LAB - SOFTWARE ARCHITECTURE

Mục lục

1 Cloud hostings

1.1 GearHost

- Sign up account from GearHost: https://www.gearhost.com

1.1.1 Database

- Menu Databases » Create Database:

- Database Name: sonkk
- Database Type: MSSQL

» Create Empty Database » Database informations:

- Database Name: sonkk
- Database Server: den1.mssql7.gear.host
- Username: sonkk
- Password: pwd

- SQL Scripting:

```
--Create table
IF OBJECT_ID('Book') IS NOT NULL DROP TABLE Book;
CREATE TABLE Book(
  Code int PRIMARY KEY IDENTITY,
  Name nvarchar(50) NOT NULL,
  Author nvarchar(50) NOT NULL,
  Price int
);
--Insert data
INSERT INTO Book(Name, Author, Price) VALUES('SA', 'HSU1', 111);
INSERT INTO Book(Name, Author, Price) VALUES('OOP', 'HSU2', 222);
INSERT INTO Book(Name, Author, Price) VALUES('OOAD', 'HSU3', 333);
--Select data
SELECT * FROM Book;
```

1.1.2 CloudSite

- Menu CloudSites » Add CloudSite:

- Application Name: sonkk (http://sonkk.gear.host)
- Application Type: Free

» Create CloudSite » Access CloudSite » Publish Tab » Application Publishing Files » Download Visual Studio publishing profile (sonkk.publishsettings file)

1.2 Google Firebase

- Sign in google account from Firebase: http://firebase.google.com » Go to console » Add project

- Project name: sonkk
- Uncheck on Enable Google Analytics for this project
- Menu Realtime Database » Create Database » Security rules: Start in test mode
- URL: https://sonkk.firebaseio.com

2 Architectural styles

2.1 nLayers/nTiers architecture

2.1.1 ADO.NET

- Create a new project:

- Project type: Windows Forms App (.NET Framework)
- Project name: BookClient
- Solution name: nLayers

BookClient project

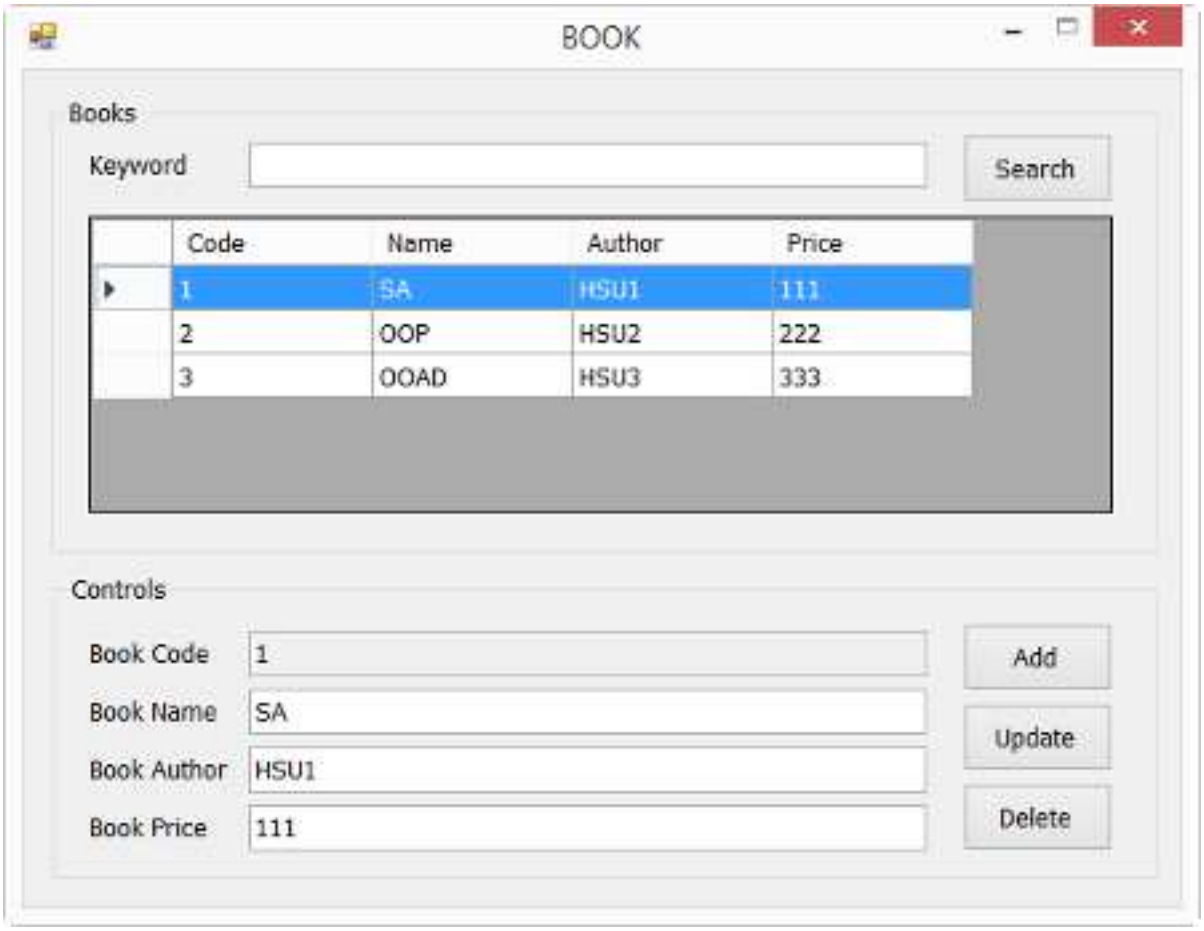
- Update App.config

```
<configuration>
  ...
  <connectionStrings>
    <add name="strCon" providerName="System.Data.SqlClient" connectionString="
       SERVER=den1.mssql7.gear.host;DATABASE=sonkk;USER=sonkk;PASSWORD=pwd" />
  </connectionStrings>
</configuration>
```

- Create Book.cs (DTO class)

```
...
class Book {
  public int Code { get; set; }
  public String Name { get; set; }
  public String Author { get; set; }
  public int Price { get; set; }
}
```

- Create BookForm.cs (presentation class)




```
...
public partial class BookForm : Form {
  public BookForm() {
    InitializeComponent();
  }
  private void BookForm_Load(object sender, EventArgs e) {
    List<Book> books = new BookBUS().GetAll();
    gridBook.DataSource = books;
  }
  private void BookForm_FormClosing(object sender, FormClosingEventArgs e) {
    Application.Exit();
  }
  private void gridBook_SelectionChanged(object sender, EventArgs e) {
    if (gridBook.SelectedRows.Count > 0) {
      int code = int.Parse(gridBook.SelectedRows[0].Cells["Code"].Value.ToString
          ());
      Book book = new BookBUS().GetDetails(code);
      if (book != null) {
        txtCode.Text = book.Code.ToString();
        txtName.Text = book.Name;
        txtAuthor.Text = book.Author;
        txtPrice.Text = book.Price.ToString();
      }
    }
```

```
  }
  private void btnSearch_Click(object sender, EventArgs e) {
    String keyword = txtKeyword.Text.Trim();
    List<Book> books = new BookBUS().Search(keyword);
    gridBook.DataSource = books;
  }
  private void btnAdd_Click(object sender, EventArgs e) {
    Book newBook = new Book() {
      Code = 0,
      Name = txtName.Text.Trim(),
      Author = txtAuthor.Text.Trim(),
      Price = int.Parse(txtPrice.Text.Trim())
    };
    bool result = new BookBUS().AddNew(newBook);
    if (result) {
      List<Book> books = new BookBUS().GetAll();
      gridBook.DataSource = books;
    } else {
      MessageBox.Show("SORRY BABY!");
    }
  }
  private void btnUpdate_Click(object sender, EventArgs e) {
    Book newBook = new Book() {
      Code = int.Parse(txtCode.Text.Trim()),
      Name = txtName.Text.Trim(),
      Author = txtAuthor.Text.Trim(),
      Price = int.Parse(txtPrice.Text.Trim())
    };
    bool result = new BookBUS().Update(newBook);
    if (result) {
      List<Book> books = new BookBUS().GetAll();
      gridBook.DataSource = books;
    } else {
      MessageBox.Show("SORRY BABY!");
    }
  }
  private void btnDelete_Click(object sender, EventArgs e) {
    DialogResult dialogResult = MessageBox.Show("ARE YOU SURE ?", "CONFIRMATION"
        , MessageBoxButtons.YesNo);
    if (dialogResult == DialogResult.Yes) {
      int code = int.Parse(txtCode.Text);
      bool result = new BookBUS().Delete(code);
      if (result) {
        List<Book> books = new BookBUS().GetAll();
        gridBook.DataSource = books;
      } else {
        MessageBox.Show("SORRY BABY!");
      }
    }
  }
}
```

- Create BookBUS.cs (business-logic class)

```
...
class BookBUS {
  public List<Book> GetAll() {
    List<Book> books = new BookDAO().SelectAll();
    return books;
  }
  public Book GetDetails(int code) {
    Book book = new BookDAO().SelectByCode(code);
    return book;
  }
  public List<Book> Search(String keyword) {
    List<Book> books = new BookDAO().SelectByKeyword(keyword);
    return books;
  }
  public bool AddNew(Book newBook) {
    bool result = new BookDAO().Insert(newBook);
    return result;
  }
  public bool Update(Book newBook) {
    bool result = new BookDAO().Update(newBook);
    return result;
  }
  public bool Delete(int code) {
    bool result = new BookDAO().Delete(code);
    return result;
  }
}
```

- Create BookDAO.cs (data-access class)

```
...
using System.Configuration; // add reference System.Configuration.dll
using System.Data.SqlClient;

class BookDAO {
  public List<Book> SelectAll() {
    List<Book> books = new List<Book>();
    String strCon = ConfigurationManager.ConnectionStrings["strCon"].
       ConnectionString;
    SqlConnection con = new SqlConnection(strCon);
    con.Open();
    String strCom = "SELECT * FROM Book";
    SqlCommand com = new SqlCommand(strCom, con);
    SqlDataReader dr = com.ExecuteReader();
    while (dr.Read()) {
      Book book = new Book() {
        Code = (int)dr["Code"],
        Name = (String)dr["Name"],
        Author = (String)dr["Author"],
```

```
19          Price = (int)dr["Price"]
20        };
21        books.Add(book);
22      }
23      return books;
24    }
25    public Book SelectByCode(int code) {
26      String strCon = ConfigurationManager.ConnectionStrings["strCon"].
           ConnectionString;
27      SqlConnection con = new SqlConnection(strCon);
28      con.Open();
29      String strCom = "SELECT * FROM Book WHERE Code=@Code";
30      SqlCommand com = new SqlCommand(strCom, con);
31      com.Parameters.Add(new SqlParameter("@Code", code));
32      SqlDataReader dr = com.ExecuteReader();
33      if (dr.Read()) {
34        Book book = new Book() {
35          Code = (int)dr["Code"],
36          Name = (String)dr["Name"],
37          Author = (String)dr["Author"],
38          Price = (int)dr["Price"]
39        };
40        return book;
41      }
42      return null;
43    }
44    public List<Book> SelectByKeyword(String keyword) {
45      List<Book> books = new List<Book>();
46      String strCon = ConfigurationManager.ConnectionStrings["strCon"].
           ConnectionString;
47      SqlConnection con = new SqlConnection(strCon);
48      con.Open();
49      String strCom = "SELECT * FROM Book WHERE Name LIKE @Keyword";
50      SqlCommand com = new SqlCommand(strCom, con);
51      com.Parameters.Add(new SqlParameter("@Keyword", "%" + keyword + "%"));
52      SqlDataReader dr = com.ExecuteReader();
53      while (dr.Read()) {
54        Book book = new Book() {
55          Code = (int)dr["Code"],
56          Name = (String)dr["Name"],
57          Author = (String)dr["Author"],
58          Price = (int)dr["Price"]
59        };
60        books.Add(book);
61      }
62      return books;
63    }
64    public bool Insert(Book newBook) {
65      String strCon = ConfigurationManager.ConnectionStrings["strCon"].
           ConnectionString;
66      SqlConnection con = new SqlConnection(strCon);
```

```
67      con.Open();
68      String strCom = "INSERT INTO Book VALUES(@Name,@Author,@Price)";
69      SqlCommand com = new SqlCommand(strCom, con);
70      com.Parameters.Add(new SqlParameter("@Name", newBook.Name));
71      com.Parameters.Add(new SqlParameter("@Author", newBook.Author));
72      com.Parameters.Add(new SqlParameter("@Price", newBook.Price));
73      try { return com.ExecuteNonQuery() > 0; }
74      catch { return false; }
75    }
76    public bool Update(Book newBook) {
77      String strCon = ConfigurationManager.ConnectionStrings["strCon"].
          ConnectionString;
78      SqlConnection con = new SqlConnection(strCon);
79      con.Open();
80      String strCom = "UPDATE Book SET Name=@Name,Author=@Author,Price=@Price
          WHERE Code=@Code";
81      SqlCommand com = new SqlCommand(strCom, con);
82      com.Parameters.Add(new SqlParameter("@Name", newBook.Name));
83      com.Parameters.Add(new SqlParameter("@Author", newBook.Author));
84      com.Parameters.Add(new SqlParameter("@Price", newBook.Price));
85      com.Parameters.Add(new SqlParameter("@Code", newBook.Code));
86      try { return com.ExecuteNonQuery() > 0; }
87      catch { return false; }
88    }
89    public bool Delete(int code) {
90      String strCon = ConfigurationManager.ConnectionStrings["strCon"].
          ConnectionString;
91      SqlConnection con = new SqlConnection(strCon);
92      con.Open();
93      String strCom = "DELETE FROM Book WHERE Code=@Code";
94      SqlCommand com = new SqlCommand(strCom, con);
95      com.Parameters.Add(new SqlParameter("@Code", code));
96      try { return com.ExecuteNonQuery() > 0; }
97      catch { return false; }
98    }
99  }
```

2.1.2 LINQ

- Menu Tools » Get Tools and Features... » Individual components tab » Search by "LINQ to SQL tools" » Install

- Create a new project:

  - Project type: Windows Forms App (.NET Framework)
  - Project name: BookClient
  - Solution name: LINQ

BookClient project

- Create MyDB.dbml (LINQ to SQL Classes)

- Add Data Connection:
  - Server Explorer tab » Right-click Data Connections » Add Connection... » Data source: Microsoft SQL Server
  - Server name: den1.mssql7.gear.host
  - Authentication: SQL Server Authentication
  - Username: sonkk & Password: pwd & Check on Save my password
  - Select or enter a database name: sonkk
- Drag and drop Book table in DB server into MyDB.dbml designer
- Warning "The connection string ... a clear text password ... security" » No
- Open MyDB.designer.cs file » Review MyDBDataContext class & Book class (DTO class)

- Create BookForm.cs (presentation class)
- Create BookBUS.cs (business-logic class)
- Create BookDAO.cs (data-access class)

```
...
using System.Configuration; // add reference System.Configuration.dll

class BookDAO {
  private MyDBDataContext db = new MyDBDataContext(ConfigurationManager.
     ConnectionStrings["strCon"].ConnectionString);

  public List<Book> SelectAll() {
    db.ObjectTrackingEnabled = false; // fix error "System.Runtime.Serialization
       .SerializationException: Type 'System.Data.Linq.ChangeTracker+
       StandardChangeTracker' in Assembly '...' is not marked as serializable"
    List<Book> books = db.Books.ToList();
    return books;
  }
  public Book SelectByCode(int code) {
    Book book = db.Books.SingleOrDefault(b => b.Code == code);
    return book;
  }
  public List<Book> SelectByKeyword(String keyword) {
    List<Book> books = db.Books.Where(b => b.Name.Contains(keyword)).ToList();
    return books;
  }
  public bool Insert(Book newBook) {
    try {
      db.Books.InsertOnSubmit(newBook);
      db.SubmitChanges();
      return true;
    }
    catch { return false; }
  }
  public bool Update(Book newBook) {
    Book dbBook = db.Books.SingleOrDefault(b => b.Code == newBook.Code);
```

```
    if (dbBook != null) {
      try {
        dbBook.Name = newBook.Name;
        dbBook.Author = newBook.Author;
        dbBook.Price = newBook.Price;
        db.SubmitChanges();
        return true;
      }
      catch { return false; }
    }
    return false;
  }
  public bool Delete(int code) {
    Book dbBook = db.Books.SingleOrDefault(b => b.Code == code);
    if (dbBook != null) {
      try {
        db.Books.DeleteOnSubmit(dbBook);
        db.SubmitChanges();
        return true;
      }
      catch { return false; }
    }
    return false;
  }
}
```

2.2 Client-Server architecture

2.2.1 Webform

- Create a new project:

- Project type: ASP.NET Web Application (.NET Framework)
- Project name: BookSite
- Solution name: Webform
- Empty project

BookSite project

- Update Web.config

```
<configuration>
  ...
  <connectionStrings>
    <add name="strCon" providerName="System.Data.SqlClient" connectionString="
       SERVER=den1.mssql7.gear.host;DATABASE=sonkk;USER=sonkk;PASSWORD=pwd" />
  </connectionStrings>
</configuration>
```

- Create BookForm.aspx (presentation page)

```
...
<form id="form1" runat="server">
  <h1>BOOK LIST</h1>
  <asp:TextBox ID="txtKeyword" runat="server"></asp:TextBox>
  <asp:Button ID="btnSearch" runat="server" Text="SEARCH" OnClick="
     btnSearch_Click" /><br/><br/>
  <asp:GridView ID="gvBooks" runat="server" AutoGenerateSelectButton="True"
     OnSelectedIndexChanged="gvBooks_SelectedIndexChanged"></asp:GridView><br/>
  <h1>BOOK DETAILS</h1>
  <table>
    <tr>
      <td>Code</td>
      <td><asp:TextBox ID="txtCode" runat="server" ReadOnly="true"></asp:TextBox
         ></td>
    </tr>
    <tr>
      <td>Name</td>
      <td><asp:TextBox ID="txtName" runat="server"></asp:TextBox></td>
    </tr>
    <tr>
      <td>Author</td>
      <td><asp:TextBox ID="txtAuthor" runat="server"></asp:TextBox></td>
    </tr>
    <tr>
      <td>Price</td>
      <td><asp:TextBox ID="txtPrice" runat="server"></asp:TextBox></td>
    </tr>
  </table><br/>
  <asp:Button ID="btnAdd" runat="server" Text="ADD NEW" OnClick="btnAdd_Click" /
     >
  <asp:Button ID="btnUpdate" runat="server" Text="UPDATE" OnClick="
     btnUpdate_Click" />
  <asp:Button ID="btnDelete" runat="server" Text="DELETE" OnClick="
     btnDelete_Click" OnClientClick="return confirm('ARE YOU SURE?');" />
</form>
...
```

```
...
public partial class Default : System.Web.UI.Page {
  protected void Page_Load(object sender, EventArgs e) {
    if (!Page.IsPostBack) {
      List<Book> books = new BookBUS().GetAll();
      gvBooks.DataSource = books;
      gvBooks.DataBind();
    }
  }
  protected void gvBooks_SelectedIndexChanged(object sender, EventArgs e) {
    txtCode.Text = gvBooks.SelectedRow.Cells[1].Text.Trim();
    txtName.Text = gvBooks.SelectedRow.Cells[2].Text.Trim();
    txtAuthor.Text = gvBooks.SelectedRow.Cells[3].Text.Trim();
```

```
    txtPrice.Text = gvBooks.SelectedRow.Cells[4].Text.Trim();
  }
  protected void btnSearch_Click(object sender, EventArgs e) {
    String keyword = txtKeyword.Text.Trim();
    List<Book> books = new BookBUS().Search(keyword);
    gvBooks.DataSource = books;
    gvBooks.DataBind();
  }
  protected void btnAdd_Click(object sender, EventArgs e) {
    Book newBook = new Book() {
      Code = int.Parse(txtCode.Text.Trim()),
      Name = txtName.Text.Trim(),
      Author = txtAuthor.Text.Trim(),
      Price = int.Parse(txtPrice.Text.Trim())
    };
    bool result = new BookBUS().AddNew(newBook);
    if (result) {
      List<Book> books = new BookBUS().GetAll();
      gvBooks.DataSource = books;
      gvBooks.DataBind();
    }
  }
  protected void btnUpdate_Click(object sender, EventArgs e) {
    Book newBook = new Book() {
      Code = int.Parse(txtCode.Text.Trim()),
      Name = txtName.Text.Trim(),
      Author = txtAuthor.Text.Trim(),
      Price = int.Parse(txtPrice.Text.Trim())
    };
    bool result = new BookBUS().Update(newBook);
    if (result) {
      List<Book> books = new BookBUS().GetAll();
      gvBooks.DataSource = books;
      gvBooks.DataBind();
    }
  }
  protected void btnDelete_Click(object sender, EventArgs e) {
    int code = int.Parse(txtCode.Text.Trim());
    bool result = new BookBUS().Delete(code);
    if (result) {
      List<Book> books = new BookBUS().GetAll();
      gvBooks.DataSource = books;
      gvBooks.DataBind();
    }
  }
}
```

- Deploy on GearHost: Right-click Project » Publish...

  - Target: Import Profile

- Browse to sonkk.publishsettings file
- Edit » Settings » File Publish Options » Check on Remove additional files at destination
- Publish » http://sonkk.gear.host/BookForm.aspx

2.2.2 Remoting

- Create 3 new projects:

- Shared:
  - Project type: Class Library (.NET Framework)
  - Project name: BookShared
  - Solution name: Remoting
- Server:
  - Project type: Console App (.NET Framework)
  - Project name: BookServer
  - Solution name: Remoting
- Client:
  - Project type: Windows Forms App (.NET Framework)
  - Project name: BookClient
  - Solution name: Remoting
- Add Reference BookShared into BookServer and BookClient

BookShared project

- Create Book.cs (DTO class)

```
...
[Serializable]
public class Book {
  public int Code { get; set; }
  public String Name { get; set; }
  public String Author { get; set; }
  public int Price { get; set; }
}
```

OR create MyDB.dbml (LINQ to SQL Classes) » Update MyDB.designer.cs

```
...
[Serializable]
[global::System.Data.Linq.Mapping.TableAttribute(Name="dbo.Book")]
public partial class Book : INotifyPropertyChanging, INotifyPropertyChanged {
  ...
}
...
```

- Create IBookBUS.cs (interface)

```
...
public interface IBookBUS {
  List<Book> GetAll();
  Book GetDetails(int code);
  List<Book> Search(String keyword);
  bool AddNew(Book newBook);
  bool Update(Book newBook);
  bool Delete(int code);
}
```

BookServer project

- Update Program.cs (entry-point)

```
...
using System.Runtime.Remoting; // add reference System.Runtime.Remoting.dll
using System.Runtime.Remoting.Channels;
using System.Runtime.Remoting.Channels.Tcp;

class Program {
  static void Main(string[] args) {
    ChannelServices.RegisterChannel(new TcpChannel(6969), false);
    RemotingConfiguration.RegisterWellKnownServiceType(typeof(BookBUS), "xxx",
       WellKnownObjectMode.SingleCall);
    RemotingConfiguration.CustomErrorsMode = CustomErrorsModes.Off;
    Console.WriteLine("SERVER is started ...");
    Console.Read();
  }
}
```

- Create BookBUS.cs (business-logic class)

```
...
using Remoting.BookShared;
class BookBUS : MarshalByRefObject, IBookBUS {
  ...
}
```

- Create BookDAO.cs (data-access class)

BookClient project

- Create BookForm.cs (presentation class)

```
...
using Remoting.BookShared;

public partial class BookForm : Form {
  const String uri = "tcp://ip_server:6969/xxx";
  IBookBUS bookBUS = (IBookBUS)Activator.GetObject(typeof(IBookBUS), uri);
```

```
7
8   private void BookForm_Load(object sender, EventArgs e) {
9     List<Book> books = bookBUS.GetAll();
10    ...
11  }
12  private void gridBook_SelectionChanged(object sender, EventArgs e) {
13    ...
14    Book book = bookBUS.GetDetails(code);
15    ...
16  }
17  private void btnSearch_Click(object sender, EventArgs e) {
18    ...
19    List<Book> books = bookBUS.Search(keyword);
20    ...
21  }
22  private void btnAdd_Click(object sender, EventArgs e) {
23    ...
24    bool result = bookBUS.AddNew(newBook);
25    if (result) {
26      List<Book> books = bookBUS.GetAll();
27      ...
28    }
29    ...
30  }
31  private void btnUpdate_Click(object sender, EventArgs e) {
32    ...
33    bool result = bookBUS.Update(newBook);
34    if (result) {
35      List<Book> books = bookBUS.GetAll();
36      ...
37    }
38    ...
39  }
40  private void btnDelete_Click(object sender, EventArgs e) {
41    ...
42    bool result = bookBUS.Delete(code);
43    if (result) {
44      List<Book> books = bookBUS.GetAll();
45      ...
46    }
47    ...
48  }
49 }
```

- Deploy on 2 separate machines:

  - Server: start BookServer
  - Client: start BookClient

2.3 Service-Oriented architecture

2.3.1 Windows Service

- Create 3 new projects:

- Shared:
  - Project type: Class Library (.NET Framework)
  - Project name: BookShared
  - Solution name: WindowsService
- Service:
  - Project type: Windows Service (.NET Framework)
  - Project name: BookService
  - Solution name: WindowsService
- Client:
  - Project type: Windows Forms App (.NET Framework)
  - Project name: BookClient
  - Solution name: WindowsService
- Add Reference BookShared into BookService and BookClient

BookService project

- Update MainService.cs (entry-point)

```
...
using System.Runtime.Remoting; // add reference System.Runtime.Remoting.dll
using System.Runtime.Remoting.Channels;
using System.Runtime.Remoting.Channels.Tcp;

public partial class MainService : ServiceBase {
  ...
  protected override void OnStart(string[] args) {
    ChannelServices.RegisterChannel(new TcpChannel(6969), false);
    RemotingConfiguration.RegisterWellKnownServiceType(typeof(BookBUS), "xxx",
       WellKnownObjectMode.SingleCall);
    RemotingConfiguration.CustomErrorsMode = CustomErrorsModes.Off;
  }
  ...
}
```

- Open MainService.cs [Design] » Right-click » Add Installer

- serviceInstaller1:
  - ServiceName: BookService
  - StartType: Automatic
- serviceProcessInstaller1:

  - Account: LocalSystem

- Install BookService: open Command Prompt (Run as administrator)

```
cd C:\Windows\Microsoft.NET\Framework64\v4.0.30319
InstallUtil /i $PROJECT_PATH$/bin/Debug/BookService.exe
```

- Uninstall BookService: open Command Prompt (Run as administrator)

```
cd C:\Windows\Microsoft.NET\Framework64\v4.0.30319
InstallUtil /u $PROJECT_PATH$/bin/Debug/BookService.exe
```

- Deploy on 2 separate machines:

  - Server: start BookService in Windows services (OR restart Server)
  - Client: start BookClient

2.3.2 SOAP Service

- Create 4 new projects:

  - Service:
    - Project type: ASP.NET Web Application (.NET Framework)
    - Project name: BookSite
    - Solution name: SoapService
    - Empty project
  - Winform Client:
    - Project type: Windows Forms App (.NET Framework)
    - Project name: BookClient
    - Solution name: SoapService
  - HTML Client:
    - Project type: HTML + Javascript
    - Project name: HtmlClient
    - Solution name: SoapService
  - Mobile Client:
    - Project type: React Native
    - Project name: MobileClient
    - Solution name: SoapService

BookSite project

- Create BookService.asmx (soap-service)

```
...
[WebService(Namespace = "http://sonkk.org/")]
public class BookService : System.Web.Services.WebService {
  [WebMethod]
  public List<Book> GetAll() {
    List<Book> books = new BookDAO().SelectAll();
    return books;
  }
  [WebMethod]
  public Book GetDetails(int code) {
    Book book = new BookDAO().SelectByCode(code);
    return book;
  }
  [WebMethod]
  public List<Book> Search(String keyword) {
    List < Book > books = new BookDAO().SelectByKeyword(keyword);
    return books;
  }
  [WebMethod]
  public bool AddNew(Book newBook) {
    bool result = new BookDAO().Insert(newBook);
    return result;
  }
  [WebMethod]
  public bool Update(Book newBook) {
    bool result = new BookDAO().Update(newBook);
    return result;
  }
  [WebMethod]
  public bool Delete(int code) {
    bool result = new BookDAO().Delete(code);
    return result;
  }
}
```

- Run on localhost: Ctrl+F5 » https://localhost:port/BookService.asmx
- Deploy on GearHost: Right-click Project » Publish... » http://sonkk.gear.host/BookService.asmx

- Postman:

- (POST) http://sonkk.gear.host/BookService.asmx
- Header: Content-Type: text/xml
- Body (raw+XML):
  - GetAll

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://
   schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <GetAll xmlns="http://sonkk.org/" />
  </soap:Body>
</soap:Envelope>
```

– GetDetails

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://
   schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <GetDetails xmlns="http://sonkk.org/">
      <code>1</code>
    </GetDetails>
  </soap:Body>
</soap:Envelope>
```

– Search

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://
   schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <Search xmlns="http://sonkk.org/">
      <keyword>a</keyword>
    </Search>
  </soap:Body>
</soap:Envelope>
```

– AddNew

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://
   schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <AddNew xmlns="http://sonkk.org/">
      <newBook>
        <Code>4</Code>
        <Name>MAD</Name>
        <Author>HSU4</Author>
        <Price>444</Price>
      </newBook>
    </AddNew>
  </soap:Body>
</soap:Envelope>
```

– Update

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://
   schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <Update xmlns="http://sonkk.org/">
      <newBook>
        <Code>4</Code>
        <Name>ADM</Name>
        <Author>4HSU</Author>
        <Price>123</Price>
      </newBook>
    </Update>
  </soap:Body>
</soap:Envelope>
```

– Delete

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://
   schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <Delete xmlns="http://sonkk.org/">
      <code>4</code>
    </Delete>
  </soap:Body>
</soap:Envelope>
```

BookClient project

- Add reference to BookService:

- Add Service Reference » Advanced » Add Web Reference...
- URL: http://sonkk.gear.host/BookService.asmx » Go
- Web reference name: gearhost » Add Reference

- Create BookForm.cs (presentation class)

```
...
public partial class BookForm : Form {
  gearhost.BookService bookService = new gearhost.BookService();

  private void BookForm_Load(object sender, EventArgs e) {
    List<gearhost.Book> books = bookService.GetAll();
    ...
  }
  private void gridBook_SelectionChanged(object sender, EventArgs e) {
    ...
```

```
    gearhost.Book book = bookService.GetDetails(code);
    ...
  }
  private void btnSearch_Click(object sender, EventArgs e) {
    ...
    List<gearhost.Book> books = bookService.Search(keyword);
    ...
  }
  private void btnAdd_Click(object sender, EventArgs e) {
    ...
    bool result = bookService.AddNew(newBook);
    if (result) {
      List<gearhost.Book> books = bookService.GetAll();
      ...
    }
    ...
  }
  private void btnUpdate_Click(object sender, EventArgs e) {
    ...
    bool result = bookService.Update(newBook);
    if (result) {
      List<gearhost.Book> books = bookService.GetAll();
      ...
    }
    ...
  }
  private void btnDelete_Click(object sender, EventArgs e) {
    ...
    bool result = bookService.Delete(code);
    if (result) {
      List<gearhost.Book> books = bookService.GetAll();
      ...
    }
    ...
  }
}
```

HtmlClient project

- Create books.html (presentation page)

```
<html>
<head>
  <script src="https://unpkg.com/axios/dist/axios.min.js"></script>
  <script src="https://cdn.rawgit.com/abdmob/x2js/master/xml2json.js"></script>
  <script src="books.js"></script>
</head>
<body onload="page_Load()">
  <h2>BOOK LIST</h2>
  <input type="text" id="txtKeyword" placeholder="Keyword" /> 
  <input type="button" value="SEARCH" onclick="btnSearch_Click()" /><br/><br/>
```

```
  <table id="lstBooks" border="1"></table><br/>
  <h2>BOOK DETAILS</h2>
  <table>
    <tr>
      <td>Code</td>
      <td><input type="text" id="txtCode" readonly="true" /></td>
    </tr>
    <tr>
      <td>Name</td>
      <td><input type="text" id="txtName" /></td>
    </tr>
    <tr>
      <td>Author</td>
      <td><input type="text" id="txtAuthor" /></td>
    </tr>
    <tr>
      <td>Price</td>
      <td><input type="text" id="txtPrice" /></td>
    </tr>
    <tr>
      <td colspan="2">
        <input type="button" value="ADD NEW" onclick="btnAdd_Click()" />
        <input type="button" value="UPDATE" onclick="btnUpdate_Click()" />
        <input type="button" value="DELETE" onclick="btnDelete_Click()" />
      </td>
    </tr>
  </table>
</body>
</html>
```

- Create books.js (event-handlers)

```
const CORS = "https://cors-anywhere.herokuapp.com/";
const URI = "http://sonkk.gear.host/BookService.asmx";
const config = { headers: { 'Content-Type': 'text/xml' } };

/* event-handler methods */
function page_Load() {
  getAll();
}
function lnkID_Click(code) {
  getDetails(code);
}
function btnSearch_Click() {
  var keyword = document.getElementById("txtKeyword").value.trim();
  if (keyword.length > 0) {
    search(keyword);
  } else {
    getAll();
  }
}
```

```
function btnAdd_Click() {
  var newBook = {
    Code: 0,
    Name: document.getElementById("txtName").value,
    Author: document.getElementById("txtAuthor").value,
    Price: document.getElementById("txtPrice").value
  };
  addNew(newBook);
}
function btnUpdate_Click() {
  var newBook = {
    Code: document.getElementById("txtCode").value,
    Name: document.getElementById("txtName").value,
    Author: document.getElementById("txtAuthor").value,
    Price: document.getElementById("txtPrice").value
  };
  update(newBook);
}
function btnDelete_Click() {
  if (confirm("ARE YOU SURE ?")) {
    var code = document.getElementById("txtCode").value;
    delete(code);
  }
}
/* fetch SOAP methods */
function getAll() {
  var body = `<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="
   http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap
   /envelope/">
  <soap:Body>
    <GetAll xmlns="http://sonkk.org/" />
  </soap:Body>
</soap:Envelope>`;
  axios.post(CORS + URI + "?op=GetAll", body, config).then((response) => {
    var xmlData = response.data;
    //alert(xmlData); // for DEBUG
    var jsonData = new X2JS().xml_str2json(xmlData);
    //alert(JSON.stringify(jsonData)); // for DEBUG
    var books = jsonData.Envelope.Body.GetAllResponse.GetAllResult.Book;
    //alert(JSON.stringify(books)); // for DEBUG
    renderBookList(books);
  });
}
function getDetails(code) {
  var body = `<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="
   http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap
   /envelope/">
  <soap:Body>
    <GetDetails xmlns="http://sonkk.org/">
```

```
      <code>${code}</code>
    </GetDetails>
  </soap:Body>
</soap:Envelope>`;
  axios.post(CORS + URI + "?op=GetDetails", body, config).then((response) => {
    var xmlData = response.data;
    var jsonData = new X2JS().xml_str2json(xmlData);
    var book = jsonData.Envelope.Body.GetDetailsResponse.GetDetailsResult;
    renderBookDetails(book);
  });
}
function search(keyword) {
  var body = `<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="
   http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap
   /envelope/">
  <soap:Body>
    <Search xmlns="http://sonkk.org/">
      <keyword>${keyword}</keyword>
    </Search>
  </soap:Body>
</soap:Envelope>`;
  axios.post(CORS + URI + "?op=Search", body, config).then((response) => {
    var xmlData = response.data;
    var jsonData = new X2JS().xml_str2json(xmlData);
    var books = jsonData.Envelope.Body.SearchResponse.SearchResult.Book;
    renderBookList(books);
  });
}
function addNew(newBook) {
  var body = `<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="
   http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap
   /envelope/">
  <soap:Body>
    <AddNew xmlns="http://sonkk.org/">
      <newBook>
        <Code>${newBook.Code}</Code>
        <Name>${newBook.Name}</Name>
        <Author>${newBook.Author}</Author>
        <Price>${newBook.Price}</Price>
      </newBook>
    </AddNew>
  </soap:Body>
</soap:Envelope>`;
  axios.post(CORS + URI + "?op=AddNew", body, config).then((response) => {
    var xmlData = response.data;
    var jsonData = new X2JS().xml_str2json(xmlData);
    var result = jsonData.Envelope.Body.AddNewResponse.AddNewResult;
    if (result) {
      getAll();
```

```
      clearTextboxes();
    } else {
      alert('SORRY BABY!');
    }
  });
}
function update(newBook) {
  var body = `<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="
   http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap
   /envelope/">
  <soap:Body>
    <Update xmlns="http://sonkk.org/">
      <newBook>
        <Code>${newBook.Code}</Code>
        <Name>${newBook.Name}</Name>
        <Author>${newBook.Author}</Author>
        <Price>${newBook.Price}</Price>
      </newBook>
    </Update>
  </soap:Body>
</soap:Envelope>`;
  axios.post(CORS + URI + "?op=Update", body, config).then((response) => {
    var xmlData = response.data;
    var jsonData = new X2JS().xml_str2json(xmlData);
    var result = jsonData.Envelope.Body.UpdateResponse.UpdateResult;
    if (result) {
      getAll();
      clearTextboxes();
    } else {
      alert('SORRY BABY!');
    }
  });
}
function delete(code) {
  var body = `<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="
   http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap
   /envelope/">
  <soap:Body>
    <Delete xmlns="http://sonkk.org/">
      <code>${code}</code>
    </Delete>
  </soap:Body>
</soap:Envelope>`;
  axios.post(CORS + URI + "?op=Delete", body, config).then((response) => {
    var xmlData = response.data;
    var jsonData = new X2JS().xml_str2json(xmlData);
    var result = jsonData.Envelope.Body.DeleteResponse.DeleteResult;
    if (result) {
      getAll();
```

```
      clearTextboxes();
    } else {
      alert('SORRY BABY!');
    }
  });
}
/* helper methods */
function renderBookList(books) {
  var rows = "";
  for (var book of books) {
    rows += "<tr onclick='lnkID_Click(" + book.Code + ")' style='cursor:pointer
       '>";
    rows += "<td>" + book.Code + "</td>";
    rows += "<td>" + book.Name + "</td>";
    rows += "<td>" + book.Author + "</td>";
    rows += "<td>" + book.Price + "</td>";
    rows += "</tr>";
  }
  var header = "<tr><th>Code</th><th>Name</th><th>Author</th><th>Price</th></tr>
     ";
  document.getElementById("lstBooks").innerHTML = header + rows;
}
function renderBookDetails(book) {
  document.getElementById("txtCode").value = book.Code;
  document.getElementById("txtName").value = book.Name;
  document.getElementById("txtAuthor").value = book.Author;
  document.getElementById("txtPrice").value = book.Price;
}
function clearTextboxes() {
  document.getElementById("txtCode").value = '';
  document.getElementById("txtName").value = '';
  document.getElementById("txtAuthor").value = '';
  document.getElementById("txtPrice").value = '';
}
```

MobileClient project

- Create React Native project by Expo-cli

```
expo init MobileClient
cd MobileClient
npm install axios --save
npm install react-native-xml2js --save
expo start -c
```

- Update App.js (entry-point)

```
import React, { Component } from 'react';
import Book from './components/BookComponent';

class App extends Component {
```

```
  render() {
    return (<Book />);
  }
}
export default App;
```

- Create components/BookComponent.js

```
import React, { Component } from 'react';
import { StyleSheet, View, ActivityIndicator, FlatList, TouchableOpacity, Text,
   TextInput, Button, Alert } from 'react-native';
import axios from 'axios';
import xml2js from 'react-native-xml2js';

class Book extends Component {
  constructor(props) {
    super(props);
    this.URI = 'http://sonkk.gear.host/BookService.asmx';
    this.config = { headers: { 'Content-Type': 'text/xml' } };
    this.state = { isLoading: true, dataSource: [], txtCode: '', txtName: '',
       txtAuthor: '', txtPrice: '' };
  }
  render() {
    if (this.state.isLoading) {
      return (
        <View style={styles.loading}>
          <ActivityIndicator />
        </View>
      );
    } else {
      return (
        <View style={styles.container}>
          <View style={styles.bookList}>
            <Text>BOOK LIST</Text>
            <FlatList contentContainerStyle={styles.flatList}
              data={this.state.dataSource}
              keyExtractor={(item) => item.Code.toString()}
              renderItem={({ item }) => (
                <TouchableOpacity onPress={() => this.pressOnRow(item)}>
                  <Text style={styles.rowItem}>{item.Code} - {item.Name} - {item
                     .Author} - {item.Price}</Text>
                </TouchableOpacity>
              )} />
          </View>
          <View style={styles.line} />
          <View style={styles.bookDetails}>
            <Text>BOOK DETAILS</Text>
            <TextInput style={styles.input} underlineColorAndroid='transparent'
               placeholder='Code' autoCapitalize='none' editable={false}
              value={this.state.txtCode} onChangeText={(txtCode) => this.
                 setState({ txtCode })} />
```

```
39            <TextInput style={styles.input} underlineColorAndroid='transparent'
                 placeholder='Name' autoCapitalize='none'
40              value={this.state.txtName} onChangeText={(txtName) => this.
                   setState({ txtName })} />
41            <TextInput style={styles.input} underlineColorAndroid='transparent'
                 placeholder='Author' autoCapitalize='none'
42              value={this.state.txtAuthor} onChangeText={(txtAuthor) => this.
                   setState({ txtAuthor })} />
43            <TextInput style={styles.input} underlineColorAndroid='transparent'
                 placeholder='Price' autoCapitalize='none'
44              value={this.state.txtPrice} onChangeText={(txtPrice) => this.
                   setState({ txtPrice })} />
45            <View style={styles.buttons}>
46              <Button title='ADD NEW' onPress={this.btnAdd} />
47              <Button title='UPDATE' onPress={this.btnUpdate} />
48              <Button title='DELETE' onPress={this.btnDelete} />
49            </View>
50          </View>
51        </View>
52      );
53    }
54  }
55  componentDidMount() {
56    this.getAll();
57  }
58  /* event-handler methods */
59  pressOnRow(item) {
60    var code = item.Code;
61    this.getDetails(code);
62  }
63  btnAdd = async () => {
64    var newBook = { Code: 0, Name: this.state.txtName, Author: this.state.
         txtAuthor, Price: this.state.txtPrice };
65    this.addNew(newBook);
66  }
67  btnUpdate = () => {
68    var newBook = { Code: this.state.txtCode, Name: this.state.txtName, Author:
         this.state.txtAuthor, Price: this.state.txtPrice };
69    this.update(newBook);
70  }
71  btnDelete = () => {
72    Alert.alert('', 'ARE YOU SURE ?',
73      [
74        { text: 'CANCEL', style: 'cancel' },
75        { text: 'OK', onPress: () => this.delete(this.state.txtCode) }
76      ],
77      { cancelable: false }
78    );
79  }
80  /* fetch SOAP methods */
81  async getAll() {
```

```
    var body = `<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="
   http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap
   /envelope/">
  <soap:Body>
    <GetAll xmlns="http://sonkk.org/" />
  </soap:Body>
</soap:Envelope>`;
    axios.post(this.URI + '?op=GetAll', body, this.config).then((response) => {
      var xmlData = response.data;
      xml2js.parseString(xmlData, { explicitArray: false, ignoreAttrs: true }, (
         err, result) => {
        var books = result['soap:Envelope']['soap:Body'].GetAllResponse.
           GetAllResult.Book;
        this.setState({ isLoading: false, dataSource: books });
      });
    });
  }
  async getDetails(code) {
    var body = `<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="
   http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap
   /envelope/">
  <soap:Body>
    <GetDetails xmlns="http://sonkk.org/">
      <code>${code}</code>
    </GetDetails>
  </soap:Body>
</soap:Envelope>`;
    axios.post(this.URI + '?op=GetDetails', body, this.config).then((response)
       => {
      var xmlData = response.data;
      xml2js.parseString(xmlData, { explicitArray: false, ignoreAttrs: true }, (
         err, result) => {
        var book = result['soap:Envelope']['soap:Body'].GetDetailsResponse.
           GetDetailsResult;
        this.setState({ txtCode: book.Code.toString(), txtName: book.Name,
           txtAuthor: book.Author, txtPrice: book.Price.toString() });
      });
    });
  }
  async addNew(newBook) {
    var body = `<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="
   http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap
   /envelope/">
  <soap:Body>
    <AddNew xmlns="http://sonkk.org/">
      <newBook>
        <Code>${newBook.Code}</Code>
        <Name>${newBook.Name}</Name>
```

```
        <Author>${newBook.Author}</Author>
        <Price>${newBook.Price}</Price>
      </newBook>
    </AddNew>
  </soap:Body>
</soap:Envelope>`;
    axios.post(this.URI + '?op=AddNew', body, this.config).then((response) => {
      var xmlData = response.data;
      xml2js.parseString(xmlData, { explicitArray: false, ignoreAttrs: true }, (
         err, result) => {
        var result = result['soap:Envelope']['soap:Body'].AddNewResponse.
           AddNewResult;
        if (result) {
          alert('OK BABY!');
          this.getAll();
        } else {
          alert('SORRY BABY!');
        }
      });
    });
  }
  async update(newBook) {
    var body = `<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="
   http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap
   /envelope/">
  <soap:Body>
    <Update xmlns="http://sonkk.org/">
      <newBook>
        <Code>${newBook.Code}</Code>
        <Name>${newBook.Name}</Name>
        <Author>${newBook.Author}</Author>
        <Price>${newBook.Price}</Price>
      </newBook>
    </Update>
  </soap:Body>
</soap:Envelope>`;
    axios.post(this.URI + '?op=Update', body, this.config).then((response) => {
      var xmlData = response.data;
      xml2js.parseString(xmlData, { explicitArray: false, ignoreAttrs: true }, (
         err, result) => {
        var result = result['soap:Envelope']['soap:Body'].UpdateResponse.
           UpdateResult;
        if (result) {
          alert('OK BABY!');
          this.getAll();
        } else {
          alert('SORRY BABY!');
        }
      });
    });
```

```
  }
  async delete(code) {
    var body = `<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="
   http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap
   /envelope/">
  <soap:Body>
    <Delete xmlns="http://sonkk.org/">
      <code>${code}</code>
    </Delete>
  </soap:Body>
</soap:Envelope>`;
    axios.post(this.URI + '?op=Delete', body, this.config).then((response) => {
      var xmlData = response.data;
      xml2js.parseString(xmlData, { explicitArray: false, ignoreAttrs: true }, (
         err, result) => {
        var result = result['soap:Envelope']['soap:Body'].DeleteResponse.
           DeleteResult;
        if (result) {
          alert('OK BABY!');
          this.getAll();
        } else {
          alert('SORRY BABY!');
        }
      });
    });
  }
}
export default Book;

const styles = StyleSheet.create({
  loading: { flex: 1, justifyContent: 'center' },
  container: { flex: 1, marginTop: 40, marginLeft: 10, marginRight: 10,
     marginBottom: 20, borderWidth: 1 },
  line: { marginTop: 10, marginBottom: 10, height: 1, backgroundColor: 'black'
     },
  bookList: { marginTop: 10, alignItems: 'center', height: 180 },
  flatList: { justifyContent: 'center' },
  rowItem: { marginTop: 10 },
  bookDetails: { alignItems: 'center' },
  input: { marginTop: 10, padding: 5, width: 250, height: 40, borderWidth: 1 },
  buttons: { flexDirection: 'row', marginTop: 10 }
});
```

2.3.3 RESTful Service

- Create 4 new projects:

- Service:
  - Project type: ASP.NET Web Application (.NET Framework)

  - Project name: BookSite
  - Solution name: RestfulService
  - Empty project » Check on Web API

- Winform Client:
  - Project type: Windows Forms App (.NET Framework)
  - Project name: BookClient
  - Solution name: RestfulService

- HTML Client:
  - Project type: HTML + Javascript
  - Project name: HtmlClient
  - Solution name: RestfulService

- Mobile Client:
  - Project type: React Native
  - Project name: MobileClient
  - Solution name: RestfulService

BookSite project

- Create BooksController.cs (Web API controller)

```
...
public class BooksController : ApiController {
  [HttpGet]
  [Route("api/books")]
  public List<Book> GetAll() {
    List<Book> books = new BookDAO().SelectAll();
    return books;
  }
  [HttpGet]
  [Route("api/books/{code}")]
  public Book GetDetails(int code) {
    Book book = new BookDAO().SelectByCode(code);
    return book;
  }
  [HttpGet]
  [Route("api/books/search/{keyword}")]
  public List<Book> Search(String keyword) {
    List<Book> books = new BookDAO().SelectByKeyword(keyword);
    return books;
  }
  [HttpPost]
  [Route("api/books")]
  public bool AddNew(Book newBook) {
    bool result = new BookDAO().Insert(newBook);
    return result;
```

```
  }
  [HttpPut]
  [Route("api/books/{code}")]
  public bool Update(int code, Book newBook) {
    if (code != newBook.Code) return false;
    bool result = new BookDAO().Update(newBook);
    return result;
  }
  [HttpDelete]
  [Route("api/books/{code}")]
  public bool Delete(int code) {
    bool result = new BookDAO().Delete(code);
    return result;
  }
```

- Run on localhost: Ctrl+F5 » https://localhost:port/api/books
- Deploy on GearHost: Right-click Project » Publish... » http://sonkk.gear.host/api/books

- Postman:

  - GetAll
    - (GET) http://sonkk.gear.host/api/books
  - GetDetails
    - (GET) http://sonkk.gear.host/api/books/1
  - Search
    - (GET) http://sonkk.gear.host/api/books/search/a
  - AddNew
    - (POST) http://sonkk.gear.host/api/books
    - Body (raw+JSON): { "Code": 4, "Name": "MAD", "Author": "HSU4", "Price": 444 }
  - Update
    - (PUT) http://sonkk.gear.host/api/books/4
    - Body (raw+JSON): { "Code": 4, "Name": "ADM", "Author": "4HSU", "Price": 123 }
  - Delete
    - (DELETE) http://sonkk.gear.host/api/books/4

BookClient project

- Install Newtonsoft.Json (James Newton-King) library:

  - Right-click project » Manage NuGet Packages...
  - Browse tab » Search "Newtonsoft.Json" » Install

- Create Book.cs (DTO class)

```
...
class Book {
  public int Code { get; set; }
  public String Name { get; set; }
  public String Author { get; set; }
  public int Price { get; set; }
}
```

- Create BookForm.cs (presentation class)

```
...
public partial class BookForm : Form {
  private void BookForm_Load(object sender, EventArgs e) {
    List<Book> books = new BookBUS().GetAll();
    ...
  }
  private void gridBook_SelectionChanged(object sender, EventArgs e) {
    ...
    Book book = new BookBUS().GetDetails(code);
    ...
  }

  private void btnSearch_Click(object sender, EventArgs e) {
    ...
    List<Book> books = new List<Book>();
    if (keyword.Length > 0) {
      books = new BookBUS().Search(keyword);
    } else {
      books = new BookBUS().GetAll();
    }
    ...
  }

  private void btnAdd_Click(object sender, EventArgs e) {
    ...
    bool result = new BookBUS().AddNew(newBook);
    if (result) {
      List<Book> books = new BookBUS().GetAll();
      ...
    }
    ...
  }

  private void btnUpdate_Click(object sender, EventArgs e) {
    ...
    bool result = new BookBUS().Update(newBook);
    if (result) {
      List<Book> books = new BookBUS().GetAll();
      ...
```

```
    }
    ...
  }

  private void btnDelete_Click(object sender, EventArgs e) {
    ...
    bool result = new BookBUS().Delete(code);
    if (result) {
      List<Book> books = new BookBUS().GetAll();
      ...
    }
    ...
  }
}
```

- Create BookBUS.cs (business-logic class)

```
...
using System.Net;
using Newtonsoft.Json;

class BookBUS {
  String URI = "http://sonkk.gear.host/api/books";

  public List<Book> GetAll() {
    WebClient client = new WebClient();
    String response = client.DownloadString(URI);
    return JsonConvert.DeserializeObject<List<Book>>(response);
  }
  public Book GetDetails(int code) {
    WebClient client = new WebClient();
    String response = client.DownloadString(URI + "/" + code);
    return JsonConvert.DeserializeObject<Book>(response);
  }
  public List<Book> Search(String keyword) {
    WebClient client = new WebClient();
    String response = client.DownloadString(URI + "/search/" + keyword);
    return JsonConvert.DeserializeObject<List<Book>>(response);
  }
  public bool AddNew(Book newBook) {
    String data = JsonConvert.SerializeObject(newBook);
    WebClient client = new WebClient();
    client.Headers[HttpRequestHeader.ContentType] = "application/json";
    String response = client.UploadString(URI, "POST", data);
    return bool.Parse(response);
  }
  public bool Update(Book newBook) {
    String data = JsonConvert.SerializeObject(newBook);
    WebClient client = new WebClient();
    client.Headers[HttpRequestHeader.ContentType] = "application/json";
    String response = client.UploadString(URI + "/" + newBook.Code, "PUT",data);
```

```
    return bool.Parse(response);
  }
  public bool Delete(int code) {
    WebClient client = new WebClient();
    String response = client.UploadString(URI + "/" + code, "DELETE", "");
    return bool.Parse(response);
  }
}
```

HtmlClient project

- Create books.html (presentation page)

```
<html>
<head>
  <script src="https://unpkg.com/axios/dist/axios.min.js"></script>
  <script src="books.js"></script>
</head>
<body onload="page_Load()">
  ...
</body>
</html>
```

- Create books.js (event-handlers)

```
const CORS = "https://cors-anywhere.herokuapp.com/";
const URI = "http://sonkk.gear.host/api/books";

/* event-handler methods */
...
/* fetch API methods */
function getAll() {
  axios.get(CORS + URI).then((response) => {
    var books = response.data;
    renderBookList(books);
  });
}
function getDetails(code) {
  axios.get(CORS + URI + "/" + code).then((response) => {
    var book = response.data;
    renderBookDetails(book);
  });
}
function search(keyword) {
  axios.get(CORS + URI + "/search/" + keyword).then((response) => {
    var books = response.data;
    renderBookList(books);
  });
}
function addNew(newBook) {
  axios.post(CORS + URI, newBook).then((response) => {
```

```
    var result = response.data;
    if (result) {
      getAll();
      clearTextboxes();
    } else {
      alert('SORRY BABY!');
    }
  });
}
function update(newBook) {
  axios.put(CORS + URI + "/" + newBook.Code, newBook).then((response) => {
    var result = response.data;
    if (result) {
      getAll();
      clearTextboxes();
    } else {
      alert('SORRY BABY!');
    }
  });
}
function delete(code) {
  axios.delete(CORS + URI + "/" + code).then((response) => {
    var result = response.data;
    if (result) {
      getAll();
      clearTextboxes();
    } else {
      alert('SORRY BABY!');
    }
  });
}
/* helper methods */
...
```

MobileClient project

- Create React Native project by Expo-cli

```
expo init MobileClient
cd MobileClient
expo start -c
```

- Update App.js (entry-point)

```
import React, { Component } from 'react';
import Book from './components/BookComponent';

class App extends Component {
  render() {
    return (<Book />);
  }
```

```
}
export default App;
```

- Create components/BookComponent.js

```
import React, { Component } from 'react';
import { StyleSheet, View, ActivityIndicator, FlatList, TouchableOpacity, Text,
   TextInput, Button, Alert } from 'react-native';

class Book extends Component {
  constructor(props) {
    super(props);
    this.URI = 'http://sonkk.gear.host/api/books';
    this.state = { isLoading: true, dataSource: [], txtCode: '', txtName: '',
       txtAuthor: '', txtPrice: '' };
  }
  render() {
    ...
  }
  componentDidMount() {
    ...
  }
  /* event-handler methods */
  pressOnRow(item) {
    ...
  }
  btnAdd = async () => {
    ...
  }
  btnUpdate = () => {
    ...
  }
  btnDelete = () => {
    ...
  }
  /* fetch API methods */
  async getAll() {
    var response = await fetch(this.URI);
    var books = await response.json();
    this.setState({ isLoading: false, dataSource: books });
  }
  async getDetails(code) {
    var response = await fetch(this.URI + '/' + code);
    var book = await response.json();
    this.setState({ txtCode: book.Code.toString(), txtName: book.Name, txtAuthor
       : book.Author, txtPrice: book.Price.toString() });
  }
  async addNew(newBook) {
    var response = await fetch(this.URI, { method: 'POST', headers: { 'Content-
       Type': 'application/json' }, body: JSON.stringify(newBook) });
    var result = await response.json();
```

```
    if (result) {
      alert('OK BABY!');
      this.getAll();
    } else {
      alert('SORRY BABY!');
    }
  }
  async update(newBook) {
    var response = await fetch(this.URI + '/' + newBook.Code, { method: 'PUT',
       headers: { 'Content-Type': 'application/json' }, body: JSON.stringify(
       newBook) });
    var result = await response.json();
    if (result) {
      alert('OK BABY!');
      this.getAll();
    } else {
      alert('SORRY BABY!');
    }
  }
  async delete(code) {
    var response = await fetch(this.URI + '/' + code, { method: 'DELETE' });
    var result = await response.json();
    if (result) {
      alert('OK BABY!');
      this.getAll();
    } else {
      alert('SORRY BABY!');
    }
  }
}
export default Book;

const styles = StyleSheet.create({
  ...
});
```

2.3.4 Backend as a Service

- Create 3 new projects:

- Winform Client:
  - Project type: Windows Forms App (.NET Framework)
  - Project name: BookClient
  - Solution name: BaaS
- HTML Client:
  - Project type: HTML + Javascript
  - Project name: HtmlClient
  - Solution name: BaaS

- Mobile Client:
  - Project type: React Native
  - Project name: MobileClient
  - Solution name: BaaS

BookClient project

- Install FireSharp (ziyasal) library:

- Right-click project » Manage NuGet Packages...
- Browse tab » Search "FireSharp" » Install

- Create Book.cs (DTO class)

```
...
class Book {
  public int Code { get; set; }
  public String Name { get; set; }
  public String Author { get; set; }
  public int Price { get; set; }
}
```

- Create BookForm.cs (presentation class)

```
...
public partial class BookForm : Form {
  private void BookForm_Load(object sender, EventArgs e) {
    new BookBUS().ListenFirebase(gridBook);
  }
  private void gridBook_SelectionChanged(object sender, EventArgs e) {
    ...
    Book book = new BookBUS().GetDetails(code);
    ...
  }
  private void btnSearch_Click(object sender, EventArgs e) {
    ...
    List<Book> books = new BookBUS().Search(keyword);
    gridBook.BeginInvoke(new MethodInvoker(delegate { gridBook.DataSource =
       books; })); // set asynchronous datasource
  }
  private void btnAdd_Click(object sender, EventArgs e) {
    ...
    bool result = new BookBUS().AddNew(newBook);
    if (!result) MessageBox.Show("SORRY BABY !!!");
  }
  private void btnUpdate_Click(object sender, EventArgs e) {
    ...
    bool result = new BookBUS().Update(newBook);
    if (!result) MessageBox.Show("SORRY BABY !!!");
```

```
  }
  private void btnDelete_Click(object sender, EventArgs e) {
    ...
    bool result = new BookBUS().Delete(code);
    if (!result) MessageBox.Show("SORRY BABY !!!");
  }
}
```

- Create BookBUS.cs (business-logic class)

```
...
using FireSharp;
using FireSharp.Config;
using FireSharp.Interfaces;
using FireSharp.Response;

class BookBUS {
  static IFirebaseConfig config = new FirebaseConfig { BasePath = "https://sonkk
     .firebaseio.com" };
  static FirebaseClient client = new FirebaseClient(config);

  public async void ListenFirebase(DataGridView gridBook) {
    EventStreamResponse response = await client.OnAsync("books",
      added: (sender, args, context) => { UpdateDataGridView(gridBook); },
      changed: (sender, args, context) => { UpdateDataGridView(gridBook); },
      removed: (sender, args, context) => { UpdateDataGridView(gridBook); }
    );
  }
  private void UpdateDataGridView(DataGridView gridBook) {
    List<Book> books = GetAll();
    gridBook.BeginInvoke(new MethodInvoker(delegate { gridBook.DataSource =
       books; })); // set asynchronous datasource
  }
  private List<Book> GetAll() {
    FirebaseResponse response = client.Get("books");
    Dictionary<String, Book> dictBooks = response.ResultAs<Dictionary<String,
       Book>>();
    return dictBooks.Values.ToList();
  }
  public Book GetDetails(int code) {
    FirebaseResponse response = client.Get("books/B" + code);
    Book book = response.ResultAs<Book>();
    return book;
  }
  private String GetKeyByCode(int code) {
    FirebaseResponse response = client.Get("books");
    Dictionary<String, Book> dictBooks = response.ResultAs<Dictionary<String,
       Book>>();
    String key = dictBooks.FirstOrDefault(x => x.Value.Code == code).Key;
    return key;
  }
```

```
  public List<Book> Search(String keyword) {
    List<Book> books = new List<Book>();
    foreach (var item in GetAll()) {
      if (item.Name.ToLower().Contains(keyword.ToLower())) {
        books.Add(item);
      }
    }
    return books;
  }
  public bool AddNew(Book newBook) {
    try {
      //client.Push("books", newBook); // auto-generated key
      client.Set("books/B" + newBook.Code, newBook); // custom key
      return true;
    } catch { return false; }
  }
  public bool Update(Book newBook) {
    try {
      String key = GetKeyByCode(newBook.Code);
      if (String.IsNullOrEmpty(key)) return false;
      client.Set("books/" + key, newBook);
      return true;
    } catch { return false; }
  }
  public bool Delete(int code) {
    try {
      String key = GetKeyByCode(code);
      if (String.IsNullOrEmpty(key)) return false;
      client.Delete("books/" + key);
      return true;
    } catch { return false; }
  }
}
```

HtmlClient project

- Create books.html (presentation page)

```
<html>
<head>
  <script src="https://www.gstatic.com/firebasejs/8.3.1/firebase.js"></script>
  <script src="books.js"></script>
</head>
<body onload="page_Load()">
  ...
</body>
</html>
```

- Create books.js (event-handlers)

```
const config = { databaseURL: "https://sonkk.firebaseio.com" };
```

```
firebase.initializeApp(config);
const dbRef = firebase.database().ref();

/* event-handler methods */
function btnAdd_Click() {
  var newBook = {
    Code: document.getElementById("txtCode").value,
    Name: document.getElementById("txtName").value,
    Author: document.getElementById("txtAuthor").value,
    Price: document.getElementById("txtPrice").value
  };
  addNew(newBook);
}
...
/* firebase methods */
function getAll() {
  dbRef.child("books").on("value", (snapshot) => {
    var books = [];
    snapshot.forEach((child) => {
      //alert(child.key);
      var book = child.val();
      books.push(book);
    });
    renderBookList(books);
  });
}
function getDetails(code) {
  dbRef.child("books").once("value", (snapshot) => {
    snapshot.forEach((child) => {
      var book = child.val();
      if (book.Code == code) {
        renderBookDetails(book);
      }
    });
  });
}
function search(keyword) {
  dbRef.child("books").once("value", (snapshot) => {
    var books = [];
    snapshot.forEach((child) => {
      var book = child.val();
      if (book.Name.toLowerCase().includes(keyword.toLowerCase())) {
        books.push(book);
      }
    });
    renderBookList(books);
  });
}
function addNew(newBook) {
  //dbRef.child("books").push(newBook); // auto-generated key
  dbRef.child("books/B" + newBook.Code).set(newBook); // custom key
```

```
}
function update(newBook) {
  dbRef.child("books").once("value", (snapshot) => {
    snapshot.forEach((child) => {
      var book = child.val();
      if (book.Code == newBook.Code) {
        var key = child.key;
        dbRef.child("books").child(key).set(newBook);
      }
    });
  });
}
function delete(code) {
  dbRef.child("books").once("value", (snapshot) => {
    snapshot.forEach((child) => {
      var book = child.val();
      if (book.Code == code) {
        var key = child.key;
        dbRef.child("books").child(key).remove();
      }
    });
  });
}
/* helper methods */
...
```

MobileClient project

- Create React Native project by Expo-cli

```
expo init MobileClient
cd MobileClient
npm install firebase --save
expo start -c
```

- Update App.js (entry-point)

```
import React, { Component } from 'react';
import Book from './components/BookComponent';

class App extends Component {
  render() {
    return (<Book />);
  }
}
export default App;
```

- Create components/BookComponent.js

```
import React, { Component } from 'react';
```

```
import { StyleSheet, View, ActivityIndicator, FlatList, TouchableOpacity, Text, TextInput, Button, Alert } from 'react-native';
import firebase from 'firebase';

class Book extends Component {
  constructor(props) {
    super(props);
    this.state = { isLoading: true, dataSource: [], txtCode: '', txtName: '', txtAuthor: '', txtPrice: '' };
  }
  render() {
    ...
    <Text>BOOK DETAILS</Text>
    <TextInput style={styles.input} underlineColorAndroid='transparent' placeholder='Code' autoCapitalize='none' value={this.state.txtCode} onChangeText={(txtCode) => this.setState({ txtCode })} />
    ...
  }
  componentDidMount() {
    this.initFirebase();
    this.getAll();
  }
  /* event-handler methods */
  pressOnRow(item) {
    ...
  }
  btnAdd = async () => {
    var newBook = { Code: this.state.txtCode, Name: this.state.txtName, Author: this.state.txtAuthor, Price: this.state.txtPrice };
    this.addNew(newBook);
  }
  btnUpdate = () => {
    ...
  }
  btnDelete = () => {
    ...
  }
  /* firebase methods */
  initFirebase() {
    const config = { databaseURL: 'https://sonkk.firebaseio.com' };
    firebase.initializeApp(config);
    this.dbRef = firebase.database().ref();
  }
  getAll() {
    this.dbRef.child("books").on("value", (snapshot) => {
      var books = [];
      snapshot.forEach((child) => {
        var book = child.val();
        books.push(book);
      });
      this.setState({ isLoading: false, dataSource: books });
```

```
48      });
49    }
50    getDetails(code) {
51      this.dbRef.child("books").once("value", (snapshot) => {
52        snapshot.forEach((child) => {
53          var book = child.val();
54          if (book.Code == code) {
55            this.setState({ txtCode: book.Code.toString(), txtName: book.Name,
                 txtAuthor: book.Author, txtPrice: book.Price.toString() });
56          }
57        });
58      });
59    }
60    addNew(newBook) {
61      //this.dbRef.child("books").push(newBook); // auto-generated key
62      this.dbRef.child("books/B" + newBook.Code).set(newBook); // custom key
63    }
64    update(newBook) {
65      this.dbRef.child("books").once("value", (snapshot) => {
66        snapshot.forEach((child) => {
67          var book = child.val();
68          if (book.Code == newBook.Code) {
69            var key = child.key;
70            this.dbRef.child("books").child(key).set(newBook);
71          }
72        });
73      });
74    }
75    delete(code) {
76      this.dbRef.child("books").once("value", (snapshot) => {
77        snapshot.forEach((child) => {
78          var book = child.val();
79          if (book.Code == code) {
80            var key = child.key;
81            this.dbRef.child("books").child(key).remove();
82          }
83        });
84      });
85    }
86  }
87  export default Book;
88
89  const styles = StyleSheet.create({
90    ...
91  });
```

2.3.5 Microservice

- Create 2 new projects:

- Service:

  - Project type: ASP.NET Web Application (.NET Framework)
  - Project name: BookSite
  - Solution name: Microservice
  - Empty project » Check on Web API

- Winform Client:
  - Project type: Windows Forms App (.NET Framework)
  - Project name: BookClient
  - Solution name: Microservice

BookClient project

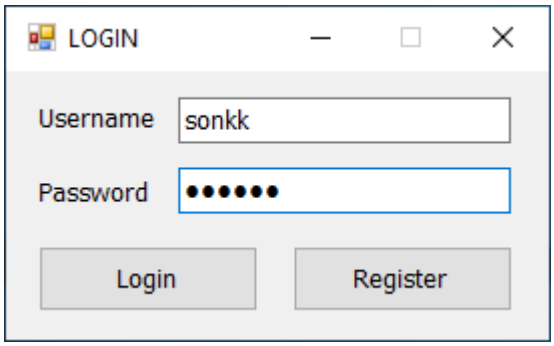
- Install Newtonsoft.Json (James Newton-King) & FireSharp (ziyasal) libraries
- Update Program.cs (entry-point)

```
...
static class Program {
  [STAThread]
  static void Main() {
    ...
    Application.Run(new LoginForm());
  }
}
```

- Create Account.cs (DTO class)

```
...
class Account {
  public String Username { get; set; }
  public String Password { get; set; }
}
```

- Create LoginForm.cs (presentation class)




```
...
public partial class LoginForm : Form {
  ...
  private void btnRegister_Click(object sender, EventArgs e) {
```

```
    Account newAccount = new Account() {
      Username = txtUsername.Text.Trim(),
      Password = txtPassword.Text.Trim()
    };
    bool result = new AccountBUS().AddNew(newAccount);
    if (result) {
      MessageBox.Show("OK BABY!");
    } else {
      MessageBox.Show("SORRY BABY!");
    }
  }
  private void btnLogin_Click(object sender, EventArgs e) {
    Account account = new Account() {
      Username = txtUsername.Text.Trim(),
      Password = txtPassword.Text.Trim()
    };
    bool result = new AccountBUS().CheckAccount(account);
    if (result) {
      new BookForm().Show();
      this.Hide();
    } else {
      MessageBox.Show("SORRY BABY!");
    }
  }
}
```

- Create AccountBUS.cs (business-logic class)

```
...
using FireSharp;
using FireSharp.Config;
using FireSharp.Interfaces;
using FireSharp.Response;

class AccountBUS {
  static IFirebaseConfig config = new FirebaseConfig { BasePath = "https://sonkk
     .firebaseio.com" };
  static FirebaseClient client = new FirebaseClient(config);

  public bool AddNew(Account newAccount) {
    try {
      client.Push("accounts", newAccount); // auto-generated key
      return true;
    }
    catch { return false; }
  }
  public bool CheckAccount(Account account) {
    FirebaseResponse response = client.Get("accounts");
    Dictionary<String, Account> dicAccounts = response.ResultAs<Dictionary<
       String, Account>>();
    String key = dicAccounts.FirstOrDefault(x => x.Value.Username == account.
```

```
        Username && x.Value.Password == account.Password).Key;
    if (String.IsNullOrEmpty(key))
      return false;
    return true;
  }
}
```

2.4 Others architecture

2.4.1 MVC

- ASP.NET MVC 5: http://www.asp.net/mvc/tutorials/mvc-5/introduction/getting-started
- ASP.NET Core MVC: http://docs.asp.net/en/latest/tutorials/first-mvc-app

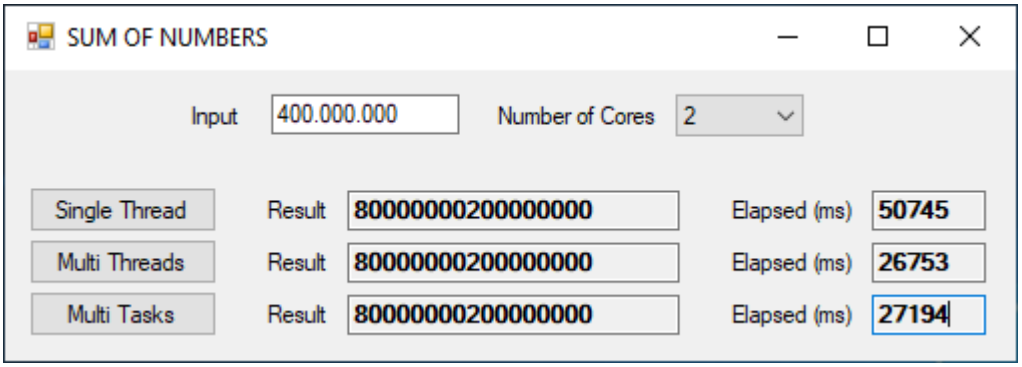
2.4.2 Parallel

- Create a new project:

- Project type: Windows Forms App (.NET Framework)
- Project name: SumOfNumbers
- Solution name: Parallel

SumOfNumbers project

- Create SumForm.cs (presentation class)




```
...
using System.Numerics; // add reference System.Numerics.dll
using System.Diagnostics;
using System.Threading;

public partial class SumForm : Form {
  public SumForm() {
    InitializeComponent();
    cmbCores.SelectedIndex = 0;
  }
  private void btnSingleThread_Click(object sender, EventArgs e) {
    btnSingleThread.Enabled = false;
    txtResult1.Text = "";
```

```
    txtElapsed1.Text = "";
    // get input
    int input = int.Parse(txtInput.Text.Replace(".", "").Trim());
    // start calculate
    Stopwatch watch = Stopwatch.StartNew();
    SumNumbers worker = new SumNumbers(1, input);
    worker.Calculate();
    BigInteger result = worker.Result;
    watch.Stop();
    long elapsed = watch.ElapsedMilliseconds;
    // display result
    txtResult1.Text = result.ToString();
    txtElapsed1.Text = elapsed.ToString();
    btnSingleThread.Enabled = true;
  }
  private void btnMultiThreads_Click(object sender, EventArgs e) {
    btnMultiThreads.Enabled = false;
    txtResult2.Text = "";
    txtElapsed2.Text = "";
    // get input
    int input = int.Parse(txtInput.Text.Replace(".", "").Trim());
    int numCores = int.Parse(cmbCores.SelectedItem.ToString().Trim());
    // assign tasks
    Thread[] threads = new Thread[numCores];
    SumNumbers[] workers = new SumNumbers[numCores];
    for (int i = 1; i <= numCores; i++) {
      int from = (i == 1) ? 1 : (input / numCores * (i - 1)) + 1;
      int to = (i == numCores) ? input : input / numCores * i;
      workers[i - 1] = new SumNumbers(from, to);
      threads[i - 1] = new Thread(new ThreadStart(workers[i - 1].Calculate));
    }
    // start calculate
    Stopwatch watch = Stopwatch.StartNew();
    foreach(Thread thread in threads) {
      thread.Start();
    }
    foreach(Thread thread in threads) {
      thread.Join();
    }
    // collect result
    BigInteger result = new BigInteger(0);
    foreach(SumNumbers worker in workers) {
      result += worker.Result;
    }
    watch.Stop();
    long elapsed = watch.ElapsedMilliseconds;
    // display result
    txtResult2.Text = result.ToString();
    txtElapsed2.Text = elapsed.ToString();
    btnMultiThreads.Enabled = true;
  }
```

```
  private void btnMultiTaks_Click(object sender, EventArgs e) {
    btnMultiTasks.Enabled = false;
    txtResult3.Text = "";
    txtElapsed3.Text = "";
    // get input
    int input = int.Parse(txtInput.Text.Replace(".", "").Trim());
    int numCores = int.Parse(cmbCores.SelectedItem.ToString().Trim());
    // assign tasks
    Task[] tasks = new Task[numCores];
    SumNumbers[] workers = new SumNumbers[numCores];
    for (int i = 1; i <= numCores; i++) {
      int from = (i == 1) ? 1 : (input / numCores * (i - 1)) + 1;
      int to = (i == numCores) ? input : input / numCores * i;
      workers[i - 1] = new SumNumbers(from, to);
      tasks[i - 1] = new Task(new Action(workers[i - 1].Calculate));
    }
    // start calculate
    Stopwatch watch = Stopwatch.StartNew();
    foreach(Task task in tasks) {
      task.Start();
    }
    Task.WaitAll(tasks);
    // collect result
    BigInteger result = new BigInteger(0);
    foreach(SumNumbers worker in workers) {
      result += worker.Result;
    }
    watch.Stop();
    long elapsed = watch.ElapsedMilliseconds;
    // display result
    txtResult3.Text = result.ToString();
    txtElapsed3.Text = elapsed.ToString();
    btnMultiTasks.Enabled = true;
  }
}
```

- Create SumNumbers.cs (business-logic class)

```
...
using System.Numerics; // add reference System.Numerics.dll

class SumNumbers {
  public int From { get; set; }
  public int To { get; set; }
  public BigInteger Result { get; set; }

  public SumNumbers(int from, int to) {
    this.From = from;
    this.To = to;
    this.Result = new BigInteger(0);
  }
```

```
14    public void Calculate() {
15      for (int i = this.From; i <= this.To; i++) {
16        this.Result += new BigInteger(i);
17      }
18    }
19  }
```