TRƯỜNG ĐẠI HỌC ĐÀ LẠT
KHOA CÔNG NGHỆ THÔNG TIN
--oOo--

Nguyễn Thị Lương (Chủ biên)
Phan Thị Thanh Nga - Trần Thị Phương Linh
Hoàng Minh Tiến - Thái Duy Quý - Tạ Hoàng Thắng

HƯỚNG DẪN THỰC HÀNH
PHÁT TRIỂN ỨNG DỤNG DESKTOP

Đà Lạt, tháng 8 năm 2024

LỜI MỞ ĐẦU

Ngày nay, cuộc Cách mạng Công nghiệp 4.0 đã thay đổi sâu sắc nhận thức của con người về cách tiếp cận công nghệ, cũng như triển khai phương thức quản lý và sản xuất trong mọi lĩnh vực. Theo đó, các ứng dụng chuyển đổi số đóng vai trò chủ chốt trong việc thúc đẩy sự phát triển mạnh mẽ và ổn định của nền kinh tế. Các phương pháp quản lý truyền thống dần dần được thay thế bởi các ứng dụng và phần mềm quản lý hiện đại, do không đáp ứng được các yếu tố như hiệu quả về thời gian và chi phí. Có thể thấy, nhiều thành tựu đáng kể của công nghệ thông tin đã len lỏi khắp mọi nơi, từ các hệ thống quản trị nội dung, phần mềm quản lý, website với đa dạng chủ đề, đến các ứng dụng tiện ích trên các thiết bị thông minh, thậm chí cả các quy trình tự động hóa, góp phần nâng cao hiệu suất và chất lượng cuộc sống.

Trước sự bùng nổ công nghệ liên tục, các ứng dụng phần mềm cần phải thay đổi và cập nhật tính năng thường xuyên để đáp ứng nhu cầu của người dùng và thích ứng với sự tiến bộ không ngừng của công nghệ. Cụ thể, phần mềm không chỉ cần có tốc độ xử lý nhanh, tính linh hoạt và thông minh, mà còn phải có khả năng xử lý lượng dữ liệu lớn và truy xuất nội dung một cách nhanh chóng, tức thời, đạt hiệu quả kinh tế cao. Do đó, bên cạnh việc cần có hệ quản trị cơ sở dữ liệu phù hợp, chúng ta cũng cần công cụ và môi trường phát triển tốt để xây dựng các ứng dụng phần mềm có chất lượng. Đồng thời, quá trình tổ chức chương trình cũng giữ vai trò quan trọng trong việc xây dựng phần mềm, để nhìn rõ tổng quan những việc cần làm để mang lại hiệu quả, chẳng hạn như tận dụng một cách tinh tế các thành tựu khoa học công nghệ.

Nhằm tạo điều kiện để sinh viên cũng như người yêu thích lập trình ứng dụng desktop sử dụng các kiến thức đã học và nâng cao kỹ năng lập trình, chúng tôi biên soạn và giới thiệu "Sách hướng dẫn thực hành Phát triển ứng dụng Desktop" giúp độc giả như một tài liệu học tập, phát triển kỹ năng và tư duy lập trình. Cuốn sách cung cấp các hướng dẫn và bài tập thực hành từ cơ bản tới nâng cao xuyên suốt từ bước thiết kế giao diện chương trình cho tới xử lý các sự kiện của form và các điều khiển. Bên cạnh đó, sách còn hướng dẫn người học phát triển các ứng dụng đọc và ghi dữ liệu từ tập tin văn bản, XML, JSON cũng như kết nối tới cơ sở dữ liệu sử dụng câu truy vấn đơn giản hoặc gọi thủ tục để thực thi các thao tác truy vấn và lưu trữ dữ liệu. Ngoài ra, cuốn sách còn giới thiệu tới độc giả một cách tiếp cận khác trong việc sử dụng mô hình ba tầng để tổ chức mã chương trình nhằm nâng cao tính chuyên nghiệp và tái sử dụng mã nguồn. Thêm vào đó, chúng tôi cũng giới thiệu cho người đọc cách sử dụng một thư viện hỗ trợ kết nối cơ sở dữ liệu đang được sử dụng khá phổ biến, đó là Entity Framework. Thêm vào đó, chúng tôi hy vọng, đây sẽ là tài liệu tham khảo tương đối đầy đủ về xây dựng ứng dụng Desktop có kết nối cơ sở dữ liệu để quý bạn đọc tự khám phá và phát huy những kỹ năng tiềm ẩn của mình.

Sách hướng dẫn được chia làm 8 chủ đề với những nội dung cơ bản như sau:

Chủ đề 1 giúp người đọc củng cố kiến thức về cơ sở dữ liệu qua việc sử dụng SQL Server. Sinh viên sẽ học cách thiết kế bảng, xác định thuộc tính và kiểu dữ liệu, và thiết lập các mối quan hệ giữa các bảng để đảm bảo tính toàn vẹn dữ liệu. Họ cũng sẽ thực hiện các thao tác truy vấn, từ đơn giản đến phức tạp, viết thủ tục lưu trữ (Stored

Procedure) và hàm tùy chỉnh (Function) để tối ưu hóa hiệu suất và xử lý dữ liệu. Chủ đề này cung cấp nền tảng vững chắc về quản lý cơ sở dữ liệu, giúp sinh viên phát triển kỹ năng thiết kế, tối ưu hóa và bảo trì cơ sở dữ liệu.

Chủ đề 2 hướng dẫn cách sử dụng các Control (điều khiển) cơ bản để xây dựng ứng dụng. Các điều khiển được giới thiệu trong chủ đề này bao gồm Label, TextBox, ComboBox, MaskedTextBox, Button, RadioButton, CheckBox, ListBox, CheckListBox, DateTimePicker, PictureBox và ListView để phát triển ứng dụng. Bên cạnh đó, chủ đề còn ôn tập lại kỹ năng lập trình hướng đối tượng và cung cấp hướng dẫn về cách xử lý các sự kiện của form và các điều khiển, nhằm hoàn thiện chức năng cơ bản của ứng dụng. Sinh viên được ôn tập và quan trọng nhất là áp dụng các kiến thức đã học về sử dụng các Control, xử lý sự kiện cho Form và Control để xây dựng một ứng dụng quản lý thông tin sinh viên đơn giản.

Chủ đề 3 hướng dẫn sinh viên xây dựng một ứng dụng quản lý sinh viên đơn giản với tính năng đọc và ghi dữ liệu từ các loại tập tin khác nhau như “txt”, “xml” và “json”, sử dụng các thư viện tích hợp sẵn hoặc gói NuGet Package trong .NET. Bên cạnh việc học các thao tác với dữ liệu tập tin, sinh viên còn được củng cố kỹ năng xây dựng các Control và Form để hiển thị dữ liệu tải lên. Điều này giúp quản lý dữ liệu hiệu quả hơn và tạo ra giao diện người dùng trực quan.

Chủ đề 4 giúp sinh viên tìm hiểu cách xây dựng ứng dụng có kết nối đến một cơ sở dữ liệu và thực thi trực tiếp một số lệnh truy vấn đơn giản. Thông qua đó, sinh viên sẽ nắm rõ các thành phần của một chuỗi kết nối và ý nghĩa của chúng, cũng như cách tạo đối tượng kết nối, cách thực thi truy vấn dùng đối tượng Command, và cách đọc dữ liệu dùng DataReader. Ngoài ra, sinh viên có thể xử lý các lỗi thường gặp khi kết nối và truy vấn thông qua phần thực hành.

Chủ đề 5 hướng dẫn sinh viên nâng cao kỹ thuật kết nối và thao tác cơ sở dữ liệu, tập trung vào việc sử dụng đối tượng Parameter để truyền tham số vào các lệnh SQL. Sinh viên sẽ học cách thực thi các lệnh truy vấn có tham số, nhận kết quả trả về và xử lý dữ liệu từ các lệnh này. Chủ đề còn bao gồm việc thực thi các thủ tục lưu trữ và nhận dữ liệu trả về từ chúng, cung cấp cho sinh viên kiến thức và kỹ năng cần thiết để làm việc hiệu quả với các truy vấn và thủ tục có tham số trong cơ sở dữ liệu.

Chủ đề 6 giúp sinh viên làm quen với mô hình đa tầng trong phát triển ứng dụng. Chủ đề này 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. Chủ đề này không chỉ giúp sinh viên hiểu rõ cách phân chia trách nhiệm trong một ứng dụng đa tầng mà còn cung cấp kỹ năng thực hành trong việc phát triển ứng dụng có cấu trúc rõ ràng, dễ bảo trì và mở rộng.

Chủ đề 7 giúp sinh viên nắm vững cách sử dụng Entity Framework để kết nối với cơ sở dữ liệu và thực hiện các thao tác truy vấn cũng như lưu trữ dữ liệu. Sinh viên sẽ học cách áp dụng Entity Framework để tạo và quản lý các đối tượng dữ liệu, thực hiện các truy vấn linh hoạt, và lưu trữ thông tin một cách hiệu quả, từ đó nâng cao khả năng phát triển ứng dụng với cơ sở dữ liệu trong môi trường .NET.

Chủ đề 8 giúp sinh viên làm quen với mẫu thiết kế Repository trong lập trình cơ sở dữ liệu, thông qua ví dụ minh họa sử dụng ADO.NET và truy vấn LINQ. Chủ đề này

còn giúp sinh viên học cách thiết kế mẫu Repository nhằm tách biệt logic truy cập dữ liệu khỏi phần còn lại của ứng dụng, từ đó cải thiện tính tổ chức, khả năng bảo trì và mở rộng của mã nguồn. Ngoài ra, sinh viên được hướng dẫn cách tích hợp mẫu Repository vào dự án lập trình, từ đó sử dụng mẫu này để thực hiện truy xuất dữ liệu từ cơ sở dữ liệu và hiển thị dữ liệu trên giao diện WinForm.

Trong phần cuối cùng, chúng tôi trình bày một case-study tổng hợp về một ứng dụng hoàn chỉnh. Thông qua nghiên cứu trường hợp này, sinh viên sẽ nắm bắt được toàn bộ quy trình xây dựng dự án lập trình cơ sở dữ liệu, từ thiết kế và phát triển đến triển khai. Case-study này không chỉ giúp sinh viên hiểu rõ cách áp dụng các kỹ thuật thiết kế đã học vào thực tiễn mà còn trang bị cho họ khả năng phát triển các ứng dụng tương tự và triển khai vào các dự án thực tế một cách tự tin và hiệu quả.

Chúng tôi hi vọng cuốn sách này sẽ là một tài liệu tham khảo có ích đối với sinh viên ngành công nghệ thông tin và những độc giả quan tâm tới lĩnh vực xây dựng và phát triển ứng dụng có kết nối cơ sở dữ liệu. Chúng tôi rất mong nhận được sự cổ vũ và những ý kiến đóng góp quý báu của các quý độc giả giúp bổ sung và hoàn thiện hơn nữa nội dung của cuốn sách.

Sau cùng, chúng tôi xin gửi lời cảm ơn đến các thầy cô và đồng nghiệp đã ủng hộ và giúp đỡ chúng tôi hoàn thành giáo trình này.

Nhóm tác giả

MỤC LỤC

DANH MỤC TỪ VIẾT TẮT

ADO.NET (ActiveX Data Objects for .NET): Đối tượng Dữ liệu ActiveX cho .NET

BL (Business Logic): Tầng Business Logic trong mô hình 3 tầng

CSDL: Cơ sở dữ liệu

CSV (Comma-separated values): Các giá trị ngăn cách bằng dấu phẩy

DA (Data Access): Truy cập Dữ liệu

DTO (Data Transfer Object): Đối tượng Truyền Dữ liệu

IoC (Inversion of Control): Đảo ngược Kiểm soát

JSON (JavaScript Object Notation): Cú pháp Đối tượng JavaScript

LINQ (Language Integrated Query): Ngôn ngữ Truy vấn Tích hợp

NT (Network Technology): Công nghệ Mạng

ODBC (Open Database Connectivity): Kết nối Cơ sở Dữ liệu Mở

OLE DB (Object Linking and Embedding Database): Cơ sở dữ liệu Nhúng Liên kết Đối tượng

RDBMS (Relational DataBase Management System): Hệ thống Quản trị Cơ sở dữ liệu Quan hệ

SQL (Structured Query Language): Ngôn ngữ Truy vấn Cấu trúc

UI (User Interface): Giao diện người dùng

XML (eXtensible Markup Language): Ngôn ngữ Đánh dấu Mở rộng

CHỦ ĐỀ 1. ÔN TẬP CƠ SỞ DỮ LIỆU

A. Mục tiêu

Chủ đề này giúp sinh viên hệ thống lại cơ sở dữ liệu bằng cách sử dụng hệ quản trị cơ sở dữ liệu (SQL Server). Sinh viên sẽ thiết kế các bảng, thiết kế các mối quan hệ ràng buộc giữa các bảng, thực hiện các thao tác truy vấn, viết các thủ tục (*Store Procedure*) và các hàm (*Function*).

Sau chủ đề này, sinh viên cần nắm rõ những vấn đề sau:

- Tìm hiểu rõ môi trường làm việc của một Hệ quản trị cơ sở dữ liệu SQL Server.
- Các cách kết nối với SQL Server: Windows Authentication hoặc SQL Server Authentication.
- Tìm hiểu cách tạo một cơ sở dữ liệu, tạo bảng, tạo truy vấn, tạo ràng buộc, tạo thủ tục, tạo hàm và sao lưu, phục hồi cơ sở dữ liệu thông qua Backup và Restore hoặc tạo tập tin sao lưu thông qua Generate Script.
- Tìm hiểu cách xây dựng một lược đồ quan hệ trên Database Diagrams, thêm các bảng vào Database Diagrams, thiết lập mối quan hệ giữa các bảng trong Database Diagrams, cụ thể quan hệ: 1-1, quan hệ: 1-nhiều và quan hệ: nhiều-1.
- Tìm hiểu cách nhập dữ liệu vào các bảng khi đã thiết lập mối quan hệ.
- Thực hiện các truy vấn trên dữ liệu thử nghiệm với các hàm: SELECT, INSERT, DELETE, UPDATE.

B. Hướng dẫn thực hành

1. Giới thiệu SQL Server

SQL Server là một hệ thống quản trị cơ sở dữ liệu quan hệ (Relational DataBase Management System - RDBMS) do công ty Microsoft phát triển, dùng để thiết kế dữ liệu bằng cách sử dụng các bảng (*tables*), các mối quan hệ (*relationships*), các truy vấn (*query*), thủ tục (*Store procedure*), hàm (*function*) và bẫy sự kiện (*trigger*). Giống như những hệ quản trị khác (Oracle, MySQL, DB2, MongoDB,…), khi làm việc với SQL Server, người dùng phải thực hiện các câu lệnh truy vấn dữ liệu Transaction - SQL để trao đổi dữ liệu giữa Client Computer và Server Computer.

SQL Server có một số đặc tính sau:

- Có tốc độ xử lý dữ liệu nhanh đáp ứng yêu cầu về thời gian, cho phép quản trị một hệ cơ sở dữ liệu lớn (lên đến vài terabyte).

- Có hệ thống phân quyền bảo mật tương thích với hệ thống bảo mật của công nghệ NT (Network Technology), tích hợp với hệ thống bảo mật của Windows NT hoặc sử dụng hệ thống bảo vệ độc lập của SQL Server.

- Cho phép nhiều người cùng khai thác trong một thời điểm đối với một cơ sở dữ liệu và toàn bộ quản trị cơ sở dữ liệu (lên đến vài chục ngàn user).

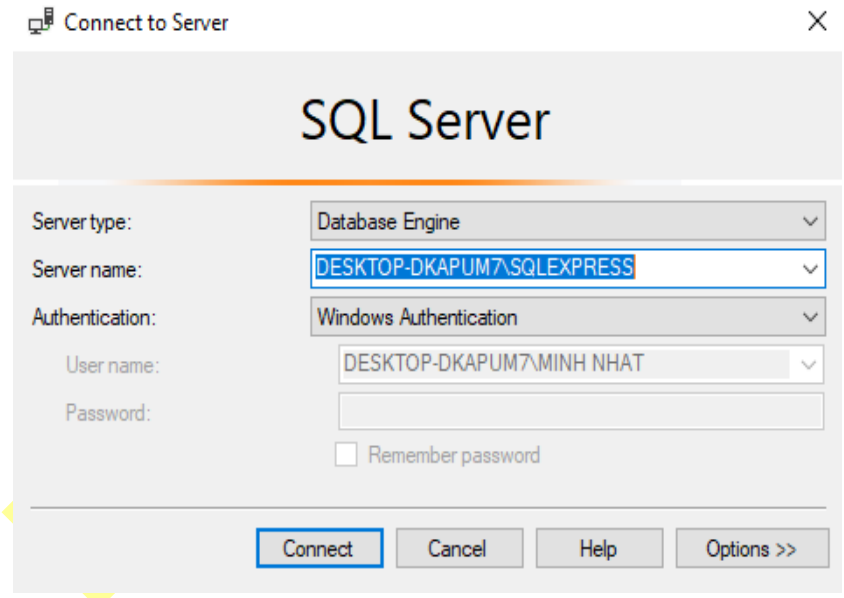
- Hỗ trợ trong việc triển khai cơ sở dữ liệu phân tán và phát triển ứng dụng trên Internet.

- Cho phép lập trình kết nối với nhiều ngôn ngữ lập trình khác với mục đích xây dựng các ứng dụng đặc thù (Visual Basic, C#, C, C++, ASP, ASP.NET, XML,...).

- Sử dụng câu lệnh truy vấn dữ liệu Transaction - SQL (trên SQL Server), PL/SQL trên Oracle.

2. Bắt đầu thao tác với SQL Server

SQL Server sau khi cài đặt thành công, lúc khởi động sẽ bật màn hình như sau:



Trong đó:

- *Server type*: Kiểu Server, thường để mặc định là *Database Engine*;

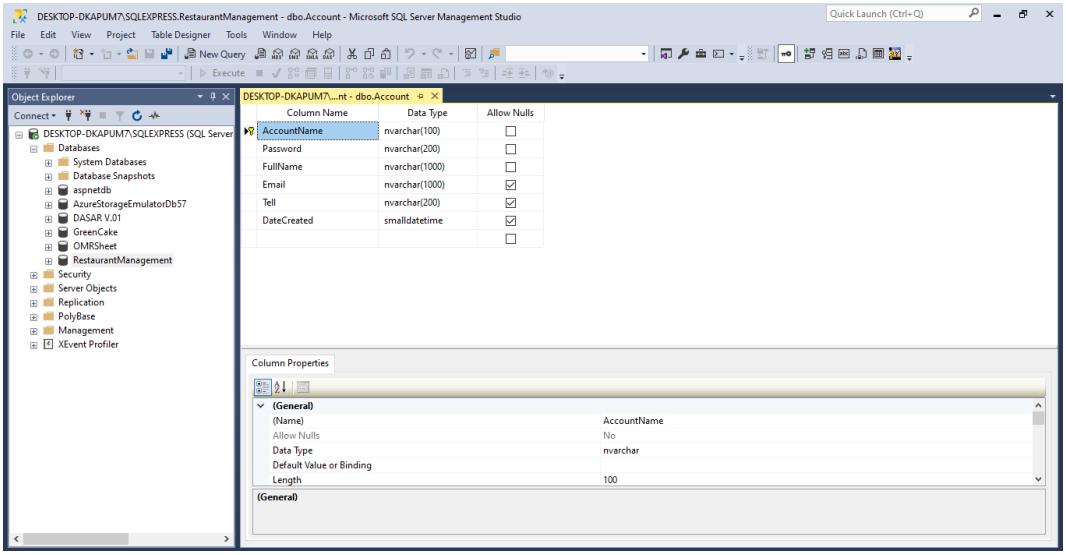
- *Server name*: Tên của Server, đây là tên mà người dùng đặt, giống như một máy tính Server chứa thông tin về cơ sở dữ liệu. Khi kết nối bằng ngôn ngữ lập trình cần lưu ý tới đối tượng này.

- *Authentication*: Thường chọn một trong hai chế độ: Nếu chọn *Windows Authentication* thì kết nối theo Window, nghĩa là khi đăng nhập vào Windows thì sẽ vào được cơ sở dữ liệu (thường thì người dùng sẽ chọn cách này); Nếu chọn *SQL*

Server Authentication thì phải nhập Username và mật khẩu để vào SQL Server (cách này áp dụng cho chế độ kết nối từ xa).

- Sau khi chọn các kiểu trên, nhấn Connect để kết nối.

  Giao diện sau khi kết nối như sau:



Một số thao tác thường dùng trên Hệ quản trị cơ sở dữ liệu SQL Server như sau:

- *Tạo Cơ sở dữ liệu*: Nhấp chuột phải lên Databases, chọn New Database, đặt tên và nhấn OK.

- *Tạo bảng*: Vào Database, nhấp chuột phải lên Table, chọn New, chọn Table: Đặt tên cột, kiểu dữ liệu, cho phép chọn Null hay không, nhấn lưu để đặt tên cho bảng.

- *Tạo truy vấn*: Nhấp chuột phải lên Database, chọn New Query (hoặc chọn New Query trên thanh công cụ).

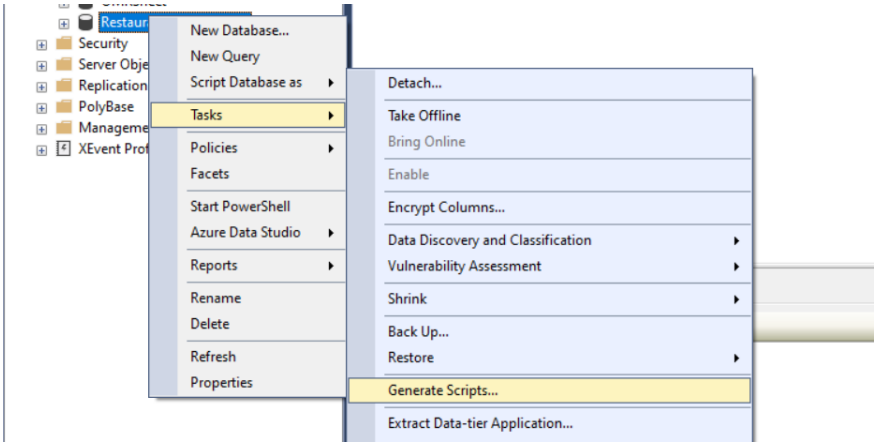
- *Tạo ràng buộc*: Nhấp chuột phải lên Database Diagrams, chọn New Database Diagram, thiết lập các mối quan hệ, sau đó lưu lại.

- *Tạo thủ tục (Store Procedure)*: Vào Programmability, nhấp chuột phải lên Store Procedures chọn Store Procedure... để viết các thủ tục.

- *Tạo hàm (Function)*: Vào Programmability, nhấp chuột phải lên Function chọn New, chọn Scalar-valued Functions để viết các hàm.

- *Sao lưu và phục hồi cơ sở dữ liệu*: Nhấp chuột phải lên Database, chọn Task, chọn Backup hoặc Restore.

Lưu ý: Có thể sao lưu bằng cách phát sinh script chứa dữ liệu và mô hình bằng cách nhấp chuột phải lên Database, chọn Task, chọn Generate Scripts:



3. Thực hành trên SQL Server

- **Bước 1**: Sinh viên mở SQL Server, kết nối bằng Windows Authentication, tạo một cơ sở dữ liệu có tên Restaurant Management.

- **Bước 2**: Tạo các bảng có tên bảng, tên cột, kiểu dữ liệu như hình sau. Lưu ý: ID là kiểu số nguyên (int), tự tăng khi có dữ liệu.

Category *

Column Name	Data Type	Allow Nulls
ID	int	☐
Name	nvarchar(1000)	☐
Type	int	☐
		☐

Table *

Column Name	Data Type	Allow Nulls
ID	int	☐
Name	nvarchar(1000)	☑
Status	int	☐
Capacity	int	☑
		☐

BillDetails *

Column Name	Data Type	Allow Nulls
ID	int	☐
InvoiceID	int	☐
FoodID	int	☐
Quantity	int	☐
		☐

Food *

Column Name	Data Type	Allow Nulls
ID	int	☐
Name	nvarchar(1000)	☐
Unit	nvarchar(100)	☐
FoodCategoryID	int	☐
Price	int	☐
Notes	nvarchar(3000)	☑
		☐

Bills *

Column Name	Data Type	Allow Nulls
ID	int	☐
Name	nvarchar(1000)	☐
TableID	int	☐
Amount	int	☐
Discount	float	☑
Tax	float	☑
Status	bit	☐
CheckoutDate	smalldatetime	☑
Account	nvarchar(100)	☐
		☐

RoleAccount *

Column Name	Data Type	Allow Nulls
RoleID	int	☐
AccountName	nvarchar(100)	☐
Actived	bit	☐
Notes	nvarchar(3000)	☑
		☐

Role *

Column Name	Data Type	Allow Nulls
ID	int	☐
RoleName	nvarchar(1000)	☐
Path	nvarchar(3000)	☑
Notes	nvarchar(3000)	☑
		☐

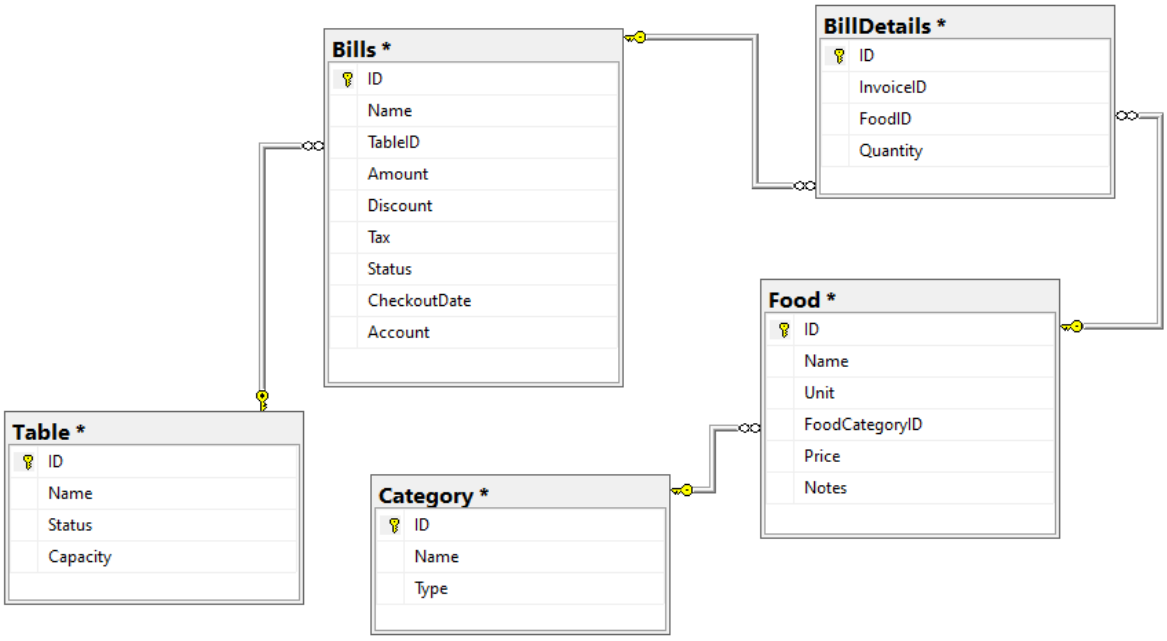
Account *

Column Name	Data Type	Allow Nulls
AccountName	nvarchar(100)	☐
Password	nvarchar(200)	☐
FullName	nvarchar(1000)	☐
Email	nvarchar(1000)	☑
Tell	nvarchar(200)	☑
DateCreated	smalldatetime	☑
		☐

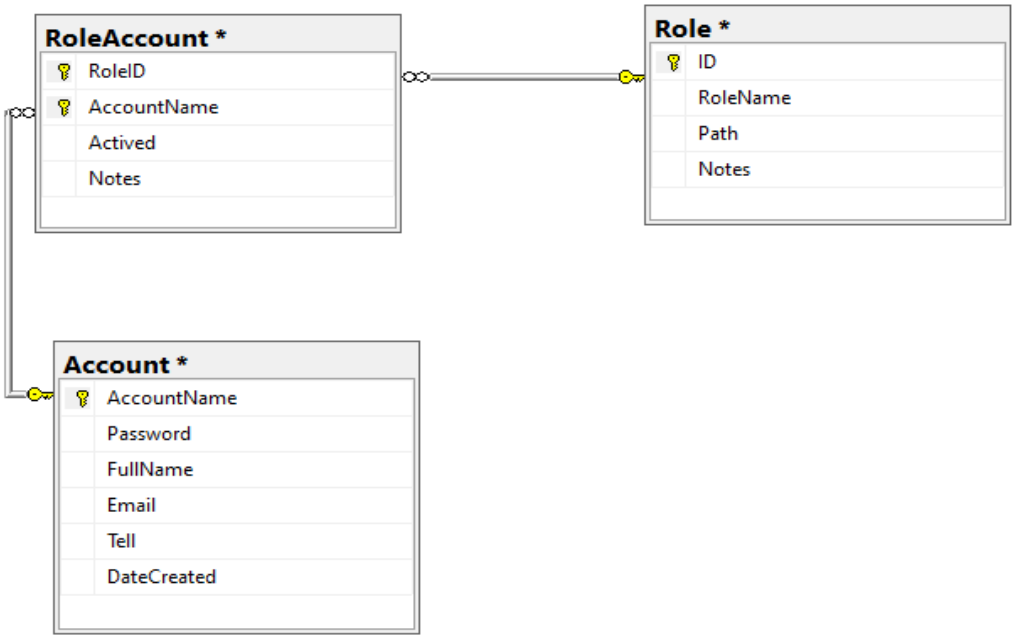
Lưu ý: Sinh viên cần tìm hiểu tác dụng của các bảng trên và ý nghĩa của mô hình trên là gì.

- **Bước 3**: Tạo lược đồ quan hệ, nhấp chuột phải lên Database Diagrams, chọn New Database Diagrams, sau đó tạo 2 lược đồ như sau:

Lược đồ 1: Lược đồ chức năng, đặt tên là *Functional Diagram*:



Lược đồ 2: Lược đồ phân quyền, đặt tên là *Role Diagram*:



- **Bước 4**: Nhập dữ liệu cho các bảng theo thứ tự từ bảng 1 đến bảng nhiều.

- **Bước 5**: Thực hiện truy vấn: Chọn New Query trên thanh công cụ, thực hiện các truy vấn như sau (xem hình ở dưới):

  - Lấy hết thông tin bằng lệnh *Select * from [Table]*
  - Lấy thông tin có điều kiện bằng lệnh: *Select * from [Table] where [column] = value*
  - Thêm dữ liệu vào bảng bằng lệnh: *Insert into [Table] values (...)*
  - Cập nhật dữ liệu bằng lệnh: *Update [Table] Set ...*
  - Xoá dữ liệu bằng lệnh: *Delete from [Table]*

- **Bước 6**: Tiến hành viết các thủ tục: Mỗi bảng sẽ viết 05 thủ tục theo cấu trúc như sau: [*Table*]_[*TenThuTuc*]. Các thủ tục bao gồm:

  - *[Table]_GetAll*: Lấy hết dữ liệu của bảng
  - *[Table]_GetByID*: Lấy thông tin dữ liệu bảng theo ID (khóa ngoại)
  - *[Table]_Insert*: Chèn dữ liệu vào bảng
  - *[Table]_Update*: Cập nhật dữ liệu bảng
  - *[Table]_Delete*: Xóa dữ liệu bảng theo khóa ngoại

  Sau khi viết xong thủ tục nào thì nhấn F5 để SQL Server ghi vào hệ thống.

  Ví dụ: Với bảng Category, các thủ tục được viết như các hình sau:

  - Category_GetAll:

```
-- Thủ tục lấy hết dữ liệu trong bảng Category
CREATE PROCEDURE Category_GetAll
AS
BEGIN
    SELECT * FROM dbo.Category
END
GO
```

- Category_GetByID:

```
-- Thủ tục lấy hết dữ liệu trong bảng Category theo ID
CREATE PROCEDURE Category_GetAll
(
  @ID INT
)
AS
BEGIN
    SELECT * FROM dbo.Category WHERE ID = @ID
END
GO
```

- Category_Insert:

```
-- Thủ tục thêm dữ liệu vào bảng Category
CREATE PROCEDURE Category_Insert
(
  @Name NVARCHAR(1000),
  @Type INT
)
AS
BEGIN
    -- Kiểm tra tồn tại Name: Lệnh này có thể không cần thiết trong một số bảng
    IF (NOT EXISTS (SELECT Name FROM dbo.Category WHERE Name = @Name))
        INSERT INTO dbo.Category (Name,Type)
                          VALUES (@Name,@Type)
END
GO
```

- Category_Update:

```
-- Thủ tục cập nhật dữ liệu trong bảng Category
CREATE PROCEDURE Category_Update
(
  @ID INT,
  @Name NVARCHAR(1000),
  @Type INT
)
AS
BEGIN
        UPDATE dbo.Category
        SET [Name] =@Name , [Type] =@Type
        WHERE ID = @ID
END
GO
```

- Category_Delete:

```
-- Thủ tục xóa mẫu tin trong bảng Category
CREATE PROCEDURE Category_Delete
(
  @ID INT
)
AS
BEGIN
        DELETE FROM dbo.Category
        WHERE ID = @ID
END
GO
```

Dùng lệnh EXEC để gọi thủ tục trên như sau:

```
EXEC dbo.Category_Insert N'Tráng miệng', 1
```

Hoặc:

```
EXEC dbo.Category_GetAll
```

Lưu ý: Thủ tục Insert nếu cần trả về ID nào vừa mới thêm, có thể viết lại như đoạn mã sau:

```
ALTER PROCEDURE dbo.Category_Insert_
(
  @ID INT OUTPUT,
  @Name NVARCHAR(1000),
  @Type INT
)
AS
BEGIN
    -- Kiểm tra tồn tại Name
    IF (NOT EXISTS (SELECT Name FROM dbo.Category WHERE Name = @Name))
        INSERT INTO dbo.Category (Name,Type)
                          VALUES (@Name,@Type)
     SET @ID = @@IDENTITY
END
```

Khi đó, ta gọi thủ tục để kiểm tra như sau:

```
DECLARE @ID INT = 0;
EXEC dbo.Category_Insert_ @ID = @ID OUTPUT,
                          @Name = N'Món rau',
                          @Type = 1
SELECT * FROM dbo.Category WHERE ID = @ID
```

176 %

Results | Messages

	ID	Name	Type
1	3	Món rau	1

Tương tự như vậy, thủ tục GetAll có thể viết chung cho tất cả các bảng (truyền vào tên bảng) như sau:

```
-- Thủ tục lấy tất cả mẫu tin theo tên bảng
CREATE PROCEDURE [dbo] [_GetAll]
(
  @TableName NVARCHAR(200)
)
AS
BEGIN
        -- Khai báo chuỗi và gán chuỗi
        DECLARE @Sql NVARCHAR(1000)
        SET @Sql = 'Select * from '+@TableName
        EXEC (@Sql) -- Thực thi
END
```

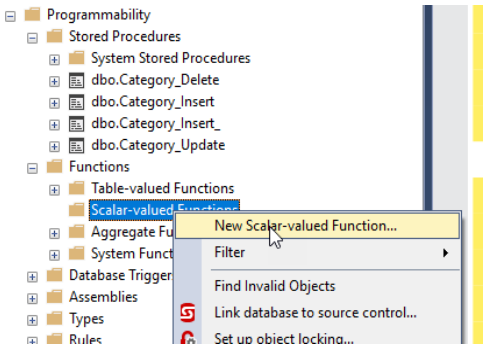
Khi đó, gọi thủ tục như sau:

```
EXEC dbo._GetAll 'Category'
```

hoặc `EXEC dbo._GetAll 'Food'`

- **Bước 7**: Viết hàm

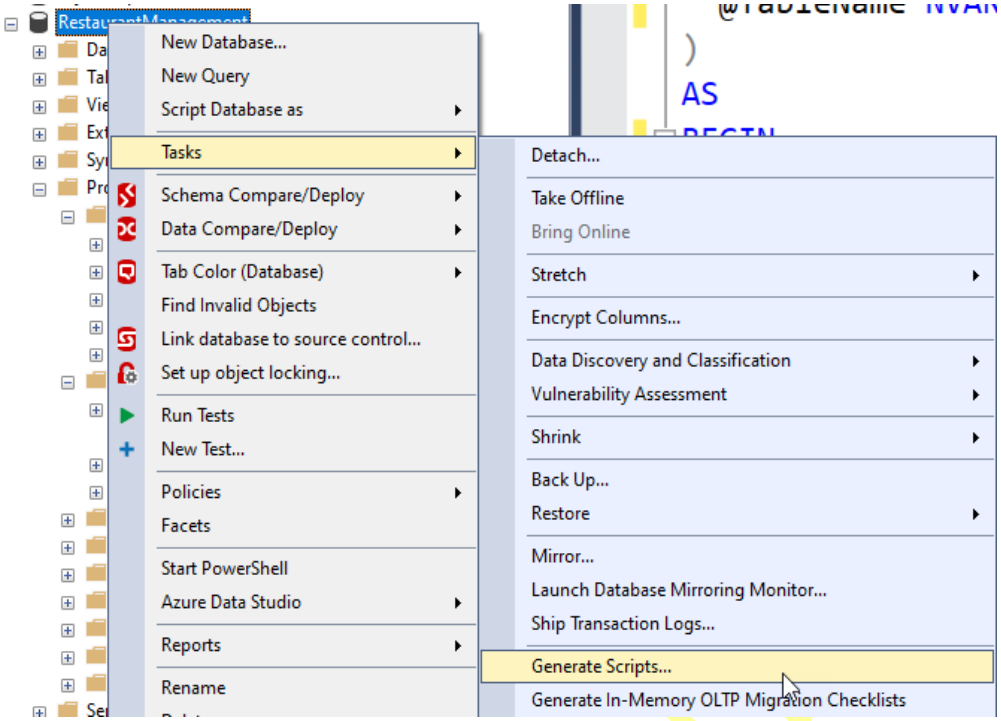
Có nhiều trường hợp cần tính toán, SQL Server cho phép viết hàm để tính, hàm sẽ trả về kiểu dữ liệu (số, ngày, tháng, chuỗi, …) hoặc trả về bảng. Hàm được tạo ra bằng cách vào Programmability, chọn Functions, nhấp chuột phải lên Scalar-valued Function, chọn New Scalar-valued Function và bắt đầu viết hàm.



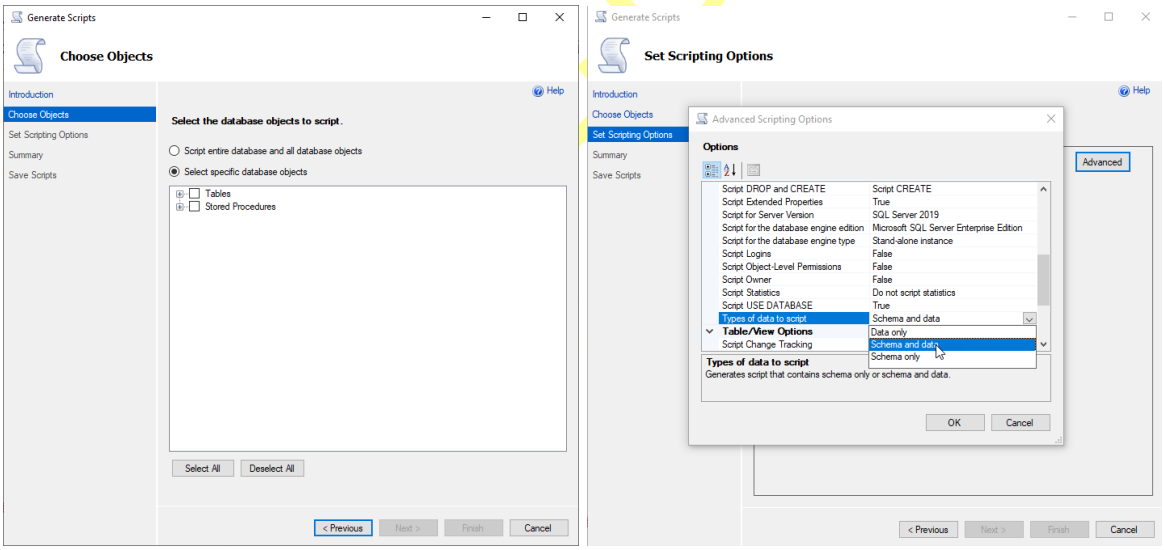
- **Bước 8**: Phát sinh Script

Sau khi đã tạo Database, tạo bảng, viết thủ tục, viết hàm, sinh viên có thể cho phát sinh Script để sử dụng cho lần sau. Cách phát sinh Script như sau:

  - Nhấp chuột phải lên Database, chọn Task, chọn Generate Script:



- Chọn ***Next***, chọn ***Select specific database objects***, sau đó đánh dấu check hết Table, View, Store Procedure, Functions… và nhấn Next. Chọn ***Advanced***, tìm đến ***Types of Data to Script***, chọn ***Scheme and Data***



- Đặt tên cho Script và nhấn Finish. Sau đó giữ lại Script này.

C. Bài tập

1. Viết hết các thủ tục Insert, Update, Delete cho tất cả các bảng nêu trên.
2. Viết một thủ tục _GetAll để lấy dữ liệu của tất cả các bảng, truyền vào tên bảng.
3. Viết một thủ tục _GetByID để lấy dữ liệu của tất cả các bảng có ID là kiểu int, khóa chính và tự tăng. Tham số truyền vào ID và tên bảng.

4. Viết thủ tục _Delete để xóa dữ liệu của bất kỳ bảng nào có ID là kiểu int, khóa chính và tự tăng. Tham số truyền vào là ID và tên bảng.
5. Viết thủ tục để khi thêm quyền vào bảng Role thì tự động gán hết quyền cho các User (Insert vào bảng User Role, nhưng để Active = false).
6. Viết hàm tính số tiền bán được theo ngày.
7. Viết hàm đếm số lượng món ăn bán được theo ngày.
8. Viết thủ tục thống kê số tiền bán được theo từng loại món ăn, theo ngày.
9. Viết thủ tục nhập hai bảng làm một.
10. Viết thủ tục tách bàn.

CHỦ ĐỀ 2. LẬP TRÌNH GIAO DIỆN C#

A. Mục tiêu

Một trong những công cụ phổ biến hỗ trợ lập trình viên tạo ra các giao diện người dùng một cách hiệu quả đó là Windows Form của Microsoft trong môi trường .NET. Windows Form cho phép các lập trình viên tạo các Form một cách dễ dàng, hỗ trợ nhiều điều khiển (controls) và các thành phần (components) giúp cho lập trình viên có thể xây dựng và tùy chỉnh giao diện một cách nhanh chóng và đa dạng, phù hợp nhu cầu và mục đích của người sử dụng. Trong chuyên đề này, hướng dẫn cơ bản trong lập trình giao diện: cách tạo Form với ngôn ngữ C#; một số trình điều khiển cơ bản và nâng cao; Các hộp thoại; Xử lý các sự kiện chuột và bàn phím trong Form… bao gồm:

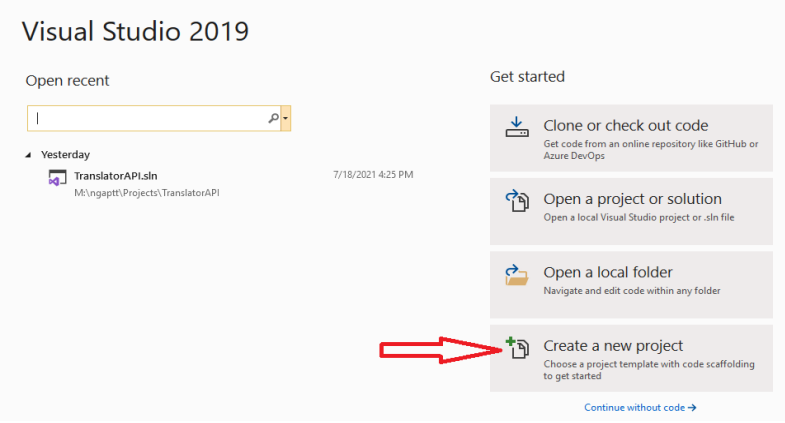
- Tạo project Windows Application
- Các điều khiển cơ bản;
- Các điều khiển nâng cao;
- Xử lý sự kiện chuột, bàn phím.

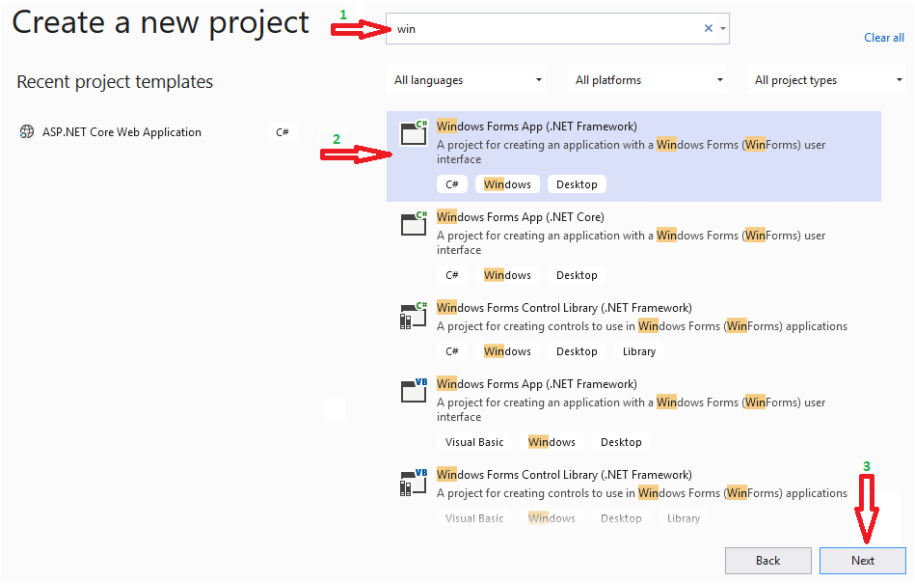
Sau khi hoàn thành chủ đề, sinh viên có thể áp dụng các kiến thức về sử dụng các control, xử lý sự kiện cho form và control để xây dựng một ứng dụng đơn giản cho phép đọc dữ liệu từ tập tin có đuôi "txt" hiển thị lên form và lưu dữ liệu vào tập tin dạng "txt".

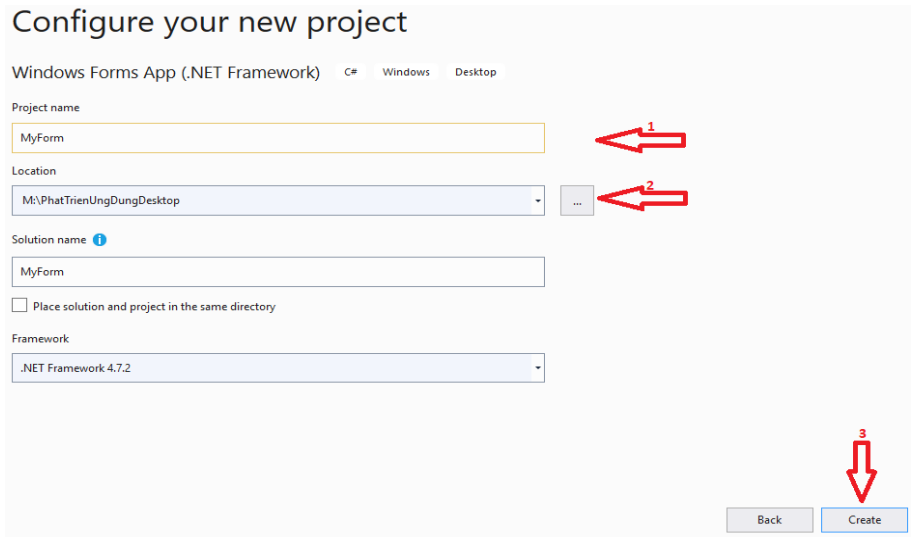
B. Hướng dẫn thực hành

Bước 1: Khởi động Microsoft Visual Studio bằng cách nhấp chuột vào biểu tượng trên desktop hoặc thanh Taskbar.

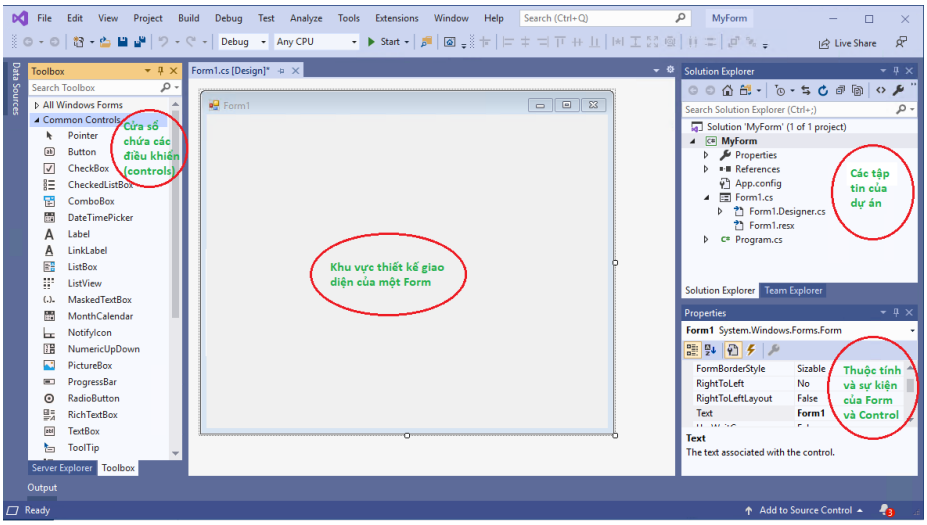
Bước 2: Tạo mới Project và chọn Windows Forms App (.NET Framework), đặt tên Project là MyForm.



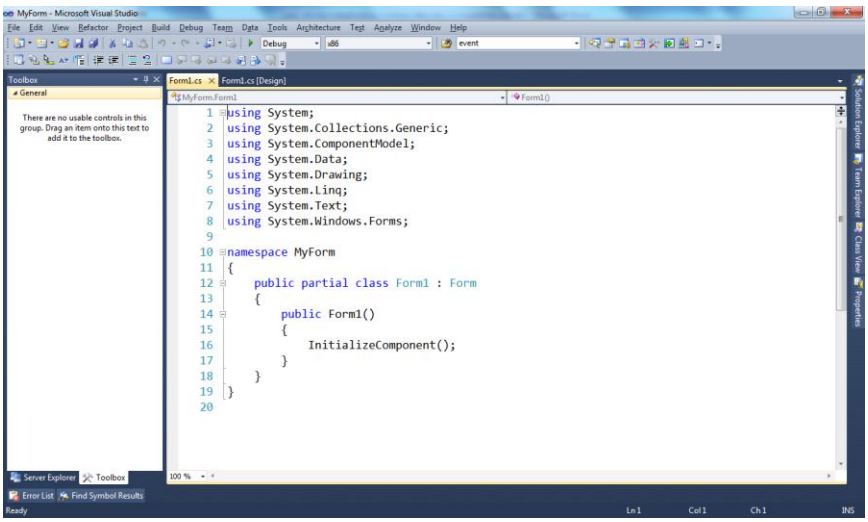




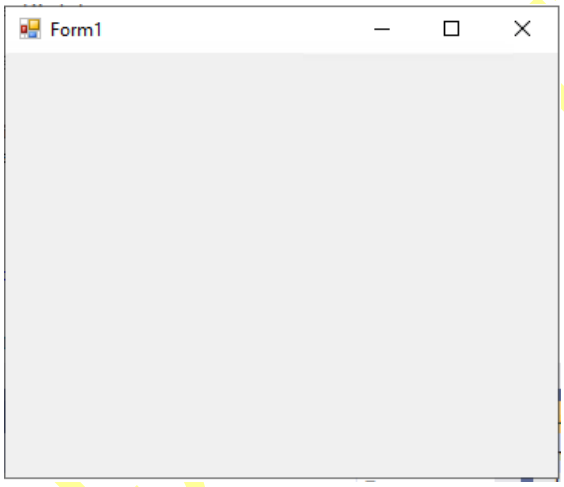
Bước 3: Giao diện chính của chương trình.



Bước 4: Để viết mã nguồn, nhấp chuột phải trên Form chọn View Code hoặc nhấn F7 hoặc vào menu View chọn Code.

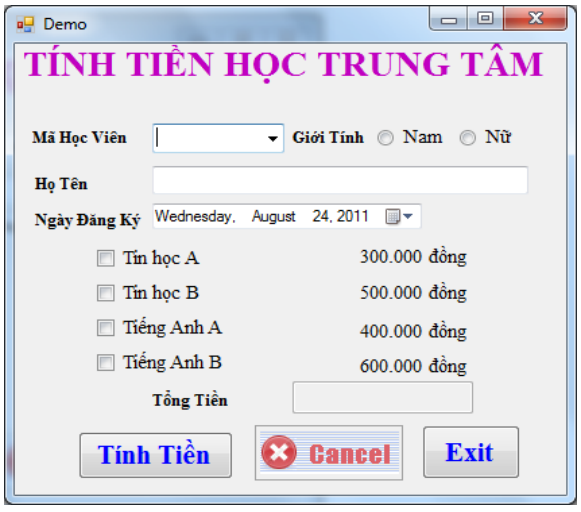


Bước 5: Nhấn Ctrl+F5 để chạy chương trình. Kết quả chạy chương trình.



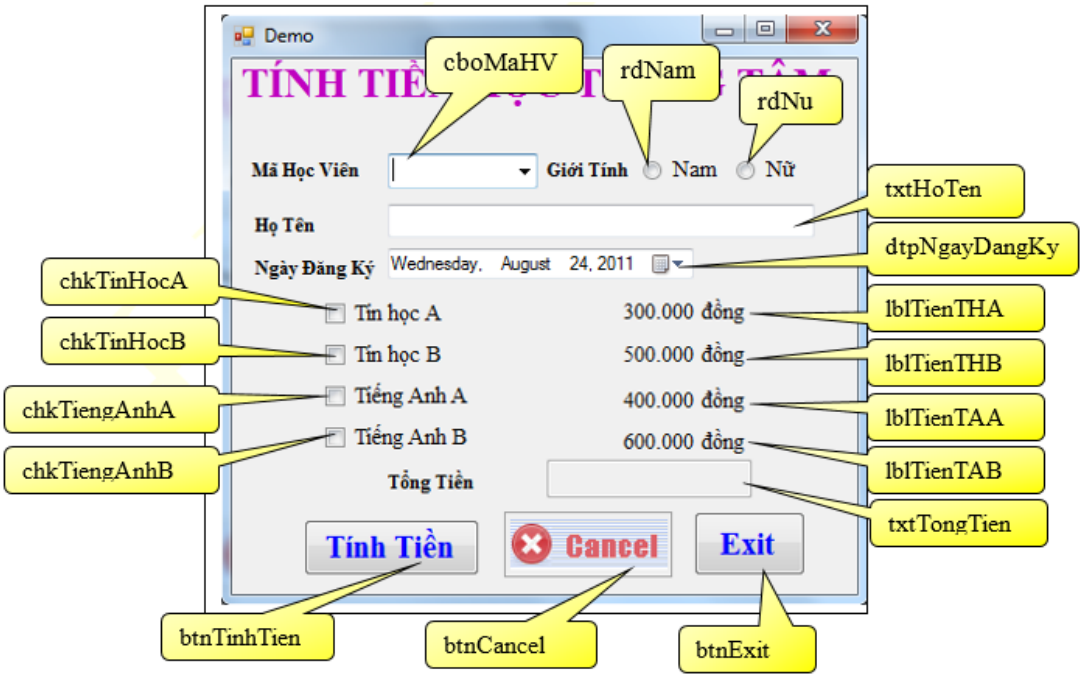
<u>Yêu cầu 1</u>: Tính tiền học trung tâm

Tạo ứng dụng đơn giản cho phép tính toán tiền học cho học viên (Mã số, Họ tên, Giới tính) khi đăng ký (kèm thông tin ngày đăng ký) các chương trình: Tin học A (300.000 đồng); Tin học B (500.000 đồng); Tiếng Anh A (400.000 đồng); Tiếng Anh B (600.000 đồng).

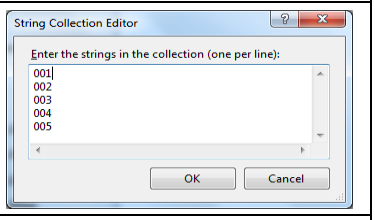


Bước 1: Thiết kế Form và đặt tên các control như hướng dẫn:

Ý nghĩa đặt tên control: LoaiControlThongTin. Ví dụ tên control: cboMaHV. Loại control sử dụng là ComboBox thông tin của Mã Học Viên.



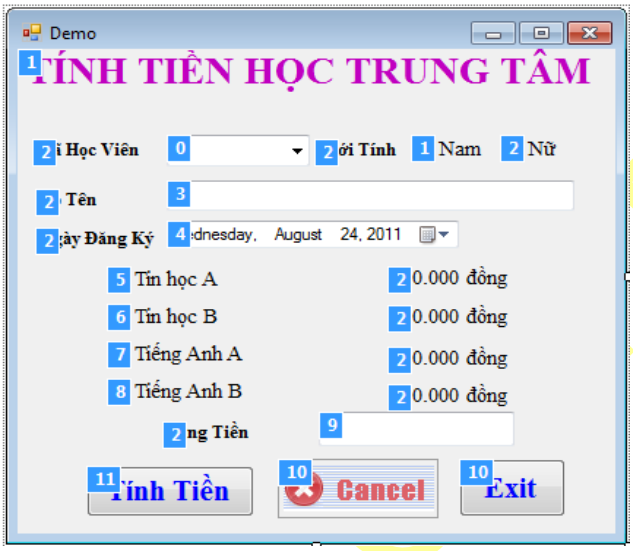
Bảng mô tả thông tin của các control

STT	Name	Loại Control	Tên Thuộc Tính	Giá Trị
1	frmTrungTam	Form	Text	Simple Drawing
2	cboMaHV	ComboBox	Items	
3	rdNam	RadioButton		
4	rdNu	RadioButton		
5	txtHoTen	TextBox		
6	chkTinHocA	CheckBox	Text	Tin học A
7	chkTinHocB	CheckBox	Text	Tin học B
8	chkTiengAnhA	CheckBox	Text	Tiếng Anh A
9	chkTiengAnhB	CheckBox	Text	Tiếng Anh B
10	lblTienTHA	Label	Text	300.000 đồng
11	lblTienTHB	Label	Text	500.000 đồng
12	lblTienTAA	Label	Text	400.000 đồng
13	lblTienTAB	Label	Text	600.000 đồng

14	txtTongTien	TextBox	Enable	False
15	btnTinhTien	Button	Text	Tính Tiền
16	btnCancel	Button	Image	***Đường dẫn hình***
17	btnExit	Button	Text	Exit

Bước 2: Thực hiện gán Tab order cho các control trên Form:

Vào View \ Tab Order \ nhấp chuột lên số để thiết lập Tab theo thứ tự sau:

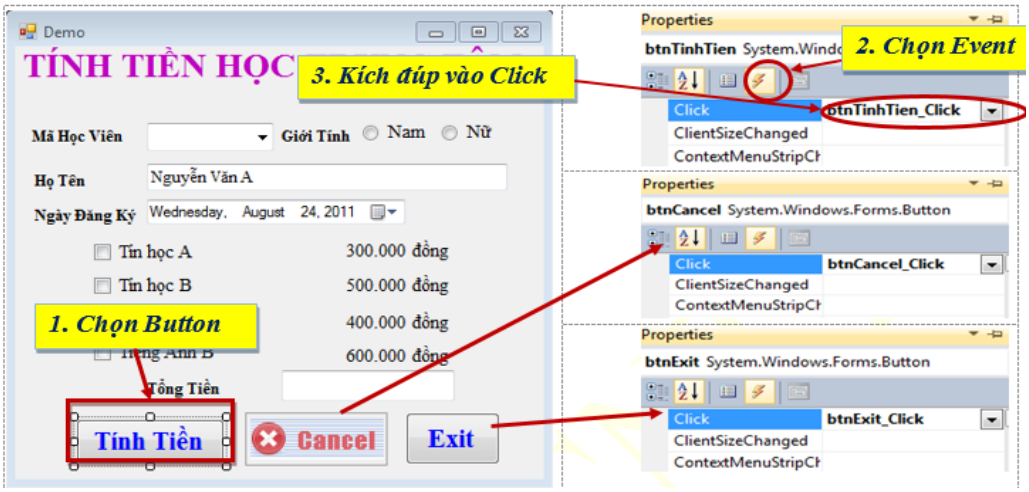


Để quay lại màn hình thiết kế Form thực hiện vào View\Tab Order.

Bước 3: Viết code chương trình.

Bước 3.1: Viết mã nguồn cho sự kiện Click của 3 nút: btnTinhTien, btnCancel, btnExit. Để chọn sự kiện cho nút có 2 cách:

Cách 1: Chọn nút cần tạo trình xử lý, sau đó nhấp chuột lên Tab event trong cửa sổ Properties, nhấp đôi chuột vào mục Click trong cửa sổ Event.



Cách 2: Nhấp đôi chuột vào nút cần tạo trình xử lý sự kiện trong màn hình Form design view. Khi đó Visual Studio sẽ tạo trình xử lý sự kiện gắn với sự kiện Click của nút "Tính Tiền" hoặc "Cancel", "Exit".

Bước 3.2: Viết code lớp frmTrungTam.cs:

- Sự kiện btnExit_Click để tắt chương trình.

```
private void btnExit_Click(object sender, EventArgs e)
{
    Application.Exit();
}
```

- Sự kiện btnCancel_Click thiết lập lại các control về giá trị mặc định.

```
private void Reset()
{
    this.cboMaHV.Text = "";
    this.txtHoTen.Text = "";
    this.dtpNgayDangKy.Value = DateTime.Now;
    this.rdNam.Checked = true;
    this.chkTiengAnhA.Checked = false;
    this.chkTiengAnhB.Checked = false;
    this.chkTinHocA.Checked = false;
    this.chkTinHocB.Checked = false;
    this.txtTongTien.Text = "";
}
1 reference
private void btnCancel_Click(object sender, EventArgs e)
{
    this.Reset();
}
```

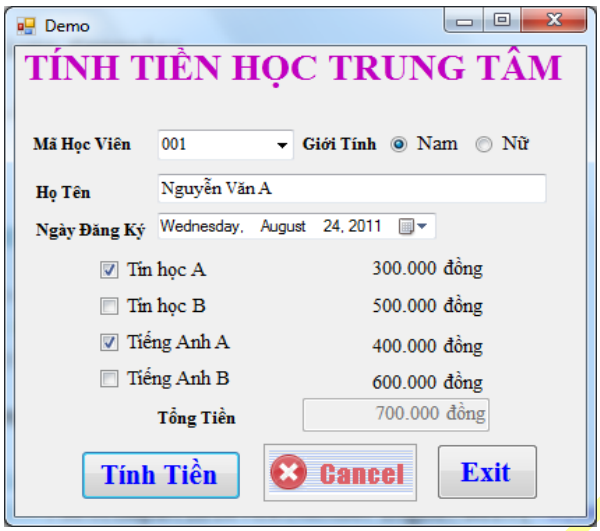
- Sự kiện btnTinhTien_Click thực hiện tính tiền dựa vào lựa chọn các khóa học và hiện thị tổng tiền.

```
private void btnTinhTien_Click(object sender, EventArgs e)
{
    int s = 0;
    if (chkTinHocA.Checked)
        s += int.Parse(lblTienTHA.Text.Split('.')[0]);
    if (chkTinHocB.Checked)
        s += int.Parse(lblTienTHB.Text.Split('.')[0]);
    if (chkTiengAnhA.Checked)
        s += int.Parse(lblTienTAA.Text.Split('.')[0]);
    if (chkTiengAnhB.Checked)
        s += int.Parse(lblTienTAB.Text.Split('.')[0]);
    this.txtTongTien.Text = s + ".000 đồng";
}
```

Bước 3.3: Phương thức ***main*** của lớp ***Program.cs*** bổ sung lệnh chạy frmTrungTam.

```
internal static class Program
{
    /// <summary>
    /// The main entry point for the application.
    /// </summary>
    [STAThread]
    0 references
    static void Main()
    {
        Application.EnableVisualStyles();
        Application.SetCompatibleTextRenderingDefault(false);
        Application.Run(new frmTrungTam());
    }
}
```

Bước 4: Nhấn Ctrl+F5 để chạy chương trình.



Yêu cầu 2: Nhập thông tin giảng viên khoa CNTT

Ứng dụng cho phép người dùng nhập một số thông tin giảng viên bao gồm: mã số, họ tên, giới tính, số điện thoại, ngày sinh, địa chỉ email, ngoại ngữ có thể sử dụng (tiếng Anh, tiếng Pháp, tiếng Nhật và tiếng Nga), Danh sách các học phần của khoa và Danh sách học phần giảng viên giảng dạy. Chức năng chính: cho phép lựa chọn từ danh sách học phần giảng viên giảng dạy từ danh sách học phần của Khoa; thông báo thông tin giảng viên đã nhập; hiển thị hộp thoại chi tiết thông tin của giảng viên; nút Cancel: reset lại toàn bộ control về trạng thái mặc định; nút Exit: tắt chương trình; và nút Liên hệ: mở website của khoa CNTT (https://cntt.dlu.edu.vn/).

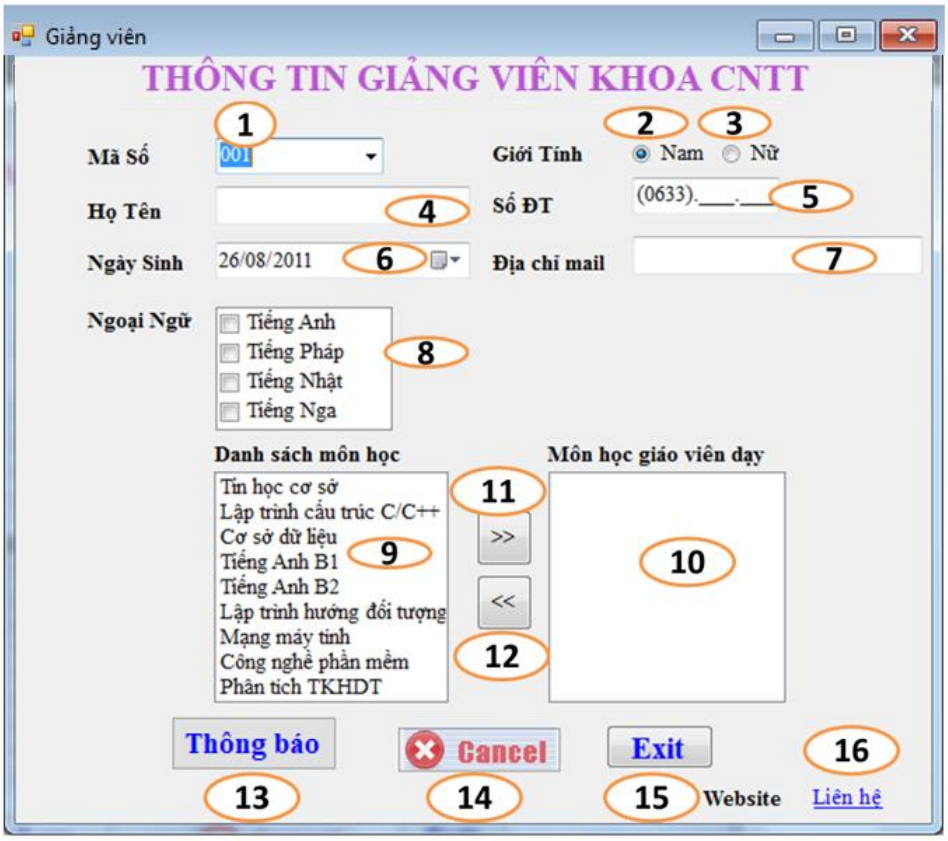
Bước 1:

- Thiết kế form thông báo thông tin giảng viên với tên **frmTBGiangVien**

Bảng mô tả thông tin control cho ***frmTBGiaoVien***

TT	Name	Loại Control
1	lblThongBao	Label

- Thiết kế giao diện form thông tin giảng viên với tên - **frmGiangVien**

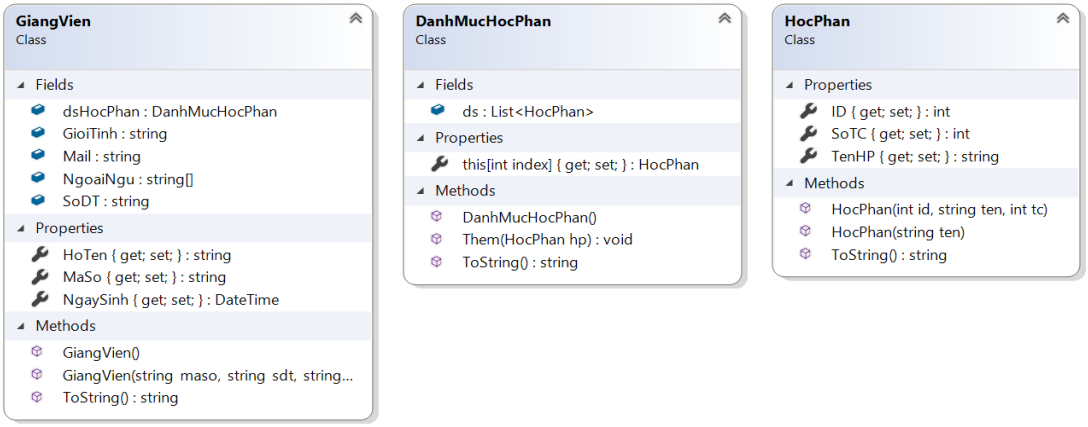


- Bảng mô tả thông tin control cho **frmGiangVien**

STT	Name	Loại Control	Tên thuộc tính	Giá Trị
1	cboMaSo	ComboBox	Items	***001*** ***002*** ***003*** ***004***
2	rdNam	CheckBox	Checked	***true***
3	rdNu	CheckBox		
4	txtHoTen	TextBox		
5	mtxtSoDT	MaskedTextBox	Mask	***(\0633).000.000***
6	dtpNgaySinh	DateTimePicker	CustomFormat Format	***dd/MM/yyyy*** ***Custom***
7	txtMail	TextBox		
8	chklbNgoaiNgu	CheckListBox	Items	***Tiếng Anh*** ***Tiếng Pháp*** ***Tiếng Nhật*** ***Tiếng Nga***
9	lbDanhSachHP	ListBox	Items SelectionMode	***"Danh sách học phần như trên Form"*** ***MultiExtended***

10	lbHocPhanDay	ListBox	ListBox	***One***
11	btnChon	Button	Text	>>
12	btnXoa	Button	Text	<<
13	btnThongBao	Button	Text	***Thông Báo***
14	btnCancel	Button	Image	
15	btnExit	Button	Text	***Exit***
16	linklbLienHe	LinkLabel	Text	***Liên hệ***

Bước 2: Xây dựng sơ đồ lớp



- Lớp **HocPhan.cs**: gồm thông tin ID, số tín chỉ và tên học phần.

```
10 references
public class HocPhan
{
    1 reference
    public int ID { get; set; }
    3 references
    public string TenHP { get; set; }
    1 reference
    public int SoTC { get; set; }
    1 reference
    public HocPhan(string ten)
    {
        this.TenHP = ten;
    }
    0 references
    public HocPhan(int id, string ten, int tc)
    {
        this.ID = id;
        this.TenHP = ten;
        this.SoTC = tc;
    }
    3 references
    public override string ToString()
    {
        return TenHP;
    }
}
```

- Lớp **DanhSachHocPhan.cs**: danh sách học phần – Đóng gói thuộc tính Index; phương thức thêm 1 học phần vào danh sách; ToString().

```
6 references
public class DanhMucHocPhan
{
    public List<HocPhan> ds;
    2 references
    public DanhMucHocPhan()
    {
        ds = new List<HocPhan>();
    }
    0 references
    public HocPhan this[int index]
    {
        get { return ds[index] as HocPhan; }
        set { ds[index] = value; }
    }
    1 reference
    public void Them(HocPhan hp)
    {
        ds.Add(hp);
    }
    3 references
    public override string ToString()
    {
        string s = "Danh sach mon hoc: ";
        foreach (object mh in ds)
        {
            s += mh as HocPhan + ";";
        }
        return s;
    }
}
```

- Lớp **GiangVien.cs**: thông tin giảng viên.

```
30 references
public class GiangVien
{
    9 references
    public string MaSo { get; set; }
    7 references
    public string HoTen { get; set; }
    4 references
    public DateTime NgaySinh { get; set; }
    public DanhMucHocPhan dsHocPhan;
    public string GioiTinh;
    public string[] NgoaiNgu;
    public string SoDT;
    public string Mail;
    5 references
    public GiangVien()
    {
        dsHocPhan = new DanhMucHocPhan();
        NgoaiNgu = new string[20];
    }
```

```
public GiangVien(string maso, string sdt, string mail, string hoten,
    DateTime ngaysinh, DanhMucHocPhan ds, string gt, string[] nn)
{
    this.MaSo = maso;
    this.HoTen = hoten;
    this.NgaySinh = ngaysinh;
    this.dsHocPhan=ds;
    this.NgoaiNgu = nn;
    this.SoDT = sdt;
    this.Mail = mail;
    this.GioiTinh = gt;
}
3 references
public override string ToString()
{
    string s = "Mã số: " + MaSo + "\n"
        + "Họ tên: " + HoTen + "\n"
        + "Ngày sinh: " + NgaySinh + "\n"
        + "Giới tính: " + GioiTinh + "\n"
        + "Số ĐT: " + SoDT + "\n"
        + "Mail: " + Mail + "\n";
    string sngoaingu = "Ngoại ngữ";
    foreach (string t in NgoaiNgu)
        sngoaingu += t + ";";
    string monDay = "Danh sách môn dạy: ";
    foreach (HocPhan hp in dsHocPhan.ds)
        monDay += hp + ";";
    s += "\n" + sngoaingu + "\n" + monDay;
    return s;
}
```

- Lớp **frmTBGiangVien.cs**, bổ sung phương thức *SetText* sau.

```
4 references
public partial class frmTBGiangVien : Form
{
    1 reference
    public frmTBGiangVien()
    {
        InitializeComponent();
    }
    //Gán chuỗi s cho thuộc text của lblThongBao
    1 reference
    public void SetText(string s)
    {
        this.lblThongBao.Text = s;
    }
}
```

- Lớp frmGiangVien.cs:
  - Sự kiện frmGiangVien_Load

```
private void frmGiangVien_Load(object sender, EventArgs e)
{
    string lienHe = "https://cntt.dlu.edu.vn/";
    this.linklbLienHe.Links.Add(0, lienHe.Length, lienHe);
    this.cboMaSo.SelectedItem = this.cboMaSo.Items[0];
}
```

  - Sự kiện *btnChon_Click* xóa học phần được chọn từ danh sách học phần và chuyển sang danh sách học phần dạy

```
private void btnChon_Click(object sender, EventArgs e)
{
    int i = this.lbDanhSachHP.SelectedItems.Count - 1;
    while(i>=0)
    {
        this.lbHocPhanDay.Items.Add(lbDanhSachHP.SelectedItems[i]);
        this.lbDanhSachHP.Items.Remove(lbDanhSachHP.SelectedItems[i]);
        i--;
    }
}
```

- Sự kiện *btnXoa_Click* xóa học phần được chọn từ danh sách học phần dạy và chuyển sang danh sách học phần.

```
private void btnXoa_Click(object sender, EventArgs e)
{
    int i = this.lbHocPhanDay.SelectedItems.Count - 1;
    while (i >= 0)
    {
        this.lbDanhSachHP.Items.Add(lbHocPhanDay.SelectedItems[i]);
        this.lbHocPhanDay.Items.Remove(lbHocPhanDay.SelectedItems[i]);
        i--;
    }
}
```

- Sự kiện *btnCancel_Click* thiết lập controls về giá trị mặc định.

```
private void btnCancel_Click(object sender, EventArgs e)
{
    this.Reset();
}
1 reference
public void Reset()
{
    this.cboMaSo.Text = "";
    this.txtHoTen.Text = "";
    this.txtMail.Text = "";
    this.mtxtSoDT.Text = "";
    this.rdNam.Checked = true;
    for (int i = 0; i < chklbNgoaiNgu.Items.Count - 1; i++)
        chklbNgoaiNgu.SetItemChecked(i, false);
    foreach (object ob in this.lbHocPhanDay.Items)
        this.lbDanhSachHP.Items.Add(ob);
    this.lbHocPhanDay.Items.Clear();
}
```

- Sự kiện *btnThongBao_Click* lấy thông tin đã nhập trên *frmGiangVien*, hiển thị thông tin lên *frmTBGiangVien*.

```
private void btnThongBao_Click(object sender, EventArgs e)
{
    frmTBGiangVien frm = new frmTBGiangVien();
    frm.SetText(GetGiangVien().ToString());
    frm.ShowDialog();
}
```

```
//Lấy thông tin của giảng viên được nhập trên form
1 reference
public GiangVien GetGiangVien()
{
    string gt = "Nam";
    if (rdNu.Checked)
        gt = "Nữ";
    GiangVien gv = new GiangVien();
    gv.MaSo = this.cboMaSo.Text;
    gv.GioiTinh = gt;
    gv.HoTen = this.txtHoTen.Text;
    gv.NgaySinh = this.dtpNgaySinh.Value;
    gv.Mail = this.txtMail.Text;
    gv.SoDT = this.mtxtSoDT.Text;
    string ngoaiNgu = "";
    for (int i = 0; i < chklbNgoaiNgu.Items.Count - 1; i++)
        if (chklbNgoaiNgu.GetItemChecked(i))
            ngoaiNgu += chklbNgoaiNgu.Items[i] + ";";
    gv.NgoaiNgu = ngoaiNgu.Split(';');
    DanhMucHocPhan dshp = new DanhMucHocPhan();
    foreach (object hp in lbHocPhanDay.Items)
        dshp.Them(new HocPhan(hp.ToString()));
    gv.dsHocPhan = dshp;
    return gv;
}
```

- Sự kiện *linklblLienHe_LinkClicked* mở website liên hệ https://cntt.dlu.edu.vn/.

```
private void linklbLienHe_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)
{
    string strlink = e.Link.LinkData.ToString();
    Process.Start(strlink);
}
```

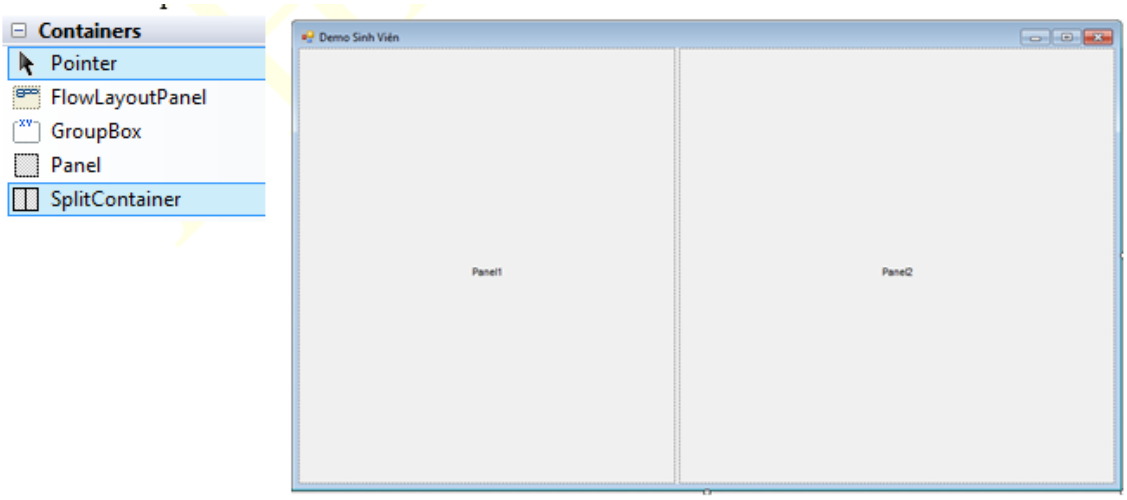
Yêu cầu 3: Quản lý sinh viên

Viết chương trình quản lý sinh viên gồm các thông tin sau: mã số, họ tên, ngày sinh, địa chỉ, lớp, hình ảnh, giới tính, chuyên ngành. Có chức năng chính sau:

- Nhập (thêm) sinh viên vào danh sách,
- Xóa sinh viên theo mã số,
- Sửa thông tin sinh viên theo mã số,
- Thiết lập các control có giá trị mặc định,

Bước 1: Thiết kế form frmSinhVien

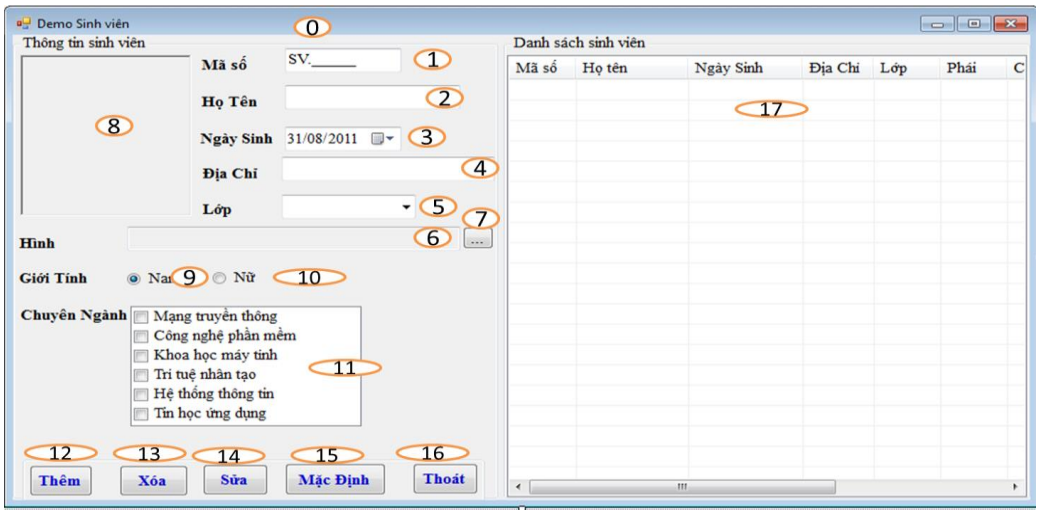
- Thêm control SplitContainer chia form thành 2 vùng:



- Thêm 1 GroupBox vào Panel 1 (groupboxTTSV); thêm 1 GroupBox (groupboxDSSV) vào Panel2. Có thuộc tính như sau:

STT	Name	Loại Control	Tên Thuộc Tính	Giá Trị
1	groupboxTTSV	GroupBox	Dock Text	Fill Thông tin sinh viên
2	groupboxDSSV	GroupBox	Dock Text	Fill Danh sách sinh viên

- Bổ sung các control bố trí như sau:



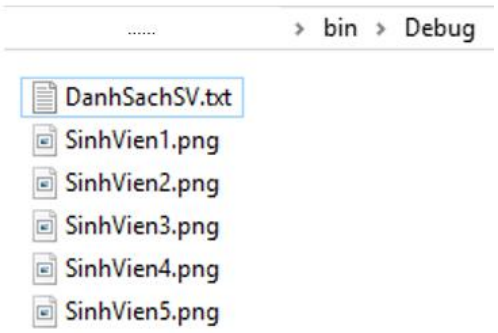
Chi tiết bảng mô tả các control:

STT	Name	Loại Control	Tên Thuộc Tính	Giá Trị
0	frmSinhVien	Form	Text	Demo Sinh viên
1	groupboxTTSV	GroupBox	Dock Text	Fill Thông tin sinh viên
2	groupboxDSSV	GroupBox	Dock Text	Fill Danh sách sinh viên

3	mtxtMaSo	MarkedTextBox	Mask	SV.00000
4	txtHoTen	TextBox		
5	dtpNgaySinh	DateTimePicker	Format	Custom
			CustomFormat	dd/MM/yyyy
6	txtDiaChi	TextBox		
7	cboLop	ComboBox	Items	CTK31 CTK32 CTK33 CTK34 CTK32CD CTK33CD CTK34CD
8	txtHinh	TextBox	ReadOnly	True
9	btnBrowse	Button	Text	…
			Chức năng	Hiển thị hộp thoại chọn hình
10	pbHinh	PictureBox	BorderStyle	Fixed3D
			SizeMode	StretchImage
11	rdNam	RadioButton	Checked	True
12	rdNu	RadioButton		
13	clbChuyenNganh	CheckListBox	CheckOnClick	True
			Items	Mạng truyền thông Công nghệ phần mềm Khoa học máy tính Trí tuệ nhân tạo Hệ thống thông tin Tin học ứng dụng
14	btnThem	Button	Text	Thêm
			Chức năng	Thêm sinh viên đã nhập
15	btnXoa	Button	Text	Xóa
			Chức năng	Xóa SV check trên ListView
16	btnSua	Button	Text	Sửa
			Chức năng	Sửa SV chọn trên ListView
17	btnMacDinh	Button	Text	Mặc Định

			Chức năng	Reset lại các controls
18	btnThoat	Button	Text	Thoát
			Chức năng	Thoát chương trình
19	lvSinhVien	ListView	CheckBox	True
			Dock	Fill
			GridLine	True
			View	Details
			Columns	Tạo các Columns: Mã số, Họ tên, Ngày sinh, Địa chỉ, Lớp, Phái, Chuyên ngành, Hình
20	OpenFileDialogHinh	OpenFileDialog	Filter	Image File(*.bmp;*.jpg;*.png)|*.bmp;*.jpg;*.png

Bước 2: Tạo tập tin và hình ảnh trong thư mục bin/Debug của Project.

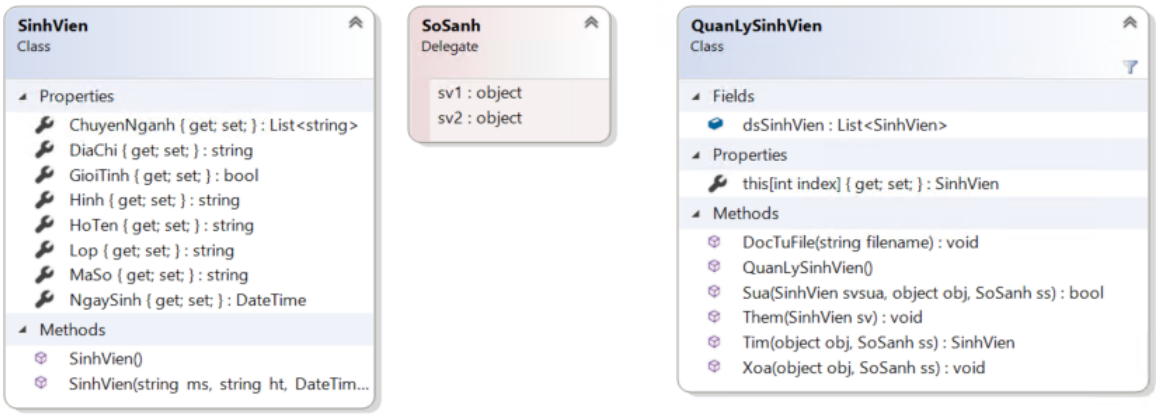


- Tạo tập tin DanhSachSV.txt có nội dung sau vào thư mục bin/Debug của Project.

MSSV	Họ và tên	Ngày sinh	Địa chỉ	Lớp	Hình	Giới tính	Chuyên ngành
SV.00002	Nguyễn Văn A	10/15/2003	Đà Lạt	CTK47	SinhVien1.png	1	Mạng truyền thông,Công nghệ phần mềm
SV.00010	Nguyễn Thị Hoa	7/20/2002	Hà Nội	CTK47	SinhVien2.png	0	Công nghệ phần mềm,Trí tuệ nhân tạo,Hệ thống thông tin
SV.00020	Nguyễn Văn Hùng	8/25/2003	Gia Lai	CTK47	SinhVien3.png	1	Mạng truyền thông,Trí tuệ nhân tạo,Hệ thống thông tin
SV.00018	Lê Thị Anh Thư	11/20/2001	Đà Lạt	CTK47	SinhVien4.png	0	Công nghệ phần mềm,Trí tuệ nhân tạo,Hệ thống thông tin
SV.00013	Hoàng Lâm Vi	9/20/2003	Bình Định	CTK47	SinhVien5.png	0	Mạng truyền thông,Trí tuệ nhân tạo,Hệ thống thông tin

- Hình ảnh đại diện mỗi sinh viên: MSSV.jpg

Bước 3: Viết code chương trình theo sơ đồ lớp về Sinh viên; Quản lý sinh viên.



Bước 3.1: Lớp SinhVien.cs.

```
public class SinhVien
{
    13 references
    public string MaSo { get; set; }
    9 references
    public string HoTen { get; set; }
    9 references
    public DateTime NgaySinh { get; set; }
    6 references
    public string DiaChi { get; set; }
    6 references
    public string Lop { get; set; }
    7 references
    public string Hinh { get; set; }
    8 references
    public bool GioiTinh { get; set; }
    7 references
    public List<string> ChuyenNganh { get; set; }
    3 references
    public SinhVien()
    {
        ChuyenNganh = new List<string>();
    }

    public SinhVien(string ms, string ht, DateTime ngay, string dc, string lop,
        string hinh, bool gt, List<string> cn)
    {
        this.MaSo = ms;
        this.HoTen = ht;
        this.NgaySinh = ngay;
        this.DiaChi = dc;
        this.Lop = lop;
        this.Hinh = hinh;
        this.GioiTinh = gt;
        this.ChuyenNganh = cn;
    }
}
```

Bước 3.2: Lớp QuanLySinhVien.cs.

```
public delegate int SoSanh(object sv1, object sv2);
6 references
class QuanLySinhVien
{
    public List<SinhVien> dsSinhVien;
    3 references
    public QuanLySinhVien()
    {
        dsSinhVien = new List<SinhVien>();
    }
    0 references
    public SinhVien this[int index]
    {
        get { return this.dsSinhVien[index]; }
        set { dsSinhVien[index] = value; }
    }
}
```

- Phương thức thêm 1 sinh viên vào danh sách.

```
public void Them(SinhVien sv)
{
    this.dsSinhVien.Add(sv);
}
```

- Phương thức tìm sinh viên đầu tiên thỏa điều kiện so sánh.

```
public SinhVien Tim(object obj, SoSanh ss)
{
    SinhVien svresult = null;
    foreach (SinhVien sv in dsSinhVien)
        if (ss(obj, sv) == 0)
        {
            svresult = sv;
            break;
        }
    return svresult;
}
```

- Phương thức sửa thông tin sinh viên thỏa điều kiện so sánh.

```
public bool Sua(SinhVien svsua, object obj, SoSanh ss)
{
    int i, count;
    bool kq = false;
    count = this.dsSinhVien.Count - 1;
    for (i = 0; i < count; i++)
        if (ss(obj, this[i]) == 0)
        {
            this[i] = svsua;
            kq = true;
            break;
        }
    return kq;
}
```

- Phương thức xóa thông tin sinh viên thỏa điều kiện so sánh.

```
public void Xoa(object obj, SoSanh ss)
{
    int i = dsSinhVien.Count - 1;
    for (; i >= 0; i--)
        if (ss(obj, this[i]) == 0)
            this.dsSinhVien.RemoveAt(i);
}
```

- Phương thức đọc danh sách sinh viên từ tập tin (có nội dung tập tin DanhSachSV đã tạo ở Bước 2.

```
public void DocTuFile(string filename)
{
    string  t;
    string[] s;
    SinhVien sv;
    using (StreamReader sr = new StreamReader(
     new FileStream(filename,
    FileMode.Open)))
    {
        while ((t = sr.ReadLine()) != null)
        {
            s = t.Split('\t');
            sv = new SinhVien();
            sv.MaSo = s[0];
            sv.HoTen = s[1];
            sv.NgaySinh = DateTime.Parse(s[2]);
            sv.DiaChi = s[3];
            sv.Lop = s[4];
            sv.Hinh = s[5];
            sv.GioiTinh = false;
            if (s[6] == "1")
                sv.GioiTinh = true;
            string[] cn = s[7].Split(',');
            foreach (string c in cn)
                sv.ChuyenNganh.Add(c);
            this.Them(sv);
        }
    }
}
```

Bước 4: Xử lý sự kiện cho form frmSinhVien

- Các phương thức bổ trợ xử lý các sự kiện:
  - Phương thức thêm thông tin sinh viên lên ListView lvSinhVien.

```
private void ThemSV(SinhVien sv)
{
    ListViewItem lvitem = new ListViewItem(sv.MaSo);
    lvitem.SubItems.Add(sv.HoTen);
    lvitem.SubItems.Add(sv.NgaySinh.ToShortDateString());
    lvitem.SubItems.Add(sv.DiaChi);
    lvitem.SubItems.Add(sv.Lop);
    string gt = "Nữ";
    if (sv.GioiTinh)
        gt = "Nam";
    lvitem.SubItems.Add(gt);
    string cn = "";
    foreach (string s in sv.ChuyenNganh)
        cn += s + ",";
    cn = cn.Substring(0, cn.Length - 1);
    lvitem.SubItems.Add(cn);
    lvitem.SubItems.Add(sv.Hinh);
    this.lvSinhVien.Items.Add(lvitem);
}
```

- Phương thức hiển thị danh sách sinh viên lên ListView lvSinhVien.

```
//Hiển thị các sinh viên trong qlsv lên ListView
4 references
private void LoadListView()
{
    this.lvSinhVien.Items.Clear();
    foreach (SinhVien sv in qlsv.dsSinhVien)
    {
        ThemSV(sv);
    }
}
```

- Phương thức lấy thông tin sinh viên từ thông tin nhập trên các control.

```
private SinhVien GetSinhVien()
{
    SinhVien sv = new SinhVien();
    bool gt = true;
    List<string> cn = new List<string>();
    sv.MaSo = this.mtxtMaSo.Text;
    sv.HoTen = this.txtHoTen.Text;
    sv.NgaySinh = this.dtpNgaySinh.Value;
    sv.DiaChi = this.txtDiaChi.Text;
    sv.Lop = this.cboLop.Text;
    sv.Hinh = this.txtHinh.Text;
    if (rdNu.Checked)
        gt = false;
    sv.GioiTinh = gt;
    for (int i = 0; i < this.clbChuyenNganh.Items.Count; i++)
        if (clbChuyenNganh.GetItemChecked(i))
            cn.Add(clbChuyenNganh.Items[i].ToString());
    sv.ChuyenNganh = cn;
    return sv;
}
```

- Phương thức lấy thông tin sinh viên từ item của ListView lvSinhVien.

```
private SinhVien GetSinhVienLV(ListViewItem lvitem)
{
    SinhVien sv = new SinhVien();
    sv.MaSo = lvitem.SubItems[0].Text;
    sv.HoTen = lvitem.SubItems[1].Text;
    sv.NgaySinh = DateTime.Parse(lvitem.SubItems[2].Text);
    sv.DiaChi = lvitem.SubItems[3].Text;
    sv.Lop = lvitem.SubItems[4].Text;
    sv.GioiTinh = false;
    if (lvitem.SubItems[5].Text == "Nam")
        sv.GioiTinh = true;
    List<string> cn = new List<string>();
    string[] s = lvitem.SubItems[6].Text.Split(',');
    foreach (string t in s)
        cn.Add(t);
    sv.ChuyenNganh = cn;
    sv.Hinh = lvitem.SubItems[7].Text;
    return sv;
}
```

- Phương thức thiết lập thông tin sinh viên lên control.

```
private void ThietLapThongTin(SinhVien sv)
{
    this.mtxtMaSo.Text = sv.MaSo;
    this.txtHoTen.Text = sv.HoTen;
    this.dtpNgaySinh.Value = sv.NgaySinh;
    this.txtDiaChi.Text = sv.DiaChi;
    this.cboLop.Text = sv.Lop;
    this.txtHinh.Text = sv.Hinh;
    this.pbHinh.ImageLocation = sv.Hinh;
    if (sv.GioiTinh)
        this.rdNam.Checked = true;
    else
        this.rdNu.Checked = true;

    for (int i = 0; i < this.clbChuyenNganh.Items.Count; i++)
        this.clbChuyenNganh.SetItemChecked(i, false);

    foreach (string s in sv.ChuyenNganh)
    {
        for (int i = 0; i < this.clbChuyenNganh.Items.Count; i++)
            if (s.CompareTo(this.clbChuyenNganh.Items[i]) == 0)
                this.clbChuyenNganh.SetItemChecked(i, true);
    }
}
```

- Xử lý sự kiện

  - Sự kiện *frmSinhVien_Load*: thực hiện đọc danh sách sinh viên từ tập tin và hiển thị lên ListView sinh viên.

```
private void frmSinhVien_Load(object sender, EventArgs e)
{
    qlsv = new QuanLySinhVien();
    qlsv.DocTuFile("DanhSachSV.txt");
    LoadListView();
}
```

- Sự kiện *lvSinhVien_SelectedIndexChanged*: khi người dùng chọn sinh viên từ ListView sinh viên thì thực hiện thiết lập thông tin sinh viên được chọn lên các control tương ứng.

```
private void lvSinhVien_SelectedIndexChanged(object sender, EventArgs e)
{
    int count = this.lvSinhVien.SelectedItems.Count;
    if (count > 0)
    {
        ListViewItem lvitem = this.lvSinhVien.SelectedItems[0];
        SinhVien sv = GetSinhVienLV(lvitem);
        ThietLapThongTin(sv);
    }
}
```

- Sự kiện btnMacDinh_Click: thiết lập các control có giá trị mặc định.

```
private void btnMacDinh_Click(object sender, EventArgs e)
{
    this.mtxtMaSo.Text = "";
    this.txtHoTen.Text = "";
    this.dtpNgaySinh.Value = DateTime.Now;
    this.txtDiaChi.Text = "";
    this.cboLop.Text = this.cboLop.Items[0].ToString();
    this.txtHinh.Text = "";
    this.pbHinh.ImageLocation = "";
    this.rdNam.Checked = true;
    for (int i = 0; i < this.clbChuyenNganh.Items.Count - 1;
    i++)
        this.clbChuyenNganh.SetItemChecked(i, false);
}
```

- Sự kiện *btnThoat_Click*: thoát chương trình.

```
private void btnThoat_Click(object sender, EventArgs e)
{
    Application.Exit();
}
```

- Sự kiện *btnXoa_Click*: xóa tất cả sinh viên đã chọn trong ListView sinh viên.

```
private int SoSanhTheoMa(object sv1, object sv2)
{
    SinhVien sv = sv2 as SinhVien;
    return sv.MaSo.CompareTo(sv1);
}
```

```
private void btnXoa_Click(object sender, EventArgs e)
{
    int count, i;
    ListViewItem lvitem;
    count = this.lvSinhVien.Items.Count - 1;
    for (i = count; i >= 0; i--)
    {
        lvitem = this.lvSinhVien.Items[i];
        if (lvitem.Checked)
            qlsv.Xoa(lvitem.SubItems[0].Text, SoSanhTheoMa);
    }
    this.LoadListView();
    this.btnMacDinh.PerformClick();

}
```

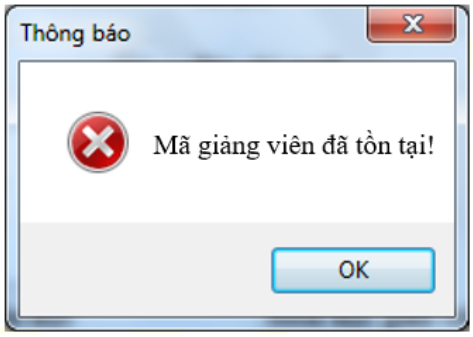
- Sự kiện btnSua_Click: sửa thông tin sinh viên được chọn.

```
private void btnSua_Click(object sender, EventArgs e)
{
    SinhVien sv = GetSinhVien();
    bool kqsua;
    kqsua = qlsv.Sua(sv, sv.MaSo, SoSanhTheoMa);
    if (kqsua)
    {
        this.LoadListView();
    }
}
```

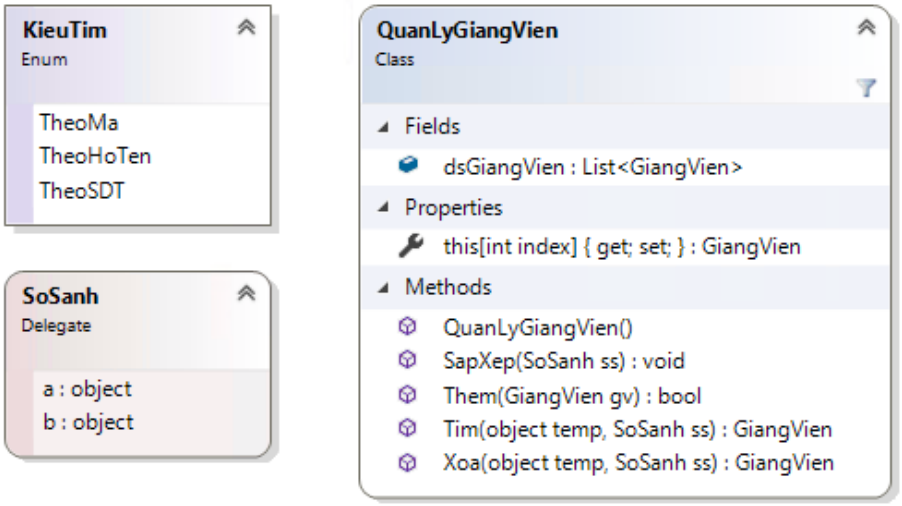
C. Bài tập

1. Thêm chức năng cho form nhập thông tin giảng viên (bài 3 phần hướng dẫn)

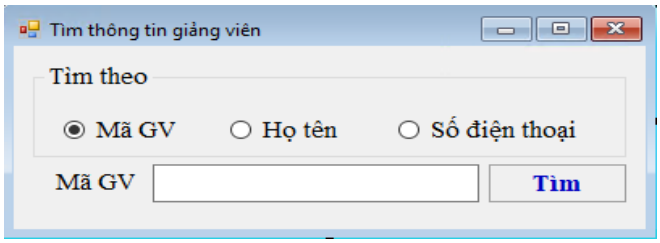
- **Chức năng 1**: Thêm nút **Thêm** xây dựng lớp QuanLyGiangVien. Khi nhấp chuột vào nút thêm thực hiện: Thêm dữ liệu giảng viên trên form cho danh sách giảng viên (mỗi giảng viên chỉ có 1 mã duy nhất). Nếu thêm giảng viên có mã trong danh sách thông báo người dùng:



Viết theo lớp sau:

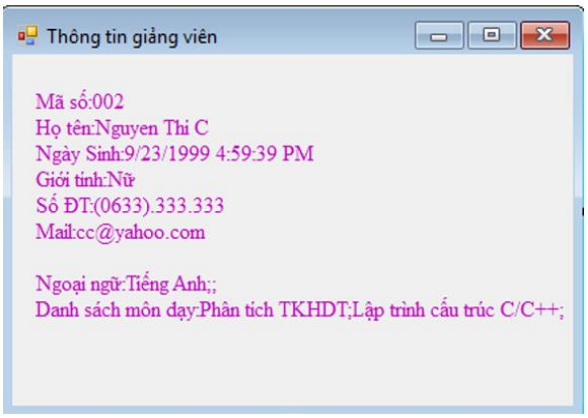


- **Chức năng 2**: Nút Tìm có chức năng hiển thị form cho phép người dùng lựa chọn chức năng tìm theo Mã giảng viên hoặc theo Họ tên hoặc theo Số điện thoại.

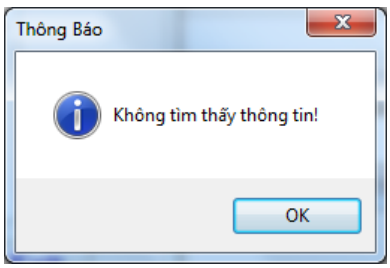


Khi nhấn Tìm:

- Nếu tìm thấy thông tin giảng viên thì hiển thị frmTBGiangVien thông tin có dạng:

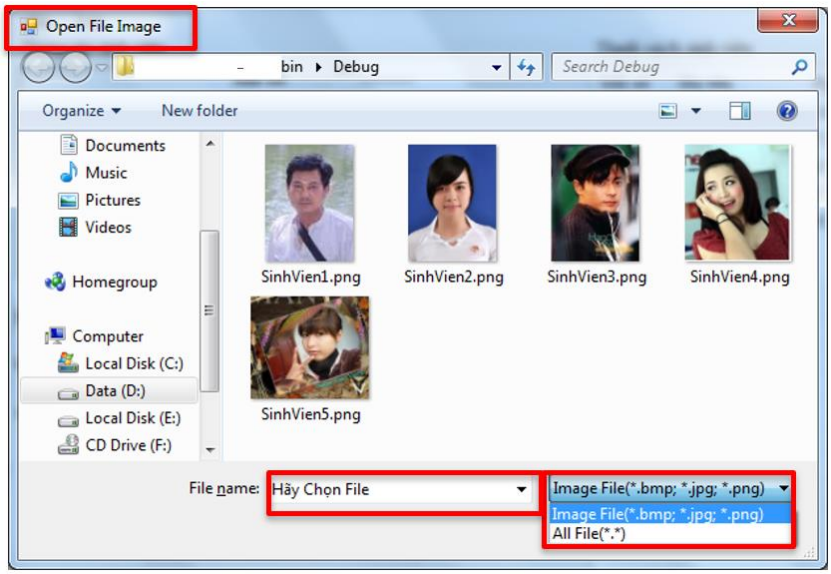


- Nếu không tìm thấy hiển thị thông báo sau:

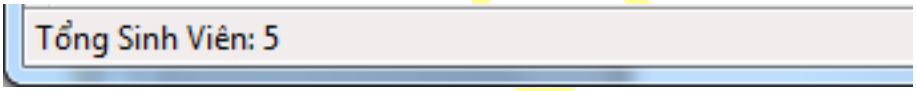


2. Thêm chức năng cho chương trình quản lý sinh viên (bài 4 phần hướng dẫn).

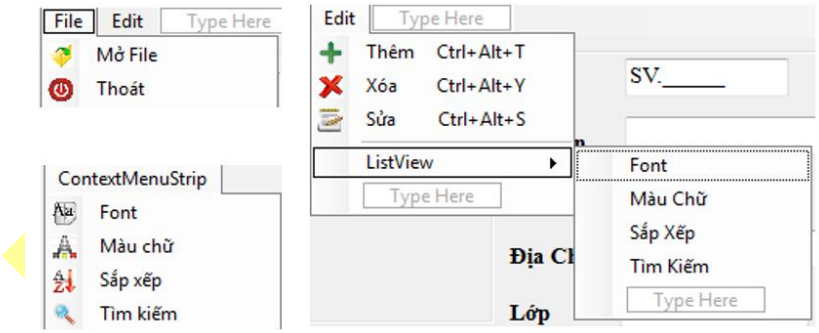
- **Chức năng 1:** Viết sự kiện cho nút **…** (btnBrowse). Chọn hình từ đĩa. Định dạng hộp thoại như sau:



Và thiết kế Statustrip: Hiển thị tổng số sinh viên trên danh sách.



- **Chức năng 2:** Thiết kế các menu

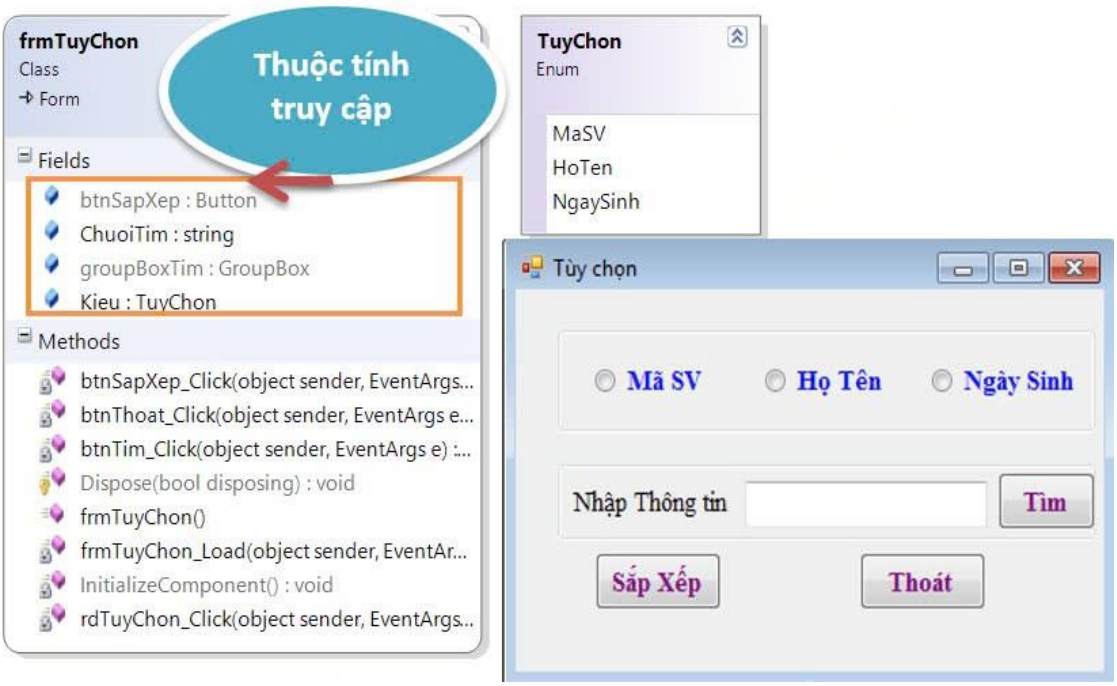


Mô tả chức năng của các menu:

STT	Menu	MenuItem	Chức năng
1	File	Mở File	Mở tập tin hình
		Thoát	Thoát chương trình
2	Edit	Thêm	Thêm sinh viên vào ListView
		Xóa	Xóa danh sách sinh viên đánh dấu Check trên ListView
		Sửa	Sửa thông tin sinh viên được chọn trong ListView

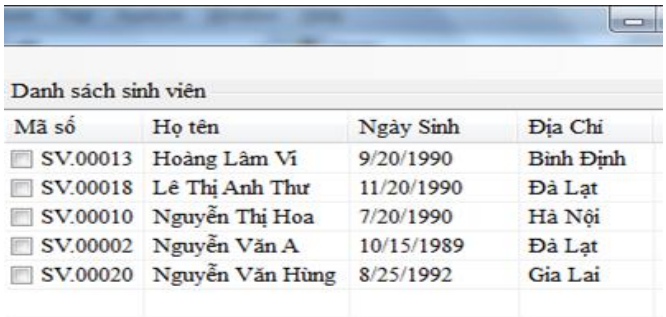
3	Edit\ListView ContextMenuStrip	Font	Chọn font chữ cho ListView
		Màu chữ	Chọn Màu chữ cho ListView
		Sắp xếp	Sắp xếp danh sách sinh viên trên ListView
		Tìm kiếm	Tìm thông tin sinh viên trên ListView

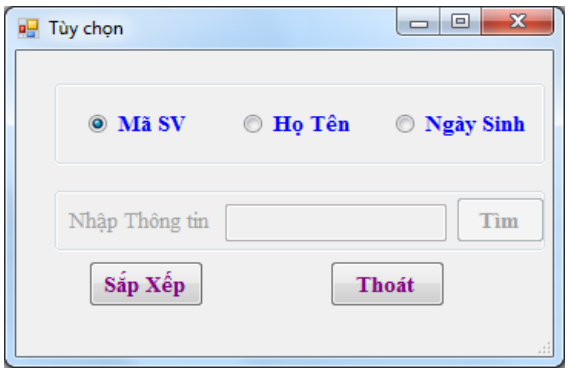
Thiết kế Form Tùy chọn với tên: frmTuyChon:



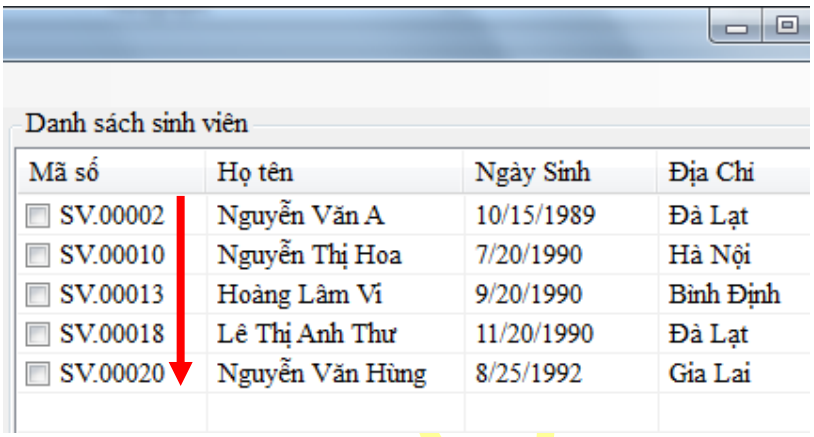
- Chi tiết chức năng Sắp xếp: thực hiện khi người dùng nhấp chuột vào menu Sắp xếp

- Hiển thị form Tùy chọn như sau:

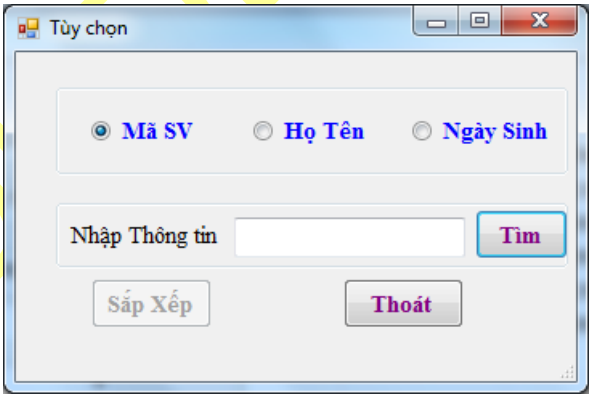




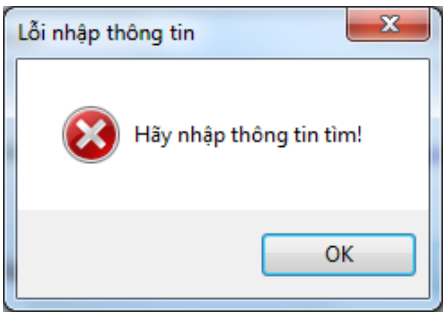
- Nhấn nút Sắp xếp thì danh sách trên ListView sắp theo kiểu chọn:



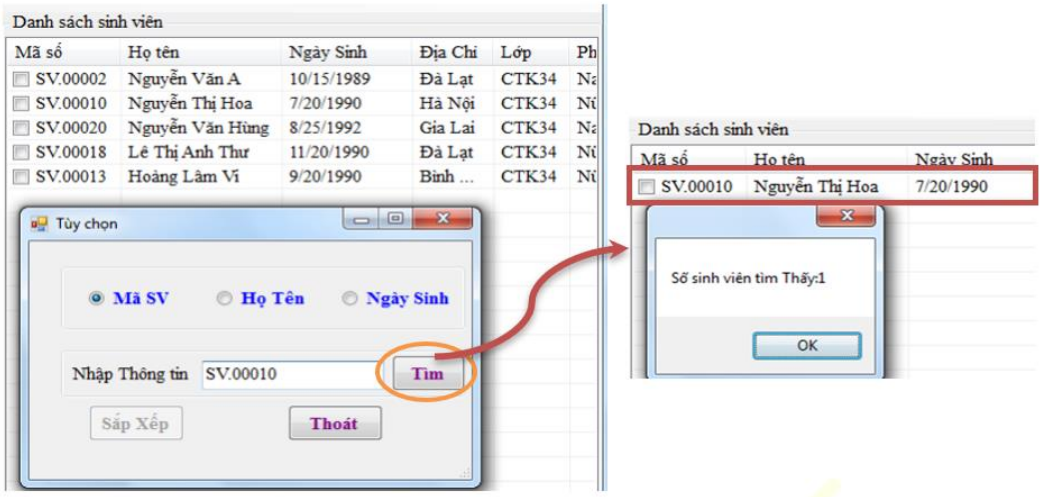
- **Chức năng Tìm kiếm:** thực hiện khi người dùng nhấp chuột vào menu Tìm kiếm.
  - Hiển thị màn hình Tùy chọn như sau:



  - Nhấn nút Tìm:
    - Nếu không nhập thông tin thông báo lỗi:

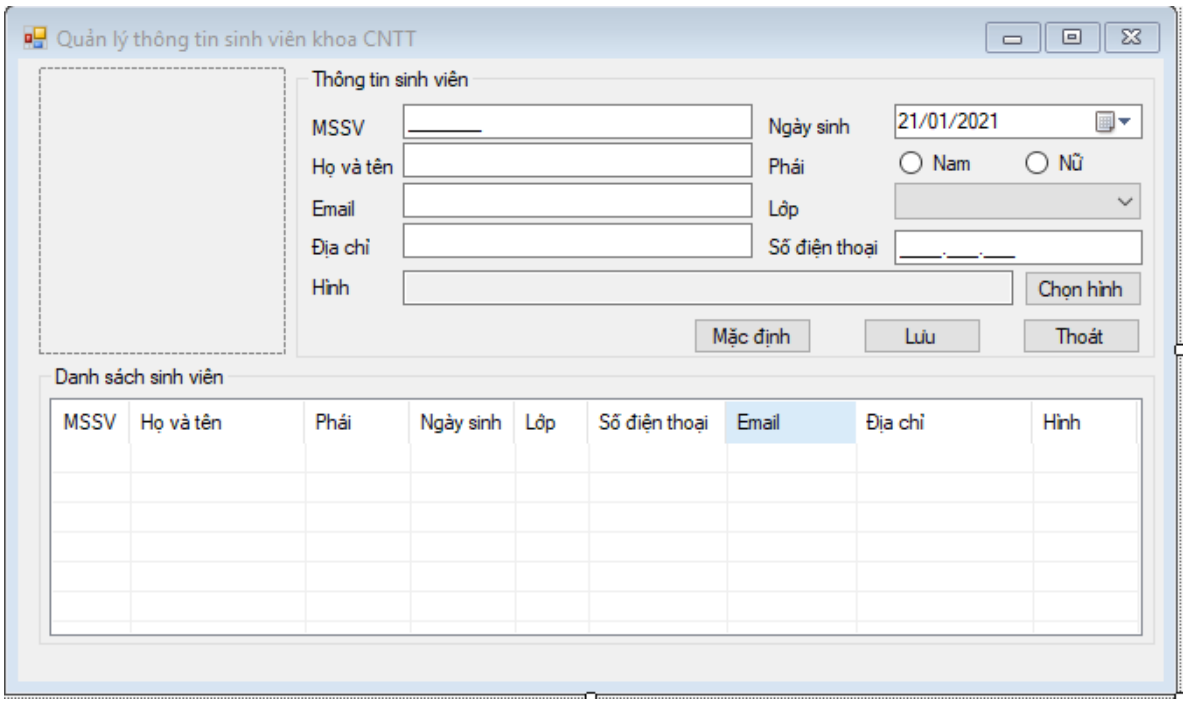


- Nếu nhập thông tin kết quả sẽ hiển thị dạng:



3. Xây dựng chương trình quản lý sinh viên khoa CNTT đơn giản.

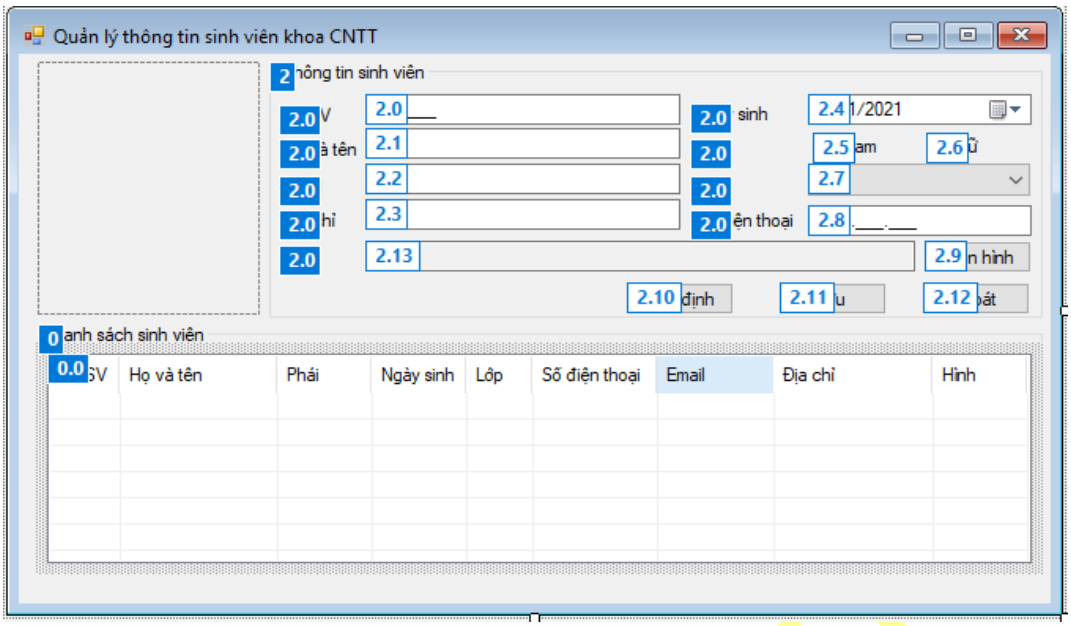
- Thiết kế form như hình sau:



- Các yêu cầu cần chú ý ở phần nhập thông tin sinh viên:

a) Mã sinh viên gồm 7 ký tự số.
b) Ngày sinh hiển thị theo định dạng "dd/mm/yyyy".
c) Giới tính: chỉ được chọn Nam hoặc Nữ.
d) Số điện thoại 10 chữ số, phân tách thành 3 nhóm.
e) Lớp: có thể chọn 1 trong các lớp sau: "CTK43", "CTK44", "CTK45", "CTK46". Lưu ý: người dùng không có quyền nhập thêm lớp khác.
f) Ô hình dùng để chứa đường dẫn của hình, không cho nhập

- Thiết lập thứ tự tab (Tab Order) như hình sau:



Yêu cầu:

- Xây dựng lớp Sinh viên, tạo một danh sách sinh viên kiểu ArrayList để lưu danh sách sinh viên.

- Các sự kiện cần xử lý:

  - Nhấp chuột vào nút "Chọn hình" cho phép người dùng chọn 1 hình ảnh, đường dẫn hình lưu vào textbox, hình ảnh được tải lên khung hình ở bên trái.
  - Nhấp chuột vào nút "Mặc định" : xóa toàn bộ nội dung được nhập ở phần nhập liệu.
  - Nhấp chuột vào nút "Thoát": đóng chương trình.
  - Nhấp chuột vào nút "Lưu": cho phép thêm sinh viên mới vào danh sách sinh viên hoặc chỉnh sửa thông tin của một sinh viên nằm trong danh sách .
  - Hướng dẫn: Trước hết, thực hiện sinh viên theo mã sinh viên, nếu tìm thấy thì cập nhật thông tin của sinh viên này; nếu không tìm thấy thực hiện thêm sinh viên này vào danh sách sinh viên và hiển thị danh sách sinh viên ở ListView.
  - Khi người dùng nhấp chuột vào một sinh viên trong danh sách sinh viên, thông tin của sinh viên sẽ được hiển thị chi tiết lên trên phần nhập liệu.
  - Khi nhấp chuột phải vào Listview, hiển thị context menu "Xóa", cho phép xóa một hoặc nhiều sinh viên đã chọn.
  - Khi form được nạp (Form_Load), tải toàn bộ danh sách sinh viên từ tập tin DSNV.txt vào ListView.
  - Cho phép người dùng tải lại danh sách sinh viên từ tập tin DSNV.txt khi người dùng nhấp chuột phải vào Listview và chọn "Tải lại danh sách".
  - Khi người dùng đóng form, kiểm tra xem người dùng có chỉnh sửa danh sách sinh viên được tải lên hay không, nếu có hiển thị MessageBox hỏi người dùng có muốn lưu danh sách đã thay đổi hay không? Nếu người dùng chọn "OK" thực hiện lưu danh sách sinh viên trên Listview vào tập tin "DSNV.txt".

CHỦ ĐỀ 3. THAO TÁC TẬP TIN

A. Mục tiêu

Chủ đề 3 giới thiệu một số thư viện hỗ trợ việc đọc và ghi tập tin dạng văn bản, tập tin nhị phân, các tập tin bán cấu trúc như CSV, XML và các bảng tính Excel. .NET Framework đã cung cấp sẵn các thư viện hàm để làm việc với tập tin dạng văn bản và nhị phân. Tuy nhiên, đối với những tập tin bán cấu trúc như CSV hay bảng tính Excel, việc xử lý tương đối phức tạp nếu không sử dụng các thư viện hàm chuyên dụng. Cộng đồng người dùng .NET đã phát triển nhiều thư viện hàm tiện ích để giải quyết vấn đề này và phân phối dưới dạng các gói Nuget Package có thể được cài đặt vào dự án khi cần thiết. Ngoài ra, rất nhiều thư viện hỗ trợ mạnh mẽ cho việc lập trình cơ sở dữ liệu quan hệ cũng đã được phát triển trong nhiều năm qua, chẳng hạn như NHibernate, BLToolkit, LinqConnect, LINQ to SQL, Entity Framework, … Chủ đề cũng giới thiệu thư viện Entity Framework được phát triển và phân phối chính thức bởi Microsoft.

Sau khi hoàn thành chủ đề, sinh viên có thể xây dựng ứng dụng Windows Form cho phép đọc và ghi các loại tập tin như *.txt, *.xml và *.json.

B. Hướng dẫn thực hành

1. Đọc và tạo tập tin dạng văn bản

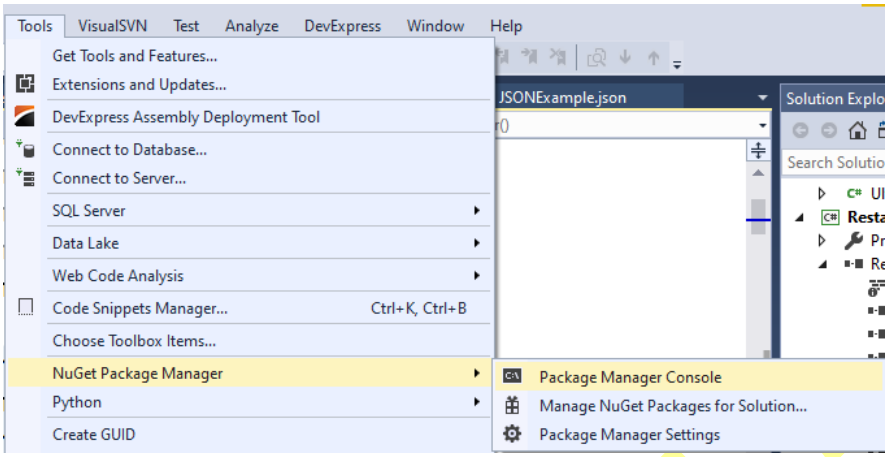
Trong .NET Framework, namespace System.IO định nghĩa sẵn một tập các lớp, các cấu trúc và các giao diện để thao tác với tập tin, thư mục và các thiết bị nhập xuất. Trong đó, hai lớp tập tin và FileInfo được dùng để làm việc với các tập tin cụ thể. Lớp tập tin cung cấp các phương thức để tạo, xóa, cập nhật, sao chép hoặc di chuyển tập tin. Trong khi đó, lớp FileInfo cho biết thêm các thuộc tính, thông tin chi tiết hơn về một tập tin, chẳng hạn như người tạo, thời gian tạo, thời điểm cập nhật,…

Ví dụ để tạo tập tin lưu dữ liệu từ mảng chuỗi text, sau đó thực hiện đọc hiển thị ra màn hình.

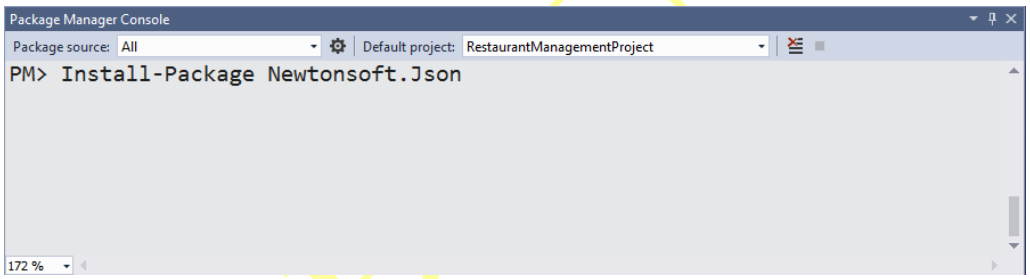
```
private void WriteReadText(string filename, string []text)
{
    //Ghi dữ liệu text vào file filename
    File.WriteAllLines(filename, text);
    //Đọc dữ liệu từ file filename hiển thị ra màn hình
    foreach (string s in File.ReadAllLines(filename))
    {
        Console.WriteLine(s);
    }
    Console.ReadLine();
}
```

2. Đọc tập tin JSON bằng Visual Studio

Để đọc được cấu trúc trong tập tin JSON bằng ngôn ngữ C#, chúng ta phải cài thêm gói thư viện Newtonsoft.Json vào dự án. Để cài gói thư viện này, thực hiện như sau: Vào menu Tools, chọn NuGet Package Manager, chọn Package Manager Console.



Khi đó cửa sổ Package Manager Console hiện ra phía dưới màn hình cho phép người dùng gõ lệnh. Từ đây có thể cài nhiều gói khác nhau vào dự án, để cài gói Newtonsoft.Json, chúng ta gõ lệnh "`Install-Package Newtonsoft.Json`" và nhấn Enter:



Hệ thống sẽ tìm phiên bản phù hợp nhất và cài vào dự án. Để sử dụng thư viện này và đọc được tập tin JSON, chúng ta sử dụng các câu lệnh using như sau:

```
using System.IO;
using Newtonsoft.Json;
using Newtonsoft.Json.Linq;
```

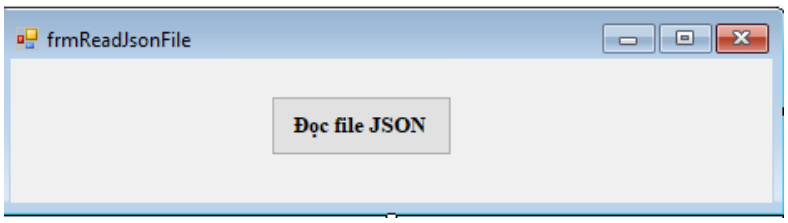
Giả sử cần đọc tập tin JSON với tên ***students.json*** như sau

```
{
  "sinhvien": [
    {
      "MSSV": "1245732", "hoten": "Thái Duy Quý",
      "tuoi": 21, "diem": 8.5, "tongiao": false
    },
    {
      "MSSV": "1245872", "hoten": "Phan Thị Thanh Nga",
      "tuoi": 20,"diem": 9.0, "tongiao": true
    }         ]
}
```

Với cấu trúc sinh viên bao gồm các trường như: MSSV, họ tên, tuổi, điểm, và tôn giáo. Khi đó, để thuận tiện cần phải xây dựng một mô hình lớp bao gồm các trường như trong tập tin JSON, mô hình lớp có thể được xây dựng như trong lớp sau đây:

```
public class StudentInfo
{
    // Các thuộc tính
    1 reference
    public string MSSV { get; set; }
    1 reference
    public string Hoten { get; set; }
    1 reference
    public int Tuoi { get; set; }
    1 reference
    public double Diem { get; set; }
    1 reference
    public bool TonGiao { get; set; }
    // Phương thức tạo lập
    0 references
    public StudentInfo(string mssv, string hoten, int tuoi,
        double diem, bool tongiao)
    {
        this.MSSV = mssv;
        this.Hoten = hoten;
        this.Tuoi = tuoi;
        this.Diem = diem;
        this.TonGiao = tongiao;
    }
}
```

Tạo form có nút Đọc tập tin JSON như hình sau:

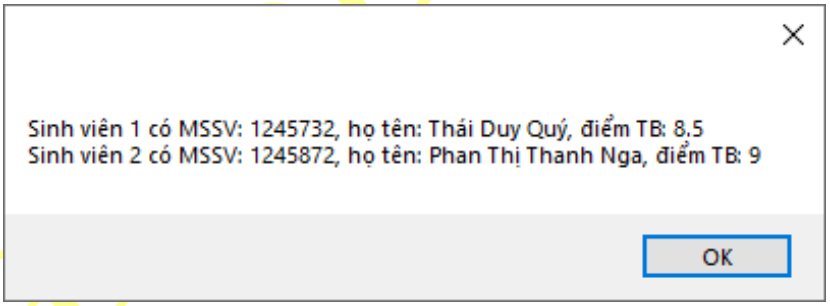


Xử lý sự kiện btnReadJSON_Click: đọc tập tin .json students.json (đã tạo ở bước trên), hiển thị trên hộp thoại.

```
private void btnReadJSON_Click(object sender, EventArgs e)
{
    string Str = ""; // chuỗi lưu trữ
    string Path = "../../students.json"; // Đường dẫn tập tin
    List<StudentInfo> List = LoadJSON(Path); // Gọi phương thức
    for (int i = 0; i < List.Count; i++) // Đọc danh sách
    {
        StudentInfo info = List[i];
        Str += string.Format("Sinh viên {0} có MSSV: {1}, họ tên: {2}," +
            " điểm TB: {3}\r\n", (i + 1), info.MSSV, info.Hoten, info.Diem);
    }
    MessageBox.Show(Str);
}
```

```
/// <summary>
/// Phương thức đọc tập tin JSON
/// </summary>
/// <param name="Path">Đường dẫn tập tin</param>
/// <returns>Danh sách các đối tượng từ tập tin JSON</returns>
0 references
private List<StudentInfo> LoadJSON(string Path)
{
    // Khai báo danh sách lưu trữ
    List<StudentInfo> List = new List<StudentInfo>();
    // Đối tượng đọc tập tin
    StreamReader r = new StreamReader(Path);
    string json = r.ReadToEnd(); // Đọc hết
                                 // Chuyển về thành mảng các đối tượng
    var array = (JObject)JsonConvert.DeserializeObject(json);
    // Lấy đối tượng sinhvien
    var students = array["sinhvien"].Children();
    foreach (var item in students) // Duyệt mảng
    {
        // Lấy các thành phần
        string mssv = item["MSSV"].Value<string>();
        string hoten = item["hoten"].Value<string>();
        int tuoi = item["tuoi"].Value<int>();
        double diem = item["diem"].Value<double>();
        bool tongiao = item["tongiao"].Value<bool>();
        // Chuyển vào đối tượng StudentInfo
        StudentInfo info = new StudentInfo(mssv, hoten, tuoi, diem, tongiao)
        List.Add(info);// Thêm vào danh sách
    }
    return List;
}
```

Kết quả chạy chương trình khi nhấp chuột vào nút "**Đọc tập tin JSON**":



3. Đọc và ghi tập tin XML

XML là một ngôn ngữ có khả năng tự mô tả. Tập tin XML vừa chứa dữ liệu, vừa chứa các quy tắc và thông tin để có thể rút trích được dữ liệu từ tập tin đó. Có hai cách phổ biến để thao tác với các tập tin XML. Thứ nhất là dùng các lớp được cung cấp sẵn bởi .NET Framework trong namespace System.Xml. Bạn phải nạp namespace này vào mã nguồn trước khi sử dụng các lớp để đọc, ghi nội dung XML. Thứ hai là sử dụng LINQ to XML. LINQ to XML là một tập hợp các thư viện hàm cho phép bạn nạp tập tin XML vào bộ nhớ để xử lý và cập nhật tài liệu XML một cách thuận tiện và hiệu quả. Để sử dụng LINQ to XML, bạn cần nạp namespace System.Xml.Linq vào mã nguồn. Ngoài ra, còn có một số thư viện của bên thứ 3 nhằm giúp xử lý nội dung XML đơn giản và dễ dàng hơn.

Việc đọc tập tin XML và phân tích nội dung bên trong các thẻ XML có thể được thực hiện bằng nhiều cách khác nhau tùy thuộc vào yêu cầu cụ thể. Bạn có thể sử dụng lớp XmlDataDocument, XmlReader, XmlTextReader, XmlDocument trong namespace System.Xml. Hoặc bạn có thể sử dụng các lớp XDocument,, XElement trong namespace System.Xml.Linq. Phần này giới thiệu cách sử dụng lớp XmlDocument để đọc, phân tích nội dung và truy vấn tài liệu XML sử dụng XPath. Lớp XmlDocument đọc toàn bộ nội dung XML vào bộ nhớ và cho phép chúng ta đọc nội dung các thẻ bất kỳ, điều hướng qua lại giữa phần tử cha-con, các phần tử cùng cấp.

Giả sử ta có tập tin books.xml như dưới đây:

```
<?xml version="1.0" encoding="utf-8" ?>
<!-- Books.xml stores information about Mahesh Chand and related books
-->
<catalog>
 <book ISBN="9831123212" yearpublished="2002">
       <title>A Programmer's Guide to ADO .Net using C#</title>
       <author>
             <first-name>Mahesh</first-name>
             <last-name>Chand</last-name>
       </author>
       <publisher>Apress</publisher>
       <price>44.99</price>
 </book>
 <book ISBN="9781484234" yearpublished="2019">
       <title>Pro Entity Framework Core 2</title>
       <author>
             <first-name>Adam</first-name>
             <last-name>Freeman</last-name>
       </author>
       <publisher>Apress</publisher>
       <price>45.09</price>
 </book>
</catalog>
```

Để đọc nội dung XML và lấy thông tin danh mục sách trong tập tin nói trên, ta có thể sử dụng lớp XmlDocument như sau:

```
public static void Main(string[] args)
{
    // Load XML file into XmlDocument instance
    var xmlDoc = new XmlDocument();
    xmlDoc.Load("..\\..\\books.xml");

    // Get list of nodes whose name is Book
    var nodeList = xmlDoc.DocumentElement.SelectNodes("/catalog/book");

    foreach (XmlNode node in nodeList)
    {
        // Read attribute value
        var isbn = node.Attributes["ISBN"].Value;
        // Read child node value
        var title = node.SelectSingleNode("title").InnerText;
        var price = node.SelectSingleNode("price").InnerText;
        // Read the descendant node value
        var firstName = node.SelectSingleNode("author/first-name").InnerText;
        var lastName = node.SelectSingleNode("author/last-name").InnerText;
        Console.WriteLine("{0,-15}{1,-50}{2,-15}{3,-15}{4,6}",
            isbn, title, firstName, lastName, price);
    }
}
```

Kết quả đọc như sau:

9831123212	A Programmer's Guide to ADO .Net using C#	Mahesh	Chand	44.99
9781484234	Pro Entity Framework Core 2	Adam	Freeman	45.09

Tương tự, việc tạo tập tin XML từ dữ liệu cho trước cũng tương đối đơn giản. Có thể sử dụng các lớp trong namespace System.Xml hoặc sử dụng LINQ to XML tùy theo yêu cầu dự án. Sử dụng lớp XmlWriter để ghi dữ liệu ra một tập tin books.xml có nội dung như sau:

```
<?xml version="1.0" encoding="utf-8"?>
<?xml-stylesheet type="text/xsl" href="book.xsl"?>
<!DOCTYPE book [
 <!ENTITY h "hardcover">
]>
<!--This is a book sample XML-->
<book ISBN="9831123212" yearpublished="2002">
 <author>Mahesh Chand</author>
 <title>Visual C# Programming</title>
 <price>44.95</price>
</book>
```

```
public static void Main(string[] args)
{
    using (XmlWriter writer = XmlWriter.Create("books.xml"))
    {
        // Write Processing Instruction
        String pi = "type=\"text/xsl\" href=\"book.xsl\"";
        writer.WriteProcessingInstruction("xml-stylesheet", pi);
        // Write DocumentType
        writer.WriteDocType("catalog", null, null, "<!ENTITY h \"hardcover\">");
        // Write a Comment
        writer.WriteComment("This is a book sample XML");
        // Root element - start tag
        writer.WriteStartElement("book");
        // Write ISBN attribute
        writer.WriteAttributeString("ISBN", "9831123212");
        // Write year attribute
        writer.WriteAttributeString("yearpublished", "2002");
        // Write title
        writer.WriteElementString("author", "Mahesh Chand");
        // Write author
        writer.WriteElementString("title", "Visual C# Programming");
        // Write price
        writer.WriteElementString("price", "44.95");
        // Root element - end tag
        writer.WriteEndElement();
        // End Documentd
        writer.WriteEndDocument();
        // Flush it
        writer.Flush();
    }
}
```

Việc ghi nội dung XML như trên sẽ trở nên phức tạp và khó kiểm soát khi lượng dữ liệu lớn và có nhiều thông tin hơn. Nếu bạn có một danh sách các đối tượng và muốn lưu trữ dữ liệu của chúng dưới dạng tập tin XML, lớp XmlSerializer sẽ là một lựa chọn phù hợp. Giả sử, ta có lớp Book chứa 5 thuộc tính mô tả về một cuốn sách như sau:

```
public class Book
{
    0 references
    public string ISBN { get; set; }
    0 references
    public string Title { get; set; }
    0 references
    public string Author { get; set; }
    0 references
    public decimal Price { get; set; }
    0 references
    public int YearPublished { get; set; }
}
```

Khi đó, để lưu dữ liệu từ một mảng đối tượng books vào tập tin books.xml, ta sử dụng đoạn mã sau:

```
private static void SaveToXmlFile(List<Book> books)
{
    var serializer = new XmlSerializer(typeof(List<Book>));

    using (var writer = new StreamWriter("books.xml"))
    {
        serializer.Serialize(writer, books, null);
        writer.Close();
    }
}
```

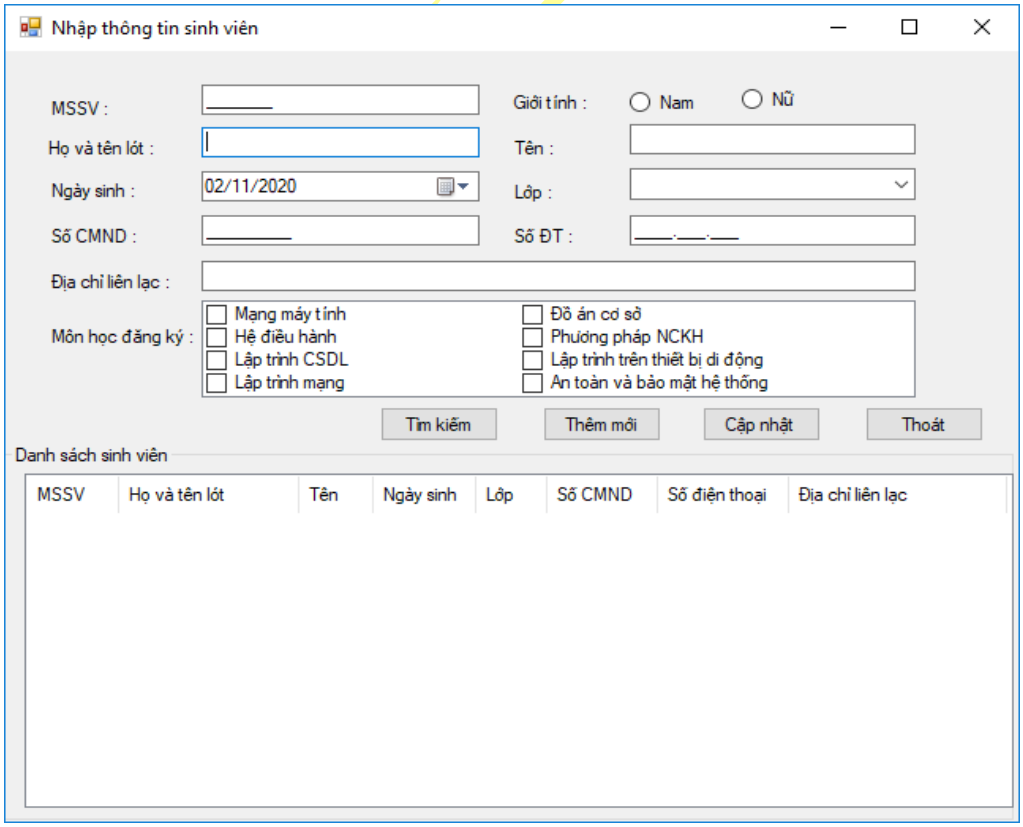
Tập tin books.xml cho ra nội dung kết quả:

```
<?xml version="1.0" encoding="utf-8"?>
<ArrayOfBook
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <Book>
       <ISBN>9831123212</ISBN>
       <Title>A Programmer's Guide to ADO .Net using C#</Title>
       <Author>Mahesh Chand</Author>
       <Price>44.99</Price>
       <YearPublished>2002</YearPublished>
 </Book>
 <Book>
       <ISBN>9781484234</ISBN>
       <Title>Pro Entity Framework Core 1</Title>
       <Author>Adam Freeman</Author>
       <Price>44.99</Price>
       <YearPublished>2018</YearPublished>
 </Book>
</ArrayOfBook>
```

C. Bài tập

1. Viết chương trình ứng dụng nhập thông tin sinh viên sử dụng đọc/ghi file

- Thiết kế form như sau:



- Thực hiện yêu cầu sau:
  - Định nghĩa lớp Student (SinhVien) có các thuộc tính và phương thức cần thiết.

- Xây dựng các lớp để đọc, ghi danh sách sinh viên từ tập tin văn bản thông thường (*.txt).
- Xây dựng lớp StudentManager(QLSinhVien) với các chức năng cơ bản như sau:
  a. Thêm và cập nhật thông tin sinh viên (có lưu vào file)
  b. Tìm kiếm sinh viên theo Tên, Lớp, MSSV
  c. Xóa một hay nhiều sinh viên (có lưu vào file)

2. Chương trình phải đảm bảo:

- Khi chạy chương trình danh sách sinh viên sẽ được tải từ tập tin.
- MSSV gồm 7 chữ số, Số CMND gồm 9 chữ số và Số ĐT nhập 10 chữ số.
- Môn đăng ký: cho phép nhấp chuột phải để mở ContextMenu cho phép xóa hoặc thêm môn.
- Khi chọn một sinh viên trong danh sách sinh viên, thông tin của sinh viên phải được tự động điền lên phần thông tin phía trên danh sách.
- Người dùng phải nhập hết thông tin rồi mới cho phép thêm mới hoặc cập nhật. Nếu người dùng chưa nhập đầy đủ thông tin mà đã nhấn nút Thêm mới/ Cập nhật thì cần thông báo cho người dùng.
- Danh sách sinh viên: cho phép chọn nhiều sinh viên dùng Checkbox, cho phép nhấp chuột phải để mở ContextMenu cho phép xóa 1 hoặc nhiều sinh viên đã chọn.
- Khi nhấp chuột vào nút Tìm kiếm, chương trình phải hiển thị form cho người dùng nhập điều kiện tìm kiếm (theo MSSV/Tên/Lớp).
- Khi người dùng nhấn nút Thoát phải hỏi lại người dùng có chắc chắn muốn thoát chương trình hay không. Nếu đồng ý thì thoát chương trình.

3. Bổ sung chức năng của chương trình như sau:

- MSSV gồm 7 chữ số có dạng AABBCCC, trong đó AA là 2 số cuối năm nhập học của sinh viên, BB = 10, CCC là số bất kỳ. Không có sinh viên nào trùng MSSV. Lưu ý: dựa vào lớp để biết năm nhập học của sinh viên.
- Xây dựng các lớp để đọc, ghi danh sách sinh viên từ tập tin. Yêu cầu: chương trình có thể hỗ trợ các định dạng tập tin sau đây:
  a. Tập tin văn bản thông thường: students.txt
  b. Tập tin xml: students.xml
  c. Tập tin JSON: students.json
- Chương trình cho phép tìm sinh viên theo một hoặc nhiều điều kiện (điều kiện bất kỳ).

CHỦ ĐỀ 4. KẾT NỐI VÀ TRUY VẤN DỮ LIỆU

A. Mục tiêu

Chủ đề 4 giúp sinh viên tìm hiểu cách kết nối đến một cơ sở dữ liệu và thực thi trực tiếp một số lệnh truy vấn đơn giản:

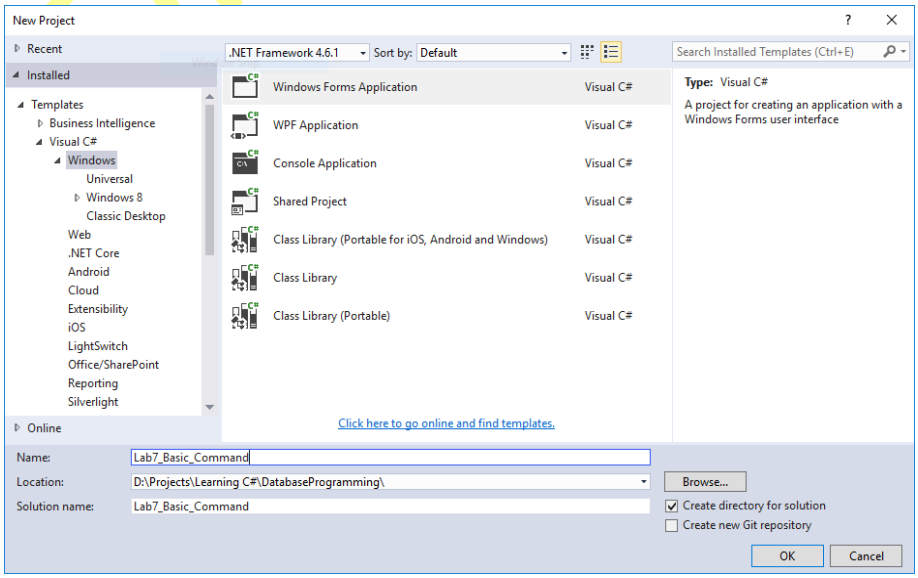
- SELECT: Lấy các mẫu tin từ một bảng hoặc khung nhìn.
- INSERT: Thêm một mẫu tin mới vào một bảng.
- UPDATE: Cập nhật một mẫu tin có sẵn trong bảng.
- DELETE: Xóa một mẫu tin khỏi bảng.

Sau hoàn thành chủ đề, sinh viên cần nắm rõ những vấn đề sau:

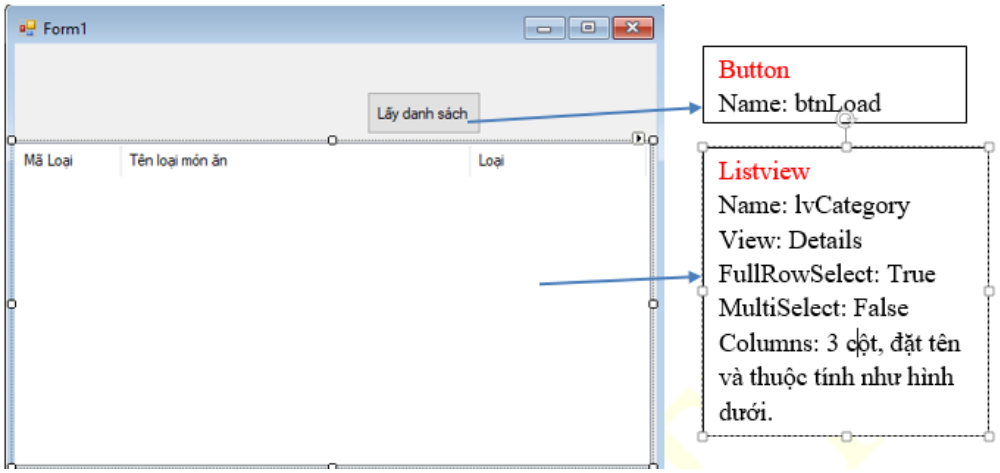
- Các thành phần của chuỗi kết nối và ý nghĩa của chúng.
  - Cách tạo đối tượng kết nối đến các cơ sở dữ liệu SQL Server, Access.
  - Cách sử dụng đối tượng Command để truy vấn, DataReader để đọc dữ liệu.
  - Cách xây dựng ứng dụng trên nền Windows Form.

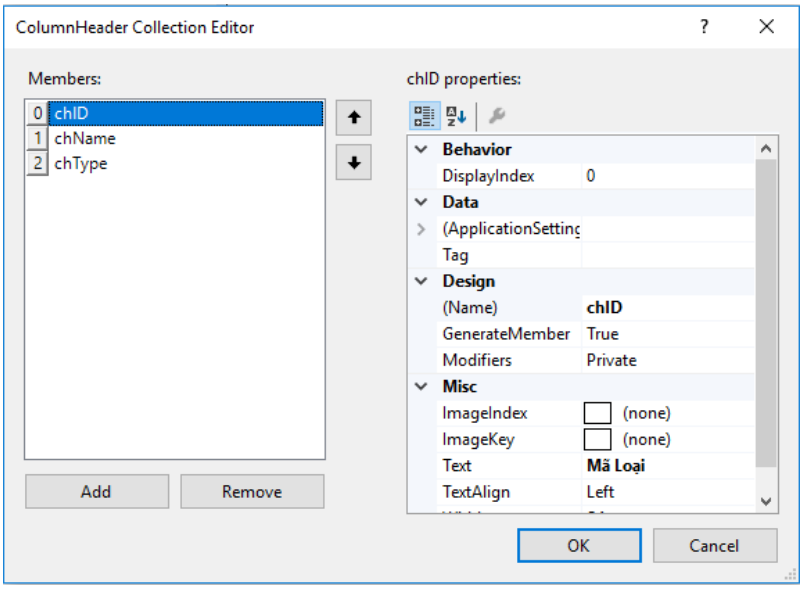
B. Hướng dẫn thực hànhb

Tạo một dự án Windows Application mới, đặt tên là ***Lab_Basic_Command.*** Phần hướng dẫn tiếp theo trong chủ đề này sẽ hướng dẫn các bạn xây dựng một form quản lý danh mục (nhóm) các món ăn. Các thao tác trên form bao gồm hiển thị danh mục món ăn; thêm, xóa, cập nhật một danh mục và xem danh sách các món ăn thuộc một danh mục nào đó bằng cách nhấp chuột vào menu chuột phải (context menu).



Tạo một form mới đặt tên là ***CategoryForm*** và thiết kế form như sau:





1. Lấy dữ liệu bằng cách dùng phương thức ExecuteReader

Phần này, chúng ta sẽ tạo ra một form đơn giản chỉ có một nút "***Lấy danh sách***" . Khi nhấp chuột vào nút này, danh sách nhóm món ăn sẽ được hiển thị lên listview. Như vậy, sự kiện cần xử lý là sự kiện Click của nút "***Lấy danh sách***" (được đặt tên là btnLoad). Danh sách các nhóm món ăn sẽ được đọc từ cơ sở dữ liệu sử dụng câu truy vấn SELECT.

Đầu tiên, nhấp đôi chuột vào nút btnLoad và thêm đoạn mã sau vào đầu lớp CategoryForm.cs:

```
using System;
using System.Data;
using System.Data.SqlClient;
using System.Collections.Generic;
…
```

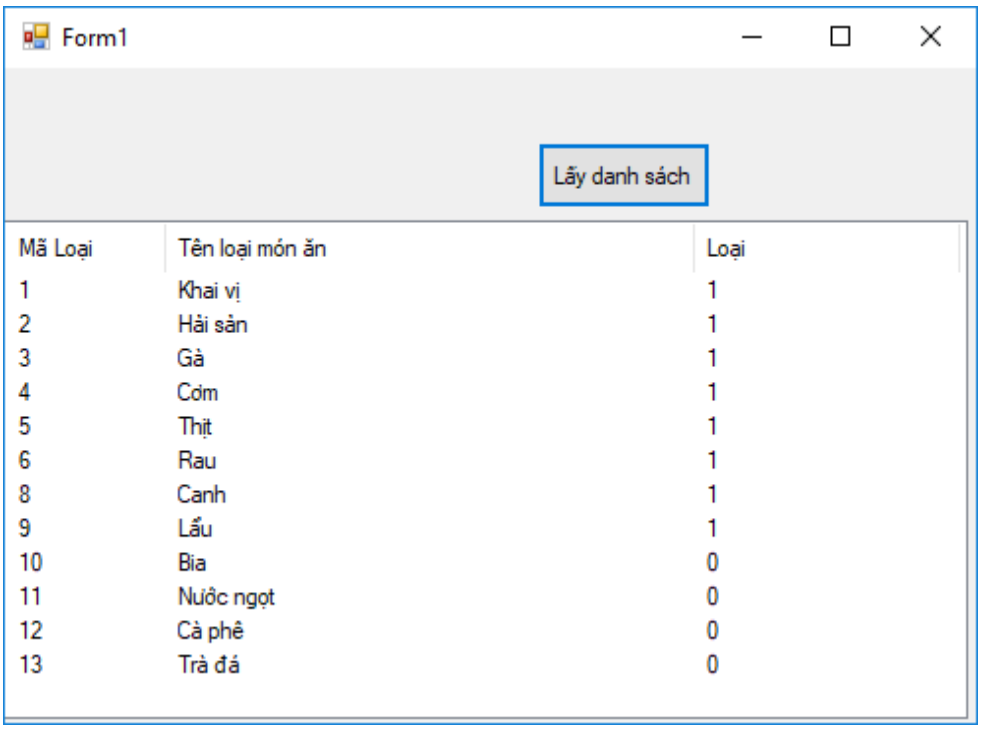
Bổ sung đoạn mã sau vào phương thức btnLoad_Click:

```
private void btnLoad_Click(object sender, EventArgs e)
{
    // Tạo chuỗi kết nối tới cơ sở dữ liệu RestaurantManagement
    string connectionString = "server=.; database = RestaurantManagement; Integrated Security = true; ";

    // Tạo đối tượng kết nối
    SqlConnection sqlConnection = new SqlConnection(connectionString);

    // Tạo đối tượng thực thi lệnh
    SqlCommand sqlCommand = sqlConnection.CreateCommand();

    // Thiết lập lệnh truy vấn cho đối tượng Command
    string query = "SELECT ID, Name, Type FROM Category";
    sqlCommand.CommandText = query;

    // Mở kết nối tới cơ sở dữ liệu
    sqlConnection.Open();

    // Thực thi lệnh bằng phương thức ExcuteReader
    SqlDataReader sqlDataReader = sqlCommand.ExecuteReader();

    // Gọi hàm hiển thị dữ liệu lên màn hình
    this.DisplayCategory(sqlDataReader);

    // Đóng kết nối
    sqlConnection.Close();
}
```

Để hiển thị dữ liệu lên ListView, bạn phải viết thêm phương thức ***DisplayCategory*** như sau:

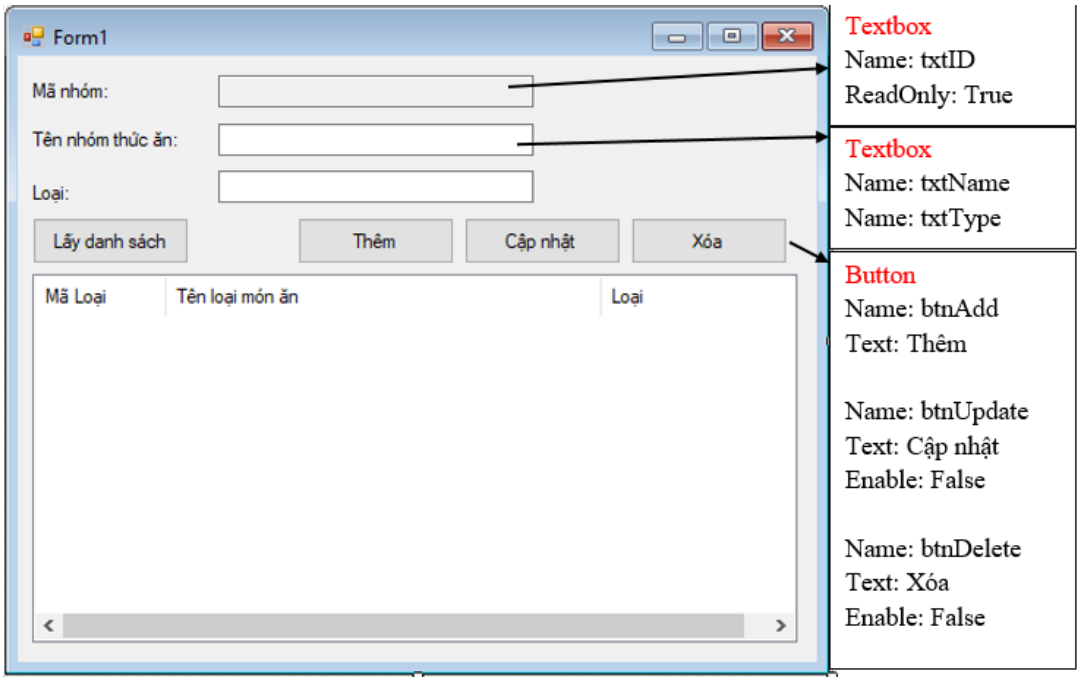
```
private void DisplayCategory(SqlDataReader reader)
{
    // Xóa tất cả các dòng hiện tại
    lvCategory.Items.Clear();

    // Đọc một dòng dữ liệu
    while (reader.Read())
    {
        // Tạo một dòng mới trong ListView
        ListViewItem item = new ListViewItem(reader["ID"].ToString());

        // Thêm dòng mới vào ListView
        lvCategory.Items.Add(item);

        // Bổ sung các thông tin khác cho ListViewItem
        item.SubItems.Add(reader["Name"].ToString());
        item.SubItems.Add(reader["Type"].ToString());
    }
}
```

Nhấn F5 để chạy chương trình. Nhấn nút “Lấy danh sách” để xem kết quả.



2. Thêm một mẫu tin dùng lệnh INSERT

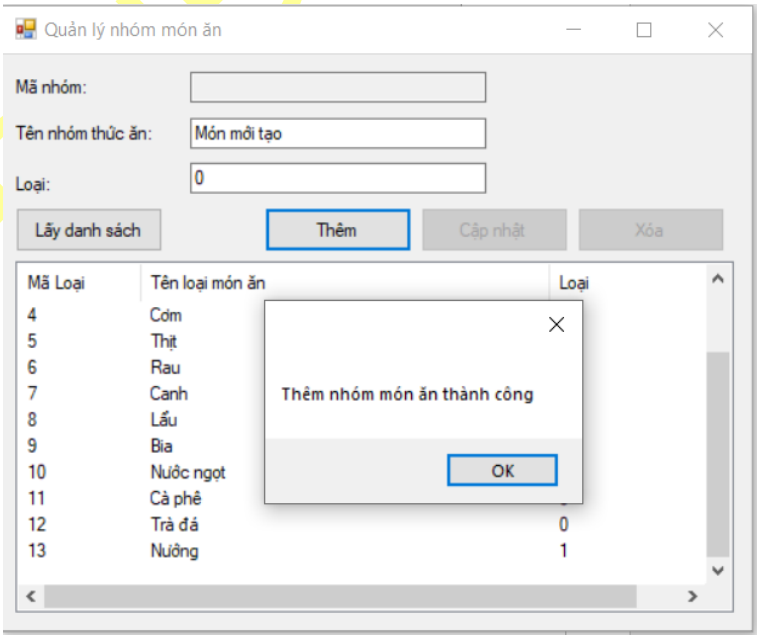
Trong phần này, chúng ta sẽ thay đổi lại giao diện cho ***CategoryForm*** như hình dưới đây. Mục đích của việc thay đổi giao diện này là chúng ta sẽ tạo ra một form hoàn chỉnh hơn, bổ sung chức năng thêm, cập nhật và xóa nhóm món ăn bên cạnh chức năng lấy danh sách đã thực hiện.

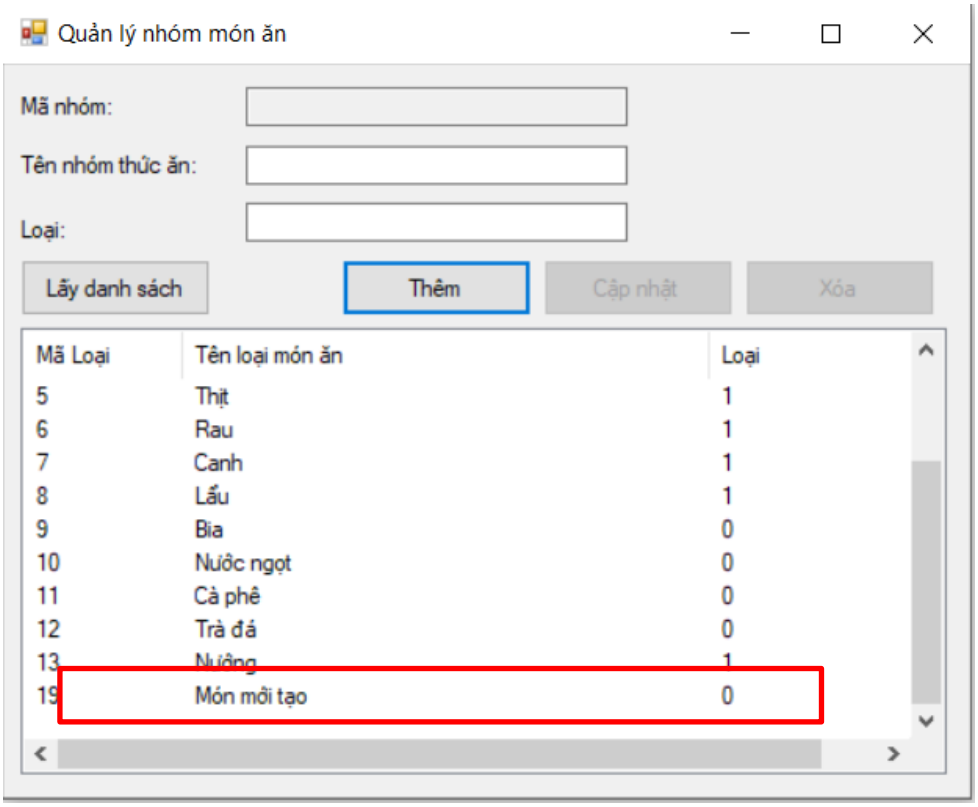


Đổi lại nội dung thanh tiêu đề của form (thuộc tính Text của form) là "***Quản lý nhóm món ăn***". Tiếp tục nhấp đôi chuột vào nút btnAdd (Thêm mới), bổ sung đoạn mã sau vào phương thức *btnAdd_Click*.

```
private void btnAdd_Click(object sender, EventArgs e)
{
    // Tạo đối tượng kết nối
    string connectionString = "server=.; database = RestaurantManagement; Integrated Security = true; ";
    SqlConnection sqlConnection = new SqlConnection(connectionString);

    // Tạo đối tượng thực thi lệnh
    SqlCommand sqlCommand = sqlConnection.CreateCommand();

    // Thiết lập lệnh truy vấn cho đối tượng Command
    sqlCommand.CommandText = "INSERT INTO Category(Name, [Type])" +
        "VALUES (N'" + txtCategoryName.Text + "', " + txtType.Text + ")";

    // Mở kết nối tới cơ sở dữ liệu
    sqlConnection.Open();

    // Thực thi lệnh bằng phương thức ExcuteReader
    int numOfRowsEffected = sqlCommand.ExecuteNonQuery();

    // Đóng kết nối
    sqlConnection.Close();

    if (numOfRowsEffected == 1)
    {
        MessageBox.Show("Thêm nhóm món ăn thành công");

        // Tải lại dữ liệu
        btnLoad.PerformClick();

        // Xóa các ô nhập
        txtCategoryName.Text = "";
        txtType.Text = "";
    }
    else
    {
        MessageBox.Show("Đã có lỗi xảy ra. Vui lòng thử lại");
    }
}
```

Nhấn nút F5 để chạy chương trình. Nhập tên món ăn mới và nhấn nút "***Thêm***". Kết quả sẽ hiển thị như hình dưới đây:





3. Cập nhật một mẫu tin dùng lệnh UPDATE

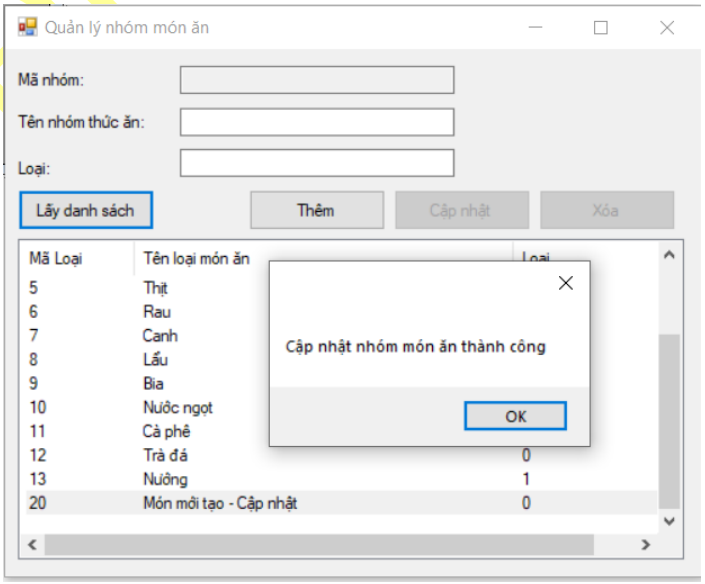
- Nhấp phải chuột vào ListView *lvCategory*, chọn *Properties*.
- Trong khung *Properties*, nhấn chọn nút *Events*. Kích đôi chuột vào sự kiện ***Click***.
- Bổ sung đoạn mã sau vào phương thức lvCategory_Click.

```
private void lvCategory_Click(object sender, EventArgs e)
{
    // Lấy dòng được chọn trong Listview
    ListViewItem item = lvCategory.SelectedItems[0];

    // Hiển thị dữ liệu lên Textbox
    txtCategoryID.Text = item.Text;
    txtCategoryName.Text = item.SubItems[1].Text;
    txtType.Text = item.SubItems[1].Text == "0" ? "Thức uống" : "Đồ ăn";

    // Hiển thị nút cập nhật và xóa
    btnUpdateCat.Enabled = true;
    btnDeleteCate.Enabled = true;
}
```

- Nhấn F5 để chạy chương trình, nhấn nút btnLoad rồi nhấp chuột vào ListView để xem kết quả chạy chương trình.
- Nhấn nút Close để tắt CategoryForm.
- Nhấp đôi chuột vào nút btnUpdate (Cập nhật) và bổ sung đoạn mã sau:

```
private void btnUpdate_Click(object sender, EventArgs e)
{
    // Tạo đối tượng kết nối
    string connectionString = "server=.; database = RestaurantManagement; Integrated Security = true; ";
    SqlConnection sqlConnection = new SqlConnection(connectionString);

    // Tạo đối tượng thực thi lệnh
    SqlCommand sqlCommand = sqlConnection.CreateCommand();

    // Thiết lập lệnh truy vấn cho đối tượng Command
    sqlCommand.CommandText = "UPDATE Category SET Name = N'" + txtCategoryName.Text +
                             "', [Type] = " + txtType.Text +
                             " WHERE ID = " + txtCategoryID.Text;

    // Mở kết nối tới cơ sở dữ liệu
    sqlConnection.Open();

    // Thực thi lệnh bằng phương thức ExcuteReader
    int numOfRowsEffected = sqlCommand.ExecuteNonQuery();

    // Đóng kết nối
    sqlConnection.Close();

    if (numOfRowsEffected == 1)
    {
        // Cập nhật lại dữ liệu trên Listview
        ListViewItem item = lvCategory.SelectedItems[0];

        item.SubItems[1].Text = txtCategoryName.Text;
        item.SubItems[2].Text = txtType.Text;

        // Xóa các ô nhập
        txtCategoryID.Text = "";
        txtCategoryName.Text = "";
        txtType.Text = "";

        // Disable các nút xóa và cập nhật
        btnUpdate.Enabled = false;
        btnDelete.Enabled = false;

        MessageBox.Show("Cập nhật nhóm món ăn thành công");
    }
    else
    {
        MessageBox.Show("Đã có lỗi xảy ra. Vui lòng thử lại");
    }
}
```

- Nhấn phím F5 để chạy và kiểm tra chương trình.



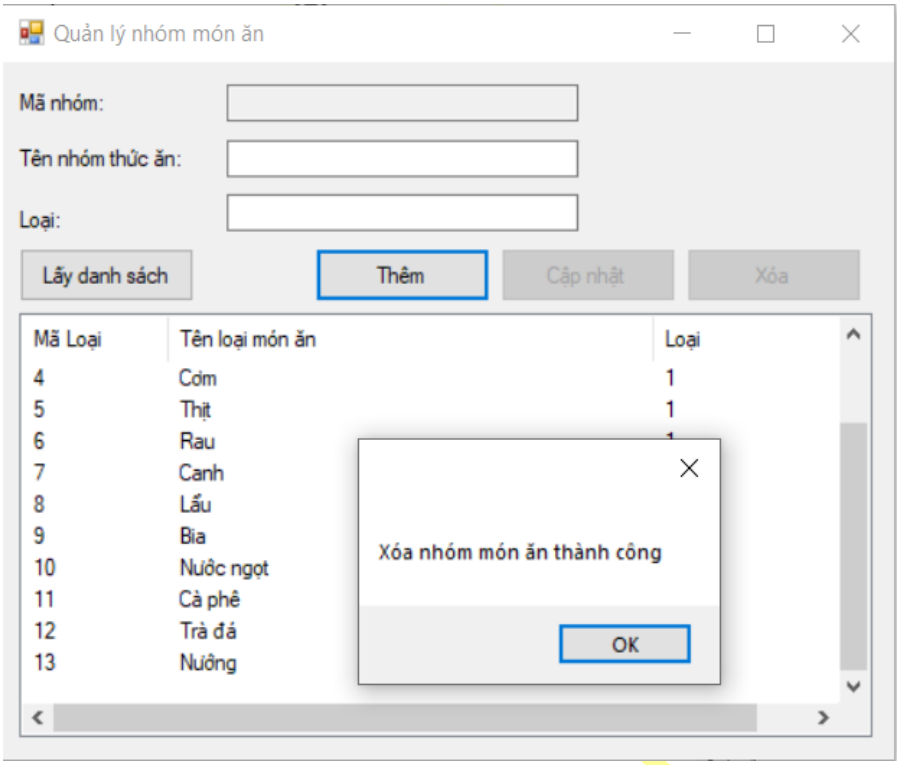
4. Xóa một mẫu tin dùng lệnh DELETE

- Nhấp đôi chuột vào nút btnDelete và bổ sung đoạn mã sau vào phương thức btnDelete_Click.

```
private void btnDelete_Click(object sender, EventArgs e)
{
    // Tạo đối tượng kết nối
    string connectionString = "server=.; database = RestaurantManagement; Integrated Security = true; ";
    SqlConnection sqlConnection = new SqlConnection(connectionString);

    // Tạo đối tượng thực thi lệnh
    SqlCommand sqlCommand = sqlConnection.CreateCommand();

    // Thiết lập lệnh truy vấn cho đối tượng Command
    sqlCommand.CommandText = "DELETE FROM Category " +
        "WHERE ID = " + txtCategoryID.Text;

    // Mở kết nối tới cơ sở dữ liệu
    sqlConnection.Open();

    // Thực thi lệnh bằng phương thức ExcuteReader
    int numOfRowsEffected = sqlCommand.ExecuteNonQuery();

    // Đóng kết nối
    sqlConnection.Close();


    if (numOfRowsEffected == 1)
    {
        // Cập nhật lại dữ liệu trên Listview
        ListViewItem item = lvCategory.SelectedItems[0];
        lvCategory.Items.Remove(item);

        // Xóa các ô nhập
        txtCategoryID.Text = "";
        txtCategoryName.Text = "";
        txtType.Text = "";

        // Disable các nút xóa và cập nhật
        btnUpdate.Enabled = false;
        btnDelete.Enabled = false;

        MessageBox.Show("Xóa nhóm món ăn thành công");
    }
    else
    {
        MessageBox.Show("Đã có lỗi xảy ra. Vui lòng thử lại");
    }
}
```

- Nhấn nút F5 để chạy chương trình.

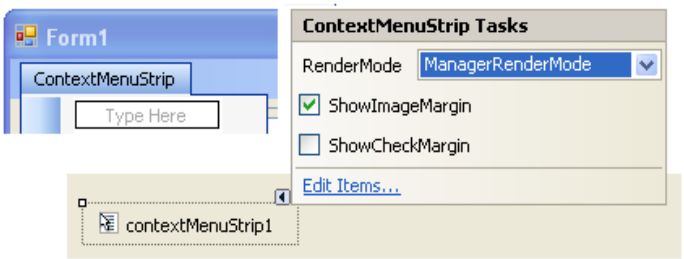
- Nhấn nút btnLoad, nhấp chuột vào ListView, chọn dòng mới được thêm vào ở phần 2 *"Món ăn mới – Cập nhật"*. Sau đó nhấn nút Xóa.



5. Lấy dữ liệu dùng bằng phương thức Fill của DataAdapter

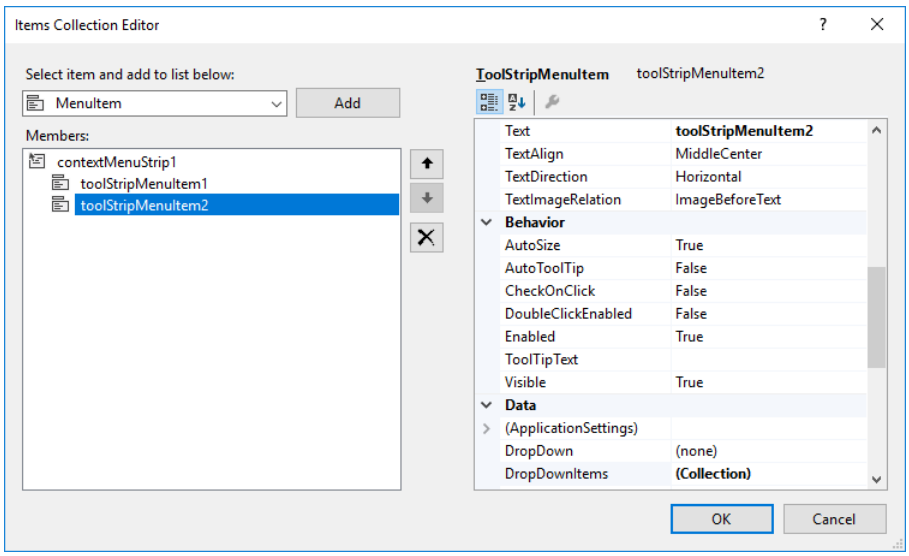
Phần này hướng dẫn thiết kế một menu chuột phải để bổ sung chức năng xóa nhóm món ăn và hiển thị danh sách các món ăn thuộc nhóm.

- Mở CategoryForm, trong nhóm All Windows Form hoặc Menu and Toolbar của thanh Toolbox, chọn Context Menu Strip và kéo nó vào giao diện.

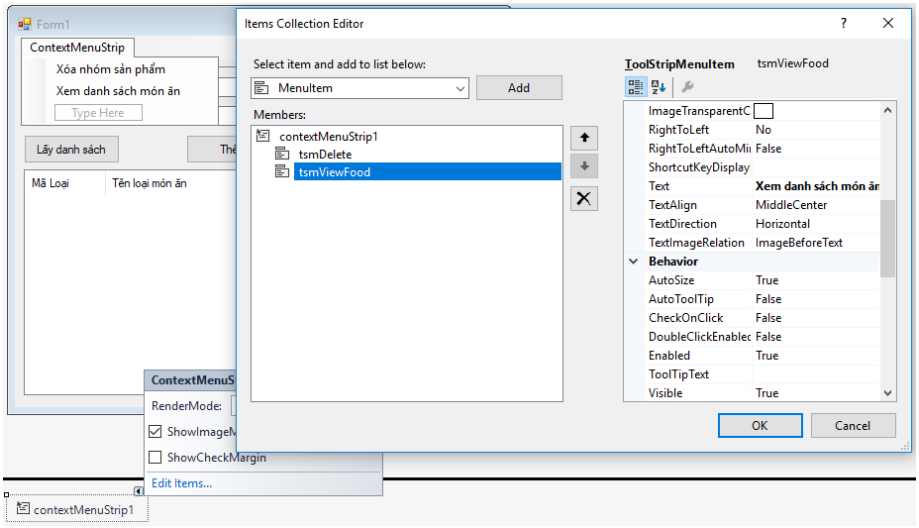


- Phía dưới CategoryForm có một component tên là contextMenuStrip1. Nhấp chuột vào dấu mũi tên hình tam giác, chọn Edit Items...

- Chọn loại Menu Item, Nhấn nút Add để tạo 2 Menu Item như hình sau.



Đổi tên (Name) và tiêu đề (Text) của các MenuItem như hình sau:



- Nhấp đôi chuột vào từng Menu để tạo phương thức xử lý sự kiện Click.

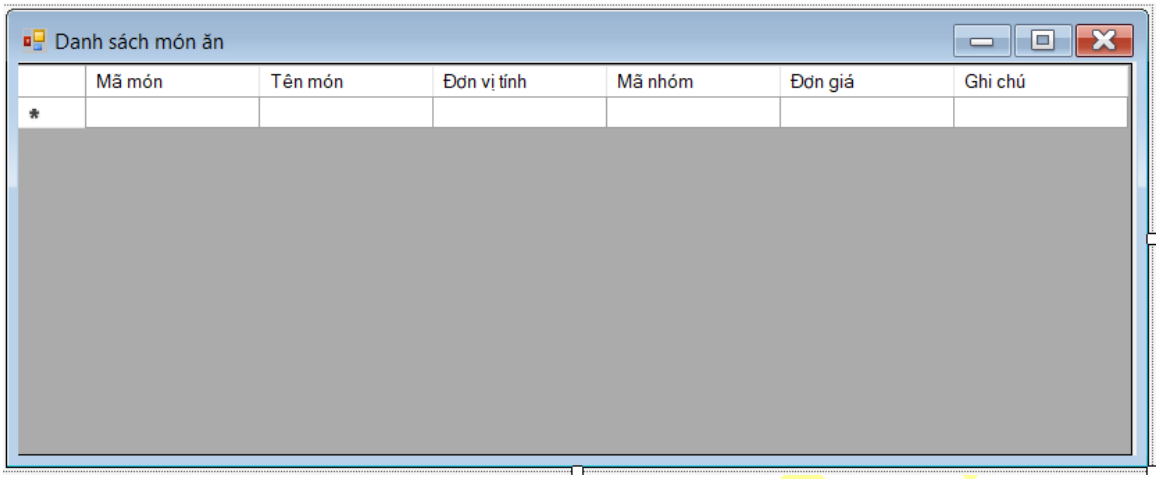
```
private void tsmDelete_Click(object sender, EventArgs e)
{
    if(lvCategory.SelectedItems.Count > 0)
        btnDelete.PerformClick();
}

1 reference
private void tsmViewFood_Click(object sender, EventArgs e)
{

}
```

- Nhấp phải chuột lên ListView, chọn Properties. Trong khung Properties, mục ContextMenuStrip, chọn contextMenuStrip1. Với thao tác này, chúng ta đã đảm bảo khi người dùng nhấp chuột phải vào listview thì menu vừa thiết kế sẽ xuất hiện.

- Để hiển thị danh sách món ăn của một nhóm món ăn cụ thể, ta cần thiết kế một form mới. Tạo một form mới, đặt tên là ***FoodForm*** (Name: FoodForm).

- Trong nhóm Data của thanh Toolbox, chọn DataGridView và kéo lên form mới.

- Đặt tên cho DataGridView là ***dgvFood*** và thiết lập thuộc tính Anchor là Top, Left, Right, Bottom. Thêm các cột cho Datagridview (***Edit Column***) như hình dưới đây. Lưu ý thiết lập thuộc tính ***DataPropertyName*** cho các cột tương ứng với tên các cột trong CSDL.



- Nhấp phải chuột lên Form mới, chọn View Code.

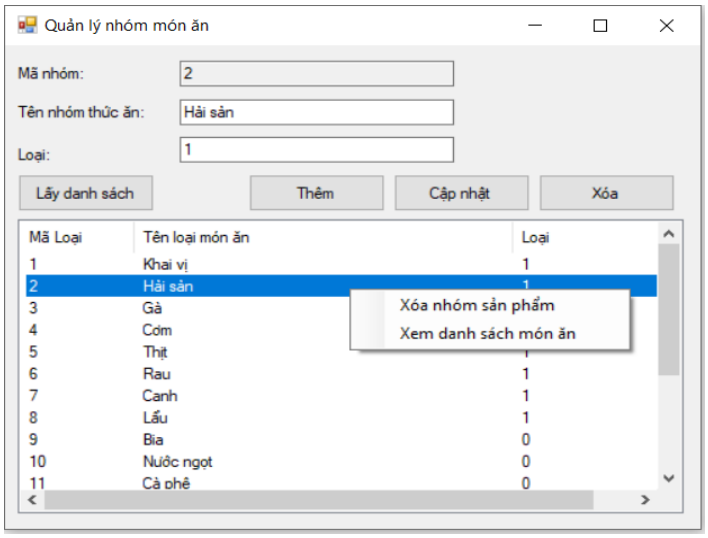
- Tạo một phương thức mới trong lớp FoodForm như sau:

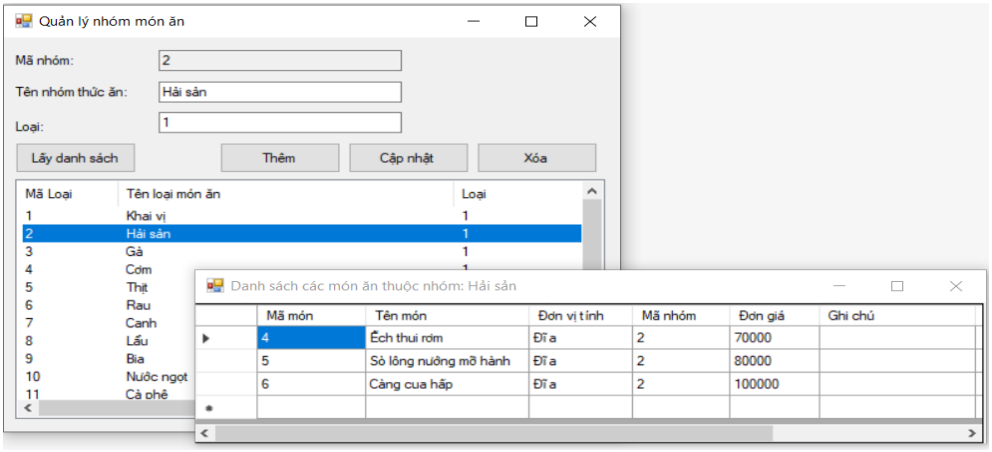
```
public void LoadFood(int categoryID)
{
    // Tạo đối tượng kết nối
    string connectionString = "server=.; database = RestaurantManagement; Integrated Security = true; ";
    SqlConnection sqlConnection = new SqlConnection(connectionString);

    // Tạo đối tượng thực thi lệnh
    SqlCommand sqlCommand = sqlConnection.CreateCommand();

    // Thiết lập lệnh truy vấn cho đối tượng Command
    sqlCommand.CommandText = "SELECT Name FROM Category where ID = " + categoryID;

    // Mở kết nối tới cơ sở dữ liệu
    sqlConnection.Open();

    // Gán tên nhóm sản phẩm cho tiêu đề
    string catName = sqlCommand.ExecuteScalar().ToString();
    this.Text = "Danh sách các món ăn thuộc nhóm: " + catName;

    sqlCommand.CommandText = "SELECT * FROM Food WHERE FoodCategoryID = " + categoryID;

    // Tạo đối tượng DataAdapter
    SqlDataAdapter da = new SqlDataAdapter(sqlCommand);

    // Tạo DataTable để chứa dữ liệu
    DataTable dt = new DataTable("Food");
    da.Fill(dt);

    // Hiển thị danh sách món ăn lên Form
    dgvFood.DataSource = dt;

    // Đóng kết nối và giải phóng bộ nhớ
    sqlConnection.Close();
    sqlConnection.Dispose();
    da.Dispose();
}
```

Trở lại lớp CategoryForm.cs, bổ sung đoạn mã sau vào phương thức tsmViewFood_Click.

```
private void tsmViewFood_Click(object sender, EventArgs e)
{
    if (txtCategoryID.Text != "")
    {
        FoodForm foodForm = new FoodForm();
        foodForm.Show(this);
        foodForm.LoadFood(Convert.ToInt32(txtCategoryID.Text));
    }
}
```

- Nhấn F5 để chạy chương trình. Nhấn nút ***btnLoad***, nhấp chuột phải lên một nhóm sản phẩm rồi chọn ***Xem danh sách món ăn***.





C. Bài tập

1. Bổ sung 2 nút vào FoodForm với chức năng cụ thể như sau:

- Nút Save: cho phép người dùng thêm hoặc sửa thông tin trong dgvFood và lưu thông tin cập nhật hoặc thêm mới vào bảng Food.

- Nút Delete: cho phép xóa dòng được chọn trên dgvFood.

2. Thiết kế Form: BillsForm và viết phương thức xử lý để:

- Hiển thị danh sách hóa đơn được bán trong một khoảng thời gian nào đó (yêu cầu có ô chọn từ ngày đến ngày – sử dụng control DateTimePicker), có hiển thị tổng số tiền chưa giảm giá, tổng số tiền giảm giá, thực thu.

- Khi nhấp đôi chuột vào một hóa đơn nào đó thì mở một Form mới (BillDetailsForm) để hiển thị danh mục các mặt hàng mua bởi hóa đơn đó.

3. Thiết kế Form AccountManager và viết phương thức xử lý để:
   - Xem danh sách tài khoản theo nhóm, theo trạng thái (Active?).
   - Thêm một tài khoản mới vào cơ sở dữ liệu.
   - Cập nhật thông tin của một tài khoản.
   - Reset mật khẩu cho tài khoản.
   - Nhấp chuột phải vào một tài khoản hiển thị menu sau:

Xóa tài khoản
Xem danh sách vai trò

   Trong đó:

   - Nếu chọn Xóa tài khoản thì toàn bộ vai trò của tài khoản này sẽ bị đánh dấu là không kích hoạt (0).
   - Xem danh sách vai trò: Mở một Form mới để hiển thị các vai trò được gán cho tài khoản này.

4. Thiết kế Form: MainForm và viết các phương thức xử lý để
   - Hiển thị danh sách các bàn.
   - Xem hóa đơn hiện tại của một bàn.
   - Thêm một bàn mới.
   - Cập nhật thông tin của bàn.
   - Xóa một bàn.
   - Khi nhấp phải chuột vào một bàn, hiển thị menu sau:

Xóa bàn
Xem danh mục hóa đơn
Xem nhật ký hóa đơn

   Trong đó:

   - Nếu chọn Xóa bàn thì dữ liệu về bàn đó sẽ bị xóa khỏi cơ sở dữ liệu.
   - Xem danh mục hóa đơn: Mở một Form mới, phần bên trái của Form là một ListBox chứa ngày lập của các hóa đơn. Khi nhấn chuột vào ngày nào thì hiển thị thông tin chi tiết (cả danh mục sản phẩm được mua) của hóa đơn ở phần bên phải Form.

- Xem nhật ký mua hàng: Liệt kê số lượng hóa đơn, tổng số tiền, tổng thuế, tổng giảm giá của tất cả các hóa đơn, thông tin liên quan đến từng hóa đơn như ngày lập, tên nhân viên lập hóa đơn. (sử dụng ListView hoặc DataGridView).

CHỦ ĐỀ 5. TRUYỀN THAM SỐ VÀ THỰC THI THỦ TỤC

A. Mục tiêu

Chủ đề 5 giúp sinh viên tìm hiểu:

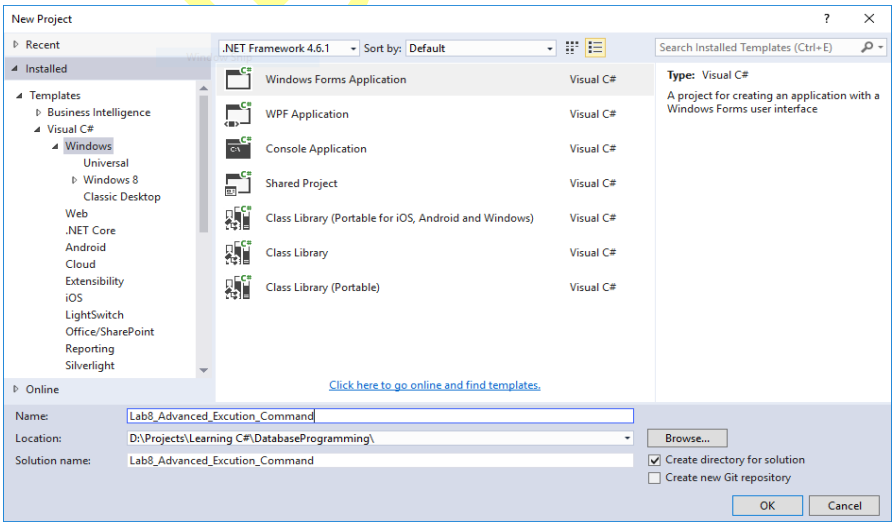
- Cách truyền tham số vào một lệnh truy vấn SQL
- Cách thực thi các lệnh SELECT, INSERT, UPDATE, DELETE có sử dụng các tham số.
- Gọi và thực thi các Stored Procedure

Sau hoàn thành chủ đề, sinh viên cần nắm rõ những vấn đề sau:

- Sử dụng đối tượng Parameter để truyền tham số vào các lệnh.
- Cách thực thi và nhận kết quả trả về từ các lệnh truy vấn có tham số
- Cách thực thi các thủ tục và nhận dữ liệu trả về.
- Cách xây dựng ứng dụng trên nền Windows Form.

B. Hướng dẫn thực hành

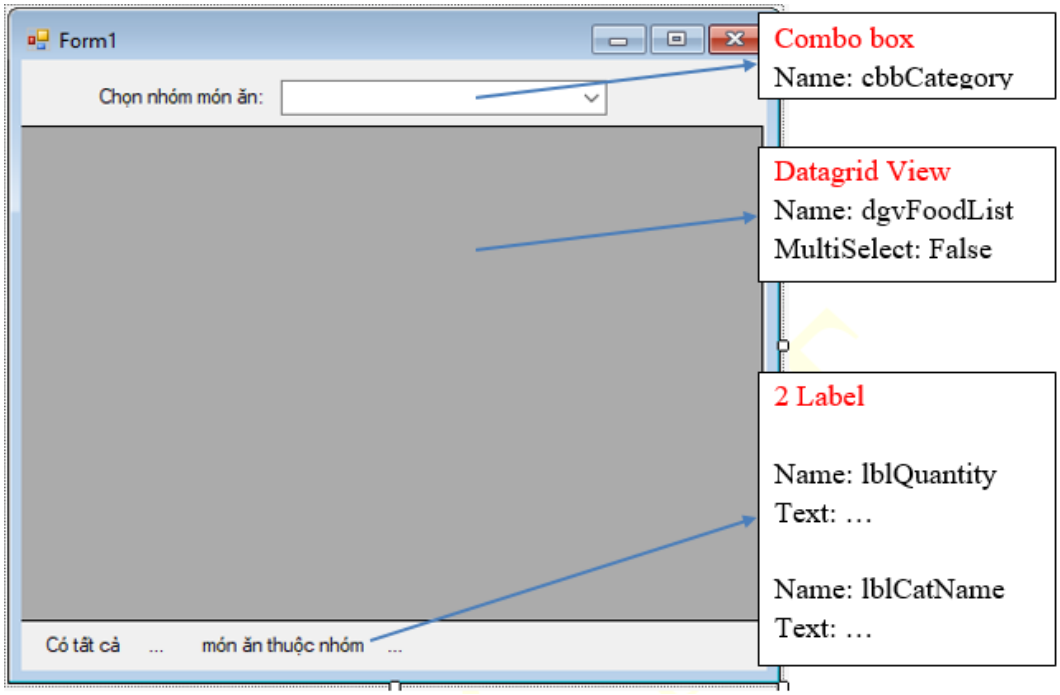
Tạo một dự án Windows Application mới, đặt tên là ***Lab_Advanced_Command.***



Phần hướng dẫn tiếp theo trong chủ đề này sẽ hướng dẫn các bạn xây dựng một form quản lý danh sách các món ăn. Cụ thể, danh sách các món ăn sẽ được hiển thị theo từng nhóm (danh mục). Người dùng chọn nhóm nào, danh sách các món ăn thuộc nhóm sẽ được hiển thị. Bên cạnh đó, người dùng có thể thực hiện các thao tác thêm, cập nhật

các món ăn trong danh sách cũng như hiển thị tổng số lượng đã bán của từng món ăn bằng cách sử dụng menu chuột phải.

Đầu tiên, đổi tên ***Form1.cs*** thành ***FoodForm.cs*** và thiết kế form có dạng như hình sau:

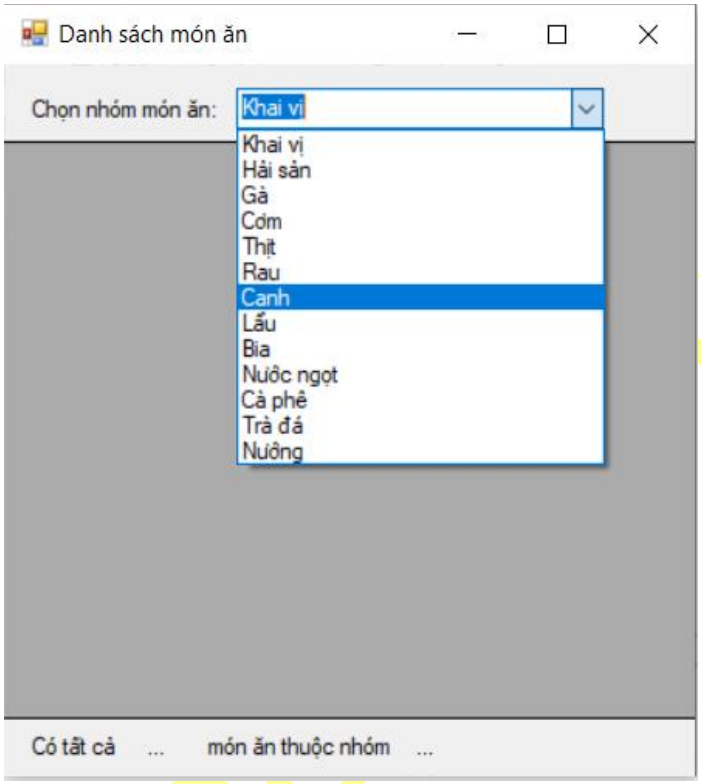


Thay đổi tiêu đề của form thành "***Danh sách món ăn***". Sau đó, nhấp đôi chuột lên ***FoodForm*** để tạo phương thức xử lý sự kiện ***FoodForm_Load***. Tạo phương thức ***LoadCategory*** để tải danh sách nhóm sản phẩm lên ComboBox và gọi phương thức này trong phương thức ***FoodForm_Load*** như hình dưới đây. Bước này đảm bảo khi form được mở thì danh sách nhóm món ăn sẽ được đọc từ cơ sở dữ liệu đổ vào combobox.

```
private void LoadCategory()
{
    string connectionString = "server=.; database = RestaurantManagement; Integrated Security = true; ";
    SqlConnection conn = new SqlConnection(connectionString);

    SqlCommand cmd = conn.CreateCommand();
    cmd.CommandText = "SELECT ID, Name FROM Category";

    SqlDataAdapter adapter = new SqlDataAdapter(cmd);
    DataTable dt = new DataTable();

    // Mở kết nối
    conn.Open();

    // Lấy dữ liệu từ csdl đưa vào DataTable
    adapter.Fill(dt);

    // Đóng kết nối và giải phóng bộ nhớ
    conn.Close();
    conn.Dispose();

    // Đưa dữ liệu vào Combo Box
    cbbCategory.DataSource = dt;

    // Hiển thị tên nhóm sản phẩm
    cbbCategory.DisplayMember = "Name";

    // Nhưng khi lấy giá trị thì lấy ID của nhóm
    cbbCategory.ValueMember = "ID";
}
```

```
private void FoodForm_Load(object sender, EventArgs e)
{
    LoadCategory();
}
```

Tiếp theo, nhấn F5 để chạy chương trình và kết quả sẽ hiển thị như hình dưới đây. Như vậy danh sách nhóm món ăn đã được tải vào combobox.



1. Truyền tham số vào đối tượng Command

Trong phần này, chúng ta sẽ thực hiện xử lý sự kiện để hiển thị danh sách các món ăn thuộc nhóm mà người dùng chọn trong combobox. Như vậy, sự kiện cần xử lý là sự kiện SelectedIndexChanged của combobox.

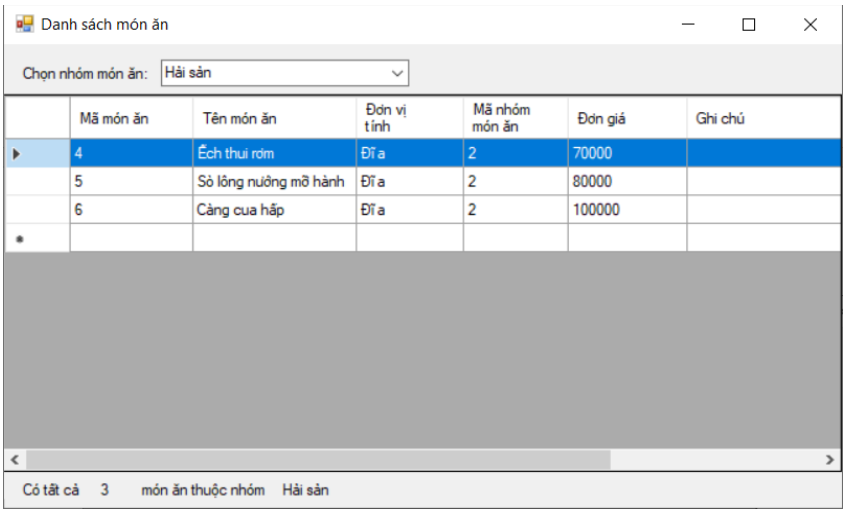
Nhấp chuột phải vào ComboBox, chọn Properties. Trong khung Properties, nhấn nút Events, nhấp đôi chuột vào SelectedIndexChanged. Khai báo biến cục bộ foodTable như sau:

```
public partial class Form1 : Form
{
    private DataTable foodTable;

    1 reference
    public Form1()
    {
        InitializeComponent();
    }
```

Bổ sung đoạn mã sau vào phương thức cbbCategories_SelectedIndexChanged.

```
private void cbbCategory_SelectedIndexChanged(object sender, EventArgs e)
{
    if (cbbCategory.SelectedIndex == -1) return;

    string connectionString = "server=.; database = RestaurantManagement; Integrated Security = true; ";
    SqlConnection conn = new SqlConnection(connectionString);

    SqlCommand cmd = conn.CreateCommand();
    cmd.CommandText = "SELECT * FROM Food WHERE FoodCategoryID = @categoryId";

    // Truyền tham số
    cmd.Parameters.Add("@categoryId", SqlDbType.Int);

    if (cbbCategory.SelectedValue is DataRowView)
    {
        DataRowView rowView = cbbCategory.SelectedValue as DataRowView;
        cmd.Parameters["@categoryId"].Value = rowView["ID"];
    }
    else
    {
        cmd.Parameters["@categoryId"].Value = cbbCategory.SelectedValue;
    }
    // Tạo bộ điều phiếu dữ liệu
    SqlDataAdapter adapter = new SqlDataAdapter(cmd);
    foodTable = new DataTable();

    // Mở kết nối
    conn.Open();

    // Lấy dữ liệu từ csdl đưa vào DataTable
    adapter.Fill(foodTable);

    // Đóng kết nối và giải phóng bộ nhớ
    conn.Close();
    conn.Dispose();

    // Đưa dữ liệu vào data gridview
    dgvFoodList.DataSource = foodTable;

    // Tính số lượng mẫu tin
    lblQuantity.Text = foodTable.Rows.Count.ToString();
    lblCatName.Text = cbbCategory.Text;
}
```

Nhấn phím F5 để chạy chương trình. Nhấp chuột vào ComboBox, chọn nhóm món ăn để xem danh sách món ăn thuộc nhóm đó.

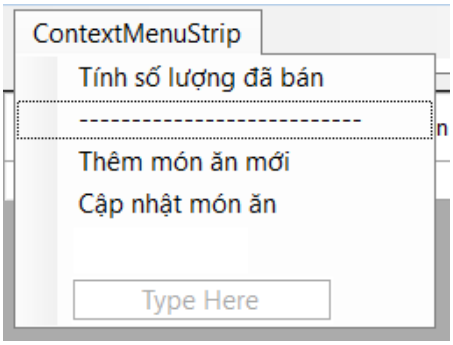


2. Nhận giá trị trả về từ tham số

Phần này, chúng ta sẽ thiết kế menu chuột phải của Listview danh sách món ăn và xử lý chức năng tính số lượng đã bán của một món ăn bất kỳ.

Từ thanh Toolbox, kéo một đối tượng ContextmenuStrip vào ***FoodForm***, đặt tên cho contextmenu là ***ctmFoodList***. Thiết kế menu có dạng như hình dưới đây (tên của

các menu lần lượt là *tsmCalculateQuantity*, *tsmSeperator*, *tsmAddFood*, *tsmUpdateFood*).

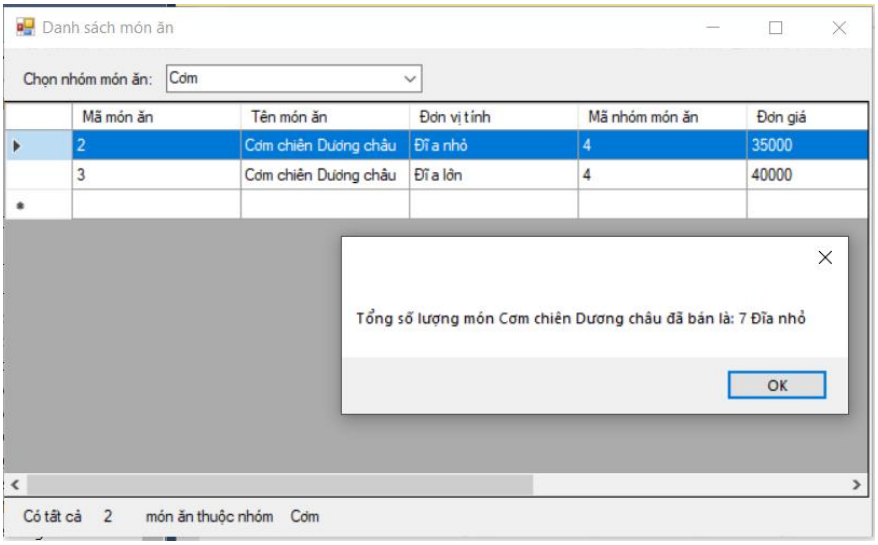


Nhấp chuột phải vào DataGridView, chọn Properties, thiết lập thuộc tính ContextMenuStrip của DataGridView là ***ctmFoodList***.

Nhấp đôi chuột lên menu "*Tính số lượng đã bán*" để tạo phương thức xử lý sự kiện Click cho menu. Bổ sung đoạn mã sau:

```
private void tsmCalculateQuantity_Click(object sender, EventArgs e)
{
    string connectionString = "server=.; database = RestaurantManagement; Integrated Security = true; ";
    SqlConnection conn = new SqlConnection(connectionString);

    SqlCommand cmd = conn.CreateCommand();
    cmd.CommandText = "SELECT @numSaleFood = sum(Quantity) FROM BillDetails WHERE FoodID = @foodId";

    // Lấy thông tin sản phẩm được chọn
    if (dgvFoodList.SelectedRows.Count > 0)
    {
        DataGridViewRow selectedRow = dgvFoodList.SelectedRows[0];

        DataRowView rowView = selectedRow.DataBoundItem as DataRowView;

        // Truyền tham số
        cmd.Parameters.Add("@foodId", SqlDbType.Int);
        cmd.Parameters["@foodId"].Value = rowView["ID"];

        cmd.Parameters.Add("@numSaleFood", SqlDbType.Int);
        cmd.Parameters["@numSaleFood"].Direction = ParameterDirection.Output;


        // Mở kết nối csdl
        conn.Open();

        // Thực thi truy vấn và lấy dữ liệu từ tham số
        cmd.ExecuteNonQuery();

        string result = cmd.Parameters["@numSaleFood"].Value.ToString();
        MessageBox.Show("Tổng số lượng món " + rowView["Name"] + " đã bán là: " + result + " " + rowView["Unit"]);

        // Đóng kết nối csdl
        conn.Close();
    }
    cmd.Dispose();
    conn.Dispose();
}
```

Nhấn F5 chạy chương trình, chọn 1 sản phẩm và nhấp chuột phải và chọn "*Tính số lượng đã bán*" và kiểm tra kết quả.



Trở lại FoodForm, nhấp đôi chuột vào 2 menu còn lại trong ***ctmFoodList*** để tạo phương thức xử lý sự kiện Click của chúng như hình dưới.

```
private void tsmAddFood_Click(object sender, EventArgs e)
{

}

1 reference
private void tsmUpdateFood_Click(object sender, EventArgs e)
{

}
```

3. Thực thi lệnh bằng cách Sử dụng Stored Procedure

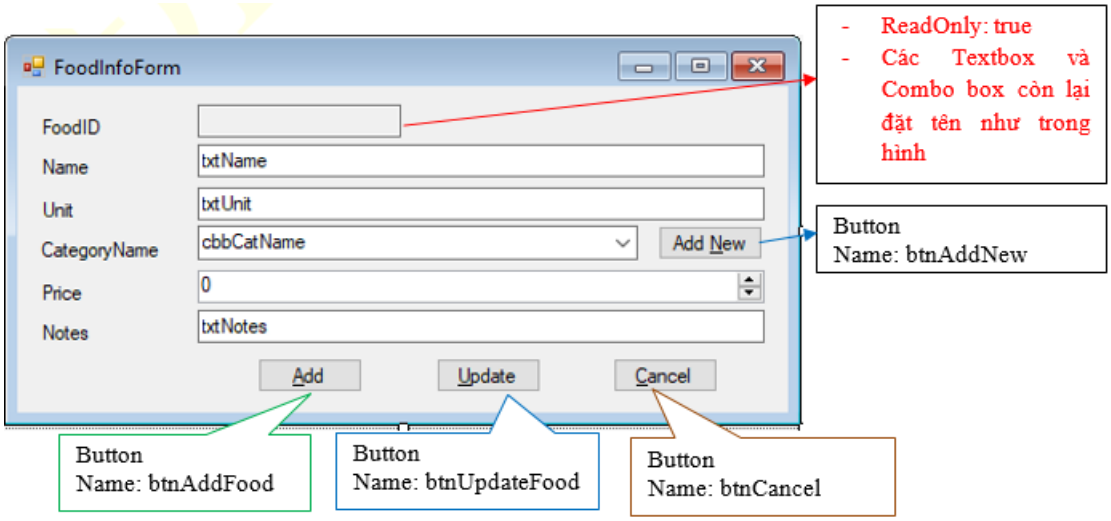
Trong phần này, chúng ta sẽ thực hiện xử lý 2 chức năng là thêm và cập nhật thông tin món ăn khi nhấp chuột chọn chức năng "*Thêm món ăn mới*" và "*Cập nhật món ăn*" trên menu chuột phải. Để thêm hoặc cập nhật món ăn ta cần tạo thêm một form mới để ghi nhận hoặc hiển thị thông tin chi tiết của món ăn. Trước hết, ta sẽ tạo Stored Procedure sau để thực hiện thêm một món ăn mới và cập nhật thông tin món ăn.

```
CREATE PROCEDURE [InsertFood]
    @ID int output,
    @Name nvarchar(1000),
    @Unit nvarchar(100),
    @FoodCategoryID int,
    @Price int,
    @Notes nvarchar(3000)
AS
INSERT INTO [Food]
([Name],[Unit],[FoodCategoryID],[Price],[Notes])
VALUES (@Name, @Unit, @FoodCategoryID, @Price,@Notes)

SELECT @ID = SCOPE_IDENTITY();
GO
```

```
CREATE PROCEDURE [UpdateFood]
    @ID int,
    @Name nvarchar(1000),
    @Unit nvarchar(100),
    @FoodCategoryID int,
    @Price int,
    @Notes nvarchar(3000)
AS
UPDATE [Food]
SET
   [Name] = @Name,
   [Unit]=@Unit,
   [FoodCategoryID]= @FoodCategoryID,
   [Price]=@Price,
   [Notes]=@Notes

WHERE ID = @ID

IF @@ERROR <> 0
RETURN 0
ELSE
RETURN 1
GO
```

Trong phần này, ta sẽ sử dụng 2 cách khác nhau để gọi một thủ tục trong SQL Server và học cách bắt lỗi SQL (hay ngoại lệ - Exception) bằng C#.

Thêm một Form mới, đặt tên là ***FoodInfoForm***. Thiết kế giao diện cho Form mới như hình sau:



Nhấp đôi chuột vào FoodInfoForm để xử lý phương thức Load. Thêm các phương thức sau:

```
private void FoodInfoForm_Load(object sender, EventArgs e)
{
    this.InitValues();
}

private void InitValues()
{
    string connectionString = "server=.; database = RestaurantManagement; Integrated Security = true; ";
    SqlConnection conn = new SqlConnection(connectionString);

    SqlCommand cmd = conn.CreateCommand();
    cmd.CommandText = "SELECT ID, Name FROM Category";

    SqlDataAdapter adapter = new SqlDataAdapter(cmd);
    DataSet ds = new DataSet();

    // mở kết nối
    conn.Open();

    // Lấy dữ liệu từ csdl đưa vào DataTable
    adapter.Fill(ds, "Category");

    // Hiển thị nhóm món ăn
    cbbCatName.DataSource = ds.Tables["Category"];
    cbbCatName.DisplayMember = "Name";
    cbbCatName.ValueMember = "ID";

    // đóng kết nối và giải phóng bộ nhớ
    conn.Close();
    conn.Dispose();
}
```

Bổ sung phương thức sau để xóa dữ liệu trên các control của Form

```
private void ResetText()
{
    txtFoodID.ResetText();
    txtName.ResetText();
    txtNotes.ResetText();
    txtUnit.ResetText();
    cbbCatName.ResetText();
    nudPrice.ResetText();
}
```

Nhấp đôi chuột vào nút btnAddFood để xử lý phương thức thêm một món ăn mới.

```
private void btnAddFood_Click(object sender, EventArgs e)
{
    try
    {
        string connectionString = "server=.; database = RestaurantManagement; Integrated Security = true; ";
        SqlConnection conn = new SqlConnection(connectionString);

        SqlCommand cmd = conn.CreateCommand();
        cmd.CommandText = "EXECUTE InsertFood @id OUTPUT, @name, @unit, @foodCategoryID, @price, @notes";

        // Thêm tham số vào đối tượng Command
        cmd.Parameters.Add("@id", SqlDbType.Int);
        cmd.Parameters.Add("@name", SqlDbType.NVarChar, 1000);
        cmd.Parameters.Add("@unit", SqlDbType.NVarChar, 100);
        cmd.Parameters.Add("@foodCategoryId", SqlDbType.Int);
        cmd.Parameters.Add("@price", SqlDbType.Int);
        cmd.Parameters.Add("@notes", SqlDbType.NVarChar,3000);

        cmd.Parameters["@id"].Direction = ParameterDirection.Output;

        // Truyền giá trị vào thủ tục qua tham số
        cmd.Parameters["@name"].Value = txtName.Text;
        cmd.Parameters["@unit"].Value = txtUnit.Text;
        cmd.Parameters["@foodCategoryId"].Value = cbbCatName.SelectedValue;
        cmd.Parameters["@price"].Value = nudPrice.Value;
        cmd.Parameters["@notes"].Value = txtNotes.Text;

        //mở kết nối
        conn.Open();

        int numRowAffected = cmd.ExecuteNonQuery();


        // Thông báo kết quả
        if (numRowAffected > 0)
        {
            string foodID = cmd.Parameters["@id"].Value.ToString();

            MessageBox.Show("Successfully adding new food. Food ID = " + foodID, "Message");

            this.ResetText();
        }
        else
        {
            MessageBox.Show("Adding food failed");
        }

        // đóng kết nối
        conn.Close();
        conn.Dispose();
    }
    // Bắt lỗi SQL và các lỗi khác
    catch (SqlException exception)
    {
        MessageBox.Show(exception.Message, "SQL Error");
    }

    catch (Exception exception)
    {
        MessageBox.Show(exception.Message, "Error");
    }
}
```

Bổ sung thêm phương thức sau vào lớp FoodInfoForm.

```
public void DisplayFoodInfo(DataRowView rowView)
{
    try
    {
        txtFoodID.Text = rowView["ID"].ToString();
        txtName.Text = rowView["Name"].ToString();
        txtUnit.Text = rowView["Unit"].ToString();
        txtNotes.Text = rowView["Notes"].ToString();
        nudPrice.Text = rowView["Price"].ToString();

        cbbCatName.SelectedIndex = -1;

        // chọn nhóm món ăn tương ứng
        for (int index = 0; index < cbbCatName.Items.Count; index++)
        {
            DataRowView cat = cbbCatName.Items[index] as DataRowView;
            if (cat["ID"].ToString() == rowView["FoodCategoryID"].ToString())
            {
                cbbCatName.SelectedIndex = index;
                break;
            }
        }
    }
    catch (Exception exception)
    {
        MessageBox.Show(exception.Message, "Error");
        this.Close();
    }
}
```

Nhấp đôi chuột vào nút btnUpdateFood để xử lý phương thức cập nhật thông tin món ăn.

```
private void btnUpdateFood_Click(object sender, EventArgs e)
{
    try
    {
        string connectionString = "server=.; database = RestaurantManagement; Integrated Security = true;";
        SqlConnection conn = new SqlConnection(connectionString);

        SqlCommand cmd = conn.CreateCommand();
        cmd.CommandText = "EXECUTE UpdateFood @id, @name, @unit, @foodCategoryID, @price, @notes";

        // Thêm tham số vào đối tượng Command
        cmd.Parameters.Add("@id", SqlDbType.Int);
        cmd.Parameters.Add("@name", SqlDbType.NVarChar, 1000);
        cmd.Parameters.Add("@unit", SqlDbType.NVarChar, 100);
        cmd.Parameters.Add("@foodCategoryId", SqlDbType.Int);
        cmd.Parameters.Add("@price", SqlDbType.Int);
        cmd.Parameters.Add("@notes", SqlDbType.NVarChar, 3000);

        // Truyền giá trị vào thủ tục qua tham số
        cmd.Parameters["@id"].Value = int.Parse(txtFoodID.Text);
        cmd.Parameters["@name"].Value = txtName.Text;
        cmd.Parameters["@unit"].Value = txtUnit.Text;
        cmd.Parameters["@foodCategoryId"].Value = cbbCatName.SelectedValue;
        cmd.Parameters["@price"].Value = nudPrice.Value;
        cmd.Parameters["@notes"].Value = txtNotes.Text;

        //mở kết nối
        conn.Open();

        int numRowAffected = cmd.ExecuteNonQuery();
```

```
            // Thông báo kết quả
            if (numRowAffected > 0)
            {

                MessageBox.Show("Successfully updating food", "Message");

                this.ResetText();
            }
            else
            {
                MessageBox.Show("Updating food failed");
            }

            // đóng kết nối
            conn.Close();
            conn.Dispose();
        }
        // Bắt lỗi SQL và các lỗi khác
        catch (SqlException exception)
        {
            MessageBox.Show(exception.Message, "SQL Error");
        }

        catch (Exception exception)
        {
            MessageBox.Show(exception.Message, "Error");
        }
    }
```

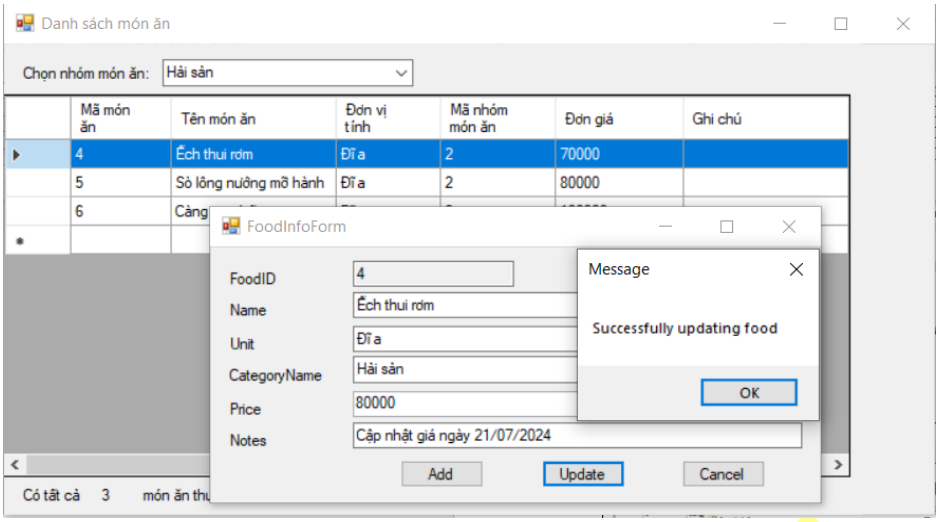
Nhấp đôi chuột vào nút btnCancel để xử lý việc thoát khỏi Form.

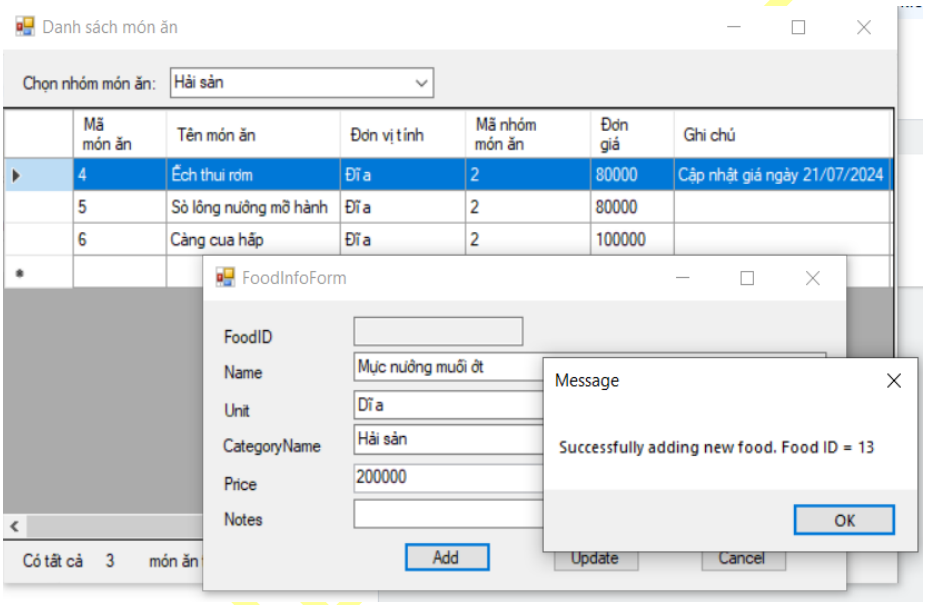
```
private void btnCancel_Click(object sender, EventArgs e)
{
    this.Close();
}
```

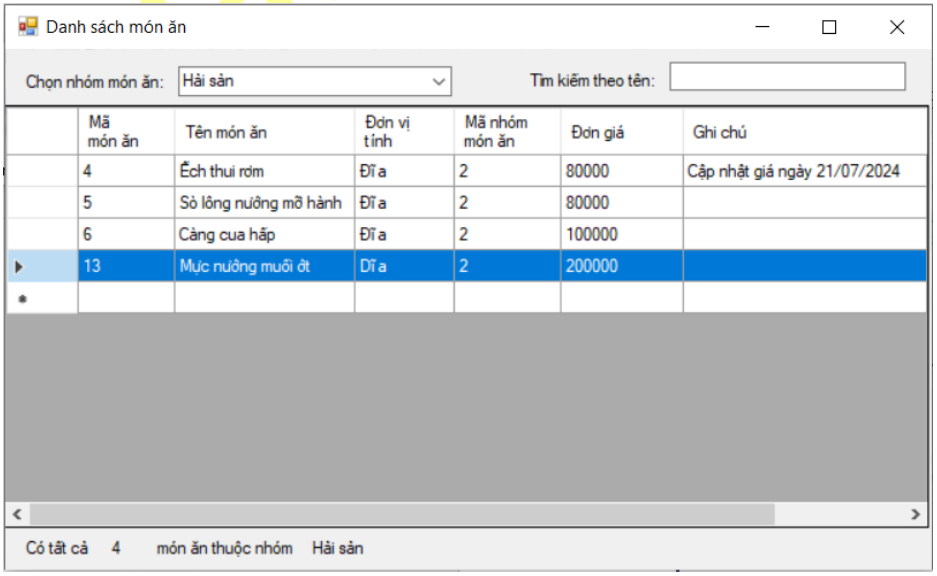
Trở lại FoodForm, bổ sung đoạn mã sau vào phương thức xử lý sự kiện Click của 2 menu Thêm mới và Cập nhật thông tin món ăn.

```
private void tsmAddFood_Click(object sender, EventArgs e)
{
    FoodInfoForm foodForm = new FoodInfoForm();
    foodForm.FormClosed += new FormClosedEventHandler(foodForm_FormClosed);

    foodForm.Show(this);
}

2 references
void foodForm_FormClosed(object sender, FormClosedEventArgs e)
{
    int index = cbbCategory.SelectedIndex;
    cbbCategory.SelectedIndex = -1;
    cbbCategory.SelectedIndex = index;
}


private void tsmUpdateFood_Click(object sender, EventArgs e)
{
    // Lấy thông tin sản phẩm được chọn
    if (dgvFoodList.SelectedRows.Count > 0)
    {
        DataGridViewRow selectedRow = dgvFoodList.SelectedRows[0];
        DataRowView rowView = selectedRow.DataBoundItem as DataRowView;

        FoodInfoForm foodForm = new FoodInfoForm();
        foodForm.FormClosed += new FormClosedEventHandler(foodForm_FormClosed);

        foodForm.Show(this);
        foodForm.DisplayFoodInfo(rowView);
    }
}
```

Chạy chương trình và kiểm tra kết quả:

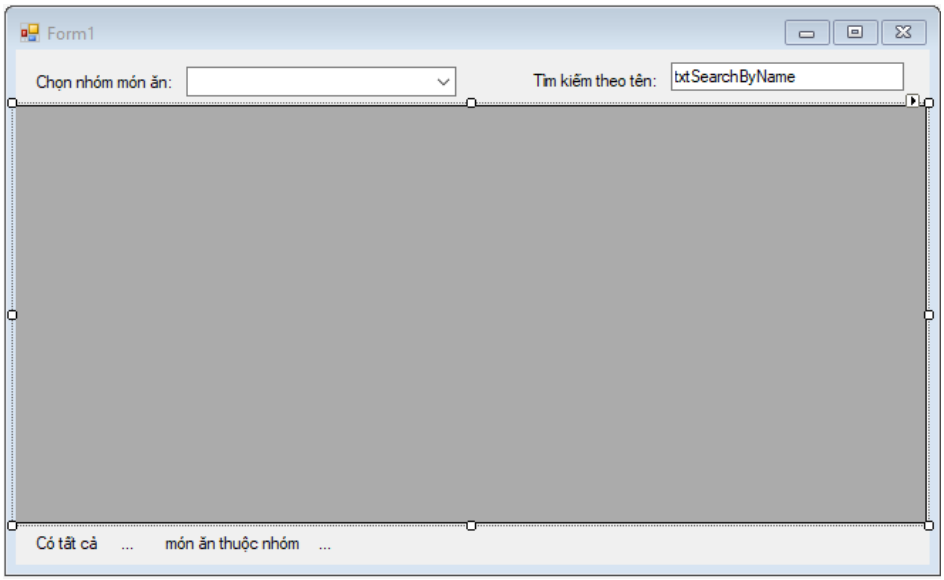

Danh sách món ăn
Chọn nhóm món ăn:
Hải sản
Mã món ăn
Tên món ăn
Đơn vị tính
Mã nhóm món ăn
Đơn giá
Ghi chú
4
Ếch thui rơm
Đĩa
2
70000
5
Sò lông nướng mỡ hành
Đĩa
2
80000
6
Càng
FoodInfoForm
FoodID
4
Name
Ếch thui rơm
Unit
Đĩa
CategoryName
Hải sản
Price
80000
Notes
Cập nhật giá ngày 21/07/2024
Add
Update
Cancel
Message
Successfully updating food
OK
Có tất cả
3
món ăn thu


Danh sách món ăn
Chọn nhóm món ăn:
Hải sản
Mã món ăn
Tên món ăn
Đơn vị tính
Mã nhóm món ăn
Đơn giá
Ghi chú
4
Ếch thui rơm
Đĩa
2
80000
Cập nhật giá ngày 21/07/2024
5
Sò lông nướng mỡ hành
Đĩa
2
80000
6
Càng cua hấp
Đĩa
2
100000
FoodInfoForm
FoodID
Name
Mực nướng muối ớt
Unit
Dĩa
CategoryName
Hải sản
Price
200000
Notes
Add
Update
Cancel
Message
Successfully adding new food. Food ID = 13
OK
Có tất cả
3
món ăn


Danh sách món ăn
Chọn nhóm món ăn:
Hải sản
Tìm kiếm theo tên:
Mã món ăn
Tên món ăn
Đơn vị tính
Mã nhóm món ăn
Đơn giá
Ghi chú
4
Ếch thui rơm
Đĩa
2
80000
Cập nhật giá ngày 21/07/2024
5
Sò lông nướng mỡ hành
Đĩa
2
80000
6
Càng cua hấp
Đĩa
2
100000
13
Mực nướng muối ớt
Dĩa
2
200000
Có tất cả
4
món ăn thuộc nhóm
Hải sản

4. Sử dụng DataView để lọc dữ liệu

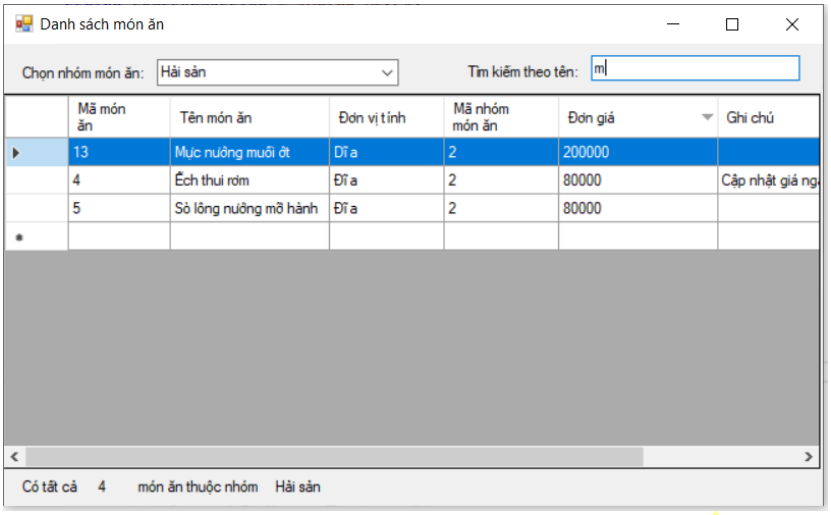
Trong phần này, ta sẽ thực hiện bổ sung phần tìm kiếm theo tên món ăn vào FoodForm như hình dưới đây. Các bạn lưu ý đặt tên cho textbox theo tên được ghi chú sẵn trong hình. Chức năng chúng ta cần xử lý là chức năng cho phép người dùng nhập một từ khóa vào ô tìm kiếm, hệ thống sẽ lọc ra danh sách những món ăn có tên chứa từ khóa được nhập và hiển thị kết quả.



Như vậy, bên cạnh việc hiển thị danh sách món ăn theo nhóm món ăn, người dùng có thể lọc món ăn dựa vào tên bằng cách sử dụng Dataview để sắp xếp và trích lọc. Bổ sung sự kiện TextChange cho ***txtSearchByName*** như sau:

```
private void txtSearchByName_TextChanged(object sender, EventArgs e)
{

    if (foodTable == null) return;

    // create filter and sort expression
    string filterExpression = "Name like '%" + txtSearchByName.Text + "%'";
    string sortExpression = "Price DESC";
    DataViewRowState rowStateFilter = DataViewRowState.OriginalRows;

    // Create a data view object to view the data in foodTable data table
    // filter by Name (contain 'ng') and sort descending by Price
    DataView foodView = new DataView(foodTable,
        filterExpression, sortExpression, rowStateFilter);

    // Assign foodTable as Data Source of data grid view
    dgvFoodList.DataSource = foodView;

}
```

Chạy chương trình, và kiểm tra kết quả:



C. Bài tập

Trong các bài tập sau, yêu cầu sinh viên tạo Stored Procedure và dùng đối tượng Command để gọi, thực thi các thủ tục đó.

1. Trong màn hình FoodInfoForm, có một nút dùng để thêm mới nhóm món ăn. Hãy thiết kế Form để thêm mới nhóm món ăn. Sau khi nhấn nút Add New để thêm mới nhóm món ăn rồi tắt Form này, nhóm món ăn mới thêm phải được đưa vào ComboBox (cbbCatName).

2. Thiết kế Form: OrdersForm và viết phương thức xử lý để

- Hiển thị danh sách hóa đơn được bán trong một khoảng thời gian nào đó (yêu cầu có ô chọn từ ngày nào tới ngày nào – sử dụng 2 DateTimePicker). Hiển thị tổng số tiền chưa giảm giá, số tiền giảm giá, thực thu doanh thu trong ngày đã chọn.
- Khi nhấp đôi chuột vào một hóa đơn nào đó thì mở một Form mới (OrderDetailsForm) để hiển thị danh mục các mặt hàng mua bởi hóa đơn đó.

3. Thiết kế Form: AccountForm và viết các phương thức xử lý để

- Hiển thị danh sách tài khoản
- Thêm một tài khoản mới. Có thể thêm vai trò mới trước khi thêm tài khoản bằng cách xử lý sự kiện chon nút "Thêm vai trò" .
- Cập nhật thông tin tài khoản.
- Reset mật khẩu cho tài khoản.
- Khi nhấp phải chuột vào một tài khoản, hiển thị menu sau:

Xem danh sách các vai trò
Xem nhật ký hoạt động

Trong đó:

- Xem danh sách các vai trò: Mở một Form mới, hiển thị danh sách chi tiết các vai trò của hệ thống.Tài khoản được gán vai trò nào thì cột đầu tiên của danh sách sẽ được check. Form này cũng chứa 3 nút là AddNew để thêm mới vai trò, Update để thay đổi danh sách vai trò gán cho người dùng, và Cancel để đóng màn hình.
- Xem nhật ký hoạt động: Phần bên trái của màn hình là một ListBox chứa ngày lập của các hóa đơn. Khi nhấn chuột vào ngày nào thì hiển thị thông tin chi tiết (cả danh mục món ăn) của hóa đơn ở phần bên phải Form. Phía dưới Form, liệt kê số lượng hóa đơn tài khoản đã lập), tổng số tiền của tất cả các hóa đơn.

CHỦ ĐỀ 6. MÔ HÌNH ĐA TẦNG

A. Mục tiêu

Chủ đề 6 giúp người học làm quen với mô hình đa tầng trong phát triển ứng dụng. Bài thực hành này hướng dẫn người học biết cách 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.

Sau hoàn thành chuyên đề, người học có khả năng:

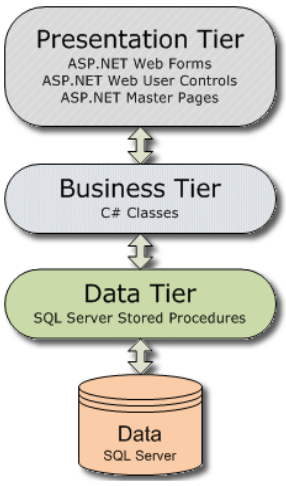
- Nắm vững kiến thức về mô hình ba tầng, hiểu được vai trò và chức năng của các tầng trong mô hình.
- Xây dựng được ứng dụng đơn giản theo mô hình ba tầng.

B. Hướng dẫn 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 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, được kết nối với nhau thông qua việc giao tiếp với 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:

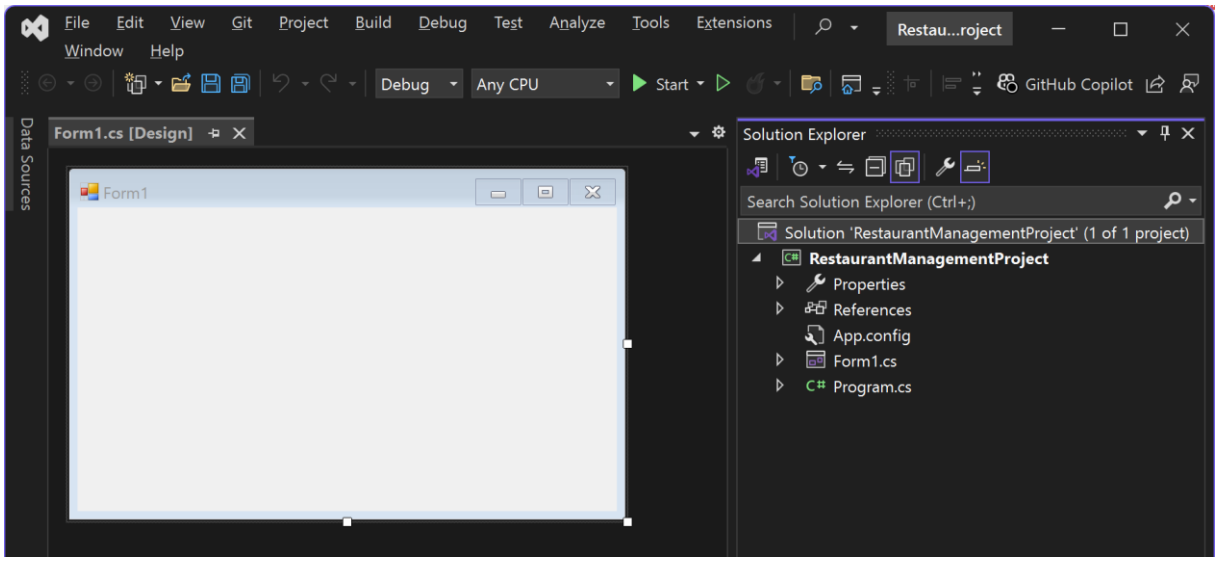


- Tầng giao diện (*Presentation*): Dùng để hiển thị các thành phần giao diện và tương tác với người dùng như tiếp nhận các thông tin đầu vào, thông báo kết quả, thông báo lỗi;
- Tầng xử lý (*Business Logic*): Thực hiện các hành động nghiệp vụ của phần mềm như tính toán, thêm, xóa, sửa dữ liệu;
- Tầng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

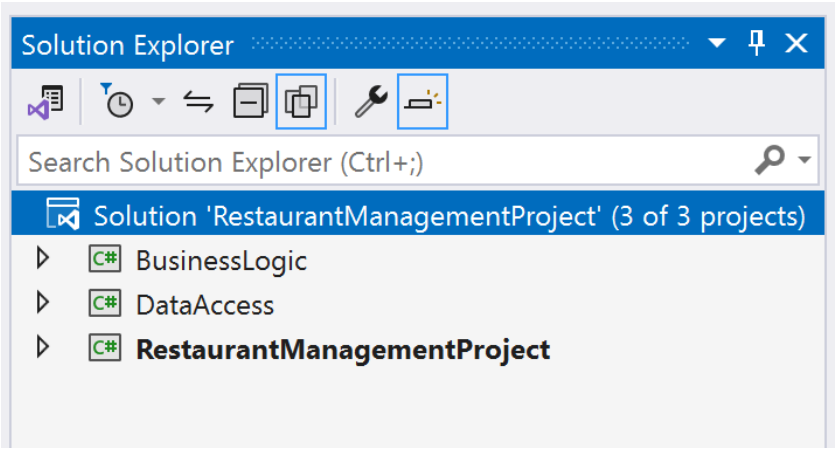
2. Tạo mô hình đa tầng với Visual Studio

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). Phần dưới đây sẽ hướng dẫn tạo một dự án Windows Form với mô hình ba tầng trên Visual Studio:

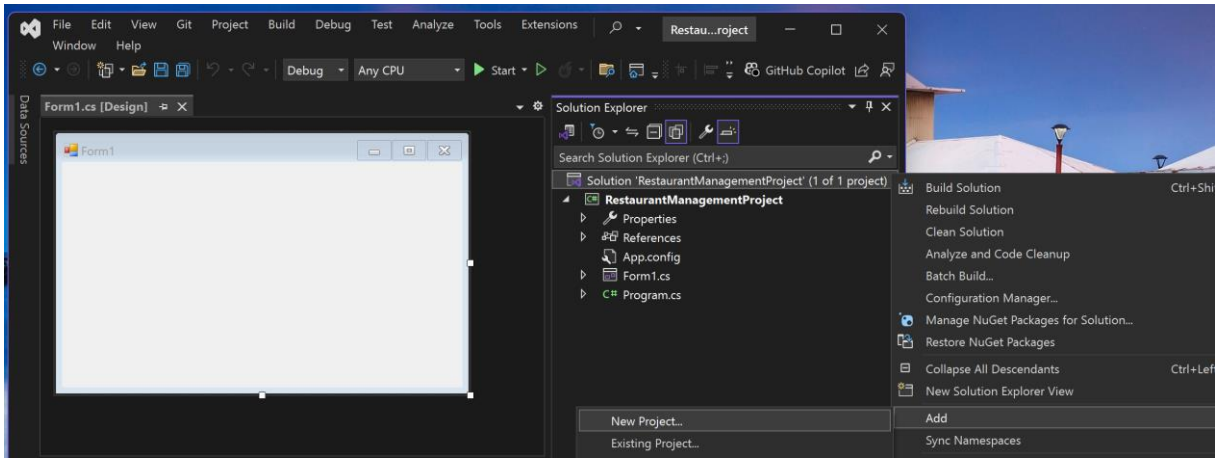
- **Bước 1**: Mở Visual Studio, tạo dự án Windows Form bằng ngôn ngữ C#, đặt tên dự án này là *RestaurantManagementProject*:

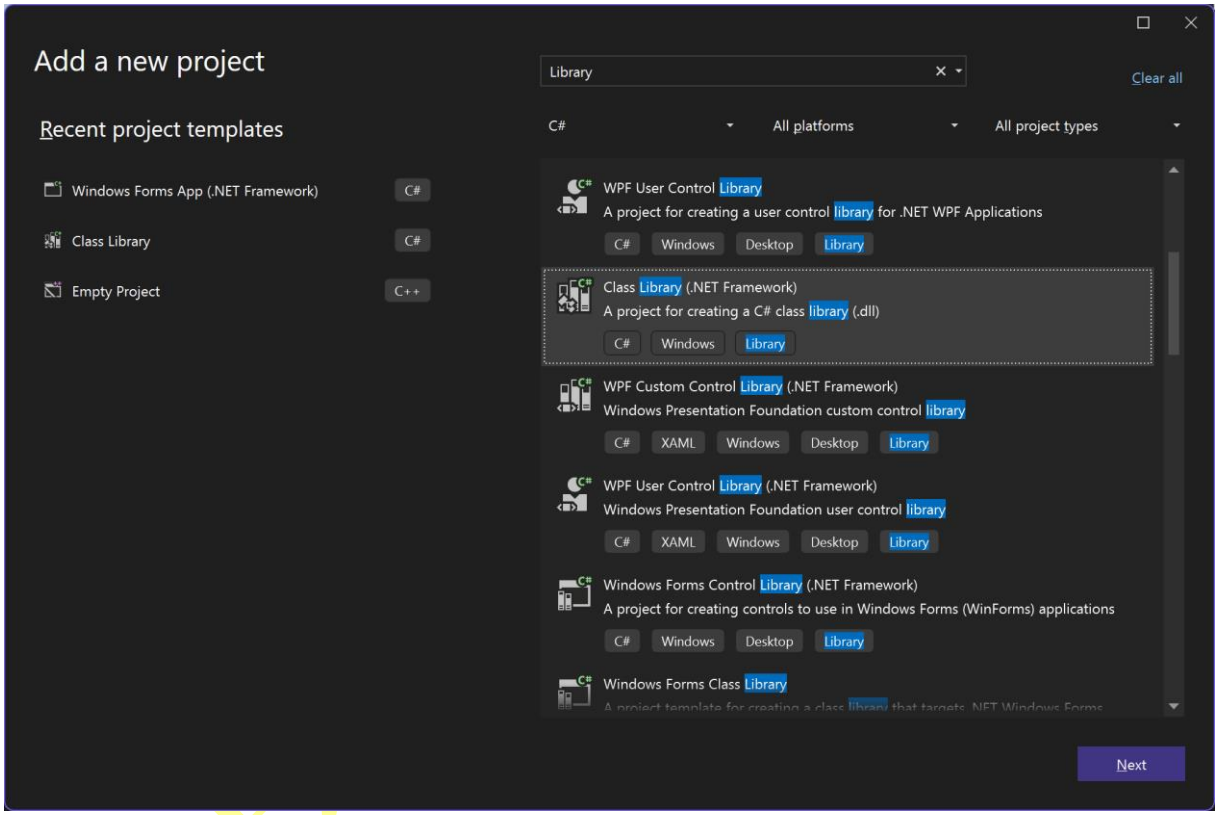


Lưu ý: Sau khi tạo xong dự án, ta thấy có một *Solution* và một *Project* được tạo ra. Từ đây, *Solution* là giải pháp chung cho cả dự án, mỗi một *Project* đại diện cho một tầng. Tầng vừa tạo chính là tầng Giao diện, dùng để xử lý các vấn đề liên quan đến giao diện chương trình.



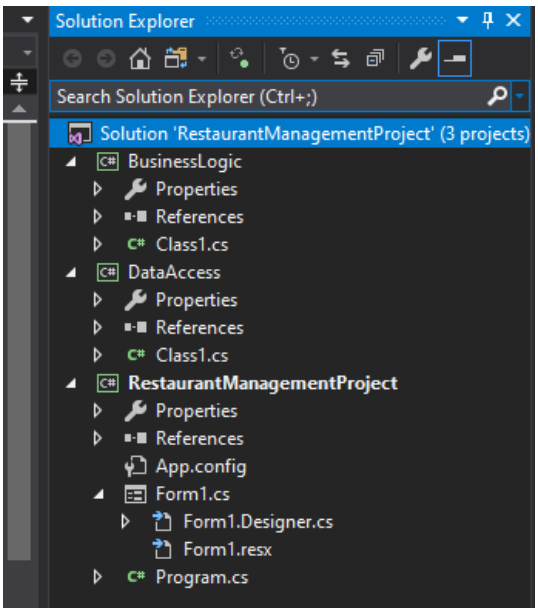
- **Bước 2**: Tạo tầng xử lý bằng cách nhấp chuột phải lên *Solution*, chọn *Add*, chọn *New Project*. Trong hộp thoại hiện lên chọn ngôn ngữ *C#* /chọn kiểu dự án là *Class Library* / bấm *Next*, đặt tên dự án là *BusinessLogic/Create*.



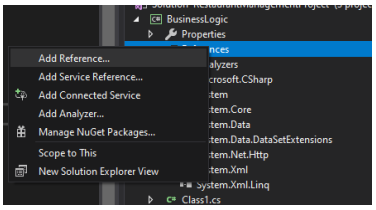


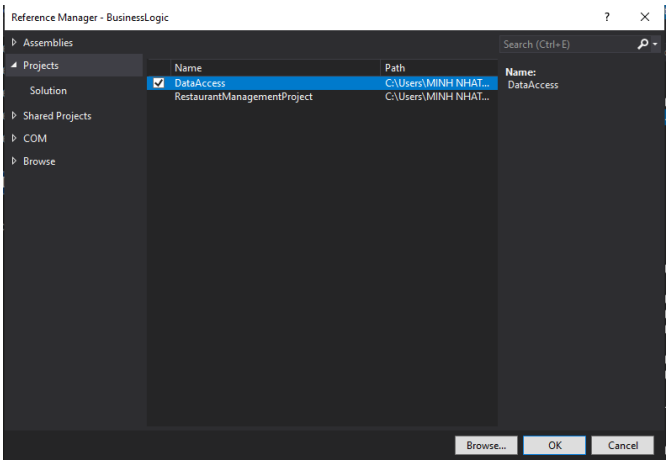
Lưu ý: Đây là một dự án dạng thư viện, thư viện này được tạo ra dưới dạng các tập tin *.dll khi chạy ứng dụng.

- **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:

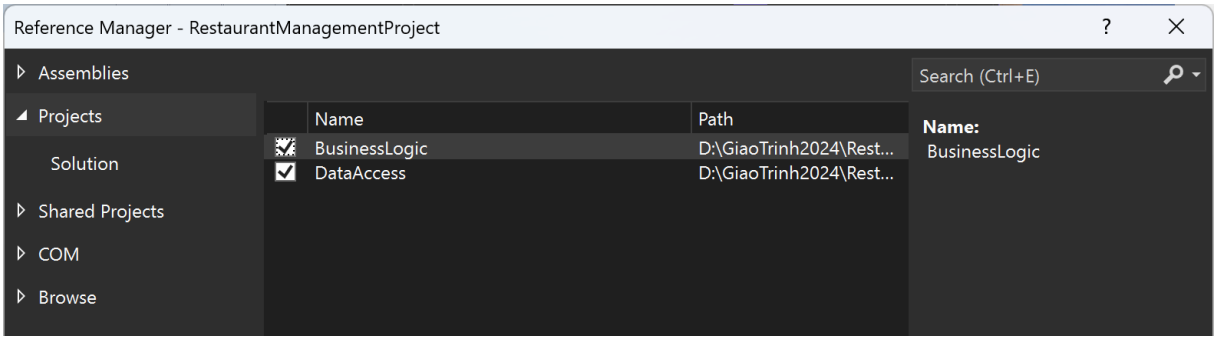


- **Bước 4**: Để có thể gọi các phương thức của tầng *DataAccess* từ tầng *BusinessLogic*, nhấp chuột phải lên thành phần *References* của Project *BusinessLogic*, chọn *Add Reference…*, đánh dấu tích vào *DataAccess* -> *OK*:





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

Nhấp chuột 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 (nhấp chuột 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

Nhấp chuột 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:

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

public class Food
{
    // ID của bảng Food
    0 references
    public int ID { get; set; }
    // Tên loại đồ ăn, thức uống
    0 references
    public string Name { get; set; }
    // Đơn vị tính
    0 references
    public string Unit { get; set; }
    //Loại thức ăn, ứng với bảng ở trên
    0 references
    public int FoodCategoryID { get; set; }
    // Giá
    0 references
    public int Price { get; set; }
    // Ghi chú
    0 references
    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
0 references
public class CategoryDA
{
    //Phương thức lấy hết dữ liệu theo thủ tục Food_GetAll
    0 references
    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
0 references
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
0 references
public class FoodDA
{
    // Phương thức lấy hết dữ liệu theo thủ tục Food_GetAll
    0 references
    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
0 references
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, và tìm theo các tiêu chí khác.

- **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
0 references
public class CategoryBL
{
    //Đối tượng CategoryDA từ DataAccess
    CategoryDA categoryDA = new CategoryDA();
    //Phương thức lấy hết dữ liệu
    0 references
    public List<Category> GetAll()
    {
        return categoryDA.GetAll();
    }
    //Phương thức thêm dữ liệu
    0 references
    public int Insert(Category category)
    {
        return categoryDA.Insert_Update_Delete(category, 0);
    }
    //Phương thức cập nhật dữ liệu
    0 references
    public int Update(Category category)
    {
        return categoryDA.Insert_Update_Delete(category, 1);
    }
    //Phương thức xoá dữ liệu truyền vào ID
    0 references
    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
0 references
public class FoodBL
{
    //Đối tượng CategoryDA từ DataAccess
    FoodDA foodDA = new FoodDA();
    //Phương thức lấy hết dữ liệu
    1 reference
    public List<Food> GetAll()
    {
        return foodDA.GetAll();
    }
    // Phương thức lấy về đối tượng Food theo khoá chính
    0 references
    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
0 references
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
0 references
public int Insert(Food food)
{
    return foodDA.Insert_Update_Delete(food, 0);
}
//Phương thức cập nhật dữ liệu
0 references
public int Update(Food food)
{
    return foodDA.Insert_Update_Delete(food, 1);
}
//Phương thức xoá dữ liệu với ID cho trước
0 references
public int Delete(Food food)
{
    return foodDA.Insert_Update_Delete(food, 2);
}
```

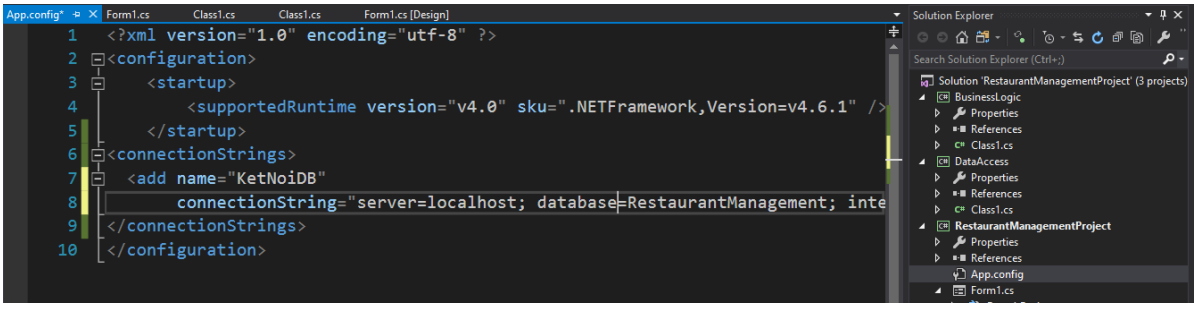
Lớp *CategoryBL* và *FoodBL* nếu có nhu cầu thêm, có thể viết thêm các phương thức khác bổ sung. Khi viết xong tầng *BusinessLogic*, có thể kiểm tra lỗi và tạo tập tin *.dll bằng cách nhấp chuột phải lên Project *BusinessLogic*, chọn *Build*, khi đó nếu có lỗi sai, hệ thống sẽ báo để sửa lỗi trước khi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

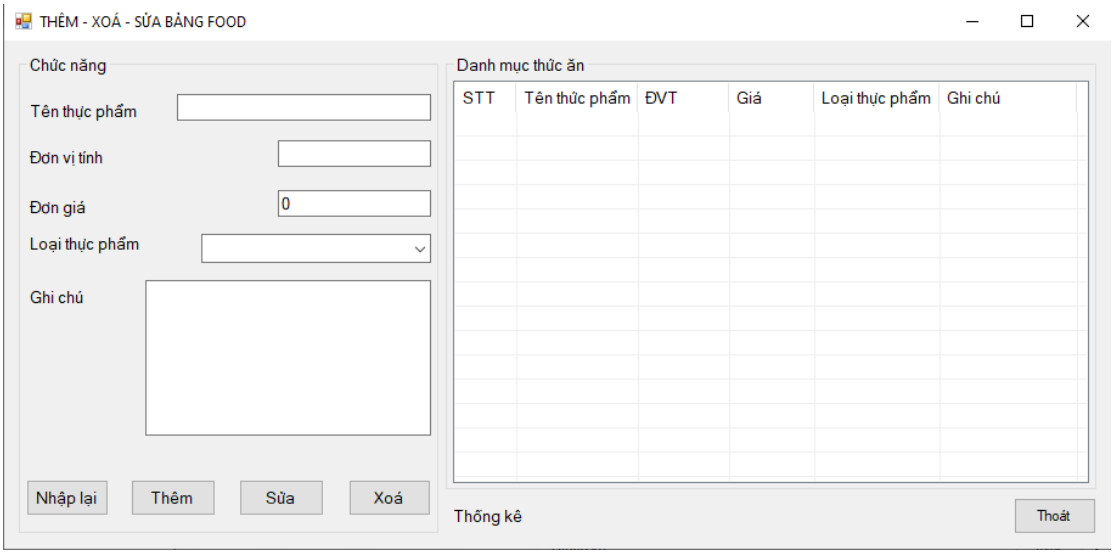
```
<add name=" ConnectionStringName" connectionString="server=localhost;
database=RestaurantManagement; integrated security=true;"/>
```



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, xóa, 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 frmFood_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++;
    }
}
```

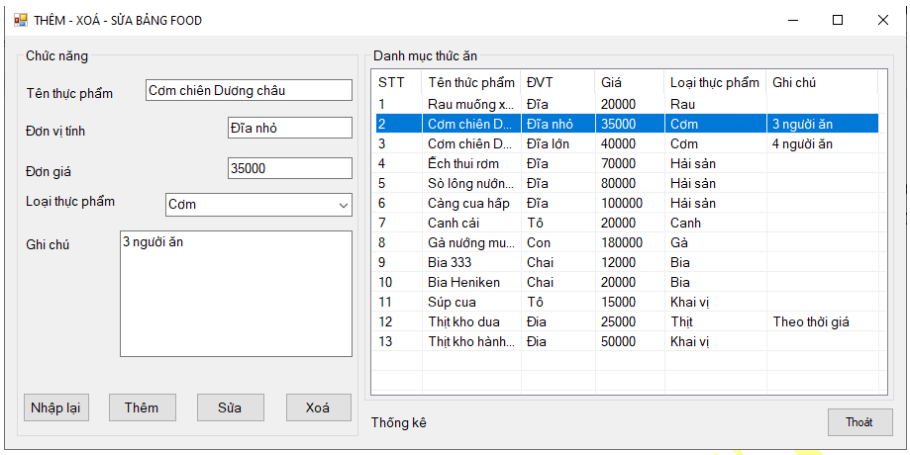
- **Bước 4**: Sự kiện Click lên dòng của ListView

Khi người dùng nhấp 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*. Nhấp chuột 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 cho bảng food, nếu thành công trả về số dương, ngược lại trả về số âm:

```
/// <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>
0 references
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 từ 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 Nhấp 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 và sự kiện nhấn nút 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");
}
```

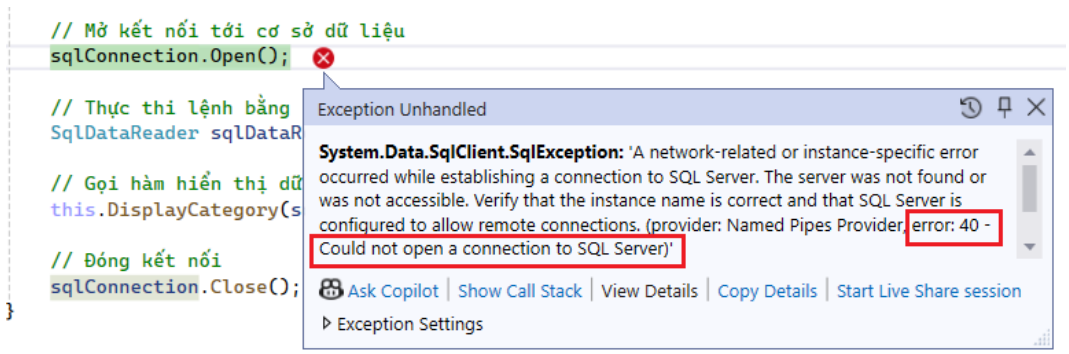
```
/// <summary>
/// Phương thức cập nhật dữ liệu cho bảng Food
/// </summary>
/// <returns>Trả về dương nếu cập nhật thành công, ngược lại là số âm</returns>
0 references
public int UpdateFood()
{
    //Khai báo đối tượng Food và lấy đối tượng hiện hành
    Food food = foodCurrent;
    // Kiểm tra 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;
}
```

6. Các lỗi thường gặp khi kết nối cơ sở dữ liệu

Phần này liệt kê và hướng dẫn khắc phục một số lỗi phổ biến khi kết nối với cơ sở dữ liệu dùng ADO.NET.

Lỗi "Could not open a connection to SQL server"

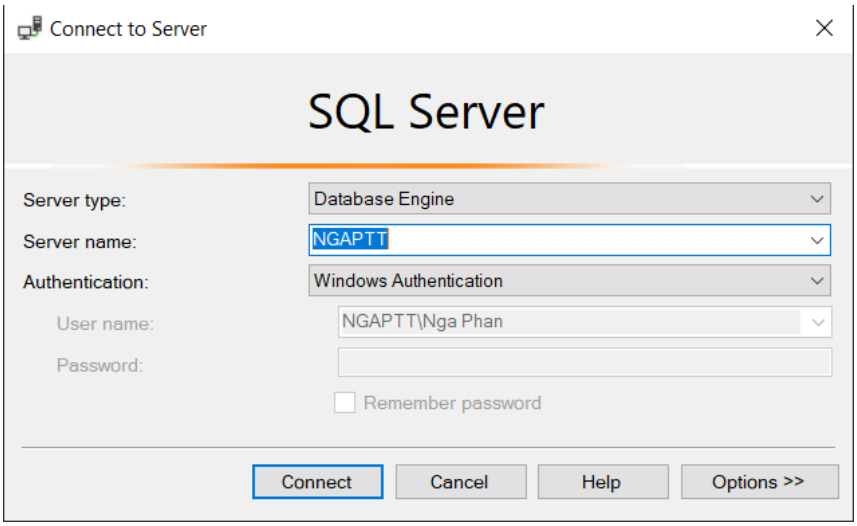
Lỗi này được hiển thị chi tiết trong hình dưới đây:



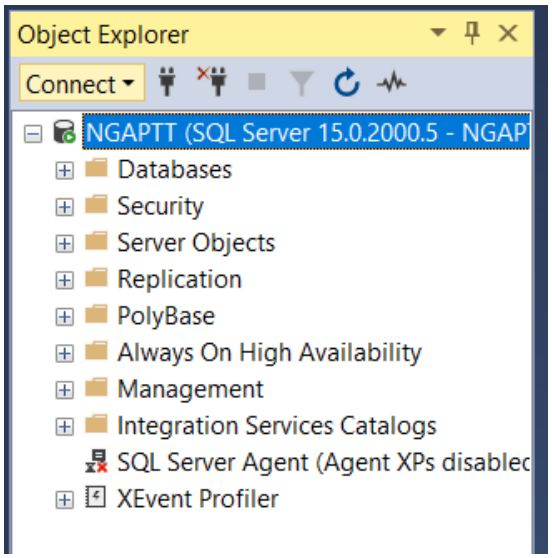
Lỗi này xuất hiện là do tên của SQL server instance đã không được cung cấp đúng. Cụ thể, lỗi sai được tô vàng như hình dưới đây:

```
// Tạo chuỗi kết nối tới cơ sở dữ liệu RestaurantManagement
string connectionString = "server=PC301; database = RestaurantManagement; Integrated Security = true; ";
```

Để khắc phục lỗi này, bạn cần kiểm tra lại thông tin kết nối tới SQL server và điền lại thông tin server name cho hợp lý. Để kiểm tra, mở phần mềm "*Microsoft SQL Server Management Studio*", cửa sổ *"Connect to Server"* sau sẽ xuất hiện:



Thực hiện chọn server name và bấm "***Connect***", kết quả sau khi kết nối thành công tới SQL server thể hiện trong hình sau:

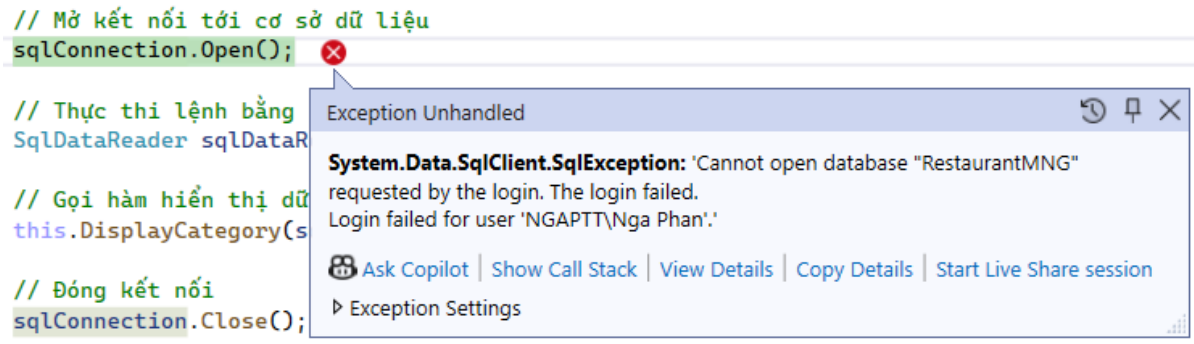


Sau khi kết nối thành công SQL server, bạn cần cập nhật lại tên server trong chuỗi kết nối đúng với tên bạn đã chọn ở bước trước. Trong ví dụ này, tên server sẽ được cập nhật lại như sau:

```
// Tạo chuỗi kết nối tới cơ sở dữ liệu RestaurantManagement
string connectionString = "server=NGAPTT; database = RestaurantManagement; Integrated Security = true; ";
```

Lỗi "Cannot open database"

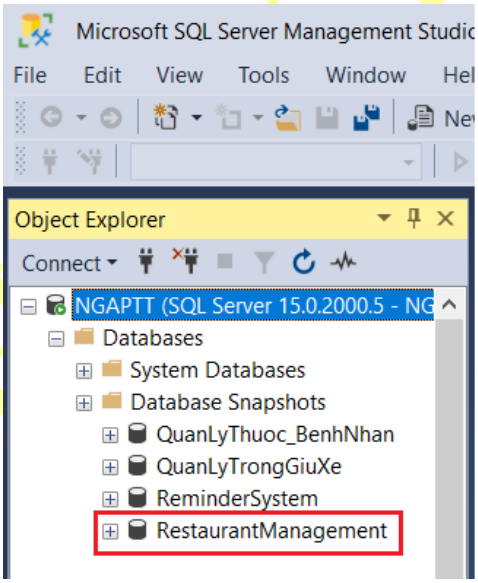
Một lỗi khác có thể xuất hiện khi thiết lập thông tin cho chuỗi kết nối là lỗi không thể mở database như hình minh họa dưới đây.



Lỗi này xuất hiện do người dùng cung cấp sai tên database trong chuỗi kết nối như hình dưới đây:

```
// Tạo chuỗi kết nối tới cơ sở dữ liệu RestaurantManagement
string connectionString = "server=NGAPTT; database = RestaurantMNG; Integrated Security = true; ";
```

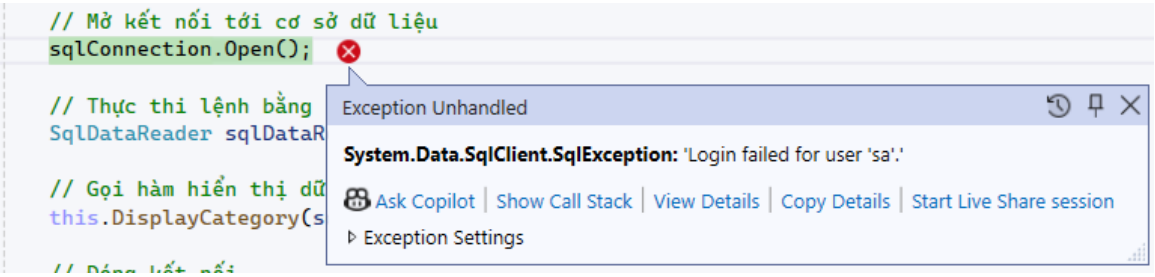
Cách khắc phục: Kiểm tra và sửa lại tên của cơ sở dữ liệu đúng với tên Database đã tạo.




```
// Tạo chuỗi kết nối tới cơ sở dữ liệu RestaurantManagement
string connectionString = "server=NGAPTT; database = RestaurantManagement; Integrated Security = true; ";
```

Lỗi "Login failed"

Lỗi này liên quan tới thông tin đăng nhập SQL server trong chuỗi kết nối bị sai. Trong trường hợp này, thay vì dùng tài khoản đăng nhập là tài khoản đăng nhập Windows (*integrated security=true*) người dùng cung cấp thông tin tài khoản và mật khẩu đăng nhập SQL server thông qua 2 tham số ***uid*** và ***pwd*** trong chuỗi kết nối. Tuy nhiên, một trong hai thông tin cung cấp chưa chính xác.

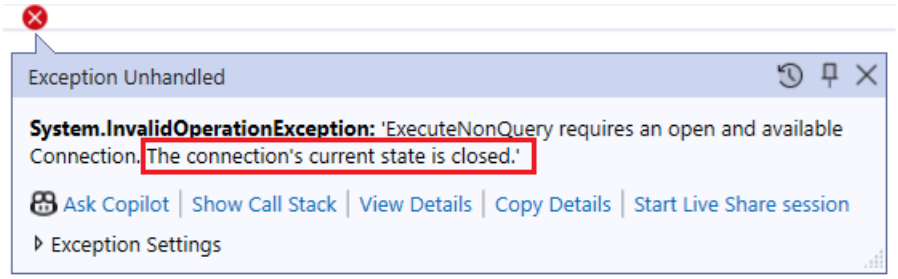


Để khắc phục lỗi này cần kiểm tra lại thông tin tài khoản đăng nhập (tên tài khoản và mật khẩu) và sửa lại thông tin đúng trong chuỗi kết nối.

```
// Tạo chuỗi kết nối tới cơ sở dữ liệu RestaurantManagement
string connectionString = "server=NGAPTT; database = RestaurantManagement; uid=tst; pwd=123";
```

Lỗi "The connection's current state is closed"

Lỗi này thường xuất hiện khi người dùng quên mở kết nối. Thông báo lỗi có thể có dạng như hình dưới đây.



Để khắc phục lỗi này, thêm dòng code mở kết nối trước đoạn mã gọi thực thi lệnh như hình minh họa dưới đây:

```
// Mở kết nối tới cơ sở dữ liệu
sqlConnection.Open();

// Thực thi lệnh bằng phương thức ExcuteNonQuery
int numOfRowsEffected = sqlCommand.ExecuteNonQuery();
```

C. 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.

CHỦ ĐỀ 7. SỬ DỤNG ENTITY FRAMEWORK

A. Mục tiêu

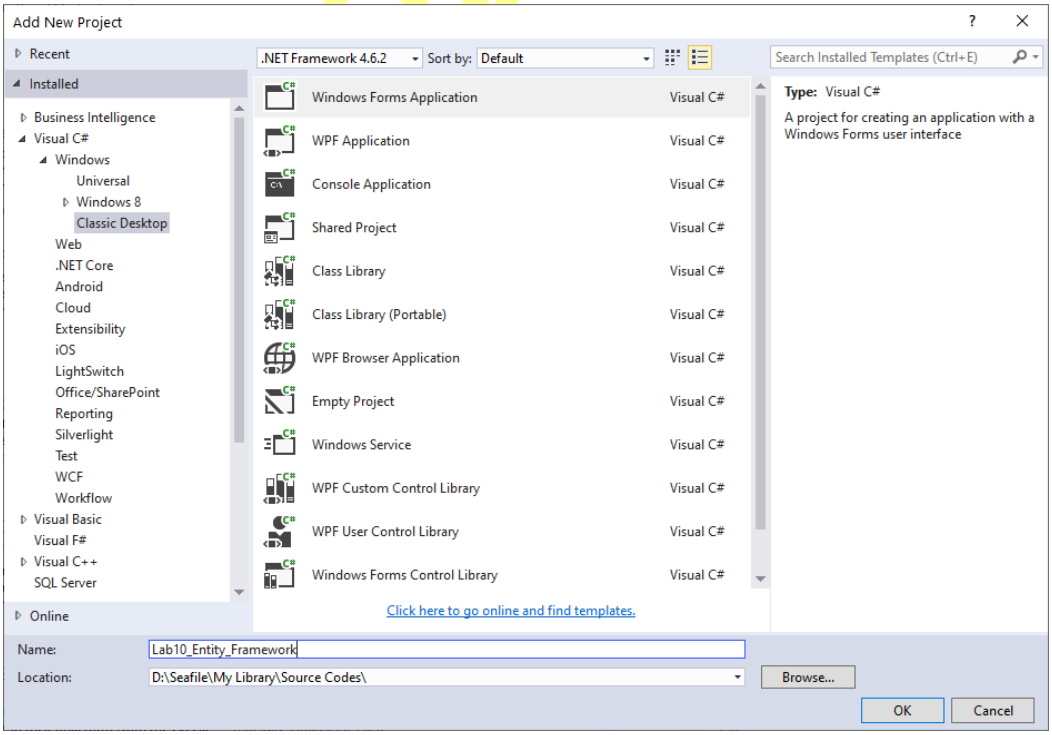
Chủ đề này giúp sinh viên tìm hiểu cách sử dụng Entity Framework để kết nối tới cơ sở dữ liệu và thực hiện các truy vấn đơn giản như SELECT, INSERT, UPDATE, DELETE.

Sau khi hoàn thành chủ đề, sinh viên có thể nắm rõ những vấn đề sau:

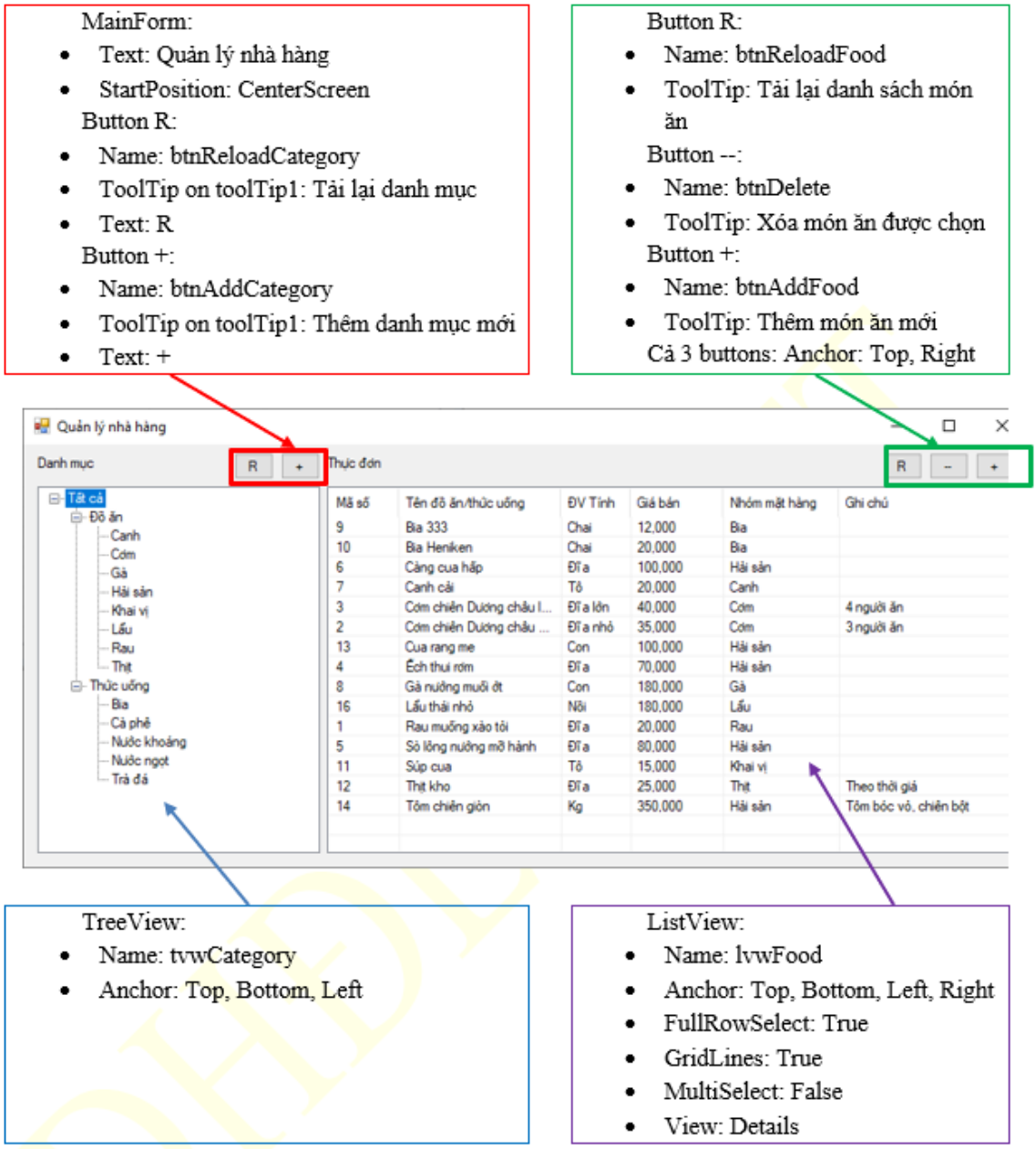
- Các thành phần của chuỗi kết nối, ý nghĩa của chúng và cách tạo chuỗi kết nối.
- Cách định nghĩa các lớp thực thể và lớp ngữ cảnh.
- Cách sử dụng đối tượng Context để thực thi truy vấn, thêm, cập nhật dữ liệu vào cơ sở dữ liệu.
- Cách sử dụng lớp DTO để lấy thông tin từ nhiều bảng.
- Cách xây dựng ứng dụng trên nền Windows Form với Entity Framework.

B. Hướng dẫn thực hành

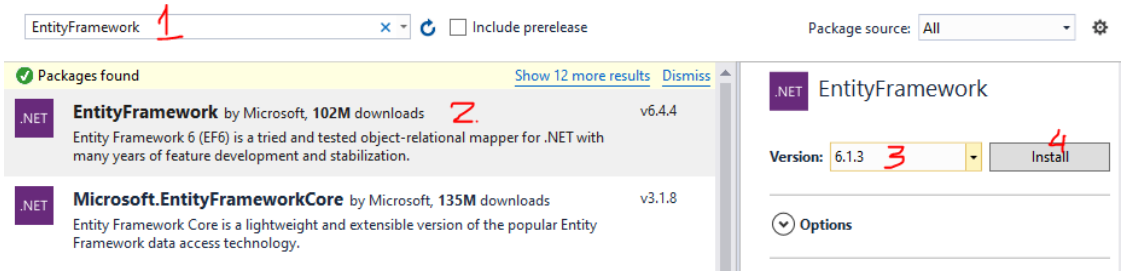
Tạo một dự án Windows Application mới, đặt tên là Lab09_Entity Framework.



Đặt lại tên Form1 thành MainForm. Kéo một điều khiển ToolTip vào MainForm, MainForm được thiết kế như sau:



1. Cài đặt gói thư viện EntityFramework



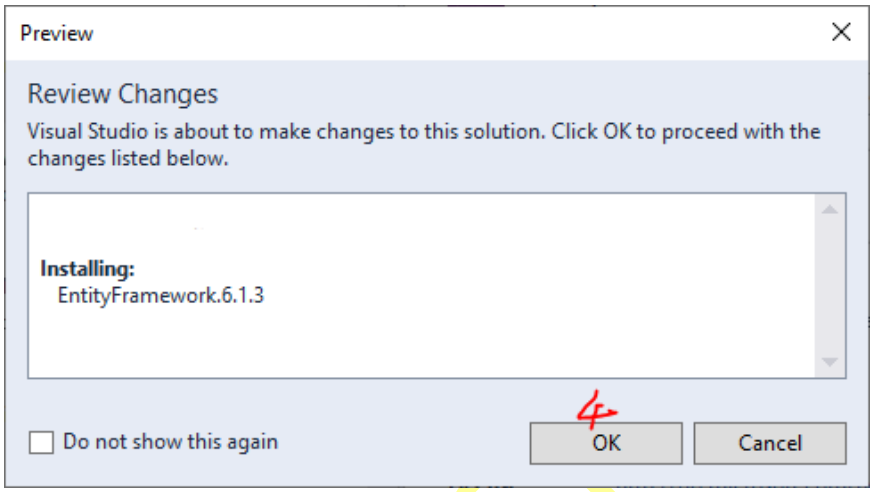
Nhấp phải chuột vào dự án, chọn Manage Nuget Packages,…

- Trong cửa sổ mới "NuGet: Lab09_Entity_Framework", chọn tab Browse. Trong mục Package source, chọn All. Nhập từ khóa EntityFramework vào ô tìm kiếm.

- Trong cửa sổ hiển thị kết quả tìm kiếm, chọn EntityFramework.

- Ở khung kế bên phải, chọn phiên bản muốn cài đặt. Sau đó nhấn nút Install. Lưu ý: tùy vào thời điểm bạn cài đặt, phiên bản của Entity Framework có thể khác với số phiên bản trong hình minh họa ở trên. Tuy nhiên, cách làm hoàn toàn tương tự.

- Trong cửa sổ popup Preview, nhấn nút OK để xác nhận và bắt đầu cài đặt thư viện.



2. Định nghĩa các lớp thực thể và lớp ngữ cảnh (DbContext)

Nhấp phải chuột vào tên dự án, chọn Add > New Folder. Đặt tên thư mục là Models. Trong thư mục Models, tạo ra các lớp thực thể sau đây:

- Lớp *Category.cs*: Biểu diễn thông tin nhóm đồ ăn, thức uống.

```
public class Category
{
    5 references
    public int Id { get; set; }

    9 references
    public string Name { get; set; }

    6 references
    public CategoryType Type { get; set; }
}

7 references
public enum CategoryType
{
    Drink,
    Food
}
```

- Lớp *Food.cs*: Biểu diễn thông tin về một món ăn, đồ uống.

```
public class Food
{
    4 references
    public int Id { get; set; }

    8 references
    public string Name { get; set; }

    6 references
    public string Unit { get; set; }

    6 references
    public int FoodCategoryId { get; set; }

    6 references
    public int Price { get; set; }

    6 references
    public string Notes { get; set; }


    4 references
    public Category Category { get; set; }
}
```

Tiếp theo, để có thể truy xuất tới cơ sở dữ liệu, ta cần định nghĩa lớp ngữ cảnh RestaurantContext.cs trong thư mục Models như sau:

```
public class RestaurantContext : DbContext
{
    // Tham chiếu tới các nhóm món ăn trong bảng Category
    4 references
    public DbSet<Category> Categories { get; set; }

    // Tham chiếu tới các món ăn, đồ uống trong bảng Food
    6 references
    public DbSet<Food> Foods { get; set; }

    0 references
    protected override void OnModelCreating(DbModelBuilder modelBuilder)
    {
        // Xóa bỏ quy tắc sử dụng danh từ số nhiều cho tên bảng
        // Lúc này, thuộc tính Categories ánh xạ tới bảng Category trong db
        // Và thuộc tính Foods tương ứng với bảng Food trong cơ sở dữ liệu
        modelBuilder.Conventions.Remove<PluralizingTableNameConvention>();

        // Định nghĩa mối quan hệ một nhiều giữa hai bảng Category và Food
        modelBuilder.Entity<Food>()
            .HasRequired(x => x.Category)
            .WithMany()
            .HasForeignKey(x => x.FoodCategoryId)
            .WillCascadeOnDelete(true);
    }
}
```

Và cuối cùng, để lấy thông tin chi tiết món ăn từ cả hai bảng Food và Category, ta tạo một lớp DTO (Data Transfer Object) làm trung gian và chứa những thuộc tính cần thiết.

Nhấp phải chuột vào thư mục Models, chọn Add > Class … Đặt tên là FoodModel. Định nghĩa các thuộc tính cho lớp FoodModel như sau:

```
public class FoodModel
{
    3 references
    public int Id { get; set; }

    3 references
    public string Name { get; set; }

    3 references
    public string Unit { get; set; }

    3 references
    public string CategoryName { get; set; }

    3 references
    public int Price { get; set; }

    3 references
    public string Notes { get; set; }
}
```

3. Định nghĩa chuỗi thông tin kết nối tới cơ sở dữ liệu

Nhấp đôi chuột vào tập tin App.config để mở nó trong cửa sổ soạn thảo. Nếu chưa có, nhấp phải chuột vào tên dự án, chọn Add > New Item. Trong cửa sổ Add New Item, chọn Visual C# Items > General > Application Configuration tập tin rồi nhấn nút Add.

Trong tập tin App.config, bổ sung thêm thẻ <connectionStrings> để thêm chuỗi kết nối tới cơ sở dữ liệu như trong hình sau:

```
<configuration>
    <configSections>
        <!-- For more information on Entity Framework configuration, visit http://go.microsoft.com/fwlink/?LinkID=237468 -->
        <section name="entityFramework" type="System.Data.Entity.Internal.ConfigFile.EntityFrameworkSection, EntityFramework,
    </configSections>
    <connectionStrings>
        <add name="RestaurantContext"
             connectionString="Data Source=(local)\SQL16E;Initial Catalog=RestaurantManagement;User ID=sa;Password=123456;"
             providerName="System.Data.SqlClient"/>
    </connectionStrings>
    <startup>
        <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.6.2" />
    </startup>
```

Có thể, bạn cần phải thay đổi thông tin trong chuỗi kết nối cho phù hợp với thông tin mà bạn đã cài đặt MS SQL Server và tạo cơ sở dữ liệu.

4. Nạp danh mục các nhóm món ăn lên TreeView

Mở MainForm, nhấp đôi chuột vào form để tạo phương thức xử lý sự kiện Load trên MainForm. Tiếp đến, nhấp đôi chuột vào nút R bên trái (***btnReloadCategory***) để tạo phương thức xử lý sự kiện Click trên nút btnReloadCategory.

```
1 reference | 0 changes | 0 authors, 0 changes
private void MainForm_Load(object sender, EventArgs e)
{

}

1 reference | 0 changes | 0 authors, 0 changes
private void btnReloadCategory_Click(object sender, EventArgs e)
{

}
```

Cũng trong lớp ***MainForm.cs***, định nghĩa các phương thức sau:

```
private List<Category> GetCategories()
{
    // Khởi tạo đối tượng context
    var dbContext = new RestaurantContext();

    // Lấy danh sách tất cả nhóm thức ăn, sắp xếp theo tên
    return dbContext.Categories.OrderBy(x => x.Name).ToList();
}
```

```
private void ShowCategories()
{
    // Xóa tất cả các nút hiện có trên cây
    tvwCategory.Nodes.Clear();

    // Tạo danh sách loại nhóm thức ăn, đồ uống
    // Tên của các loại này được hiển thị trên các nút mức 2
    var cateMap = new Dictionary<CategoryType, string>()
    {
        [CategoryType.Food] = "Đồ ăn",
        [CategoryType.Drink] = "Thức uống"
    };

    // Tạo nút gốc của cây
    var rootNode = tvwCategory.Nodes.Add("Tất cả");

    // Lấy danh sách nhóm đồ ăn, thức uống
    var categories = GetCategories();

    // Duyệt qua các loại nhóm thức ăn
    foreach (var cateType in cateMap)
    {
        // Tạo các nút tương ứng với loại nhóm thức ăn
        var childNode = rootNode.Nodes.Add(cateType.Key.ToString(), cateType.Value);
        childNode.Tag = cateType.Key;

        // Duyệt qua các nhóm thức ăn
        foreach (var category in categories)
        {
            // Nếu nhóm đang xét không cùng loại thì bỏ qua
            if (category.Type != cateType.Key) continue;

            // Ngược lại, tạo các nút tương ứng trên cây
            var grantChildNode = childNode.Nodes.Add(category.Id.ToString(), category.Name);
            grantChildNode.Tag = category;
```

```
        }
    }

    // Mở rộng các nhánh của cây để thấy hết tất cả các nhóm thức ăn
    tvwCategory.ExpandAll();

    // Đánh dấu nút gốc đang được chọn
    tvwCategory.SelectedNode = rootNode;
}
```

Sau đó, gọi phương thức ***ShowCategories()*** trong trong hai phương thức xử lý sự kiện đã tạo ở phía trên:

```
private void MainForm_Load(object sender, EventArgs e)
{
    ShowCategories();
}

1 reference
private void btnReloadCategory_Click(object sender, EventArgs e)
{
    ShowCategories();
}
```

Nhấn F5 để chạy chương trình và xem kết quả.

Nhấn nút "R" ở phía bên trái để tải lại danh mục nhóm món ăn.

5. Hiển thị danh sách món ăn, đồ uống khi chọn một danh mục

Trong phần này, ta sẽ định nghĩa các phương thức và xử lý sự kiện để hiển thị danh sách các món ăn, đồ uống vào ListView phía bên phải khi người dùng chọn một nút trong TreeView bên trái.

- Nếu người dùng chọn nút Tất cả, ListView bên phải sẽ hiển thị tất cả các món ăn, thức uống.

- Nếu người dùng chọn nút Đồ ăn, tất cả các món ăn trong các danh mục đồ ăn sẽ được hiển thị trong ListView.

- Nếu người dùng chọn nút Thức uống, ListView sẽ cho thấy tất cả các đồ uống trong thực đơn.

- Nếu người dùng chọn một danh mục, các món ăn hoặc đồ uống trong danh mục (nhóm) đó sẽ được liệt kê trong ListView.

Như vậy, ta cần phải phân biệt các nút được chọn để có thể truy vấn danh sách các món ăn, đồ uống thích hợp. Ở đây, ta dùng mức (level) của các nút trên cây để phân biệt. Trước tiên, ta định nghĩa các phương thức sau:

- ***GetFoodByCategory(int? categoryId):*** Lấy danh sách món ăn theo mã danh mục. Nếu mã này bằng null thì lấy tất cả các món ăn. Sắp xếp món ăn tăng dần theo tên.

- ***GetFoodByCategoryType(CategoryType cateType)***: Lấy danh sách món ăn hoặc đồ uống, tùy theo loại danh mục.

```
private List<FoodModel> GetFoodByCategory(int? categoryId)
{
    // Khởi tạo đối tượng context
    var dbContext = new RestaurantContext();

    // Tạo truy vấn lấy danh sách món ăn
    var foods = dbContext.Foods.AsQueryable();

    // Nếu mã nhóm món ăn khác null và hợp lệ
    if (categoryId != null && categoryId > 0)
    {
        // Thì tìm theo mã số nhóm thức ăn
        foods = foods.Where(x => x.FoodCategoryId == categoryId);
    }

    // Sắp xếp đồ ăn, thức uống theo tên và trả về
    // danh sách chứa đầy đủ thông tin về món ăn.
    return foods
        .OrderBy(x => x.Name)
        .Select(x => new FoodModel()
        {
            Id = x.Id,
            Name = x.Name,
            Unit = x.Unit,
            Price = x.Price,
            Notes = x.Notes,
            CategoryName = x.Category.Name
        })
        .ToList();
}
private List<FoodModel> GetFoodByCategoryType(CategoryType cateType)
{
    var dbContext = new RestaurantContext();

    // Tìm các món ăn theo loại nhóm thức ăn (Category Type).
    // Sắp xếp đồ ăn, thức uống theo tên và trả về
    // danh sách chứa đầy đủ thông tin về món ăn.
    return dbContext.Foods
        .Where(x => x.Category.Type == cateType)
        .OrderBy(x => x.Name)
        .Select(x => new FoodModel()
        {
            Id = x.Id,
            Name = x.Name,
            Unit = x.Unit,
            Price = x.Price,
            Notes = x.Notes,
            CategoryName = x.Category.Name
        })
        .ToList();
}
```

Tiếp đến, ta định nghĩa phương thức ***ShowFoodsForNode*** nhận đầu vào là một nút của cây. Kiểm tra xem nút đó ở mức nào để gọi các phương thức tương ứng đã định nghĩa ở trên. Lưu ý rằng, trong phương thức ***ShowCategories***(), ta đã lưu các thông tin cần thiết vào nút của cây.

```
private void ShowFoodsForNode(TreeNode node)
{
    // Xóa danh sách thực đơn hiện tại khỏi listview
    lvwFood.Items.Clear();

    // Nếu node = null, không cần xử lý gì thêm
    if (node == null) return;

    // Tạo danh sách để chứa danh sách các món ăn tìm được
    List<FoodModel> foods = null;

    // Nếu nút được chọn trên TreeView tương ứng với
    // loại nhóm thức ăn (Category Type) (mức thứ 2 trên cây)
    if (node.Level == 1)
    {
        // Thì lấy danh sách món ăn theo loại nhóm
        var categoryType = (CategoryType)node.Tag;
        foods = GetFoodByCategoryType(categoryType);
    }
    else
    {
        // Ngược lại, lấy danh sách món ăn theo thể loại
        // Nếu nút được chọn là 'Tất cả' thì lấy hết
        var category = node.Tag as Category;
        foods = GetFoodByCategory(category?.Id);
    }

    // Gọi hàm để hiển thị các món ăn lên ListView
    ShowFoodsOnListView(foods);
}

private void ShowFoodsOnListView(List<FoodModel> foods)
{
    // Duyệt qua từng phần tử của danh sách food
    foreach (var foodItem in foods)
    {
        // Tạo item tương ứng trên ListView
        var item = lvwFood.Items.Add(foodItem.Id.ToString());

        // Và hiển thị các thông tin của món ăn
        item.SubItems.Add(foodItem.Name);
        item.SubItems.Add(foodItem.Unit);
        item.SubItems.Add(foodItem.Price.ToString("##,###"));
        item.SubItems.Add(foodItem.CategoryName);
        item.SubItems.Add(foodItem.Notes);
    }
}
```

Để xử lý sự kiện chọn một nút của cây, ta làm như sau:

- Nhấp phải chuột vào TreeView, chọn Properties.
- Trong cửa sổ Properties, nhấp chuột chọn chọn nút Events (hình tia sét).
- Sau đó nhấp đôi chuột vào sự kiện AfterSelect để tạo phương thức xử lý sự kiện đó.

Sau đó, gọi phương thức ***ShowFoodsForNode***() trong phương thức xử lý sự kiện và truyền nút được chọn vào đối số của phương thức. Nút được chọn được lưu trong thuộc tính Node của tham số e.

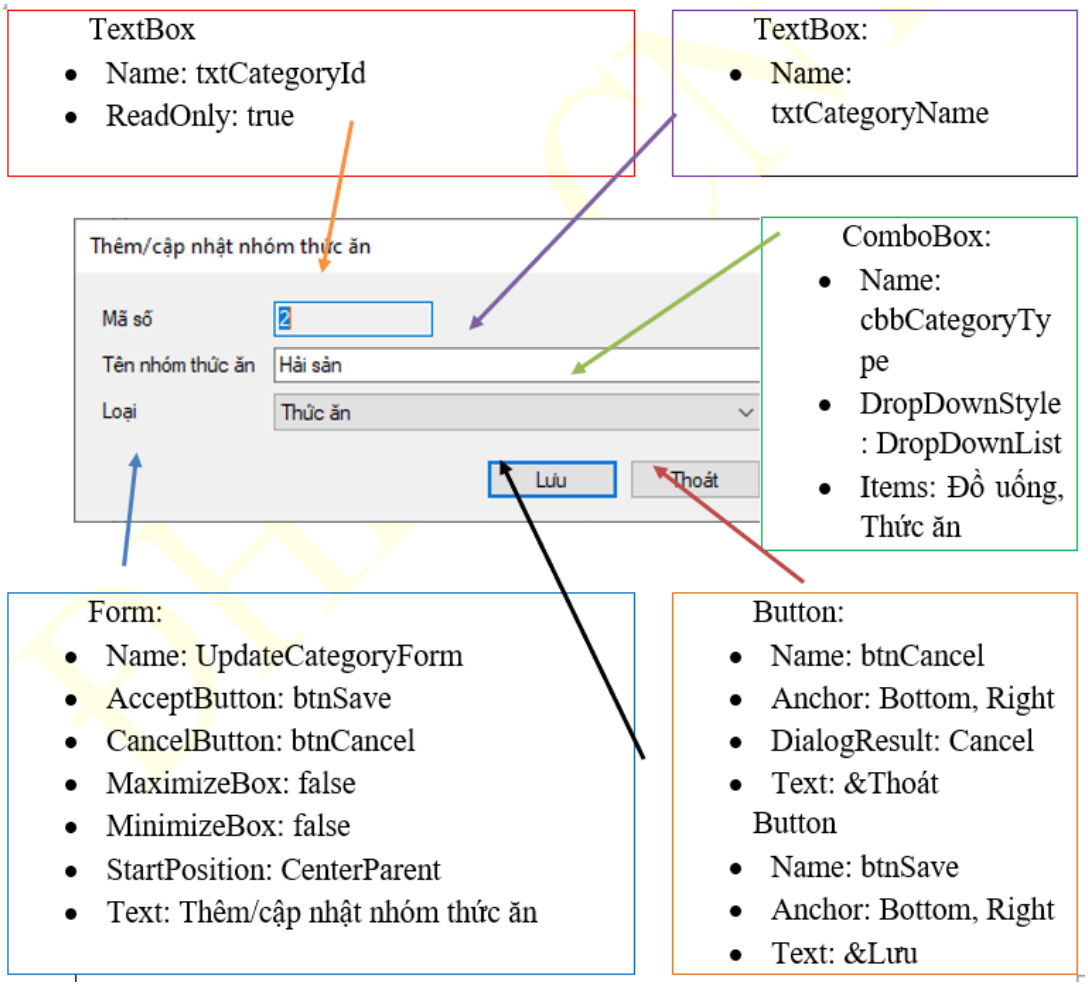
```
private void tvwCategory_AfterSelect(object sender, TreeViewEventArgs e)
{
    ShowFoodsForNode(e.Node);
}
```

Nhấn phím F5 để chạy chương trình và kiểm tra kết quả.

6. Xây dựng Form cập nhật thông tin danh mục (nhóm) món ăn, đồ uống

Phần này sẽ hướng dẫn bạn cách xây dựng một form mới – có tên ***UpdateCategoryForm*** – dùng để thêm mới hoặc cập nhật thông tin một nhóm thức ăn, đồ uống.

- Nhấp phải chuột vào tên dự án, chọn Add > Windows Form.
- Đặt tên cho form mới là UpdateCategoryForm.
- Thiết kế form có giao diện như sau:



Tiếp theo, trong lớp ***UpdateCategoryForm***, khai báo thêm 2 biến:

```
private RestaurantContext _dbContext;
private int _categoryId;
```

Và cập nhật lại phương thức khởi tạo của form như sau:

```
public UpdateCategoryForm(int? categoryId = null)
{
    InitializeComponent();

    _dbContext = new RestaurantContext();
    _categoryId = categoryId ?? 0;
}
```

Tham số ***categoryId*** có kiểu Nullable<int> và nhận giá trị mặc định là null. Điều này cho phép ta khởi tạo form mới mà không cần phải truyền vào mã danh mục món ăn. Điều này được áp dụng khi muốn tạo mới một danh mục.

Trong trường hợp tham số này nhận một giá trị dương, form sẽ ở chế độ cập nhật thông tin của một danh mục có sẵn. Lúc này, khi form được mở, ta cần hiển thị thông tin của danh mục đã được chọn.

Để truy vấn và hiển thị thông tin của một danh mục có mã cho trước, ta định nghĩa các phương thức sau đây:

```
private Category GetCategoryById(int categoryId)
{
    // Nếu ID được truyền vào là hợp lệ, ta tìm thông tin theo ID
    // Ngược lại, chỉ đơn giản trả về null, cho biết không thấy.
    return categoryId > 0 ? _dbContext.Categories.Find(categoryId) : null;
}

1 reference
private void ShowCategory()
{
    // Lấy thông tin của nhóm thức ăn
    var category = GetCategoryById(_categoryId);

    // Nếu không tìm thấy thông tin, không cần làm gì cả
    if (category == null) return;

    // Ngược lại, nếu tìm thấy, hiển thị lên form
    txtCategoryId.Text = category.Id.ToString();
    txtCategoryName.Text = category.Name;
    cbbCategoryType.SelectedIndex = (int)category.Type;
}
```

Muốn hiển thị thông tin của danh mục lên màn hình khi form được mở, ta xử lý sự kiện Load của UpdateCategoryForm như sau:

```
private void UpdateCategoryForm_Load(object sender, EventArgs e)
{
    // Hiển thị thông tin nhóm thức ăn lên form
    ShowCategory();
}
```

Khi người dùng nhấn nút Lưu, ta cần phải kiểm tra dữ liệu nhập có hợp lệ hay không. Nếu dữ liệu không hợp lệ, chương trình cần hiển thị hộp thoại thông báo cho người dùng biết. Ngược lại, ta lấy dữ liệu nhập và thực thi truy vấn để lưu chúng vào cơ sở dữ liệu. Trong form này, ta cần phải xử lý cả hai tình huống:

- Biến _categoryId > 0: nghĩa là khi form bật lên, bạn đã chọn một danh mục để cập nhật. Trong trường hợp này, chương trình cần cập nhật thông tin mới cho danh mục đó.

- Biến _categoryId = 0: nghĩa là form được mở ra để tạo mới một danh mục.

Ta định nghĩa thêm hai phương thức để kiểm tra dữ liệu nhập và lấy thông tin đã nhập như sau:

```
private Category GetUpdatedCategory()
{
    // Tạo đối tượng Category với thông tin đã nhập
    var category = new Category()
    {
        Name = txtCategoryName.Text.Trim(),
        Type = (CategoryType)cbbCategoryType.SelectedIndex
    };

    // Gán giá trị của ID ban đầu (nếu đang cập nhật)
    if (_categoryId > 0)
    {
        category.Id = _categoryId;
    }

    return category;
}

1 reference
private bool ValidateUserInput()
{
    // Kiểm tra tên nhóm thức ăn đã được nhập hay chưa
    if (string.IsNullOrWhiteSpace(txtCategoryName.Text))
    {
        MessageBox.Show("Tên nhóm thức ăn không được để trống", "Thông báo");
        return false;
    }

    // Kiểm tra loại nhóm thức ăn đã được chọn hay chưa
    if (cbbCategoryType.SelectedIndex < 0)
    {
        MessageBox.Show("Bạn chưa chọn loại nhóm thức ăn", "Thông báo");
        return false;
    }

    return true;
}
```

Tiếp đến, nhấp đôi chuột vào nút Lưu (btnSave) để tạo phương thức xử lý sự kiện Click và định nghĩa phương thức đó như sau:

```
private void btnSave_Click(object sender, EventArgs e)
{
    // Kiểm tra nếu dữ liệu nhập vào là hợp lệ
    if (ValidateUserInput())
    {
        // Thì lấy thông tin người dùng nhập vào
        var newCategory = GetUpdatedCategory();

        // Và thử tìm xem đã có nhóm thức ăn trong CSDL chưa
        var oldCategory = GetCategoryById(_categoryId);

        // Nếu chưa có (chưa tồn tại)
        if (oldCategory == null)
        {
            // Thì thêm nhóm thức ăn mới
            _dbContext.Categories.Add(newCategory);
        }
        else
        {
            // Ngược lại, ta chỉ cần cập nhật thông tin cần thiết
            oldCategory.Name = newCategory.Name;
            oldCategory.Type = newCategory.Type;
        }

        // Lưu các thay đổi xuống CSDL
        _dbContext.SaveChanges();

        // Đóng hộp thoại
        DialogResult = DialogResult.OK;
    }
}
```

7. Thêm mới, cập nhật một danh mục món ăn

Phần này sẽ hướng dẫn bạn cách sử dụng UpdateCategoryForm để thêm mới và cập nhật một danh mục món ăn.

- Để thêm mới một danh mục, ta xử lý sự kiện Click nút dấu + (***btnAddCategory***) trên MainForm và gọi phương thức để hiển thị hộp thoại ***UpdateCategoryForm***.

- Để cập nhật một danh mục, người dùng nhấp đôi chuột vào một nút trên cây. Chương trình cần xử lý sự kiện ***NodeMouseDoubleClick*** để mở form cập nhật.

- Trong cả 2 trường hợp, sau khi Lưu dữ liệu thành công, chương trình cần tải và hiển thị lại danh mục thức ăn trên TreeView.

Nhấp đôi chuột vào nút ***btnAddCategory*** và định nghĩa phương thức xử lý sự kiện ***Click*** như sau:

```
private void btnAddCategory_Click(object sender, EventArgs e)
{
    var dialog = new UpdateCategoryForm();
    if (dialog.ShowDialog(this) == DialogResult.OK)
    {
        ShowCategories();
    }
}
```

Nhấp phải chuột vào TreeView, chọn Properties > Events. Nhấp đôi chuột vào sự kiện ***NodeMouseDoubleClick*** để tạo phương thức xử lý sự kiện. Định nghĩa phương thức đó như sau:

```
private void tvwCategory_NodeMouseDoubleClick(object sender, TreeNodeMouseClickEventArgs e)
{
    if (e.Node == null || e.Node.Level < 2 || e.Node.Tag == null) return;

    var category = e.Node.Tag as Category;
    var dialog = new UpdateCategoryForm(category?.Id);
    if (dialog.ShowDialog(this) == DialogResult.OK)
    {
        ShowCategories();
    }
}
```

Nhấn phím F5 để chạy chương trình. Click vào nút dấu + hoặc nhấp đôi chuột vào 1 nút trên cây để xem cách hoạt động của chương trình.

8. Xử lý việc xóa một món ăn và nạp lại danh sách món ăn, đồ uống

Việc nạp lại danh sách đồ ăn, thức uống tương đối đơn giản vì ta đã có phương thức ***ShowFoodsForNode*** được định nghĩa ở phần trước. Trong màn hình thiết kế ***MainForm***, nhấp đôi chuột vào nút R (***btnReloadFood***) để tạo phương thức xử lý sự kiện ***Click***. Gọi phương thức ***ShowFoodsForNode*** với tham số là nút hiện đang được chọn trong TreeView.

```
private void btnReloadFood_Click(object sender, EventArgs e)
{
    ShowFoodsForNode(tvwCategory.SelectedNode);
}
```

Để xóa một món ăn, đồ uống đang được chọn trong ListView, ta cần xử lý sự kiện Click cho nút "--" (btnDeleteFood) trong MainForm.

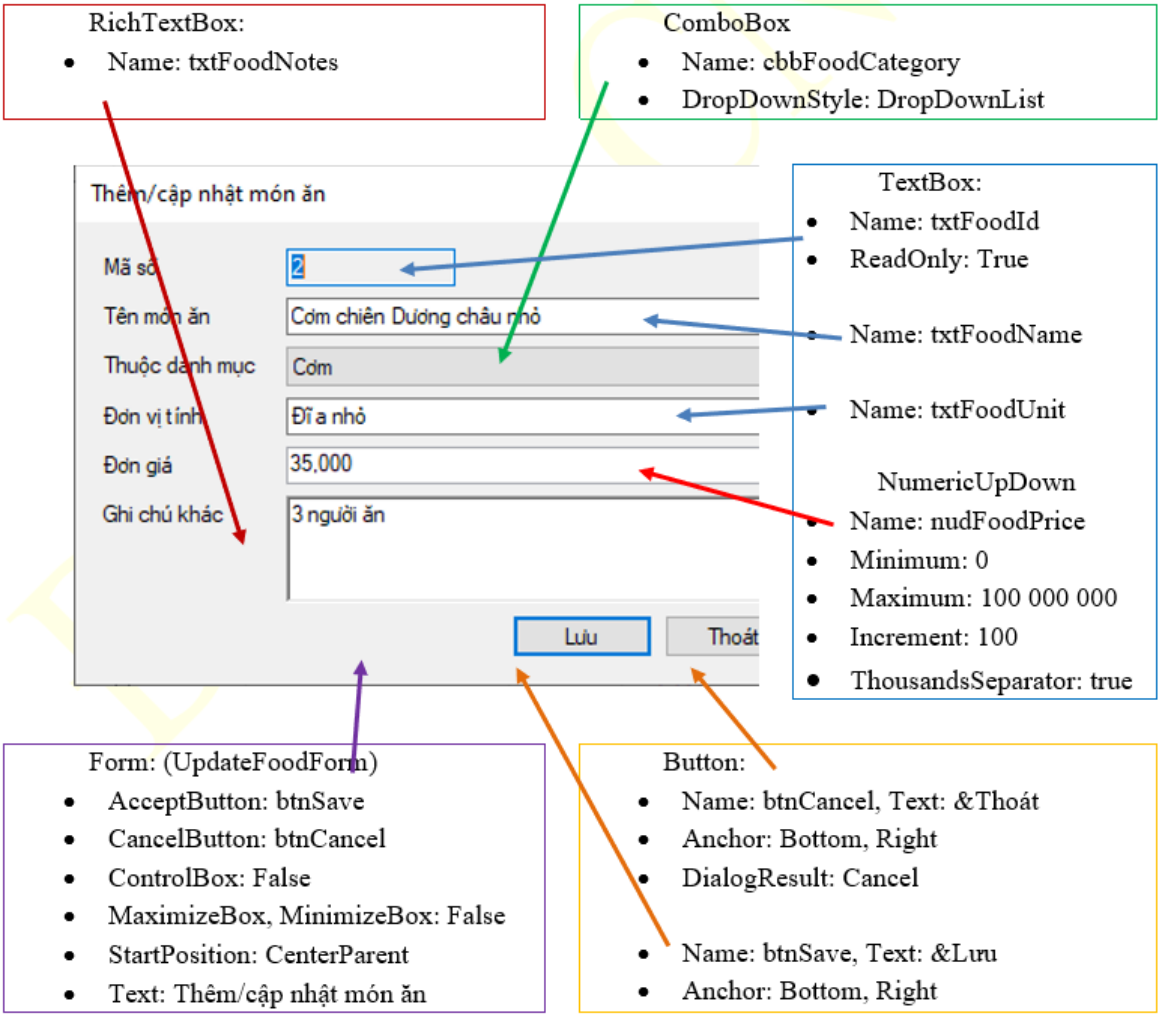
```
private void btnDeleteFood_Click(object sender, EventArgs e)
{
    // Nếu không có món ăn nào được chọn, không cần làm gì cả
    if (lvwFood.SelectedItems.Count == 0) return;

    // Ngược lại, lấy mã số của món ăn được chọn
    var dbContext = new RestaurantContext();
    var selectedFoodId = int.Parse(lvwFood.SelectedItems[0].Text);

    // Truy vấn để lấy thông tin của món ăn đó
    var selectedFood = dbContext.Foods.Find(selectedFoodId);
```

```
        // Nếu tìm thấy thông tin món ăn
        if (selectedFood != null)
        {
            // Thì xóa nó khỏi cơ sở dữ liệu
            dbContext.Foods.Remove(selectedFood);
            dbContext.SaveChanges();

            // Và đồng thời xóa khỏi ListView
            lvwFood.Items.Remove(lvwFood.SelectedItems[0]);
        }
    }
```

Chạy chương trình và thử nhấn vào nút R và nút "--" ở bên phải để xem hoạt động của chương trình.

9. Xây dựng form cập nhật thông tin món ăn, đồ uống

Để thêm và cập nhật thông tin món ăn, đồ uống, ta xây dựng form mới như hình dưới đây:



- Nhấp phải chuột vào tên dự án, chọn Add > Windows Form.
- Đặt tên cho form mới là UpdateFoodForm.

- Thiết kế form có giao diện như hình trên/
- Trong lớp ***UpdateFoodForm***, khai báo thêm 2 biến sau:

```
private RestaurantContext _dbContext;
private int _foodId;
```

- Khởi tạo hai biến đó trong phương thức khởi tạo của form như sau

```
public UpdateFoodForm(int? foodId = null)
{
    InitializeComponent();

    _dbContext = new RestaurantContext();
    _foodId = foodId ?? 0;
}
```

Tiếp theo, ta cần lấy danh sách các nhóm đồ ăn, thức uống và nạp chúng vào ComboBox khi form được mở lên. Ngoài ra, nếu người dùng mở form để cập nhật thông tin một món ăn đã chọn thì chương trình cũng cần phải truy vấn và hiển thị thông tin món ăn đó.

Để nạp danh mục đồ ăn, thức uống vào ComboBox, ta định nghĩa phương thức sau:

```
private void LoadCategoriesToCombobox()
{
    // Lấy tất cả danh mục thức ăn, sắp tăng theo tên
    var categories = _dbContext.Categories
        .OrderBy(x => x.Name).ToList();

    // Nạp danh mục vào combobox, hiển thị tên cho người
    // dùng xem nhưng khi được chọn thì lấy giá trị là ID
    cbbFoodCategory.DisplayMember = "Name";
    cbbFoodCategory.ValueMember = "Id";
    cbbFoodCategory.DataSource = categories;
}
```

Sau đó gọi tới phương thức vừa tạo trong phương thức xử lý sự kiện ***Load*** của ***UpdateFoodForm***. Bạn cần nhấp đôi chuột vào form để tạo phương thức xử lý sự kiện này.

```
private void UpdateFoodForm_Load(object sender, EventArgs e)
{
    // Nạp danh sách nhóm thức ăn vào combobox
    LoadCategoriesToCombobox();

}
```

Việc truy vấn để lấy thông tin món ăn dựa vào mã số cho trước sẽ được thực hiện nhiều lần. Vì vậy, ta có thể định nghĩa phương thức dưới đây để có thể sử dụng lại nó.

```
private Food GetFoodById(int foodId)
{
    // Tìm món ăn theo mã số
    return foodId > 0 ? _dbContext.Foods.Find(foodId) : null;
}
```

Phương thức này kiểm tra nếu mã số là hợp lệ thì lấy thông tin món ăn từ cơ sở dữ liệu. Ngược lại, chỉ đơn giản là trả về null.

Để hiển thị thông tin món ăn người dùng đã chọn lên form để cập nhật, ta định nghĩa thêm một phương thức mới:

```
private void ShowFoodInformation()
{
    // Tìm món ăn theo mã số đã được truyền vào form
    var food = GetFoodById(_foodId);

    // Nếu không tìm thấy, không cần làm gì cả
    if (food == null) return;

    // Ngược lại, hiển thị thông tin món ăn lên form
    txtFoodId.Text = food.Id.ToString();
    txtFoodName.Text = food.Name;
    cbbFoodCategory.SelectedValue = food.FoodCategoryId;
    txtFoodUnit.Text = food.Unit;
    nudFoodPrice.Value = food.Price;
    txtFoodNotes.Text = food.Notes;
}
```

Sau đó, bổ sung lời gọi phương thức tới phương thức này trong phương thức xử lý sự kiện form load.

```
private void UpdateFoodForm_Load(object sender, EventArgs e)
{
    // Nạp danh sách nhóm thức ăn vào combobox
    LoadCategoriesToCombobox();

    // Hiển thị thông tin món ăn lên form
    ShowFoodInformation();
}
```

Việc tiếp theo cần xử lý là kiểm tra dữ liệu nhập khi người dùng nhấn nút Lưu và hiển thị thông báo nếu dữ liệu nhập chưa đúng. Nếu mọi thông tin đều hợp lệ, ta cần tái tạo lại đối tượng Food từ dữ liệu nhập và thêm hoặc cập nhật vào cơ sở dữ liệu.

```
private bool ValidateUserInput()
{
    // Kiểm tra tên món ăn đã được nhập hay chưa
    if (string.IsNullOrWhiteSpace(txtFoodName.Text))
    {
        MessageBox.Show("Tên món ăn, đồ uống không được để trống", "Thông báo");
        return false;
    }

    // Kiểm tra đơn vị tính đã được nhập hay chưa
    if (string.IsNullOrWhiteSpace(txtFoodUnit.Text))
    {
        MessageBox.Show("Đơn vị tính không được để trống", "Thông báo");
        return false;
    }


    // Kiểm tra giá món ăn đã được nhập hay chưa
    if (nudFoodPrice.Value.Equals(0))
    {
        MessageBox.Show("Giá của thức ăn phải lớn hơn 0", "Thông báo");
        return false;
    }

    // Kiểm tra nhóm món ăn đã được chọn hay chưa
    if (cbbFoodCategory.SelectedIndex < 0)
    {
        MessageBox.Show("Bạn chưa chọn nhóm thức ăn", "Thông báo");
        return false;
    }

    return true;
}

private Food GetUpdatedFood()
{
    // Tạo đối tượng food với thông tin được lấy từ
    // các điều khiển trên form
    var food = new Food()
    {
        Name = txtFoodName.Text.Trim(),
        FoodCategoryId = (int)cbbFoodCategory.SelectedValue,
        Unit = txtFoodUnit.Text,
        Price = (int)nudFoodPrice.Value,
        Notes = txtFoodNotes.Text
    };

    // Gán giá trị của ID ban đầu (nếu đang cập nhật)
    if (_foodId > 0)
    {
        food.Id = _foodId;
    }

    return food;
}
```

Cuối cùng, để hoàn thành form cập nhật thông tin món ăn, bạn cần viết mã cho phương thức xử lý khi người dùng nhấn nút Lưu. Nhấp đôi chuột vào nút Lưu (btnSave) để tạo phương thức xử lý sự kiện Click và bổ sung đoạn mã sau:

```
private void btnSave_Click(object sender, EventArgs e)
{
    // Kiểm tra nếu dữ liệu nhập vào là hợp lệ
    if (ValidateUserInput())
    {
        // Thì lấy thông tin người dùng nhập vào
        var newFood = GetUpdatedFood();

        // Và thử tìm xem đã có món ăn trong CSDL chưa
        var oldFood = GetFoodById(_foodId);

        // Nếu chưa có (chưa tồn tại)

        if (oldFood == null)
        {
            // Thì thêm món ăn mới
            _dbContext.Foods.Add(newFood);
        }
        else
        {
            // Ngược lại, cập nhật thông tin món ăn
            oldFood.Name = newFood.Name;
            oldFood.FoodCategoryId = newFood.FoodCategoryId;
            oldFood.Unit = newFood.Unit;
            oldFood.Price = newFood.Price;
            oldFood.Notes = newFood.Notes;
        }

        // Lưu các thay đổi xuống CSDL
        _dbContext.SaveChanges();

        // Đóng hộp thoại
        DialogResult = DialogResult.OK;
    }
}
```

10. Gọi form cập nhật thông tin món ăn, đồ uống

Trong phần này, ta bổ sung hai phương thức trong ***MainForm.cs*** để xử lý sự kiện nhấn nút + (***btnAddFood***) để thêm món ăn mới và sự kiện nhấn đôi chuột vào một item của ListView để cập nhật thông tin món ăn, đồ uống.

```
private void btnAddFood_Click(object sender, EventArgs e)
{
    var dialog = new UpdateFoodForm();
    if (dialog.ShowDialog(this) == DialogResult.OK)
    {
        ShowFoodsForNode(tvwCategory.SelectedNode);
    }
}

1 reference
private void lvwFood_DoubleClick(object sender, EventArgs e)
{
    if (lvwFood.SelectedItems.Count == 0) return;

    var foodId = int.Parse(lvwFood.SelectedItems[0].Text);
    var dialog = new UpdateFoodForm(foodId);
    if (dialog.ShowDialog(this) == DialogResult.OK)
    {
        ShowFoodsForNode(tvwCategory.SelectedNode);
    }
}
```

Chạy chương trình và kiểm tra kết quả.

C. Bài tập

Thêm các chức năng cho chương trình như sau.

1. Thiết kế Form: RoleForm và viết phương thức xử lý để:
   - Hiển thị danh sách các nhóm (hay vai trò) người dùng
   - Thêm role mới, cập nhật một role đã có
   - Xem danh sách người dùng thuộc role được chọn

2. Thiết kế Form: AccountForm và viết phương thức xử lý để:
   - Xem danh sách tài khoản, có cho phép tìm theo role và tên người dùng
   - Thêm một tài khoản mới, trong đó cho phép chọn role cho người dùng
   - Cập nhật thông tin của một tài khoản
   - Reset mật khẩu cho tài khoản
   - Thay đổi mật khẩu của tài khoản
   - Nhấp chuột phải vào một tài khoản hiển thị menu sau:

Xóa tài khoản
Xem danh sách vai trò

Trong đó:

   - Nếu chọn Xóa tài khoản thì xóa Password, gán về null. Lúc này, người dùng không thể đăng nhập được, tài khoản ở trạng thái Inactive.
   - Xem danh sách vai trò: Mở một Form mới để hiển thị các vai trò (role) được gán cho tài khoản này.

3. Thiết kế Form: BillsForm và viết phương thức xử lý để:
   - Hiển thị danh sách hóa đơn được bán trong một khoảng thời gian nào đó (yêu cầu có ô chọn từ ngày đến ngày – sử dụng control DateTimePicker), có hiển thị tổng số tiền chưa giảm giá, tổng số tiền giảm giá, thực thu.
   - Khi nhấp đôi chuột vào một hóa đơn nào đó thì mở một Form mới (BillDetailsForm) để hiển thị danh mục các mặt hàng mua bởi hóa đơn đó.

4. Thiết kế Form: FoodForm với chức năng cụ thể như sau:
   - Hiển thị món ăn theo nhóm món ăn
   - Tìm kiếm món ăn theo tên
   - Thêm, sửa món ăn
   - Xóa các món ăn đã chọn từ danh sách
   - Thêm nhóm món ăn

5. Thiết kế Form: TableForm và viết các phương thức xử lý để:

- Hiển thị danh sách các bàn
- Xem hóa đơn hiện tại của một bàn
- Thêm một bàn mới
- Cập nhật thông tin của bàn
- Xóa một bàn
- Khi nhấp chuột phải vào một bàn, hiển thị menu sau:

Xóa bàn
Xem danh mục hóa đơn
Xem nhật ký bán hàng

Trong đó:

- Nếu chọn Xóa bàn thì dữ liệu về bàn đó sẽ bị xóa khỏi cơ sở dữ liệu.
- Xem danh mục hóa đơn: Mở một Form mới, phần bên trái của Form là một ListBox chứa ngày lập của các hóa đơn. Khi nhấn chuột vào ngày nào thì hiển thị thông tin chi tiết (cả danh mục sản phẩm được mua) của hóa đơn ở phần bên phải Form.
- Xem nhật ký bán hàng: Liệt kê số lượng hóa đơn, tổng số tiền, tổng thuế, tổng giảm giá của tất cả các hóa đơn, thông tin liên quan đến từng hóa đơn như ngày lập, tên nhân viên lập hóa đơn. (sử dụng ListView hoặc DataGridView).

6. Thiết kế Form: MainForm và viết các phương thức xử lý để xem trạng thái hoạt động của các bàn, quản lý việc đặt món cho các bàn, thanh toán tiền lập hóa đơn, in hóa đơn, chuyển bàn, nhập bàn. Từ main Form, tạo menu để mở các form quản trị. Nếu người dùng đăng nhập với vai trò quản trị viên sẽ truy cập được các form này. Nếu người dùng là nhân viên, chỉ cho xem thông tin tài khoản của họ.

7. Thiết kế Form: MenuForm để xem và in thực đơn.

CHỦ ĐỀ 8. THIẾT KẾ MẪU TRONG LẬP TRÌNH CƠ SỞ DỮ LIỆU

Trong chủ đề này, chúng ta sẽ tìm hiểu và làm quen với Repository Pattern (tạm dịch là mẫu kho lưu trữ), một mẫu thiết kế được sử dụng phổ biến trong các dự án phần mềm thông qua các ví dụ minh họa với ADO.NET và LINQ. Ngoài ra, trong lập trình xây dựng phần mềm nói chung còn có nhiều mẫu khác chẳng hạn như Inversion of Control (IoC), Singleton, Factory, Decorator, Observer,… Tuy nhiên, vì tính chất liên quan môn học lập trình cơ sở dữ liệu, chương này chỉ trình bày kiến thức và ví dụ về Repository Pattern.

A. Mục tiêu

Trong nhiều ứng dụng phần mềm, tầng Business Logic là tầng truy cập dữ liệu thông qua các kho dữ liệu như hệ thống cơ sở dữ liệu, các dịch vụ web, và danh sách SharePoint (điểm chia sẻ dữ liệu). Thông thường, nhiều ứng dụng cho phép truy cập trực tiếp dữ liệu, tuy nhiên điều đó có thể gây ra một số hậu quả như mã nguồn truy cập trùng lặp, tiềm năng xảy ra lỗi trong quá trình lập trình, kiểu dữ liệu yếu, gây khó khăn khi triển khai các chính sách tập trung hóa dữ liệu như bộ nhớ đệm, khó tách biệt tầng Business Logic với các phần phụ thuộc bên ngoài. Vì vậy, người ta thường sử dụng Repository Pattern để tách biệt rõ ràng giữa logic truy cập dữ liệu và logic nghiệp vụ, giúp mã nguồn dễ đọc, dễ bảo trì và kiểm thử khi cần thiết. Hơn nữa, mẫu thiết kế này giảm sự phụ thuộc vào công nghệ cụ thể, cho phép thay đổi cách truy cập dữ liệu mà không ảnh hưởng đến ứng dụng. Repository Pattern tổ chức mã nguồn có cấu trúc, hỗ trợ tái sử dụng, quản lý dễ dàng, và nâng cao tính linh hoạt, mở rộng và bảo trì của ứng dụng. Các mục tiêu chính của Repository Pattern đó là:

- Tối đa hóa lượng mã có thể được kiểm thử tự động và cô lập tầng dữ liệu để hỗ trợ kiểm thử đơn vị.
- Truy cập nguồn dữ liệu từ nhiều vị trí khác nhau và muốn áp dụng các quy tắc truy cập và logic nhất quán, được quản lý tập trung.
- Triển khai và tập trung hóa chiến lược lưu trữ tạm (caching) cho nguồn dữ liệu.
- Cải thiện khả năng bảo trì và tính dễ đọc của mã bằng cách tách biệt logic nghiệp vụ khỏi logic truy cập dữ liệu hoặc dịch vụ.
- Sử dụng các thực thể nghiệp vụ có kiểu dữ liệu mạnh (strongly typed) để có thể phát hiện các vấn đề trong quá trình biên dịch thay vì trong thời gian chạy.
- Liên kết một hành vi cụ thể với dữ liệu liên quan. Ví dụ, tính toán các trường dữ liệu hoặc thực thi các mối quan hệ phức tạp hoặc quy tắc business (nghiệp vụ) giữa các yếu tố dữ liệu trong một thực thể.
- Áp dụng mô hình miền để đơn giản hóa logic nghiệp vụ phức tạp.

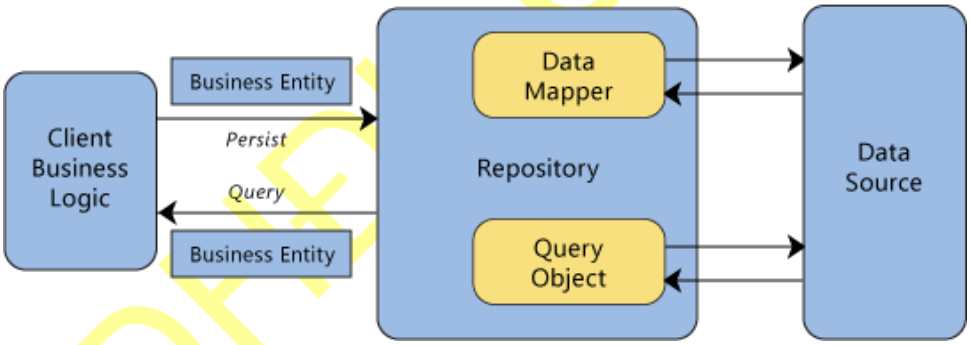
B. Hướng dẫn thực hànhb

1. Cách hoạt động của mẫu Repository

Repository Pattern tách biệt logic truy xuất và ánh xạ dữ liệu khỏi logic nghiệp vụ. Nó làm trung gian giữa tầng nguồn dữ liệu (như cơ sở dữ liệu, danh sách SharePoint, hoặc dịch vụ Web) và các lớp nghiệp vụ, xử lý việc truy vấn dữ liệu, ánh xạ vào thực thể nghiệp vụ, và lưu trữ thay đổi. Điều này giúp logic nghiệp vụ không phụ thuộc vào nguồn dữ liệu cụ thể. Sự tách biệt giữa tầng dữ liệu (data access layer) và tầng nghiệp vụ (business logic layer) có ba lợi ích:

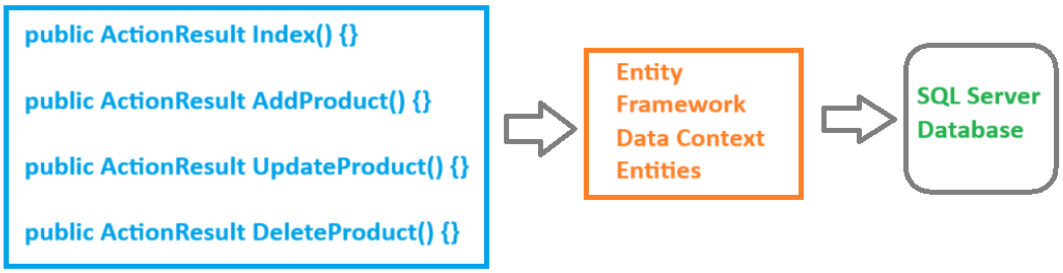
- Tập trung hóa logic dữ liệu hoặc logic truy cập dịch vụ Web.
- Cung cấp điểm thay thế cho các kiểm thử đơn vị.
- Cung cấp một kiến trúc linh hoạt có thể được điều chỉnh khi thiết kế tổng thể của ứng dụng phát triển.

Có hai cách mà Repository Pattern có thể truy vấn các thực thể nghiệp vụ. Một là gửi một đối tượng truy vấn đến logic nghiệp vụ của khách hàng, hoặc hai là sử dụng các phương thức xác định các tiêu chí nghiệp vụ. Trong trường hợp thứ hai, Repository Pattern tạo truy vấn thay mặt cho khách hàng. Repository Pattern trả về một tập hợp các thực thể phù hợp với truy vấn.



Hình trên minh họa sự tương tác của Repository Pattern với khách hàng và nguồn dữ liệu. Theo đó, Repository đóng vai trò là trung gian giữa nguồn dữ liệu và các thực thể nghiệp vụ, sử dụng đối tượng ánh xạ dữ liệu (Data Mapper) và truy vấn dữ liệu (Query Object) để tương tác với nguồn dữ liệu và chuyển dữ liệu từ nguồn dữ liệu đến các thực thể nghiệp vụ hoặc ngược lại.

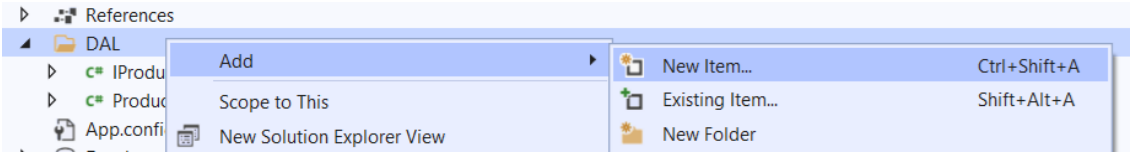
Trong ví dụ sau, Repository Pattern được áp dụng để hoạt động trung gian giữa cơ sở dữ liệu SQL và lớp **ProductController** (khung xanh nhạt) theo mô hình như sau.



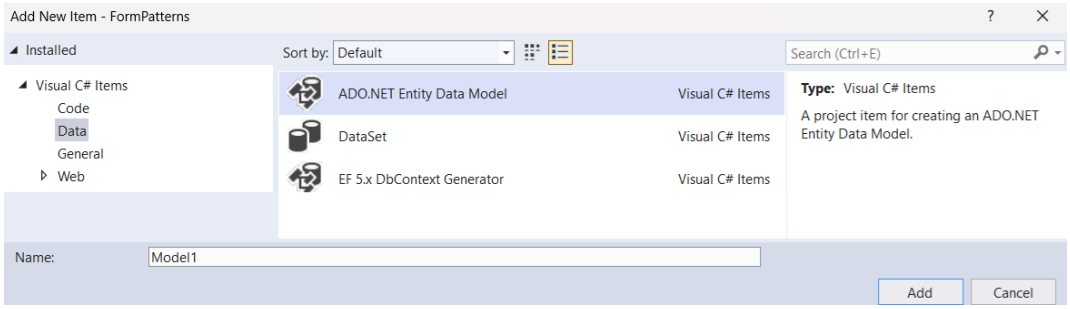
2. Xây dựng mẫu Repository

Trước hết, chúng ta sẽ thực hiện các thao tác tương tác với dữ liệu ở bảng Food. Tuy nhiên thay vì sử dụng cách kết nối database truyền thống, chúng ta sẽ tạo một mô hình dữ liệu ADO.NET và truy vấn bằng LINQ. Cách làm này nhanh hơn vì Visual Studio sẽ tự động tạo kết nối và gieo các lớp mã nguồn cần thiết để thực hiện truy vấn.

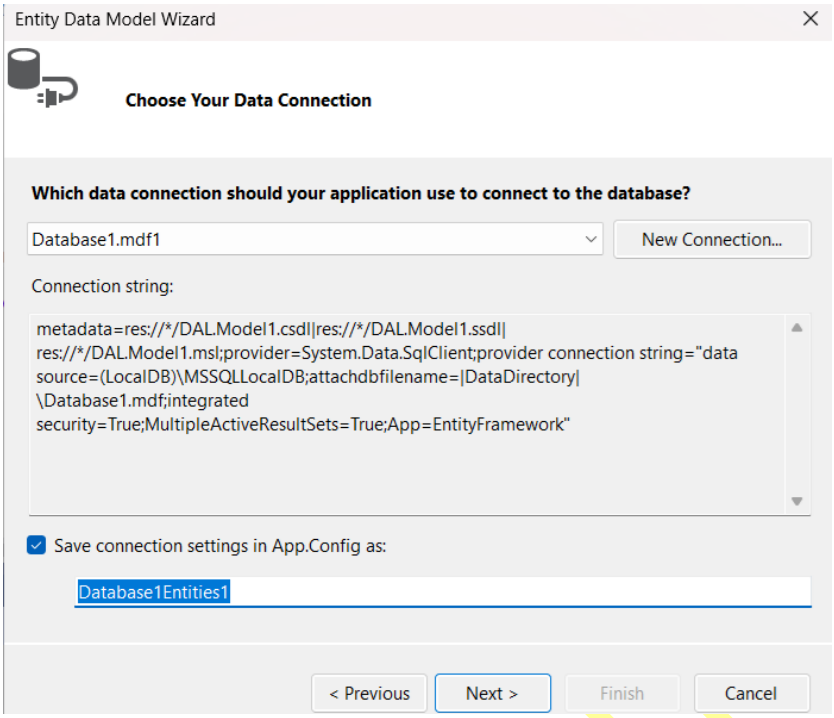
Đầu tiên, ở khung duyệt dự án, tạo 1 thư mục mới tên là **DAL**, nhấp chuột chọn thư mục đó và chuột phải chọn **Add->New Item** để tạo mô hình ADO.NET Entity Data Model như hình sau:



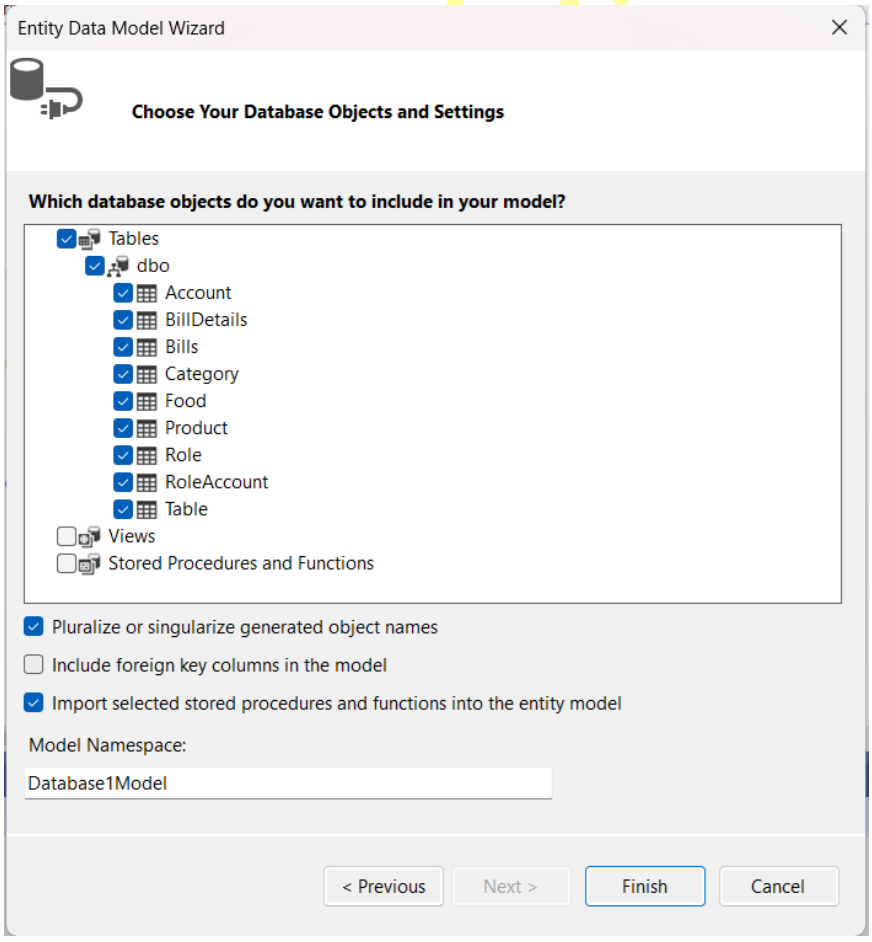
Sau đó, chọn thêm mới 1 mô hình dữ liệu thực thể ADO.NET (**ADO.NET Entity Data Model**) và đặt tên là **Model1** như hình sau.



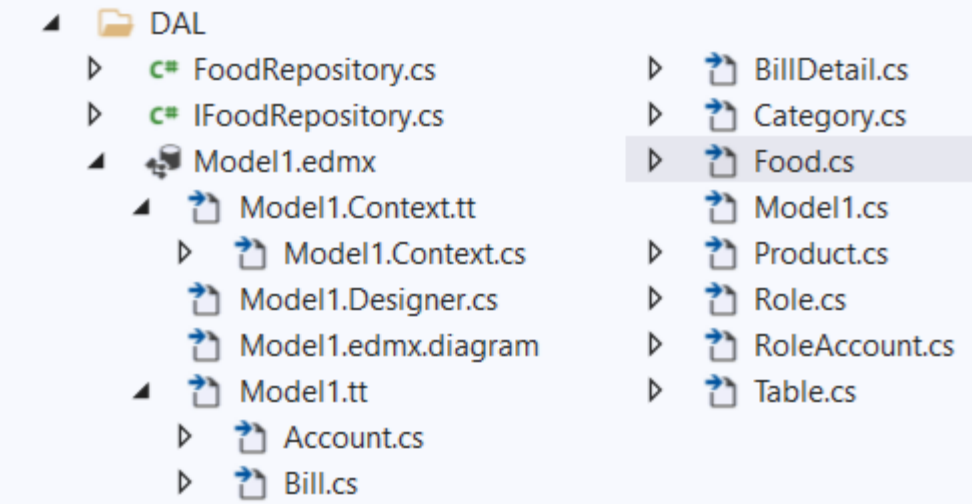
Tiếp theo chọn mục **EF Designer from database**, chọn tên database tương ứng và đặt tên cho kết nối là **Database1Entities1**. Lưu ý bạn có thể đặt bất kỳ tên gì cho kết nối Database.



Tiếp theo, bạn chọn bảng để kết nối với database, và đặt tên namespace là **Database1Model**.



Sau khi thực hiện xong các bước kết nối với database, Visual Studio sẽ tự động gieo mã nguồn, trong đó có tập tin **Model1.edmx** với cấu trúc thư mục như hình sau.



Theo đó, khi mở tập tin **Food.cs**, chúng ta sẽ có nội dung do Visual Studio dự gieo như sau:

```
namespace FormPatterns.DAL
{
    using System;
    using System.Collections.Generic;

    5 references
    public partial class Food
    {
        0 references
        public Food()
        {
            this.BillDetails = new HashSet<BillDetail>();
        }

        0 references
        public int ID { get; set; }
        0 references
        public string Name { get; set; }
        0 references
        public string Unit { get; set; }
        0 references
        public int Price { get; set; }
        0 references
        public string Notes { get; set; }

        1 reference
        public virtual ICollection<BillDetail> BillDetails { get; set; }
        0 references
        public virtual Category Category { get; set; }
    }
}
```

Tương tự, khi mở tập tin **Model1.Context.cs**, chúng ta cũng có nội dung tập tin được Visual Studio tự gieo như sau:

```
namespace FormPatterns.DAL
{
    using System;
    using System.Data.Entity;
    using System.Data.Entity.Infrastructure;

    1 reference
    public partial class Database1Entities1 : DbContext
    {
        0 references
        public Database1Entities1()
            : base("name=Database1Entities1")
        {
        }

        0 references
        protected override void OnModelCreating(DbModelBuilder modelBuilder)
        {
            throw new UnintentionalCodeFirstException();
        }

        0 references
        public virtual DbSet<Account> Accounts { get; set; }
        0 references
        public virtual DbSet<BillDetail> BillDetails { get; set; }
        0 references
        public virtual DbSet<Bill> Bills { get; set; }
        0 references
        public virtual DbSet<Category> Categories { get; set; }
        0 references
        public virtual DbSet<Food> Foods { get; set; }
        0 references
        public virtual DbSet<Product> Products { get; set; }
        0 references
        public virtual DbSet<Role> Roles { get; set; }
        0 references
        public virtual DbSet<RoleAccount> RoleAccounts { get; set; }

        public virtual DbSet<Table> Tables { get; set; }
    }
}
```

Tiếp theo, chúng ta tiến hành xây dựng Repository Pattern. Chọn folder DAL ở thanh dự án, chuột phải chọn Add -> New Item, chọn 1 tập tin mã nguồn mới tên là **IFoodRepository.cs** với nội dung như sau:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace FormPatterns.DAL
{
    1 reference
    public interface IFoodRepository
    {
        2 references
        IEnumerable<Food> GetAll();
        1 reference
        Food GetById(int FoodID);
        1 reference
        void Insert(Food food);
        1 reference
        void Update(Food food);
        1 reference
        void Delete(int FoodID);
        1 reference
        void Save();
    }
}
```

Lúc này, chúng ta có thể thấy lớp IProductRepository là 1 lớp interface (giao diện) chứa các phương thức cần thiết để tương tác với cơ sở dữ liệu, và mỗi phương thức sẽ không chứa bất kỳ dòng code nào.

Tiếp theo, tạo 1 tập tin mã nguồn **FoodRepository.cs** (chọn thư mục DAL, Add -> New Item) để kế thừa lớp IFoodRepository và tiến hành viết mã nguồn tương tác giữa cơ sở dữ liệu và thành phần khác trong 1 dự án. Tập tin FoodRepository.cs với biến **_context** có kiểu **DbContext** như sau:

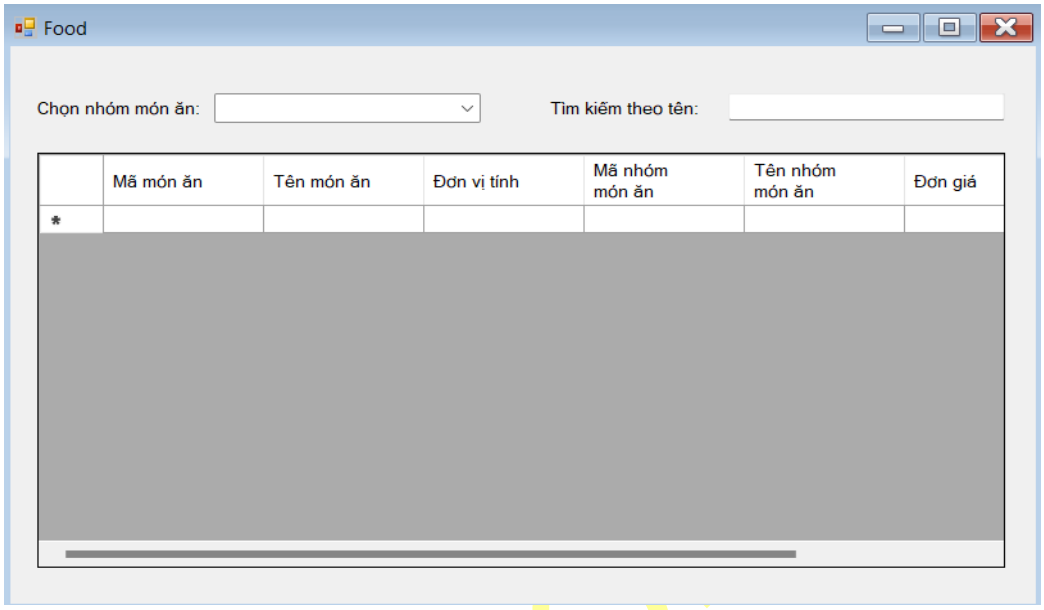
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Data.Entity;

namespace FormPatterns.DAL
{
    4 references
    public class FoodRepository : IFoodRepository
    {
        //The following variable is going to hold the Database1Entities1 instance
        private readonly Database1Entities1 _context;
        //Initialzing the Database1Entities1 instance
        1 reference
        public FoodRepository()
        {
            _context = new Database1Entities1();
        }
        //Initializing the Database1Entities1 instance which it received as an argument
        1 reference
        public FoodRepository(Database1Entities1 context)
        {
            _context = context;
        }
        //This method will return all the Foods from the Food table
        1 reference
        public IEnumerable<Food> GetAll()
        {
            return _context.Foods.ToList();
        }
        //This method will return one Food's information from the Food table
        //based on the FoodID which it received as an argument
        0 references
        public Food GetById(int FoodID)
        {
            return _context.Foods.Find(FoodID);
        }
        //This method will Insert one Food object into the Food table
        //It will receive the Food object as an argument which needs to be inserted into the database
        0 references
        public void Insert(Food Food)
        {
            //The State of the Entity is going to be Added State
            _context.Foods.Add(Food);
        }
        //This method is going to update the Food data in the database
        //It will receive the Food object as an argument
        0 references
```

```
        //This method is going to update the Food data in the database
        //It will receive the Food object as an argument
        0 references
        public void Update(Food Food)
        {
            //It will mark the Entity State as Modified
            _context.Entry(Food).State = EntityState.Modified;
        }
        //This method is going to remove the Food Information from the Database
        //It will receive the FoodID as an argument whose information needs to be removed from the database
        1 reference
        public void Delete(int FoodID)
        {
            //First, fetch the Food details based on the FoodID id
            Food Food = _context.Foods.Find(FoodID);
            //If the Food object is not null, then remove the Food
            if (Food != null)
            {
                //This will mark the Entity State as Deleted
                _context.Foods.Remove(Food);
            }

        }
        //This method will make the changes permanent in the database
        //That means once we call Insert, Update, and Delete Methods, then we need to call
        //the Save method to make the changes permanent in the database
        1 reference
        public void Save()
        {
            //Based on the Entity State, it will generate the corresponding SQL Statement and
            //Execute the SQL Statement in the database
            //For Added Entity State: It will generate INSERT SQL Statement
            //For Modified Entity State: It will generate UPDATE SQL Statement
            //For Deleted Entity State: It will generate DELETE SQL Statement
            _context.SaveChanges();
        }
        private bool disposed = false;
        //As a context object is a heavy object or you can say time-consuming object
        //So, once the operations are done we need to dispose of the same using Dispose method
        //The Database1Entities1 class inherited from DbContext class and the DbContext class
        //is Inherited from the IDisposable interface
        1 reference
        protected virtual void Dispose(bool disposing)
        {
            if (!this.disposed)
            {
                if (disposing)
                {
                    _context.Dispose();
                }
            }
            this.disposed = true;
        }
        0 references
        public void Dispose()
        {
            Dispose(true);
            GC.SuppressFinalize(this);
        }

    }
}
```

Lưu ý có thể sử dụng lớp FoodRepository hoặc dùng truy vấn LINQ trực tiếp, thậm chí dùng cả 2 phương pháp để tương tác với database và các thành phần khác trong dự án. Ở các dự án phần mềm thực tế, việc sử dụng cách nào để tương tác với database tùy thuộc theo quy ước của người quản lý hoặc của công ty yêu cầu. Ví dụ, chúng ta có thể lấy tất cả dữ liệu của bảng Food thông qua lớp FoodRepository với 2 dòng truy vấn đơn giản như sau:

```
var productRepo = new ProductRepository(new Database1Entities1());
var result = productRepo.GetAll();
```

Để minh họa trực quan, tiếp theo bạn hãy tạo 1 Form mới tên là **Food.cs**, sau đó thêm 1 control DataGridView, 1 ComboBox (**cbbCategory**), 1 trường Text (**txtByCategoryName**), và 2 label tương ứng với giao diện sau.



Sau đó, nhấp chuột lên các controller dataGridView1, cbbCategory và txtByCategoryName để tạo các phương thức sự kiện trong mã nguồn Form **Food.cs** với nội dung như sau.

```
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 FormPatterns.DAL;

namespace FormPatterns
{
    3 references
    public partial class Food : Form
    {
        Database1Entities1 db_context;
        1 reference
        public Food()
        {
            InitializeComponent();
            db_context = new Database1Entities1();

        }
```

```
2 references
private void LoadData()
{
    // Cách 1: Dùng phương thức GetAllWithCategory()
    // ở lớp FoodRepository
    var foodRepo = new FoodRepository(db_context);
    var result = foodRepo.GetAllWithCategory();

    // Cách 2 dùng LINQ trực tiếp
    //var result = db_context.Foods.Select(f => new
    //{
    //    f.ID,
    //    f.Name,
    //    f.Unit,
    //    CategoryID = f.Category.ID,
    //    CategoryName = f.Category.Name,
    //    f.Price,
    //   f.Notes,
    //}).ToList();

    dataGridView1.DataSource = result;
}
1 reference
private void Food_Load(object sender, EventArgs e)
{
    // Dùng LINQ trực tiếp
    var categories = db_context.Categories
                    .Select(c => new { c.ID, c.Name })
                    .ToList();
    categories.Insert(0, new { ID = 0, Name = "Tất cả" });

    cbbCategory.DataSource = categories;
    LoadData();
}
1 reference
private void textBox1_TextChanged(object sender, EventArgs e)
{
    string Name = txtSearchByName.Text;

    //Cách 1: Dùng phương thức GetByName ở lớp FoodRepository
    var foodRepo = new FoodRepository(db_context);
    var result = foodRepo.GetByName(Name);

    //Cách 2: Dùng LINQ trực tiếp
    //var result = db_context.Foods.Where(f => f.Name.Contains(Name))
    //          .ToList();

    dataGridView1.DataSource = result;
}
```

```
1 reference
private void cbbCategory_SelectedIndexChanged(object sender, EventArgs e)
{
    // Indexes start with 0
    int category = cbbCategory.SelectedIndex;
    if (category == 0)
    {
        LoadData();
        return;
    }

    // Cách 1:  Dùng phương thức GetByCategory ở lớp FoodRepository
    var foodRepo = new FoodRepository(db_context);
    var result = foodRepo.GetByCategory(category);

    // Cách 2: Dùng LINQ trực tiếp
    //var result = db_context.Foods.
    //    Where(f => f.Category.ID == category).
    //    Select(f => new
    //{
    //    f.ID,
    //   f.Name,
    //    f.Unit,
    //    CategoryID = f.Category.ID,
    //    CategoryName = f.Category.Name,
    //    f.Price,
    //    f.Notes,
    //}).ToList();

    dataGridView1.DataSource = result;
}
```

Lưu ý ở các phương thức LoadData(), textBox1_TextChanged(), cbbCategory_SelectedIndexChanged(), bạn có thể các phương thức ở lớp FoodRepository (tập tin **FoodRepository.cs**) có nội dung như sau:

```
//This method will return all the Foods from the Food table
1 reference
public IEnumerable<object> GetAllWithCategory()
{
    return _context.Foods.Select(f => new
    {
        f.ID,
        f.Name,
        f.Unit,
        CategoryID = f.Category.ID,
        CategoryName = f.Category.Name,
        f.Price,
        f.Notes,
    }).ToList();

}
```

```
        //This method will return all the Foods by a given CategoryID
        1 reference
        public IEnumerable<object> GetByCategory(int CategoryID)
        {
            return _context.Foods.Where(f => f.Category.ID == CategoryID).
                Select(f => new
                {
                    f.ID,
                    f.Name,
                    f.Unit,
                    CategoryID = f.Category.ID,
                    CategoryName = f.Category.Name,
                    f.Price,
                    f.Notes,
                }).ToList();
        }

        //This method will return all the Foods by a given Name
        1 reference
        public IEnumerable<Food> GetByName(string Name)
        {
            return _context.Foods.
                Where(f => f.Name.Contains(Name)).
                ToList();
        }
```

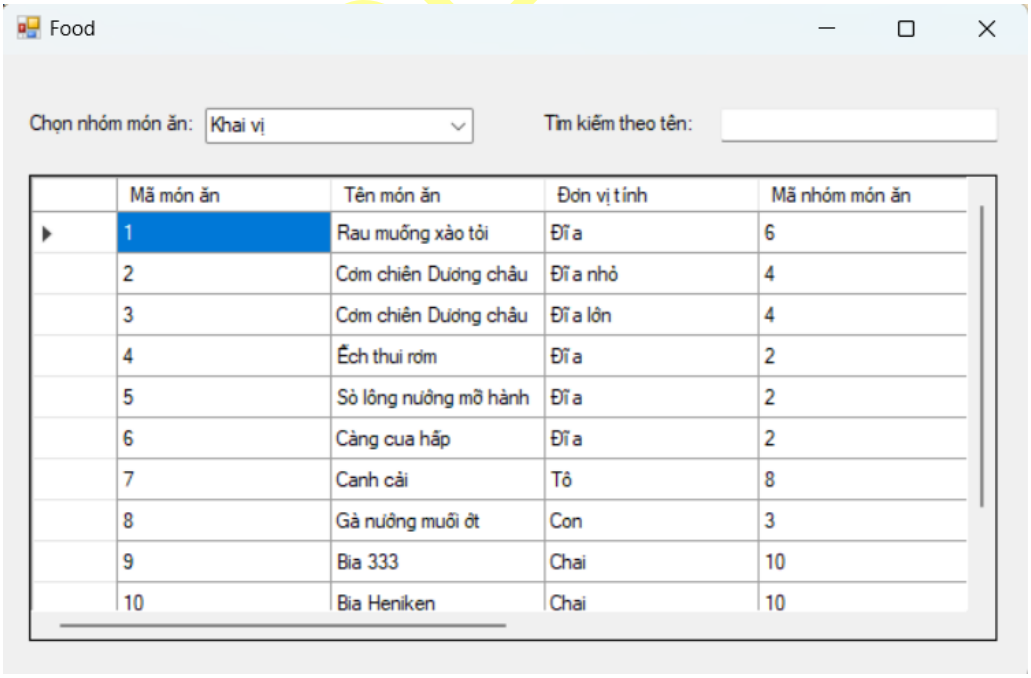
Trong tập tin **Food.Designer.cs**, 2 controller cbbCategory và textSearchByName có nội dung như sau:

```
//
// cbbCategory
//
this.cbbCategory.DisplayMember = "Name";
this.cbbCategory.FormattingEnabled = true;
this.cbbCategory.Location = new System.Drawing.Point(159, 38);
this.cbbCategory.Name = "cbbCategory";
this.cbbCategory.Size = new System.Drawing.Size(208, 24);
this.cbbCategory.TabIndex = 3;
this.cbbCategory.ValueMember = "ID";
this.cbbCategory.SelectedIndexChanged +=
    new System.EventHandler(this.cbbCategory_SelectedIndexChanged);
//
// txtSearchByName
//
this.txtSearchByName.Location = new System.Drawing.Point(560, 38);
this.txtSearchByName.Name = "txtSearchByName";
this.txtSearchByName.Size = new System.Drawing.Size(216, 22);
this.txtSearchByName.TabIndex = 4;
this.txtSearchByName.TextChanged +=
    new System.EventHandler(this.textBox1_TextChanged);
```

Cũng trong tập tin **Food.Designer.cs**, bạn cần tùy chỉnh thứ tự các cột trong controller dataGridView1 tương ứng với dữ liệu lấy ra từ truy vấn như sau:

```
//
// Column1
//
this.Column1.DataPropertyName = "ID";
this.Column1.HeaderText = "Mã món ăn";
this.Column1.MinimumWidth = 6;
this.Column1.Name = "Column1";
this.Column1.Width = 125;
//
// Column2
//
this.Column2.DataPropertyName = "Name";
this.Column2.HeaderText = "Tên món ăn";
this.Column2.MinimumWidth = 6;
this.Column2.Name = "Column2";
this.Column2.Width = 125;
//
// Column3
//
this.Column3.DataPropertyName = "Unit";
this.Column3.HeaderText = "Đơn vị tính";
this.Column3.MinimumWidth = 6;
this.Column3.Name = "Column3";
this.Column3.Width = 125;
//
// Column4
//
this.Column4.DataPropertyName = "CategoryID";
this.Column4.HeaderText = "Mã nhóm món ăn";
this.Column4.MinimumWidth = 6;
this.Column4.Name = "Column4";
this.Column4.Width = 125;
//
// Column5
//
this.Column5.DataPropertyName = "CategoryName";
this.Column5.HeaderText = "Tên nhóm món ăn";
this.Column5.MinimumWidth = 6;
this.Column5.Name = "Column5";
this.Column5.Width = 125;
//
// Column6
//
this.Column6.DataPropertyName = "Unit";
this.Column6.HeaderText = "Đơn giá";
this.Column6.MinimumWidth = 6;
this.Column6.Name = "Column6";
this.Column6.Width = 125;
//
// Column7
//
this.Column7.DataPropertyName = "Notes";
this.Column7.HeaderText = "Ghi chú";
this.Column7.MinimumWidth = 6;
this.Column7.Name = "Column7";
```

Cuối cùng, nhấn F5 để chạy thì Form Food có giao diện như sau:



Nếu có thay đổi về kiến trúc dữ liệu Database hoặc cách lấy dữ liệu, chúng ta chỉ cần sửa chữa các phương thức kết nối trong lớp ProductRepository.cs và IProductRepository.cs như đã nêu.

Như vậy, Repository Pattern giúp can thiệp vào việc tương tác với cơ sở dữ liệu thông qua một lớp trung gian chứa các phương thức truy vấn dữ liệu khác nhau. Lớp repository này không chỉ thực hiện các thao tác CRUD (tạo, đọc, cập nhật, xóa) mà còn cung cấp các phương thức tùy chỉnh để truy vấn dữ liệu theo các tiêu chí nghiệp vụ cụ thể. Điều này cho phép mã nguồn quản lý các thao tác dữ liệu một cách có cấu trúc, giảm thiểu sự phụ thuộc vào chi tiết triển khai của cơ sở dữ liệu và dễ dàng thay đổi hoặc mở rộng các chức năng truy vấn mà không ảnh hưởng đến các phần khác của ứng dụng.

C. Bài tập

Xây dựng Repository Pattern cho bảng Food, Category theo hướng dẫn trong bài. Sau đó thêm 1 số chức năng:

1. Đếm tổng số Food có trong bảng Food và hiển thị thông qua 1 controller Label.

2. Đếm tổng số Category có trong bảng Category và hiển thị thông qua 1 controller Label.

CHỦ ĐỀ 9. CASE STUDY ỨNG DỤNG QUẢN LÝ NHÀ HÀNG

A. Mục tiêu

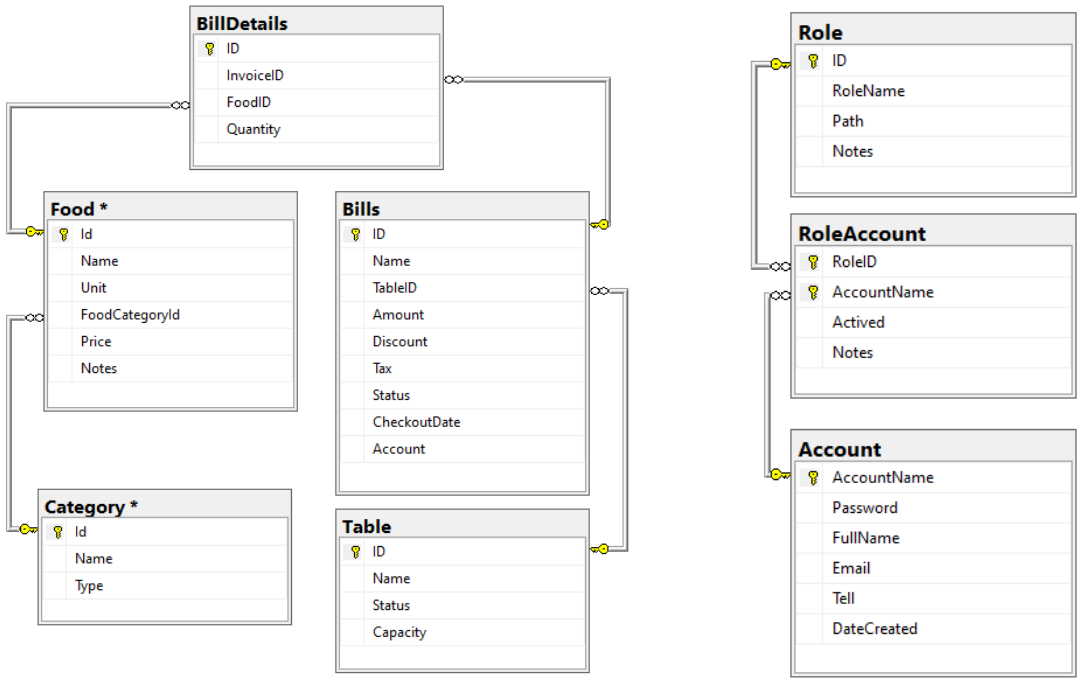
Chủ đề này này sẽ giúp sinh viên xây dựng một trường hợp cụ thể là ứng dụng quản lý nhà hàng hoàn chỉnh với cơ sở dữ liệu được cho như trong chủ đề 1. Bài hướng dẫn này bao gồm các nội dung sau đây:

- Xây dựng cơ sở dữ liệu
- Xây dựng các tầng trong mô hình n-tier
- Xây dựng phần giao diện quản lý nhà hàng
- Xây dựng phần phân quyền
- Triển khai ứng dụng

B. Hướng dẫn thực hành

1. Xây dựng cơ sở dữ liệu

Cơ sở dữ liệu ở đây chúng tôi sử dụng là RestaurantManagement (Quản lý nhà hàng) được cho trong chủ đề 1 với hai lược đồ quan hệ là Chức năng và Phân quyền như sau:



Các thủ tục được sử dụng trong bài hướng dẫn bao gồm tên, chức năng và tham số như trong bảng sau đây:

Tên thủ tục	**Chức năng**	**Tham số**
Find	Tìm kiếm thông tin theo bảng	*TableName*: Tên của bảng *ColumnName*: Tên cột tìm kiếm *Key*: Từ khóa tìm kiếm
GetAll	Thủ tục lấy hết dữ liệu trong một bảng	*TableName*: Tên của bảng *OrderBy*: Kiểu sắp xếp (asc, desc)
TotalRowCount	Thủ tục dùng để đếm số lượng mẫu tin trong một bảng theo bộ lọc	*TableName*: Tên của bảng *Filter*: Bộ lọc
InsertUpdateDelete_Table	Thủ tục dùng để thêm, xóa, sửa nội dung bảng Table	Toàn bộ các trường của bảng Table
InsertUpdateDelete_Bills	Thủ tục dùng để thêm, xóa, sửa nội dung bảng Bills	Toàn bộ các trường của bảng Bills
InsertUpdateDelete_BillDetails	Thủ tục dùng để thêm, xóa, sửa nội dung bảng BillsDetails	Toàn bộ các trường của bảng BillsDetails

InsertUpdateDelete_Food	Thủ tục dùng để thêm, xóa, sửa nội dung bảng Food	Toàn bộ các trường của bảng Food
InsertUpdateDelete_Category	Thủ tục dùng để thêm, xóa, sửa nội dung bảng Category	Toàn bộ các trường của bảng Category
InsertUpdateDelete_Role	Thủ tục dùng để thêm, xóa, sửa nội dung bảng Role	Toàn bộ các trường của bảng Role
InsertUpdateDelete_Account	Thủ tục dùng để thêm, xóa, sửa nội dung bảng Account	Toàn bộ các trường của bảng Account
InsertUpdateDelete_RoleAccount	Thủ tục dùng để thêm, xóa, sửa nội dung bảng RoleAccount	Toàn bộ các trường của bảng RoleAccount

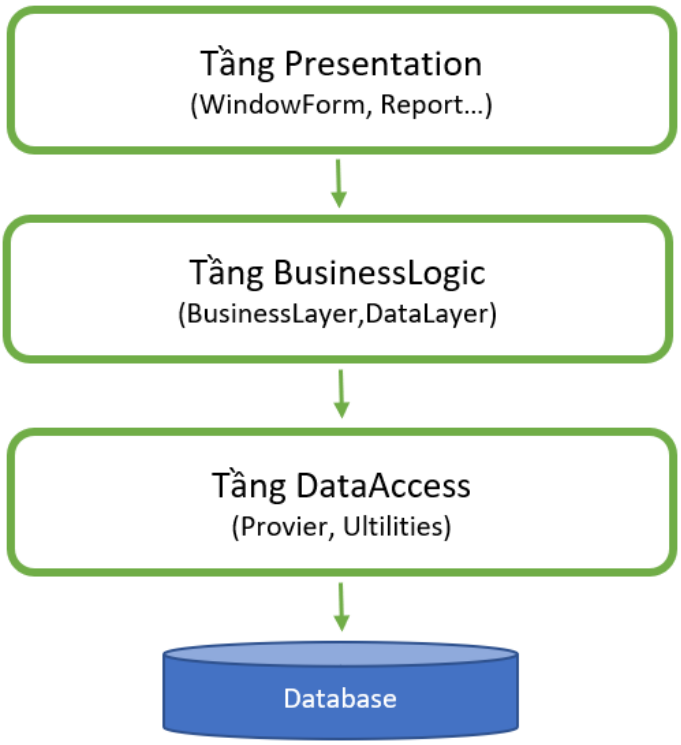
Lưu ý: Để dễ xử lý, chúng tôi gom chung các nội dung thêm, xóa, sửa thành một thủ tục duy nhất là *InsertUpdateDelete_[Tên bảng]*, sau đó dùng biến Action để kiểm soát. Ví dụ cho thủ tục thêm, xóa, sửa bảng Category:

```
CREATE PROCEDURE [dbo].[InsertUpdateDelete_Category]
    @Id int output,
    @Name nvarchar(250),
    @Type int,
    @Action int
AS
IF @Action = 0
   BEGIN
       INSERT INTO [Category] (Name],[Type])VALUES (@Name,@Type)
       SET @Id = @@identity
   END
ELSE IF @Action = 1
   BEGIN
       UPDATE [Category] SET
            [Name] = @Name,
            [Type] = @Type
       WHERE [Id] = @Id
   END
ELSE IF @Action = 2
   BEGIN
       DELETE [Category] WHERE [Id] = @Id
   END
```

Độc giả và sinh viên có thể xem thêm trong Lab 1 để biết cách tạo một cơ sở dữ liệu hoàn chỉnh với các bảng và các thủ tục.

2. Tạo mô hình các tầng trong chương trình

Chúng tôi xây dựng ứng dụng theo kiến trúc mô hình *n-tier* (tương tự như trong Lab 8). Các tầng của kiến trúc được mô tả như hình sau:



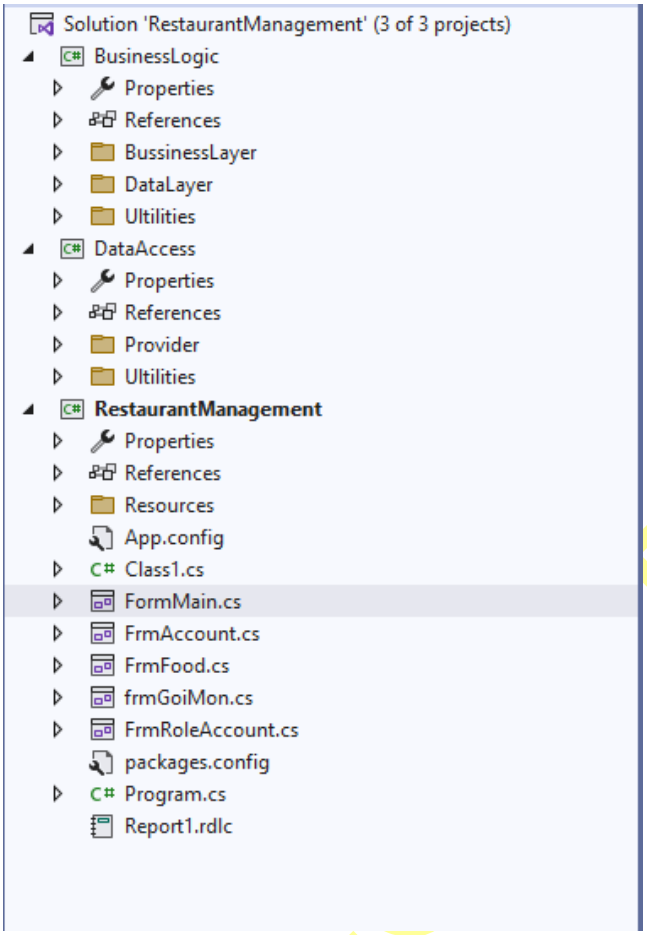
Tầng Presentation được tạo bằng cách tạo mới một ứng dụng WindowForm, đây chính là giao diện chính đầu tiên, sau này sẽ bổ sung các giao diện còn lại. Hai tầng còn lại được tạo ra bằng cách thêm Solution cho dự án bằng ClassLibrary (chi tiết xem trong Lab 8). Tiếp theo, tầng BusinessLogic chứa ba thư mục, bao gồm:

- *BussinessLayer*: Chứa các lớp quản lý các bảng dữ liệu, mỗi một bảng trong cơ sở dữ liệu sẽ được ánh xạ thành một lớp

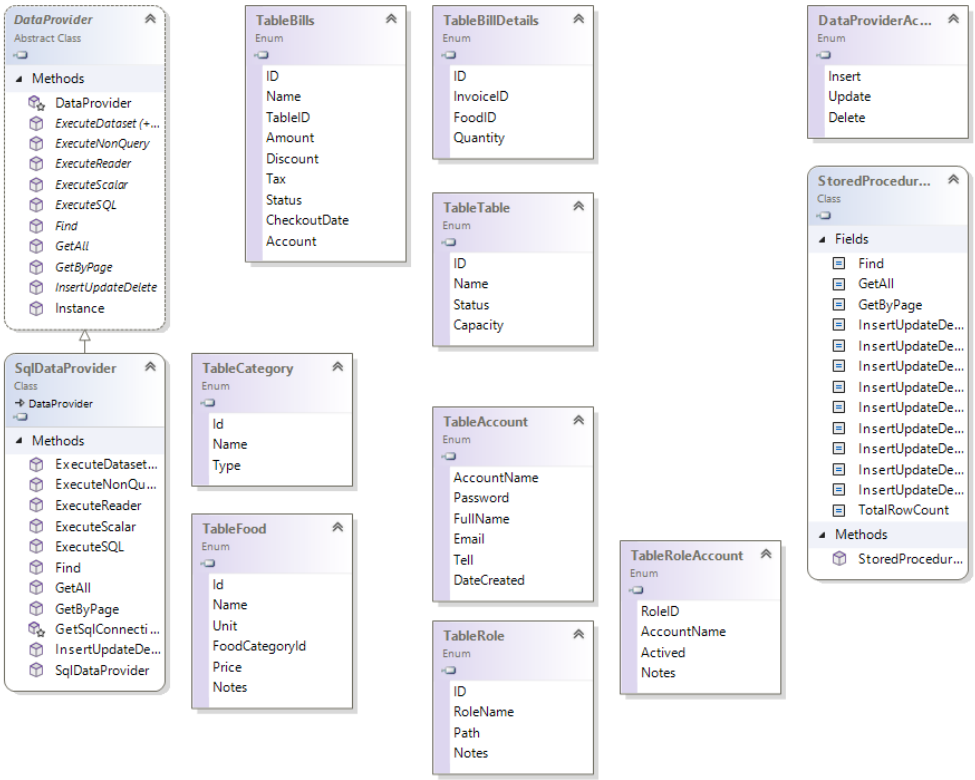
- *DataLayer*: Chứa các lớp xử lý dữ liệu, mỗi một thủ tục (StoreProcedure) trong cơ sở dữ liệu sẽ tương ứng với một phương thức trong từng lớp.

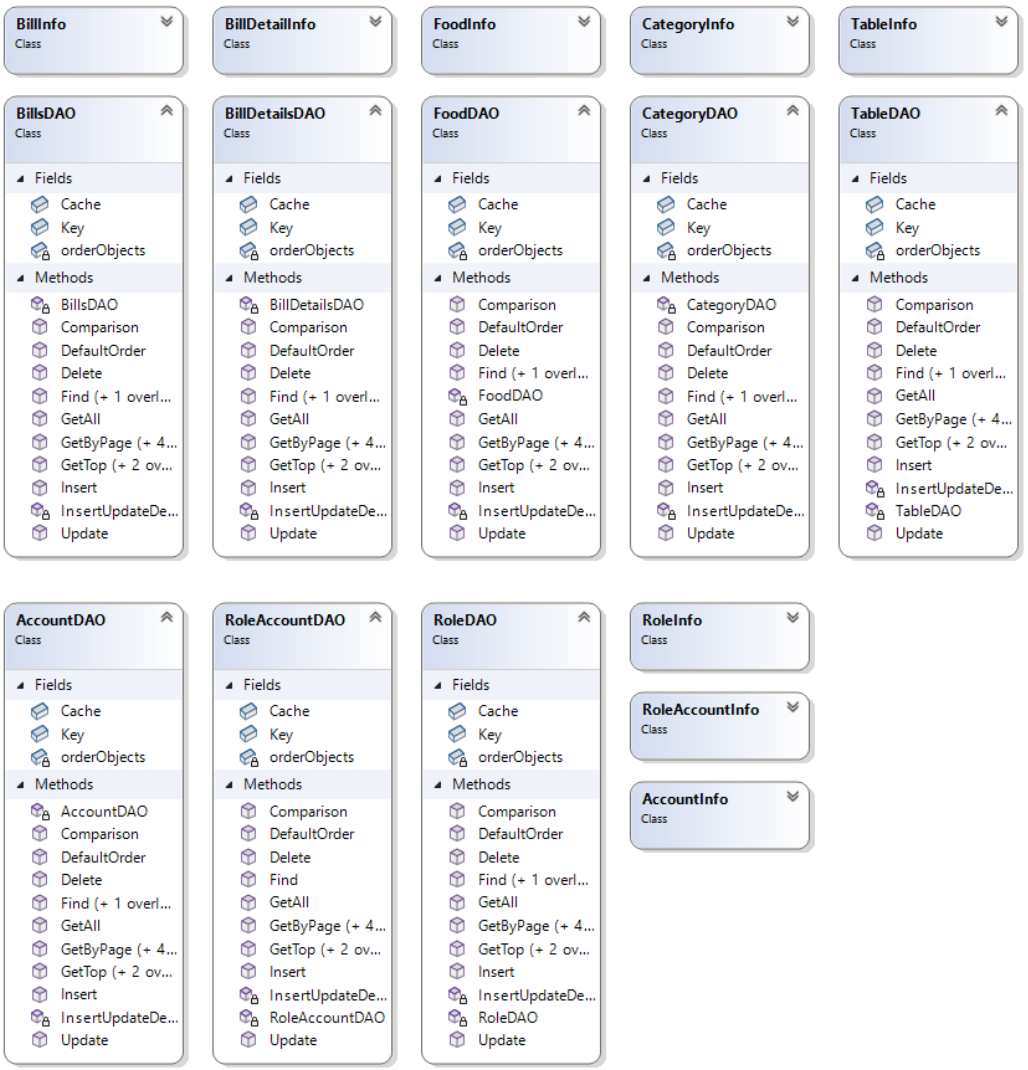
- *Ultilities*: Chứa các lớp đặt tên cho bảng hoặc đặt tên cho các thủ tục

Còn lại là tầng DataAccess, tầng này có các lớp tổng quát dùng để kết nối và truy xuất, xử lý dữ liệu ở mức tổng quát. Các tầng được biểu diễn trong dự án của WindowForm như sau:



Hai hình sau đây mô tả phần Class Diagram của tầng DataAccess và tầng BusinessLogic.





3. Xây dựng chương trình tầng Presentation

Tầng Presentation được xây dựng bằng cách xây dựng một Form chính bao gồm các thẻ, mỗi thẻ đảm nhiệm một chức năng nhất định. Các chức Mô đun của giao diện chính bao gồm:

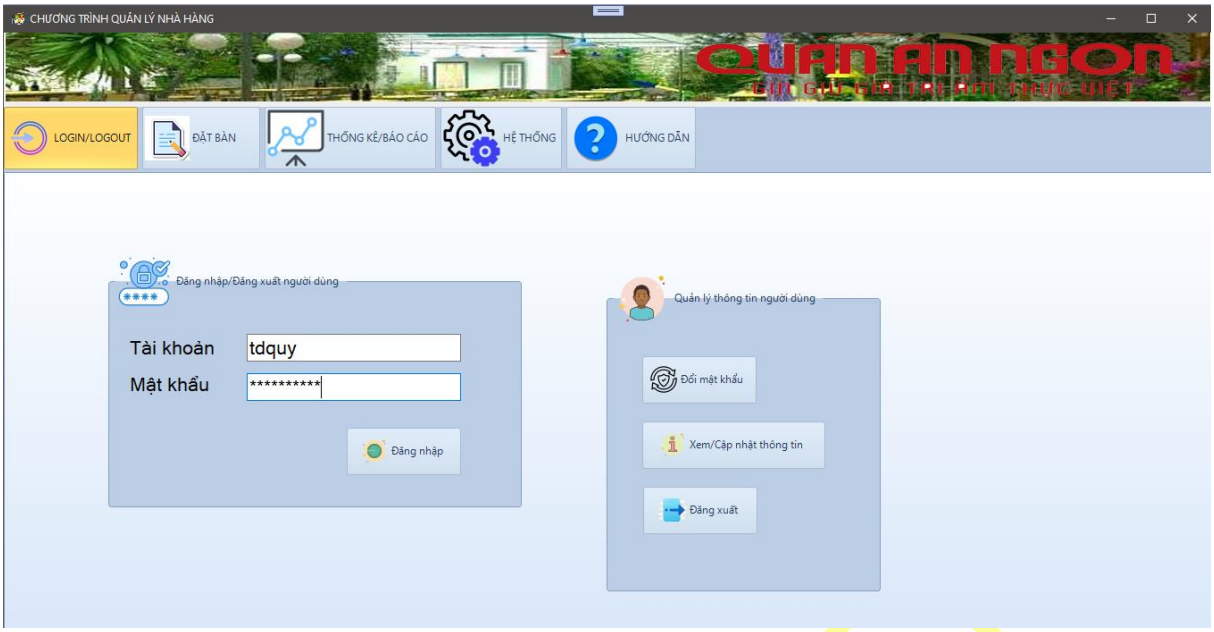
- ***Mô đun Login/Logout***: Bao gồm các chức năng như Đăng nhập, đăng xuất, đổi mật khẩu, xem thông tin tài khoản…

- ***Mô đun Đặt bàn***: Bao gồm các chức năng quản lý bàn ăn uống của khách như lập bàn, gọi món, thanh toán, in hóa đơn…

- ***Mô đun Thống kê, báo cáo***: Bao gồm các chức năng thống kê báo cáo doanh thu của quán theo ngày, tuần, tháng hoặc năm.

- ***Mô đun Hệ thống***: Bao gồm các chức năng liên quan đến quản trị hệ thống cơ sở dữ liệu như thêm xóa, sửa các bảng, hệ thống quản lý tài khoản và phân quyền.

Giao diện chương trình tổng thể như trong hình sau đây:

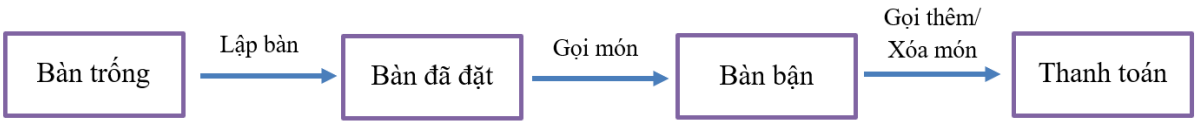


Mô đun Login/Logout: Mô đun này dùng để đăng nhập người dùng, khi người dùng nhập thông tin đăng nhập và nhấn nút, nếu đăng nhập thành công, hệ thống sẽ kiểm tra quyền của người dùng và hiển thị các chức năng tương ứng với quyền đó.

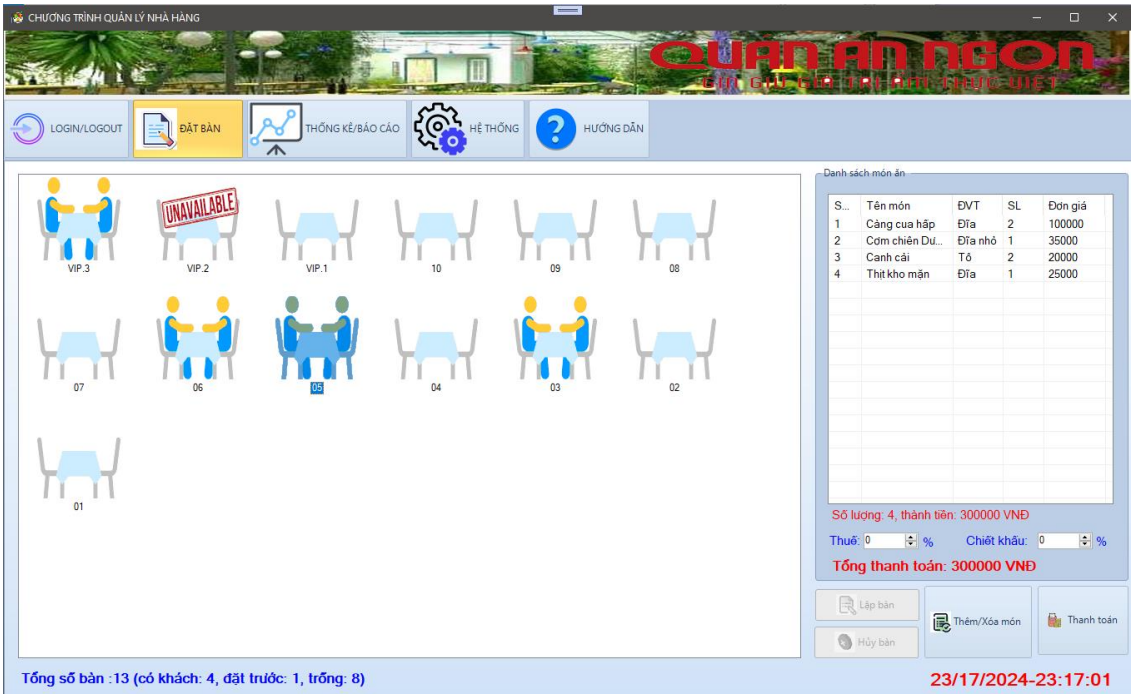
Sau khi đăng nhập xong, hệ thống sẽ lưu lại tài khoản đã đăng nhập để lưu vào cơ sở dữ liệu khi xuất hóa đơn và lưu nhật ký hệ thống. Mô đun này cũng cho phép người đã đăng nhập đổi mật khẩu, xem và cập nhật thông tin của tài khoản đăng nhập cũng như đăng xuất khỏi hệ thống.

Mô đun đặt bàn: Mô đun này cho phép nhân viên và người quản lý đặt bàn, gọi món, thêm món và thanh toán. Chương trình dùng ListView để hiển thị các bàn trong hệ thống với ba trạng thái: Trạng thái bàn trống, trạng thái bàn đã đặt chỗ và trạng thái bàn bận (đang có khách).

Quy trình đặt bàn bắt đầu khi bàn trống, nhân viên sẽ lập bàn và bắt đầu gọi món, khi có khách tới, bàn sẽ bận và có thể gọi thêm hoặc xóa món, bước cuối cùng là thanh toán. Quy trình thể hiện như hình sau đây:



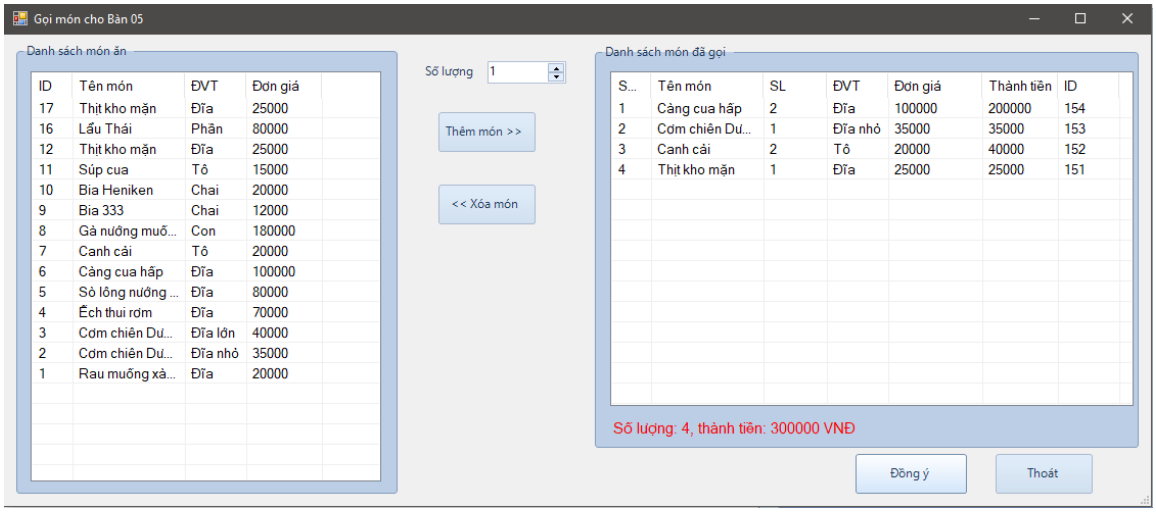
Giao diện của Mô đun này như hình sau:



Các chức năng của giao Mô đun này như sau:

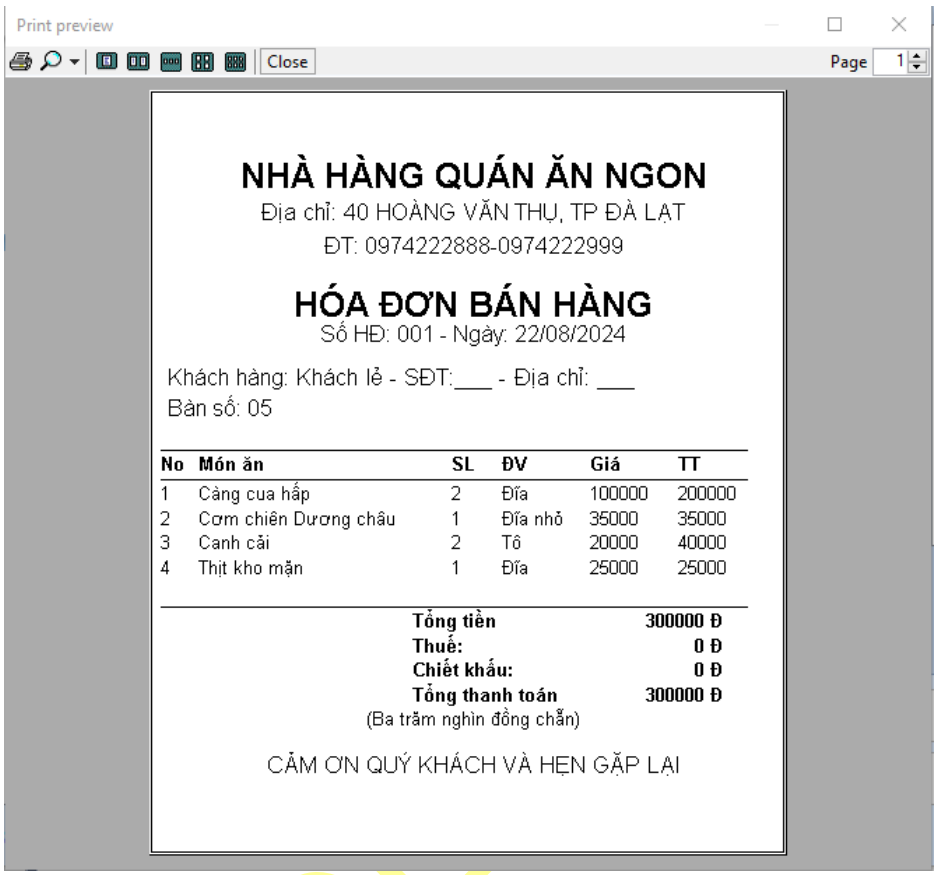
- **Lập bàn**: Chuyển đổi thông tin trường Status trong bảng Table từ 0 qua 2. Hủy bàn sẽ làm ngược lại, chuyển từ 2 về 0.

- **Thêm/Xóa món**: Sẽ cho phép gọi món, khi gọi món hệ thống sẽ thêm mới món ăn vào bảng Hóa đơn và chi tiết hóa đơn, sau khi đã gọi món, nhân viên có thể thêm bớt hoặc xóa bớt các món đã gọi. Đồng thời ở bước này, bàn sẽ được chuyển về trạng thái 1 đồng nghĩa với việc khách đã tới nên bàn sẽ có khách và bận. Giao diện phần thêm, xóa món như sau:



- **Thanh toán**: Bước thanh toán sẽ làm ba việc: 1) Chuyển bàn từ trạng thái 1 về trạng thái 0; 2) Cập nhật trạng thái Status về True (đã thanh toán) và 3) Cho phép in hóa đơn thanh toán. Phần này chúng tôi có bổ sung thêm phần thuế và chiết khấu, nếu nhân viên nhập vào ô, hệ thống sẽ cập nhật đầy đủ xuống cơ sở dữ liệu, được lưu trong bảng Bill.

Để in ra được hóa đơn, chúng tôi sử dụng công cụ PrintPreview trong WindowsForm và kết hợp với phương thức vẽ chuỗi với công cụ Graphics để vẽ ra vị trí phù hợp. Giao diện in hóa đơn như sau:



Mã nguồn sau đây thể hiện việc hóa đơn trong hình trên:

```
1 reference
private void DrawingPayment(System.Drawing.Printing.PrintPageEventArgs e)
{
    // Khai báo các đối tượng liên quan như Graphics, các loại font, các loại tham số để vẽ
    Graphics graphics = e.Graphics;
    Font font20 = new Font("Arial", 20, FontStyle.Bold);
    Font font15 = new Font("Arial", 15);
    Font font13 = new Font("Arial", 13);
    Font font10 = new Font("Arial", 10);
    float leading = 4;
    float lineheight20 = font20.GetHeight() + leading;
    float lineheight15 = font15.GetHeight() + leading;
    float lineheight13 = font13.GetHeight() + leading;
    float lineheight10 = font10.GetHeight() + leading;

    float startX = 5, startY = 50, Offset = 0;
    StringFormat formatLeft = new StringFormat(StringFormatFlags.NoClip);
    StringFormat formatCenter = new StringFormat(formatLeft);
    StringFormat formatRight = new StringFormat(formatLeft);

    formatCenter.Alignment = StringAlignment.Center;
    formatRight.Alignment = StringAlignment.Far;
    formatLeft.Alignment = StringAlignment.Near;

    SizeF layoutSize = new SizeF(500 - Offset * 2, lineheight20);
    RectangleF layout = new RectangleF(new PointF(startX, startY + Offset), layoutSize);
    Brush brush = Brushes.Black;
```

```
// Vẽ thông tin nhà hàng
string Name = "NHÀ HÀNG " + RestaurantInformation.RestaurantName.ToUpper();
graphics.DrawString(Name, font20, brush, layout, formatCenter);

string Add = RestaurantInformation.RestaurantAdd.ToUpper();
Offset = Offset + lineheight20;
layout = new RectangleF(new PointF(startX, startY + Offset), layoutSize);
graphics.DrawString("Địa chỉ: " + Add, font13, brush, layout, formatCenter);

string Phone = RestaurantInformation.RestaurantPhone.ToUpper();
Offset = Offset + lineheight15;
layout = new RectangleF(new PointF(startX, startY + Offset), layoutSize);
graphics.DrawString("ĐT: " + Phone, font13, brush, layout, formatCenter);

// Vẽ thông tin hóa đơn và khách hàng
string BillName = "Hóa đơn bán hàng".ToUpper();
Offset = Offset + lineheight20;
layout = new RectangleF(new PointF(startX, startY + Offset+5), layoutSize);
graphics.DrawString(BillName, font20, brush, layout, formatCenter);

string BillInfor = "Số HĐ: 001 - Ngày: 22/08/2024";
Offset = Offset + lineheight20;
layout = new RectangleF(new PointF(startX, startY + Offset), layoutSize);
graphics.DrawString(BillInfor, font13, brush, layout, formatCenter);

string Customer = "Khách hàng: Khách lẻ - SĐT:___ - Địa chỉ: ___ ";
Offset = Offset + lineheight20;
layout = new RectangleF(new PointF(startX+5, startY + Offset), layoutSize);
graphics.DrawString(Customer, font13, brush, layout, formatLeft);

string Table = "Bàn số: " + TableSelected.Name;
Offset = Offset + lineheight13;
layout = new RectangleF(new PointF(startX + 5, startY + Offset), layoutSize);
graphics.DrawString(Table, font13, brush, layout, formatLeft);
```

```
// Vẽ thông tin món ăn
Offset = Offset + lineheight15;
Font font10Bold = new Font(font10.FontFamily, font10.Size, FontStyle.Bold);
layout = new RectangleF(new PointF(startX, startY + Offset), layoutSize);
graphics.DrawString("".PadRight(60, '_'), font10Bold, brush, layout, formatLeft);

Offset = Offset + lineheight10;
float x = startX;
layout = new RectangleF(new PointF(x, startY + Offset), layoutSize);
graphics.DrawString("No", font10Bold, brush, layout, formatLeft);

x = x + 30;
layout = new RectangleF(new PointF(x, startY + Offset), layoutSize);
graphics.DrawString("Món ăn", font10Bold, brush, layout, formatLeft);

x = x + 200;
layout = new RectangleF(new PointF(x, startY + Offset), layoutSize);
graphics.DrawString("SL", font10Bold, brush, layout, formatLeft);

x = x + 40;
layout = new RectangleF(new PointF(x, startY + Offset), layoutSize);
graphics.DrawString("ĐV", font10Bold, brush, layout, formatLeft);

x = x + 70;
layout = new RectangleF(new PointF(x, startY + Offset), layoutSize);
graphics.DrawString("Giá", font10Bold, brush, layout, formatLeft);
```

```
x = x + 70;
layout = new RectangleF(new PointF(x, startY + Offset), layoutSize);
graphics.DrawString("TT", font10Bold, brush, layout, formatLeft);

Offset = Offset+ leading;
layout = new RectangleF(new PointF(startX, startY + Offset), layoutSize);
graphics.DrawString("".PadRight(60,'_'), font10Bold, brush, layout, formatLeft);


Offset = Offset + lineheight10;
// Vòng lặp hiển thị các món ăn trên hóa đơn
foreach (ListViewItem item in lsvOrdered.Items)
{
    string No = item.SubItems[0].Text;
    x = startX;
    layout = new RectangleF(new PointF(x, startY + Offset), layoutSize);
    graphics.DrawString(No, font10, brush, layout, formatLeft);

    string ItemName = item.SubItems[1].Text;  x = x + 30;
    layout = new RectangleF(new PointF(x, startY + Offset), layoutSize);
    graphics.DrawString(ItemName, font10, brush, layout, formatLeft);

    string ItemQuantity = item.SubItems[3].Text; x = x + 200;
    layout = new RectangleF(new PointF(x, startY + Offset), layoutSize);
    graphics.DrawString(ItemQuantity, font10, brush, layout, formatLeft);

    string ItemUnit = item.SubItems[2].Text; x = x + 40;
    layout = new RectangleF(new PointF(x, startY + Offset), layoutSize);
    graphics.DrawString(ItemUnit, font10, brush, layout, formatLeft);

    string ItemPrice = item.SubItems[4].Text; x = x + 70;
    layout = new RectangleF(new PointF(x, startY + Offset), layoutSize);
    graphics.DrawString(ItemPrice, font10, brush, layout, formatLeft);

    string Total = (int.Parse(ItemQuantity) * int.Parse(ItemPrice)).ToString();
    x = x + 70;
    layout = new RectangleF(new PointF(x, startY + Offset), layoutSize);
    graphics.DrawString(Total, font10, brush, layout, formatLeft);
    Offset = Offset + lineheight10;
}


// Phần thông tin chi phí thanh toán
Offset = Offset + leading;
layout = new RectangleF(new PointF(startX, startY + Offset), layoutSize);
graphics.DrawString("".PadRight(60, '_'), font10, brush, layout, formatLeft);

Offset = Offset + lineheight10;
layout = new RectangleF(new PointF(startX+200, startY + Offset), layoutSize);
graphics.DrawString("Tổng tiền",
    new Font(font10.FontFamily, font10.Size, FontStyle.Bold), brush, layout, formatLeft);
layout = new RectangleF(new PointF(startX-50, startY + Offset), layoutSize);
graphics.DrawString(totalMoney.ToString() + " Đ",
    new Font(font10.FontFamily, font10.Size, FontStyle.Bold), brush, layout, formatRight);


Offset = Offset + lineheight10;
layout = new RectangleF(new PointF(startX + 200, startY + Offset), layoutSize);
graphics.DrawString("Thuế: ",
    new Font(font10.FontFamily, font10.Size, FontStyle.Bold), brush, layout, formatLeft);
float tax = (float)txtTax.Value;
layout = new RectangleF(new PointF(startX - 50, startY + Offset), layoutSize);
graphics.DrawString(tax.ToString() + " Đ",
    new Font(font10.FontFamily, font10.Size, FontStyle.Bold), brush, layout, formatRight);
```

```
        Offset = Offset + lineheight10;
        layout = new RectangleF(new PointF(startX + 200, startY + Offset), layoutSize);
        graphics.DrawString("Chiết khấu: ",
            new Font(font10.FontFamily, font10.Size, FontStyle.Bold), brush, layout, formatLeft);
        layout = new RectangleF(new PointF(startX - 50, startY + Offset), layoutSize);
        graphics.DrawString(tax.ToString() + " Đ",
            new Font(font10.FontFamily, font10.Size, FontStyle.Bold), brush, layout, formatRight);

        Offset = Offset + lineheight10;
        double money_new = totalMoney + (double)(totalMoney * (txtTax.Value / 100))
                        - (double)(totalMoney * (txtDiscount.Value / 100));
        layout = new RectangleF(new PointF(startX + 200, startY + Offset), layoutSize);
        graphics.DrawString("Tổng thanh toán",
            new Font(font10.FontFamily, font10.Size, FontStyle.Bold), brush, layout, formatLeft);
        layout = new RectangleF(new PointF(startX - 50, startY + Offset), layoutSize);
        graphics.DrawString(money_new.ToString() + " Đ",
            new Font(font10.FontFamily, font10.Size, FontStyle.Bold), brush, layout, formatRight);

        Offset = Offset + lineheight10;
        string str = NumberToText(money_new);
        String c = str[0].ToString().ToUpper();
        str = str.Substring(1);
        string money_new_str = "("+c+str+")";
        layout = new RectangleF(new PointF(startX, startY + Offset), layoutSize);
        graphics.DrawString(money_new_str,
            new Font(font10.FontFamily, font10.Size), brush, layout, formatCenter);
        // Phần cuối hóa đơn
        Offset = Offset + lineheight20;
        string thankyou = "CẢM ƠN QUÝ KHÁCH VÀ HẸN GẶP LẠI";
        layout = new RectangleF(new PointF(startX, startY + Offset), layoutSize);
        graphics.DrawString(thankyou,
            font13, brush, layout, formatCenter);

        font20.Dispose(); font15.Dispose(); font13.Dispose();
        font10.Dispose();
    }
```

Mô đun thống kê/Báo cáo: Mô đun này cho phép nhân viên và người quản lý thống kê doanh thu theo các thông tin cho phép. Chúng tôi có sử dụng công cụ LiveCharts.Winforms và công cụ vẽ biểu đồ trong WindowForm để biểu diễn hai loại biểu đồ Bar và Pie. Các chức năng cho phép thống kê bao gồm:

- ***Thống kê doanh thu theo ngày***: Cho phép chọn ngày và thống kê doanh thu từng bàn của ngày hôm đó.
- ***Thống kê doanh thu theo tuần***: Cho phép chọn tháng và thống kê doanh thu theo từng tuần của tháng đó.
- ***Thống kê doanh thu theo tháng***: Cho phép chọn năm và thống kê doanh thu của 12 tháng trong năm
- ***Thống kê doanh thu theo 5 năm gần nhất***: Hệ thống sẽ vẽ biểu đồ doanh thu của 5 năm gần nhất.

Ngoài ra, mô đun này còn cho phép người dùng kết xuất tập tin JSON và kết xuất tập tin XML.

Phương thức sau đây là một ví dụ về mã nguồn thống kê theo ngày, để hiển thị lên hai biểu đồ:

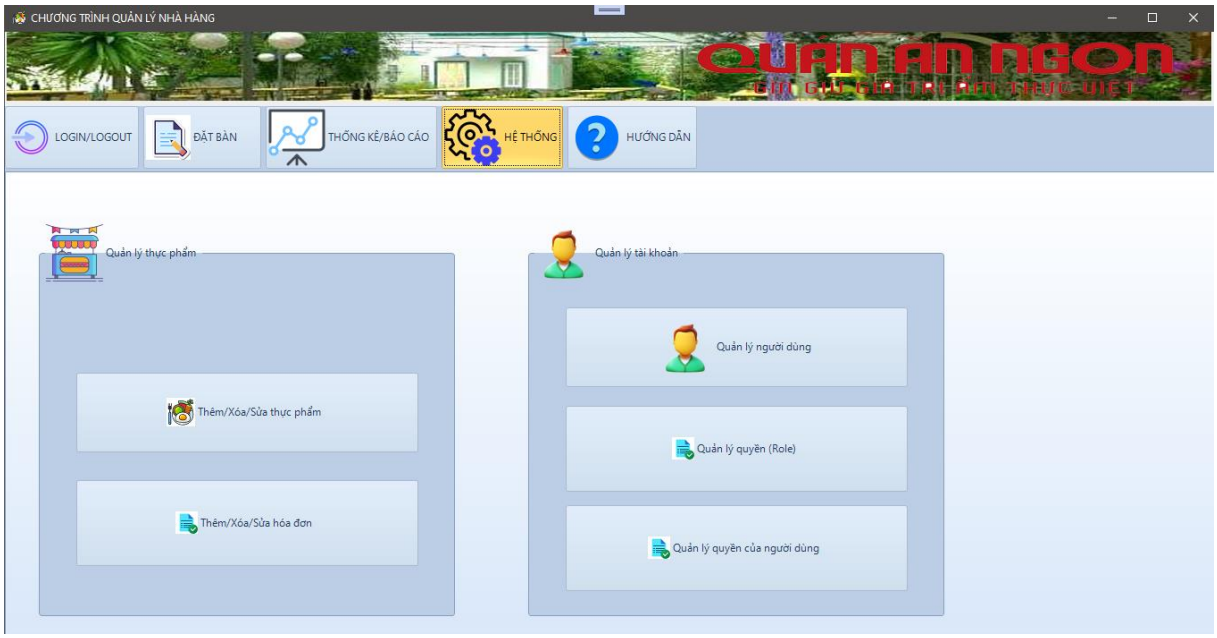
```
2 references
private void DateStatistical()
{
    // Phần hiển thị các tiêu đề
    chartBar.Series[0].Points.Clear();
    SeriesCollection series = new SeriesCollection();
    List<BillInfo> billList = BillsDAO.GetAll();
    DateTime dt = dtDayChosen.Value;
    string str = "";
    if (dt.Day == DateTime.Now.Day && dt.Month == DateTime.Now.Month && dt.Year==DateTime.Now.Year)
        str = string.Format("DOANH THU THEO BÀN CỦA NGÀY HÔM NAY");
    else str = string.Format("DOANH THU THEO BÀN CỦA NGÀY " + dt.Day + "/" + dt.Month + "/" + dt.Year);

    lblDisplay.Text = str;

    // Hiển thị Bar chart
    chartBar.Series[0].Name = "Bàn";
    chartBar.Series[0].Color = Color.Red;
    chartBar.ChartAreas[0].AxisX.Title = "Bàn";
    chartBar.ChartAreas[0].AxisY.Title = "Doanh thu (VNĐ)";

    // Duyệt vòng lặp theo hóa đơn
    foreach (var item in billList)
    {
        if (dt.Day == item.CheckoutDate.Day
            && dt.Month == item.CheckoutDate.Month
            && dt.Year == item.CheckoutDate.Year)
        {
            TableInfo table = TableDAO.Find(item.TableID);
            chartBar.Series[0].Points.AddXY(table.Name, item.Amount);
        }
    }

    // Hiển thị PieChart
    foreach (var item in billList)
    {
        if (dt.Day == item.CheckoutDate.Day
            && dt.Month == item.CheckoutDate.Month
            && dt.Year == item.CheckoutDate.Year)
        {
            TableInfo table = TableDAO.Find(item.TableID);
            series.Add(new PieSeries()
            {
                Title = table.Name,
                Values = new ChartValues<int> { item.Amount },
                DataLabels = true,
                LabelPoint = label
            });
            chartPie.Series = series;
        }
    }
}
```

Hai phương thức sau đây cho phép kết xuất tập tin JSON và XML:

```
private void JSONExport(string filepath)
{
    List<BillInfo> billList_Export = BillsDAO.GetAll();
    string json = JsonSerializer.Serialize(billList_Export);
    File.WriteAllText(filepath, json, Encoding.UTF8);
}
0 references
private void JXMLExport(string filePath)
{
    List<BillInfo> billList_Export = BillsDAO.GetAll();
    XmlSerializer serializer = new XmlSerializer(typeof(List<BillInfo>));
    using (StreamWriter writer = new StreamWriter(filePath))
    {
        serializer.Serialize(writer, billList_Export);
    }
}
```

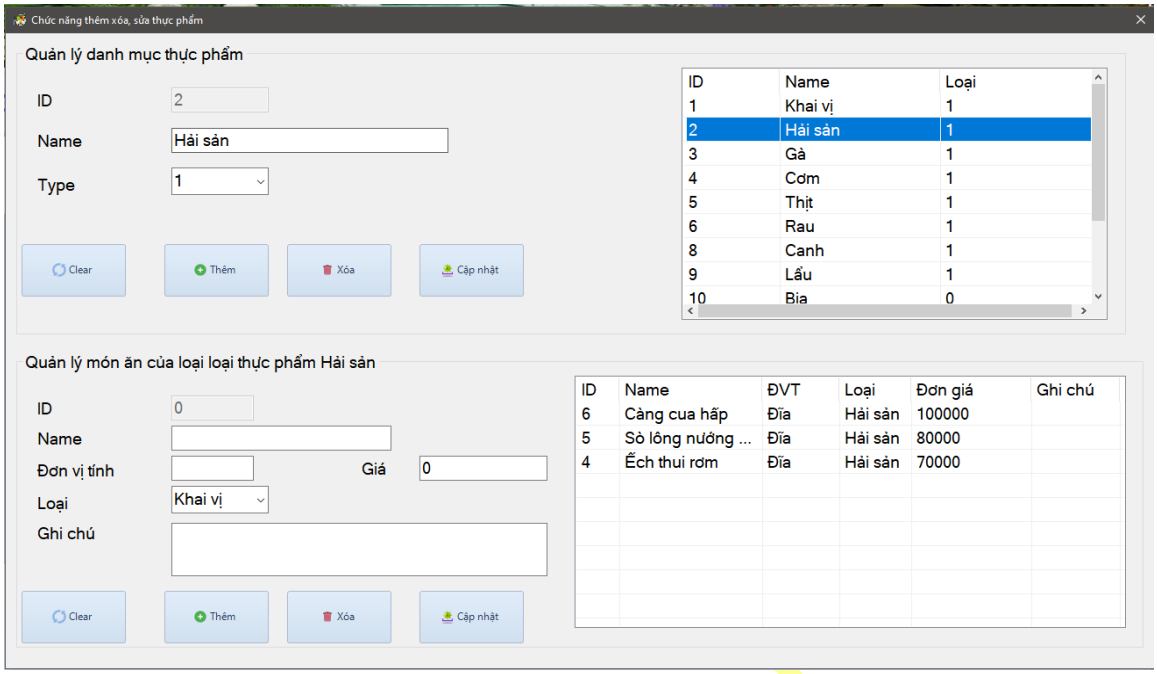
Kết quả giao diện mô đun thống kê theo tuần như sau:

Mô đun Hệ thống: Mô đun này cho phép nhân viên và người quản lý quản trị hệ thống các bảng. Hệ thống các bảng bao gồm: 1) Phần quản lý thực phẩm và hóa đơn và 2) Phần quản lý tài khoản và phân quyền. Mỗi phần sẽ có các nút tương ứng. Giao diện của Mô đun này như hình sau:



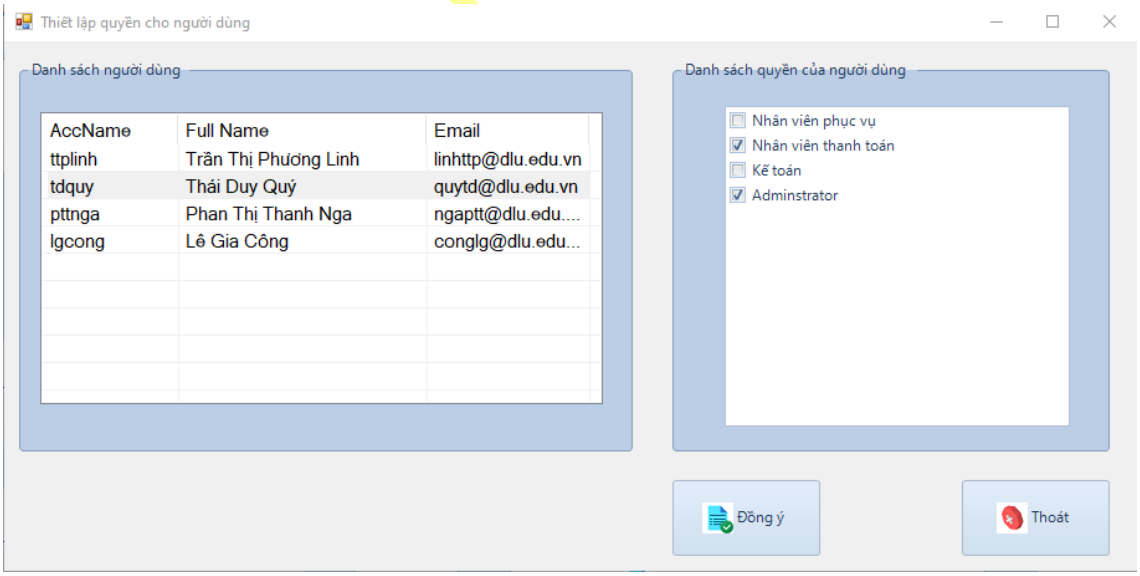
Phần quản lý thực phẩm sẽ được gom quản lý theo hai bảng, nghĩa là sẽ quản lý theo hình thức khóa ngoại. Chương trình sử dụng các ListView và các công cụ phù hợp để xử lý khi người dùng lựa chọn từng sản phẩm. Phần quản lý thực phẩm sẽ thực hiện trên hai bảng là Category và Food, trước tiên cho phép người dùng thêm xóa, sửa thông tin trên bảng Category, sau đó nếu người dùng chọn loại thì các thông tin về thực phẩm loại đó sẽ được hiển thị ở phía dưới. Chương trình đồng thời cho phép thêm, xóa, sửa thông tin ở bảng Food.

Chi tiết về chức năng quản lý thực phẩm, được minh họa như trong hình sau:



Chức năng quản lý người dùng sẽ bao gồm các chức năng như quản lý Account, quản lý quyền và phân quyền. Chức năng phân quyền sẽ cho phép người sử dụng lựa chọn tài khoản, sau đó chọn hoặc bỏ chọn các quyền trong hệ thống.

Chức năng phân quyền được thể hiện như trong hình sau:

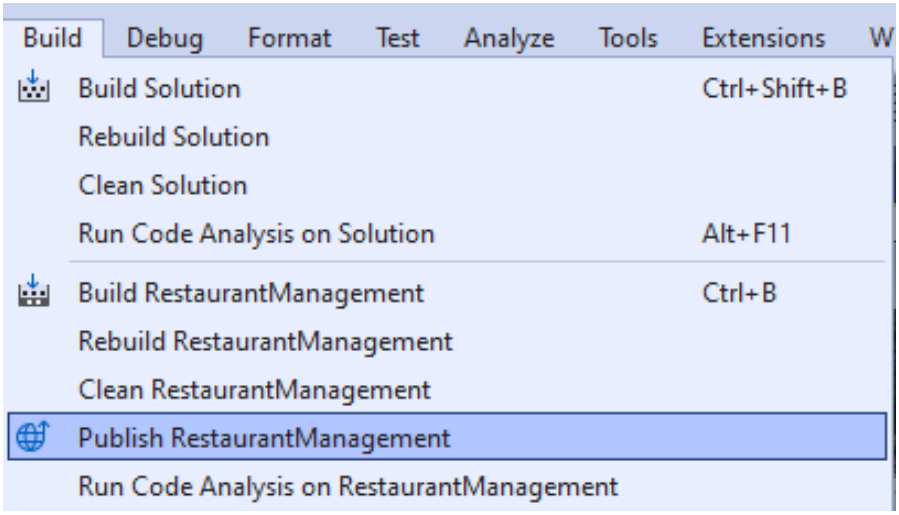


4. Đóng gói chương trình

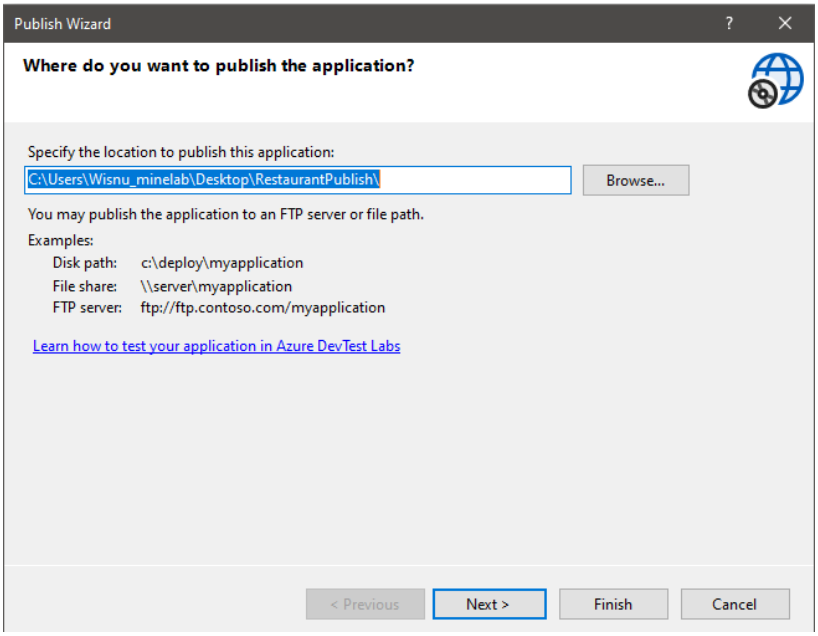
Để đóng gói và triển khai chương trình, người dùng cần tạo ra một sản phẩm để cho phép cài đặt lên máy tính khác. Một lưu ý là muốn đóng gói chương trình hiệu quả, thì dòng lệnh kết nối CSDL phải được viết trong tập tin *App.config*. Khi muốn xuất bản để cài đặt ở máy nào, người dùng cần sửa lại địa chỉ Server, cũng như tên Database có trong đó.

Một lưu ý khi muốn triển khai chương trình đó là chương trình phải không còn các lỗi logic, nên chương trình sẽ tiến hành kiểm tra trước khi xuất bản, nếu còn tồn tại lỗi thì khi xuất bản sẽ không xuất bản được.

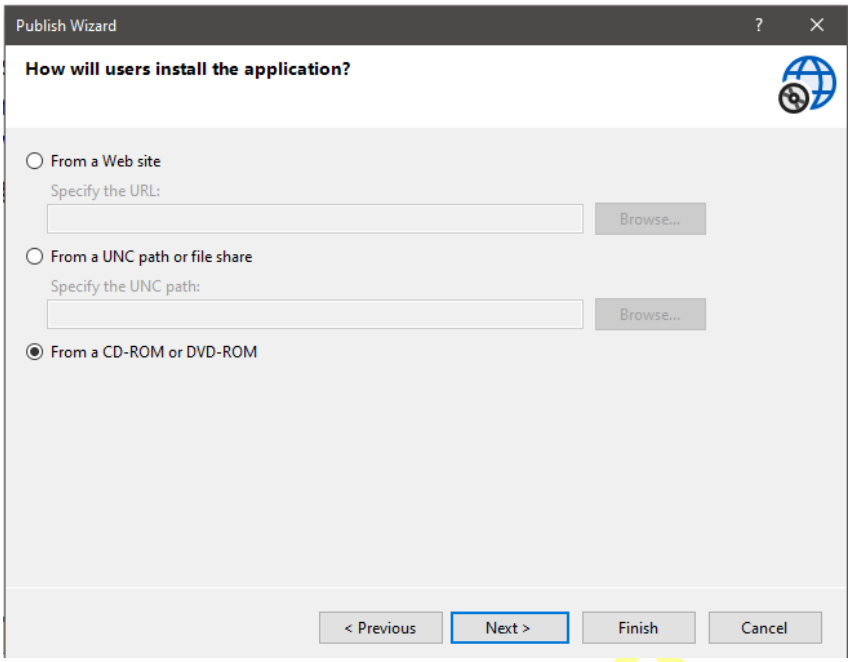
Trong Visual Studio, muốn đóng gói để xuất bản, vào **Build**, chọn **Publish [tên chương trình]** như hình sau:



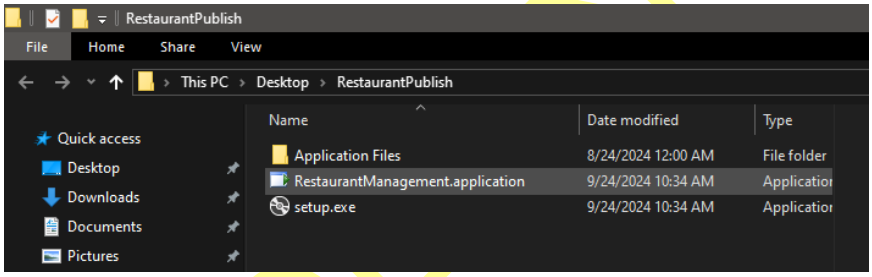
Khi đó, hệ thống yêu cầu chọn nơi để xuất bản, bạn chọn thư mục xuất bản, nhấn Next:



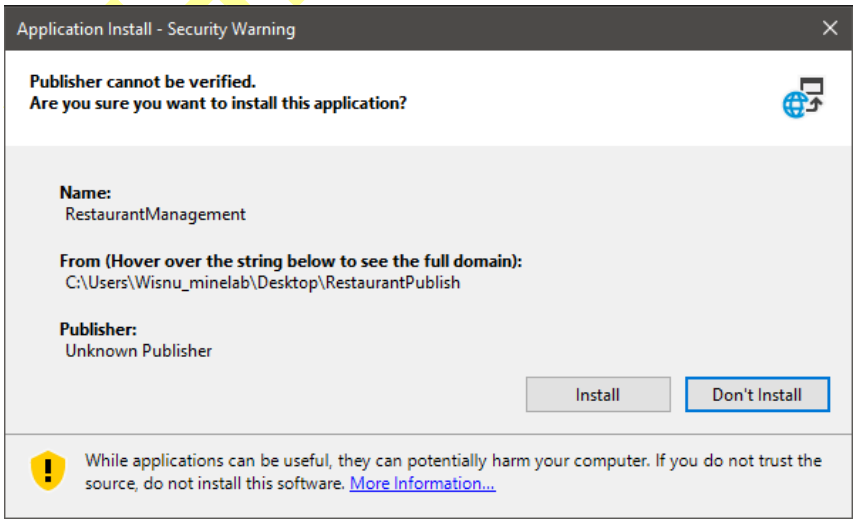
Lựa chọn From a CD-ROM or DVD-ROOM:



Kết quả như sau:



Lúc này, chương trình đã được triển khai tại một thư mục trên máy tính. Nhấp chuột vào setup.exe để tiến hành cài đặt:



Sau khi cài đặt xong, người dùng có thể vào Menu Start để chạy ứng dụng như thường lệ. Lưu ý là việc cài đặt và thiết lập Database được tiến hành như tại Chủ đề 1.

C. Bài tập

1. Theo hướng dẫn như trên, hãy xây dựng chức năng Login, Logout cho một ứng dụng bất kỳ.

2. Xây dựng chức năng đặt bàn, gọi món theo quy trình nêu trên

3. Sử dụng các công cụ PrintDocument và PrintPreview để tạo hóa đơn cho bảng Bill.

4. Thực hành tập vẽ các loại biểu đồ bằng cách sử dụng công cụ vẽ biểu đồ hoặc cài đặt LiveCharts.Winforms trong NuGet Package Manager.

5. Phân quyền cho người dùng bằng cách sử dụng mô hình phân quyền được gợi ý trong phần cơ sở dữ liệu.

TÀI LIỆU THAM KHẢO

Bai, Y. (2010). Database programming practical database programming with Visual C#.NET. John Wiley & Sons, Inc.

Connection Strings. (n.d.). Retrieved from https://www.connectionstrings.com

Entity Framework Tutorial. (n.d.). Retrieved from https://www.entityframework tutorial.net/

FunctionX. (n.d.). Visual C# 2008 tutorial. Retrieved from http://www.functionx. com/vcsharp2008/index.htm

FunctionX. (n.d.). Visual C# 2010 tutorial. Retrieved from http://www.functionx.com/vcsharp2010/index.htm

GeeksforGeeks. (n.d.). *Repository design pattern*. Retrieved from https://www.geeksforgeeks.org/repository-design-pattern/

Hidayat, A. (n.d.). N-tier layer architecture in C#. Medium. Retrieved from https://medium.com/@hidayatarg/n-tier-layer-architecture-in-c-15b8fe97283c#:~:text=N%20represent%20a%20number%20and,(Layers)%20of%20an%20application.&text=It%20comprise%20of%203%2Dtiers,and%20Layers%20are%20used%20interchangeably

Net-Informations. (n.d.). C# GUI programming: Windows forms. Retrieved from http://csharp.net-informations.com/gui/cs_forms.htm

Nguyễn, M. H., & Nguyễn, V. P. (2009). Giáo trình lập trình cơ sở dữ liệu. Trường Đại học Đà Lạt.

Microsoft. (n.d.). Entity Framework 6. Retrieved from https://docs.microsoft.com/en-us/ef/ef6/

Price, J. (2004). Mastering C# database programming. Retrieved from https://epdf.pub/mastering-c-database-programming55279.html

PHỤ LỤC: CƠ SỞ DỮ LIỆU QUẢN LÝ NHÀ HÀNG

STT	Tên trường	Kiểu	Null	Ghi chú
Bảng **Restaurant**				
1	ID	int		Tự tăng; Khóa chính
2	Name	nvarchar(1000)		Tên nhà hàng, mặc định "Chưa đặt tên"
3	Address	nvarchar(3000)		Mặc định "Chưa có địa chỉ"
4	Phone	nvarchar(100)		Mặc định "02633…"
5	Website	nvarchar(1000)	x	
Ghi chú: Bảng này lưu thông tin về nhà hàng. Một nhà hàng có thể bao gồm nhiều sảnh				
Bảng **Hall**				
1	ID	int		Tự tăng; Khóa chính
2	Name	nvarchar(1000)	X	Mặc định "Chưa đặt tên"
3	RestaurantID	int		Thuộc nhà hàng nào
Ghi chú: Bảng này lưu thông tin về các sảnh, trong trường hợp nhà hàng có nhiều sảnh. Nếu như một nhà hàng lớn có nhiều chi nhánh thì đây chính là chi nhánh.				
Bảng **Table**				
1	ID	int		Tự tăng; Khóa chính
2	TableCode	nvarchar(200)		Ký hiệu bàn: Ví dụ: Bàn 2.11, bàn số 11 sảnh 2.
3	Name	nvarchar(1000)	x	Tên bàn: Ví dụ: Bàn VIP, Bàn Khách đặc biệt.
4	Status	Int		Trạng thái: 0: chưa đặt; 1: Đã đặt; 2: Có khách
5	Seats	int	x	Số chỗ ngồi tối đa
6	HallID	int		Bàn thuộc sảnh nào
Ghi chú: Bảng này lưu thông tin về bàn ăn. Một bàn sẽ thuộc sảnh nào, bàn chứa tối đa bao nhiêu chỗ ngồi.				
Bảng **FoodCategory**				
1	ID	int		Tự tăng; Khóa chính
2	Name	nvarchar(1000)		Tên đồ ăn, thức uống, mặc định "Chưa đặt tên"
3	Type	int		Đồ ăn (1) hay thức uống (2)
Ghi chú: Bảng này lưu thông tin về loại đồ ăn, thức uống. Nếu Type = 1 là thức ăn; nếu Type = 2 là đồ uống.				

Bảng **Food**				
1	ID	int		Tự tăng; Khóa chính
2	Name	nvarchar(1000)		Tên đồ ăn, mặc định “Chưa đặt tên”
3	FoodCategoryID	int		Thuộc loại đồ ăn, thức uống nào
4	Price	int		Giá, mặc định 0
5	Notes	Nvarchar(3000)	X	Ghi chú về món ăn
Ghi chú: Bảng này lưu thông tin về món ăn, món uống và giá của từng món.				

Bảng **Invoice**				
1	ID	int		Tự tăng; Khóa chính
2	Name	nvarchar(1000)		Tên hóa đơn, mặc định “Hóa đơn thanh toán”
3	TableID	Int		Ngồi bàn nào
4	Total	Int		Tổng tiền phải thanh toán, mặc định 0
5	Discount	Float	X	Số tiền giảm giá, mặc định 0
6	Tax	float	X	Thuế, mặc định 0
7	Status	bit		Đã thanh toán hay chưa (1 hoặc 0), mặc định 0
8	AccountID	int		Tài khoản nào đang đăng nhập, mặc định 1
9	CheckoutDate	smalldatetime		Ngày in phiếu, mặc định ngày hiện tại
Ghi chú: Bảng này lưu thông tin về phiếu thu tiền; Biên lai chỉ lưu các thông tin cơ bản, chi tiết món ăn sẽ có trong bảng **InvoiceDetails.**				

Bảng **InvoiceDetails**				
1	ID	int		Tự tăng; Khóa chính
2	InvoiceID	int		Thuộc hóa đơn nào
3	FoodID	int		Đồ ăn, thức uống nào
4	Amount	int		Số lượng, mặc định 0
Ghi chú: Bảng này lưu thông tin về các món ăn và số lượng trong phiếu;				

Bảng **Account**				
1	AccountName	nvarchar(100)		Tên tài khoản; Khóa chính
2	Password	nvarchar(200)		Mật khẩu
3	FullName	nvarchar(1000)		Họ tên
4	Email	nvarchar(1000)	X	
5	Phone	nvarchar(200)		

6	DateCreated	smalldatetime		Ngày tạo tài khoản, mặc định là ngày hôm nay
Ghi chú: Bảng này lưu thông tin về các tài khoản đăng nhập; Thông tin tài khoản sẽ kèm theo Họ và tên, cũng như các thông tin cơ bản khác.				

Bảng **Role**				
1	ID	int		Tự tăng, khóa chính
2	RoleName	nvarchar(1000)		Tên quyền
3	Path	nvarchar(3000)	X	Đường dẫn (hoặc tên Control)
4	Notes		X	
Ghi chú: Bảng này lưu thông tin về các quyền trong chương trình. Mỗi quyền sẽ có một tên quyền: Ví dụ: Quản lý, Kế toán, Nhân viên, … Path lưu trữ thông tin về Control hoặc đường dẫn chứa quyền.				

Bảng **RoleAccount**				
1	AccountName	nvarchar(100)		Khóa chính
2	RoleID	Int		Khóa chính
3	Actived	bit		Có kích hoạt hay không
4	Notes	nvarchar(3000)	X	Ghi chú
Ghi chú: Bảng này dùng để phân quyền. Mỗi tài khoản được gắn với 1 quyền và có được kích hoạt hay không.				