au contribuit la efectuarea operațiilor consemnate;
- numărul și data documentului;
- descrierea operațiilor economico-financiare, uneori cu specificarea bazei actului normativ care a stat la baza înregistrării; semnăturile persoanelor care răspund de efectuarea operațiilor economice, și ale celor cu funcții de control financiar preventiv;
- alte elemente care asigură o consemnare completă a operațiilor în documente.

**Formatted:** Bulleted + Level: 1 + Aligned at: 0,63 cm + Indent at: 1,27 cm, Tab stops: 1 cm, Left

Condițiile necesare pentru ca un document contabil să fie considerat valabil sunt următoarele: să fie scris clar și citeț, eliminându-se orice controversă sau posibilitate de interpretare, să nu conțină ștersături sau corecturi, să aibă anulate rândurile libere, să fie întocmit în termenul legal, să conțină date exacte și reale, sumele pentru valori bănești să fie scrise în cifre și litere.

După **criteriul de clasificare**, documentele contabile se pot cataloga în felul următor **[bibliografie...]**:

**Formatted:** Font: Bold

**a) a)** După modul de întocmire și rolul lor în cadrul sistemului informațional-decizional pe care îl îndeplinesc în cadrul unității patrimoniale, deosebim:

**Formatted:** Numbered + Level: 1 + Numbering Style: a, b, c, ... + Start at: 1 + Alignment: Left + Aligned at: 0,75 cm + Indent at: 1,38 cm, Tab stops: 1,25 cm, Left

- *documente justificative* care asigură datele de intrare în sistemul informațional contabil și care sunt întocmite în momentul efectuării operațiilor economice (exemple: ordinul de plată, chitanța);
- *documente de evidență contabilă (registre de contabilitate)* care realizează înregistrarea și stocarea datelor în conturi;
- *documente de sinteză și raportare contabilă (situații financiare)* prin care se centralizează și transmit informațiile. Ele servesc la cumularea informațiilor cuprinse în documentele justificative, și se întocmesc la termene precise.

**Formatted:** Indent: Left: 1,37 cm, Bulleted + Level: 2 + Aligned at: 1,9 cm + Indent at: 2,54 cm, Tab stops: 1,75 cm, Left

**b) b)** După regimul de tipărire deosebim:

**Formatted:** Numbered + Level: 1 + Numbering Style: a, b, c, ... + Start at: 1 + Alignment: Left + Aligned at: 0,75 cm + Indent at: 1,38 cm, Tab stops: 1,25 cm, Left

- *documente tipizate*, al căror conținut, formă și format sunt standardizate prin Nomenclator. Informațiile cuprinse în ele sunt fixe (cu caracter repetitiv și tipărite) și variabile (completate de fiecare agent economic, în funcție de specificul operațiunilor consemnate). De asemenea, informațiile pot fi repartizate sub formă de chestionar (exemple: procese-verbale, acte de constatare etc.), tabel (exemple: state de plată a

**Formatted:** Bulleted + Level: 1 + Aligned at: 0,63 cm + Indent at: 1,27 cm, Tab stops: 1 cm, Left

salariilor, liste de avans chenzinal etc.), sau prin grupare (exemple: articolele înscrise în factură). Formatul formularelor tipizate variază în funcție de mărimea suportului de hârtie și carton utilizate (A3, A4, A5, X4, X5, X6).

- *documente netipizate* la care conținutul și forma nu sunt prestabilite, acestea lăsându-se la latitudinea fiecărei unități patrimoniale (exemple: situațiile de repartizare a cheltuielilor indirecte, de fundamentare a indicatorilor bugetului de venituri și cheltuieli etc.).

c) După regimul de utilizare și păstrare deosebim:

- *documente cu regim special* pentru care există un sistem unitar de înseriere și numerotare, stabilit de Ministerul Finanțelor;
- *documente fără regim special* pentru care nu s-au stabilit norme stricte, ci numai unele generale.

d) După destinația sau funcția pe care o îndeplinesc, deosebim:

- *documente de dispoziție* prin care se ordonă efectuarea unei anumite operații (exemple: ordinul de plată);
- *documente mixte (combinat)* care atestă atât dispoziția cât și execuția operației (exemple: cec de numerar, foaie de vărsământ cu chitanță);

e) După circuitul lor deosebim:

- *documente interne* care circulă numai în interiorul unității (exemplu: fișa mijlocului fix);
- *documente externe* care circulă și în afara unității (exemplu: avizul de însoțire a mărfii).

f) După conținutul lor, deosebim:

- *documente primare*, care înregistrează informațiile în momentul producerii lor (exemplu: nota de plată);
- *documente centralizatoare*, care cumulează informațiile din mai multe documente de aceeași natură și din aceeași perioadă (exemple: centralizatorul lunar al bonurilor de consum).

g) După natura elementelor patrimoniale deosebim:

- *documente privind mijloacele fixe*;
- *documente privind stocurile*;
- *documente privind disponibilitățile bănești etc.*

**Formatted:** Numbered + Level: 1 + Numbering Style: a, b, c, ... + Start at: 1 + Alignment: Left + Aligned at: 0,75 cm + Indent at: 1,38 cm, Tab stops: 1,25 cm, Left

**Formatted:** Bulleted + Level: 1 + Aligned at: 0,63 cm + Indent at: 1,27 cm, Tab stops: 1 cm, Left

**Formatted:** Normal, Numbered + Level: 1 + Numbering Style: a, b, c, ... + Start at: 1 + Alignment: Left + Aligned at: 0,75 cm + Indent at: 1,38 cm, Tab stops: 1,25 cm, Left

**Formatted:** Bulleted + Level: 1 + Aligned at: 0,63 cm + Indent at: 1,27 cm, Tab stops: 1 cm, Left

**Formatted:** Indent: Left: 0,62 cm, Bulleted + Level: 1 + Aligned at: 1,9 cm + Indent at: 2,54 cm, Tab stops: 1 cm, Left

**Formatted:** Font: Italic

**Formatted:** Indent: Left: 0,62 cm, Bulleted + Level: 1 + Aligned at: 1,9 cm + Indent at: 2,54 cm, Tab stops: 1 cm, Left

**Formatted:** Font: Italic

**Formatted:** Font: Italic

**Formatted:** Font: Italic

**Formatted:** Indent: Left: 0,62 cm, Bulleted + Level: 1 + Aligned at: 1,9 cm + Indent at: 2,54 cm, Tab stops: 1 cm, Left

**Formatted:** Font: Not Italic

**Formatted:** Font: Not Italic

**Formatted:** Font: Not Italic

## 1.2.2 NOȚIUNI TEORETICE DESPRE CONTABILITATEA PRIMARĂ PFA

Persoana fizică autorizată, adică PFA, se definește ca persoana autorizată să desfășoare orice activitate economică permisă de lege, folosind în principal forța sa de muncă. Activitățile economice pe care o persoană fizică le poate desfășura ca PFA sunt cele prevăzute de codul CAEN.

PFA-ul este cea mai simplă formă de organizare. Această formă de organizare a fost gândită pentru prestări de servicii de mică anvergură. În acest caz contabilitatea poate fi ținută teoretic și de proprietar deoarece numărul de documente obligatorii este mai mic decât în cazul unei societăți comerciale, etc. Persoana fizică are libertatea de a alege locul și modul de



desfășurare a activității precum și programului de lucru, poate să își desfășoare activitatea pentru mai mulți clienți.

Contabilitatea în partidă simplă se aplică următoarelor categorii de contribuabili:

- persoanele fizice ale căror venituri sunt supuse impozitului pe venit în conformitate cu Codul Fiscal, al căror venit net anual este determinat în sistem real, pe baza datelor din contabilitate, fiind obținut din următoarele surse:

- activități independente;
- cedarea folosinței bunurilor
- activități agricole, sivicultura și piscicultura.

- persoanele sau entitățile care, prin actul normativ de înființare, prin legi speciale sau prin alte acte normative, au obligația ținerii contabilității în partidă simplă.

Persoanele fizice autorizate pot utiliza toate formularele sau doar o parte, în funcție de activitatea desfășurată. Înregistrarea veniturilor în contabilitate în partidă simplă se ține în funcție de tipul veniturilor, astfel:

- venituri din activități de comerț;
- venituri din profesii libere
- alte venituri.

Contribuabilii care realizează venituri dintr-o singură activitate trebuie să treacă totalul veniturilor în Fisa de operațiuni diverse, iar pentru cei care încasează în numerar toate veniturile, le pot evidenția numai în Registrul-Jurnal de încasări și plăți. Fișa de operațiuni diverse nu mai este obligatorie în acest caz.

Contribuabilii trebuie să întocmească, în funcție de activitatea desfășurată, de frecvența încasării sau de tipul serviciilor prestate, formularele Factură, Factură fiscală sau Chitanță. Ei au obligația să întocmească factura pentru sumele încasate, atât pe bază de chitanță, bon fiscal cât și prin bancă în relațiile cu persoanele juridice, iar în relațiile cu persoanele fizice trebuie să întocmească factura, chitanța fiind întocmită doar în cazul încasării în numerar.

Evidența cheltuielilor se ține în funcție de felul acestora:

- cheltuieli în interesul direct al activității;
- cheltuieli de sponsorizare;
- cheltuieli de protocol;
- alte cheltuieli deductibile.

Persoanele fizice care realizează venituri din activități independente au următoarele obligații:

- să organizeze contabilitatea în partidă simplă conform reglementărilor în vigoare;
- să completeze Registrul-Jurnal de încasări și plăți, Registrul Inventar cât și alte documente contabile.

Registrele utilizate în evidența unui PFA sunt:

- Registrul-Jurnal de încasări și plăți în care contribuabilii înregistrează operațiunile efectuate;
- Registrul Inventar în care se înregistrează bunurile achiziționate sau realizate;
- Registrul Unic de Control – solicitarea acestui registru se face în termen de 30 zile de la constituirea punctului de lucru;
- Registrul de evidență a salariaților în cazul în care PFA are angajați, trebuie să completeze acest registru și să-l înregistreze la Inspectoratul de muncă de pe raza județului Timiș.

Mai exact încasările din Registrul-Jurnal de încasări și plăți vor cuprinde:

- sume încasate din desfășurarea activității;

**Formatted:** Bulleted + Level: 1 +  
Aligned at: 0,63 cm + Indent at: 1,27  
cm, Tab stops: 1 cm, Left

**Formatted:** Bulleted + Level: 1 +  
Aligned at: 0,63 cm + Indent at: 1,27  
cm, Tab stops: 1 cm, Left

**Formatted:** Indent: First line: 1,27  
cm

**Formatted:** Indent: First line: 0 cm

**Formatted:** Bulleted + Level: 1 +  
Aligned at: 0,63 cm + Indent at: 1,27  
cm, Tab stops: 1 cm, Left

**Formatted:** Indent: First line: 0 cm

**Formatted:** Bulleted + Level: 1 +  
Aligned at: 0,63 cm + Indent at: 1,27  
cm, Tab stops: 1 cm, Left

**Formatted:** Indent: First line: 0 cm

**Formatted:** Indent: Left: 1,62 cm,  
Bulleted + Level: 1 + Aligned at: 0,63  
cm + Indent at: 1,27 cm, Tab stops:  
2 cm, Left

- aporturile în numerar sau prin conturi făcute la începerea activității sau în cursul desfășurării acesteia;
- sume primite sub forma de credite bancare sau împrumuturi;
- sume primite ca despăgubiri;
- sume primite sub forma de sponsorizări, mecenat sau donații;
- subvenții;
- alte sume încasate.

În cazul plăților, acestea vor cuprinde:

- plățile efectuate în cadrul activității desfășurate în scopul realizării de venituri;
- sumele reprezentând restituirea aporturilor în numerar și prin conturi bancare;
- sumele reprezentând rambursarea de credite bancare sau alte împrumuturi;
- alte plăți efectuate (de exemplu penalități, amenzi).

### 1.2.3 NOȚIUNI PRACTICE ÎN CONTABILITATEA PRIMARĂ PFA

Registrul de evidență a lucrărilor este important pentru o bună organizare a lucrărilor preluate de la clienți. Acest registru, în cazul nostru, face legătura între lucrarea preluată de la client și depusă la OCPI/ANCPI, numărul primit și numărul de contract primit. În acest tabel se poate vedea cu ușurință stadiul lucrării la OCPI/ANCPI, avizatorii/registratorii care se ocupă de ea, felul lucrării precum și termenul de soluționare. O parte din acest tabel se poate vedea mai jos:

Formatted: Justified

**Tabel 1-1. Stadiul lucrărilor înregistrate la OCPI/ANCPI**

Formatted: Caption

Lucrare acceptata/respinsa	Nr. Inregistrare la OCPI/ANCPI	Termen solutionare	Avizator/Registrator	Tipul lucrării	Nr.pr.
acceptata	11067/26.01.2015	16.02.2015	Oana/ Kakuja/ Roseti	Intab.	01/2015
acceptata	11079/26.01.2015	16.02.2015	Rancov/ Ciocani/ Erk	Intab.	02/2015
acceptata	11080/26.01.2015	16.02.2015	Rancov/ Ciocani/ Erk	Intab.	03/2015
acceptata	33332/27.02.2015	20.03.2015	Rancov/ Tolcea/ Roseti	Edificare constructie	04/2015
acceptata	46787/17.03.2015		Bazosan/	Apartamentare	05/2015
acceptata	28482/05.03.2015	25.03.2015	Bazosan/ Roseti/ Roseti	Apartamentare	06/2015

La preluarea unei lucrări, după o discuție în prealabil cu clientul, se semnează un *Contract-Notă de comandă*. Acesta va cuprinde la început datele de identificare ale părților, apoi pe rând obiectul lucrării, termenul de execuție și onorariul. Acest contract va fi semnat de ambele părți în dublu exemplar. La obiectul lucrării se descrie ce anume se dorește de la PFA să execute, iar la termenul de execuție se stabilește împreună cu clientul durata contractului. Aici în contract se mai pot preciza diverse condiții în funcție de situație. Un exemplu de contract este redat mai jos:

Formatted: Justified

Formatted: Font: 3 pt



<b>P.F.A. TĂRAN CONSTANTIN</b> Certificat de Autorizare Seria TM Nr.116, C.I.F. 30178884 Str. Sirius Nr.21, Ap.9, Timisoara, Cod postal: 300685	Tel: 0726.174.783 Fax: 0356.818.380 E-mail: cosmintaran@yahoo.com
---	---

**CONTRACT- NOTA DE COMANDA**  
Nr. 02 din 22.02.2016

**P.F.A. TARAN CONSTANTIN** cu sediul in Timisoara, str. Sirius,Nr.21, Ap.9, C.I.F.: 30178884/2012, persoana fizica autorizata reprezentata legal de catre domnul Ing. Dipl. **TARAN CONSTANTIN**, in calitate de **PRESTATOR** si

**CRACIUN ADRIAN** cu domiciliul in mun. Timisoara, strada Gavril Musicescu nr.16, Sc. B, etaj. 3, ap. 14, jud. Timis, identificat prin CNP: 1871111350070, C.I. Seria TZ nr. 182498, in calitate de **BENEFICIAR** au convenit incheierea prezentei note de comanda-contract:

**OBIECTUL LUCRARII:**

- intocmire documentatie de **dezlipire**, pentru imobilul cu nr. cad: 404341, in scris in C.F. nr. 404341, UAT Remetea Mare loc. Remetea Mare [S=1000 mp], avizarea acestuia de catre OCPI Timis si obtinerea Referatului de Admitere.

**TERMEN DE EXECUTIE:**

Termenul de executie al lucrarii este de 20 zile lucratoare/documentatie, de la data inmanarii actelor necesare operatiunilor de mai sus, de catre beneficiar.

In cazul imposibilitatii realizarii lucrarii la termen, din motive independente de prestator, adica:

- imposibilitatea realizarii operatiunilor de teren datorita conditiilor meteorologice nefavorabile
- lipsa unor documente ce trebuie furnizate de catre beneficiar sau de catre O.C.P.I. Timis
- timpul de reactie intarziat ale institutiilor emitente de avize si acorduri
- existenta unor documente cu inregistrari incorecte etc., termenul de predare va fi decalat.

**Figura 1-1. Exemplu de contract**

După realizarea în teren a unei lucrări și după finalizarea lucrării la OCPI se va semna cu proprietarul un *Proces-verbal* în care se va trece foarte clar ce acte au fost primite și predate și ce anume s-a executat de către PFA. Acest proces verbal diferă de la caz la caz în funcție de ce anume se dorește să se execute. Unul din modelele de proces-verbal este prezentat mai jos, în Figura 1-2:

Formatted Table

Formatted: Centered

Formatted: Caption, Indent: First line: 0 cm

Formatted: Justified

## PROCES VERBAL de trasare

Nr.01 din 09.04.2016

Incheiat in vederea receptiei operatiunii de trasare si materializare in teren, a punctelor ce definesc intersectia axelor precum si cota zero, pentru obiectivul de investitie: **Construire casa P+1E si imprejurimi** pe amplasamentul imobilului cu nr.cad.410878, inregistrat in C.F. nr. 410878, localitatea Timisoara, strada Linistei, nr. FN.

Astfel au fost trasate si materializate in teren prin tarusi metalici, punctele ce definesc intersectia axelor A-4, B-1, E-1 si E-4 ale obiectivului de investitie, in conformitate cu plansa **A-03 Faza D.T.A.C. – Plan Parter** din proiectul **nr. 14/2015** pusa la dispozitie de catre beneficiar.

Cota zero a fost stabilita la **+0.70 m** fata de cota actuala a carosabilului din fata imobilului, rezultand o cota absoluta **90.978m** – plan de referinta **Marea Neagra 1975**, conform specificatiilor proiectantului.

Trasarea axelor fundatiei s-a executat cu statia totala Nikon NPL-362 cu precizia de 3" din punctul de statie 201 cu vize spre punctele de statie 200 si 202 .

Coordonatele punctelor de statie:

**Figura 1-2. Model de proces verbal**

Dupa ce au fost finalizate cele de mai sus, la final de luna PFA-ul isi noteaza, pe baza facturilor, bonurilor fiscale, chitanțelor operatiunile efectuate in acea luna, indiferent daca sunt incasari sau plati. Acest tabel este copia fidelă a registrului de incasari și plati analogic.

Documentul (felul, nr.)	Felul operatiunii (explicatii)	Incasari				numerar
		numerar	Pret fara TVA	Valoare TVA	banca	
2	3	4			5	6
CH TM3901872/ FF TM 3067841	Plata servicii OCPI					60.00
BF 0095	Plata servicii xerox					16.00
CI nr. 6/ FF nr. 6	Prestari servicii	1529.85	1233.75	296.10		
CI nr.7/ FF nr. 7	Prestari servicii	3850.00	3104.84	745.16		
DP 13	Plata utilitati					123.42
DP 14	Cheltuieli transport in comun					76.40
<b>Total</b>		<b>5379.85</b>	<b>4338.59</b>	<b>1041.26</b>	<b>0.00</b>	<b>275.82</b>

**Figura 1-3. Registru de incasari si plati**

### 1.3 PREZENTAREA MEDIULUI DE DEZVOLTARE A APLICAȚIEI

Pentru dezvoltarea prezentei aplicații s-a optat pentru limbajul de programare C# (#-Sharp-) dezvoltat de către compania Microsoft, limbajul fiind unul modern, foarte puternic și total orientat pe obiect (O.O.P). Acesta vine cu un I.D.E excepțional, Visual Studio și are în spate unul din cele mai puternice Framework-uri de pe piață la ora actuală și anume .NET.

### 1.3.1 SCURTĂ PREZENTARE A PLATFORMEI .NET

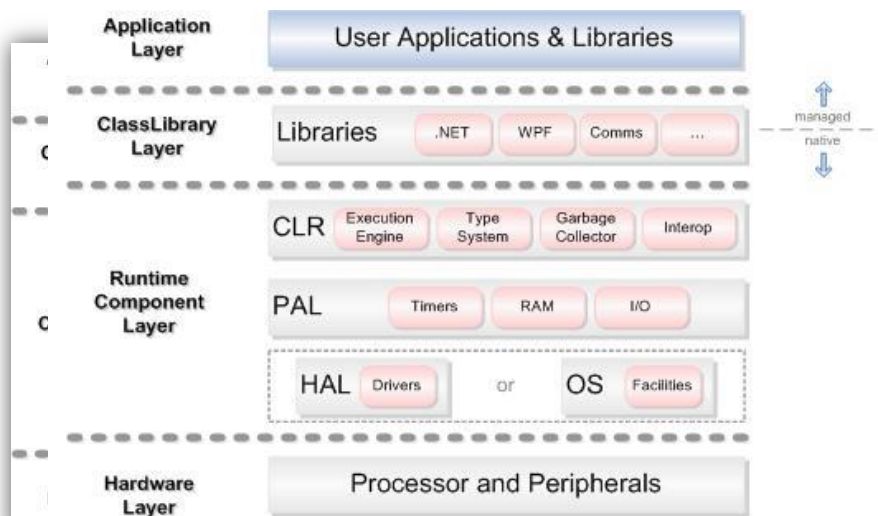
.NET este un Framework pentru dezvoltarea software unitară care permite atât dezvoltarea precum și rularea aplicațiilor Windows, atât desktop cât și WEB.

Framework-ul îmbină mai multe tehnologii laolaltă (ASP,XAML,OOP, SOAP, etc) și limbaje de programare (C#, VB.NET, SQL, C++, F#, C++/CLI), asigurând portabilitate a codului compilat pe orice calculator ce rulează Windows.

Toate limbajele de programare incluse în platforma .NET trebuie să respecte o serie de specificații OOP numite "Common Type System" . Elementele de bază ale "Common Type System" sunt:

- Clase , Interfețe, Delegări
- Tipuri valoare, Tipuri Referință
- Mecanisme: moștenire, polimorfism, gestionarea și tratarea excepțiilor.

Platforma comună pe care se execută programele este numită "Common Language Runtime" CLR, din care face parte și CTS –ul.



Figură 1-1. Figura 1-4. Arhitectura .NET Sursa: Microsoft MSDN

În prezent, se află în dezvoltare proiectul MONO, care este o variantă open source a platformei .NET și se adresează sistemelor de operare UNIX. Varianta curentă și stabilă a platformei MONO este 4.2.

Formatted: Justified

### 1.3.2 COMPILAREA PROGRAMELOR ÎN ARHITECTURA .NET

Orice program scris într-un limbaj de programare suportat de platforma .NET cu respectarea specificațiilor OOP "Common Language Specification" va fi compilat într-un limbaj intermediar numit "Microsoft Intermediate Language", rezultând fișiere cu extensia .exe sau .dll. Fișierul .exe nu este un executabil nativ ca cel generat de C++ de exemplu ci respectă un format IL care va fi rulat de către CLR, care include o mașină virtuală ce execută instrucțiunile din fișierul rezultat în urma compilării.

Formatted: Justified

Mașina virtuală a CLR-ului se folosește de un compilator special numit "Just In Time" (JIT), care analizează codul și produce codul mașină al aplicației.

Datorita limbajului IL care este la fel indiferent de limbajul de programare care la produs, obiectele și clasele scrise într-un limbaj pot fi folosite și în altul. Pe lângă aceasta CLR se ocupă și de gestionarea automată a memoriei degrevându-l astfel pe programator de această problemă și astfel eliminând riscul de apariție a memory leak-urilor, cum se întâmplă în limbajele unmanaged ex, C++.

În limbajele de programare unmanaged ex. C++, cu care vom face și o comparație, cum spuneam și mai sus cade în sarcina programatorului să gestioneze memoria (alocări dealocări), prin intermediul operatorilor new și delete. Și în limbajul C# există operatorul new pentru alocare de memorie dar nu și operatorul delete, ne mai fiind nevoie de acesta datorită garbage collector-ului.

Formatted: Justified

Gestionarea manuală a memoriei are atât avantaje cât și dezavantaje, după cum urmează:

- Avantaje:
  - permite un control asupra momentului în care un obiect se naște sau moare, astfel se folosește exact atâta memorie cât este necesare, un lucru foarte bun în special în programarea embedded. Ca o bună practică în acest domeniu se recomandă a nu se folosi memorie alocată dinamic, deci evitarea operatorilor gen: new, malloc, calloc, realloc.
  - permite folosirea destructorilor rezultând o ștergere corespunzătoare a memorie.
- Dezavantaje:

Formatted: Justified, Indent: Hanging: 0,81 cm, Tab stops: 3,5 cm, Left

- poate apărea riscul de a apela prea târziu operatorul delete sau chiar niciodată. În general nu este o eroare fatală dar duce la ocuparea memoriei cu "gunoaie" și poate duce la o creștere necontrolată a memoriei ocupate de respectivul program, în final putându-se ajunge și la crash-uri.
- poate apărea riscul apelării prea rapide a operatorului delete ceea ce este o problemă mult mai gravă decât cea de mai sus, ducând direct închiderea aplicației.

### 1.3.3 SCURTĂ PREZENTARE MEDIULUI DE PROGRAMARE MICROSOFT VISUAL STUDIO

Mediul de programare Microsoft Visual Studio este un mediu de dezvoltare integrat modern, flexibil care permite crearea de aplicații în limbajele de programare C#, Visual Basic. NET, C++ sau F#, atât în consolă cât și aplicații cu interfață grafică pentru toate platformele Windows.

Pe lângă limbajele amintite mai sus Visual Studio mai oferă suport și pentru alte limbaje: M, Python, Ruby, XML/XSLT, HTML/XHTML, JavaScript, CSS etc.

#### Istoricul versiunilor:

**Tabel 1-2. Istoricul versiunilor**

Denumire	Versiune	Versiunea .NET Framework	Anul lansării
Visual Studio	4.0	N/A	1995
Visual Studio 97	5.0	N/A	1997
Visual Studio 6.0	6.0	N/A	1998
Visual Studio .NET (2002)	7.0	1.0	2002
Visual Studio .NET (2003)	7.1	1.1	2003
Visual Studio 2005	8.0	2.0	2005
Visual Studio 2008	9.0	3.5	2007
Visual Studio 2010	10.0	4.0	2010
Visual Studio 2012	11.0	4.5	2011
Visual Studio 2013	12.0	4.5.1	2013
Visual Studio 2015	14.0	4.6	2015

#### 1. Configurarea IDE

La prima rulare a aplicației există posibilitatea de a pre-seta IDE-ul în funcție de limbajul de programare sau tipul de aplicații dezvoltate. Opțiunea „General Development Settings” aplică cele mai uzuale setări valabile pentru toate limbajele de programare suportate de Visual Studio.

După deschidere se pot identifica ferestrele meniurile, barele de meniu precum și fereastra principală, după cum se poate vedea și în imaginea de mai jos.

Formatted Table

Formatted: Justified, Indent: Hanging: 0,81 cm, Tab stops: 3,5 cm, Left

Formatted: Justified, Indent: Left: 0 cm, First line: 1,27 cm

Formatted: Justified, Indent: First line: 1,27 cm

Formatted: Justified

Formatted: Caption, Indent: First line: 0 cm

Formatted: Font: 5 pt, Not Bold

Formatted: Font: 11 pt

Formatted: Font: 11 pt

Formatted: Font: 11 pt

Formatted: Font: 11 pt

Formatted: Font: 11 pt

Formatted: Font: 11 pt

Formatted: Font: 11 pt

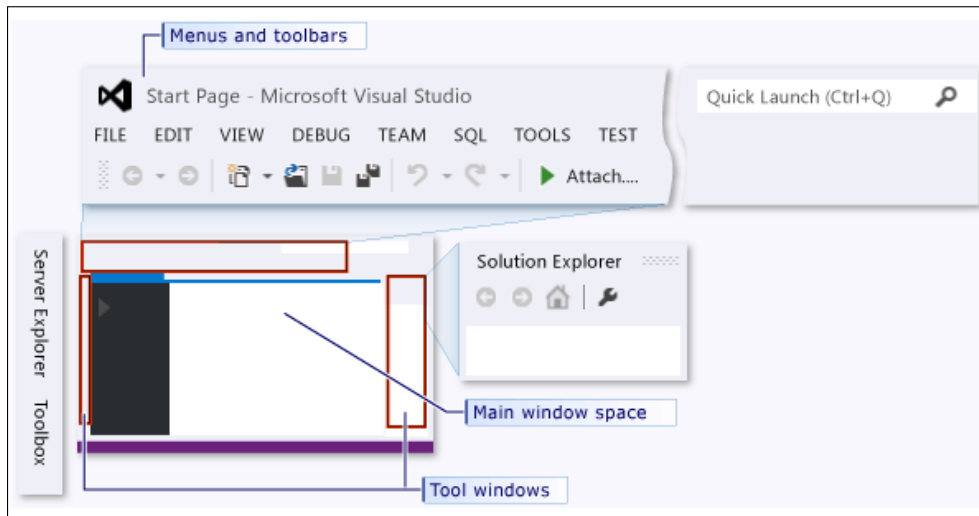
Formatted: Font: 11 pt

Formatted: Font: 11 pt

Formatted: Font: 11 pt

Formatted: Font: 11 pt

Formatted: Justified



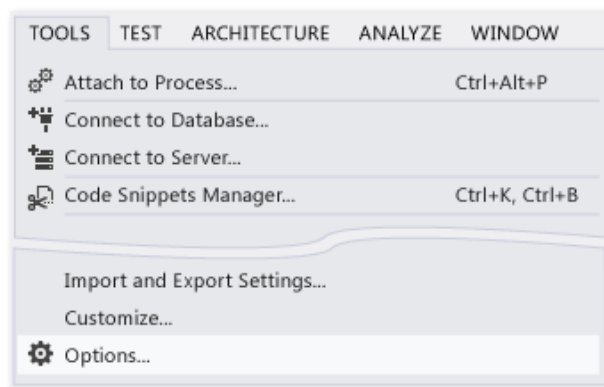
**Figura 1-5. Start Page în MVS / sursa:- Sursă:** <https://msdn.microsoft.com/en-us/library/jj153219.aspx>

**Figură 1-2**

Pentru a schimba layout-ul aplicației din meniul Tools se selectează Options – General și din Color Theme se pot alege opțiunile Dark sau Light.

Formatted: Justified

### Meniul Tools

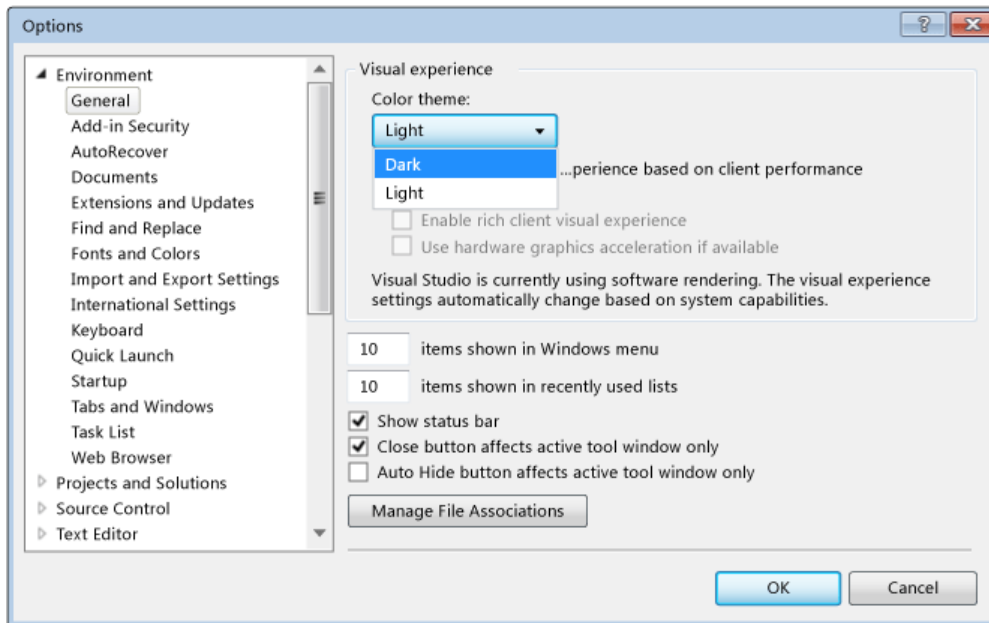


Formatted: Centered

—Figură 1-3

**Figura 1-6. Meniul Tools / sursa: —** <https://msdn.microsoft.com/en-us/library/jj153219.aspx>

### Meniul Options



Figură 1-4

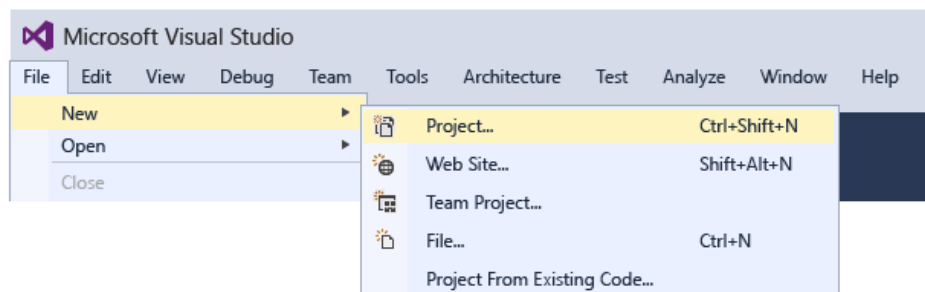
**Figura 1-7. Meniul Options / sursa:** <https://msdn.microsoft.com/en-us/library/jj153219.aspx>

## 2. Crearea unei simple aplicații în Visual Studio și C#

Atunci când se creează o nouă aplicație prima dată se generează un proiect nou precum și o soluție nouă. De exemplu vom crea un proiect Windows Presentation Foundation (WPF)

Formatted: Justified

Din meniul File se aleg opțiunile New, Project...

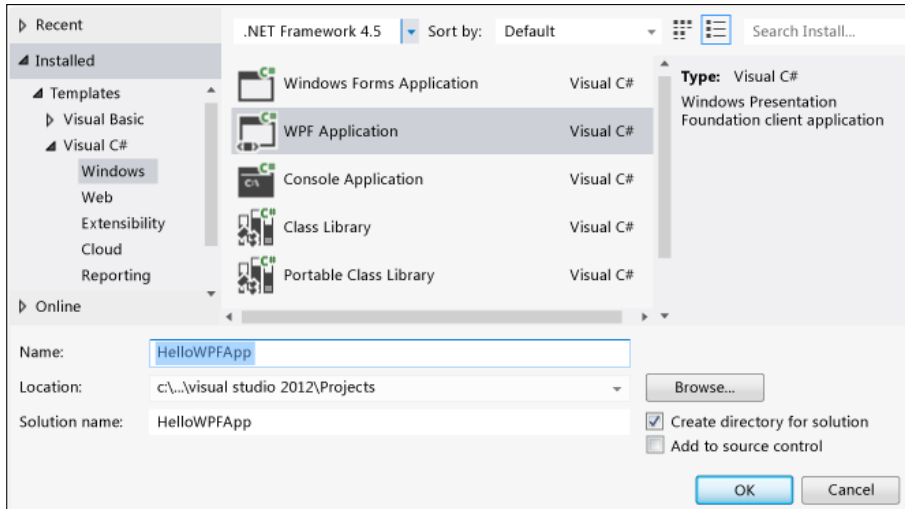


Formatted: Centered

Figura 1-8. Figură 1-5

**Meniuri / sursa: Sursa:** <https://msdn.microsoft.com/en-us/library/jj153219.aspx>

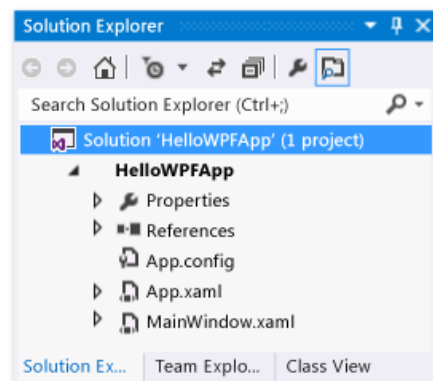
Din fereastra nou deschisă se alege C# cu șablonul „Visual WPF” iar în partea de jos a ferestrei se alege numele proiectului.



**Figura 1-9. / Figură 1-6**

Sursa: <https://msdn.microsoft.com/en-us/library/jj153219.aspx>

După acești pași Visual Studio creează proiectul „HelloWPFApp” precum și soluția.



**Figură 1-7**

**Figura 1-10. / Sursa:** <https://msdn.microsoft.com/en-us/library/jj153219.aspx>

În acest moment Visual Studio este pregătit pentru a crea aplicații C# WPF

### 3. Variante disponibile de Visual Studio

Visual Studio este disponibil în mai multe variante:



- Community fost Express,
- Professional,
- Premium,
- Ultimate.

### 1.3.4 LIMBAJUL C#

#### 1.3.4.1 TIPURI DE DATE FUNDAMENTALE IN C#

Limbajul C# include două categorii de tipuri de date și anume tipuri valorice și tipuri referință.

Conform MSDN (Microsoft Developer Network) putem clasifica tipurile de valoare după cum urmează:

##### 1. Tipuri Întregi:

În C# sunt definite nouă tipuri întregi: char, byte, sbyte, short, ushort, int, uint, long, ulong. Cu excepția lui char toate sunt folosite în calculele numerice.

**Tabel 1-3. Domeniul tipurilor întregi:**

Tipul de dată	Biți	Domeniul
byte	8	0 la 255 (fără semn)
sbyte	8	-128 la 127
short	16	-32768 la 32767
ushort	16	0 la 65535 (fără semn)
int	32	-2147483648 la 2147483647
uint	32	0 la 4294967296 (fără semn)
long	64	-9223372036854775808 la 9223372036854775807
ulong	64	0 la 18446744073709551616 (fără semn)

##### 2. Tipuri în virgulă mobilă:

Se folosesc pentru reprezentarea fracțională a numerelor și sunt de două tipuri: float și double.

**Tabel 1-4. Tipuri în virgulă mobilă**

Tipul de dată	Biți	Domeniul
float	32	-3.402823e38 la 3.402823e38
double	64	-1.79769313486232e308 la 1.79769313486232e308

##### 3. Tipul decimal:

Tipul decimal nu se găsește în C++, C sau Java și este folosit în calculele monetare pentru a se evita erorile de rotunjire. Se reprezintă precis până la 28 de poziții zecimale.

Formatted Table

Formatted: Justified, Bulleted + Level: 1 + Aligned at: 1,9 cm + Indent at: 2,54 cm

Formatted: Indent: Hanging: 0,77 cm

Formatted: Indent: Left: 1,27 cm

Formatted: Justified

Formatted: Justified, Indent: Left: 0 cm, First line: 1 cm

Formatted: Justified, Indent: Left: 0 cm

Formatted: Justified, Indent: Left: 1,27 cm

Formatted: Caption, Indent: Left: 0 cm

Formatted: Font: 5 pt

Formatted: Indent: Left: 0 cm, First line: 1,25 cm

Formatted: Caption, Indent: Left: 0 cm

Formatted: Font: 5 pt

**Tabel 1-5. Tipul decimal**

Tipul de dată	Biți	Domeniul
<b>decimal</b>	128	-79228162514264337593543950335 la 79228162514264337593543950335

Formatted: Font: 5 pt

## 4. Tipuri bool:

Poate reține valori de adevăr "true sau false" și nu poate fi convertită.

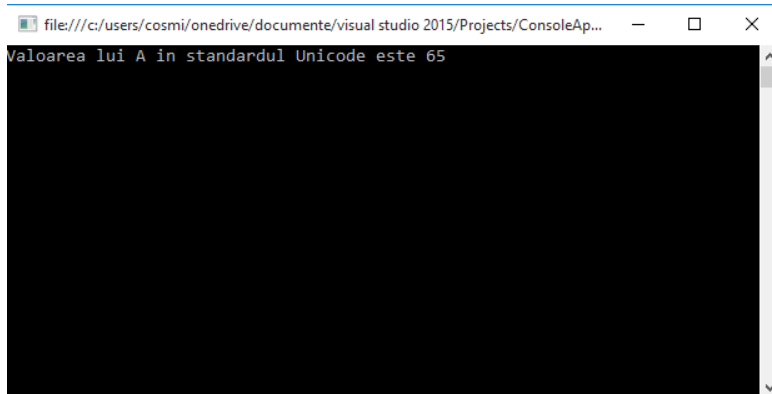
## 5. Tipul caracter:

În C# pentru caractere se utilizează tabela Unicode.

ex.

```
using System;
namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            //Declaraarea unei variabile de tip caracter
            //si atribuirea caracterului A
            const char caracter = 'A';
            //Afișarea valorii in standard UNICODE pentru variabila
            caracter
            //prima data are loc conversia la int
            Console.WriteLine("Valoarea lui {0} in standardul Unicode este
            {1}", caracter, (int)caracter);
            Console.ReadKey();
        }
    }
}
```

Acest mic program va afișa:



Figură 1-8

-Figura 1-11. Afișare mesaj

## 6. Tipul Structură –tip valoare propriu

Acest tip este asemănător cu cel din C++. Ea este stocată pe stivă, nu în heap și nu pune probleme de gestionare, atâta timp cât are o dimensiune rezonabilă.

Poate conține:

- declarații de constante
- câmpuri
- metode
- proprietăți
- indexatori
- operatori supraîncărcați
- constructori

ex.

```
struct Nod
{
    public readonly int Value;
    public Nod Stânga;
    public Nod Dreapta;

    public Nod(int initial)
    {
        Value = initial;
        Stânga = null;
        Dreapta = null;
    }
};
```

Privind cele de mai sus putem spune că structurile sunt identice cu clasele, deosebirea fiind locul de memorie unde se alocă acestea, stiva sau heap.

## 7. Tipul Enumerare:

Formatted Table

Formatted: Centered

Formatted: Left, Indent: First line: 0 cm

Formatted: Caption, Indent: Left: 0 cm

Formatted: Indent: Left: 0 cm, First line: 1,25 cm

Formatted: Indent: Left: 0 cm, First line: 1,25 cm

Este definit de programator și este asemănător cu cel din C++. Permite folosirea unor nume cărora li se atribuie o valoare. Ca bună practică este recomandată folosirea enumerației direct în Namespace, pentru a putea fi folosită de toate clasele.

Sintaxa enumerației este:

[atribute] [modificatori acces] enum NumeEnumeratie : [tip]

ex.

```
using System;
namespace ConsoleApplication1
{
    public enum Săptămâna : byte
    {
        Luni = 0,
        Marți,
        Miercuri,
        Joi,
        Vineri,
        Sâmbătă,
        Duminică
    }
    class Program
    {
        static void Main(string[] args)
        {

        }
    }
}
```

Tipul enumerării este implicit `int` și este opțional. Întotdeauna valoarea primului membru din enumerare este 0 iar tipurile `enum` sunt derivate din clasa `System.Enum` care și ea la rândul ei este derivată din `System.ValueType`.

#### 8. Tipuri nullable:

Sunt acele tipuri care pot reține atât valori din domeniul lor cât și valori `null`. La declarare valoarea unei variabile conține valoarea implicită a tipului (ex. `int` are valoarea lui `0` sau `0.0`), dacă nu se inițializează de programator. Tipurile nullable se folosesc atunci când se dorește ca valoarea implicită a variabilei să nu fie definită.

ex.

```
using System;
namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            //Prima metoda de declarare
            Nullable<int> a = null;

            //A doua metoda de declarare
            int? b = null;
        }
    }
}
```

Formatted Table

Formatted: Justified, Indent: Left: 0 cm

Formatted: Indent: First line: 2,24 cm

Formatted: Justified

```

//Verificare

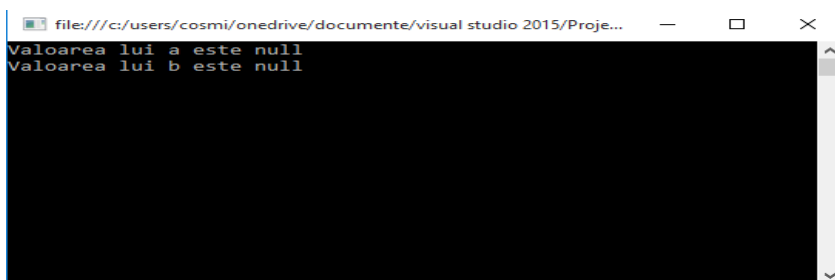
if (a == null)
{
    Console.WriteLine("Valoarea lui a este {0}", "null");
}

if (b == null)
{
    Console.WriteLine("Valoarea lui b este {0}", "null");
}

Console.ReadKey();
    }
}

```

Programul de mai sus va afișa:



```

file:///c:/users/cosmi/onedrive/documente/visual studio 2015/Proje...
Valoarea lui a este null
Valoarea lui b este null

```

**Figura 1-12. Afișare mesaj**

**Figură 1-9**

### 1.3.4.2 ASEMĂNĂRI ȘI DEOSEBIRI ÎNTRE C++ ȘI C#

C++ este un limbaj de programare cu capacități atât „high level” cât și „low level”, adică permite programare atât OOP cât și lucrul cu adresele de memorie pointeri etc, de aceea el fiind privit ca un limbaj de programare „mid level”.

C# este un limbaj modern, simplu, orientat pe obiect, type safe (nu permite declarații care în C++ generează erori de tipul „Acces Violation”).

1. În .NET este implementat, ca și în Java de altfel, o unealtă numită „Garbage Collector” care permite gestionarea automată a memoriei, fără intervenția programatorului pe când în C++ este necesară dealocarea memoriei de către programator folosind operatorii delete sau delete[] direct sau prin folosirea destructorilor.

Destructorii există și în C# dar nu au aceeași eficiență ca cei din C++, ei doar chemând GC, care oricum nu se execută în acel moment.

2. C# nu are tipuri de date primitive ca C++, tipurile de date fiind structuri. De exemplu tipul `int` din C# este de fapt un alias al tipului `Int32` din Framework, care este o structură ce se alocă pe stivă. Ca o dovadă se poate vedea că tipul `int` are metode proprii precum și metode moștenite din clasa `Object`, clasa de bază a tuturor claselor din C#.

Formatted: Justified

Implementarea tipului `Int32`:

```
(Microsoft Corporation)
using System.Globalization;
using System.Runtime;
using System.Runtime.InteropServices;
using System.Security;

namespace System
{
    [ComVisible(true)]
    public struct Int32 : IComparable, IFormattable, IConvertible,
        IComparable<Int32>, IEquatable<Int32>
    {
        public const Int32 MaxValue = 2147483647;
        public const Int32 MinValue = -2147483648;
        public static Int32 Parse(string s);
        [TargetedPatchingOptOut("Performance critical to inline across NGen image boundaries")]
        public static Int32 Parse(string s, IFormatProvider provider);
        public static Int32 Parse(string s, NumberStyles style);
        [TargetedPatchingOptOut("Performance critical to inline across NGen image boundaries")]
        public static Int32 Parse(string s, NumberStyles style, IFormatProvider provider);
        public static bool TryParse(string s, out Int32 result);

        [TargetedPatchingOptOut("Performance critical to inline across NGen image boundaries")]
        public static bool TryParse(string s, NumberStyles style, IFormatProvider provider, out Int32 result);
        public Int32 CompareTo(object value);
        [TargetedPatchingOptOut("Performance critical to inline across NGen image boundaries")]
        public Int32 CompareTo(Int32 value);
        [TargetedPatchingOptOut("Performance critical to inline across NGen image boundaries")]
        public override bool Equals(object obj);
        [TargetedPatchingOptOut("Performance critical to inline across NGen image boundaries")]
        public bool Equals(Int32 obj);
        [TargetedPatchingOptOut("Performance critical to inline across NGen image boundaries")]
        public override Int32 GetHashCode();
        [TargetedPatchingOptOut("Performance critical to inline across NGen image boundaries")]
        public TypeCode GetTypeCode();
        [SecuritySafeCritical]
        [TargetedPatchingOptOut("Performance critical to inline across NGen image boundaries")]
    }
}
```

```

    public override string ToString();
    [SecuritySafeCritical]
    [TargetedPatchingOptOut("Performance critical to inline across NGen image boundaries")]
    public string ToString(IFormatProvider provider);
    [SecuritySafeCritical]
    [TargetedPatchingOptOut("Performance critical to inline across NGen image boundaries")]
    public string ToString(string format);
    [SecuritySafeCritical]
    public string ToString(string format, IFormatProvider provider);
}

```

3. În C++ un tablou (array) este doar un pointer pe când în C# tablourile sunt obiecte care includ metode și proprietăți. Totodată există și diferențe de sintaxă:
  - în C++ `int a[]`
  - în C# `int[] a`
4. În C# pointerii sunt permisi, dar doar în modul „unsafe”. Poate fi declarată o singură metodă ca unsafe sau tot proiectul.
5. În C# metoda `main()` din C++ este declarată astfel `Main()` și întotdeauna este de tip static. Totodată diferă și argumentele din linia de comandă.
6. În C# toate variabilele și metodele trebuie să fie conținute de o clasă sau structură iar variabilele globale existente în C++ nu există în C#.
7. C# suportă operatori noi cum sunt `is`, `as`, `typeof`, iar supraîncărcarea operatorilor se face diferit.
8. Pe lângă instrucțiunile `if/else`, `while`, `for`, care sunt la fel ca în C++, la instrucțiunea `switch` trebuie să se pună obligatoriu `break` după fiecare caz. Totodată C# mai conține și instrucțiunea `foreach` pentru parcurgerea colecțiilor instrucțiune care a apărut mai nou și în C++ 11.

ex.

```

using System;
namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            int[] array = {0, 1, 2, 3, 4, 5};
            foreach (var x in array) //se creează de fiecare dată un obiect virtual
            {
                Console.WriteLine(x.ToString());
            }

            Console.ReadKey();
        }
    }
}

```

Această instrucțiune creează de fiecare dată un obiect nou de tipul celui din colecție, astfel parcurgerea unei colecții fiind mai lentă decât o instrucțiune `for` clasică, dar are avantajul unui cod mai clar.

Formatted: Justified

Formatted: Justified

Formatted: Justified

Formatted: Justified

Formatted: Justified

Formatted: Justified

Formatted: Justified

9. În C++ tipul bool este în esență un întreg pe când în C# bool este un tip de sine stătător și nu poate fi convertit într-un alt tip.
10. În C# poate fi folosit tipul object pentru a reține orice tip de dată. Procedeul poartă denumirea de boxing respectiv unboxing.

Formatted: Justified

ex.

```
using System;
namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            //boxing
            object o1 = "un text";
            object o2 = 12.3F;
            object[] tab = new object[] { "un text", 100, null, DateTime.Now };

            //unboxing
            string s = (string) o1;
            float f = (float) 02;
        }
    }
}
```

11. În C#, la try-catch s-a mai adăugat blocul finally pentru cod sursă care trebuie executat indiferent dacă o excepție este aruncată sau nu. Este foarte util spre exemplu la lucru cu baze de date (conexiunile se închid întotdeauna în finally) sau stream-uri.
12. În C# se pot folosi proprietățile sau auto proprietățile, care sunt o formă mai elegantă pentru metodele de tipul get, set.

ex.

```
namespace ConsoleApplication1
{
    public class Class1
    {
        private int m_Value ;

        public int Valoare { get; set; } //Auto proprietate

        public int Value //proprietate
        {
            get { return m_Value; }
            set { m_Value = value; }
        }

        public int GetValue() //clasic getter
        {
            return m_Value;
        }

        public void SetValue(int value) //clasic setter
        {
            this.m_Value = value;
        }
    }
}
```



```

    }

```

13. În C++ clasele și structurile sunt practic identice în timp ce în C# ele sunt destul de diferite. În C# clasele pot implementa orice număr de interfețe, dar nu pot face moștenire multiplă (pot moșteni de la o singură clasă de bază). În C# structurile nu acceptă moștenire, și nu acceptă constructori expliți (unul este prevăzut în mod implicit). De asemenea, în C# nu se poate deriva decât cu public. În C#, într-o clasă derivată, se poate utiliza cuvântul rezervat *base* pentru a apela membrii suprascriși ai clasei de bază. Suprascrerea metodelor virtuale sau abstracte se face în mod explicit folosind cuvântul rezervat *override*.

Formatted: Justified

Formatted: Font: Italic

Formatted: Font: Italic

14. În C# compilarea condiționată se realizează prin directivele pentru procesor. Nu se utilizează fișiere antet.

Formatted: Justified

15. Dacă în C++ *typedef* este folosit pentru a crea denumiri mai scurte sau mai convenabile pentru tipurile care au fost deja declarate, în C# se va utiliza directiva *using*.

Formatted: Justified

Formatted: Font: Italic

Formatted: Font: Italic

16. În C++ directiva *#define* utilizată pentru a declara valori constante. În C# această directivă nu poate fi utilizată în acest scop. În C# constantele sunt definite ca enumerări sau ca membri statici ai unei clase sau structuri.

Formatted: Justified

Formatted: Font: Italic

17. În C# locul template-urilor este luat de tipurile și metodele generice.

Formatted: Justified

Exemplu de clasă generică:

```

namespace ConsoleApplication1
{
    public class Stack<T>
    {
        private T[] array;
        private int index;
        public int Length { get { return index; } }
        public Stack() { index = 0; array = new T[5]; }
        public void Push(T value )
        {
            if (index > array.Length - 1)
            {
                T[] temp = new T[array.Length*2];
                for (int i = 0; i < array.Length; i++)
                    temp[i] = array[i];
                array = temp;
            }
            array[index] = value;
            index++;
        }
        public T Pop()
        {
            T temp;
            if (index > 0)
            {
                temp = array[index];
                index--;
            }
            else
                temp = array[index];
            return temp;
        }
    }
}

```

|

|

}

}

Formatted Table

## 2 PARTEA PRACTICĂ A LUCRĂRII

### 2.1 COMPONENTELE APLICAȚIEI „CONTAPFA”

#### 2.1.1 ARHITECTURA BAZEI DE DATE

Aplicația ContaPFA a fost gândită în jurul unei baze de date SQL locale, în care să reținem toate intrările într-un mod sigur și eficient. S-a ales o bază de date SQL locală deoarece este oferită gratuit de către Microsoft, volumul de date nu este unul așa de mare încât aplicația să necesite o conexiune către un server de baze de date și nu este nici necesar un control al concurenței oferit de server deoarece aplicația nu este destinată lucrului mai multor utilizatori deodată. Dacă totuși în viitor va fi necesară o astfel de abordare, migrarea către o bază de date de tip concurențial (-folosirea unui server MSSQL) fiind foarte ușoară, prin simpla modificare a string-ului de conexiune.

Tehnologia folosită pentru crearea și gestionarea bazei de date este **Entity Framework 6** –code first, ceea ce permite generarea automată a bazei de date dacă aceasta nu există în locația indicată în fișierul de configurare (xml).

Baza de date folosită în aplicație numită **ContaDb** este compusă din 9 tabele structurate astfel:

- **Tabela Acceptată Refuzată:**

Stochează date despre statusul lucrării Acceptată respectiv Refuzată de către persoana fizică autorizată. S-a optat pentru această soluție pentru o standardizare a datelor aduse în baza de date dar și pentru a economisii spațiu, tabela fiind legată de tabela Lucrări printr-o relație de tipul one to many.

	Column Name	Data Type	Allow Nulls
▶	AcceptataRefuzatald	int	<input type="checkbox"/>
	StatusAccept	nvarchar(10)	<input type="checkbox"/>

**Figura 2-1. Tabela Acceptată / Refuzată**

Stochează date despre statusul lucrării Acceptată respectiv Refuzată de către persoana fizică autorizată. S-a optat pentru această soluție pentru o standardizare a datelor aduse în baza de date dar și pentru a economisii spațiu, tabela fiind legată de tabela Lucrări printr-o relație de tipul one to many.

- **Tabela Beneficiar:**

Stochează datele de identificare a beneficiarilor lucrărilor. Este legată de tabela Contract printr-o relație de tipul one to many. Pot exista unul sau mai multe contracte care au același beneficiar astfel se economisește spațiu în baza de date.

	Column Name	Data Type	Allow Nulls
🔑	BeneficiarlId	int	<input type="checkbox"/>
	Nume	nvarchar(150)	<input type="checkbox"/>
	CNP	nvarchar(14)	<input type="checkbox"/>
	TipActId	int	<input checked="" type="checkbox"/>
	Serie	nvarchar(3)	<input checked="" type="checkbox"/>
	Numar	nvarchar(6)	<input checked="" type="checkbox"/>
	Adresa	text	<input checked="" type="checkbox"/>
	AtributFiscal	nvarchar(3)	<input checked="" type="checkbox"/>
	NrRegComert	nvarchar(25)	<input checked="" type="checkbox"/>
	Telefon	nvarchar(14)	<input checked="" type="checkbox"/>
	AdresaEmail	nvarchar(50)	<input checked="" type="checkbox"/>
	PersoanaFizica	bit	<input type="checkbox"/>

**Figura 2-2. Tabela Beneficiar**

Stochează datele de identificare a beneficiarilor lucrărilor. Este legată de tabela Contract printr-o relație de tipul one to many. Pot exista unul sau mai multe contracte care au același beneficiar astfel se economisește spațiu în baza de date.

- **Tabela Contract:**

Stochează datele de identificare a contractelor și asociază fiecărui contract un beneficiar. Totodată Contract este și tabela de legătură între Lucrări , Beneficiari, Plăți și Încasări.

	Column Name	Data Type	Allow Nulls
🔑	ContractId	int	<input type="checkbox"/>
	NrContract	nvarchar(4)	<input type="checkbox"/>
	Data	date	<input type="checkbox"/>
	BeneficiarlId	int	<input checked="" type="checkbox"/>
	ObiectulContractului	text	<input type="checkbox"/>
	Suma	decimal(18, 0)	<input type="checkbox"/>
	Observatii	text	<input checked="" type="checkbox"/>

**Figura 2-3. Tabela Contract**

Stochează datele de identificare a contractelor și asociază fiecărui contract un beneficiar. Totodată Contract este și tabela de legătură între Lucrări , Beneficiari, Plăți și Încasări.

- **Tabela Încasări:**

Stochează date cu privire la operațiunea de încasare a unei sume de bani în urma unui serviciu prestat conform unui contract. Datele stocate aici împreună cu cele din tabela Plăți, sunt folosite pentru generea ulterioară a unor rapoarte către fisc cum ar fi: Declarația 200, Declarația 392/b, etc.

	Column Name	Data Type	Allow Nulls
?	IncasareId	int	<input type="checkbox"/>
	Data	datetime	<input type="checkbox"/>
	TipDocument	nvarchar(16)	<input type="checkbox"/>
	Serie	nvarchar(6)	<input type="checkbox"/>
	[Numar Document]	nvarchar(MAX)	<input type="checkbox"/>
	Suma	decimal(18, 2)	<input type="checkbox"/>
	TransferBancar	bit	<input type="checkbox"/>
	[Plateste T.V.A.]	bit	<input type="checkbox"/>
	[Valoare T.V.A.]	decimal(18, 2)	<input type="checkbox"/>
	[Suma T.V.A.]	decimal(18, 2)	<input type="checkbox"/>
	ContractId	int	<input checked="" type="checkbox"/>
	[Felul Operatiunii]	nvarchar(255)	<input type="checkbox"/>

**Figura 2-4. Tabela Încasări**

Stochează date cu privire la operațiunea de încasare a unei sume de bani în urma unui serviciu prestat conform unui contract. Datele stocate aici împreună cu cele din tabela Plăți sunt folosite pentru generea ulterioară a unor rapoarte către fisc cum ar fi: Declarația 200, Declarația 392/b, etc.

#### • Tabela Plati

Stochează date despre plățile efectuate de către persoana fizică autorizată având aceleași câmpuri ca și tabela Încasări și aceeași folosință.

	Column Name	Data Type	Allow Nulls
?	PlataId	int	<input type="checkbox"/>
	Data	datetime	<input type="checkbox"/>
	TipDocument	nvarchar(16)	<input type="checkbox"/>
	Serie	nvarchar(6)	<input type="checkbox"/>
	[Numar Document]	nvarchar(MAX)	<input type="checkbox"/>
	Suma	decimal(18, 2)	<input type="checkbox"/>
	TransferBancar	bit	<input type="checkbox"/>
	[Plateste T.V.A.]	bit	<input type="checkbox"/>
	[Valoare T.V.A.]	decimal(18, 2)	<input type="checkbox"/>
	[Suma T.V.A.]	decimal(18, 2)	<input type="checkbox"/>
	ContractId	int	<input checked="" type="checkbox"/>
	[Felul Operatiunii]	nvarchar(255)	<input type="checkbox"/>

**Figura 2-5. Tabela Plăți**

~~Stochează date despre plățile efectuate de către persoana fizică autorizată având aceleași câmpuri ca și tabela Încasări și aceeași folosință.~~

- **Tabela Lucrare:**

Stochează date și informații despre lucrările efectuate sau în desfășurarea ale persoanei fizice autorizate în domeniul lucrărilor de cadastru și topografie conform cerințelor A.N.C.P.I (Agenția Națională de Cadastru și Publicitate Imobiliară). Aceasta este legată printr-o relație de tipul one to one de tabela Contract, iar de tabela AcceptatăRespinsă, tabela ReceptionatRespins și de tabela TipLucrare printr- relație de tipul one to many.

Formatted Table

Formatted: Centered

Formatted: Caption

Formatted: Font: Bold

Formatted: Justified

	Column Name	Data Type	Allow Null
🔑	LucrareId	int	<input type="checkbox"/>
	AcceptataRefuzataId	int	<input type="checkbox"/>
	[Nr.OCPI]	nvarchar(10)	<input checked="" type="checkbox"/>
	DataInregistrare	date	<input checked="" type="checkbox"/>
	TermenSolutionare	date	<input checked="" type="checkbox"/>
	AvizatorRegistrator	text	<input checked="" type="checkbox"/>
	TipLucrareId	int	<input checked="" type="checkbox"/>
	[Nr.Proiect]	nvarchar(6)	<input type="checkbox"/>
	AnProiect	nvarchar(4)	<input type="checkbox"/>
	ContractId	int	<input checked="" type="checkbox"/>
	[Cad/Top]	text	<input type="checkbox"/>
	UAT	nvarchar(100)	<input type="checkbox"/>
	Observatii	text	<input checked="" type="checkbox"/>
	ReceptionatRespinsId	int	<input type="checkbox"/>

**Figura 2-6. Tabela Lucrare**

Stochează date și informații despre lucrările efectuate sau în desfășurarea ale persoanei fizice autorizate în domeniul lucrărilor de cadastru și topografie conform cerințelor A.N.C.P.I (Agenția Națională de Cadastru și Publicitate Imobiliară). Aceasta este legată printr-o relație de tipul one to one de tabela Contract, iar de tabela AcceptataRefuzata, tabela ReceptionatRespins și de tabela TipLucrare printr-relație de tipul one to many.

#### • Tabela ReceptionatRespins

Stochează date despre statusul lucrării Recepționată respectiv Respinsă de către O.C.P.I. (Oficiul de Cadastru și Publicitate Imobiliară). S-a optat pentru această soluție pentru o standardizare a datelor aduse în baza de date dar și pentru a economisii spațiu, tabela fiind legată de tabela Lucrări printr-o relație de tipul one to many.

	Column Name	Data Type	Allow Nulls
🔑	ReceptionatRespinsId	int	<input type="checkbox"/>
	StatusRec	nvarchar(11)	<input type="checkbox"/>

**Figura 2-7. Tabela ReceptionatRespins**

Stochează date despre statusul lucrării Recepționată respectiv Respinsă de către O.C.P.I. (Oficiul de Cadastru și Publicitate Imobiliară). S-a optat pentru această

Formatted Table

Formatted: Centered

Formatted: Caption, Indent: Left: 0 cm

Formatted: Font: Bold

Formatted: Justified

Formatted: Centered

Formatted: Caption, Indent: Left: 0 cm

soluție pentru o standardizare a datelor aduse în baza de date dar și pentru a economisii spațiu, tabela fiind legată de tabela Lucrări printr-o relație de tipul one to many.

- **Tabela TipAct:**

Stochează tipul de act de identificare cu care s-a legitimat beneficiarul unei lucrări și în baza căruia s-a semnat un contract.

	Column Name	Data Type	Allow Nulls
🔑	TipActId	int	<input type="checkbox"/>
	TipAct	nvarchar(22)	<input type="checkbox"/>

**Figura 2-8. Tabela TipAct**

Stochează tipul de act de identificare cu care s-a legitimat beneficiarul unei lucrări și în baza căruia s-a semnat un contract.

- **Tabela TipLucrare:**

Stochează datele cu privire la tipul lucrărilor și codul acestora, lucrări aflate în nomenclatorul A.N.C.P.I. Tabela este legată de tabela Lucrări printr-o relație de tipul one to many.

	Column Name	Data Type	Allow Nulls
🔑	TipLucrareId	int	<input type="checkbox"/>
	CodLucrare	nvarchar(5)	<input type="checkbox"/>
	TipLucrare	nvarchar(125)	<input type="checkbox"/>

**Figura 2-9. Tabela TipLucrare**

Stochează datele cu privire la tipul lucrărilor și codul acestora, lucrări aflate în nomenclatorul A.N.C.P.I. Tabela este legată de tabela Lucrări printr-o relație de tipul one to many.

Normalizarea datelor s-a făcut avându-se în vedere o obținere a unui raport viteză de răspuns/spațiu ocupat pe disc optim, precum și evitarea introducerii de date redundante sau greșit formate de către utilizator. De exemplu numele unei lucrări poate fi scris de către același utilizator sau utilizatori diferiți în feluri diferite, făcând mai dificile interogările bazei de date, dar și o ocupare de spațiu pe disc mai mare, putând să existe în felul acesta mai multe înregistrări pentru aceeași lucrare.

Relația între tabele se poate observa și din diagrama de mai jos:

Formatted Table

Formatted: Font: Bold

Formatted: Normal, Indent: Left: 1,75 cm, No bullets or numbering

Formatted: Normal, Justified, Indent: Left: 1,75 cm, No bullets or numbering

Formatted: Centered

Formatted: Caption, Indent: Left: 0 cm

Formatted: Font: Bold

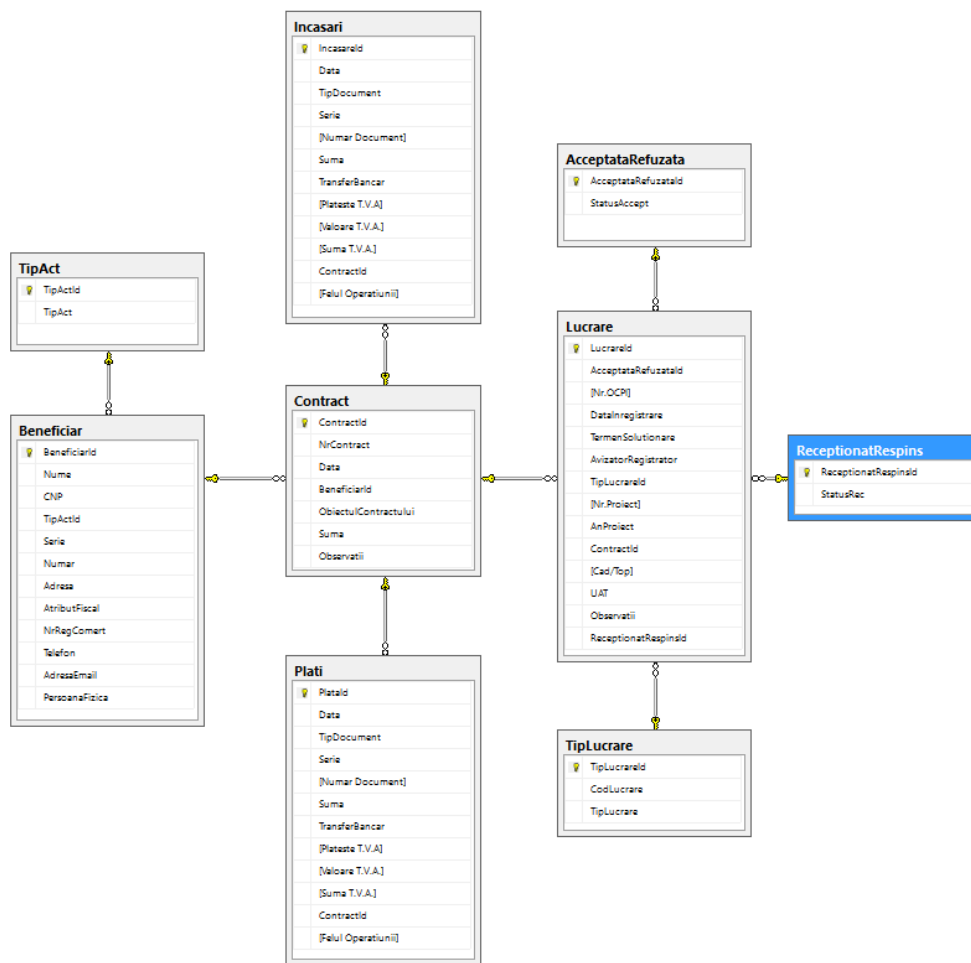
Formatted: Justified

Formatted: Centered

Formatted: Caption, Indent: Left: 0 cm

Formatted: Justified





**Figura 2-10. Relația de legătură între tabelele bazei de date ContaDb**

Pentru crearea bazei de date descrisă mai sus s-a folosit un Framework dezvoltat de Microsoft, numit **Entity Framework**. Entity Framework este un Object Relational Mapping (ORM) Framework care este o îmbunătățire a lui ADO.NET și pune la dispoziția programatorului un mecanism automat de accesare a datelor dintr-o bază de date.

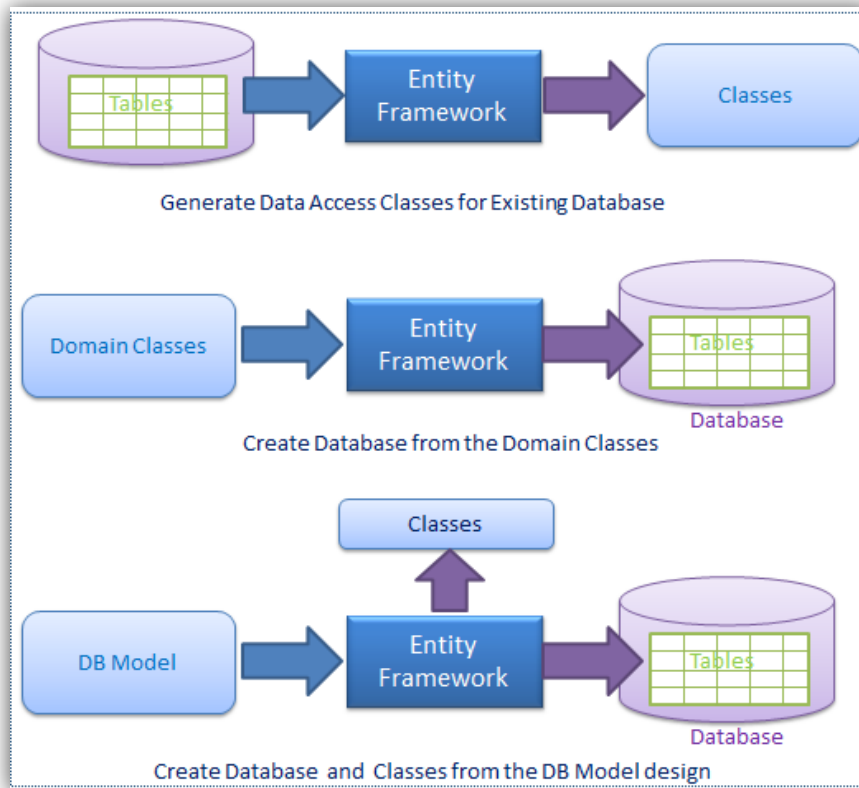
Entity Framework se poate folosi în trei scenarii:

1. Generare de cod (clase) după schema unei baze de date existente (database first).
2. Crearea bazei de date după crearea claselor modelului (code first).
3. Crearea schemei unei baze de date în modul visual designer și apoi crearea bazei de date și a claselor.

Formatted: Justified

Formatted: Font: Bold, Italic

Formatted: Justified



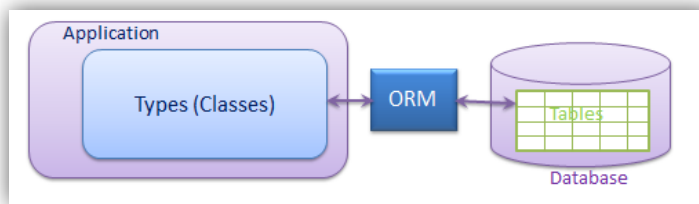
**Figura 2-2** **Figura 2-11. Entity Framework /** Sursă: <http://www.entityframeworktutorial.net/what-is-entityframework.aspx>

Dar totuși ce este ORM? Este o unealtă care permite salvarea obiectelor într-o bază de date de tip relațional cum ar fi MSSQL într-un mod automat fără scriere de prea mult cod.

ORM permite separarea design-ului bazei de date de domain class design. Acest lucru face ca aplicația să se întrețină ușor și să se extindă la nevoie.

De asemenea conține un standard automat C.R.U.D, (Create, Read, Update, Delete) deci programatorul nu trebuie să le mai scrie manual.

Mai multe informații despre arhitectura, precum și modul de lucru cu Entity Framework se pot găsi la: <http://www.entityframeworktutorial.net/EntityFramework5/entity-framework5-introduction.aspx>

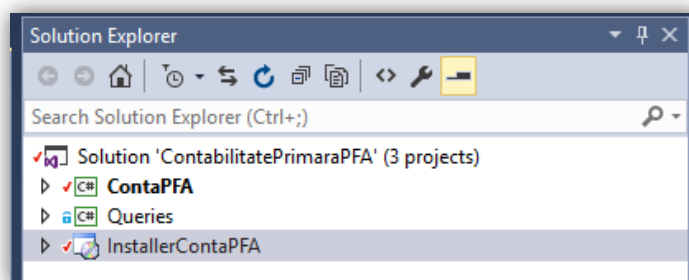


**Figura 2-3** **Figura 2-12. ORM** / Sursă: <http://www.entityframeworktutorial.net/what-is-entityframework.aspx>

## 2.1.2 ARHITECTURA APLICAȚIEI CONTAPFA

Soluția aplicației „ContaPFA” este compusă din trei proiecte și anume:

- ContaPFA care gestionează interfața grafică (forms și user controls) precum și logica aplicației (business logic).
- Queries gestionează comunicarea dintre business logic-ul aplicației și baza de date.
- InstallerContaPFA așa cum îi spune și numele se ocupă de generarea installer-ului aplicației.



**Figura 2-4** **Figura 2-13. Soluția aplicației ContaPFA**

**1. Proiectul ContaPFA** conține patru form-uri: ContaPfaForm, BeneficiariForm, FilterForm

1. FilterForm și SelectTipLucrări, precum și trei user controls: ucContracte, ucLucrări, ucOperațiuni. Ca design pattern pentru comportamentul user controls și forms s-a utilizat Singleton pentru a preveni instanțierea multiplă a unei clase. Durata de viață a obiectului ContaPfaForm și a obiectelor de tip user control enumerate mai sus este pe toată durata de execuție a programului.

**Formatted Table**

**Formatted:** Centered, Don't keep with next

**Formatted:** Justified

**Formatted:** Centered, Don't keep with next

**Formatted:** Justified, Indent: First line: 1,27 cm

**Formatted:** Justified, Indent: Hanging: 0,02 cm, Numbered + Level: 1 + Numbering Style: 1, 2, 3, ... + Start at: 1 + Alignment: Left + Aligned at: 0,63 cm + Indent at: 1,27 cm

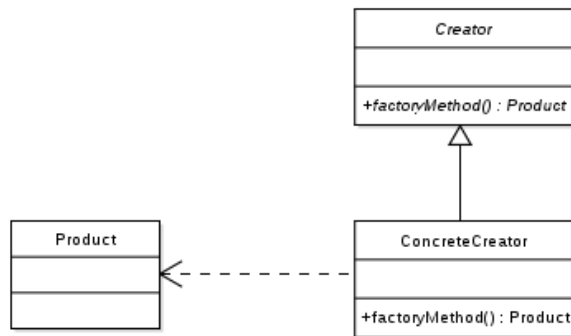
Form-ul ContaPfaForm este folosit ca un șablon pentru elementele de interfață grafică comune iar în urma unor evenimente generate de utilizator sau aplicație părțile dinamice (user controls) apar în form.

Ca design pattern pentru obținere instanțelor obiectelor de tip user control s-a folosit un pattern denumit Factory.

-Definiția acestui design pattern este conform link-ului de mai jos:

([http://cursuri.cs.pub.ro/~poo/wiki/index.php/Design\\_Patterns\\_Basics](http://cursuri.cs.pub.ro/~poo/wiki/index.php/Design_Patterns_Basics))

„Patternul Factory face parte din categoria *Creational Patterns* și ca atare rezolvă problema creării unui obiect fără a specifica exact clasa obiectului ce urmează a fi creat. Acest lucru este implementat prin definirea unei metode al cărei scop este crearea obiectelor. Metoda va avea specificat ca parametru de întors în antet un obiect de tip părinte, urmând ca, în funcție de alegerea programatorului, aceasta să creeze și să întoarcă obiecte noi de tip copil.”



Sursa: [http://cursuri.cs.pub.ro/~poo/wiki/index.php/Design\\_Patterns\\_Basics](http://cursuri.cs.pub.ro/~poo/wiki/index.php/Design_Patterns_Basics)

**Figură 2-5** Figura 2-14. Diagrame UML cu implementare design pattern-ului Factory / Sursa: [http://cursuri.cs.pub.ro/~poo/wiki/index.php/Design\\_Patterns\\_Basics](http://cursuri.cs.pub.ro/~poo/wiki/index.php/Design_Patterns_Basics)

Pentru o mai bună înțelegere voi arăta și o scurtă implementare a acestui pattern în C#

```

using System.Collections.Generic;
using System.Windows.Forms;
using View.View.UserControls;

namespace View.View.Classes
{
    public class UIFactory
    {
        private static Dictionary<string, UserControl> _mDUserControl = new
        Dictionary<string, UserControl>();

        public static UserControl GetUserControl(string uiName)
        {
            if (_mDUserControl.Count != 0) return _mDUserControl[uiName];
            _mDUserControl.Add("Lucrari", UcLucrari.GetUiLucrari);
            _mDUserControl.Add("Contracte", UcContracte.GetUiContracte);
            _mDUserControl.Add("Operatiuni", ucOperatiuni.GetUiOperatiuni);
            return _mDUserControl[uiName];
        }
    }
}
  
```

După cum se observă clasa UIFactory are o metodă statică numită GetUserControl care returnează un obiect de tip UserControl în funcție de un nume. Se mai poate observa că

obiectele de tipul UserControl se instanțiază doar în momentul apelării metodei și nu la pornirea aplicației (lazy initialization).

Posibilitatea de a aduce într-un dicționar a trei tipuri de obiecte diferite este dată de faptul că toate moștenesc aceeași clasă de bază și anume UserControl urmând ca clientul să facă cast către tipul dorit.

- exemplu de implementare în clasa client:

```
private void PaintUserControl(object sender, EventArgs e)
{
    mainPanel.Controls.Clear();

    switch(sender.ToString())
    {
        case "Lucrari":
            _userControl = UIFactory.GetUserControl(sender.ToString());
            break;

        case "Contracte":
            _userControl = UIFactory.GetUserControl(sender.ToString());
            break;
        default:
            {
                var currentUserControl = _userControl.Name;
                currentUserControl = currentUserControl.Remove(0, 2);
                var button = sender as Button;
                if (button != null && button.Text != currentUserControl)
                {
                    _userControl = UIFactory.GetUserControl(((Button)
                        sender).Text);
                }
            }
            break;
    }

    if (!mainPanel.Controls.Contains(_userControl))
    {
        if (_userControl == null) return;
        mainPanel.Controls.Add(_userControl);
        _userControl.Dock = DockStyle.Fill;
        _userControl.BringToFront();
    }
    else
    {
        _userControl.BringToFront();
    }
}
```

Metoda PaintUserControl este de fapt un event handler care prinde mesajele trimise de obiectele de tip buton la evenimentul „click”. În funcție de butonul care a declanșat evenimentul, metoda *PaintUserControl* cere un obiect după un nume clasei UIFactory și îl face curent în form-ul principal.

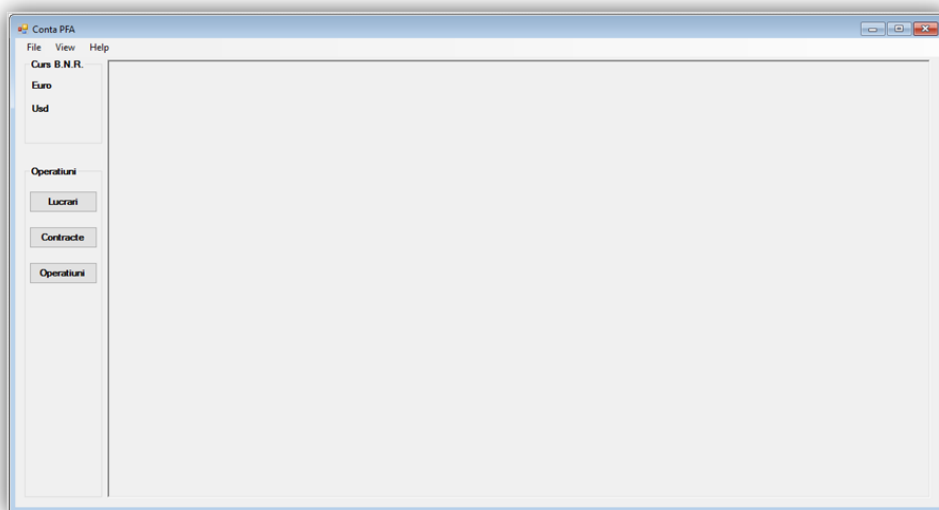
Clasa **ContaPfaForm** mai conține pe lângă constructor și o metodă GetBNR care cheamă o clasă „CursBNR” din care se obține cursul BNR al zilei curente și îl afișează în form.

Formatted: Justified

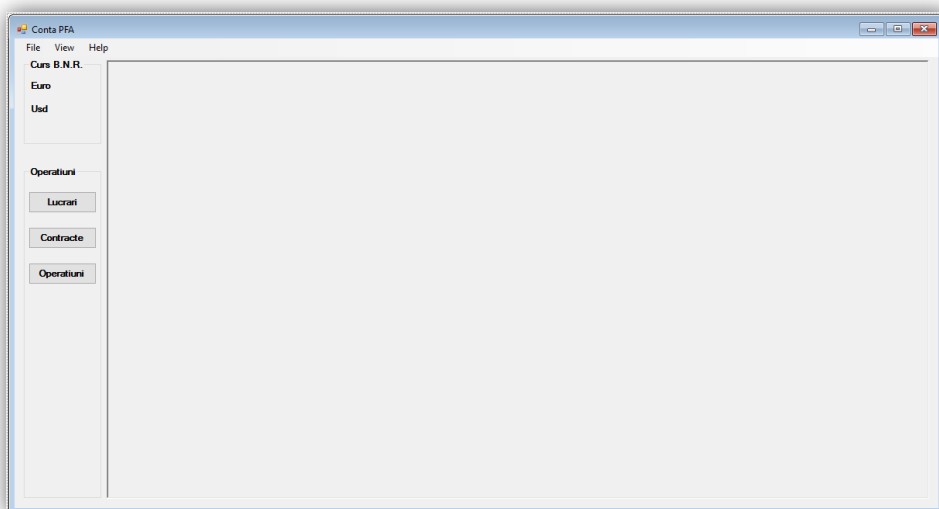
Formatted: Font: Italic

Formatted: Justified

Formatted: Font: Bold



**Figura 2-15. Form-ul principal ContaPfaForm fără nici un UserControl instantiat**



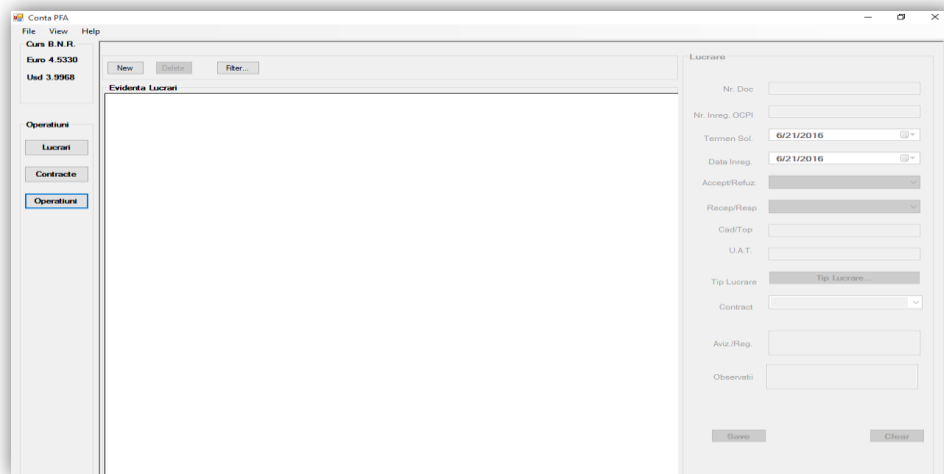
**Figură 2-6 Form-ul principal ContaPfaForm fără nici un UserControl instanțiat**

Formatted Table

Formatted: Indent: First line: 0 cm

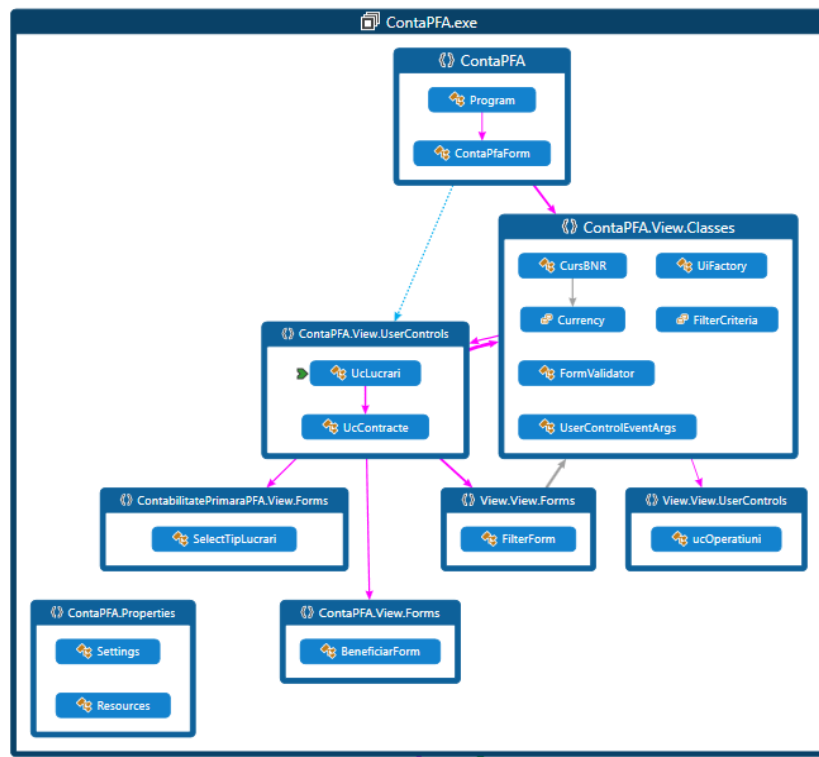
Formatted: Indent: First line: 0 cm

Formatted: Don't keep with next

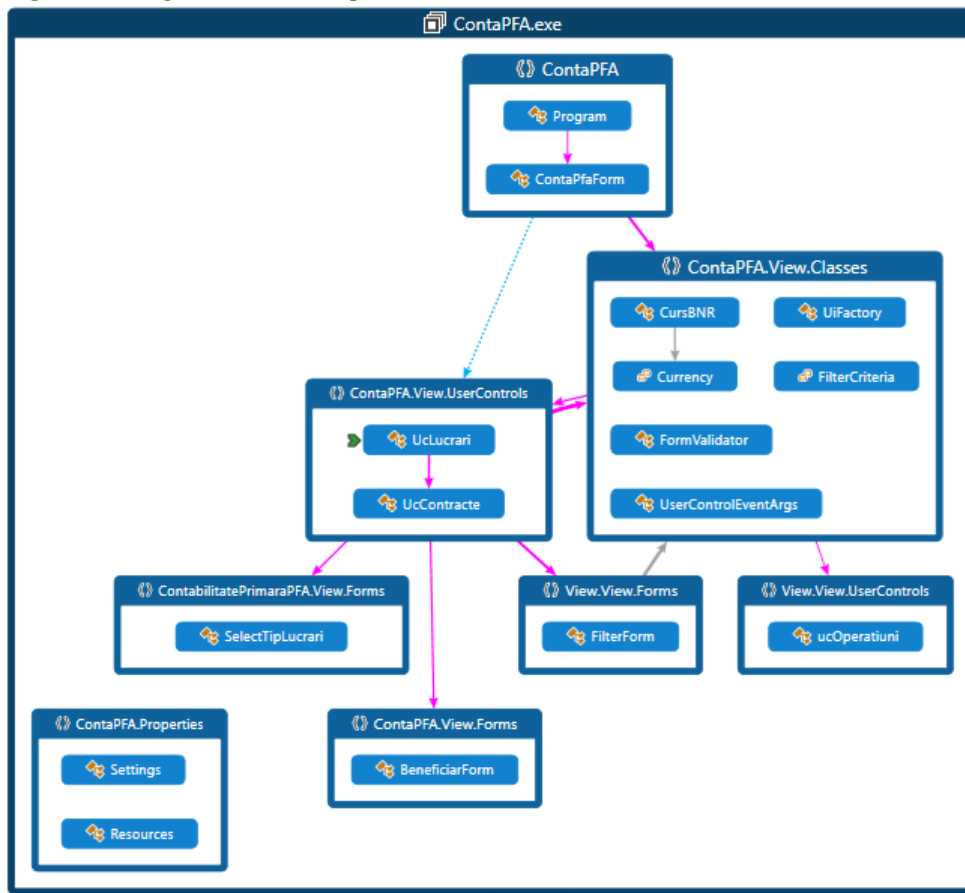


**Figura 2-7-Figura 2-16.** Form-ul principal ContaPfaForm cu UserControl ucLucrari instanțiat

User Controls ucLucrari, ucContracte și ucOperatiuni oferă controale și metode prin intermediul cărora utilizatorul comunică cu baza de date, executând diferite operațiuni asupra acestora. Tot în această zonă există și metode de validare a datelor introduse de utilizator în scopul evitării introducerii unor date invalide pentru anumite câmpuri din baza de date.



~~Figură 2.8 Diagrama claselor in proiectul ContaPFA~~



**Figură 2-8-Figura 2-17. Diagrama claselor în proiectul ContaPFA**



2. **Proiectul Queries** se ocupă cu operațiile cu baza de date și se bazează pe tehnologia Entity Framework care este un ORM. ORM-ul presupune o tehnică de programare prin accesarea și manipularea obiectelor fără ca programatorii să fie interesați de sursa de date de unde provin aceste obiecte. Cu această tehnologie procesul de stocare a datelor într-o baza de date relațională se face automat folosind un Framework ORM (Entity Framework) și constă în maparea obiectelor la tabele corespunzătoare, asocierea dintre acestea fiind descrisă folosind metadate.

Pornind de la cele descrise mai sus s-au creat în proiect clase care descriu tabelele din baza de date precum și relațiile dintre tabele.

- exemplu clasa pentru tabela contracte:

```
namespace Queries.Core.Domain
{
    using System;
    using System.Collections.Generic;
    using System.ComponentModel.DataAnnotations;
    using System.ComponentModel.DataAnnotations.Schema;
    [Table("Contract")]
    public partial class Contract
    {
        public Contract()
        {
            Plata = new HashSet<Plata>();
            Incasare = new HashSet<Incasare>();
            Lucrare = new HashSet<Lucrare>();
        }

        public int ContractId { get; set; }

        [Required]
        [StringLength(4)]
        public string NrContract { get; set; }

        [Column(TypeName = "date")]
        public DateTime Data { get; set; }

        public int? BeneficiarId { get; set; }

        [Column(TypeName = "text")]
        [Required]
        public string ObiectulContractului { get; set; }

        public decimal Suma { get; set; }

        [Column(TypeName = "text")]
        public string Observatii { get; set; }
    }
}
```

Formatted: Justified

Formatted: Justified, Numbered +  
Level: 1 + Numbering Style: 1, 2, 3, ...  
+ Start at: 1 + Alignment: Left +  
Aligned at: 0,63 cm + Indent at: 1,27  
cm

Formatted: Justified

```

public virtual Beneficiar Beneficiar { get; set; }

public virtual ICollection<Plata> Plata { get; set; }
public virtual ICollection<Incasare> Incasare { get; set; }
public virtual ICollection<Lucrare> Lucrare { get; set; }
}

```

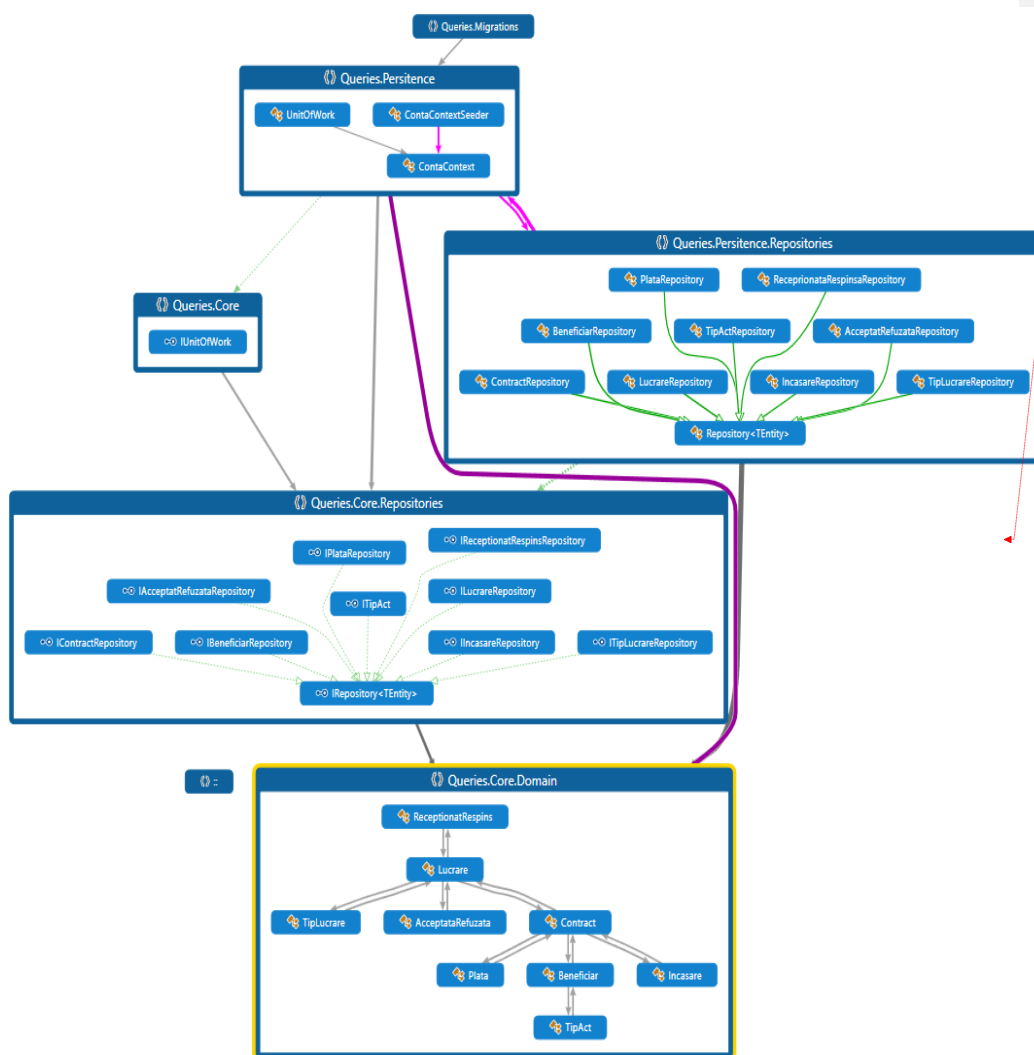
Atribuția proiectului Queries este aceea de a aduce din, respectiv de a salva în, baza de date a unor obiecte sau liste de obiecte care servesc proiectul principal ContaPFA. Design pattern-ul ales pentru acest scop se numește „**Repository**”.

Acest design pattern separă „data access logic” de „business logic” astfel făcând aplicația mai ușor de întreținut, extins și totodată se evită scrierea codului duplicat care poate într-un final duce și la erori de programare. Comunicarea dintre ”business logic” și ”data access logic” se face printr-o interfață, astfel folosind acest design pattern partea de business logic nu are cunoștință cum lucrează partea de data access logic, (folosește LINQ to SQL , sau ADO.NET sau Entity Framework ) ea doar așteaptă prin intermediul interfeței date.

Formatted Table

Formatted: Justified

Formatted: Font: Bold



**Figura 2-9** **Figura 2-18.** Diagrama claselor din proiectul Queries

## 2.2 FUNCȚIONAREA APLICAȚIEI

Aplicația permite înregistrarea editarea și ștergerea în sau din baza de date a evidențelor contabile ale unui PFA autorizat în lucrări de topografie și cadastru precum și cea a lucrărilor executate de acesta. Aplicația este împărțită în patru zone:

- Afișare curs BNR.

- Zonă de comandă.
- Afișare date înregistrate în baza de date.
- Adăugare editare date.

**Figură 2-10-Figura 2-19.** Interfața principală a aplicației

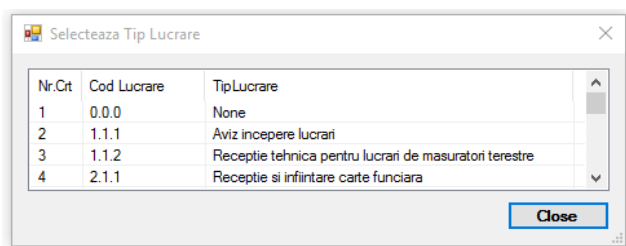
Din zona de operațiuni se poate alege tipul de înregistrare dorit lucrare, contract sau operațiuni financiare (plată - încasare).

Formatted: Justified

În cazul lucrărilor, conform figurii 2-10 butonul new activează zona lucrare care permite adăugarea unei noi lucrări și salvarea acesteia. Tipurile de informații cum sunt Acceptată Refuzată, Recepționată Respinsă și Tip Lucrare sunt predefinite în baza de date utilizatorul trebuind doar să selecteze informația corespunzătoare.

Contractul aferent lucrării se va selecta din baza de date dacă acesta există deja iar dacă nu se va crea unul nou pentru această nouă lucrare folosind new.

Pentru selectarea tipului de lucrare se va deschide un nou form care conține toate lucrările și codurile acestora conform nomenclatorului de lucrări A.N.C.P.I.



**Figura 2- 20. Fereastră pentru selectare tipului de lucrare**

~~Figura 2- 11 Fereastră pentru selectare tipului de lucrare~~

În momentul salvării unei lucrări aceasta este automat adusă în zona de afișare date din baza de date.

Nr.Crt	Cod Lucrare	Tip Lucrare
3	1.1.2	Receptie tehnica pentru lucrari de masuratori terestre
4	2.1.1	Receptie si infiintare carte funciara
5	2.1.2	Receptie (nr. cadastral)
6	2.1.3	Infiintare carte funciara

**Figura 2- 21. Figură 2-12 Crearea și salvarea unei noi lucrări**

Pentru editarea unei lucrări se va da dublu click pe lucrarea dorită aceasta urmând a fi încărcată în zona lucrare pentru a fi editate datele. ~~Vezi fig.2-13~~

Aceptata/Refuzata	NrInregOCPI	TermenSol	AvizatorRegistrator	TipLucrare	NrProiect	Beneficiari
Acceptata	3698-6/17/2016	6/24/2016	Boruga Viorel	Aviz incepere lucrari	01/2016	Taran Constantin

**Lucrare**

Nr. Doc: 01

Nr. Inreg. OCPI: 3698

Termen Sol: 6/24/2016

Data Inreg: 6/17/2016

Accept/Refuz: Acceptata

Recep/Resp: In Lucru

Cad/Top: 1236/2/a

U.A.T.: Timisoara

Tip Lucrare: Tip Lucrare...

Contract: 0101

Aviz./Reg: Boruga Viorel

Observatii: nu sunt

Edit Cancel

**Figura 2- 22. Figură 2-13** Editarea unei lucrări existente în baza de date

Filtrarea datelor afișate se face cu butonul **Filter**, buton ce deschide o nouă fereastră din care se va selecta un criteriu de filtrare precum și în unele cazuri un cuvânt cheie. Dacă lista de lucrări este filtrată v-a apărea în dreptul butonului Filter un avertisment „Filter On”.

Exemplu filtrare lucrări după anul întocmirii documentației:

New Delete Filter... Filter On

**Evidenta Lucrari**

Aceptata/Refuzata	NrInregOCPI	TermenSol	AvizatorRegistrator	TipLucrare	NrProiect	Beneficiari
Acceptata	3698-6/17/2016	6/24/2016	Boruga Viorel	Aviz incepere lucrari	01/2016	Taran Constantin

**Cautare Lucrari**

2016 Cauta

Renunta

Nr. Ct Criteriu Cautare

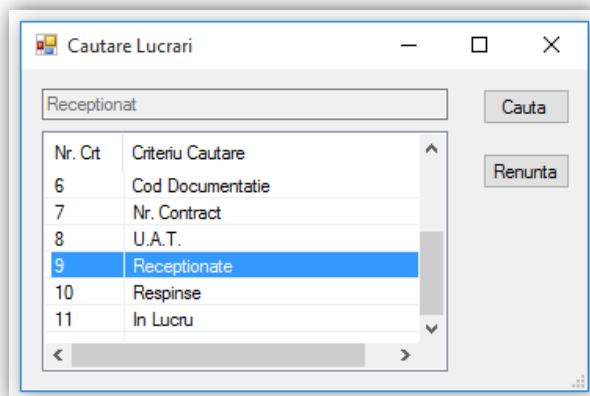
- 1 None
- 2 An Documentatie
- 3 Nr. Documentatie
- 4 Nume Beneficiar
- 5 C.N.P. Beneficiar
- 6 Cod Documentatie

**Figura 2- 23. Figură 2-14** Filtrarea afișării înregistrărilor din baza de date

Formatted: Justified

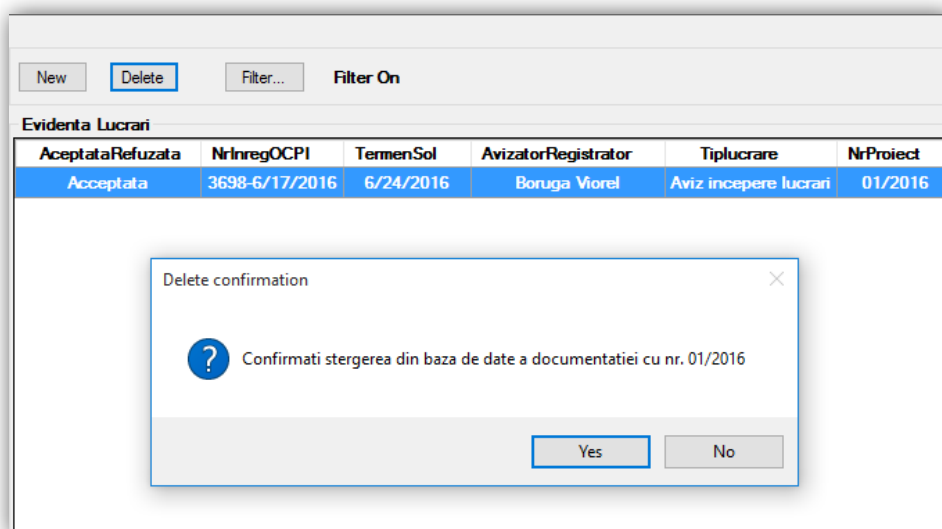
Formatted: Font: Bold

După filtrare se poate observa că există o singură înregistrare în baza de date pe anul 2016. Totodată lucrările pot fi filtrate și după statusul acestora: în lucru, recepționată, respinsă, caz în care se va elimina necesitatea unui cuvânt cheie pentru filtrare iar câmpul destinat intrării acestuia va fi dezactivat.



**Figura 2- 24. Filtrare după statusul lucrării**

Pentru ștergerea unei înregistrări din baza de date se selectează lucrarea butonul Delete devine activ și se poate șterge selecția.



**Figura 2- 25. Figură 2-16 Ștergere unei înregistrări din baza de date**

Afișarea cursului BNR se face automat la deschidere prin descărcarea de pe internet a unui fișier .xml ce conține cursul valutar al acelei zile. În cazul lipsei unei conexiuni de internet, dacă fișierul există, aplicația v-a afișa ultimul curs valutar pe care îl are în fișier, iar dacă acest fișier nu există, se va afișa o avertizare la pornire, iar câmpurile Euro și Usd vor rămâne goale.

Principiul de funcționare descris mai sus pentru lucrări se moștenește și la contracte.

Formatted Table

Formatted: Centered

Formatted: Justified

Formatted: Justified



NrContract	Data	Titular	Suma
01	6/10/2016	Taran Constantin	100

**Figura 2- 26. Figură 2-17** Fereastra contracte

Adăugarea unui nou beneficiar al lucrării se face din fereastra de contracte prin butonul „Beneficiar”.

—Astfel se deschide o nouă fereastră ce permite introducerea în baza de date a unui nou beneficiar sau selectarea unuia deja existent. ~~Vezi figura 2-18~~

Formatted: Justified

**Figura 2- 27. Figură 2-18** Fereastra beneficiar

—Aplicația permite filtrarea beneficiarilor înscrși în baza de date pe măsura ce se completează numele acestuia (AutoComplete).

Formatted: Justified

## Concluzii

Aplicația ContaPFA oferă persoanelor fizice autorizate în domeniul lucrărilor de topografie și cadastru o modalitate ușoară de a gestiona și vizualiza lucrările executate sau în curs de executare, contractele și beneficiarii acestora. În acest fel se menține o evidență acestora într-un mod unitar și ușor de gestionat.

Ca îmbunătățiri ulterioare se are în vedere

- finalizarea părții de gestionare a plăților și încasărilor.
- posibilitatea de a ține aceste evidențe pentru mai multe persoane fizice autorizate.
- adăugarea unui modul propriu de emitere facturi și chitanțe
- adăugarea unui modul pentru generare de rapoarte în format pdf, doc și excel
- adăugarea unui modul pentru importul în baza de date a conținutului unui fișier excel.

De menționat că această aplicație așa cum este ea momentan nu poate fi funcțională în producție, lipsindu-i modulele menționate mai sus precum și un limbaj specific de contabilitate – topografie și cadastru, precum și o testare corespunzătoare dat fiind faptul că aplicația procesează date cu caracter confidențial.

## Bibliografie

- [1]. <https://msdn.microsoft.com>
- [2]. Programarea Orientată pe Obiecte și Programarea Vizuală Microsoft
- [3]. Introduction to C#, The New Language for Microsoft.net, H.Mossenbock, University of Linz, Austria.
- [4]. Teoria Normalizării, Despi I., Petrov G., Reisz R., Stepan A. – Teoria generala a bazelor de date, Ed. Mirton, Timișoara, 2000.

### 3—Cuprins

<i>Facultatea de Calculatoare și Informatică Aplicată</i> .....	4
<i>Departamentul de Informatică</i> .....	4
<b>1—PARTEA TEORETICĂ A LUCRĂRII</b> .....	<b>5</b>
1.1—INTRODUCERE.....	5
1.2—NOȚIUNI TEORETICE ȘI PRACTICE DESPRE CONTABILITATEA PRIMARĂ ÎN PARTIDĂ SIMPLĂ.....	6
1.2.1—NOȚIUNI TEORETICE DESPRE CONTABILITATE.....	6
1.2.2—NOȚIUNI TEORETICE DESPRE CONTABILITATEA PRIMARĂ PFA.....	10
1.2.3—NOȚIUNI PRACTICE ÎN CONTABILITATEA PRIMARĂ PFA.....	11
1.3—PREZENTAREA MEDIULUI DE DEZVOLTARE A APLICAȚIEI.....	14
1.3.1—SCURTĂ PREZENTARE A PLATFORMEI .NET.....	14
1.3.2—COMPILAREA PROGRAMELOR ÎN ARHITECTURA .NET.....	15
1.3.3—SCURTĂ PREZENTARE MEDIULUI DE PROGRAMARE MICROSOFT VISUAL STUDIO.....	16
1.3.4—LIMBAJUL C#.....	19

<b>2</b>	<b>PARTEA PRACTICĂ A LUCRĂRII</b>	<b>28</b>
2.1	COMPONENTELE APLICAȚIEI „CONTAPFA”	28
2.1.1	ARHITECTURA BAZEI DE DATE	28
2.1.2	ARHITECTURA APLICAȚIEI CONTAPFA	34
2.2	FUNCȚIONAREA APLICAȚIEI	40
2.3	Concluzii	45
<b>3</b>	<b>Bibliografie</b>	<b>46</b>