Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Кафедра Програмної інженерії

КУРСОВА РОБОТА

ПОЯСНЮВАЛЬНА ЗАПИСКА

з дисципліни “Об’єктно - орієнтоване програмування”

“Посібник начальника в'язниці”

Керівник , проф. Бондарєв В.М.

Студент гр. ПІ - 15 – 1 Черкас Б. І.

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***

*Черкаса Бориса Ігоровича*

1. Тема проекту: «Посібник начальника в'язниці».
2. Термін здачі студентом закінченого проекту: ***“2” - червня - 2016 р***.
3. Вихідні дані до проекту: анкетні дані ув'язнених, стаття, дата взяття під варту, місце в тюремній ієрархії, камера, відомості про родичів, особливості характеру. Формування статичних зведень про склад, вибір за довільною ознакою.

**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 ув'язнених. | 8-03-2016 – 25-03-2016 р. |
| 4 | Розробка підсистеми пошуку ув'язнених, формування зведень про склад. | 26-03-2016 – 15-04-2016 р. |
| 5 | Розробка функцій додавання нових, редагування та видалення ув'язнених. | 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 – 20-05-2016 р. |
| 8 | Оформлення пояснювальної записки, додатків, графічного матеріалу | 20-05-2016 – 1-06-2016 р. |
| 9 | Захист | 02-06-2016 р. |

Студент\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Черкас Б. І.

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

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

Реферат

Пояснювальна записка до курсової роботи: 50 с., 21 рис., 1 додаток, 6 джерел.

Метою роботи є розробка програми “ Посібник начальника в'язниці ” 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/cyber2370

В результаті отримана програма під назвою “Посібник начальника в'язниці ”, яка дозволяє знайти ув'язнених, які задовольняють різним критеріям пошуку. Ув'язнених можна шукати за анкетними даними, статтею, датою взяття під варту, місцем в тюремній ієрархії, камерою, відомостями про родичів, особливостями характеру. Є можливість додавати, видаляти та редагувати інформацію про ув'язнених, робити звіт, а також зберігати оновлений перелік ув'язнених після редагування.

ПРОГРАМА, ОБ’ЕКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ, КЛАС, МОВА ПРОГРАМУВАННЯ C#, ПЛАТФОРМА .NET, УВ'ЯЗНЕНІ, 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. 15

2.1 Архітектурний стиль 15

2.2. Об’єктна структура програми 15

2.3 Формат даних 17

3 Інструкція користувача 19

Висновки 26

Перелік посилань 27

Додаток А – Код програми.. 28

Вступ

В'язниця – це таке місце, де більше, ніж де б то не було потрібен порядок, сувора дисципліна та чіткі інструкції. Велика відповідальність за додержання всіх правил та законів лягає на плечі начальника в'язниці. Він повинен чітко знати склад камер, кількість ув'язнених та звільнених. Це дуже непроста робота.

Саме тому посібник ув'язнених є просто незамінним у роботі начальника в'язниці, адже завдяки йому можна легко та швидко знайти інформацію про будь-якого ув'язненого, його характер, склад злочину, статтю. Також при повторному ув'язненні, скориставшись посібником, начальник з легкістю може оновити у пам'яті всі особливості попередніх ув'язнень та підібрати потрібну камеру, враховуючи особливості характеру, статтю та місце в тюремній ієрархії. Крім цього, у посібнику міститься інформація про родину ув'язненого, що особливо стає актуальним при підготовці до виходу на свободу.

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

Електронний посібник ув'язнених вміщає у собі інформацію про ув'язнених, 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 начальника в'язниці».

Посібник має містити у собі детальну інформацію щодо ув'язнених, а саме: анкетні дані ув'язнених, стаття, дата взяття під варту, місце в тюремній ієрархії, камера, відомості про родичів, особливості характеру.

Програма повинна виконувати наступні функції: редагування інформації, зберігання та зчитування її з файлу, пошук ув'язнених за вибраними критеріями, формування статичних зведень про склад, вибір за довільною ознакою.

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 ув'язнених;
2. додавання нового ув'язненого;
3. редагування ув'язненого;
4. видалення ув'язненого;
5. формування статичних зведень про склад;
6. пошук ув'язненого за будь-яким з критеріїв, перелічених нижче:

- анкетні дані;

- стаття;

- камера;

- місце в тюремній ієрархії;

- дата взяття під варту;

- відомості про родичів.

1.3 Інтерфейс користувача

Після запуску програми користувач побачить головну форму, вона ж форма пошуку та управління базою ув'язнених (Рисунок 1.1). В таблиці автоматично з'явиться список всіх ув'язнених, занесених до бази даних.

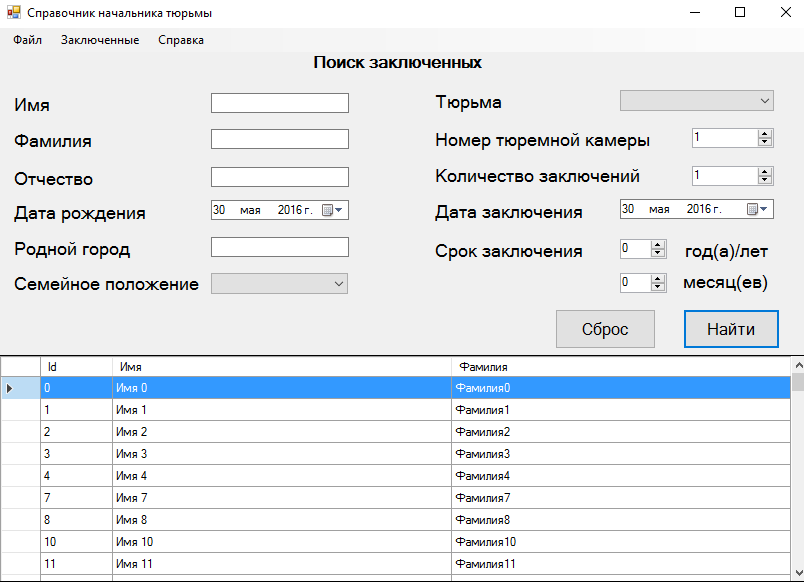


Рисунок 1.1 – Головна форма

Після заповнення полів форми та натискання кнопки «Найти» користувач побачить тих ув'язнених, які задовольняють критеріям пошуку. Вибравши певного ув'язненого, користувач має можливість проглянути детальну інформацію про нього, або відредагувати його, або видалити. Це робиться одиничним кліком правої кнопки миші по обраному ув'язненому або вибором у меню: «Заключенные → Просмотреть\Изменить\Удалить» відповідно. Також наявні функції «Удалить всех\ Обобщенная информация» (Рисунок 1.2, 1.3).

Кнопка «Сброс» очищує всі поля форми, в таблиці автоматично з'явиться список всіх ув'язнених, занесених до бази даних.

Детальну інформацію про ув'язненого можна також отримати, зробивши подвійний клік лівою кнопкою миші по обраному ув'язненому.

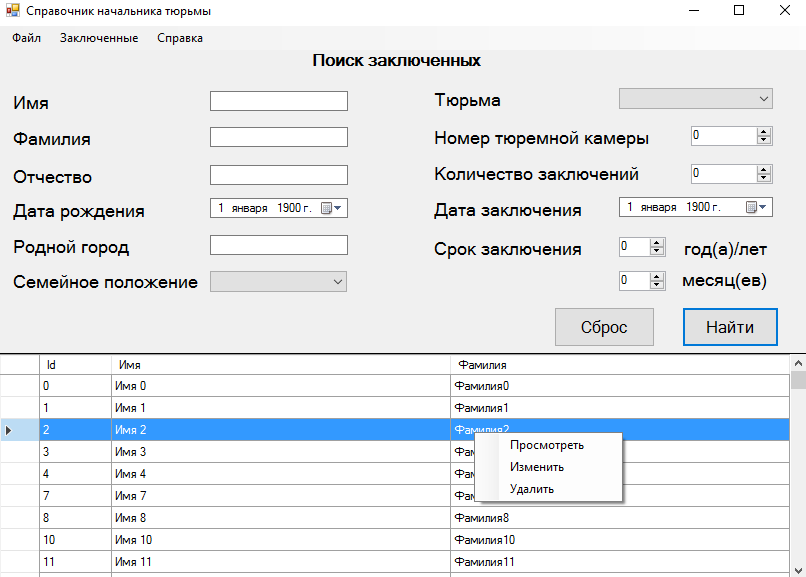


Рисунок 1.2 – Вибір функцій.

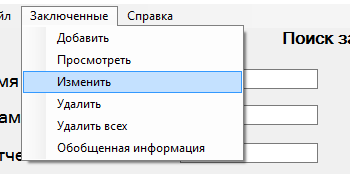


Рисунок 1.3 – Вибір функцій в головному меню.

При виборі «Просмотреть» або «Изменить» з’являється форма перегляду та редагування. На цій формі є дві вкладки: персональна інформація та інформація про ув'язнення (Рисунок 1.4), (Рисунок 1.5). Якщо користувач хоче відредагувати ув'язненого, то треба натиснути кнопку «Изменить», це зробить всі поля форми доступними для запису інформації. Кнопка «ОК» зберігає зміни, внесені користувачем, кнопка «Закрыть» закриває форму без збереження змін.

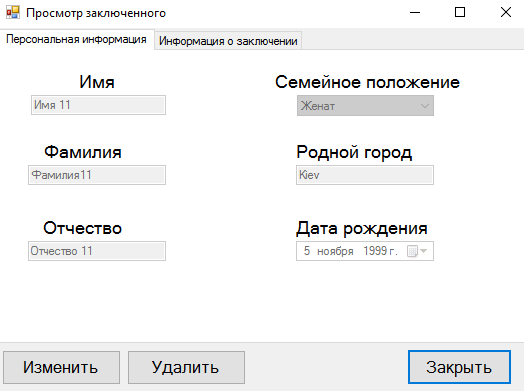


Рисунок 1.4 – Форма просмотру ув'язненого (персональна інформація).

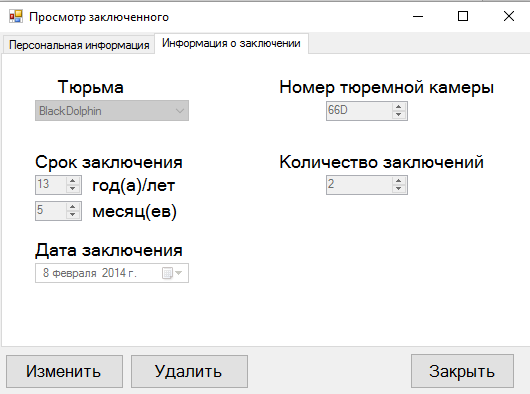


Рисунок 1.5 – Форма просмотру ув'язненого (інформація про ув'язнення).

На головній формі в таблиці користувач побачить зроблені ним зміни у даному ув'язненому.

При виборі функції «Удалить» відкриється діалогове вікно підтвердження видалення ув'язненого з бази.(Рисунок 1.6).

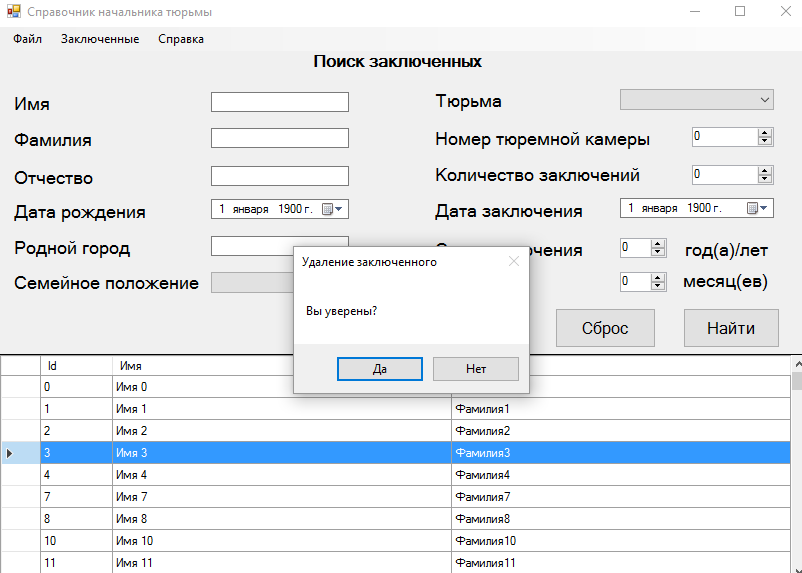


Рисунок 1.6 – Підтвердження видалення ув'язненого.

При виборі «Добавить» з’являється форма додавання (Рисунок 1.8), (Рисунок 1.9). Кнопка «ОК» зберігає нове підприємство, внесене користувачем, кнопка «Закрыть» закриває форму без збереження підприємства.

Також на головній формі є в меню «Справка» є додаткові функції для користувача (Рисунок 1.7).

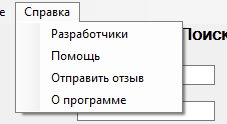


Рисунок 1.7 – Додаткові функції для користувача.

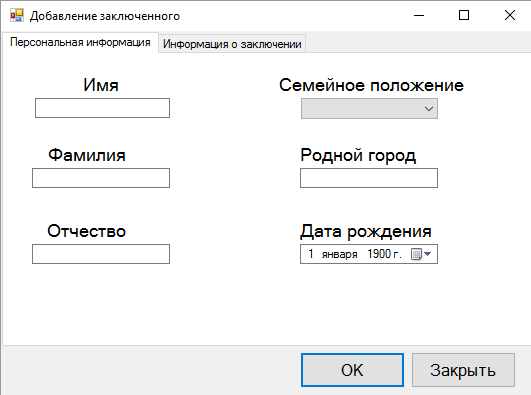


Рисунок 1.8 – Форма додавання ув'язненого (персональна інформація).

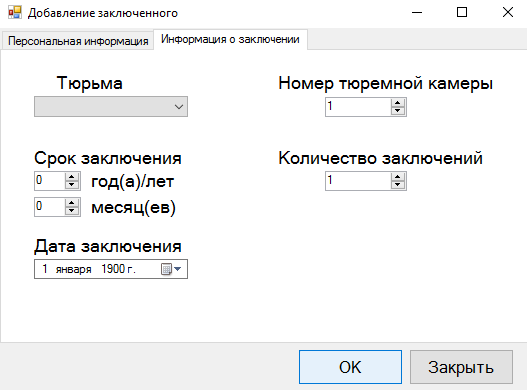


Рисунок 1.9 – Форма додавання ув'язненого (інформація про ув'язненого).

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).

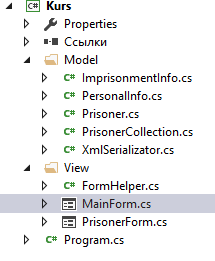


Рисунок 2.1. – Архітектура проекту

У програмі реалізовані такі класи: Prisoner, PersonalInfo, ImprisonmentInfo, PrisonerCollection, XmlSerializator (Рисунок 2.2).

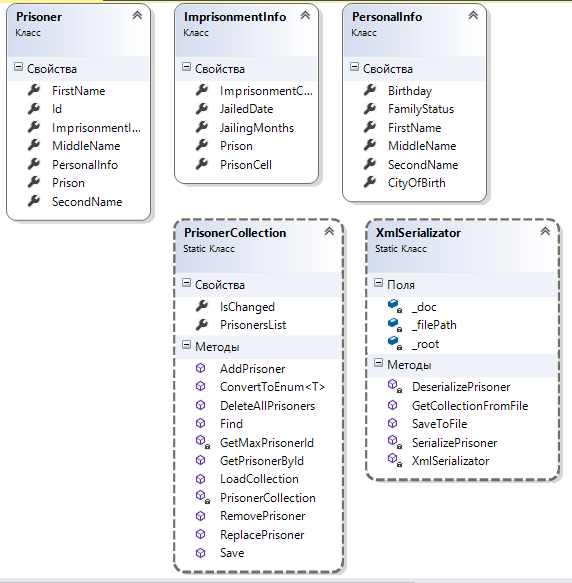


Рисунок 2.2. – Класи

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). Головна форма – це форма пошуку та керування базою ув'язнених.

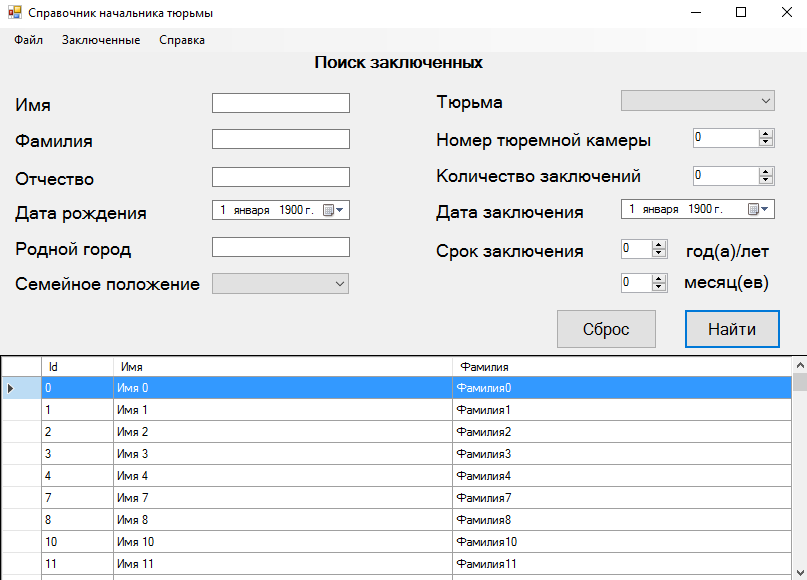


Рисунок 3.1 – Головне вікно

Пошук ув'язненого:

1. заповнити будь-які з критеріїв пошуку, відображених на головному вікні.
2. у разі невірного заповнення полів натиснути «Сброс». Всі поля форми будуть очищені, у таблиці з'явиться список всіх ув'язнених, внесених до бази даних.
3. натиснути «Найти». У вигляді таблиці будуть відображені всі ув'язнені, яку задовольняють критеріям пошуку. У разі відсутності заповнених критеріїв пошуку будуть відображені всі ув'язнені, які збережені в базі даних.

Додавання ув'язненого:

1. натиснути Заключенные → Добавить»;
2. відкриється форма додавання «Добавить» (Рисунок 3.2, 3.3);

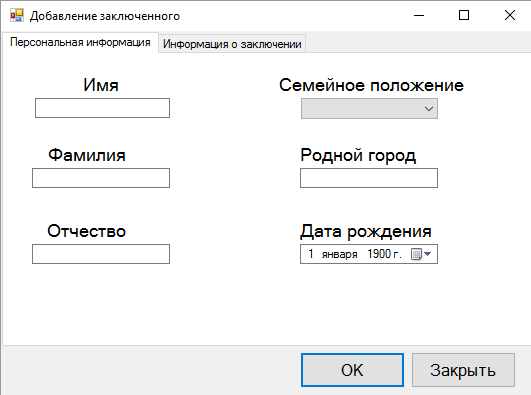


Рисунок 3.2 – Форма додавання ув'язненого (персональна інформація).

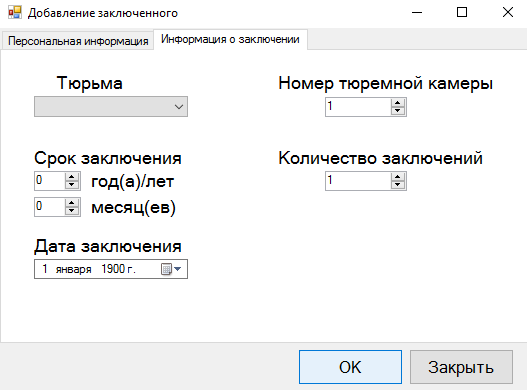


Рисунок 3.3 – Форма додавання ув'язненого (інформація про ув'язненого).

1. потрібно заповнити поля в обох вкладках і натиснути «ОК»;

є) нове ув'язнений відобразиться у базі на головній формі.

Перегляд та редагування ув'язненого:

а) обрати в таблиці бажаного ув'язненого;

б) натиснути «Заключенные → Просмотреть / Изменить» або правою кнопкою миші на обраного ув'язненого та вибрати «Просмотреть/Изменить»;

в) відкриється форма перегляду та редагування «Просмотреть / Изменить» (Рисунок 3.5, 3.6);

г) Для внесення змін натиснути кнопку «Изменить»;

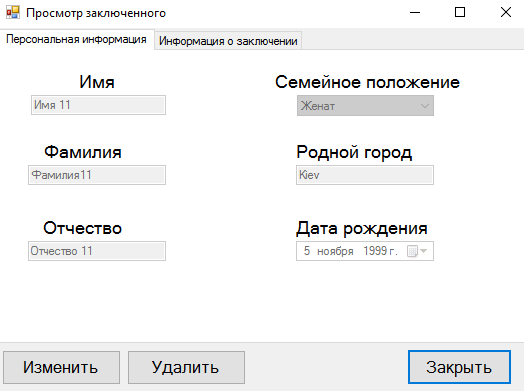


Рисунок 3.5 – Форма просмотру\редагування ув'язненого (персональна інформація).

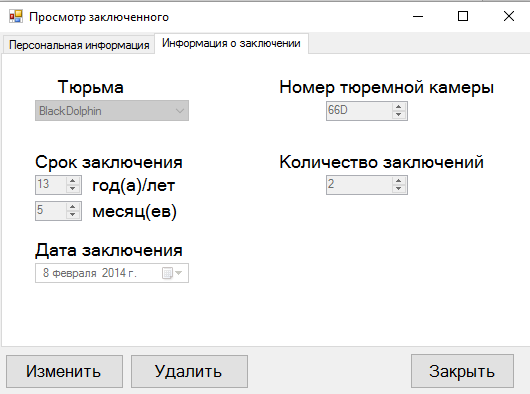


Рисунок 3.6 – Форма просмотру\редагування ув'язненого (інформація про ув'язнення).

д) внести необхідні зміни;

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

є) інформація про ув'язненого буде змінена, результат можна буде побачити у базі на головній формі.

Видалення ув'язненого:

1. вибрати з бази бажаного ув'язненого;
2. натиснути «Заключенные → Удалить» або правою кнопкою миші на обраного ув'язне6ного та обрати «Удалить»;
3. з’явиться вікно для підтвердження видалення (Рисунок 3.7).

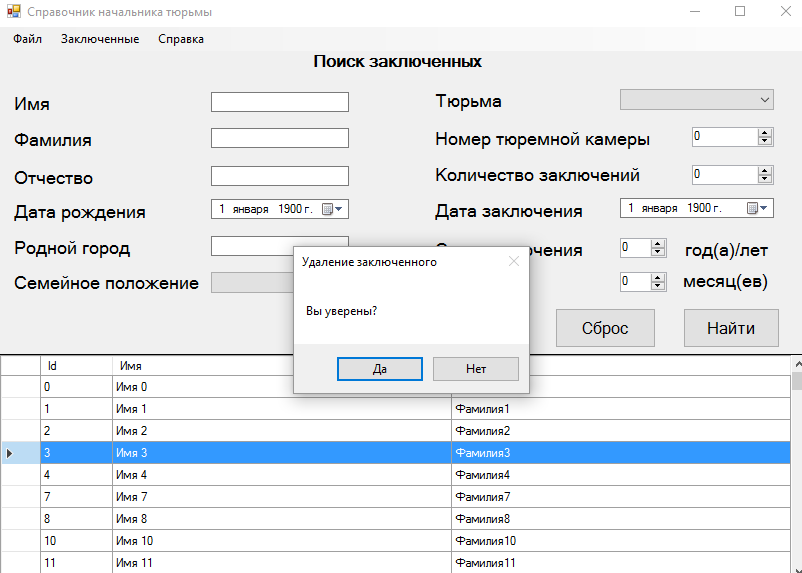


Рисунок 3.7 – Вікно підтвердження вибору

1. підтвердити свій вибір.
2. якщо потрібно видалити всіх ув'язнених вибрати «Заключенные → Удалить всех». Для видалення всіх ув'язнених потрібно ввести пароль адміністратора “GlobalLogicPassword” (без лапок) та натиснути “ОК”.

Для отримання узагальненої інформації про ув'язнених обрати в головному меню «Заключенные → Обобщенная информация».(Рисунок 3.8).

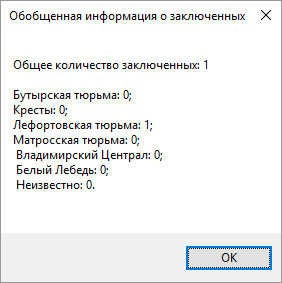


Рисунок 3.8 – Узагальнена інформація про ув'язнених

Для збереження внесених змін потрібно обрати в головному меню «Файл →Сохранить». (Рисунок 3.9). Якщо користувач вирішив завершити програму без збереження змін, тоді перед завершенням потрібно буде обрати зберігати дані чи ні.(Рисунок 3.10)

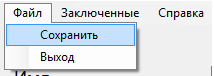


Рисунок 3.9 – Кнопка збереження змін

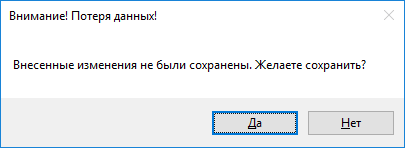


Рисунок 3.10 – Вікно підтвердження збереження змін

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 начальника в'язниці» на платформі .Net з використанням мови C#.

Посібник містить у собі детальну інформацію щодо ув'язнених, а саме анкетні дані, статтю, дату взяття під варту, місце в тюремній ієрархії, камеру, відомості про родичів, особливості характеру.

Програма виконує наступні функції: редагування інформації, зберігання та зчитування її з файлу, пошук ув'язнених 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. Ульман, Дж. Д. Основи систем баз даних [Текст] / Дж. Д. Ульман - М. : Фінанси і статистика, 1983. – 334 с.
6. http://metanit.com/sharp/tutorial/16.2.php

Додаток А – Код програми

using System;

/// <summary>

/// Действующие тюрьмы.

/// </summary>

public enum Prisons

{

Unknown = 0,

Butyrskaya = 1,

Crosses = 2,

Lefortovo = 3,

Sailor = 4,

VladimirskyCentral = 5,

WhiteSwan = 6

}

public class ImprisonmentInfo

{

public Prisons Prison { get; set; }

public int PrisonCell { get; set; }

public DateTime JailedDate { get; set; }

public int JailingMonths { get; set; }

public int ImprisonmentCount { get; set; }

}

public enum FamilyStatus

{

Unknown = 0,

Married = 1,

NotMarried = 2,

Divorced = 3

}

public class PersonalInfo

{

public string FirstName { get; set; }

public string SecondName { get; set; }

public string MiddleName { get; set; }

public DateTime Birthday { get; set; }

public string СityOfBirth { get; set; }

public FamilyStatus FamilyStatus { get; set; }

}

/// <summary>

/// Модель заключенного.

/// </summary>

public class Prisoner

{

public int Id { get; set; }

public PersonalInfo PersonalInfo { get; set; }

public ImprisonmentInfo ImprisonmentInfo { get; set; }

public string FirstName => PersonalInfo.FirstName;

public string SecondName => PersonalInfo.SecondName;

public string MiddleName => PersonalInfo.MiddleName;

public string Prison => ImprisonmentInfo.Prison.ToString();

}

/// <summary>

/// Класс для работы со списком заключенными.

/// </summary>

public static class PrisonerCollection

{

public static bool IsChanged { get; private set; }

public static BindingList<Prisoner> PrisonersList

{ get; private set; }

/// <summary>

/// Конструктор статического класса PrisonerCollection.

/// </summary>

static PrisonerCollection()

{

PrisonersList = new BindingList<Prisoner>();

IsChanged = false;

LoadCollection();

}

/// <summary>

/// Метод поиска объекта Prisoner по полю Id.

/// </summary>

/// <param name="id">Идентификатор заключенного.

/// Должен существовать заключенный с таким ID.</param>

/// <returns></returns>

public static Prisoner GetPrisonerById(int id)

{

var pris = PrisonersList.SingleOrDefault

(x => x.Id == id);

if(pris == null)

throw new NullReferenceException

("Неверный идентификатор {id}");

return pris;

}

/// <summary>

/// Метод удаления заключенного по полю id.

/// </summary>

/// <param name="id">Идентификатор заключенного</param>

public static void RemovePrisoner(int id)

{

PrisonersList.Remove(GetPrisonerById(id));

IsChanged = true;

}

/// <summary>

/// Метод добавления нового заключенного в коллекцию.

/// </summary>

/// <param name="prisoner"></param>

public static void AddPrisoner(Prisoner prisoner)

{

var newId = GetMaxPrisonerId();

prisoner.Id = (newId != -1) ? (newId + 1) : 1;

PrisonersList.Add(prisoner);

IsChanged = true;

}

/// <summary>

/// Метод замены заключенного.

/// </summary>

/// <param name="id">ID заменяемого заключенного.</param>

/// <param name="prisoner">Заключенный для замены.</param>

public static void ReplacePrisoner(Prisoner prisoner)

{

var replaceThis = GetPrisonerById(prisoner.Id);

PrisonersList[PrisonersList.IndexOf(replaceThis)]

= prisoner;

IsChanged = true;

}

/// <summary>

/// Удаляет всех заключенных.

/// </summary>

public static void DeleteAllPrisoners()

{

PrisonersList = new BindingList<Prisoner>();

IsChanged = true;

}

/// <summary>

/// Метод получения наибольшего идентификатора заключенного.

/// </summary>

/// <returns>Возвращает идентификатор

/// последнего заключенного

/// или -1 при отсутствии заключенных.</returns>

private static int GetMaxPrisonerId()

{

return (PrisonersList.Count > 0)

? PrisonersList.Max(x => x.Id) : 0;

}

/// <summary>

/// Метод поиска по разным критериям.

/// Фильтрует основную коллекцию по заполненным критериям в модели.

///

/// </summary>

/// <param name="prisoner"></param>

/// <returns></returns>

public static BindingList<Prisoner> Find(Prisoner prisoner)

{

var prisPersInfo = prisoner.PersonalInfo;

var prisImprInfo = prisoner.ImprisonmentInfo;

var collection = PrisonersList.Where

(x => x.PersonalInfo.FirstName.Contains(prisPersInfo.FirstName)

&& x.PersonalInfo.SecondName.Contains(prisPersInfo.SecondName)

&& x.PersonalInfo.MiddleName.Contains(prisPersInfo.MiddleName)

&& x.PersonalInfo.СityOfBirth.Contains(prisPersInfo.СityOfBirth)

&& (prisPersInfo.Birthday == new DateTime(1800, 1, 1)

|| x.PersonalInfo.Birthday.Equals(prisPersInfo.Birthday))

&& (prisPersInfo.FamilyStatus == FamilyStatus.Unknown

|| x.PersonalInfo.FamilyStatus == prisPersInfo.FamilyStatus)

&& (prisImprInfo.JailedDate == new DateTime(1800, 1, 1)

|| x.ImprisonmentInfo.JailedDate

.Equals(prisImprInfo.JailedDate))

&& (prisImprInfo.JailingMonths == 0

|| x.ImprisonmentInfo.JailingMonths ==

prisImprInfo.JailingMonths)

&& (prisImprInfo.ImprisonmentCount == 0

|| x.ImprisonmentInfo.ImprisonmentCount ==

prisImprInfo.ImprisonmentCount)

&& (prisImprInfo.Prison == Prisons.Unknown

|| x.ImprisonmentInfo.Prison == prisImprInfo.Prison)

&& (prisImprInfo.PrisonCell == 0

|| x.ImprisonmentInfo.PrisonCell == prisImprInfo

.PrisonCell));

var list = new BindingList<Prisoner>();

foreach (var z in collection)

list.Add(z);

return list;

}

/// <summary>

/// Записывает изменения в файл.

/// </summary>

public static void Save()

{

XmlSerializator.SaveToFile(PrisonersList);

IsChanged = false;

}

/// <summary>

/// Заполняет коллекцию из файла.

/// </summary>

public static void LoadCollection()

{

var col = XmlSerializator.GetCollectionFromFile();

PrisonersList = col ?? new BindingList<Prisoner>();

IsChanged = false;

}

/// <summary>

/// Метод приводит строку к типу перечисления.

/// </summary>

/// <typeparam name="T"></typeparam>

/// <param name="value"></param>

/// <returns></returns>

public static T ConvertToEnum<T>(string value)

{

return (T)Enum.Parse(typeof(T), value, true);

}

}

using System;

using System.ComponentModel;

using System.IO;

using System.Linq;

using System.Xml.Linq;

/// <summary>

/// Класс для работы с файлом XML.

/// </summary>

public static class XmlSerializator

{

//Путь к Xml-документу.

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

//Объявление Xml-документа.

private static XDocument \_doc;

//Корневой элемент Xml-документа.

private static XElement \_root;

/// <summary>

/// Статический конструктор без параметров.

/// Создание или подключение документа.

/// </summary>

static XmlSerializator()

{

if (File.Exists(\_filePath))

{

\_doc = XDocument.Load(\_filePath);

if (!\_doc.Elements().Any())

{

\_doc = new XDocument(new XElement("Prisoners"));

\_doc.Save(\_filePath);

}

}

else

{

\_doc = new XDocument(new XElement("Prisoners"));

}

\_root = \_doc.Element("Prisoners");

}

/// <summary>

/// Метод сохранения коллекции заключенных в файл.

/// </summary>

///

public static void SaveToFile(BindingList<Prisoner> list)

{

\_root.RemoveAll();

foreach (var elem in list)

\_root.Add(SerializePrisoner(elem));

\_doc.Save(\_filePath);

}

/// <summary>

/// Метод создания коллекции заключенных

/// с помощью чтения из файла.

/// </summary>

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

public static BindingList<Prisoner> GetCollectionFromFile()

{

var list = new BindingList<Prisoner>();

foreach (var xElem in \_root.Elements())

{

var elem = DeserializePrisoner(xElem);

if (elem != null)

list.Add(elem);

}

return list;

}

/// <summary>

/// Метод создает объект заключенного (Prisoner)

/// из представления XML-объекта.

/// </summary>

/// <param name="pris"></param>

/// <returns></returns>

private static XElement SerializePrisoner(Prisoner pris)

{

var persInfo = new XElement("PersonalInfo",

new XElement("FirstName",

pris.PersonalInfo.FirstName),

new XElement("SecondName",

pris.PersonalInfo.SecondName),

new XElement("MiddleName",

pris.PersonalInfo.MiddleName),

new XElement("Birthday",

pris.PersonalInfo.Birthday),

new XElement("СityOfBirth”,

pris.PersonalInfo.СityOfBirth),

new XElement("FamilyStatus",

pris.PersonalInfo.FamilyStatus));

var imprInfo = new XElement("ImprisonmentInfo",

new XElement("JailedDate",

pris.ImprisonmentInfo.JailedDate),

new XElement("ImprisonmentCount",

pris.ImprisonmentInfo.ImprisonmentCount),

new XElement("JailingMonths",

pris.ImprisonmentInfo.JailingMonths),

new XElement("Prison",

pris.ImprisonmentInfo.Prison),

new XElement("PrisonCell",

pris.ImprisonmentInfo.PrisonCell));

return new XElement("Prisoner",

new XAttribute("Id", pris.Id),

persInfo,

imprInfo);

}

/// <summary>

/// Метод создает объект XML заключенного (XML) из Prisoner.

/// </summary>

/// <param name="pris"></param>

/// <returns></returns>

private static Prisoner DeserializePrisoner(XElement xPris)

{

var xPersInfo = xPris.Element("PersonalInfo");

var persInfo = new PersonalInfo()

{

FirstName = xPersInfo.Element("FirstName").Value,

SecondName = xPersInfo.Element("SecondName").Value,

MiddleName = xPersInfo.Element("MiddleName").Value,

FamilyStatus =

PrisonerCollection.ConvertToEnum<FamilyStatus>

(xPersInfo.Element("FamilyStatus").Value),

Birthday = Convert.ToDateTime

(xPersInfo.Element("Birthday").Value),

СityOfBirth = xPersInfo.Element("СityOfBirth").Value

};

var xImprInfo = xPris.Element("ImprisonmentInfo");

var imprInfo = new ImprisonmentInfo()

{

JailedDate = Convert.ToDateTime

(xImprInfo.Element("JailedDate").Value),

JailingMonths = Convert.ToInt32

(xImprInfo.Element("JailingMonths").Value),

PrisonCell = Convert.ToInt32

(xImprInfo.Element("PrisonCell").Value),

ImprisonmentCount = Convert.ToInt32

(xImprInfo.Element("ImprisonmentCount").Value),

Prison = PrisonerCollection.ConvertToEnum<Prisons>

(xImprInfo.Element("Prison").Value)

};

return new Prisoner()

{

Id = Convert.ToInt32(xPris.FirstAttribute.Value),

ImprisonmentInfo = imprInfo,

PersonalInfo = persInfo

};

}

}

static class FormHelper

{

public static FamilyStatus GetFamilyValueByIndex(int index)

{

string str;

switch (index)

{

case 1:

str = "Married";

break;

case 2:

str = "NotMarried";

break;

case 3:

str = "Divorced";

break;

default:

str = "Unknown";

break;

}

return PrisonerCollection

.ConvertToEnum<FamilyStatus>(str);

}

public static int GetFamilyIndexByValue(string value)

{

switch (value)

{

case "Married":

return 1;

case "NotMarried":

return 2;

case "Divorced":

return 3;

default:

return 0;

}

}

public static Prisons GetPrisonValueByIndex(int index)

{

string str;

switch (index)

{

case 1:

str = "Butyrskaya";

break;

case 2:

str = "Crosses";

break;

case 3:

str = "Lefortovo";

break;

case 4:

str = "Sailor";

break;

case 5:

str = "VladimirskyCentral";

break;

case 6:

str = "WhiteSwan";

break;

default:

str = "Unknown";

break;

}

return PrisonerCollection.ConvertToEnum<Prisons>(str);

}

public static int GetPrisonIndexByValue(string value)

{

switch (value)

{

case "Butyrskaya":

return 1;

case "Crosses":

return 2;

case "Lefortovo":

return 3;

case "Sailor":

return 4;

case "VladimirskyCentral":

return 5;

case "WhiteSwan":

return 6;

default:

return 0;

}

}

}

public partial class MainForm : Form

{

/// <summary>

/// Стандартный конструктор.

/// </summary>

public MainForm()

{

InitializeComponent();

}

/// <summary>

/// Событие FormLoad главной формы.

/// Привязывает главную коллекцию заключенных к таблице.

/// </summary>

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

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

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

{

BindCollectionToDataGridView

(PrisonerCollection.PrisonersList);

PrisonCB.SelectedIndex = FamilyCB.SelectedIndex = 0;

}

/// <summary>

/// Событие Click кнопки Найти. Собирает параметры поиска с

/// формы,

/// перепривязывает форму к коллекции заключенных.

/// </summary>

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

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

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

{

var col = risonerCollection

.Find(CreatePrisonerFromForm());

MessageBox.Show(!col.Any()

? "Не найдено заключенных по вашему запросу"

: "По вашему запросу найдено "

+ $"{col.Count()} заключенных.",

"Результаты поиска");

BindCollectionToDataGridView(col);

}

/// <summary>

/// Событие таблицы CellDoubleClick.

/// Открывает форму для отображения/изменения/удаления

/// заключенного.

/// </summary>

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

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

private void DGV\_CellDoubleClick(object sender,

DataGridViewCellEventArgs e)

{

ShowPrisoner();

}

/// <summary>

/// Привязывает коллекцию к таблице DataGridView.

/// </summary>

/// <param name="list"></param>

private void BindCollectionToDataGridView

(BindingList<Prisoner> list)

{

var bindingList = list;

var source = new BindingSource(bindingList, null);

DGV.DataSource = source;

}

/// <summary>

/// Событие FormClosing формы MainForm.

/// Сохраняет коллекцию заключенных в файл.

/// </summary>

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

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

private void MainForm\_FormClosing

(object sender, FormClosingEventArgs e)

{

if (!PrisonerCollection.IsChanged) return;

DialogResult dialogResult = MessageBox.Show

(@"Внесенные изменения не были сохранены. «

+ @”Желаете сохранить?",

@"Внимание! Потеря данных!", MessageBoxButtons.YesNo);

if (dialogResult != DialogResult.Yes) return;

PrisonerCollection.Save();

}

/// <summary>

/// Событие вызова ContextMenu таблицы DGV (DataGridView).

/// </summary>

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

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

private void DGV\_CellContextMenuStripNeeded

(object sender,

DataGridViewCellContextMenuStripNeededEventArgs e)

{

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

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

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

}

/// <summary>

/// Изменить заключенного (контекстное меню таблицы).

/// </summary>

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

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

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

{

ChangePrisoner();

}

/// <summary>

/// Удалить заключенного заключенного (контекстное меню таблицы).

/// </summary>

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

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

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

{

DeletePrisoner();

}

/// <summary>

/// Открывает форму для добавления нового заключенного.

/// </summary>

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

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

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

{

AddPrisoner();

}

/// <summary>

/// Открывает форму для изменения заключенного,

/// который выделен в таблице.

/// </summary>

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

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

private void изменитьToolStripMenuItem\_Click

(object sender, EventArgs e)

{

ChangePrisoner();

}

/// <summary>

/// Кнопка из меню "Разработчики".

/// Выводит информацию о разработчиках программы.

/// </summary>

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

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

private void разработчикиToolStripMenuItem\_Click

(object sender, EventArgs e)

{

MessageBox.Show(@"Программа разработана студентом “

+ “группы ПИ-15-1"

+ "Черкасом Борисом.",

"Справочник начальника тюрьмы.");

}

/// <summary>

/// Кнопка из меню "О программе...".

/// Отображает основную информацию о программе.

/// </summary>

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

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

private void оПрограммеToolStripMenuItem\_Click

(object sender, EventArgs e)

{

MessageBox.Show

(@"Програма дозволяє знайти ув'язнених, які"

+ "задовольняють різним критеріям пошуку. ”

+ “Ув'язнених можна шукати"

+ "за анкетними даними, статтею, "

+ "датою взяття під варту, місцем в тюремній ієрархії, камерою,"

+ "відомостями про родичів, особливостями характеру. “

+ “Є можливість "

+ "додавати, видаляти та редагувати інформацію про ув'язнених, "

+ "робити звіт, а також зберігати оновлений перелік ув'язнених"

+ "після редагування.", "Справочник начальника тюрьмы.");

}

/// <summary>

/// Сохраняет текущее состояние основной коллекции

/// заключенных в файл.

/// </summary>

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

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

private void сохранитьToolStripMenuItem\_Click

(object sender, EventArgs e)

{

PrisonerCollection.Save();

}

/// <summary>

/// Собирает данные поиска с формы в один объект.

/// </summary>

/// <returns></returns>

private Prisoner CreatePrisonerFromForm()

{

var persInfo = new PersonalInfo()

{

FirstName = FirstNameTB.Text,

SecondName = SecondNameTB.Text,

MiddleName = MiddleNameTB.Text,

Birthday = BirthdayDTP.Value,

СityOfBirth = CityBornTB.Text,

FamilyStatus = FormHelper

.GetFamilyValueByIndex(FamilyCB.SelectedIndex)

};

var imprInfo = new ImprisonmentInfo()

{

JailedDate = JailedDTP.Value,

JailingMonths = Convert.ToInt32

(JailingYearsNUD.Value) \* 12

+ Convert.ToInt32(JailingMonthsNUD.Value),

PrisonCell = Convert.ToInt32(PrisCellNUD.Value),

ImprisonmentCount = Convert.ToInt32

(ImprCountNUD.Value),

Prison = FormHelper.GetPrisonValueByIndex

(PrisonCB.SelectedIndex)

};

return new Prisoner()

{

ImprisonmentInfo = imprInfo,

PersonalInfo = persInfo

};

}

/// <summary>

/// Кнопка Сброс. Очищает поля поиска и отображает

/// полную коллекцию заключенных.

/// </summary>

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

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

private void button1\_Click(object sender, EventArgs e)

{

FirstNameTB.Text = SecondNameTB.Text

= MiddleNameTB.Text = CityBornTB.Text = "";

BirthdayDTP.Value = JailedDTP.Value

= new DateTime(1800, 1, 1);

JailingYearsNUD.Value = ImprCountNUD.Value

= JailingMonthsNUD.Value = PrisCellNUD.Value = 0;

FamilyCB.SelectedIndex = PrisonCB.SelectedIndex = 0; BindCollectionToDataGridView

(PrisonerCollection.PrisonersList);

}

/// <summary>

/// Просмотреть (контекстное меню таблицы).

/// </summary>

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

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

private void просмотретьToolStripMenuItem\_Click\_1

(object sender, EventArgs e)

{

ShowPrisoner();

}

/// <summary>

/// Кнопка из меню программы "Удалить заключенного".

/// Удаляет выбранного в таблице заключенного.

/// </summary>

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

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

private void удалитьToolStripMenuItem\_Click

(object sender, EventArgs e)

{

DeletePrisoner();

}

/// <summary>

/// Кнопка Удалить всех заключенных из меню программы.

/// Полностью очищает коллекцию от заключенных.

/// </summary>

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

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

private void удалитьВсехЗаключенныхToolStripMenuItem\_Click

(object sender, EventArgs e)

{

DialogResult dialogResult =

MessageBox.Show

(@"Вы точно уверены, что хотите “

+ “удалить всех заключенных?",

@"Удаление ВСЕХ заключенных!", MessageBoxButtons.YesNo);

if (dialogResult != DialogResult.Yes) return;

PrisonerCollection.DeleteAllPrisoners();

BindCollectionToDataGridView

(PrisonerCollection.PrisonersList);

}

/// <summary>

/// Открывает форму отображения детальной

/// информации по заключенному.

/// </summary>

private void ShowPrisoner()

{

if (!CheckTable()) return;

var form = new PrisonerForm

(Convert.ToInt32(DGV[0,GV.SelectedRows[0].Index].Value), false);

form.ShowDialog();

}

/// <summary>

/// Открывает форму добавления заключенного.

/// </summary>

private void AddPrisoner()

{

var form = new PrisonerForm();

form.ShowDialog();

}

/// <summary>

/// Открывает форму изменения заключенного.

/// </summary>

private void ChangePrisoner()

{

if (!CheckTable()) return;

var form = new PrisonerForm

(Convert.ToInt32(DGV[0,DGV.SelectedRows[0]

.Index].Value), true);

form.ShowDialog();

}

/// <summary>

/// Удаляет заключенного из коллекции.

/// </summary>

private void DeletePrisoner()

{

if (!CheckTable()) return;

DialogResult dialogResult

= MessageBox.Show(@"Вы уверены?",

@"Удаление заключенного",

MessageBoxButtons.YesNo);

if (dialogResult != DialogResult.Yes) return;

PrisonerCollection.RemovePrisoner

(Convert.ToInt32(DGV[0, DGV.SelectedRows[0]

.Index].Value));

}

/// <summary>

/// Проверяет, выбран ли заключенный в таблице.

/// </summary>

/// <returns></returns>

private bool CheckTable()

{

if (DGV.SelectedRows.Count == 0)

{

MessageBox.Show("Сначала выберите заключенного!",

"Ошибка");

return false;

}

return true;

}

/// <summary>

/// Показывает MessageBox с обобщенной информацией о заключенных.

/// </summary>

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

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

private void обобщенныеДанныеToolStripMenuItem\_Click

(object sender, EventArgs e)

{

var prisLst = PrisonerCollection.PrisonersList;

var buturka = prisLst.Count

(x => x.Prison.ToString().Equals("Butyrskaya"));

var krestu = prisLst.Count

(x => x.Prison.ToString().Equals("Crosses"));

var lefort = prisLst.Count

(x => x.Prison.ToString().Equals("Lefortovo"));

var matros = prisLst.Count

(x => x.Prison.ToString().Equals("Sailor"));

var vladimir = prisLst.Count

(x => x.Prison.ToString()

.Equals("VladimirskyCentral"));

var lebed = prisLst.Count

(x => x.Prison.ToString().Equals("WhiteSwan"));

var unknown = prisLst.Count

(x => x.Prison.ToString().Equals("Unknown"));

MessageBox.Show($"Общее количество заключенных: "

+$"{prisLst.Count}

{Environment.NewLine}

{Environment.NewLine}"

+ $"Бутырская тюрьма: {buturka}; {Environment.NewLine}"

+ $"Кресты: {krestu}; {Environment.NewLine}"

+ $"Лефортовская тюрьма: {lefort};

{Environment.NewLine}"

+ $"Матросская тюрьма: {matros}; {Environment.NewLine} "

+ $"Владимирский Централ: {vladimir};

{Environment.NewLine} "

+ $"Белый Лебедь: {lebed}; {Environment.NewLine} "

+ $"Неизвестно: {unknown}. {Environment.NewLine} ",

"Обобщенная информация о заключенных");

}

}

/// <summary>

/// Класс формы заключенного. Просмотр, добавление, изменение.

/// </summary>

public partial class PrisonerForm : Form

{

private readonly Prisoner prisoner = null;

/// <summary>

/// Конструктор (добавление заключенного).

/// </summary>

public PrisonerForm()

{

InitializeComponent();

Text = @"Добавление заключенного";

deleteBtn.Visible = writableBtn.Visible = false;

okBtn.Visible = true;

FamilyCB.SelectedIndex = prisonCB.SelectedIndex = 0;

}

/// <summary>

/// Конструктор просмотр или изменения данных заключенного.

/// </summary>

/// <param name="prisoner"></param>

/// <param name="writable"></param>

public PrisonerForm(int prisonerId, bool writable)

{

InitializeComponent();

if (prisonerId <= 0)

{

MessageBox.Show(@"Sorry, but ID is wrong!");

Close();

}

var pris = PrisonerCollection

.GetPrisonerById(prisonerId);

if (pris == null)

{

MessageBox.Show

(@"Ошибка! Такого заключенного не существует!");

Close();

}

Text = writable

? @"Изменение заключенного"

: "Просмотр заключенного";

prisoner = pris;

FillFields();

deleteBtn.Visible = true;

okBtn.Visible = writable;

writableBtn.Visible = !writable;

WritableFields(writable);

}

private void FillFields()

{

var persControls = Controls["tabControl1"]

.Controls[0].Controls;

var imprisControls = Controls["tabControl1"]

.Controls[1].Controls;

persControls["FirstNameTB"].Text =

prisoner.PersonalInfo.FirstName;

persControls["SecondNameTB"].Text = prisoner.PersonalInfo.SecondName;

persControls["MiddleNameTB"].Text =

prisoner.PersonalInfo.MiddleName;

persControls["CityBornTB"].Text = prisoner.PersonalInfo.СityOfBirth;

((DateTimePicker)persControls["birthdayDTP"]).Value =

prisoner.PersonalInfo.Birthday;

((ComboBox) persControls["FamilyCB"]).SelectedIndex =

FormHelper.GetFamilyIndexByValue

(prisoner.PersonalInfo

.FamilyStatus.ToString());

((ComboBox)imprisControls["prisonCB"]).SelectedIndex =

FormHelper.GetPrisonIndexByValue

(prisoner.ImprisonmentInfo

.Prison.ToString());

((NumericUpDown) imprisControls["PrisCellNUD"]).Value =

prisoner.ImprisonmentInfo.PrisonCell;

((NumericUpDown) imprisControls["ImprCountNUD"]).Value =

prisoner.ImprisonmentInfo.ImprisonmentCount;

((NumericUpDown) imprisControls["JailingYearsNUD"])

.Value = Convert.ToInt32(prisoner.ImprisonmentInfo

.JailingMonths/12);

((NumericUpDown) imprisControls["JailingMonthsNUD"]).Value =

prisoner.ImprisonmentInfo.JailingMonths%12;

((DateTimePicker)imprisControls["JailedDTP"]).Value =

prisoner.ImprisonmentInfo.JailedDate;

}

/// <summary>

/// Событие Click кнопки OK. Добавляет/заменяет заключенного в коллекцию.

/// </summary>

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

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

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

{

if (!CheckFields()) return;

var personalInf = new PersonalInfo()

{

FamilyStatus = FormHelper.GetFamilyValueByIndex

(FamilyCB.SelectedIndex),

FirstName = FirstNameTB.Text,

SecondName = SecondNameTB.Text,

MiddleName = MiddleNameTB.Text,

Birthday = birthdayDTP.Value,

СityOfBirth = CityBornTB.Text

};

var imprisInfo = new ImprisonmentInfo()

{

Prison = FormHelper.GetPrisonValueByIndex

(prisonCB.SelectedIndex),

ImprisonmentCount =

Convert.ToInt32(ImprCountNUD.Value),

PrisonCell = Convert.ToInt32(PrisCellNUD.Value),

JailedDate = JailedDTP.Value,

JailingMonths = Convert.ToInt32

(JailingYearsNUD.Value\*12

+ JailingMonthsNUD.Value)

};

var prisoner = new Prisoner()

{

Id = this.prisoner != null ? this.prisoner.Id : 0,

PersonalInfo = personalInf,

ImprisonmentInfo = imprisInfo

};

if(this.prisoner == null)

PrisonerCollection.AddPrisoner(prisoner);

else

PrisonerCollection.ReplacePrisoner(prisoner);

Close();

}

/// <summary>

/// Событие Click кнопки Cancel. Закрывает текущую форму.

/// </summary>

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

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

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

{

Close();

}

/// <summary>

/// Делает поля формы (не)доступными для записи.

/// </summary>

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

private void WritableFields(bool verif)

{

WritableFieldsHelper(Controls["tabControl1"]

.Controls[0].Controls, verif);

WritableFieldsHelper(Controls["tabControl1"]

.Controls[1].Controls, verif);

}

/// <summary>

/// Вспомогательный метод для метода writableFields.

/// Изменяет свойство Enabled у полей на форме.

/// </summary>

/// <param name="control"></param>

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

private void WritableFieldsHelper

(Control.ControlCollection control,

bool verif)

{

foreach (var c1 in control)

if (c1 is TextBox)

((TextBox)c1).Enabled = verif;

else if (c1 is NumericUpDown)

((NumericUpDown)c1).Enabled = verif;

else if (c1 is ComboBox)

((ComboBox)c1).Enabled = verif;

else if (c1 is DateTimePicker)

((DateTimePicker)c1).Enabled = verif;

}

/// <summary>

/// Событие Click кнопки Удалить.

/// </summary>

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

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

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

{

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

@"Удаление заключенного", MessageBoxButtons.YesNo);

if (dialogResult != DialogResult.Yes) return;

PrisonerCollection.RemovePrisoner(prisoner.Id);

MessageBox.Show(@"Success");

Close();

}

/// <summary>

/// Событие Click кнопки Изменить. Делает поля Enabled.

/// </summary>

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

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

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

{

writableBtn.Visible = false;

okBtn.Visible = true;

WritableFields(true);

}

private bool CheckFields()

{

if (FirstNameTB.Text == "" || SecondNameTB.Text == ""

|| MiddleNameTB.Text == "" || CityBornTB.Text == ""

|| birthdayDTP.Value == new DateTime(1800, 1, 1)

|| (JailingYearsNUD.Value == 0

&& JailingMonthsNUD.Value == 0)

|| PrisCellNUD.Value == 0)

{

MessageBox.Show("Заполните все поля!",

"Ошибка добавления заключенного.");

return false;

}

if(birthdayDTP.Value > JailedDTP.Value)

{

MessageBox.Show("Дата рождения не может быть "

+ "раньше даты заключения!",

"Ошибка добавления заключенного.");

return false;

}

return true;

}

}