**Міністерство освіти і науки України**

**Національний університет «Запорізька політехніка»**

Кафедра програмних засобів

**ЗВІТ**

Дисципліна «Фреймворки розробки програмного забезпечення»

Робота №8

Тема «Оформлення авторського свідоцтва на розроблену програму»

**Виконав варіант 19**

Студент КНТ-122 Онищенко О. А.

**Прийняли**

Викладач Зелік О. В.

2024

Мета

Засвоїти основні поняття про порядок та особливості оформлення авторського свідоцтва в Україні на розроблену програму, навчитися виділяти та представляти отримані результати.

Завдання

Розглянути розроблену програмну систему на предмет відповідності поставленому ТЗ.

Оформити настанову щодо використання створеного програмного продукту, що має складатися з наступних елементів:

* Призначення програми;
* Структура системи;
* Функціонування системи;
* Структура та органіація даних;
* Основні функції;
* Інтерфейсні засоби;
* Вимоги до програмного та апаратного забезпечення;
* Методика роботи користувача з системою;

Оформити заяву про реєстрацію авторського права на твір.

Оформити договір між роботодавцем та авторами комп’ютерної програми.

Створити презентацію для представлення результатів розробленого продукту.

Задача

Додати графічну взаємодію із системою.

Виконання

1 Опис

По виконанню роботи благодаттю Господа нашого Ісуса Христа було розроблено графічний застосунок взаємодії з базою даних за попередньо визначеними вимогами до програми. Програма розроблена мовою C# із застосуванням фреймворку WPF та бази даних MySQL.

2 Код

Database.cs

using MySql.Data.MySqlClient;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace seven

{

public class Database {

string query;

MySqlCommand command;

MySqlDataReader reader;

MySqlConnection connection;

public const string LastCreatedID="SELECT LAST\_INSERT\_ID();";

// SYSTEM

public Database(MySqlConnection connection) { this.connection = connection; }

public MySqlDataReader read(string query){

command=new MySqlCommand(query,connection);

return command.ExecuteReader();

}

public void write(string query){

command = new MySqlCommand(query, connection);

command.ExecuteNonQuery();

}

// CREATE

public User createUser(string name, int admin = 0, int balance=0) {

write($"INSERT INTO user (name,admin,balance) VALUES ('{name}',{admin},{balance});");

reader=read(LastCreatedID);

var user = new User();

while (reader.Read()) {

user.ID = reader.GetInt32(0);

user.Name = name;

user.Admin = admin;

user.Balance=balance;

}

reader.Close();

return user;

}

public Estate createEstate(string title, string kind, User owner, int price) {

write($"INSERT INTO estate (title,kind,owner\_id,price) VALUES ('{title}','{kind}',{owner.ID},{price});");

reader=read(LastCreatedID);

var estate = new Estate();

while (reader.Read()) {

estate.ID = reader.GetInt32(0);

estate.Title = title;

estate.Kind = kind;

estate.Owner = owner;

estate.Price=price;

}

reader.Close();

return estate;

}

public Meeting createMeeting(User sender,Estate target){

write($"INSERT INTO meeting (sender\_id,target\_id) VALUES ({sender.ID},{target.ID});");

reader=read(LastCreatedID);

var meeting = new Meeting();

while (reader.Read()) {

meeting.ID = reader.GetInt32(0);

meeting.Sender=sender;

meeting.Target=target;

}

reader.Close();

return meeting;

}

// READ

public List<User> getUsers(){

reader=read("SELECT id,name,admin,balance FROM user;");

var users=new List<User>();

while (reader.Read()) {

var user = new User();

user.ID = reader.GetInt32(0);

user.Name = reader.GetString(1);

user.Admin = reader.GetInt32(2);

user.Balance=reader.GetInt32(3);

users.Add(user);

}

reader.Close();

return users;

}

public User getUser(int id){

return getUsers().Where(u=>u.ID==id).ToList().ElementAtOrDefault(0);

}

public List<Estate> getEstates(){

reader=read("SELECT id,owner\_id,title,kind,price FROM estate;");

var estates=new List<Estate>();

var owners=new List<int>();

while (reader.Read()){

var estate=new Estate();

estate.ID=reader.GetInt32(0);

owners.Add(reader.GetInt32(1));

estate.Title=reader.GetString(2);

estate.Kind=reader.GetString(3);

estate.Price=reader.GetInt32(4);

estates.Add(estate);

}

reader.Close();

for (int i=0;i<estates.Count;i++){

estates[i].Owner=getUser(owners[i]);

}

return estates;

}

public Estate getEstate(int id){

return getEstates().Where(e=>e.ID==id).ToList().ElementAtOrDefault(0);

}

public List<Meeting> getMeetings(){

reader=read("SELECT id,sender\_id,target\_id,score,status FROM meeting;");

var meetings=new List<Meeting>();

var senders = new List<int>();

var targets = new List<int>();

while (reader.Read()) {

var meeting = new Meeting();

meeting.ID = reader.GetInt32(0);

senders.Add(reader.GetInt32(1));

targets.Add(reader.GetInt32(2));

meeting.Score = reader.GetString(3);

meeting.Status = reader.GetString(4);

meetings.Add(meeting);

}

reader.Close();

for (int i = 0; i < meetings.Count; i++) {

meetings[i].Sender = getUser(senders[i]);

meetings[i].Target = getEstate(targets[i]);

}

return meetings;

}

public Meeting getMeeting(int id){

return getMeetings().Where(m=>m.ID==id).ToList().ElementAtOrDefault(0);

}

// UPDATE

public void updateUser(User user) {

write($"UPDATE user SET name='{user.Name}',admin={user.Admin},balance={user.Balance} WHERE id={user.ID};");

}

public void updateEstate(Estate estate) {

write($"UPDATE estate SET owner\_id={estate.Owner.ID},title='{estate.Title}',kind='{estate.Kind}',price={estate.Price} WHERE id={estate.ID};");

}

public void updateMeeting(Meeting meeting) {

write($"UPDATE meeting SET sender\_id={meeting.Sender.ID},target\_id={meeting.Target.ID},score='{meeting.Score}',status='{meeting.Status}' WHERE id={meeting.ID};");

}

// DELETE

public void deleteUser(int id) {

write($"DELETE FROM user WHERE id={id};");

}

public void deleteEstate(int id) {

write($"DELETE FROM estate WHERE id={id};");

}

public void deleteMeeting(int id) {

write($"DELETE FROM meeting WHERE id={id};");

}

}

}

Estate.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace seven

{

public class Estate {

public int ID { get; set; }

public User Owner { get; set; }

public string Title { get; set; }

public string Kind { get; set; }

public int Price { get; set; } = 0;

}

}

EstateKind.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace seven

{

public static class EstateKind {

public const string Home = "Home";

public const string Flat = "Flat";

public const string New = "New";

}

}

LoginPage.xaml

<Page x:Class="seven.LoginPage"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:local="clr-namespace:seven"

mc:Ignorable="d"

Title="LoginPage" Height="85" Width="199">

<Grid>

<TextBox x:Name="UserNameInput" HorizontalAlignment="Left" Margin="10,40,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="120" Height="19" FontFamily="Verdana"/>

<Button Content="Log in" HorizontalAlignment="Left" Margin="135,40,0,0" VerticalAlignment="Top" FontFamily="Verdana" Width="47" Click="Button\_Click"/>

<Label Content="User name" HorizontalAlignment="Left" Margin="10,10,0,0" VerticalAlignment="Top" FontFamily="Verdana" Width="90" FontStretch="Normal"/>

</Grid>

</Page>

LoginPage.xaml.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace seven

{

/// <summary>

/// Interaction logic for LoginPage.xaml

/// </summary>

public partial class LoginPage : Page

{

public LoginPage()

{

InitializeComponent();

UserNameInput.Focus();

}

private void Button\_Click(object sender, RoutedEventArgs e)

{

var userName = UserNameInput.Text;

this.NavigationService.Navigate(new ProfilePage(userName));

}

}

}

Meeting.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace seven

{

public class Meeting {

public int ID { get; set; }

public User Sender { get; set; }

public Estate Target { get; set; }

public string Score { get; set; } = "Unrated";

public string Status { get; set; } = MeetingStatus.Wait;

}

}

MeetingScore.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace seven

{

public static class MeetingScore {

public const string Bad = "Bad";

public const string Okay = "Okay";

public const string Fine = "Fine";

}

}

MeetingStatus.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace seven

{

public static class MeetingStatus {

public const string Wait = "Wait";

public const string Done = "Done";

public const string Skip = "Skip";

}

}

ProfilePage.xaml

<Page x:Class="seven.ProfilePage"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:local="clr-namespace:seven"

mc:Ignorable="d"

d:DesignHeight="450" d:DesignWidth="800"

Title="ProfilePage">

<TabControl>

<TabItem Selector.Selected="homeTabOpened" x:Name="HomeTab" Header="Home">

<Grid Background="White">

<Label x:Name="UserNameLabel" Content="Name" HorizontalAlignment="Left" Margin="10,15,0,0" VerticalAlignment="Top" FontFamily="Verdana"/>

<Label x:Name="UserBalanceLabel" Content="Balance" HorizontalAlignment="Left" Margin="10,44,0,0" VerticalAlignment="Top" FontFamily="Verdana"/>

<Button x:Name="ChangeNameButton" Content="Change" HorizontalAlignment="Left" Margin="181,19,0,0" VerticalAlignment="Top" Height="17" Width="53" FontFamily="Verdana" FontSize="11" Click="ChangeNameButton\_Click"/>

<TextBox x:Name="UserNameBox" HorizontalAlignment="Left" Margin="56,19,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="120" FontFamily="Verdana"/>

<TextBox x:Name="UserBalanceBox" HorizontalAlignment="Left" Margin="70,48,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="120" FontFamily="Verdana"/>

<Button x:Name="ChangeBalanceButton" Content="Change" HorizontalAlignment="Left" Margin="195,48,0,0" VerticalAlignment="Top" Height="17" Width="53" FontFamily="Verdana" FontSize="11" Click="ChangeBalanceButton\_Click"/>

<CheckBox x:Name="UserStatusToggle" Content="Manager?" HorizontalAlignment="Left" Margin="59,74,0,0" VerticalAlignment="Top" Click="UserStatusToggle\_Click" />

<Label x:Name="UserManagerLabel" Content="Status" HorizontalAlignment="Left" Margin="10,69,0,0" VerticalAlignment="Top" FontFamily="Verdana"/>

</Grid>

</TabItem>

<TabItem Selector.Selected="availableEstatesTabOpened" x:Name="AvailableEstatesTab" Header="Available Estate">

<Grid Background="White">

<ListBox x:Name="AvailableEstatesContainer" d:ItemsSource="{d:SampleData ItemCount=5}" Margin="0,35,0,0" FontFamily="Verdana"/>

<Button x:Name="BuyEstateButton" Content="Buy" HorizontalAlignment="Left" Margin="531,7,0,0" VerticalAlignment="Top" FontFamily="Verdana" Width="43" Click="BuyEstateButton\_Click" Height="20"/>

<Button x:Name="SetMeetingButton" Content="View" HorizontalAlignment="Left" Margin="579,7,0,0" VerticalAlignment="Top" FontFamily="Verdana" Width="43" Height="20" Click="SetMeetingButton\_Click"/>

<TextBox x:Name="SellTitleInput" HorizontalAlignment="Left" Margin="44,7,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="140" FontFamily="Verdana" Height="20"/>

<TextBox x:Name="SellPriceInput" HorizontalAlignment="Left" Margin="189,7,0,0" TextWrapping="Wrap" Text="0" VerticalAlignment="Top" Width="57" FontFamily="Verdana" Height="20"/>

<ComboBox x:Name="SellKindInput" HorizontalAlignment="Left" Margin="251,7,0,0" VerticalAlignment="Top" Width="77" Height="20" FontFamily="Verdana">

<ComboBoxItem Content="Home" IsSelected="True"/>

<ComboBoxItem Content="Flat"/>

<ComboBoxItem Content="New"/>

</ComboBox>

<Button x:Name="SellButton" Content="Sell" HorizontalAlignment="Left" Margin="333,7,0,0" VerticalAlignment="Top" Height="20" FontFamily="Verdana" Width="48" Click="AeSellButton\_Click"/>

<Label x:Name="AeSelectedHeading" Content="Selected" HorizontalAlignment="Left" Margin="460,5,0,0" VerticalAlignment="Top" FontFamily="Verdana" FontWeight="Bold"/>

<Label x:Name="SellHeading" Content="Sell" HorizontalAlignment="Left" Margin="10,4,0,0" VerticalAlignment="Top" FontWeight="Bold"/>

</Grid>

</TabItem>

<TabItem Selector.Selected="ownedEstatesTabOpened" x:Name="OwnedEstatesTab" Header="Owned Estate">

<Grid Background="White">

<Grid.ColumnDefinitions>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<ListBox x:Name="OwnedEstatesContainer" d:ItemsSource="{d:SampleData ItemCount=5}" Margin="0,34,0,0" FontFamily="Verdana" SelectionChanged="OwnedEstatesContainer\_SelectionChanged"/>

<TextBox x:Name="EditTitleInput" HorizontalAlignment="Left" Margin="51,6,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="140" FontFamily="Verdana" Height="20"/>

<TextBox x:Name="EditPriceInput" HorizontalAlignment="Left" Margin="196,6,0,0" TextWrapping="Wrap" Text="0" VerticalAlignment="Top" Width="57" FontFamily="Verdana" Height="20"/>

<ComboBox x:Name="EditKindInput" HorizontalAlignment="Left" Margin="258,6,0,0" VerticalAlignment="Top" Width="77" Height="20" FontFamily="Verdana">

<ComboBoxItem Content="Home" IsSelected="True"/>

<ComboBoxItem Content="Flat"/>

<ComboBoxItem Content="New"/>

</ComboBox>

<Button x:Name="EditButton" Content="Edit" HorizontalAlignment="Left" Margin="340,6,0,0" VerticalAlignment="Top" Height="20" FontFamily="Verdana" Width="48" Click="OeEditButton\_Click"/>

<Label x:Name="EditHeading" Content="Edit" HorizontalAlignment="Left" Margin="10,4,0,0" VerticalAlignment="Top" FontWeight="Bold" FontFamily="Verdana"/>

</Grid>

</TabItem>

<TabItem Selector.Selected="incomingMeetingsTabOpened" x:Name="IncomingMeetingsTab" Header="Incoming Meetings">

<Grid Background="White">

<Grid.ColumnDefinitions>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<ListBox x:Name="IncomingMeetingsContainer" d:ItemsSource="{d:SampleData ItemCount=5}" Margin="0,40,0,0" FontFamily="Verdana" SelectionChanged="IncomingMeetingsContainer\_SelectionChanged"/>

<Button x:Name="ProcessButton" Content="Process" HorizontalAlignment="Left" Margin="162,12,0,0" VerticalAlignment="Top" Height="20" FontFamily="Verdana" Width="48" Click="ProcessButton\_Click"/>

<Label x:Name="ProcessHeading" Content="Process" HorizontalAlignment="Left" Margin="10,10,0,0" VerticalAlignment="Top" FontWeight="Bold" FontFamily="Verdana"/>

<ComboBox x:Name="ProcessInput" HorizontalAlignment="Left" Margin="80,12,0,0" VerticalAlignment="Top" Width="77" Height="20" FontFamily="Verdana">

<ComboBoxItem Content="Wait" IsSelected="True" FontFamily="Verdana"/>

<ComboBoxItem Content="Done" FontFamily="Verdana"/>

<ComboBoxItem Content="Skip" FontFamily="Verdana"/>

</ComboBox>

</Grid>

</TabItem>

<TabItem Selector.Selected="outgoingMeetingsTabOpened" x:Name="OutgoingMeetingsTab" Header="Outgoing Meetings">

<Grid Background="White">

<ListBox x:Name="OutgoingMeetingsContainer" d:ItemsSource="{d:SampleData ItemCount=5}" Margin="0,38,0,0" FontFamily="Verdana" SelectionChanged="OutgoingMeetingsContainer\_SelectionChanged"/>

<Button x:Name="RateButton" Content="Rate" HorizontalAlignment="Left" Margin="138,10,0,0" VerticalAlignment="Top" Height="20" FontFamily="Verdana" Width="48" Click="RateButton\_Click"/>

<Label x:Name="RateHeading" Content="Rate" HorizontalAlignment="Left" Margin="10,8,0,0" VerticalAlignment="Top" FontWeight="Bold" FontFamily="Verdana"/>

<ComboBox x:Name="RateInput" HorizontalAlignment="Left" Margin="56,10,0,0" VerticalAlignment="Top" Width="77" Height="20" FontFamily="Verdana">

<ComboBoxItem Content="Bad" FontFamily="Verdana"/>

<ComboBoxItem Content="Okay" FontFamily="Verdana" IsSelected="True"/>

<ComboBoxItem Content="Fine" FontFamily="Verdana"/>

</ComboBox>

</Grid>

</TabItem>

</TabControl>

</Page>

ProfilePage.xaml.cs

using MySql.Data.MySqlClient;

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.Linq;

using System.Security.Cryptography.X509Certificates;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using System.Xml.Linq;

namespace seven

{

/// <summary>

/// Interaction logic for ProfilePage.xaml

/// </summary>

public partial class ProfilePage : Page

{

public User client;

public MySqlConnection connection = new MySqlConnection(UtilityVariables.connectionString);

public Database database;

public ProfilePage(string userName) {

InitializeComponent();

connection.Open();

database = new Database(connection);

User found=database.getUsers().Where(u => u.Name == userName).ToList().ElementAtOrDefault(0);

if (found != null) {

client = found;

} else {

client = database.createUser(userName);

}

}

public void homeTabOpened(object sender, RoutedEventArgs e) {

showUserData();

}

public void ownedEstatesTabOpened(object sender, RoutedEventArgs e) {

showOwnedEstates();

}

public void availableEstatesTabOpened(object sender, RoutedEventArgs e) {

showAvailableEstates();

}

public void incomingMeetingsTabOpened(object sender, RoutedEventArgs e){

showIncomingMeetings();

}

public void outgoingMeetingsTabOpened(object sender, RoutedEventArgs e){

showOutgoingMeetings();

}

public void showUserData() {

UserNameBox.Text=client.Name;

UserBalanceBox.Text= client.Balance.ToString();

UserStatusToggle.IsChecked=Convert.ToBoolean(client.Admin);

}

public void showOwnedEstates() {

var data = database.getEstates().Where(e => e.Owner.ID == client.ID).ToList();

OwnedEstatesContainer.Items.Clear();

foreach (var e in data) {

OwnedEstatesContainer.Items.Add($"{e.ID}. {e.Title} of kind {e.Kind} price {e.Price} owned by {e.Owner.Name}");

}

EditTitleInput.Text = "";

EditKindInput.Text = EstateKind.Home;

EditPriceInput.Text = "0";

}

public void showAvailableEstates() {

var data = database.getEstates().Where(e=>e.Owner.ID!=client.ID).ToList();

AvailableEstatesContainer.Items.Clear();

foreach (var e in data) {

AvailableEstatesContainer.Items.Add($"{e.ID}. {e.Title} of kind {e.Kind} price {e.Price} owned by {e.Owner.Name}");

}

}

public void showIncomingMeetings() {

var data = database.getMeetings().Where(m => m.Target.Owner.ID == client.ID).OrderBy(m=>m.ID).Reverse().ToList();

IncomingMeetingsContainer.Items.Clear();

foreach (var m in data) {

IncomingMeetingsContainer.Items.Add($"{m.ID}. For {m.Target.Title} by {m.Sender.Name} to {m.Target.Owner.Name} rated {m.Score} status {m.Status}");

}

}

public void showOutgoingMeetings() {

var data = database.getMeetings().Where(m=>m.Sender.ID==client.ID).OrderBy(m => m.ID).Reverse().ToList();

OutgoingMeetingsContainer.Items.Clear();

foreach (var m in data) {

OutgoingMeetingsContainer.Items.Add($"{m.ID}. For {m.Target.Title} by {m.Sender.Name} to {m.Target.Owner.Name} rated {m.Score} status {m.Status}");

}

}

private void ChangeNameButton\_Click(object sender, RoutedEventArgs e) {

var name = UserNameBox.Text;

client.Name = name;

database.updateUser(client);

client = database.getUser(client.ID);

showUserData();

}

private void ChangeBalanceButton\_Click(object sender, RoutedEventArgs e) {

int balance=client.Balance;

try {

balance = Convert.ToInt32(UserBalanceBox.Text);

} catch { MessageBox.Show("Wrong balance. Please enter a number"); }

client.Balance = balance;

database.updateUser(client);

client = database.getUser(client.ID);

showUserData();

}

private void UserStatusToggle\_Click(object sender, RoutedEventArgs e) {

var status = UserStatusToggle.IsChecked;

client.Admin = Convert.ToInt32(status);

database.updateUser(client);

client = database.getUser(client.ID);

showUserData();

}

private void BuyEstateButton\_Click(object sender, RoutedEventArgs e) {

if (AvailableEstatesContainer.SelectedIndex<0) { return; }

string id = AvailableEstatesContainer.SelectedValue.ToString().Split('.')[0];

var estate = database.getEstate(int.Parse(id));

if (client.Balance < estate.Price) {

MessageBox.Show("Not enough money");

return;

}

estate.Owner = client;

client.Balance -= estate.Price;

database.updateEstate(estate);

showAvailableEstates();

}

private void SetMeetingButton\_Click(object sender, RoutedEventArgs e) {

if (AvailableEstatesContainer.SelectedIndex < 0) { return; }

string id = AvailableEstatesContainer.SelectedValue.ToString().Split('.')[0];

var estate = database.getEstate(int.Parse(id));

database.createMeeting(client, estate);

showOutgoingMeetings();

}

private void AeSellButton\_Click(object sender, RoutedEventArgs e) {

var title = SellTitleInput.Text;

var priceInput = SellPriceInput.Text;

int price;

var kind = SellKindInput.Text;

try {

price = int.Parse(priceInput);

} catch {

MessageBox.Show("Incorrect price, please enter a number");

return;

}

if (kind==EstateKind.New && client.Admin == 0) {

MessageBox.Show($"Estate of kind {EstateKind.New} may be added only by managers");

return;

}

database.createEstate(title, kind, client, price);

showAvailableEstates();

}

private void OwnedEstatesContainer\_SelectionChanged(object sender, SelectionChangedEventArgs e) {

if (OwnedEstatesContainer.SelectedIndex<0) { return; }

string id = OwnedEstatesContainer.SelectedValue.ToString().Split('.')[0];

var estate = database.getEstate(int.Parse(id));

EditTitleInput.Text = estate.Title;

EditKindInput.Text = estate.Kind;

EditPriceInput.Text = estate.Price.ToString();

}

private void OeEditButton\_Click(object sender, RoutedEventArgs e) {

if (OwnedEstatesContainer.SelectedIndex < 0) { return; }

string id = OwnedEstatesContainer.SelectedValue.ToString().Split('.')[0];

var estate = database.getEstate(int.Parse(id));

var title = EditTitleInput.Text;

var kind = EditKindInput.Text;

var priceInput = EditPriceInput.Text;

int price;

try {

price = int.Parse(priceInput);

} catch {

MessageBox.Show("Incorrect price, please enter a number");

return;

}

estate.Title = title;

estate.Price = price;

estate.Kind = kind;

database.updateEstate(estate);

showOwnedEstates();

}

private void ProcessButton\_Click(object sender, RoutedEventArgs e) {

if (IncomingMeetingsContainer.SelectedIndex < 0) { return; }

string id = IncomingMeetingsContainer.SelectedValue.ToString().Split('.')[0];

var meeting = database.getMeeting(int.Parse(id));

var status = ProcessInput.Text;

meeting.Status = status;

database.updateMeeting(meeting);

showIncomingMeetings();

}

private void RateButton\_Click(object sender, RoutedEventArgs e) {

if (OutgoingMeetingsContainer.SelectedIndex < 0) { return; }

string id = OutgoingMeetingsContainer.SelectedValue.ToString().Split('.')[0];

var meeting = database.getMeeting(int.Parse(id));

var score = RateInput.Text;

meeting.Score = score;

database.updateMeeting(meeting);

showOutgoingMeetings();

}

private void IncomingMeetingsContainer\_SelectionChanged(object sender, SelectionChangedEventArgs e) {

if (IncomingMeetingsContainer.SelectedIndex < 0) { return; }

string id = IncomingMeetingsContainer.SelectedValue.ToString().Split('.')[0];

var meeting = database.getMeeting(int.Parse(id));

ProcessInput.Text = meeting.Status;

}

private void OutgoingMeetingsContainer\_SelectionChanged(object sender, SelectionChangedEventArgs e) {

if (OutgoingMeetingsContainer.SelectedIndex < 0) { return; }

string id = OutgoingMeetingsContainer.SelectedValue.ToString().Split('.')[0];

var meeting = database.getMeeting(int.Parse(id));

RateInput.Text = meeting.Score;

}

}

}

User.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace seven

{

public class User {

public int ID { get; set; }

public string Name { get; set; }

public int Admin { get; set; } = 0;

public int Balance { get; set; } = 0;

}

}

UtilityFunctions.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace seven

{

public class UtilityFunctions {

// CHECKERS

public bool checkEstateKind(string kind)

{

return kind != EstateKind.Home && kind != EstateKind.Flat && kind != EstateKind.New ? false : true;

}

public bool checkMeetingStatus(string status)

{

return status != MeetingStatus.Wait && status != MeetingStatus.Done && status != MeetingStatus.Skip ? false : true;

}

public bool checkMeetingScore(string score)

{

return score != MeetingScore.Bad && score != MeetingScore.Okay && score != MeetingScore.Fine ? false : true;

}

// INPUT

public string getInputString(string hint)

{

Console.Write($"{hint}: ");

return Console.ReadLine();

}

public int getInputNumber(string hint)

{

var userInput = getInputString(hint);

try

{

return int.Parse(userInput);

}

catch { return -1; }

}

// FORMATTERS

public string getUserStatusString(User client)

{

return client.Admin == 0 ? "Client" : "Manager";

}

public string getEstateKindString(User client)

{

return client.Admin == 0 ? $"Estate kind ({EstateKind.Home} or {EstateKind.Flat})" : $"Estate kind ({EstateKind.Home} or {EstateKind.Flat} or {EstateKind.New})";

}

// DISPLAYS

// Estate

public void showEstates(List<Estate> estates, string header = "")

{

if (header != "")

{

Console.WriteLine($"{header} estates ({estates.Count})");

}

foreach (var e in estates)

{

Console.WriteLine($"{e.ID}. {e.Title} of kind {e.Kind} price {e.Price} owned by {e.Owner.Name}");

}

}

public bool showOwnedEstates(Database db, User user)

{

var ownedEstates = db.getEstates().Where(e => e.Owner.ID == user.ID).ToList();

if (ownedEstates.Count < 1)

{

Console.WriteLine("No owned estates");

return false;

}

else

{

showEstates(ownedEstates, "Owned");

return true;

}

}

public bool showAvailableEstates(Database db, User user)

{

var availableEstates = db.getEstates().Where(e => e.Owner.ID != user.ID).ToList();

if (availableEstates.Count < 1)

{

Console.WriteLine("No available estates");

return false;

}

else

{

showEstates(availableEstates, "Available");

return true;

}

}

// Meeting

public void showMeetings(List<Meeting> meetings, string header = "")

{

if (header != "")

{

Console.WriteLine($"{header} meetings ({meetings.Count})");

}

foreach (var m in meetings)

{

Console.WriteLine($"{m.ID}. For {m.Target.Title} by {m.Sender.Name} to {m.Target.Owner.Name} rated {m.Score} status {m.Status}");

}

}

public bool showIncomingMeetings(Database db, User user)

{

var incomingMeetings = db.getMeetings().Where(m => m.Target.Owner.ID == user.ID).ToList();

if (incomingMeetings.Count < 1)

{

Console.WriteLine("No incoming meetings");

return false;

}

else

{

showMeetings(incomingMeetings, "Incoming");

return true;

}

}

public bool showOutgoingMeetings(Database db, User user)

{

var incomingMeetings = db.getMeetings().Where(m => m.Sender.ID == user.ID).ToList();

if (incomingMeetings.Count < 1)

{

Console.WriteLine("No outgoing meetings");

return false;

}

else

{

showMeetings(incomingMeetings, "Outgoing");

return true;

}

}

}

}

UtilityVariables.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace seven

{

public static class UtilityVariables {

public const string connectionString = "uid=root;pwd=1313;host=localhost;port=3306;database=fr\_data";

}

}

3 Результати

Вигляд користувацького інтерфейсу подано нижче. Процес взаємодії із застосунком у вигляді відео [за посиланням](https://youtu.be/ZgJmRFba9wk).

Зображення, що містить текст, знімок екрана, білий, дизайн

Автоматично згенерований опис

Рисунок 3.1 – Вхід

Зображення, що містить текст, знімок екрана, монітор, програмне забезпечення

Автоматично згенерований опис

Рисунок 3.2 – Профіль

Зображення, що містить текст, знімок екрана, монітор, програмне забезпечення

Автоматично згенерований опис

Рисунок 3.3 – Ринок



Рисунок 3.4 – Наявні об’єкти нерухомості

Зображення, що містить текст, знімок екрана, Шрифт, програмне забезпечення

Автоматично згенерований опис

Рисунок 3.5 – Вхідні зустрічі

Зображення, що містить текст, знімок екрана, програмне забезпечення

Автоматично згенерований опис

Рисунок 3.6 – Вихідні зустрічі