BÀI THỰC HÀNH SỐ 8 (8 tiết)

MÔ HÌNH ĐA TẦNG TRONG PHÁT TRIỂN ỨNG DỤNG

I. Mục tiêu:

Bài thực hành này giúp sinh viên làm quen với mô hình đa tầng trong phát triển ứng dụng. Bài thực hành sẽ giúp sinh viên xây dựng và phát triển một số chức năng của ứng dụng theo mô hình ba tầng là Tầng giao diện, Tầng xử lý logic và tầng Truy xuất dữ liệu.

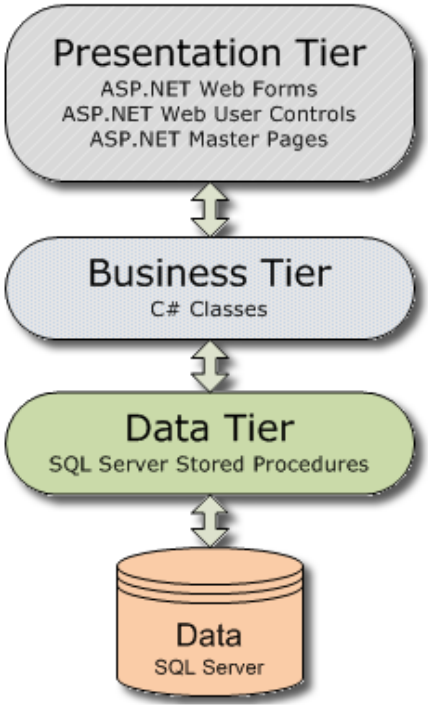
II. Thực hành

1. Giới thiệu mô hình đa tầng

Mô hình đa tầng (*n-tier*) là một kiến trúc phần mềm được phân chia thành nhiều các thành phần, trong đó các phần như giao diện người dùng (*User Interface - UI*), quy tắc xử lý (*Business Logic – BL*), và lưu trữ dữ liệu (*Data Access – DA*) được phát triển như là những mô đun độc lập. Các mô đun này có thể được xây dựng và phát triển trên các nền tảng riêng biệt, chúng được nối với nhau thông qua việc giao tiếp qua các tập tin dạng thư viện (*.dll).

Khi triển khai ứng dụng ở mức vật lý, kiến trúc đa tầng thường đưa về dạng kiến trúc với ba tầng riêng biệt bao gồm:

- Tầng giao diện (*Presentation*): Dùng để hiển thị các thành phần giao diện và tương tác với người dùng như tiếp nhận thông tin nhập, thông báo kết quả, thông báo lỗi;
- Tầng xử lý (*Business Logic*): Thực hiện các hành động nghiệp vụ của phần mềm như tính toán, thêm, xóa, sửa dữ liệu;
- Tầng truy cập dữ liệu (*Data Access*): Bao gồm hai thành phần: thứ nhất là thành phần trực tiếp truy xuất dữ liệu thường là các lệnh, các hàm trong hệ quản trị cơ sở dữ liệu; Thành phần thứ hai truy xuất là các lớp tổng quát dùng để gọi các hàm và lệnh từ cơ sở dữ liệu.



Để hiểu hơn về mô hình ba tầng, sinh viên có thể tham khảo thêm các link sau:

https://topdev.vn/blog/mo-hinh-3-lop-la-gi/

https://viblo.asia/p/gioi-thieu-mo-hinh-3-lop-trong-c-gDVK2Q9w5Lj

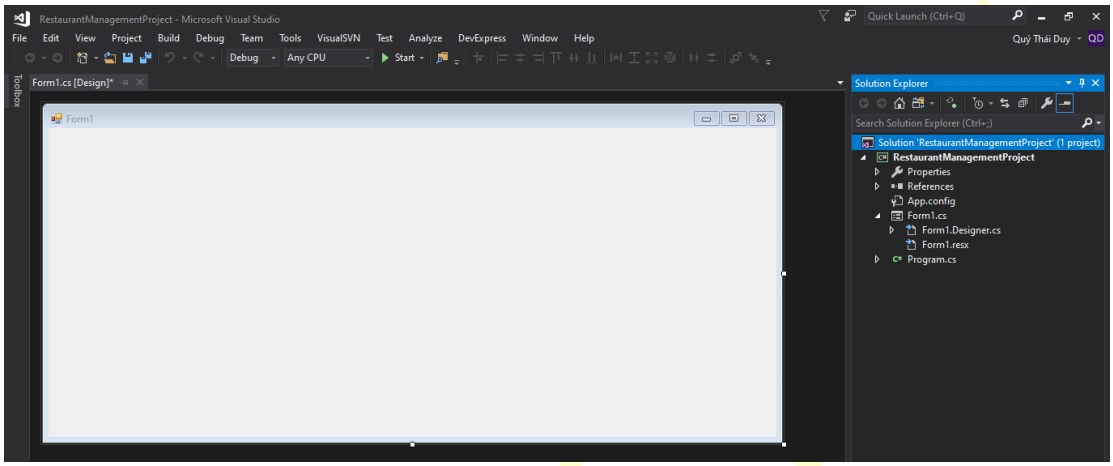
https://stackify.com/n-tier-architecture/

https://docs.microsoft.com/en-us/visualstudio/data-tools/walkthrough-creating-an-n-tier-data-application?view=vs-2019

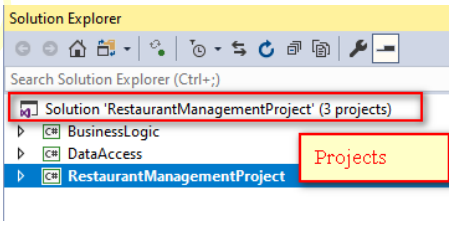
2. Tạo mô hình đa tầng với Visual Studio

Khi xây dựng mô hình đa tầng, Solution được tạo ra khi tạo tầng đầu tiên (*Web, Form*). Các tầng còn lại có thể thêm mới hoặc thêm từ dự án có sẵn (tập tin *.dll). Các tầng được liên kết với nhau thông qua việc kết nối bởi các tập tin thư viện (*.dll). Để tạo dự án với mô hình ba tầng trên Windows Form được thực hiện theo các bước như sau:

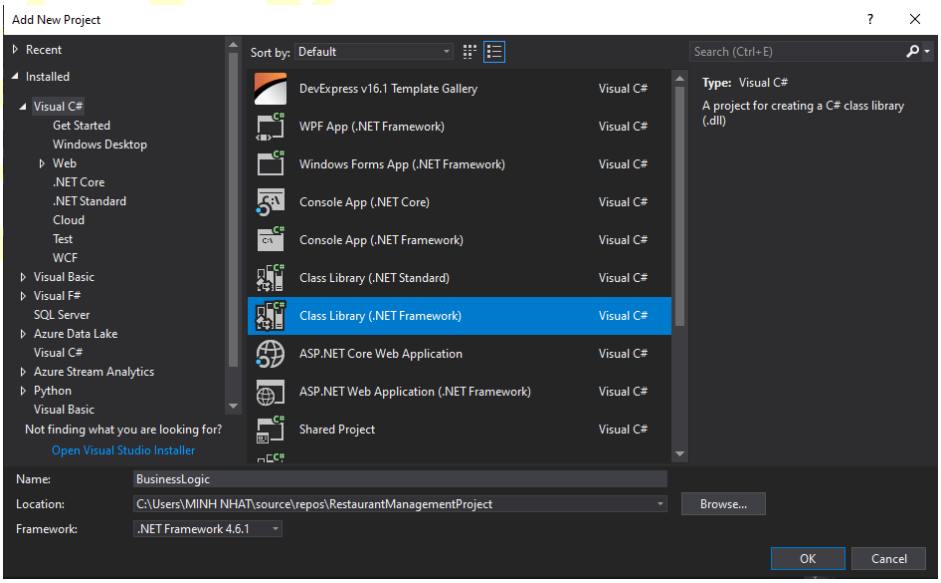
- **Bước 1**: Mở Visual Studio, tạo dự án Windows Form bằng ngôn ngữ C#, đặt tên dự án này là *RestaurantManagementProject*:



Lưu ý: Sau khi tạo xong dự án, ta thấy có một *Solution* và một *Project* được tạo ra. Từ đây, *Solution* là giải pháp chung cho cả dự án, mỗi một *Project* đại diện cho một tầng. Tầng vừa tạo chính là tầng Giao diện, dùng để xử lý các vấn đề liên quan đến giao diện chương trình.

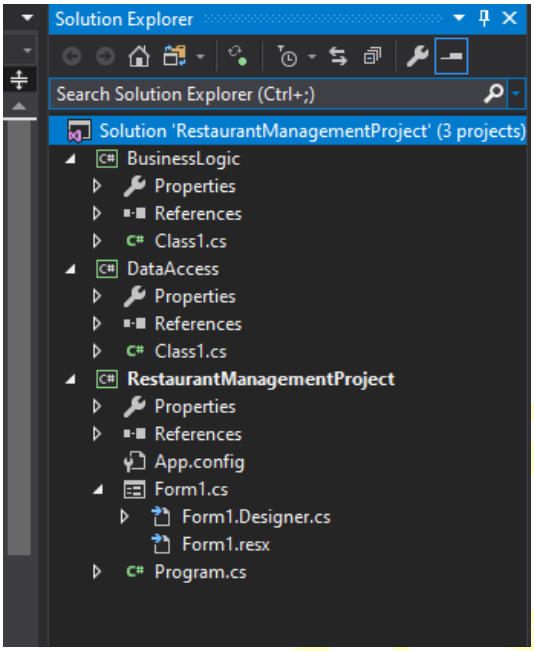


- **Bước 2**: Tạo tầng xử lý bằng cách click phải lên *Solution*, chọn *Add*, chọn *New Project*. Trong hộp thoại hiện lên chọn kiểu dự án là *Class Library* (.Net Framework), đặt tên dự án là *BusinessLogic*.

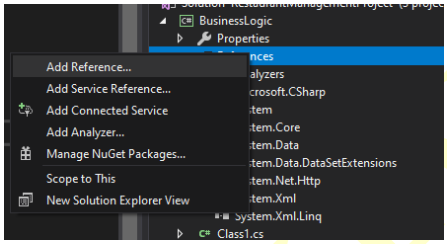


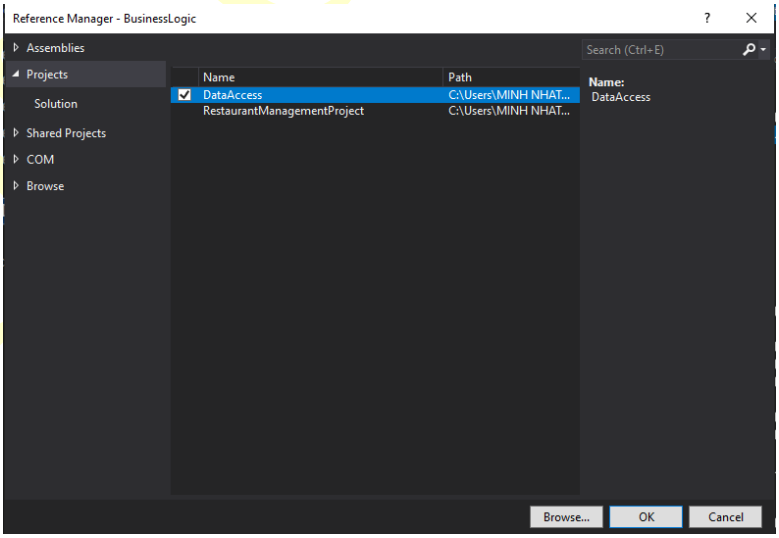
Lưu ý: Đây là một dự án dạng thư viện, thư viện này được tạo ra dưới dạng các tập tin *.dll khi chạy ứng dụng.

- **Bước 3**: Thực hiện thao tác tương tự Bước 2 để tạo tầng truy cập dữ liệu có tên là *DataAccess*. Kết quả trên Solution Explorer được thể hiện như sau:

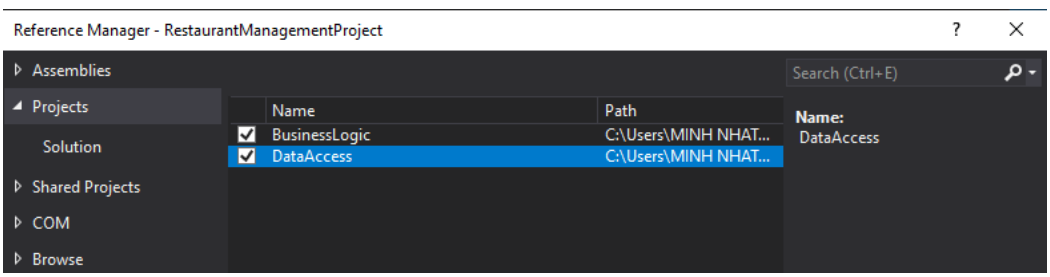


- **Bước 4**: Để có thể gọi các phương thức của tầng *DataAccess* từ tầng *BusinessLogic*, click phải lên thành phần *References* của Project *BusinessLogic*, chọn *Add Reference…*, đánh dấu tích và *DataAccess*:





Thực hiện tương tự để kết nối hai tầng *BusinessLayer* và *DataAccess* với tầng giao diện:



Sau khi đã kết nối các tầng, chúng ta có thể gọi các đối tượng, phương thức, và thuộc tính từ các tầng với nhau bằng cách sử dụng lệnh *using* như sau:

```
using BusinessLogic; hoặc using DataAccess;
```

Chi tiết về cách xây dựng các tầng có thể xem tại các mục tiếp theo.

3. Tầng truy cập dữ liệu (*DataAccess*)

Tầng *DataAccess* bao gồm hai thành phần là Cơ sở dữ liệu (*Data*) xây dựng bằng SQL Server và phần Truy xuất (Access) xây dựng bằng ngôn ngữ C#.

- **Bước 1: Xây dựng cơ sở dữ liệu**

Cơ sở dữ liệu được dùng trong mô hình này là *RestaurantManagement* (sinh viên xem lại Lab 1 hoặc phần Phụ lục của giáo trình Lập trình cơ sở dữ liệu). Để cho đơn giản, hai bảng được dùng để minh họa là *Category* và *Food*, đây là hai bảng có kết nối khóa ngoại, có giá trị ID tự tăng làm khóa chính. Các chức năng khác, sinh viên dựa theo cách xây dựng từ hai bảng này để phát triển thêm. Các thủ tục (*Store Procedure*) cần xây dựng như sau:

Tên thủ tục	Kiểu trả về	Giải thích
`Category_GetAll`	Toàn bộ mẫu tin bảng Category	Thủ tục này trả về tất cả mẫu tin trong bảng Category.
`Category_InsertUpdateDelete`	Kiểu số nguyên	Thủ tục này nhận vào các tham số bảng Category, và biến action, nếu action = 0 thì thêm, nếu bằng 1 thì sửa và nếu bằng 2 thì xoá.
`Food_GetAll`	Toàn bộ mẫu tin bảng Food	Thủ tục này trả về tất cả mẫu tin trong bảng Food.
`Food_InsertUpdateDelete`	Kiểu số nguyên	Thủ tục này nhận vào các tham số bảng Food, và biến action, nếu action = 0 thì thêm, nếu bằng 1 thì sửa và nếu bằng 2 thì xoá.

Mã nguồn của các thủ tục trên được viết trong SQL Server như sau:

```
--Thủ tục lấy tất cả dữ liệu bảng Category
CREATE PROCEDURE [dbo].[Category_GetAll]
AS

 SELECT * FROM Category

-----------------------------------------
--Thủ tục lấy tất cả dữ liệu bảng Food
ALTER PROCEDURE [dbo].[Food_GetAll]
AS

 SELECT * FROM Food

-----------------------------------------
```

```
-- Thủ tục thêm, xóa, sửa bảng Category
ALTER PROCEDURE [dbo].[Category_InsertUpdateDelete]
    @ID int output, -- Biến ID tự tăng, khi thêm xong phải lấy ra
    @Name nvarchar(200),
    @Type int,
    @Action int -- Biến cho biết thêm, xóa, hay sửa
AS
-- Nếu Action = 0, thực hiện thêm dữ liệu
IF @Action = 0
 BEGIN
       INSERT INTO [Category] ([Name],[Type])
       VALUES (@Name, @Type)
       SET @ID = @@identity -- Thiết lập ID tự tăng
 END
-- Nếu Action = 1, thực hiện cập nhật dữ liệu
ELSE IF @Action = 1
 BEGIN
       UPDATE [Category] SET [Name] = @Name, [Type]=@Type
       WHERE [ID] = @ID
 END
-- Nếu Action = 2, thực hiện xóa dữ liệu
ELSE IF @Action = 2
 BEGIN
       DELETE FROM [Category] WHERE [ID] = @ID

 END
-----------------------------------------
-- Thủ tục thêm, xóa, sửa bảng Food
ALTER PROCEDURE [dbo].[Food_InsertUpdateDelete]
    @ID int output, -- Biến ID tự tăng, khi thêm xong phải lấy ra
    @Name nvarchar(1000),
    @Unit nvarchar(100),
    @FoodCategoryID int,
    @Price int,
    @Notes nvarchar(3000),
    @Action int -- Biến cho biết thêm, xóa, hay sửa
AS
IF @Action = 0 -- Nếu Action = 0, thêm dữ liệu
 BEGIN
     INSERT INTO [Food] ([Name],[Unit],[FoodCategoryID],[Price],[Notes])
     VALUES (@Name, @Unit,@FoodCategoryID,@Price,@Notes)
     SET @ID = @@identity -- Thiết lập ID tự tăng
 END
ELSE IF @Action = 1 -- Nếu Action = 1, cập nhật dữ liệu
 BEGIN
     UPDATE [Food]
     SET [Name] = @Name,[Unit]=@Unit,[FoodCategoryID]=@FoodCategoryID,
         [Price]=@Price,[Notes]=@Notes
     WHERE [ID] = @ID
 END
ELSE IF @Action = 2 -- Nếu Action = 2, xóa dữ liệu
 BEGIN
     DELETE FROM [Food] WHERE [ID] = @ID
 END
```

- **Bước 2: Xây dựng các lớp tham số chung**

Click phải lên Project *DataAccess*, chọn Add, chọn Class, đặt tên lớp là *Ultilities*, viết mã nguồn như sau:

```
public class Ultilities
{
    // lấy chuỗi kết nối từ tập tin App.Config
    private static string StrName = "ConnectionStringName";
    public static string ConnectionString = ConfigurationManager
                                            .ConnectionStrings[StrName]
                                           .ConnectionString;
    // Các biến của bảng Food
    public static string Food_GetAll = "Food_GetAll";
    public static string Food_InsertUpdateDelete = "Food_InsertUpdateDelete";
    // Các biến của bảng Food
    public static string Category_GetAll = "Category_GetAll";
    public static string Category_InsertUpdateDelete=
                                        "Category_InsertUpdateDelete";
}
```

Lưu ý: Để gọi được lớp *ConfigurationManager* cần phải thêm thư viện *System.Configuration* vào dự án (click phải lên *References*, chọn *Add Reference…*, tìm đến thư viện *System.Configuration* và thêm vào dự án), sau đó gọi thư viện này bằng lệnh using:

```
using System.Configuration;
```

- **Bước 3: Xây dựng các lớp ánh xạ bảng Food và Category**

Click phải lên Project *DataAccess*, chọn Add, chọn Class, tạo hai tập tin chứa hai lớp với tên lần lượt là *Food.cs* và *Category.cs*. Khai báo các thuộc tính và kiểu dữ liệu như sau:

```
/// <summary>
/// Lớp ánh xạ bảng Category
/// </summary>
public class Category
{
    // ID của bảng, tự tăng trong CSDL
    public int ID { get; set;}
    // Tên của loại thức ăn
    public string Name { get; set; }
    // Kiểu: 0 là đố uống; 1 là thức ăn...
    public int Type { get; set;}
}
/// <summary>
/// Lớp ánh xạ bảng Food
/// </summary>
public class FoodRecord
{
    // ID của bảng Food
    public int ID { get; set; }
    // Tên loại đồ ăn, thức uống
```

```
        public string Name { get; set;}
        // Đơn vị tính
        public string Unit { get; set; }
        //Loại thức ăn, ứng với bảng ở trên
        public int FoodCategoryID { get; set; }
        // Giá
        public int Price { get; set; }
        // Ghi chú
        public string Notes { get; set; }

    }
```

- **Bước 4: Xây dựng các lớp truy xuất dữ liệu**

Bảng Category có hai thủ tục dùng để lấy tất cả (*GetAll*) và thêm, xoá, sửa mẫu tin (*InsertUpdateDelete*) thì sẽ có hai phương thức tương ứng là *GetAll()* và *Insert_Update_Delete*(…). Phương thức GetAll thì không truyền tham số và trả về là một danh cách còn phương thức *Insert_Update_Delete* thì truyền vào một đối tượng là ánh xạ của bảng và một biến *action*:

```
//Lớp quản lý Category: DA = DataAccess
public class CategoryDA
{
    //Phương thức lấy hết dữ liệu theo thủ tục Food_GetAll
    public List<Category> GetAll()
    {
        // Khai báo đối tượng SqlConnection và mở kết nối
        // Đối tượng SqlConnection truyền vào chuỗi kết nối trong App.config
        SqlConnection sqlConn=new SqlConnection(Ultilities.ConnectionString);
        sqlConn.Open();
        //Khai báo đối tượng SqlCommand có kiểu xử lý là StoredProcedure
        SqlCommand command = sqlConn.CreateCommand();
        command.CommandType = CommandType.StoredProcedure;
        command.CommandText = Ultilities.Category_GetAll;
        // Đọc dữ liệu, trả về danh sách các đối tượng Category
        SqlDataReader reader = command.ExecuteReader();
        List<Category> list = new List<Category>();
        while (reader.Read())
        {
            Category category = new Category();
            category.ID = Convert.ToInt32(reader["ID"]);
            category.Name = reader["Name"].ToString();
            category.Type = Convert.ToInt32(reader["Type"]);
            list.Add(category);
        }
        // Đóng kết nối và trả về danh sách
        sqlConn.Close();
        return list;
    }
//Phương thức thêm, xoá, sửa theo thủ tục Category_InsertUpdateDelete
    public int Insert_Update_Delete(Category category, int action)
    {
        // Khai báo đối tượng SqlConnection và mở kết nối
        // Đối tượng SqlConnection truyền vào chuỗi kết nối trong App.config
        SqlConnection sqlConn=new SqlConnection(Ultilities.ConnectionString);
```

```
            sqlConn.Open();
            //Khai báo đối tượng SqlCommand có kiểu xử lý là StoredProcedure
            SqlCommand command = sqlConn.CreateCommand();
            command.CommandType = CommandType.StoredProcedure;
            command.CommandText = Ultilities.Category_InsertUpdateDelete;
            // Thêm các tham số cho thủ tục; Các tham số này chính là các tham số
trong thủ tục;
        //ID là tham số có giá trị lấy ra khi thêm và truyền vào khi xoá, sửa
            SqlParameter IDPara = new SqlParameter("@ID", SqlDbType.Int);
            IDPara.Direction = ParameterDirection.InputOutput; // Vừa vào vừa ra
            command.Parameters.Add(IDPara).Value = category.ID;
            command.Parameters.Add("@Name", SqlDbType.NVarChar,200)
                              .Value = category.Name;
            command.Parameters.Add("@Type", SqlDbType.Int)
                              .Value = category.Type;
            command.Parameters.Add("@Action", SqlDbType.Int)
                              .Value = action;
            // Thực thi lệnh
            int result = command.ExecuteNonQuery();
            if (result > 0) // Nếu thành công thì trả về ID đã thêm
                return (int)command.Parameters["@ID"].Value;
            return 0;
        }

    }

    //Lớp quản lý Food: DA = DataAccess
    public class FoodDA
    {
        // Phương thức lấy hết dữ liệu theo thủ tục Food_GetAll
        public List<Food> GetAll()
        {
            //Khai báo đối tượng SqlConnection và mở kết nối
            //Đối tượng SqlConnection truyền vào chuỗi kết nối trong App.config
            SqlConnection sqlConn=new SqlConnection(Ultilities.ConnectionString);
            sqlConn.Open();
            //Khai báo đối tượng SqlCommand có kiểu xử lý là StoredProcedure
            SqlCommand command = sqlConn.CreateCommand();
            command.CommandType = CommandType.StoredProcedure;
            command.CommandText = Ultilities.Food_GetAll;
            // Đọc dữ liệu, trả về danh sách các đối tượng Food
            SqlDataReader reader = command.ExecuteReader();
            List<Food> list = new List<Food>();
            while (reader.Read())
            {
                Food food = new Food();
                food.ID = Convert.ToInt32(reader["ID"]);
                food.Name = reader["Name"].ToString();
                food.Unit = reader["Unit"].ToString();
                food.FoodCategoryID = Convert.ToInt32(reader["FoodCategoryID"]);
                food.Price = Convert.ToInt32(reader["Price"]);
                food.Notes = reader["Notes"].ToString();
                list.Add(food);
            }
            // Đóng kết nối và trả về danh sách
```

```
            sqlConn.Close();
            return list;
        }
        // Phương thức thêm, xoá, sửa theo thủ tục Food_InsertUpdateDelete
        public int Insert_Update_Delete(Food food, int action)
        {
            // Khai báo đối tượng SqlConnection và mở kết nối
            // Đối tượng SqlConnection truyền vào chuỗi kết nối trong App.config
            SqlConnection sqlConn=new SqlConnection(Ultilities.ConnectionString);
            sqlConn.Open();
            //Khai báo đối tượng SqlCommand có kiểu xử lý là StoredProcedure
            SqlCommand command = sqlConn.CreateCommand();
            command.CommandType = CommandType.StoredProcedure;
            command.CommandText = Ultilities.Food_InsertUpdateDelete;
            // Thêm các tham số cho thủ tục; Các tham số này chính là các tham số
trong thủ tục;
        //ID là tham số có giá trị lấy ra khi thêm và truyền vào khi xoá, sửa
        SqlParameter IDPara = new SqlParameter("@ID", SqlDbType.Int);
        IDPara.Direction = ParameterDirection.InputOutput;
            command.Parameters.Add(IDPara).Value = food.ID;
            //Các biến còn lại chỉ truyền vào
            command.Parameters.Add("@Name", SqlDbType.NVarChar, 1000)
                              .Value = food.Name;
            command.Parameters.Add("@Unit", SqlDbType.NVarChar)
                              .Value = food.Unit;
            command.Parameters.Add("@FoodCategoryID", SqlDbType.Int)
                              .Value = food.FoodCategoryID;
            command.Parameters.Add("@Price", SqlDbType.Int)
                              .Value = food.Price;
            command.Parameters.Add("@Notes", SqlDbType.NVarChar,3000)
                              .Value = food.Notes;
            command.Parameters.Add("@Action", SqlDbType.Int)
                              .Value = action;
            int result = command.ExecuteNonQuery();
            // Thực thi lệnh
            if (result > 0) // Nếu thành công thì trả về ID đã thêm
                return (int)command.Parameters["@ID"].Value;
            return 0;
        }
    }
```

4. Tầng xử lý Logic (*BusinessLogic)*

Tầng *BusinessLogic* là tầng trung gian giữa tầng *DataAccess* và tầng *Presentation*. Tầng này có thể thêm một số phương thức khác như Tìm kiếm, tìm theo khoá chính, tìm theo trường…

- **Bước 1: Lớp *CategoryBL***

Lớp *CategoryBL* là lớp dùng để thực hiện các chức năng của bảng *Category*, các chức năng chủ yếu của bảng này bao gồm các phương thức cơ bản như: lấy hết, thêm, xoá, sửa…

```
// Lớp CategoryBL có các phương thức xử lý bảng Category
public class CategoryBL
{
    //Đối tượng CategoryDA từ DataAccess
    CategoryDA categoryDA = new CategoryDA();
    //Phương thức lấy hết dữ liệu
    public List<Category> GetAll()
    {
        return categoryDA.GetAll();
    }
    //Phương thức thêm dữ liệu
    public int Insert(Category category)
    {
        return categoryDA.Insert_Update_Delete(category, 0);
    }
    //Phương thức cập nhật dữ liệu
    public int Update(Category category)
    {
        return categoryDA.Insert_Update_Delete(category, 1);
    }
    //Phương thức xoá dữ liệu truyền vào ID
    public int Delete(Category category)
    {
        return categoryDA.Insert_Update_Delete(category, 2);
    }
}
```

- **Bước 2: Lớp *FoodBL***

```
// Lớp FoodBL có các phương thức xử lý bảng Food
public class FoodBL
{
    //Đối tượng CategoryDA từ DataAccess
    FoodDA foodDA = new FoodDA();
    //Phương thức lấy hết dữ liệu
    public List<Food> GetAll()
    {
        return foodDA.GetAll();
    }
    // Phương thức lấy về đối tượng Food theo khoá chính
    public Food GetByID(int ID)
    {
        // Lấy hết
        List<Food> list = GetAll();
        // Duyệt để tìm kiếm
        foreach (var item in list)
        {
            if (item.ID == ID) // Nếu gặp khoá chính
                return item; // thì trả về kết quả
        }
        return null;
    }
    //Phương thức tìm kiếm theo khoá
    public List<Food> Find(string key)
    {
```

```
            List<Food> list = GetAll(); // Lấy hết
            List<Food> result = new List<Food>();
            // Duyệt theo danh sách
            foreach (var item in list)
            {
                // Nếu từng trường chứa từ khoá
                if (item.ID.ToString().Contains(key)
                    || item.Name.Contains(key)
                    || item.Unit.Contains(key)
                    || item.Price.ToString().Contains(key)
                    || item.Notes.Contains(key))
                    result.Add(item); // Thì thêm vào danh sách kết quả
            }
            return result;
        }
        //Phương thức thêm dữ liệu
        public int Insert(Food food)
        {
            return foodDA.Insert_Update_Delete(food, 0);
        }
        //Phương thức cập nhật dữ liệu
        public int Update(Food food)
        {
            return foodDA.Insert_Update_Delete(food, 1);
        }
        //Phương thức xoá dữ liệu với ID cho trước
        public int Delete(Food food)
        {
            return foodDA.Insert_Update_Delete(food,2);
        }
    }
```

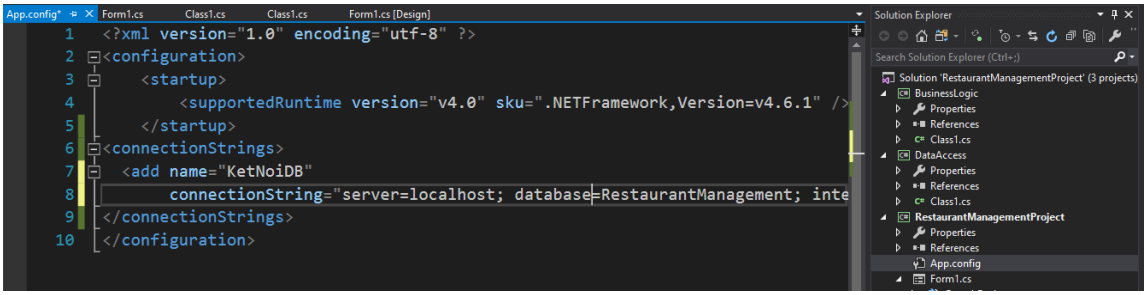
Lớp *CategoryBL* và *FoodBL* nếu có nhu cầu thêm, có thể viết thêm các phương thức khác bổ sung. Khi viết xong tầng *BusinessLogic*, có thể kiểm tra lỗi và tạo tập tin *.dll bằng cách click phải lên Project *BusinessLogic*, chọn *Build*, khi đó nếu có lỗi sai, hệ thống sẽ báo để sửa lỗi trước khi viết qua tầng mới.

5. Tầng Giao diện người dùng (*UserInterface*)

- **Bước 1: Tạo chuỗi kết nối**

  Thêm thẻ sau vào trong tập tin *App.config* của dự án:

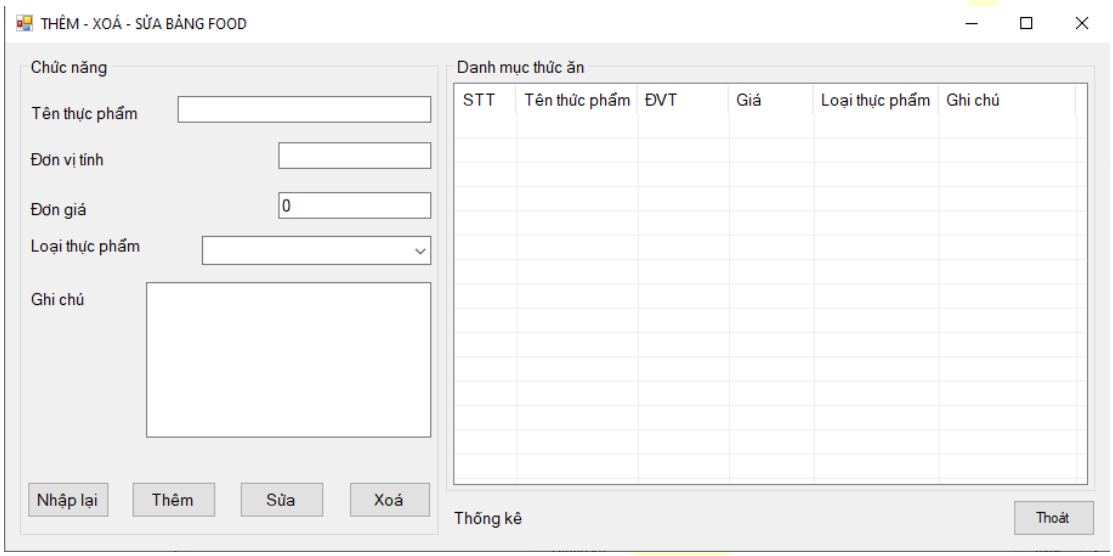
```
<add name=" ConnectionStringName" connectionString="server=localhost;
database=RestaurantManagement; integrated security=true;"/>
```



Lưu ý: Giá trị "*ConnectionStringName*" là tên của chuỗi kết nối, các giá trị được in đậm là các giá trị thay đổi theo tên của Server và tên cơ sở dữ liệu trong SQL Server.

- **Bước 2: Giao diện chương trình**

Giao diện chương trình minh hoạ ở đây dùng để quản lý bảng Food, bao gồm các chức năng chính như: Thêm, xoá, sửa, tìm kiếm… Để hiển thị thông tin, có thể sử dụng ListView hoặc DataGridView, ở đây sử dụng ListView. Dữ liệu Loại thực phẩm được đọc từ bảng Category và đưa vào Combobox:



Các thành phần trên giao diện được đặt các thuộc tính như sau:

Thành phần	Tên thành phần	Thuộc tính
Form	frmFood	Text= THÊM - XOÁ - SỬA BẢNG FOOD
GroupBox	grpLeft	Text = Chức năng; Anchor = Top, Bottom, Left
GroupBox	grpRight	Text = Danh mục thức ăn; Anchor = Top, Bottom, Left, Right
Label	Tên mặc định	Text là tiêu đề cho các thành phần như "Tên thực phẩm", "Đơn vị tính", "Đơn giá", "Loại thực phẩm", "Ghi chú"
TextBox	txtName txtUnit txtPrice	Anchor= Top, Left
ComboBox	cbbCategory	Anchor= Top, Left
TextBox	txtNotes	Anchor= Top, Left; Multiline = True
Button	cmdClear cmdAdd cmdUpdate cmdDelete	Anchor= Bottom; Text là tiêu đề cho các nút bấm như "Nhập lại", "Thêm", "Sửa", "Xoá".
ListView	lsvFood	Anchor= Top, Bottom, Left, Right; Columns={STT, Tên thực phẩm, ĐVT, Giá, Loại thực phẩm, Ghi chú};

		FullRowSelect= true; GridLines= true; MultiSelect= false; View= Details
Label	lblStatistic	Text= Thống kê
Button	cmdExit	Text= Thoát; Anchor= Bottom, Right

Sự kiện cho nút Thoát và Nhập lại được viết như sau:

```
private void cmdExit_Click(object sender, EventArgs e)
{
    Application.Exit();
}
private void cmdClear_Click(object sender, EventArgs e)
{
    //Gán các ô bằng giá trị mặc định
    txtName.Text = "";
    txtPrice.Text = "0";
    txtUnit.Text = "";
    txtNotes.Text = "";
    // Thiết lập index = 0 cho ComboBox
    if(cbbCategory.Items.Count >0)
        cbbCategory.SelectedIndex = 0;
}
```

- **Bước 3: Xây dựng các phương thức tự động tải dữ liệu**

Để gọi các đối tượng từ tầng *DataAccess* và *BusinessLogic*, phải sử dụng lệnh using như sau trong lớp frmFood:

```
using BusinessLogic;
using DataAccess;
```

Đầu tiên, khai báo các đối tượng toàn cục như sau trong lớp frmFood:

```
// Danh sách toàn cục bảng Category
List<Category> listCategory = new List<Category>();
// Danh sách toàn cục bảng Food
List<Food> listFood = new List<Food>();
// Đối tượng Food đang chọn hiện hành
Food foodCurrent = new Food();
```

Tiếp theo, xây dựng các phương thức tự động tải dữ liệu và gọi trong sự kiện Form_Load:

```
private void Form1_Load(object sender, EventArgs e)
{
    //Đổ dữ liệu vào ComboBox
    LoadCategory();
    // Đổ dữ liệu vào ListView
    LoadFoodDataToListView();
}
```

```
private void LoadCategory()
{
    //Gọi đối tượng CategoryBL từ tầng BusinessLogic
    CategoryBL categoryBL = new CategoryBL();
    // Lấy dữ liệu gán cho biến toàn cục listCategory
    listCategory = categoryBL.GetAll();
    // Chuyển vào Combobox với dữ liệu là ID, hiển thị là Name
    cbbCategory.DataSource = listCategory;
    cbbCategory.ValueMember = "ID";
    cbbCategory.DisplayMember = "Name";
 }
 public void LoadFoodDataToListView()
 {
     //Gọi đối tượng FoodBL từ tầng BusinessLogic
     FoodBL foodBL = new FoodBL();
     // Lấy dữ liệu
     listFood= foodBL.GetAll();
     int count = 1; // Biến số thứ tự
     // Xoá dữ liệu trong ListView
     lsvFood.Items.Clear();
     // Duyệt mảng dữ liệu để đưa vào ListView
     foreach (var food in listFood)
        {
        // Số thứ tự
        ListViewItem item = lsvFood.Items.Add(count.ToString());
        // Đưa dữ liệu Name, Unit, price vào cột tiếp theo
        item.SubItems.Add(food.Name);
        item.SubItems.Add(food.Unit);
        item.SubItems.Add(food.Price.ToString());
        // Theo dữ liệu của bảng Category ID, lấy Name để hiển thị
        string foodName = listCategory
                          .Find(x => x.ID == food.FoodCategoryID).Name;
        item.SubItems.Add(foodName);
        // Đưa dữ liệu Notes vào cột cuối
        item.SubItems.Add(food.Notes);
        count++;
    }

 }
```

- **Bước 4: Sự kiện click lên dòng của ListView**

Khi người dùng click chuột vào một dòng trên *ListView*, dữ liệu sẽ được lấy ra và gán cho biến của đối tượng Food hiện hành, đồng thời giá trị của các trường được đưa vào các ô nhập. Trong sự kiện OnClick của ListView mã nguồn được viết như sau:

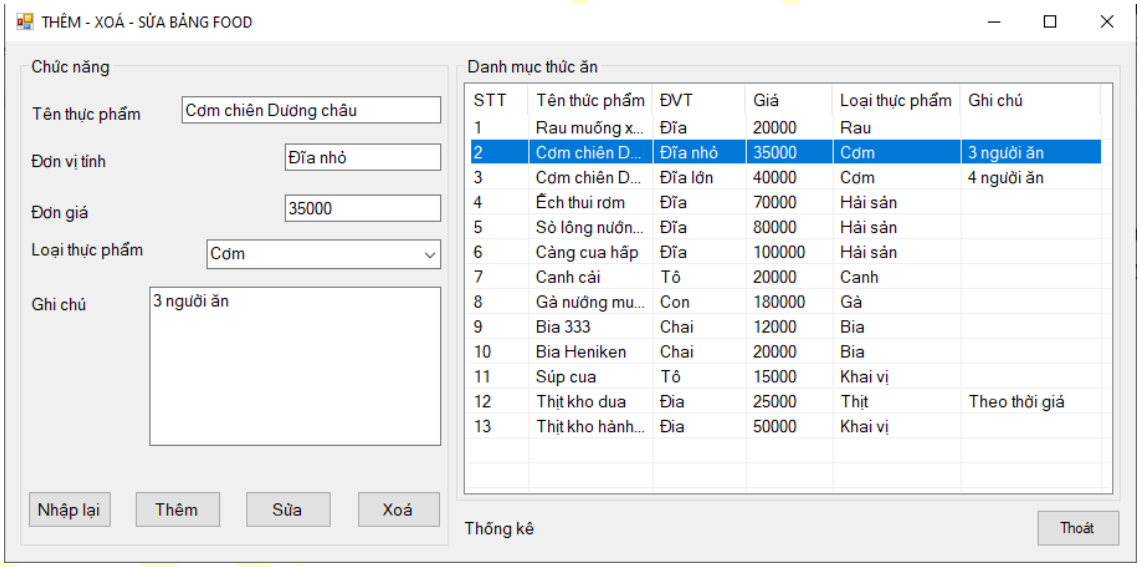
```
private void lsvFood_Click(object sender, EventArgs e)
{
    // Duyệt toàn bộ dữ liệu trong ListView
    for (int i = 0; i < lsvFood.Items.Count; i++)
    {
        // Nếu có dòng được chọn thì lấy dòng đó
        if (lsvFood.Items[i].Selected)
        {
```

```
                // Lấy các tham số và gán dữ liệu vào  các ô
                foodCurrent = listFood[i];
                txtName.Text = foodCurrent.Name;
                txtUnit.Text = foodCurrent.Unit;
                txtPrice.Text = foodCurrent.Price.ToString();
                txtNotes.Text = foodCurrent.Notes;
                // Lấy index của Combobox theo FoodCategoryID
                cbbCategory.SelectedIndex = listCategory
                            .FindIndex(x => x.ID == foodCurrent.FoodCategoryID);
            }

        }
}
```

Lưu ý: Do đã thiết lập từ trước các thuộc tính của *ListView* là *FullRowSelect*= true và *MultiSelect*= false (xem bảng ở Bước 2) nên khi người dùng nhấp chuột vào vùng dữ liệu thì chỉ 1 dòng được chọn và dòng đó phải chọn cả dòng.

Sau khi viết hết lệnh, nhấn F5 và chạy thử chương trình để xem dữ liệu được hiển thị trong *ListView* và *ComboBox*. Click chọn dòng trong *ListView* để dữ liệu được đưa vào trong các ô như sau:



- **Bước 5: Xử lý nút thêm**

Trước tiên, viết phương thức thêm dữ liệu:

```
/// <summary>
/// Phương thức thêm dữ liệu cho bảng Food
/// </summary>
/// <returns>Trả về số dương nếu thành công, ngược lại trả về số âm</returns>
public int InsertFood()
{
    //Khai báo đối tượng Food từ tầng DataAccess
    Food food = new Food();
    food.ID = 0;
    // Kiểm tra nếu các ô nhập khác rỗng
```

```
        if (txtName.Text == "" || txtUnit.Text == "" || txtPrice.Text == "")
            MessageBox.Show("Chưa nhập dữ liệu cho các ô, vui lòng nhập lại");
        else {
            //Nhận giá trị Name, Unit, và Notes từ người dùng nhập vào
            food.Name = txtName.Text;
            food.Unit = txtUnit.Text;
            food.Notes = txtNotes.Text;
           // Giá trị price là giá trị số nên cần bắt lỗi khi người dùng nhập sai
            int price = 0;
            try
            {
                // Cố gắng lấy giá trị
                price = int.Parse(txtPrice.Text);
            }
            catch
            {
                // Nếu sai, gán giá = 0
                price = 0;
            }
            food.Price = price;
            // Giá trị FoodCategoryID được lấy từ ComboBox
           food.FoodCategoryID = int.Parse(cbbCategory.SelectedValue.ToString());
            // Khao báo đối tượng FoodBL từ tầng Business
            FoodBL foodBL = new FoodBL();
            // Chèn dữ liệu vào bảng
            return foodBL.Insert(food);
        }
        return -1;

    }
```

Lưu ý: Vì *ID* là tự tăng nên giá trị nhận vào được gán mặc định (bằng 0).

Sự kiện khi nhấn nút Thêm được viết như sau:

```
    private void cmdAdd_Click(object sender, EventArgs e)
    {
        // Gọi phương thức thêm dữ liệu
        int result = InsertFood();
        if(result > 0) // Nếu thêm thành công
        {
            // Thông báo kết quả
            MessageBox.Show("Thêm dữ liệu thành công");
            // Tải lại dữ liệu cho ListView
            LoadFoodDataToListView();
        }
        // Nếu thêm không thành công thì thông báo cho người dùng
        else MessageBox.Show("Thêm dữ liệu không thành công. Vui lòng kiểm tra lại
dữ liệu nhập");
    }
```

- **Bước 6: Xử lý nút xoá**

```
private void cmdDelete_Click(object sender, EventArgs e)
{
    // Hỏi người dùng có chắc chắn xoá hay không? Nếu đồng ý thì
    if(MessageBox.Show("Bạn có chắc chắn muốn xoá mẫu tin này?","Thông báo",
        MessageBoxButtons.YesNo,MessageBoxIcon.Warning) == DialogResult.Yes)
    {
        // Khai báo đối tượng FoodBL từ BusinessLogic
        FoodBL foodBL = new FoodBL();
        if (foodBL.Delete(foodCurrent) > 0)// Nếu xoá thành công
        {
            MessageBox.Show("Xoá thực phẩm thành công");
            // Tải tữ liệu lên ListView
            LoadFoodDataToListView();
        }
        else MessageBox.Show("Xoá không thành công");
    }
}
```

Lưu ý: Thuộc tính *foodCurrent* là thuộc tính được khai báo ở Bước 3 và mỗi khi Click chuột lên dòng của ListView thì thuộc tính này sẽ lưu giữ giá trị được chọn.

- **Bước 7: Xử lý nút sửa**

Khi nhấn vào nút Sửa, hệ thống sẽ lấy giá trị hiện hành (biến *foodCurrent*) làm giá trị để sửa, ID được giữ lại, các giá trị còn lại được cập nhật bằng cách dựa trên dữ liệu mà người dùng nhập vào. Phương thức cập nhật được viết như sau:

```
/// <summary>
/// Phương thức cập nhật dữ liệu cho bảng Food
/// </summary>
/// <returns>Trả về dương nếu cập nhật thành công, ngược lại là số âm</returns>
public int UpdateFood()
{
    //Khai báo đối tượng Food và lấy đối tượng hiện hành
    Food food = foodCurrent;
    // Kiểm tra nếu các ô nhập khác rỗng
    if (txtName.Text == "" || txtUnit.Text == "" || txtPrice.Text == "")
        MessageBox.Show("Chưa nhập dữ liệu cho các ô, vui lòng nhập lại");
    else
    {
        //Nhận giá trị Name, Unit, và Notes khi người dùng sửa
        food.Name = txtName.Text;
        food.Unit = txtUnit.Text;
        food.Notes = txtNotes.Text;
        //Giá trị price là giá trị số nên cần bắt lỗi khi người dùng nhập sai
        int price = 0;
        try
        {
            // Chuyển giá trị từ kiểu văn bản qua kiểu int
            price = int.Parse(txtPrice.Text);
        }
        catch
```

```
            {
                // Nếu sai, gán giá = 0
                price = 0;
            }
            food.Price = price;
            // Giá trị FoodCategoryID được lấy từ ComboBox
           food.FoodCategoryID = int.Parse(cbbCategory.SelectedValue.ToString());
            // Khao báo đối tượng FoodBL từ tầng Business
            FoodBL foodBL = new FoodBL();
            // Cập nhật dữ liệu trong bảng
            return foodBL.Update(food);
        }
        return -1;
    }
```

Sự kiện khi người dùng click vào nút sửa được viết như sau:

```
    private void cmdUpdate_Click(object sender, EventArgs e)
    {
        // Gọi phương thức cập nhật dữ liệu
        int result = UpdateFood();
        if (result > 0) // Nếu cập nhật thành công
        {
            // Thông báo kết quả
            MessageBox.Show("Cập nhật dữ liệu thành công");
            // Tải lại dữ liệu cho ListView
            LoadFoodDataToListView();
        }
        // Nếu thêm không thành công thì thông báo cho người dùng
        else MessageBox.Show("Cập nhật dữ liệu không thành công. Vui lòng kiểm tra
lại dữ liệu nhập");
    }
```

III. Bài tập

1. Từ các bước minh họa ở trên, xây dựng chương trình mô hình ba tầng cho bảng Category và bảng Food.

2. Xây dựng hết các chức năng của chương trình cho phép nhập liệu tất cả các bảng (Xem cơ sở dữ liệu phần Phụ lục trong giáo trình).

3. Xây dựng phần phân quyền, cho phép gán các quyền như: Quản lý, Kế toán, Nhân viên, Admin.

4. Xây dựng chương trình quản lý nhà hàng hoàn chỉnh với các chức năng như: Đặt món, tách bàn, thanh toán theo mô hình 3 tầng như trên.

5. Kết hợp DevExpress, cải tiến một số chức năng của chương trình để chương trình thân thiện, dễ sử dụng.

--Hết--