МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ, МОЛОДЕЖИ И СПОРТА УКРАИНЫ

ДОНБАССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

КАФЕДРА СПЕЦИАЛИЗИРОВАННЫХ КОМПЬЮТЕРНЫХ СИСТЕМ

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе по дисциплине

«Программирование»

Выполнил: студент группы СКС-10з

Тронов М. А.

Проверил: Мочалин А. Е.

Алчевск, 2012

РЕФЕРАТ

Пояснительная записка - 133 с., количество источников - 2, количество рисунков - 10.

Курсовая работа иллюстрирует процесс разработки программы "АРМ табельщика", используемой для учета выполняемых работ, отработанного времени и расчета заработной платы сотрудников ремонтных организаций.

В процессе работы с программой, пользователь вводит данные, которые в последствии доступны для просмотра, изменения или удаления.

Для хранения данных в программе используются файлы баз данных типа .mdb. Предусмотрена реализация механизма защиты целостности данных.

СОДЕРЖАНИЕ

1. СИСТЕМНЫЙ АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

Необходимо разработать приложение для учета выполняемых работ, затрат рабочего времени и расчета заработной платы работников ремонтных организаций. Учет ведется на основе составления электронных версий нарядов на выполнение работ. Каждый наряд содержит информацию о заказчике, номер заказа, дату составления или последней редакции наряда, перечень исполнителей работ, затраченное время и перечень выполненых работ. Образец наряда представлен на рисунках 1 и 2.

Сдельный наряд

за *Апрель 2011*

№ док.	Участок	Бригада	К.Ф.М.	Корреспонд счет	Гарантия
	01	04	014		

Ф.И.О.	Отработано часов				1							Профес-сия	Таб. №	Разряд	Виды оплат			
															100			
															Часовая тарифная ставка / Часы фактические			
Иванов И. И.					*8*							*101*	*111*	*6*	*9,999* / *8*			
Петров П. П.					*8*							*101*	*222*	*5*	*8,888* / *8*			
Сидоров С. С.					*8*							*101*	*333*	*4*	*7,777* / *8*			
Тронов М. А.					*8*							*101*	*444*	*4*	*6,666* / *8*			

Рисунок 1 - Образец наряда (личевая сторона)

Доменный 11 Ст 45 Ø40

	Наименование работ	Разряд работ	Тарифная сетка звена	Ед. изм.	Задано	Норма на единицу		К-во выполнено
						время	расценка	
	Отковать							
6	болт ГОСТ 7917	6544	8,333	шт	100	0,5	4,167	100

Начальник цеха _____ Мастер _____

Начальник БОТ (инженер по ОиНТ) _____

Работу сдал _____ Работу принял _____

Рисунок 2 - Образец наряда (тыльная сторона)

Каждый элемент перечня исполнителей работ содержит следующую информацию:

- личный табельный номер сотрудника;
- фамилию, имя и отчество сотрудника;
- шифр профессии, по которой сотрудник фигурирует в наряде;
- разряд по указанной профессии.

Каждый элемент перечня затрат времени содержит следующую информацию:

- дату ведения работ;
- время в часах, затраченное на производство работ.

Каждый элемент перечня выполненых работ содержит следующую информацию:

- наименование работ;
- номер чертежа или схемы на произведенные работы;

- количественную составляющую работ;

- норму времени на единицу работ;

- расценку на единицу работ.

В целях обеспечения дружественного пользовательского интерфейса, необходимо создание вспомогательных справочных таблиц, таких как:

- справочник сотрудников;

- справочник профессий;

- таблица текущего состояния производственной структуры организации.

Необходимо обеспечить возможность генерации табеля выхождаемости сотрудников на основе данных полученных из нарядов. Табель должен содержать перечень сотрудников выбранного подразделения с указанием времени отработанного в каждый из дней учетного месяца. Поскольку просмотр суммарной информации может выявить ошибку в данных, обусловленую человеческим фактором, необходимо обеспечить оперативный доступ к документам, содержащим данные на основе которых генерируется табель.

Производственная структура имеет иерархический вид, в котором корнями иерархии являются производственные участки. Каждый участок может содержать неограниченное количество бригад. Каждая бригада, в свою очередь, содержит в своем составе сотрудников, которые замыкают иерархию производственной структуры.

2. РАЗРАБОТКА СТРУКТУРЫ БАЗЫ ДАННЫХ ПРИЛОЖЕНИЯ

Использование реляционной базы данных задает ряд ограничений относительно объектных баз данных. В их числе - необходимость косвенной реализации историчности данных. В случае реляционной модели историчность (версионность) данных можно обеспечить посредством добавления дополнительных полей указывающих дату ввода данных (версию данных). Поскольку справочные данные могут быть подвержены изменениям, в целях сохранения целостности документов использующих эти данные, все справочники имеют двойную индексацию в виде полей "Id" - уникального идентификатора в таблице, и "Code" - шифра записи, который уникален только на протяжении указанного промежутка времени. Этот промежуток устанавливается в двух дополнительных полях каждой таблицы базы данных, используемой справочником.

Таким образом, с учетом требований предъявленных к приложению, база данных содержит следующие таблицы:

Areas - содержит информацию об участках. Включает такие поля как:

Id - уникальный идентификатор участка;

Code - шифр участка;

Title - название участка;

Begin - дата создания участка;

End - дата удаления участка.

Brigades - содержит информацию о бригадах. Включает поля:

Id - уникальный идентификатор бригады;

AreaId - уникальный идентификатор участка, содержащего

данную бригаду;

Code - шифр бригады;

Title - название бригады;

Begin - дата создания бригады;

End - дата удаления бригады.

BrigadePersons - содержит описания текущих связей сотрудников с бригадами, иными словами описывает состав каждой бригады в определенный момент времени. Включает поля:

Id - уникальный идентификатор связи;

BrigadeId - идентификатор бригады;

PersonId - идентификатор сотрудника;

Begin - дата включения сотрудника в состав бригады;

End - дата исключения сотрудника из состава бригады.

Persons - содержит данные о сотрудниках:

Id - уникальный идентификатор сотрудника;

Code - табельный номер сотрудника;

FirstName - имя сотрудника;

MiddleName - отчество сотрудника;

LastName - фамилия сотрудника;

Begin - дата приема сотрудника в организацию;

End - дата увольнения сотрудника.

PersonProfessions - описывает текущую квалификацию сотрудников. Включает поля:

Id - уникальный идентификатор связи;

PersonId - идентификатор сотрудника;

ProfessionId - идентификатор сотрудника;

Begin - дата включения сотрудника в состав бригады;

End - дата исключения сотрудника из состава бригады.

Professions - содержит записи о профессиях и их тарифах, в том числе:

Id - уникальный идентификатор профессии;

Code - шифр профессии;

Title - название профессии;

Rank1 - тариф первого разряда профессии;

Rank2 - тариф второго разряда профессии;

Rank3 - тариф третьего разряда профессии;

Rank4 - тариф четвертого разряда профессии;

Rank5 - тариф пятого разряда профессии;

Rank6 - тариф шестого разряда профессии;

Begin - дата включения сотрудника в состав бригады;

End - дата исключения сотрудника из состава бригады.

Warranties - содержит данные о нарядах, выраженные полями:

Id - уникальный идентификатор записи;

Customer - название заказчика;

Order - номер заказа;

WarrantyDate - дата последней редакции наряда;

AreaId - идентификатор участка на котором производились работы;

BrigadeId - идентификатор бригады, производившей работы.

Positions - содержит позиции нарядов, выраженные следующимим

полями:

Id - уникальный идентификатор позиции;

WarrantyId - идентификатор наряда, которому принадлежит позиция;

Title - наименование работ;

Draw - номер чертежа;

Matherial - материал, затраченный при производстве работ;

Number - количественная составляющая произведенных работ;

Mass - масса единицы результата труда;

Norm - норма времени на единицу результата труда;

Price - расценка на единицу результата труда.

Executors - содержит записи об исполнителях работ. Поля таблицы:

Id - уникальный идентификатор исполнителя;

WarrantyId - идентификатор наряда, которому принадлежит запись об исполнителе работ;

PersonId - идентификатор сотрудника;

ProfessionId - идентификатор профессии, по которой сотрудник фигурирует в наряде;

Rank - разряд по профессии сотрудника, определенный сложностью работ.

Labors - содержит записи о затратах времени, отмеченных в нарядах. Описана следующими полями:

Id - идентификатор записи;

WarrantyId - идентификатор наряда, которому соответствует запись;

LaborDate - дата ведения работ;

Hours - затраченое время.

3. РЕАЛИЗАЦИЯ ДОСТУПА К ДАННЫМ В ПРИЛОЖЕНИИ

Среда разработки обладает средством визуального проектирования кэша базы данных в объекте типа DataSet. Однако DataSet отображает реляционную структуру данных, а при разработке приложения удобно использовать объектно-ориентированый источник данных. Поэтому, в приложении был разработан собственный объектный кэш данных. Его структура описывается нижеуказанными принципами.

Проект приложения содержит три файла обеспечивающих взаимодействие приложения с базами данных. В их число входят:

- файл Elements.cs - содержит описание строк каждой из таблиц базы данных, поддерживающее интерфейсы для проверки эквивалентности строк. Диаграмма классов файла Elements.cs представлена на рисунке 3;

- файл Tables.cs - содержит методы доступа к данным каждой таблицы. Таблицы в кэше поддерживают интерфейс System.Collections.Generic.IEnumerable<T>, что позволяет с помощью методов расширения получить оперативный доступ к типизированному содержимому таблиц. Диаграмма классов файла Tables.cs представлена на рисунке 4;

- файл Data.cs - содержит механизмы управления базами данных.

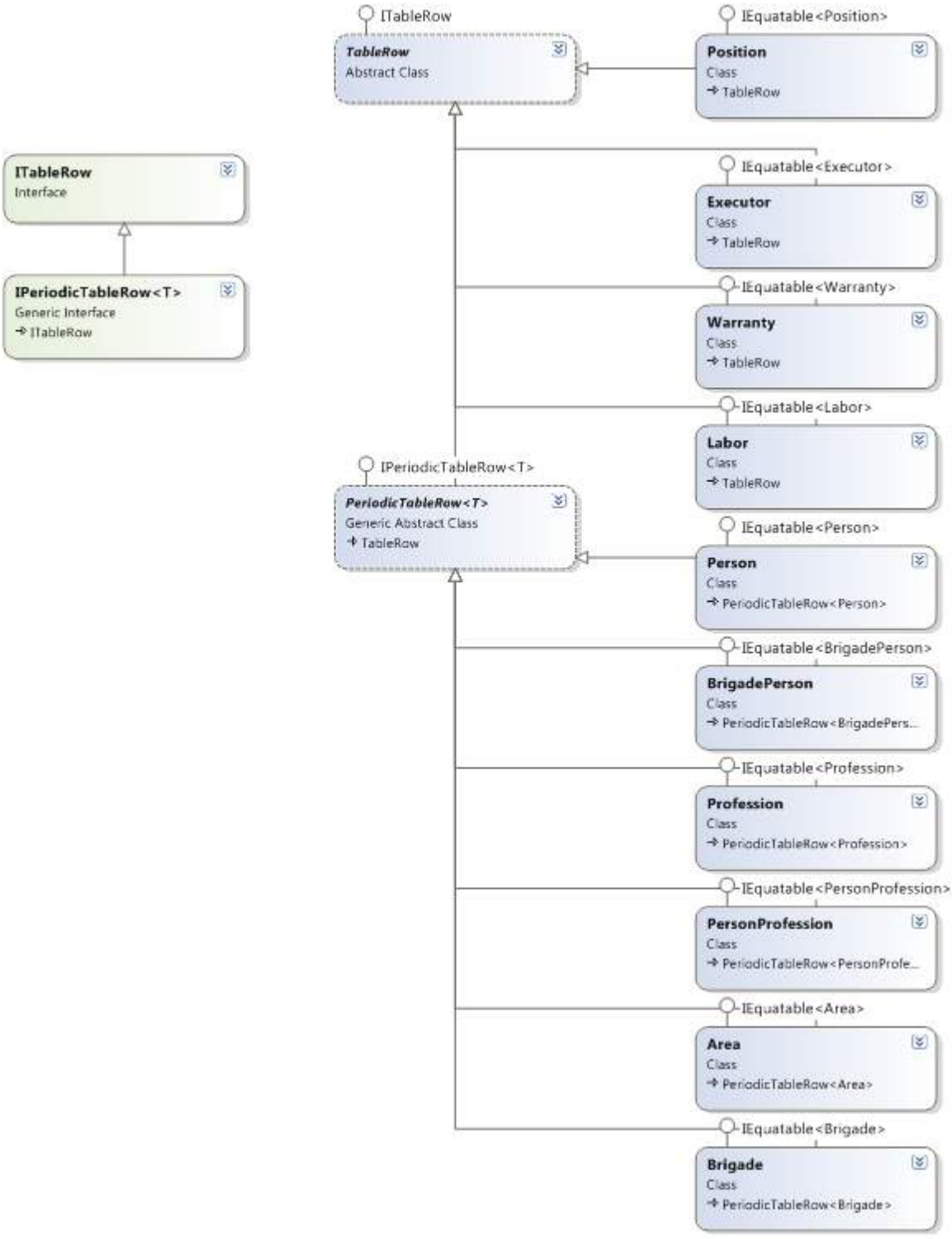

Рисунок 3 - Диаграмма классов файла Elements.cs

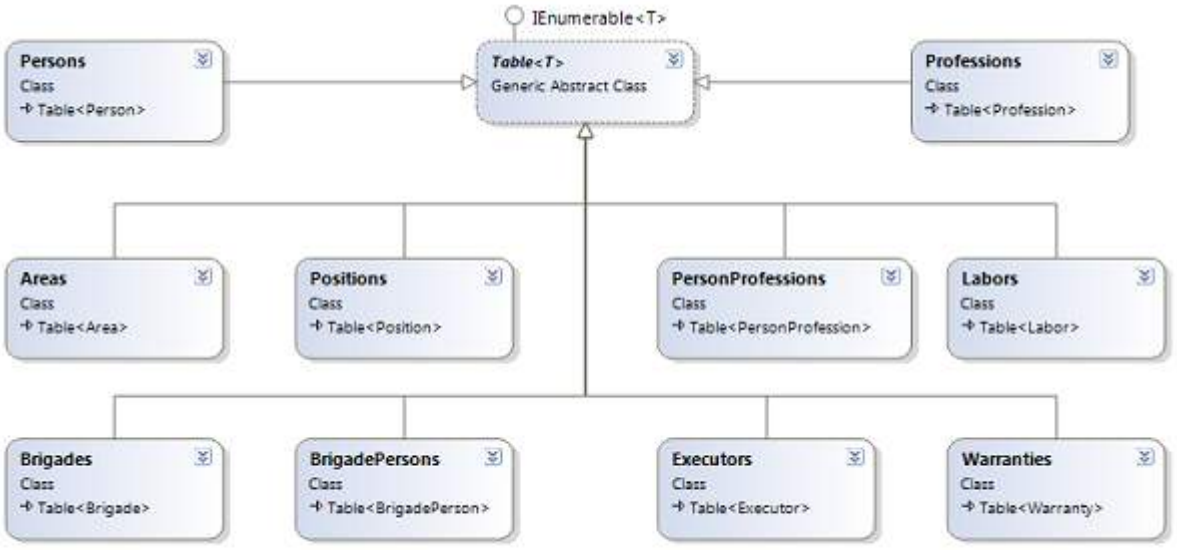

Рисунок 4 - Диаграмма классов файла Elements.cs

4. ОСОБЕННОСТИ РЕАЛИЗАЦИИ ИНТЕРФЕЙСА ПОЛЬЗОВАТЕЛЯ

Взаимодействие пользователя с программой происходит в MDI интерфейсе.

Для создания и редактирования строк каждой таблицы текущей базы данных используется одна форма. В соответствующих формах предусмотрено открытое свойство CatalogMode, настраивающее поведение и отображение формы в соответствии с выбраной задачей.

Для упрощения проектирования форм обслуживающих справочные таблицы, создан пользовательский элемент управления TableControl наследующий класс System.Windows.Forms.UserControl. Для каждой справочной таблицы создан пользовательский элемент управления наследующий TableControl и дополняющий его в соответствии со

спецификой таблицы.

Формы отображающие содержимое таблиц базы данных используют готовые пользовательские элементы управления таблиц.

5. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ ПРОГРАММЫ "АРМ ТАБЕЛЬЩИКА"

При запуске программы, после демонстрации заставки, на экране появляется главная форма, которая содержит в себе пункты главного меню:

- файл;
- вид;
- справка.

Меню "Файл" предоставляет возможность выхода из программы через подпункт "Выход".

Меню "Вид" (рисунок 5) содержит подпункты предоставляющие доступ к форме структуры организации, справочникам и документам.

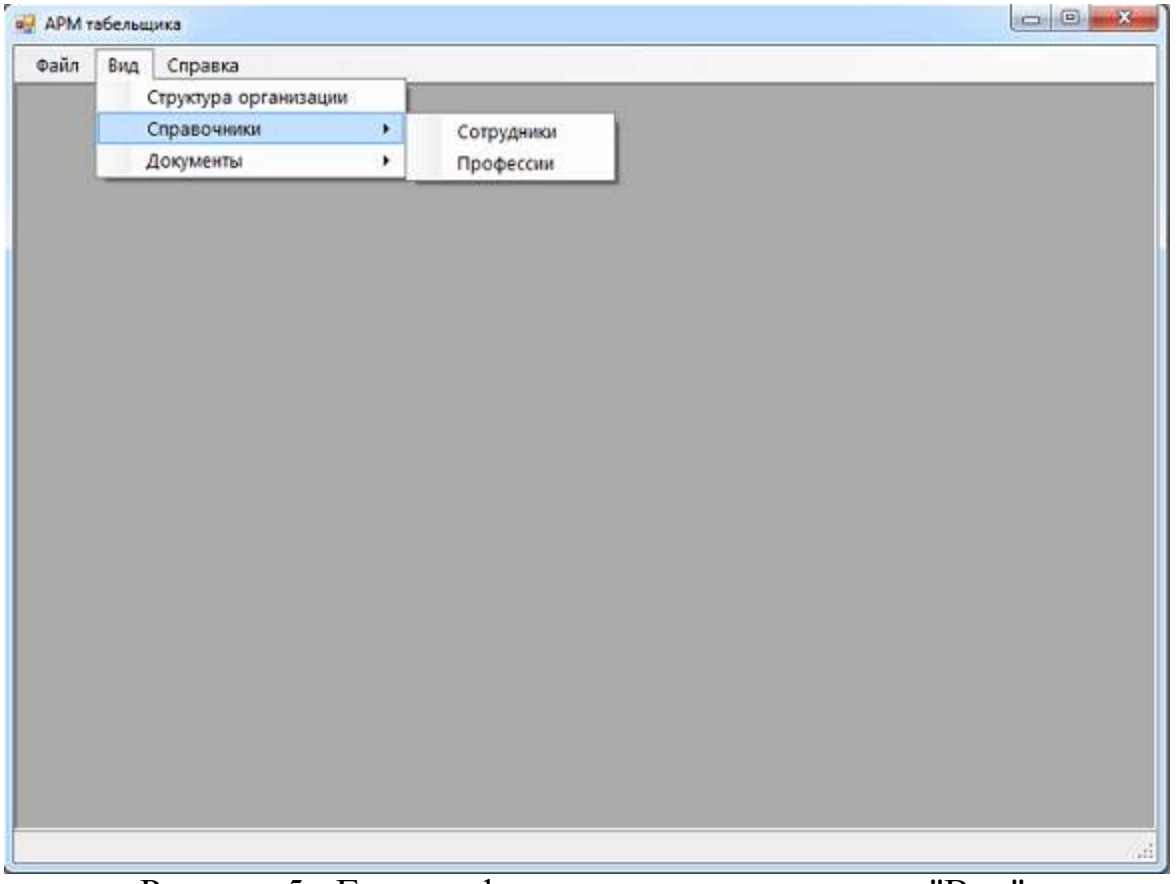

Рисунок 5 - Главная форма приложения и меню "Вид".

Меню "Справка" предоставляет возможность просмотра сведений о программе через подпункт "О программе".

Далее рассмотрены формы доступные из меню "Вид".

При выборе подпункта меню - "Структура организации", открывается форма "Структура организации" (рисунок 6). Чтобы добавить участок, необходимо нажатьна кнопку "Добавить участок". Добавление бригад возможно посредством выбора соответствующего пункта контекстного меню после щелчка правой кнопкой мыши на участке. Редактирование и удаление участков и бригад также доступно через контекстное меню.

При выборе подпункта меню - "Справочник сотрудников", открывается форма "Справочник сотрудников" (рисунок 7).

Редактирование и удаление сотрудников, а также их квалификаций доступно через соответствующие кнопки или через контекстные меню.

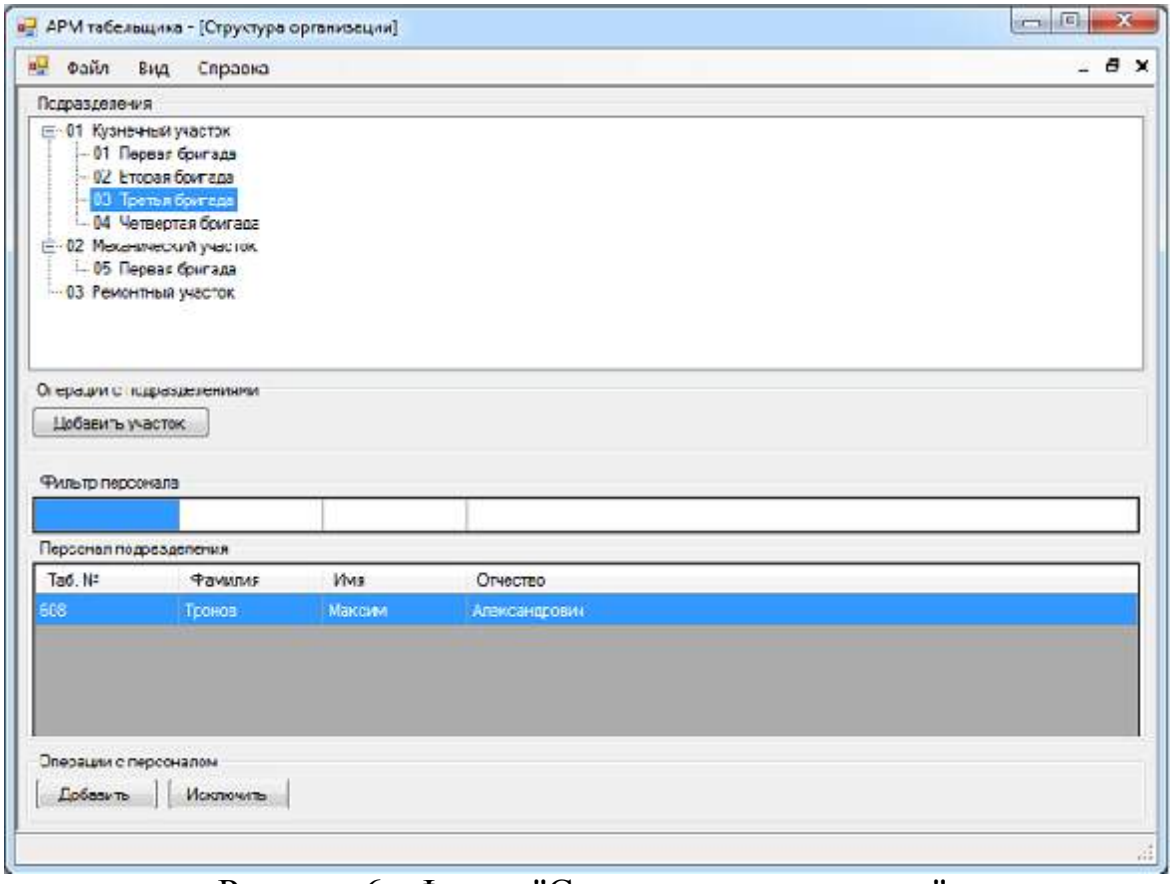

Рисунок 6 - Форма "Структура организации".

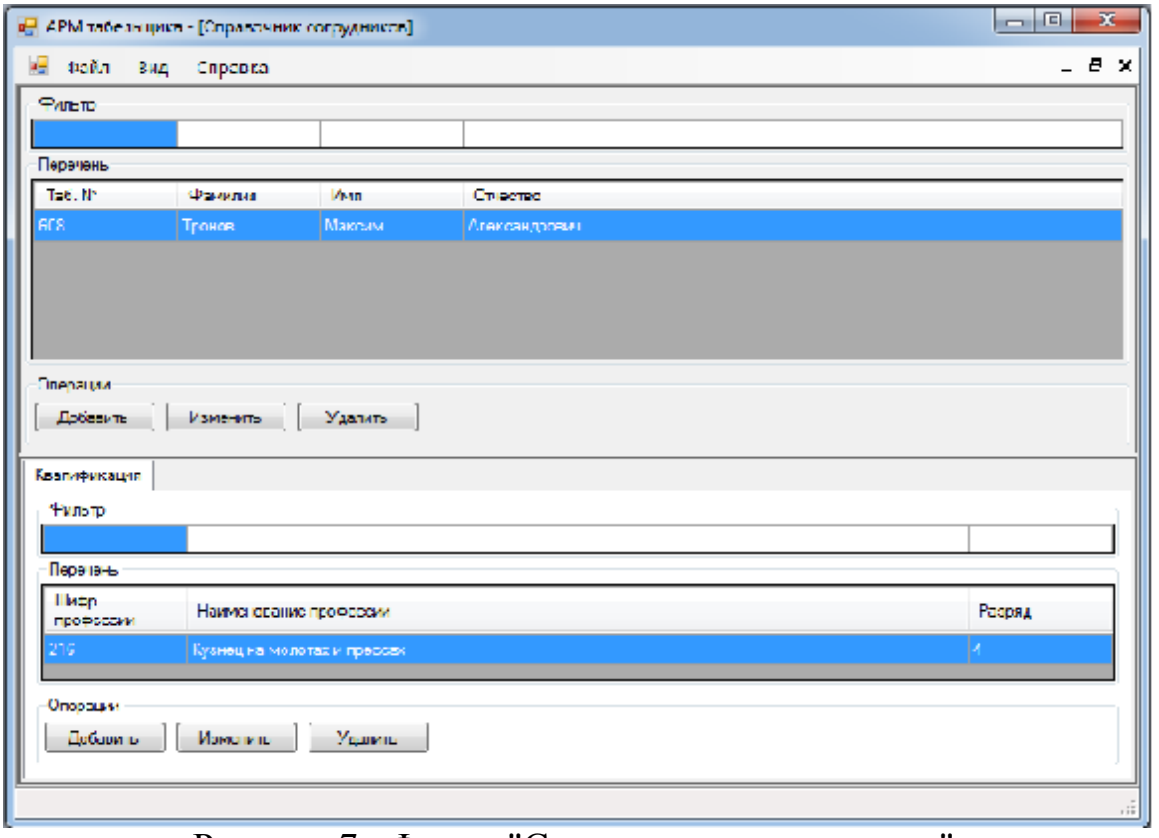

Рисунок 7 - Форма "Справочник сотрудников".

При выборе подпункта меню - "Справочник профессий", открывается форма "Справочник профессий" (рисунок 8).

Добавление, редактирование и удаление профессий доступно через соответствующие кнопки или через контекстные меню.

При выборе подпункта меню - "Справочник нарядов", открывается форма "Справочник нарядов" (рисунок 9).

Добавление, редактирование и удаление нарядов доступно через соответствующие кнопки или через контекстные меню.

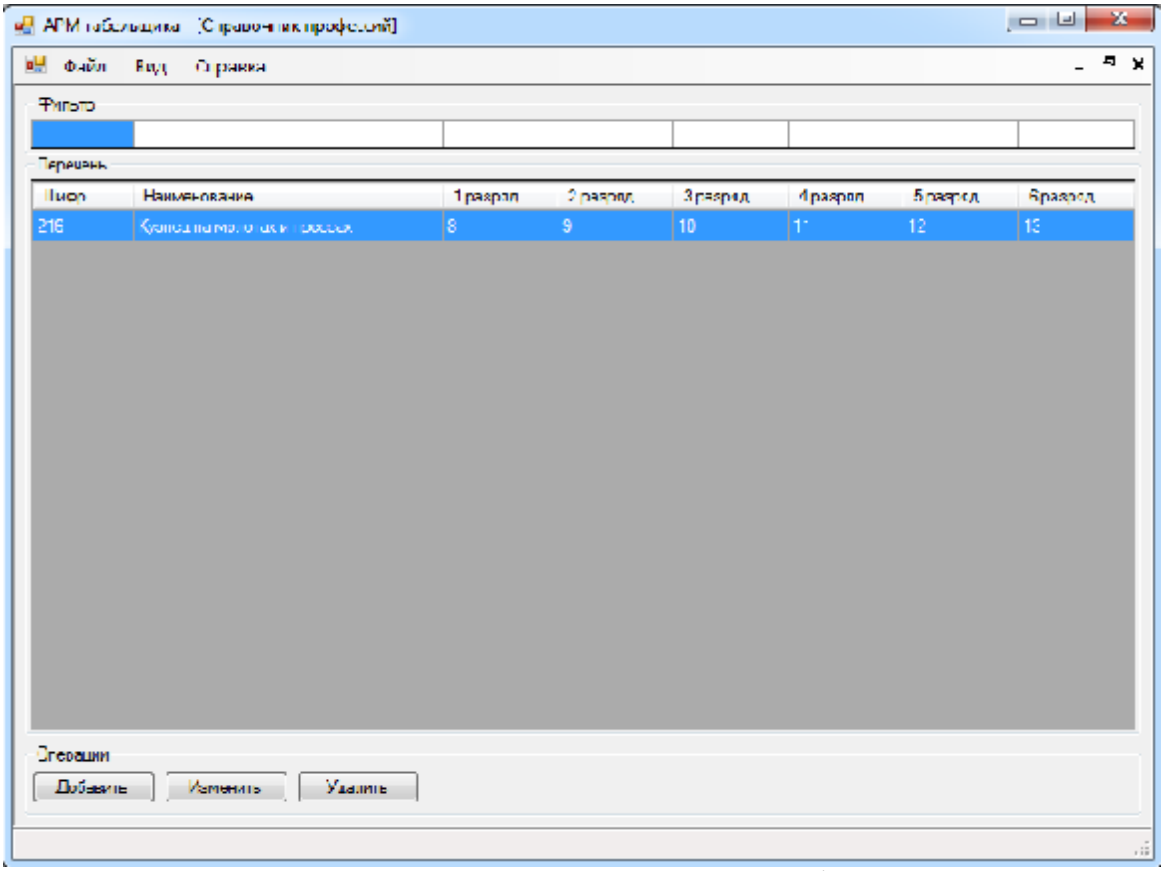

Рисунок 8 - Форма "Справочник профессий".

При выборе подпункта меню - "Табель", открывается форма "Табель" (рисунок 10).

Для генерации табеля необходимо указать интересующую бригаду посредством нажания на кнопку "Бригада" и отчетный месяц в элементе управления "Месяц".

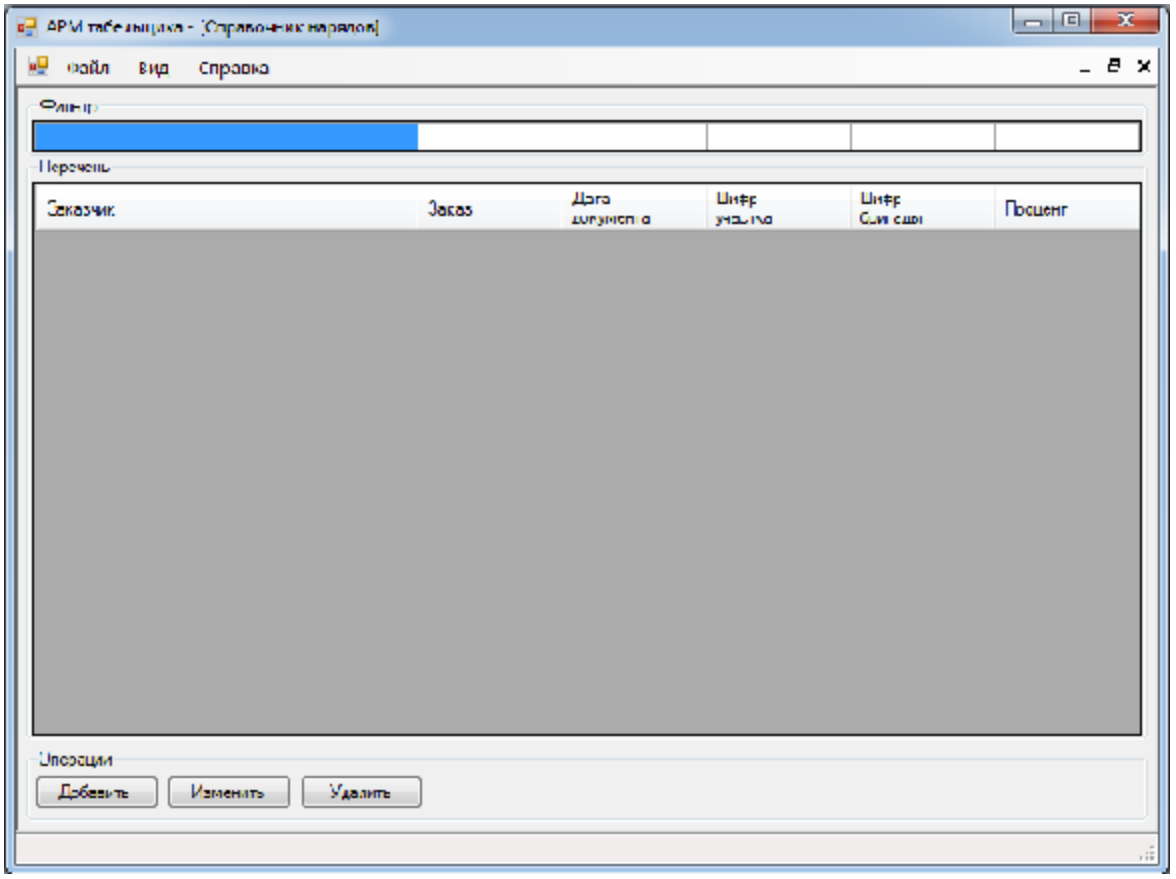

Рисунок 9 - Форма "Справочник нарядов".

При выборе подпункта меню - "Табель", открывается форма "Табель" (рисунок 10).

Для генерации табеля необходимо указать интересующую бригаду посредством нажания на кнопку "Бригада" и отчетный месяц в элементе управления "Месяц".

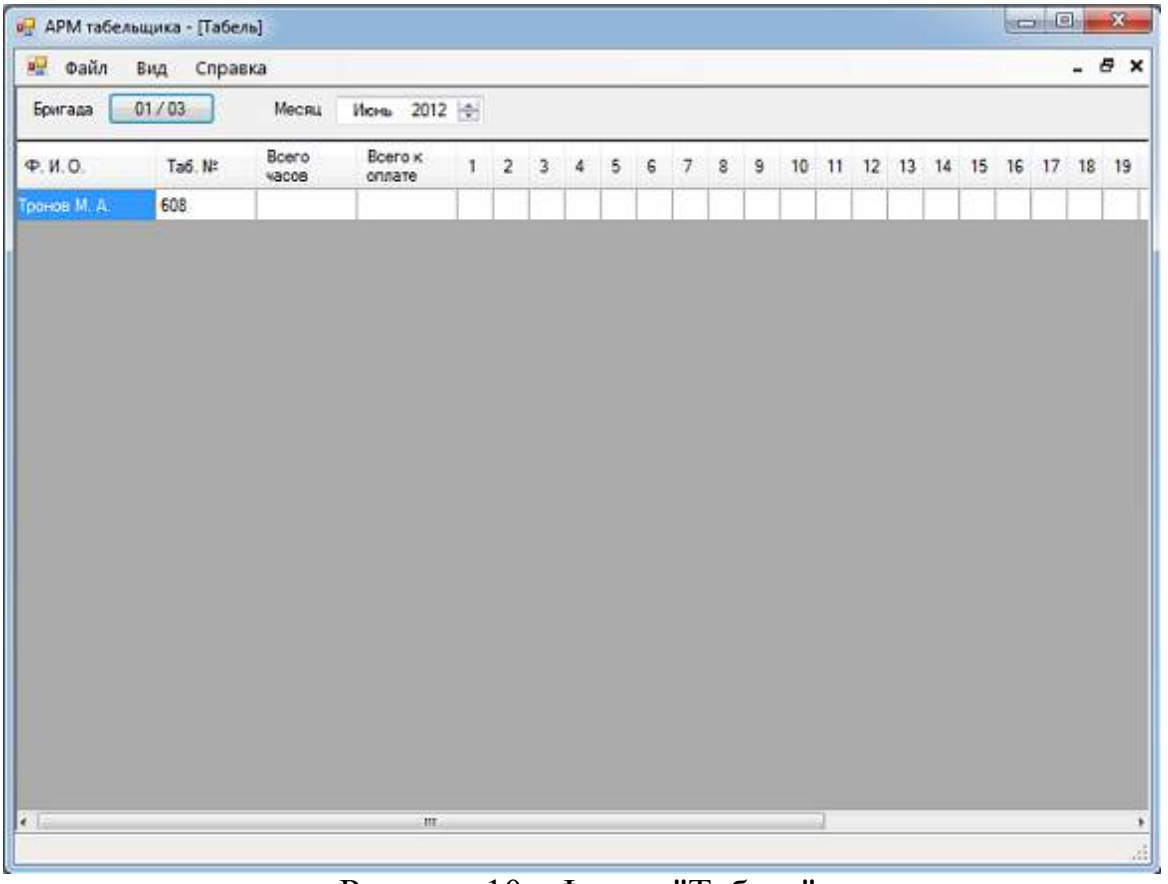

Рисунок 10 - Форма "Табель".

6. ИСХОДНЫЙ КОД ПРОГРАММЫ

Файл Project\Databases\Elements.cs:

```
using System;
using System.Data.OleDb;
using System.Linq;

namespace Project.Databases
{
 public interface ITableRow
 {
  int Id { get; set; }
 }

 public abstract class TableRow : ITableRow
 {
  protected int _Id;
  public int Id
  {
   get
   {
    return this._Id;
   }
   set
   {
    this._Id = value;
   }
  }
 }

 public interface IPeriodicTableRow<T> : ITableRow
 {
  DateTime Begin { get; }
  DateTime End { get; }
  bool IsActive { get; }
  bool IsUsed { get; }
  T Clone();
  void Update(T item);
  void Delete();
 }

 public abstract class PeriodicTableRow<T> : TableRow,
IPeriodicTableRow<T> where T : PeriodicTableRow<T>
 {
  protected DateTime _Begin;
  protected DateTime _End;

  public DateTime Begin
  {
   get
```

```
        {
         return this._Begin;
        }
        set
        {
         T item = this.Clone();
         item._Begin = value;
         this.Update(item);
        }
       }
       public DateTime End
       {
        get
        {
         return this._End;
        }
        set
        {
         T item = this.Clone();
         item._End = value;
         this.Update(item);
        }
       }

       public bool IsActive
       {
        get
        {
         return this.Begin.CompareTo(DateTime.Now) <= 0 &&
this.End.CompareTo(DateTime.Now) > 0;
        }
       }
       public abstract bol IsUsed { get; }
       public abstract T Clone();
       public abstract void Update(T item);
       public abstract void Delete();

      }

      public class Person : PeriodicTableRow<Person>,
IEquatable<Person>
      {
       public Person(OleDbDataReader reader)
       {
        this._Id = Convert.ToInt32(reader["Id"]);
        this._Code = Convert.ToInt16(reader["Code"]);
        this._FirstName =
Convert.ToString(reader["FirstName"]);
        this._MiddleName =
Convert.ToString(reader["MiddleName"]);
        this._LastName = Convert.ToString(reader["LastName"]);
```

```
            this._Begin = Convert.ToDateTime(reader["Begin"]);
            this._End = Convert.ToDateTime(reader["End"]);
        }
        public Person(short Code, string FirstName, string
MiddleName, string LastName)
        {
            this._Id = 0;
            this._Code = Code;
            this._FirstName = FirstName;
            this._MiddleName = MiddleName;
            this._LastName = LastName;
            this._Begin = DateTime.Now.Date;
            this._End = new DateTime(9000, 1, 1);
        }
        public Person(short Code, string FirstName, string
MiddleName, string LastName, DateTime Begin, DateTime
End)
        {
            this._Id = 0;
            this._Code = Code;
            this._FirstName = FirstName;
            this._MiddleName = MiddleName;
            this._LastName = LastName;
            this._Begin = Begin;
            this._End = End;
        }

        private short _Code;
        private string _FirstName;
        private string _MiddleName;
        private string _LastName;

        public short Code { get { return this._Code; } }
        public string FirstName { get { return this._FirstName;
} }
        public string MiddleName { get { return
this._MiddleName; } }
        public string LastName { get { return this._LastName; }
}
        public BrigadePersons BrigadePersons
        {
            get
            {
                BrigadePersons brigadePersons = new BrigadePersons();
                brigadePersons.Clear();
                var items = Data.Tables.BrigadePersons._Items.Where(r
=> r.Value.PersonId.Equals(this.Id)).ToArray();
                foreach (var item in items)
brigadePersons._Items.Add(item.Key, item.Value);
                return brigadePersons;
            }
```

```
    }
    public Executors Executors
    {
     get
     {
      Executors executors = new Executors();
      var items = Data.Tables.Executors._Items.Where(r =>
r.Value.PersonId.Equals(this.Id)).ToArray();
      foreach (var item in items)
executors._Items.Add(item.Key, item.Value);
      return executors;
     }
    }
    public PersonProfessions PersonProfessions
    {
     get
     {
      PersonProfessions personeProfessions = new
PersonProfessions();
      var items =
Data.Tables.PersonProfessions._Items.Where(r =>
r.Value.PersonId.Equals(this.Id)).ToArray();
      foreach (var item in items)
personeProfessions._Items.Add(item.Key, item.Value);
      return personeProfessions;
     }
    }

    public override bool IsUsed
    {
     get
     {
      return !this.Executors.Count().Equals(0);
     }
    }

    public override Person Clone()
    {
     return new Person(this.Code, this.FirstName,
this.MiddleName, this.LastName, this.Begin, this.End);
    }

    public override void Update(Person newItem)
    {
     Data.Tables.Persons.Update(this, newItem);
    }
    public override void Delete()
    {
     foreach (BrigadePerson brigadePerson in
this.BrigadePersons)
      brigadePerson.Delete();
```

```
    foreach (PersonProfession personProfession in
this.PersonProfessions)
     personProfession.Delete();

    if (this.IsUsed)
    {
     Person item = this.Clone();
     item._End = DateTime.Now.Date;
     this.Update(item);
    }
    else
     Data.Tables.Persons.Delete(this);
   }

   public bool Equals(Person other)
   {
    if (
     this._Code.Equals(other._Code) &&
     this._FirstName.Equals(other._FirstName) &&
     this._MiddleName.Equals(other._MiddleName) &&
     this._LastName.Equals(other._LastName) &&
     this._Begin.Equals(other._Begin) &&
     this._End.Equals(other._End)
     )
     return true;
    else return false;
   }
   public override bool Equals(Object obj)
   {
    if (obj == null) return base.Equals(obj);

    if (!(obj is Person))
     throw new InvalidCastException("The 'obj' argument is
not a Person object.");
    else
     return Equals(obj as Person);
   }
   public override int GetHashCode()
   {
    string hash =
     this.Id.GetHashCode().ToString() +
     this.Code.GetHashCode().ToString() +
     this.FirstName.GetHashCode().ToString() +
     this.MiddleName.GetHashCode().ToString() +
     this.LastName.GetHashCode().ToString() +
     this.Begin.GetHashCode().ToString() +
     this.End.GetHashCode().ToString();
    return hash.GetHashCode();
   }
  }
```

```
 public class Profession : PeriodicTableRow<Profession>,
IEquatable<Profession>
 {
  public Profession(OleDbDataReader reader)
  {
   this._Id = Convert.ToInt32(reader["Id"]);
   this._Code = Convert.ToInt16(reader["Code"]);
   this._Title = Convert.ToString(reader["Title"]);
   this._Rank1 = Convert.ToSingle(reader["Rank1"]);
   this._Rank2 = Convert.ToSingle(reader["Rank2"]);
   this._Rank3 = Convert.ToSingle(reader["Rank3"]);
   this._Rank4 = Convert.ToSingle(reader["Rank4"]);
   this._Rank5 = Convert.ToSingle(reader["Rank5"]);
   this._Rank6 = Convert.ToSingle(reader["Rank6"]);
   this._Begin = Convert.ToDateTime(reader["Begin"]);
   this._End = Convert.ToDateTime(reader["End"]);
  }
  public Profession(short Code, string Title, float
Rank1, float Rank2, float Rank3, float Rank4, float
Rank5, float Rank6)
  {
   this._Id = 0;
   this._Code = Code;
   this._Title = Title;
   this._Rank1 = Rank1;
   this._Rank2 = Rank2;
   this._Rank3 = Rank3;
   this._Rank4 = Rank4;
   this._Rank5 = Rank5;
   this._Rank6 = Rank6;
   this._Begin = DateTime.Now.Date;
   this._End = new DateTime(9000, 1, 1);
  }
  public Profession(short Code, string Title, float
Rank1, float Rank2, float Rank3, float Rank4, float
Rank5, float Rank6, DateTime Begin, DateTime End)
  {
   this._Id = 0;
   this._Code = Code;
   this._Title = Title;
   this._Rank1 = Rank1;
   this._Rank2 = Rank2;
   this._Rank3 = Rank3;
   this._Rank4 = Rank4;
   this._Rank5 = Rank5;
   this._Rank6 = Rank6;
   this._Begin = Begin;
   this._End = End;
  }

  private short _Code;
```

```
    private string _Title;
    private float _Rank1;
    private float _Rank2;
    private float _Rank3;
    private float _Rank4;
    private float _Rank5;
    private float _Rank6;


    public short Code { get { return this._Code; } }
    public string Title { get { return this._Title; } }
    public float Rank1 { get { return this._Rank1; } }
    public float Rank2 { get { return this._Rank2; } }
    public float Rank3 { get { return this._Rank3; } }
    public float Rank4 { get { return this._Rank4; } }
    public float Rank5 { get { return this._Rank5; } }
    public float Rank6 { get { return this._Rank6; } }
    public PersonProfessions PersonProfessions
    {
     get
     {
      PersonProfessions personProfessions = new
PersonProfessions();
      var items =
Data.Tables.PersonProfessions._Items.Where(r =>
r.Value.ProfessionId.Equals(this.Id)).ToArray();
      foreach (var item in items)
personProfessions._Items.Add(item.Key, item.Value);
      return personProfessions;
     }
    }
    public Executors Executors
    {
     get
     {
      Executors executors = new Executors();
      var items = Data.Tables.Executors._Items.Where(r =>
r.Value.ProfessionId.Equals(this.Id)).ToArray();
      foreach (var item in items)
executors._Items.Add(item.Key, item.Value);
      return executors;
     }
    }

    public override bool IsUsed
    {
     get
     {
      return !this.Executors.Count().Equals(0);
     }
    }
```

```
    public override Profession Clone()
    {
     return new Profession(this.Code, this.Title,
      this.Rank1, this.Rank2, this.Rank3, this.Rank4,
this.Rank5, this.Rank6,
      this.Begin, this.End);
    }

    public override void Update(Profession newItem)
    {
     Data.Tables.Professions.Update(this, newItem);
    }
    public override void Delete()
    {
     foreach (PersonProfession personProfession in
this.PersonProfessions)
      personProfession.Delete();
     if (this.IsUsed)
     {
      Profession item = this.Clone();
      item._End = DateTime.Now.Date;
      this.Update(item);
     }
     else
      Data.Tables.Professions.Delete(this);
    }

    public bool Equals(Profession other)
    {
     if (
      this._Code.Equals(other._Code) &&
      this._Title.Equals(other._Title) &&
      this._Rank1.Equals(other._Rank1) &&
      this._Rank2.Equals(other._Rank2) &&
      this._Rank3.Equals(other._Rank3) &&
      this._Rank4.Equals(other._Rank4) &&
      this._Rank5.Equals(other._Rank5) &&
      this._Rank6.Equals(other._Rank6) &&
      this._Begin.Equals(other._Begin) &&
      this._End.Equals(other._End)
      )
      return true;
     else return false;
    }
    public override bool Equals(Object obj)
    {
     if (obj == null) return base.Equals(obj);

     if (!(obj is Profession))
      throw new InvalidCastException("The 'obj' argument is
```

```
not a Profession object...");
    else
     return Equals(obj as Profession);
   }
   public override int GetHashCode()
   {
    string hash =
     this.Id.GetHashCode().ToString() +
     this.Code.GetHashCode().ToString() +
     this.Title.GetHashCode().ToString() +
     this.Rank1.GetHashCode().ToString() +
     this.Rank2.GetHashCode().ToString() +
     this.Rank3.GetHashCode().ToString() +
     this.Rank4.GetHashCode().ToString() +
     this.Rank5.GetHashCode().ToString() +
     this.Rank6.GetHashCode().ToString() +
     this.Begin.GetHashCode().ToString() +
     this.End.GetHashCode().ToString();
    return hash.GetHashCode();
   }
  }

  public class Warranty : TableRow, IEquatable<Warranty>
  {
   public Warranty(OleDbDataReader reader)
   {
    this._Id = Convert.ToInt32(reader["Id"]);
    this._Customer = Convert.ToString(reader["Customer"]);
    this._Order = Convert.ToInt16(reader["Order"]);
    this._Percent = Convert.ToSingle(reader["Percent"]);
    this._WarrantyDate =
Convert.ToDateTime(reader["WarrantyDate"]);
    this._AreaId = Convert.ToInt32(reader["AreaId"]);
    this._BrigadeId = Convert.ToInt32(reader["BrigadeId"]);
   }
   public Warranty(string Customer, short Order, float
Percent, DateTime WarrantyDate, int AreaId, int
BrigadeId)
   {
    this._Id = 0;
    this._Customer = Customer;
    this._Order = Order;
    this._Percent = Percent;
    this._WarrantyDate = WarrantyDate;
    this._AreaId = AreaId;
    this._BrigadeId = BrigadeId;
   }

   private string _Customer;
   private short _Order;
   private float _Percent;
```

```
    private DateTime _WarrantyDate;
    private int _AreaId;
    private int _BrigadeId;

    public string Customer { get { return this._Customer; }
}
    public short Order { get { return this._Order; } }
    public float Percent { get { return this._Percent; } }
    public DateTime WarrantyDate { get { return
this._WarrantyDate; } }
    public int AreaId { get { return this._AreaId; } }
    public int BrigadeId { get { return this._BrigadeId; } }
    public Executors Executors
    {
     get
     {
      Executors executors = new Executors();
      var items = Data.Tables.Executors._Items.Where(r =>
r.Value.WarrantyId.Equals(this.Id)).ToArray();
      foreach (var item in items)
executors._Items.Add(item.Key, item.Value);
      return executors;
     }
    }
    public Labors Labors
    {
     get
     {
      Labors labors = new Labors();
      var items = Data.Tables.Labors._Items.Where(r =>
r.Value.WarrantyId.Equals(this.Id)).ToArray();
      foreach (var item in items)
labors._Items.Add(item.Key, item.Value);
      return labors;
     }
    }
    public Positions Positions
    {
     get
     {
      Positions positions = new Positions();
      var items = Data.Tables.Positions._Items.Where(r =>
r.Value.WarrantyId.Equals(this.Id)).ToArray();
      foreach (var item in items)
positions._Items.Add(item.Key, item.Value);
      return positions;
     }
    }

    public void Update(Warranty newItem)
    {
```

```
        Data.Tables.Warranties.Update(this, newItem);
      }
      public void Delete()
      {
       foreach (Position position in this.Positions)
        position.Delete();
       foreach (Executor executor in this.Executors)
        executor.Delete();
       Data.Tables.Warranties.Delete(this);
      }

      public bool Equals(Warranty other)
      {
       if (this._Customer.Equals(other._Customer) &&
        this._Order.Equals(other._Order) &&
        this._Percent.Equals(other._Percent) &&
        this._WarrantyDate.Equals(other._WarrantyDate) &&
        this._AreaId.Equals(other._AreaId) &&
        this._BrigadeId.Equals(other._BrigadeId))
        return true;
       else return false;
      }
      public override bool Equals(Object obj)
      {
       if (obj == null) return base.Equals(obj);

       if (!(obj is Warranty))
        throw new InvalidCastException("The 'obj' argument is
    not a Warranty object.");
       else
        return Equals(obj as Warranty);
      }
      public override int GetHashCode()
      {
       string hash =
        this.Id.GetHashCode().ToString() +
        this.Customer.GetHashCode().ToString() +
        this.Order.GetHashCode().ToString() +
        this.Percent.GetHashCode().ToString() +
        this.WarrantyDate.GetHashCode().ToString() +
        this.AreaId.GetHashCode().ToString() +
        this.BrigadeId.GetHashCode().ToString();
       return hash.GetHashCode();
      }
     }

     public class Area : PeriodicTableRow<Area>,
    IEquatable<Area>
     {
      public Area(OleDbDataReader reader)
      {
```

```
            this._Id = Convert.ToInt32(reader["Id"]);
            this._Code = Convert.ToByte(reader["Code"]);
            this._Title = Convert.ToString(reader["Title"]);
            this._Begin = Convert.ToDateTime(reader["Begin"]);
            this._End = Convert.ToDateTime(reader["End"]);
        }
        public Area(byte Code, string Title)
        {
            this._Id = 0;
            this._Code = Code;
            this._Title = Title;
            this._Begin = DateTime.Now.Date;
            this._End = new DateTime(9000, 1, 1);
        }
        public Area(byte Code, string Title, DateTime Begin,
    DateTime End)
        {
            this._Id = 0;
            this._Code = Code;
            this._Title = Title;
            this._Begin = Begin;
            this._End = End;
        }

        private byte _Code;
        private string _Title;

        public byte Code { get { return this._Code; } }
        public string Title { get { return this._Title; } }
        public Brigades Brigades
        {
            get
            {
                Brigades brigades = new Brigades();
                var items = Data.Tables.Brigades._Items
                    .Where(r => r.Value.AreaId.Equals(this.Id))
                    .ToArray();
                foreach (var item in items)
    brigades._Items.Add(item.Key, item.Value);
                return brigades;
            }
        }
        public Warranties Warranties
        {
            get
            {
                Warranties warranties = new Warranties();
                var items = Data.Tables.Warranties._Items.Where(r =>
    r.Value.BrigadeId.Equals(this.Id)).ToArray();
                foreach (var item in items)
    warranties._Items.Add(item.Key, item.Value);
```

```
      return warranties;
     }
    }

    public override bool IsUsed
    {
     get
     {
      return !this.Warranties.Count().Equals(0);
     }
    }

    public override Area Clone()
    {
     return new Area(this.Code, this.Title, this.Begin,
  this.End);
    }

    public override void Update(Area newItem)
    {
     Data.Tables.Areas.Update(this, newItem);
    }
    public override void Delete()
    {
     foreach (Brigade brigade in this.Brigades)
      brigade.Delete();
     if (this.IsUsed)
     {
      Area item = this.Clone();
      item._End = DateTime.Now.Date;
      this.Update(item);
     }
     else
      Data.Tables.Areas.Delete(this);
    }

    public bool Equals(Area other)
    {
     if (
      this._Code.Equals(other._Code) &&
      this._Title.Equals(other._Title) &&
      this._Begin.Equals(other._Begin) &&
      this._End.Equals(other._End)
      )
      return true;
     else return false;
    }
    public override bool Equals(Object obj)
    {
     if (obj == null) return base.Equals(obj);
```

```
    if (!(obj is Area))
     throw new InvalidCastException("The 'obj' argument is
not a Area object.");
    else
     return Equals(obj as Area);
   }
   public override int GetHashCode()
   {
    string hash =
     this.Id.GetHashCode().ToString() +
     this.Code.GetHashCode().ToString() +
     this.Title.GetHashCode().ToString() +
     this.Begin.GetHashCode().ToString() +
     this.End.GetHashCode().ToString();
    return hash.GetHashCode();
   }
  }

  public class Position : TableRow, IEquatable<Position>
  {
   public Position(OleDbDataReader reader)
   {
    this._Id = Convert.ToInt32(reader["Id"]);
    this._WarrantyId =
Convert.ToInt32(reader["WarrantyId"]);
    this._Title = Convert.ToString(reader["Title"]);
    this._Draw = Convert.ToString(reader["Draw"]);
    this._Matherial =
Convert.ToString(reader["Matherial"]);
    this._Number = Convert.ToInt32(reader["Number"]);
    this._Mass = Convert.ToSingle(reader["Mass"]);
    this._Norm = Convert.ToSingle(reader["Norm"]);
    this._Price = Convert.ToSingle(reader["Price"]);
   }
   public Position(int WarrantyId, string Title, string
Draw, string Matherial, int Number, float Mass, float
Norm, float Price)
   {
    this._Id = 0;
    this._WarrantyId = WarrantyId;
    this._Title = Title;
    this._Draw = Draw;
    this._Matherial = Matherial;
    this._Number = Number;
    this._Mass = Mass;
    this._Norm = Norm;
    this._Price = Price;
   }

   private int _WarrantyId;
   private string _Title;
```

```
    private string _Draw;
    private string _Matherial;
    private int _Number;
    private float _Mass;
    private float _Norm;
    private float _Price;

    public int WarrantyId { get { return this._WarrantyId;
} }
    public string Title { get { return this._Title; } }
    public string Draw { get { return this._Draw; } }
    public string Matherial { get { return this._Matherial;
} }
    public int Number { get { return this._Number; } }
    public float Mass { get { return this._Mass; } }
    public float Norm { get { return this._Norm; } }
    public float Price { get { return this._Price; } }


    public void Update(Position newItem)
    {
     Data.Tables.Positions.Update(this, newItem);
    }
    public void Delete()
    {
     Data.Tables.Positions.Delete(this);
    }

    public bool Equals(Position other)
    {
     if (this._WarrantyId.Equals(other._WarrantyId) &&
      this._Title.Equals(other._Title) &&
      this._Draw.Equals(other._Draw) &&
      this._Matherial.Equals(other._Matherial) &&
      this._Number.Equals(other._Number) &&
      this._Mass.Equals(other._Mass) &&
      this._Norm.Equals(other._Norm) &&
      this._Price.Equals(other._Price))
      return true;
     else return false;
    }
    public override bool Equals(Object obj)
    {
     if (obj == null) return base.Equals(obj);

     if (!(obj is Position))
      throw new InvalidCastException("The 'obj' argument is
not a Position object.");
     else
      return Equals(obj as Position);
    }
```

```
    public override int GetHashCode()
    {
     string hash =
      this.Id.GetHashCode().ToString() +
      this.WarrantyId.GetHashCode().ToString() +
      this.Title.GetHashCode().ToString() +
      this.Draw.GetHashCode().ToString() +
      this.Matherial.GetHashCode().ToString() +
      this.Number.GetHashCode().ToString() +
      this.Mass.GetHashCode().ToString() +
      this.Norm.GetHashCode().ToString() +
      this.Price.GetHashCode().ToString();
     return hash.GetHashCode();
    }
   }

   public class Executor : TableRow, IEquatable<Executor>
   {
    public Executor(OleDbDataReader reader)
    {
     this._Id = Convert.ToInt32(reader["Id"]);
     this._WarrantyId =
Convert.ToInt32(reader["WarrantyId"]);
     this._PersonId = Convert.ToInt32(reader["PersonId"]);
     this._ProfessionId =
Convert.ToInt32(reader["ProfessionId"]);
     this._Rank = Convert.ToByte(reader["Rank"]);
    }
    public Executor(int WarrantyId, int PersonId, int
ProfessionId, byte Rank)
    {
     this._Id = 0;
     this._WarrantyId = WarrantyId;
     this._PersonId = PersonId;
     this._ProfessionId = ProfessionId;
     this._Rank = Rank;
    }

    private int _WarrantyId;
    private int _PersonId;
    private int _ProfessionId;
    private byte _Rank;

    public int WarrantyId { get { return this._WarrantyId;
} }
    public int PersonId { get { return this._PersonId; } }
    public int ProfessionId { get { return
this._ProfessionId; } }
    public byte Rank { get { return this._Rank; } }
    public Person Person
    {
```

```
        get
        {
         return Data.Tables.Persons[this._PersonId];
        }
       }
       public Profession Profession
       {
        get
        {
         return Data.Tables.Professions[this._ProfessionId];
        }
       }
       public Labors Labors
       {
        get
        {
         Labors labors = new Labors();
         var items = Data.Tables.Labors._Items.Where(r =>
r.Value.WarrantyId.Equals(this.Id)).ToArray();
         foreach (var item in items)
labors._Items.Add(item.Key, item.Value);
         return labors;
        }
       }

       public void Update(Executor newItem)
       {
        Data.Tables.Executors.Update(this, newItem);
       }
       public void Delete()
       {
        foreach (Labor labor in this.Labors)
         labor.Delete();
        Data.Tables.Executors.Delete(this);
       }

       public bool Equals(Executor other)
       {
        if (this._WarrantyId.Equals(other._WarrantyId) &&
         this._PersonId.Equals(other._PersonId) &&
         this._ProfessionId.Equals(other._ProfessionId) &&
         this._Rank.Equals(other._Rank))
         return true;
        else return false;
       }
       public override bool Equals(Object obj)
       {
        if (obj == null) return base.Equals(obj);

        if (!(obj is Executor))
         throw new InvalidCastException("The 'obj' argument is
```

```
not a Executor object.");
    else
     return Equals(obj as Executor);
   }
   public override int GetHashCode()
   {
    string hash =
     this.Id.GetHashCode().ToString() +
     this.WarrantyId.GetHashCode().ToString() +
     this.PersonId.GetHashCode().ToString() +
     this.ProfessionId.GetHashCode().ToString() +
     this.Rank.GetHashCode().ToString();
    return hash.GetHashCode();
   }
  }

  public class Labor : TableRow, IEquatable<Labor>
  {
   public Labor(OleDbDataReader reader)
   {
    this._Id = Convert.ToInt32(reader["Id"]);
    this._WarrantyId =
Convert.ToInt32(reader["WarrantyId"]);
    this._LaborDate =
Convert.ToDateTime(reader["LaborDate"]);
    this._Hours = Convert.ToSingle(reader["Hours"]);
   }
   public Labor(int WarrantyId, DateTime LaborDate, byte
Hours)
   {
    this._Id = 0;
    this._WarrantyId = WarrantyId;
    this._LaborDate = LaborDate;
    this._Hours = Hours;
   }

   private int _WarrantyId;
   private DateTime _LaborDate;
   private float _Hours;

   public int WarrantyId { get { return this._WarrantyId;
} }
   public DateTime LaborDate { get { return
this._LaborDate; } }
   public float Hours { get { return this._Hours; } }

   public void Update(Labor newItem)
   {
    Data.Tables.Labors.Update(this, newItem);
   }
   public void Delete()
```

```
    {
     Data.Tables.Labors.Delete(this);
    }

    public bool Equals(Labor other)
    {
     if (this._WarrantyId.Equals(other._WarrantyId) &&
      this._LaborDate.Equals(other._LaborDate) &&
      this._Hours.Equals(other._Hours)
      )
      return true;
     else return false;
    }
    public override bool Equals(Object obj)
    {
     if (obj == null) return base.Equals(obj);

     if (!(obj is Labor))
      throw new InvalidCastException("The 'obj' argument is
not a Labor object.");
     else
      return Equals(obj as Labor);
    }
    public override int GetHashCode()
    {
     string hash =
      this.Id.GetHashCode().ToString() +
      this.WarrantyId.GetHashCode().ToString() +
      this.LaborDate.GetHashCode().ToString() +
      this.Hours.GetHashCode().ToString();
     return hash.GetHashCode();
    }
   }

   public class Brigade : PeriodicTableRow<Brigade>,
IEquatable<Brigade>
   {
    public Brigade(OleDbDataReader reader)
    {
     this._Id = Convert.ToInt32(reader["Id"]);
     this._AreaId = Convert.ToInt32(reader["AreaId"]);
     this._Code = Convert.ToByte(reader["Code"]);
     this._Title = Convert.ToString(reader["Title"]);
     this._Begin = Convert.ToDateTime(reader["Begin"]);
     this._End = Convert.ToDateTime(reader["End"]);
    }
    public Brigade(int AreaId, byte Code, string Title)
    {
     this._Id = 0;
     this._AreaId = AreaId;
     this._Code = Code;
```

```
        this._Title = Title;
        this._Begin = DateTime.Now.Date;
        this._End = new DateTime(9000, 1, 1);
      }
      public Brigade(int AreaId, byte Code, string Title,
    DateTime Begin, DateTime End)
      {
        this._Id = 0;
        this._AreaId = AreaId;
        this._Code = Code;
        this._Title = Title;
        this._Begin = Begin;
        this._End = End;
      }

      private int _AreaId;
      private byte _Code;
      private string _Title;

      public int AreaId { get { return this._AreaId; } }
      public byte Code { get { return this._Code; } }
      public string Title { get { return this._Title; } }
      public Area Area { get { return
    Data.Tables.Areas[this.AreaId]; } }
      public BrigadePersons BrigadePersons
      {
        get
        {
          BrigadePersons brigadePersons = new BrigadePersons();
          brigadePersons.Clear();
          var items = Data.Tables.BrigadePersons._Items.Where(r
    => r.Value.BrigadeId.Equals(this.Id)).ToArray();
          foreach (var item in items)
    brigadePersons._Items.Add(item.Key, item.Value);
          return brigadePersons;
        }
      }
      public Warranties Warranties
      {
        get
        {
          Warranties warranties = new Warranties();
          var items = Data.Tables.Warranties._Items.Where(r =>
    r.Value.BrigadeId.Equals(this.Id)).ToArray();
          foreach (var item in items)
    warranties._Items.Add(item.Key, item.Value);
          return warranties;
        }
      }

      public override bool IsUsed
```

```
    {
     get
     {
      return !this.Warranties.Count().Equals(0);
     }
    }

    public override Brigade Clone()
    {
     return new Brigade(this.AreaId, this.Code, this.Title,
this.Begin, this.End);
    }

    public override void Update(Brigade newItem)
    {
     Data.Tables.Brigades.Update(this, newItem);
    }
    public override void Delete()
    {
     foreach (BrigadePerson brigadePerson in
this.BrigadePersons)
      brigadePerson.Delete();
     if (this.IsUsed)
     {
      Brigade item = this.Clone();
      item._End = DateTime.Now.Date;
      this.Update(item);
     }
     else
      Data.Tables.Brigades.Delete(this);
    }

    public bool Equals(Brigade other)
    {
     if (
      this._Code.Equals(other._Code) &&
      this._Title.Equals(other._Title) &&
      this._AreaId.Equals(other._AreaId) &&
      this._Begin.Equals(other._Begin) &&
      this._End.Equals(other._End)
      )
      return true;
     else return false;
    }
    public override bool Equals(Object obj)
    {
     if (obj == null) return base.Equals(obj);

     if (!(obj is Brigade))
      throw new InvalidCastException("The 'obj' argument is
not a Brigade object.");
```

```
      else
       return Equals(obj as Brigade);
     }
     public override int GetHashCode()
     {
      string hash =
       this.Id.GetHashCode().ToString() +
       this.AreaId.GetHashCode().ToString() +
       this.Code.GetHashCode().ToString() +
       this.Title.GetHashCode().ToString() +
       this.Begin.GetHashCode().ToString() +
       this.End.GetHashCode().ToString();
      return hash.GetHashCode();
     }
    }

    public class BrigadePerson :
   PeriodicTableRow<BrigadePerson>,
   IEquatable<BrigadePerson>
    {
     public BrigadePerson(OleDbDataReader reader)
     {
      this._Id = Convert.ToInt32(reader["Id"]);
      this._BrigadeId = Convert.ToInt32(reader["BrigadeId"]);
      this._PersonId = Convert.ToInt32(reader["PersonId"]);
      this._Begin = Convert.ToDateTime(reader["Begin"]);
      this._End = Convert.ToDateTime(reader["End"]);
     }
     public BrigadePerson(int BrigadeId, int PersonId)
     {
      this._Id = 0;
      this._BrigadeId = BrigadeId;
      this._PersonId = PersonId;
      this._Begin = DateTime.Now.Date;
      this._End = new DateTime(9000, 1, 1);
     }
     public BrigadePerson(int BrigadeId, int PersonId,
   DateTime Begin, DateTime End)
     {
      this._Id = 0;
      this._BrigadeId = BrigadeId;
      this._PersonId = PersonId;
      this._Begin = Begin;
      this._End = End;
     }

     private int _BrigadeId;
     private int _PersonId;

     public int BrigadeId { get { return this._BrigadeId; } }
     public int PersonId { get { return this._PersonId; } }
```

```
    public Brigade Brigade
    {
     get
     {
      return Data.Tables.Brigades[this.BrigadeId];
     }
    }
    public Person Person
    {
     get
     {
      return Data.Tables.Persons[this._PersonId];
     }
    }

    public override bool IsUsed
    {
     get
     {
      return (!this.Brigade.Warranties.Count().Equals(0))
||  (!this.Person.Executors.Count().Equals(0));

     }
    }

    public override BrigadePerson Clone()
    {
     return new BrigadePerson(this.BrigadeId,
this.PersonId, this.Begin, this.End);
    }

    public override void Update(BrigadePerson newItem)
    {
     Data.Tables.BrigadePersons.Update(this, newItem);
    }
    public override void Delete()
    {
     if (this.IsUsed)
     {
      BrigadePerson item = this.Clone();
      item._End = DateTime.Now.Date;
      this.Update(item);
     }
     else
      Data.Tables.BrigadePersons.Delete(this);
    }

    public bool Equals(BrigadePerson other)
    {
     if (this._BrigadeId.Equals(other._BrigadeId) &&
      this._PersonId.Equals(other._PersonId) &&
```

```
        this._Begin.Equals(other._Begin) &&
        this._End.Equals(other._End))
        return true;
      else return false;
    }
    public override bool Equals(Object obj)
    {
      if (obj == null) return base.Equals(obj);

      if (!(obj is BrigadePerson))
        throw new InvalidCastException("The 'obj' argument is
not a BrigadePerson object.");
      else
        return Equals(obj as BrigadePerson);
    }
    public override int GetHashCode()
    {
      string hash =
        this.Id.GetHashCode().ToString() +
        this.BrigadeId.GetHashCode().ToString() +
        this.PersonId.GetHashCode().ToString() +
        this.Begin.GetHashCode().ToString() +
        this.End.GetHashCode().ToString();
      return hash.GetHashCode();
    }
  }

  public class PersonProfession :
PeriodicTableRow<PersonProfession>,
IEquatable<PersonProfession>
  {
    public PersonProfession(OleDbDataReader reader)
    {
      this._Id = Convert.ToInt32(reader["Id"]);
      this._PersonId = Convert.ToInt32(reader["PersonId"]);
      this._ProfessionId =
Convert.ToInt32(reader["ProfessionId"]);
      this._Rank = Convert.ToByte(reader["Rank"]);
      this._Begin = Convert.ToDateTime(reader["Begin"]);
      this._End = Convert.ToDateTime(reader["End"]);
    }
    public PersonProfession(int PersonId, int ProfessionId,
byte Rank)
    {
      this._Id = 0;
      this._PersonId = PersonId;
      this._ProfessionId = ProfessionId;
      this._Rank = Rank;
      this._Begin = Begin;
      this._End = End;
    }
```

```
    public PersonProfession(int PersonId, int ProfessionId,
  byte Rank, DateTime Begin, DateTime End)
    {
     this._Id = 0;
     this._PersonId = PersonId;
     this._ProfessionId = ProfessionId;
     this._Rank = Rank;
     this._Begin = Begin;
     this._End = End;
    }

    private int _PersonId;
    private int _ProfessionId;
    private byte _Rank;

    public int PersonId { get { return this._PersonId; } }
    public int ProfessionId { get { return
  this._ProfessionId; } }
    public byte Rank { get { return this._Rank; } }
    public Person Person
    {
     get
     {
      return Data.Tables.Persons[this.PersonId];
     }
    }
    public Profession Profession
    {
     get
     {
      return Data.Tables.Professions[this._ProfessionId];
     }
    }

    public override bool IsUsed
    {
     get
     {
      return this.Person.IsUsed;
     }
    }

    public override PersonProfession Clone()
    {
     return new PersonProfession(this.PersonId,
  this.ProfessionId, this.Rank, this.Begin, this.End);
    }

    public override void Update(PersonProfession newItem)
    {
     Data.Tables.PersonProfessions.Update(this, newItem);
```

```
    }
    public override void Delete()
    {
     if (this.IsUsed)
     {
      PersonProfession item = this.Clone();
      item._End = DateTime.Now.Date;
      this.Update(item);
     }
     else
      Data.Tables.PersonProfessions.Delete(this);
    }

    public bool Equals(PersonProfession other)
    {
     if (this._PersonId.Equals(other._PersonId) &&
      this._ProfessionId.Equals(other._ProfessionId) &&
      this._Rank.Equals(other._Rank) &&
      this._Begin.Equals(other._Begin) &&
      this._End.Equals(other._End))
      return true;
     else return false;
    }
    public override bool Equals(Object obj)
    {
     if (obj == null) return base.Equals(obj);

     if (!(obj is PersonProfession))
      throw new InvalidCastException("The 'obj' argument is
not a PersonProfession object.");
     else
      return Equals(obj as PersonProfession);
    }
    public override int GetHashCode()
    {
     string hash =
      this.Id.GetHashCode().ToString() +
      this.PersonId.GetHashCode().ToString() +
      this.ProfessionId.GetHashCode().ToString() +
      this.Begin.GetHashCode().ToString() +
      this.End.GetHashCode().ToString();
     return hash.GetHashCode();
    }
   }
  }
```

Файл Project\Databases\Tables.cs:

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Data.OleDb;
using System.Linq;

namespace Project.Databases
{
 public abstract class Table<T> : IEnumerable<T>
  where T : ITableRow
 {
  protected internal Dictionary<int, T> _Items = new
Dictionary<int, T>();

  protected internal OleDbDataReader reader;

  public bool IsEmpty
  {
   get { return this._Items.Count.Equals(0); }
  }

  public T this[int id]
  {
   get { return this._Items[id]; }
  }

  public int GetIdByValue(T item)
  {
   foreach (var i in this._Items)
    if (i.Value.Equals(item)) return i.Key;
   return 0;
  }

  public int NextId
  {
   get
   {
    int id;
    if (this._Items.Keys.Count != 0) id =
this._Items.Keys.Max() + 1;
    else id = 1;
    return id;
   }
  }

  internal void Clear()
  {
   this._Items.Clear();
  }

  IEnumerator IEnumerable.GetEnumerator()
```

```
    {
     return this._Items.GetEnumerator();
    }
    public IEnumerator<T> GetEnumerator()
    {
     return this._Items.Values.GetEnumerator();
    }


    public string TableName;

    public OleDbCommand Command = new OleDbCommand();

    public List<OleDbParameter> Parameters = new
List<OleDbParameter>();

    public int Insert(T item)
    {
     int id = this.NextId;
     item.Id = id;
     OleDbParameter pId = new OleDbParameter("Id",
OleDbType.Integer);
     pId.Value = id;
     Command.Parameters.Add(pId);
     string comParams = "", comVals = "";
     comParams += "([Id], ";
     comVals += "(?, ";
     foreach (OleDbParameter parameter in Parameters)
     {
      comParams += String.Format("[{0}], ",
parameter.ParameterName);
      comVals += "?, ";
     }
     comParams = comParams.Remove(comParams.Length - 2);
     comVals = comVals.Remove(comVals.Length - 2);
     comParams += ")";
     comVals += ")";

     string comText = String.Format("INSERT INTO {0} {1}
VALUES {2};", TableName, comParams, comVals);
     foreach (OleDbParameter parameter in Parameters)
     {
      parameter.Value =
item.GetType().GetProperty(parameter.ParameterName).GetVa
lue(item, null);
      Command.Parameters.Add(parameter);
     }

     Command.CommandText = comText;
     Command.Connection = Data.Connection;
     Command.Connection.Open();
```

```
    Command.ExecuteNonQuery();
    Command.Parameters.Clear();
    Command.Connection.Close();

    this._Items.Add(id, item);
    return id;
   }

   public void Update(T item, T newItem)
   {
    newItem.Id = item.Id;
    int id = GetIdByValue(item);
    OleDbParameter pId = new OleDbParameter("Id",
OleDbType.Integer);
    pId.Value = id;


    string comItems = "";

    //comItems += "(";
    foreach (OleDbParameter parameter in Parameters)
    {
     comItems += String.Format(" [{0}] = ?, ",
parameter.ParameterName);
     parameter.Value =
newItem.GetType().GetProperty(parameter.ParameterName).Ge
tValue(newItem, null);
     Command.Parameters.Add(parameter);
    }
    comItems = comItems.Remove(comItems.Length - 2);
    //comItems += ")";

    Command.Parameters.Add(pId);


    Command.CommandText = String.Format("UPDATE {0} SET
{1} WHERE [Id] = ?;", TableName, comItems);
    Command.Connection = Data.Connection;
    Command.Connection.Open();
    Command.ExecuteNonQuery();
    Command.Parameters.Clear();
    Command.Connection.Close();

    this._Items[id] = newItem;
   }

   public void Delete(T item)
   {
    int id = GetIdByValue(item);
    if (id != 0)
    {
```

```
    OleDbParameter pId = new OleDbParameter("Id",
OleDbType.Integer);
    pId.Value = id;

    Command.Connection = Data.Connection;
    Command.CommandText = String.Format("DELETE * FROM
[{0}] WHERE [Id] = ?;", TableName);
    Command.Parameters.Add(pId);

    Command.Connection.Open();
    Command.ExecuteNonQuery();
    Command.Parameters.Clear();
    Command.Connection.Close();

    this._Items.Remove(id);
   }
  }

  public void Load(OleDbConnection connection)
  {
   Command.Connection = connection;
   Command.CommandText = String.Format("SELECT * FROM
{0};", TableName);
   OleDbDataReader reader = Command.ExecuteReader();
   while (reader.Read())
   {
    int id = Convert.ToInt32(reader["Id"]);
    T item = (T)Activator.CreateInstance(typeof(T), new
object[] { reader });

    this._Items.Add(id, item);
   }
  }
 }

 public class Persons : Table<Person>
 {
  public Persons()
  {
   TableName = "Persons";
   Parameters.Add(new OleDbParameter("Code",
OleDbType.SmallInt));
   Parameters.Add(new OleDbParameter("FirstName",
OleDbType.VarWChar, sizeof(char) * 31));
   Parameters.Add(new OleDbParameter("MiddleName",
OleDbType.VarWChar, sizeof(char) * 31));
   Parameters.Add(new OleDbParameter("LastName",
OleDbType.VarWChar, sizeof(char) * 31));
   Parameters.Add(new OleDbParameter("Begin",
OleDbType.DBDate));
   Parameters.Add(new OleDbParameter("End",
```

```
OleDbType.DBDate));
  }

  public Persons Active
  {
   get
   {
    Persons persons = new Persons();
    var _Items = Data.Tables.Persons._Items
     .Where(r => r.Value.Begin.CompareTo(DateTime.Now) <=
0 && r.Value.End.CompareTo(DateTime.Now) > 0)
     .ToArray();
    foreach (var item in _Items)
     persons._Items.Add(item.Key, item.Value);
    return persons;
   }
  }

  internal void Optimize()
  {
   Person[] persons = Data.Tables.Persons
    .Except(Data.Tables.Persons.Active)
    .Where(r => r.Executors.Count().Equals(0))
    .ToArray();

   foreach (Person person in persons)
    person.Delete();
  }
 }

 public class Professions : Table<Profession>
 {
  public Professions()
  {
   TableName = "Professions";
   Parameters.Add(new OleDbParameter("Code",
OleDbType.SmallInt));
   Parameters.Add(new OleDbParameter("Title",
OleDbType.VarWChar, sizeof(char) * 31));
   Parameters.Add(new OleDbParameter("Rank1",
OleDbType.Single));
   Parameters.Add(new OleDbParameter("Rank2",
OleDbType.Single));
   Parameters.Add(new OleDbParameter("Rank3",
OleDbType.Single));
   Parameters.Add(new OleDbParameter("Rank4",
OleDbType.Single));
   Parameters.Add(new OleDbParameter("Rank5",
OleDbType.Single));
   Parameters.Add(new OleDbParameter("Rank6",
OleDbType.Single));
```

```
    Parameters.Add(new OleDbParameter("Begin",
OleDbType.DBDate));
    Parameters.Add(new OleDbParameter("End",
OleDbType.DBDate));
   }

   public Professions Active
   {
    get
    {
     Professions professions = new Professions();
     var _Items = Data.Tables.Professions._Items
      .Where(r => r.Value.Begin.CompareTo(DateTime.Now) <=
0 && r.Value.End.CompareTo(DateTime.Now) > 0)
      .ToArray();
     foreach (var item in _Items)
      professions._Items.Add(item.Key, item.Value);
     return professions;
    }
   }

   internal void Optimize()
   {
    Profession[] professions = Data.Tables.Professions
     .Except(Data.Tables.Professions.Active)
     .ToArray();

    foreach (Profession profession in professions)
     profession.Delete();
   }
  }

  public class Warranties : Table<Warranty>
  {
   public Warranties()
   {
    TableName = "Warranties";
    Parameters.Add(new OleDbParameter("Customer",
OleDbType.VarWChar, sizeof(char) * 50));
    Parameters.Add(new OleDbParameter("Order",
OleDbType.SmallInt));
    Parameters.Add(new OleDbParameter("Percent",
OleDbType.Single));
    Parameters.Add(new OleDbParameter("WarrantyDate",
OleDbType.DBDate));
    Parameters.Add(new OleDbParameter("AreaId",
OleDbType.Integer));
    Parameters.Add(new OleDbParameter("BrigadeId",
OleDbType.Integer));
   }
  }
```

```
 public class Areas : Table<Area>
 {
  public Areas()
  {
   TableName = "Areas";
   Parameters.Add(new OleDbParameter("Code",
OleDbType.SmallInt));
   Parameters.Add(new OleDbParameter("Title",
OleDbType.VarWChar, sizeof(char) * 50));
   Parameters.Add(new OleDbParameter("Begin",
OleDbType.DBDate));
   Parameters.Add(new OleDbParameter("End",
OleDbType.DBDate));
  }

  public Areas Active
  {
   get
   {
    Areas areas = new Areas();
    var temp = Data.Tables.Areas._Items.Where(r =>
r.Value.Begin.CompareTo(DateTime.Now) <= 0 &&
r.Value.End.CompareTo(DateTime.Now) > 0).ToArray();
    foreach (var area in temp) areas._Items.Add(area.Key,
area.Value);
    return areas;
   }
  }

  internal void Optimize()
  {
   Area[] areas = Data.Tables.Areas
    .Except(Data.Tables.Areas.Active)
    .Where(r => r.Warranties.Count().Equals(0))
    .ToArray();

   foreach (Area area in areas)
    Data.Tables.Areas.Delete(area);
  }
 }

 public class Positions : Table<Position>
 {
  public Positions()
  {
   TableName = "Positions";
   Parameters.Add(new OleDbParameter("WarrantyId",
OleDbType.Integer));
   Parameters.Add(new OleDbParameter("Title",
OleDbType.VarWChar, sizeof(char) * 50));
```

```
    Parameters.Add(new OleDbParameter("Draw",
OleDbType.VarWChar, sizeof(char) * 50));
    Parameters.Add(new OleDbParameter("Matherial",
OleDbType.VarWChar, sizeof(char) * 50));
    Parameters.Add(new OleDbParameter("Number",
OleDbType.SmallInt));
    Parameters.Add(new OleDbParameter("Mass",
OleDbType.Single));
    Parameters.Add(new OleDbParameter("Norm",
OleDbType.Single));
    Parameters.Add(new OleDbParameter("Price",
OleDbType.Single));
   }
  }

  public class Executors : Table<Executor>
  {
   public Executors()
   {
    TableName = "Executors";
    Parameters.Add(new OleDbParameter("WarrantyId",
OleDbType.Integer));
    Parameters.Add(new OleDbParameter("PersonId",
OleDbType.Integer));
    Parameters.Add(new OleDbParameter("ProfessionId",
OleDbType.Integer));
    Parameters.Add(new OleDbParameter("Rank",
OleDbType.UnsignedTinyInt));
   }
  }

  public class Labors : Table<Labor>
  {
   public Labors()
   {
    TableName = "Labors";
    Parameters.Add(new OleDbParameter("WarrantyId",
OleDbType.Integer));
    Parameters.Add(new OleDbParameter("LaborDate",
OleDbType.DBDate));
    Parameters.Add(new OleDbParameter("Hours",
OleDbType.Single));
   }
  }

  public class Brigades : Table<Brigade>
  {
   public Brigades()
   {
    TableName = "Brigades";
    Parameters.Add(new OleDbParameter("AreaId",
```

```
OleDbType.Integer));
    Parameters.Add(new OleDbParameter("Code",
OleDbType.UnsignedTinyInt));
    Parameters.Add(new OleDbParameter("Title",
OleDbType.VarWChar, sizeof(char) * 63));
    Parameters.Add(new OleDbParameter("Begin",
OleDbType.DBDate));
    Parameters.Add(new OleDbParameter("End",
OleDbType.DBDate));
   }

   public Brigades Active
   {
    get
    {
     Brigades brigades = new Brigades();
     var _Items = Data.Tables.Brigades._Items.Where(r =>
r.Value.Begin.CompareTo(DateTime.Now) <= 0 &&
r.Value.End.CompareTo(DateTime.Now) > 0).ToArray();
     foreach (var item in _Items)
brigades._Items.Add(item.Key, item.Value);
     return brigades;
    }
   }

   internal void Optimize()
   {
    Brigade[] brigades = Data.Tables.Brigades
     .Except(Data.Tables.Brigades.Active)
     .Where(r => r.Warranties.Count().Equals(0))
     .ToArray();

    foreach (Brigade brigade in brigades)
     Data.Tables.Brigades.Delete(brigade);
   }
  }

  public class BrigadePersons : Table<BrigadePerson>
  {
   public BrigadePersons()
   {
    TableName = "BrigadePersons";

    Parameters.Add(new OleDbParameter("BrigadeId",
OleDbType.Integer));
    Parameters.Add(new OleDbParameter("PersonId",
OleDbType.Integer));
    Parameters.Add(new OleDbParameter("Begin",
OleDbType.DBDate));
    Parameters.Add(new OleDbParameter("End",
OleDbType.DBDate));
```

```
    }

    public BrigadePersons Active
    {
     get
     {
      BrigadePersons brigadePersons = new BrigadePersons();
      var _Items =
Data.Tables.BrigadePersons._Items.Where(r =>
r.Value.Begin.CompareTo(DateTime.Now) <= 0 &&
r.Value.End.CompareTo(DateTime.Now) > 0).ToArray();
      foreach (var item in _Items)
brigadePersons._Items.Add(item.Key, item.Value);
      return brigadePersons;
     }
    }
   }

   public class PersonProfessions : Table<PersonProfession>
   {
    public PersonProfessions()
    {
     TableName = "PersonProfessions";

     Parameters.Add(new OleDbParameter("PersonId",
OleDbType.Integer));
     Parameters.Add(new OleDbParameter("ProfessionId",
OleDbType.Integer));
     Parameters.Add(new OleDbParameter("Rank",
OleDbType.UnsignedTinyInt));
     Parameters.Add(new OleDbParameter("Begin",
OleDbType.DBDate));
     Parameters.Add(new OleDbParameter("End",
OleDbType.DBDate));
    }

    public PersonProfessions Active
    {
     get
     {
      PersonProfessions personProfessions = new
PersonProfessions();
      var _Items =
Data.Tables.PersonProfessions._Items.Where(r =>
r.Value.Begin.CompareTo(DateTime.Now) <= 0 &&
r.Value.End.CompareTo(DateTime.Now) > 0).ToArray();
      foreach (var item in _Items)
personProfessions._Items.Add(item.Key, item.Value);
      return personProfessions;
     }
    }
```

```
    }

    public class Tables
    {
     public Professions Professions = new Professions();
     public Persons Persons = new Persons();
     public PersonProfessions PersonProfessions = new
  PersonProfessions();
     public Areas Areas = new Areas();
     public Brigades Brigades = new Brigades();
     public BrigadePersons BrigadePersons = new
  BrigadePersons();
     public Warranties Warranties = new Warranties();
     public Positions Positions = new Positions();
     public Executors Executors = new Executors();
     public Labors Labors = new Labors();

     public bool IsEmpty
     {
      get
      {
       if (
        Professions.IsEmpty &&
        Persons.IsEmpty &&
        PersonProfessions.IsEmpty &&
        Areas.IsEmpty &&
        Brigades.IsEmpty &&
        BrigadePersons.IsEmpty &&
        Warranties.IsEmpty &&
        Positions.IsEmpty &&
        Executors.IsEmpty &&
        Labors.IsEmpty)
        return true;
       else return false;
      }
     }

     internal void Load(OleDbConnection connection)
     {
      Professions.Load(connection);
      Persons.Load(connection);
      PersonProfessions.Load(connection);
      Areas.Load(connection);
      Brigades.Load(connection);
      BrigadePersons.Load(connection);
      Warranties.Load(connection);
      Positions.Load(connection);
      Executors.Load(connection);
      Labors.Load(connection);
     }
```

```
    internal void Clear()
    {
     Professions.Clear();
     Persons.Clear();
     PersonProfessions.Clear();
     Areas.Clear();
     Brigades.Clear();
     BrigadePersons.Clear();
     Warranties.Clear();
     Positions.Clear();
     Executors.Clear();
     Labors.Clear();
    }

    public void Optimize()
    {
     Areas.Optimize();
     Brigades.Optimize();
     Persons.Optimize();
     Professions.Optimize();
    }
   }
}
```

Файл Project\Databases\Data.cs:

```
using System;
using System.Collections.Generic;
using System.Data.OleDb;
using System.IO;
using System.Linq;
using System.Text.RegularExpressions;
using JRO;

namespace Project.Databases
{
 public static class Data
 {
  private static DirectoryInfo _DataDirectory = new
DirectoryInfo(@".\Databases");

  private static OleDbConnectionStringBuilder
_ConnectionStringBuilder =
   new
OleDbConnectionStringBuilder("Provider=Microsoft.Jet.OLED
B.4.0;");

  public static bool IsEmpty
  {
```

```
    get { return Tables.IsEmpty; }
   }

   public static Tables Tables = new Tables();

   internal static OleDbConnection Connection = new
OleDbConnection();

   private static List<string> _AvailableDatabases = new
List<string>();

   public static List<string> AvailableDatabases
   {
    get { return _AvailableDatabases; }
   }

   /// <summary>
   /// Оптимизирует базу данных текущего года
   /// посредством удаления устаревших данных
   /// и последующего сжатия файла базы данных.
   /// </summary>
   public static void Optimize()
   {
    Load(Connection);
    Tables.Optimize();
    Clear();

    // Сжатие базы данных
    string tempPath =
Path.Combine(_DataDirectory.FullName, "temp.mdb");
    _ConnectionStringBuilder.DataSource = tempPath;
    JRO.JetEngine jro = new JetEngine();
    jro.CompactDatabase(Connection.ConnectionString,
_ConnectionStringBuilder.ConnectionString);
    File.Delete(Connection.DataSource);
    File.Copy(tempPath, Connection.DataSource);
    File.Delete(tempPath);
   }

   private static void ClearDocs()
   {
    List<Warranty> docs = Tables.Warranties.ToList();
    foreach (Warranty doc in docs) doc.Delete();
   }

   private static void Init()
   {
    int currentYear = DateTime.Now.Year;

    // Получить список путей к каждому из *.mdb файлов в
каталоге данных
```

```
    FileInfo[] files = _DataDirectory.GetFiles("*.mdb");

    Regex reg = new Regex(@"^\d\d\d\d.mdb$");

    foreach (FileInfo file in files)
    {
     if (reg.IsMatch(file.Name))
     {
      string fName = file.Name.Remove(file.Name.Length -
4);
      int year = Convert.ToInt32(fName);
      if (year <= currentYear)
_AvailableDatabases.Add(fName);
     }
    }

    if (_AvailableDatabases.Count != 0)
    {
     int maxYear = _AvailableDatabases.Select(r =>
Convert.ToInt32(r)).Max();
     if (maxYear != currentYear)
     {
      string ifPath =
Path.Combine(_DataDirectory.FullName, maxYear.ToString()
+ ".mdb");
      string ofPath =
Path.Combine(_DataDirectory.FullName,
currentYear.ToString() + ".mdb");

      File.Copy(ifPath, ofPath);
      _AvailableDatabases.Add(currentYear.ToString());
      _ConnectionStringBuilder.DataSource = ofPath;
      Connection.ConnectionString =
_ConnectionStringBuilder.ConnectionString;

      ClearDocs();
      Optimize();
     }
     else
     {
      string fPath = Path.Combine(_DataDirectory.FullName,
currentYear.ToString() + ".mdb");
      _ConnectionStringBuilder.DataSource = fPath;
      Connection.ConnectionString =
_ConnectionStringBuilder.ConnectionString;
     }
    }
    else
    {
     string ifPath = Path.Combine(_DataDirectory.FullName,
"template.mdb");
```

```
    string ofPath = Path.Combine(_DataDirectory.FullName,
currentYear.ToString() + ".mdb");
    File.Copy(ifPath, ofPath);
    _AvailableDatabases.Add(currentYear.ToString());
    _ConnectionStringBuilder.DataSource = ofPath;
    Connection.ConnectionString =
_ConnectionStringBuilder.ConnectionString;
   }

  }

  private static void Load(OleDbConnection connection)
  {
   Connection.Open();
   Tables.Load(connection);
   Connection.Close();
  }

  public static void Clear()
  {
   Tables.Clear();
  }

  static Data()
  {
   Init();

   Load(Connection);
  }
 }
}
```

Файл Project\Controls\BrigadePersonsControl.cs:

```
using System.Collections.Generic;
using System.ComponentModel;
using System.Linq;
using System.Windows.Forms;
using Project.Databases;
using Project.Forms.Tables;

namespace Project.Controls
{
 public partial class BrigadePersonsControl : TableControl
 {
  private DataGridViewColumn _Id = new
DataGridViewTextBoxColumn();
  private DataGridViewColumn _PersonCode = new
DataGridViewTextBoxColumn();
```

```
  private DataGridViewColumn _PersonFirstName = new
DataGridViewTextBoxColumn();
  private DataGridViewColumn _PersonMiddleName = new
DataGridViewTextBoxColumn();
  private DataGridViewColumn _PersonLastName = new
DataGridViewTextBoxColumn();

  private int _BrigadeId = 0;
  private List<BrigadePerson> _Deletions = new
List<BrigadePerson>();


[DesignerSerializationVisibility(DesignerSerializationVis
ibility.Hidden)]
  public int BrigadeId
  {
   get { return this._BrigadeId; }
   set
   {
    this._BrigadeId = value;
    Init();
   }
  }


[DesignerSerializationVisibility(DesignerSerializationVis
ibility.Hidden)]
  public List<BrigadePerson> Deletions
  {
   get { return this._Deletions; }
   set
   {
    this._Deletions = value;
    Init();
   }
  }

  public BrigadePersonsControl()
  {
   InitializeComponent();

   this.gbFilter.Text = "Фильтр персонала";
   this.gbItems.Text = "Персонал подразделения";
   this.gbOperations.Text = "Операции с персоналом";

   this._Id.Name = "Id";
   this._Id.Visible = false;

   this._PersonCode.Name = "PersonCode";
   this._PersonCode.HeaderText = "Таб. №";
```

```
        this._PersonLastName.Name = "PersonLastName";
        this._PersonLastName.HeaderText = "Фамилия";

        this._PersonFirstName.Name = "PersonFirstName";
        this._PersonFirstName.HeaderText = "Имя";

        this._PersonMiddleName.Name = "PersonMiddleName";
        this._PersonMiddleName.HeaderText = "Отчество";
        this._PersonMiddleName.AutoSizeMode =
    DataGridViewAutoSizeColumnMode.Fill;


        this.dgvItems.Columns.Add(this._Id);
        this.dgvItems.Columns.Add(this._PersonCode);
        this.dgvItems.Columns.Add(this._PersonLastName);
        this.dgvItems.Columns.Add(this._PersonFirstName);
        this.dgvItems.Columns.Add(this._PersonMiddleName);



    this.dgvFilter.Columns.Add((DataGridViewColumn)this._Pers
    onCode.Clone());

    this.dgvFilter.Columns.Add((DataGridViewColumn)this._Pers
    onLastName.Clone());

    this.dgvFilter.Columns.Add((DataGridViewColumn)this._Pers
    onFirstName.Clone());

    this.dgvFilter.Columns.Add((DataGridViewColumn)this._Pers
    onMiddleName.Clone());
        this.dgvFilter.Rows.Add();

        this.bNew.Text = "Добавить";
        this.bDelete.Text = "Исключить";
        this.bDelete.Location = this.bEdit.Location;
        this.bEdit.Visible = false;

        Init();
      }


    [DesignerSerializationVisibility(DesignerSerializationVis
    ibility.Hidden)]
      public override void New()
      {
        if (this.BrigadeId != 0)
        {
          Brigade brigade = Data.Tables.Brigades[BrigadeId];
          frmPersons form = new frmPersons();
```

```
    form.ctrlPersons.Deletions = (from person in
Data.Tables.Persons.Active
                from brigadePerson in
Data.Tables.BrigadePersons.Active
                where person.Equals(brigadePerson.Person)
                select person).ToList();

    form.CatalogMode = CatalogMode.Select;
    form.ShowDialog();
    if (form.SelectedPersonId != 0)
    {
     BrigadePerson brigadePerson = new
BrigadePerson(this.BrigadeId, form.SelectedPersonId);
     Data.Tables.BrigadePersons.Insert(brigadePerson);
    }
    Init();
   }
  }


[DesignerSerializationVisibility(DesignerSerializationVis
ibility.Hidden)]
  public override void Delete()
  {
   if (this.CurrentId != 0)
   {
    if (MessageBox.Show("Вы действительно хотите удалить
запись?", "Удаление записи",
MessageBoxButtons.OKCancel).Equals(DialogResult.OK))
    {
     BrigadePerson brigadePerson =
Data.Tables.BrigadePersons[this.CurrentId];
     brigadePerson.Delete();
     Init();
    }
   }
  }


[DesignerSerializationVisibility(DesignerSerializationVis
ibility.Hidden)]
  public override void Init()
  {
   this.dgvItems.DataSource = (from brigadePerson in
Data.Tables.BrigadePersons.Active.Except(this.Deletions.A
sEnumerable())
           where
           (brigadePerson.BrigadeId == this.BrigadeId) &&

brigadePerson.Person.Code.ToString().Contains(this.GetFil
ter("PersonCode")) &&
```

```
brigadePerson.Person.FirstName.ToUpper().Contains(this.Ge
tFilter("PersonFirstName").ToUpper()) &&

brigadePerson.Person.MiddleName.ToUpper().Contains(this.G
etFilter("PersonMiddleName").ToUpper()) &&

brigadePerson.Person.LastName.ToUpper().Contains(this.Get
Filter("PersonLastName").ToUpper())
           select new
           {
            Id = brigadePerson.Id,
            PersonCode = brigadePerson.Person.Code,
            PersonFirstName =
brigadePerson.Person.FirstName,
            PersonMiddleName =
brigadePerson.Person.MiddleName,
            PersonLastName = brigadePerson.Person.LastName
           }).ToList();

   foreach (DataGridViewColumn column in
this.dgvItems.Columns)
    {
     column.DataPropertyName = column.Name;
    }
   }
  }
}
```

Файл Project\Controls\CalendarColumn.cs:

```
using System;
using System.Windows.Forms;

namespace Project.Controls
{

 public class CalendarColumn : DataGridViewColumn
 {
  public CalendarColumn()
   : base(new CalendarCell())
  {
  }

  public override DataGridViewCell CellTemplate
  {
   get
   {
    return base.CellTemplate;
```

```
    }
    set
    {
     if (value != null &&
!value.GetType().IsAssignableFrom(typeof(CalendarCell)))
     {
      throw new InvalidCastException("Must be a
CalendarCell");
     }
     base.CellTemplate = value;
    }
   }
  }

  public class CalendarCell : DataGridViewTextBoxCell
  {

   public CalendarCell()
    : base()
   {
    this.Style.Format = "d";
   }

   public override void InitializeEditingControl(int
rowIndex, object
    initialFormattedValue, DataGridViewCellStyle
dataGridViewCellStyle)
   {
    base.InitializeEditingControl(rowIndex,
initialFormattedValue,
     dataGridViewCellStyle);
    CalendarEditingControl ctl =
     DataGridView.EditingControl as CalendarEditingControl;
   }

   public override Type EditType
   {
    get
    {
     return typeof(CalendarEditingControl);
    }
   }

   public override Type ValueType
   {
    get
    {
     return typeof(DateTime);
    }
   }
```

```
    public override object DefaultNewRowValue
    {
     get
     {
      return null;
     }
    }
   }

   class CalendarEditingControl : DateTimePicker,
  IDataGridViewEditingControl
   {
    DataGridView dataGridView;
    private bool valueChanged = false;
    int rowIndex;

    public CalendarEditingControl()
    {
     this.Format = DateTimePickerFormat.Short;
    }

    public object EditingControlFormattedValue
    {
     get
     {
      return this.Value.ToShortDateString();
     }
     set
     {
      if (value is String)
      {
       this.Value = DateTime.Parse((String)value);
      }
     }
    }

    public object GetEditingControlFormattedValue(
     DataGridViewDataErrorContexts context)
    {
     return EditingControlFormattedValue;
    }

    public void ApplyCellStyleToEditingControl(
     DataGridViewCellStyle dataGridViewCellStyle)
    {
     this.Font = dataGridViewCellStyle.Font;
     this.CalendarForeColor =
  dataGridViewCellStyle.ForeColor;
     this.CalendarMonthBackground =
  dataGridViewCellStyle.BackColor;
```

```
    }

    public int EditingControlRowIndex
    {
     get
     {
      return rowIndex;
     }
     set
     {
      rowIndex = value;
     }
    }

    public bool EditingControlWantsInputKey(
     Keys key, bool dataGridViewWantsInputKey)
    {
     switch (key & Keys.KeyCode)
     {
      case Keys.Left:
      case Keys.Up:
      case Keys.Down:
      case Keys.Right:
      case Keys.Home:
      case Keys.End:
      case Keys.PageDown:
      case Keys.PageUp:
       return true;
      default:
       return false;
     }
    }

    public void PrepareEditingControlForEdit(bool
  selectAll) { }

    public bool RepositionEditingControlOnValueChange
    {
     get { return false; }
    }

    public DataGridView EditingControlDataGridView
    {
     get { return dataGridView; }
     set { dataGridView = value; }
    }

    public bool EditingControlValueChanged
    {
     get { return valueChanged; }
     set { valueChanged = value; }
```

```
    }

    public Cursor EditingPanelCursor
    {
     get { return base.Cursor; }
    }

    protected override void OnValueChanged(EventArgs
eventargs)
    {
     valueChanged = true;

this.EditingControlDataGridView.NotifyCurrentCellDirty(tr
ue);
     base.OnValueChanged(eventargs);
    }
   }
}
```

Файл Project\Controls\ExecutorsControl.cs:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Linq;
using System.Windows.Forms;
using Project.Databases;
using Project.Forms.Tables;

namespace Project.Controls
{
 public partial class ExecutorsControl : UserControl
 {
  private int _BrigadeId = 0;


[DesignerSerializationVisibility(DesignerSerializationVis
ibility.Hidden)]
    public int BrigadeId
    {
     get { return this._BrigadeId; }
     set
     {
      this._BrigadeId = value;
     }
    }

    public ExecutorsControl()
    {
```

```
    InitializeComponent();
   }

   private void dgvExecutors_CellContentClick(object
sender, DataGridViewCellEventArgs e)
   {
    if (e.ColumnIndex >= 0 && e.RowIndex >= 0)
    {
     DataGridView grid = (DataGridView)sender;
     DataGridViewColumn column =
grid.Columns[e.ColumnIndex];
     DataGridViewRow row = grid.Rows[e.RowIndex];
     DataGridViewCell cell = grid[e.ColumnIndex,
e.RowIndex];
     switch (column.Name)
     {
      case "PersonCode":
       frmBrigadePersons fbp = new frmBrigadePersons();
       fbp.CatalogMode = CatalogMode.Select;
       fbp.ctrlBrigadePersons.BrigadeId = this.BrigadeId;

       List<BrigadePerson> deletions = new
List<BrigadePerson>();
       List<short> deletionsCodes = new List<short>();

       foreach (DataGridViewRow r in this.dgvItems.Rows)
       {

deletionsCodes.Add(Convert.ToInt16(r.Cells["PersonCode"].
Value));
       }
       foreach (short deletionCode in deletionsCodes)
       {

deletions.AddRange(Data.Tables.BrigadePersons.Active.Wher
e(r => r.Person.Code == deletionCode).AsEnumerable());
       }
       fbp.ctrlBrigadePersons.Deletions = deletions;
       fbp.ShowDialog();

       if (fbp.ctrlBrigadePersons.SelectedId != 0)
       {
        BrigadePerson brigadePerson =
Data.Tables.BrigadePersons[fbp.ctrlBrigadePersons.Selecte
dId];
        DataGridViewButtonCell button =
(DataGridViewButtonCell)cell;
        button.Value =
brigadePerson.Person.Code.ToString();
        button.Tag = brigadePerson.Person;
        row.Cells["PersonLastName"].Value =
```

```
brigadePerson.Person.LastName;
            row.Cells["PersonFirstName"].Value =
brigadePerson.Person.FirstName;
            row.Cells["PersonMiddleName"].Value =
brigadePerson.Person.MiddleName;
          }
          break;
        case "ProfessionCode":
          if (row.Cells["PersonCode"].Value != null)
          {
            frmPersonProfessions ppf = new
frmPersonProfessions();
            Person p =
((DataGridViewButtonCell)row.Cells["PersonCode"]).Tag as
Person;
            ppf.ctrlPersonProfessions.PersonId = p.Id;
            ppf.ShowDialog();
            if (ppf.SelectedId != 0)
            {
              PersonProfession pp =
Data.Tables.PersonProfessions[ppf.SelectedId];
              DataGridViewButtonCell b =
(DataGridViewButtonCell)cell;
              b.Tag = pp;
              b.Value = pp.Profession.Code.ToString();
              grid.CurrentCell = row.Cells["Rank"];
              grid.BeginEdit(true);
              DataGridViewTextBoxEditingControl rankControl =
(DataGridViewTextBoxEditingControl)grid.EditingControl;
              rankControl.Text = pp.Rank.ToString();
              grid.EndEdit();
            }
          }
          break;
        default:
          break;
      }
    }
  }

  private void dgvItems_RowValidating(object sender,
DataGridViewCellCancelEventArgs e)
  {
    DataGridViewRow row =
((DataGridView)sender).CurrentRow;
    foreach (DataGridViewCell cell in row.Cells)
      if (cell.Value == null) e.Cancel = true;
    if (row.Cells["ExecutorId"].Value == null) e.Cancel =
false;
  }
```

```
  private void dgvItems_EditingControlShowing(object
sender, DataGridViewEditingControlShowingEventArgs e)
  {
   if
(dgvItems.Columns[dgvItems.CurrentCell.ColumnIndex].Name
== "Rank")
   {
    DataGridViewTextBoxEditingControl ctrl =
(DataGridViewTextBoxEditingControl)e.Control;
    ctrl.KeyPress += new
KeyPressEventHandler(rankCell_KeyPress);
   }
  }

  void rankCell_KeyPress(object sender, KeyPressEventArgs
e)
  {
   char c = e.KeyChar;
   if (c != (char)1 &&
    c != (char)2 &&
    c != (char)3 &&
    c != (char)4 &&
    c != (char)5 &&
    c != (char)6)
    e.Handled = true;
  }
 }
}
```

Файл Project\Controls\LaborsControl.cs:

```
using System;
using System.Windows.Forms;

namespace Project.Controls
{
 public partial class LaborsControl : UserControl
 {
  public LaborsControl()
  {
   InitializeComponent();
  }

  private void dgvLabors_EditingControlShowing(object
sender, DataGridViewEditingControlShowingEventArgs e)
  {
   if
(dgvItems.Columns[dgvItems.CurrentCell.ColumnIndex].Name
== "Hours")
   {
```

```
    DataGridViewTextBoxEditingControl cell =
(DataGridViewTextBoxEditingControl)e.Control;
    cell.KeyPress += new
KeyPressEventHandler(hours_KeyPress);
   }
  }

  void hours_KeyPress(object sender, KeyPressEventArgs e)
  {
   string separator =
System.Globalization.CultureInfo.CurrentCulture.NumberFor
mat.NumberDecimalSeparator;

   DataGridViewTextBoxEditingControl ctrl =
(DataGridViewTextBoxEditingControl)sender;

   if (ctrl.Text.Contains(separator) && (e.KeyChar !=
(char)Keys.Back) && ctrl.SelectionLength == 0)
   {
    if
(ctrl.Text.Substring(ctrl.Text.IndexOf(separator)).Length
> 1) e.Handled = true;
   }

   if (!Char.IsDigit(e.KeyChar) && (e.KeyChar !=
(char)Keys.Back) && (e.KeyChar != '.') && (e.KeyChar !=
','')) e.Handled = true;

   if (e.KeyChar == '.' || e.KeyChar == ',')
   {
    if (!ctrl.Text.Contains(separator) &&
!(ctrl.Text.Length == 0))
    {
     ctrl.Text += separator;
     ctrl.SelectionStart = ctrl.Text.Length;
    }
    e.Handled = true;
   }
   if (e.KeyChar == '0' && ctrl.Text.Equals("0"))
e.Handled = true;
  }

  private void dgvItems_RowValidating(object sender,
DataGridViewCellCancelEventArgs e)
  {
   DataGridViewRow row =
((DataGridView)sender).CurrentRow;
   foreach(DataGridViewCell cell in row.Cells)
    if (cell.Value == null) e.Cancel = true;
   if (row.Cells["LaborId"].Value == null) e.Cancel =
false;
```

```
    }
  }
}
```

Файл Project\Forms\Controls\PersonProfessionsControl.cs

```
using System.ComponentModel;
using System.Linq;
using System.Windows.Forms;
using Project.Databases;

namespace Project
{
 public partial class PersonProfessionsControl :
TableControl
 {
  private int _PersonId = 0;


[DesignerSerializationVisibility(DesignerSerializationVis
ibility.Hidden)]
  public int PersonId
  {
   get { return this._PersonId; }
   set
   {
    this._PersonId = value;
    Init();
   }
  }

  private DataGridViewColumn _Id = new
DataGridViewTextBoxColumn();
  private DataGridViewColumn _ProfessionCode = new
DataGridViewTextBoxColumn();
  private DataGridViewColumn _ProfessionTitle = new
DataGridViewTextBoxColumn();
  private DataGridViewColumn _Rank = new
DataGridViewTextBoxColumn();

  public PersonProfessionsControl()
  {
   InitializeComponent();

   this._Id.Name = "Id";
   this._Id.Visible = false;

   this._ProfessionCode.Name = "ProfessionCode";
   this._ProfessionCode.HeaderText = "Шифр профессии";
```

```
    this._ProfessionTitle.Name = "ProfessionTitle";
    this._ProfessionTitle.HeaderText = "Наименование
профессии";
    this._ProfessionTitle.AutoSizeMode =
DataGridViewAutoSizeColumnMode.Fill;

    this._Rank.Name = "Rank";
    this._Rank.HeaderText = "Разряд";

    this.dgvItems.Columns.Add(this._Id);
    this.dgvItems.Columns.Add(this._ProfessionCode);
    this.dgvItems.Columns.Add(this._ProfessionTitle);
    this.dgvItems.Columns.Add(this._Rank);


this.dgvFilter.Columns.Add((DataGridViewColumn)this._Prof
essionCode.Clone());

this.dgvFilter.Columns.Add((DataGridViewColumn)this._Prof
essionTitle.Clone());

this.dgvFilter.Columns.Add((DataGridViewColumn)this._Rank
.Clone());
    this.dgvFilter.Rows.Add();

    Init();
   }


[DesignerSerializationVisibility(DesignerSerializationVis
ibility.Hidden)]
  public override void New()
  {
   if (_PersonId != 0)
   {
    Person person = Data.Tables.Persons[_PersonId];
    frmPersonProfession form = new
frmPersonProfession(person);
    form.ShowDialog(this);
    Init();
   }
  }


[DesignerSerializationVisibility(DesignerSerializationVis
ibility.Hidden)]
  public override void Edit()
  {
   if (this.CurrentId != 0)
   {
```

```
    PersonProfession personProfession =
Data.Tables.PersonProfessions[this.CurrentId];
    frmPersonProfession form = new
frmPersonProfession(personProfession);
    form.ShowDialog(this);
    Init();
   }
  }


[DesignerSerializationVisibility(DesignerSerializationVis
ibility.Hidden)]
  public override void Delete()
  {
   if (this.CurrentId != 0)
   {
    if (MessageBox.Show("Вы действительно хотите удалить
запись?", "Удаление записи",
MessageBoxButtons.OKCancel).Equals(DialogResult.OK))
    {
     PersonProfession personProfession =
Data.Tables.PersonProfessions[this.CurrentId];
     personProfession.Delete();
     Init();
    }
   }
  }


[DesignerSerializationVisibility(DesignerSerializationVis
ibility.Hidden)]
  public override void Init()
  {
   this.dgvItems.DataSource = (from personProfession in
Data.Tables.PersonProfessions
          where
          personProfession.PersonId.Equals(_PersonId) &&

personProfession.Profession.Code.ToString().Contains(this
.GetFilter("ProfessionCode")) &&

personProfession.Profession.Title.ToUpper().Contains(this
.GetFilter("ProfessionTitle").ToUpper()) &&

personProfession.Rank.ToString().Contains(this.GetFilter(
"Rank"))
          select new
           {
            Id = personProfession.Id,
            ProfessionCode =
personProfession.Profession.Code,
```

```
                ProfessionTitle =
personProfession.Profession.Title,
                Rank = personProfession.Rank
               }).ToList();

    foreach (DataGridViewColumn column in
this.dgvItems.Columns)
    {
     column.DataPropertyName = column.Name;
    }
   }
  }
}
```

Файл Project\Controls\PersonsControl.cs

```
using System.Collections.Generic;
using System.ComponentModel;
using System.Linq;
using System.Windows.Forms;
using Project.Databases;

namespace Project
{
 public partial class PersonsControl : TableControl
 {
  private DataGridViewColumn _Id = new
DataGridViewTextBoxColumn();
  private DataGridViewColumn _Code = new
DataGridViewTextBoxColumn();
  private DataGridViewColumn _FirstName = new
DataGridViewTextBoxColumn();
  private DataGridViewColumn _MiddleName = new
DataGridViewTextBoxColumn();
  private DataGridViewColumn _LastName = new
DataGridViewTextBoxColumn();
  private List<Person> _Deletions = new List<Person>();


[DesignerSerializationVisibility(DesignerSerializationVis
ibility.Hidden)]
  public List<Person> Deletions
  {
   get { return this._Deletions; }
   set
   {
    this._Deletions = value;
    Init();
   }
  }
```

```
  public PersonsControl()
  {
   InitializeComponent();

   this._Id.Name = "Id";
   this._Id.Visible = false;

   this._Code.Name = "Code";
   this._Code.HeaderText = "Таб. №";

   this._LastName.Name = "LastName";
   this._LastName.HeaderText = "Фамилия";

   this._FirstName.Name = "FirstName";
   this._FirstName.HeaderText = "Имя";

   this._MiddleName.Name = "MiddleName";
   this._MiddleName.HeaderText = "Отчество";
   this._MiddleName.AutoSizeMode =
DataGridViewAutoSizeColumnMode.Fill;

   this.dgvItems.Columns.Add(this._Id);
   this.dgvItems.Columns.Add(this._Code);
   this.dgvItems.Columns.Add(this._LastName);
   this.dgvItems.Columns.Add(this._FirstName);
   this.dgvItems.Columns.Add(this._MiddleName);


this.dgvFilter.Columns.Add((DataGridViewColumn)this._Code
.Clone());

this.dgvFilter.Columns.Add((DataGridViewColumn)this._Last
Name.Clone());

this.dgvFilter.Columns.Add((DataGridViewColumn)this._Firs
tName.Clone());

this.dgvFilter.Columns.Add((DataGridViewColumn)this._Midd
leName.Clone());
   this.dgvFilter.Rows.Add();

   Init();
  }


[DesignerSerializationVisibility(DesignerSerializationVis
ibility.Hidden)]
  public override void New()
  {
   frmPerson form = new frmPerson();
```

```
    form.ShowDialog(this);
    Init();
   }


[DesignerSerializationVisibility(DesignerSerializationVis
ibility.Hidden)]
   public override void Edit()
   {
    if (this.CurrentId != 0)
    {
     Person person = Data.Tables.Persons[this.CurrentId];
     frmPerson form = new frmPerson(person);
     form.ShowDialog(this);
     Init();
    }
   }


[DesignerSerializationVisibility(DesignerSerializationVis
ibility.Hidden)]
   public override void Delete()
   {
    if (this.CurrentId != 0)
    {
     if (MessageBox.Show("Вы действительно хотите удалить
запись?", "Удаление записи",
MessageBoxButtons.OKCancel).Equals(DialogResult.OK))
     {
      Person person = Data.Tables.Persons[this.CurrentId];
      person.Delete();
      Init();
     }
    }
   }


[DesignerSerializationVisibility(DesignerSerializationVis
ibility.Hidden)]
   public override void Init()
   {
    this.dgvItems.DataSource =
Data.Tables.Persons.Except(Deletions.AsEnumerable()).Wher
e(r =>
     r.Code.ToString().Contains(this.GetFilter("Code")) &&

r.FirstName.ToUpper().Contains(this.GetFilter("FirstName"
).ToUpper()) &&

r.MiddleName.ToUpper().Contains(this.GetFilter("MiddleNam
e").ToUpper()) &&
```

```
r.LastName.ToUpper().Contains(this.GetFilter("LastName").
ToUpper())
     ).ToList();

   foreach (DataGridViewColumn column in
this.dgvItems.Columns)
    {
     column.DataPropertyName = column.Name;
    }
   }
  }
}
```

Файл Project\Controls\PositionsControl.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows.Forms;
using Project.Databases;

namespace Project.Controls
{
 public partial class PositionsControl : UserControl
 {
  public PositionsControl()
  {
   InitializeComponent();
  }

  private void dgvItems_RowValidating(object sender,
DataGridViewCellCancelEventArgs e)
  {
   DataGridViewRow row = dgvItems.Rows[e.RowIndex];

   bool title = row.Cells["Title"].Value != null ? true :
false;
   bool draw = row.Cells["Draw"].Value != null ? true :
false;
   bool matherial = row.Cells["Matherial"].Value != null
? true : false;
   bool number = row.Cells["Number"].Value != null ? true
: false;
   bool mass = row.Cells["Mass"].Value != null ? true :
false;
   bool norm = row.Cells["Norm"].Value != null ? true :
false;
```

```
    if (title && draw && matherial && number && mass &&
norm) e.Cancel = false;
    else if (title || draw || matherial || number || mass
|| norm) e.Cancel = true;

    //if (row.Cells["PositionId"].Value == null) e.Cancel
= false;
   }

  void floatCell_KeyPress(object sender,
KeyPressEventArgs e)
   {
    string separator =
System.Globalization.CultureInfo.CurrentCulture.NumberFor
mat.NumberDecimalSeparator;

    DataGridViewTextBoxEditingControl ctrl =
(DataGridViewTextBoxEditingControl)sender;

    if (ctrl.Text.Contains(separator) && (e.KeyChar !=
(char)Keys.Back) && ctrl.SelectionLength == 0)
    {
     if
(ctrl.Text.Substring(ctrl.Text.IndexOf(separator)).Length
> 3) e.Handled = true;
    }

    if (!Char.IsDigit(e.KeyChar) && (e.KeyChar !=
(char)Keys.Back) && (e.KeyChar != '.') && (e.KeyChar !=
','')) e.Handled = true;

    if (e.KeyChar == '.' || e.KeyChar == ',')
    {
     if (!ctrl.Text.Contains(separator) &&
!(ctrl.Text.Length == 0))
     {
      ctrl.Text += separator;
      ctrl.SelectionStart = ctrl.Text.Length;
     }
     e.Handled = true;
    }
    if (e.KeyChar == '0' && ctrl.Text.Equals("0"))
e.Handled = true;
   }

  void intCell_KeyPress(object sender, KeyPressEventArgs
e)
   {
    DataGridViewTextBoxEditingControl ctrl =
(DataGridViewTextBoxEditingControl)sender;
```

```
    if (
     (!Char.IsDigit(e.KeyChar) && (e.KeyChar !=
(char)Keys.Back)) ||
     (ctrl.Text.Length == 0 && e.KeyChar == '0')
     ) e.Handled = true;
   }

   private void dgvItems_EditingControlShowing(object
sender, DataGridViewEditingControlShowingEventArgs e)
   {
    DataGridViewColumn column =
dgvItems.Columns[dgvItems.CurrentCell.ColumnIndex];
    AutoCompleteStringCollection acsCollection = new
AutoCompleteStringCollection();
    List<string> strings = new List<string>();

    DataGridViewTextBoxEditingControl ctrl =
(DataGridViewTextBoxEditingControl)e.Control;

    switch (column.Name)
    {
     case "Title":
      {
       ctrl.AutoCompleteMode = AutoCompleteMode.Append;
       ctrl.AutoCompleteSource =
AutoCompleteSource.CustomSource;
       strings = (from position in Data.Tables.Positions
            select position.Title).ToList();
       foreach (string str in strings)
acsCollection.Add(str);
       ctrl.AutoCompleteCustomSource = acsCollection;
       ctrl.KeyPress -= new
KeyPressEventHandler(floatCell_KeyPress);
       ctrl.KeyPress -= new
KeyPressEventHandler(intCell_KeyPress);
      }
     break;
    case "Draw":
     {
      ctrl.AutoCompleteMode = AutoCompleteMode.Append;
      ctrl.AutoCompleteSource =
AutoCompleteSource.CustomSource;
      strings = (from position in Data.Tables.Positions
           select position.Draw).ToList();
      foreach (string str in strings)
acsCollection.Add(str);
      ctrl.AutoCompleteCustomSource = acsCollection;
      ctrl.KeyPress -= new
KeyPressEventHandler(floatCell_KeyPress);
      ctrl.KeyPress -= new
KeyPressEventHandler(intCell_KeyPress);
```

```
            }
            break;
        case "Matherial":
            {
                ctrl.AutoCompleteMode = AutoCompleteMode.Append;
                ctrl.AutoCompleteSource =
AutoCompleteSource.CustomSource;
                strings = (from position in Data.Tables.Positions
                           select position.Matherial).ToList();
                foreach (string str in strings)
acsCollection.Add(str);
                ctrl.AutoCompleteCustomSource = acsCollection;
                ctrl.KeyPress -= new
KeyPressEventHandler(floatCell_KeyPress);
                ctrl.KeyPress -= new
KeyPressEventHandler(intCell_KeyPress);
            }
            break;
        case "Number":
            {
                ctrl.KeyPress -= new
KeyPressEventHandler(floatCell_KeyPress);
                ctrl.KeyPress += new
KeyPressEventHandler(intCell_KeyPress);
            }
            break;
        case "Mass":
        case "Norm":
        case "Price":
            {
                ctrl.KeyPress -= new
KeyPressEventHandler(intCell_KeyPress);
                ctrl.KeyPress += new
KeyPressEventHandler(floatCell_KeyPress);
            }
            break;
        default: break;
    }
  }
 }
}
```

Файл Project\Controls\ProfessionsControl.cs

```
using System.ComponentModel;
using System.Linq;
using System.Windows.Forms;
using Project.Databases;

namespace Project
```

```
{
 public partial class ProfessionsControl : TableControl
 {
  private DataGridViewColumn _Id = new
DataGridViewTextBoxColumn();
  private DataGridViewColumn _Code = new
DataGridViewTextBoxColumn();
  private DataGridViewColumn _Title = new
DataGridViewTextBoxColumn();
  private DataGridViewColumn _Rank1 = new
DataGridViewTextBoxColumn();
  private DataGridViewColumn _Rank2 = new
DataGridViewTextBoxColumn();
  private DataGridViewColumn _Rank3 = new
DataGridViewTextBoxColumn();
  private DataGridViewColumn _Rank4 = new
DataGridViewTextBoxColumn();
  private DataGridViewColumn _Rank5 = new
DataGridViewTextBoxColumn();
  private DataGridViewColumn _Rank6 = new
DataGridViewTextBoxColumn();

  public ProfessionsControl()
  {
   InitializeComponent();

   this._Id.Name = "Id";
   this._Id.Visible = false;

   this._Code.Name = "Code";
   this._Code.Width = 70;
   this._Code.HeaderText = "Шифр";

   this._Title.Name = "Title";
   this._Title.HeaderText = "Наименование";
   this._Title.AutoSizeMode =
DataGridViewAutoSizeColumnMode.Fill;

   this._Rank1.Name = "Rank1";
   this._Rank1.Width = 80;
   this._Rank1.HeaderText = "1 разряд";

   this._Rank2.Name = "Rank2";
   this._Rank2.Width = 80;
   this._Rank2.HeaderText = "2 разряд";

   this._Rank3.Name = "Rank3";
   this._Rank3.Width = 80;
   this._Rank3.HeaderText = "3 разряд";

   this._Rank4.Name = "Rank4";
```

```
            this._Rank4.Width = 80;
            this._Rank4.HeaderText = "4 разряд";

            this._Rank5.Name = "Rank5";
            this._Rank5.Width = 80;
            this._Rank5.HeaderText = "5 разряд";

            this._Rank6.Name = "Rank6";
            this._Rank6.Width = 80;
            this._Rank6.HeaderText = "6 разряд";

            this.dgvItems.Columns.Add(this._Id);
            this.dgvItems.Columns.Add(this._Code);
            this.dgvItems.Columns.Add(this._Title);
            this.dgvItems.Columns.Add(this._Rank1);
            this.dgvItems.Columns.Add(this._Rank2);
            this.dgvItems.Columns.Add(this._Rank3);
            this.dgvItems.Columns.Add(this._Rank4);
            this.dgvItems.Columns.Add(this._Rank5);
            this.dgvItems.Columns.Add(this._Rank6);


        this.dgvFilter.Columns.Add((DataGridViewColumn)this._Code
        .Clone());

        this.dgvFilter.Columns.Add((DataGridViewColumn)this._Titl
        e.Clone());

        this.dgvFilter.Columns.Add((DataGridViewColumn)this._Rank
        1.Clone());

        this.dgvFilter.Columns.Add((DataGridViewColumn)this._Rank
        2.Clone());

        this.dgvFilter.Columns.Add((DataGridViewColumn)this._Rank
        3.Clone());

        this.dgvFilter.Columns.Add((DataGridViewColumn)this._Rank
        4.Clone());

        this.dgvFilter.Columns.Add((DataGridViewColumn)this._Rank
        5.Clone());

        this.dgvFilter.Columns.Add((DataGridViewColumn)this._Rank
        6.Clone());
            this.dgvFilter.Rows.Add();

            Init();
          }
```

```
[DesignerSerializationVisibility(DesignerSerializationVis
ibility.Hidden)]
  public override void New()
  {
   frmProfession form = new frmProfession();
   form.ShowDialog(this);
   Init();
  }


[DesignerSerializationVisibility(DesignerSerializationVis
ibility.Hidden)]
  public override void Edit()
  {
   if (this.CurrentId != 0)
   {
    Profession profession =
Data.Tables.Professions[this.CurrentId];
    frmProfession form = new frmProfession(profession);
    form.ShowDialog(this);
    Init();
   }
  }


[DesignerSerializationVisibility(DesignerSerializationVis
ibility.Hidden)]
  public override void Delete()
  {
   if (this.CurrentId != 0)
   {
    if (MessageBox.Show("Вы действительно хотите удалить
запись?", "Удаление записи",
MessageBoxButtons.OKCancel).Equals(DialogResult.OK))
    {
     Profession profession =
Data.Tables.Professions[this.CurrentId];
     profession.Delete();
     Init();
    }
   }
  }


[DesignerSerializationVisibility(DesignerSerializationVis
ibility.Hidden)]
  public override void Init()
  {
   this.dgvItems.DataSource =
Data.Tables.Professions.Where(r =>
     r.Code.ToString().Contains(this.GetFilter("Code")) &&
```

```
r.Title.ToUpper().Contains(this.GetFilter("Title").ToUppe
r()) &&
      r.Rank1.ToString().Contains(this.GetFilter("Rank1"))
&&
      r.Rank2.ToString().Contains(this.GetFilter("Rank2"))
&&
      r.Rank3.ToString().Contains(this.GetFilter("Rank3"))
&&
      r.Rank4.ToString().Contains(this.GetFilter("Rank4"))
&&
      r.Rank5.ToString().Contains(this.GetFilter("Rank5"))
&&
      r.Rank6.ToString().Contains(this.GetFilter("Rank6"))
      ).ToList();

    foreach (DataGridViewColumn column in
this.dgvItems.Columns)
    {
     column.DataPropertyName = column.Name;
    }
   }
  }
}
```

Файл Project\Controls\StructureControl.cs

```
using System;
using System.ComponentModel;
using System.Linq;
using System.Windows.Forms;
using Project.Databases;
using Project.Forms.Elements;

namespace Project.Controls
{
 public partial class StructureControl : UserControl
 {
  private CatalogMode _CatalogMode;
  private ContextMenuStrip _cmArea = new
ContextMenuStrip();
  private ContextMenuStrip _cmBrigade = new
ContextMenuStrip();
  private int _SelectedAreaId = 0;
  private int _SelectedBrigadeId = 0;


[DesignerSerializationVisibility(DesignerSerializationVis
ibility.Hidden)]
```

```
  public int SelectedAreaId
  {
   get
   {
    return this._SelectedAreaId;
   }
   private set
   {
    this._SelectedAreaId = value;
   }
  }


[DesignerSerializationVisibility(DesignerSerializationVis
ibility.Hidden)]
  public int SelectedBrigadeId
  {
   get
   {
    return this._SelectedBrigadeId;
   }
   private set
   {
    this._SelectedBrigadeId = value;
   }
  }


[DesignerSerializationVisibility(DesignerSerializationVis
ibility.Hidden)]
  public CatalogMode CatalogMode
  {
   get { return this._CatalogMode; }
   set
   {
    this._CatalogMode = value;
    if (value == CatalogMode.Select)
    {
     this.scMain.Panel2Collapsed = true;
     this.Height = this.scMain.Panel1.Height;
     this._cmArea = new ContextMenuStrip();
     this._cmBrigade = this.cmBrigadeSelect;

    }
    if (value == CatalogMode.View)
    {
     this.scMain.Panel2Collapsed = false;
     this._cmArea = cmAreaView;
     this._cmBrigade = cmBrigadeView;
    }
    Init();
```

```
    }
   }

   public StructureControl()
   {
    InitializeComponent();
    this.CatalogMode = CatalogMode.View;
    Init();
   }


[DesignerSerializationVisibility(DesignerSerializationVis
ibility.Hidden)]
   public void NewArea()
   {
    frmArea form = new frmArea();
    form.ShowDialog(this);
    Init();
   }


[DesignerSerializationVisibility(DesignerSerializationVis
ibility.Hidden)]
   public void EditArea()
   {
    if (this.SelectedAreaId != 0)
    {
     Area area = Data.Tables.Areas[this.SelectedAreaId];
     frmArea form = new frmArea(area);
     form.ShowDialog(this);
     Init();
    }
   }


[DesignerSerializationVisibility(DesignerSerializationVis
ibility.Hidden)]
   public void DeleteArea()
   {
    if (this.SelectedAreaId != 0)
    {
     if (MessageBox.Show("Вы действительно хотите удалить
запись?", "Удаление записи",
MessageBoxButtons.OKCancel).Equals(DialogResult.OK))
     {
      Area area = Data.Tables.Areas[this.SelectedAreaId];
      area.Delete();
      Init();
     }
    }
   }
```

```
[DesignerSerializationVisibility(DesignerSerializationVis
ibility.Hidden)]
  public void NewBrigade()
  {
   if (this.SelectedAreaId != 0)
   {
    Area area = Data.Tables.Areas[this.SelectedAreaId];
    frmBrigade form = new frmBrigade(area);
    form.ShowDialog(this);
    Init();
   }
  }


[DesignerSerializationVisibility(DesignerSerializationVis
ibility.Hidden)]
  public void EditBrigade()
  {
   if (this.SelectedBrigadeId != 0)
   {
    Brigade brigade =
Data.Tables.Brigades[this.SelectedBrigadeId];
    frmBrigade form = new frmBrigade(brigade);
    form.ShowDialog(this);
    Init();
   }
  }


[DesignerSerializationVisibility(DesignerSerializationVis
ibility.Hidden)]
  public void DeleteBrigade()
  {
   if (this.SelectedBrigadeId != 0)
   {
    if (MessageBox.Show("Вы действительно хотите удалить
запись?", "Удаление записи",
MessageBoxButtons.OKCancel).Equals(DialogResult.OK))
    {

Data.Tables.Brigades[this.SelectedBrigadeId].Delete();
     Init();
    }
   }
  }


[DesignerSerializationVisibility(DesignerSerializationVis
ibility.Hidden)]
```

```
  public virtual void Init()
  {
   if (tvStructure.Nodes.Count != 0)
tvStructure.Nodes.Clear();
   Areas areas = Data.Tables.Areas.Active;

   foreach (Area area in areas.OrderBy(r => r.Code))
   {
    TreeNode areaNode = new TreeNode();
    areaNode.Text = String.Format("{0}  {1}",
area.Code.ToString("D2"), area.Title);
    areaNode.Tag = area;

    this.tvStructure.Nodes.Add(areaNode);

    foreach (Brigade brigade in area.Brigades)
    {
     if (brigade.IsActive)
     {
      TreeNode brigadeNode = new TreeNode();
      brigadeNode.Text = String.Format("{0}  {1}",
brigade.Code.ToString("D2"), brigade.Title);

      brigadeNode.Tag = brigade;

      areaNode.Nodes.Add(brigadeNode);
     }
    }
   }
   this.tvStructure.ExpandAll();
  }

  private void tvStructure_NodeMouseClick(object sender,
TreeNodeMouseClickEventArgs e)
  {
   TreeView tv = (TreeView)sender;
   tv.SelectedNode = e.Node;
   TreeNode node = e.Node;

   if (e.Button.Equals(MouseButtons.Right))
   {
    if (node.Tag is Area)
    {
     _cmArea.Show(tv, e.Location);
    }
    if (node.Tag is Brigade)
    {
     _cmBrigade.Show(tv, e.Location);
    }
   }
  }
```

```
  private void miBrigadeSelect_Click(object sender,
EventArgs e)
  {
   this.SelectedBrigadeId =
((Brigade)tvStructure.SelectedNode.Tag).Id;
   this.FindForm().Close();
  }

  private void tvStructure_NodeMouseDoubleClick(object
sender, TreeNodeMouseClickEventArgs e)
  {
   if (this.CatalogMode == CatalogMode.Select &&
tvStructure.SelectedNode.Tag is Brigade)
   {
    this.SelectedBrigadeId =
((Brigade)tvStructure.SelectedNode.Tag).Id;
    this.FindForm().Close();
   }

  }

  private void miBrigadeAdd_Click(object sender,
EventArgs e)
  {
   NewBrigade();
  }

  private void miAreaEdit_Click(object sender, EventArgs
e)
  {
   EditArea();
  }

  private void miAreaDelete_Click(object sender,
EventArgs e)
  {
   DeleteArea();
  }

  private void miBrigadeEdit_Click(object sender,
EventArgs e)
  {
   EditBrigade();
  }

  private void miBrigadeDelete_Click(object sender,
EventArgs e)
  {
   DeleteBrigade();
  }
```

```
  private void tvStructure_BeforeSelect(object sender,
TreeViewCancelEventArgs e)
  {
   if (e.Node.Tag is Area)
   {
    SelectedAreaId = (e.Node.Tag as Area).Id;
    SelectedBrigadeId = 0;
   }
   if (e.Node.Tag is Brigade)
   {
    SelectedBrigadeId = (e.Node.Tag as Brigade).Id;
   }
  }

  private void bAreaNew_Click(object sender, EventArgs e)
  {
   frmArea form = new frmArea();
   if (form.ShowDialog() == DialogResult.OK)
    Init();
  }
 }
}
```

Файл Project\Controls\TableControl.cs

```
using System;
using System.Windows.Forms;

namespace Project
{
 /// <summary>
 /// Режим просмотра справочника
 /// </summary>
 public enum CatalogMode
 {
  View = 0,
  Select = 1
 }

 public partial class TableControl : UserControl
 {
  public virtual void New() { }
  public virtual void Edit() { }
  public virtual void Delete() { }
  public virtual void Init() { }

  private CatalogMode _CatalogMode;
  private int _SelectedId = 0;
```

```
    private ContextMenuStrip cms = new ContextMenuStrip();

    /// <summary>
    /// Возвращает идентификатор строки выделенной в
текущий момент
    /// </summary>
    public int CurrentId
    {
     get
     {
      if (!this.dgvItems.SelectedRows.Count.Equals(0))
       return
(int)this.dgvItems.SelectedRows[0].Cells["Id"].Value;
      else return 0;
     }
    }

    /// <summary>
    /// Возвращает идентификатор выбранной строки
    /// </summary>
    public int SelectedId
    {
     get
     {
      return this._SelectedId;
     }
    }

    /// <summary>
    /// Возвращает или задает режим просмотра справочника
    /// </summary>
    public CatalogMode CatalogMode
    {
     get
     {
      return this._CatalogMode;
     }
     set
     {
      this._CatalogMode = value;
      if (value == CatalogMode.Select)
      {
       scMain.Panel2Collapsed = true;
       cms = cmsSelect;
       dgvItems.MouseDoubleClick += new
MouseEventHandler(dgvItems_MouseDoubleClick);
      }
      if (value == CatalogMode.View)
      {
       scMain.Panel2Collapsed = false;
```

```
            cms = cmsView;
            dgvItems.MouseDoubleClick -= new
MouseEventHandler(dgvItems_MouseDoubleClick);
        }
      }
    }

    public TableControl()
    {
      InitializeComponent();
      CatalogMode = CatalogMode.View;
      dgvItems.AutoGenerateColumns = false;
    }

    /// <summary>
    /// Возвращает значение ячейки фильтра в указанном
столбце
    /// </summary>
    /// <param name="columnName">
    /// Имя столбца фильтра
    /// </param>
    /// <returns></returns>
    public string GetFilter(string columnName)
    {
      try
      {
        DataGridViewCell cell =
dgvFilter.Rows[0].Cells[columnName];
        return cell.Value == null ? String.Empty :
cell.Value.ToString().Trim();
      }
      catch
      {
        return String.Empty;
      }
    }

    private void dgvFilter_KeyUp(object sender,
KeyEventArgs e)
    {
      if (e.KeyCode == Keys.Delete)
        dgvFilter.SelectedCells[0].Value = null;
    }
    private void dgvItems_CellMouseDown(object sender,
DataGridViewCellMouseEventArgs e)
    {
      if (e.ColumnIndex >= 0 && e.RowIndex >= 0)
      {
        DataGridView grid = (DataGridView)sender;
        DataGridViewRow row = grid.Rows[e.RowIndex];
        row.ContextMenuStrip = cms;
```

```
      row.Selected = true;
     }
    }
    private void dgvItems_CellMouseUp(object sender,
DataGridViewCellMouseEventArgs e)
    {
     if (e.ColumnIndex >= 0 && e.RowIndex >= 0 && e.Button
== MouseButtons.Right)

((DataGridView)sender).Rows[e.RowIndex].ContextMenuStrip.
Show();
    }

    private void miSelect_Click(object sender, EventArgs e)
    {
     if (this.CurrentId != 0)
      this._SelectedId = this.CurrentId;
     this.FindForm().Close();
    }

    private void bNew_Click(object sender, EventArgs e) {
New(); }

    private void bEdit_Click(object sender, EventArgs e) {
Edit(); }

    private void bDelete_Click(object sender, EventArgs e)
{ Delete(); }

    private void miEdit_Click(object sender, EventArgs e) {
Edit(); }

    private void miDelete_Click(object sender, EventArgs e)
{ Delete(); }

    private void dgvFilter_CellValueChanged(object sender,
DataGridViewCellEventArgs e) { Init(); }

    private void dgvItems_ColumnWidthChanged(object sender,
DataGridViewColumnEventArgs e)
    {
     dgvFilter.Columns[e.Column.Name].Width =
dgvItems.Columns[e.Column.Name].Width;
    }

    void dgvItems_MouseDoubleClick(object sender,
MouseEventArgs e)
    {
     if (this.CurrentId != 0)
     {
      this._SelectedId = this.CurrentId;
```

```
      this.FindForm().Close();
     }
    }
   }
  }
```

Файл Project\Controls\WarrantiesControl.cs

```
using System.ComponentModel;
using System.Linq;
using System.Windows.Forms;
using Project.Databases;
using Project.Forms.Elements;

namespace Project.Controls
{
 public partial class WarrantiesControl : TableControl
 {
  private DataGridViewColumn _Id = new
DataGridViewTextBoxColumn();
  private DataGridViewColumn _Customer = new
DataGridViewTextBoxColumn();
  private DataGridViewColumn _Order = new
DataGridViewTextBoxColumn();
  private DataGridViewColumn _Percent = new
DataGridViewTextBoxColumn();
  private DataGridViewColumn _WarrantyDate = new
DataGridViewTextBoxColumn();
  private DataGridViewColumn _AreaCode = new
DataGridViewTextBoxColumn();
  private DataGridViewColumn _BrigadeCode = new
DataGridViewTextBoxColumn();

  private System.Collections.Generic.List<Warranty>
_Includes;


[DesignerSerializationVisibility(DesignerSerializationVis
ibility.Hidden)]
  public System.Collections.Generic.List<Warranty>
Includes
  {
   set
   {
    this._Includes = value;
    Init();
   }
  }
```

```
    public WarrantiesControl()
    {
     InitializeComponent();

     this._Id.Name = "Id";
     this._Id.Visible = false;

     this._Customer.Name = "Customer";
     this._Customer.HeaderText = "Заказчик";
     this._Customer.AutoSizeMode =
DataGridViewAutoSizeColumnMode.Fill;

     this._Order.Name = "Order";
     this._Order.HeaderText = "Заказ";

     this._WarrantyDate.Name = "WarrantyDate";
     this._WarrantyDate.HeaderText = "Дата документа";

     this._AreaCode.Name = "AreaCode";
     this._AreaCode.HeaderText = "Шифр участка";

     this._BrigadeCode.Name = "BrigadeCode";
     this._BrigadeCode.HeaderText = "Шифр бригады";

     this._Percent.Name = "Percent";
     this._Percent.HeaderText = "Процент";

     this.dgvItems.Columns.Add(this._Id);
     this.dgvItems.Columns.Add(this._Customer);
     this.dgvItems.Columns.Add(this._Order);
     this.dgvItems.Columns.Add(this._WarrantyDate);
     this.dgvItems.Columns.Add(this._AreaCode);
     this.dgvItems.Columns.Add(this._BrigadeCode);
     this.dgvItems.Columns.Add(this._Percent);


this.dgvFilter.Columns.Add((DataGridViewColumn)this._Cust
omer.Clone());

this.dgvFilter.Columns.Add((DataGridViewColumn)this._Orde
r.Clone());

this.dgvFilter.Columns.Add((DataGridViewColumn)this._Warr
antyDate.Clone());

this.dgvFilter.Columns.Add((DataGridViewColumn)this._Area
Code.Clone());

this.dgvFilter.Columns.Add((DataGridViewColumn)this._Brig
adeCode.Clone());
```

```
this.dgvFilter.Columns.Add((DataGridViewColumn)this._Perc
ent.Clone());
    this.dgvFilter.Rows.Add();

    Init();

   }


[DesignerSerializationVisibility(DesignerSerializationVis
ibility.Hidden)]
   public override void New()
   {
    frmWarranty form = new frmWarranty();
    if (form.ShowDialog(this) == DialogResult.OK)
     Init();
   }


[DesignerSerializationVisibility(DesignerSerializationVis
ibility.Hidden)]
   public override void Edit()
   {
    if (this.CurrentId != 0)
    {
     Warranty warranty =
Data.Tables.Warranties[this.CurrentId];
     frmWarranty form = new frmWarranty(warranty);
     if (form.ShowDialog(this) == DialogResult.OK)
      Init();
    }
   }


[DesignerSerializationVisibility(DesignerSerializationVis
ibility.Hidden)]
   public override void Delete()
   {
    if (this.CurrentId != 0)
    {
     if (MessageBox.Show("Вы действительно хотите удалить
запись?", "Удаление записи",
MessageBoxButtons.OKCancel).Equals(DialogResult.OK))
     {
      Warranty warranty =
Data.Tables.Warranties[this.CurrentId];
      warranty.Delete();
      Init();
     }
    }
   }
```

```
[DesignerSerializationVisibility(DesignerSerializationVis
ibility.Hidden)]
  public override void Init()
  {
   System.Collections.Generic.List<Warranty> Source;
   if (this._Includes != null) Source = this._Includes;
   else Source = Data.Tables.Warranties.ToList();

    this.dgvItems.DataSource = (from warranty in Source
           where

warranty.Customer.ToUpper().Contains(this.GetFilter("Cust
omer").ToUpper()) &&

warranty.Order.ToString().Contains(this.GetFilter("Order"
)) &&

warranty.WarrantyDate.ToString().Contains(this.GetFilter(
"WarrantyDate")) &&

warranty.Percent.ToString().Contains(this.GetFilter("Perc
ent")) &&

Data.Tables.Areas[warranty.AreaId].Code.ToString().Contai
ns(this.GetFilter("AreaCode")) &&

Data.Tables.Brigades[warranty.BrigadeId].Code.ToString().
Contains(this.GetFilter("BrigadeCode"))
           select new
           {
            Id = warranty.Id,
            Customer = warranty.Customer,
            Order = warranty.Order,
            WarrantyDate =
warranty.WarrantyDate.ToShortDateString(),
            AreaCode =
Data.Tables.Areas[warranty.AreaId].Code,
            BrigadeCode =
Data.Tables.Brigades[warranty.BrigadeId].Code,
            Percent = warranty.Percent
           }).ToList();

   foreach (DataGridViewColumn column in
this.dgvItems.Columns)
   {
    column.DataPropertyName = column.Name;
   }
  }
```

```
  }
}
```

Файл Project\Forms\Elements\frmArea.cs

```
using System.Windows.Forms;
using Project.Databases;
using System.Linq;

namespace Project.Forms.Elements
{
 public partial class frmArea : Form
 {
  private Area _Area;

  public frmArea()
  {
   InitializeComponent();
  }

  public frmArea(Area area)
  {
   InitializeComponent();
   this.Text = "Изменение данных об участке.";
   this._Area = area;
   this.mtbCode.Text = area.Code.ToString("D2");
   this.tbTitle.Text = area.Title;
  }

  private bool Check()
  {
   byte code = System.Convert.ToByte(mtbCode.Text);
   string title = tbTitle.Text;
   if (code == 0)
   {
    (new ToolTip()).Show("Шифр 00 не допускается", this,
this.mtbCode.Location, 2000);
    return false;
   }
   else if (!Data.Tables.Areas.Active.Where(r => r.Code
== code).Count().Equals(0))
   {
    (new ToolTip()).Show("Участок с таким шифром уже
существует.", this, this.mtbCode.Location, 2000);
    return false;
   }
   else if (!Data.Tables.Areas.Active.Where(r =>
r.Title.Equals(title)).Count().Equals(0))
   {
```

```
    (new ToolTip()).Show("Участок с таким названием уже
существует.", this, this.mtbCode.Location, 2000);
     return false;
    }
    else return true;
   }

   private void bSave_Click(object sender,
System.EventArgs e)
   {
    if (Check())
    {
     Area area = new
Area(System.Convert.ToByte(mtbCode.Text),
tbTitle.Text.Trim());
     if (this._Area == null)
Data.Tables.Areas.Insert(area);
     else this._Area.Update(area);
     this.DialogResult = DialogResult.OK;
     this.Close();
    }
   }

   private void bCancel_Click(object sender,
System.EventArgs e)
   {
    this.DialogResult = DialogResult.Cancel;
    this.Close();
   }
  }
}
```

Файл Project\Forms\Elements\frmBrigade.cs

```
using System;
using System.Linq;
using System.Windows.Forms;
using Project.Databases;

namespace Project.Forms.Elements
{
 public partial class frmBrigade : Form
 {
  private Area _Area;
  private Brigade _Brigade;

  public frmBrigade(Brigade brigade)
  {
   InitializeComponent();
```

```
        this.Text = "Изменение данных о бригаде.";
        this._Area = brigade.Area;
        this._Brigade = brigade;
        this.mtbCode.Text = brigade.Code.ToString("D2");
        this.tbTitle.Text = brigade.Title;
      }

      public frmBrigade(Area area)
      {
        InitializeComponent();
        this._Area = area;
      }

      private void bSave_Click(object sender, EventArgs e)
      {
        if (Check())
        {
          Brigade brigade = new Brigade(this._Area.Id,
Convert.ToByte(mtbCode.Text), tbTitle.Text);
          if (this._Brigade == null)
            Data.Tables.Brigades.Insert(brigade);
          else
            this._Brigade.Update(brigade);
          this.DialogResult = DialogResult.OK;
          this.Close();
        }
      }

      private bool Check()
      {
        if (mtbCode.Text.Length == 0 && tbTitle.Text.Length ==
0)
        {
          (new ToolTip()).Show("Необходимо указать шифр и
название бригады", this, this.bSave.Location, 2000);
          return false;
        }
        else if (mtbCode.Text.Length == 0)
        {
          (new ToolTip()).Show("Необходимо указать шифр
бригады", this, this.mtbCode.Location, 2000);
          return false;
        }
        else if (tbTitle.Text.Length == 0)
        {
          (new ToolTip()).Show("Необходимо указать название
бригады", this, this.tbTitle.Location, 2000);
          return false;
        }
        else
        {
```

```
    byte code = Convert.ToByte(mtbCode.Text);
    string title = tbTitle.Text;
    if(code == 0)
    {
     (new ToolTip()).Show("Шифр 00 не допускается", this,
this.mtbCode.Location, 2000);
     return false;
    }
    else if (!this._Area.Brigades.Where(r => r.Code ==
code && r.IsActive == true).Count().Equals(0))
    {
     (new ToolTip()).Show("Бригада с таким шифром уже
существует на участке", this, this.mtbCode.Location,
2000);
     return false;
    }
    else if (!this._Area.Brigades.Active.Where(r =>
r.Title.Equals(title)).Count().Equals(0))
    {
     (new ToolTip()).Show("Бригада с таким названием уже
существует на участке", this, this.mtbCode.Location,
2000);
     return false;
    }
    else return true;
   }
  }

  private void bCancel_Click(object sender, EventArgs e)
  {
   this.DialogResult = DialogResult.Cancel;
   this.Close();
  }
 }
}
```

Файл Project\Forms\Elements\frmPerson.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Linq;
using System.Windows.Forms;
using Project.Databases;

namespace Project
{
 public partial class frmPerson : Form
 {
  public frmPerson()
```

```
    {
     InitializeComponent();
    }

    public frmPerson(Person item)
    {
     InitializeComponent();
     this.Text = "Изменение данных об сотруднике.";
     this.mtbCode.Text = item.Code.ToString("D4");
     this.tbFirstName.Text = item.FirstName;
     this.tbMiddleName.Text = item.MiddleName;
     this.tbLastName.Text = item.LastName;
    }

    private bool Check()
    {
     bool x0 = mtbCode.Text.Length != 0 ? true : false;
// codeIsSet?
     bool x1 = tbFirstName.Text.Trim().Length != 0 ? true :
false;  // firstNameIsSet?
     bool x2 = tbMiddleName.Text.Trim().Length != 0 ? true
: false; // middleNameIsSet?
     bool x3 = tbLastName.Text.Trim().Length != 0 ? true :
false;   // lastNameIsSet?

     bool f1 = x0 && x1 && x2 && x3;         // Form is
valid?

     if (!f1)
     {
      bool f2 = x1 && x2 && x3 ||         // Here is not one
error?
           x1 && x3 && x0 ||
           x1 && x2 && x0 ||
           x2 && x3 && x0;

      string code = x0 ? "" : " табельный номер";
      string fn = x1 ? "" : " имя";
      string ln = x3 ? "" : " фамилию";
      string mn = x2 ? "" : " отчество";
      string and = " и";

      List<string> mes = new List<string>();

      if (!x0) mes.Add(code);
      if (!x3) mes.Add(ln);
      if (!x1) mes.Add(fn);
      if (!x2) mes.Add(mn);

      int c = mes.Count;
```

```
        if (c > 1)
        {
         for (int i = 0; i < c - 2; i++)
         {
          string s = (string)mes[i] + ",";
          mes[i] = s;
         }

         mes.Insert(c - 1, and);
        }

        string message = "Необходимо указать";
        foreach (string s in mes) message += s;
        message += " сотрудника.";

         (new ToolTip()).Show(message, this, bSave.Location,
2000);

         return false;
       }
       else
       {
        short code = Convert.ToInt16(mtbCode.Text);
        string firstName = tbFirstName.Text.Trim();
        string middleName = tbMiddleName.Text.Trim();
        string lastName = tbLastName.Text.Trim();

        bool valid = Data.Tables.Persons
         .Count(r => r.Code.Equals(code)).Equals(0);

        if (!valid)
        {

          (new ToolTip()).Show("Указанный табельный номер уже
присвоен другому сотруднику.", this, bSave.Location,
2000);
         mtbCode.Focus();
         return false;
        }
        else return true;
       }
     }

     private void mtbCode_MaskInputRejected(object sender,
MaskInputRejectedEventArgs e)
     {
      (new ToolTip()).Show("Табельный номер сотрудника
должен состоять из четырех цифр.", this,
mtbCode.Location, 2000);
     }
```

```
  private void mtbCode_Validating(object sender,
CancelEventArgs e)
   {
    if (mtbCode.Text.Length != 0)
    {
     int code;
     if (!Int32.TryParse(mtbCode.Text, out code))
     {
      (new ToolTip()).Show("Табельный номер сотрудника
должен состоять из четырех цифр.", this,
mtbCode.Location, 2000);
      mtbCode.Clear();
      mtbCode.Focus();
      mtbCode.BringToFront();
     }
     else mtbCode.Text = code.ToString("D4");
    }
   }

  private void bSave_Click(object sender, EventArgs e)
   {
    if (Check())
    {
     short code = Convert.ToInt16(mtbCode.Text);
     string firstName = tbFirstName.Text;
     string middleName = tbMiddleName.Text;
     string lastName = tbLastName.Text;

     Person person = new Person(code, firstName,
middleName, lastName);
     Data.Tables.Persons.Insert(person);

     this.Close();
    }
   }
  }
}
```

Файл Project\Forms\Elements\frmPersonProfession.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows.Forms;
using Project.Databases;
using Project.Forms.Tables;

namespace Project
{
```

```
 public partial class frmPersonProfession : Form
 {
  private Person _Person;
  private PersonProfession _PersonProfession;

  public frmPersonProfession(Person person)
  {
   InitializeComponent();
   this._Person = person;
  }

  public frmPersonProfession(PersonProfession
personProfession)
  {
   InitializeComponent();
   this.Text = "Изменение квалификации сотрудника.";
   this._PersonProfession = personProfession;
   this._Person = personProfession.Person;
   this.bProfessionCode.Text =
personProfession.Profession.Code.ToString();
   this.bProfessionCode.Tag = personProfession.Profession;
   this.tbProfessionTitle.Text =
personProfession.Profession.Title;
   this.cbRank.Text = personProfession.Rank.ToString();
  }

  private void bProfessionCode_Click(object sender,
EventArgs e)
  {
   frmProfessions f = new frmProfessions();

   f.professionsControl.CatalogMode = CatalogMode.Select;

   IEnumerable<Profession> ds =
f.professionsControl.dgvItems.DataSource as
IEnumerable<Profession>;
   var nds =
ds.Except(this._Person.PersonProfessions.Select(r =>
r.Profession)).ToList();
   f.professionsControl.dgvItems.DataSource = nds;
   f.ShowDialog(this);
   if (f.professionsControl.CurrentId != 0)
   {
    Profession p =
Data.Tables.Professions[f.professionsControl.CurrentId];
    bProfessionCode.Text = p.Code.ToString();
    bProfessionCode.Tag = p;
    tbProfessionTitle.Text = p.Title;
   }
  }
```

```
  private void bSave_Click(object sender, EventArgs e)
  {
   if (Check())
   {
    int PersonId = this._Person.Id;
    int ProfessionId = (this.bProfessionCode.Tag as
Profession).Id;
    byte Rank = Convert.ToByte(cbRank.Text);
    PersonProfession personProfession = new
PersonProfession(PersonId, ProfessionId, Rank);

    if (this._PersonProfession == null)

Data.Tables.PersonProfessions.Insert(personProfession);
    else this._PersonProfession.Update(personProfession);

    this.Close();
   }
  }

  private bool Check()
  {
   bool x1 = this.bProfessionCode.Text.Length.Equals(0) ?
false : true;
   bool x2 = this.cbRank.Text.Length.Equals(0) ? false :
true;
   return (x1 && x2);
  }
 }
}
```

Файл Project\Forms\Elements\frmProfession.cs

```
using System;
using System.ComponentModel;
using System.Windows.Forms;
using Project.Databases;

namespace Project
{
 public partial class frmProfession : Form
 {
  private Profession _Profession;
  private string _Code
  {
   get { return this.mtbCode.Text; }
  }

  private string _Title
```

```
    {
     get { return this.tbTitle.Text.Trim(); }
    }

    public frmProfession()
    {
     InitializeComponent();
    }

    public frmProfession(Profession item)
    {
     InitializeComponent();
     this.Text = "Изменение данных о профессии.";
     this._Profession = item;
     this.mtbCode.Text = item.Code.ToString("D3");
     this.tbTitle.Text = item.Title;
     this.tbRank1.Text = item.Rank1.ToString();
     this.tbRank2.Text = item.Rank2.ToString();
     this.tbRank3.Text = item.Rank3.ToString();
     this.tbRank4.Text = item.Rank4.ToString();
     this.tbRank5.Text = item.Rank5.ToString();
     this.tbRank6.Text = item.Rank6.ToString();
    }

    private bool Check()
    {
     if (this._Code.Length == 0)
     {
      (new ToolTip()).Show("Необходимо указать код
профессии.", this, mtbCode.Location, 2000);
      mtbCode.Focus();
      return false;
     }
     if (this._Title.Length == 0)
     {
      (new ToolTip()).Show("Необходимо указать название
профессии.", this, tbTitle.Location, 2000);
      tbTitle.Focus();
      return false;
     }
     if (this.tbRank1.Text.Length == 0)
     {
      (new ToolTip()).Show("Необходимо указать тариф 1
разряда.", this, tbRank1.Location, 2000);
      tbRank1.Focus();
      return false;
     }
     if (this.tbRank2.Text.Length == 0)
     {
      (new ToolTip()).Show("Необходимо указать тариф 2
разряда.", this, tbRank2.Location, 2000);
```

```
            tbRank2.Focus();
            return false;
        }
        if (this.tbRank3.Text.Length == 0)
        {
            (new ToolTip()).Show("Необходимо указать тариф 3
разряда.", this, tbRank3.Location, 2000);
            tbRank3.Focus();
            return false;
        }
        if (this.tbRank4.Text.Length == 0)
        {
            (new ToolTip()).Show("Необходимо указать тариф 4
разряда.", this, tbRank4.Location, 2000);
            tbRank4.Focus();
            return false;
        }
        if (this.tbRank5.Text.Length == 0)
        {
            (new ToolTip()).Show("Необходимо указать тариф 5
разряда.", this, tbRank5.Location, 2000);
            tbRank5.Focus();
            return false;
        }
        if (this.tbRank6.Text.Length == 0)
        {
            (new ToolTip()).Show("Необходимо указать тариф 6
разряда.", this, tbRank6.Location, 2000);
            tbRank6.Focus();
            return false;
        }
        return true;
    }

    private void bSave_Click(object sender, EventArgs e)
    {
        if (Check())
        {
            Profession profession = new Profession(
                Convert.ToInt16(this._Code),
                this._Title,
                Convert.ToSingle(this.tbRank1.Text),
                Convert.ToSingle(this.tbRank2.Text),
                Convert.ToSingle(this.tbRank3.Text),
                Convert.ToSingle(this.tbRank4.Text),
                Convert.ToSingle(this.tbRank5.Text),
                Convert.ToSingle(this.tbRank6.Text)
                );
            if (this._Profession == null)
                Data.Tables.Professions.Insert(profession);
            else this._Profession.Update(profession);
```

```
        this.Close();
      }
    }

    private void bCancel_Click(object sender, EventArgs e)
    {
      this.Close();
    }

    private void mtbCode_MaskInputRejected(object sender,
MaskInputRejectedEventArgs e)
    {
      (new ToolTip()).Show("Код вида оплаты должен состоять
из трех цифр.", this, mtbCode.Location, 2000);
    }

    private void mtbCode_Validating(object sender,
CancelEventArgs e)
    {
      if (mtbCode.Text.Length != 0)
      {
        short code;
        if (!Int16.TryParse(mtbCode.Text, out code))
        {
          (new ToolTip()).Show("Код профессии должен состоять
из трех цифр.", this, mtbCode.Location, 2000);
          mtbCode.Clear();
          mtbCode.Focus();
          mtbCode.BringToFront();
        }
        else mtbCode.Text = code.ToString("D3");
      }
    }

    private void tbRank_KeyPress(object sender,
KeyPressEventArgs e)
    {
      string separator =
System.Globalization.CultureInfo.CurrentCulture.NumberFor
mat.NumberDecimalSeparator;

      TextBox ctrl = (TextBox)sender;

      if (ctrl.Text.Contains(separator) && (e.KeyChar !=
(char)Keys.Back) && ctrl.SelectionLength == 0)
      {
        if
(ctrl.Text.Substring(ctrl.Text.IndexOf(separator)).Length
> 3) e.Handled = true;
      }
```

```
    if (!Char.IsDigit(e.KeyChar) && (e.KeyChar !=
(char)Keys.Back) && (e.KeyChar != '.') && (e.KeyChar !=
','')) e.Handled = true;

    if (e.KeyChar == '.' || e.KeyChar == ',')
    {
     if (!ctrl.Text.Contains(separator) &&
!(ctrl.Text.Length == 0))
     {
      ctrl.Text += separator;
      ctrl.SelectionStart = ctrl.Text.Length;
     }
     e.Handled = true;
    }
    if (e.KeyChar == '0' && ctrl.Text.Equals("0"))
e.Handled = true;
   }
  }
}
```

Файл Project\Forms\Elements\frmTable.cs

```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Linq;
using System.Windows.Forms;
using Project.Databases;
using Project.Forms.Tables;

namespace Project.Forms.Elements
{
 public partial class frmTable : Form
 {
  public frmTable()
  {
   InitializeComponent();
  }

  void Calculate()
  {
   this.dgvTable.Columns.Clear();

   DataGridViewColumn PersonName = new
DataGridViewTextBoxColumn();
   PersonName.Name = "PersonName";
   PersonName.MinimumWidth = 100;
   PersonName.Width = 100;
```

```
    PersonName.Frozen = true;
    PersonName.HeaderText = "Ф. И. О.";
    this.dgvTable.Columns.Add(PersonName);

    DataGridViewColumn PersonCode = new
DataGridViewTextBoxColumn();
    PersonCode.Name = "PersonCode";
    PersonCode.Width = 70;
    PersonName.Frozen = true;
    PersonCode.HeaderText = "Таб. №";
    this.dgvTable.Columns.Add(PersonCode);

    DataGridViewColumn SummaryHours = new
DataGridViewTextBoxColumn();
    SummaryHours.Name = "SummaryHours";
    SummaryHours.Width = 70;
    SummaryHours.HeaderText = "Всего часов";
    SummaryHours.Tag = DateTime.Now;
    this.dgvTable.Columns.Add(SummaryHours);

    DataGridViewColumn SummaryMoney = new
DataGridViewTextBoxColumn();
    SummaryMoney.Name = "SummaryMoney";
    SummaryMoney.Width = 70;
    SummaryMoney.HeaderText = "Всего к оплате";
    SummaryMoney.Tag = DateTime.Now;
    this.dgvTable.Columns.Add(SummaryMoney);

    int days = DateTime.DaysInMonth(dtpMonth.Value.Year,
dtpMonth.Value.Month);

    for (int day = 1; day <= days; day++)
    {
     DataGridViewColumn column = new
DataGridViewTextBoxColumn();
     column.Name = day.ToString();
     column.HeaderText = day.ToString();
     column.Width = 25;
     column.Tag = new DateTime(dtpMonth.Value.Year,
dtpMonth.Value.Month, day);
     this.dgvTable.Columns.Add(column);
    }

    foreach (BrigadePerson brigadePerson in (bBrigade.Tag
as Brigade).BrigadePersons)
    {
     Person person = brigadePerson.Person;
     DataGridViewRow row = new DataGridViewRow();

     DataGridViewCell personName = new
DataGridViewTextBoxCell();
```

```
    personName.Value = String.Format("{0} {1}. {2}.",
person.LastName, person.FirstName[0],
person.MiddleName[0]);
    row.Cells.Add(personName);

    DataGridViewCell personCode = new
DataGridViewTextBoxCell();
    personCode.Value = person.Code;
    personCode.Tag = person;
    row.Cells.Add(personCode);

    this.dgvTable.Rows.Add(row);
   }

   foreach (DataGridViewRow row in dgvTable.Rows)
   {
    Person person = row.Cells["PersonCode"].Tag as Person;
    List<Warranty> personWarranties = new
List<Warranty>();
    float personHours = 0F;
    float personMoney = 0F;

    foreach (DataGridViewCell cell in row.Cells)
    {
     if (dgvTable.Columns[cell.ColumnIndex].Tag is
DateTime)
     {
      DateTime date =
(DateTime)dgvTable.Columns[cell.ColumnIndex].Tag;
      float hours = 0F;
      float money = 0F;
      List<Warranty> warranties = (from warranty in
Data.Tables.Warranties
                from executor in warranty.Executors
                from labor in warranty.Labors
                where
                executor.PersonId == person.Id &&
                labor.LaborDate.Equals(date)
                select warranty).ToList();


personWarranties.AddRange(warranties.AsEnumerable());
      foreach (Warranty warranty in warranties)
      {
       Executor executor = (from e in warranty.Executors
             where e.Person.Equals(person)
             select e).First();
       float tariff = 0F;
       switch (executor.Rank)
       {
        case (byte)1:
```

```
                tariff = executor.Profession.Rank1;
                break;
              case (byte)2:
                tariff = executor.Profession.Rank2;
                break;
              case (byte)3:
                tariff = executor.Profession.Rank3;
                break;
              case (byte)4:
                tariff = executor.Profession.Rank4;
                break;
              case (byte)5:
                tariff = executor.Profession.Rank5;
                break;
              case (byte)6:
                tariff = executor.Profession.Rank6;
                break;
              default:
                break;
            }

            foreach (Labor labor in warranty.Labors)
            {
              if (labor.LaborDate.Equals(date))
                hours += labor.Hours;
            }

            money += (warranty.Percent / 100) * tariff;
          }
          cell.Tag = warranties;
          if (hours != 0) cell.Value = hours.ToString();
          personHours += hours;
          personMoney += money;
        }
      }

      row.Cells["SummaryHours"].Tag = personWarranties;
      if (personHours != 0) row.Cells["SummaryHours"].Value
= personHours;
      row.Cells["SummaryMoney"].Tag = personWarranties;
      if (personMoney != 0) row.Cells["SummaryMoney"].Value
= personMoney;
    }
  }

  private void bBrigade_Click(object sender, EventArgs e)
  {
    frmStructure f = new frmStructure();
    f.CatalogMode = CatalogMode.Select;
    f.ShowDialog();
    if (f.ctrlStructure.SelectedBrigadeId != 0)
```

```
    {
     Brigade brigade =
Data.Tables.Brigades[f.ctrlStructure.SelectedBrigadeId];
     bBrigade.Text = String.Format("{0} / {1}",
brigade.Area.Code.ToString("D2"),
brigade.Code.ToString("D2"));
     bBrigade.Tag = brigade;
    }
    if (this.bBrigade.Tag != null)
     Calculate();
   }

   private void dtpMonth_ValueChanged(object sender,
EventArgs e)
   {
    if (this.bBrigade.Tag != null)
     Calculate();
   }

   private void dgvTable_CellMouseClick(object sender,
DataGridViewCellMouseEventArgs e)
   {
    if (e.ColumnIndex >= 0 && e.RowIndex >= 0)
    {
     DataGridViewColumn column =
dgvTable.Columns[e.ColumnIndex];
     DataGridViewCell cell = dgvTable[e.ColumnIndex,
e.RowIndex];

     if (e.Button == MouseButtons.Right &&
      column.Tag is DateTime &&
      cell.Tag is List<Warranty> &&
      cell.Value != null
      )
     {
      dgvTable.ClearSelection();
      cell.Selected = true;
      Rectangle r =
dgvTable.GetCellDisplayRectangle(e.ColumnIndex,
e.RowIndex, false);
      Point location = new Point(r.Location.X +
e.Location.X, r.Location.Y + e.Location.Y);
      this.cmsSelect.Show(dgvTable, location);
     }
    }
   }

   private void miWarranties_Click(object sender,
EventArgs e)
   {
    frmWarranties form = new frmWarranties();
```

```
    form.ctrlWarranties.Includes =
(List<Warranty>)dgvTable.SelectedCells[0].Tag;
    form.ShowDialog();
    Calculate();
   }
  }
}
```

Файл Project\Forms\Elements\frmWarranty.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows.Forms;
using Project.Controls;
using Project.Databases;
using Project.Forms.Tables;

namespace Project.Forms.Elements
{
 public partial class frmWarranty : Form
 {
  private Warranty _Warranty;

  private int _WarrantyId
  {
   get
   {
    if (this._Warranty == null) return 0;
    else return this._Warranty.Id;
   }
  }

  private int _newExecutorId = -1;
  private int _newLaborId = -1;
  private int _newPositionId = -1;
  private float _chainTariff
  {
   get
   {
    float summMoney = 0F;
    float chainTariff = 0F;
    foreach (DataGridViewRow row in
this.ctrlExecutors.dgvItems.Rows)
    {
     if (row.Cells["ProfessionCode"].Tag != null &&
row.Cells["Rank"].Value != null)
     {
      Profession profession =
```

```
(row.Cells["ProfessionCode"].Tag as
PersonProfession).Profession;
      float personTariff = 0F;
      switch (Convert.ToByte(row.Cells["Rank"].Value))
      {
       case 1:
        personTariff = profession.Rank1;
        break;
       case 2:
        personTariff = profession.Rank2;
        break;
       case 3:
        personTariff = profession.Rank3;
        break;
       case 4:
        personTariff = profession.Rank4;
        break;
       case 5:
        personTariff = profession.Rank5;
        break;
       case 6:
        personTariff = profession.Rank6;
        break;
       default:
        break;
      }
      summMoney += personTariff;
     }
    }
    chainTariff = (float)Math.Round((double)(summMoney /
(this.ctrlExecutors.dgvItems.Rows.Count - 1)), 3);
    return chainTariff;
   }
  }
  private float _percent
  {
   get
   {
    float summaryLaborTime = 0F;
    float summaryNormTime = 0F;
    foreach (DataGridViewRow row in
this.ctrlLabors.dgvItems.Rows)
    {
     if (row.Cells["Hours"].Value != null)
      summaryLaborTime +=
Convert.ToSingle(row.Cells["Hours"].Value);
    }
    foreach (DataGridViewRow row in
this.ctrlPositions.dgvItems.Rows)
    {
     if (row.Cells["Number"].Value != null &&
```

```
row.Cells["Norm"].Value != null)
      summaryNormTime +=
       Convert.ToSingle(row.Cells["Number"].Value) *
       Convert.ToSingle(row.Cells["Norm"].Value);
    }
    return
     summaryLaborTime != 0 ?
     (float)Math.Round((double)summaryNormTime /
summaryLaborTime, 5) * 100 : 0;
   }
  }

  public frmWarranty()
  {
   InitializeComponent();
   this.ctrlExecutors.Enabled = false;
   this.ctrlLabors.Enabled = false;
   this.ctrlPositions.Enabled = false;
   this.bSave.Enabled = false;
   Sinc();
  }

  private void Sinc()
  {
   this.ctrlExecutors.dgvItems.CellValueChanged += new
DataGridViewCellEventHandler(Executors_CellValueChanged);
   this.ctrlExecutors.dgvItems.UserAddedRow += new
DataGridViewRowEventHandler(Executors_UserAddedRow);
   this.ctrlExecutors.dgvItems.RowsRemoved += new
DataGridViewRowsRemovedEventHandler(PriceNeeded);
   this.ctrlLabors.dgvItems.CellValueChanged += new
DataGridViewCellEventHandler(Labors_CellValueChanged);
   this.ctrlLabors.dgvItems.UserAddedRow += new
DataGridViewRowEventHandler(Labors_UserAddedRow);
   this.ctrlLabors.dgvItems.RowsRemoved += new
DataGridViewRowsRemovedEventHandler(PriceNeeded);
   this.ctrlPositions.dgvItems.CellValueChanged += new
DataGridViewCellEventHandler(Positions_CellValueChanged);
   this.ctrlPositions.dgvItems.UserAddedRow += new
DataGridViewRowEventHandler(Positions_UserAddedRow);
  }

  void PriceNeeded(object sender,
DataGridViewRowsRemovedEventArgs e)
  {
   CalculatePrice();
  }

  void CalculatePrice()
  {
   foreach (DataGridViewRow row in
```

```
this.ctrlPositions.dgvItems.Rows)
    {
     if (row.Cells["Norm"].Value != null)
     {
      float norm =
Convert.ToSingle(row.Cells["Norm"].Value);
      row.Cells["Price"].Value =
(float)Math.Round((double)norm * this._chainTariff, 3);
     }
    }
    this.lPercentTitle.Visible = true;
    this.lPercent.Text = String.Format("{0}%",
this._percent);
   }

   public frmWarranty(Warranty warranty)
   {
    InitializeComponent();
    this.Text = "Изменение наряда.";
    this._Warranty = warranty;
    List<Position> positions = warranty.Positions.ToList();
    List<Executor> executors = warranty.Executors.ToList();
    List<Labor> labors = (from executor in executors
           from labor in executor.Labors
           select labor).ToList();

    this.tbCustomer.Text = warranty.Customer;
    this.mtbOrder.Text = warranty.Order.ToString("D4");
    Brigade brigade =
Data.Tables.Brigades[warranty.BrigadeId];
    this.bBrigade.Tag = brigade;
    this.bBrigade.Text = String.Format("{0} / {1}",
brigade.Area.Code.ToString("D2"),
brigade.Code.ToString("D2"));

    foreach (Executor executor in executors)
    {
     DataGridViewRow row = new DataGridViewRow();
     DataGridViewCell ExecutorId = new
DataGridViewTextBoxCell();
     DataGridViewCell PersonCode = new
DataGridViewButtonCell();
     DataGridViewCell PersonLastName = new
DataGridViewTextBoxCell();
     DataGridViewCell PersonFirstName = new
DataGridViewTextBoxCell();
     DataGridViewCell PersonMiddleName = new
DataGridViewTextBoxCell();
     DataGridViewCell ProfessionCode = new
DataGridViewButtonCell();
     DataGridViewCell Rank = new DataGridViewTextBoxCell();
```

```
            ExecutorId.Value = executor.Id;
            PersonCode.Value = executor.Person.Code;
            PersonCode.Tag = executor.Person;
            PersonLastName.Value = executor.Person.LastName;
            PersonFirstName.Value = executor.Person.FirstName;
            PersonMiddleName.Value = executor.Person.MiddleName;
            ProfessionCode.Tag =
        Data.Tables.PersonProfessions.First(r =>
        r.Profession.Code == executor.Profession.Code);
            ProfessionCode.Value = executor.Profession.Code;
            Rank.Value = executor.Rank;

            row.Cells.Add(ExecutorId);
            row.Cells.Add(PersonCode);
            row.Cells.Add(PersonLastName);
            row.Cells.Add(PersonFirstName);
            row.Cells.Add(PersonMiddleName);
            row.Cells.Add(ProfessionCode);
            row.Cells.Add(Rank);

            this.ctrlExecutors.dgvItems.Rows.Add(row);
          }
          foreach (Labor labor in labors)
          {
            DataGridViewRow row = new DataGridViewRow();

            DataGridViewCell LaborId = new
        DataGridViewTextBoxCell();
            DataGridViewCell LaborDate = new CalendarCell();
            DataGridViewCell Hours = new
        DataGridViewTextBoxCell();

            LaborId.Value = labor.Id;
            LaborDate.Value = labor.LaborDate;
            Hours.Value = labor.Hours;

            row.Cells.Add(LaborId);
            row.Cells.Add(LaborDate);
            row.Cells.Add(Hours);
            this.ctrlLabors.dgvItems.Rows.Add(row);
          }
          foreach (Position position in positions)
          {
            this.ctrlPositions.dgvItems.Rows
              .Add(position.Id, position.Title, position.Draw,
        position.Matherial, position.Number, position.Mass,
        position.Norm, position.Price);
          }
          Sinc();
        }
```

```
  private void bBrigade_Click(object sender, EventArgs e)
  {
   frmStructure f = new frmStructure();
   f.CatalogMode = CatalogMode.Select;
   f.ShowDialog();
   if (f.ctrlStructure.SelectedBrigadeId != 0)
   {
    Brigade brigade =
Data.Tables.Brigades[f.ctrlStructure.SelectedBrigadeId];
    bBrigade.Text = String.Format("{0} / {1}",
brigade.Area.Code.ToString("D2"),
brigade.Code.ToString("D2"));

    if(bBrigade.Tag != null)
     if (!(bBrigade.Tag as Brigade).Equals(brigade))
     {
      this.ctrlExecutors.dgvItems.Rows.Clear();
      this.ctrlLabors.dgvItems.Rows.Clear();
     }

    bBrigade.Tag = brigade;
    this.ctrlExecutors.BrigadeId = brigade.Id;
    this.ctrlExecutors.Enabled = true;
   }
  }

  private void bSave_Click(object sender, EventArgs e)
  {
   if (Check())
   {
    string customer = this.tbCustomer.Text;
    short order = Convert.ToInt16(this.mtbOrder.Text);
    DateTime warrantyDate = DateTime.Now.Date;
    int areaId = (this.bBrigade.Tag as Brigade).AreaId;
    int brigadeId = (this.bBrigade.Tag as Brigade).Id;
    float percent = this._percent;

    Warranty warranty = new Warranty(customer, order,
percent, warrantyDate, areaId, brigadeId);

    if (this._WarrantyId == 0)
    {
     Data.Tables.Warranties.Insert(warranty);
     this._Warranty = warranty;
    }
    else
    {
     if (!warranty.Equals(this._Warranty))
      this._Warranty.Update(warranty);
    }
```

```
        List<Executor> executors = new List<Executor>();
        List<Labor> labors = new List<Labor>();
        List<Position> positions = new List<Position>();

        foreach (DataGridViewRow executorsRow in
this.ctrlExecutors.dgvItems.Rows)
        {
         if (executorsRow.Cells["ExecutorId"].Value != null)
         {
          int executorId =
Convert.ToInt32(executorsRow.Cells["ExecutorId"].Value);
          int personId =
(executorsRow.Cells["PersonCode"].Tag as Person).Id;
          int professionId =
(executorsRow.Cells["ProfessionCode"].Tag as
PersonProfession).ProfessionId;
          byte rank =
Convert.ToByte(executorsRow.Cells["Rank"].Value);

          Executor executor = new Executor(this._WarrantyId,
personId, professionId, rank);

          if (executorId < 0)
          {
           Data.Tables.Executors.Insert(executor);
           executors.Add(executor);
          }
          else if (executorId > 0)
          {
           Data.Tables.Executors[executorId].Update(executor);
           executors.Add(executor);
          }
         }
        }

        foreach (DataGridViewRow laborsRow in
this.ctrlLabors.dgvItems.Rows)
        {
         if (laborsRow.Cells["LaborId"].Value != null)
         {
          int laborId =
Convert.ToInt32(laborsRow.Cells["LaborId"].Value);
          DateTime laborDate =
Convert.ToDateTime(laborsRow.Cells["LaborDate"].Value).Da
te;
          byte hours =
Convert.ToByte(laborsRow.Cells["Hours"].Value);

          Labor labor = new Labor(this._WarrantyId,
laborDate, hours);
```

```
            if (laborId < 0)
            {
             Data.Tables.Labors.Insert(labor);
             labors.Add(labor);
            }
            else if (laborId > 0)
            {
             Data.Tables.Labors[laborId].Update(labor);
             labors.Add(labor);
            }
           }

         }

         foreach (DataGridViewRow positionsRow in
this.ctrlPositions.dgvItems.Rows)
         {
          if (positionsRow.Cells["Id"].Value != null)
          {
           int positionId =
Convert.ToInt32(positionsRow.Cells["Id"].Value);
           string title =
positionsRow.Cells["Title"].Value.ToString();
           string draw =
positionsRow.Cells["Draw"].Value.ToString();
           string matherial =
positionsRow.Cells["Matherial"].Value.ToString();
           int number =
Convert.ToInt32(positionsRow.Cells["Number"].Value);
           float mass =
Convert.ToSingle(positionsRow.Cells["Mass"].Value);
           float norm =
Convert.ToSingle(positionsRow.Cells["Norm"].Value);
           float price =
Convert.ToSingle(positionsRow.Cells["Price"].Value);

           Position position = new Position(this._WarrantyId,
title, draw, matherial, number, mass, norm, price);

           if (positionId < 0)
           {
            Data.Tables.Positions.Insert(position);
            positions.Add(position);
           }
           else if (positionId > 0)
           {
            Data.Tables.Positions[positionId].Update(position);
            positions.Add(position);
           }
```

```
        }
      }

      if (this._Warranty != null)
      {
       foreach (Executor executor in
this._Warranty.Executors.ToList().Except(executors.AsEnum
erable()))
        executor.Delete();

       foreach (Position position in
this._Warranty.Positions.ToList().Except(positions.AsEnum
erable()))
        position.Delete();
      }
      this.DialogResult = DialogResult.OK;
      this.Close();
     }
    }

    private bool Check()
    {
     if (this.tbCustomer.Text.Length == 0)
     {
      (new ToolTip()).Show("Необходимо указать заказчика",
this, this.tbCustomer.Location, 2000);
      return false;
     }
     if (this.mtbOrder.Text.Length == 0)
     {
      (new ToolTip()).Show("Необходимо указать номер
заказа", this, this.mtbOrder.Location, 2000);
      return false;
     }
     return true;
    }

    void Executors_UserAddedRow(object sender,
DataGridViewRowEventArgs e)
    {
     CalculatePrice();
     this.ctrlLabors.Enabled = true;
    }

    void Executors_CellValueChanged(object sender,
DataGridViewCellEventArgs e)
    {
     CalculatePrice();
     if (e.ColumnIndex >= 0 && e.RowIndex >= 0)
     {
      DataGridView grid = sender as DataGridView;
```

```
    DataGridViewColumn column =
grid.Columns[e.ColumnIndex];
    DataGridViewRow row =
((DataGridView)sender).Rows[e.RowIndex];
    if(column.Name == "PersonCode")
    {
     row.Cells["ProfessionCode"].Value = null;
     row.Cells["ProfessionCode"].Tag = null;
     row.Cells["Rank"].Value = null;
    }
    if (row.Cells["ExecutorId"].Value == null)
    {
     row.Cells["ExecutorId"].Value = this._newExecutorId;
     this._newExecutorId--;
    }
   }
  }

  void Labors_UserAddedRow(object sender,
DataGridViewRowEventArgs e)
  {
   CalculatePrice();
   this.ctrlPositions.Enabled = true;
  }

  void Labors_CellValueChanged(object sender,
DataGridViewCellEventArgs e)
  {
   CalculatePrice();
   if (e.ColumnIndex >= 0 && e.RowIndex >= 0)
   {
    DataGridViewRow row =
((DataGridView)sender).Rows[e.RowIndex];

    if (row.Cells["LaborId"].Value == null)
    {
     row.Cells["LaborId"].Value = this._newLaborId;
     this._newLaborId--;
    }
   }
  }

  void Positions_UserAddedRow(object sender,
DataGridViewRowEventArgs e)
  {
   CalculatePrice();
   this.bSave.Enabled = true;
  }

  void Positions_CellValueChanged(object sender,
DataGridViewCellEventArgs e)
```

```
    {
     CalculatePrice();
     if (e.ColumnIndex >= 0 && e.RowIndex >= 0)
     {
      DataGridView grid = sender as DataGridView;
      DataGridViewColumn column =
grid.Columns[e.ColumnIndex];
      DataGridViewRow row = grid.Rows[e.RowIndex];
      DataGridViewCell cell = grid[e.ColumnIndex,
e.RowIndex];
      if (row.Cells["Id"].Value == null)
      {
       row.Cells["Id"].Value = this._newPositionId;
       this._newPositionId--;
      }
     }
    }
   }
}
```

Файл Project\Forms\Tables\frmBrigadePersons.cs

```
using System.ComponentModel;
using System.Windows.Forms;

namespace Project.Forms.Tables
{
 public partial class frmBrigadePersons : Form
 {

[DesignerSerializationVisibility(DesignerSerializationVis
ibility.Hidden)]
  public CatalogMode CatalogMode
  {
   set
   {
    this.ctrlBrigadePersons.CatalogMode = value;
   }
  }

  public frmBrigadePersons()
  {
   InitializeComponent();
  }
 }
}
```

Файл Project\Forms\Tables\frmPersonProfessions.cs

```
using System.ComponentModel;
using System.Windows.Forms;

namespace Project.Forms.Tables
{
 public partial class frmPersonProfessions : Form
 {

[DesignerSerializationVisibility(DesignerSerializationVis
ibility.Hidden)]
  public int SelectedId
  {
   get { return this.ctrlPersonProfessions.SelectedId; }
  }

  public frmPersonProfessions()
  {
   InitializeComponent();
   this.ctrlPersonProfessions.CatalogMode =
CatalogMode.Select;
  }
 }
}
```

Файл Project\Forms\Tables\frmPersons.cs

```
using System;
using System.ComponentModel;
using System.Windows.Forms;

namespace Project.Forms.Tables
{
 public partial class frmPersons : Form
 {
  private CatalogMode _CatalogMode = CatalogMode.View;


[DesignerSerializationVisibility(DesignerSerializationVis
ibility.Hidden)]
  public int SelectedPersonId
  {
   get
   {
    return this.ctrlPersons.SelectedId;
   }
  }
```

```
[DesignerSerializationVisibility(DesignerSerializationVis
ibility.Hidden)]
  public CatalogMode CatalogMode
  {
   get
   {
    return this._CatalogMode;
   }
   set
   {
    this._CatalogMode = value;
    this.ctrlPersons.CatalogMode = value;
    this.scMain.Panel2Collapsed = value ==
CatalogMode.Select ? true : false;
   }
  }

  public frmPersons()
  {
   InitializeComponent();
   this.ctrlPersons.dgvItems.SelectionChanged += new
EventHandler(personsControl_SelectionChanged);
  }

  private void personsControl_SelectionChanged(object
sender, EventArgs e)
  {
   this.ctrlPersonProfessions.PersonId =
this.ctrlPersons.CurrentId;
  }
 }
}
```

Файл Project\Forms\Tables\frmProfessions.cs

```
using System;
using System.Windows.Forms;

namespace Project.Forms.Tables
{
 public partial class frmProfessions : Form
 {
  public frmProfessions()
  {
   InitializeComponent();
  }
 }
}
```

Файл Project\Forms\Tables\frmStructure.cs

```
using System.ComponentModel;
using System.Windows.Forms;

namespace Project.Forms.Tables
{
 public partial class frmStructure : Form
 {

[DesignerSerializationVisibility(DesignerSerializationVis
ibility.Hidden)]
  public CatalogMode CatalogMode
  {
   set
   {
    this.ctrlStructure.CatalogMode = value;
    if (value == CatalogMode.Select)
    {
     this.scMain.Panel2Collapsed = true;
    }
    if (value == CatalogMode.View)
    {
     this.scMain.Panel2Collapsed = false;
    }
   }
  }

  public frmStructure()
  {
   InitializeComponent();
   this.ctrlStructure.tvStructure.AfterSelect += new
TreeViewEventHandler(tvStructure_AfterSelect);
  }

  void tvStructure_AfterSelect(object sender,
TreeViewEventArgs e)
  {
   this.ctrlBrigadePersons.BrigadeId =
this.ctrlStructure.SelectedBrigadeId;
  }
 }
}
```

Файл Project\Forms\Tables\frmWarranties.cs

```
using System.Windows.Forms;
```

```
namespace Project.Forms.Tables
{
 public partial class frmWarranties : Form
 {
  public frmWarranties()
  {
   InitializeComponent();
  }
 }
}
```

Файл Project\Forms\frmMain.cs

```
using System;
using System.Windows.Forms;
using Project.Forms.Elements;
using Project.Forms.Tables;

namespace Project.Forms
{
 public partial class frmMain : Form
 {
  public frmMain()
  {
   InitializeComponent();
  }

  private void ShowChild(Type type)
  {
   bool isFirst = true;
   for (int i = 0; i < this.MdiChildren.Length; i++)
    if (this.MdiChildren[i].GetType().Equals(type))
     isFirst = false;
   if (isFirst)
   {
    Form form = (Form)Activator.CreateInstance(type);
    form.MdiParent = this;
    form.WindowState = FormWindowState.Maximized;
    form.Show();
   }
  }

  private void mainMenuExit_Click(object sender,
EventArgs e)
  {
   this.Close();
  }

  private void miProfessions_Click(object sender,
```

```
EventArgs e)
   {
    ShowChild(typeof(frmProfessions));
   }

   private void miPersons_Click(object sender, EventArgs e)
   {
    ShowChild(typeof(frmPersons));
   }

   private void miStructure_Click(object sender, EventArgs
e)
   {
    ShowChild(typeof(frmStructure));
   }

   private void miWarranties_Click(object sender,
EventArgs e)
   {
    ShowChild(typeof(frmWarranties));
   }

   private void miTables_Click(object sender, EventArgs e)
   {
    ShowChild(typeof(frmTable));
   }

   private void miAbout_Click(object sender, EventArgs e)
   {
    frmAbout form = new frmAbout();
    form.ShowDialog(this);
   }

   private void frmMain_Load(object sender, EventArgs e)
   {
    frmSplash splash = new frmSplash();
    splash.Show();
    System.Threading.Thread.Sleep(2000);
    splash.Close();
    this.TopMost = true;
    this.TopMost = false;
   }
  }
}
```

7. ПЕРЕЧЕНЬ ССЫЛОК

1. П. В. Шумаков - ADO.NET и создание приложений баз данных в среде Microsoft Visual Studio .NET. *Руководство разработчика с примерами на C#*. – М.: ДИАЛОГ-МИФИ, 2003. – 528 с.

2. Д. С. Раттц - LINQ: язык интегрированных запросов в C# 2008 для профессионалов. : Пер. с англ. - М. : ООО "И. Д. Вильямс", 2008. - 560 с. : ил. - Парал. тит. англ.