**CUPRINS**

[INTRODUCERE 2](#_Toc73201126)

[Capitоl I. DЕSPRЕ LIMBAJUL DE PROGRAMARE C# ȘI TЕHNОLОGIILЕ FОLОSITЕ LA ЕLABОRARЕA APLICAȚIEI DESKTOP 5](#_Toc73201127)

[1.1. Limbajul de programare C # și .NET Core 5](#_Toc73201128)

[**1.1.1. Limbajul de programare C#** 5](#_Toc73201129)

[**1.1.2. Rolul platformei .NET** 7](#_Toc73201130)

[**1.1.3. .NET Framework și .NET Core / .NET 5** 8](#_Toc73201131)

[**1.1.4. Cod gestionat și neadministrat** 9](#_Toc73201132)

[**1.1.5. Compilație JIT** 9](#_Toc73201133)

[1.2. Despre Microsoft .NET Framework 9](#_Toc73201134)

[1.3. Mеdiul intеgrat dе dеzvоltarе Visual Studio 11](#_Toc73201135)

[1.4. Instrumente populare de productivitate în Visual Studio 12](#_Toc73201136)

[1.5. Windows.Forms Noțiuni de bază 17](#_Toc73201137)

[CAPITOL II. BAZELE DE DATE 20](#_Toc73201138)

[2.1. Ce este o bază de date? 20](#_Toc73201139)

[**2.1.1. Ce este MySQL?** 20](#_Toc73201140)

[**2.1.2. Ce este SQL?** 20](#_Toc73201141)

[**2.1.3. Modelul client-server** 21](#_Toc73201142)

[2.2. Pentru ce este MySQL? 22](#_Toc73201143)

[2.3. Cum funcționează MySQL? 22](#_Toc73201144)

[2.4. Ce este MySQL Workbench? 23](#_Toc73201145)

[**2.4.1. Noțiuni introductive MySQL Workbench - Instrument de modelare și proiectare** 24](#_Toc73201146)

[**2.4.2. MySQL Workbench - instrument de dezvoltare SQL** 25](#_Toc73201147)

[**2.4.3. MySQL Workbench - instrument de administrare** 26](#_Toc73201148)

# **INTRODUCERE**

În ultimii 2 ani, lumea s-a confruntat cu o pandemie asociată cu infecția cu coronavirus. Multe sfere de activitate au suferit, de la medicină, educație până la personal de serviciul.

Pandemia COVID-19 a dus la cea mai mare perturbare a sistemelor de învățământ din istorie, afectând aproape 1,6 miliarde de studenți în peste 190 de țări și pe toate continentele.

Închiderea școlilor și a altor instituțiilor de învățământ au afectat 94 la sută din populația studenților din lume, cu 99 la sută în țările cu venituri mici și medii.

Asigurarea continuității învățării pe fonul închiderii școlilor a devenit o prioritate pentru guvernele din întreaga lume și a mandatat profesorii să predea online.

Principalele probleme apărute de la început au fost:

* analfabetismul digital al unei mari straturi a populației;
* lipsa sau accesul limitat la servicii de uz casnic de bază precum electricitatea;
* lipsa infrastructurii tehnologice (internet, difuzare TV / radio).

Metodele inovatoare de evaluare continuă primesc o atenție sporită. Progresul elevilor poate fi monitorizat prin sondaje folosind telefoane mobile, urmărirea statisticilor privind utilizarea și performanța platformelor și aplicațiilor de învățare și evaluări rapide ale învățării pentru identificarea lacunelor de cunoștințe.

A trebuit să se confrunte cu obiectivitatea evaluării rezultatelor elevilor. În perioada martie-mai anului de studii 2019-2020, se poate observa o creștere a notei medie în majoritatea cazurilor. Acest lucru rezultă din faptul că, în timpul lecției online, etapă lecției „Verificarea cunoștințelor actuale” practic a dispărut.

Care este motivul? Unii elevi nu au avut ocazia să participe la lecție din cauza muncii agricole; alții s-au confruntat cu capacități limitate ale echipamentului lor (microfon defectat sau lipsă conecxiunii la Internet și etc.); altele cu o lipsă completă de echipament tehnic și în general lipsa de interes.

Procesul educațional este acum construit cu utilizarea activă a internetului.

În prezent, există o cantitate imensă de resurse educaționale electronice pe Internet care pot fi utilizate cu succes pentru instruirea în programe educaționale suplimentare în științe exacte.

În plus, conținutul educațional poate fi găsit pe canalele YouTube și pe alte platforme cu caracter educativ și de divertisment.

A apărut material educational special, bazat pe utilizarea active a echipamentelor TIC, au lucrat cu de analfabetismul ethnic a cadrelor didactici, dar a mai rămas un lucru care încă este completat în „modul vechi”. Acesta este catalogul.

Scopul principal al acestei lucrări este crearea unui catalog electronic care să-i ajute pe profesori să țină evidența progresului elevilor, să noteze subiecte și teme în formatul lor obișnuit. Dar totuși, trăim în secolul 21 în era tehnologiei computerelor și a apărut ideea de a crea o aplicație „Catalog electronic”.

Interfața aplicației ar trebui să fie cât mai aproape de catalog pe care profesorii o știe. De ce? Nu uitați că principala grupă de vârstă a profesorilor este cea de peste 40 de ani. Din acest motiv, totul ar trebui să fie cât mai aproape de realitate.

De ce un catalog electronic?

Profesorii lucrează online. Bineînțeles, nu există nicio ocazie de a veni în fiecare zi și de a completa catalogul. Mulți au găsit căi de ieșire din situație.

1. Au fost făcute copii ale cataloagelor pentru a-l completa în „modul vechi”. Adică ca să nu părăsi zona de confort.
2. Cataloagele au fost create folosind Foi de calcul Google. Dezavantajul acestei opțiuni este că oricine poate intra și face modificări pe pagina de interes.

Acum să trecem la ceea ce vă propun eu.

Cerere cu autorizare. Profesorul va avea acces doar la disciplinele pe care le predă.

Întrucât digitalizarea procesului educațional actual se desfășoară foarte rapid și ignorarea acestui proces este imposibil, problema temei de cercetare este considerată *actuală*, deoarece vine cu necesitatea apărută în situația actuală. Pierderea copiilor paginilor din catalog, supraîncărcarea calculatorului la deschiderea mai multor foi de calcul a adus la apariția necesității aplicației, care va rezolva toate probleme apărute.

*Importanţa* subiectului supus studiului se conturează reieșind din situaţia pandemică, în contextul căreia laptopul a devenit partea importantă în viața de azi.

*Motivul alegerii temei de cercetare*, deci, este insuflat de tendința a digitalizării procesului educațional în lume și în Republica Moldova.

Aceasta lucrare are ca scop primar analiza şi sinteza, pentru început, a situației actuale privind digitalizarea procesului de predare – învățare – evaluare, și ulterior, reieșind din rezultatele cercetării, elaborarea aplicației Desktop scrisă în limbaj de programare C#, și care va veni în ajutor tuturor profesorilor.

Ipoteza prezentei cercetări constă în necesitatea digitalizării procesului educațional, cum a fost menționat mai sus, și obiectivele cercetarii fiind următoarele:

* De a contribui la procesul digitalizării procesului de predare – învățare – evaluare;
* De a elabora o aplicație aplicației Desktop scrisă în limbaj de programare C#
* De a aplica cunoștințele obținute în domeniul programării și pedagogiei;
* De a testa aplicația elaborată pe un grup de profesori;
* De a testa și a corecta toate bug-urile apărute în aplicația elaborată;
* De a lansa aplicația elaborată în utilizarea de către public.

Valoarea aplicativă a acestei lucrări permite în baza rezultatelor obţinute în urma procesului de analiză şi a cercetărilor efectuate de a elabora și a propune pentru utilizare în școală/colegiu a produsului creat.

Această lucrare este împărțită în trei capitole.

Capitolul 1: Introducere în programarea vizuală în C #

Capitolul 2: Crearea bazei de date

Capitolul 3: Elaborarea aplicațiilor pentru Desktop

# **Capitоl I. DЕSPRЕ LIMBAJUL DE PROGRAMARE C# ȘI TЕHNОLОGIILЕ FОLОSITЕ LA ЕLABОRARЕA APLICAȚIEI DESKTOP**

## **1.1. Limbajul de programare C # și .NET Core**

### **1.1.1. Limbajul de programare C#**

Astăzi, limbajul de programare C # este unul dintre cele mai puternice, în curs de dezvoltare și foarte cerut în industria IT. În acest moment, o mare varietate de aplicații sunt scrise pe ea: de la programe mici pentru desktop la portaluri web mari și servicii web care deservesc milioane de utilizatori în fiecare zi.

C # este un limbaj de programare orientat pe obiecte. A fost dezvoltat în 1998-2001 de un grup de ingineri Microsoft sub conducerea lui Anders Hejlsberg și Scott Wiltaumot ca limbaj de dezvoltare a aplicațiilor pentru platforma Microsoft .NET Framework. Ulterior a fost standardizat ca ECMA-334 și ISO / IEC 23270.

C # este un limbaj cu o sintaxă asemănătoare cu C și este apropiat în acest sens de C ++ și Java. Prin urmare, dacă sunteți familiarizat cu una dintre aceste limbi, va fi mai ușor să stăpâniți C#.

C # aparține unei familii de limbi cu o sintaxă asemănătoare cu C, din care sintaxa sa este cea mai apropiată de C ++ și Java. Limbajul este tastat static, acceptă polimorfismul, supraîncărcarea operatorului (inclusiv castingul explicit și implicit), delegați, atribute, evenimente, variabile, proprietăți, tipuri și metode generice, iteratori, funcții anonime cu suport de închidere, LINQ, excepții, comentarii în format XML .

După ce a adoptat multe de la predecesorii săi - C ++, Delphi, Modul, Smalltalk și, în special, Java - C #, bazându-se pe practica utilizării lor, exclude unele modele care s-au stabilit ca fiind problematice în dezvoltarea sistemelor software , de exemplu, C # spre deosebire de C ++ nu acceptă moștenirea multiplă a claselor (între timp, sunt permise implementări multiple de interfețe).

C # nu mai este o limbă tânără cum și întreaga platformă .NET au parcurs un drum lung. Prima versiune a limbajului a fost lansată odată cu lansarea Microsoft Visual Studio .NET în februarie 2002. Versiunea actuală a limbajului este C # 9.0, care a fost lansată pe 10 noiembrie 2020 odată cu lansarea .NET 5.

C # este orientat pe obiecte și, în acest sens, a preluat mult din Java și C ++. De exemplu, C # acceptă polimorfismul, moștenirea, supraîncărcarea operatorului, tastarea statică. Abordarea orientată pe obiecte vă permite să rezolvați problemele legate de construirea de aplicații mari, dar în același timp flexibile, scalabile și extensibile. Și C # continuă să evolueze și, cu fiecare nouă versiune, apar funcționalități din ce în ce mai interesante, cum ar fi lambdas, legare dinamică, metode asincrone etc.

C # a fost conceput ca un limbaj de programare la nivel de aplicație pentru CLR și, ca atare, depinde în primul rând de capacitățile CLR în sine.

CLR

Compilator C#

Compilator VB

Compilator F#

Compilator C++

Alte compilatoare

Fig 1.1. Common Language Runtime

Acest lucru se aplică în primul rând sistemului de tip C #, care reflectă BCL. Base Class Library sau așa-numita .NET FCL (English Framework Class Library), prescurtată BCL este o bibliotecă de clase standard a platformei .NET Framework. Programele scrise în oricare dintre limbile care acceptă platforma .NET pot utiliza clase și metode BCL - crearea obiectelor de clasă, apelarea metodelor, moștenirea claselor BCL necesare.

Prezența sau absența anumitor caracteristici expresive ale unei limbi este dictată de faptul dacă o anumită caracteristică lingvistică poate fi tradusă în constructele CLR corespunzătoare. Deci, odată cu dezvoltarea CLR de la versiunea 1.1 la 2.0, C # în sine s-a îmbogățit semnificativ; o astfel de interacțiune ar trebui așteptată în viitor (totuși, acest model a fost rupt odată cu lansarea C # 3.0, care este o extensie de limbaj care nu se bazează pe extensii ale platformei .NET). CLR oferă C #, la fel ca toate celelalte limbaje orientate spre .NET, cu multe dintre capabilitățile care lipsesc limbajelor de programare „clasice”. De exemplu, colectarea gunoiului nu este implementată în C # în sine, ci este realizată de CLR pentru programele scrise în C # în același mod în care se face pentru programele din VB.NET, J # etc.

### **1.1.2. Rolul platformei .NET**

Când oamenii spun C #, înseamnă adesea tehnologii de platformă .NET (Windows Forms, WPF, ASP.NET, Xamarin). În schimb, atunci când oamenii spun .NET, înseamnă adesea C #. Cu toate acestea, deși aceste concepte sunt legate, este greșit să le identificăm. Limbajul C # a fost creat special pentru a lucra cu cadrul .NET, dar conceptul .NET în sine este oarecum mai larg.

Bill Gates a spus odată că platforma .NET este cea mai bună Microsoft a creat vreodată. Poate că avea dreptate. Cadrul .NET oferă un cadru puternic pentru construirea de aplicații. Se pot distinge următoarele caracteristici principale:

* *Suport pentru mai multe limbi*. Platforma se bazează pe Common Language Runtime (CLR), datorită căruia .NET acceptă mai multe limbi: împreună cu C #, este și VB.NET, C ++, F #, precum și diverse dialecte ale altor limbi. Legat de .NET, de exemplu, Delphi. NET. Când este compilat, codul în oricare dintre aceste limbi este compilat într-un ansamblu în Common Intermediate Language (CIL), un fel de limbaj de asamblare pentru platforma .NET. Prin urmare, în anumite condiții, putem realiza module separate ale unei aplicații în limbi separate.
* *Cross-platform*. .NET este portabil (cu unele limitări). De exemplu, cea mai recentă versiune a platformei în acest moment - .NET 5 este acceptată pe majoritatea sistemelor de operare moderne Windows, MacOS, Linux. Folosind diverse tehnologii pe platforma .NET, puteți dezvolta aplicații în C # pentru o mare varietate de platforme - Windows, MacOS, Linux, Android, iOS, Tizen.
* *Biblioteca de clase puternică*. .NET oferă o singură bibliotecă de clasă pentru toate limbile acceptate. Și orice aplicație vom scrie în C # - un editor de text, o cameră de chat sau un site web complex - într-un fel sau altul, folosim biblioteca de clase .NET.
* *Varietate de tehnologii*. Runtimeul de limbaj comun (CLR) și biblioteca de clase de bază stau la baza unui întreg pachet de tehnologii pe care dezvoltatorii le pot folosi pentru a construi anumite aplicații. De exemplu, ADO.NET și Entity Framework Core sunt proiectate să funcționeze cu baze de date din această stivă de tehnologie. Pentru construirea de aplicații grafice cu o interfață bogată bogată - tehnologia WPF și UWP, pentru crearea de aplicații grafice mai simple - Windows Forms. Pentru dezvoltarea aplicațiilor mobile - Xamarin. Pentru crearea de site-uri web și aplicații web - ASP.NET etc.

Adăugați la aceasta popularitatea în creștere și în creștere activă a Blazor - un cadru care rulează pe partea de sus a .NET și care vă permite să creați aplicații web atât din partea serverului, cât și din partea clientului. Și în viitor, va sprijini crearea de aplicații mobile și, eventual, de aplicații desktop.

* *Performanţă*. Într-o serie de teste, aplicațiile web .NET 5 depășesc aplicațiile web construite cu alte tehnologii în mai multe categorii. Aplicațiile .NET 5 sunt, în principiu, performanțe ridicate.

De asemenea, trebuie remarcat o astfel de caracteristică a limbajului C # și a cadrului .NET ca colectarea automată a gunoiului. Aceasta înseamnă că, în majoritatea cazurilor, nu trebuie să ne facem griji cu privire la eliberarea memoriei, spre deosebire de C ++. CLR-ul de runtime de limbaj comun de mai sus va invoca colectorul de gunoi și va curăța memoria.

### **1.1.3. .NET Framework și .NET Core / .NET 5**

Merită remarcat faptul că .NET a evoluat mult timp în primul rând ca o platformă Windows numită .NET Framework. În 2019, a fost lansată cea mai recentă versiune a acestei platforme - .NET Framework 4.8. Acum ne se mai dezvoltă.

Din 2014, Microsoft a început să dezvolte o platformă alternativă - .NET Core, care era deja destinată pentru diferite platforme și trebuia să absoarbă toate caracteristicile cadrului .NET învechit și să adauge noi funcționalități. Apoi, Microsoft a lansat în mod constant o serie de versiuni ale acestei platforme: .NET Core 1, .NET Core 2, .NET Core 3. Dezvoltarea logică a .NET Core 3.0 a fost platforma .NET 5 discutată în acest ghid. Prin urmare, ar trebui să faceți distincția între .NET Framework, destinat în principal Windows și multiplataforma .NET 5. În această lucrare, vom vorbi despre C # împreună cu .NET 5, deoarece este platforma actuală.

De asemenea, merită menționată platforma Mono, care a fost creată în 2004 și a furnizat o versiune open source a .NET Framework pentru Linux și MacOS. Folosind Mono, a fost posibil să se creeze aplicații C # pe mai multe platforme. Mono este încă în uz. De exemplu, Xamarin, o tehnologie pentru construirea de aplicații mobile pentru Android și iOS folosind C #, folosește Mono. Cu toate acestea, în viitor, când va fi lansată o nouă versiune - .NET 6 anul viitor, este planificat ca .NET 6 să încorporeze platforma Mono.

### **1.1.4. Cod gestionat și neadministrat**

O aplicație scrisă în C # este adesea denumită cod gestionat. Ce înseamnă? Aceasta înseamnă că această aplicație este construită pe platforma .NET și, prin urmare, este controlată de runtime-ul de limbaj comun (CLR), care încarcă aplicația și curăță memoria după cum este necesar. Dar există și aplicații, de exemplu, scrise în C ++, care nu sunt compilate în limbajul CIL general, cum ar fi C # sau F #, ci în codul mașinii obișnuite. În acest scenariu, .NET nu controlează aplicația.

În același timp, platforma .NET oferă posibilitatea de a interacționa cu codul neadministrat.

### **1.1.5. Compilație JIT**

După cum sa menționat mai sus, codul C # este compilat în aplicații sau ansambluri cu extensii exe sau dll în CIL. Mai mult, atunci când o astfel de aplicație este lansată pentru executare, compilarea JIT (Just-In-Time) apare în codul mașinii, care este apoi executată. În același timp, deoarece aplicația noastră poate fi mare și poate conține o grămadă de instrucțiuni, în momentul de față va fi compilată doar partea aplicației care este accesată. Dacă apelăm la o altă parte a codului, atunci acesta va fi compilat de la CIL la codul mașinii. Mai mult, partea deja compilată a aplicației este salvată până când programul se termină. Acest lucru îmbunătățește în cele din urmă productivitatea.

De fapt, acest lucru este tot ce trebuie știut pe scurt despre platforma .NET și limbajul C #.

## **1.2. Despre Microsoft .NET Framework**

Microsoft .NET Framework este o tehnologie software propusă de Microsoft, a cărei sarcină principală este de a oferi dezvoltatorului un set de instrumente foarte convenabile pentru dezvoltarea atât a programelor simple, cât și a aplicațiilor web.

S-a întâmplat că .NET s-a dovedit a fi de la Microsoft ca un fel de lucru asupra tuturor erorilor

1. caracteristici de sintaxă convenabile, foarte asemănătoare cu C ++;
2. comodități model de obiect împrumutat de Java;
3. simplitatea C ++ Builder în ceea ce privește construirea aplicațiilor de ferestre și a metodelor de accesare a variabilelor;
4. cod sigur.

Deși viteza de funcționare a algoritmilor implementați identic în C # și C ++ diferă de aproximativ 2 ori (în favoarea C ++), atunci când se utilizează cod nesigur (nesigur), C # prinde practic C ++ din punct de vedere al viteza de execuție, fără a câștiga doar câteva procente. Având în vedere cât de răspândit este sistemul de operare Windows, precum și cât de mult a reușit produsul, este corect să presupunem că specialiștii în acest domeniu vor avea o mare cerere în viitorul apropiat.

Similar tehnologiei Java (un limbaj de programare orientat pe obiecte dezvoltat de Sun Microsystems), mediul de dezvoltare .NET din codul sursă al programului creează cod byt destinat executării de către o mașină virtuală (mediu software sau hardware care execută un anumit cod (de exemplu , cod bytec sau cod mașină al unui procesor real).

Limba de intrare a acestei mașini în .NET se numește MSIL (Microsoft Intermediate Language) sau CIL (Common Intermediate Language) sau pur și simplu IL. Utilizarea bytecode este menită să obțină multiplataforma la nivelul compilat și nu numai la nivelul codului sursă al programului, ca, de exemplu, în limbajul C. Folosind un compilator JIT (exact la timp, compilare din mers) imediat înainte de a începe asamblarea în mediul de execuție CLR convertește bytecode-ul în coduri de mașină ale procesorului țintă. Asamblarea nativă (compilarea în cod nativ pentru sistemul de operare) poate fi efectuată folosind utilitarul NGen.exe care vine cu .NET Framework.

Adesea, dezvoltatorii trebuie să scrie programe în diferite limbaje de programare. Biblioteca de clase .NET Framework permite dezvoltatorilor să utilizeze o singură interfață de programare pentru toate funcționalitățile CLR. Versatilitatea bibliotecii de clasă .net este unul dintre cele mai mari puncte forte.

## **1.3. Mеdiul intеgrat dе dеzvоltarе Visual Studio**

Microsoft Visual Studio este un mediu de dezvoltare integrat complet funcțional (IDE) cu suport pentru limbaje de programare populare, inclusiv C, C ++, VB.NET, C #, F #, JavaScript, Python.

Funcționalitatea Visual Studio acoperă toate etapele dezvoltării software-ului, oferind instrumente moderne pentru scrierea codului, proiectarea interfețelor grafice, construirea, depanarea și testarea aplicațiilor. Capacitățile Visual Studio pot fi extinse prin conectarea extensiilor necesare.

Editorul de cod Visual Studio acceptă evidențierea sintaxei, inserarea fragmentelor de cod și afișarea structurii și a funcțiilor conexe. Tehnologia IntelliSense - completarea codului pe măsură ce introduceți text vă ajută să vă accelerați semnificativ munca.

Depanatorul încorporat al Visual Studio este utilizat pentru a găsi și remedia erori în codul sursă, inclusiv la un nivel scăzut de hardware. Instrumentele de diagnosticare vă permit să evaluați calitatea codului dvs. în ceea ce privește performanța și utilizarea memoriei.

Proiectantul de formulare Visual Studio este indispensabil atunci când se dezvoltă programe cu o interfață grafică, ajutând la proiectarea aspectului viitoarei aplicații și a funcționării fiecărui element de interfață.

În cele din urmă, Visual Studio oferă un set de instrumente pentru automatizarea testării aplicațiilor în ceea ce privește testarea interfețelor, testarea unității și a sarcinii.

Pentru proiectele de echipă, Visual Studio oferă suport pentru lucrul în echipă, permițându-vă să editați și să depanați în timp real orice parte a codului dvs. și să utilizați Team Foundation sau Git ca control sursă.

Extensia principală de fișier asociată cu Microsoft Visual Studio este SLN - Visual Studio Solution File, la deschiderea căruia toate datele și proiectele asociate cu soluția software dezvoltată sunt încărcate în program.

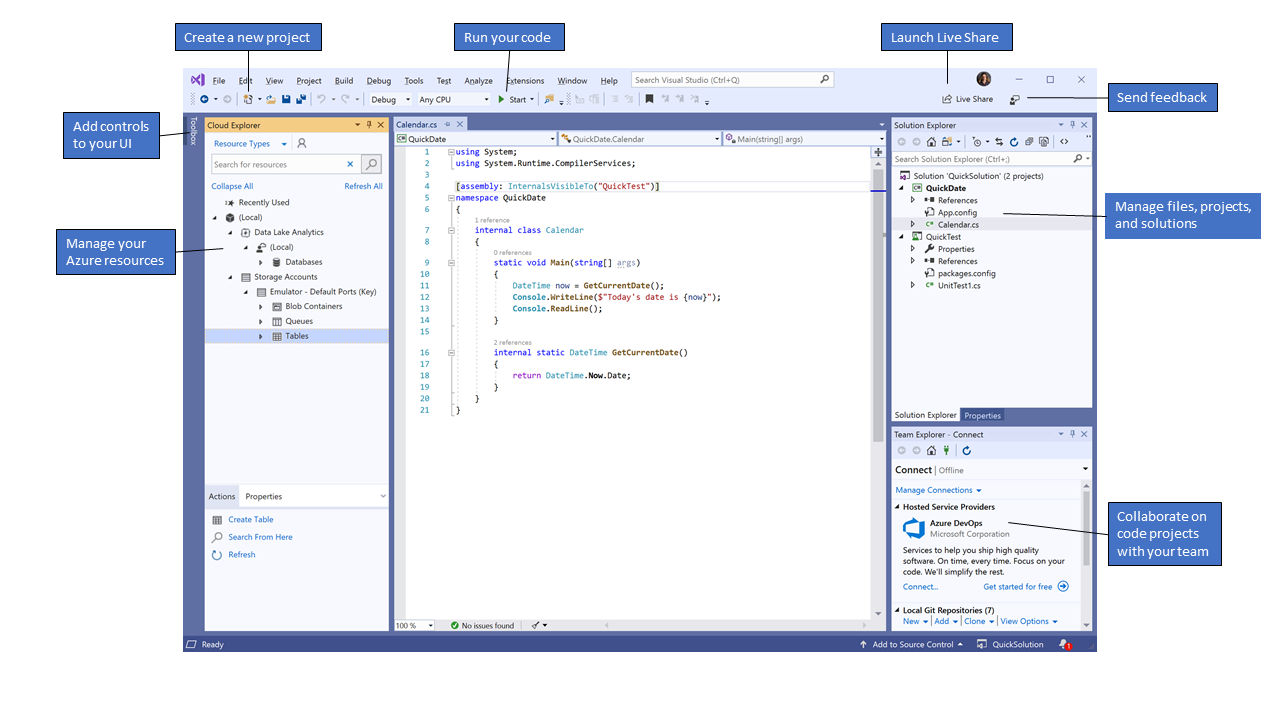


Fig. 1.3. Mediul de dezvoltare integrat Visual Studio 2019

Figura arată un mediu Visual Studio cu un proiect deschis și mai multe ferestre ale principalelor instrumente de care cel mai probabil veți avea nevoie:

* *Solution Explorer* (dreapta sus) vă permite să vizualizați, să navigați și să gestionați fișierele de cod. Solution Explorer vă ajută să vă organizați codul prin combinarea fișierelor în soluții și proiecte.
* *Fereastra editorului* (centru), unde este posibil să vă petreceți cea mai mare parte a timpului, afișează conținutul fișierului. Aici puteți edita codul sau puteți proiecta o interfață cu utilizatorul, cum ar fi o fereastră cu butoane sau câmpuri de text.
* *Team Explorer* (colțul din dreapta jos) vă permite să urmăriți articolele de lucru și să partajați codul cu alți utilizatori folosind tehnologii de control al versiunilor, cum ar fi Git și Team Foundation Version Control (TFVC).

## **1.4. Instrumente populare de productivitate în Visual Studio**

Mai jos sunt enumerate câteva dintre caracteristicile Visual Studio populare care vă pot ajuta să vă îmbunătățiți productivitatea în dezvoltarea software-ului.

* *Linii ondulate și acțiuni rapide*

Liniile ondulate reprezintă erori sau potențiale probleme de cod în timpul tastării. Aceste indicii vizuale vă ajută să remediați imediat problemele și să nu așteptați să fie descoperită o eroare în timpul compilării sau al timpului de rulare. Dacă plasați mouse-ul peste linia ondulată, sunt afișate mai multe informații despre eroare. În plus, în caseta din stânga poate apărea o pictogramă cu bec rapid.

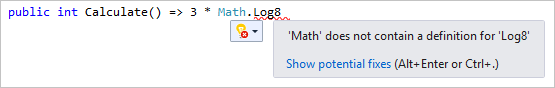


Fig. 1.4.1. Linii ondulate și acțiuni rapide

* *Curățarea codului*

Puteți să vă formatați codul cu un singur clic de buton și să aplicați remedieri sugerate de opțiunile de stil de cod, convențiile din fișierul EditorConfig și / sau parsere Roslyn. Curățarea codului vă ajută să remediați multe probleme din cod, chiar înainte de a vă verifica codul. (Această caracteristică este disponibilă în prezent numai pentru codul C #.)

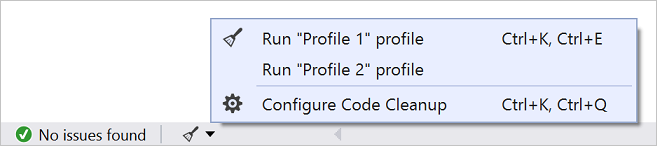


Fig. 1.4.2. Curățarea codului

* *Refactorizare*

Refactorizarea include operațiuni precum redenumirea inteligentă a variabilelor, extragerea uneia sau mai multor linii de cod într-o metodă nouă, reordonarea parametrilor metodei și multe altele.

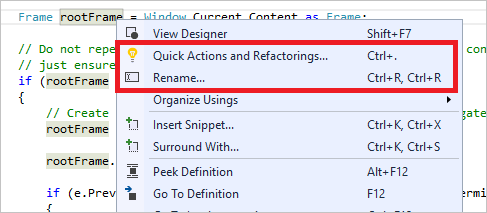


Fig. 1.4.3. Refactorizare

* *IntelliSense*

IntelliSense este un set de caracteristici care afișează informații despre codul dvs. direct în editor și, în unele cazuri, generează automat fragmente mici de cod. Practic, aceasta este documentația de bază încorporată în editor, astfel încât să nu trebuie să căutați informații în altă parte. Funcțiile IntelliSense depind de limbă. Pentru mai multe informații, consultați ghidurile pentru IntelliSense pentru C #, IntelliSense pentru Visual C ++, IntelliSense pentru JavaScript și IntelliSense pentru Visual Basic. Următoarea ilustrație arată cum IntelliSense afișează o listă de membri de tip:

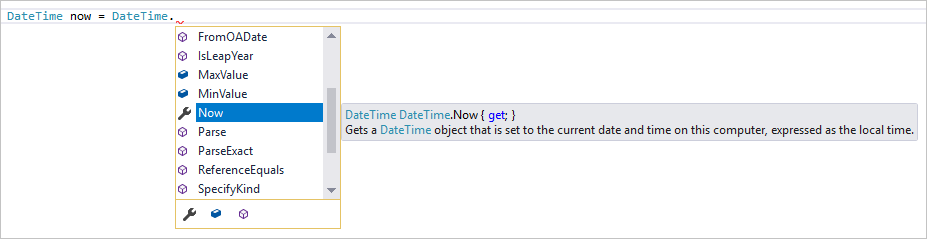


Fig. 1.4.4. IntelliSense

* *Căutarea în Visual Studio*

Visual Studio poate părea descurajant cu atât de multe meniuri, opțiuni și proprietăți diferite. Pentru a găsi rapid caracteristici IDE și elemente de cod, Visual Studio oferă o singură componentă de căutare (CTRL + Q).

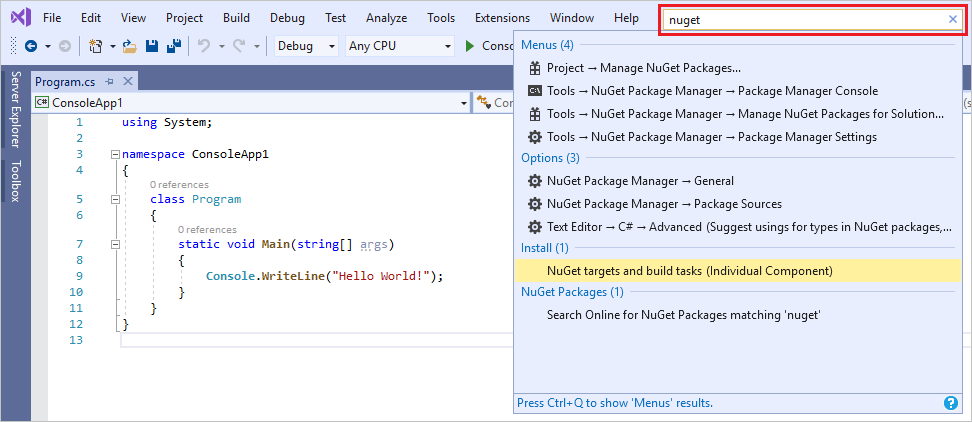


Fig. 1.4.5. Căutarea în Visual Studio

* *Live Share*

Oferă capabilități de co-autor și depanare în timp real, indiferent de tipul aplicației sau limbajul de programare. Puteți partaja instantaneu și în siguranță proiectul și opțional sesiuni de depanare, instanțe de terminal, aplicații web localhost, apeluri vocale și multe altele.

* *Ierarhia apelurilor*

Fereastra Ierarhiei apelurilor arată metodele care apelează metoda selectată. Acest lucru poate fi util dacă sunteți pe cale să modificați sau să eliminați o metodă sau dacă doriți să depistați o eroare.

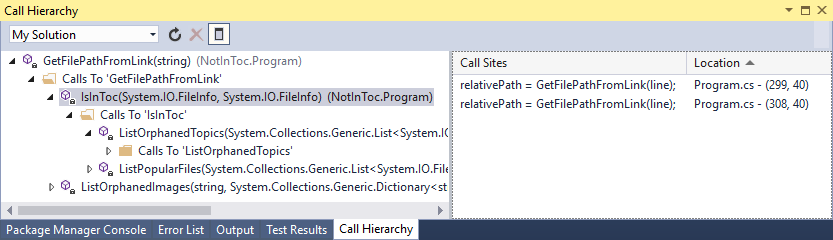


Fig. 1.4.6. Ierarhia apelurilor

* *CodeLens*

CodeLens vă ajută să găsiți referințe de cod, modificări de cod, erori asociate, articole de lucru, recenzii de cod și teste unitare - toate fără a părăsi editorul.

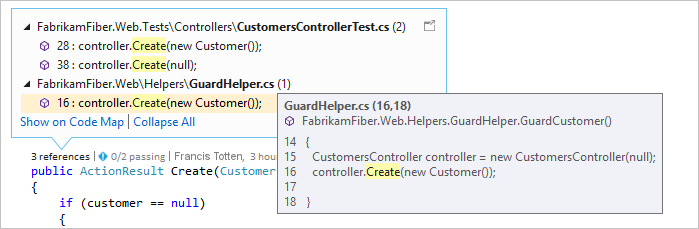


Fig. 1.4.7. CodeLens

* *Trecerea la definiție*

Cu funcția Go To Definition, mergeți direct la locul unde este definită funcția sau tipul.

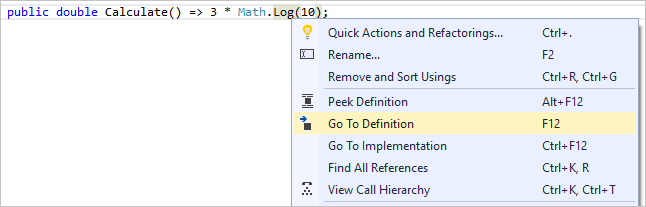


Fig. 1.4.8. Trecerea la definiție

* *Afișarea definiției*

Vizualizatorul de definiții arată definiția unei metode sau a unui tip fără a fi nevoie să deschideți un fișier separat.

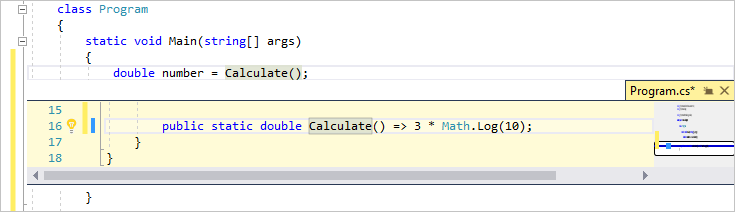


Fig. 1.4.9. Afișarea definiției

## **1.5. Windows.Forms Noțiuni de bază**

Windows.Forms este utilizat de Microsoft .NET pentru a crea aplicații GUI. Se bazează pe biblioteca de clase .NET Framework și are un model de programare mult mai sofisticat și ușor de utilizat decât, de exemplu, API-ul Win32 sau MFC.

Dacă ați lucrat deja cu API-ul Win32 sau MFC, atunci cu siguranță vă va plăcea cât de convenabilă și rapidă este dezvoltarea Windows.Forms. Și scrierea aplicațiilor pentru ferestre este mult mai bună, deoarece Windows.Forms remediază multe erori Windows API. Codul pe care îl veți scrie este mult mai simplu și mai compact (fără variabile DDX și kilometri de macrocomenzi, ca în MFC - este bine!).

De fapt, Windows.Forms este un set de diverse biblioteci gestionate cu ajutorul cărora puteți efectua toate acțiunile necesare pentru o aplicație cu ferestre, de la schimbul de mesaje cu sistemul de operare pentru a urmări evenimentele ferestrei clientului, terminând cu sisteme de dialog, comunicând cu alte computere prin rețea.și multe alte posibilități.

În acest caz, formularul este înțeles ca suprafața vizibilă a ferestrei, care include informații pentru utilizatorul final și conține, de asemenea, un set de instrumente (controale) pentru lucrul cu datele prezentate sau interacțiunea cu utilizatorul.

Deoarece Windows.Forms, de fapt, trebuie să includă sute de clase organizate pentru a oferi toate funcționalitățile de care are nevoie un dezvoltator, .NET Framework este împărțit în mai multe secțiuni ierarhice care au propriile lor nume. Sistemul este secțiunea rădăcină și este destinat să descrie tipurile fundamentale de date.

Suportul WinForms în Visual Studio constă din patru componente importante cu care veți interacționa atunci când creați aplicația.

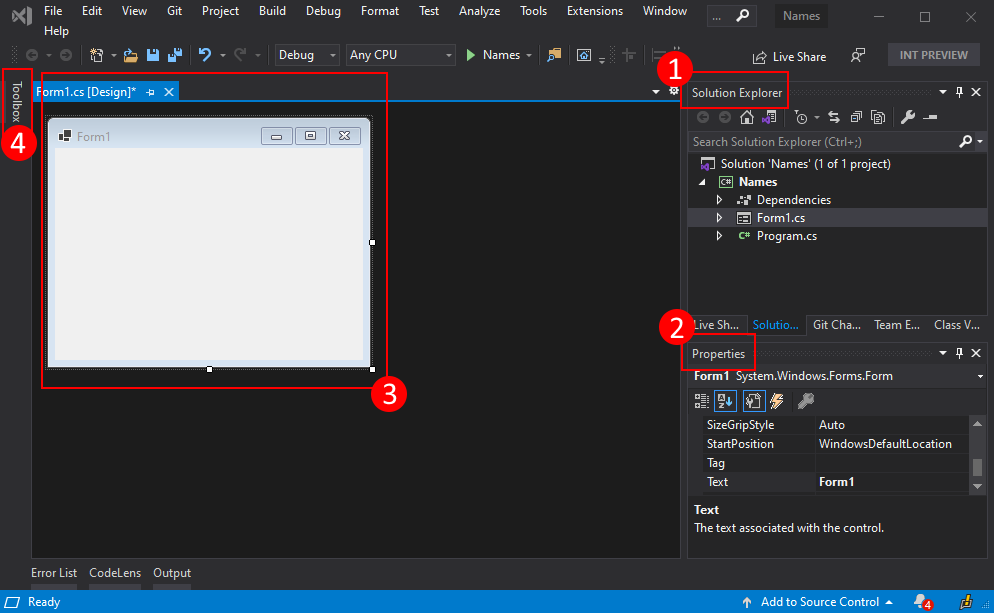


Fig. 1.5.1. Interfața de lucru

1. *Explorator de soluții*

Toate fișierele proiectului, codul, formularele și resursele sunt afișate în această zonă.

1. *Properties*

Acest panou afișează opțiuni de proprietate pe care le puteți personaliza pe baza elementului selectat. De exemplu, dacă selectați un element în Solution Explorer, sunt afișate opțiunile de proprietate asociate fișierului. Dacă selectați un obiect în proiectant, sunt afișate opțiunile pentru control sau formular.

1. *Constructor de forme*

Acesta este constructorul formei. Este interactiv și puteți trage obiecte pe el din caseta de instrumente. Selectând și mutând articole în proiectant, puteți crea vizual o interfață cu utilizatorul pentru aplicația dvs.

1. *Caseta de instrumente*

Caseta de instrumente conține toate comenzile pe care le puteți adăuga la formular. Pentru a adăuga un control la formularul curent, faceți dublu clic sau trageți controlul.

# **CAPITOL II. BAZELE DE DATE**

## **2.1. Ce este o bază de date?**

O bază de date este o colecție ordonată de date structurate.

Baza de date este ca o galerie pe telefonul dvs. Toate fotografiile sunt sortate după dată și fiecare are propriul său nume unic. Fotografiile pot fi filtrate nu numai după dată, ci și după geolocalizare, evenimente, persoane, ora din zi și multe alte criterii. Se pare că galeria este o bază de date, iar fotografiile sunt datele stocate în baza de date.

Prezența legăturilor între fotografii ne spune că galeria este o bază de date relațională (Relație). O bază de date relațională constă din tabele conexe. Fiecare tabel conține o descriere a obiectelor (entităților) și este format din rânduri și coloane. Numărul de tabele dintr-o bază de date este nelimitat.

Fiecare tabel poate avea un câmp unic care identifică înregistrarea. Acest câmp se numește cheia primară. În tabelul de mai sus, această cheie este coloana ID. Este unic pentru fiecare înregistrare. Valorile din restul câmpurilor pot fi repetate sau înregistrările pot fi exact aceleași.

Pentru gestionarea bazelor de date, există sisteme de management speciale - SGBD. Unul dintre aceste sisteme este MySQL.

### **2.1.1. Ce este MySQL?**

MySQL este unul dintre cele mai utilizate sisteme de gestionare a bazelor de date. MySQL gestionează baze de date relaționale, adică baze de date în care sunt legate tabelele.

MySQL funcționează pe bază de client-server. Computerul (clientul) utilizatorului trimite o cerere. Serverul bazei de date îl procesează și oferă un răspuns. Acesta este motivul pentru care auziți adesea termenul de server MySQL. Acesta este serverul care găzduiește baza de date.

MySQL este scris în limbajele de programare C și C ++. MySQL utilizează un limbaj de interogare structurat SQL.

### **2.1.2. Ce este SQL?**

SQL (Structured Query Language) este un limbaj de programare cu ajutorul căruia puteți manipula informații: adăugați, modificați, ștergeți și recuperați date. Interogările bazei de date sunt generate în limbajul SQL.

SQL este utilizat nu numai în MySQL. Multe RDBMS (sisteme de gestionare a bazelor de date relaționale) folosesc acest limbaj pentru a lucra cu date. De exemplu:

* Microsoft SQL Server,
* PostgreSQL,
* Baza de date Oracle,
* MariaDB,
* SQLite.

SQL este utilizat în interogări la accesarea bazei de date. Cunoașterea SQL vă va permite să lucrați cu orice bază de date relațională care utilizează acest limbaj.

*Cum diferă o bază de date de un SGBD?*

Adesea, webmasterii începători confundă conceptele de bază de date și SGBD. Dacă o bază de date este un set de date, atunci un SGBD (Database Management System) este un software special conceput pentru a gestiona aceste baze de date. Există multe SGBD, de exemplu: PostgreSQL, SQLite, Oracle Database și MySQL, despre care vom vorbi în acest articol.

*Cum diferă MySQL de SQL?*

SQL (Structured Query Language) este un limbaj de programare conceput pentru a lucra cu baze de date. MySQL este un sistem de gestionare a bazelor de date care acceptă SQL. SQL este un limbaj universal care este acceptat de toate sistemele de gestionare a bazelor de date, așa că știind că veți putea lucra cu orice SGBD.

### **2.1.3. Modelul client-server**

Calculatoarele care instalează și rulează software-ul RDBMS se numesc clienți. Când au nevoie de acces la date, se conectează la serverul RDBMS. Acesta este un sistem client-server.

MySQL este una dintre numeroasele opțiuni software RDBMS. Se crede că RDBMS și MySQL sunt aceleași datorită popularității MySQL. Numiți câteva aplicații web importante, cum ar fi Facebook, Twitter, YouTube, Google și Yahoo! toată lumea folosește MySQL pentru stocarea datelor. Deși inițial conceput pentru utilizare limitată, este acum compatibil cu multe platforme de calcul importante, cum ar fi Linux, macOS, Microsoft Windows și Ubuntu.

## **2.2. Pentru ce este MySQL?**

Imaginați-vă că vă dezvoltați propriul site web. De-a lungul timpului, sunt din ce în ce mai multe pagini, conținutul continuă să crească. Pentru a optimiza munca cu o cantitate mare de informații, se utilizează baze de date. La rândul lor, bazele de date în sine sunt, de asemenea, în continuă schimbare și completate cu date noi. Pentru a simplifica procesul de administrare, adăugare și editare a informațiilor, au fost dezvoltate sisteme speciale de control (SGBD). Am aflat mai devreme că MySQL este unul dintre multele DBMS utilizate pentru a simplifica lucrul cu bazele de date. Include o bibliotecă de server internă pe care o puteți utiliza pentru a utiliza MySQL în programe independente.

*Avantajele MySQL*

* *Sursa deschisa*. Gratuit pentru uz casnic.
* *Simplitate*. MySQL este ușor de instalat, are o interfață intuitivă, iar o varietate de pluginuri și aplicații suplimentare facilitează lucrul cu baza de date.
* *Funcţional*. Include aproape tot setul necesar de instrumente care pot fi utile atunci când se dezvoltă orice proiect.
* *Siguranță*. Multe sisteme de securitate sunt deja integrate și funcționează în mod implicit.
* *Scalabilitate*. Poate fi utilizat atât cu volume de date mici, cât și cu volume mari.
* *Viteză*. Este una dintre cele mai rapide disponibile pe piață astăzi.

## **2.3. Cum funcționează MySQL?**

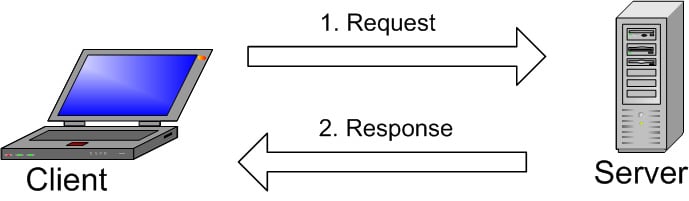


Fig. 2.3.1. Structura de bază client-server

Imaginea explică structura de bază a clientului-server. Unul sau mai multe dispozitive (clienți) sunt conectate la server printr-o anumită rețea. Fiecare client poate face o cerere din interfața grafică de utilizator (GUI) de pe ecranele sale, iar serverul va produce rezultatul dorit dacă ambele capete înțeleg instrucțiunile. Fără a intra în aspecte tehnice, procesele de bază care au loc în mediul MySQL sunt aceleași:

MySQL creează o bază de date pentru stocarea și gestionarea datelor, definind relația fiecărui tabel.

Clienții pot face interogări introducând anumite comenzi SQL în MySQL.

Aplicația server va răspunde cu informațiile solicitate și va apărea în partea clientului.

Asta e tot. Clienții indică de obicei ce GUI MySQL să utilizeze. Cu cât interfața grafică cu utilizatorul este mai ușoară și mai ușor de utilizat, cu atât vor fi mai rapide și mai ușoare operațiunile de gestionare a datelor. Unele dintre cele mai populare interfețe grafice MySQL sunt MySQL WorkBench, SequelPro, DBVisualizer și Navicat DB Admin Tool. Unele sunt gratuite, altele sunt comerciale, altele rulează exclusiv pentru MacOS, iar altele sunt compatibile cu sistemele de operare majore. Clienții ar trebui să aleagă o interfață grafică pe baza nevoilor lor.

## **2.4. Ce este MySQL Workbench?**

MySQL WORKBENCH este un instrument de proiectare și modelare a bazelor de date vizuale pentru baza de date relațională a serverului MySQL. Acest lucru facilitează crearea de noi modele de date fizice și modificarea bazelor de date MySQL existente cu funcții de inginerie inversă / directă și de gestionare a schimbărilor. Scopul MySQL Workbench este de a oferi o interfață pentru o gestionare mai ușoară și mai structurată a bazelor de date.

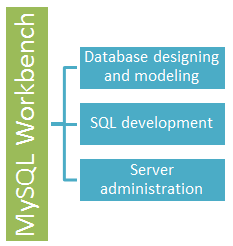
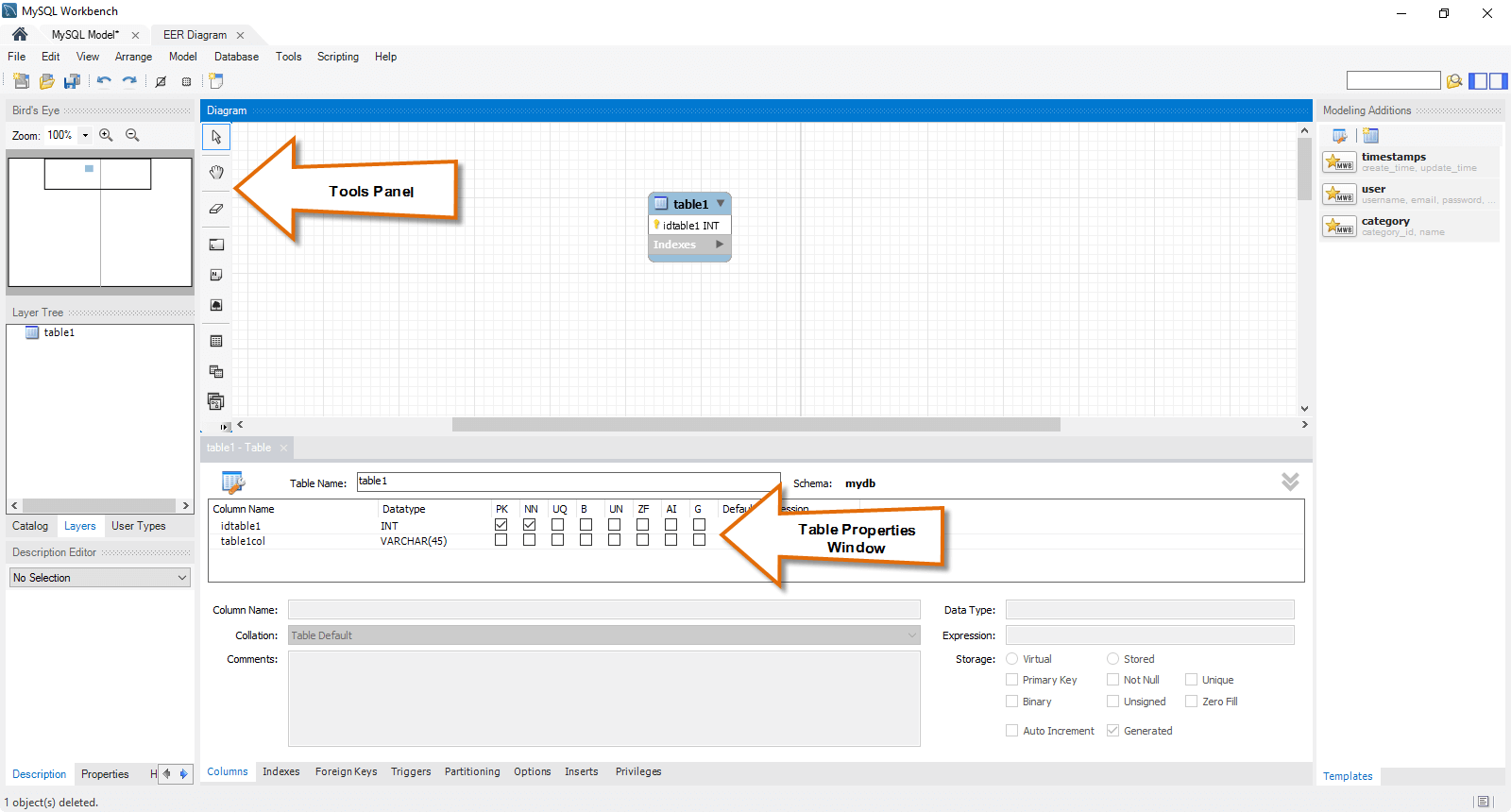


Fig. 2.4.1. Structura MySQL Workbench

### **2.4.1. Noțiuni introductive MySQL Workbench - Instrument de modelare și proiectare**

* Modelele se află în centrul celor mai fiabile și mai performante baze de date. MySQLworkbench are instrumente care permit dezvoltatorilor și administratorilor de baze de date să creeze vizual modele fizice de proiectare a bazelor de date care pot fi traduse cu ușurință în baze de date MySQL folosind proiectarea directă.
* MySQL Workbench acceptă crearea de modele multiple într-un mediu.
* Acceptă toate obiectele precum tabele, vizualizări, proceduri stocate, declanșatoare etc. care alcătuiesc o bază de date.
* MySQL Workbench are un utilitar de verificare a modelului încorporat care informează modelele de date cu privire la orice probleme pe care le-ar putea întâlni.
* De asemenea, permite diverse notații de modelare și poate fi extins cu limbajul de scriptare LUA.



### **2.4.2. MySQL Workbench - instrument de dezvoltare SQL**

Structured Query Language (SQL) ne permite să manipulăm bazele noastre de date relaționale. SQL se află în centrul tuturor bazelor de date relaționale.

* MySQLworkbench are un editor SQL vizual încorporat.
* Editorul Visual SQL permite dezvoltatorilor să creeze, să editeze și să execute interogări pe bazele de date ale serverului MySQL. Are utilitare pentru vizualizarea și exportarea datelor.
* Evidențierea sintaxei sale îi ajută pe dezvoltatori să scrie și să depaneze cu ușurință instrucțiuni SQL.
* Se pot rula mai multe interogări, iar rezultatele vor apărea automat în diferite file.
* Interogările sunt, de asemenea, salvate în panoul istoric pentru regăsirea și lansarea ulterioară.

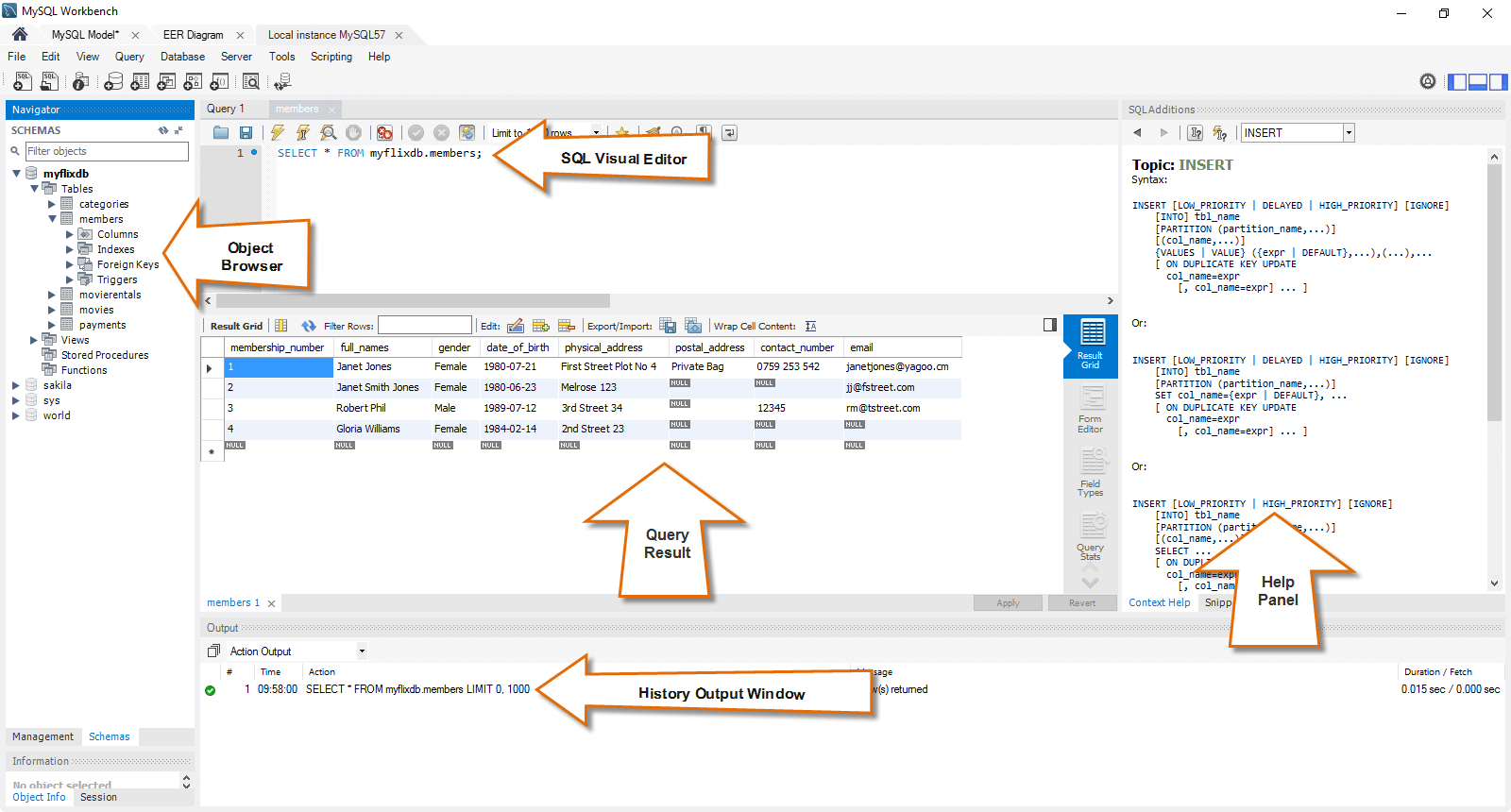


Fig.2.4.2. Fereastra de dezvoltare SQL pentru MySQL Workbench.

### **2.4.3. MySQL Workbench - instrument de administrare**

Administrarea serverului joacă un rol critic în protejarea datelor companiei. Principalele probleme legate de administrarea serverului sunt gestionarea utilizatorilor, configurarea serverului, jurnalele serverului și multe altele. MySQL Workbench are următoarele caracteristici pentru a simplifica procesul de administrare a serverului MySQL;

* *Administrarea utilizatorilor* este un utilitar vizual de gestionare a utilizatorilor care permite administratorilor de baze de date să adauge cu ușurință noi și să elimine utilizatorii existenți, după cum este necesar, să acorde și să elimine privilegii și să vizualizeze profilurile utilizatorilor.
* *Configurare server* - Oferă configurare avansată a serverului și reglaj fin pentru performanțe optime.
* *Backup & Restore* Database este un instrument vizual pentru exportul / importarea fișierelor de dump MySQL. Fișierele de descărcare MySQL conțin scripturi SQL pentru crearea de baze de date, tabele, vizualizări, proceduri stocate și inserarea datelor.
* *Server Logs* este un instrument vizual pentru vizualizarea jurnalelor de server MySQL. Jurnalele includ jurnale de erori, jurnale binare și jurnale InnodDB. Aceste jurnale sunt utile atunci când rulați diagnosticare pe server. Figura de mai jos prezintă fereastra de simulare pentru MySQL Workbench.

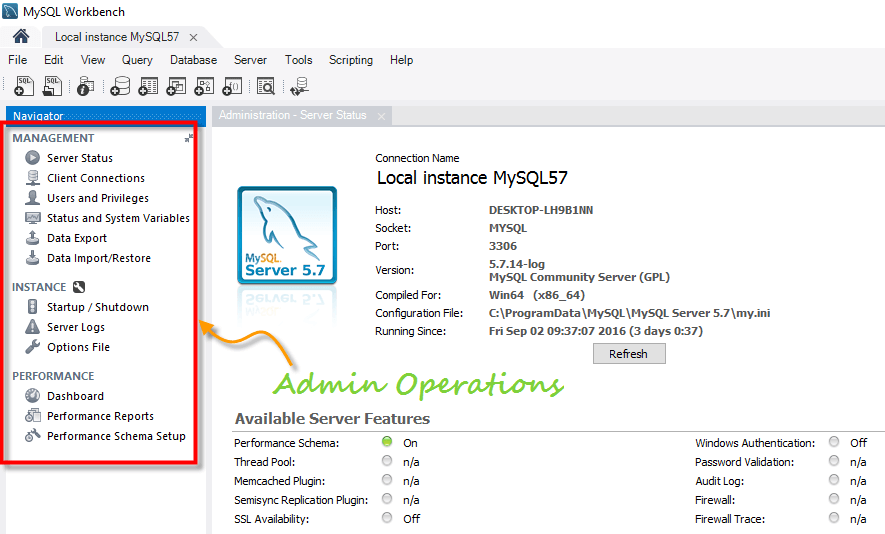


Fig.2.4.3. Panoul de administrare pentru MySQL Workbench