**ĐẠI HỌC QUỐC GIA TP.HCM**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**



**Báo cáo đồ án lập trình trực quan**

**Phần mềm quản lý ghi chú**

**Giảng viên hướng dẫn: Huỳnh Tuấn Anh**

**Thành viên nhóm: Phạm Đăng Khoa 18520930**

**Thôi Đặng Thắng Tường 18521620**

**Lớp: IT008.K11.PMCL**

**Sài Gòn, 21 tháng 12 năm 2019**

# Tổng quan

## Lý do chọn đề tài:

Nhằm để hiểu rõ và thành thạo việc xử lý các sự kiện (thao tác click, di chuyển), cũng như nhu cầu nâng cao hiệu quả trong học tập nên nhóm chúng em quyết định tạo ra phần mềm quản lý ghi chú chi tiết để phục vụ việc chú thích những nội dung trong quá trình học hỏi lâu dài.

## Phạm vi nghiên cứu đề tài:

Ngôn ngữ C# với các thao tác xử lý sự kiện đăng nhập, đăng ký, nhấp chuột cũng như tạo và quản lý được nhiều ghi chú, các ghi chú được lưu trữ vào database. Được phát triển phần mềm dựa trên design pattern là MVVM

## Công nghệ sử dụng:

Công nghệ WPF trên nền tảng .NET framework 4, được phát triển trên IDE Visual Studio 2019

## Hướng phát triển:

Nhóm em dự định sẽ them tính năng nhận dạng âm thanh và chuyển thành văn bản trên note để tiện lợi hơn cho người dùng

# Giới thiệu cơ bản về MVVM

Cũng như trong mọi kiến trúc phần mềm khác, MVVM có data binding rất cần thiết để dữ liệu có tính trực quan, người dung có thể thao tác với dữ liệu trực tiếp trên giao diện một cách sinh động. Data binding là kĩ thuật dung dể tạo gắn kết giữa phần giao diện (UI) và dữ liệu thông qua phần business logoc. Do đó, UI có thể tự động cập nhật sự thay đổi trong dữ liệu và hiển thị lại. Hơn nữa, trong mô hình MVVM thì các tầng bên dưới sẽ không biết được các thông tin gì về tầng ở trên nó. Sau đây là sơ lược về các phần trong MVVM

## View:

View là phần giao diện trong ứng dụng có chức năng hiển thị dữ liệu và đảm nhận việc xử lý khi gặp các thao tác từ người dung. Ngoài ra điểm mạnh của View trong mô hình này là nó có khả năng thực hiện hành vi và phản hồi lại người dung bằng các tính năng binding và command.

## Model:

Model là phần chứa các phương thức xử lý logic, kết nối và truy xuất đến database, mô tả dữ liệu

## ViewModel:

Là lớp trung gian giữa View và Model. ViewModel là thành phần chứa các mã lệnh cần thiết dung để thực hiện data binding và xử lý các command

# Cấu trúc của ứng dụng

## Model:

### Note:

|  |  |  |
| --- | --- | --- |
| **Tên thuộc tính** | **Mô tả thuộc tính** | **Ghi chú** |
| ID | Mã số ID dung để phân biệt với các notes khác | Khoá chính và tự động tang dần trong danh sách (AutoIncrement) |
| NotebookID | ID của các notebook | Khoá ngoại tham chiếu đến Note (1 notebook có thể có nhiều notes) |
| Title | Tiêu đề của note |  |
| CreatedTime | Thời gian tạo ra note |  |
| UpdatedTime | Thời gian note được chỉnh sửa |  |
| fileLocation | Địa chỉ mà file note được lưu |  |

### Notebook:

|  |  |  |
| --- | --- | --- |
| **Tên thuộc tính** | **Mô tả thuộc tính** | **Ghi chú** |
| Id | Mã số ID của notebook dung để phân biệt duy nhất với các notebook khác | Khoá chính của notebook và tự tang dần trong danh sách(AutoIncrement) |
| UserId | Mã số của người dung notebook đó | Khoá ngoại tham chiếu đến User (1 user có thể tạo nhiều notebooks) |
| Name | Tiêu đề của Notebook |  |

### User

|  |  |  |
| --- | --- | --- |
| **Tên thuộc tính** | **Mô tả thuộc tính** | **Ghi chú** |
| Id | Mã số của người dung để phân biệt duy nhất | Khoá chính và tự tang dần trong danh sách |
| Name | Tên người dung |  |
| Lastname | Họ của người dùng |  |
| Username | Tên tài khoản người dung tạo ra |  |
| Email | Email khi người dung đăng ký |  |
| Password | Mật khẩu |  |

## ViewModel:

### Commands:

|  |  |  |
| --- | --- | --- |
| **STT** | **Tên gọi** | **Mô tả chức năng** |
| 1 | BeginEditCommand | Bắt đầu thực hiện chức năng chỉnh sửa |
| 2 | HasEditingCommand | Kiểm tra xem người dung đã có chỉnh sửa nội dung note chưa |
| 3 | LoginCommand | Thực hiện thao tác đăng nhập cho người dung (đã đăng ký) |
| 4 | NewNotebookCommand | Tạo mới 1 notebook |
| 5 | NewNoteCommand | Tạo mới 1 note trong notebook đã lựa chọn tương ứng |
| 6 | RegisterCommand | Thực hiện thao tác đăng ký cho người dùng |

### Converters:

Thực hiện kiểm tra dữ liệu đã được chuyển đổi phù hợp để có thể hiện thị trực quan cho người dung chưa

using System;

using System.Collections.Generic;

using System.Globalization;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Data;

namespace NoteApps.ViewModel.Converters

{

class BoolToVisConverter : IValueConverter

{

//đã check hết

public object Convert(object value, Type targetType, object parameter, CultureInfo culture)

{

bool isVisible = (bool)value;

if (isVisible)

return Visibility.Visible;

return Visibility.Collapsed;

}

public object ConvertBack(object value, Type targetType, object parameter, CultureInfo culture)

{

return false;

}

}

}

### DatabaseHelper

Class này dung để thực hiện các thao tác insert, update, delete dữ liệu và đồng thời sẽ upload dữ liệu lên phần View một cách trực quan, do đó ta cần phải có phương thức Path từ class System.IO để làm việc với đường dẫn file note được lưu

### LoginVM

using SQLite;

using System;

using System.IO;

namespace NoteApps.ViewModel

{

public class DatabaseHelper

{

public static string dbFile = Path.Combine(Environment.CurrentDirectory, "notesDb.db3");

public static bool Insert<T>(T item)

{

bool result = false;

using (SQLiteConnection conn = new SQLiteConnection(dbFile))

{

conn.CreateTable<T>();

int numberOfRows=conn.Insert(item);

if (numberOfRows > 0)

return true;

}

return result;

}

public static bool Update<T>(T item)

{

bool result = false;

using (SQLiteConnection conn = new SQLiteConnection(dbFile))

{

conn.CreateTable<T>();

int numberOfRows=conn.Update(item);

if (numberOfRows > 0)

return true;

}

return result;

}

public static bool Delete<T>(T item)

{

bool result = false;

using (SQLiteConnection conn = new SQLiteConnection(dbFile))

{

conn.CreateTable<T>();

int numberOfRows=conn.Delete(item);

if (numberOfRows > 0)

return true;

}

return result;

}

}

}

Class này dung để đảm nhận chức năng binding đến LoginWindow

using NoteApps.Model;

using NoteApps.ViewModel.Commands;

using System;

namespace NoteApps.ViewModel

{

public class LoginVM

{

private User user;

public User User

{

get { return user; }

set { user = value; }

}

public RegisterCommand RegisterCommand { get; set; }

public LoginCommand LoginCommand { get; set; }

public event EventHandler HasLogIn;

public LoginVM()

{

User = new User();

RegisterCommand = new RegisterCommand(this);

LoginCommand = new LoginCommand(this);

}

public void Login()

{

try

{

using (SQLite.SQLiteConnection conn = new SQLite.SQLiteConnection(DatabaseHelper.dbFile))

{

conn.CreateTable<User>();

var user = conn.Table<User>().Where(u =>u.Username == User.Username).FirstOrDefault();

if (user.Password == User.Password)

{

App.UserID = user.Id.ToString();

HasLogIn(this, new EventArgs());

}

}

}

catch(Exception ex)

{

}

}

public void Register()

{

try

{

using (SQLite.SQLiteConnection conn = new SQLite.SQLiteConnection(DatabaseHelper.dbFile))

{

conn.CreateTable<User>();

var result = DatabaseHelper.Insert(User);

if (result)

{

App.UserID = User.Id.ToString();

HasLogIn(this, new EventArgs());

}

}

}

catch (Exception ex){//}

}

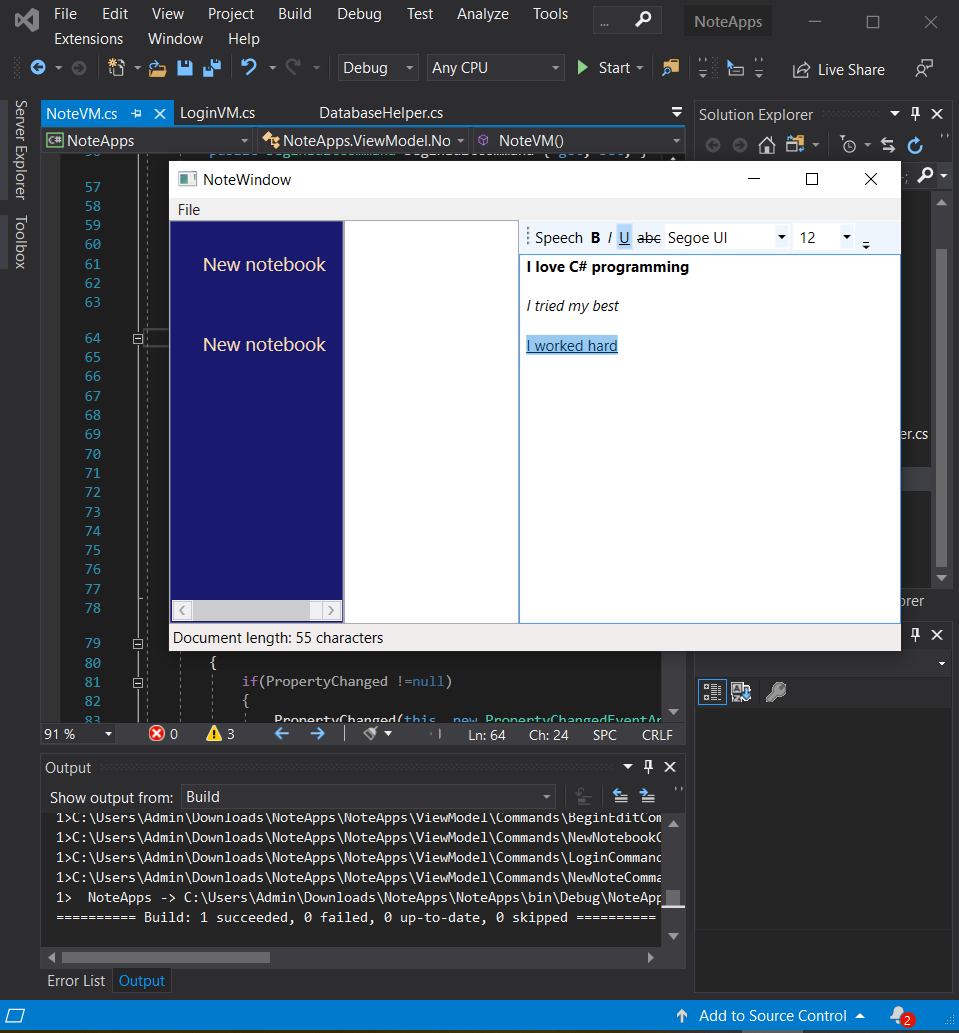
}

}

## Các Toggle Buttons:

Toggle Button có thể xem như 1 loại control có trạng thái On/Off (theo nhiều hình thức khác nhau) đồng thời những chữ minh hoạ trên Button để cho người dung biết là hiện tại có đang sử dụng nó hay không

|  |  |  |
| --- | --- | --- |
| **STT** | **Tên Button** | **Chức năng** |
| 1 | B | Làm đậm chữ |
| 2 | *I* | Làm nghiêng chữ |
| 3 | U | Gạch chân chữ |
| 4 | ~~Strikethrough~~ | Gạch xuyên chữ |



*Ảnh minh hoạ cho ToggleButton*

Như hình trên ta có thể thấy khi người dung chọn vùng chữ và click nút vào đó, Button sẽ được phủ màu lên (chứng tỏ Button đó đang ở trạng thái ON)

# Khác

## Thuận lợi và khó khan:

1. Thuận lợi:

* Một số thao tác xử lý đã có sẵn
* WPF cung cấp hệ thống API giúp xây dựng giao diện vừa đẹp vừa thuận tiện hơn

1. Khó khan:

* Mô hình MVVM còn nhiều mới lạ nên nhóm không tránh khỏi sai sót
* Chưa hoàn chỉnh được vài chức năng

## Phương hướng cải thiện:

* Tìm hiểu nhiều thuật toán để tối ưu
* Tìm hiểu them về xử lý đa luồng
* Thêm chức năng chỉnh sửa tên (notebook, note)
* Thêm nhiều fonts và cỡ chữ hơn

## Nguồn tham khảo:

Học cơ bản về WPF: <https://www.youtube.com/watch?v=Vjldip84CXQ&list=PLrW43fNmjaQVYF4zgsD0oL9Iv6u23PI6M>

Mô hình MVVM: <https://viblo.asia/p/co-ban-ve-mvvm-model-view-viewmodel-pattern-Do754wdWlM6>

Cùng một số nguồn tài liệu khác ví dụ như GitHub, Stackoverflow,…