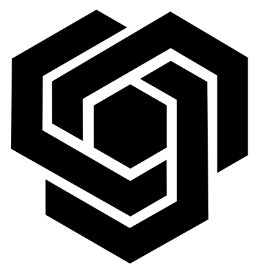
**Технически Университет София**

Факултет Компютърни системи и технологии

**Курсов Проект**

**По пРС**

**Вътрешен Софтуер за Книжарница**

Изготвил:

Йоанна Светлозарова Василева

Специалност: КСИ

Курс: Четвърти

Група: 42

Фак. № 121214108

София, Януари 2018 г.

СЪДЪРЖАНИЕ

Увод.......................................................................................................................................... 3

Проектиране............................................................................................................................ 3

Употреба на SOAP................................................................................................................... 5

Реализация............................................................................................................................... 6

Потребителски ръководство.................................................................................................. 17

Заключение............................................................................................................................. 19

Литература...............................................................................................................................20

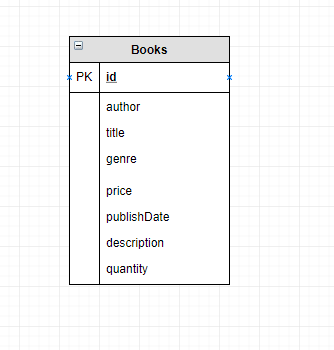
Приложение............................................................................................................................ 21

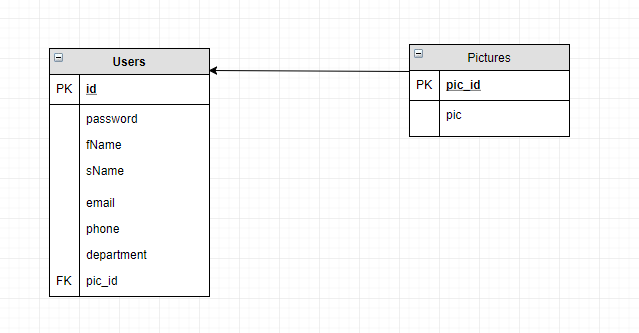
УВОД

В технологичното време, в което живеем, всяко едно действие ни е неимоверно свързано с интернет. От това да си поръчаме храна за вкъщи чак до това да купуваме книги онлайн. Но освен за крайния потребител, трябва да се помисли и от какво се нуждае персонала, а именно едно уеб приложение, в което да се дава информация кои книги ги има в наличност в магазина.

ПРОЕКТИРАНЕ

Базата данни има следния изглед:





Продуктът ще се използва от служители в магазин.

При стартиране, се появява уеб страница за влизане на профила. Когато потребителя достъпи профила си се отваря страница , съдържаща списък с имената на книгите. Потребителя има възможността да търси книга по жанр, автор и сценарий. При натискане на бутона search, списъка се обновява с желания резултат. Потребителя може да изтрие книга, която вече не е в наличност в магазина, като селектира дадена книга от списъка и натисне бутона Delete. Потребителя може да разгледа повече детайли за някоя книга като избере ред от списъка и натисне бутона View Details. При натискане на този бутон, се отваря втора страница, съдържаща подробности за книгата. Потребителя може да коригира информацията за дадена книга, чрез промяна на желано текстово поле и натискане на бутона „Update“. Потребителя може да добави нова книга, като въведе на мястото на запълнените полета, коректните данни. Ако такава книга вече съществува, тя няма да бъде добавена. Потребителя може да се върне към начална страница с бутона “Back”. Потребителя може да види профилната си страница при натискане на бутона-картинка. Отваря се профилната страница на потребителя. Той може да си обнови профилната снимка, телефона, отдела и паролата. Полетата с имената са неактивни. При натискане на бутона update, се обновява информацията на потребителя и се връща в страницата със списъка книги.

Употреба на SOAP:

Приложението използва SQL Server. Данните се извличат чрез WebService-SOAP.

Данни са предварително добавени в базата с тестова цел.

BooksSoap:

Добавяне на книга:

|  |  |  |  |
| --- | --- | --- | --- |
| **Метод** | **Адрес** | **Тяло на заявката** | **Тяло на отговора** |
| POST | BooksSoap.asmx/insert | XML Обект | - |

Четене на книгите:

|  |  |  |  |
| --- | --- | --- | --- |
| **Метод** | **Адрес** | **Тяло на заявката** | **Тяло на отговора** |
| POST | BooksSoap.asmx/read | - | XML Обект |

Триене на книга:

|  |  |  |  |
| --- | --- | --- | --- |
| **Метод** | **Адрес** | **Тяло на заявката** | **Тяло на отговора** |
| POST | BooksSoap.asmx/delete | XML Обект | - |

Ъпдейтване на книга:

|  |  |  |  |
| --- | --- | --- | --- |
| **Метод** | **Адрес** | **Тяло на заявката** | **Тяло на отговора** |
| POST | BooksSoap.asmx/update | XML Обект | - |

Users:

Откриване потребител по еmail и парола

|  |  |  |  |
| --- | --- | --- | --- |
| **Метод** | **Адрес** | **Тяло на заявката** | **Тяло на отговора** |
| POST | Users.asmx/findUsr | XML Обект | XML Обект |

Откриване потребител по email

|  |  |  |  |
| --- | --- | --- | --- |
| **Метод** | **Адрес** | **Тяло на заявката** | **Тяло на отговора** |
| POST | Users.asmx/getUsr | XML Обект | XML Обект |

Ъпдейтване на потребител:

|  |  |  |  |
| --- | --- | --- | --- |
| **Метод** | **Адрес** | **Тяло на заявката** | **Тяло на отговора** |
| POST | Users.asmx/updateUsr | XML Обект | - |

Намиране картинка по id

|  |  |  |  |
| --- | --- | --- | --- |
| **Метод** | **Адрес** | **Тяло на заявката** | **Тяло на отговора** |
| POST | Users.asmx/getImage | XML Обект | XML Обект |

Намиране картинка по url

|  |  |  |  |
| --- | --- | --- | --- |
| **Метод** | **Адрес** | **Тяло на заявката** | **Тяло на отговора** |
| POST | Users.asmx/getImgId | XML Обект | XML Обект |

Качване на картинка:

|  |  |  |  |
| --- | --- | --- | --- |
| **Метод** | **Адрес** | **Тяло на заявката** | **Тяло на отговора** |
| POST | Users.asmx/uploadImage | XML Обект | - |

Обновяване на картинка:

|  |  |  |  |
| --- | --- | --- | --- |
| **Метод** | **Адрес** | **Тяло на заявката** | **Тяло на отговора** |
| POST | Users.asmx/updateImage | XML Обект | - |

Изтриване на картинка:

|  |  |  |  |
| --- | --- | --- | --- |
| **Метод** | **Адрес** | **Тяло на заявката** | **Тяло на отговора** |
| POST | Users.asmx/deleteImage | XML Обект | - |

РЕАЛИЗАЦИЯ

Има класове-модели, които представляват oписание на данните, които идват от таблиците от базата.

Book:

public class Book

{

private int \_id;

private string \_author;

private string \_title;

private string \_genre;

private double \_price;

private DateTime \_publishDate;

private string \_description;

public int Id

{

get

{

return \_id;

}

set

{

\_id = value;

}

}

public string Author

{

get

{

return \_author;

}

set

{

\_author = value;

}

}

public string Title

{

get

{

return \_title;

}

set

{

\_title = value;

}

}

public string Genre

{

get

{

return \_genre;

}

set

{

\_genre = value;

}

}

public double Price

{

get

{

return \_price;

}

set

{

\_price = value;

}

}

public DateTime PublishDate

{

get

{

return \_publishDate;

}

set

{

\_publishDate = value;

}

}

public string Description

{

get

{

return \_description;

}

set

{

\_description = value;

}

}

public Book() { }

}

User:

public class User

{

private int \_id;

private string \_password;

private string \_fName;

private string \_sName;

private string \_email;

private string \_phone;

private string \_department;

private int \_picId;

public int Id

{

get

{

return \_id;

}

set

{

\_id = value;

}

}

public string Password

{

get

{

return \_password;

}

set

{

\_password = value;

}

}

public string FName

{

get

{

return \_fName;

}

set

{

\_fName = value;

}

}

public string SName

{

get

{

return \_sName;

}

set

{

\_sName = value;

}

}

public string Email

{

get

{

return \_email;

}

set

{

\_email = value;

}

}

public string Phone

{

get

{

return \_phone;

}

set

{

\_phone = value;

}

}

public string Department

{

get

{

return \_department;

}

set

{

\_department = value;

}

}

public int PicId

{

get

{

return \_picId;

}

set

{

\_picId = value;

}

}

public User() { }

}

Picture:

public class Picture

{

private int \_id;

private byte[] \_byteArr;

public int Id

{

get

{

return \_id;

}

set

{

\_id = value;

}

}

public byte[] ByteArr

{

get

{

return \_byteArr;

}

set

{

\_byteArr = value;

}

}

}

Има папка LogicImplementation, която съдържа класове имплементиращи логика необходима ни за приложението.

В CatalogC имаме listTitles, srchBk,delete,read. Read използва уеб метода и запазва книгите от базата в List<Books>. listTitles ползва този списък, като извлича само имената на кнгите и прави втори списък с имената на книгите. srchBk претърсва от всички книги по зададените му критерии дали има съответстващи книги и връща списък със съответстващите заглавия. Delete използва уеб сървиса и за изтрива избрана книга.

.

..

.

public List<Book> readBk()

{

ServiceReference1.BooksSoapSoapClient mySvc = new ServiceReference1.BooksSoapSoapClient();

var myEnumerable = mySvc.read().AsEnumerable();

List<Book> catalog = new List<Book>();

catalog=

(from item in myEnumerable

select new Book

{

Id = item.Field<int>("Id"),

Author = item.Field<string>("author"),

Title = item.Field<string>("title"),

Genre = item.Field<string>("genre"),

Price = Convert.ToDouble(item.Field<decimal>("price")),

PublishDate = item.Field<DateTime>("publishDate"),

Description = item.Field<string>("description")

}).ToList();

return catalog;

}

public List<Book> srchBk(string option)

{

string searchedWord=null;

searchedWord = SrchWord;

string name = null;

string genre = null;

string author = null;

List<Book> foundBooks = new List<Book>();

if (option.Equals("title"))

{

name = searchedWord;

} else if (option.Equals("genre"))

{

genre = searchedWord;

} else if (option.Equals("author"))

{

author = searchedWord;

}

List<Book> allBooks = readBk();

foreach (Book searchedBk in allBooks)

{

if (searchedWord.Equals(""))

{

foundBooks = allBooks;

}

else if (name!=null&&name.Equals(searchedBk.Title))

{

foundBooks.Add(searchedBk);

}else if (author!=null&&author.Equals(searchedBk.Author))

{

foundBooks.Add(searchedBk);

}else if (genre!=null&&genre.Equals(searchedBk.Genre))

{

foundBooks.Add(searchedBk);

}

}

Books = foundBooks;

return Books;

}

public List<string> listTitles(string option)

{

List<Book> bks = srchBk(option);

List<Book> forDispaly = new List<Book>();

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

if (bks.Equals(null)) {

forDispaly = readBk();

}

else

{

forDispaly = Books;

}

string title = null;

foreach(Book bk in forDispaly)

{

title = bk.Title;

titles.Add(title);

}

Ttl = titles;

return Ttl;

}

public void delete()

{

ServiceReference1.BooksSoapSoapClient mySvc = new ServiceReference1.BooksSoapSoapClient();

int id=0;

string selected = Selected;

List<Book> books =readBk();

foreach(Book bk in books)

{

if (selected.Equals(bk.Title))

{

id = bk.Id;

mySvc.delete(id);

}

}

}

..

.

.

В Insert имаме един метод insert, имплементиращ уеб метода.

.

..

…

public void insert()

{

ServiceReference1.BooksSoapSoapClient mySvc = new ServiceReference1.BooksSoapSoapClient();

//int id = Book.Id;

string title = Book.Title;

string author = Book.Author;

string genre = Book.Genre;

double price = Book.Price;

DateTime publDate = Book.PublishDate;

string description = Book.Description;

CatalogC c = new CatalogC();

List<Book> books = c.readBk();

string yes = "";

string no = "";

foreach (Book b in books)

{

if (title == b.Title&&author==b.Author)

{

yes = "y";

}

else

{

}

}

if (yes.Equals(""))

{

mySvc.insert(author, title, genre, price, publDate, description);

}

}

…

..

.

В Update имаме методите getBook и update. getBook намира книгата избрана за ъпдейтване. Update имплементира уеб метода и ъпдейтва.

..

…

public void update()

{

ServiceReference1.BooksSoapSoapClient mySvc = new ServiceReference1.BooksSoapSoapClient();

int id=Book.Id;

string title=Book.Title;

string author = Book.Author;

string genre=Book.Genre;

double price=Book.Price;

DateTime publDate=Book.PublishDate;

string description=Book.Description;

mySvc.update(id, author, title, genre, price, publDate, description);

}

public Book getBook()

{

CatalogC c = new CatalogC();

List<Book> books = c.readBk();

Book returned = new Book();

string title = Title;

foreach(Book b in books)

{

if (title.Equals(b.Title))

{

returned = b;

Book=b;

}

}

return returned;

}

….

В Login имаме метод, checkUser, който ни имплементира уеб метода-търси потребител по парола и email:

public int checkUsr(string mail, string pass)

{

UsersSOAP.UsersSoapClient mysvc = new UsersSOAP.UsersSoapClient();

var myEnumerable = mysvc.findUsr(mail,pass).AsEnumerable();

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

User u = new User();

lUsr =

(from item in myEnumerable

select new User {

Id=item.Field<int>("id"),

Password=item.Field<string>("password"),

FName=item.Field<string>("fName"),

SName=item.Field<string>("sName"),

Email=item.Field<string>("email"),

Phone=item.Field<string>("phone"),

Department=item.Field<string>("department")

}).ToList();

if (lUsr.Count!=0)

{

return 1;

}

else

{

return 0;

}

}

В UserInfo имаме метод, с който търсим потребителя по email. Намерения потребител го използваме за ъпдейтването му-чрез имплементация на уеб метода. Освен това имаме методи за работа с картинката-upload, update, delete спрямо това, какви са действията на потребителя. Те също са имплементации на уеб сървиси.

public class UserInfo

{

private User \_usr;

public User Usr

{

get

{

return \_usr;

}

set

{

\_usr = value;

}

}

public User getUser(string email)

{

UsersSOAP.UsersSoapClient mysvc = new UsersSOAP.UsersSoapClient();

var myEnumerable = mysvc.getUsr(email).AsEnumerable();

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

User u = new User();

lUsr =

(from item in myEnumerable

select new User

{

Id = item.Field<int>("id"),

Password = item.Field<string>("password"),

FName = item.Field<string>("fName"),

SName = item.Field<string>("sName"),

Email = item.Field<string>("email"),

Phone = item.Field<string>("phone"),

Department = item.Field<string>("department"),

PicId=item.Field<int>("pic\_id")

}).ToList();

u = lUsr.First();

Usr = u;

return u;

}

public string byteArrayToString(string email)

{

User u = getUser(email);

var picid =Usr.PicId;

Picture p = new Picture();

string img;

p.Id = Convert.ToInt32(picid);

UsersSOAP.UsersSoapClient mysvc = new UsersSOAP.UsersSoapClient();

byte[] imgByte = mysvc.getImage(p.Id);

img = Convert.ToBase64String(imgByte, 0, imgByte.Length);

return img;

}

public void updateImg(string email, string imgUrl)

{

UsersSOAP.UsersSoapClient mysvc = new UsersSOAP.UsersSoapClient();

User u = getUser(email);

var picid = Usr.PicId;

Picture p = new Picture();

if (picid == 6)

{

mysvc.uploadImage(imgUrl);

}

else

{

mysvc.updateImage(p.Id, imgUrl);

}

}

public void deleteImg(int old)

{

UsersSOAP.UsersSoapClient mysvc = new UsersSOAP.UsersSoapClient();

mysvc.deleteImage(old);

}

public void updateUsers(string imgUrl)

{

UsersSOAP.UsersSoapClient mysvc = new UsersSOAP.UsersSoapClient();

int id = Usr.Id;

string phone = Usr.Phone;

string department = Usr.Department;

string pass = Usr.Password;

int picId;

if (imgUrl.Equals(null) || imgUrl == "")

{

picId = Usr.PicId;

}

else

{

picId = mysvc.getImgId(imgUrl);

}

//int picId = Usr.PicId;

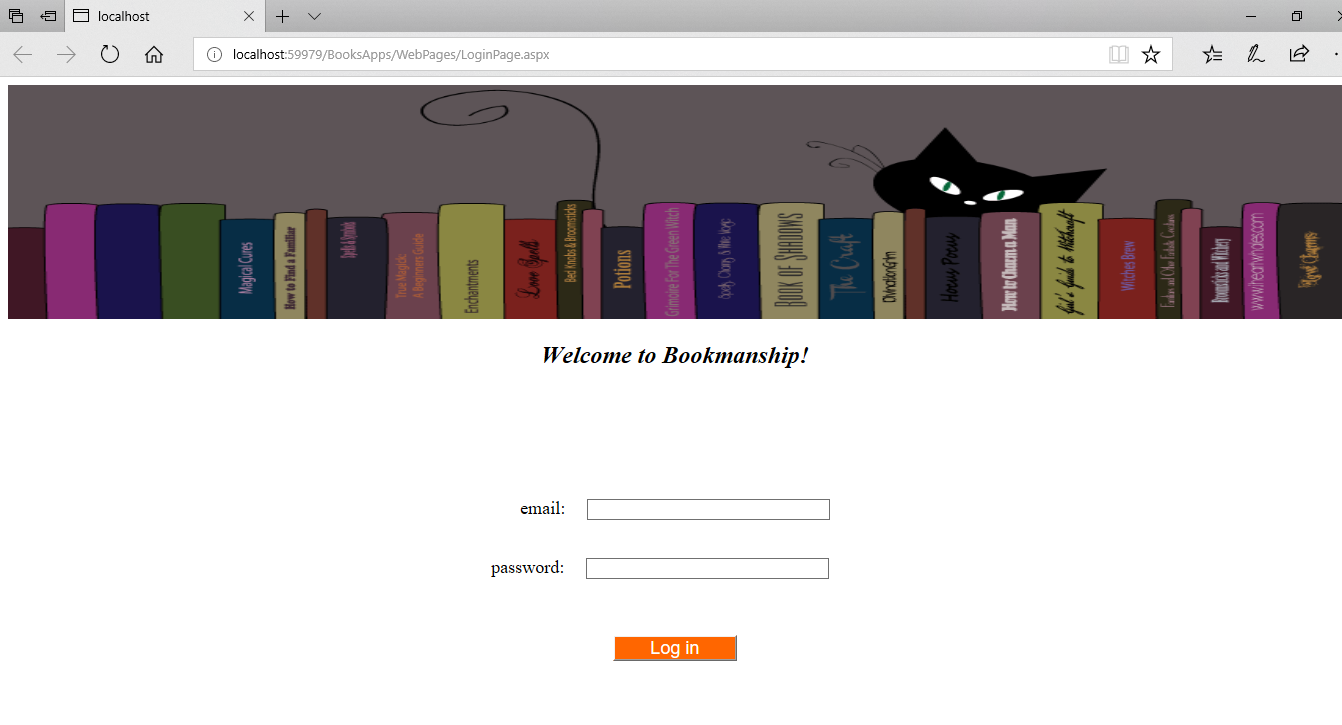
mysvc.updateUsr(id, phone, department, pass, picId);

}

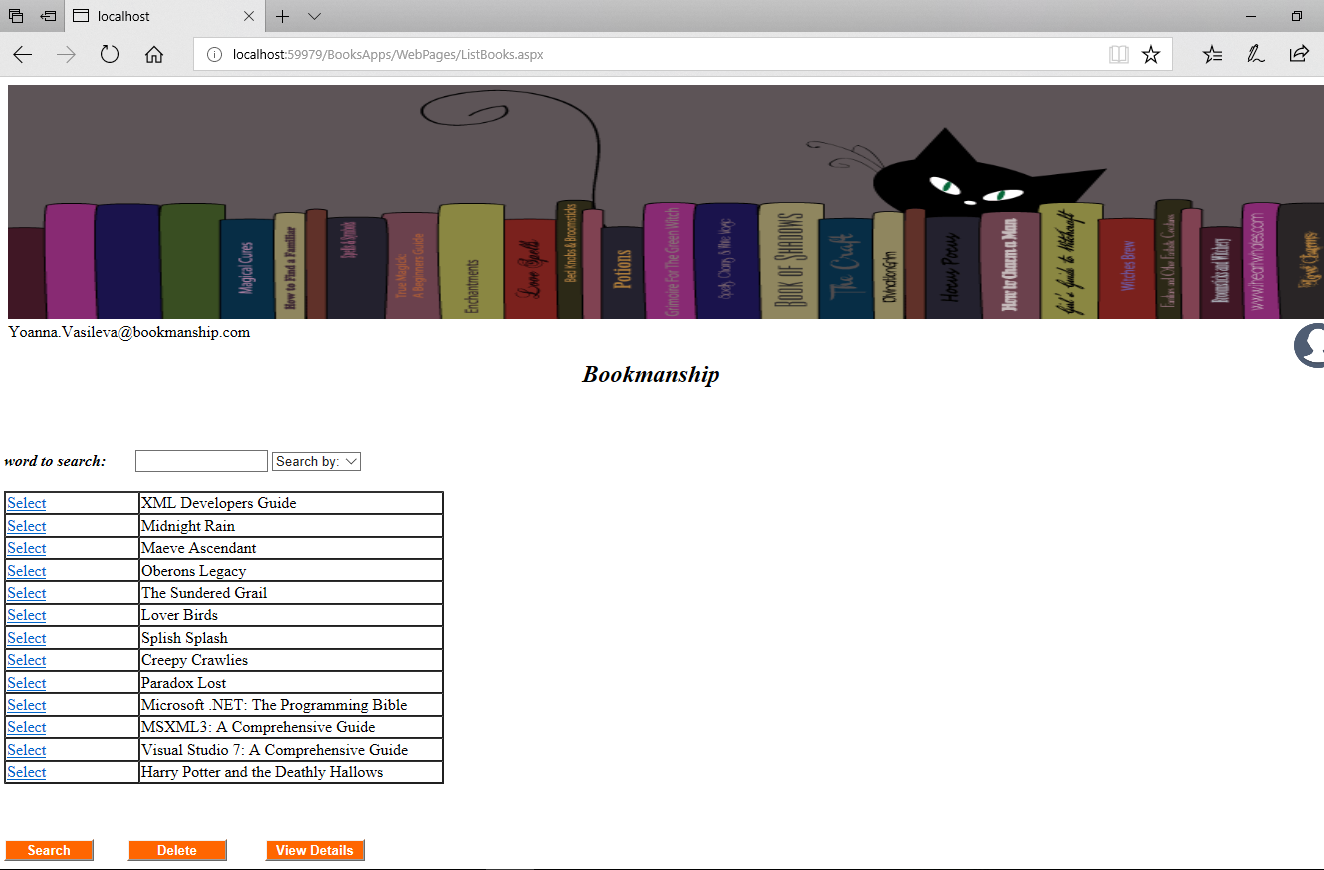
}

Има и две уеб страници ListBooks и Overview. Чрез тях потребителя комуникира.

ПОТРЕБИТЕЛСКО РЪКОВОДСТВО



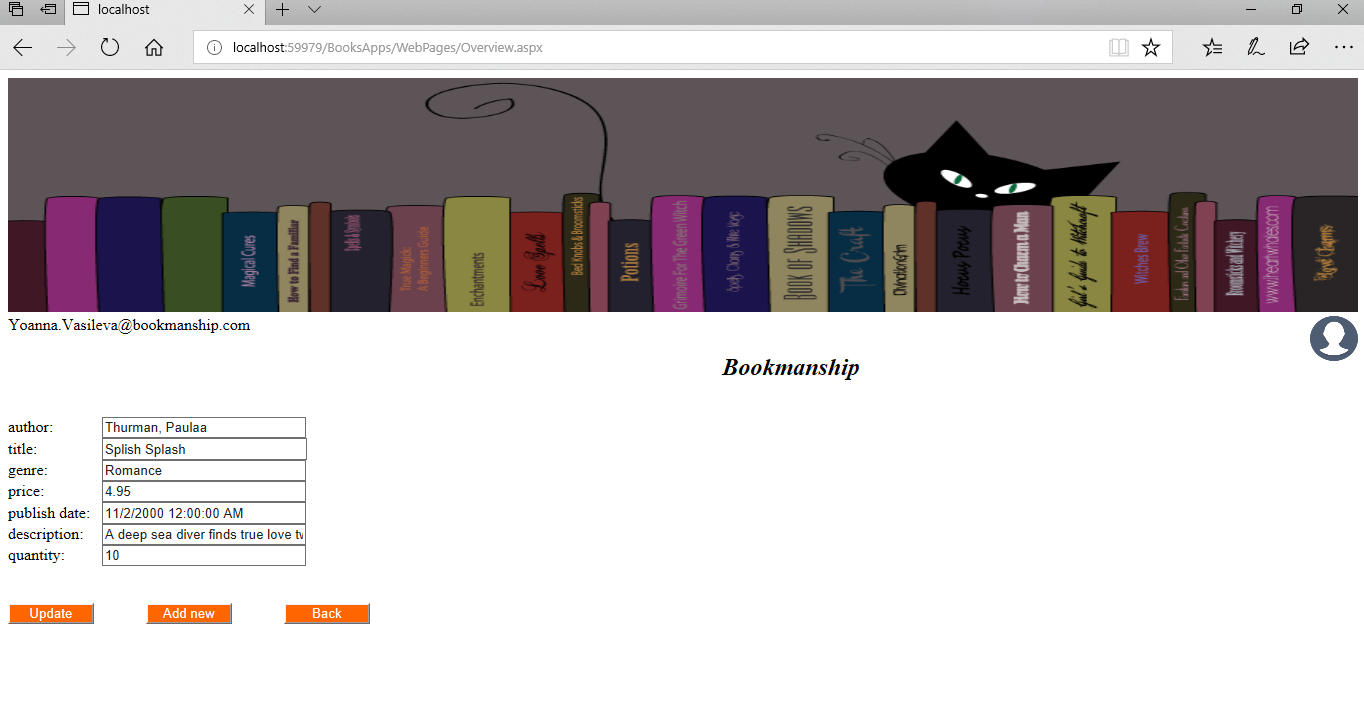
Това е първата страница, която се стартира. Ако потребителя има профил-може да влезе е него. Ако потребителя няма профил, той няма права за достъп и трябва да говори със сис-админа, за да му осигурят. При грешна комбинация от email и парола се появява съобщение за грешка.



При коректно вписване се зарежда втората страница. Потребителя може да избере категория, по която да търси и името, съответстващо и. При натискане на бутона search, резултата в списъка ще се обнови.

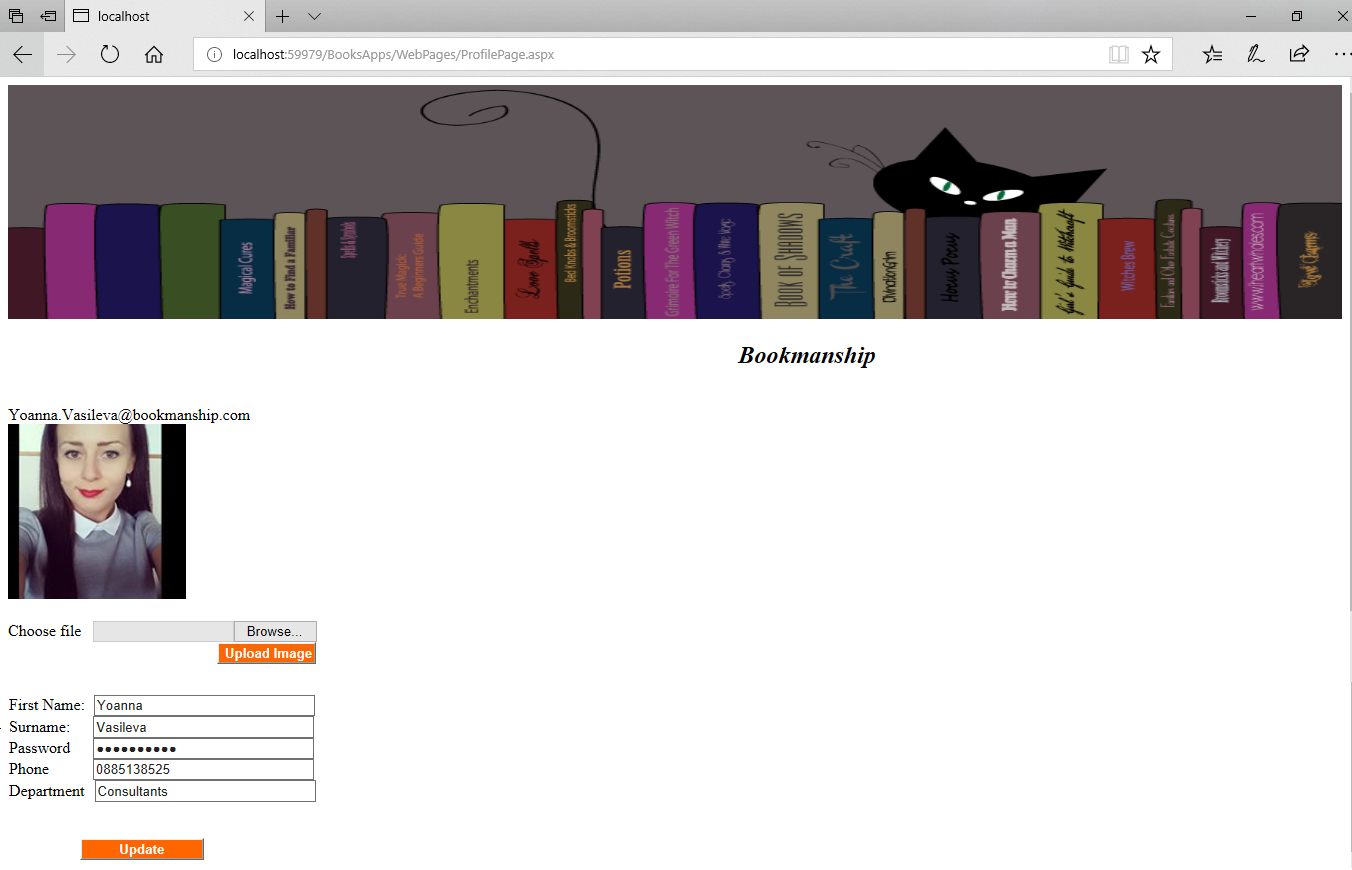
При избиране на ред от списъка потребителя има два варианта за този запис- да види повече детайли за него- чрез натискане на бутона view details или да го изтрие-чрез натискане на бутона Delete.

Потребителя може да достъпи профилната си страница, чрез натискане на профилния бутон.



При натискане на бутона View Details се появява страница, носеща ни информация за съответната книга. Потребителя може да обнови информацията, чрез промяна на текстово поле и натискане на бутон ъпдейт. Потребителя може да добави нова книга, чрез промяна на текстовите полета и натискане на бутон Add. При опит за добавяне на книга с вече съществуващ автор и същото заглавие, книгата няма да бъде добавяна. Потребителя може да се върне в началната страница с бутон back.

Потребителя може да достъпи профилната си страница, чрез натискане на профилния бутон.



При натискане на профилния бутон се появява профилната страница. Потребителя може да обнови информацията си. Потребителя не може да обнови имената си и email-a си.

Ако потребителя избере празна снимка, профила му ще се обнови с запис, който е dummy в базата. При натискане на бутона update, потребителя обновява данните си и сайта го пренасочва към страницата със списъка книги.

ЗАКЛЮЧЕНИЕ

Приложението изпълнява целите, които са поставени при проектиране. Използван е лесен за употреба интерфейс. Основната функционалности работа с книги е изпълнена. Освен това е добавена и страница за вписване и страница на потребителя. Възможните подобрения са планирани в следващи разработки.

ЛИТЕРАТУРА

1. https://stackoverflow.com
2. http://tasheva.info/exercises.php
3. https://msdn.microsoft.com/en-us/library/hh848246.aspx
4. https://www.tutorialspoint.com/sqlserver/

ПРИЛОЖЕНИЕ

https://github.com/jvassileva/Bookmanship

* Модел Book:

public class Book

{

private int \_id;

private string \_author;

private string \_title;

private string \_genre;

private double \_price;

private DateTime \_publishDate;

private string \_description;

private int \_quantity;

public int Id

{

get

{

return \_id;

}

set

{

\_id = value;

}

}

public string Author

{

get

{

return \_author;

}

set

{

\_author = value;

}

}

public string Title

{

get

{

return \_title;

}

set

{

\_title = value;

}

}

public string Genre

{

get

{

return \_genre;

}

set

{

\_genre = value;

}

}

public double Price

{

get

{

return \_price;

}

set

{

\_price = value;

}

}

public DateTime PublishDate

{

get

{

return \_publishDate;

}

set

{

\_publishDate = value;

}

}

public string Description

{

get

{

return \_description;

}

set

{

\_description = value;

}

}

public int Quantity

{

get

{

return \_quantity;

}

set

{

\_quantity = value;

}

}

public Book() { }

}

* Модел User:

public class User

{

private int \_id;

private string \_password;

private string \_fName;

private string \_sName;

private string \_email;

private string \_phone;

private string \_department;

private int \_picId;

public int Id

{

get

{

return \_id;

}

set

{

\_id = value;

}

}

public string Password

{

get

{

return \_password;

}

set

{

\_password = value;

}

}

public string FName

{

get

{

return \_fName;

}

set

{

\_fName = value;

}

}

public string SName

{

get

{

return \_sName;

}

set

{

\_sName = value;

}

}

public string Email

{

get

{

return \_email;

}

set

{

\_email = value;

}

}

public string Phone

{

get

{

return \_phone;

}

set

{

\_phone = value;

}

}

public string Department

{

get

{

return \_department;

}

set

{

\_department = value;

}

}

public int PicId

{

get

{

return \_picId;

}

set

{

\_picId = value;

}

}

public User() { }

}

* Модел Picture:

public class Picture

{

private int \_id;

private byte[] \_byteArr;

public int Id

{

get

{

return \_id;

}

set

{

\_id = value;

}

}

public byte[] ByteArr

{

get

{

return \_byteArr;

}

set

{

\_byteArr = value;

}

}

}

* LogicImplementation CatalogC:

public class CatalogC : ApiController

{

private string \_srchWord;

private string \_selected;

private List<Book> \_books;

private List<string> \_ttl;

public List<Book> Books

{

get

{

return \_books;

}

set

{

\_books = value;

}

}

public List<string> Ttl

{

get

{

return \_ttl;

}

set

{

\_ttl=value;

}

}

public string SrchWord

{

get

{

return \_srchWord;

}

set

{

\_srchWord = value;

}

}

public string Selected

{

get

{

return \_selected;

}

set

{

\_selected = value;

}

}

public List<Book> readBk()

{

ServiceReference1.BooksSoapSoapClient mySvc = new ServiceReference1.BooksSoapSoapClient();

var myEnumerable = mySvc.read().AsEnumerable();

List<Book> catalog = new List<Book>();

catalog=

(from item in myEnumerable

select new Book

{

Id = item.Field<int>("Id"),

Author = item.Field<string>("author"),

Title = item.Field<string>("title"),

Genre = item.Field<string>("genre"),

Price = Convert.ToDouble(item.Field<decimal>("price")),

PublishDate = item.Field<DateTime>("publishDate"),

Description = item.Field<string>("description"),

Quantity=item.Field<int>("quantity")

}).ToList();

return catalog;

}

public List<Book> srchBk(string option)

{

string searchedWord=null;

searchedWord = SrchWord;

string name = null;

string genre = null;

string author = null;

List<Book> foundBooks = new List<Book>();

if (option.Equals("title"))

{

name = searchedWord;

} else if (option.Equals("genre"))

{

genre = searchedWord;

} else if (option.Equals("author"))

{

author = searchedWord;

}

List<Book> allBooks = readBk();

foreach (Book searchedBk in allBooks)

{

if (searchedWord.Equals(""))

{

foundBooks = allBooks;

}

else if (name!=null&&name.Equals(searchedBk.Title))

{

foundBooks.Add(searchedBk);

}else if (author!=null&&author.Equals(searchedBk.Author))

{

foundBooks.Add(searchedBk);

}else if (genre!=null&&genre.Equals(searchedBk.Genre))

{

foundBooks.Add(searchedBk);

}

}

Books = foundBooks;

return Books;

}

public List<string> listTitles(string option)

{

List<Book> bks = srchBk(option);

List<Book> forDispaly = new List<Book>();

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

if (bks.Equals(null)) {

forDispaly = readBk();

}

else

{

forDispaly = Books;

}

string title = null;

foreach(Book bk in forDispaly)

{

title = bk.Title;

titles.Add(title);

}

Ttl = titles;

return Ttl;

}

public void delete()

{

ServiceReference1.BooksSoapSoapClient mySvc = new ServiceReference1.BooksSoapSoapClient();

int id=0;

string selected = Selected;

List<Book> books =readBk();

foreach(Book bk in books)

{

if (selected.Equals(bk.Title))

{

id = bk.Id;

mySvc.delete(id);

}

}

}

}

* LogicImplementation Insert:

public class Insert

{

private Book \_book;

public Book Book

{

get

{

return \_book;

}

set

{

\_book = value;

}

}

public void insert()

{

ServiceReference1.BooksSoapSoapClient mySvc = new ServiceReference1.BooksSoapSoapClient();

//int id = Book.Id;

string title = Book.Title;

string author = Book.Author;

string genre = Book.Genre;

double price = Book.Price;

DateTime publDate = Book.PublishDate;

string description = Book.Description;

int quant = Book.Quantity;

CatalogC c = new CatalogC();

List<Book> books = c.readBk();

string yes = "";

string no = "";

foreach (Book b in books)

{

if (title == b.Title&&author==b.Author)

{

yes = "y";

}

else

{

}

}

if (yes.Equals(""))

{

mySvc.insert(author, title, genre, price, publDate, description, quant);

}

}

}

* LogicImplementation Update:

public class Update

{

private string \_title;

public string Title

{

get

{

return \_title;

}

set

{

\_title = value;

}

}

private Book \_book;

public Book Book

{

get

{

return \_book;

}

set

{

\_book = value;

}

}

public void update()

{

ServiceReference1.BooksSoapSoapClient mySvc = new ServiceReference1.BooksSoapSoapClient();

int id=Book.Id;

string title=Book.Title;

string author = Book.Author;

string genre=Book.Genre;

double price=Book.Price;

DateTime publDate=Book.PublishDate;

string description=Book.Description;

int quant = Book.Quantity;

mySvc.update(id, author, title, genre, price, publDate, description, quant);

}

public Book getBook()

{

CatalogC c = new CatalogC();

List<Book> books = c.readBk();

Book returned = new Book();

string title = Title;

foreach(Book b in books)

{

if (title.Equals(b.Title))

{

returned = b;

Book=b;

}

}

return returned;

}

}

* LogicImplementation Login:

public class LogIn

{

public int checkUsr(string mail, string pass)

{

UsersSOAP.UsersSoapClient mysvc = new UsersSOAP.UsersSoapClient();

var myEnumerable = mysvc.findUsr(mail,pass).AsEnumerable();

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

User u = new User();

lUsr =

(from item in myEnumerable

select new User {

Id=item.Field<int>("id"),

Password=item.Field<string>("password"),

FName=item.Field<string>("fName"),

SName=item.Field<string>("sName"),

Email=item.Field<string>("email"),

Phone=item.Field<string>("phone"),

Department=item.Field<string>("department")

}).ToList();

if (lUsr.Count!=0)

{

return 1;

}

else

{

return 0;

}

}

}

}

* LogicImplementation UserInfo:

public class UserInfo

{

private User \_usr;

public User Usr

{

get

{

return \_usr;

}

set

{

\_usr = value;

}

}

public User getUser(string email)

{

UsersSOAP.UsersSoapClient mysvc = new UsersSOAP.UsersSoapClient();

var myEnumerable = mysvc.getUsr(email).AsEnumerable();

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

User u = new User();

lUsr =

(from item in myEnumerable

select new User

{

Id = item.Field<int>("id"),

Password = item.Field<string>("password"),

FName = item.Field<string>("fName"),

SName = item.Field<string>("sName"),

Email = item.Field<string>("email"),

Phone = item.Field<string>("phone"),

Department = item.Field<string>("department"),

PicId=item.Field<int>("pic\_id")

}).ToList();

u = lUsr.First();

Usr = u;

return u;

}

public string byteArrayToString(string email)

{

User u = getUser(email);

var picid =Usr.PicId;

Picture p = new Picture();

string img;

p.Id = Convert.ToInt32(picid);

UsersSOAP.UsersSoapClient mysvc = new UsersSOAP.UsersSoapClient();

byte[] imgByte = mysvc.getImage(p.Id);

img = Convert.ToBase64String(imgByte, 0, imgByte.Length);

return img;

}

public void updateImg(string email, string imgUrl)

{

UsersSOAP.UsersSoapClient mysvc = new UsersSOAP.UsersSoapClient();

User u = getUser(email);

var picid = Usr.PicId;

Picture p = new Picture();

if (picid == 6)

{

mysvc.uploadImage(imgUrl);

}

else

{

mysvc.updateImage(p.Id, imgUrl);

}

}

public void deleteImg(int old)

{

UsersSOAP.UsersSoapClient mysvc = new UsersSOAP.UsersSoapClient();

mysvc.deleteImage(old);

}

public void updateUsers(string imgUrl)

{

UsersSOAP.UsersSoapClient mysvc = new UsersSOAP.UsersSoapClient();

int id = Usr.Id;

string phone = Usr.Phone;

string department = Usr.Department;

string pass = Usr.Password;

int picId;

if (imgUrl.Equals(null) || imgUrl == "")

{

picId = Usr.PicId;

}

else

{

picId = mysvc.getImgId(imgUrl);

}

//int picId = Usr.PicId;

mysvc.updateUsr(id, phone, department, pass, picId);

}

}

* WebPage LoginPage:

public partial class Login : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

lblErr.Text = "";

}

public void check()

{

LogIn lgIn = new LogIn();

string email;

string pass;

email = txtEmail.Text;

pass = txtPass.Text;

int check = lgIn.checkUsr(email, pass);

if (check == 0)

{

lblErr.Text = "Wrong email or passord. Please try again!";

}

}

protected void btnLogin\_Click(object sender, EventArgs e)

{

check();

string email = txtEmail.Text;

if (lblErr.Text.Equals("")) {

if (email != null)

{

Session["se1"] = email;

Response.Redirect("ListBooks.aspx");

}

}

}

}

* WebPage ListBooks:

public partial class ListBooks : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

string field = (string)(Session["se1"]);

lblMail.Text = field;

showList();

lblErr.Text = "";

}

protected void showList()

{

string option = DropDownOption.SelectedValue;

string srched = searched.Text;

CatalogC c = new CatalogC();

c.SrchWord = srched;

List<string> data = c.listTitles(option);

titles.DataSource = null;

titles.DataSource = data;

titles.DataBind();

}

protected void btnSrc\_Click(object sender, EventArgs e)

{

showList();

}

protected void btnDelete\_Click(object sender, EventArgs e)

{

if (titles.SelectedIndex > 0) {

ListViewItem item = titles.Items[titles.SelectedIndex];

Label lbl = (Label)item.FindControl("lblSel");

string result = lbl.Text;

CatalogC c = new CatalogC();

c.Selected = result;

c.delete();

}

else

{

lblErr.Text = "Please select an item!";

}

}

protected void titles\_SelectedIndexChanged(object sender, EventArgs e)

{

if (titles.SelectedIndex >= 1) { }

// ViewState["SelectedKey"] = titles.SelectedValue;

else

ViewState["SelectedKey"] = null;

}

protected void titles\_SelectedIndexChanging(object sender, ListViewSelectEventArgs e)

{

// ListViewItem selItem = titles.Items[titles.SelectedIndex];

// string selItem = titles.SelectedDataKey.Value.ToString();

}

protected void btnOvrvw\_Click(object sender, EventArgs e)

{

if (titles.SelectedIndex >0) {

ListViewItem item = titles.Items[titles.SelectedIndex];

Label lbl = (Label)item.FindControl("lblSel");

string email = lblMail.Text;

string result = lbl.Text;

if (result != null&&email!=null)

{

Session["se1"] = email;

Session["se2"] = result;

Response.Redirect("Overview.aspx");

}

}

else

{

lblErr.Text = "Please select an item!";

}

}

protected void btnProfile\_Click(object sender, EventArgs e)

{

string email = lblMail.Text;

if (email != null)

{

Session["se3"] = email;

Response.Redirect("ProfilePage.aspx");

}

}

}

* WebPage Overview:

public partial class Overview : System.Web.UI.Page

{

Update upd = new Update();

Book bk = new Book();

protected void Page\_Load(object sender, EventArgs e)

{

string email = (string)(Session["se1"]);

lblMail.Text = email;

string field = (string)(Session["se2"]);

upd.Title = field;

bk = upd.getBook();

display(field);

}

public void display(string field)

{

upd.Title = field;

bk = upd.getBook();

if (!IsPostBack)

{

txtTit.Text = bk.Title;

txtAuth.Text = bk.Author;

txtGenre.Text = bk.Genre;

txtPrc.Text = bk.Price.ToString();

txtPubDt.Text = bk.PublishDate.ToString();

txtDescr.Text = bk.Description;

txtQnt.Text = bk.Quantity.ToString();

}

}

protected void btnUpdate\_Click(object sender, EventArgs e)

{

Book mybk = upd.Book;

bk.Id = mybk.Id;

bk.Author = txtAuth.Text;

bk.Title = txtTit.Text;

bk.Price = Convert.ToDouble(txtPrc.Text);

bk.Genre = txtGenre.Text;

bk.PublishDate = Convert.ToDateTime(txtPubDt.Text);

bk.Description = txtDescr.Text;

bk.Quantity = Convert.ToInt32(txtQnt.Text);

upd.update();

}

protected void btnInsert\_Click(object sender, EventArgs e)

{

Insert i = new Insert();

// bk.Id = mybk.Id;

bk.Author = txtAuth.Text;

bk.Title = txtTit.Text;

bk.Price = Convert.ToDouble(txtPrc.Text);

bk.Genre = txtGenre.Text;

bk.PublishDate = Convert.ToDateTime(txtPubDt.Text);

bk.Description = txtDescr.Text;

bk.Quantity = Convert.ToInt32(txtQnt.Text);

i.Book = bk;

i.insert();

}

protected void Button1\_Click(object sender, EventArgs e)

{

Response.Redirect("ListBooks.aspx");

}

protected void btnProfile\_Click(object sender, EventArgs e)

{

string email = lblMail.Text;

if (email != null)

{

Session["se3"] = email;

Response.Redirect("ProfilePage.aspx");

}

}

}

* WebPage ProfilePage:

public partial class ProfilePage : System.Web.UI.Page

{

UserInfo u = new UserInfo();

//User myUsr = new Model.User();

protected void Page\_Load(object sender, EventArgs e)

{

string email = (string)(Session["se3"]);

lblMail.Text = email;

getData(email);

}

public void getData(string email)

{

if (!IsPostBack)

{

UserInfo info = new UserInfo();

string img = info.byteArrayToString(email);

info.Usr = info.getUser(email);

txtFN.Text = info.Usr.FName;

txtSN.Text = info.Usr.SName;

txtPass.Text = info.Usr.Password;

txtPhone.Text = info.Usr.Phone;

txtDepartm.Text = info.Usr.Department;

Image1.ImageUrl = "data:image/png;base64," + img;

}

}

public void displayPic()

{

System.IO.Stream fs = FileUpload1.PostedFile.InputStream;

string name = FileUpload1.FileName;

if (name.Equals(null)||name=="")

{

Image1.Visible = false;

}

else {

FileUpload1.SaveAs(@"D:\My Documents\Visual Studio 2017\Projects\BooksApps\BooksApps\WebPages\" + name);

string pth = Server.MapPath(FileUpload1.FileName);

path.Text = pth;

System.IO.BinaryReader br = new System.IO.BinaryReader(fs);

Byte[] bytes = br.ReadBytes((Int32)fs.Length);

string base64String = Convert.ToBase64String(bytes, 0, bytes.Length);

Image1.ImageUrl = "data:image/png;base64," + base64String;

Image1.Visible = true;

}

}

protected void btnUpload\_Click(object sender, EventArgs e)

{

displayPic();

}

protected void btnUpd\_Click(object sender, EventArgs e)

{

string email = lblMail.Text;

updateUsr(email);

if (email != null)

{

Session["se1"] = email;

Response.Redirect("ListBooks.aspx");

}

}

public void updateUsr(string email)

{

UserInfo info = new UserInfo();

info.Usr = info.getUser(email);

int old = info.Usr.PicId;

string imgUrl = null;

string picPath = path.Text;

if (picPath == "" && Image1.ImageUrl.Equals(null))

{

info.Usr.Password = txtPass.Text;

info.Usr.Department = txtDepartm.Text;

info.Usr.Phone = txtPhone.Text;

info.Usr.PicId = 6;

info.updateUsers(picPath);

info.deleteImg(old);

}

else if (picPath == "" && Image1.ImageUrl != null && Image1.ImageUrl != "") {

info.Usr.Password = txtPass.Text;

info.Usr.Department = txtDepartm.Text;

info.Usr.Phone = txtPhone.Text;

imgUrl = Image1.ImageUrl;

info.updateUsers(picPath);

}

else

{

info.updateImg(email, picPath);

info.Usr.Password = txtPass.Text;

info.Usr.Department = txtDepartm.Text;

info.Usr.Phone = txtPhone.Text;

imgUrl = Image1.ImageUrl;

info.updateUsers(picPath);

}

}

}