Quản lý chương trình đào tạo

# 1. Cơ sở dữ liệu

## 1.1. Tạo CSDL và bảng bảng

* CSDL: **CREATE DATABASE EduManager;**
* Bảng

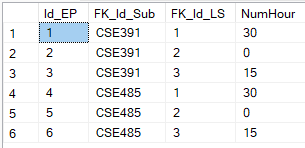
| **-- Bảng Subjects**  CREATE TABLE Subjects (  Id\_Sub VARCHAR(10) PRIMARY KEY,  Name\_Sub NVARCHAR(100)  );  **-- Bảng LearningStyle**  CREATE TABLE LearningStyle (  Id\_LS INT PRIMARY KEY,  Name\_LS NVARCHAR(50)  );  **-- Bảng EduProgram**  CREATE TABLE EduProgram (  Id\_EP INT PRIMARY KEY IDENTITY(1,1),  FK\_Id\_Sub VARCHAR(10),  FK\_Id\_LS INT,  NumHour INT,  FOREIGN KEY (FK\_Id\_Sub) REFERENCES Subjects(Id\_Sub),  FOREIGN KEY (FK\_Id\_LS) REFERENCES LearningStyle(Id\_LS)  ); |
| --- |

## 1.2. Chèn dữ liệu

| **-- Thêm dữ liệu vào bảng LearningStyle**  INSERT INTO LearningStyle (Id\_LS, Name\_LS) VALUES  (1, N'Lý thuyết / lý luận'),  (2, N'Bài tập / Thảo luận'),  (3, N'Thực hành'); |
| --- |

## 1.3. Tạo view

| CREATE VIEW EduProgramView AS  WITH SourceData AS (  SELECT  FK\_Id\_Sub,  Name\_Sub,  Name\_LS,  NumHour  FROM  EduProgram  JOIN  Subjects ON EduProgram.FK\_Id\_Sub = Subjects.Id\_Sub  JOIN  LearningStyle ON EduProgram.FK\_Id\_LS = LearningStyle.Id\_LS  )  SELECT  FK\_Id\_Sub AS "Mã môn học",  Name\_Sub AS "Tên môn học",  [Lý thuyết / lý luận] AS "Lý thuyết",  [Bài tập / Thảo luận] AS "Bài tập",  [Thực hành] AS "Thực hành"  FROM  SourceData  PIVOT (  MAX(NumHour)  FOR Name\_LS IN ([Lý thuyết / lý luận], [Bài tập / Thảo luận], [Thực hành])  ) AS PivotTable; |
| --- |



*Hình trên biểu thị dữ liệu ban đầu của bảng EduProgram khi chưa PIVOT*

Đoạn mã lệnh SQL trên tạo một khung nhìn (VIEW) trong cơ sở dữ liệu với tên là EduProgramView. Nó sử dụng cú pháp WITH...AS... để định nghĩa dữ liệu nguồn và sau đó áp dụng **PIVOT** để xoay dữ liệu từ định dạng hàng thành định dạng cột.



Cụ thể hơn:

* **Khung nhìn (VIEW)**: Tạo một dạng lưu trữ ảo của dữ liệu, cho phép truy xuất dễ dàng dưới dạng bảng mà không cần truy vấn lại cơ sở dữ liệu mỗi lần.
* **WITH SourceData AS ...**: Đây là một câu lệnh con (subquery) được gọi là SourceData. Nó thu thập dữ liệu từ bảng EduProgram, sau đó nối (JOIN) với bảng Subjects để lấy tên môn học (Name\_Sub) và với bảng LearningStyle để lấy kiểu học (Name\_LS). Nó cũng lấy thông tin số giờ học (NumHour).
* **SELECT ... FROM SourceData PIVOT ...**: Đây là phần chính của truy vấn. Nó sử dụng PIVOT để xoay dữ liệu:
* Phần **SELECT** xác định các cột sẽ xuất hiện trong kết quả. Nó cũng chuyển đổi tên cột thành tên thân thiện hơn với người dùng (ví dụ, "Mã môn học" thay vì FK\_Id\_Sub).
* Phần **PIVOT** định nghĩa cách xoay dữ liệu. Nó sử dụng hàm tổng hợp MAX để lấy số giờ lớn nhất cho từng môn học theo từng kiểu học. FOR Name\_LS IN ([Lý thuyết / lý luận], [Bài tập / Thảo luận], [Thực hành]) chỉ định rằng bạn muốn xoay dữ liệu trên cột Name\_LS và tạo các cột mới cho từng giá trị duy nhất (ở đây là "Lý thuyết / lý luận", "Bài tập / Thảo luận", "Thực hành").
* Tên "**PivotTable**" được đặt cho bảng kết quả sau khi áp dụng PIVOT.
* Kết quả cuối cùng là một bảng với các cột: "Mã môn học", "Tên môn học", "Lý thuyết", "Bài tập", "Thực hành", hiển thị số giờ cho từng kiểu học theo mỗi môn học.

Truy vấn này giúp bạn thấy rõ ràng số giờ học theo từng loại hoạt động (lý thuyết, bài tập, thực hành) cho mỗi môn học trong chương trình giáo dục.

# 2. Mã nguồn

## 2.1. Services

**ConnectDatabase.cs**

| using System;  using System.Collections.Generic;  using System.Data;  using System.Data.SqlClient;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace EduManager.Services  {  internal class ConnectDatabase  {  private static ConnectDatabase instance;  private string \_connectionString = @"Data Source=DUNGHAQUANG;Initial Catalog=EduManager;Integrated Security=True";  private ConnectDatabase() { }  **// Tạo kết nối**  public static ConnectDatabase getInstance()  {  if (instance == null)  {  instance = new ConnectDatabase();  }  return instance;  }  **// Sử dụng để truy vấn câu lệnh SELECT**  public DataTable ExecuteQuery(string query, SqlParameter[] parameter = null)  {  DataTable data = new DataTable();  using (SqlConnection sqlConnection = new SqlConnection(\_connectionString))  {  sqlConnection.Open();  SqlCommand cmd = new SqlCommand(query, sqlConnection);  if (parameter != null)  cmd.Parameters.AddRange(parameter);  SqlDataAdapter adapter = new SqlDataAdapter(cmd);  data.Clear();  adapter.Fill(data);  sqlConnection.Close();  }  return data;  }  **// Sử dụng để thêm sửa xóa**  public int ExecuteNonQuery(string query, SqlParameter[] parameter = null)  {  int rowsAffected = 0;  using (SqlConnection sqlConnection = new SqlConnection(\_connectionString))  {  sqlConnection.Open();  SqlCommand cmd = new SqlCommand(query, sqlConnection);  if (parameter != null)  cmd.Parameters.AddRange(parameter);  rowsAffected = cmd.ExecuteNonQuery();  sqlConnection.Close();  }  return rowsAffected;  }  public string GetValue(string query)  {  string value = null;  using (SqlConnection sqlConnection = new SqlConnection(\_connectionString))  {  sqlConnection.Open();  SqlCommand cmd = new SqlCommand(query, sqlConnection);  SqlDataReader dataReader = cmd.ExecuteReader();  if (dataReader.Read())  {  value = dataReader.GetValue(0).ToString();  }  sqlConnection.Close();  }  return value;  }  public SqlDataReader ExecuteReader(string query)  {  SqlConnection sqlConnection = new SqlConnection(\_connectionString);  sqlConnection.Open();  SqlCommand cmd = new SqlCommand(query, sqlConnection);  SqlDataReader reader = cmd.ExecuteReader();  return reader;  }  }  } |
| --- |

## 2.2. Models

### 2.2.1. Subject

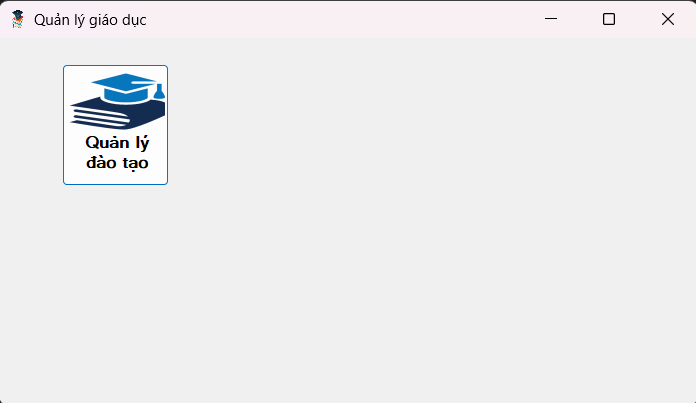
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace EduManager.Models  {  internal class Subject  {  public string Id\_Sub { get; set; }  public string Name\_Sub { get; set; }  public Subject() { }  public Subject(string id\_Sub)  {  Id\_Sub = id\_Sub;  }  public Subject(string id\_Sub, string name\_Sub)  {  Id\_Sub = id\_Sub;  Name\_Sub = name\_Sub;  }  }  } |
| --- |

### 2.2.2. EduProgram

| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace EduManager.Models  {  internal class EduProgram  {  public int Id\_EP { get; set; }  public string FK\_Id\_Sub { get; set; }  public int FK\_Id\_LS { get; set; }  public int NumHour { get; set; }  public EduProgram() { }  public EduProgram(string fK\_Id\_Sub)  {  FK\_Id\_Sub = fK\_Id\_Sub;  }  public EduProgram(string fK\_Id\_Sub, int fK\_Id\_LS, int numHour)  {  FK\_Id\_Sub = fK\_Id\_Sub;  FK\_Id\_LS = fK\_Id\_LS;  NumHour = numHour;  }  public EduProgram(int id\_EP, string fK\_Id\_Sub, int fK\_Id\_LS, int numHour)  {  Id\_EP = id\_EP;  FK\_Id\_Sub = fK\_Id\_Sub;  FK\_Id\_LS = fK\_Id\_LS;  NumHour = numHour;  }  }  } |
| --- |

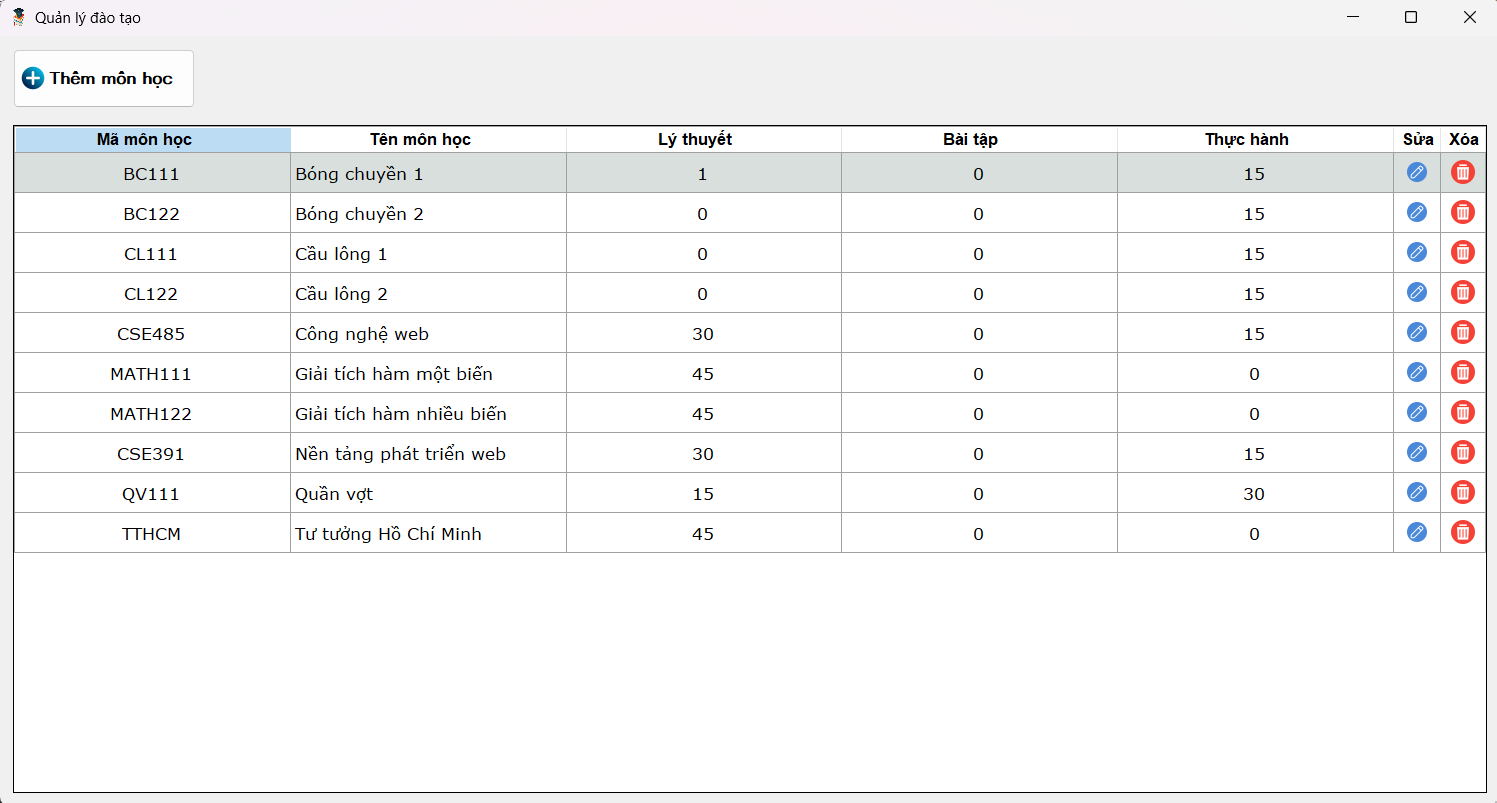
## 2.3. Views

### 2.3.1. Form Main



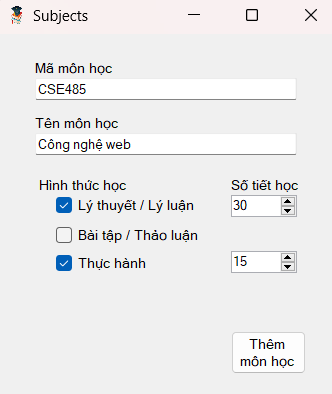
| using System;  using System.Drawing;  using System.Windows.Forms;  using EduManager.Views;  namespace EduManager  {  public partial class Main : Form  {  public Main()  {  InitializeComponent();  }  private void btnEduManage\_Click(object sender, EventArgs e)  {  EduProgram\_Form f = new EduProgram\_Form();  this.Hide();  f.ShowDialog();  this.Show();  }  private void Main\_Load(object sender, EventArgs e)  {  **// Điều chỉnh kích thước hình ảnh để vừa với nút**  Size buttonSize = btnEduManage.Size;  Image resizedIcon = ResizeImage(Properties.Resources.QLDT, new Size(buttonSize.Width, buttonSize.Height / 2)); // Chia đôi chiều cao  **// Gán hình ảnh đã được điều chỉnh vào nút**  btnEduManage.Image = resizedIcon;  **// Chỉnh văn bản hiển thị phía dưới hình ảnh**  btnEduManage.TextImageRelation = TextImageRelation.ImageAboveText;  }  private Image ResizeImage(Image img, Size newSize)  {  Bitmap resizedImage = new Bitmap(newSize.Width, newSize.Height);  using (Graphics g = Graphics.FromImage(resizedImage))  {  g.DrawImage(img, 0, 0, newSize.Width, newSize.Height);  }  return resizedImage;  }  }  } |
| --- |

### 2.3.2. Form EduProgram



| using System;  using System.Drawing;  using System.IO;  using System.Linq;  using System.Windows.Forms;  using EduManager.Controllers;  using EduManager.Models;  namespace EduManager.Views  {  public partial class EduProgram\_Form : Form  {  public **EduProgram\_Form**()  {  InitializeComponent();  dtgvEduProgram.RowTemplate.Height = 40;  }  **// FORM LOAD**  private void **EduProgram\_Load**(object sender, EventArgs e)  {  LoadData();  // Chọn dòng toàn bộ khi click  dtgvEduProgram.SelectionMode = DataGridViewSelectionMode.FullRowSelect;  dtgvEduProgram.DefaultCellStyle.SelectionBackColor = ColorTranslator.FromHtml("#d8dfdc");  dtgvEduProgram.DefaultCellStyle.SelectionForeColor = ColorTranslator.FromHtml("#000");  btnAddSubject.Image = Properties.Resources.ICON\_ADD\_24x24;  btnAddSubject.TextImageRelation = TextImageRelation.ImageBeforeText;  }  **// LOAD DATA**  public void **LoadData**()  {  EduProgramController.Instance().ShowData(dtgvEduProgram);  ConfigureDataGridViewReadOnly(dtgvEduProgram, "Mã môn học");  ConfigureColumnHeaders(dtgvEduProgram);  ConfigureColumnAlignment(dtgvEduProgram, new string[] { "Mã môn học", "Lý thuyết", "Bài tập", "Thực hành" });  AddActionColumns(dtgvEduProgram);  }  **// ADD & CONFIGURATION DATAGRIDVIEW**  private void **AddActionColumns**(DataGridView dgv)  {  if (dgv.Columns["Edit"] == null) // Chỉ thêm nếu chưa tồn tại  {  string imagePath = Path.Combine(AppDomain.CurrentDomain.BaseDirectory, "..", "..", "Resources", "ICON\_EDIT-24x24.png");  var editColumn = new DataGridViewImageColumn  {  Name = "Edit",  HeaderText = "Sửa",  Image = Image.FromFile(imagePath),  AutoSizeMode = DataGridViewAutoSizeColumnMode.AllCells  };  dgv.Columns.Add(editColumn);  }  if (dgv.Columns["Delete"] == null) // Chỉ thêm nếu chưa tồn tại  {  string imagePath = Path.Combine(AppDomain.CurrentDomain.BaseDirectory, "..", "..", "Resources", "ICON\_TRASH-24x24.png");  var deleteColumn = new DataGridViewImageColumn  {  Name = "Delete",  HeaderText = "Xóa",  Image = Image.FromFile(imagePath),  AutoSizeMode = DataGridViewAutoSizeColumnMode.AllCells  };  dgv.Columns.Add(deleteColumn);  }  }  private void **ConfigureDataGridViewReadOnly**(DataGridView dgv, string columnName)  {  var column = dgv.Columns[columnName];  if (column != null)  {  column.ReadOnly = true; // Đặt cột là chỉ đọc  }  }  private void **ConfigureColumnHeaders**(DataGridView dgv)  {  var headerStyle = new DataGridViewCellStyle  {  Alignment = DataGridViewContentAlignment.MiddleCenter,  BackColor = SystemColors.GrayText,  ForeColor = SystemColors.WindowText,  Font = new Font("Arial", 10, FontStyle.Bold)  };  dgv.ColumnHeadersDefaultCellStyle = headerStyle;  }  private void **ConfigureColumnAlignment**(DataGridView dgv, string[] columnNames)  {  foreach (var columnName in columnNames)  {  var column = dgv.Columns[columnName];  if (column != null)  {  column.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;  column.HeaderCell.Style.Alignment = DataGridViewContentAlignment.MiddleCenter;  }  }  }  **// EVENT HANDLERS**  private void **dtgvEduProgram\_CellClick**(object sender, DataGridViewCellEventArgs e)  {  if (e.RowIndex < 0) return; // Bỏ qua khi click vào tiêu đề  var dgv = sender as DataGridView;  var headerText = dgv.Columns[e.ColumnIndex].HeaderText;  if (headerText == "Xóa")  {  HandleDelete(dgv, e.RowIndex);  }  else if (headerText == "Sửa")  {  HandleEdit(dgv, e.RowIndex);  }  }  private void **HandleDelete**(DataGridView dgv, int rowIndex)  {  var subjectId = dgv.Rows[rowIndex].Cells[0].Value.ToString();  var subjectName = dgv.Rows[rowIndex].Cells[1].Value.ToString();  // Xác nhận trước khi xóa  var confirmation = MessageBox.Show(  $"Bạn có chắc chắn muốn xóa môn \"{subjectName}\"?",  "Xác nhận",  MessageBoxButtons.YesNo,  MessageBoxIcon.Question // Thay đổi biểu tượng để cho biết đây là một câu hỏi xác nhận  );  if (confirmation == DialogResult.Yes)  {  var isDeleted = EduProgramController.Instance().RemoveAllData(new EduProgram(subjectId)) &&  SubjectController.Instance().RemoveData(new Subject(subjectId));  if (isDeleted)  {  LoadData(); // Tải lại dữ liệu  MessageBox.Show("Xóa môn học thành công!", "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Information);  }  else  {  MessageBox.Show("Xóa môn học thất bại.", "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Error);  }  }  }  private void **HandleEdit**(DataGridView dgv, int rowIndex)  {  var subjectId = dgv.Rows[rowIndex].Cells[0].Value.ToString();  var subjectName = dgv.Rows[rowIndex].Cells[1].Value.ToString();  var lectureHours = Convert.ToInt32(dgv.Rows[rowIndex].Cells[2].Value);  var exerciseHours = Convert.ToInt32(dgv.Rows[rowIndex].Cells[3].Value);  var practiceHours = Convert.ToInt32(dgv.Rows[rowIndex].Cells[4].Value);  var subject = new Subject(subjectId, subjectName);  var eduProgram1 = new EduProgram(subjectId, 1, lectureHours);  var eduProgram2 = new EduProgram(subjectId, 2, exerciseHours);  var eduProgram3 = new EduProgram(subjectId, 3, practiceHours);  var isUpdated = SubjectController.Instance().editData(subject) &&  EduProgramController.Instance().EditData(eduProgram1) &&  EduProgramController.Instance().EditData(eduProgram2) &&  EduProgramController.Instance().EditData(eduProgram3);  if (isUpdated)  {  LoadData();  MessageBox.Show("Sửa môn học thành công!", "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Information);  }  else  {  MessageBox.Show("Sửa môn học thất bại.", "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Error);  }  }  private void **btnAddSubject\_Click**(object sender, EventArgs e)  {  var subjectForm = new Subjects(this);  subjectForm.ShowDialog();  }  **// VALIDATION**  private object previousValue;  private void **dtgvEduProgram\_CellBeginEdit**(object sender, DataGridViewCellCancelEventArgs e)  {  previousValue = dtgvEduProgram[e.ColumnIndex, e.RowIndex].Value;  }  private void **dtgvEduProgram\_CellEndEdit**(object sender, DataGridViewCellEventArgs e)  {  var currentValue = dtgvEduProgram[e.ColumnIndex, e.RowIndex].Value;  if (currentValue == null || string.IsNullOrWhiteSpace(currentValue.ToString()))  {  dtgvEduProgram[e.ColumnIndex, e.RowIndex].Value = previousValue;  }  else  {  if (!currentValue.Equals(previousValue))  {  dtgvEduProgram[e.ColumnIndex, e.RowIndex].Style.ForeColor = System.Drawing.Color.Blue;  }  }  }  private void **dtgvEduProgram\_EditingControlShowing**(object sender, DataGridViewEditingControlShowingEventArgs e)  {  var textBox = e.Control as TextBox;  if (textBox != null)  {  textBox.KeyPress -= TextBox\_KeyPress;  int columnIndex = dtgvEduProgram.CurrentCell.ColumnIndex;  int[] numericColumns = { 2, 3, 4 };  if (numericColumns.Contains(columnIndex))  {  textBox.KeyPress += TextBox\_KeyPress;  }  }  }  private void **TextBox\_KeyPress**(object sender, KeyPressEventArgs e)  {  if (!char.IsDigit(e.KeyChar) && !char.IsControl(e.KeyChar))  {  e.Handled = true;  lbMessage.Text = "Vui lòng chỉ nhập số!";  lbMessage.ForeColor = Color.Red;  }  else  {  lbMessage.Text = "";  }  }  }  } |
| --- |

### 2.3.3. Form Subjects



| using System;  using System.Windows.Forms;  using EduManager.Models;  using EduManager.Controllers;  using System.Text.RegularExpressions;  namespace EduManager.Views  {  public partial class Subjects : Form  {  private EduProgram\_Form **\_eduProgramForm**;  public **Subjects**(EduProgram\_Form eduProgramForm)  {  \_eduProgramForm = eduProgramForm;  InitializeComponent();  }  private void **chbLT\_CheckedChanged**(object sender, EventArgs e)  {  nmLT.Visible = chbLT.Checked ? true : false;  }  private void **chbBT\_CheckedChanged**(object sender, EventArgs e)  {  nmBT.Visible = chbBT.Checked ? true : false;  }  private void **chbTH\_CheckedChanged**(object sender, EventArgs e)  {  nmTH.Visible = chbTH.Checked ? true : false;  }  private void **btnAddSubject\_Click**(object sender, EventArgs e)  {  // Kiểm tra nếu các trường bắt buộc là rỗng  if (string.IsNullOrWhiteSpace(txbId\_Sub.Text) || string.IsNullOrWhiteSpace(txbName\_Sub.Text))  {  MessageBox.Show("Vui lòng điền ID môn học và tên môn học hợp lệ.", "Lỗi", MessageBoxButtons.OK, MessageBoxIcon.Warning);  return;  }  var subjectAdded = AddSubject();  // Thêm chương trình giáo dục cho các checkbox được chọn  bool ltAdded = AddEduProgram(1, (int)nmLT.Value);  bool btAdded = AddEduProgram(2, (int)nmBT.Value);  bool thAdded = AddEduProgram(3, (int)nmTH.Value);  // Hiển thị thông báo dựa trên kết quả của các thao tác  if (subjectAdded && ltAdded && btAdded && thAdded)  {  MessageBox.Show("Thêm môn học và chương trình liên quan thành công!", "Thành công", MessageBoxButtons.OK, MessageBoxIcon.Information);  \_eduProgramForm.LoadData();  }  else  {  MessageBox.Show("Thêm môn học hoặc một số chương trình thất bại.", "Lỗi", MessageBoxButtons.OK, MessageBoxIcon.Error);  }  ResetInputFields(); // Đặt lại các trường nhập liệu  }  private bool **AddSubject**() // Thêm môn học  {  var subject = new Subject  {  Id\_Sub = txbId\_Sub.Text,  Name\_Sub = txbName\_Sub.Text  };  return SubjectController.Instance().addSubject(subject);  }  private bool **AddEduProgram**(int lessonType, int numHours) // Thêm chương trình giáo dục  {  var eduProgram = new EduProgram  {  FK\_Id\_Sub = txbId\_Sub.Text,  FK\_Id\_LS = lessonType,  NumHour = numHours  };  return EduProgramController.Instance().AddData(eduProgram);  }  private void **ResetInputFields**() // Đặt lại các trường nhập liệu  {  txbId\_Sub.Text = "";  txbName\_Sub.Text = "";  chbLT.Checked = false;  chbBT.Checked = false;  chbTH.Checked = false;  nmLT.Value = 0;  nmBT.Value = 0;  nmTH.Value = 0;  }  private void **txbId\_Sub\_TextChanged**(object sender, EventArgs e)  {  var regex = new Regex("^[a-zA-Z0-9]+$");  // Nếu đầu vào không hợp lệ, cảnh báo hoặc điều chỉnh  try  {  if (!regex.IsMatch(txbId\_Sub.Text) && txbId\_Sub.Text != "")  {  MessageBox.Show("Vui lòng chỉ nhập chữ in hoa và số.", "Lỗi", MessageBoxButtons.OK, MessageBoxIcon.Warning);  // Loại bỏ ký tự cuối cùng (vì nó không hợp lệ)  txbId\_Sub.Text = txbId\_Sub.Text.Substring(0, txbId\_Sub.Text.Length - 1);  txbId\_Sub.SelectionStart = txbId\_Sub.Text.Length; // Đặt con trỏ cuối cùng  }  }  catch (Exception ex)  {  MessageBox.Show("Lỗi không xác định", "Lỗi", MessageBoxButtons.OK, MessageBoxIcon.Error);  }  }  }  } |
| --- |

## 2.4. Controllers

### 2.4.1. EduProgramController

| using System.Data.SqlClient;  using System.Windows.Forms;  using EduManager.Models;  using EduManager.Services;  namespace EduManager.Controllers  {  internal class EduProgramController  {  private static EduProgramController **instance**;  private ConnectDatabase connectDatabase = ConnectDatabase.getInstance();  private EduProgramController() { }  **// Khởi tạo một instance mới nếu chưa tồn tại, ngược lại trả về instance hiện tại**  public static EduProgramController **Instance**()  {  if (instance == null)  {  instance = new EduProgramController();  }  return instance;  }  **// Hiển thị dữ liệu**  public void **ShowData**(DataGridView dgv)  {  string query = "SELECT \* FROM EduProgramView";  dgv.Columns.Clear();  dgv.DataSource = ConnectDatabase.getInstance().ExecuteQuery(query);  }  **// Thêm dữ liệu**  public bool **AddData**(EduProgram ep)  {  string query = "INSERT INTO EduProgram VALUES (@FK\_Id\_Sub, @FK\_Id\_LS, @NumHour)";  SqlParameter[] para = new SqlParameter[]  {  new SqlParameter("FK\_Id\_Sub", ep.FK\_Id\_Sub),  new SqlParameter("FK\_Id\_LS", ep.FK\_Id\_LS),  new SqlParameter("NumHour", ep.NumHour)  };  int rowAffect = connectDatabase.ExecuteNonQuery(query, para);  return rowAffect > 0 ? true : false;  }  **// Sửa dữ liệu**  public bool **EditData**(EduProgram ep)  {  string query = "UPDATE EduProgram " +  "SET NumHour = @NumHour " +  "WHERE FK\_Id\_Sub = @FK\_Id\_Sub AND FK\_Id\_LS = @FK\_Id\_LS;";  SqlParameter[] para = new SqlParameter[]  {  new SqlParameter("FK\_Id\_Sub", ep.FK\_Id\_Sub),  new SqlParameter("FK\_Id\_LS", ep.FK\_Id\_LS),  new SqlParameter("NumHour", ep.NumHour)  };  int rowAffect = connectDatabase.ExecuteNonQuery(query, para);  return rowAffect > 0 ? true : false;  }  **// Xóa dữ liệu**  public bool **RemoveData**(EduProgram ed)  {  string query = "DELETE FROM EduProgram WHERE Id\_EP = @Id\_EP";  SqlParameter[] para = new SqlParameter[]  {  new SqlParameter("Id\_EP", ed.Id\_EP)  };  int rowAffect = connectDatabase.ExecuteNonQuery(query, para);    return rowAffect > 0 ? true : false;  }  **// Xóa tất cả dữ liệu**  public bool **RemoveAllData**(EduProgram ed)  {  string query = "DELETE FROM EduProgram WHERE FK\_Id\_Sub = @FK\_Id\_Sub";  SqlParameter[] para = new SqlParameter[]  {  new SqlParameter("FK\_Id\_Sub", ed.FK\_Id\_Sub)  };  int rowAffect = connectDatabase.ExecuteNonQuery(query, para);  return rowAffect > 0 ? true : false;  }  }  } |
| --- |

### 2.4.2. SubjectController

| using System.Data.SqlClient;  using System.Windows.Forms;  using EduManager.Models;  using EduManager.Services;  namespace EduManager.Controllers  {  internal class SubjectController  {  private static SubjectController instance;  private ConnectDatabase connectDatabase = ConnectDatabase.getInstance();  private **SubjectController**() { }  **// Khởi tạo**  public static SubjectController **Instance**()  {  if (instance == null)  {  instance = new SubjectController();  }  return instance;  }  **// Thêm dữ liệu**  public bool **addSubject**(Subject subject)  {  string query = "INSERT INTO Subjects VALUES (@Id\_Sub , @name\_Sub)";  SqlParameter[] para = new SqlParameter[]  {  new SqlParameter("Id\_Sub", subject.Id\_Sub),  new SqlParameter("name\_Sub", subject.Name\_Sub)  };  int rowAffect = connectDatabase.ExecuteNonQuery(query, para);  // ExecuteNonQuery sẽ trả về số dòng được thay đổi tại Database. Ví dụ không có dữ liệu nào được thay đổi hay là thay đổi lỗi thì số dòng thay đổi sẽ = 0 => Lỗi  return rowAffect > 0 ? true : false;  }  **// Hiển thị dữ liệu**  public void **showData**(DataGridView dgv)  {  string query = "Select \r\n\tId\_Sub AS N'Mã môn học', \r\n\tName\_Sub AS N'Tên môn học'\r\nFROM Subjects";  dgv.DataSource = connectDatabase.ExecuteQuery(query);  }  **// Sửa dữ liệu**  public bool **editData**(Subject s)  {  string query = "UPDATE Subjects SET Name\_Sub = @name\_Sub WHERE Id\_Sub = @Id\_Sub";  SqlParameter[] para = new SqlParameter[]  {  new SqlParameter("Id\_Sub", s.Id\_Sub),  new SqlParameter("name\_Sub", s.Name\_Sub)  };  int rowAffect = connectDatabase.ExecuteNonQuery(query, para);  return (rowAffect > 0 ? true : false);  }  **// Xóa dữ liệu**  public bool **RemoveData**(Subject s)  {  string query = "DELETE FROM Subjects WHERE Id\_Sub = @Id\_Sub";  SqlParameter[] para = new SqlParameter[]  {  new SqlParameter("Id\_Sub", s.Id\_Sub)  };  int rowAffect = connectDatabase.ExecuteNonQuery(query, para);  return rowAffect > 0 ? true : false;  }  }  } |
| --- |

### 