Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Кафедра Програмної інженерії

КУРСОВА РОБОТА

ПОЯСНЮВАЛЬНА ЗАПИСКА

з дисципліни “ Об’єктно - орієнтоване програмування”

“Посібник підприємств побутового обслуговування”

Керівник , проф. Бондарєв В.М.

Студент гр. ПІ - 15 – 1 Михневич Т.К.

Комісія:

проф. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Дудар З.В.

<посада> \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_<прізвище, ініціали >

<посада> \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_<прізвище, ініціали >

Харків 2016

**ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ**

Кафедра: ***Програмної інженерії***

Дисципліна: ***Об’єктно-орієнтоване програмування***

Спеціальність: ***Програмна інженерія***

Курс 1 .Група *ПІ-15-1* . Семестр *2* .

***ЗАВДАННЯ***

***на курсовий проект студента***

*Михневич Тетяни Костянтинівни*

1. Тема проекту: «Посібник підприємств побутового обслуговування»
2. Термін здачі студентом закінченого проекту: ***“25” - травня - 2016 р***.
3. Вихідні дані до проекту: база підприємств побутового обслуговування міста: назва, розряд, адреса і телефони, спеціалізація, перелік послуг, що надаються, форма власності, години та дні роботи. Пошук підприємств за послугою та іншим критеріям.

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |
| --- | --- | --- |
| *№* | *Назва етапу* | *Термін виконання* |
| 1 | Видача теми, узгодження і затвердження теми | 1-03-2016 р. |
| 2 | Формулювання вимог до програми | 1-03-2016 – 7-03-2016 р. |
| 3 | Розробка підсистеми зберігання підприємств. | 8-03-2016 – 25-03-2016 р. |
| 4 | Розробка підсистеми пошуку підприємств. | 26-03-2016 – 15-04-2016 р. |
| 5 | Розробка функцій додавання нових, редагування та видалення підприємств. | 16-04-2016 – 25-04-2016 р. |
| 6 | Розробка функцій зберігання та завантаження даних. | 26-04-2016 – 1-05-2016 р. |
| 7 | Тестування і доопрацювання розробленої програмної системи. | 1-05-2016 – 10-05-2016 р. |
| 8 | Оформлення пояснювальної записки, додатків, графічного матеріалу | 11-05-2016 – 24-05-2016 р. |
| 9 | Захист | 25-05-2016 р. |

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Михневич Т. К.

Керівник \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Бондарєв В. М.

« 1 »\_березня\_\_\_\_\_\_\_\_\_\_ 2016 р.

РЕФЕРАТ

Пояснювальна записка до курсової роботи: 37 с., 17 рис., 1 додаток, 6 джерел.

Метою роботи є розробка програми “Посібник підприємств побутового обслуговування ” на засадах об'єктно-орієнтованого програмування.

Методи розробки базуються на використанні середи розробки Microsoft Visual Studio 2015, Windows Forms, платформи .NET Framework 4.5, мови програмування C#.

Профіль на github: https://github.com/TanyaMy

В результаті отримана програма під назвою “Посібник підприємств побутового обслуговування ”, яка дозволяє знайти підприємства, які задовольняють різним критеріям пошуку. Підприємства можна шукати за назвою, адресою, номером телефона, спеціалізацією, розрядом, формою власності, послугами, які вони надають, а також за годинами та днями роботи. Є можливість додавати, видаляти та редагувати підприємства, а також зберігати оновлений перелік підприємств після редагування.

ПРОГРАМА, ОБ’ЕКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ, КЛАС, МОВА ПРОГРАМУВАННЯ C#, ПЛАТФОРМА .NET, ПІДПРИЄМСТВО, XML, ПОШУК, КРИТЕРІЙ.

ЗМІСТ

Вступ 6

1 Специфікація програми 7

1.1 Список задач 7

1.2 Функції програми 7

1.3 Інтерфейс користувача 8

2 Проектна специфікація. 13

2.1 Архітектурний стиль 13

2.2. Об’єктна структура програми 13

2.3 Формат даних 15

3 Інструкція користувача 16

ВИСНОВКИ 21

ПЕРЕЛІК ПОСИЛАНЬ.. 22

Додаток А – Код програми.. 23

ВСТУП

Кожна людина хоче мати якісне обслуговування у різних сферах свого життя. В наш час існує велика кількість підприємств побутового обслуговування. Всі вони мають свою спеціалізацію, перелік послуг та інші критерії. Тому потрібно вміти вибрати те підприємство, яке щонайточніше буде відповідати вимогам користувача, задовольняти всі його бажання щодо обслуговування.

Саме тому посібник підприємств всього міста необхідний у повсякденному житті кожного, адже завдяки йому можна легко та швидко знайти інформацію про будь-яке підприємство, навіть якщо пам’ятати лише потрібні послуги або спеціалізацію. Іноді буває потрібно знайти підприємства, які працюють у вихідні дні, або підприємства тільки вищого розряду, або підприємства, розташовані на певній вулиці. Завдяки електронному посібнику все це стає можливим.

Дослідивши переваги та недоліки паперових носіїв інформації, будо прийнято рішення створити прикладну програму, щоб вирішити проблеми, перелічені вище.

Електронний посібник підприємств вміщає у собі інформацію про велику кількість підприємств, компаній міста, а пошук необхідного відбувається швидко і просто – це основна перевага електронного посібника перед паперовим.

1 СПЕЦИФІКАЦІЯ ПРОГРАМИ

1.1 Список задач

Метою курсового проекту є створення програми «Посібник підприємств побутового обслуговування».

Посібник має містити у собі детальну інформацію щодо підприємств, а саме: назву, адресу, номер телефона, спеціалізацію, розряд, форма власності, послуги, які вони надають, а також за години та дні роботи.

Програма повинна виконувати наступні функції: редагування інформації, зберігання та зчитування її з файлу, пошук підприємств за вибраними критеріями.

Не менш важливим є інтерфейс програми. Він повинен бути інтуїтивно зрозумілим та простим у використанні. Також необхідно, щоб користувач міг легко використати будь-яку функцію програми. Однією із найважливіших задач є правильна реакція програми на будь-які дії користувача.

1.2 Функції програми

Програма, що буде розроблена, повинна мати наступні функції:

1. відображення бази підприємств;
2. додавання нового підприємства;
3. редагування підприємства;
4. видалення підприємства;
5. пошук підприємства за будь-яким з критеріїв, перелічених нижче:

-назва;

-номер телефона;

-адреса;

-розряд;

-спеціалізація;

-послуги;

-дні та години роботи;

-форма власності.

1.3 Інтерфейс користувача

Після запуску програми користувач побачить головну форму, вона ж форма пошуку та управління базою підприємств (рисунок 1.1).

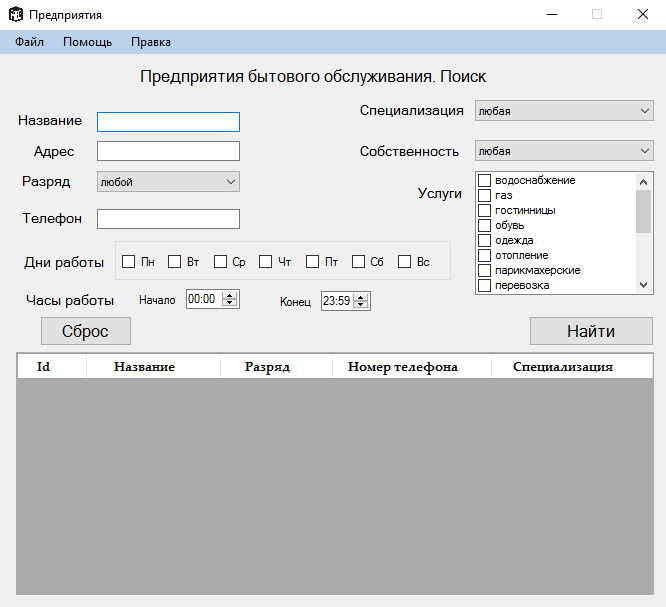


Рисунок 1.1 – Головна форма

Після заповнення полів форми та натискання кнопки «Поиск» користувач побачить ті підприємства, які задовольняють критеріям пошуку. Вибравши певне підприємство, користувач має можливість проглянути детальну інформацію про нього, або відредагувати його, або видалити. Це робиться одиничним кліком правої кнопки миші по обраному підприємству або вибором у меню: «Правка → Просмотреть\Изменить\Удалить» відповідно. (Рисунок 1.2, 1.3).

Кнопка «Сброс» очищує всі поля форми.

Детальну інформацію про підприємство можна також отримати, зробивши подвійний клік лівою кнопкою миші по обраному підприємству.

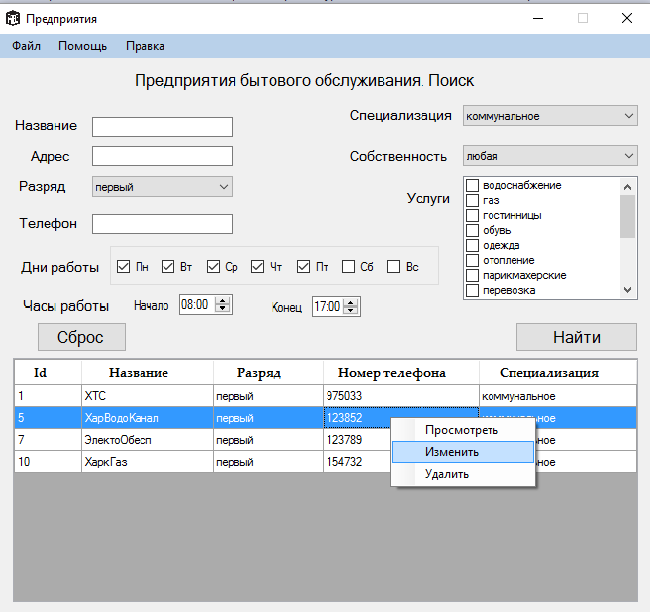


Рисунок 1.2 – Вибір функцій.

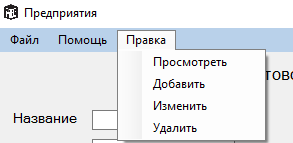


Рисунок 1.3 – Вибір функцій в головному меню.

При виборі «Просмотреть» або «Изменить» з’являється форма перегляду та редагування (Рисунок 1.4). Якщо користувач хоче відредагувати підприємство, то треба ще раз натиснути кнопку «Изменить», це зробить всі поля форми доступними для запису інформації. Кнопка «ОК» зберігає зміни, внесені користувачем, кнопка «Отмена» закриває форму без збереження змін, кнопка «Сброс» очищує всі поля форми.

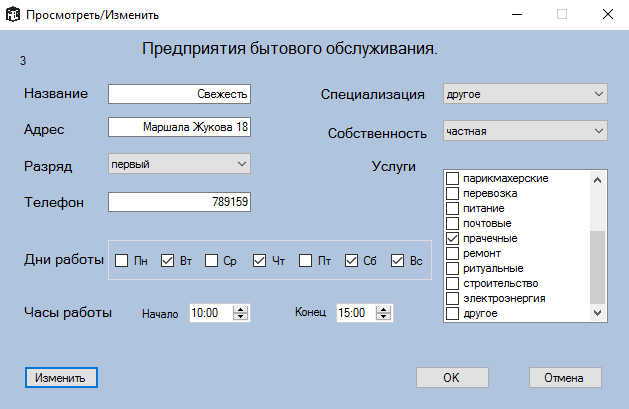


Рисунок 1.4 – Форма просмотру та редагування підприємства.

На головній формі в таблиці користувач побачить зроблені ним зміни у даному підприємстві.

Якщо користувач натискне будь-яку кнопку з меню «Правка → Просмотреть\Изменить\Удалить» не вибравши певну компанію для перегляду, редагування або видалення, то відкриється відповідне діалогове вікно «Попередження» (Рисунок 1.5, 1.6, 1.7).

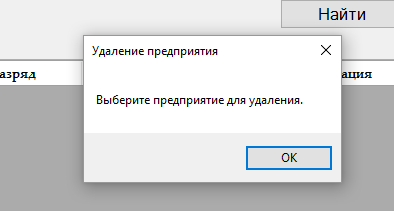
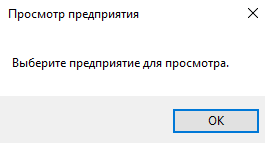
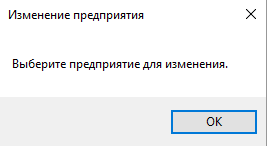


Рисунок 1.5 – Діалогове вікно «Попередження» для видалення підприємства.

Рисунки 1.6, 1.7 – Діалогове вікно «Попередження» для перегляду та редагування підприємства.

При виборі функції «Удалить» відкриється діалогове вікно підтвердження видалення компанії з бази.(Рисунок 1.8).

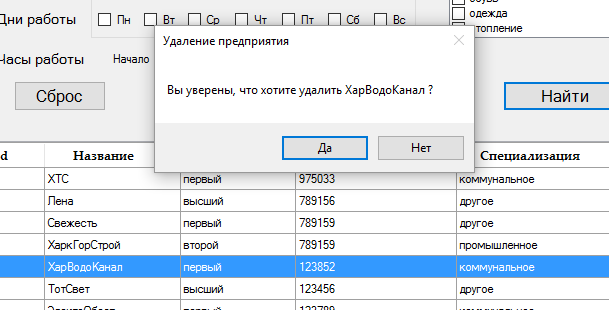


Рисунок 1.8 – Підтвердження видалення підприємства.

При виборі «Добавить» з’являється форма додавання (Рисунок 1.5). Кнопка «ОК» зберігає нове підприємство, внесене користувачем, кнопка «Отмена» закриває форму без збереження підприємства, кнопка «Сброс» очищує всі поля форми.

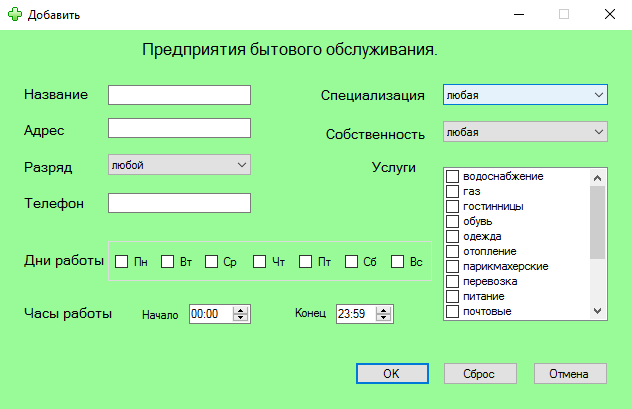


Рисунок 1.9 – Форма додавання підприємства.

2 ПРОЕКТНА СПЕЦИФІКАЦІЯ

2.1 Архітектурний стиль

Для реалізації даного проекту найкраще підходить архітектура MVC. У ній вся логіка поділяється на три окремих компоненти: Model, View, Controller.

Під Model (модель) розуміють частину програми, в якій міститься вся функціональна логіка програми.

Компонент View (зображення) відповідає за відображення даних моделі. Зображення має вільний доступ до моделі, але тільки для зчитування.

Controller (контролер) реагує на зовнішні події та змінює модель та зображення. Варто зазначити, що контролер не є посередником між моделлю і зображенням, не передає ніякої інформації з одного компонента в інший.

У процесі розробки було вирішено об’єднати контролер і зображення. Це ніяк не порушує ідею MVC. Головна задача – зробити модель повністю незалежною від інших частин програми, що дозволить робити зміни у контролері або зображенні, не впливаючи на модель.

Зараз проект працює з файлами, але у майбутньому може виникнути необхідність підключити базу даних. Дуже часто також виникає необхідність змінити зображення, не змінюючи інші частини програми. Саме через ці причини була вибрана архітектура MVC.

2.2. Об’єктна структура програми

Програма представлена набором класів, що взаємодіють між собою (рисунок 2.1).

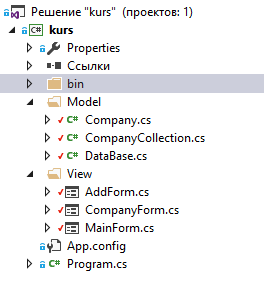


Рисунок 2.1. – Архітектура проекту

У програмі реалізовані такі класи: Company, CompanyCollection, DataBase. Класи представлені на рисунку 2.2.

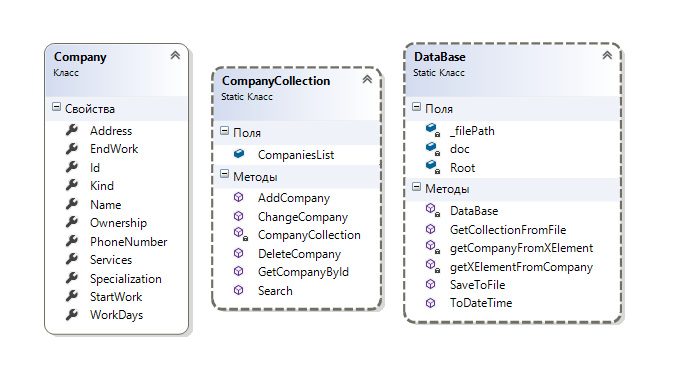


Рисунок 2.2. – Класи

2.3. Формат даних

Для зберігання інформації використовується .xml файл.

У даній програмі для запису інформації в .xml файл використовується технологія LINQ to XML. Вся функціональність LINQ to XML міститься в просторі імен System.Xml.Linq. Цей спосіб дозволяє отримувати вкладені елементи і керувати ними.

У програмі всі дані зберігаються у .xml файлі, як показано на рисунку 2.3.

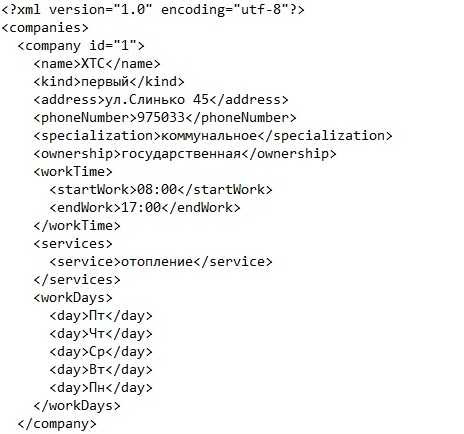


Рисунок 2.3 – Зберігання даних, фрагмент файлу XML

3 ІНСТРУКЦІЯ КОРИСТУВАЧА

Встановлення програми:

* 1. Запускаємо «Мій комп'ютер».
  2. Заходимо на носій з програмним продуктом.
  3. Розпаковуємо архів у бажану папку на комп'ютері.
  4. Можна запускати «.exe» файл і починати роботу з програмою.

Робота з програмою

Запускаємо програму. Відкривається головне вікно роботи з програмою. (Рисунок 3.1). Час, розряд, спеціалізація та форма власності вибрані за замовчуванням. Головна форма – це форма пошуку та керування базою підприємств.

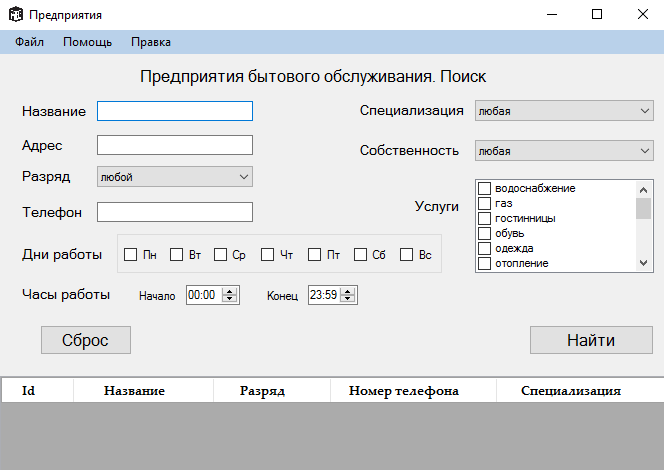


Рисунок 3.1 – Головне вікно

Пошук підприємства:

1. заповнити будь-які з критеріїв пошуку, відображених на головному вікні.;
2. у разі невірного заповнення полів натиснути «Сброс». Всі поля форми будуть очищені.
3. натиснути «Найти». У вигляді таблиці будуть відображені всі підприємства, яку задовольняють критеріям пошуку. У разі відсутності заповнених критеріїв пошуку будуть відображені всі підприємства, які збережені в базі даних.
4. якщо не існує жодного підприємства, яке задовольняє всі критерії пошуку, з’явиться вікно з інформацією про відсутність шуканого підприємства (Рисунок 3.2).

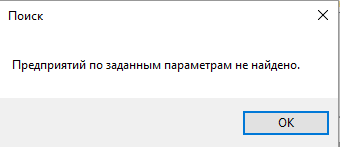


Рисунок 3.2 – Інформація про відсутність шуканого підприємства

Додавання підприємства:

1. натиснути «Правка → Добавить»;
2. відкриється форма додавання «Добавить» (Рисунок 3.3);

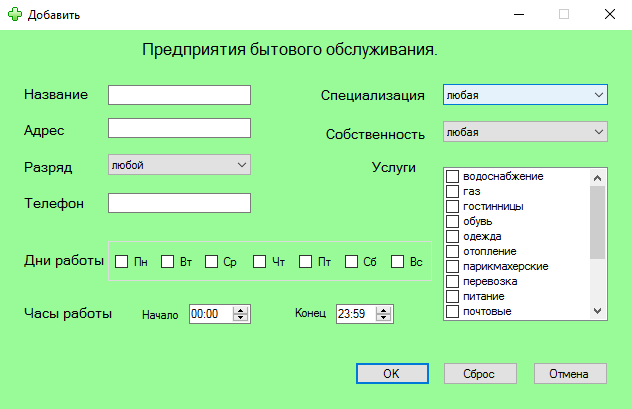


Рисунок 3.3 – Форма додавання підприємства

1. потрібно заповнити поля і натиснути «ОК»;
2. у разі невірного заповнення полів натиснути «Сброс». Всі поля форми будуть очищені.
3. нове підприємство відобразиться у базі на головній формі.

Перегляд та редагування підприємства:

а) вибрати з бази бажане підприємство;

б) натиснути «Правка → Просмотреть / Изменить» або правою кнопкою миші на обране підприємство та вибрати «Просмотреть/Изменить»;

в) відкриється форма перегляду та редагування «Просмотреть / Изменить» (Рисунок 3.4);

г) Для внесення змін натиснути кнопку «Изменить»;

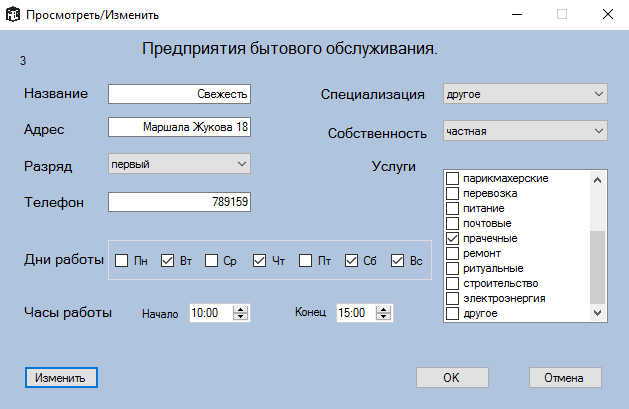


Рисунок 3.4 – Форма перегляду та редагування підприємства

д) внести необхідні зміни;

е) натиснути «ОК» для збереження змін. Для відміни операції натиснути «Отмена»;

є) підприємство буде змінено, результат можна буде побачити у базі на головній формі.

Видалення підприємства:

1. вибрати з бази бажане підприємство;
2. натиснути «Правка → Удалить» або правою кнопкою миші на обране підприємство та вибрати «Удалить»;
3. з’явиться вікно для підтвердження видалення (Рисунок 3.5).

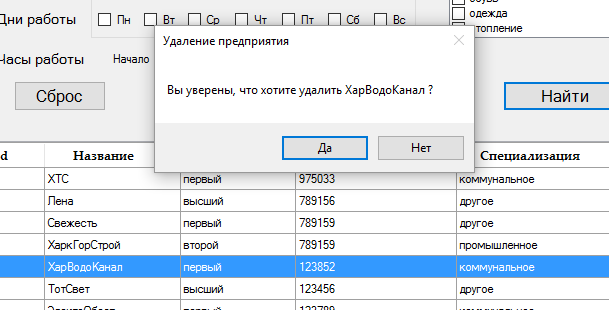


Рисунок 3.5 – Вікно підтвердження вибору

1. підтвердити свій вибір.

ВИСНОВКИ

Під час виконання курсового проекту та супроводження програмного продукту мною були пройдені всі шляхи розробки даної програми, починаючи з постановки задачі до тестування і налагодження програми.

У результаті розробки проекту була створена прикладна програма «Посібник підприємств побутового обслуговування» на платформі .Net з використанням мови C#.

Посібник містить у собі детальну інформацію щодо підприємств, а саме: назву, адресу, номер телефона, спеціалізацію, розряд, форма власності, послуги, які вони надають, а також за години та дні роботи.

Програма виконує наступні функції: редагування інформації, зберігання та зчитування її з файлу, пошук підприємств за вибраними критеріями..

Мною були докладно розглянуті такі аспекти .Net: основні принципи об’єктно-орієнтованого програмування, LINQ запити, робота з Windows Forms, запис та зчитування даних з XML-файлу, події. Був отриманий досвід у проектуванні, реалізації, документуванні повноцінної програми.

У майбутньому планується виконання деяких пунктів для покращення програми, а саме: переробка дизайну, додавання функціоналу і використання бази даних.

ПЕРЕЛІК ПОСИЛАНЬ

1. А. Хейлсберг, М. Торгерсен, С. Вилтамут, П. Голд. Язык программирования C#. Классика Computers Science. 4-е издание= C# Programming Language (Covering C# 4.0), 4th Ed. – СПб.: «Питер», 2012. – 784 с.
2. Бондарев В. М. «Объектно-ориентированное программирование на С#» [Текст]: учеб. пособ / В. М. Бондарев. – Х.: Смит, 2009. – 224 с.;
3. Дейт, К. Дж. Введення у системи баз даних [Текст] / К. Дж. Дейт.- 7-е вид. – М. : Вид. дім «Вільямс», 2001. – 846 с.
4. Троелсен: «Мова програмування С# 2005 та платформа .NET 2.0» [Текст]: учеб. пособ, 3е видання. : Пер. з англ. – Москва. : ООО «И.Д. Вільямс», 2007. – 1168с.
5. 5. Microsoft Developer Network. Бібліотека MSDN. Розробка на .NET [Електронний ресурc] / MSDN – мережа розробників Microsoft. – Доступ : www/URL:https://msdn.microsoft.com/ru-ru/ – 25.05.2015 г.
6. http://metanit.com/sharp/tutorial/16.2.php

Додаток А – Код програми

using System;

namespace kurs.Model

{

//перечисления

//Разряд

public enum Kind

{

любой,

высший,

первый,

второй

}

//Специализация

public enum Specialization

{

любая,

коммунальное,

промышленное,

производственное,

коммерческое,

другое

}

//Собственность

public enum Ownership

{

любая,

частная,

государственная,

муниципальная

}

public class Company

{

public int Id { get; set; }

public string Name { get; set; }

public Kind Kind { get; set; }

public string Address { get; set; }

public string PhoneNumber { get; set; }

public Specialization Specialization { get; set; }

public string[] Services { get; set; }

public Ownership Ownership { get; set; }

//Рабочие дни компании

public string[] WorkDays { get; set; }

//Время начала рабочего дня

public DateTime StartWork { get; set; }

//Время окончания рабочего дня

public DateTime EndWork { get; set; }

}

}

using System.Collections.Generic;

using System.Linq;

using System.ComponentModel;

namespace kurs.Model

{

/// <summary>

/// Данный класс содержит коллекцию объектов Company и методы для работы с

/// этой коллекцией, такие как удаление, добавление, изменение, поиск

/// компании по заданным параметрам.

/// </summary>

public static class CompanyCollection

{

//Список компаний, поддерживающий привязку данных.

public static BindingList<Company> CompaniesList

= new BindingList<Company>();

//Конструктор без параметров.

static CompanyCollection()

{

CompaniesList = DataBase.GetCollectionFromFile();

}

//Добавление компании в список.

public static void AddCompany(Company comp)

{

comp.Id = CompaniesList.LastOrDefault()?.Id + 1 ?? 1;

CompaniesList.Add(comp);

}

//Удаление компании по Id.

public static void DeleteCompany(int id)

{

var foundComp = CompaniesList.SingleOrDefault(x => x.Id == id);

if (foundComp != null)

CompaniesList.Remove(foundComp);

else

throw new KeyNotFoundException("Ooops... Id was not found!");

}

//Замена компании с указанным Id друой компанией.

public static void ChangeCompany(int id, Company company)

{

var foundComp = CompaniesList.SingleOrDefault(x => id == x.Id);

if (foundComp != null)

{

company.Id = id;

CompaniesList[(CompaniesList.IndexOf(foundComp))] = company;

}

else

throw new KeyNotFoundException("Ooops... Id was not found!");

}

/// <summary>

/// Поиск компаний в коллекции по заданным параметрам. Создание копии

/// списка компаний и удаление из него компаний, не удовлетворяющих

///заданным

/// критериям поиска с помощью лямбда-выражения.

/// </summary>

public static BindingList<Company> Search(Company company)

{

var collection = new List<Company>();

foreach (var z in CompaniesList)

collection.Add(z);

if (company.Services.Count() == 1 && company.Services[0] == "")

company.Services = new string[0];

if (company.WorkDays.Count() == 1 && company.WorkDays[0] == "")

company.WorkDays = new string[0];

collection.RemoveAll(x => !(x.Address.Contains(company.Address) &&

((company.Kind.ToString() != "любой") ?

x.Kind.Equals(company.Kind) : true) &&

((company.Ownership.ToString() != "любая") ?

x.Ownership.Equals(company.Ownership) : true) &&

((company.Specialization.ToString() != "любая") ?

x.Specialization.Equals(company.Specialization) : true) &&

((company.StartWork.ToString().Substring(11, 4) == "0:00" &&

company.EndWork.ToString().Substring(11, 5) == "23:59") ?

true : (x.StartWork.TimeOfDay <= company.StartWork.TimeOfDay))&&

((company.StartWork.ToString().Substring(11, 4) == "0:00" &&

company.EndWork.ToString().Substring(11, 5) == "23:59") ?

true : (x.EndWork.TimeOfDay >= company.EndWork.TimeOfDay)) &&

x.PhoneNumber.Contains(company.PhoneNumber) &&

x.Name.Contains(company.Name) &&

x.Address.Contains(company.Address) &&

x.Services.Except(company.Services).Count() ==

(x.Services.Count() - company.Services.Count()) &&

x.WorkDays.Except(company.WorkDays).Count() ==

(x.WorkDays.Count() - company.WorkDays.Count())));

var res = new BindingList<Company>();

foreach (var z in collection)

res.Add(z);

return res;

}

//получение объекта Company по указанному Id.

public static Company GetCompanyById(int id)

{

return CompaniesList.SingleOrDefault(x => x.Id == id);

}

}

}

using System;

using System.Xml.Linq;

using System.ComponentModel;

namespace kurs.Model

{

/// <summary>

/// Для работы с XML-документом: запись/удаление/изменение/чтение

/// (поиск по различным параметрам, формирование объекта Company).

/// </summary>

static class DataBase

{

//Объявление Xml-документа.

private static XDocument doc;

//Путь к Xml-документу.

private static string \_filePath = @"db.xml";

//Корневой элемент Xml-документа.

private static XElement Root;

/// <summary>

/// Статический конструктор без параметров.

/// Создание или подключение документа.

/// </summary>

static DataBase()//

{

if (System.IO.File.Exists(\_filePath))

doc = XDocument.Load(\_filePath);

else {

doc = new XDocument(new XElement("companies"));

doc.Save("db.xml");

}

Root = doc.Element("companies");

}

/// <summary>

/// Метод сохранения коллекции компаний в файл.

/// </summary>

public static void SaveToFile(BindingList<Company> list)

{

Root.RemoveAll();

foreach (var el in list)

{

Root.Add(getXElementFromCompany(el));

}

doc.Save(\_filePath);

}

/// <summary>

/// Метод создания коллекции компаний

/// с помощью чтения из файла.

/// </summary>

/// <returns>Возвращает список компаний.</returns>

public static BindingList<Company> GetCollectionFromFile()

{

var list = new BindingList<Company>();

foreach (var elem in Root.Elements())

{

var tmp = getCompanyFromXElement(elem);

if(tmp != null)

list.Add(tmp);

}

return list;

}

//Создание Xml-элемента <company> из объекта Company

private static XElement getXElementFromCompany(Company company)

{

XElement services = new XElement("services");

XElement workDays = new XElement("workDays");

foreach (string S in company.Services)

services.Add(new XElement("service", S));

foreach (string WD in company.WorkDays)

workDays.Add(new XElement("day", WD));

return new XElement

(new XElement("company",

new XAttribute("id", company.Id),

new XElement("name", company.Name),

new XElement("kind", company.Kind),

new XElement("address", company.Address),

new XElement("phoneNumber", company.PhoneNumber),

new XElement("specialization", company.Specialization),

new XElement("ownership", company.Ownership),

new XElement("workTime",

new XElement("startWork",

company.StartWork.TimeOfDay.ToString().

Substring(0, 5)),

new XElement("endWork",

company.EndWork.TimeOfDay.ToString().

Substring(0, 5))),

services,

workDays));

}

//Создание объекта Company из Xml-элемента <company>

//с помощью чтения из файла.

private static Company getCompanyFromXElement(XElement xCompany)

{

if (xCompany != null)

{

string services = "", days = "";

foreach (XElement element in xCompany.Element("services")?

.Elements("service"))

services += element.Value + " ";

foreach (XElement element in xCompany.Element("workDays")?

.Elements("day"))

days += element.Value + " ";

var company = new Company(){

Id = Convert.ToInt32(xCompany.FirstAttribute?.Value ?? "-1"),

Name = xCompany.Element("name")?.Value ?? "emptyName",

Kind = (Kind)Enum.Parse(typeof(Kind),

xCompany.Element("kind")?

.Value ?? "любой", true),

Address = xCompany.Element("address")?.Value ?? "empty",

Ownership = (Ownership)Enum.Parse(typeof(Ownership),

xCompany.Element("ownership")?.Value ?? "любая", true),

PhoneNumber = xCompany.Element("phoneNumber")?.Value ??

"empty",

Specialization =

(Specialization)Enum.Parse(typeof(Specialization),

xCompany.Element("specialization")?.Value ?? "любая",

true),

Services = services.Trim().Split(' '),

WorkDays = days.Trim().Split(' '),

StartWork = ToDateTime(xCompany.Element("workTime")

.Element("startWork")?.Value ?? "00:00"),

EndWork = ToDateTime(xCompany.Element("workTime")

.Element("endWork")?.Value ?? "23:59")

};

return company;

}

return null;

}

// Метод перевода строки формата "hh:mm" в формат DateTime

public static DateTime ToDateTime(string time)

{

return Convert.ToDateTime("01/01/1800 " + time + ":00.00");

}

}

}

using kurs.Model;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Windows.Forms;

namespace kurs.View

{

// Форма для поиска по различным параметрам, вызова форм

//просмотра/изменения/добавления компании.

public partial class MainForm : Form

{

// Конструктор без параметров.

public MainForm()

{

InitializeComponent();

}

//Заполнение элементов управления ComboBox значениями

//по умолчанию при загрузке формы.

private void MainForm\_Load(object sender, EventArgs e)

{

foreach (Control c in this.Controls)

{

if (c is ComboBox)

{

ComboBox combo = (ComboBox)c;

combo.SelectedIndex = 0;

}

}

}

//Информация о программе в MessagaBox при выборе

//в меню "Помощь -> О программе".

private void aboutTStrMenu\_Click(object sender, EventArgs e)

{

MessageBox.Show("Курсовая работа\nПрограмма \"Справочник

потребителя\"" +

" содержит базу предприятий бытового обслуживания города.

Возможен" +

" поиск предприятий по заданным параметрам. \nВыполнила:" +

" Михневич Т.К.\nст.гр ПИ-15-1\n2016", "О программе");

}

// Зактрытие формы по нажатию "Выход".

private void exitTStrMenu\_Click(object sender, EventArgs e)

{

Close();

}

// Открытие формы для добавления компании

//по выбору в меню "Правка -> Добавить".

private void addTStrMenu\_Click(object sender, EventArgs e)

{

AddForm form = new AddForm();

form.ShowDialog();

var collection = CompanyCollection.Search(getCompanyForSearch());

bindListToDataGridView(collection);

}

/// <summary>

/// По нажатию кнопки "Найти" вызов метода создания экземпляра класса

/// Company с заполненной формы для поиска, затем вызов метода поиска

/// компаний, удовлетворяющих критериям поиска

/// и заполнение DataGridView информацией о найденных компаниях.

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void findbtn\_Click(object sender, EventArgs e)

{

var comp = getCompanyForSearch();

var collection = CompanyCollection.Search(comp);

bindListToDataGridView(collection);

if (collection.Count == 0)

{

DialogResult dialogRes =

MessageBox.Show("Предприятий по заданным параметрам не

найдено.",

"Поиск", MessageBoxButtons.OK);

return;

}

}

// Вызов формы с информацией по выбранной компании

//по двойному щелчку по строке в DataGridView.

private void dataGV\_CellDoubleClick(object sender,

DataGridViewCellEventArgs e)

{

if (e.RowIndex < 0) return;

int ID = Convert.ToInt32(dataGV[0, e.RowIndex].Value);

CompanyForm form =

new CompanyForm(CompanyCollection.

GetCompanyById(Convert.ToInt32(dataGV[0, e.RowIndex].Value)));

form.Writable(false);

form.ShowDialog();

var collection = CompanyCollection.Search(getCompanyForSearch());

bindListToDataGridView(collection);

}

//Сохранение изменений в документе при выборе в меню "Файл -> Сохранить".

private void saveTStrMenu\_Click(object sender, EventArgs e)

{

DataBase.SaveToFile(CompanyCollection.CompaniesList);

}

//Сохранение изменений в документе при закрытии формы.

private void MainForm\_FormClosing(object sender, FormClosingEventArgs e)

{

DataBase.SaveToFile(CompanyCollection.CompaniesList);

}

//Вызов контекстного меню по событию "Нажатие правой клавишей

//мыши по выбранной компании в DataGridView"

private void dataGV\_CellContextMenuStripNeeded(object sender,

DataGridViewCellContextMenuStripNeededEventArgs e)

{

if (e.ColumnIndex < 0 || e.RowIndex < 0) return;

dataGV[e.ColumnIndex, e.RowIndex].Selected = true;

dataGV.ContextMenuStrip.Show(new System.Drawing.Point());

}

//Удаление выбранной компании в DataGridView по выбору

//в контекстном меню "Удалить".

private void deleteTSM\_Click(object sender, EventArgs e)

{

try

{

int ID = Convert.ToInt32(dataGV[0,

dataGV.SelectedRows[0].Index].Value);

DialogResult dialogResult = MessageBox.Show("Вы уверены?",

"Удаление предприятия", MessageBoxButtons.YesNo);

if (dialogResult == DialogResult.Yes)

{

CompanyCollection.DeleteCompany

(Convert.ToInt32(dataGV[0,

dataGV.SelectedRows[0].Index].Value));

var collection = CompanyCollection.

Search(getCompanyForSearch());

bindListToDataGridView(collection);

}

}

catch (ArgumentOutOfRangeException)

{

DialogResult dialogRes =

MessageBox.Show("Выберите предприятие для удаления.",

"Удаление предприятия", MessageBoxButtons.OK);

return;

}

catch (KeyNotFoundException)

{

DialogResult dialogRes =

MessageBox.Show("Выберите предприятие для удаления.",

"Удаление предприятия", MessageBoxButtons.OK);

return;

}

}

//Изменение выбранной компании в DataGridView по выбору в контекстном меню

//"Изменить". Вызов формы для внесения изменений в выбранную компанию.

private void changeTSM\_Click(object sender, EventArgs e)

{

try {

int ID = Convert.ToInt32(dataGV[0,

dataGV.SelectedRows[0].Index].Value);

CompanyForm form = new CompanyForm

(CompanyCollection.GetCompanyById(ID));

form.Writable(false);

form.ShowDialog();

var collection = CompanyCollection.Search(getCompanyForSearch());

bindListToDataGridView(collection);

}

catch (ArgumentOutOfRangeException)

{

DialogResult dialogResult =

MessageBox.Show("Выберите предприятие для изменения.",

"Изменение предприятия", MessageBoxButtons.OK);

return;

}

catch (KeyNotFoundException)

{

DialogResult dialogResult =

MessageBox.Show("Выберите предприятие для изменения.",

"Изменение предприятия", MessageBoxButtons.OK);

return;

}

}

//Показ информации о выбранной компании в DataGridView по

//выбору в контекстном меню "Просмотреть".

private void showTSM\_Click(object sender, EventArgs e)

{

try {

int ID = Convert.ToInt32(dataGV[0,

dataGV.SelectedRows[0].Index].Value);

CompanyForm form = new CompanyForm

(CompanyCollection.GetCompanyById

(Convert.ToInt32(dataGV[0,

dataGV.SelectedRows[0].Index].Value)));

form.Writable(false);

form.ShowDialog();

var collection = CompanyCollection.Search(getCompanyForSearch());

bindListToDataGridView(collection);

}

catch (ArgumentOutOfRangeException)

{

DialogResult dialogResult =

MessageBox.Show("Выберите предприятие для просмотра.",

"Просмотр предприятия", MessageBoxButtons.OK);

return;

}

}

//Создание объекта Company с информацией, введенной в форме для поиска.

private Company getCompanyForSearch()

{

var company = new Company()

{

Name = nameTBox.Text,

Address = addressTBox.Text,

PhoneNumber = phoneNumberTBox.Text,

Kind = (Kind)Enum.Parse(typeof(Kind), kindCBox.Text, true),

Specialization =

(Specialization)Enum.Parse(typeof(Specialization),

specializationCBox.Text, true),

Ownership = (Ownership)Enum.Parse(typeof(Ownership),

ownershipCBox.Text, true),

StartWork = startDTPicker.Value,

EndWork = endDTPicker.Value

};

string days = "";

foreach (var ch in workdaysGB.Controls)

{

CheckBox TB = (CheckBox)ch;

if (TB.Checked)

days += TB.Text + " ";

}

company.WorkDays = days.Trim().Split(' ');

string serv = "";

foreach (var ch in servicesChLB.CheckedItems)

serv += ch + " ";

company.Services = serv.Trim().Split(' ');

return company;

}

//Привязка списка компаний к DataGridView.

private void bindListToDataGridView(BindingList<Company> list)

{

var bindingList = list;

var source = new BindingSource(bindingList, null);

dataGV.DataSource = source;

}

//Сброс введенной в форме информации.

private void resetbtn\_Click(object sender, EventArgs e)

{

foreach (Control c in Controls)

if (c is TextBox)

((TextBox)c).Text = "";

else if (c is ComboBox)

((ComboBox)c).SelectedIndex = 0;

else if (c is GroupBox)

{

GroupBox gb = (GroupBox)c;

foreach (CheckBox ch in gb.Controls)

ch.Checked = false;

}

else if (c is DateTimePicker)

{

DateTimePicker dtp = (DateTimePicker)c;

if (dtp.Name == "startDTPicker")

dtp.Value =

Convert.ToDateTime("01/01/1800 " + "00:00" +

":00.00");

else if (dtp.Name == "endDTPicker")

dtp.Value =

Convert.ToDateTime("01/01/1800 " + "23:59" +

":00.00");

}

else if (c is CheckedListBox)

{

CheckedListBox clb = (CheckedListBox)c;

for (int i = 0; i < clb.Items.Count; i++)

clb.SetItemChecked(i, false);

}

}

//Проверка на правильность введенной даты

//(дата начала рабочего дня не должна опережать дату завершения рабочего

//дня)по событию "Изменение даты начала рабочего дня".

private void startDTPicker\_ValueChanged(object sender, EventArgs e)

{

DateTime startValue = endDTPicker.Value;

startDTPicker.MaxDate = startValue.AddMinutes(-1); ;

if (startDTPicker.MaxDate >= endDTPicker.Value)

{

if (startValue.Minute > 0)

startValue.AddMinutes(-1);

else {

startValue.AddMinutes(59);

startValue.AddHours(-1);

}

}

else

startValue.AddMinutes(-1);

}

//Проверка на правильность введенной даты

//(дата начала рабочего дня не должна опережать дату завершения рабочего

//дня)по событию "Изменение даты конца рабочего дня".

private void endDTPicker\_ValueChanged(object sender, EventArgs e)

{

if(startDTPicker.Value >= endDTPicker.Value)

{

endDTPicker.Value = startDTPicker.Value.AddMinutes(1);

}

}

}

}

using kurs.Model;

using System;

using System.Windows.Forms;

namespace kurs.View

{

// Форма для просмотра и изменения компании с заполненными полями.

public partial class CompanyForm : Form

{

// Конструктор без параметров.

public CompanyForm()

{

InitializeComponent();

}

//Конструктор с параметром. Принимает компанию и заполняет поля

//формы информацией об этой компании.

public CompanyForm(Company comp)

{

InitializeComponent();

fillFields(comp);

}

//Создание новой компании и запись в нее информации, введенной в форме.

private Company createCompanyFromForm()

{

Company comp = new Company()

{

Id = Convert.ToInt32(IDlbl.Text),

Name = nameTBox.Text,

Address = addressTBox.Text,

PhoneNumber = phoneNumberTBox.Text,

Kind = (Kind)Enum.Parse(typeof(Kind), kindCBox.Text, true),

Specialization =

(Specialization)Enum.Parse(typeof(Specialization),

specializationCBox.Text, true),

Ownership = (Ownership)Enum.Parse(typeof(Ownership),

ownershipCBox.Text, true),

StartWork = startDTPicker.Value,

EndWork = endDTPicker.Value

};

string days = "";

foreach (var ch in workdaysGB.Controls)

{

CheckBox TB = (CheckBox)ch;

if (TB.Checked)

days += TB.Text + " ";

}

comp.WorkDays = days.Trim().Split(' ');

string serv = "";

foreach (var ch in servicesChLB.CheckedItems)

serv += ch + " ";

comp.Services = serv.Trim().Split(' ');

return comp;

}

//Принимает компанию и заполняет поля формы информацией об этой компании.

private void fillFields(Company comp)

{

IDlbl.Text = "" + comp.Id;

nameTBox.Text = comp.Name;

phoneNumberTBox.Text = comp.PhoneNumber;

addressTBox.Text = comp.Address;

kindCBox.Text = comp.Kind.ToString();

ownershipCBox.Text = comp.Ownership.ToString();

specializationCBox.Text = comp.Specialization.ToString();

endDTPicker.Value = comp.EndWork;

startDTPicker.Value = comp.StartWork;

foreach (var s in comp.WorkDays)

foreach (var el in workdaysGB.Controls)

{

var El = el as CheckBox;

if (El.Text.Equals(s, StringComparison.OrdinalIgnoreCase))

El.Checked = true;

}

var arr = servicesChLB.Items;

foreach (var s in comp.Services)

for (int i = 0; i < servicesChLB.Items.Count; i++)

if (arr[i].ToString().Equals(s,

StringComparison.OrdinalIgnoreCase))

servicesChLB.SetItemChecked(i, true);

}

// Закрытие окна по нажатию кнопки "Отмена".

private void cancelbtn\_Click(object sender, EventArgs e)

{

Close();

}

// Сохранение изменений по нажатию кнопки "ОК".

private void okbtn\_Click(object sender, EventArgs e)

{

var comp = createCompanyFromForm();

CompanyCollection.ChangeCompany(comp.Id, comp);

}

// Делает поля доступными для записи по нажатию кнопки "Изменить".

private void changebtn\_Click(object sender, EventArgs e)

{

Writable(true);

}

/// <summary>

/// Делает поля формы доступными только для чтения и для записи

/// в зависимости от введенного параметра - булевого значения.

/// </summary>

/// <param name="verif"></param>

public void Writable(bool verif)

{

foreach (var element in Controls)

{

if (element is TextBox)

{

TextBox tb = (TextBox)element;

tb.ReadOnly = !verif;

}

else if (element is DateTimePicker)

{

DateTimePicker dtp = (DateTimePicker)element;

dtp.Enabled = verif;

}

else if (element is ComboBox)

{

ComboBox cb = (ComboBox)element;

cb.Enabled = verif;

}

else if (element is CheckBox)

{

CheckBox cb = (CheckBox)element;

cb.Enabled = verif;

}

else if (element is CheckedListBox)

{

CheckedListBox cb = (CheckedListBox)element;

cb.Enabled = verif;

}

else if (element is GroupBox)

{

GroupBox gb = (GroupBox)element;

foreach (var tmp in gb.Controls)

{

var ch = (CheckBox)tmp;

ch.Enabled = verif;

}

}

}

}

//Проверка на правильность введенной даты

//(дата начала рабочего дня не должна опережать дату завершения рабочего

//дня)по событию "Изменение даты начала рабочего дня".

private void startDTPicker\_ValueChanged(object sender, EventArgs e)

{

DateTime startValue = endDTPicker.Value;

startDTPicker.MaxDate = startValue.AddMinutes(-1); ;

if (startDTPicker.MaxDate >= endDTPicker.Value)

{

if (startValue.Minute > 0)

startValue.AddMinutes(-1);

else {

startValue.AddMinutes(59);

startValue.AddHours(-1);

}

}

else

startValue.AddMinutes(-1);

}

//Проверка на правильность введенной даты

//(дата начала рабочего дня не должна опережать дату завершения рабочего

//дня)по событию "Изменение даты конца рабочего дня".

private void endDTPicker\_ValueChanged(object sender, EventArgs e)

{

if (startDTPicker.Value >= endDTPicker.Value)

{

endDTPicker.Value = startDTPicker.Value.AddMinutes(1);

}

}

}

}

using kurs.Model;

using System;

using System.Windows.Forms;

namespace kurs.View

{

// Форма для добавления компании.

public partial class AddForm : Form

{

//Конструктор без параметров.

public AddForm()

{

InitializeComponent();

}

// Закрытие окна по нажатию кнопки "Отмена".

private void cancelbtn\_Click(object sender, EventArgs e)

{

Close();

}

//Создание новой компании и запись в нее информации, введенной в форме.

private Company createCompanyFromForm()

{

Company comp = new Company()

{

Id = -1,

Name = nameTBox.Text,

Address = addressTBox.Text,

PhoneNumber = phoneNumberTBox.Text,

Kind = (Kind)Enum.Parse(typeof(Kind), kindCBox.Text, true),

Specialization =

(Specialization)Enum.Parse(typeof(Specialization),

specializationCBox.Text, true),

Ownership = (Ownership)Enum.Parse(typeof(Ownership),

ownershipCBox.Text, true),

StartWork = startDTPicker.Value,

EndWork = endDTPicker.Value

};

string days = "";

foreach (var ch in workdaysGB.Controls)

{

CheckBox TB = (CheckBox)ch;

if (TB.Checked)

days += TB.Text + " ";

}

comp.WorkDays = days.Trim().Split(' ');

string serv = "";

foreach (var ch in servicesChLB.CheckedItems)

serv += ch + " ";

comp.Services = serv.Trim().Split(' ');

return comp;

}

// Добавление компаниии по нажатию кнопки "ОК".

private void okbtn\_Click(object sender, EventArgs e)

{

CompanyCollection.AddCompany(createCompanyFromForm());

}

//Сброс введенной в форме информации.

private void resetbtn\_Click(object sender, EventArgs e)

{

foreach (Control c in Controls)

if (c is TextBox)

((TextBox)c).Text = "";

else if (c is ComboBox)

((ComboBox)c).SelectedIndex = 0;

else if (c is GroupBox)

{

GroupBox gb = (GroupBox)c;

foreach (CheckBox ch in gb.Controls)

ch.Checked = false;

}

else if (c is DateTimePicker)

{

DateTimePicker dtp = (DateTimePicker)c;

if (dtp.Name == "startDTPicker")

dtp.Value =

Convert.ToDateTime("01/01/1800 " + "00:00" +

":00.00");

else if (dtp.Name == "endDTPicker")

dtp.Value =

Convert.ToDateTime("01/01/1800 " + "23:59" +

":00.00");

}

else if (c is CheckedListBox)

{

CheckedListBox clb = (CheckedListBox)c;

for (int i = 0; i < clb.Items.Count; i++)

clb.SetItemChecked(i, false);

}

}

//Заполнение элементов управления ComboBox значениями

//по умолчанию при загрузке формы.

private void AddForm\_Load(object sender, EventArgs e)

{

foreach (Control c in this.Controls)

{

if (c is ComboBox)

{

ComboBox combo = (ComboBox)c;

combo.SelectedIndex = 0;

}

}

}

//Проверка на правильность введенной даты

//(дата начала рабочего дня не должна опережать дату завершения рабочего

//дня)по событию "Изменение даты начала рабочего дня".

private void startDTPicker\_ValueChanged(object sender, EventArgs e)

{

DateTime startValue = endDTPicker.Value;

startDTPicker.MaxDate = startValue.AddMinutes(-1); ;

if (startDTPicker.MaxDate >= endDTPicker.Value)

{

if (startValue.Minute > 0)

startValue.AddMinutes(-1);

else {

startValue.AddMinutes(59);

startValue.AddHours(-1);

}

}

else

startValue.AddMinutes(-1);

}

//Проверка на правильность введенной даты

//(дата начала рабочего дня не должна опережать дату завершения рабочего

//дня)по событию "Изменение даты конца рабочего дня".

private void endDTPicker\_ValueChanged(object sender, EventArgs e)

{

if (startDTPicker.Value >= endDTPicker.Value)

{

endDTPicker.Value = startDTPicker.Value.AddMinutes(1);

}

}

}

}