МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний аерокосмічний університет ім. М. Є. Жуковського

«Харківський авіаційний інститут»

Факультет систем управління літальними апаратами

Кафедра 302 «Інформаційні управляючі системи»

**Лабораторна робота № 1**

з дисципліни «Технологія створення програмного продукту»

на тему: «Створення додатку БД»

Виконала: студентка 2 курсу групи 327 ст

напряму підготовки (спеціальності)

6.050101 Комп’ютерні науки

Крейзе М.В,

Прийняв: доц. каф 302 Момот М.О\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Національна шкала: \_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_\_\_

Оцінка: ECTS \_\_\_\_\_

Харків – 2018

**Лабораторная работа № 1**

Тема: Разработка приложений БД .

**Вариант заданий:** Venue-Event-Reservation-Customer

**Дополнительное задание:**

1. Главная форма содержит меню, в котором есть обязательные пункты File (Exit) Help(About –с информацией о создателях, Help context) – для связи с последующей лабораторной работой.
2. Приложение содержит несколько окон для отображения данных, причем некоторые из них модальные.
3. Формы отображения таблиц содержат управляющие элементы DBNavigator, DBGrid и др. из закладки компонентов Data Controls.
4. Заголовки полей в DBGrid – по-русски.
5. Помеченные \* таблицы имеют графическое поле – отображать содержимое на форме.

**ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**

**Толстый клиент**

*Толстый или Rich-клиент* в архитектуре клиент-сервер -это приложение, обеспечивающее (в противовес тонкому клиенту) расширенную функциональность независимо от центрального сервера. Часто сервер в этом случае является лишь хранилищем данных, а вся работа по обработке и представлению этих данных переносится на машину клиента.

***Достоинства:***

* Толстый клиент обладает широким функционалом в отличие от тонкого.
* Режим многопользовательской работы.
* Предоставляет возможность работы даже при обрывах связи с сервером.
* Имеет возможность подключения к банкам без использования сети Интернет.
* Высокое быстродействие.

***Недостатки***

* Большой размер дистрибутива.
* Многое в работе клиента зависит от того, для какой платформы он разрабатывался.
* При работе с ним возникают проблемы с удаленным доступом к данным.
* Довольно сложный процесс установки и настройки.
* Сложность обновления и связанная с ней неактуальность данных.

**Тонкий клиент**

*Тонкий клиент* (англ. *thin client*) в компьютерных технологиях — компьютер или программа-клиент в сетях с клиент-серверной или терминальной архитектурой, который переносит все или большую часть задач по обработке информации на сервер. Примером тонкого клиента может служить компьютер с браузером, использующийся для работы с веб-приложениями.

***Web-клиенты (англ. WEB clients)***

Web-клиент как программа — браузер. Web-клиент как устройство — устройство, основным приложением которого (с точки зрения разработчика устройства или маркетолога) является браузер.

***Тонкие клиенты, работающие в терминальном режиме***

Под термином «тонкий клиент» подразумевается достаточно широкий с точки зрения системной архитектуры ряд устройств и программ, которые объединяются общим свойством: возможность работы в терминальном режиме. Таким образом, для работы тонкого клиента необходим терминальный сервер. Этим тонкий клиент отличается от толстого клиента, который, напротив, производит обработку информации независимо от сервера, используя последний в основном лишь для хранения данных.

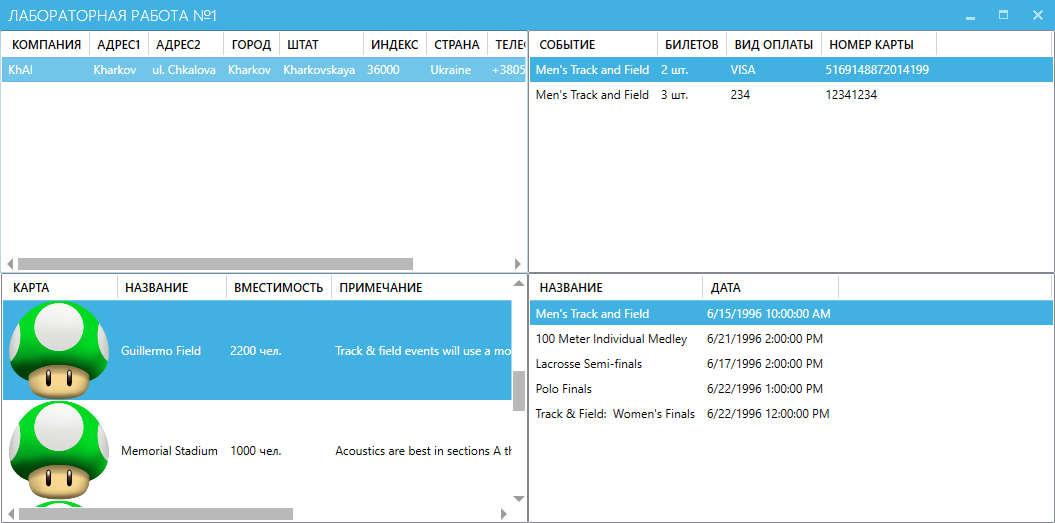
Сравнение размеров — тонкий клиент (справа) в сравнении с традиционным настольным ПК

Кроме общего случая, следует выделить аппаратный тонкий клиент (например, Windows- и Linux-терминалы) — специализированное устройство, принципиально отличное от ПК. Аппаратный тонкий клиент не имеет жёсткого диска, использует специализированную локальную ОС (одна из задач которой организовать сессию с терминальным сервером для работы пользователя), не имеет в своём составе подвижных деталей, выполняется в специализированных корпусах с полностью пассивным охлаждением.

Для расширения функциональности тонкого клиента прибегают к его «утолщению», например, добавляют возможности автономной работы, сохраняя главное отличие — работу в сессии с терминальным сервером. Когда в клиенте появляются подвижные детали (жёсткие диски), появляются возможности автономной работы, он перестаёт быть тонким клиентом в чистом виде, а становится универсальным клиентом.

Тонкий клиент в большинстве случаев обладает минимальной аппаратной конфигурацией, вместо жёсткого диска для загрузки локальной специализированной ОС используется DOM (DiskOnModule) [модуль с разъёмом IDE, флэш-памятью и микросхемой, реализующей логику обычного жёсткого диска — в BIOS определяется как обычный жёсткий диск, только размер его обычно в 2-3 раза меньше]. В некоторых конфигурациях системы тонкий клиент загружает операционную систему по сети с сервера, используя протоколы PXE, BOOTP, DHCP, TFTP и Remote Installation Services (RIS).

**ХОД РАБОТЫ**



**Листинг программы:**

using Lab1.Model;

using System;

using System.Collections.Generic;

using System.ComponentModel;

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;

using MahApps.Metro.Controls;

using MahApps.Metro.Controls.Dialogs;

namespace Lab1

{

/// <summary>

///    MainWindow.xaml

/// </summary>

public partial class MainWindow : MetroWindow, INotifyPropertyChanged

{

private EventContext context = new EventContext();

private List<Venue> venues;

public List<Venue> Venues { get { return venues; } set { venues = value; RaisePropertyChanged("Venues"); } }

private List<Event> events;

public List<Event> Events { get { return events; } set { events = value; RaisePropertyChanged("Events"); } }

private List<Reservation> reservations;

public List<Reservation> Reservations { get { return reservations; } set { reservations = value; RaisePropertyChanged("Reservations"); } }

private List<Customer> customers;

public List<Customer> Customers { get { return customers; } set { customers = value; RaisePropertyChanged("Customers"); } }

public MainWindow()

{

InitializeComponent();

venues = context.Venues.ToList();

customers = context.Customers.ToList();

DataContext = this;

//dataGrid.ItemsSource = c.Venues.ToList();

//dataGrid\_Copy.ItemsSource = c.Events.ToList();

//dataGrid\_Copy2.ItemsSource = c.Reservations.ToList();

//dataGrid\_Copy3.ItemsSource = c.Customers.ToList();

}

#region INotifyPropertyChanged

public event PropertyChangedEventHandler PropertyChanged;

public void RaisePropertyChanged(string propertyName)

{

PropertyChangedEventHandler handler = PropertyChanged;

if (handler != null)

{

PropertyChanged(this, new PropertyChangedEventArgs(propertyName));

}

}

#endregion

private void exitButton\_Click(object sender, RoutedEventArgs e)

{

Application.Current.Shutdown();

}

private async void aboutButton\_Click(object sender, RoutedEventArgs e)

{

}

private void listBox\_MouseDoubleClick(object sender, MouseButtonEventArgs e)

{

try

{

int cID = (listBox\_Copy2.SelectedItem as Customer).CustomerID;

Reservations = context.Reservations.Where(x => x.CustomerID == cID).ToList();

}catch { }

}

private void listBox\_MouseDoubleClick\_1(object sender, MouseButtonEventArgs e)

{

try

{

int vID = (listBox.SelectedItem as Venue).VenueID;

Events = context.Events.Where(x => x.VenueID == vID).ToList();

}

catch { }

}

private void listBox\_Copy\_MouseDoubleClick(object sender, MouseButtonEventArgs e)

{

try

{

int eID = (listBox\_Copy.SelectedItem as Event).EventID;

Reservations = context.Reservations.Where(x => x.EventID == eID).ToList();

}

catch { }

}

}

**ВЫВОД:**

Во время выполнения лабораторной работы мы научились работать с БД, подключать ее к программному продукту, работать с ней, в разработанном ПП. Ознакомились с понятиями «толстый» и «тонкий» клиент.