**TRƯỜNG ĐẠI HỌC HẠ LONG**

**KHOA CÔNG NGHỆ THÔNG TIN**

ĐỀ TÀI

**PHẦN MỀM GHI CHÚ VIỆC CẦN LÀM**

|  |  |
| --- | --- |
| **Sinh viên thực hiện:**  **Lớp:**  **Giảng viên hướng dấn:** | Nguyễn Trường Minh  KHMT K4  Ts. Lê Anh Tú |

**Quảng Ninh, tháng 04 năm 2021**

# **MỤC LỤC**

[MỤC LỤC 2](#_Toc73522640)

[MỤC LỤC HÌNH 3](#_Toc73522641)

[LỜI MỞ ĐẦU 4](#_Toc73522642)

[CHƯƠNG I: CƠ SỞ LÝ THUYẾT VÀ MÔ TẢ BÀI TOÁN 5](#_Toc73522643)

[A. Cơ sở lý thuyết 5](#_Toc73522644)

[1. Giới thiệu về ngôn ngữ lập trình C# 5](#_Toc73522645)

[2. Môi trường lập trình C# 5](#_Toc73522646)

[3. Ngôn ngữ truy vấn cơ sở dữ liệu Sql Server 6](#_Toc73522647)

[4. Kiến trúc xây dựng phần mềm 6](#_Toc73522648)

[B. Mô tả bài toán 10](#_Toc73522649)

[1. Lý do chọn đề tài 10](#_Toc73522650)

[2. Những yêu cầu đặt ra đối với hệ thống 10](#_Toc73522651)

[CHƯƠNG II: PHÂN TÍCH THIẾT KẾ HỆ THỐNG VÀ THIẾT KẾ CSDL 11](#_Toc73522652)

[A. Phân tích thiết kế hệ thống 11](#_Toc73522653)

[1. Các chức năng của hệ thống 11](#_Toc73522654)

[2. Mô hình hệ thống 11](#_Toc73522655)

[B. Thiết kế cơ sở dữ liệu 12](#_Toc73522656)

[1. Bảng User 12](#_Toc73522657)

[2. Bảng Avatar 12](#_Toc73522658)

[3. Bảng CustomMenu 13](#_Toc73522659)

[4. Bảng Task 13](#_Toc73522660)

[CHƯƠNG III: XÂY DỰNG PHẦN MỀM 14](#_Toc73522661)

[A. Giao diện phần mềm 14](#_Toc73522662)

[1. Giao diện đăng nhập 14](#_Toc73522663)

[2. Giao diện đăng kí 15](#_Toc73522664)

[3. Giao diện trang chủ 16](#_Toc73522665)

[4. Giao diện hồ sơ cá nhân của người dung 17](#_Toc73522666)

[5. Giao diện cửa sổ chỉnh sửa 18](#_Toc73522667)

[B. Mô tả thực thi phần mềm 19](#_Toc73522668)

[1. DTO Layers 19](#_Toc73522669)

[2. DAL (Data Access Layer) 19](#_Toc73522670)

[3. BLL (Business Logic Layer) 19](#_Toc73522671)

[4. GUI Layer () 26](#_Toc73522672)

[TỔNG KẾT VÀ ĐÁNH GIÁ 36](#_Toc73522673)

[LỜI CẢM ƠN 37](#_Toc73522674)

[TÀI LIỆU THAM KHẢO 38](#_Toc73522675)

[Mã Nguồn 38](#_Toc73522676)

[Github Repository 38](#_Toc73522677)

[Preview 38](#_Toc73522678)

# MỤC LỤC HÌNH

[Hình 1: Sơ đồ tổng quan mô hình ba lớp 7](file:///C:\Users\Gr4y\Desktop\Báo%20Cáo%20Lập%20trình%20Cơ%20sở%20dữ%20liệu.docx#_Toc73522394)

[Hình 2: Chi tiết mô hình ba lớp 8](file:///C:\Users\Gr4y\Desktop\Báo%20Cáo%20Lập%20trình%20Cơ%20sở%20dữ%20liệu.docx#_Toc73522395)

[Hình 3: Biểu đồ phân cấp chức năng 11](file:///C:\Users\Gr4y\Desktop\Báo%20Cáo%20Lập%20trình%20Cơ%20sở%20dữ%20liệu.docx#_Toc73522396)

[Hình 4: mô hình ER 12](file:///C:\Users\Gr4y\Desktop\Báo%20Cáo%20Lập%20trình%20Cơ%20sở%20dữ%20liệu.docx#_Toc73522397)

[Hình 5: Giao diện đăng nhập 14](file:///C:\Users\Gr4y\Desktop\Báo%20Cáo%20Lập%20trình%20Cơ%20sở%20dữ%20liệu.docx#_Toc73522398)

[Hình 6: Giao diện trang đăng kí 15](file:///C:\Users\Gr4y\Desktop\Báo%20Cáo%20Lập%20trình%20Cơ%20sở%20dữ%20liệu.docx#_Toc73522399)

[Hình 7: Thông báo đăng kí tài khoản 15](file:///C:\Users\Gr4y\Desktop\Báo%20Cáo%20Lập%20trình%20Cơ%20sở%20dữ%20liệu.docx#_Toc73522400)

[Hình 8: Thông báo lỗi đăng kí 16](file:///C:\Users\Gr4y\Desktop\Báo%20Cáo%20Lập%20trình%20Cơ%20sở%20dữ%20liệu.docx#_Toc73522401)

[Hình 9: Giao diện trang chủ 16](file:///C:\Users\Gr4y\Desktop\Báo%20Cáo%20Lập%20trình%20Cơ%20sở%20dữ%20liệu.docx#_Toc73522402)

[Hình 10: Giao diện hồ sơ cá nhân của người dùng 17](file:///C:\Users\Gr4y\Desktop\Báo%20Cáo%20Lập%20trình%20Cơ%20sở%20dữ%20liệu.docx#_Toc73522403)

[Hình 11: Cửa sổ popup 18](file:///C:\Users\Gr4y\Desktop\Báo%20Cáo%20Lập%20trình%20Cơ%20sở%20dữ%20liệu.docx#_Toc73522404)

# LỜI MỞ ĐẦU

Việc biết rõ được việc mình cần làm là gì, và luôn ghi nhớ điều đó là việc quan trọng. Điều đó giúp ta biết được đích đến, hướng đi của mình là gì, và luôn tập trung vào mục tiêu đó. Các phần mềm hỗ trợ ghi chú việc cần làm trên thế giới rất phổ biến, cụ thể như Trello, Micosoft To Do,... Tuy nhiên chưa có một phần mềm nào của Việt Nam được người dùng lựa chọn.

Để phục vụ việc báo cáo thi kết thúc học phần môn Lập trình Cơ sở dữ liệu do thầy Lê Anh Tú giảng dạy tại trường Đại học Hạ Long, tôi xin lựa chọn đề tài xây dựng phần mềm ghi chú việc cần làm. Đề tài sẽ giúp tôi tìm hiểu sâu sắc hơn về lập trình cơ sở dữ liệu và hướng xây dựng của một phần mềm Window.

Tôi xin chân thành cảm ơn những người đã giúp đỡ tôi hoàn thiện phần mềm này!

# CƠ SỞ LÝ THUYẾT VÀ MÔ TẢ BÀI TOÁN

## Cơ sở lý thuyết

### Giới thiệu về ngôn ngữ lập trình C#

C# (hay C sharp) là một ngôn ngữ lập trình đơn giản, được phát triển bởi đội ngũ kỹ sư của Microsoft vào năm 2000, trong đó người dẫn đầu là Anders Hejlsberg và Scott Wiltamuth.

Là ngôn ngữ lập trình hiện đại, hướng đối tượng và nó được xây dựng trên nền tảng của hai ngôn ngữ mạnh nhất là C++ và Java.

C# được thiết kế cho Common Language Infrastructure (CLI), mà gồm Executable Code và Runtime Environment, cho phép chúng ta sử dụng các ngôn ngữ high-level đa dạng trên các nền tảng và cấu trúc máy tính khác nhau.

Với sự hỗ trợ mạnh mẽ của .NET Framework giúp cho việc tạo một ứng dụng Windows Forms hay WPF (Windows Presentation Foundation), . . . trở nên rất dễ dàng.

C# là một ngôn ngữ khá tốt cho người mới học lập trình bởi vì nó rất mạnh , không nhiều lệnh và học nhanh

C# có hỗ trợ mạnh mẽ về .NET Framework cho nên việc tạo và lập trình ứng dựng và phần mềm dựa trên Windows Froms sẽ đơn giản hơn những ngôn ngữ khác.

C# chính là 1 sự lựa chọn tuyệt vời cho sinh viên làm đồ án tốt nghiệp về phần mềm hay ứng dung.

### Môi trường lập trình C#

Để lập trình bằng ngôn ngữ C# thì đầu tiên chúng ta cần phải cài đặt môi trường lập trình. Chúng ta cần các công cụ cần thiết cho việc lập trình.

#### .NET framework

.NET Framework là một môi trường thực thi được quản lý cho Windows, cung cấp nhiều dịch vụ cho các ứng dụng đang chạy của nó. Nó bao gồm hai thành phần chính: bộ thực thi ngôn ngữ chung (CLR), là công cụ thực thi xử lý các ứng dụng đang chạy và thư viện lớp .NET Framework, cung cấp một thư viện mã được kiểm tra, có thể sử dụng lại mà các nhà phát triển có thể gọi từ các ứng dụng của riêng họ. Các dịch vụ mà .NET Framework cung cấp cho các ứng dụng đang chạy bao gồm:

* Quản lý bộ nhớ. Trong nhiều ngôn ngữ lập trình, lập trình viên chịu trách nhiệm phân bổ và giải phóng bộ nhớ và xử lý vòng đời của đối tượng. Trong các ứng dụng .NET Framework, CLR cung cấp các dịch vụ này thay mặt cho ứng dụng.
* Một hệ thống loại phổ biến. Trong các ngôn ngữ lập trình truyền thống, các loại cơ bản được xác định bởi trình biên dịch, làm phức tạp khả năng tương tác giữa các ngôn ngữ. Trong .NET Framework, các loại cơ bản được xác định bởi hệ thống loại .NET Framework và phổ biến cho tất cả các ngôn ngữ nhắm vào .NET Framework.
* Một thư viện lớp học rộng lớn. Thay vì phải viết một lượng lớn mã để xử lý các hoạt động lập trình cấp thấp phổ biến, các lập trình viên sử dụng một thư viện các loại có thể truy cập dễ dàng và các thành viên của họ từ Thư viện lớp .NET Framework.
* Khung phát triển và công nghệ .NET Framework bao gồm các thư viện cho các lĩnh vực phát triển ứng dụng cụ thể, chẳng hạn như ASP.NET cho các ứng dụng web, ADO.NET để truy cập dữ liệu, Windows Communication Foundation cho các ứng dụng hướng dịch vụ và Windows Presentation Foundation cho các ứng dụng máy tính để bàn Windows.
* Khả năng tương tác ngôn ngữ. Các trình biên dịch ngôn ngữ nhắm vào .NET Framework phát ra một mã trung gian có tên là Ngôn ngữ trung gian chung (CIL), đến lượt nó, được biên dịch trong thời gian chạy bởi thời gian chạy ngôn ngữ chung. Với tính năng này, các thói quen được viết bằng một ngôn ngữ có thể truy cập được bằng các ngôn ngữ khác và các lập trình viên tập trung vào việc tạo các ứng dụng bằng ngôn ngữ ưa thích của họ.
* Phiên bản tương thích. Với các trường hợp ngoại lệ hiếm hoi, các ứng dụng được phát triển bằng cách sử dụng một phiên bản .NET Framework cụ thể chạy mà không sửa đổi trên phiên bản mới hơn.
* Thi công bên lề. .NET Framework giúp giải quyết xung đột phiên bản bằng cách cho phép nhiều phiên bản thời gian chạy ngôn ngữ chung tồn tại trên cùng một máy tính. Điều này có nghĩa là nhiều phiên bản ứng dụng có thể cùng tồn tại và một ứng dụng có thể chạy trên phiên bản .NET Framework được xây dựng. Việc thực hiện song song áp dụng cho các nhóm phiên bản .NET Framework 1.0 / 1.1, 2.0 / 3.0 / 3.5 và 4 / 4.5.x / 4.6.x / 4.7.x / 4.8.
* Đa mục tiêu. Bằng cách nhắm mục tiêu [.NET Standard](https://docs.microsoft.com/en-us/dotnet/standard/net-standard), các nhà phát triển tạo các thư viện lớp hoạt động trên nhiều nền tảng .NET Framework được hỗ trợ bởi phiên bản tiêu chuẩn đó. Ví dụ: các thư viện nhắm mục tiêu .NET Standard 2.0 có thể được sử dụng bởi các ứng dụng nhắm mục tiêu .NET Framework 4.6.1, .NET Core 2.0 và UWP 10.0.16299.

#### Trình soạn thảo (IDE)

IDE viết tắt của cụm từ Integrated Development Environment là phần mềm cung cấp cho các lập trình viên một môi trường tích hợp bao gồm nhiều công cụ khác nhau như chương trình viết mã lệnh hay code editor, chương trình sửa lỗi hay debugger, chương trình mô phỏng ứng dụng khi chạy thực tế hay simulator.... Nói cách khác thì IDE là một phần mềm bao gồm những gói phần mềm khác giúp phát triển ứng dụng phần mềm.

* Microsoft cung cấp các công cụ phát triển sau cho lập trình C#:
* Visual Studio (VS)
* Visual Studio Code
* Visual C# Express (VCE)
* Visual Web Developer

Để được hỗ trợ tốt nhất về C# thì ta nên dung Visual Studio.

* Một số thông tin về Microsoft Visual Studio:
* **Visual studio** là một trong những công cụ hỗ trợ lập trình rất nổi tiếng nhất hiện nay của Mcrosoft và chưa có một phần mềm nào có thể thay thế được nó. Visual Studio được viết bằng 2 ngôn ngữ đó chính là C# và VB+. Đây là 2 ngôn ngữ lập trình giúp người dùng có thể lập trình được hệ thống một các dễ dàng và nhanh chóng nhất thông qua Visual Studio.
* Visual Studio là một phần mềm lập trình hệ thống được sản xuất trực tiếp từ [Microsoft](https://www.microsoft.com/vi-vn/). Từ khi ra đời đến nay, Visual Studio đã có rất nhiều các phiên bản sử dụng khác nhau. Điều đó, giúp cho người dùng có thể lựa chọn được phiên bản tương thích với dòng máy của mình cũng như cấu hình sử dụng phù hợp nhất.
* Bên cạnh đó, Visual Studio còn cho phép người dùng có thể tự chọn lựa giao diện chính cho máy của mình tùy thuộc vào nhu cầu sử dụng.

#### Tính năng của Microsoft Visual Studio:

* Biên tập mã: Giống như bất kỳ [IDE](https://vi.wikipedia.org/wiki/IDE) khác, nó bao gồm một [trình soạn thảo mã](https://vi.wikipedia.org/w/index.php?title=Tr%C3%ACnh_so%E1%BA%A1n_th%E1%BA%A3o_m%C3%A3&action=edit&redlink=1" \o "Trình soạn thảo mã (trang chưa được viết)) hỗ trợ [tô sáng cú pháp](https://vi.wikipedia.org/w/index.php?title=T%C3%B4_s%C3%A1ng_c%C3%BA_ph%C3%A1p&action=edit&redlink=1" \o "Tô sáng cú pháp (trang chưa được viết)) và [hoàn thiện mã](https://vi.wikipedia.org/w/index.php?title=Ho%C3%A0n_thi%E1%BB%87n_m%C3%A3&action=edit&redlink=1" \o "Hoàn thiện mã (trang chưa được viết)) bằng cách sử dụng [IntelliSense](https://vi.wikipedia.org/w/index.php?title=IntelliSense&action=edit&redlink=1) không chỉ cho các [biến](https://vi.wikipedia.org/wiki/Bi%E1%BA%BFn_(l%E1%BA%ADp_tr%C3%ACnh)" \o "Biến (lập trình)), [hàm](https://vi.wikipedia.org/wiki/Ch%C6%B0%C6%A1ng_tr%C3%ACnh_con" \o "Chương trình con) và các [phương pháp](https://vi.wikipedia.org/wiki/Ph%C6%B0%C6%A1ng_th%E1%BB%A9c_l%E1%BA%ADp_tr%C3%ACnh) mà còn các cấu trúc ngôn ngữ như [vòng điều khiển](https://vi.wikipedia.org/w/index.php?title=Ki%E1%BB%83m_so%C3%A1t_l%C6%B0u_l%C6%B0%E1%BB%A3ng&action=edit&redlink=1" \o "Kiểm soát lưu lượng (trang chưa được viết)) hoặc [truy vấn](https://vi.wikipedia.org/w/index.php?title=Language_Integrated_Query&action=edit&redlink=1" \o "Language Integrated Query (trang chưa được viết)).
* Trình gỡ lỗi: Visual Studio có một trình gỡ lỗi hoạt động vừa là một trình gỡ lỗi cấp mã nguồn và là một trình gỡ lỗi cấp máy. Nó hoạt động với cả hai mã quản lý cũng như [ngôn ngữ máy](https://vi.wikipedia.org/wiki/Ng%C3%B4n_ng%E1%BB%AF_m%C3%A1y" \o "Ngôn ngữ máy) và có thể được sử dụng để gỡ lỗi các ứng dụng được viết bằng các ngôn ngữ được hỗ trợ bởi Visual Studio. Ngoài ra, nó cũng có thể đính kèm theo quy trình hoạt động và theo dõi và gỡ lỗi những quy trình.

### Ngôn ngữ truy vấn cơ sở dữ liệu Sql Server

SQL là viết tắt của Structured Query Language có nghĩa là ngôn ngữ truy vấn có cấu trúc, là một ngôn ngữ máy tính để lưu trữ, thao tác và truy xuất dữ liệu được lưu trữ trong một cơ sở dữ liệu quan hệ.

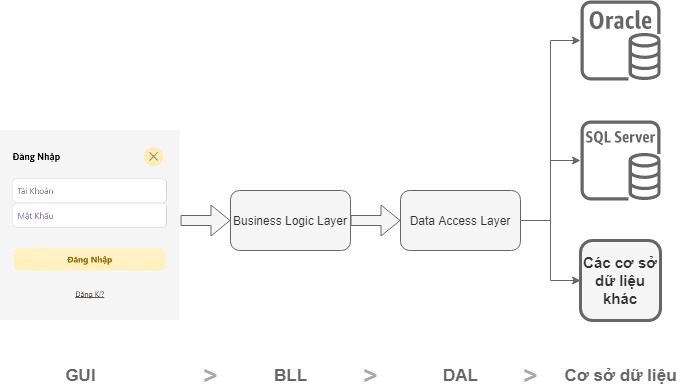
SQL là ngôn ngữ chuẩn cho hệ cơ sở dữ liệu quan hệ. Tất cả các hệ thống quản lý cơ sở dữ liệu quan hệ (RDMS) như MySQL, MS Access, Oracle, Sybase, Informix, Postgres và SQL Server đều sử dụng SQL làm ngôn ngữ cơ sở dữ liệu chuẩn.

### Kiến trúc xây dựng phần mềm

Kiến trúc cơ bản trong xây dựng phần mềm trong C# là mô hình 3 lớp.

Mô hình ba lớp là một kiến trúc kiểu client/server mà trong đó giao diện người dùng (UI-user interface), các quy tắc xử lý (BR-business rule hay BL-business logic), và việc lưu trữ dữ liệu được phát triển như những module độc lập, và hầu hết là được duy trì trên các nền tảng độc lập, và mô hình 3 tầng (3-tiers) được coi là một kiến trúc phần mềm và là một mẫu thiết kế.

#### Tổng quan về mô hình 3 lớp

Mô hình 3 lớp trong C# giúp chúng ta tách biệt các phần code trong chương trình của chúng ta thành các lớp với chức năng riêng biệt theo mô hình như sau:

Hình 1: Sơ đồ tổng quan mô hình ba lớp

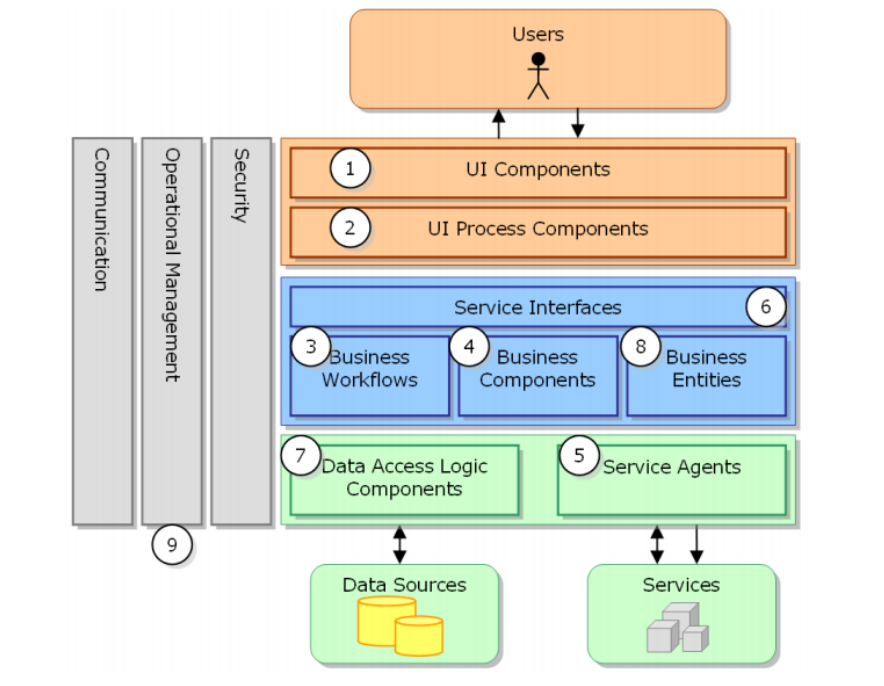
Mô hình này tương tự như mô hình MVC của web, trong đó: Business như là Controller, GUI (Presentation) là View, Data Access là Model.

Mô hình 3-layer gồm có 3 phần chính :

* Presentation Layer (GUI) : Lớp này có nhiệm vụ chính giao tiếp với người dùng. Nó gồm các thành phần giao diện (win form, web form,…) và thực hiện các công việc như nhập liệu, hiển thị dữ liêu, kiểm tra tính đúng đắn dữ liệu trước khi gọi lớp Business Logic Layer (BLL).
* Business Logic Layer (BLL) : Layer này phân ra 2 thành nhiệm vụ :
* Đáp ứng các yêu cầu thao tác dữ liệu của GUI layer, xử lý dữ liệu từ Presentation Layer trước khi truyền xuống Data Access Layer và lưu xuống hệ quản trị CSDL.
* Kiểm tra các ràng buộc, tính toàn vẹn và hợp lệ dữ liệu, thực hiện tính toán và xử lý các yêu cầu nghiệp vụ, trước khi trả kết quả về Presentation Layer.
* Data Access Layer (DAL) : Lớp này có chức năng giao tiếp với hệ quản trị CSDL như thực hiện các công việc liên quan đến lưu trữ và truy vấn dữ liệu (tìm kiếm, thêm, xóa, sửa,…).

*Ghi chú:* Ngoài 3 lớp trên, đôi khi người ta tổ chức thêm một lớp Data Transfer Objects (DTO) Layer. Lớp này định nghĩa các table trong database. Có thể ví DTO như 1 dạng Object Relation Mapping (ORM) cơ bản. Tuy nhiên, thông thường DTO được tích hợp trong DAL.

#### Phân tích chi tiết từng layer trong mô hình ba lớp



Hình 2: Chi tiết mô hình ba lớp

##### Presentation Layer (GUI)

Có hai thành phần chính:

* UI Components : gồm các thành phần tạo nên giao diện của ứng dụng (GUI). Chúng chịu trách nhiệm thu nhận và hiển thị dữ liệu cho người dùng… Ví dụ : textbox, button, combobox, …
* UI Process Components : là thành phần chịu trách nhiệm quản lý các quá trình chuyển đổi giữa các UI… Ví dụ : Sắp xếp quá trình kiểm tra thông tin khách hàng: (1) Hiển thị màn hình tra cứu ID, (2) Hiển thị màn hình thông tin chi tiết khách hàng tương ứng, (3) Hiển thị màn hình liên lạc với khách hàng.

##### Business Logic Layer (BLL)

Lớp này gồm 4 thành phần:

* Service Interface : là thành phần giao diện lập trình mà lớp này cung cấp cho lớp Presentation sử dụng.
* Bussiness Workflows : chịu trách nhiệm xác định và điều phối các quy trình nghiệp vụ gồm nhiều bước và kéo dài. Những quy trình này phải được sắp xếp và thực hiện theo một thứ tự chính xác.
* Bussiness Components : chịu trách nhiệm kiểm tra các quy tắc nghiệp vụ, ràng buộc logic và thực hiện các công việc . Các thành phần này cũng thực hiện các dịch vụ mà Service Interface cung cấp và Business Workflows sẽ sử dụng nó.
* Bussiness Entities : thường được sử dụng như Data Transfer Objects (DTO) . Ccó thể sử dụng để truyền dữ liệu giữa các lớp (Presentation và Data Layer). Chúng thường là cấu trúc dữ liệu (DataSets, XML,… ) hay các lớp đối tượng đã được tùy chỉnh

##### Data Access Layer (DAL)

Lớp này gồm 2 thành phần:

* Data Access Logic Components : lưu trữ và truy xuất dữ liệu từ các nguồn dữ liệu (Data Sources) như XML, file system,….
* Service Agents : gọi và tương tác với các dịch vụ từ bên ngoài.

##### Data Transfer Object (DTO)(không bắt buộc)

DTO hoàn toàn không chứa behavior/ logic, chỉ được sử dụng để truyền dữ liệu và map dữ liệu từ các Domain Model trước khi truyền tới Client. Trong các ứng dụng đơn giản, các Domain Model thường có thể được sử dụng lại trực tiếp dưới dạng DTO và được truyền trực tiếp đến lớp hiển thị, do đó chỉ có một Data Model thống nhất.

Một số lợi ích từ DTO:

* Tách biệt logic một cách rõ ràng : Transfer Object chỉ chứa data, còn logic được implement trong phần khác.
* Cãi thiện hiệu suất ứng dụng : chi phí của mỗi request/ response là lớn, chúng ta nên cố gắng gửi nhiều nhất có thể. Để làm điều này, chúng ta có thể tạo một Transfer Object để gửi data từ Client lên Server hay từ Server đến Client một lần duy nhất, thay vì phải gửi từng phần riêng lẻ.

## Mô tả bài toán

### Lý do chọn đề tài

Xác định mục tiêu, không ngừng nỗ lực và kiên trì để đạt được mục tiêu. Đây chính là chìa khóa đem lại cuộc sống ý nghĩa cho mỗi người. Tuy nhiên, để giữ được mục tiêu luôn trong tiềm thức của mình là chuyện khó.

Là một sinh viên công nghệ thông tin, tôi muốn xây dựng một phần mềm hỗ trợ tôi ghi nhớ mục tiêu của bản thân, giúp tôi phát triển bản thân mỗi ngày.

Phần mềm ghi chú công việc cần làm có các chức năng như đăng nhập, đăng kí, ghi chú công việc, hoàn thành công việc, đánh dấu công việc, tạo danh sách công việc cần làm, tìm kiếm…

### Những yêu cầu đặt ra đối với hệ thống

Hệ thống phải đáp ứng được những yêu cầu: Dễ sử dụng, thao tác nhanh gọn tránh tốn thời gian suy nghĩ, dễ dàng bổ sung, sửa đổi, tìm kiếm.

# PHÂN TÍCH THIẾT KẾ HỆ THỐNG VÀ THIẾT KẾ CSDL

## Phân tích thiết kế hệ thống

### Các chức năng của hệ thống

Qua quá trình nghiên cứu, em đã xây dựng được một hệ thống CSDL của phần mềm ghi chú công việc cần làm, cơ bản đã đáp ứng một số yêu cầu của người dung:

* Quản lý thông tin cần thiết của một công việc cần làm như: việc cần làm là gì, đã hoàn thành hay chưa, việc cần làm có quan trọng không.
* Cập nhật công việc cần làm theo ngày, theo tầm quan trọng, theo danh sách,…
* Tìm kiếm công việc cần làm.
* Thêm, sửa, xóa công việc nhanh.

### Mô hình hệ thống

#### Biểu đồ phân cấp chức năng

Hình 3: Biểu đồ phân cấp chức năng

#### Xây dựng các thực thể trong cơ sở dữ liệu

User(**id\_user**, displayName, id\_avatar, username, password);

Avatar(**id\_avatar**, data, fileName);

CustomMenu(**id\_customMenu**, id\_user, title, iconFont, backColor);

Task(**id\_task**, id\_user, taskContent, timeCreated, category, status, isImportant)

#### Mô hình ER

Hình 4: mô hình ER

## Thiết kế cơ sở dữ liệu

### Bảng User

|  |  |  |
| --- | --- | --- |
| ***Tên cột*** | ***Kiểu dữ liệu*** | ***Mô tả*** |
| **id\_user** | int\* | Mã người dùng |
| displayName | nvarchar(30)\* | Họ tên người dùng |
| id\_avatar | int\* | Mã của ảnh đại diện |
| username | varchar(30)\* | Tên đăng nhập |
| password | nchar(16)\* | Mật khẩu đăng nhập |

### Bảng Avatar

|  |  |  |
| --- | --- | --- |
| ***Tên cột*** | ***Kiểu dữ liệu*** | ***Mô tả*** |
| **id\_avatar** | int\* | Mã người dùng |
| data | image\* | Họ tên người dùng |
| fileName | nchar(50)\* | Đường dẫn của ảnh |

### Bảng CustomMenu

|  |  |  |
| --- | --- | --- |
| ***Tên cột*** | ***Kiểu dữ liệu*** | ***Mô tả*** |
| **id\_customMenu** | int\* | Mã danh sách |
| id\_user | int\* | Mã người dùng |
| title | nvarchar(20)\* | Tên danh sách |
| iconFont | nvarchar(10) | Icon của danh sách |
| backColor | nvarchar(7)\* | Màu của danh sách |

### Bảng Task

|  |  |  |
| --- | --- | --- |
| ***Tên cột*** | ***Kiểu dữ liệu*** | ***Mô tả*** |
| **id\_task** | int\* | Mã công việc |
| id\_user | int\* | Mã người dùng |
| taskContent | nvarchar(50)\* | Chi tiết công việc |
| timeCreated | datetime | Ngày giờ tạo công việc |
| category | nvarchar(50) | Loại công việc |
| status | bit\* | Trạng thái công việc |
| isImportant | bit\* | Trạng thái quan trọng |

# XÂY DỰNG PHẦN MỀM

## Giao diện phần mềm

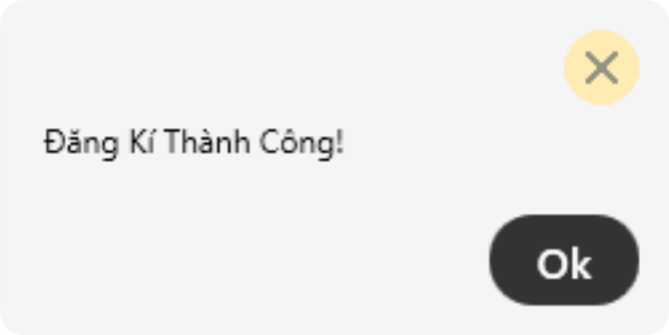
### Giao diện đăng nhập

Hình 5: Giao diện đăng nhập

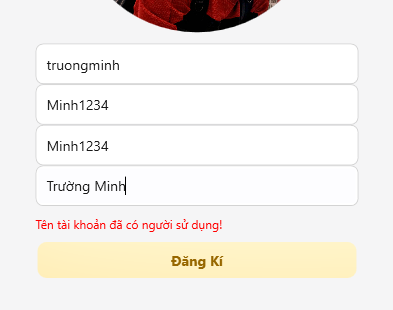
* Đây là giao diện khi phần mềm khởi động. Muốn vào trong trang chủ của phần mềm ta cần đăng nhập qua giao diện này.
* Cần phải có tài khoản đã có trong hệ thống để đăng nhập.

### Giao diện đăng kí

Hình 6: Giao diện trang đăng kí

* Trang này sẽ giúp người dùng đăng kí nếu người dùng chưa có tài khoản trong hệ thống. Người dùng có thể có nhiều tài khoản, tuy nhiên các tên tài khoản không được trùng nhau.
* Sau khi đăng kí, nếu đăng kí thành công, người dùng sẽ nhận được thông báo sau:

Hình 7: Thông báo đăng kí tài khoản

* Còn nếu các thông tin hoàn thiện form đăng kí bị lỗi, Giao diện sẽ thông báo các lỗi như sau:

Hình 8: Thông báo lỗi đăng kí

### Giao diện trang chủ

Hình 9: Giao diện trang chủ

* Sau khi đăng nhập thành công thì đây là giao diện mà người dùng sẽ thấy.
* Hướng đến sự tối giản hóa, không cầu kì, hầu hết thao tác của người dùng sẽ chỉ ở trên một trang duy nhất.

### Giao diện hồ sơ cá nhân của người dung

Hình 10: Giao diện hồ sơ cá nhân của người dùng

* Đây là cửa sổ người dùng sẽ thấy để chỉnh sửa hồ sơ cá nhân.
* Nếu người dùng không muốn dùng phần mềm nữa, có lựa chọn xóa tài khoản cá nhân.

### Giao diện cửa sổ chỉnh sửa

Hình 11: Cửa sổ popup

* Đây là cửa sổ popup nhỏ dùng để chỉnh sửa các việc cần làm hoặc danh sách.

## Mô tả thực thi phần mềm

### DTO Layers

#### MenuModel

namespace **ToDoApp**.**DTO**

{

    public class **MenuModel** : ViewModelBase

    {

        public int ID { get; set; }

        public string SearchString { get; set; }

        public string IconFont { get; set; }

        public string Title { get; set; }

        public string BackColor { get; set; }

        public int CurrentUserID { get; set; }

        public bool Display { get; set; } = true;

        private ObservableCollection<TaskInfo> \_taskInfos = new ObservableCollection<TaskInfo>();

        public ObservableCollection<TaskInfo> TaskInfos

        {

            get

            {

                BLL bus = new BLL();

                return bus.**getTaskForMenuModel**(Title, CurrentUserID, SearchString);

            }

            set { \_taskInfos = value; **RaisePropertyChanged**(); }

        }

        public object ModifyTag { get { return new { ID, Title }; } }

    }

}

#### User

namespace **ToDoApp**.**DTO**

{

    public class **User**

    {

        public **User**() { }

        public **User**(string displayName, Image avatar)

        {

            this.DisplayName = displayName;

            this.Avatar = avatar;

        }

        public int ID { get; set; }

        public string DisplayName { get; set; }

        internal Image Avatar { get; set; }

        internal string UserName { get; set; }

        internal string PassWord { get; set; }

        public BitmapSource DisplayImage

        {

            get

            {

                if (this.Avatar == null)

                {

                    var outPutDirectory = Path.**GetDirectoryName**(Assembly.**GetExecutingAssembly**().CodeBase);

                    var unknownUserPath = Path.**Combine**(outPutDirectory, "Assets\\Images\\unknownUser.jpg");

                    BitmapSource unknownUserSource = new BitmapImage(new Uri(new Uri(unknownUserPath).LocalPath));

                    return unknownUserSource;

                }

                else

                    return ImgConverter.**GetAvatarBitSource**(Avatar);

            }

        }

    }

}

#### Task

namespace **ToDoApp**.**DTO**

{

    public class **Task**

    {

        public **Task**(string content, DateTime timeCreated, bool status, bool isImportant)

        {

            Content = content;

            TimeCreated = timeCreated;

            IsStatusDone = status;

            IsImportant = isImportant;

        }

        public int Id { get; set; }

        public string Content { get; set; }

        public string Category { get; set; }

        public DateTime TimeCreated { get; set; }

        public bool IsStatusDone { get; set; }

        public bool IsImportant { get; set; }

        public string TimeString

        {

            get

            {

                DateTime resDate;

                if (!DateTime.**TryParseExact**(TimeCreated.**ToString**("MM/dd/yyyy"), "MM/dd/yyyy", CultureInfo.InvariantCulture, DateTimeStyles.None, out resDate))

*// lỗi tryParseExact thì sẽ trả về ""*

                    return "";

                if (DateTime.Today == resDate)

                    return "Hôm nay, " + TimeCreated.**ToString**("hh:mm tt");

                else

                    return TimeCreated.**ToString**();

            }

        }

        public object UpdateField

        {

            get

            {

                return new

                {

                    Id = this.Id,

                    Content = this.Content,

                };

            }

        }

        public object BooleanField

        {

            get

            {

                return new

                {

                    Id = this.Id,

                    IsStatusDone = this.IsStatusDone,

                    IsImportant = this.IsImportant

                };

            }

        }

    }

}

### DAL (Data Access Layer)

namespace **ToDoApp**.**Controls**

{

*// Data Access Layer*

    class **DAL**

    {

        static private SqlConnection conn = null;

        private SqlCommand cmd = null;

        public **DAL**()

        {

            conn = new SqlConnection(Properties.Settings.Default.connStr);

        }

        public void **OpenConnector**()

        {

            try

            {

                if (conn.State == ConnectionState.Closed) conn.**Open**();

            }

            catch (Exception ex)

            {

                throw ex;

            }

        }

        public void **CloseConnector**()

        {

            try

            {

                if (conn.State == ConnectionState.Open) conn.**Close**();

            }

            catch (Exception ex)

            {

                throw ex;

            }

        }

        public static SqlCommand **newSqlCommand**(string strSql)

        {

            return new SqlCommand(strSql, conn);

        }

        public SqlDataReader **getDataReader**(string tablename)

        {

            string strSQL = "SELECT \* FROM [dbo].[" + tablename + "]";

            cmd = new SqlCommand(strSQL, conn);

            this.**OpenConnector**();

            SqlDataReader dr = cmd.**ExecuteReader**();

            return dr;

        }

        public DataTable **getDataTableQuery**(SqlCommand command)

        {

            SqlDataReader dr = **getDataQuery**(command);

            DataTable dt = new DataTable();

            dt.**Load**(dr);

            return dt;

        }

        public SqlDataReader **getDataQuery**(string strSql)

        {

            cmd = new SqlCommand(strSql, conn);

            this.**OpenConnector**();

            SqlDataReader dr = cmd.**ExecuteReader**();

            return dr;

        }

        public SqlDataReader **getDataQuery**(SqlCommand command)

        {

            SqlDataReader dr = null;

            try

            {

                this.**OpenConnector**();

                dr = command.**ExecuteReader**();

                return dr;

            }

            catch (Exception ex)

            {

                Console.**WriteLine**(ex.Message);

            }

            return dr;

        }

    }

}

### BLL (Business Logic Layer)

Cấu trúc:

namespace **ToDoApp**.**Control**

{

*/// Business Logic Layer*

    partial class **BLL**

    {

    }

}

#### CustomMenu

partial class **BLL**

{

    DAL dbProcess = new DAL();

    public bool **isCustomMenuValid**(string customMenuTitle, int userID)

    {

        try

        {

            var cmd = DAL.**newSqlCommand**("SELECT \* FROM [dbo].[CustomMenu] WHERE [CustomMenu].title = @Title AND [CustomMenu].id\_user = @userID");

            cmd.Parameters.**AddWithValue**("@Title", customMenuTitle);

            cmd.Parameters.**AddWithValue**("@userID", userID);

            DataTable table = dbProcess.**getDataTableQuery**(cmd);

            if (table.Rows.Count > 0)

            {

                new PromptDialog("Danh sách " + customMenuTitle.**ToUpper**() + " đã tồn tại", promptInput: false, cancelButton: false).**ShowDialog**();

                return false;

            }

            else if (table.Rows.Count == 0)

            {

                return true;

            }

        }

        catch (Exception e)

        {

            Console.**WriteLine**(e.Message);

        }

        return false;

    }

    public ObservableCollection<MenuModel> **getCustomMenu**(int userID)

    {

        var cmd = DAL.**newSqlCommand**("SELECT \* FROM CustomMenu WHERE [CustomMenu].id\_user = @userID");

        cmd.Parameters.**AddWithValue**("@userID", userID);

        SqlDataReader dr = dbProcess.**getDataQuery**(cmd);

        ObservableCollection<MenuModel> customMenuModels = new ObservableCollection<MenuModel>();

        while (dr.**Read**())

        {

            int id = (int)dr.**GetValue**(dr.**GetOrdinal**("id\_customMenu"));

            string title = dr.**GetString**(dr.**GetOrdinal**("title"));

            string iconFont = dr.**GetString**(dr.**GetOrdinal**("iconFont"));

            string backColor = dr.**GetString**(dr.**GetOrdinal**("backColor"));

            MenuModel menu = new MenuModel() { Title = title, IconFont = iconFont, BackColor = backColor, ID = id };

            customMenuModels.**Add**(menu);

        }

        dbProcess.**CloseConnector**();

        return customMenuModels;

    }

    public ObservableCollection<MenuModel> **createCustomMenu**(int userID, MenuModel menu, MainViewModel viewModel)

    {

        if (**isCustomMenuValid**(menu.Title, userID))

        {

            try

            {

                var cmd = DAL.**newSqlCommand**("INSERT INTO [dbo].[CustomMenu] ([id\_user], [title], [iconFont], [backColor]) VALUES (@UserID, @Title, @IconFont, @BackColor) SELECT SCOPE\_IDENTITY();");

                cmd.Parameters.**AddWithValue**("@UserID", userID);

                cmd.Parameters.**AddWithValue**("@Title", menu.Title);

                cmd.Parameters.**AddWithValue**("@IconFont", menu.IconFont);

                cmd.Parameters.**AddWithValue**("@BackColor", menu.BackColor);

                DataTable table = dbProcess.**getDataTableQuery**(cmd);

                if (table.Rows.Count == 1)

                {

                    Console.**WriteLine**("Thêm Menu Thành Công");

                }

                else if (table.Rows.Count < 1)

                {

                    Console.**WriteLine**("THẤT BẠI: Thêm Menu");

                }

            }

            catch (ArgumentNullException ex)

            {

                Console.**WriteLine**(ex);

            }

        }

        return **getCustomMenu**(userID);

    }

    public ObservableCollection<MenuModel> **updateCustomMenu**(int userID, int menuID, string title)

    {

        if (**isCustomMenuValid**(title, userID))

        {

            try

            {

                var cmd = DAL.**newSqlCommand**("UPDATE [dbo].[CustomMenu] SET [title] = @Title WHERE id\_customMenu = @MenuID SELECT SCOPE\_IDENTITY();");

                cmd.Parameters.**AddWithValue**("@Title", title);

                cmd.Parameters.**AddWithValue**("@MenuID", menuID);

                DataTable table = dbProcess.**getDataTableQuery**(cmd);

                if (table.Rows.Count == 1)

                {

                    Console.**WriteLine**("Sửa Task Thành Công");

                }

                else if (table.Rows.Count < 1)

                {

                    Console.**WriteLine**("THẤT BẠI: Sửa Task");

                }

            }

            catch (ArgumentNullException ex)

            {

                Console.**WriteLine**(ex);

            }

        }

        return **getCustomMenu**(userID);

    }

    public ObservableCollection<MenuModel> **deleteCustomMenu**(int userID, int menuID)

    {

        PromptDialog promptDialog = new PromptDialog("Bạn có muốn xóa công việc này?", promptInput: false);

        if (promptDialog.**ShowDialog**() == true)

        {

            try

            {

                var cmd = DAL.**newSqlCommand**("DELETE FROM [dbo].[CustomMenu] WHERE id\_customMenu = @MenuID; SELECT SCOPE\_IDENTITY();");

                cmd.Parameters.**AddWithValue**("@MenuID", menuID);

                DataTable table = dbProcess.**getDataTableQuery**(cmd);

                if (table.Rows.Count == 1)

                {

                    new PromptDialog("Xóa Thành Công!", promptInput: false, cancelButton: false).**ShowDialog**();

                    Console.**WriteLine**("Xóa Menu Thành Công");

                }

                else if (table.Rows.Count < 1)

                {

                    Console.**WriteLine**("THẤT BẠI: Xóa Menu");

                }

            }

            catch (ArgumentNullException ex)

            {

                Console.**WriteLine**(ex);

            }

        }

        return **getCustomMenu**(userID);

    }

}

#### Task

partial class **BLL**

{

    DAL dbProcess = new DAL();

    public ObservableCollection<Task> **getTaskForMenuModel**(string title, int CurrentUserID, string searchString)

    {

        var result = new ObservableCollection<Task>();

        switch (title.**ToLower**())

        {

            case "hôm nay":

                result = **getTaskListOfToday**(userID: CurrentUserID);

                break;

            case "tất cả":

                result = **getTaskListOfUser**(userID: CurrentUserID);

                break;

            case "quan trọng":

                result = **getImportantTaskList**(userID: CurrentUserID);

                break;

            case "hoàn thành":

                result = **getDoneTaskList**(userID: CurrentUserID);

                break;

            default:

                result = **getTaskListOfUser**(userID: CurrentUserID, category: title);

                break;

        }

        if (searchString != "" && searchString != null)

            result = LanguageConverter.**filterSearch**(result, searchString);

        return new ObservableCollection<Task>(result.**OrderBy**(o => o.IsStatusDone)); ;

    }

    public ObservableCollection<Task> **getTaskListOfUser**(int userID, string category = "")

    {

        string cmdCategory = category == "" ? "" : " AND Task.category = @Category";

        var cmd = DAL.**newSqlCommand**("SELECT \* FROM Task WHERE Task.id\_user = @IdUser" + cmdCategory);

        if (cmdCategory != "" && category != "")

        {

            cmd.Parameters.**AddWithValue**("@Category", category);

        }

        cmd.Parameters.**AddWithValue**("@IdUser", userID);

        SqlDataReader dr = dbProcess.**getDataQuery**(cmd);

        ObservableCollection<Task> taskList = new ObservableCollection<Task>();

        while (dr.**Read**())

        {

            Task task = new Task(

                content: dr.**GetString**(dr.**GetOrdinal**("taskContent")),

                timeCreated: dr.**GetDateTime**(dr.**GetOrdinal**("timeCreated")),

                status: dr.**GetBoolean**(dr.**GetOrdinal**("status")),

                isImportant: dr.**GetBoolean**(dr.**GetOrdinal**("isImportant")));

            int categoryColumn = dr.**GetOrdinal**("category");

            task.Category = dr.**IsDBNull**(categoryColumn) ? "" : dr.**GetString**(categoryColumn);

            task.Id = dr.**GetInt32**(dr.**GetOrdinal**("id\_task"));

            taskList.**Add**(task);

*//Console.WriteLine(task.Id + " " + task.Content + " " + task.TimeString);*

        }

        dbProcess.**CloseConnector**();

        return taskList;

    }

    public ObservableCollection<Task> **getTaskListOfToday**(int userID)

    {

        var cmd = DAL.**newSqlCommand**("SELECT \* FROM [dbo].[Task] WHERE [Task].id\_user = @UserID AND CAST(Task.timeCreated AS DATE) = CAST(GETDATE() AS DATE)");

        cmd.Parameters.**AddWithValue**("@UserID", userID);

        SqlDataReader dr = dbProcess.**getDataQuery**(cmd);

        ObservableCollection<Task> taskList = new ObservableCollection<Task>();

        while (dr.**Read**())

        {

            Task task = new Task(

                content: dr.**GetString**(dr.**GetOrdinal**("taskContent")),

                timeCreated: dr.**GetDateTime**(dr.**GetOrdinal**("timeCreated")),

                status: dr.**GetBoolean**(dr.**GetOrdinal**("status")),

                isImportant: dr.**GetBoolean**(dr.**GetOrdinal**("isImportant")));

            task.Id = dr.**GetInt32**(dr.**GetOrdinal**("id\_task"));

            task.Category = dr.**IsDBNull**(dr.**GetOrdinal**("category")) ? "" : dr.**GetString**(dr.**GetOrdinal**("category"));

            taskList.**Add**(task);

        }

        dbProcess.**CloseConnector**();

        return taskList;

    }

    public ObservableCollection<Task> **getImportantTaskList**(int userID)

    {

        var cmd = DAL.**newSqlCommand**("SELECT \* FROM [dbo].[Task] WHERE [Task].id\_user = @UserID AND [Task].isImportant = 1");

        cmd.Parameters.**AddWithValue**("@UserID", userID);

        SqlDataReader dr = dbProcess.**getDataQuery**(cmd);

        ObservableCollection<Task> taskList = new ObservableCollection<Task>();

        while (dr.**Read**())

        {

            Task task = new Task(

                content: dr.**GetString**(dr.**GetOrdinal**("taskContent")),

                timeCreated: dr.**GetDateTime**(dr.**GetOrdinal**("timeCreated")),

                status: dr.**GetBoolean**(dr.**GetOrdinal**("status")),

                isImportant: dr.**GetBoolean**(dr.**GetOrdinal**("isImportant")));

            task.Id = dr.**GetInt32**(dr.**GetOrdinal**("id\_task"));

            task.Category = dr.**IsDBNull**(dr.**GetOrdinal**("category")) ? "" : dr.**GetString**(dr.**GetOrdinal**("category"));

            taskList.**Add**(task);

        }

        dbProcess.**CloseConnector**();

        return taskList;

    }

    public ObservableCollection<Task> **getDoneTaskList**(int userID)

    {

        var cmd = DAL.**newSqlCommand**("SELECT \* FROM [dbo].[Task] WHERE [Task].id\_user = @UserID AND [Task].status = 1");

        cmd.Parameters.**AddWithValue**("@UserID", userID);

        SqlDataReader dr = dbProcess.**getDataQuery**(cmd);

        ObservableCollection<Task> taskList = new ObservableCollection<Task>();

        while (dr.**Read**())

        {

            Task task = new Task(

                content: dr.**GetString**(dr.**GetOrdinal**("taskContent")),

                timeCreated: dr.**GetDateTime**(dr.**GetOrdinal**("timeCreated")),

                status: dr.**GetBoolean**(dr.**GetOrdinal**("status")),

                isImportant: dr.**GetBoolean**(dr.**GetOrdinal**("isImportant")));

            task.Id = dr.**GetInt32**(dr.**GetOrdinal**("id\_task"));

            task.Category = dr.**IsDBNull**(dr.**GetOrdinal**("category")) ? "" : dr.**GetString**(dr.**GetOrdinal**("category"));

            taskList.**Add**(task);

*//Console.WriteLine(task.Id + " " + task.Content + " " + task.TimeString);*

        }

        dbProcess.**CloseConnector**();

        return taskList;

    }

    public ObservableCollection<Task> **addTaskForUser**(int userID, Task task, MainViewModel viewModel)

    {

        try

        {

            var cmd = DAL.**newSqlCommand**("INSERT INTO [dbo].[Task] ([id\_user], [taskContent], [timeCreated], [status], [isImportant]" + (task.Category == "" ? "" : ", [category]") + ") VALUES (@UserID, @TaskContent, @TimeCreated, @Status, @IsImportant" + (task.Category == "" ? "" : ", @Category") + ") SELECT SCOPE\_IDENTITY();");

            cmd.Parameters.**AddWithValue**("@UserID", userID);

            cmd.Parameters.**AddWithValue**("@TaskContent", task.Content);

            cmd.Parameters.**AddWithValue**("@TimeCreated", task.TimeCreated);

            cmd.Parameters.**AddWithValue**("@Status", task.IsStatusDone);

            cmd.Parameters.**AddWithValue**("@IsImportant", task.IsImportant);

            if (task.Category != "")

            {

                cmd.Parameters.**AddWithValue**("@Category", task.Category);

            }

            DataTable table = dbProcess.**getDataTableQuery**(cmd);

            if (table.Rows.Count == 1)

            {

                Console.**WriteLine**("Thêm Task Thành Công");

            }

            else if (table.Rows.Count < 1)

            {

                Console.**WriteLine**("THẤT BẠI: Thêm Task");

            }

        }

        catch (ArgumentNullException ex)

        {

            Console.**WriteLine**(ex);

        }

        return viewModel.MenuModel?.TaskInfos;

    }

    public ObservableCollection<Task> **updateTaskForUser**(int taskID, MainViewModel viewModel, bool taskStatus, bool isImportant, string updateContent)

    {

        try

        {

            DataTable table = null;

            if (updateContent != "")

            {

                var cmd = DAL.**newSqlCommand**("UPDATE [dbo].[Task] SET [taskContent] = @TaskContent WHERE [Task].id\_task = @IDTask; SELECT SCOPE\_IDENTITY();");

                cmd.Parameters.**AddWithValue**("@TaskContent", updateContent);

                cmd.Parameters.**AddWithValue**("@IDTask", taskID);

                table = dbProcess.**getDataTableQuery**(cmd);

            }

            else

            {

                var cmd = DAL.**newSqlCommand**("UPDATE [dbo].[Task] SET [status] = @Status, [isImportant] = @IsImportant WHERE [Task].id\_task = @IDTask; SELECT SCOPE\_IDENTITY();");

                cmd.Parameters.**AddWithValue**("@Status", taskStatus);

                cmd.Parameters.**AddWithValue**("@IsImportant", isImportant);

                cmd.Parameters.**AddWithValue**("@IDTask", taskID);

                table = dbProcess.**getDataTableQuery**(cmd);

            }

            if (table.Rows.Count == 1)

            {

                Console.**WriteLine**("Sửa Task Thành Công");

            }

            else if (table.Rows.Count < 1)

            {

                Console.**WriteLine**("THẤT BẠI: Sửa Task");

            }

        }

        catch (ArgumentNullException ex)

        {

            Console.**WriteLine**(ex);

        }

        return viewModel.MenuModel.TaskInfos;

    }

    public ObservableCollection<Task> **deleteTaskForUser**(int taskID, MainViewModel viewModel)

    {

        PromptDialog promptDialog = new PromptDialog("Bạn có muốn xóa công việc này?", promptInput: false);

        if (promptDialog.**ShowDialog**() == true)

        {

            if (promptDialog.DialogResult == true)

            {

                try

                {

                    var cmd = DAL.**newSqlCommand**("DELETE FROM [dbo].[Task] WHERE id\_task = @IDTask; SELECT SCOPE\_IDENTITY();");

                    cmd.Parameters.**AddWithValue**("@IDTask", taskID);

                    DataTable table = dbProcess.**getDataTableQuery**(cmd);

                    if (table.Rows.Count == 1)

                    {

                        Console.**WriteLine**("Xóa Task Thành Công");

                    }

                    else if (table.Rows.Count < 1)

                    {

                        Console.**WriteLine**("THẤT BẠI: Xóa Task");

                    }

                }

                catch (ArgumentNullException ex)

                {

                    Console.**WriteLine**(ex);

                }

            }

        }

        return viewModel.MenuModel.TaskInfos;

    }

}

#### User

partial class **BLL**

{

    DAL dbProcess = new DAL();

    [ReliabilityContract(Consistency.WillNotCorruptState, Cer.MayFail)]

    private unsafe bool **UnSafeCompareByteArray**(byte[] strA, byte[] strB)

    {

        int length = strA.Length;

        if (length != strB.Length)

        {

            return false;

        }

**fixed** (byte\* str = strA)

        {

            byte\* chPtr = str;

**fixed** (byte\* str2 = strB)

            {

                byte\* chPtr2 = str2;

                byte\* chPtr3 = chPtr;

                byte\* chPtr4 = chPtr2;

                while (length >= 10)

                {

                    if ((((\*(((int\*)chPtr3)) != \*(((int\*)chPtr4))) || (\*(((int\*)(chPtr3 + 2))) != \*(((int\*)(chPtr4 + 2))))) || ((\*(((int\*)(chPtr3 + 4))) != \*(((int\*)(chPtr4 + 4)))) || (\*(((int\*)(chPtr3 + 6))) != \*(((int\*)(chPtr4 + 6)))))) || (\*(((int\*)(chPtr3 + 8))) != \*(((int\*)(chPtr4 + 8)))))

                    {

                        break;

                    }

                    chPtr3 += 10;

                    chPtr4 += 10;

                    length -= 10;

                }

                while (length > 0)

                {

                    if (\*(((int\*)chPtr3)) != \*(((int\*)chPtr4)))

                    {

                        break;

                    }

                    chPtr3 += 2;

                    chPtr4 += 2;

                    length -= 2;

                }

                return (length <= 0);

            }

        }

    }

    public int **insertImage**(byte[] image, string fileName)

    {

        var cmd = DAL.**newSqlCommand**("SELECT \* FROM [dbo].[Avatar]");

        DataTable dt = dbProcess.**getDataTableQuery**(cmd);

        foreach (DataRow row in dt.Rows)

        {

            byte[] imgData = (byte[])row["data"];

            if (**UnSafeCompareByteArray**(image, imgData))

            {

                Console.**WriteLine**("Ảnh Đã Tồn Tại!");

                return (int)row["id\_avatar"];

            }

            else

            {

                Console.**WriteLine**("Không Giống");

            }

        }

        try

        {

            cmd = DAL.**newSqlCommand**("INSERT INTO [dbo].[Avatar]([data], [fileName]) VALUES (@Image, @Filename) SELECT SCOPE\_IDENTITY();");

            cmd.Parameters.**AddWithValue**("@Image", image);

            cmd.Parameters.**AddWithValue**("@Filename", fileName);

            dt = dbProcess.**getDataTableQuery**(cmd);

            if (dt.Rows.Count == 1)

            {

                Console.**WriteLine**("Thêm Ảnh Thành Công!");

                return int.**Parse**(dt.Rows[0][0].**ToString**());

            }

            Console.**WriteLine**("THẤT BẠI: Thêm Ảnh");

        }

        catch (Exception)

        {

*//throw;*

        }

        return 0;

    }

    public bool **createAccount**(string username, string password, string displayName, Image avatar, string fileName)

    {

        try

        {

            dbProcess.**OpenConnector**();

            int idAvatar = 0;

            if (avatar != null)

            {

                idAvatar = **insertImage**(ImgConverter.**ConvertImageToBytes**(avatar), fileName);

            }

            else

            {

                Console.**WriteLine**("null image");

            }

            var cmd = DAL.**newSqlCommand**("INSERT INTO [dbo].[User]([displayName], [id\_avatar], [username], [password]) VALUES (@displayName, @idAvatar, @username, @password)");

            cmd.Parameters.**AddWithValue**("@displayName", displayName);

            cmd.Parameters.**AddWithValue**("@idAvatar", idAvatar);

            cmd.Parameters.**AddWithValue**("@username", username);

            cmd.Parameters.**AddWithValue**("@password", password);

            int rowsAffected = cmd.**ExecuteNonQuery**();

            if (rowsAffected > 0)

            {

                Console.**WriteLine**("Thêm Thành Công");

                return true;

            }

            else if (rowsAffected <= 0)

            {

                Console.**WriteLine**("THẤT BẠI: Thêm người dùng");

                return false;

            }

            dbProcess.**CloseConnector**();

        }

        catch (Exception e)

        {

            Console.**WriteLine**(e);

        }

        return false;

    }

    public bool **updateAccount**(int id, string password, string displayName, string fileName, Image avatar = null)

    {

        try

        {

            dbProcess.**OpenConnector**();

            int idAvatar = 0;

            if (avatar != null)

            {

                idAvatar = **insertImage**(ImgConverter.**ConvertImageToBytes**(avatar), fileName);

            }

            else

            {

                Console.**WriteLine**("null image");

            }

            var cmd = DAL.**newSqlCommand**("UPDATE [dbo].[User] SET [displayName] = @displayName, [password] = @password " + (avatar != null ? ", [id\_avatar] = @idAvatar" : "") + " WHERE [User].id\_user = @userID");

            cmd.Parameters.**AddWithValue**("@displayName", displayName);

            if (idAvatar != 0 && avatar != null)

            {

                Console.**WriteLine**("add image to command");

                cmd.Parameters.**AddWithValue**("@idAvatar", idAvatar);

            }

            cmd.Parameters.**AddWithValue**("@password", password);

            cmd.Parameters.**AddWithValue**("@userID", id);

            int rowsAffected = cmd.**ExecuteNonQuery**();

            if (rowsAffected > 0)

            {

                Console.**WriteLine**("Chỉnh Sửa Thành Công");

                return true;

            }

            else if (rowsAffected <= 0)

            {

                Console.**WriteLine**("THẤT BẠI: Chỉnh sửa người dùng");

                return false;

            }

            dbProcess.**CloseConnector**();

        }

        catch (Exception e)

        {

            Console.**WriteLine**(e);

        }

        return false;

    }

    public bool **deleteAccount**(int id)

    {

        try

        {

            dbProcess.**OpenConnector**();

            var cmd = DAL.**newSqlCommand**("DELETE FROM [dbo].[User] WHERE [User].id\_user = @userID");

            cmd.Parameters.**AddWithValue**("@userID", id);

            int rowsAffected = cmd.**ExecuteNonQuery**();

            if (rowsAffected > 0)

            {

                Console.**WriteLine**("Xóa Thành Công");

                return true;

            }

            else

            {

                Console.**WriteLine**("THẤT BẠI: Xóa người dùng");

                return false;

            }

        }

        catch (Exception e)

        {

            Console.**WriteLine**(e.Message);

        }

        finally

        {

            dbProcess.**CloseConnector**();

        }

        return false;

    }

    public User **getUser**(int userID)

    {

        var cmd = DAL.**newSqlCommand**("SELECT \* FROM [dbo].[User] WHERE [User].id\_user = @UserID");

        cmd.Parameters.**AddWithValue**("@UserID", userID);

        DataTable dt = dbProcess.**getDataTableQuery**(cmd);

        if (dt.Rows.Count == 1)

        {

            var row = dt.Rows[0];

            var cmdAvatar = DAL.**newSqlCommand**("SELECT [Avatar].data, [Avatar].fileName FROM [dbo].[Avatar] WHERE [Avatar].id\_avatar = @IdAvatar");

            cmdAvatar.Parameters.**AddWithValue**("@IdAvatar", row["id\_avatar"]);

            DataTable dtAvatar = dbProcess.**getDataTableQuery**(cmdAvatar);

            Image avatar = null;

            string displayName = row["displayName"].**ToString**(), fileName = "";

            int id = (int)row["id\_user"];

            if (dtAvatar.Rows.Count == 1)

            {

                avatar = ImgConverter.**ConvertByteArrayToImage**((byte[])dtAvatar.Rows[0]["data"]);

                fileName = dtAvatar.Rows[0]["fileName"].**ToString**();

            }

            User resUser = new User(displayName, avatar);

            resUser.ID = id;

            resUser.UserName = (string)row["username"];

            resUser.PassWord = (string)row["password"];

            return resUser;

        }

        else if (dt.Rows.Count < 1)

        {

            return null;

        }

        return null;

    }

    public User **getUser**(string userName)

    {

        var cmd = DAL.**newSqlCommand**("SELECT [User].id\_user, [User].displayName, [User].id\_avatar FROM [dbo].[User] WHERE [User].username = @UserName");

        cmd.Parameters.**AddWithValue**("@UserName", userName);

        DataTable dt = dbProcess.**getDataTableQuery**(cmd);

        if (dt.Rows.Count == 1)

        {

            var row = dt.Rows[0];

            var cmdAvatar = DAL.**newSqlCommand**("SELECT [Avatar].data, [Avatar].fileName FROM [dbo].[Avatar] WHERE [Avatar].id\_avatar = @IdAvatar");

            cmdAvatar.Parameters.**AddWithValue**("@IdAvatar", row["id\_avatar"]);

            DataTable dtAvatar = dbProcess.**getDataTableQuery**(cmdAvatar);

            Image avatar = null;

            string displayName = row["displayName"].**ToString**(), fileName = "";

            int id = (int)row["id\_user"];

            if (dtAvatar.Rows.Count == 1)

            {

                avatar = ImgConverter.**ConvertByteArrayToImage**((byte[])dtAvatar.Rows[0]["data"]);

                fileName = dtAvatar.Rows[0]["fileName"].**ToString**();

            }

            User resUser = new User(displayName, avatar);

            resUser.ID = id;

            return resUser;

        }

        else if (dt.Rows.Count < 1)

        {

            return null;

        }

        return null;

    }

    public string **signIn**(string username, string password)

    {

        var cmd = DAL.**newSqlCommand**("SELECT [User].id\_user, [User].displayName, [User].id\_avatar FROM [dbo].[User] WHERE [User].username = @Username AND [User].password = @Password");

        cmd.Parameters.**AddWithValue**("@Username", username.**Trim**());

        cmd.Parameters.**AddWithValue**("@Password", password.**Trim**());

        DataTable dt = dbProcess.**getDataTableQuery**(cmd);

        try

        {

            if (dt.Rows.Count == 1)

            {

                var row = dt.Rows[0];

                User authenticatedUser = **getUser**((int)row["id\_user"]);

                MainWindow mainWindow = new MainWindow(authenticatedUser);

                mainWindow.**Show**();

                return "";

            }

            else if (dt.Rows.Count < 1)

            {

                return "Sai thông tin đăng nhập!";

            }

        }

        finally

        {

            dbProcess.**CloseConnector**();

        }

        return "";

    }

    public void **signOut**()

    {

        Login login = new Login();

        login.**Show**();

    }

}

### GUI Layer ()

#### MainViewModel

MainViewModel giống một lớp DTO để chứa dữ liệu, nhưng làm với framework WPF nên không đưa vào namespace DTO. Gộp chung vào GUI Layer để dùng làm DataContext cho GUI.

namespace **ToDoApp**.**View**

{

    public class **MainViewModel** : ViewModelBase

    {

        private BLL bus = new BLL();

        public **MainViewModel**(User \_currentUser)

        {

            MenuModels = new ObservableCollection<MenuModel>()

            {

                new MenuModel() { IconFont = "\xe755", Title = "Hôm Nay", BackColor = "#E4572E", },

                new MenuModel() { IconFont = "\xe6b6", Title = "Quan Trọng", BackColor = "#48ACF0", },

                new MenuModel() { IconFont = "\xe6e1", Title = "Hoàn Thành", BackColor = "#2E933C", },

                new MenuModel() { IconFont = "\xe614", Title = "Tất Cả", BackColor = "#1E212B", }

            };

            CustomMenuModels = new ObservableCollection<MenuModel>();

            customMenuModels = bus.**getCustomMenu**(\_currentUser.ID);

            this.CurrentUser = \_currentUser;

            this.MenuModel = MenuModels[0];

**Refresh**(\_currentUser.ID);

*//Console.WriteLine(this.CurrentUser.ID);*

            SelectedCommand = new RelayCommand<MenuModel>(t => **Select**(t));

            SelectedTaskCommand = new RelayCommand<Task>(t => **SelectedTask**(t));

        }

        private ObservableCollection<MenuModel> menuModels;

        public ObservableCollection<MenuModel> MenuModels

        {

            get { return menuModels; }

            set { menuModels = value; **RaisePropertyChanged**(); }

        }

        private ObservableCollection<MenuModel> customMenuModels;

        public ObservableCollection<MenuModel> CustomMenuModels

        {

            get { return customMenuModels; }

            set { customMenuModels = value; **RaisePropertyChanged**(); }

        }

        public RelayCommand<MenuModel> SelectedCommand { get; set; }

        public RelayCommand<Task> SelectedTaskCommand { get; set; }

        private MenuModel menuModel;

        public MenuModel MenuModel

        {

            get { return menuModel; }

            set { menuModel = value; **RaisePropertyChanged**(); }

        }

        private User currentUser = new User();

        public User CurrentUser

        {

            get { return (User)currentUser; }

            set { currentUser = value; **RaisePropertyChanged**(); }

        }

        private Task info;

        public Task Info

        {

            get { return info; }

            set { info = value; **RaisePropertyChanged**(); }

        }

        private void **Select**(MenuModel model)

        {

            MenuModel = model;

**SearchTaskInfos**("");

**Refresh**();

        }

        public void **AddCustomMenu**(string title, string icon, string backColor)

        {

            var menu = new MenuModel() { Title = title, IconFont = icon, BackColor = backColor };

            this.CustomMenuModels = bus.**createCustomMenu**(CurrentUser.ID, menu, this);

**Refresh**(CurrentUser.ID);

**Select**(MenuModels[0]);

        }

        public void **UpdateCustomMenu**(int menuID, string title)

        {

            this.CustomMenuModels = bus.**updateCustomMenu**(currentUser.ID, menuID, title);

**Refresh**();

**Select**(MenuModels[0]);

        }

        public void **DeleteCustomMenu**(int menuID)

        {

            this.CustomMenuModels = bus.**deleteCustomMenu**(currentUser.ID, menuID);

**Refresh**();

**Select**(MenuModels[0]);

        }

        public void **Refresh**(int id = 0)

        {

            if (this.MenuModel != null)

            {

                this.MenuModel.TaskInfos = this.MenuModel.TaskInfos;

            }

            foreach (MenuModel mainItem in this.MenuModels)

            {

                mainItem.TaskInfos = mainItem.TaskInfos;

*// Console.WriteLine(mainItem.TaskInfosCount);*

            }

            if (id != 0)

            {

                foreach (MenuModel mainItem in this.MenuModels)

                {

                    mainItem.CurrentUserID = id;

                    mainItem.SearchString = "";

                }

                foreach (MenuModel customItem in this.CustomMenuModels)

                {

                    customItem.CurrentUserID = id;

                    customItem.SearchString = "";

                }

            }

        }

        public void **SearchTaskInfos**(string searchText)

        {

            this.MenuModel.SearchString = searchText;

**Refresh**();

        }

        public void **AddTaskInfo**(string content, bool isImportant = false, string category = "")

        {

            if (content.Length > 0)

            {

                if (this.MenuModel?.Title != null)

                    category = this.MenuModel.Title;

                bool isComplete = false;

                switch (category.**ToLower**())

                {

                    case "hôm nay":

                    case "tất cả":

                        category = "";

                        break;

                    case "quan trọng":

                        category = "";

                        isImportant = true;

                        break;

                    case "hoàn thành":

                        category = "";

                        isComplete = true;

                        break;

                    default:

                        break;

                }

                var task = new Task(

                    content: content.**Trim**(),

                    timeCreated: DateTime.Now,

                    status: false,

                    isImportant: isImportant

                );

                task.Category = category;

                task.IsStatusDone = isComplete;

                this.MenuModel.TaskInfos = bus.**addTaskForUser**(CurrentUser.ID, task, viewModel: this);

            }

            else

                new PromptDialog("Hãy nhập thông tin vào nhé", promptInput: false, cancelButton: false).**ShowDialog**();

**Refresh**();

        }

        public void **UpdateTaskInfo**(int taskID, bool status = false, bool isImportant = false, string content = "")

        {

            this.MenuModel.TaskInfos = bus.**updateTaskForUser**(taskID: taskID, viewModel: this, updateContent: content, taskStatus: status, isImportant: isImportant);

**Refresh**();

        }

        public void **DeleteTaskInfo**(int taskID)

        {

            this.MenuModel.TaskInfos = bus.**deleteTaskForUser**(taskID, viewModel: this);

**Refresh**();

        }

        public void **SelectedTask**(Task task)

        {

            Info = task;

            Messenger.Default.**Send**(task, "Expand");

        }

    }

}

#### MainWindow

Đây là lớp chính để giao tiếp với C#.

public partial class **MainWindow** : Window

{

    BLL bus = new BLL();

    private User \_currentUser = null;

    public **MainWindow**(User user)

    {

**InitializeComponent**();

        this.MouseDown += (sender, e) =>

        {

            if (e.LeftButton == MouseButtonState.Pressed)

                this.**DragMove**();

        };

        Messenger.Default.**Register**<Task>(this, "Expand", ExpandColumn);

        \_currentUser = user;

        MainViewModel mainViewModel = new MainViewModel(\_currentUser: user);

*//mainViewModel.CurrentUser = \_currentUser;*

        this.DataContext = mainViewModel;

    }

    public void **RefreshUser**()

    {

        this.\_currentUser = bus.**getUser**(\_currentUser.ID);

    }

    private void **TextBox\_KeyDown**(object sender, KeyEventArgs e)

    {

        if (e.Key == Key.Enter)

        {

            string inputValue = inputText.Text;

            if (inputValue == "") return;

            var vm = this.DataContext as MainViewModel;

            vm.**AddTaskInfo**(inputValue);

            inputText.Text = string.Empty;

        }

    }

    private void **ExpandColumn**(Task task)

    {

        var cdf = grc.ColumnDefinitions;

        if (cdf[1].Width == new GridLength(0))

        {

            cdf[1].Width = new GridLength(280);

            btnMinimize.Foreground = new SolidColorBrush(Colors.Black);

            btnMaximize.Foreground = new SolidColorBrush(Colors.Black);

            btnClose.Foreground = new SolidColorBrush(Colors.Black);

        }

        else

        {

            cdf[1].Width = new GridLength(100);

            btnMinimize.Foreground = new SolidColorBrush(Colors.Red);

            btnMaximize.Foreground = new SolidColorBrush(Colors.Red);

            btnClose.Foreground = new SolidColorBrush(Colors.Red);

        }

    }

    private void **btnMinClick**(object sender, RoutedEventArgs e)

    {

        this.WindowState = WindowState.Minimized;

    }

    private void **btnMaxClick**(object sender, RoutedEventArgs e)

    {

        if (this.WindowState == WindowState.Maximized)

            this.WindowState = WindowState.Normal;

        else

            this.WindowState = WindowState.Maximized;

    }

    private void **btnCloseClick**(object sender, RoutedEventArgs e)

    {

        Application.Current.**Shutdown**();

    }

    private void **btnAddTask\_MouseDown**(object sender, MouseButtonEventArgs e)

    {

        string inputValue = inputText.Text;

        if (inputValue == "") return;

        var viewModel = this.DataContext as MainViewModel;

        viewModel.**AddTaskInfo**(inputValue);

        inputText.Text = string.Empty;

    }

    private void **MenuItemProfile\_Click**(object sender, RoutedEventArgs e)

    {

        var viewModel = this.DataContext as MainViewModel;

        Profile profile = new Profile(this, viewModel.CurrentUser);

        profile.**Show**();

    }

    public void **signOut**()

    {

        bus.**signOut**();

        this.**Close**();

    }

    private void **MenuItemSignOut\_Click**(object sender, RoutedEventArgs e)

    {

**signOut**();

    }

    private void **CheckBox\_CheckedChanged**(object sender, RoutedEventArgs e)

    {

        object taskBind = (object)((CheckBox)sender).Tag;

        int ID = (int)taskBind.**GetType**().**GetProperty**("Id").**GetValue**(taskBind, null);

        bool isImportant = (bool)taskBind.**GetType**().**GetProperty**("IsImportant").**GetValue**(taskBind, null);

        var viewModel = this.DataContext as MainViewModel;

        var newStatus = (bool)((CheckBox)sender).IsChecked;

        viewModel.**UpdateTaskInfo**(ID, status: newStatus, isImportant: isImportant);

        if (newStatus == true)

        {

            var outPutDirectory = Path.**GetDirectoryName**(Assembly.**GetExecutingAssembly**().CodeBase);

            var soundPath = Path.**Combine**(outPutDirectory, "Assets\\taskDone.wav");

            SoundPlayer player = new SoundPlayer(soundPath);

            player.**Play**();

        }

    }

    private void **CheckBoxImportant\_CheckedChanged**(object sender, RoutedEventArgs e)

    {

        object taskBind = (object)((CheckBox)sender).Tag;

        int ID = (int)taskBind.**GetType**().**GetProperty**("Id").**GetValue**(taskBind, null);

        bool isDone = (bool)taskBind.**GetType**().**GetProperty**("IsStatusDone").**GetValue**(taskBind, null);

        var viewModel = this.DataContext as MainViewModel;

        var isImportant = (bool)((CheckBox)sender).IsChecked;

        viewModel.**UpdateTaskInfo**(ID, status: isDone, isImportant: isImportant);

    }

    private void **UpdateTask\_Click**(object sender, RoutedEventArgs e)

    {

        var taskBind = (object)((MenuItem)sender).Tag;

*// Console.WriteLine(taskBind.ToString());*

        int ID = (int)taskBind.**GetType**().**GetProperty**("Id").**GetValue**(taskBind, null);

        string taskContent = (string)taskBind.**GetType**().**GetProperty**("Content").**GetValue**(taskBind, null);

        PromptDialog inputDialog = new PromptDialog("Chỉnh sửa Task:", taskContent);

        if (inputDialog.**ShowDialog**() == true)

        {

            var res = inputDialog.Answer;

            if (res == "")

                new PromptDialog("Hãy nhập thông tin nào", promptInput: false, cancelButton: false).**ShowDialog**();

            else if (res == taskContent) { }

            else

            {

                var viewModel = this.DataContext as MainViewModel;

                viewModel.**UpdateTaskInfo**(ID, content: res);

            }

        }

    }

    private void **DeleteTask\_Click**(object sender, RoutedEventArgs e)

    {

        int ID = (int)((MenuItem)sender).Tag;

        Console.**WriteLine**(ID.**ToString**());

        var viewModel = this.DataContext as MainViewModel;

        viewModel.**DeleteTaskInfo**(ID);

    }

    private void **CreateNewCustomMenu\_Click**(object sender, RoutedEventArgs e)

    {

        PromptDialog inputDialog = new PromptDialog("Thêm Danh Sách Mới:");

        if (inputDialog.**ShowDialog**() == true)

        {

            var res = inputDialog.Answer;

            if (res == "")

                new PromptDialog("Hãy nhập thông tin nào", promptInput: false, cancelButton: false).**ShowDialog**();

            else

            {

                var viewModel = this.DataContext as MainViewModel;

                viewModel.**AddCustomMenu**(res, "🚀", "#FFBE1A");

            }

        }

    }

    private void **UpdateCustomMenuItem\_Click**(object sender, RoutedEventArgs e)

    {

        var updateTag = (object)((MenuItem)sender).Tag;

        int ID = (int)updateTag.**GetType**().**GetProperty**("ID").**GetValue**(updateTag, null);

        string title = (string)updateTag.**GetType**().**GetProperty**("Title").**GetValue**(updateTag, null);

        PromptDialog inputDialog = new PromptDialog("Chỉnh sửa Danh Sách:", title);

        if (inputDialog.**ShowDialog**() == true)

        {

            var res = inputDialog.Answer;

            if (res == "")

                new PromptDialog("Hãy nhập thông tin nào", promptInput: false, cancelButton: false).**ShowDialog**();

            else if (res == title) { }

            else

            {

                var viewModel = this.DataContext as MainViewModel;

                viewModel.**UpdateCustomMenu**(ID, res);

            }

        }

    }

    private void **DeleteCustomMenuItem\_Click**(object sender, RoutedEventArgs e)

    {

        int ID = (int)((MenuItem)sender).Tag;

        var viewModel = this.DataContext as MainViewModel;

        viewModel.**DeleteCustomMenu**(ID);

    }

    private void **SearchText\_TextChanged**(object sender, TextChangedEventArgs e)

    {

        string searchText = txtSearch.Text;

        var viewModel = this.DataContext as MainViewModel;

        viewModel.**SearchTaskInfos**(searchText);

    }

}

**TỔNG KẾT VÀ ĐÁNH GIÁ**

* Ưu điểm:
* Chương trình có giao diện thân thiện, đơn giản cho người sử dụng.
* Hỗ trợ nhiều chức năng phù hợp với thực tế của việc lưu trữ thông tin của việc cần làm.
* Dễ dàng sử dụng và quản lý.
* Khuyết điểm:
* Do thời gian nghiên cứu có hạn và khả năng còn hạn chế nên một số vẫn chỉ còn trên ý tưởng, chưa được thực thi.
* Một số chức năng còn chưa hoạt động ổn định.
* **Hướng phát triển của phần mềm :**
* Tìm hiểu và hoàn thiện một số chức năng chưa thực hiện trong ứng dụng.
* Hỗ trợ cơ sở dữ liệu Oracle để khả năng bảo mật, tốc độ truy vấn, khả năng quản lý dữ liệu tốt hơn.

**LỜI CẢM ƠN**

Đề tài *Phần mềm ghi chú việc cần làm* là nội dung em chọn đề làm bài tập lớn kết thúc học phần theo chương trình Đại học chuyên ngành công nghệ thông tin tại trường Đại Học Hạ Long.

Để hoàn thành quá trình và hoàn thiện đề tài, em xin chân thành cảm ơn thầy Lê Anh Tú đã hướng dẫn em cách xây dựng một phần mềm và những nội dung liên quan.

Nhân dịp này, em xin chân thành cảm ơn khoa công nghệ thông tin trường Đại học Hạ Long đã tạo nhiều điều kiện trong suốt thời gian học, do dịch bệnh kéo dài nhưng với sự hướng dẫn của thầy cô và sự quan tâm của nhà trường em đã hoàn thành đề tài để báo cáo kết thúc học phần.

**Em xin trân thành cảm ơn!**

|  |  |
| --- | --- |
| **Giảng viên hướng dẫn** | **Sinh viên thực hiện**  **Nguyễn Trường Minh** |

**TÀI LIỆU THAM KHẢO**

1. Beginning Visual C# 2010, Karli Watson, Christian Nagel, Jacob Pedersen, Jon D. Reid, Morgan Skinner.
2. Lập trình viên công nghệ .Net, NXB Đại Học Khoa Học Tự Nhiên năm 2014.
3. Document về WPF: [Windows Presentation Foundation documentation](https://docs.microsoft.com/en-us/dotnet/desktop/wpf/?view=netdesktop-5.0)
4. Lý thuyết C#: <https://freetuts.net/c-sharp-la-gi-tong-quan-ve-c-sharp-1045.html>

# Mã Nguồn

## Github Repository

[TMinh25/ToDo: Microsoft To Do clone with SQL Server implement (github.com)](https://github.com/TMinh25/ToDo)

## Preview

[[Preview] \README.md - TMinh25/ToDo - GitHub1s](https://github1s.com/TMinh25/ToDo)