**Universitatea de Stat din Moldova**

**fACULTATEA DE MaTEMATICĂ ȘI INFORMATICĂ**

**DEPARTAMENTUL DE INFORMATICĂ**

**DASCĂL ALEXANDRU**

**INTERACȚIUNEA CU BAZELE DE DATE ÎN CADRUL LIMBAJULUI C#**

**Teză de licenţă**

Șef Departament: **\_\_\_\_\_\_\_\_\_\_\_\_\_ Vsevolod Arnaut, dr., conf.**

(semnătura)

Conducător ştiinţific: **\_\_\_\_\_\_\_\_\_\_\_\_\_ Vsevolod Arnaut, dr., conf.**

(semnătura)

Autorul: \_\_\_\_\_\_\_\_\_\_\_\_\_ **Dascăl Alexandru**

(semnătura)

**Chişinău-2016**

CUPRINS

[LISTA ABREVIERILOR 3](#_Toc453190774)

[ADNOTARE 4](#_Toc453190775)

[ADNOTATION 5](#_Toc453190776)

[INTRODUCERE 6](#_Toc453190777)

[I. TEHNOLOGIILE .NET PENTRU BAZELE DE DATE 8](#_Toc453190778)

[1.1. Concepte fundamentale ale modelelor de stocare a datelor 8](#_Toc453190779)

[1.2. Tehnologia ADO.NET 12](#_Toc453190780)

[1.2.2. ADO.NET metoda conectată 17](#_Toc453190781)

[1.2.2. ADO.NET metoda deconectată 18](#_Toc453190782)

[1.3. Tehnologia Entity Framework 19](#_Toc453190783)

[II. DEZVOLTAREA UNEI APLICAȚII UTILIZÎND TEHNOLGIA ENTITY FRAMEWORK 22](#_Toc453190784)

[2.1. Proiectarea Aplicației 22](#_Toc453190785)

[2.1.1. Structura Organizației 22](#_Toc453190786)

[2.1.2. Cerințele față de sistemul informatic 24](#_Toc453190787)

[2.2. Realizarea Aplicației 25](#_Toc453190788)

[2.3. Funcționalitățile Aplicației 31](#_Toc453190789)

[CONCLUZII 35](#_Toc453190790)

[BIBLIOGRAFIE 37](#_Toc453190791)

[Anexe 38](#_Toc453190792)

# LISTA ABREVIERILOR

C# - C șarp.

ADO.NET – Obiecte de Date ActiveX (eng. ActiveX Data Objects).

IT – Tehnologii Informaționale.

UDA – Accesul Universal al Datelor (eng. Universal Data Acces).

ORM – Maparea relațională a obiectelor (eng. Object Relational Maping).

DRSA – Directie raionala pentru siguranta alimentelor.

ȘI – Sistem Informatic.

SQL – Limbaj Standard pentru Interogarea datelor (eng. Standard Query Language) .

LINQ – Limbaj pentru interogarea datelor în C# (eng. Language Integrated Query).

ASVF – Autorizație Sanitară Veterinară de Funcționare.

EF – Entity Framework.

# ADNOTARE

La teza de licență cu tema „Interacțiunea cu bazele de date în cadrul limbajului C#” a studentului grupei I31, Dascăl Alexandru.

*Structura*. Teza de licență este compuă din introducere, lista de abrevieri, două capitole, concluzii și bibliografie fiind perfectată pe 41 pagini A4, cuprinde 6 imagini, 8 tabele, 5 diagrame, 2 anexe și 11 surse bibliografice.

*Cuvinte cheie:* bază de date, date, informație, stocare a datelor, clasă, tabel, aplicație, sistem, tehnologie, limbaj, relație.

*Primul capitol* este format din două parți, în prima parte sunt enumerate și descrise avantajele bazei de date față de alte sisteme de stocare a datelor, partea a doua reprezintă introducerea în cîteva din tehnologiile existente în limbajul C# pentru interacțiunea cu bazele de date, tehnolgiile sunt descrise din punct de vedere teoretic și cu exemple practice, pe lîngă descrierea tehnologiilor sunt facute și comparații dintre acestea, evidențiind aspectele fiecăreia dintre ele, argumentînd avantajele și dezavantajele lor.

*Capitolul doi* reprezintă partea practică a lucrării. După cercetarea tehnologiilor, efectuată în primul capitol și determinarea celei mai eficiente tehnologii, a fost realizată o aplicție ce are ca scop rezolvarea unei probleme existente.

*Metodele de cercetare* utilizate în teză includ studieriea tehnologiilor din materiale video, materiale scrise precum articole și cărți, au fost consultate forumuri despre tehnologiile respective și experimentarea lor în exemple practice.

*Importanța teoretică* a lucrării constă în argumentarea beneficiilor utilizării bazelor de date și cercetarea aprofundată a tehnologiilor de utilizare a bazelor de date în limbajul C#.

*Valoarea aplicativă* constă în proiectarea corectă a aplicației și realizare ei în conformitate cu cerințele utilizatorilor folosind tehnologia aleasă în urma invesțigării realizate în capitolul teoretic.

# ADNOTATION

On the licence thesis „Interaction with databases using C# language” of student Dascal Alexandru, grupe I31.

*Structure*. This thesis consists of introduction, list of abbreviations, two chapters, conclusion and bibliography, being perfected on 41 A4 pages containing 6 images, 8 tables, 5 charts, 2 annexes and 11 bibliographical sources.

*Keywords*: database, data, information, data storage, class, table, application, system, technology, language, relationship.

*The first chapter* consists of two parts, in the first part are listed and described some advantages of using databases instead of other date storage systems, the second part is the introduction in the existing technologies of the C# language for interacting with databases, technologies are described theoretically and in practical examples, beside the description of technologies are highlighted some positive an negative aspects of each technolog.

*Chapter two* is the practical part of the licence thesis. After researching technologies in the first chapter and determining the most efficient technology, was developed an aplication that solved a real problem using that technology.

*Research methods* used in this thesis was the study of technologies from sources such as videos, written materials such as articles and books, were consulted forums to see different opinions and understanding the aspects of technologies from point of view of other people, also practicing technologies in real examples.

*The theoretical importance* of this thesis is to argue the benefits of using databases and to find the best database technology in C# language.

*Practical value* consists in developing the rigth application in accordance with user requirements, using the best technology chosen after investigation made in theoretical chapter.

# INTRODUCERE

Lucrarea dată este o analiză a metodelor de utilizare a bazelor de date în cadrul limbajului C#, evidențiind avantajele utilizării bazelor de date în favoarea altor posibilități de stocare a datelor în format electronic, evoluția metodelor din limbajul C# pentru interacțiunea cu bazele de date și argumentarea celei mai eficiente și des utilizate metode în elaborarea aplicațiilor. Partea practică constă în dezvoltarea unei aplicații în care se utilizează una din tehnologiile descrise.

***Actualitatea și importanța****.* Acestă temă are o importanță majoră, deoarece aplicațiile ce sunt realizate în zilele noastre au nevoie de metode și spații de stocare a datelor, care pe parcursul utilizării sunt adăugate, șterese, actualizate și efectuate alte acțiuni ce necesită interacțiunea peramanentă cu datele stocate ale aplicației. Pe parcursul utilizării aplicației numarul datelor stocate crește rapid, ceea ce necesită spații largi de stocare, acești doi factori numărul mare de date și volumul mare de memorie ocupată duc la frînarea aplicației și ineficienței acestiea. Pentru soluționare acestei probleme, cea mai eficientă metodă de stocare a datelor este utilizarea unei baze date. Bazele de date au ca principii structurarea datelor, pentru a modela aspecte ale realitații cu ajutotul entităților și legaturilor dintre ele, acest fapt duce la eliminarea repetării datelor și eficientizarea întrețineri stocurilor mari de date. Din aceste motive majoritatea aplicațiilor ce necesită stocarea și utilizarea volumului mare de date utilizează pentru metode de stocare a datelor bazele de date.

***Scopul și obiectivele propuse*** suntde a demonstra de ce bazele de date sunt vitale pentru aplicații cu volum mare de date, deasemenea cercetarea tehnolgiilor existente pentru interacțiunea cu bazele de date în cadrul limbajului C#, dezvoltarea unei aplicații utilizînd una din tehnologiile analizate în urma comparațiilor și găsirii aveantajelor acestei tehnologii. Această aplicație va rezolva o problemă reală de stocare a datelor și va fi dezvoltată în conformitate cu cerințele utilizatorului.

***Suportul metodologic şi teoretico-ştiinţific al lucrării*** include studieria tehnologiilor din surse precum materiale video prezentate de dezvoltatori sau persoane ce au contribuit la dezvoltarea tehnolgiilor date, materiale scrise precum articole și cărți, consultarea forumurilor pentru a vedea opinii diferite și a înțelege aspectele tehnologiilor din practica altor oameni, experimentarea tehnolgiilor în diferite exemple practice.

***Noutatea temei investigate*** este cercetearea unui domeniu vital în dezvoltarea aplicațiilor, precum sunt datele și stocarea lor. Deasemenea cercetarea, evidențierea avantajelor și utilizarea tehnologiei Enity Framework care de la lansarea ei a reuțit să acumuleze de 11 milioane de descărcări ceea ce demonstrază încă o data eficacitatea acestei tehnologii și necesitatea lucrului cu date.

***Structura tezei***. Primul capitol al acestei teze are ca scop analiza amplă a factorilor ce favorizează stocarea datelor în baze de date, elementelor componente și crearea corectă a bazei de date în concordanță cu anumite reguli și pași. În partea a doua a acestui capitol se analizează evoluția modalitaților de utilizare a bazei de date în cadrul limbajului C#, se accentuiază tehnologiile ADO.NET care este un set de spații de denumiri ce conțin clase cu ajutorul cărora se realizează conectarea la baza de date, extragerea datelor necesare, salvarea modificarilor și deconectarea de la baza de date, cea mai importantă tehnologie analizată este tehnologia Entity Framework care este cea mai utilizată și oferă cele mai mari facilitați pentru lucrul cu bazele de date, este tehnologia preferată de majoritatea companiilor IT și cea mai recomandata de către specialiști.

Capitolul doi reprezintă partea practică a tezei, a tezei ce are ca scop cercetarea unei probleme și solutionarea ei prin implementarea unei aplicații utilizînd una din tehnologiile cercetate. În urma cercetărilor sa ajuns la decizia de a implementa o aplicație pentru direcția raională pentru siguranță alimentelor Cimișlia, aceasta aplicație are ca rol salvarea în format electronic a autorizatiilor sanitar veterinare de funcționare și generarea rapoartelor în format excel, această implementare va ușura munca angajaților deoarece autorizațiile eliberate se vor păstra în format electronic, ceea ce reduce riscul pierderii autorizațiile eliberate și oferă posibilitatea de generare a rapoartelor.

În partea a doua a capitolului este descrisa proiectarea aplicației, ce cuprinde descrierea direcției raionale pentru siguranța alimentelor Cimișlia, organigrama și funcțiile fiecarui departament din această organizație. Sunt prezentate cerințele funcționale și cele nefunționale ale aplicației. Acest lucru a fost realizat deoarece este esențial de a cunoaște cine vor fi utilizatorii aplicației și a ințelege care este procesul pe care aplicația trebuie să îl înlocuiască, pentru a putea realiza un produs cît mai calitativ pentru cerințele utilizatorului. În acest capitol este explicată și structura proiectului realizat care are două module primul este responsabil de structura și interacținuea cu baze de date și al doilea modul este responsabil de interacțiunea cu utilizatorul oferidui posibilitatea de a efectua acțiunile dorite. Este reprezentat și rezultatul proiectului, interfața utilizatorului cu posibilitatile pe care le oferă acesta aplicație introducerea unei autorizatii noi, filtrarea cautarea și generarea rapoartelor în format excel.

# I. TEHNOLOGIILE .NET PENTRU BAZELE DE DATE

## 1.1. Concepte fundamentale ale modelelor de stocare a datelor

Pentru a înțelge necesitatea și importanța modelelor de structurare și stocare a datelor se va analiza noțiunile de dată și informație ce au semnificații diferite. *Datele* sunt fapte colectate din viaţa cotidiană bazate pe observaţii și măsurători. *Informaţia* este rezultatul prelucrării și interpretării datelor de către un subiect, în dependența de subiect datele prelucrate pot oferi inforamții diferete. Datele au un caracter obiectiv, în timp ce informaţia este subiectivă. Pentru a putea obține inforamații cit mai corecte datele trebuie colectate, memorate, organizate și prelucrate, pentru efectuarea acestor acțiuni deobicei se utilizează fișiere și baze de date.

Organizarea datelor înseamnă definirea şi structurarea datelor în memoria internă sau în memoria externă şi stabilirea legăturilor între date, conform unui model de date. În decursul dezvoltării metodelor de structurare au fost utilizate mai multe concepte, prezentate în tabelul 1.1.

Tabelul 1.1. *Evolutia sistemelor de stocare a datelor* [2].

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Etape | Pina la 1960 | Anii 1960 | Anii 1970 | Sfîrșitul 1970 prezent | Sfîrșitul 1980  prezent |
| Mod de organizare | Fișiere secvențiale | Fișiere secvențiale indexate | Baze de date arborescente | Baze de date relaţionale | Baze de date orientate obiect |

Pe parcursul dezvoltării modulelelor de organizare a datelor, inițial datele se stocau în fișiere, acest model permite stocarea datelor în mai multe fișiere [1]. Pentru o orgnizație se generaza fișiere diferite în depentență de department, în schimbul păstrării lor centralizat, de exemplu:

Departamentul Contabilitate stochiaza:

(Nume, Prenume, Adresă, DataNaşterii, Salariu, NumărSecţie)

Departamentul Resurse Umane stochează:

(Nume, Prenume, Adresă , DataNaşterii, NumărTelefon, Funcţie, NumărSecţie)

În acest exemplu se observă o dublare a datelor, defapt dublarea datelor există în toate sistemele bazate pe fișiere pe lînga acest neajuns, acest model de organizare are mai multe limitari:

1. *Separarea și izolarea datelor* are ca efect faptul că, pentru anumite aplicaţii este necesară prelucrarea şi sincronizarea a două sau mai multe fişiere.
2. *Dublarea datelor* implică două dezavantaje:
   1. risipa de spaţiu de stocare;
   2. posibila alterare a integrităţii datelor.
3. *Dependenţa de date* implică modificarea tuturor programelor de aplicaţie în cazul modificării tipului sau dimensiunii unui câmp de date, activitate ce necesită mult timp și este supusă apariţiei erorilor.
4. *Incompatibilitatea fişierelor,* structura fişierelor este încorporată în programele aplicaţie, ea este dependentă de limbajul în care sunt scrise acestea.
5. *Interogarea fixă a programelor aplicaţie* a fost generată de faptul că sistemele bazate pe fişiere sunt în mare măsură dependente de programatorul de aplicaţii. Interogările sau rapoartele necesare trebuie să fie scrise de către acesta. Din acest motiv apăr două situaţii:
   1. tipurile de interogări sau rapoarte care pot fi realizate sunt fixe și nu exista nici un fel de facilitate de a realiza interogări neplanificate;
   2. apăritia unui mod specific pentru fiecare organizatie de scriere în fișiere și a aplicațiilor.

Toate limitările bazate pe fişiere se datorează următorilor doi factori:

1. Definiţia datelor este încorporată în programele aplicaţie, în loc de a fi stocată separat și independent;
2. Nu există un control al accesului și manipulării datelor, în afară de cel impus de către programele aplicaţie.

Pentru a evita toate aceste probleme ce apar în urma utilizării unui sistem bazat pe fișiere, cel mai eficient este utilizarea unei baze de date.

*Baza de date* este o colecţie partajată de date elementare sau structurate, între care există relaţii logice, mai detaliat este un depozit de date unic, care este definit o singură dată și este utilizabil simultan de mai mulţi utilizatori, toate datele sunt integrate, cu o dublare minimă și este o resursă comună, partajată, ea conţine nu numai datele operaţionale, ci și o descrierea acestora.

Caracteristica principală a aplicaţiilor ce utilizează baze de date constă în faptul că accentul este pus pe operaţiile de memorare și regăsire efectuate asupra unui volum mare de date şi mai puţin asupra operaţiilor de prelucrare a acestora. Principala operaţie în aceste aplicaţiile este regăsirea datelor în scopul obţinerii informaţiilor din baza de date. Utilizarea unei baze de date oferă urmatoarele avantaje:

1. *Centralizarea datelor* ofera urmatoarele avantaje:
2. *Reducerea redundanţei datelor memorate*. În cazul sistemelor bazate pe fişiere, este posibil ca aceleaşi date să apară de mai multe ori în fişiere diferite, aparţinând unor aplicaţii diferite.
3. *Evitarea inconsistenţei datelor memorate*: atunci când există mai multe copii ale aceleiaşi date este posibil ca prin actualizarea numai a unora dintre ele, să existe valori diferite pentru una şi aceeaşi dată. Aceasta atrage după sine inconsistenţa bazei de date.
4. *Posibilitatea partajării datelor*, se referă nu numai la posibilitatea utilizării în comun a datelor de către mai mulţi utilizatori, ci și la posibilitatea de a dezvolta aplicaţii noi folosind datele deja existente în baza de date.
5. *Posibilitatea aplicării restricţiilor de securitate*: existând controlul centralizat al datelor, se pot introduce verificări de autorizare a accesului la date. Se pot impune restricţii diferite pentru fiecare tip de acces (regăsire, actualizare, ştergere), pentru fiecare tip de dată şi la nivelul fiecărui utilizator.
6. *Posibilitatea introducerii unor proceduri de validare a datelor* în cazul operaţiilor de actualizare, pentru a asigura cerinţa ca baza de date să conţină date corecte.

2*. Independenţa dintre date și program*. Baza de date, ca imagine a unei anumite realităţi, trebuie actualizată permanent. Acest lucru nu trebuie să afecteze aplicația. Pentru aceasta trebuie ca fiecare program să aibă o viziune proprie asupra bazei de date.

3. *Realizarea legăturilor între entităţile de date*, necesare pentru exploatarea eficientă a bazei de date.

4. *Integritatea datelor asigură fiabilitatea şi coerenţa bazei de date*. Pentru aceasta trebuie definite restricţii de integritate cum ar fi:

* Apartenenţa la o listă de valori sau la un interval.
* Apartenenţa la un anumit format.
* Reguli de coerenţă cu alte date, aceste reguli trebuie să respecte atât aspectele statice, cât şi cele dinamice. De exemplu, după o actualizare de creştere a salariului, nivelul acestuia trebuie să fie mai mare decât cel precedent.

5*. Securitatea datelor*. Baza de date trebuie să fie protejată împotriva unei distrugeri logice (anomalii de actualizare) sau fizice. Pentru aceasta, există instrumente care permit:

* Crearea unor *puncte de repriză* ce reprezintă salvarea din timp în timp a unor copii coerente ale bazei de date.
* Gestiunea unui *jurnal de tranzacţii* ce este lista operaţiilor realizate asupra bazei de date după ultimul punct de repriză. Dacă apare o anomalie care ar putea distruge baza de date, pe baza copiei de la ultima repriză și a jurnalului de tranzacţii se poate reface baza de date în forma iniţială.

6. *Confidenţialitatea datelor* este asigurată prin proceduri de:

* Identificare a utilizatorilor prin nume sau cod.
* Autentificarea prin parole.
* Autorizarea accesului diferenţiat prin drepturi de creare, consultare, modificare sau ştergere pentru anumite sectoare de date.

*7. Partajarea datelor* permite înlănţuirea tranzacţiilor solicitate simultan pe aceeaşi înregistrare din baza de date, prin blocarea cererilor în aşteptare și deservirea ulterioară a acestora.

Unul dintre cele mai importante și des utilizate sisteme de baze de date este sistemul de baze de date relațional. Modelul relaţional a fost propus de către IBM și a revoluţionat reprezentarea datelor făcând trecerea la generaţia a doua de baze de date. Modelul are o solidă fundamentare teoretică fiind bazat pe teoria seturilor și logica matematică. Pot fi reprezentate toate tipurile de structuri de date de mare complexitate, din diferite domenii de activitați. Modelul relaţional este definit prin structura de date, operatorii ce acţionează asupra structurii și restricţiilor de integritate [9].

O dată cu dezvoltarea foarte rapidă a aplicațiilor web și numarului imens de date, a apărut conceptul de baze de date nerelaționale pentru a putea menține performanța aplicațiilor. O bază de date nerelațională nu stochează date folosind tabele ci folosind chei de identificare. Datele pot fi regăsite în funcţie de cheile asignate. Acest tip de baze de date evadează din rigorile relaţionale prin lipsa unei scheme, lipsa necesităţii de normalizare a datelor şi de stocare a relaţiilor dintre tabele aducând astfel performanţe sporite aplicaţiilor care le folosesc. De asemenea acest tip de baze de date îmbunătăţesc şi răspunsul la schimbări de-a lungul timpului. Într-un sistem relaţional nu există flexibilitatea necesară pentru a asimila modificări în modelul de date. Faptul că bazele de date NoSQL nu au o schemă de date fixă face aceste baze de date să fie mult mai flexibile şi adaptabile la schimbări de model în cursul anilor.

## 1.2. Tehnologia ADO.NET

În zielele de astăzi, datele sunt păstrate în diferite formate, variind de la date păstrate în baze de date pînă la informații salvate în documente word, mesaje de pe poșta electronică și multe alte tipuri. ADO sau denumirea din limba engleză ActiveX Data Objects este o tehnologie de acces a datelor ce simplifică utilizarea lor de la diferiți furnizori, eliberînd astfel programatorii de învățarea tehnologiilor pentru utilizarea datelor de la diferiți furnizori și dezvoltînd aplicații nedependente de furnizorul de date. Filosofia celor de la Microsoft din spatele ecestei tehnologii este Accesul Universal al Datelor în engleza Universal Data Access (UDA). UDA nu este o tehnologie dar mai degrabă o strategie pentru a rezolva problema accesului la date, al cărui scop este un acces eficient și puternic de date, indiferent de sursa de date sau de limbaj de dezvoltare. Mai mult decât atât, acest acces universal are menirea de a elimina nevoia transformării datelor existente dintr-un format în altul. Cum sa specificat mai sus ADO este o tehnologie independentă față de limbajul de programare. Acest lucru înseamnă că, indiferent de limbajul de dezvoltare a aplicației, Visual Basic, VBScript, Visual Basic for Applications (VBA), Visual C ++, Visual J ++, sau JavaScript - interfața de dezvoltare este identică [4].

O dată cu apariția platformei .NET dezvoltată la fel de către cei de la Microsoft a fost schimbată și tehnologia ADO, care din ADO a devenit ADO.NET și bine înțeles cu multe alte îmbunătațiri care au fost făcut în aceasta tehnologie.

Tabel 1.2. *Evolutia tehnologiei ADO* [5].

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Anul lansării | 1996 | 1998 | 2000 | 2002 | 2004 | 2008 | 2010 |
| Versiunea tehnologiei | ADO 1.0 | ADO 2.1 | ADO 2.5 | ADO.NET | ADO.NET 2.0 | ADO.NET  3.5 | ADO.NET 4.0 |

Platforama .NET definește un set de spații de denumiri și clase care fac perte din tehnologia ADO.NET ce permite interacțiunea cu bazele de date cum sunt Microsoft SQL Server, Oracle Server, MySQL Server și altele. ADO.NET are la bază două metode de a interacționa cu baza de date, prima fiind conectarea permanentă la baza de date și a doua este conectarea doar în cazul reînoirii datelor [3].

ADO.NET nu are un sigur set de spații de denumiri și clase care comunică cu toate tipurile de sisteme de baze de date. Dar oricum, ADO.NET suportă mai mulți furnizori, fiecare furnizor avînd un spațiu de denumiri cu clase similare și fiind specifice fiecărui sistem, pe lînga spațiile de denumiri pentru fiecare furnizor ADO.NET conține și un spațiu de denumiri *System.Data.Common* cu toate componenetele cumune ale tuturor furnizorilor.

Primul beneficiu al acestei abordări este posibilitatea de a utiliza spațiul de denumiri pentru un furnizor spefic și a utiliza toate funcționalitățile spefice acestui sistem.

Tabel 1.3 *Principalele componente comune ale sistemelor de baze de date în ADO.NET* [3].

|  |  |  |  |
| --- | --- | --- | --- |
| Tipul obiectului | Clasa de baza | Interfața | Descrierea obiectului |
| Connection | DbConnection | IDbConnection | Oferă posibilitatea de conectare la și deconectare de la sistemul de date. |
| Command | DbCommand | IDbCommand | Reprezintă o interogare SQL sau o procedură stocată. De asemenea, oferă acces la datele citite de la furnizor |
| DataReader | DbDataReader | IDbDataReader | Oferă posibilitatea de citire unidirecțională a datelor direct de la server |
| DataAdapter | DbDataAdapter | IDbDataAdapter | Administrează transferurile de date între aplicație și sistemul de date. DataAdaptor conține o conexiune și un set de patru obiecte pentru a selecta, însera, actualiza și șterge datele din sistemul de date. |
| Parameter | DbParameter | IDbParameter | Reprezintă un parametru definit pentru o interogare |
| Transaction | DbTransaction | IDbTransaction | Reprezintă incapsularea unei tranzacții |

În dependență de furnizorul de date elementele au denumiri specifice, dar implementeză și derivă de la aceleași clase de baze specificate mai sus (ex. SqlConnection, OracleConnection, OdbcConnection, MySqlConnection) acest lucru permite utilizarea diferitor furnizori utilizînd aceleași metode de utilizare. Cum sa specificat mai sus ADO.NET este un set de spații de denumiri care permit lucrul cu bazele de date. În tabelul 1.4 sunt analizate doar cele mai importante spații de denumiri și rolul acestora .

Tabel 1.4 *Principalele spatii de denumiri din ADO.NET* [4].

|  |  |
| --- | --- |
| Spațiul de denumiri | Descrirea spațiului de denumiri |
| Microsoft.SqlServer.Server | Acest spațiu de denumiri furnizează tipurile care facilitează CLR și Microsoft SQL Server |
| System.Data | Acest spațiu de denumiri definește tipurile de bază ADO.NET utilizate de către toți furnizorii de date, inclusiv interfețe comune și numeroase tipuri care reprezintă stratul deconectat (de exemplu, DataSet și DataTable). |
| System.Data.Common | Acest spațiu de denumiri conține tipuri partajate între toți furnizorii de date ADO.NET, inclusiv clasele abstracte de bază comune. |
| System.Data.SqlTypes | Acest spațiu de denumiri conține tipuri de date native folosite de Microsoft SQL Server. Se pot utiliza întotdeauna tipurile de date CLR corespunzătoare, dar sunt SQLTYPE-uri optimizate pentru a lucra cu SQL Server (de exemplu, în cazul în care baza de date SQL Server conține o valoare întreagă, aveți posibilitatea să o reprezentați folosind fie int sau SqlTypes.SqlInt32). |

Dintre toate spațiile de denumiri din ADO.NET, spațiul de denimuri *System.Data* este numitorul comun pentru toate funcționalitățile oferite de ADO.NET. Nu se pot construi aplicațiile utilizînd ADO.NET fără a specifica acest spațiu de denumiri. *System.Data* conține tipurile care sunt partajate între toate sistemele de baze de date care se utilizează cu ajutorul tehnologiei ADO.NET.În plus acest spațiu de denumiri conține și un set de excepții specifice în utilizarea bazelor de date (de exemplu, NoNullAllowedException, RowNotInTableException). *System.Data* conține tipuri care reprezintă componente a bazei de date (de exemplu tabele, rânduri, coloane și constrângeri), precum și interfețe comune implementate de clasele specifice ficărui sistem de date implementat în ADO.NET. În Tabelul 1.5 sunt reprezentate principalele componente ale acestui spațiu de denumiri *System.Data.*

Tabel 1.5 *Componentele principale ale spațiului de denumiri System.Data* [4].

|  |  |
| --- | --- |
| Tipul | Descrierea |
| DataSet | Reprezintă baza de date construită din un oarecare număr de obiecte de tipul DataTable ce reprezintă tabelele, toate acestea sunt păstrate în memoria cache. |
| DataTable | Reprezintă un tabel din baza de date |
| DataColumn | Reprezintă o coloană a unui tabel de tip DataTable |
| DataRow | Reprezintă o linie dintrun obiect DataTabel |
| Constraint | Reprezintă o constringere asupra unei obiect de tip DataColumn |
| DataRelation | Reprezintă relația copil părinte dintre daouă obiecte de tip DataTable |
| IdataAdapter | Definește funcționalitățile unui obiect de tipul DataAdapter |
| IdataReader | Definește funcționalitățile unui obiect de tipul DdataReader |
| IdbCommnad | Definește funcționalitățile unui obiect de tipul Command |

Pentru a putea utiliza toate aceste componente oferite de *System.Data* primul pas ce trebuie făcut este conectarea la baza de date, pentru a face acest lucru fiecare sistem de baze de date are implementat o clasa specifică Connection, în dependență de furnizor ea are denumire difereită (ex. SqlConnection, OracleConnection, MySqlConnection ), funcționalitățile de bază ale acestei clase care sunt comune sunt definite de interfața:

*public interface IDbConnection : IDisposable*

*{*

*string ConnectionString { get; set; }*

*void Close();*

*void Open();*

*}*

Acestea sunt doar cîteva din cele mai importante proprietăți și funcționalități ale acestei interfețe, metoda *Open()* și *Close()*, deschid și respectiv închid conexiunea la baza de date, specificare bazei de date se face cu ajutorul proprietății *ConnectionString* care reprezintă un șir de caracte unde sunt specificate numele bazei de date și numele serverului. În exemplul urmator este realizat conectarea la un anumit tip de sistem de baze de date și anume *Microsoft SQL Server*

*SqlConnection connection = new SqlConnection (@"Data Source = (local)\SQLEXPRESS; Initial Catalog = USM; Integrated Security = True");*

Această metodă de conectare face aplicația construită dependenată de un anumit sistem de baze de date, în cazul în care este necesar schimbarea altui sistem de date va fi nevoie de schimbat o mare parte din cod în dependență de noul sistem de date, pentru a ușura acest lucru se paote folosi spațiul de denumiri *System.Data.Common* ce conține un set de clase și interfețe ce permite tratarea unor componente polimorfic, în cazul abordat se procedează în modul următor, în fișierul \*.config se poate specifica tipul sistemului de baze de date și stringul de conecatere.

*<appSettings>*

*<add key="provider" value="System.Data.SqlClient" />*

*<add key="cnStr" value= "Data Source=(local)\SQLEXPRESS;*

*Initial Catalog=USM; Integrated Security=True"/>*

*</appSettings>*

Acesta este un fragment de fișier de tip XML ce conține unele configurații ale aplicației, cu ajutorul unor clase specifice se pot extrage unele date în momentul cînd deja aplicația este pornită.

*string dataProvider = ConfigurationManager.AppSettings["provider"];*

*string connectionString = ConfigurationManager.AppSettings["cnStr"];*

*DbProviderFactory dataFactory = DbProviderFactories.GetFactory(dataProvider);*

*using (DbConnection connection = dataFactory.CreateConnection())*

*{*

*connection.ConnectionString = connectionString;*

*}*

În acest fragment de cod C# se extrag din fisierul \*.config cu ajutorul clasei *ConfigurationManager* numele sistemului de date și stringul de conectare după care se crează obiectul *DbConnection* care oferă toate funcționalitățile pentru a lucra cu baza de date.

### 1.2.2. ADO.NET metoda conectată

În cele descrise mai sus au fost descrise daor componentele comune ale celor două metode de utilizare a bazelor de date utilizînd tehnologia ADO.NET, prima fiind metoda conectată iar a doua metoda deconectată, care constă în conectare doar în cazul cînd sunt extrase date din baza de date și atunci cînd se modifică datele din baza de date propriu-zisă.

Pentru utilizarea metodei conectată mai întii este nevoie de a crea o conexiune utilizînd una din metodele descrie mai sus, principalul component pentru utilizarea acestei metode este clasa DataReader care se obține cu ajutorul clasei Command care oferă posibilitatea de a executa cereri SQL asupra bazei noastre de date și în urma execuției se obține un obiect de tip DataReader ce permite accesarea rezultatului unei cererii SQL executată. În figura de mai jos Fig. 1.1 este reprezentată arhitectura acestei metode:

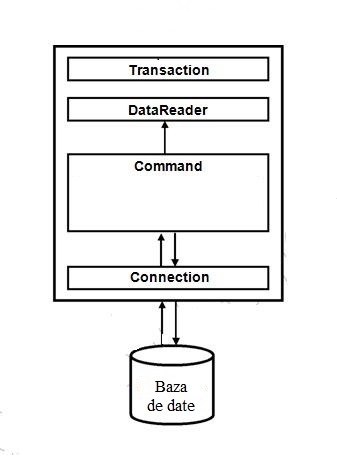


Fig. 1.1. *Arhitectura ADO.NET metoda conectată.* [6]

Această metodă permite de a avea toate tiprile de comenzi SQL precum de selecție, înserare, actualizare și stergere, doar în urma selecției vom obține un obiect DataReader, care accesează datele direct din baza de date, utilizînd metoda Read() din acestă clasă, se obține cîte un tuplu din toate care sau obținut în urma cererii SQL.

Această metodă de utilizare a bazei de date este eficientă în cazuri în care este nevoie de iterat rapid asupra unui set mare de date și nu este nevoie de a păstra toate datele în memoria aplicației, un aspect negativ este că trebuie să existe o conectare permanentă la baza de date plus trebuie să se deschidă și închidă conexiunea, ceea ce în metoda deconectată acest lucru este efectuat de către sistem.

### 1.2.2. ADO.NET metoda deconectată

Conceptul care stă la baza acestei metode și de unde vine denumirea ei este de a optimiza interacțiunea cu bazei de date, dacă în metoda conectată conexiunea este deschisă și inchisă de către programator, atunci în metoda deconectată aceste acțiuni sunt efectuate automat de către sistem. Inițial datele necesare sunt copiate în memoria aplicației și apoi de fiecare dată cînd este nevoie de a reactualiza datele din baza de date sau din aplicație, se deschide o conexiune, se efectuiează toate acțiunile necesare după care conexiunea se închide, acesta metoda înlătură necesitatea de a fi conectați permanent la baza de date.

În tabelul 1.5 au fost enumerate cîteva tipuri de date din spațiul de denumiri *System.Data*, o mare parte din ele sunt utilizate în acestă metodă, aceste tipuri sunt *DataSet, DataTable, DataColumn, DataRow, Constrints, DataRelation* etc. Toate aceste tipuri sunt reprezentarea bazei de date în aplicație, DataSet reprezintă baza de date propriu-zisă, iar DataTable tabelele din baza de date și respectiv celelalte clase sunt alte componente ale baze de date, cu ajutorul acestui set de clase se pot efectua diferite acțiuni asupra bazei de date. În figura 1.2 este reprezintă arhitectura acestei metode.

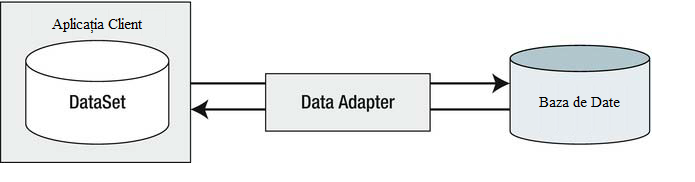


Figura 1.2 *Arhitectura ADO.NET metodei deconectată*

Clasa care este responsabilă de sincronizarea datelor din aplicație și a celor din baza de date este DataAdapter, în Tabelul 1.6 sunte reprezentate principalele metode care sunt necesare pentru efectuarea sincronizării.

Tabel 1.6 *Principalele metode și propreietăți ale clasei DataReader* [3].

|  |  |
| --- | --- |
| Metoda | Descrierea |
| Fill() | Această metodă execută o comandă SQL SELECT (așa cum este specificat de către proprietatea SelectCommand) pentru a interoga baza de date și a înscri datele într-un obiect DataTable. |
| SelectCommand | Interogare de tipul SQL SELECT care se execută cu ajutorul metodei Fill() |
| InsertCommand | Interogare SQL INSERT care se execută cu ajutorul metodei Update() |
| UpdateCommand | Interogare SQL UPDATE care se execută cu ajutorul metodei Update() |
| DeleteCommand | Interogare SQL DELETE care se execută cu ajutorul metodei Update() |
| Update() | Execută comenzile specificate de proprietățile  InsertCommant, UpdateCommand și DeleteCommand |

Următorul fragment arată cum pot fi utilizate metodele enumerate în Tabelul 1.6, crearea unui obiect DataSet ce reprezintă baza de date și apoi popularea acesteia, utilizînd un obiect DataReader

*// Crearea unui obiect DataSet*

*DataSet dataSet = new DataSet("USM");*

*// Crearea unui obiect DataAdapter*

*SqlDataAdapter dataAdapter = new SqlDataAdapter("Select \* From USM", ConnectionString);*

*// Apelarea metodei Fill*

*dataAdapt.Fill(dataSet, "USM");*

## 1.3. Tehnologia Entity Framework

Majoritate aplicațiilor elaborate folosesc baze de date în care sunt stocate, prelucrate datele necesare pentru aplicație, de fiecare dată cînd este necesar adaugarea sau prelucrarea dateleor este nevoie de a elabora metode care vor realiza aceste funcționalități, pentru realizarea acestor metode se consumă foarte mult timp, pe lînga elaborarea funcționalității aplicațiilor care este defapt scopul principal în elaboararea unei aplicații. Utilizînd tehnologia ADO.NET menționată mai sus va fi nevoie de a scris foarte mult cod care de multe ori este foarte dificil și costisitor de menținut. Pentru a simplifica modalitatea de utilizare a bazelor de date, se paote utiliza arhitectura reprezentata în figura 1.3

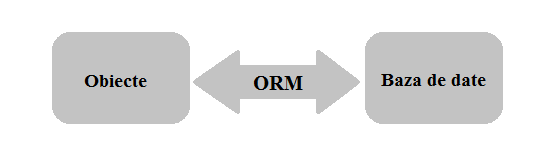


Figura 1.3. ORM

Această arhitectură stabilește relația dintre obiectele și clasele care sunt similare cu tabelele și colaonele din baza de date, ceea ce ne ofera posibilitatea de a lucra cu clase și obiecte fara a utiliza o sumedenie de clase și interfețe adiționale pentru a ne conecta și apoi deconecta de la baza de date, plus de a scri interogari SQL direct în codul aplicației. În anul 2008 Microsoft lansează ORM-ul Entity Framework care are rolul de a simplifica utilizarea bazelor de date, acestă noua tehnologie are un succes foarte mare și a reusit să înlocuiască utilizarea tehnologiei ADO.NET, ultima versiune EF 6 a reușit să adune pîna în prezent peste 11 milioane de descărcări, iar ultima subversiune EF 6.1.3 peste 750 000 de descărcări doar în primele 3 luni. În tabelul 1.7 sunt reprezentate toate versiunile Entity Framework lansate de către Microsoft.

Tabel 1.7. *Versiuni Entity Framework* [8].

|  |  |  |
| --- | --- | --- |
| Virsiunea Entity Faramework | Versiunea .Net Framework | Anul lansării |
| Enity Framework v1.0 | .Net Framework 3.5 | 2008 |
| Enity Framework v4.0 | .Net Framework 4.0 | 2010 |
| Enity Framework v5.0 | .Net Framework 5.5 | 2012 |
| Enity Framework v6.0 | .Net Framework 4.6 | 2013 |

Pe parcursul dezvoltării acestei tehnologii, pîna în prezent sa ajuns la trei metode de utilizare a bazelor de date folosind Entity Framework, prima posibilitate oferită în versiununea v1.0, bineînțeles care a evoluat foarte mult pîna la versiunea curentă v.6.0, este crearea mai întii a bazei de date și apoi integrarea acesteia în aplicație.

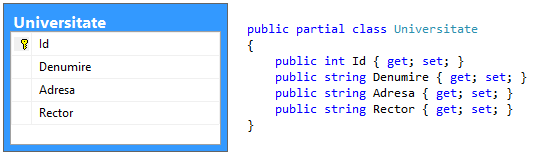


Figura 1.4. *Maparea clasei din aplicație cu tabelul din baza de date*

Întegrarea constă în maparea bazei de date cu clasele și obiectele aplicației, în figura 1.4 este reprezentată maparea unui tabel din baza de date cu o clasă care este identică cu tabelul respectiv, denumirile coloanelor fiind proprietățile clasei, iar denumira tabelului numele clasei.

Cea mai utilizată metodă însă este crearea claselor în aplicație și apoi generarea bazei de date în concordanță cu clasele create. Acest lucru se face cu ajutorul unei clase abstracte *DbContext* în înteriorul carea se efectuiază conxiunea la baza de date și generarea bazei de date propriu-zise. Această metoda ne permite ca aplicația noastră sa nu conțină clase elaborate pentru a putea lucra cu baza de date, doar sa creem clase ce vor reprezenta entitatile aplicației.

A treia metodă care are la baza metoda crearii claselor și apoi generarea bazei de date este crearea modelului prin metode interactive de tragere și plasare și apoi generarea bazei de date. Aceasta metoda ne permite crearea bazei de date fara a scri vreo careva clasa doar de a genera tabel și crea legaturi dintre acestea, în spatele acestei metoda defapt se genereaza un cod XML, din care apoi se genereaza clasele și se revine la metoda a doua din clasele create se genereaza cod SQL care și crează baza de date mapata cu clasele din aplicație.

Toate tabelele care sunt generate se genereaza conform unor convenții stabilite, pentru schimbarea acestor conventii sunt folosite două tehnologii *Data Anotation* și *Fluent API*.

Pentru ultimile două metode mentionate apare notiunea de migratie care reprezintă defapt schimabrea structurii claselor ce reprezintă tabelele din baza de date.

# II. DEZVOLTAREA UNEI APLICAȚII UTILIZÎND TEHNOLGIA ENTITY FRAMEWORK

Acest capitol reprezintă partea practică a tezei care constă în realizarea unie aplicații utilizînd tehnologiile menționate în primul capitol. Aplicația a fost realizată în urma consultarii și ințelegerii cu utilizatorii acesteia care sunt angajații direcției raională pentru siguranța alimentelor Cimișlia (DRSA Cimișlia) care nu dispun de produse soft pentru inregistrarea autorizațiilor sanitar veterinare de funcționare (Anexa 1).

## 2.1. Proiectarea Aplicației

Sa gasit necesar implemntarea unui sistem informatic deoarece în urma eliberarii acestor autorizații datele sunt păstrate în formă scrisă, în caiete sau alte metode de păstrare a datelor, dar nu sunt utilizate metodele de păstrare în format electronic. Evident toate aceste informații sunt vulnerabile și pot fi foarte ușor pierdute, de asemenea angajații au obligația de a face rapoarte, rapoartele includ în sine informații despre autorizații.

Pentru soluționarea acestei probleme cea mai eficienta soluție este bineințeles realizarea unui produs soft ce ar perminte angajaților realizarea acetui proces la calculator și salvarea datelor în format electoronic. Salvarea datelor în format electornic reduce riscul pierderii acestor informatii, de asemenea permite cu usurință gasirea lor după anumite criterii precum sortarea și generearea rapoartelor în diferite formate fie fișiere word sau excel.

### 2.1.1. Structura Organizației

Agenția regională DRSA Cimișlia responsabilă de reglementarea și control pentru sigutanța alimentelor și în domeniul sanitar-veterinar, zootehnic, al protecției plantelor și carantinei fitosanitare, controlul semincer, calitații produselor primare, produselor alimentare și a hranei pentru animale. Agenția dispune de o serie de drepturi care sunt puse în practica prin angajatii sai.

* *Să dispună accesul* inspectorilor aflați în exercițiu, conform legislaţiei în vigoare, în incintele şi pe teritoriul staţiilor de cale ferată, aeroporturilor, staţiilor auto, oficiilor poştale, elevatoarelor, depozitelor, în vagoanele de marfă și de pasageri, autobuze, pe aeronavele civile și pe teritoriul întreprinderilor, instituţiilor şi organizaţiilor cu orice tip de proprietate, care produc, achiziţionează, prelucrează şi comercializează producţie şi materiale supuse supravegherii și controlului în domeniile vizate; [10]
* *Sa interzică*

1. importul, comercializarea şi utilizarea produselor de uz fitosanitar şi a fertilizanţilor, produselor farmaceutice de uz veterinar care nu corespund standardelor naţionale, reglementărilor tehnice și altor acte tehnico-normative;
2. prelucrarea şi comercializarea produselor agroalimentare şi folosirea apei în cazul depistării de reziduri de produse de uz fitosanitar, fertilizanţi, reziduri de hormoni și produse farmaceutice de uz veterinar ce depăşesc limitele maxim admisibile;
3. utilizarea produselor genetic modificate etc.

* *Să solicite eliberarea din funcție a persoanelor* antrenate în activițati cu produse de uz fitosanitar și cu fertilizanți care nu au fost supuse examenului medical și nu au fost instruite în privinta securitații muncii în cadrul acestor activitați etc.



Figura 2.1 *Organigrama Directiei sanitar veterinare [10]* În figura 2.1 este reprezentată organigrama instituției ce reprezintă o prezentare amplă a structurii orgnizatorice.Șeful subdiviziunii are trei șefi adjuncți, consacrați activitaților după cum urmeaza

* Pentru probleme sanitar-veterinare.
* Pentru probleme fitosanitare și controlul calitații.
* Pentru problemele siguranței alimentelor.

### 2.1.2. Cerințele față de sistemul informatic

Destinația acestui sistem informatic este organizația DRSA Cimișlia, pentru care este necesar dezvoltarea unui soft, ce va avea o serie de funcționalități accesibile utilizatorilor:

* Colectarea, stocare datelor introduse de către utilizator.
* Viziualizarea a datelor.
* Filtrarea datelor.
* Generarea rapoartelor.

Există doar un tip de utilizatori care vor avea o singură interfață de acces:

* Angajații.

Dezvoltarea acestui SI soluționează urmatoarele probleme:

* Siguranța datelor.
* Vizualizarea rapidă a datelor.
* Căutarea după anumite criterii.
* Generarea rapoartelor.

Tabel 2.2 *Cerințe funcționale*

|  |  |  |
| --- | --- | --- |
| Nr. | Descrierea cerinței | Prioritatea |
|  | Posibilitatea de a introduce și salva datele autorizației | înaltă |
|  | Avertizarea utilizatorului cînd datele pentru salvare nu sunt complete | înaltă |
|  | Posibilitatea de a sterge toate datele inserate, dar nesalvate | medie |
|  | Vizualizarea datelor salvate | înaltă |
|  | Căutarea autorizațiilor pe anumite perioade de timp | înaltă |
|  | Căutarea autorizațiilor după numele subdiviziunii teritoriale | medie |
|  | Căutarea după alte criterii | medie |

*Cerințe nefuncționale:*

1. Cerințe de confidențialitate:

* Datele din baza de date nu trebuie să fie disponibile utlizatorilor, doar administratorului bazei de date.
* Aplicația nu trebuie să dea datele persoanei care a elibrat autorizația.

1. Cerințe față de interfața utilizator:

* Casetele pentru introducerea aplicației să nu conțină fonturi stridente.
* Textul să fie de dimensiuni citibile.
* Dimensiunea ferestrei să fie suficinet de mare pentru a putea introduce, vizualiza datele.

1. Cerințe de operabilitate:

* Orice acțiune să nu dureze mai mult de 3 secunde.
* Intorducerea datelor și vizualizarea lor să fie separate.
* În cazul generării unui raport numele fișierului să fie ales de utilizator.
* Adresa de salvare a raportului să fie ales de utilizator.

1. Cerințe de fiabiliate:

* În cazul deconectării datele intorduse să nu fie salvate, se salvează doar în cazul cînd utilizatorul va da acordul.

## 2.2. Realizarea Aplicației

În urma analizei cerințelor înaintate de către utlizatori sa decis pentru realizarea acestei aplicații să se utilizeze urmatoarele limbaje și tehnologii, limbajul de programare C#, tehnologia Windows Forms deoarece toate aceste mecanisme permit realizarea cerințelor cerute foarte rapid și calitativ. Crearea și interacțiunea cu baza de date a fost realizată cu ajutorul tehnologiei Entity Framework ce permite generarea bazei de date de la o structură de clase care sunt create în limbajul de programare, acest lucru aduce flexibilitate bazei de date în conformitate cu modificările aplicației, a fost utilizată versiunea de Entity Framework 6.1.3 care este ultima versiune lansată, instalarea acestei tehnologii este foarte simplă, mediul de dezvoltare utilizat Visul Studio dispune de un set foarte larg de instrumente printre care NuGet, acest instrument permite instalarea foarte rapidă a tuturor tehnologiilor existente pentru un anumit limbaj sau platformă de dezvoltare a aplicațiilor.

Soluția aplicației este formată din două proiecte, sepărarea realizării aplicației în mai multe proiecte permite dezvoltarea și menținerea aplicației eficient fără a implica toate proiectele aplicației. Primul proiect este *Authorization.Data* în care sunt realizate toate acțiunile ce țin de datele aplicației inclusiv interacțiunea cu baza de date. Proiectul doi este *Authorization.WinForms* în care este realizat logica aplicației, prelucrearea datelor și interfața utilizatorului.

Proiectul *Authorization.Data* ține de datele aplicației, este compus din mai multe clase, dintre care 4 din ele reprezintă tabelele din baza de date iar una din clase este baza de date propriu-zisă sau în termenii tehnologiei respective este numit și contextual bazei de date. Clasele respective sunt *ActivityProfile, Agent, Authrization, Subdivision.* Un element foarte important este realizarea legăturilor corecte dintre tabelele unei baze de date, în structura creată se întilnesc două tipuri de legături 1-n și m-n ele sunt reprezentate în figura 2.2, pentru realizarea acestor legături sunt respectate urmatoarele reguli. Pentru legătura 1-n, exemplu clasele *Authorization* și *Agent*, în care *Agent* este legătura 1 iar *Authorization* n.

|  |  |
| --- | --- |
| 1 | n |
| *public class Authorization*  *{*  *[ForeignKey(nameof(Agent))]*  *public int AgentId {get;set;}*  *public Agent Agent {get;set;}* | *public class Agent*  *{*  *public ICollection<Authorization> Authorizations {get;set;}*  *}* |

Clasa *Authorization* are două proprietăți care sunt strict legate de realizarea legăturii cu clasa *Agent*, proprietatea Agent este referința obiectului cu care se face legătura și respectiv proprietatea AgentId este idul acestui obiect, aceste două proprietăți fac legătura de tip 1, în clasa *Agent* pentru realizarea legăturii de tip n este o colecție de obiecte de tip Authorization.

Pentru realizarea legăturii de tip n-m, nu este nevoie de a crea careva clase intermediare, în baza de date automat se va genera un table ce va realiza legătura de tip n-m, în aplicația realizată legătura n-m există între clasele *ActivityProfile* și *Authorization*.

|  |
| --- |
| n |
| *public class Authorization*  *{*  *public virtual ICollection<ActivityProfile> ActivityProfiles { get; set; }*  *}* |
| m |
| *public class ActivityProfile*  *{*  *public virtual ICollection<Authorization> Authorizations { get; set; }*  *}* |

În exemplul dat legătura este realizată prin proprietăți ce reprezintă cîte o colecție de obiecte de tipul respectiv, în baza de date este creat un tabel intermediar cu ajutorul căruia se va efectua legătura. Clasa care zeprezintă baza de date în aplicație este clasa *AuthorizationDbContext* care moștenește clasa abstractă *DbContext* din spațiul de denumire *System.Data.Entity* care conține un set de metode care permit manipularea bazei de date. Proprietățile acestei clase sunt respectiv tabelele din baza de date care sunt reprezentate de un set de colecții de tipul *DbSet* care reprezintă un tabel.

*public class AuthorizationDbContext : DbContext*

*{*

*public DbSet<Authorization> Authorizations { get; set; }*

*public DbSet<Subdivision> Subdivisions { get; set; }*

*public DbSet<Agent> Agents { get; set; }*

*public DbSet<ActivityProfile> ActivityProfiles { get; set; }*

*}*

În exemplul de mai sus este reprezentată clasa *AuthorizationDbContext* care are 4 proprietăți de tipul DbSet, care sunt tabelele din baza de date reprezentate sub formă de obiecte.

Un alt element important pentru această tehnologie sunt migrțiile ce reprezintă schimbările efectuate asupra bazei de date, realizarea lor se face prin modificarea contextului, modificări precum adăugarea, stergerea claselor și proprietaților. Pentru utilizarea migrațiilor sunt necesare trei comenzi:

1. *Enable-Migrations*, permite utilizarea migrațiilor și se utilizează doar o singură dată în proiect.
2. *Add-Migration*, de fiecare dată după efectuarea unui modificări, această comandă va adăuga schimbarea respectivă.
3. *Update-Database*, aseastă comandă sincronizează modificările din aplicație și baza de date.

În momentul creării unei migrații se generează o clasa ce moștenește clasa abstracta *DbMigration*

*public partial class InitialCreate : DbMigration*

*{*

*public override void Up()*

*{*

*CreateTable("dbo.ActivityProfiles", c => new*

*{*

*Id = c.Int(nullable: false, identity: true),*

*Activity = c.String(),*

*})*

*.PrimaryKey(t => t.Id);*

*}*

*public override void Down()*

*{*

*DropTable("dbo.ActivityProfiles");*

*}*

*}*

Această clasă conține două metode, în prima metodă *Up()* se generează tot codul ce va efectua modificările în baza de date conform schimbărilor apărute în contextul dat, metoda *Down()* este codul ce anulează toate schimbările din metoda *Up()*, metoda *Down()* se aplică în cazul cînd este nevoie de a reveni la o anumită stare a bazei de date, respectiv toate migrațiile aplicate vor putea fi anulate cu ajutorul acestei metode.

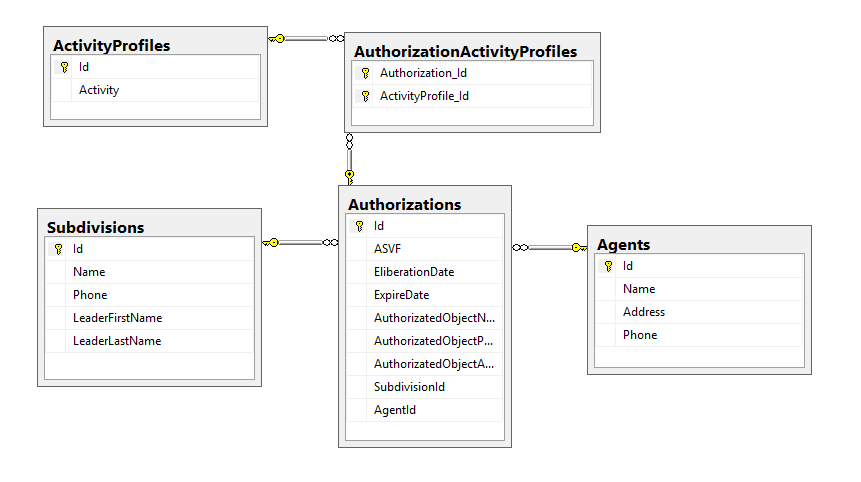


Figura 2.2 *Structura bazei de date*

În figura 2.2 este reprezentată structura bazei de date a aplicației realizate care a fost generată în urma creării claselor în proiectul *Authorization.Data* tabelele date au denumirile claselor la plural.

În proiectul *Authorization.WinForms* interacțiunea cu baza de de date se realizează prin intermediul unui obiect de tipul *AuthorizationDbContext,* prelucrarea datelor din baza de date se poate realiza utilizînd 3 metode:

1. SQL.
2. LINQ.
3. Metode LINQ.

În aplicația dată au fost utilizate metode LINQ, care permit interogarea bazei de date cu ajutorul unor metode specifice limbajului, iar în spatele lor se generează cereri SQL care sunt optimizate și permit obținerea rezultatului cu eficiență și performanță maximă.

Pentru reprezentarea grafică și interacțiunea cu utilizatorul a fost utilizată tehnologia Windows Forms ce oferă realizarea aplicațiilor desktop pentru Windows într-un mod eficient. Visual Studio oferă pentru această tehnologie posibilitatea de tragere și plasare a componentelor, iar codul fiind generat de infrastructura tehnologiei. În această tehnologie trebuie să existe o clasă de bază ce moștenește calsa *Form,* această clasă este fereasta de bază pe care vor fi plasate toate componentele necesare, în momentul unei acțiuni din partea utilizatorului se va arunca un eveniment ce va lansa o anumită metodă cu instrucțiunile așteptate de către utilizator. Pentru a delimita codul generat de infrastructura tehnolgiei și codul scris de programator tehnologia utlizaează clase parțiale care sunt scrise în fișiere diferite, iar în timpul compilării sunt compilate ca un fișier unic.

Aplicația are doua secțiuni care sunt reprezentate prin taburi, prima este crearea unei autorizații și introducerea datelor, a doua este filtrarea, căutarea și generarea rapoartelor. În prima secțiune este foarte importantă verificarea datelor introduse, o autorizație nu poate fi introdusă cu careva lipse de date, de aceea acestea sunt verificate în metoda de mai jos

*private bool IsInputDataNull()*

*{*

*var result = string.IsNullOrEmpty(ASVFTextBox.Text)*

*&& string.IsNullOrEmpty(AgentNameTextBox.Text)*

*...*

*}*

Această metodă se lansează cînd apăsăm butonul ok pentru a salva datele introduse, în momentul apăsării acestui buton se lansează metoda *OkButton\_Click.*

*private void OkButton\_Click(object sender, EventArgs e)*

*{*

*if (!IsInputDataNull())*

*{*

*InsertDataToDataBase();*

*SetSubdivisions();*

*ClearTextBoxes();*

*}*

*}*

Această metodă pe lingă verificarea datelor, apelează și alte metode care salvează datele introduse în baza de date și pregătește interfața grafică pentru introducerea altor autorizații. În cazul în care datele introduse nu sunt complete utilizatorul va primi un mesaj de eroare pentru a introduce toate datele corecte.

Secțiunea de filtrare, căutare și generare de rapoarte este în tabul doi care este compus din două secțiuni, prima elementele de filtrare și secțiunea unde utilizatorul poate vedea rezultatele obținute, filtrarea se face după mai multe criteri, acest lucru se efectuiază în metodă de mai jos.

*private ICollection GetFilteredData()*

*{*

*if (ASVFSearchBox.Text != String.Empty)*

*{*

*result = result.Where(a => a.ASVF.ToLower().*

*Contains(ASVFSearchBox.Text.ToLower()));*

*}*

*...*

*}*

În acest fragment este arătat cum are loc filtrarea după criterii, ea este efectuată doar dacă sunt introduse date pentru criteriul respectiv, în caz contrar filtrarea după criteriul dat nu are loc, pentru filtrare sunt utilizate metode LINQ ce permit eficient efectuarea instrucțiunilor. Datele obținute sunt atribuite obiectul de tip *DataGridView* care afișiază datele în formă de tabel, metoda de mai jos setează doar sursa datelor iar restul instrucțiunilor sunt efectuate de infrastructura tehnologiei.

*private void SearchButton\_Click(object sender, EventArgs e)*

*{*

*dataGridView.DataSource = GetFilteredData();*

*}*

Generarea rapoartelor are loc în fișier excel, aceste operațiuni sunt efectuate în metoda *SaveButton\_Click,* unde este creat un fișier de tip excel și sunt inserate datele.

Toate aceste tehnologii utilizate oferă posibilitetea de a utiliza clase deja existente ce permit efectuarea instrucțiunilor și lucrul cu datele mult mai eficient și rapid.

## 2.3. Funcționalitățile Aplicației

Aplicația are două cerințe de bază, prima este inregistrarea datelor introduse în baza de date, a doua cerință este filtrarea, căutarea și generarea rapoartelor în fișier excel. La lansarea aplicației prima opțiune este de a înscri o autorizație nouă, în figura 2.3 este reprezentată fereastra principală și tabul „Inserare Autorizație”, datele autorizației care sunt necesare de a fi introduse sunt grupate în categorii:

* Date autorizație.
* Agent economic.
* Obiect autorizat.
* Date subdiviziune.
* Șef subdivisiune.
* Profilul activitații.

În partea stingă sunt toate casetele ce trebuie să fie completate pentru a putea însera autorizația în baza de date, în partea dreaptă sunt toate activitățile existente, ele pot fi selectate fară a fi scrise manual.



Figura 2.4 *Secțiunea Inserare Autorizație.*

Dacă casetele date nu sunt completate sau numele subdiviziunii noi deja există aplicația va afișa un mesaj de eoroare din fugura 2.4 care va atenționa utilizatorul sa completeze toate casetele necesare pentru crearea unei autorizații.

|  |  |
| --- | --- |
|  |  |

Figura 2.5 Mesaje de eroare

În tabul doi care se numește „Căutare Autorizație” permite utilizatorului manipularea datelor existente în baza de date și generarea rapoartelor în fișire excel, această fereastră este reprezentată în fugura 2.6 și este compusă din două secțiuni, criterile de filtrare și rezultatul căutarii.

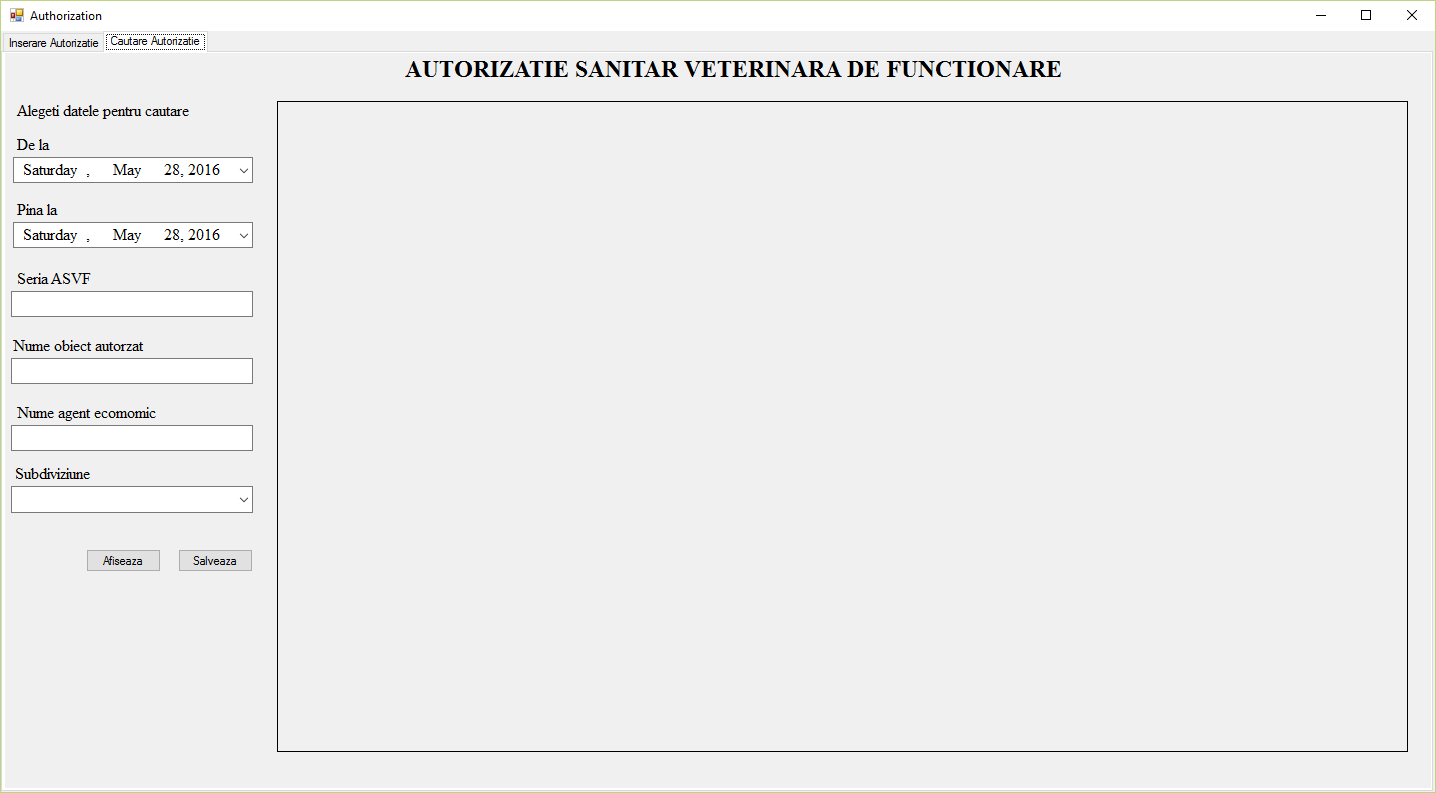


Figura 2.6. *Secțiunea Căutare Autorizație*

În prima secțiune filtrarea datelor poate fi efectuată după urmatoarele criterii:

* Data eliberarii.
* Data expirarii.
* Seria ASVF.
* Nume obiect autorizat.
* Nume agent economic.
* Subdiviziune.

Dacă caseta pentru unul din criteriile enumerate nu este completată atunci filtrarea după acest criteriu nu se ia în considerație, respectiv filtrarea se va face în dependență de casetele completate, dacă nici una din casete nu este completată, atunci vor fi afișate toate autorizațiile existente. În figura 2.7 este reprezentat rezultatul unei căutări, datele sunt afișate în formă de tabel, iar denumirile cîmpurilor sunt marcate cu bold.

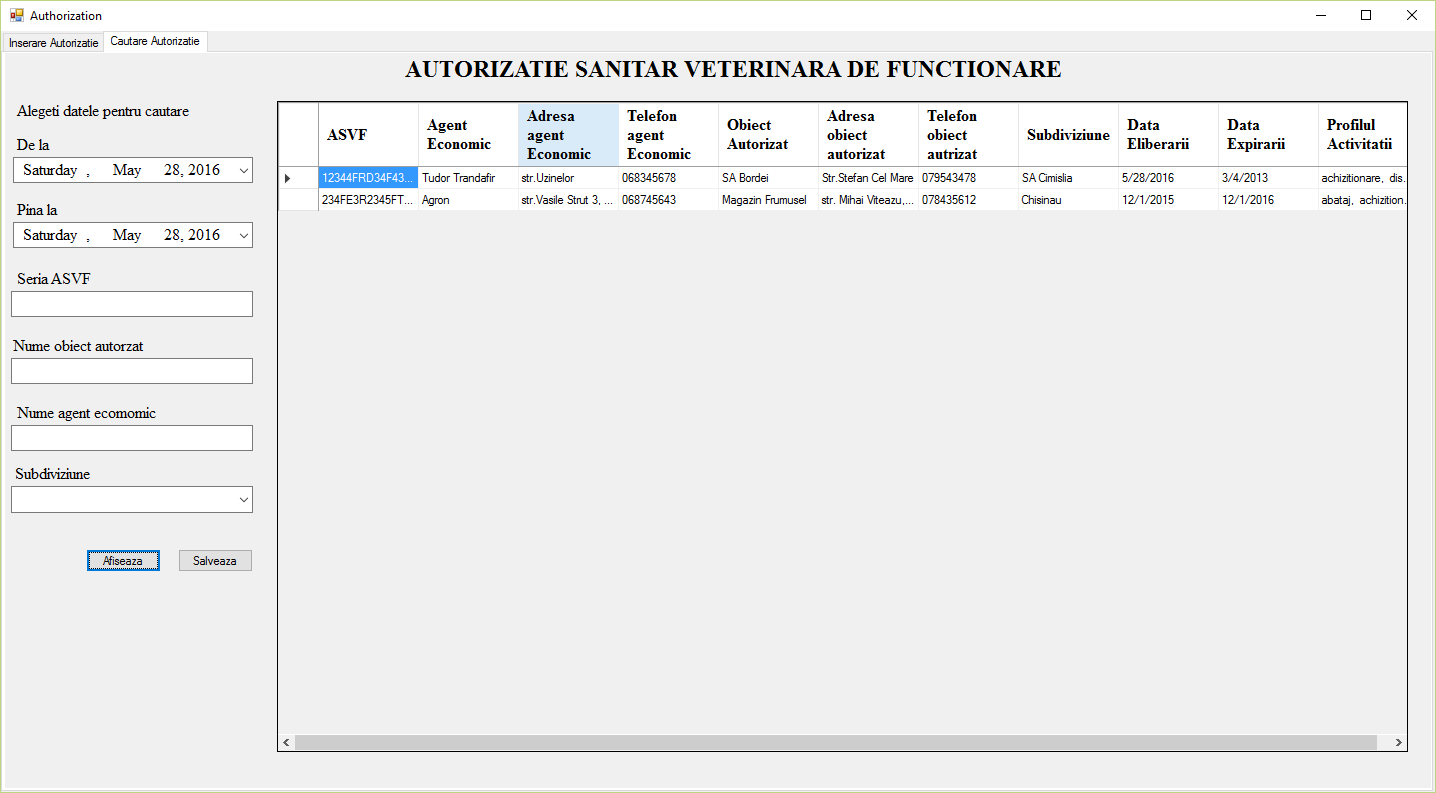


Figura 2.7. *Rezultatul căutării autorizațiilor*

Pentru rezultatul obținut în urma unei căutări este și posibilitatea de a salva datele din tabel în fișier excel, acest lucru se face prin apăsarea butonului „Salvează” care deschide fereastra de salvare și permite utilizatorului să aleagă destinația și numele fișierul. În fereastra respectivă formatul de salvare al fișierului este setat implicit excel, această fereastră este reprezentată în figura 2.8.

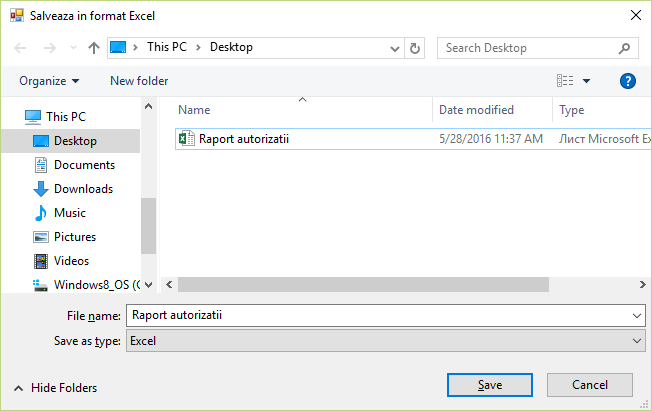


Figura 2.8. *Salvarea raportului în format excel.*

Aplicația a fost realizată în conformitate cu cerințele utilizatorilor și permite eficientizarea lucrului cu tipul dat de autorizații, au fot utilizate mai multe tehnologii ce permit realizarea aplicațiilor de nivel înalt. Interfața grafică este foarte comună cu elementele din sistemul de operare Windows ceea ce îi permite utilizatorului confort în folosirea ei și nu este necesar de a citi instrucțiuni de utilizare. De asemenea rezultatele obținute în urma căutării pot fi salvate în fișiere excel.

# CONCLUZII

Această teză a fost realizată în urma analizei componentelor comune ale aplicațiilor existente, sa ajuns la concluzia că lucrul cu datele și stocarea lor este necesară în majoritetea aplicațiilor, din acest motiv sa ales investigarea și găsirea celei mai bune și eficiente soluții pentru stocarea și prelucrarea datelor, sau adus argumente de ce bazele de date sunt mai eficiente decît alte sisteme de stocare a datelor, de asemenea au fost comparate tehnologiile existente de interacțiune cu bazele de date în cadrul limbajului C#.

Pentru relizarea tezei sau cercetat și studiat diferitor tehnologii, din diferite surse dintre care materiale video prezentate de dezvoltători profesioniști sau persoane ce au contribuit la dezvoltarea tehnolgiilor date, de asemenea diferite materiale scrise precum articole și cărți, au fost consultate diferite forumuri pentru a vedea opinii diferite și a ințelege aspectele tehnologiilor din practica altor oameni, după studierea tehnologiilor sa încercat utilizarea lor în diferite exemple practice, în final sa decis care dintre tehnolgii vor fi utilizate.

Pentru realizarea unei aplicații în primul rînd este nevoie de cerințe care vor defini aplicația realizată, este foarte important ca aplicația să îndeplinească cerințele înaintate, de multe ori aplicația realizată de către dezvoltatori nu corespunde cu cerințele clientului sau clientul iși schimbă opinia și cerințele în urma unui rezultat prealabil al viitoarei aplicații. Pentru aplicația descrisă în lucrarea dată, inițial sa discutat cu utilizatorii, care au avut urmatoarele cerințe de bază, crearea și salvarea autorizațiilor, filtrarea și căutarea autorizatiilor, generarea rapoartelor, acest lucru a dus la întelegerea mai profundă a problemei pe care trebuie să o rezolve aplicația dată.

Scopul acestei lucrări a fost de a găsi cele mai eficiente metode de utilizare a bazei de date în cadrul limbajului C# și de a motiva care sunt beneficiile păstrării datelor în baza de date în favoarea altor metode de păstrare, în capitolul întîi au fost aduse argumente de ce păstrarea datelor în baza de date este mult mai ieficientă dectît păstrarea lor în fișiere, a fost definită noțiunea de bază de date și componentele ei. Au fost analizate toate posibilitățile și tehnologiile existente în cadrul limbajului pentru utilizarea bazelor de date, tehnologia de bază este ADO.NET, această tehnologie fiind foarte dificil de utilizat și nevoiea de a ințelege foarte bine structura bazei de date cu toate componenetele ei, pentru a eficientiza lucrul cu bazele de date au apărut tehnologiile de mapare a tabelelor din baza de date cu obiecte și clase din limbajele de programare, tehnologia utilizată în cadrul limbajului C# este Enitity Framework ce permite lucrul cu bazele de date foarte eficient. În urma analizei ambelor tehnolgii sa ajuns la concluzia că pentru realizarea aplicației de utilizat tehnologia Enity Framework fiind o tehnologie nouă și recomandată de mulți dezvoltatori.

Capitolul doi reprezintă descrierea procesul de realizare a aplicației propuse inițial, pentru acest scop în prima parte este descrisă structura organizației ce urmează să utilizeze aseastă aplicație, care este compusă din mai multe compartimente fiecare dintre ele avînd rolul său, în urma analizei organizației și problemei existente sunt descrise un set de cerințe față de aplicația dată, ele sunt divizate în cerințe funcționale și cerințe nefuncționale, acest lucru sa făcut pentru a defini inițial un set de reguli pentru a dezvolta corect aplicația, de a acoperi exact cerințele cerute și a satisface dorințe utilizatorilor față de aplicație. În partea a doua a capitolului doi sunt descrise succint cele mai importante aspecte ale realizării aplicației, printre ele sunt metodele de utilizare a tehnologiei Entity Framework, particularitațile acestei tehnologii sunt mult mai clare și ușor de ințeles cînd sunt aplicate în practică. În finalul capitolui doi sunt menționate functionalitațile aplicației realizate și structura aplicației.

Datele reprezintă unul din compartimentele de baza în realizarea produselor softwere, din acest motiv este necesar de a găsi cele mai bune soluții pentru a avea un rezultat calitaiv și eficient.

# BIBLIOGRAFIE

1. Angel Cațaron , Baze de Date, 2004, extras 25 februarie 2016, din sursa

<http://vega.unitbv.ro/~cataron/Courses/BD/BD_Cap_1.pdf>

1. Lungu Ion, Butha Iuliana, Curs de Baze de Date, 2005, extras 28 martie 2016, din sursa <http://bd.ase.ro/uploads/bd_curs/Curs_BD_Lungu,Botha.pdf>
2. Andrew Troelsen, Pro C# 5.0 and The .NET 4.5 Framework, 2012, p.807, extras 3 martie 2016
3. Jason T Roff, ADO: ActiveX Data Objects, 2001, extras 3 martie 2016, din sursa <https://www.safaribooksonline.com/library/view/ado-activex-data/1565924150/ch01.html>
4. Julie Lerman,Geting started with Entity Framewk 5, 2013, extras 10 martie 2016, din sursa

<https://www.pluralsight.com/courses/entity-framework5-getting-started>

1. Connected Arhitecture of ADO.NET, 2012, extras 10 martie 2016, din sursa

<http://programcall.com/4/adonet/connected-architecture-of-adonet.aspx>

1. Mosh Homedani, Entity Framework in depth, 2015, extras 25 martie 2016, din sursa

<https://www.udemy.com/entity-framework-tutorial/learn/v4/overview>

1. Julie Lerman,Geting started with Entity Framewk 6, 2015, extras 25 martie, din sursa

<https://www.pluralsight.com/courses/entity-framework-6-getting-started>

1. Valeriu Lupu, Teoria bazelor de date, 2003, extras 26 aprilie 2016, din sursa

<http://www.seap.usv.ro/~valeriul/lupu/cafec/capitolul1.pdf>

1. Agenția Națională pentru Siguranta elimentelor, extras 25 aprilie 2016, din sursa

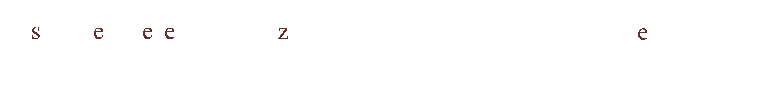
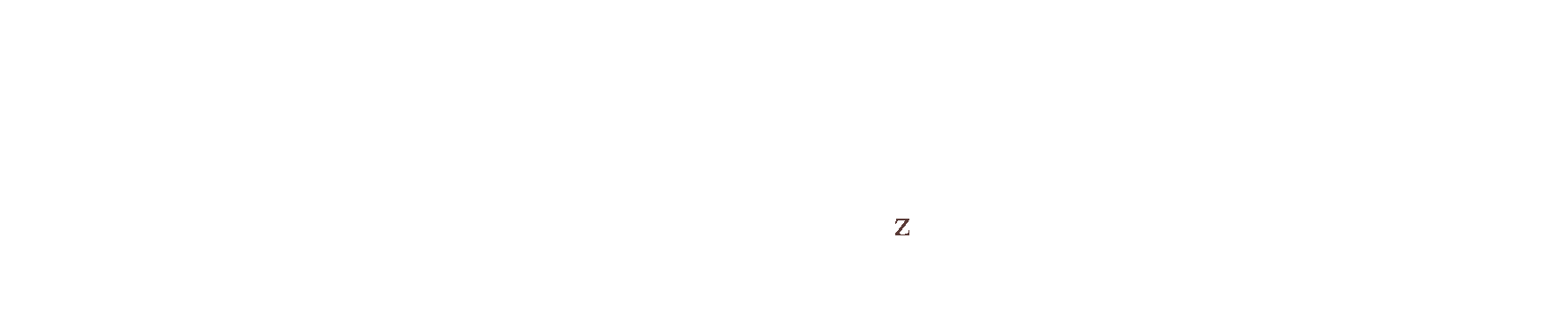
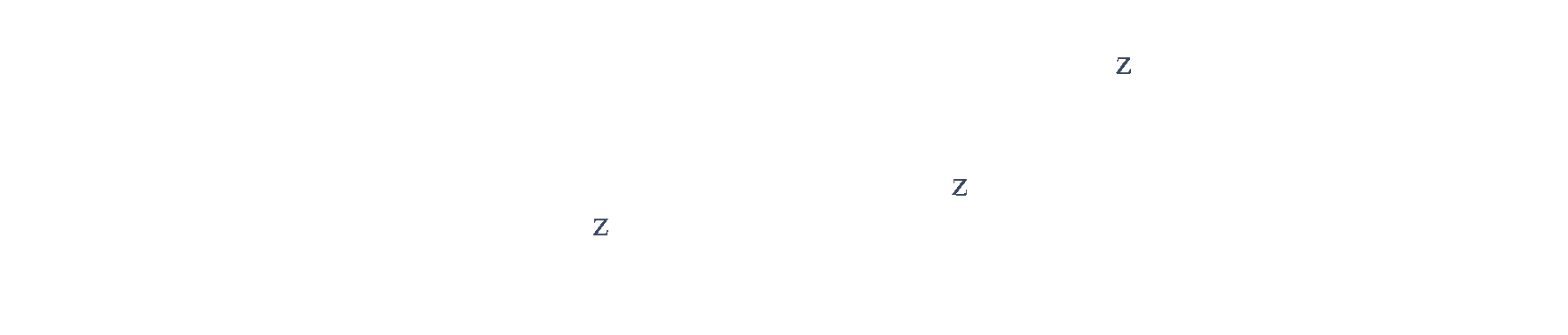
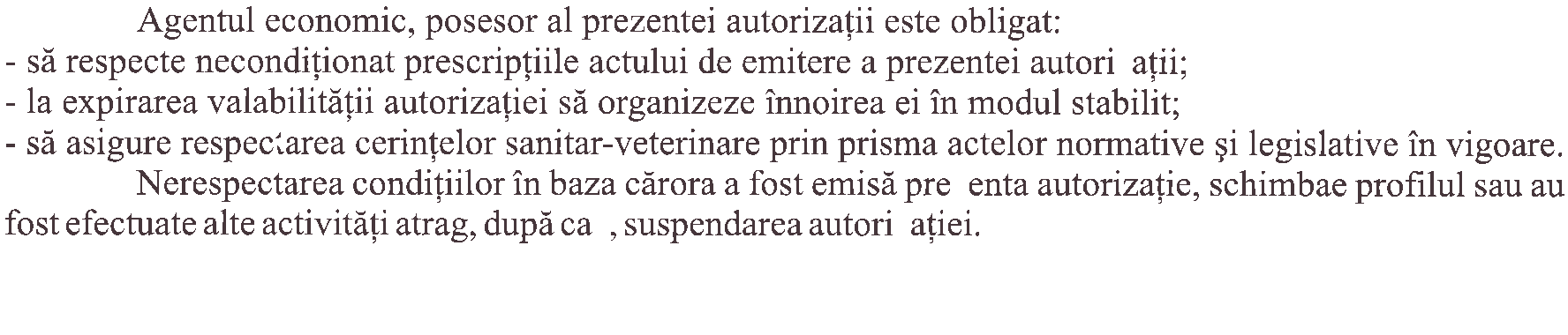
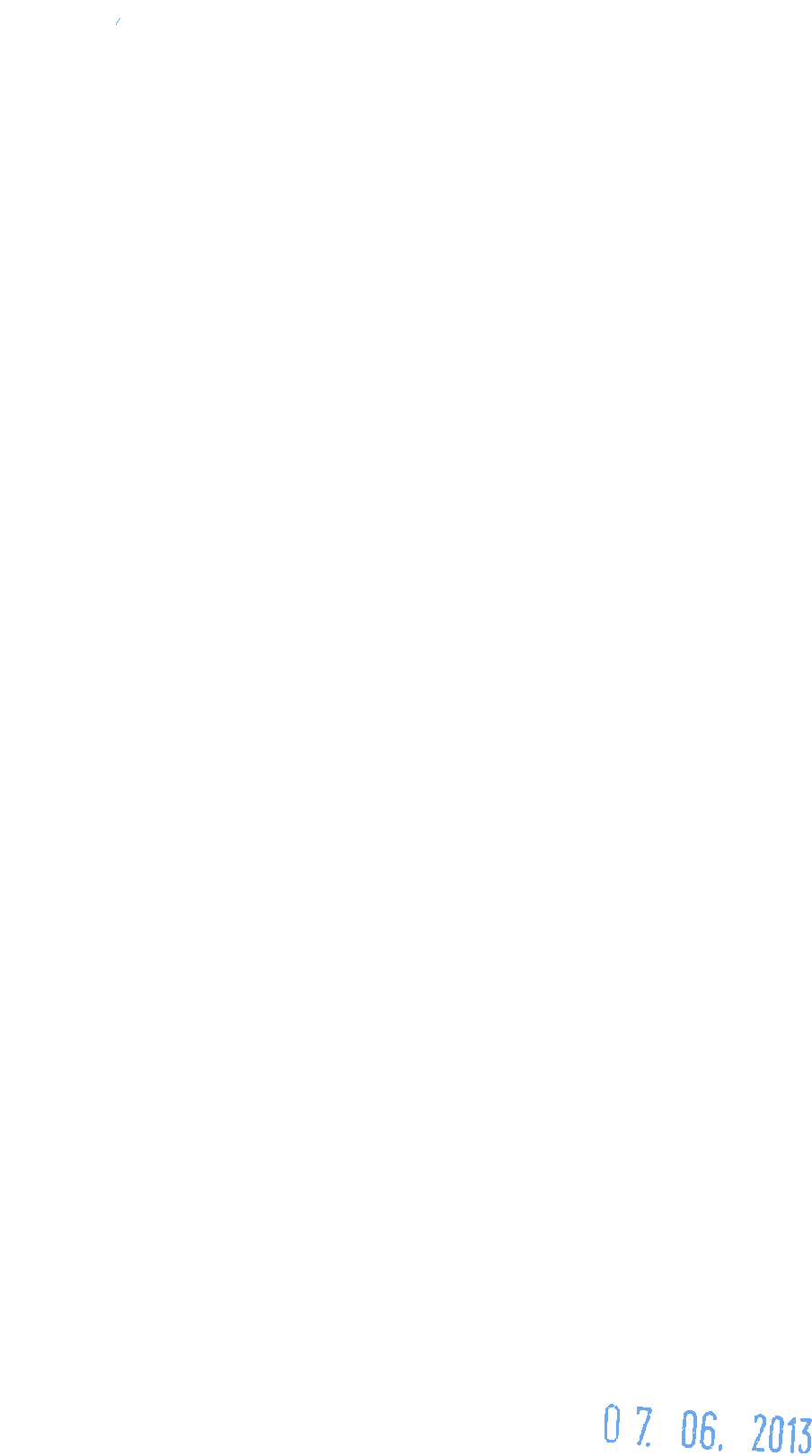
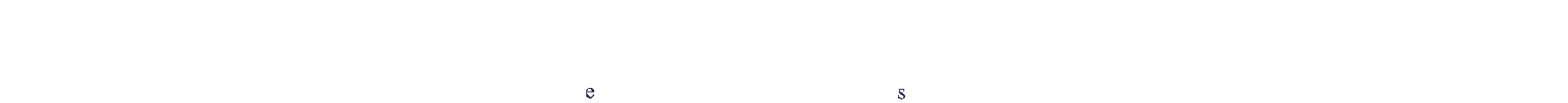
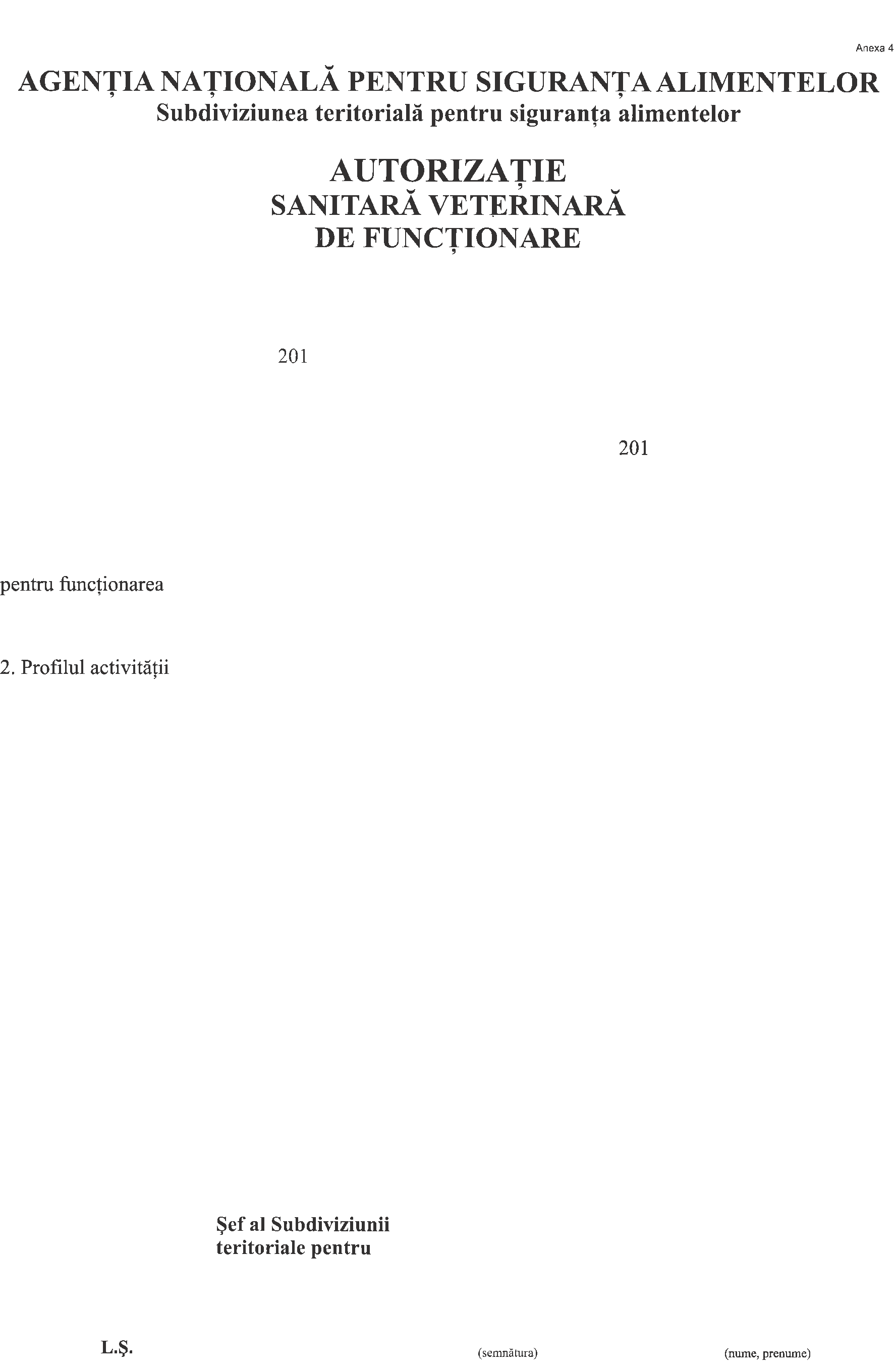
<http://www.ansa.gov.md/ro/structuri-teritoriale.html>

1. Autorizație sanitară veternară de funcționare, extras 25 aprilie 2016, din sursa

http://www.ansa.gov.md/uploads/files/Siguranta/Autorizatie%20San-Vet\_Functionare.pdf

# Anexe

**Anexa I. Autorizație Sanitară Veterinară de funcționare**



**Anexa 2. Metoda de salvare a datelor în fișier excel**

*private void SaveButton\_Click(object sender, EventArgs e)*

*{*

*saveFileDialog.Title = @"Salveaza in format Excel";*

*saveFileDialog.Filter = @"Excel |\*.xlsx";*

*if ( saveFileDialog.ShowDialog() != DialogResult.Cancel )*

*{*

*Excel.Application excelFile = new Excel.Application();*

*excelFile.Application.Workbooks.Add(Type.Missing);*

*excelFile.Columns.ColumnWidth = 25;*

*for (int i = 0; i < dataGridView.Columns.Count; i++)*

*{*

*excelFile.Cells[1, i + 1] = dataGridView.Columns[i].HeaderText;*

*}*

*for (int i = 0; i < dataGridView.Rows.Count; i++)*

*{*

*for (int j = 0; j < dataGridView.Columns.Count; j++)*

*{*

*excelFile.Cells[i + 2, j + 1] = dataGridView.Rows[i].Cells[j].Value.ToString();*

*}*

*}*

*excelFile.ActiveWorkbook.SaveCopyAs(saveFileDialog.FileName);*

*excelFile.ActiveWorkbook.Saved = true;*

*excelFile.Quit();*

*}*

**Anexa 3. Metoda de salvare a datelor în baza de date.**

*private void InsertDataToDataBase()*

*{*

*Subdivision subdivision;*

*if (SubdivisionsComboBoxInsertTab.Text != Empty)*

*{*

*var existingSubdivision = \_dbContext*

*.Subdivisions*

*.SingleOrDefault(s => s.Name == SubdivisionsComboBoxInsertTab.Text);*

*subdivision = existingSubdivision;*

*}*

*else*

*{*

*var newSubdivision = new Subdivision*

*{*

*Name = SubdivisionNameTextBox.Text,*

*Phone = SubdivizionPhoneTextBox.Text,*

*LeaderFirstName = LeaderFirstNameTextbox.Text,*

*LeaderLastName = LeaderLastNameTextBox.Text*

*};*

*subdivision = \_dbContext.Subdivisions.Add(newSubdivision);*

*}*

*var agent = new Agent*

*{*

*Name = AgentNameTextBox.Text,*

*Address = AgentAddressTextBox.Text,*

*Phone = AgentPhoneTextBox.Text*

*};*

*var addesAgent = \_dbContext.Agents.Add(agent);*

*var authorization = new Authorization*

*{*

*ASVF = ASVFTextBox.Text,*

*EliberationDate = EliberationDateTime.Value.Date,*

*ExpireDate = ExpirationDateTime.Value.Date,*

*AuthorizatedObjectName = ObjectNameTextBox.Text,*

*AuthorizatedObjectAddress = ObjectAddressTextBox.Text,*

*AuthorizatedObjectPhone = ObjectPhoneTextBox.Text,*

*SubdivisionId = subdivision.Id,*

*AgentId = addesAgent.Id,*

*};*

*var addedAuthorization = \_dbContext.Authorizations.Add(authorization);*

*var checkedActivities = ActivityListInsertTab.CheckedItems;*

*foreach (string item in checkedActivities)*

*{var activity = \_dbContext.ActivityProfiles.Where(*

*ap => ap.Activity == item).SingleOrDefault();*

*\_dbContext.Authorizations.Find(addedAuthorization.Id).ActivityProfiles.Add(activity);*

*}*

*\_dbContext.SaveChanges();*

*}*

**Anexa 4. Metoda de filtrare a datelor din baza de date.**

*private ICollection GetFilteredData()*

*{*

*var result = \_dbContext.Authorizations.Include(ap => ap.ActivityProfiles)*

*.Include(ap => ap.Agent)*

*.Include(ap => ap.Subdivision)*

*.Select(at => new AuthorisationModel*

*{*

*ASVF = at.ASVF,*

*AgentName = at.Agent.Name,*

*AgentAddress = at.Agent.Address,*

*AgentPhone = at.Agent.Phone,*

*AuthorzatedObjectName = at.AuthorizatedObjectName,*

*AuthorizatedObjectAddress = at.AuthorizatedObjectAddress,*

*AuthorizatedObjectPhone = at.AuthorizatedObjectPhone,*

*SubdivisionName = at.Subdivision.Name,*

*EliberationDate = at.EliberationDate,*

*ExpirationDate = at.ExpireDate,*

*Activities = at.ActivityProfiles.Select(ac => ac.Activity).ToList()*

*.Aggregate((current, next) => Concat(current, ", ", next))*

*});*

*if (EliberationDateSearch.Value.Date < ExpirationDateSearch.Value.Date)*

*{*

*result = result.Where(at => at.EliberationDate >= EliberationDateSearch.Value.Date)*

*.Where(at => at.ExpirationDate <= ExpirationDateSearch.Value.Date);*

*}*

*if (ASVFSearchBox.Text != Empty)*

*{*

*result = result.Where(a => a.ASVF.ToLower().Contains(ASVFSearchBox.Text.ToLower()));*

*}*

*if (ObjectSearchBox.Text != Empty)*

*{*

*result = result.Where(a => a.AuthorzatedObjectName.ToLower().Contains(ObjectSearchBox.Text.ToLower()));*

*}*

*if (AgentSearchBox.Text != Empty)*

*{*

*result = result.Where(a => a.AgentName.ToLower().Contains(AgentSearchBox.Text.ToLower()));*

*}*

*if (SubdivisionsComboBoxISearchTab.Text != Empty)*

*{*

*result = result.Where(a => a.SubdivisionName.ToLower().Contains(SubdivisionsComboBoxISearchTab.Text.ToLower()));*

*}*

*return result.ToList();*

*}*

**Anexa 4.Verificarea datelor introduse.**

*private bool IsInputDataNull()*

*{*

*var result = IsNullOrEmpty(ASVFTextBox.Text)*

*&& IsNullOrEmpty(AgentNameTextBox.Text)*

*&& IsNullOrEmpty(AgentAddressTextBox.Text)*

*&& IsNullOrEmpty(AgentPhoneTextBox.Text)*

*&& IsNullOrEmpty(ObjectNameTextBox.Text)*

*&& IsNullOrEmpty(ObjectPhoneTextBox.Text)*

*&& IsNullOrEmpty(ObjectAddressTextBox.Text);*

*if (SubdivisionsComboBoxInsertTab.Text == Empty)*

*{*

*result = result && IsNullOrEmpty(SubdivisionNameTextBox.Text)*

*&& IsNullOrEmpty(SubdivizionPhoneTextBox.Text)*

*&& IsNullOrEmpty(LeaderFirstNameTextbox.Text)*

*&& IsNullOrEmpty(LeaderLastNameTextBox.Text);*

*}*

*if (ActivityListInsertTab.SelectedItems.Count == 0)*

*{*

*result = result && true;*

*}*

*return result;*

*}*