МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО

Факультет автоматизированных и информационных систем

Кафедра «Информатика»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 1

по дисциплине «ИГИ»

на тему: «Разработка слоя доступа к данным в приложении с многослойной архитектурой»

Выполнил: студент гр. ИП-32

Суховенко Э. С.

Принял: преподаватель

Процкая М.А.

Гомель 2022

**Цель работы.** Разработать модель заданной предметной области, ознакомиться с особенностями архитектуры приложений, получить навыки реализации паттерна «Репозиторий» .

**Задание.**

1 Создать файлы с данными в формате CSV.

2 Создать базу данных не менее чем из трех таблиц и заполнить таблицы данными.

3 Разработать слой DAL (Data Access Layer) в виде библиотеки классов для создаваемой системы в рамках выбранной предметной области.

Библиотека должна содержать:

* Классы сущностей (Entities), моделирующие не менее чем три таблицы базы данных;
* Интерфейсы и классы, реализующие паттерн **Репозиторий** для доступа к данным в файле в формате CSV и к БД с использованием ADO.Net.

2 Разработать консольное приложение (слой UI), предоставляющее пользователю удобный интерфейс для работы с системой. Это приложение должно работать под управлением меню и выполнять следующие функции:

* вывод информации по запросу пользователя;
* добавление информации;
* удаление указанных пользователем данных;
* поиск информации.

Код задания

Program.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using ClassLibrary.Entities;

using ClassLibrary.Interfaces;

using ClassLibrary.Repository;

using ClassLibrary.Repository.SCVsRepository;

namespace App

{

internal class Program

{

static void Main(string[] args)

{

Console.WriteLine("С какими данными вы хотите работать?");

Console.WriteLine("1. CSV\n2. БД (Access)");

int choose = int.Parse(Console.ReadLine());

//while()

//if (int.TryParse(Console.ReadLine(), out int res))

// choose = res;

//else Console.WriteLine("Неправильный ввод");

while (true)

{

switch (choose)

{

case 1:

ChooseTableCSV();

break;

case 2:

ChooseTableDB();

break;

}

}

Console.ReadKey();

}

private static void ChooseTableCSV()

{

Console.WriteLine("Выберите таблицу");

Console.WriteLine("1. Store\n2. Users\n3. Clients");

int choose = int.Parse(Console.ReadLine());

switch (choose)

{

case 1:

IRepository<Store> store = new CsvRepStore();

ChooseOptionStore(store);

break;

case 2:

IRepository<User> user = new CsvRepUser();

ChooseOptionUser(user);

break;

case 3:

IRepository<Client> client = new CsvRepClient();

ChooseOptionClient(client);

break;

}

}

private static void ChooseTableDB()

{

Console.WriteLine("Выберите таблицу");

Console.WriteLine("1. Store\n2. Users\n3. Clients");

int choose = int.Parse(Console.ReadLine());

switch (choose)

{

case 1:

IRepository<Store> store = new RepositoryStore();

ChooseOptionStore(store);

break;

case 2:

IRepository<User> user = new RepositoryUser();

ChooseOptionUser(user);

break;

case 3:

IRepository<Client> client = new RepositoryClient();

ChooseOptionClient(client);

break;

}

}

private static void ChooseOptionStore(IRepository<Store> obj)

{

Console.WriteLine("1. SelectAll\n2. Search\n3. Delete\n4. Insert");

int choose = int.Parse(Console.ReadLine());

switch (choose)

{

case 1:

List<Store> lst = obj.SelectAll();

lst.ForEach(x => Console.WriteLine(x.ToString()));

break;

case 2:

Console.Write("Введите название товара: ");

string inp = Console.ReadLine();

List<Store> res = obj.Search(inp);

if (res.Count == 0)

Console.WriteLine("Данных нет");

else

res.ForEach(x => Console.WriteLine(x.ToString()));

break;

case 3:

Console.Write("Введите id: ");

int id = int.Parse(Console.ReadLine());

obj.Delete(id);

obj.SelectAll().ForEach(x => Console.WriteLine(x.ToString()));

break;

case 4:

Store data = new Store();

Console.Write("NameTovar: ");

data.NameTovar = Console.ReadLine();

Console.WriteLine("Cost: ");

data.Cost = int.Parse(Console.ReadLine());

Console.WriteLine("Count: ");

data.Count = int.Parse(Console.ReadLine());

obj.Insert(data);

break;

}

}

private static void ChooseOptionClient(IRepository<Client> obj)

{

Console.WriteLine("1. SelectAll\n2. Search\n3. Delete\n4. Insert");

int choose = int.Parse(Console.ReadLine());

switch (choose)

{

case 1:

List<Client> lst = obj.SelectAll();

lst.ForEach(x => Console.WriteLine(x.ToString()));

break;

case 2:

Console.Write("Введите clientName: ");

string inp = Console.ReadLine();

List<Client> res = obj.Search(inp);

if (res.Count == 0)

Console.WriteLine("Данных нет");

else

res.ForEach(x => Console.WriteLine(x.ToString()));

break;

case 3:

Console.Write("Введите id: ");

int id = int.Parse(Console.ReadLine());

obj.Delete(id);

obj.SelectAll().ForEach(x => Console.WriteLine(x.ToString()));

break;

case 4:

Client data = new Client();

Console.Write("ClientName: ");

data.ClientName = Console.ReadLine();

Console.Write("Surname: ");

data.Surname = Console.ReadLine();

Console.Write("UserId: ");

data.UserId = int.Parse(Console.ReadLine());

Console.Write("Age");

data.Age = int.Parse(Console.ReadLine());

obj.Insert(data);

break;

}

}

private static void ChooseOptionUser(IRepository<User> obj)

{

Console.WriteLine("1. SelectAll\n2. Search\n3. Delete\n4. Insert");

int choose = int.Parse(Console.ReadLine());

switch (choose)

{

case 1:

List<User> lst = obj.SelectAll();

lst.ForEach(x => Console.WriteLine(x.ToString()));

break;

case 2:

Console.Write("Введите login: ");

string inp = Console.ReadLine();

List<User> res = obj.Search(inp);

if (res.Count == 0)

Console.WriteLine("Данных нет");

else

res.ForEach(x => Console.WriteLine(x.ToString()));

break;

case 3:

Console.Write("Введите id: ");

int id = int.Parse(Console.ReadLine());

obj.Delete(id);

obj.SelectAll().ForEach(x => Console.WriteLine(x.ToString()));

break;

case 4:

User data = new User();

Console.Write("Login: ");

data.Login = Console.ReadLine();

Console.Write("Password: ");

data.Password = Console.ReadLine();

Console.Write("Rank: ");

data.Rank = Console.ReadLine();

obj.Insert(data);

break;

}

}

}

}

GetDb.cs

using System;

using System.Collections.Generic;

using System.Data.OleDb;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ClassLibrary.Connect

{

class GetDb

{

private readonly static string connectString = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=\"D:\\3\_course\_2\_sem\\IGI\\Lab 1\\Database\_Lab1.mdb\"";

private static readonly OleDbConnection myConnection = new OleDbConnection(connectString);

static public OleDbConnection Connection => myConnection;

}

}

CsvWorking.cs

using ClassLibrary.Entities;

using ClassLibrary.Interfaces;

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Reflection;

using System.Text;

using System.Threading.Tasks;

namespace ClassLibrary.CSV

{

public class CsvWorking

{

public void WriteFile<T>(string path, T item)

{

using (FileStream fs = new FileStream(path, FileMode.OpenOrCreate, FileAccess.Write))

{

using (StreamWriter writer = new StreamWriter(fs, Encoding.Default))

{

var propArr = item.GetType().GetProperties(BindingFlags.DeclaredOnly | BindingFlags.Public | BindingFlags.Instance);

string resultString = "";

for (int i = 0; i < propArr.Length; i++)

{

resultString += propArr[i].GetValue(item);

if (i != propArr.Length - 1)

{

resultString += ',';

}

}

writer.WriteLine(resultString);

fs.Seek(0, SeekOrigin.End);

writer.Close();

fs.Close();

}

}

}

public IEnumerable<T> ReadFile<T>(string path) where T : IPropertys, new()

{

using (FileStream fs = new FileStream(path, FileMode.OpenOrCreate, FileAccess.Read))

{

using (StreamReader reader = new StreamReader(fs, Encoding.Default))

{

IEnumerable<T> list = new List<T>();

string line = "";

while ((line = reader.ReadLine()) != null)

{

var valuesArr = line.Split(',');

T item = new T();

item.SetValue(valuesArr);

list.ToList().Add(item);

fs.Seek(0, SeekOrigin.End);

}

reader.Close();

fs.Close();

return list;

}

}

}

}

}

Client.cs

using ClassLibrary.Interfaces;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ClassLibrary.Entities

{

public class Client : IPropertys

{

public int Id { get; set; }

public string ClientName { get; set; }

public string Surname { get; set; }

public int Age { get; set; }

public int UserId { get; set; }

public void SetValue(string[] props)

{

this.Id = int.Parse(props[0]);

this.ClientName = props[1];

this.Surname = props[2];

this.Surname = props[3];

this.Age = int.Parse(props[4]);

this.UserId = int.Parse(props[5]);

}

public override string ToString()

{

return Id + " " + ClientName + " " + Surname + " " + Age + " " + UserId;

}

}

}

Store.cs

using ClassLibrary.Interfaces;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ClassLibrary.Entities

{

public class Store : IPropertys

{

public int Id { get; set; }

public string NameTovar { get; set; }

public double Cost { get; set; }

public double Count { get; set; }

public int ClientId { get; set; }

public void SetValue(string[] props)

{

this.Id = int.Parse(props[0]);

this.NameTovar = props[1];

this.Cost = int.Parse(props[2]);

this.Count = int.Parse(props[3]);

this.ClientId = int.Parse(props[4]);

}

public override string ToString()

{

return Id + " " + NameTovar + " " + Cost + " " + Count + " " + ClientId;

}

}

}

User.cs

using ClassLibrary.Interfaces;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ClassLibrary.Entities

{

public class User : IPropertys

{

public int Id { get; set; }

public string Login {set;get;}

public string Password { set; get; }

public string Rank { get; set; }

public void SetValue(string[] props)

{

this.Id = int.Parse(props[0]);

this.Login = props[1];

this.Password = props[2];

this.Rank = props[3];

}

public override string ToString()

{

return Id + " " + Login + " " + Password + " " + Rank;

}

}

}

IProperys.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ClassLibrary.Interfaces

{

public interface IPropertys

{

void SetValue(string[] props);

}

}

IRepository.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ClassLibrary.Interfaces

{

public interface IRepository<T>

{

void Insert(T entity);

void Delete(int id);

List<T> SelectAll();

List<T> Search(string parametr);

}

}

CsvRepClient.cs

using ClassLibrary.CSV;

using ClassLibrary.Entities;

using ClassLibrary.Interfaces;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ClassLibrary.Repository.SCVsRepository

{

public class CsvRepClient : IRepository<Client>

{

private static readonly string path = @"D:\3\_course\_2\_sem\IGI\Lab 1\CSVs\clients.csv";

public void Delete(int id)

{

CsvWorking csv = new CsvWorking();

var lst = csv.ReadFile<Client>(path);

List<Client> res = new List<Client>();

foreach (var item in lst)

{

if (item.Id != id)

res.Add(item);

}

csv.WriteFile(path, res);

}

public void Insert(Client entity)

{

CsvWorking csv = new CsvWorking();

var lst = csv.ReadFile<Client>(path);

List<Client> res = new List<Client>();

res.Add(entity);

csv.WriteFile(path, res);

}

public List<Client> Search(string clientName)

{

CsvWorking csv = new CsvWorking();

var lst = csv.ReadFile<Client>(path);

var res = new List<Client>();

foreach(var item in lst)

{

if (item.ClientName == clientName)

res.Add(item);

}

return res;

}

public List<Client> SelectAll()

{

CsvWorking csv = new CsvWorking();

var lst = csv.ReadFile<Client>(path);

return lst.ToList();

}

}

}

CsvRepStore.cs

using ClassLibrary.CSV;

using ClassLibrary.Entities;

using ClassLibrary.Interfaces;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ClassLibrary.Repository.SCVsRepository

{

public class CsvRepStore : IRepository<Store>

{

private static readonly string path = @"D:\3\_course\_2\_sem\IGI\Lab 1\CSVs\store.csv";

public void Delete(int id)

{

CsvWorking csv = new CsvWorking();

var lst = csv.ReadFile<Store>(path);

List<Store> res = new List<Store>();

foreach (var item in lst)

{

if (item.Id != id)

res.Add(item);

}

csv.WriteFile(path, res);

}

public void Insert(Store entity)

{

CsvWorking csv = new CsvWorking();

var lst = csv.ReadFile<Store>(path);

List<Store> res = new List<Store>();

res.Add(entity);

csv.WriteFile(path, res);

}

public List<Store> Search(string nameTovar)

{

CsvWorking csv = new CsvWorking();

var lst = csv.ReadFile<Store>(path);

var res = new List<Store>();

foreach (var item in lst)

{

if (item.NameTovar == nameTovar)

res.Add(item);

}

return res;

}

public List<Store> SelectAll()

{

CsvWorking csv = new CsvWorking();

var lst = csv.ReadFile<Store>(path);

return lst.ToList();

}

}

}

CsvRepUser.cs

using ClassLibrary.CSV;

using ClassLibrary.Entities;

using ClassLibrary.Interfaces;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ClassLibrary.Repository.SCVsRepository

{

public class CsvRepUser : IRepository<User>

{

private static readonly string path = @"D:\3\_course\_2\_sem\IGI\Lab 1\CSVs\users.csv";

public void Delete(int id)

{

CsvWorking csv = new CsvWorking();

var lst = csv.ReadFile<User>(path);

List<User> res = new List<User>();

foreach (var item in lst)

{

if (item.Id != id)

res.Add(item);

}

csv.WriteFile(path, res);

}

public void Insert(User entity)

{

CsvWorking csv = new CsvWorking();

var lst = csv.ReadFile<User>(path);

List<User> res = new List<User>();

res.Add(entity);

csv.WriteFile(path, res);

}

public List<User> Search(string login)

{

CsvWorking csv = new CsvWorking();

var lst = csv.ReadFile<User>(path);

var res = new List<User>();

foreach (var item in lst)

{

if (item.Login == login)

res.Add(item);

}

return res;

}

public List<User> SelectAll()

{

CsvWorking csv = new CsvWorking();

var lst = csv.ReadFile<User>(path);

return lst.ToList();

}

}

}

RepositoryClient.cs

using ClassLibrary.Connect;

using ClassLibrary.Entities;

using System;

using System.Collections.Generic;

using System.Data.OleDb;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ClassLibrary.Repository

{

public class RepositoryClient : Interfaces.IRepository<Entities.Client>

{

public void Delete(int id)

{

try

{

GetDb.Connection.Open();

string query = "delete from Clients where Id = @id";

OleDbCommand delete = new OleDbCommand(query, GetDb.Connection);

delete.Parameters.AddWithValue("@id", id);

delete.ExecuteNonQuery();

GetDb.Connection.Close();

}

catch (Exception mes)

{

GetDb.Connection.Close();

throw new Exception(mes.Message);

}

}

public void Insert(Client entity)

{

try

{

GetDb.Connection.Open();

string query = "INSERT INTO Clients (ClientName, Surname, UserId, Age) " +

"VALUES (@cn, @sn, @uId, @age)";

OleDbCommand insert = new OleDbCommand(query, GetDb.Connection);

insert.Parameters.AddWithValue("@cn", entity.ClientName);

insert.Parameters.AddWithValue("@sn", entity.Surname);

insert.Parameters.AddWithValue("@uId", entity.UserId);

insert.Parameters.AddWithValue("@age", entity.Age);

insert.ExecuteNonQuery();

}

catch(Exception mes)

{

throw new Exception(mes.Message);

}

finally

{

GetDb.Connection.Close();

}

}

public List<Client> Search(string clientName)

{

try

{

GetDb.Connection.Open();

List<Client> clients = new List<Client>();

string query = "select \* from Clients where ClientName = @cn";

OleDbCommand find = new OleDbCommand(query, GetDb.Connection);

find.Parameters.AddWithValue("@cn", clientName);

List<Client> lst = new List<Client>();

OleDbDataReader reader = find.ExecuteReader();

if (reader.HasRows)

{

while (reader.Read())

{

Client client = new Client();

client.Id = Convert.ToInt32(reader.GetValue(0));

client.ClientName = reader.GetValue(1).ToString();

client.Surname = reader.GetValue(2).ToString();

client.UserId = Convert.ToInt32(reader.GetValue(3));

client.Age = Convert.ToInt32(reader.GetValue(4));

lst.Add(client);

}

}

return lst;

}

catch(Exception mes)

{

throw new Exception(mes.Message);

}

finally

{

GetDb.Connection.Close();

}

}

public List<Client> SelectAll()

{

try

{

GetDb.Connection.Open();

List<Client> lst = new List<Client>();

string query = "select \* from Clients";

OleDbCommand select = new OleDbCommand(query, GetDb.Connection);

OleDbDataReader reader = select.ExecuteReader();

if (reader.HasRows)

{

while (reader.Read())

{

Client client = new Client();

client.Id = Convert.ToInt32(reader.GetValue(0));

client.ClientName = reader.GetValue(1).ToString();

client.Surname = reader.GetValue(2).ToString();

client.UserId = Convert.ToInt32(reader.GetValue(3));

client.Age = Convert.ToInt32(reader.GetValue(4));

lst.Add(client);

}

}

reader.Close();

return lst;

}

catch (Exception mes)

{

throw new Exception(mes.Message);

}

finally

{

GetDb.Connection.Close();

}

}

}

}

RepositoryStore.cs

using ClassLibrary.Connect;

using ClassLibrary.Entities;

using System;

using System.Collections.Generic;

using System.Data.OleDb;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ClassLibrary.Repository

{

public class RepositoryStore : Interfaces.IRepository<Store>

{

public void Delete(int id)

{

GetDb.Connection.Open();

string query = "delete from Store where Id = @id";

OleDbCommand delete = new OleDbCommand(query, GetDb.Connection);

delete.Parameters.AddWithValue("@id", id);

delete.ExecuteNonQuery();

GetDb.Connection.Close();

}

public void Insert(Store entity)

{

try

{

GetDb.Connection.Open();

string query = "INSERT INTO Store (NameTovar, Cost, Count, ClientId) " +

"VALUES (@nt, @cost, @count, @cId)";

OleDbCommand insert = new OleDbCommand(query, GetDb.Connection);

insert.Parameters.AddWithValue("@nt", entity.NameTovar);

insert.Parameters.AddWithValue("@cost", entity.Cost);

insert.Parameters.AddWithValue("@count", entity.Count);

insert.Parameters.AddWithValue("@cID", entity.ClientId);

insert.ExecuteNonQuery();

GetDb.Connection.Close();

}

catch(Exception mes)

{

GetDb.Connection.Close();

throw new Exception(mes.Message);

}

}

public List<Store> Search(string nameTovar)

{

try

{

GetDb.Connection.Open();

List<Store> stores = new List<Store>();

string query = "select \* from Store where NameTovar = @nt";

OleDbCommand find = new OleDbCommand(query, GetDb.Connection);

find.Parameters.AddWithValue("@nt", nameTovar);

List<Store> lst = new List<Store>();

OleDbDataReader reader = find.ExecuteReader();

if (reader.HasRows)

{

while (reader.Read())

{

Store obj = new Store();

obj.Id = Convert.ToInt32(reader.GetValue(0));

obj.NameTovar = reader.GetValue(1).ToString();

obj.Cost = Convert.ToInt32(reader.GetValue(2));

obj.Count = Convert.ToInt32(reader.GetValue(3));

obj.ClientId = Convert.ToInt32(reader.GetValue(4));

lst.Add(obj);

}

}

reader.Close();

GetDb.Connection.Close();

return lst;

}

catch (Exception mes)

{

GetDb.Connection.Close();

throw new Exception(mes.Message);

}

}

public List<Store> SelectAll()

{

GetDb.Connection.Open();

List<Store> lst = new List<Store>();

string query = "select \* from Store";

OleDbCommand select = new OleDbCommand(query, GetDb.Connection);

OleDbDataReader reader = select.ExecuteReader();

if (reader.HasRows)

{

while (reader.Read())

{

Store obj = new Store();

obj.Id = Convert.ToInt32(reader.GetValue(0));

obj.NameTovar = reader.GetValue(1).ToString();

obj.Cost = Convert.ToInt32(reader.GetValue(2));

obj.Count = Convert.ToInt32(reader.GetValue(3));

obj.ClientId = Convert.ToInt32(reader.GetValue(4));

lst.Add(obj);

}

}

reader.Close();

GetDb.Connection.Close();

return lst;

}

}

}

RepositoryUser.cs

using ClassLibrary.Connect;

using ClassLibrary.Entities;

using System;

using System.Collections.Generic;

using System.Data.OleDb;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ClassLibrary.Repository

{

public class RepositoryUser : Interfaces.IRepository<Entities.User>

{

public void Delete(int id)

{

GetDb.Connection.Open();

string query = "delete from Store where Id = @id";

OleDbCommand delete = new OleDbCommand(query, GetDb.Connection);

delete.Parameters.AddWithValue("@id", id);

delete.ExecuteNonQuery();

GetDb.Connection.Close();

}

public void Insert(User entity)

{

try

{

GetDb.Connection.Open();

string query = "INSERT INTO Users (Login, Password, Rank) " +

"VALUES (@log, @pass, @rank)";

OleDbCommand insert = new OleDbCommand(query, GetDb.Connection);

insert.Parameters.AddWithValue("@log", entity.Login);

insert.Parameters.AddWithValue("@pass", entity.Password);

insert.Parameters.AddWithValue("@rank", entity.Rank);

insert.ExecuteNonQuery();

}

catch(Exception mes)

{

throw new Exception(mes.Message);

}

finally

{

GetDb.Connection.Close();

}

}

public List<User> Search(string login)

{

try

{

GetDb.Connection.Open();

List<User> lst = new List<User>();

string query = "select \* from Users where Login = @log";

OleDbCommand select = new OleDbCommand(query, GetDb.Connection);

select.Parameters.AddWithValue("@log", login);

OleDbDataReader reader = select.ExecuteReader();

if (reader.HasRows)

{

while (reader.Read())

{

User obj = new User();

obj.Id = Convert.ToInt32(reader.GetValue(0));

obj.Login = reader.GetValue(1).ToString();

obj.Password = reader.GetValue(2).ToString();

obj.Rank = reader.GetValue(3).ToString();

lst.Add(obj);

}

}

reader.Close();

return lst;

}

catch(Exception mes) { throw new Exception(mes.Message); }

finally { GetDb.Connection.Close(); }

}

public List<User> SelectAll()

{

GetDb.Connection.Open();

List<User> lst = new List<User>();

string query = "select \* from Users";

OleDbCommand select = new OleDbCommand(query, GetDb.Connection);

OleDbDataReader reader = select.ExecuteReader();

if (reader.HasRows)

{

while (reader.Read())

{

User obj = new User();

obj.Id = Convert.ToInt32(reader.GetValue(0));

obj.Login = reader.GetValue(1).ToString();

obj.Password = reader.GetValue(2).ToString();

obj.Rank = reader.GetValue(3).ToString();

lst.Add(obj);

}

}

reader.Close();

GetDb.Connection.Close();

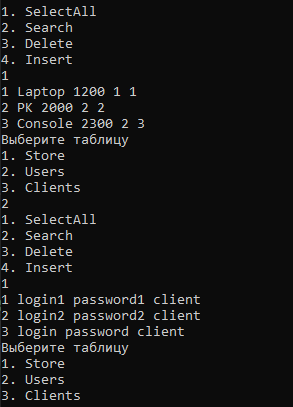
return lst;

}

}

}

Результат выполнения программы



Выводы:научился создавать модель заданной предметной области, ознакомился с особенностями архитектуры приложений, получил навыки реализации паттерна «Репозиторий» .