МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «Гомельский государственный технический

университет имени П.О. Сухого»

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

Отчёт по лабораторной работе №8

По дисциплине «Объектно-ориентированное программирование»

**«****Технологии доступа к базам данных. LINQ to SQL.»**

Выполнил:

Студент группы ИТИ-21

Говядкова П.Ю.

Принял: ст. преподаватель Стефановский И.Л.

Гомель 2021

**Цель работы:** изучить технологию доступа к базам данным.

**Задание:**

• На основании предметной области разработать БД, состоящую из 3-5 таблиц

(кол-во таблиц и схему данных предварительно согласовать с

преподавателем). Заполнить каждую таблицу 20-30 записями.

• Реализовать CRUD операции используя технологию ADO.NET. В этой

лабораторной работе запрещено использовать компоненты, которые

скрывают работу с ADO.NET.

• При реализации обязательно использовать обобщенные интерфейсы

методы и классы (в рамках темы 6ой лабораторной работы)

• Разработать GUI. Приложение должно быть простым в использовании и

включать в себя полную обработку исключений. Обязательно использовать

подгрузку данных в элементы управления из справочных таблиц (Например:

выпадающие списки).

• Написать unit-тесты для тестирования разработанных библиотечных классов,

тестирование должно покрывать более 80% библиотечного кода.

• При написании и оформлении кода обязательно руководствоваться Code

Convention, принципами ООП, SOLID и использовать элементы авто

документирования с генерацией соответствующих файлов.

Примечание: при реализации 7 и 8 работ приветствуется использование DAO

слоя и паттернов проектирования.



Рисунок 1 – Вариант задания

На рисунке 2 изображена структура решения.

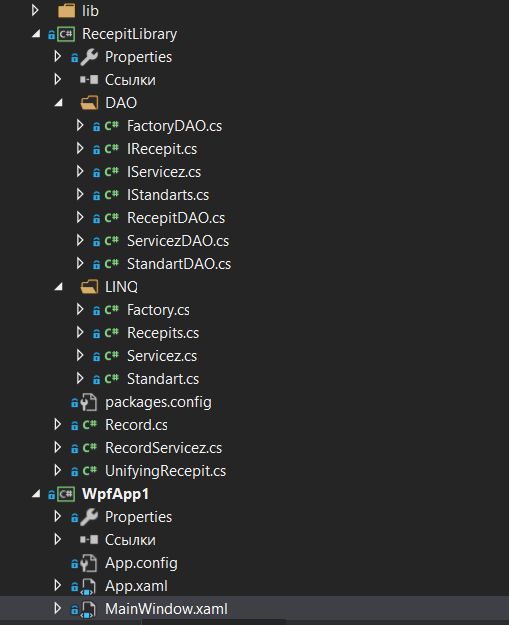


Рисунок 2 – Структура решения

На рисунке 3 изображена схема базы данных.

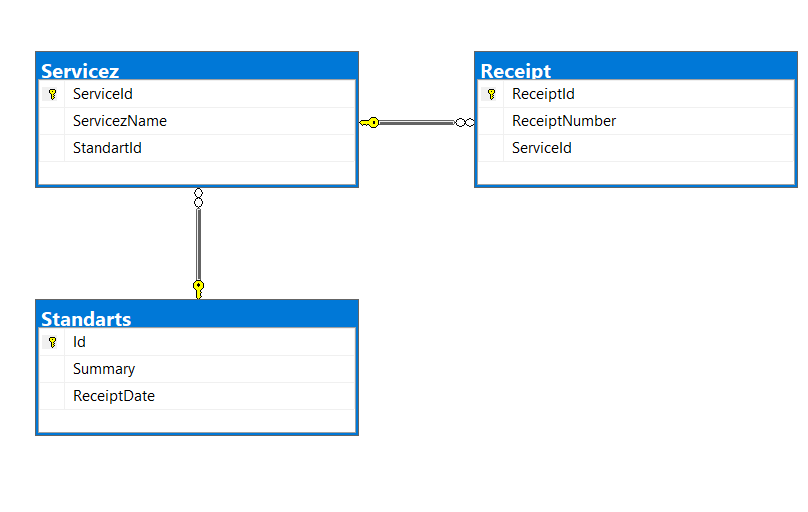


Рисунок 3 – Схема базы данных

При запуске программы открывается основное меню (Рисунок 4).

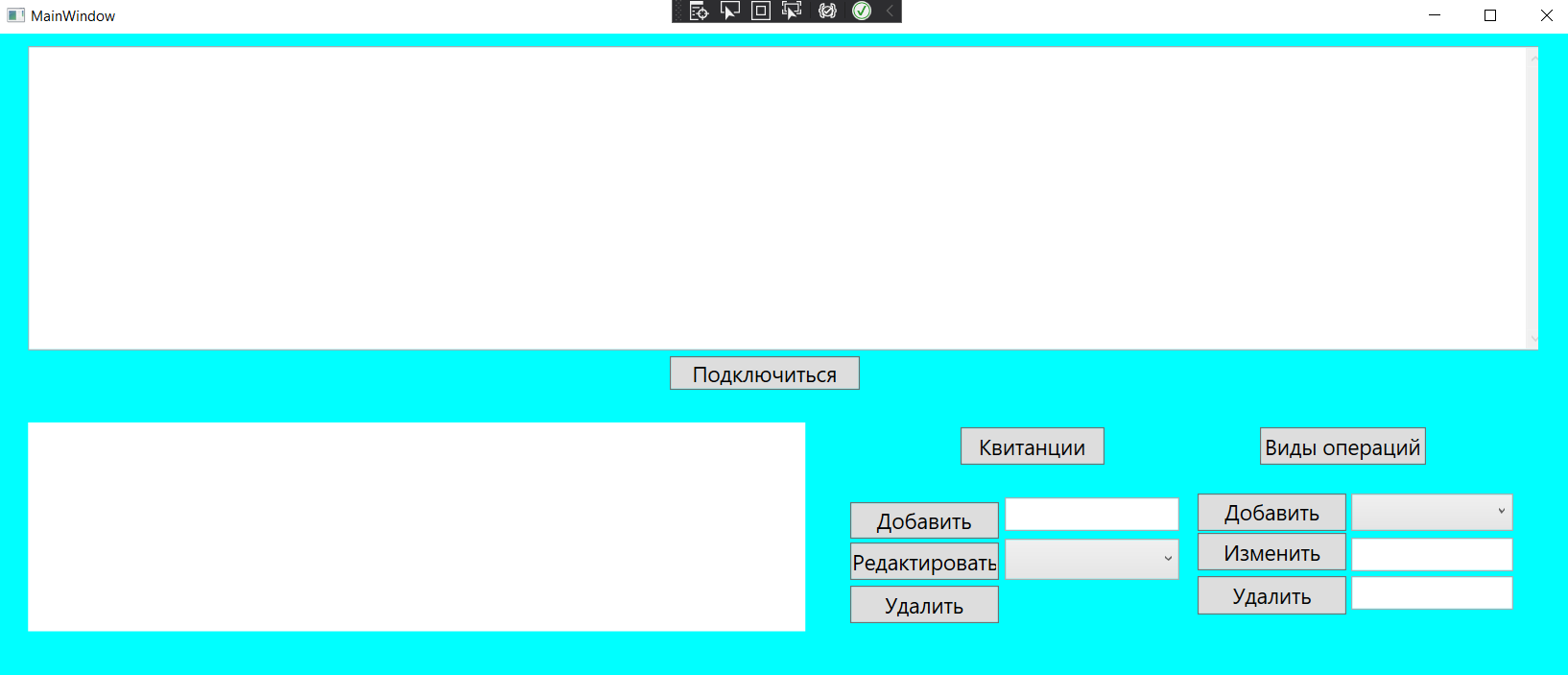


Рисунок 4 – Вывод основного меню

При нажатии кнопки подключиться записи, выводится БД (Рисунок 5).

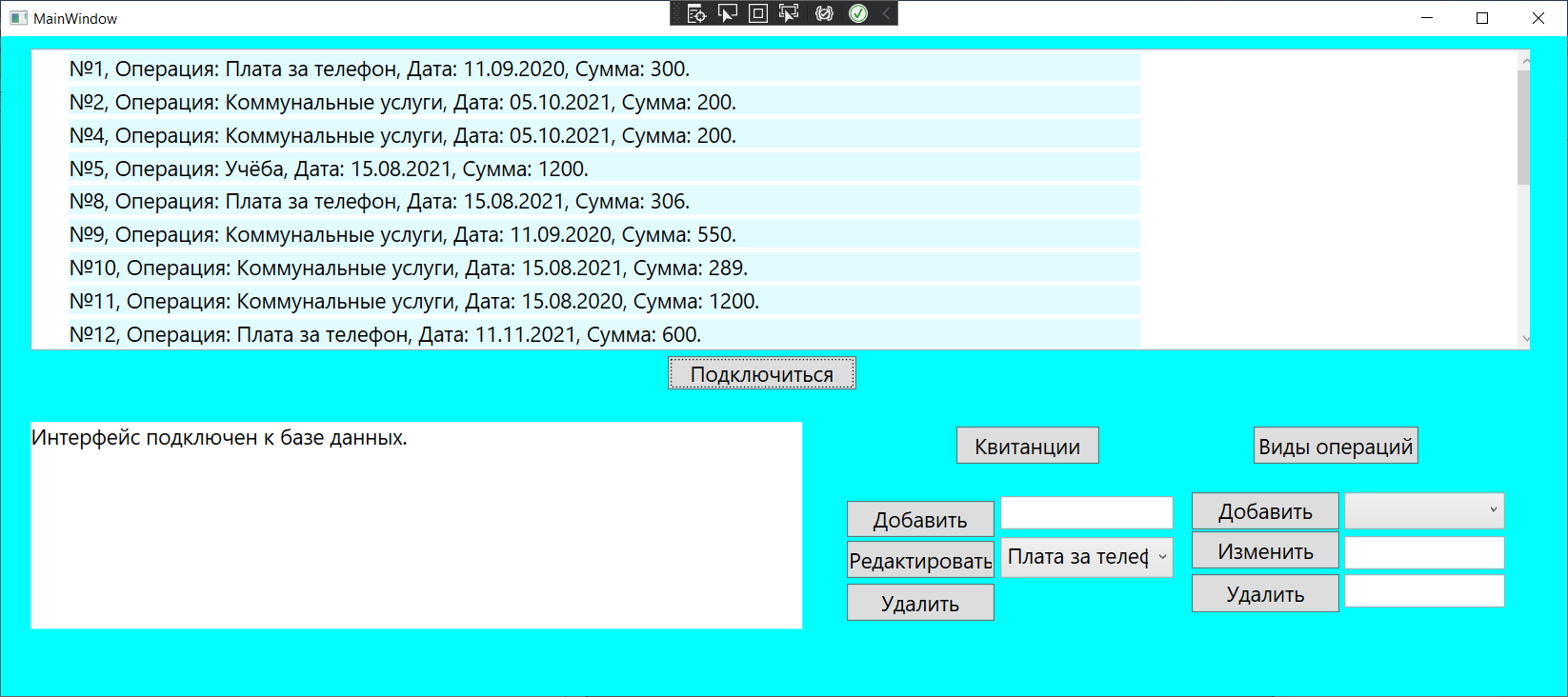


Рисунок 5 – Результат добавления записи

При нажатии кнопки удаления, выбранная запись удаляется из БД (Рисунок 6).

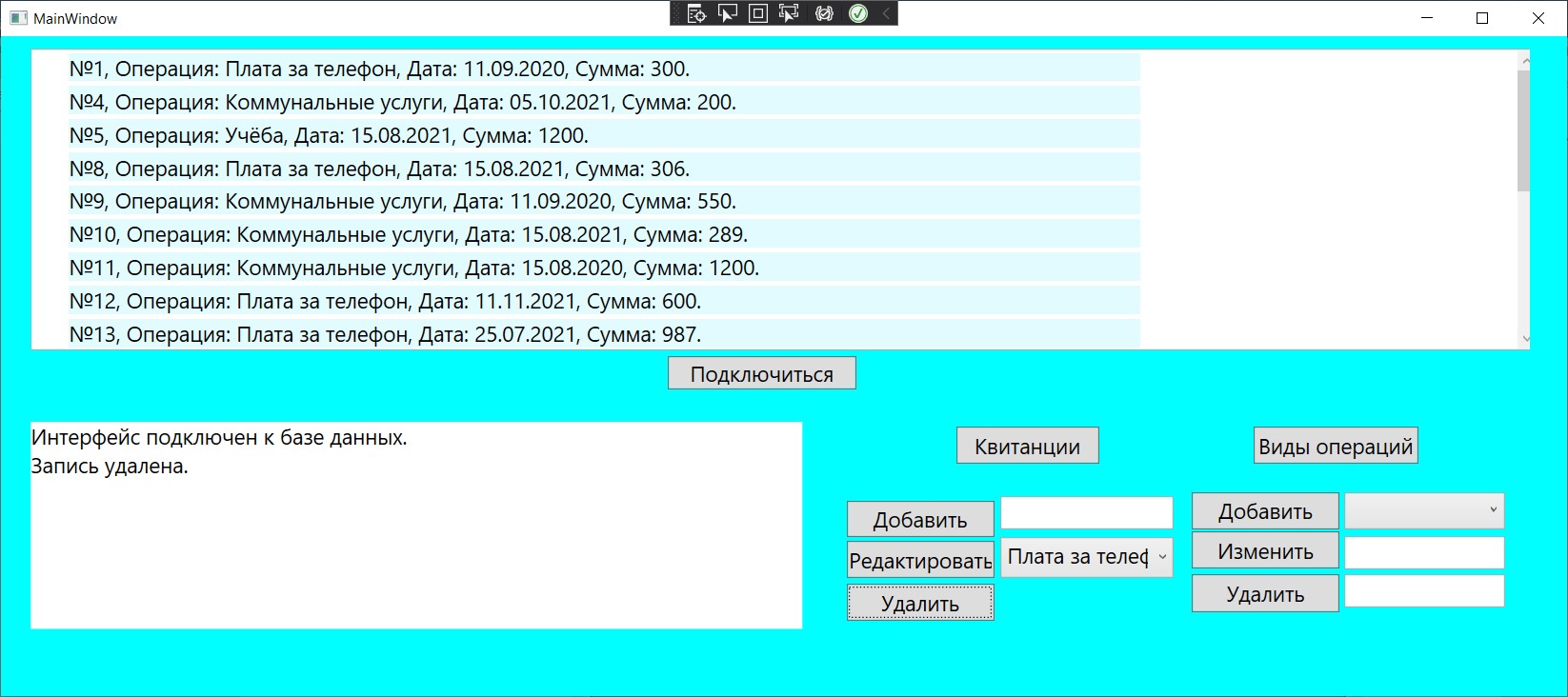


Рисунок 6 – Результат удаления элемента записи

При нажатии кнопки редактировать, в БД записываются изменения (Рисунок 7).

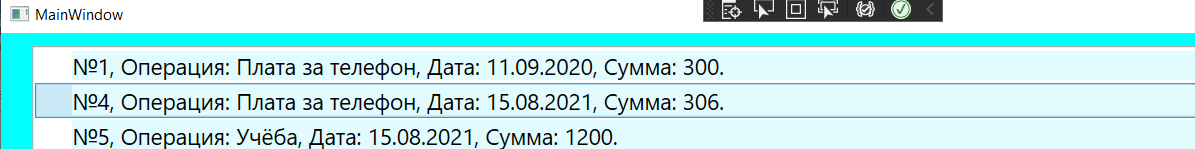


Рисунок 7 – Результат редактирования записей БД

При нажатии кнопки добавления, выбранная запись добавляется в БД (Рисунок 8).

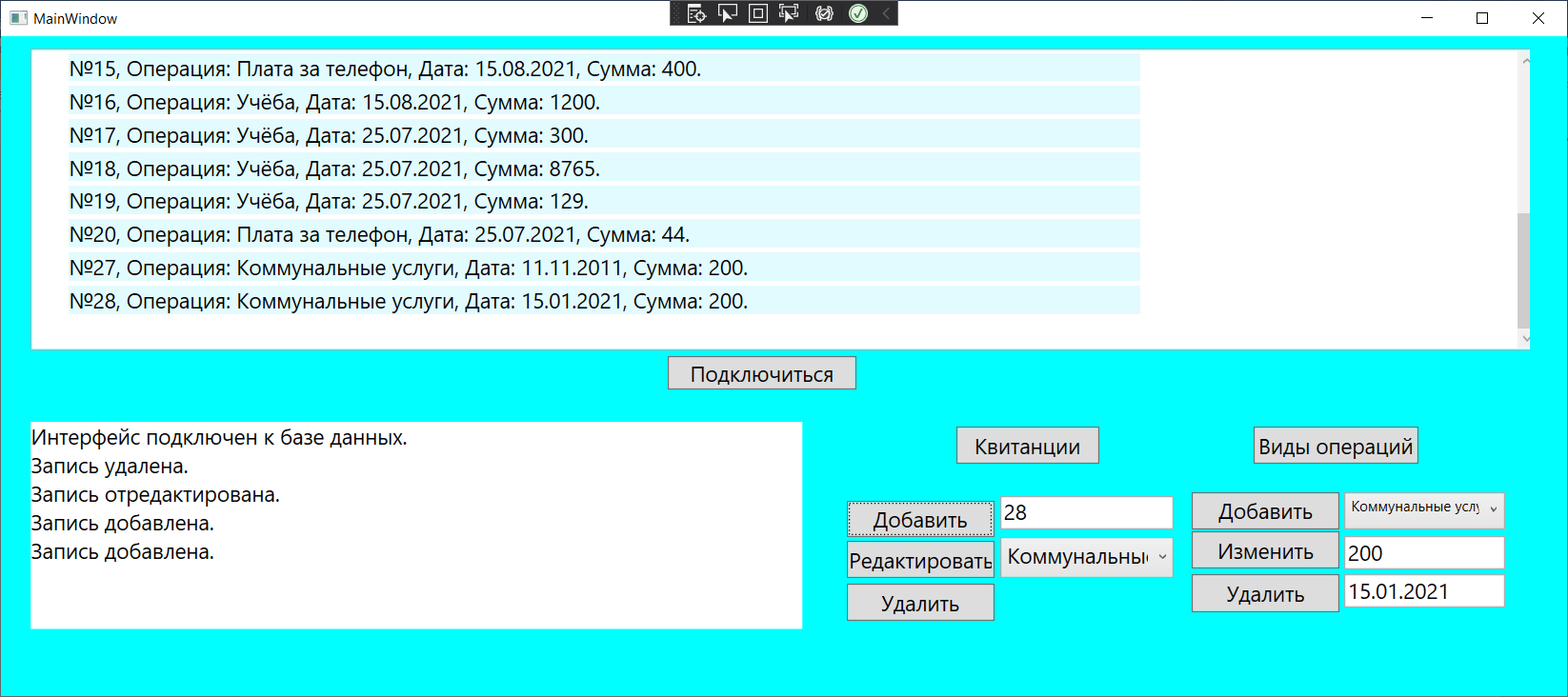


Рисунок 8 – Результат добавления записей БД

На рисунке 9 изображены результаты прохождения модульных тестов.

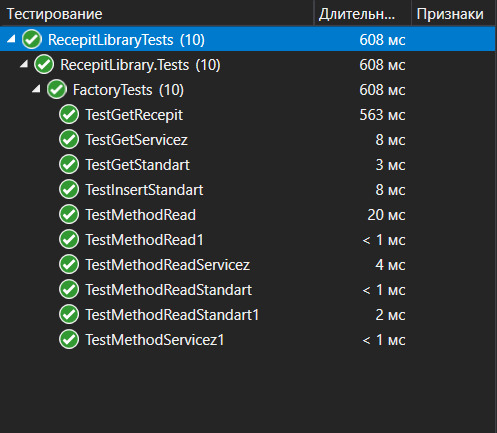


Рисунок 9 – Результат прохождения модульных тестов.

**Вывод:** в ходе выполнения лабораторной работыизучена технология доступа к базам данным.

**ПРИЛОЖЕНИЕ А**

Листинг класса Factory.

using RecepitLibrary.ADO;

using RecepitLibrary.DAO;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace RecepitLibrary.LINQ

{

/// <summary>

/// Factory

/// </summary>

/// <seealso cref="RecepitLibrary.DAO.FactoryDAO" />

public class Factory : FactoryDAO

{

/// <summary>

/// The connection string

/// </summary>

private string \_connectionString;

/// <summary>

/// Initializes a new instance of the <see cref="Factory"/> class.

/// </summary>

/// <param name="\_connectionString">The connection string.</param>

public Factory(string \_connectionString)

{

this.\_connectionString = \_connectionString;

}

/// <summary>

/// Gets the standarts.

/// </summary>

/// <returns></returns>

public override IStandarts GetStandarts()

{

return new Standart(\_connectionString);

}

/// <summary>

/// Gets the servicez.

/// </summary>

/// <returns></returns>

public override IServicez GetServicez()

{

return new Servicez(\_connectionString);

}

/// <summary>

/// Gets the recepits.

/// </summary>

/// <returns></returns>

public override IRecepit GetRecepits()

{

return new Recepits(\_connectionString);

}

}

}

Листинг класса Recepits.

using RecepitLibrary.DAO;

using System;

using System.Collections.Generic;

using System.Data.Linq;

using System.Data.SqlClient;

using System.Globalization;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace RecepitLibrary.LINQ

{

/// <summary>

/// Recepits

/// </summary>

/// <seealso cref="RecepitLibrary.DAO.IRecepit" />

public class Recepits : IRecepit

{

/// <summary>

/// The database

/// </summary>

private DataContext \_database;

/// <summary>

/// Конструктор класса

/// </summary>

/// <param name="connectionString">Строка подключения</param>

public Recepits(string connectionString)

{

\_database = new DataContext(connectionString);

}

/// <summary>

/// Метод, возращающий все записи рецептов

/// </summary>

/// <returns>

/// Массив записей

/// </returns>

public RecepitDAO[] GetAllRecepits()

{

var list = from element in \_database.GetTable<RecepitDAO>() select element;

List<RecepitDAO> recepits = new List<RecepitDAO>();

foreach (RecepitDAO recepit in list)

{

recepits.Add(recepit);

}

return recepits.ToArray();

}

/// <summary>

/// Метод, возращающий запись рецепта

/// </summary>

/// <param name="id">Индефикатор записи</param>

/// <returns>

/// Запись

/// </returns>

public RecepitDAO GetRecepit(int id)

{

var list = from element in \_database.GetTable<RecepitDAO>() where element.ReceiptId == id select element;

return list.First();

}

/// <summary>

/// Gets the recepit.

/// </summary>

/// <param name="recepit">The recepit.</param>

/// <returns></returns>

public RecepitDAO GetRecepit(RecepitDAO recepit)

{

return GetRecepit(recepit.ReceiptId);

}

/// <summary>

/// Inserts the specified add receipt.

/// </summary>

/// <param name="addReceipt">The add receipt.</param>

/// <returns></returns>

public bool Insert(RecepitDAO addReceipt)

{

\_database.GetTable<RecepitDAO>().InsertOnSubmit(addReceipt);

\_database.SubmitChanges();

var recepits = GetAllRecepits();

addReceipt.ReceiptId = recepits[recepits.Length - 1].ReceiptId;

return true;

}

/// <summary>

/// Updates the specified edit recepit.

/// </summary>

/// <param name="editRecepit">The edit recepit.</param>

/// <param name="newRecepit">The new recepit.</param>

/// <returns></returns>

public bool Update(RecepitDAO editRecepit, RecepitDAO newRecepit)

{

\_database.GetTable<RecepitDAO>().DeleteOnSubmit(editRecepit);

\_database.GetTable<RecepitDAO>().InsertOnSubmit(newRecepit);

\_database.SubmitChanges();

var recepits = GetAllRecepits();

newRecepit.ReceiptId = recepits[recepits.Length - 1].ReceiptId;

return true;

}

/// <summary>

/// Deletes the specified delete recepit.

/// </summary>

/// <param name="deleteRecepit">The delete recepit.</param>

/// <returns></returns>

public bool Delete(RecepitDAO deleteRecepit)

{

\_database.GetTable<RecepitDAO>().DeleteOnSubmit(deleteRecepit);

\_database.SubmitChanges();

return true;

}

}

}

Листинг класса Servicez.

using RecepitLibrary.DAO;

using System;

using System.Collections.Generic;

using System.Data.Linq;

using System.Data.SqlClient;

using System.Globalization;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace RecepitLibrary.LINQ

{

/// <summary>

/// Servicez

/// </summary>

/// <seealso cref="RecepitLibrary.DAO.IServicez" />

public class Servicez : IServicez

{

/// <summary>

/// The database

/// </summary>

private DataContext \_database;

/// <summary>

/// Конструктор класса

/// </summary>

/// <param name="connectionString">Строка подключения</param>

public Servicez(string connectionString)

{

\_database = new DataContext(connectionString);

}

/// <summary>

/// Метод, возращающий все записи операций

/// </summary>

/// <returns>

/// Массив записей

/// </returns>

public ServicezDAO[] GetAllServicez()

{

var list= from elements in \_database.GetTable<ServicezDAO>() select elements;

List<ServicezDAO> servicezs = new List<ServicezDAO>();

foreach (ServicezDAO servicez in list)

{

servicezs.Add(servicez);

}

return servicezs.ToArray();

}

/// <summary>

/// Метод, возращающий запись операции

/// </summary>

/// <param name="id">Индефикатор записи</param>

/// <returns>

/// Запись

/// </returns>

public ServicezDAO GetServicez(int id)

{

var list = from elements in \_database.GetTable<ServicezDAO>() where elements.ServiceId == id select elements;

return list.First();

}

/// <summary>

/// Метод, возращающий запись операции

/// </summary>

/// <param name="servicez">Объект записи</param>

/// <returns>

/// Запись

/// </returns>

public ServicezDAO GetServicez(ServicezDAO servicez)

{

return GetServicez(servicez.ServiceId);

}

/// <summary>

/// Inserts the specified add service.

/// </summary>

/// <param name="addService">The add service.</param>

/// <returns></returns>

public bool Insert(ServicezDAO addService)

{

\_database.GetTable<ServicezDAO>().InsertOnSubmit(addService);

\_database.SubmitChanges();

var servicezs = GetAllServicez();

addService.ServiceId = servicezs[servicezs.Length - 1].ServiceId;

return true;

}

/// <summary>

/// Updates the specified edit service.

/// </summary>

/// <param name="editService">The edit service.</param>

/// <param name="newService">The new service.</param>

/// <returns></returns>

public bool Update(ServicezDAO editService, ServicezDAO newService)

{

\_database.GetTable<ServicezDAO>().DeleteOnSubmit(editService);

\_database.GetTable<ServicezDAO>().InsertOnSubmit(newService);

\_database.SubmitChanges();

var breeds = GetAllServicez();

newService.ServiceId = breeds[breeds.Length - 1].ServiceId;

return true;

}

/// <summary>

/// Удаление записи

/// </summary>

/// <param name="deleteServicez">Объект для удаления</param>

/// <returns>

/// Реакци. на добавление

/// </returns>

public bool Delete(ServicezDAO deleteServicez)

{

\_database.GetTable<ServicezDAO>().DeleteOnSubmit(deleteServicez);

\_database.SubmitChanges();

return true;

}

}

}

Листинг класса Standart.

using RecepitLibrary.DAO;

using System;

using System.Collections.Generic;

using System.Data.Linq;

using System.Data.SqlClient;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace RecepitLibrary.LINQ

{

/// <summary>

///

/// </summary>

/// <seealso cref="RecepitLibrary.DAO.IStandarts" />

public class Standart : IStandarts

{

/// <summary>

/// The database

/// </summary>

private DataContext \_database;

/// <summary>

/// Конструктор класса

/// </summary>

/// <param name="connectionString">Строка подключеняия</param>

public Standart(string connectionString)

{

\_database = new DataContext(connectionString);

}

/// <summary>

/// Метод, возращающий все записи стандартов

/// </summary>

/// <returns>

/// Массив записей

/// </returns>

public StandartDAO[] GetAllStandarts()

{

Table<StandartDAO> dao = \_database.GetTable<StandartDAO>();

var list = from elements in dao select elements;

List<StandartDAO> standarts = new List<StandartDAO>();

foreach (StandartDAO standart in list)

{

standarts.Add(standart);

}

return standarts.ToArray();

}

/// <summary>

/// Метод, возращающий запись стандарта

/// </summary>

/// <param name="id">Индефикатор записи</param>

/// <returns>

/// Запись

/// </returns>

public StandartDAO GetStandart(int id)

{

var list = from elements in \_database.GetTable<StandartDAO>() where elements.Id == id select elements;

return list.First();

}

/// <summary>

/// Метод, возращающий запись стандарта

/// </summary>

/// <param name="standart">Объект записи</param>

/// <returns>

/// Запись

/// </returns>

public StandartDAO GetStandart(StandartDAO standart)

{

return GetStandart(standart.Id);

}

/// <summary>

/// Вставка записи

/// </summary>

/// <param name="addStandart">Объект для добавления записи</param>

/// <returns>

/// Реакция на добавление

/// </returns>

public bool Insert(StandartDAO addStandart)

{

\_database.GetTable<StandartDAO>().InsertOnSubmit(addStandart);

\_database.SubmitChanges();

var standarts = GetAllStandarts();

addStandart.Id = standarts[standarts.Length - 1].Id;

return true;

}

/// <summary>

/// Обновление записи

/// </summary>

/// <param name="editStandart">Редактируемый объект</param>

/// <param name="newStandart">Новый объект</param>

/// <returns>

/// Реакция на добавление

/// </returns>

public bool Update(StandartDAO editStandart, StandartDAO newStandart)

{

\_database.GetTable<StandartDAO>().DeleteOnSubmit(editStandart);

\_database.GetTable<StandartDAO>().InsertOnSubmit(newStandart);

\_database.SubmitChanges();

var standarts = GetAllStandarts();

newStandart.Id = standarts[standarts.Length - 1].Id;

return true;

}

/// <summary>

/// Удаление записи

/// </summary>

/// <param name="deleteStandart">Объект для удаления</param>

/// <returns>

/// Реакция на добавление

/// </returns>

public bool Delete(StandartDAO deleteStandart)

{

\_database.GetTable<StandartDAO>().DeleteOnSubmit(deleteStandart);

\_database.SubmitChanges();

return true;

}

}

}

Листинг класса FactoryDAO.

using RecepitLibrary.ADO;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace RecepitLibrary.DAO

{

/// <summary>

/// FactoryDAO

/// </summary>

public abstract class FactoryDAO

{

/// <summary>

/// Gets the factory.

/// </summary>

/// <param name="connectionString">The connection string.</param>

/// <returns></returns>

public static FactoryDAO GetFactory(string connectionString)

{

return new Factory(connectionString);

}

/// <summary>

/// Gets the standarts.

/// </summary>

/// <returns></returns>

public abstract IStandarts GetStandarts();

/// <summary>

/// Gets the servicez.

/// </summary>

/// <returns></returns>

public abstract IServicez GetServicez();

/// <summary>

/// Gets the recepits.

/// </summary>

/// <returns></returns>

public abstract IRecepit GetRecepits();

}

}

Листинг класса IRecepit.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace RecepitLibrary.DAO

{

public interface IRecepit

{

/// <summary>

/// Метод, возвращающий все записи квитанции

/// </summary>

/// <returns>Массив записей</returns>

RecepitDAO[] GetAllRecepits();

/// <summary>

/// Метод, возвращающий запись квитанции

/// </summary>

/// <param name="id">Индефикатор записи</param>

/// <returns>Запись</returns>

RecepitDAO GetRecepit(int id);

/// <summary>

/// Метод, возвращающий запись квитанции

/// </summary>

/// <param name="receipt">Объект записи</param>

/// <returns>Запись</returns>

RecepitDAO GetRecepit(RecepitDAO receipt);

/// <summary>

/// Вставка записи

/// </summary>

/// <param name="addStandart">Объект для добавления записи</param>

/// <returns>Реакци. на добавление</returns>

bool Insert(RecepitDAO addStandart);

/// <summary>

/// Обновление записи

/// </summary>

/// <param name="editReceipt">Редактируемый объект</param>

/// <param name="newReceipt">Новый объект</param>

/// <returns>Реакция на добавление</returns>

bool Update(RecepitDAO editReceipt, RecepitDAO newReceipt);

/// <summary>

/// Удаление рецепта

/// </summary>

/// <param name="deleteReceipt">Объект для удаления</param>

/// <returns>Реакция на добавление</returns>

bool Delete(RecepitDAO deleteReceipt);

}

}

Листинг класса IServicez.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace RecepitLibrary.DAO

{

public interface IServicez

{

/// <summary>

/// Метод, возращающий все записи операций

/// </summary>

/// <returns>Массив записей</returns>

ServicezDAO[] GetAllServicez();

/// <summary>

/// Метод, возращающий запись операции

/// </summary>

/// <param name="id">Индефикатор записи</param>

/// <returns>Запись</returns>

ServicezDAO GetServicez(int id);

/// <summary>

/// Метод, возращающий запись операции

/// </summary>

/// <param name="servicez">Объект записи</param>

/// <returns>Запись</returns>

ServicezDAO GetServicez(ServicezDAO servicez);

/// <summary>

/// Вставка записи

/// </summary>

/// <param name="addServicez">Объект для добавления записи</param>

/// <returns>Реакция на добавление</returns>

bool Insert(ServicezDAO addServicez);

/// <summary>

/// Обновление записи

/// </summary>

/// <param name="editServicez">Редактируемый объект</param>

/// <param name="newServicez">Новый объект</param>

/// <returns>Реакци. на добавление</returns>

bool Update(ServicezDAO editServicez, ServicezDAO newServicez);

/// <summary>

/// Удаление записи

/// </summary>

/// <param name="deleteServicez">Объект для удаления</param>

/// <returns>Реакци. на добавление</returns>

bool Delete(ServicezDAO deleteServicez);

}

}

Листинг класса IStandarts.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace RecepitLibrary.DAO

{

public interface IStandarts

{

/// <summary>

/// Метод, возращающий все записи стандартов

/// </summary>

/// <returns>Массив записей</returns>

StandartDAO[] GetAllStandarts();

/// <summary>

/// Метод, возращающий запись стандарта

/// </summary>

/// <param name="id">Индефикатор записи</param>

/// <returns>Запись</returns>

StandartDAO GetStandart(int id);

/// <summary>

/// Метод, возращающий запись стандарта

/// </summary>

/// <param name="standart">Объект записи</param>

/// <returns>Запись</returns>

StandartDAO GetStandart(StandartDAO standart);

/// <summary>

/// Вставка записи

/// </summary>

/// <param name="addStandart">Объект для добавления записи</param>

/// <returns>Реакци. на добавление</returns>

bool Insert(StandartDAO addStandart);

/// <summary>

/// Обновление записи

/// </summary>

/// <param name="editStandart">Редактируемый объект</param>

/// <param name="newStandart">Новый объект</param>

/// <returns>Реакци. на добавление</returns>

bool Update(StandartDAO editStandart, StandartDAO newStandart);

/// <summary>

/// Удаление записи

/// </summary>

/// <param name="deleteStandart">Объект для удаления</param>

/// <returns>Реакци. на добавление</returns>

bool Delete(StandartDAO deleteStandart);

}

}

Листинг класса RecepitDAO.

using System;

using System.Data.Linq.Mapping;

namespace RecepitLibrary.DAO

{

/// <summary>

/// RecepitDAO

/// </summary>

/// <seealso cref="System.IComparable{RecepitLibrary.DAO.RecepitDAO}" />

[Table(Name = "Receipt")]

public class RecepitDAO : IComparable<RecepitDAO>

{

/// <summary>

/// Gets or sets the receipt identifier.

/// </summary>

/// <value>

/// The receipt identifier.

/// </value>

[Column(Name = "ReceiptId", IsPrimaryKey = true, IsDbGenerated = true, DbType = "int")]

public int ReceiptId { get; set; }

/// <summary>

/// Gets or sets the receipt number.

/// </summary>

/// <value>

/// The receipt number.

/// </value>

[Column(Name = "ReceiptNumber", DbType = "int")]

public int ReceiptNumber { get; set; }

/// <summary>

/// Gets or sets the service identifier.

/// </summary>

/// <value>

/// The service identifier.

/// </value>

[Column(Name = "ServiceId", DbType = "int")]

public int ServiceId { get; set; }

/// <summary>

/// Сравнивает текущий экземпляр с другим объектом того же типа и возвращает целое число, которое показывает, расположен ли текущий экземпляр перед, после или на той же позиции в порядке сортировки, что и другой объект.

/// </summary>

/// <param name="other">Объект для сравнения с данным экземпляром.</param>

/// <returns>

/// Значение, указывающее, каков относительный порядок сравниваемых объектов.

/// Возвращаемые значения представляют следующие результаты сравнения.

/// Значение

/// Значение

/// Меньше нуля

/// Данный экземпляр предшествует параметру <paramref name="other" /> в порядке сортировки.

/// Нуль

/// В той же позиции в порядке сортировки, что происходит этот экземпляр <paramref name="other" />.

/// Больше нуля

/// Данный экземпляр следует за параметром <paramref name="other" /> в порядке сортировки.

/// </returns>

public int CompareTo(RecepitDAO other)

{

return ReceiptId.CompareTo(other.ReceiptId);

}

/// <summary>

/// Determines whether the specified <see cref="System.Object" />, is equal to this instance.

/// </summary>

/// <param name="obj">The <see cref="System.Object" /> to compare with this instance.</param>

/// <returns>

/// <c>true</c> if the specified <see cref="System.Object" /> is equal to this instance; otherwise, <c>false</c>.

/// </returns>

public override bool Equals(object obj)

{

return obj is RecepitDAO dAO &&

ReceiptId == dAO.ReceiptId &&

ReceiptNumber == dAO.ReceiptNumber &&

ServiceId == dAO.ServiceId;

}

/// <summary>

/// Returns a hash code for this instance.

/// </summary>

/// <returns>

/// A hash code for this instance, suitable for use in hashing algorithms and data structures like a hash table.

/// </returns>

public override int GetHashCode()

{

var hashCode = 153695222;

hashCode = hashCode \* -1521134295 + ReceiptId.GetHashCode();

hashCode = hashCode \* -1521134295 + ReceiptNumber.GetHashCode();

hashCode = hashCode \* -1521134295 + ServiceId.GetHashCode();

return hashCode;

}

}

}

Листинг класса ServicezDAO.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Data.Linq.Mapping;

namespace RecepitLibrary.DAO

{

/// <summary>

/// ServicezDAO

/// </summary>

/// <seealso cref="System.IComparable{RecepitLibrary.DAO.ServicezDAO}" />

[Table(Name = "Servicez")]

public class ServicezDAO : IComparable<ServicezDAO>

{

/// <summary>

/// Gets or sets the service identifier.

/// </summary>

/// <value>

/// The service identifier.

/// </value>

[Column(Name = "ServiceId", IsPrimaryKey = true, IsDbGenerated = true, DbType = "int")]

public int ServiceId { get; set; }

/// <summary>

/// Gets or sets the name of the servicez.

/// </summary>

/// <value>

/// The name of the servicez.

/// </value>

[Column(Name = "ServicezName", DbType = "varchar(50)")]

public string ServicezName { get; set; }

/// <summary>

/// Gets or sets the standart identifier.

/// </summary>

/// <value>

/// The standart identifier.

/// </value>

[Column(Name = "StandartId", DbType = "int")]

public int StandartId { get; set; }

/// <summary>

/// Сравнивает текущий экземпляр с другим объектом того же типа и возвращает целое число, которое показывает, расположен ли текущий экземпляр перед, после или на той же позиции в порядке сортировки, что и другой объект.

/// </summary>

/// <param name="other">Объект для сравнения с данным экземпляром.</param>

/// <returns>

/// Значение, указывающее, каков относительный порядок сравниваемых объектов.

/// Возвращаемые значения представляют следующие результаты сравнения.

/// Значение

/// Значение

/// Меньше нуля

/// Данный экземпляр предшествует параметру <paramref name="other" /> в порядке сортировки.

/// Нуль

/// В той же позиции в порядке сортировки, что происходит этот экземпляр <paramref name="other" />.

/// Больше нуля

/// Данный экземпляр следует за параметром <paramref name="other" /> в порядке сортировки.

/// </returns>

public int CompareTo(ServicezDAO other)

{

return ServiceId.CompareTo(other.ServiceId);

}

/// <summary>

/// Determines whether the specified <see cref="System.Object" />, is equal to this instance.

/// </summary>

/// <param name="obj">The <see cref="System.Object" /> to compare with this instance.</param>

/// <returns>

/// <c>true</c> if the specified <see cref="System.Object" /> is equal to this instance; otherwise, <c>false</c>.

/// </returns>

public override bool Equals(object obj)

{

return obj is ServicezDAO dAO &&

ServiceId == dAO.ServiceId &&

ServicezName == dAO.ServicezName &&

StandartId == dAO.StandartId;

}

/// <summary>

/// Returns a hash code for this instance.

/// </summary>

/// <returns>

/// A hash code for this instance, suitable for use in hashing algorithms and data structures like a hash table.

/// </returns>

public override int GetHashCode()

{

var hashCode = 153695222;

hashCode = hashCode \* -1521134295 + ServiceId.GetHashCode();

hashCode = hashCode \* -1521134295 + EqualityComparer<string>.Default.GetHashCode(ServicezName);

hashCode = hashCode \* -1521134295 + StandartId.GetHashCode();

return hashCode;

}

}

}

Листинг класса StandartDAO.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Data.Linq.Mapping;

namespace RecepitLibrary.DAO

{

/// <summary>

/// StandartDAO

/// </summary>

/// <seealso cref="System.IComparable{RecepitLibrary.DAO.StandartDAO}" />

[Table(Name = "Standarts")]

public class StandartDAO : IComparable<StandartDAO>

{

/// <summary>

/// Gets or sets the identifier.

/// </summary>

/// <value>

/// The identifier.

/// </value>

[Column(Name = "Id", IsPrimaryKey = true, IsDbGenerated = true, DbType = "int")]

public int Id { get; set; }

/// <summary>

/// Gets or sets the receipt date.

/// </summary>

/// <value>

/// The receipt date.

/// </value>

[Column(Name = "ReceiptDate", DbType = "varchar(50)")]

public string ReceiptDate { get; set; }

/// <summary>

/// Gets or sets the summary.

/// </summary>

/// <value>

/// The summary.

/// </value>

[Column(Name = "Summary", DbType = "float")]

public float Summary { get; set; }

/// <summary>

/// Сравнивает текущий экземпляр с другим объектом того же типа и возвращает целое число, которое показывает, расположен ли текущий экземпляр перед, после или на той же позиции в порядке сортировки, что и другой объект.

/// </summary>

/// <param name="other">Объект для сравнения с данным экземпляром.</param>

/// <returns>

/// Значение, указывающее, каков относительный порядок сравниваемых объектов.

/// Возвращаемые значения представляют следующие результаты сравнения.

/// Значение

/// Значение

/// Меньше нуля

/// Данный экземпляр предшествует параметру <paramref name="other" /> в порядке сортировки.

/// Нуль

/// В той же позиции в порядке сортировки, что происходит этот экземпляр <paramref name="other" />.

/// Больше нуля

/// Данный экземпляр следует за параметром <paramref name="other" /> в порядке сортировки.

/// </returns>

public int CompareTo(StandartDAO other)

{

return Id.CompareTo(other.Id);

}

/// <summary>

/// Determines whether the specified <see cref="System.Object" />, is equal to this instance.

/// </summary>

/// <param name="obj">The <see cref="System.Object" /> to compare with this instance.</param>

/// <returns>

/// <c>true</c> if the specified <see cref="System.Object" /> is equal to this instance; otherwise, <c>false</c>.

/// </returns>

public override bool Equals(object obj)

{

return obj is StandartDAO dAO &&

Summary == dAO.Summary &&

ReceiptDate == dAO.ReceiptDate &&

Id == dAO.Id;

}

/// <summary>

/// Returns a hash code for this instance.

/// </summary>

/// <returns>

/// A hash code for this instance, suitable for use in hashing algorithms and data structures like a hash table.

/// </returns>

public override int GetHashCode()

{

var hashCode = 153695222;

hashCode = hashCode \* -1521134295 + Summary.GetHashCode();

hashCode = hashCode \* -1521134295 + EqualityComparer<string>.Default.GetHashCode(ReceiptDate);

hashCode = hashCode \* -1521134295 + Id.GetHashCode();

return hashCode;

}

}

}

Листинг класса ConnectionString.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace RecepitLibrary

{

/// <summary>

/// ConnectionString

/// </summary>

public static class ConnectionString

{

/// <summary>

/// The connection string

/// </summary>

public static string connectionString = @"Data Source=.\SQLEXPRESS;Initial Catalog=Recepits;Integrated Security=True";

}

}

Листинг класса Record.

using RecepitLibrary.DAO;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace RecepitLibrary

{

/// <summary>

/// Record

/// </summary>

/// <seealso cref="System.IComparable{RecepitLibrary.Record}" />

public class Record : IComparable<Record>

{

/// <summary>

/// Gets the receipt identifier.

/// </summary>

/// <value>

/// The receipt identifier.

/// </value>

public int ReceiptId { get; private set; }

/// <summary>

/// Gets the name of the servicez.

/// </summary>

/// <value>

/// The name of the servicez.

/// </value>

public string ServicezName { get; private set; }

/// <summary>

/// Gets the service identifier.

/// </summary>

/// <value>

/// The service identifier.

/// </value>

public int ServiceId { get; private set; }

/// <summary>

/// Gets the standart identifier.

/// </summary>

/// <value>

/// The standart identifier.

/// </value>

public int StandartId { get; private set; }

/// <summary>

/// Gets the identifier.

/// </summary>

/// <value>

/// The identifier.

/// </value>

public int Id { get; private set; }

/// <summary>

/// Gets the receipt date.

/// </summary>

/// <value>

/// The receipt date.

/// </value>

public string ReceiptDate { get; private set; }

/// <summary>

/// Gets the summary.

/// </summary>

/// <value>

/// The summary.

/// </value>

public float Summary { get; private set; }

/// <summary>

/// Gets the receipt number.

/// </summary>

/// <value>

/// The receipt number.

/// </value>

public int ReceiptNumber { get; private set; }

/// <summary>

/// Gets the information.

/// </summary>

/// <value>

/// The information.

/// </value>

public string Info { get; private set; }

/// <summary>

/// Initializes a new instance of the <see cref="Record"/> class.

/// </summary>

/// <param name="receipt">The receipt.</param>

/// <param name="service">The service.</param>

/// <param name="standart">The standart.</param>

public Record(RecepitDAO receipt, ServicezDAO service, StandartDAO standart)

{

ReceiptId = receipt.ReceiptId;

ReceiptNumber = receipt.ReceiptNumber;

ServiceId = receipt.ServiceId;

ServicezName = service.ServicezName;

StandartId = service.StandartId;

Summary = standart.Summary;

ReceiptDate = standart.ReceiptDate;

Info = $"№{ReceiptNumber}, Операция: {ServicezName}, Дата: {ReceiptDate}, Сумма: {Summary}.";

}

/// <summary>

/// Сравнивает текущий экземпляр с другим объектом того же типа и возвращает целое число, которое показывает, расположен ли текущий экземпляр перед, после или на той же позиции в порядке сортировки, что и другой объект.

/// </summary>

/// <param name="other">Объект для сравнения с данным экземпляром.</param>

/// <returns>

/// Значение, указывающее, каков относительный порядок сравниваемых объектов.

/// Возвращаемые значения представляют следующие результаты сравнения.

/// Значение

/// Значение

/// Меньше нуля

/// Данный экземпляр предшествует параметру <paramref name="other" /> в порядке сортировки.

/// Нуль

/// В той же позиции в порядке сортировки, что происходит этот экземпляр <paramref name="other" />.

/// Больше нуля

/// Данный экземпляр следует за параметром <paramref name="other" /> в порядке сортировки.

/// </returns>

public int CompareTo(Record other)

{

return ReceiptId.CompareTo(other.ReceiptId);

}

/// <summary>

/// Determines whether the specified <see cref="System.Object" />, is equal to this instance.

/// </summary>

/// <param name="obj">The <see cref="System.Object" /> to compare with this instance.</param>

/// <returns>

/// <c>true</c> if the specified <see cref="System.Object" /> is equal to this instance; otherwise, <c>false</c>.

/// </returns>

public override bool Equals(object obj)

{

return obj is Record record &&

ReceiptId == record.ReceiptId &&

ReceiptNumber == record.ReceiptNumber &&

ServicezName == record.ServicezName &&

ReceiptDate == record.ReceiptDate &&

Id == record.Id &&

ServiceId == record.ServiceId &&

Summary == record.Summary;

}

/// <summary>

/// Returns a hash code for this instance.

/// </summary>

/// <returns>

/// A hash code for this instance, suitable for use in hashing algorithms and data structures like a hash table.

/// </returns>

public override int GetHashCode()

{

var hashCode = -2130739023;

hashCode = hashCode \* -1521134295 + Id.GetHashCode();

hashCode = hashCode \* -1521134295 + ServiceId.GetHashCode();

hashCode = hashCode \* -1521134295 + EqualityComparer<string>.Default.GetHashCode(ServicezName);

hashCode = hashCode \* -1521134295 + Summary.GetHashCode();

hashCode = hashCode \* -1521134295 + EqualityComparer<string>.Default.GetHashCode(ReceiptDate);

hashCode = hashCode \* -1521134295 + ReceiptNumber.GetHashCode();

hashCode = hashCode \* -1521134295 + ReceiptId.GetHashCode();

return hashCode;

}

}

}

Листинг класса StandartDAO.

using RecepitLibrary.DAO;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace RecepitLibrary

{

/// <summary>

/// RecordServicez

/// </summary>

/// <seealso cref="System.IComparable{RecepitLibrary.RecordServicez}" />

public class RecordServicez : IComparable<RecordServicez>

{

/// <summary>

/// Gets the service identifier.

/// </summary>

/// <value>

/// The service identifier.

/// </value>

public int ServiceId { get; private set; }

/// <summary>

/// Gets the standart identifier.

/// </summary>

/// <value>

/// The standart identifier.

/// </value>

public int StandartId { get; private set; }

/// <summary>

/// Gets the name of the servicez.

/// </summary>

/// <value>

/// The name of the servicez.

/// </value>

public string ServicezName { get; private set; }

/// <summary>

/// Gets the receipt date.

/// </summary>

/// <value>

/// The receipt date.

/// </value>

public string ReceiptDate { get; private set; }

/// <summary>

/// Gets the summary.

/// </summary>

/// <value>

/// The summary.

/// </value>

public float Summary { get; private set; }

/// <summary>

/// Gets the information.

/// </summary>

/// <value>

/// The information.

/// </value>

public string Info { get; private set; }

/// <summary>

/// Initializes a new instance of the <see cref="RecordServicez"/> class.

/// </summary>

/// <param name="breed">The breed.</param>

/// <param name="standart">The standart.</param>

public RecordServicez(ServicezDAO breed, StandartDAO standart)

{

ServiceId = breed.ServiceId;

ServicezName = breed.ServicezName;

StandartId = breed.StandartId;

ReceiptDate = standart.ReceiptDate;

Summary = standart.Summary;

Info = $"{ServicezName}, {ReceiptDate}, {Summary} ";

}

/// <summary>

/// Сравнивает текущий экземпляр с другим объектом того же типа и возвращает целое число, которое показывает, расположен ли текущий экземпляр перед, после или на той же позиции в порядке сортировки, что и другой объект.

/// </summary>

/// <param name="other">Объект для сравнения с данным экземпляром.</param>

/// <returns>

/// Значение, указывающее, каков относительный порядок сравниваемых объектов.

/// Возвращаемые значения представляют следующие результаты сравнения.

/// Значение

/// Значение

/// Меньше нуля

/// Данный экземпляр предшествует параметру <paramref name="other" /> в порядке сортировки.

/// Нуль

/// В той же позиции в порядке сортировки, что происходит этот экземпляр <paramref name="other" />.

/// Больше нуля

/// Данный экземпляр следует за параметром <paramref name="other" /> в порядке сортировки.

/// </returns>

public int CompareTo(RecordServicez other)

{

return ServiceId.CompareTo(other.ServiceId);

}

/// <summary>

/// Determines whether the specified <see cref="System.Object" />, is equal to this instance.

/// </summary>

/// <param name="obj">The <see cref="System.Object" /> to compare with this instance.</param>

/// <returns>

/// <c>true</c> if the specified <see cref="System.Object" /> is equal to this instance; otherwise, <c>false</c>.

/// </returns>

public override bool Equals(object obj)

{

return obj is RecordServicez breed &&

ServiceId == breed.ServiceId &&

ServicezName == breed.ServicezName &&

StandartId == breed.StandartId &&

ReceiptDate == breed.ReceiptDate &&

Summary == breed.Summary;

}

/// <summary>

/// Returns a hash code for this instance.

/// </summary>

/// <returns>

/// A hash code for this instance, suitable for use in hashing algorithms and data structures like a hash table.

/// </returns>

public override int GetHashCode()

{

var hashCode = -1062094777;

hashCode = hashCode \* -1521134295 + ServiceId.GetHashCode();

hashCode = hashCode \* -1521134295 + EqualityComparer<string>.Default.GetHashCode(ServicezName);

hashCode = hashCode \* -1521134295 + StandartId.GetHashCode();

hashCode = hashCode \* -1521134295 + Summary.GetHashCode();

hashCode = hashCode \* -1521134295 + EqualityComparer<string>.Default.GetHashCode(ReceiptDate);

return hashCode;

}

}

}

Листинг класса UnifyingRecepit.

using RecepitLibrary.DAO;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace RecepitLibrary

{

/// <summary>

/// UnifyingRecepit

/// </summary>

public class UnifyingRecepit

{

/// <summary>

/// Gets the standarts interface.

/// </summary>

/// <value>

/// The standarts interface.

/// </value>

public IStandarts StandartsInterface { get; private set; }

/// <summary>

/// Gets the recepit interface.

/// </summary>

/// <value>

/// The recepit interface.

/// </value>

public IRecepit RecepitInterface { get; private set; }

/// <summary>

/// Gets the servicez interface.

/// </summary>

/// <value>

/// The servicez interface.

/// </value>

public IServicez ServicezInterface { get; private set; }

/// <summary>

/// Initializes a new instance of the <see cref="UnifyingRecepit"/> class.

/// </summary>

/// <param name="factory">The factory.</param>

public UnifyingRecepit(FactoryDAO factory)

{

RecepitInterface = factory.GetRecepits();

ServicezInterface = factory.GetServicez();

StandartsInterface = factory.GetStandarts();

}

/// <summary>

/// Gets all records.

/// </summary>

/// <returns></returns>

public List<Record> GetAllRecords()

{

List<Record> records = new List<Record>();

var recepits = RecepitInterface.GetAllRecepits();

foreach (var recepit in recepits)

{

var servicez = ServicezInterface.GetServicez(recepit.ServiceId);

var standart = StandartsInterface.GetStandart(servicez.StandartId);

records.Add(new Record(recepit, servicez, standart));

}

return records;

}

/// <summary>

/// Gets all servicez.

/// </summary>

/// <returns></returns>

public List<RecordServicez> GetAllServicez()

{

List<RecordServicez> records = new List<RecordServicez>();

var servicezs = ServicezInterface.GetAllServicez();

foreach (var servicez in servicezs)

{

var standart = StandartsInterface.GetStandart(servicez.StandartId);

records.Add(new RecordServicez(servicez, standart));

}

return records;

}

}

}

Листинг класса MainWindow.

using RecepitLibrary;

using RecepitLibrary.DAO;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace WpfApp1

{

/// <summary>

/// Логика взаимодействия для MainWindow.xaml

/// </summary>

/// <seealso cref="System.Windows.Window" />

/// <seealso cref="System.Windows.Markup.IComponentConnector" />

public partial class MainWindow : Window

{

/// <summary>

/// The path connection string

/// </summary>

private string PATH\_CONNECTION\_STRING = ConnectionString.connectionString;

/// <summary>

/// The accounting

/// </summary>

private UnifyingRecepit accounting;

/// <summary>

/// The is database loaded

/// </summary>

private bool IsDbLoaded = false;

/// <summary>

/// The view mode

/// </summary>

private ViewMode viewMode;

/// <summary>

/// Initializes a new instance of the <see cref="MainWindow"/> class.

/// </summary>

public MainWindow()

{

InitializeComponent();

}

/// <summary>

/// Recepits the viewer.

/// </summary>

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

/// <param name="e">The <see cref="RoutedEventArgs"/> instance containing the event data.</param>

private void RecepitViewer(object sender, RoutedEventArgs e)

{

}

/// <summary>

/// Shows all records.

/// </summary>

private void ShowAllRecords()

{

if (!IsDbLoaded) return;

ViewOfRecords.Items.Clear();

switch (viewMode)

{

case ViewMode.Receipt:

List<Record> records = accounting.GetAllRecords();

foreach (Record record in records)

{

ViewOfRecords.Items.Add(record);

}

break;

case ViewMode.Servicez:

List<RecordServicez> recordServicezs = accounting.GetAllServicez();

foreach (RecordServicez record in recordServicezs)

{

ViewOfRecords.Items.Add(record);

}

break;

}

}

/// <summary>

/// Connects to database.

/// </summary>

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

/// <param name="e">The <see cref="RoutedEventArgs"/> instance containing the event data.</param>

private void ConnectToDatabase(object sender, RoutedEventArgs e)

{

FactoryDAO factory = FactoryDAO.GetFactory(PATH\_CONNECTION\_STRING);

accounting = new UnifyingRecepit(factory);

accounting.RecepitInterface.GetAllRecepits();

IsDbLoaded = true;

ShowAllRecords();

LoadServicez();

OutputTextBox.Text += "Интерфейс подключен к базе данных." + Environment.NewLine;

}

/// <summary>

/// Handles the SelectionChanged event of the Recepit control.

/// </summary>

/// <param name="sender">The source of the event.</param>

/// <param name="e">The <see cref="SelectionChangedEventArgs"/> instance containing the event data.</param>

private void Recepit\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

}

/// <summary>

/// Handles the SelectionChanged event of the ViewOfRecords control.

/// </summary>

/// <param name="sender">The source of the event.</param>

/// <param name="e">The <see cref="SelectionChangedEventArgs"/> instance containing the event data.</param>

private void ViewOfRecords\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

}

/// <summary>

/// Loads the servicez.

/// </summary>

private void LoadServicez()

{

if (!IsDbLoaded) return;

RecepitComboBox.Items.Clear();

var servicezs = accounting.ServicezInterface.GetAllServicez();

foreach (var servicez in servicezs)

{

RecepitComboBox.Items.Add(servicez);

}

RecepitComboBox.SelectedIndex = 0;

}

/// <summary>

/// Adds the record.

/// </summary>

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

/// <param name="e">The <see cref="RoutedEventArgs"/> instance containing the event data.</param>

private void AddRecord(object sender, RoutedEventArgs e)

{

if (!IsDbLoaded)

{

MessageBox.Show("База данных не загружена");

return;

}

if (viewMode != ViewMode.Receipt)

{

MessageBox.Show("Индекс принадлежит представлению опрерации");

return;

}

if (NameInputTextBox.Text == "" || DateInputTextBox.Text == "")

{

MessageBox.Show("Неверные данные");

return;

}

RecepitDAO recepit = new RecepitDAO()

{

ReceiptNumber = Convert.ToInt32(NameInputTextBox.Text),

ServiceId = (RecepitComboBox.SelectedItem as ServicezDAO).ServiceId,

};

accounting.RecepitInterface.Insert(recepit);

ShowAllRecords();

OutputTextBox.Text += "Запись добавлена." + Environment.NewLine;

}

/// <summary>

/// Edits the record.

/// </summary>

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

/// <param name="e">The <see cref="RoutedEventArgs"/> instance containing the event data.</param>

private void EditRecord(object sender, RoutedEventArgs e)

{

if (!IsDbLoaded)

{

MessageBox.Show("База данных не загружена");

return;

}

if (viewMode != ViewMode.Receipt)

{

MessageBox.Show("Индекс принадлежит представлениию квитанции");

return;

}

if (NameInputTextBox.Text == "" || DateInputTextBox.Text == "")

{

MessageBox.Show("Неверные данные.");

return;

}

if (ViewOfRecords.SelectedValue == null)

{

MessageBox.Show("Объект для изменения не выбран.");

return;

}

RecepitDAO newReceipt = new RecepitDAO()

{

ReceiptNumber = Convert.ToInt32(NameInputTextBox.Text),

ServiceId = (RecepitComboBox.SelectedItem as ServicezDAO).ServiceId,

};

RecepitDAO oldReceipt = accounting.RecepitInterface.GetRecepit((ViewOfRecords.SelectedValue as Record).ReceiptId);

accounting.RecepitInterface.Update(oldReceipt, newReceipt);

ShowAllRecords();

OutputTextBox.Text += "Запись отредактирована." + Environment.NewLine;

}

/// <summary>

/// Deletes the record.

/// </summary>

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

/// <param name="e">The <see cref="RoutedEventArgs"/> instance containing the event data.</param>

private void DeleteRecord(object sender, RoutedEventArgs e)

{

if (!IsDbLoaded)

{

MessageBox.Show("База данных не загружена");

return;

}

if (viewMode != ViewMode.Receipt)

{

MessageBox.Show("Индекс принадлежит представления операции");

return;

}

if (ViewOfRecords.SelectedValue == null)

{

MessageBox.Show("Объект для изменения не выбран.");

return;

}

RecepitDAO recepit = accounting.RecepitInterface.GetRecepit((ViewOfRecords.SelectedValue as Record).ReceiptId);

accounting.RecepitInterface.Delete(recepit);

ShowAllRecords();

OutputTextBox.Text += "Запись удалена." + Environment.NewLine;

}

/// <summary>

/// Handles the TextChanged event of the SummaryTextBox control.

/// </summary>

/// <param name="sender">The source of the event.</param>

/// <param name="e">The <see cref="TextChangedEventArgs"/> instance containing the event data.</param>

private void SummaryTextBox\_TextChanged(object sender, TextChangedEventArgs e)

{

}

/// <summary>

/// Handles the TextChanged event of the NameTextBox control.

/// </summary>

/// <param name="sender">The source of the event.</param>

/// <param name="e">The <see cref="TextChangedEventArgs"/> instance containing the event data.</param>

private void NameTextBox\_TextChanged(object sender, TextChangedEventArgs e)

{

}

/// <summary>

/// Adds the service record.

/// </summary>

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

/// <param name="e">The <see cref="RoutedEventArgs"/> instance containing the event data.</param>

private void AddServiceRecord(object sender, RoutedEventArgs e)

{

if (!IsDbLoaded)

{

MessageBox.Show("База данных не загружена!");

return;

}

if (viewMode != ViewMode.Servicez)

{

MessageBox.Show("Индекс принадлежит представлению операции");

return;

}

if (NameTextBox.Text == "")

{

MessageBox.Show("Неверные или пустые данные. Проверьте поля ввода.");

return;

}

StandartDAO standart = new StandartDAO()

{

Summary = (float)Convert.ToDouble(SummaryTextBox.Text),

ReceiptDate = DateInputTextBox.Text,

};

accounting.StandartsInterface.Insert(standart);

int standartId = accounting.StandartsInterface.GetStandart(standart).Id;

ServicezDAO servicez = new ServicezDAO()

{

ServicezName = NameTextBox.Text,

StandartId = standartId,

};

accounting.ServicezInterface.Insert(servicez);

ShowAllRecords();

LoadServicez();

OutputTextBox.Text += "Запись добавлена." + Environment.NewLine;

}

/// <summary>

/// Changes the view recepits.

/// </summary>

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

/// <param name="e">The <see cref="RoutedEventArgs"/> instance containing the event data.</param>

private void ChangeViewRecepits(object sender, RoutedEventArgs e)

{

viewMode = ViewMode.Receipt;

ShowAllRecords();

}

/// <summary>

/// Changes the view servicez.

/// </summary>

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

/// <param name="e">The <see cref="RoutedEventArgs"/> instance containing the event data.</param>

private void ChangeViewServicez(object sender, RoutedEventArgs e)

{

viewMode = ViewMode.Servicez;

ShowAllRecords();

}

/// <summary>

/// Edits the service record.

/// </summary>

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

/// <param name="e">The <see cref="RoutedEventArgs"/> instance containing the event data.</param>

private void EditServiceRecord(object sender, RoutedEventArgs e)

{

if (!IsDbLoaded)

{

MessageBox.Show("База данных не загружена!");

return;

}

if (viewMode != ViewMode.Servicez)

{

MessageBox.Show("Индекс принадлежит представления квитанции");

return;

}

if (NameTextBox.Text == "" || DateInputTextBox.Text == "")

{

MessageBox.Show("Неверные данные. Проверьте поля ввода.");

return;

}

if (ViewOfRecords.SelectedValue == null)

{

MessageBox.Show("Объект для изменения не выбран.");

return;

}

StandartDAO newStandart = new StandartDAO()

{

Summary = (float)Convert.ToDouble(SummaryTextBox.Text),

ReceiptDate = DateInputTextBox.Text,

};

StandartDAO standart = accounting.StandartsInterface.GetStandart((ViewOfRecords.SelectedValue as RecordServicez).StandartId);

accounting.StandartsInterface.Update(standart, newStandart);

int standartId = accounting.StandartsInterface.GetStandart(newStandart).Id;

ServicezDAO newServicez = new ServicezDAO()

{

ServicezName = NameTextBox.Text,

StandartId = standartId,

};

ServicezDAO oldServicez = accounting.ServicezInterface.GetServicez((ViewOfRecords.SelectedValue as RecordServicez).ServiceId);

accounting.ServicezInterface.Update(oldServicez, newServicez);

ShowAllRecords();

LoadServicez();

OutputTextBox.Text += "Запись изменена." + Environment.NewLine;

}

/// <summary>

/// Deletes the service record.

/// </summary>

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

/// <param name="e">The <see cref="RoutedEventArgs"/> instance containing the event data.</param>

private void DeleteServiceRecord(object sender, RoutedEventArgs e)

{

if (!IsDbLoaded)

{

MessageBox.Show("База данных не загружена");

return;

}

if (viewMode != ViewMode.Servicez)

{

MessageBox.Show("Индекс принадлежит представления квитанции");

return;

}

if (ViewOfRecords.SelectedValue == null)

{

MessageBox.Show("Объект для изменения не выбран.");

return;

}

ServicezDAO servicez = accounting.ServicezInterface.GetServicez((ViewOfRecords.SelectedValue as RecordServicez).ServiceId);

accounting.ServicezInterface.Delete(servicez);

accounting.StandartsInterface.Delete(accounting.StandartsInterface.GetStandart(servicez.StandartId));

ShowAllRecords();

LoadServicez();

OutputTextBox.Text += "Запись удалена." + Environment.NewLine;

}

}

}

Листинг класса EnumViewMode.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace WpfApp1

{

internal enum ViewMode

{

Receipt,

Servicez

}

}

Листинг класса FactoryTests.

using Microsoft.VisualStudio.TestTools.UnitTesting;

using RecepitLibrary.ADO;

using RecepitLibrary.DAO;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace RecepitLibrary.Tests

{

[TestClass()]

public class FactoryTests

{

private static string connectionString = ConnectionString.connectionString;

[TestMethod]

public void TestMethodRead()

{

Recepits recepits = new Recepits(connectionString);

Assert.IsNotNull(recepits.GetAllRecepits());

}

[TestMethod]

public void TestMethodRead1()

{

Recepits recepits = new Recepits(connectionString);

Assert.IsNotNull(recepits.GetRecepit(1));

}

[TestMethod]

public void TestGetRecepit()

{

Recepits recepits = new Recepits(connectionString);

RecepitDAO recepit = new RecepitDAO();

try

{

Assert.IsNotNull(recepits.GetRecepit(recepit));

}

catch(Exception ex)

{

Assert.IsNotNull(ex);

}

}

public void TestInsert()

{

Recepits recepits = new Recepits(connectionString);

RecepitDAO recepit = new RecepitDAO();

Assert.IsFalse(recepits.Insert(recepit));

}

[TestMethod]

public void TestMethodReadServicez()

{

Servicez servicez = new Servicez(connectionString);

Assert.IsNotNull(servicez.GetAllServicez());

}

[TestMethod]

public void TestMethodServicez1()

{

Servicez servicez = new Servicez(connectionString);

Assert.IsNotNull(servicez.GetServicez(1));

}

[TestMethod]

public void TestGetServicez()

{

Servicez servicez = new Servicez(connectionString);

ServicezDAO service = new ServicezDAO();

try

{

Assert.IsNotNull(servicez.GetServicez(service));

}

catch (Exception ex)

{

Assert.IsNotNull(ex);

}

}

[TestMethod]

public void TestMethodReadStandart()

{

Recepits standarts = new Recepits(connectionString);

Assert.IsNotNull(standarts.GetAllRecepits());

}

[TestMethod]

public void TestMethodReadStandart1()

{

Standart standarts = new Standart(connectionString);

Assert.IsNotNull(standarts.GetStandart(1));

}

[TestMethod]

public void TestGetStandart()

{

Standart standarts = new Standart(connectionString);

StandartDAO standart = new StandartDAO();

try

{

Assert.IsNotNull(standarts.GetStandart(standart));

}

catch (Exception ex)

{

Assert.IsNotNull(ex);

}

}

[TestMethod]

public void TestInsertStandart()

{

try

{

Standart standarts = new Standart(connectionString);

StandartDAO standart = new StandartDAO();

Assert.IsFalse(standarts.Insert(standart));

}

catch (Exception ex)

{

Assert.IsNotNull(ex);

}

}

}

}