

Contents

1. Date generale.....	3
2. Fereastra principala « FrmMainWindow »	3
2.1. Design.....	3
2.2. Dinamica ecranului.....	6
3. Fereastra secundara « FrmSearchWindow »	10
3.1. Design.....	10
3.2. Dinamica ecranului.....	10
4. Indicatii tehnice.....	11

Versiune	Data	Autor
1.0	30.11.2015	I VLAD Creare document

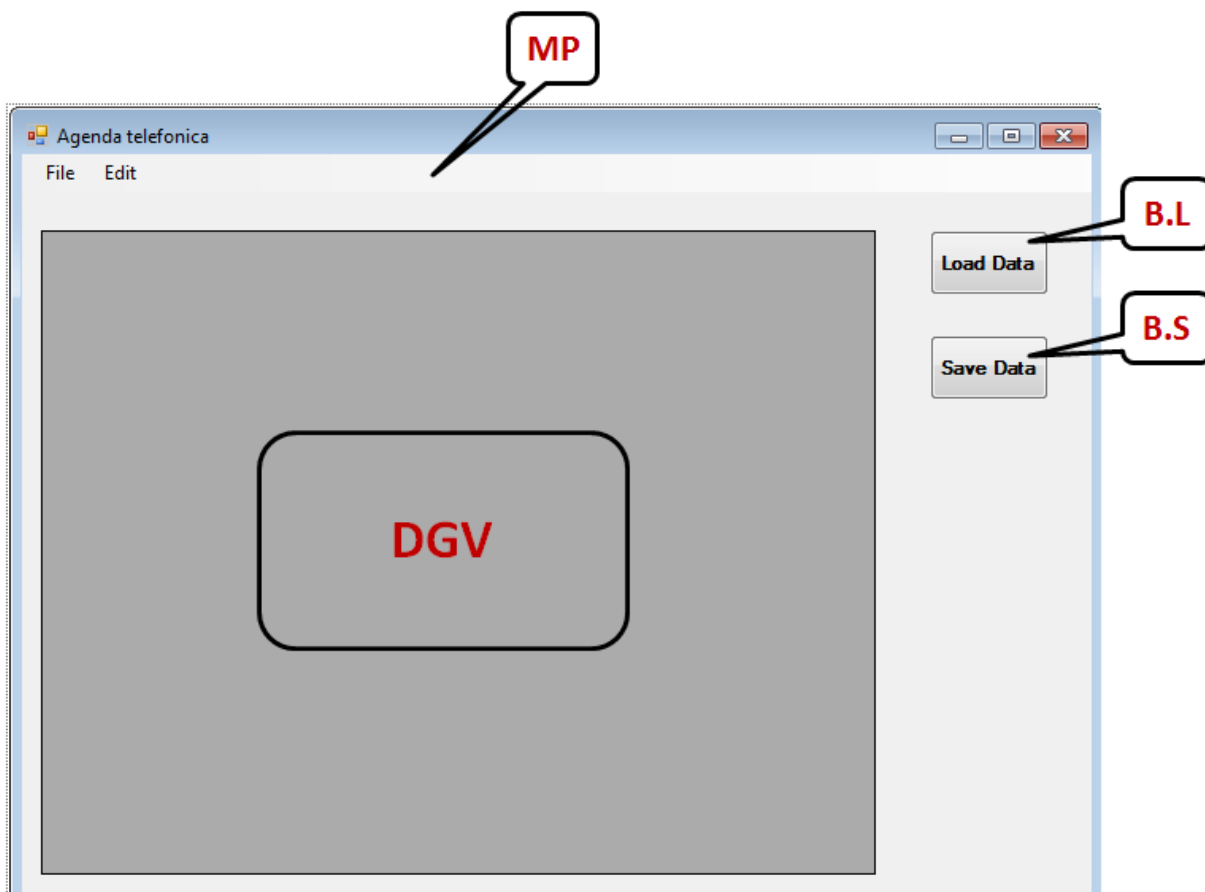
1. Date generale

Sa se creeze o aplicatie "**Agenda telefonica**" cu urmatoarele caracteristici:

- Va fi o aplicatie Windows Desktop Application
- Aplicatia (solutia din VS) va contine doua ferestre : una principala si una secundara;
- Fereastra principala se va numi « **FrmMainWindow** », iar a doua se va numi « **FrmSearchWindow** »
- « **FrmMainWindow** » este cea care se va afisa atunci cand aplicatia este rulata ;
(cele doua ferestre pot fi create in acelasi proiect Visual Studio, sau in proiecte diferite; oricare solutie este acceptata);

2. Fereastra principala « **FrmMainWindow** »

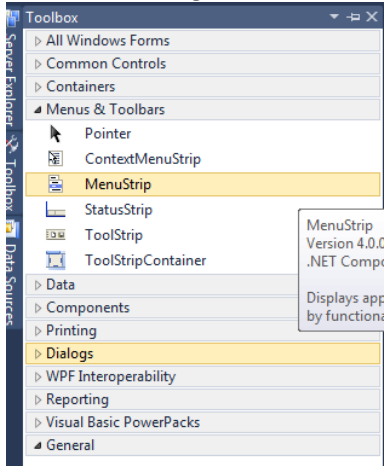

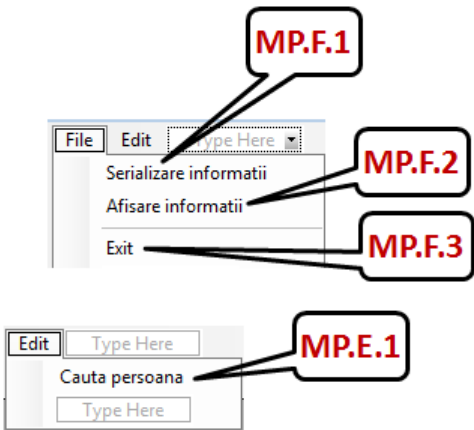
2.1. Design





Semnificatia campurilor :

(in toate explicatiile de mai jos, **VS** = **Microsoft Visual Studio**)

Camp	Tip Obiect VS	Descriere
MP	MenuStrip	<p>Obiectul este un <u>meniul principal</u>. Pentru a-l adauga in fereastra la design time, se alege din Toolbox:</p>  <p>O data adaugat ferestrei, meniul MP este adaugat si ca un control separat in fereastra de design, adica :</p>  <p>Meniul va contine:</p> <ul style="list-style-type: none"> - doua optiuni principale : File (MP.F) si Edit (MP.E); - fiecare optiune va avea sub-meniurile de mai jos: 

MP.F.1	ToolStripMenuItem	Reprezinta un meniu care contine prima optiune a meniului « File ». Textul optiunii va fi « Serializare informatii ». Ceea ce se intampla cand utilizatorul va alege aceasta optiune din meniu este descris in capitolul « 2.2 Dinamica ecranului »
MP.F.2	ToolStripMenuItem	Reprezinta un meniu care contine a doua optiune a meniului « File ». Textul optiunii va fi « Afisare informatii ». Ceea ce se intampla cand utilizatorul va alege aceasta optiune din meniu este descris in capitolul « 2.2 Dinamica ecranului »
MP.F.3	ToolStripMenuItem	Reprezinta un meniu care contine a doua optiune a meniului « File ». Textul optiunii va fi « Exit ». Ceea ce se intampla cand utilizatorul va alege aceasta optiune din meniu este descris in capitolul « 2.2 Dinamica ecranului »
MP.E.1	ToolStripMenuItem	Reprezinta un meniu care contine prima optiune a meniului « Edit ». Textul optiunii va fi « Cauta persoana ». Ceea ce se intampla cand utilizatorul va alege aceasta optiune din meniu este descris in capitolul « 2.2 Dinamica ecranului »
B.L	Button	Reprezinta butonul care va incarca din baza de date informatii si le va afisa in datagridview-ul DGV . Acest buton este tot timpul activ : B.L.Enabled = True . Ceea ce se intampla cand utilizatorul da clic pe acest buton este descris in capitolul « 2.2 Dinamica ecranului ».
B.S	Button	Reprezinta butonul care va salva ceea ce se afla in DGV in baza de date. Acest buton este tot timpul activ : B.S.Enabled = True . Ceea ce se intampla cand utilizatorul da clic pe acest buton este descris in capitolul « 2.2 Dinamica ecranului ».
DGV	DataGridView	Acest control va afisa, in ecran, infomatiile din tabela « Abonati » din baza de date. Modul in care este incarcat acest control este descris in capitolul « 2.2 Dinamica ecranului ».
SV	SaveFileDialog	Acest control va permite salvarea inregistrarilor din agenda, adica serializarea agendei, intr-o locatie aleasa de utilizator, intr-un fisier ce va avea un nume predefinit. A se vedea capitolul « 2.2 Dinamica ecranului ».
OPN	OpenFileDialog	Acest control va permite deschiderea inregistrarilor din agenda, asa cum au fost ele salvate la un moment dat din trecut, dupa ce informatia din Agenda a fost serializata. A se vedea capitolul « 2.2 Dinamica ecranului ».



OBS :

- in GUI, nu sunt admise erori de tip exceptie netratate ; codul care va trata operatiuni in baza de date, scrierea in fisiere (serializare) etc va fi pus in **try{...} catch()** ;
- fiecare obiect (meniu, buton etc) din ferestre va avea un nume semnificativ ; de exemplu butonul **B.L** poate fi numit **btnLoadData**, butonul **B.S** poate fi numit **btnSaveData** etc ; aceste nume vor fi practic variabilele utilizate in cod cu aceste nume ;

2.2. Dinamica ecranului

În tabelul de mai jos sunt descrise acțiunile pe care le poate face utilizatorul și modul în care ecranul reacționează pentru fiecare acțiune în parte :

Acțiune	Comportament ecran
Rularea aplicației	<p>Atunci când aplicația este rulată, fereastra « FrmMainWindow » este afișată.</p> <p>La afișarea ferestrei pe ecran (evenimentul FrmMainWindow_Load) se vor face :</p> <ul style="list-style-type: none">- construirea coloanelor controlului DGV (a se vedea exemplul atasat din portal) ; controlul DGV va permite ca utilizatorul să poată adăuga direct linii în el (a se vedea exemplul atasat din portal);- interogare în baza de date și se extrag toate înregistrările din tabela Agenda. Dacă se găsește cel puțin o înregistrare, atunci controlul DGV este populat cu linii. Dacă nu se găsește nicio înregistrare în tabela, DGV rămâne fără linii (doar cu coloane).
Clic pe butonul B.L	<p>Când utilizatorul dă clic pe acest buton, se va face o interogare în baza de date, tabela Abonați.</p> <p>Dacă se găsesc înregistrări în această tabelă, controlul DGV este populat cu liniile găsite în tabelă ;</p> <p>Dacă nu se găsește nicio înregistrare în această tabelă, controlul DGV este afișat fără linii și se afișează un mesaj utilizatorului « Niciun abonat găsit în bază ». Mesajul este afișat cu MessageBox.Show().</p>
Clic pe butonul B.S	<p>Acest buton este folosit pentru a salva în baza de date, tabela Abonați, modificările făcute în controlul DGV.</p> <p>Atenție : în controlul DGV, userul poate să : adăuge înregistrări, modifice înregistrări, ștergă linii. Deci, în baza de date, se pot face :</p> <ul style="list-style-type: none">- INSERT-uri- UPDATE-uri- DELETE-uri <p>Dacă operația este făcută cu succes în baza de date, atunci mesajul « Date salvate cu succes în baza de date » este afișat cu MessageBox.Show().</p> <p>În caz de eroare, un mesaj corespunzător este afișat utilizatorului, cu aceeași funcție MessageBox.Show().</p>
DGV	<p>Acest control DataGridView este construit în maniera următoare :</p> <ul style="list-style-type: none">- Coloanele îi sunt construite o singură dată, de preferat în funcția care tratează evenimentul FrmMainWindow_Load ;- De fiecare dată când este nevoie să se afișeze liniile acestui control cu informații din baza de date (la încărcarea ferestrei, la clic pe butonul B.L etc), înainte de a face aceasta, linii eventuale existente în control se vor șterge : DGV.Rows.Clear() ; <p>Fiecare celulă din acest control este editabilă, adică utilizatorul poate da dublu-clic într-o celulă și poate modifica conținutul celulei.</p> <p>De asemenea, utilizatorul poate adăuga linii acestui control, și la fel poate șterge o linie întreagă din control.</p>

	Column1	Column2
	fdg	hgfh
	ghgf	gfhjf
	jghj	ighj
		

Coloanele acestui control pot fi adaugate fie la run-time (adica cod scris pentru asta), fie direct din designer – ambele variante sunt acceptate.

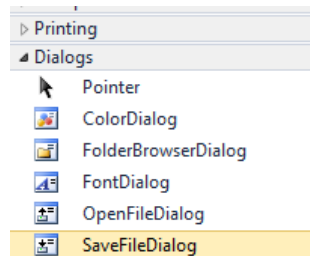
MP.F.1

Aceast meniu serializarea datelor din **DGV** intr-un fisier pe disc, acolo unde userul va alege sa se salveze (folderul va fi ales de user, de preferat acolo unde este executabilul aplicatiei). Serualizarea va fi de tip **XML**.

Aceasta optiune are legatura cu controlul **SV** din poza de mai sus.

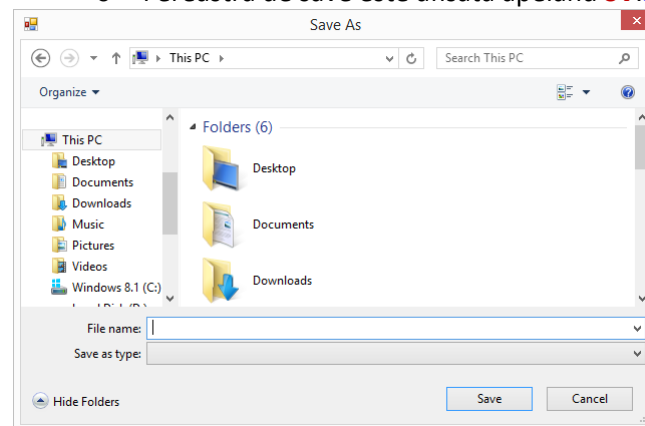
Cand userul alege aceasta optiune din meniu, atunci :

- ✓ Daca nicio linie nu este in **DGV**, atunci se afiseaza un mesaj de eroare « Nimic de serializat », cu `MessageBox.Show()`. Userul inchide mesajul si nimic nu se intampla.
- ✓ Daca exista cel putin o linie in **DGV**, atunci :
 - La alegerea optiunii, o noua fereastra se va deschide ; fereastra noua este de tipul « SaveFileDialog », control clasic din VS ; acest control se gaseste in Toolbox :



Aceasta fereastra permite salvarea fisierului in care se va face searilizarea obiectelor, undeva pe disc, acolo unde alege utilizatorul.

- Fereastra de save este afisata apeland `SV.ShowDialog();`, iar rezultatul este :



Acestei ferestre i se pot transmite diversi parametri, ca de exemplu numele fisierului in care se vor salva datele ;

	<ul style="list-style-type: none"> ○ Numele fisierului in care se vor serializa datele va fi « Abonati_ 'timestamp'.xml », unde 'timestamp' este de forma 'YYYYMMDDHHMISS', adica : <ul style="list-style-type: none"> - YYYY = anul curent in format de 4 cifre ; - MM = luna curenta, 2 cifre ; - DD = ziua curenta, 2 cifre ; - HH = ora curenta, in format 24h ; - MI = minutul orei curente, 2 cifre ; - SS = secunda orei curente, 2 cifre ; <p>Logica este ca in acest fisier se salveaza starea curenta a agendei, stare asa cum exista atunci cand userul alege optiunea MP.F.1 (momentul in care da clic pe meniu). Pentru ca userul poate sa salveze continutul lui DGV in momente diferite, fiecare salvare va crea un fisier nou, iar ca fisierele sa poate fi diferite, solutia cea mai buna este ca numele acestor fisiere sa contina chair momentul in care userul a ales optiunea din meniu. Acest moment din timp se numeste <i>TIMESTAMP</i> si contine data curenta, cat mai detaliata, pana la nivel de secunda. Ca sa obtinem in C#.NET momentul curent, pana la nivel de secunda, se poate utiliza functia <code>DateTime.Now()</code> astfel : <code>DateTime.Now.ToString("yyyyMMddHHmmss")</code></p> <p>Rezultatul este un string care contine data curenta, exact in formatul descris mai sus, necesar pt a da numele noului fisier de serializare.</p> <p>Astfel, daca se fac mai multe salvari (serializari) ale tabelui, privind retrospectiv, doar din numele fisierului putem deduce foarte usor la ce moment de timp a fost creat, deci putem deduce continutul tabelui la un moment de timp din trecut.</p> <p>Prin urmare, la clic pe meniul MP.F.1 se va afisa aceasta fereastra de Save care va avea automat (deci initializata) :</p> <ul style="list-style-type: none"> - numele fisierului de serializare egal cu cel creat mai sus, in care intra <i>TIMESTAMP</i>-ul ; - folderul propus de salvare al acestui fisier va fi folderul in care se afla executabilul aplicatiei (a se vedea exemplul atasat din portal) <p>(a se vedea exemplul atasat din portal)</p>
MP.F.2	TO BE DEFINED
MP.F.3	<p>Aceasta optiune va inchide aplicatia, cu un mesaj de confirmare. Adica, la alegerea acestui meniu, se va afisa un mesaj catre user « Sunteti sigur ca doriti sa iesiti ? (Yes/No) » (tot cu o una dintre implementarile lui <code>MessageBox.Show()</code>)</p> <p>Daca userul alege No, mesajul este inchis si nimic nu se intampla.</p> <p>Daca userul alege Yes, aplicatia se inchide apeland <code>Application.Exit()</code>.</p>
SV	<p>Acest control windows este cel folosit in toate aplicatiile windows atunci cand se cere utilizatorului sa salveze un fisier.</p> <p>Acest control poate primi o serie de parametri de intrare, prin setarea unor proprietati, astfel incat sa corespunda cu cerintele proiectului :</p> <pre>saveFileDialog1.Title = "Salvare fisier serializare"; //titlul ferestrei Save saveFileDialog1.Filter = "XML files *.xml"; //filtrul pentru tipul de</pre>

fișier

```
saveFileDialog1.InitialDirectory = Application.StartupPath; //directoul  
initial propus de fereastra save; in exemplul nostrum este vorba de  
directoul fișierului exe al aplicației  
saveFileDialog1.DefaultExt = ".xml"; //extensia propusa implicit pt  
fișierul ce se va salva  
saveFileDialog1.FileName = "file"; //numele implicit al fișierului
```

(a se vedea exemplul atasat din portal)

Se va merge pe principiul , in cadrul aplicației noastre, ca utilizatorului i se va deschide aceasta fereastră direct cu tot ce are nevoie : nume fișier, director in care sa se salveze fișierul., astfel incat el, userul, practic nu va mai modifica nimic, ci pus si simplu va apasa butonul OK din fereastră.

Dupa ce userul alege directorul de salvare, deci dupa ce da OK, numele complet al fișierului (cale completa + nume) care va contine serializarea obiectelor din **DGV** este recuperat cu **saveFileDialog1.FileName**. Acesta va fi folosit mai departe ca si fișier de serializare (a se vedea exemplul din lectia 4 din portal).

3. Fereastra secundara « **FrmSearchWindow** »

TO BE DEFINED

3.1. Design

TO BE DEFINED


3.2. Dinamica ecranului

TO BE DEFINED

4. Indicatii tehnice

Ceea ce va utiliza aplicatia:

- Aplicatia va fi realizata folosind Visual Studio 2010 sau 2013 ;
- Proiectul va fi 3 tier, adica layer-e separate: GUI Layer, Business logic Layer si Data Access Layer ;
- Solutia din VS va fi astfel proiectata incat nu se vor permite apeluri de metode (functii) apartinand layer-ului DAL direct din GUI. Adica :
 - Din GUI nu se pot instantia obiecte si apela functii decat din GUI si din Business logic Layer ;
 - Din Business logic Layer nu se pot instantia obiecte si apela functii decat din Business layer din DAL ;
 - Din DAL nu se pot apela functii decat din DAL ;
- Serverul SQL va fi de preferat versiunea 2008 ;
- Baza de date se va numi « **Training** » ;
- Aplicatia se va conecta la baza de date folosind Windows Authentication ;
- Tabela din aceasta baza de date se va numi « **Agenda** »
- Structura tabelului va fi :

	Column Name	Data Type	Allow Nulls
	ID	int	<input type="checkbox"/>
	Nume	nchar(100)	<input type="checkbox"/>
	Prenume	nchar(100)	<input type="checkbox"/>
	Telefon	nchar(12)	<input type="checkbox"/>
			<input type="checkbox"/>

Campul ID este folosit ca si cheie primara a tabelului ;