КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА ОПТИКО-МЕХАНІЧНИЙ КОЛЕДЖ

Циклова комісія програмування та інформаційних технологій

**КУРСОВА РОБОТА**

**з дисципліни «Інструментальні засобі візуального програмування»**

на тему “Автоматизація роботи приймальної комісії навчального закладу”

|  |  |  |
| --- | --- | --- |
| ЗАХИЩЕНО З ОЦІНКОЮ  Національна шкала \_\_\_\_\_\_\_\_\_\_\_\_  Кількість балів \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Керівник  \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Цапко Д.В.  (підпис) |  | ВИКОНАВ |
| студент 3-го курсу гр. ІПЗ-32 спеціальність 121 «Інженерія програмного забезпечення» |
| Пушня Олександр Сергійович  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (підпис виконавця) |

Члени комісії \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Колесник Т.Г

(підпис)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Заболотний В.О.

(підпис)

м. Київ — 2020

**Зміст**

[**Реферат** 3](#_Toc41490899)

[**Умовні позначення** 4](#_Toc41490900)

[**ВСТУП** 5](#_Toc41490901)

[**Постановка задачі** 6](#_Toc41490902)

[**Аналіз і характеристика** 7](#_Toc41490903)

[**Дослідження програм-аналогів, визначення їх переваг та недоліків** 8](#_Toc41490904)

[**Інструментальні засоби створення проекту.** 10](#_Toc41490905)

[1.1Опис С# 10](#_Toc41490906)

[1.2 Опис MySQL 16](#_Toc41490907)

[**Статистика** 30](#_Toc41490908)

[**Розробка програмного забезпечення** 32](#_Toc41490909)

[Код програми 36](#_Toc41490910)

[**Висновок** 41](#_Toc41490911)

[**Список використаної літератури** 42](#_Toc41490912)

# **Реферат**

Курсова робота складається із вступу, 9 пунктів, 1 висновку, списку використаних джерел (7) та коду програми. Робота містить 16 рисунків та 4 таблиці. Загальний обсяг роботи становить 44 сторінки, основний текст викладено на 35 сторінках.

# **Умовні позначення**

БД-База даних

ВНЗ –вищий навчальний заклад

# **ВСТУП**

В цьому проекті описується автоматизована система для пришвидшення продуктивності роботи прийому абітурієнтів до коледжу та полегшення роботи для приймальної комісії. Представлено функції програми і засоби за допомогою яких було зроблено проект.

Дана система мінімізує затрати часу та сили на роботу для навчального персоналу коледжу .Інформація в програмі завжди вірна і актуальна ,що забезпечує пришвидшенню прийняття рішень для зарахування до навчального закладу.

Дає змогу абітурієнтам та їхнім батькам перевіряти стан процесу зарахування до навчального закладу .

Приймальна комісія вищого навчального закладу (далі – Приймальна комісія) створюється для проведення прийому студентів на навчання. Термін повноважень Приймальної комісії становить один рік.

Приймальна комісія працює в умовах демократичності, прозорості та відкритості відповідно до законодавства України, у тому числі Умов прийому до вищих навчальних закладів та положення про Приймальну комісію.

Положення про Приймальну комісію затверджується керівником вищого навчального закладу та повинно передбачати кількісний склад комісії, необхідний для прийняття рішень. Рішення приймальної комісії вищого навчального закладу оформлюються протоколами, які підписуються головою і відповідальним секретарем Приймальної комісії.

Склад Приймальної комісії затверджується наказом керівника вищого навчального закладу, який є головою комісії.

Голова Приймальної комісії несе персональну відповідальність за виконання покладених на Приймальну комісію завдань і здійснення нею своїх функцій.

Відповідальний секретар Приймальної комісії та його заступники призначаються головою Приймальної комісії з числа провідних науково-педагогічних (педагогічних) працівників вищого навчального закладу.

# **Постановка задачі**

Автоматизовані системи набули широкого використання у різних сферах життя. Не становить виключення і освітня сфера, а саме вступ до навчальних закладів. Процес прийому абітурієнтів є достатньо трудомістким, тому питання спрощення та оптимізації даного процесу є дуже актуальним. Одним із шляхів вирішення даної задачі є впровадження автоматизованої системи приймальної комісії навчальних закладів за допомогою сучасних інформаційних технологій. Автоматизація дозволить пришвидшити прийом документів, обробку інформації, забезпечити надійність її збереження та відновлення втрачених даних. Приймальні комісії отримають можливість автоматизувати весь процес, починаючи з прийому документів до формування остаточних списків студентів на зарахування. Основною метою є забезпечення якісних показників роботи програми: створення зручного графічного інтерфейсу користувача, розробка супроводжувальної документації, засобів для розгортання програмної системи в інформаційному середовищі користувачів

**Основні можливості:**

* ведення електронних карток абітурієнтів
* обробка інформації про поточний стан прийому документів від абітурієнтів
* облік результатів вступних іспитів

Використання системи автоматизації суттєво впливає на конкурентні переваги навчального закладу на ринку освітніх послуг. Підвищуєшвидкість і якість роботи з контингентом учнів і співробітників, забезпечує надійність і цілісність управління в цілому.

Створення даного програмного продукту насамперед передбачає створення гнучкого і простого у використанні інтерфейсу користувача, що дасть змогу використовувати програму без спеціальних вмінь та навичок.

# **Аналіз і характеристика**

**Анкета абітурієнта**

У програмі створюється та заповнюється анкета абітурієнта, де зберігаються основні особові дані та результати складання іспитів.

**Робота з документацією**

Працюючи з відомостями, оператор має змогу реєструвати бали з сертифікатів УЦОЯО, оцінки з атестату чи диплому, сортувати абітурієнтів за набраним балом, груповим способом проставляти рекомендацію до зарахування, назву навчальної групи та номер наказу про зарахування.

Автоматизована система формує велику кількість документів у вигляді списків та таблиць, серед яких такі стандартизовані звіти, як журнал реєстрації абітурієнтів, а також аналітичні, статистичні звіти та усі Міністерські форми звітів.

**Переваги використання програми**

* Економія часу. Не доведеться заповнювати особові дані кожного абітурієнта вручну
* підрахунок загальної кількості абітурієнтів.
* Можливість легко та без зусиль редагувати внутрішню інформацію про абітурієнта. А також додавати та видаляти їх.
* Економія місця. Всі дані зберігаються в електронному варіанті

Сама система складається з бази даних Database1 на MySQL .База даних складається з однієї таблиці – Student.

# **Дослідження програм-аналогів, визначення їх переваг та недоліків**

**АС «Деканат»**

АС «Деканат» – це програмно-технологічний комплекс управління навчальним процесом закладу освіти, призначений для організації роботи методистів та зменшення кількості документації на паперових носіях.

В процесі роботи АС «Деканат», до персональних даних студента додаються інші поля, що відображають поточний стан навчання студента згідно навчального плану. Тому доцільно використовувати дану програму у комплексі з АС «Приймальна комісія».

Дані зарахованих студентів пакетнами переносяться з АС «Приймальна комісія». Це позбавляє працівників деканатів від необхідності вводити руками дані студентів. Таким чином, персональна картка для кожного студента вводиться лише один раз – під час подання документів до приймальної комісії, далі картка лише набуває додаткових даних.

**Переваги**

* Високий рівень надійності та безпеки бази даних, регулярне оновлення і постійна сервісна підтримка;
* Управління та розподіл прав доступу до системи з використанням апаратно-програмних засобів;
* Індивідуальне навчання по роботі з системою;
* У системі використовується потужна та наднадійна база даних Oracle.

**Високий рівень безпеки**

Особливістю АСУ «ВНЗ» є унікальна запатентована система захисту інформації від несанкціонованого доступу. Правовою основою створення системи захисту в АСУ «ВНЗ» є закон України «Про захист інформації в автоматизованих системах».

* Вхід у систему тільки зі «Smart Card Reader» та карткою доступу;
* Управління користувачами в системі;
* Розподіл даних між працівниками;
* Передача зашифрованих даних;
* Журналізація дій користувача

В порівняні з розроблюваним проектом ця програма має набагато більше переваг і не має видимих недоліків. В програмі реалізовано все ,що потрібно для швидкої та легкої роботи працівникам ВНЗ.

Єдиним рішенням цієї задачі є автоматизація діяльності вузу і створення інформаційного середовища, яке з одного боку, надає доступ до необхідної користувачам актуальної, несуперечливої і повної інформації, з іншої – є необхідним інструментом діяльності співробітників вузу і навчання студентів. Таке інформаційне середовище дозволяє управляти процесами, даними і людьми, тобто інформаційне середовище можна розглядати і з погляду підтримки життєдіяльності університету і як інструмент управління. На даний момент, велика частина вузів використовує «часткову» автоматизацію, тобто комп’ютеризовані лише окремі бізнес-процеси. Це веде до дублювання і втрати частини інформації, відсутності даних для оперативного аналізу, втрати часу на обслуговування і підтримку застаріваючих технологій, високого ступеню залежності від людського чинника.

# **Інструментальні засоби створення проекту.**

**Опис мов програмування**

**Мо́ва програмува́ння** (англ. *Programming language*) — це штучна мова, створена для передачі команд машинам, зокрема комп'ютерам. Мови програмування використовуються для створення програм, котрі контролюють поведінку машин, та запису алгоритмів.

Більш строге визначення: **мова програмування** — це система позначень для опису алгоритмів та структур даних, певна штучна формальна система, засобами якої можна виражати алгоритми. Мову програмування визначає набір лексичних, синтаксичних і семантичних правил, що задають зовнішній вигляд програми і дії, які виконує виконавець (комп'ютер) під її управлінням.

З часу створення перших програмованих машин було створено понад дві з половиною тисячі мов програмування. Щороку їх кількість поповнюється новими. Деякими мовами вміє користуватись тільки невелике число їх власних розробників, інші стають відомі мільйонам людей. Професійні програмісти зазвичай застосовують в своїй роботі декілька мов програмування.

# 1.1Опис С#

**C#** - це мова програмування створена спеціально для роботи у середовищі Microsoft .NET Framework.

Мова C# була розроблена з урахуванням сильних і слабких особливостей інших мов, зокрема Java і C++. Специфікація мови C# була написана Андерсом Гейлсбергом, Скотом Вілтамутом та Пітером Гольде. Андерс Хейлсберг відомий у світі програмування як автор компілятора Turbo Pascal і лідер команди, яка створила Delphi.

[Синтаксис](https://uk.wikipedia.org/wiki/%D0%A1%D0%B8%D0%BD%D1%82%D0%B0%D0%BA%D1%81%D0%B8%D1%81) C# близький до [С++](https://uk.wikipedia.org/wiki/%D0%A1%2B%2B) і [Java](https://uk.wikipedia.org/wiki/Java). [Мова](https://uk.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B2%D0%B0) має строгу статичну типізацію, підтримує [поліморфізм](https://uk.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D1%96%D0%BC%D0%BE%D1%80%D1%84%D1%96%D0%B7%D0%BC_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)), перевантаження операторів, вказівники на функції-члени класів, атрибути, події, властивості, [винятки](https://uk.wikipedia.org/wiki/%D0%9E%D0%B1%D1%80%D0%BE%D0%B1%D0%BA%D0%B0_%D0%B2%D0%B8%D0%BD%D1%8F%D1%82%D0%BA%D1%96%D0%B2), коментарі у форматі [XML](https://uk.wikipedia.org/wiki/XML). Перейнявши багато що від своїх попередників — мов [С++](https://uk.wikipedia.org/wiki/%D0%A1%2B%2B), [Object Pascal](https://uk.wikipedia.org/wiki/Object_Pascal), [Модула](https://uk.wikipedia.org/w/index.php?title=%D0%9C%D0%BE%D0%B4%D1%83%D0%BB%D0%B0&action=edit&redlink=1) і [Smalltalk](https://uk.wikipedia.org/wiki/Smalltalk) — С#, спираючись на практику їхнього [використання](https://uk.wikipedia.org/wiki/%D0%92%D0%B8%D0%BA%D0%BE%D1%80%D0%B8%D1%81%D1%82%D0%B0%D0%BD%D0%BD%D1%8F) виключає деякі моделі, що зарекомендували себе як проблематичні при розробці програмних систем, наприклад [множинне спадкування](https://uk.wikipedia.org/wiki/%D0%9C%D0%BD%D0%BE%D0%B6%D0%B8%D0%BD%D0%BD%D0%B5_%D1%81%D0%BF%D0%B0%D0%B4%D0%BA%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F) класів (на відміну від C++).

**Історія розвитку**

Назва мови C# трактується як наступне покоління розвитку С++, а символ # – символізує “++++”. Спочатку в назві фігурував дієз - “♯”(англійською sharp), однак через відсутність цього символу на клавіатурі, використовується знак для позначення номеру “#”. Загалом назву мови можна позначати обома символами.

Розробка мови C# розпочалася в грудні 1998 року, та готувався до випуску разом з продуктами групи Millenium. Проект мав назву COOL(**C**-style **O**bject **O**riented **L**anguage), та розроблявся як аналог Java від компанії Oracle. C# був анонсований, широкому загалу, в 2000 році, як основна мова платформи Microsoft .Net Framework. В цьому ж році з’явилася перша загальнодоступна бета-версія.

Перша фінальна версія мови програмування C# була випущена в 2002 році разом з середовищем інтегрованої розробки програмного забезпечення **Visual Studio .Net**.

Подібно до Java, C# отримав наступні концепції:

* віртуальна машина – платформа .Net виконує програму подібно до віртуальної машини від Java;
* байт-код – програмний код компілюється в проміжкову мову MSIL(Microsoft Intermediate Language), а вже потім перетворюється на машинну мову, в залежності від платформи на якій запускається програма;
* керований код – оскільки програми, написані на C#, виконуються виключно у віртуальному середовищі CLR(Common Language Runtime), це дає змогу контролювати виконання програми та у будь який момент зупинита її, а також контролювати використання пам’яті програмою, за необхідності - збільшувати, чи видаляти частини пам’яті, які використовує програма.

Нововведенням C# стала можливість легшої взаємодії, порівняно з мовами-попередниками, з кодом програм, написаних на інших мовах, що є важливим при створенні великих проектів. Якщо програми на різних мовах виконуються на платформі .NET, .NET бере на себе клопіт щодо сумісності програм (тобто типів даних, за кінцевим рахунком).

Станом на сьогодні C# визначено флагманською мовою корпорації Microsoft, бо вона найповніше використовує нові можливості .NET. Решта мов програмування, хоч і підтримуються, але визнані такими, що мають спадкові прогалини щодо використання .NET. Рядок в C # є посилальним типом.

Разом з тим C# передусім розроблялась як мова програмування прикладного рівня для CLR і тому вона залежить, перш за все, від можливостей самої CLR. Це стосується, перш за все, системи типів C#. Присутність або відсутність тих або інших виразних особливостей мови диктується тим, чи може конкретна мовна особливість бути трансльована у відповідні конструкції CLR. Так, з розвитком CLR від версії 1.1 до 2.0 значно збагатився і сам C#; подібної взаємодії слід чекати і надалі. Проте ця закономірність була порушена з виходом C# 3.0, що є розширеннями мови, що не спираються на розширення платформи .NET. CLR надає C#, як і всім іншим .NET-орієнтованим мовам, багато можливостей, яких позбавлені «класичні» мови програмування. Наприклад, збірка сміття не реалізована в самому C#, а проводиться CLR для програм, написаних на C# точно так, як і це робиться для програм на VB.NET, J#.

**Синтаксис**

Синтаксис мови C# схожий на інші мови C-стилів, такі як C, C ++ і Java. Зокрема:

* Крапки з комою використовуються для позначення кінця строки коду.
* Фігурні дужки використовуються для угруповання операторів.
* Строки коду зазвичай групуються в методи (функції), методи в класи і класи в простір імен.
* Змінні присвоюються з використанням знака рівності, а порівнюються з використанням двох послідовних знаків рівності.
* Квадратні дужки використовуються для роботи з масивами, як для їх оголошування, так і для отримування значення за заданим індексом в одному з них.

**Властивості**

C# надає [властивості](http://fitm.nusta.edu.ua/mediawiki/index.php?title=En:Property_(programming)&action=edit&redlink=1) як [синтаксичного цукру](http://fitm.nusta.edu.ua/mediawiki/index.php?title=%D0%A1%D0%B8%D0%BD%D1%82%D0%B0%D0%BA%D1%81%D0%B8%D1%87%D0%BD%D0%B8%D0%B9_%D1%86%D1%83%D0%BA%D0%BE%D1%80&action=edit&redlink=1) для загального шаблону, в якому пара методів, [accessor (getter) і mutator (setter)](http://fitm.nusta.edu.ua/mediawiki/index.php?title=En:Mutator_method&action=edit&redlink=1) інкапсулює операції по одному атрибуту класу. Не потрібно писати надлишкові сигнатури методів для реалізацій геттера / сетера і до цієї властивості можна отримати доступ, використовуючи синтаксис атрибутів, а не більш докладні виклики методів.

**Методи та функції**

Методи можуть бути недійсними або можуть повертати щось на зразок рядка, цілого, подвійного, десяткового, float і bool. Якщо метод недійсний, це означає, що метод не повертає жодного типу даних.

Подібно C++, і на відміну від Java, програмісти на C # повинні використовувати ключове слово virtual, щоб дозволити перевизначати методи підкласами.

Методи розширення в C# дозволяють програмістам використовувати статичні методи, як якщо б вони були методами з таблиці методів класу, дозволяючи програмістам додавати методи до об'єкта, який, на їхню думку, повинен існувати на цьому об'єкті і його похідних.

Динамічний тип dynamic допускає прив'язку методу під час виконання, що дозволяє використовувати JavaScript-подібні виклики методів і склад часу виконання.

У C# є підтримка строго типізованих покажчиків функцій через delegate ключового слова. Подібно псевдо-C ++ - *signal* і *slot* фрейма Ct, C# має семантику, спеціально пов'язану з подіями стилю публікації-підписки, хоча C# використовує делегати для цього.

C # пропонує Java-подібні синхронізовані synchronized виклики методів через атрибут [MethodImpl (MethodImplOptions.Synchronized)] і підтримує взаємовиключні блокування за допомогою блокування ключових слів.

**MenuStrip**

Меню надають користувачам функціональні можливості, зберігаючи команди, згруповані за загальною темою.

Елемент управління MenuStrip був представлений у версії 2,0 .NET Framework. За допомогою елемента управління MenuStrip можна легко створювати меню, аналогічні тим, які знаходяться в Microsoft Office.

Елемент управління MenuStrip підтримує багатодокументний інтерфейс (MDI) і злиття меню, спливаючі підказки і переповнення. Ви можете зробити його зручнішим для використання і легкість для читання меню, додавши ключі доступу, поєднання клавіш, прапорці, зображення і роздільники.

Елемент управління MenuStrip замінює і розширює функціональні можливості елемента управління MainMenu; Однак елемент управління MainMenu зберігається для забезпечення зворотної сумісності і використання в майбутньому при виборі.



**TabControl**

TabControl містить сторінки вкладок, які представлені TabPage об'єктами, які додаються за допомогою властивості TabPages. Порядок сторінок вкладок в цій колекції відображає порядок, в якому вкладки відображаються в елементі управління.

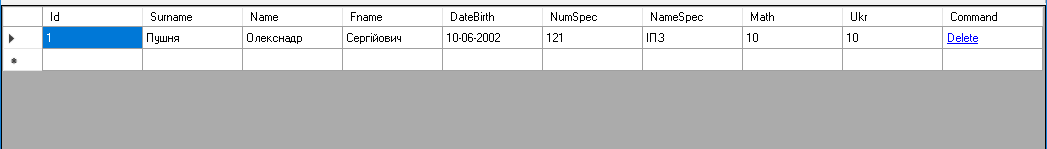
Користувач може змінити поточний TabPage, клацнувши одну з вкладок в елементі управління.



**DataGridView**

Елемент управління DataGridView можна використовувати для виведення даних з базовим джерелом даних або без нього. Без вказівки джерела даних можна створювати стовпці і рядки, що містять дані, і додавати їх безпосередньо в DataGridView за допомогою властивостей Rows і Columns. Можна також використовувати колекцію Rows для доступу до DataGridViewRow об'єктів і властивості DataGridViewRow.Cells для прямого читання або запису значень осередків. Item [String, Int32] індексатор також надає прямий доступ до осередків.

Елемент управління DataGridView надає можливість налаштувати таблицю для відображення даних. Клас DataGridView дозволяє налаштовувати осередки, рядки, стовпці і кордони за допомогою таких властивостей, як DefaultCellStyle, ColumnHeadersDefaultCellStyle, CellBorderStyleі GridColor.



# 1.2 Опис MySQL

**MySQL** — вільна система керування реляційними базами даних.

MySQL був розроблений компанією «ТсХ» для підвищення швидкодії обробки великих баз даних. Ця система керування базами даних (СКБД) з відкритим кодом була створена як альтернатива комерційним системам. MySQL з самого початку була дуже схожою на mSQL, проте з часом вона все розширювалася і зараз MySQL — одна з найпоширеніших систем керування базами даних. Вона використовується, в першу чергу, для створення динамічних веб-сторінок, оскільки має чудову підтримку з боку різноманітних мов програмування.

**Історія**

MySQL виникла як спроба застосувати mSQL до власних розробок компанії: таблиць, для яких використовувалися ISAM — підпрограми низького рівня для індексного доступу до даних. У результаті був вироблений новий SQL-інтерфейс, але API-інтерфейс залишився в спадок від mSQL. Звідки походить назва «MySQL» — достеменно не відомо. Розробники дають два варіанти: або тому, що практично всі напрацювання компанії починалися з префікса My, або на честь дівчинки на ім'я My, дочки Майкла Монті Віденіуса, одного з розробників системи.

Логотип MySQL у вигляді дельфіна носить ім'я «Sakila». Він був обраний з великого списку запропонованих користувачами «імен дельфіна». Ім'я «Sakila» було відправлено Open Source-розробником Ambrose Twebaze.

В січні-лютому 2008 Sun Microsystems придбала розробника системи керування базами даних MySQL за $1 млрд. Після поглинання у 2009 році Sun Microsystems компанією Oracle Corporation MySQL стала власністю Oracle.

За час розвитку під орудою Oracle дедалі більше відокремлює MySQL від спільноти і робить процес розробки все менш прозорим. Наприклад, повернута практика поставки власницьких розширених функцій в Enterprise-версії MySQL, спостерігається приховування інформації про вразливості, зі складу виключений тестовий набір, закритий доступ до більшої частини системи відстеження помилок та припинено публікація згрупованого логу змін, що дозволяє судити про прив'язку патчів до конкретних змін.

MySQL — характеризується великою швидкістю, стійкістю і простотою використання, була розроблена для підвищення швидкодії обробки великих баз даних.

Вихідні коди сервера компілюються на багатьох платформах. Найповніше можливості сервера виявляються в UNІХ-системах, де є підтримка багатоканальності, що підвищує продуктивність системи в цілому.

Для некомерційного використання MySQL є безкоштовним.

Можливості сервера MySQL:

1. простота у встановленні та використанні;
2. підтримується необмежена кількість користувачів, що одночасно працюють із БД;
3. кількість рядків у таблицях може досягати 50 млн.;
4. висока швидкість виконання команд;
5. наявність простої і ефективної системи безпеки.

**Основні переваги MySQL.**Серед основних переваг MySQL відзначають наступні:

* Масштабованість. MySQL може підтримувати роботу БД значних розмірів, що підтверджують її реалізації в Yahoo!, Google,HP, Associated Press. Згідно документації, що додається до MySQL, деякі БД, що використовуються компанією MySQL АВ (розробником MySQL), зберігають до 50 млн. записів.
* Переносність. MySQL працює на різних платформах, серед яких Unix, Linux, Windows, OS/2, Solaris, Mac OC. Окрім того, MySQL працює на різних платформах.
* Зв’язаність. MySQL має мережеву структуру. До MySQL можна одержувати доступ із будь-якої точки Internet кільком користувачам одночасно. MySQL має цілий ряд програмних інтерфейсів додатків (Application Programming Interface –API ), які дозволяють встановлювати з’єднання з MySQL із додатків, написаних на таких мовах як С, С++, Perl, PHP, Java, Python.
* Бeзпека. MySQL має систему контроля доступу до даних, забезпечує шифрування даних при передаванні.
* Швидкість функціонування.
* Зручність експлуатації. MySQL досить зручно встановлюється та реалізується, легко адмініструється.
* Відкритий код.

Сервер MySQL може працювати зручно на робочому столі або ноутбуці, поруч з іншими Вашими додатками, веб-серверами і так далі, вимагаючи невеликої або ніякої уваги. Ви можете скорегувати налаштування, щоб використовувати в своїх інтересах всю пам'ять, процесор і доступну пропускну здатність введення / виведення

MySQL Server був спочатку розвинений, щоб обробити великі бази даних набагато швидше, ніж існуючі рішення і успішно використовувався в дуже вимогливих виробничих середовищах протягом декількох років. Хоча і перебуває в постійній розробці, MySQL сьогодні пропонує багатий і корисний набір функцій. Зв'язок, швидкість і безпеку роблять MySQL надзвичайно відповідним для того, щоб отримати доступ до баз даних в Інтернеті.

MySQL складається з двох частин: серверної і клієнтської.

Cтруктура MySQL трирівнева: бази даних - таблиці - записи. Бази даних і таблиці MySQL фізично представляються файлами з розширеннями frm, MYD, MYI. Логічно таблиця являє собою сукупність записів. А записи - це сукупність полів різного типу. Ім'я бази даних MySQL унікально в межах системи, а таблиці - в межах бази даних, поля - в межах таблиці. Один сервер MySQL може підтримувати одразу декілька баз даних, доступ до яких може розмежовуватись логіном і паролем.

**Поля і їхні типи в MySql**

База даних з точки зору MySQL (і деяких інших СУБД) – це звичайний каталог, що містить бінарні файли певного формату – таблиці. Таблиці складаються із записів, а записи, у свою чергу, складаються з полів. Поле має два атрибути – ім'я та тип.

Тип поля може бути:

* числовим цілим;
* числовим дійсним;
* рядковим;
* бінарним;
* дата й час;
* переліком і множиною.

Цілочисельні типи даних

|  |  |
| --- | --- |
| **TINYINT** | -128...+127 |
| **SMALLINT** | -32768...+32767 |
| **MEDIUMINT** | -8 388 608...+8 388 607 |
| **INT** | -2 147 483 648...+2 147 483 647 |
| **BIGINT** | -9 223 372 036 854 775 808...+9 223 372 036 854 775 807 |

Дійсні числа

Числові дійсні типи записуються у вигляді:

ТИП (ДОВЖИНА, ЗНАКИ) [UNSIGNED]

Тут і надалі у квадратних дужках необов'язкові параметри. ДОВЖИНА - це кількість позицій, у яких буде розташоване число, а ЗНАКИ – це кількість знаків після десяткової крапки, які будуть ураховуватися. Якщо присутній модифікатор UNSIGNED, знак числа враховуватися не буде.

|  |  |
| --- | --- |
| **FLOAT** | Невелика точність |
| **DOUBLE** | Подвійна точність |
| **REAL** | Те ж, що й DOUBLE |
| **DECIMAL** | Дробове число, що зберігається у вигляді рядка |
| **NUMERIC** | Те ж, що й DECIMAL |

Рядкові типи

Будь-який рядок – це масив символів. У більшості випадків застосовується тип VARCHAR або просто CHAR, що дозволяє зберігати рядки, що містять до 255 символів. У дужках після типу вказується довжина рядка:

VARCHAR(48);

CHAR(73);

Якщо 255 символів для вашого завдання недостатньо, можна використовувати інші типи, наприклад, TEXT.

|  |  |
| --- | --- |
| **TINYTEXT** | Максимальна довжина 255 символів |
| **TEXT** | Максимальна довжина 65535 символів (64 Кб) |
| **MEDIUMTEXT** | Максимальна довжина 16 777 215 символів |
| **LONGTEXT** | Максимальна довжина 4 294 967 295 символів |

Бінарні типи даних

Бінарні типи даних також можна використовувати для зберігання тексту. У такому випадку при пошуку буде враховуватися регістр символів. До того ж, будь-який текстовий тип можна перетворити в бінарний, указавши модифікатор BINARY:

VARCHAR(30) BINARY;

|  |  |
| --- | --- |
| **TINYBLOB** | Максимум 255 символів |
| **BLOB** | Максимум 65535 символів |
| **MEDIUMBLOB** | Максимум 16 777 215 символів |
| **LONGBLOB** | Максимум 4 294 967 295 |

Типи дати та часу

|  |  |
| --- | --- |
| **DATE** | Дата у форматі РРР-ММ-ДД |
| **TIME** | Час у форматі ГГ:ХХ:СС |
| **TIMESTAMP** | Дата й час у форматі timestamp, виводиться у вигляді РРРРММДДГГХХСС |
| **DATETIME** | Дата й час у форматі РРРР-ММ-ДД-ГГ:ХХ:СС |

Стовпець – мульти множина значень, яку можна змінюватити і ,яка змінюється в реальному часі. Всі значення одного стовпця є значеннями одного типу і значеннями в одній таблиці. Значення стовпця - мінімальний елемент даних, які можуть бути обрані з таблиці, і мінімальний елемент даних, які можуть бути змінені.

Стовпець має опис і порядкову позицію в таблиці. Опис стовпця включає його тип даних і вказівка ​​про те, чи поширюється на стовпець обмеження. Опис стовпця символьних рядків визначає його атрибут довжини. Опис стовпця приблизних чисел визначає точність цих чисел. Опис стовпця точних чисел визначає точність і масштаб цих чисел.

Таблиці

Таблиця є мультимножина рядків. Усі рядки однієї таблиці містять значення кожного стовпця цієї таблиці. I- е значення кожного рядка таблиці є значенням i- ого стовпця цієї таблиці. Рядок є мінімальним елементом даних, які можна занести в таблицю і видалити з таблиці. Міра таблиці є число стовпців цієї таблиці. У будь-який момент часу міра таблиці та ж, що і потужність кожної з її рядків, і потужність таблиці та ж, що і потужність кожного з її стовпців. Таблиця має опис. Цей опис включає опис кожного із стовпців таблиці. Базова таблиця є іменована таблиця, визначена через <визначення таблиці> (<table definition>). Опис базової таблиці включає її ім'я. Породжена таблиця - це таблиця, породжена прямо або побічно з однієї або декількох інших таблиць шляхом обчислення <специфікації запиту> (<query specification>).Значеннями породженої таблиці є значення визначальних таблиць при її породженні. Таблиця, що представляється, є іменована таблиця, визначена через <визначення представлення> (<view definition>). Опис таблиці, що представляється, включає її ім'я. Таблиця являється або змінюваною (updatable), або тільки читаною. Операції занесення, модифікації і видалення допускаються для змінюваних таблиць і не допускаються для тільки читаних таблиць. Згрупована таблиця є безліч груп, що породжуються в ході обчислення <розділу group by> (<group by clause>). Група - це мульти-множина рядків, в яких усі значення стовпця (стовпців) групування рівні. Згруповану таблицю можна розглядати як набір таблиць. Множинні функції можуть застосовуватися до індивідуальних груп усередині згрупованої таблиці. Згруповане представлення - це таблиця, що представляється, породжена із згрупованої таблиці.

**Імена**

Функція специфікує імена

формат

<Table name> :: = [<authorization identifier>.]

<Table identifier>

**синтаксичні правила**

<Ім'я таблиці> (<table name>) ідентифікує іменовану таблицю.

Якщо <ім'я таблиці> (<table name>) не включає <ідентифікатора повноважень> (<authorization identifier>), то:

<Ім'я таблиці> (<table name>) оголошується в <визначенні таблиці> (<table definition>) або в <визначенні подання> (<view definition>).

<Ім'я таблиці> (<table name>) в <операторі SQL> (<SQL statement>) має ідентифікувати таблицю, певну в <схемою> (<schema>).

<Ім'я стовпця> (<column name>) ідентифікує іменований стовпець.

**Запити та функції**

Підтримка DELETE, INSERT, REPLACE і UPDATE, щоб повернути число рядків, які були змінені (порушені) або повернути замість цього число відповідних рядків, встановлюючи прапор, з'єднуючись з сервером.

**Безпека**

Безпека пароля з шифруванням всього трафіку, коли Ви з'єднуєтеся з сервером.

З'єднання і від'єднання від сервера

Щоб з'єднатися з сервером, Ви повинні будете зазвичай забезпечувати ім'я користувача MySQL, коли Ви викличете mysql і, найімовірніше, пароль.

host і user представляють ім'я хоста, де Ваш сервер MySQL працює і ім'я користувача Вашого облікового запису MySQL. Замініть на відповідні значення для Вашої установки.

Ви повинні бачити деяку ввідну інформацію, супроводжувану запрошенням mysql>:

shell> mysql -h host -u user -p

Enter password: \*\*\*\*\*\*\*\*

Welcome to the MySQL monitor. Commands end with; or \ g.

Your MySQL connection id is 25338 to server version: 8.0.1-standard

Type 'help;' or '\ h' for help. Type '\ c' to clear the buffer.

mysql>

Запрошення mysql> говорить Вам, що mysql готовий до роботи із запитами.

mysql> SELECT VERSION (), CURRENT\_DATE;

+ ----------- + -------------- +

| VERSION () | CURRENT\_DATE |

+ ----------- + -------------- +

| 5.8.0-m17 | 2015-12-21 |

+ ----------- + -------------- +

1 row in set (0.02 sec)

mysql>

Цей запит ілюструє кілька речей про mysql:

Запит зазвичай складається із запиту SQL, супроводжуваного крапкою з комою. Є деякі винятки, де крапка з комою може бути опущена. QUIT, згаданий раніше, одне з них. Ми доберемося до інших пізніше.

Коли Ви робите запит, mysql посилає це в сервер для виконання і виводить на екран результати, потім друкує інше запрошення mysql>, щоб вказати, що

Ключові слова можуть бути введені в будь-якому регістрі. Наступні запити еквівалентні:

mysql> SELECT VERSION (), CURRENT\_DATE;

mysql> select version (), current\_date;

mysql> SeLeCt vErSiOn (), current\_DATE;

Підказки

mysql> Готовий до нового запиту.

-> Очікування наступного рядка багатострокового запиту.

'> Очікування наступного рядка, чекаючи завершення рядка, яка почалася з одинарної лапки (').

"> Очікування наступного рядка, чекаючи завершення рядка, яка почалася з подвійної лапки (").

`> Очікування наступного рядка, чекаючи завершення ідентифікатора, який почався зі зворотним лапки (`).

/ \*> Очікування наступного рядка, чекаючи завершення коментаря, який почався з / \*.

Багаторядкові запити зазвичай відбуваються випадково, коли Ви маєте намір випустити запит на одному рядку, але забули завершальну крапку з комою. В цьому випадку mysql чекає більше введення:

mysql> SELECT USER ()

->

Якщо це відбувається з Вами (Ви думаєте, що ввели запит, але єдина відповідь ->), найбільш ймовірно, що mysql чекає крапку з комою. Якщо Ви не помічаєте те, що підказка говорить Вам, Ви могли б сидіти так деякий час перш, ніж зрозуміти, що Ви повинні зробити.Введіть крапку з комою, щоб завершити запит, і mysql його негайно буде:

mysql> SELECT USER ()

->;

+ --------------- +

| USER () |

+ --------------- +

| join @ localhost |

DELETE

DELETE це запит DML, який видаляє рядки з таблиці.

[WHERE where\_condition]

DELETE видаляє рядки з tbl\_name і повертає число вилучених рядків. Щоб перевірити число вилучених рядків, викличте ROW\_COUNT ().

Головні пропозиції

Умови в додатковому WHERE ідентифікує, які рядки видалити. Без WHERE всі рядки видалені.

where\_condition вираз, яке оцінюється до істини для кожного рядка, яка буде видалена. .

INSERT

INSERT [LOW\_PRIORITY | DELAYED | HIGH\_PRIORITY] [IGNORE]

[INTO] tbl\_name

UPDATE

Однотабличний варіант:

UPDATE [LOW\_PRIORITY] [IGNORE] table\_reference

SET col\_name1 = {expr1 | DEFAULT}

[ Col\_name2 = {expr2 | DEFAULT}] ...

[WHERE where\_condition]

Для однотабличного синтаксису UPDATE оновлює стовпці існуючих рядків в названій таблиці з новими значеннями. SET вказує, які стовпці змінити і значення їм потрібно дати. Кожне значення може бути дано як вираз або ключове слово DEFAULT, щоб встановити стовпець явно в його значення за замовчуванням. WHERE, якщо дано, визначає умови, які ідентифікують який рядки оновити. Без WHERE всі рядки оновлені. Якщо ORDER BY визначено, рядки оновлені в порядку, який визначений. LIMIT встановлює кордон числа рядків, які можуть бути оновлені.

where\_condition вираз, яке оцінюється до істини для кожного рядка, яка буде оновлена.

ALTER TABLE

ALTER TABLE tbl\_name

[Alter\_specification [ alter\_specification] ...]

[Partition\_options]

alter\_specification:

table\_options

| ADD [COLUMN] col\_name column\_definition

[FIRST | AFTER col\_name]

| ADD [COLUMN] (col\_name column\_definition, ...

| DROP [COLUMN] col\_name

ALTER TABLE змінює структуру таблиці. Наприклад, Ви можете додати або видалити стовпці, створити або зруйнувати індекс, змінити тип існуючих стовпців, перейменувати стовпці або таблицю безпосередньо. Ви можете також змінити характеристики, такі як механізм зберігання, використовуваний для таблиці або табличного коментаря.

Після імені таблиці, визначте зміни, які будуть зроблені. Якщо жодне не дано, ALTER TABLE нічого не робить.

DROP TABLE

DROP [TEMPORARY] TABLE [IF EXISTS]

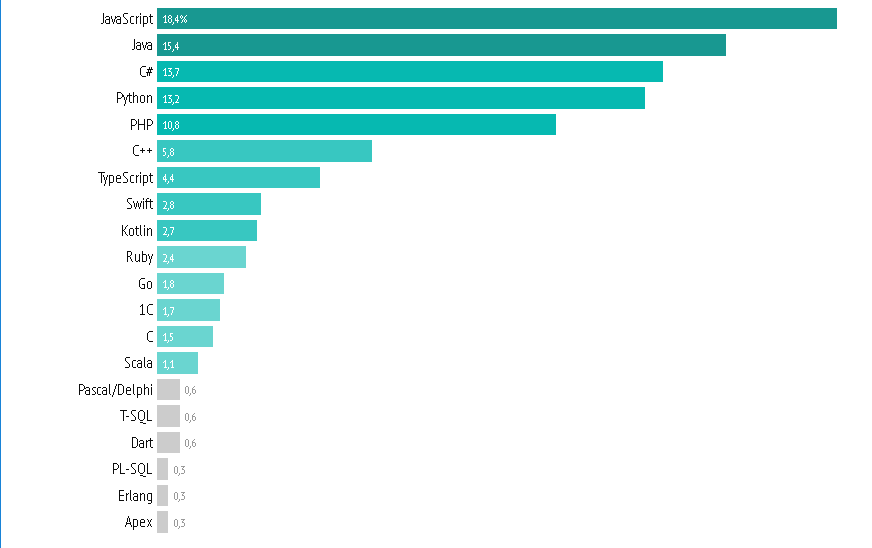
tbl\_name [ tbl\_name] ...

[RESTRICT | CASCADE]

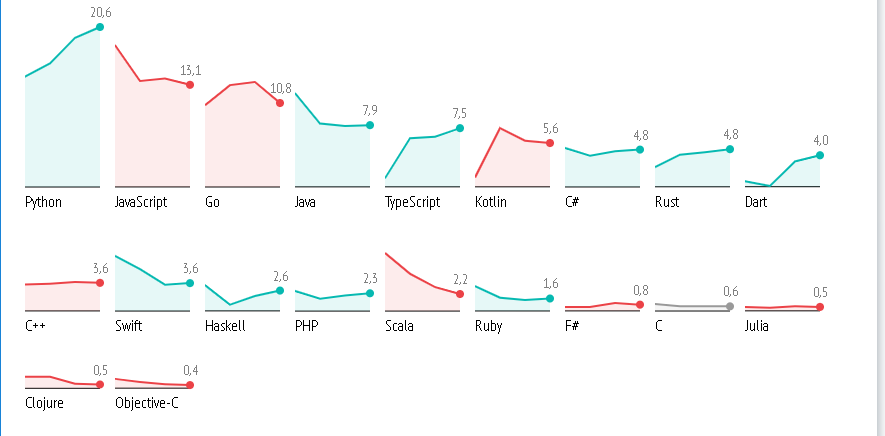
DROP TABLE видаляє одну або більше таблиць. Ви повинні мати привілей DROP для кожної таблиці. Всі табличні дані і табличне визначення видалені, так що будьте обережні! Якщо будь-яка з таблиць, названих в списку параметрів, не існує, MySQL повертає помилку, вказуючи на ім'я, які неіснуючі таблиці не вдалося видалити, але видаляє всі таблиці в списку, які дійсно існують.

# **Статистика**

Рейтинг мов які використовують для написання зараз



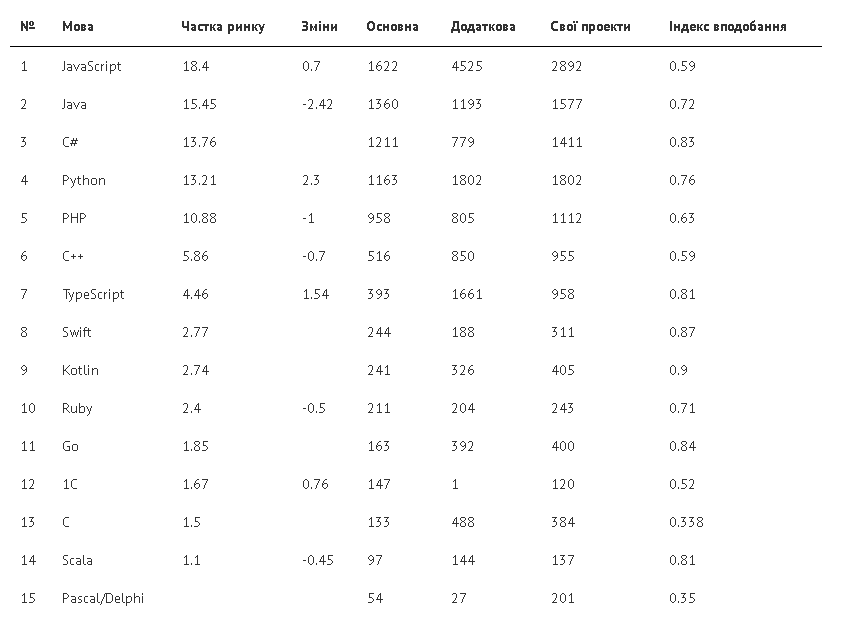
Бачимо, що JavaScript нарешті суттєво випередив Java і став найпопулярнішою мовою програмування. Також серед нових мов слід звернути увагу на Dart — він одразу зайняв, не сказати щоби велику, але досить помітну нішу.

Цікаво ,яку мову збираються вивчати наступного року . На це питання є відповідь.

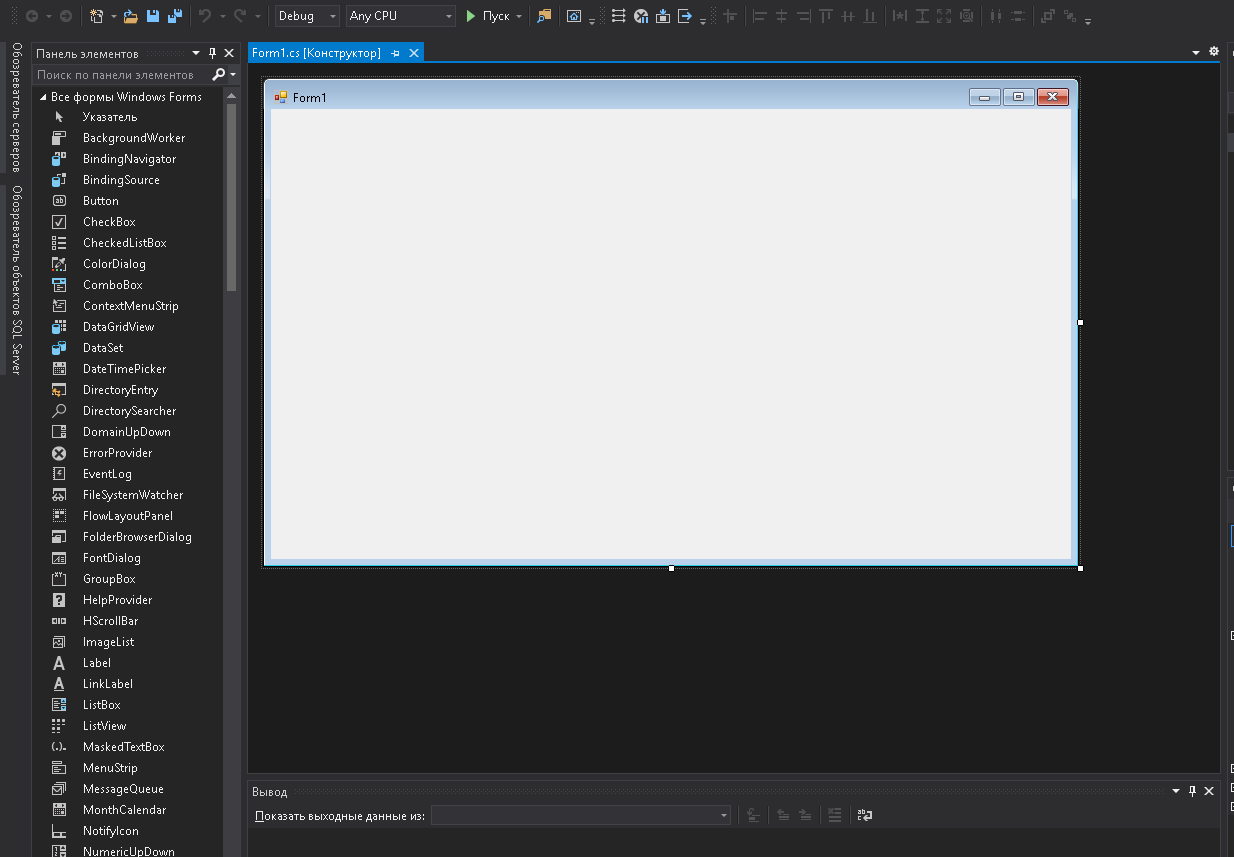
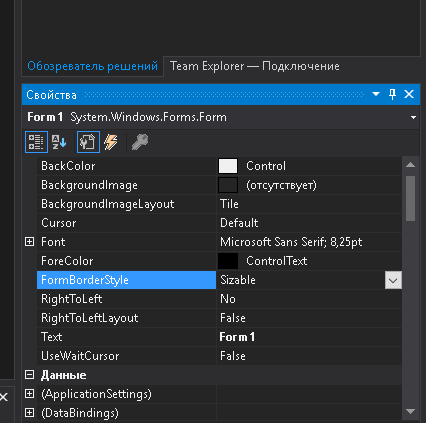
Лідери — Python та JavaScript — тут зрозуміло. Також можна звернути увагу на порівняно високу частку (у порівнянні з актуальним розповсюдженням) інтересу до Dart, Rust, ну і традиційно — Haskell.

Як і в попередні роки, більшість респондентів (54%) збираються вивчати нову мову самостійно, з допомогою книг та документації. На онлайн-курси орієнтуються майже 32%, а 7% збираються використовувати офлайн-підхід із локальними викладачами.

Рейтинг мов програмування

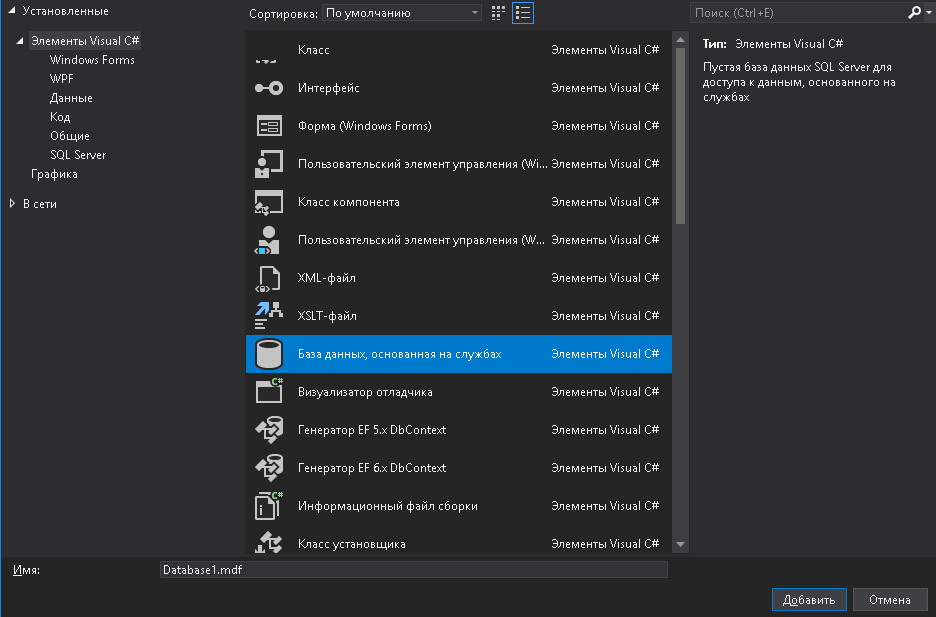


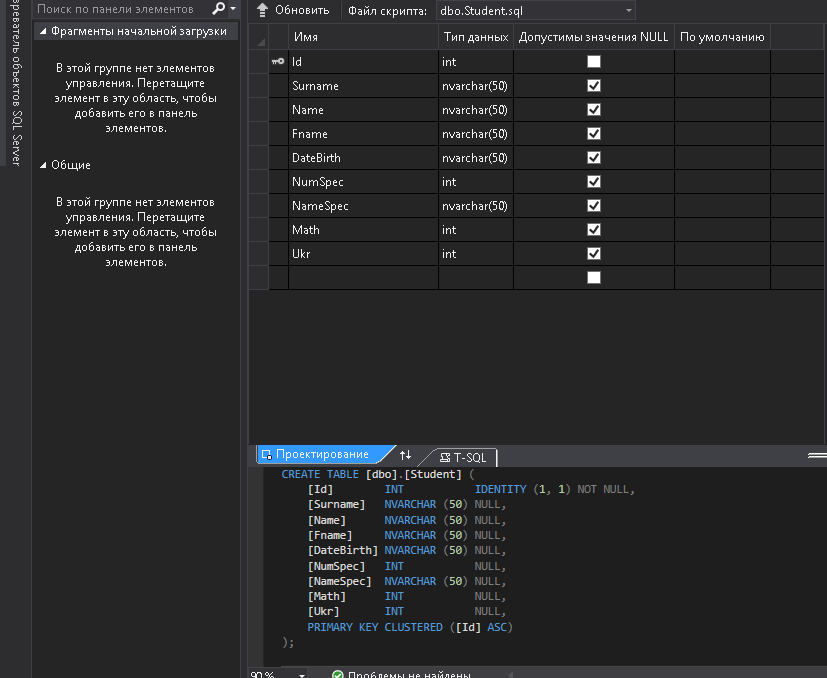
# **Розробка програмного забезпечення**

Першим кроком створюємо додаток Windows Forms в середовищі Visual StudioДалі змінюємо формат шапки на Sizable

Після цього підключаємо базу даних Database1 і створюємо таблицю Student

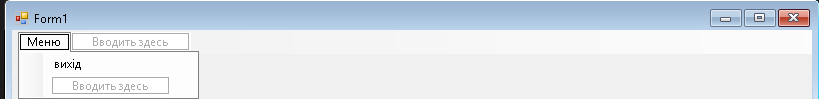
,яку заповняємо відповідною інформацією



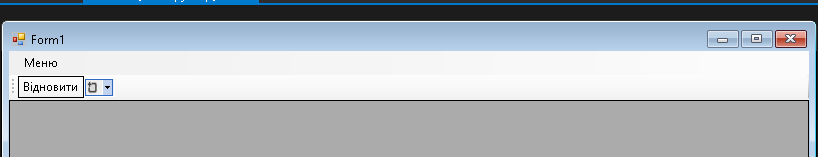


Після підключення займаємося оформленням зовнішнього виду програми.

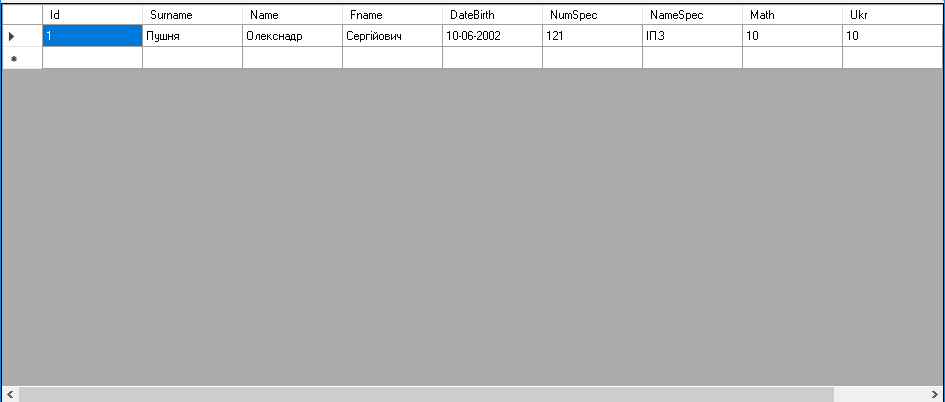
Додаємо MenuStrip і кнопку для виходу.



Створюємо toolStrip з кнопкою відновити.



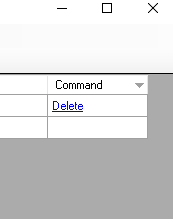
Додаємо DataGridView і підписуємо параметри зрозумілими назвами



Додаємо кнопку в DataGridView для видалення ,додавання та редагування

Інформації в таблиці.

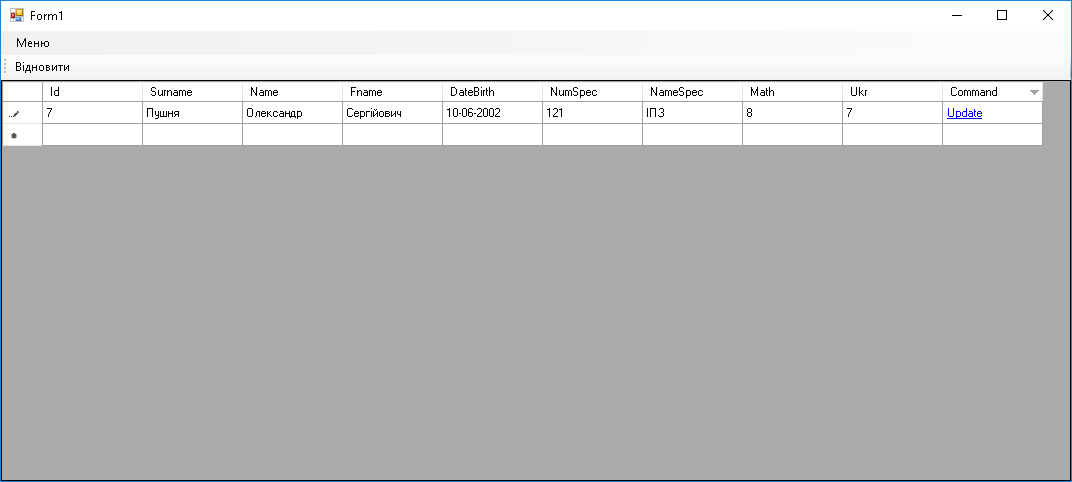
Кнопка видалення з’являється тільки якщо в таблиці заповненні всі стрічки



При додаванні нової інформації з’являється кнопка додавання ,вставки (Insert) в таблицю .



При редагуванні інформації з'являється кнопка Update.



# Код програми

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Data.SqlClient;

namespace collageDB

{

public partial class Form1 : Form

{

private SqlConnection sqlConnection = null;

private SqlCommandBuilder sqlBuilder = null;

private SqlDataAdapter sqlDataAdapter = null;

private DataSet dataSet = null;

private bool newRowAdding = false;

public Form1()

{

InitializeComponent();

}

private void LoadData()

{

try

{

sqlDataAdapter = new SqlDataAdapter("SELECT \*,'Delete' AS [Command] From Student", sqlConnection);

sqlBuilder = new SqlCommandBuilder(sqlDataAdapter);

sqlBuilder.GetInsertCommand();

sqlBuilder.GetUpdateCommand();

sqlBuilder.GetDeleteCommand();

dataSet = new DataSet();

sqlDataAdapter.Fill(dataSet, "Student");

dataGridView1.DataSource = dataSet.Tables["Student"];

for (int i = 0; i < dataGridView1.Rows.Count; i++)

{

DataGridViewLinkCell linkCell = new DataGridViewLinkCell();

dataGridView1[9, i] = linkCell;

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Помилка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void UpdateData()

{

try

{

dataSet.Tables["Student"].Clear();

sqlDataAdapter.Fill(dataSet, "Student");

dataGridView1.DataSource = dataSet.Tables["Student"];

for (int i = 0; i < dataGridView1.Rows.Count; i++)

{

DataGridViewLinkCell linkCell = new DataGridViewLinkCell();

dataGridView1[9, i] = linkCell;

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Помилка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void Form1\_Load(object sender, EventArgs e)

{

sqlConnection = new SqlConnection(@"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=C:\Users\lolol\source\repos\collageDB\collageDB\Database1.mdf;Integrated Security=True");

sqlConnection.Open();

LoadData();

}

private void toolStripButton1\_Click(object sender, EventArgs e)

{

UpdateData();

}

private void dataGridView1\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

try

{

if (e.ColumnIndex == 9)

{

string task = dataGridView1.Rows[e.RowIndex].Cells[9].Value.ToString();

if (task == "Delete")

{

if (MessageBox.Show("Видалити цю стрічку?", "Видалити", MessageBoxButtons.YesNo, MessageBoxIcon.Question)

== DialogResult.Yes)

{

int rowIndex = e.RowIndex;

dataGridView1.Rows.RemoveAt(rowIndex);

dataSet.Tables["Student"].Rows[rowIndex].Delete();

sqlDataAdapter.Update(dataSet, "Student");

}

}

else if (task == "Insert")

{

int rowIndex = dataGridView1.Rows.Count - 2;

DataRow row = dataSet.Tables["Student"].NewRow();

row["Surname"] = dataGridView1.Rows[rowIndex].Cells["Surname"].Value;

row["Name"] = dataGridView1.Rows[rowIndex].Cells["Name"].Value;

row["Fname"] = dataGridView1.Rows[rowIndex].Cells["Fname"].Value;

row["DateBirth"] = dataGridView1.Rows[rowIndex].Cells["DateBirth"].Value;

row["NumSpec"] = dataGridView1.Rows[rowIndex].Cells["NumSpec"].Value;

row["NameSpec"] = dataGridView1.Rows[rowIndex].Cells["NameSpec"].Value;

row["Math"] = dataGridView1.Rows[rowIndex].Cells["Math"].Value;

row["Ukr"] = dataGridView1.Rows[rowIndex].Cells["Ukr"].Value;

dataSet.Tables["Student"].Rows.Add(row);

dataSet.Tables["Student"].Rows.RemoveAt(dataSet.Tables["Student"].Rows.Count - 1);

dataGridView1.Rows.RemoveAt(dataGridView1.Rows.Count - 2);

dataGridView1.Rows[e.RowIndex].Cells[9].Value = "Delete";

sqlDataAdapter.Update(dataSet, "Student");

newRowAdding = false;

}

else if (task == "Update")

{

int r = e.RowIndex;

dataSet.Tables["Student"].Rows[r]["Surname"] = dataGridView1.Rows[r].Cells["Surname"].Value;

dataSet.Tables["Student"].Rows[r]["Name"] = dataGridView1.Rows[r].Cells["Name"].Value;

dataSet.Tables["Student"].Rows[r]["Fname"] = dataGridView1.Rows[r].Cells["Fname"].Value;

dataSet.Tables["Student"].Rows[r]["DateBirth"] = dataGridView1.Rows[r].Cells["DateBirth"].Value;

dataSet.Tables["Student"].Rows[r]["NumSpec"] = dataGridView1.Rows[r].Cells["NumSpec"].Value;

dataSet.Tables["Student"].Rows[r]["NameSpec"] = dataGridView1.Rows[r].Cells["NameSpec"].Value;

dataSet.Tables["Student"].Rows[r]["Math"] = dataGridView1.Rows[r].Cells["Math"].Value;

dataSet.Tables["Student"].Rows[r]["Ukr"] = dataGridView1.Rows[r].Cells["Ukr"].Value;

sqlDataAdapter.Update(dataSet, "Student");

dataGridView1.Rows[e.RowIndex].Cells[9].Value = "Delete";

}

UpdateData();

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Помилка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void dataGridView1\_UserAddedRow(object sender, DataGridViewRowEventArgs e)

{

try

{

if (newRowAdding == false)

{

newRowAdding = true;

int lastRow = dataGridView1.Rows.Count - 2;

DataGridViewRow row = dataGridView1.Rows[lastRow];

DataGridViewLinkCell linkCell = new DataGridViewLinkCell();

dataGridView1[9, lastRow] = linkCell;

row.Cells["Command"].Value = "Insert";

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Помилка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void dataGridView1\_CellValueChanged(object sender, DataGridViewCellEventArgs e)

{

try

{

if (newRowAdding == false)

{

int rowIndex = dataGridView1.SelectedCells[0].RowIndex;

DataGridViewRow editingRow = dataGridView1.Rows[rowIndex];

DataGridViewLinkCell linkCell = new DataGridViewLinkCell();

dataGridView1[9, rowIndex] = linkCell;

editingRow.Cells["Command"].Value = "Update";

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Помилка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void dataGridView1\_EditingControlShowing(object sender, DataGridViewEditingControlShowingEventArgs e)

{

e.Control.KeyPress -= new KeyPressEventHandler(Column\_KeyPress);

if (dataGridView1.CurrentCell.ColumnIndex == 5)

{

TextBox textBox = e.Control as TextBox;

if (textBox != null)

{

textBox.KeyPress += new KeyPressEventHandler(Column\_KeyPress);

}

}

if (dataGridView1.CurrentCell.ColumnIndex == 7)

{

TextBox textBox = e.Control as TextBox;

if (textBox != null)

{

textBox.KeyPress += new KeyPressEventHandler(Column\_KeyPress);

}

}

if (dataGridView1.CurrentCell.ColumnIndex == 8)

{

TextBox textBox = e.Control as TextBox;

if (textBox != null)

{

textBox.KeyPress += new KeyPressEventHandler(Column\_KeyPress);

}

}

}

private void Column\_KeyPress(object sender,KeyPressEventArgs e)

{

if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar))

{

e.Handled = true;

}

}

private void вихідToolStripMenuItem\_Click(object sender, EventArgs e)

{

if (sqlConnection != null && sqlConnection.State != ConnectionState.Closed)

sqlConnection.Close();

Close();

}

}

}

# **Висновок**

Для облегшення праці робітникам вищих навчальних закладів програма дасть змогу мати менше клопоту з інформацією в паперовому вигляді, а тим більш замінити її в електронному вигляді. Головна задача пришвидшити роботу приймальної комісії для зарахування абітурієнта на навчання. Надалі планується розробити функцію, яка буде вираховувати середній бал, екзамени, пільги з коефіцієнтами ,а також здійснювати пошук по БД. Інтерфейс програми дуже простий і зручний у використанні хоча і не має багато функціоналу і яскравого зовнішнього вигляду.

# **Список використаної літератури**

1. https://ela.kpi.ua/bitstream/123456789/16671/1/OOP\_manual.pdf

2. https://dou.ua/lenta/articles/language-rating-jan-2020/

3. http://csc.knu.ua/en/library/dissertations/poliakov.pdf

4. http://www.mysql.ru/docs/man/Security.html

5. https://www.apserver.org.ua/peregl.php?d=view&tid=25

6. https://uk.wikipedia.org/wiki/C\_Sharp

7. https://ru.wikipedia.org/wiki/MySQL