DAVID CUADRA

Technological University of Dublin  2020

HDip Final Project documentation

**Content**

**Overview…………………………………………………………………………………..3**

**Models…………………………………………………………………………………..…4**

**Database.…………………………………………………………………………………12**

**Controllers……………………………………………………………………………….35**

**Views…………………………………………………………………………………......52**

**API……………………………………………………………………………………….68**

**Unit Testing……………………………………………………………………………...75**

**Overview**

**Application that books Clients with Counsellors and generates bills for these bookings**

**Change Counselling** is a booking site where Counsellors can offer their services to Clients who are looking for counselling sessions.

The System admin can create, read, update and delete Booking data, Counsellors and Clients. There is also a Bills tabel for each Booking that can be read.

**GITHUB LINKS**

GitHub was used to manage the project and it’s code. The following links will open the code in GitHub:

* For the main MVC project: <https://github.com/DavidCuadra/ChangeCounselling>
* For the Console Client Application: <https://github.com/DavidCuadra/ChangeCounselling/blob/master/ApiClient/Program.cs>

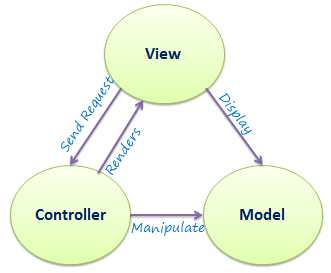
**AZURE LINKS**

Azure was used to publish the project. The following links will direct you to the app in Azure:

* Azure app link: <https://changecounsellingweb12345.azurewebsites.net/>

**The project was created with MVC 5 application**.

MVC is an application design model comprised of three interconnected parts. They include the model (data), the view (user interface), and the controller (processes that handle input).



**1.2 Models**

A model is data used by a program. This may be a database, file, or a simple object, such as an icon or a character.

This project has 4 Models: Counsellor/s, Client/s, Book/s and Bill/s. We will also have an Enum class named RateType, these are not a database table.



This program will allow the counsellors to enter the details like: name, address, phone, email and rate type. Constraints have been also added as seen below:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.ComponentModel.DataAnnotations;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Web.Mvc;

namespace ChangeCounselling.Data.Models

{

public class Counsellor

{

[Key]

[DisplayName("Counsellor ID")]

public int CounsellorID { get; set; }

[Required]

[MinLength(1)]

[MaxLength(50)]

[Display(Name ="Counsellor First Name")]

public string CouncellorFirstName { get; set; }

public IEnumerable<SelectListItem> CounsellorList { get; set; }

[Required]

[MinLength(1)]

[MaxLength(50)]

[DisplayName("Counsellor Last Name")]

public string CouncellorLastName { get; set; }

[Required]

[MinLength(1)]

[MaxLength(50)]

[Display(Name = "Counsellor Address Line 1")]

public string CouncellorAddressLine1 { get; set; }

[Required]

[MinLength(1)]

[MaxLength(50)]

[Display(Name = "Counsellor Address Line 2")]

public string CouncellorAddressLine2 { get; set; }

[Required]

[MinLength(1)]

[MaxLength(50)]

[Display(Name = "Counsellor Address Line 3")]

public string CouncellorAddressLine3 { get; set; }

[Required]

[MinLength(2)]

[MaxLength(10)]

[Display(Name = "Counsellor Phone Number")]

public string CouncellorPhone { get; set; }

[Required]

[EmailAddress]

[Display(Name = "Counsellor Email")]

public string CouncellorEmail { get; set; }

[Required]

[Display(Name = "Counsellor Rate")]

public RateType CouncellorRate { get; set; }

public virtual ICollection<Book> Books { get; set; }

}

}

This program will allow the clients to enter the details like: name, address, phone and email. Constraints have been also added as seen below:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.ComponentModel.DataAnnotations;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Web.Mvc;

namespace ChangeCounselling.Data.Models

{

public class Client

{

[Key]

[DisplayName("Client ID")]

public int ClientID { get; set; }

[Required]

[MinLength(1)]

[MaxLength(50)]

[Display(Name = "Client First Name")]

public string ClientFirstName { get; set; }

public IEnumerable<SelectListItem> ClientList { get; set; }

[Required]

[MinLength(1)]

[MaxLength(50)]

[Display(Name = "Client Last Name")]

public string ClientLastName { get; set; }

[Required]

[MinLength(1)]

[MaxLength(50)]

[Display(Name = "Client Address Line 1")]

public string ClientAddressLine1 { get; set; }

[Required]

[MinLength(1)]

[MaxLength(50)]

[Display(Name = "Client Address Line 2")]

public string ClientAddressLine2 { get; set; }

[Required]

[MinLength(1)]

[MaxLength(50)]

[Display(Name = "Client Address Line 3")]

public string ClientAddressLine3 { get; set; }

[Required]

[MinLength(2)]

[MaxLength(10)]

[Display(Name = "Client Phone Number")]

public string ClientPhone { get; set; }

[Required]

[EmailAddress]

[Display(Name = "Client Email")]

public string ClientEmail { get; set; }

public virtual ICollection<Book> Books { get; set; }

}

}

This program will allow the bookings to be made by entering details like: session date, counsellor and client. Constraints have been also added as seen below:

using ChangeCounselling.Data.CustomValidation;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Web.Mvc;

namespace ChangeCounselling.Data.Models

{

public class Book

{

[Key]

[DisplayName("Book ID")]

public int BookID { get; set; }

[Required]

[DisplayName("Session Date")]

[DisplayFormat(DataFormatString = "{0:dd/MM/yyyy hh:mm }", ApplyFormatInEditMode = true)]

[CustomDateValidation(ErrorMessage = "Date not Valid")]

public DateTime DateTime { get; set; }

[Required]

[Range(1, int.MaxValue,ErrorMessage ="You must select a Client")]

public int ClientID { get; set; }

[Required]

[Range(1, int.MaxValue, ErrorMessage = "You must select a Counsellor")]

public int CounsellorID { get; set; }

public virtual Client Client { get; set; }

public virtual Counsellor Counsellor { get; set; }

public virtual ICollection <Bill> Bills { get; set; }

}

}

Custom Validation was used for the DateTime variable in Book as to not be able to select a previous date or time from Date.Now:

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ChangeCounselling.Data.CustomValidation

{

public class CustomDateValidation : ValidationAttribute

{

public override bool IsValid(object value)

{

DateTime dateTime = Convert.ToDateTime(value);

var result = DateTime.Compare(DateTime.Now, dateTime);

if (result < 0 || result == 0)

{

return true;

}

return false;

}

}

}

This program will allow the bills to be created from a booking showing: bill id, |session date, counsellor and client information. Constraints have been also added as seen below:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ChangeCounselling.Data.Models

{

public class Bill

{

[Key]

[DisplayName("Bill ID")]

public int BillID { get; set; }

[Required]

//Data from book

[DisplayName("Bill Created Date Time")]

public DateTime DateTime { get; set; }

[Required]

[DisplayName("Client First Name")]

public string ClientFirstName { get; set; }

[Required]

[DisplayName("Client Last Name")]

public string ClientLastName { get; set; }

[Required]

[DisplayName("Counsellor First Name")]

public string ClientEmail { get; set; }

[Required]

[DisplayName("Counsellor First Name")]

public string CounsellorFirstName { get; set; }

[Required]

[DisplayName("Counsellor Last Name")]

public string CounsellorLastName { get; set; }

[Required]

[DisplayName("Counsellor Email")]

public string CounsellorEmail { get; set; }

[Required]

public int BookID { get; set; }

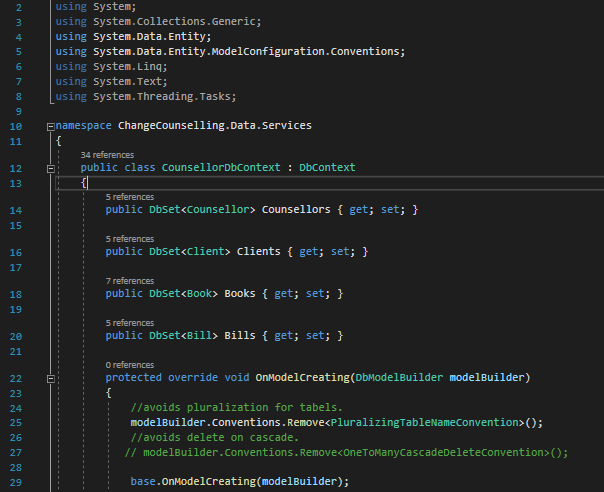
public virtual Book Book { get; set; }

}

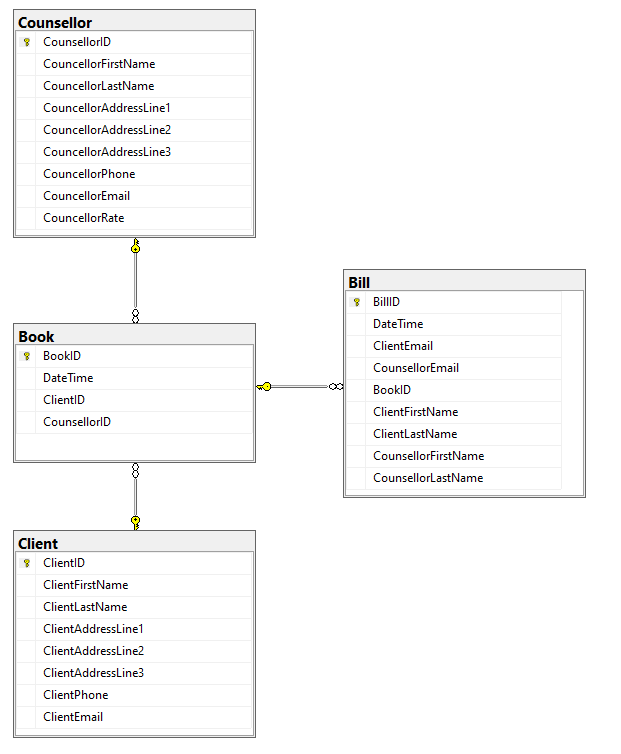
}

**1.3 Database**

I used Microsoft SQL Server Management Studio to store the database. Code First approche was used to create the database through Entity Frameworks of Microsoft using a DbContext as shown below:



Keys and foreing keys where placed as per diagram below as well as relations between tables.



A Migrations folder was included to enable future changes in the database through a Code First approach:

namespace ChangeCounselling.Data.Migrations

{

using System;

using System.Data.Entity;

using System.Data.Entity.Migrations;

using System.Linq;

internal sealed class Configuration : DbMigrationsConfiguration<ChangeCounselling.Data.Services.CounsellorDbContext>

{

public Configuration()

{

AutomaticMigrationsEnabled = true;

AutomaticMigrationDataLossAllowed = true;

}

protected override void Seed(ChangeCounselling.Data.Services.CounsellorDbContext context)

{

// This method will be called after migrating to the latest version.

// You can use the DbSet<T>.AddOrUpdate() helper extension method

// to avoid creating duplicate seed data.

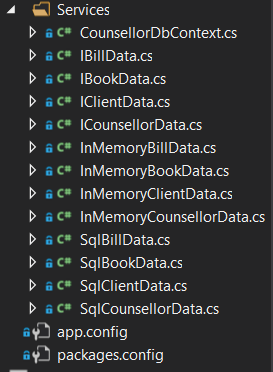
}

}

}

**1.4 Services**

I also created a Services folder that stores not only the DbContext but also interfaces that are linked to classes named "Sql...Data" for each tabel. For initial testing purposes "InMemory" classes were also intanciated for each tabel.



**1.4.1 Interfaces**

ICounsellorData

using ChangeCounselling.Data.Models;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ChangeCounselling.Data.Services

{

public interface ICounsellorData

{

IEnumerable<Counsellor> GetAll();

Counsellor Get(int id);

void Add(Counsellor counsellor);

void Update(Counsellor counsellor);

void Delete(int id);

}

}

IClientData

using ChangeCounselling.Data.Models;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Web.Mvc;

namespace ChangeCounselling.Data.Services

{

public interface IClientData

{

IEnumerable<Client> GetAll();

Client Get(int id);

void Add(Client client);

void Update(Client client);

void Delete(int id);

}

}

IBookData

using ChangeCounselling.Data.Models;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ChangeCounselling.Data.Services

{

public interface IBookData

{

IEnumerable<Book> GetAll();

IEnumerable<Book> GetAllWithClientCounsellor();

Book Get(int id);

void Add(Book book);

void Update(Book book);

void Delete(int id);

}

}

IBillData

using ChangeCounselling.Data.Models;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ChangeCounselling.Data.Services

{

public interface IBillData

{

IEnumerable<Bill> GetAll();

IEnumerable<Book> GetAllWithClientCounsellor();

Bill Get(int id);

void Add(Bill bill);

void Update(Bill bill);

void Delete(int id);

}

}

**1.4.2 Sql Data Classes**

SqlCounsellorData

using System;

using System.Collections.Generic;

using System.Data.Entity;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using ChangeCounselling.Data.Models;

namespace ChangeCounselling.Data.Services

{

public class SqlCounsellorData : ICounsellorData

{

private readonly CounsellorDbContext db;

public SqlCounsellorData(CounsellorDbContext db)

{

this.db = db;

}

public void Add(Counsellor counsellor)

{

db.Counsellors.Add(counsellor);

db.SaveChanges();

}

public void Delete(int id)

{

var counsellor = db.Counsellors.Find(id);

db.Counsellors.Remove(counsellor);

db.SaveChanges();

}

public Counsellor Get(int id)

{

return db.Counsellors.FirstOrDefault(c => c.CounsellorID == id);

}

public IEnumerable<Counsellor> GetAll()

{

return from c in db.Counsellors

orderby c.CouncellorFirstName

select c;

}

public void Update(Counsellor counsellor)

{

var entry = db.Entry(counsellor);

entry.State = EntityState.Modified;

db.SaveChanges();

}

}

}

SqlClientData

using ChangeCounselling.Data.Models;

using System;

using System.Collections.Generic;

using System.Data.Entity;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Web.Mvc;

namespace ChangeCounselling.Data.Services

{

public class SqlClientData:IClientData

{

private readonly CounsellorDbContext db;

public SqlClientData(CounsellorDbContext db)

{

this.db = db;

}

public void Add(Client client)

{

db.Clients.Add(client);

db.SaveChanges();

}

public void Delete(int id)

{

var client = db.Clients.Find(id);

db.Clients.Remove(client);

db.SaveChanges();

}

public Client Get(int id)

{

return db.Clients.FirstOrDefault(c => c.ClientID == id);

}

public IEnumerable<Client> GetAll()

{

return from c in db.Clients

orderby c.ClientFirstName

select c;

}

public void Update(Client client)

{

var entry = db.Entry(client);

entry.State = EntityState.Modified;

db.SaveChanges();

}

}

}

SqlBookData

using System;

using System.Collections.Generic;

using System.Data.Entity;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using ChangeCounselling.Data.Models;

namespace ChangeCounselling.Data.Services

{

public class SqlBookData : IBookData

{

private readonly CounsellorDbContext db;

public SqlBookData(CounsellorDbContext db)

{

this.db = db;

}

public void Add(Book book)

{

db.Books.Add(book);

db.SaveChanges();

}

public void Delete(int id)

{

var book = db.Books.Find(id);

db.Books.Remove(book);

db.SaveChanges();

}

public Book Get(int id)

{

return db.Books.FirstOrDefault(b => b.BookID == id);

}

public IEnumerable<Book> GetAll()

{

return from b in db.Books

orderby b.BookID

select b;

}

public IEnumerable<Book> GetAllWithClientCounsellor()

{

var result = db.Books

.Include(p => p.Client)

.Include(p => p.Counsellor)

.ToList();

return result;

}

public void Update(Book book)

{

var entry = db.Entry(book);

entry.State = EntityState.Modified;

db.SaveChanges();

}

}

}

SqlBillData

using System;

using System.Collections.Generic;

using System.Data.Entity;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using ChangeCounselling.Data.Models;

namespace ChangeCounselling.Data.Services

{

public class SqlBillData: IBillData

{

private readonly CounsellorDbContext db;

public SqlBillData(CounsellorDbContext db)

{

this.db = db;

}

public void Add(Bill bill)

{

db.Bills.Add(bill);

db.SaveChanges();

}

public void Delete(int id)

{

var bill = db.Bills.Find(id);

db.Bills.Remove(bill);

db.SaveChanges();

}

public Bill Get(int id)

{

return db.Bills.FirstOrDefault(b => b.BillID == id);

}

public IEnumerable<Bill> GetAll()

{

return db.Bills.ToList();

}

public IEnumerable<Book> GetAllWithClientCounsellor()

{

var result = db.Books

.Include(p => p.Client)

.Include(p => p.Counsellor)

.ToList();

return result;

}

public void Update(Bill bill)

{

var entry = db.Entry(bill);

entry.State = EntityState.Modified;

db.SaveChanges();

}

}

}

**1.4.3 Autofac**

The interfaces and the correspondiong "Sql...Data" classes where each registered in a seperate container using Autofact scafolding and instanced per request in the ContainerConfig App\_Start file as shown below:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Http;

using System.Web.Mvc;

using Autofac;

using Autofac.Integration.Mvc;

using Autofac.Integration.WebApi;

using ChangeCounselling.Data.Services;

namespace ChangeCounselling.Web

{

public class ContainerConfig

{

internal static void RegisterContainer(HttpConfiguration httpConfiguration)

{

var builder = new ContainerBuilder();

builder.RegisterControllers(typeof(MvcApplication).Assembly);

builder.RegisterApiControllers(typeof(MvcApplication).Assembly);

builder.RegisterType<SqlCounsellorData>()

.As<ICounsellorData>()

.InstancePerRequest();

builder.RegisterType<SqlClientData>()

.As<IClientData>()

.InstancePerRequest();

builder.RegisterType<SqlBookData>()

.As<IBookData>()

.InstancePerRequest();

builder.RegisterType<SqlBillData>()

.As<IBillData>()

.InstancePerRequest();

builder.RegisterType<CounsellorDbContext>().InstancePerRequest();

var container = builder.Build();

DependencyResolver.SetResolver(new AutofacDependencyResolver(container));

httpConfiguration.DependencyResolver = new AutofacWebApiDependencyResolver(container);

}

}

}

**1.4.4 InMemory classes**

As mentioned above these classes where used for initial testing:

InMemoryCounsellorData

using ChangeCounselling.Data.Models;

using System;

using System.Collections.Generic;

using System.Linq;

namespace ChangeCounselling.Data.Services

{

public class InMemoryCounsellorData : ICounsellorData

{

List<Counsellor> counsellors;

public InMemoryCounsellorData()

{

//counsellors = new List<Counsellor>()

// {

// new Counsellor {CounsellorID = 1, CouncellorFirstName = "David", CouncellorLastName = "Cuadra", CouncellorAddressLine1 = "6 Anner Road", CouncellorAddressLine2 = "Kilmainham", CouncellorAddressLine3 = "Dublin", CouncellorEmail = "davidcuadra@yahoo.com", CouncellorPhone = "086777234", CouncellorRate = RateType.Plus},

// new Counsellor {CounsellorID = 2, CouncellorFirstName = "Catherine", CouncellorLastName = "Mcfadden", CouncellorAddressLine1 = "12 Nice Road", CouncellorAddressLine2 = "Graystones", CouncellorAddressLine3 = "Wiclow", CouncellorEmail = "Catherine@yahoo.com", CouncellorPhone = "677000333", CouncellorRate = RateType.Economy },

// new Counsellor {CounsellorID = 3, CouncellorFirstName = "Carol", CouncellorLastName = "Cromer", CouncellorAddressLine1 = "8 Where Road", CouncellorAddressLine2 = "Lixslip", CouncellorAddressLine3 = "Kilder", CouncellorEmail = "Carol@yahoo.com", CouncellorPhone = "55555555", CouncellorRate = RateType.Regular}

// };

}

public void Add(Counsellor counsellor)

{

counsellors.Add(counsellor);

counsellor.CounsellorID = counsellors.Max(c => c.CounsellorID) + 1;

}

public void Update(Counsellor counsellor)

{

var existing = Get(counsellor.CounsellorID);

if(existing != null)

{

existing.CounsellorID = counsellor.CounsellorID;

existing.CouncellorFirstName = counsellor.CouncellorFirstName;

existing.CouncellorLastName = counsellor.CouncellorLastName;

existing.CouncellorAddressLine1 = counsellor.CouncellorAddressLine1;

existing.CouncellorAddressLine2 = counsellor.CouncellorAddressLine2;

existing.CouncellorAddressLine3 = counsellor.CouncellorAddressLine3;

existing.CouncellorEmail = counsellor.CouncellorEmail;

existing.CouncellorPhone = counsellor.CouncellorPhone;

existing.CouncellorRate = counsellor.CouncellorRate;

}

}

public Counsellor Get(int id)

{

return counsellors.FirstOrDefault(c => c.CounsellorID == id);

}

public IEnumerable<Counsellor> GetAll()

{

return counsellors.OrderBy(r => r.CouncellorFirstName);

}

public void Delete(int id)

{

var counsellor = Get(id);

if(counsellor != null )

{

counsellors.Remove(counsellor);

}

}

}

}

InMemoryClientData

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Web.Mvc;

using ChangeCounselling.Data.Models;

namespace ChangeCounselling.Data.Services

{

public class InMemoryClientData : IClientData

{

List<Client> clients;

public string Clients { get => throw new NotImplementedException(); set => throw new NotImplementedException(); }

public InMemoryClientData()

{

//clients = new List<Client>()

// {

// new Client {ClientID = 1, ClientFirstName = "Juan", ClientLastName = "Oldani", ClientAddressLine1 = "26 Basil Road", ClientAddressLine2 = "Swiss", ClientAddressLine3 = "Basil", ClientEmail = "juan@oldani.com", ClientPhone = "55555555"},

// new Client {ClientID = 2, ClientFirstName = "Lily", ClientLastName = "Cuadra", ClientAddressLine1 = "76 Venice Road", ClientAddressLine2 = "Cosa Nostra", ClientAddressLine3 = "Italia", ClientEmail = "lily@cuadra.com", ClientPhone = "333333333"},

// new Client {ClientID = 3, ClientFirstName = "Pipa", ClientLastName = "Cuadra", ClientAddressLine1 = "6 Anner Road", ClientAddressLine2 = "Kilmainham", ClientAddressLine3 = "Dublin", ClientEmail = "pipa@cuadra.com", ClientPhone = "999999999"},

// };

}

public void Add(Client client)

{

clients.Add(client);

client.ClientID = clients.Max(c => c.ClientID) + 1;

}

public void Update(Client client)

{

var existing = Get(client.ClientID);

if (existing != null)

{

existing.ClientID = client.ClientID;

existing.ClientFirstName = client.ClientFirstName;

existing.ClientLastName = client.ClientLastName;

existing.ClientAddressLine1 = client.ClientAddressLine1;

existing.ClientAddressLine2 = client.ClientAddressLine2;

existing.ClientAddressLine3 = client.ClientAddressLine3;

existing.ClientEmail = client.ClientEmail;

existing.ClientPhone = client.ClientPhone;

}

}

public Client Get(int id)

{

return clients.FirstOrDefault(c => c.ClientID == id);

}

public IEnumerable<Client> GetAll()

{

return clients.OrderBy(c => c.ClientFirstName);

}

public void Delete(int id)

{

var client = Get(id);

if (client != null)

{

clients.Remove(client);

}

}

public IEnumerable<SelectListItem> ClientList()

{

throw new NotImplementedException();

}

}

}

InMemoryBookData

using ChangeCounselling.Data.Models;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ChangeCounselling.Data.Services

{

public class InMemoryBookData : IBookData

{

List<Book> books;

public InMemoryBookData()

{

//books = new List<Book>()

// {

// new Book {BookID = 1, ClientID = 1, CounsellorID = 1, DateTime = 1/1/2001 12:00:00 AM , Booked = true},

// new Book {BookID = 2, ClientID = 2, CounsellorID = 2, DateTime = 15/2/2011 20:00:00 AM , Booked = false},

// new Book {BookID = 3, ClientID = 3, CounsellorID = 3, DateTime = 20/3/2031 18:00:00 AM , Booked = true},

// };

}

public void Add(Book book)

{

books.Add(book);

book.BookID = books.Max(c => c.BookID) + 1;

}

public void Update(Book book)

{

var existing = Get(book.BookID);

if (existing != null)

{

existing.BookID = book.BookID;

existing.ClientID = book.ClientID;

existing.CounsellorID = book.CounsellorID;

existing.DateTime = book.DateTime;

}

}

public Book Get(int id)

{

return books.FirstOrDefault(c => c.BookID == id);

}

public IEnumerable<Book> GetAll()

{

return books.OrderBy(c => c.BookID);

}

public void Delete(int id)

{

var book = Get(id);

if (book != null)

{

books.Remove(book);

}

}

public IEnumerable<Book> GetAllWithClientCounsellor()

{

throw new NotImplementedException();

}

}

}

InMemoryBillData

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using ChangeCounselling.Data.Models;

namespace ChangeCounselling.Data.Services

{

public class InMemoryBillData : IBillData

{

List<Bill> bills;

public InMemoryBillData()

{

//bills = new List<Bill>()

//{

// new Bill {BillID = 1, DateTime = 1/1/2001 12:00:00 AM, ClientEmail = "Cuadra@hotmail.com", CounsellorEmail = "Mcfadden@yahoo.com", BookID = 1},

// new Bill {BillID = 2, DateTime = 1/1/2001 12:00:00 AM, ClientEmail = "Mcfadden@yahoo.com", CounsellorEmail = "Cuadra@hotmail.com", BookID = 2},

// new Bill {BillID = 3, DateTime = 1/1/2001 12:00:00 AM, ClientEmail = "Cromer@gmail.com", CounsellorEmail = "Cuadra@hotmail.com", BookID = 3}

//};

}

public void Add(Bill bill)

{

bills.Add(bill);

bill.BillID = bills.Max(c => c.BillID) + 1;

}

public void Update(Bill bill)

{

var existing = Get(bill.BillID);

if (existing != null)

{

existing.BillID = bill.BillID;

existing.DateTime = bill.DateTime;

existing.ClientEmail = bill.ClientEmail;

existing.CounsellorEmail = bill.CounsellorEmail;

existing.BookID = bill.BookID;

}

}

public Bill Get(int id)

{

return bills.FirstOrDefault(b => b.BillID == id);

}

public IEnumerable<Bill> GetAll()

{

return bills.OrderBy(b => b.BillID);

}

public void Delete(int id)

{

var bill = Get(id);

if (bill != null)

{

bills.Remove(bill);

}

}

IEnumerable<Bill> IBillData.GetAll()

{

throw new NotImplementedException();

}

public IEnumerable<Book> GetAllWithClientCounsellor()

{

throw new NotImplementedException();

}

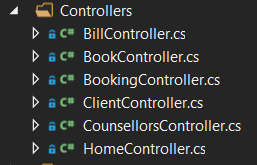
}

}

**1.5 Controllers**

A controller updates both models and views. It accepts inputs and performs the corresponding update. For example, a controller can update a model by changing the attributes of a character in a program. It may modify the view by displaying the updated character in the system.

In this application we used one Controller per Model and also have a Home Controller as shown below:



For the Home controller we used the following code:

using ChangeCounselling.Data.Services;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

namespace ChangeCounselling.Web.Controllers

{

public class HomeController : Controller

{

ICounsellorData db;

public HomeController(ICounsellorData db)

{

this.db = db;

}

public ActionResult Index()

{

var model = db.GetAll();

return View(model);

}

public ActionResult About()

{

ViewBag.Message = "Welcome to Change Counselling!";

return View();

}

public ActionResult Contact()

{

ViewBag.Message = "Your contact page.";

return View();

}

}

}

For the Counsellors controller we used as follows:

using ChangeCounselling.Data.Models;

using ChangeCounselling.Data.Services;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

using System.Web.Security;

namespace ChangeCounselling.Web.Controllers

{

public class CounsellorsController : Controller

{

private readonly ICounsellorData db;

public CounsellorsController(ICounsellorData db)

{

this.db = db;

}

// GET: Counsellors

[HttpGet]

public ActionResult Index()

{

var model = db.GetAll();

return View(model);

}

[HttpGet]

public ActionResult Details(int id)

{

var model = db.Get(id);

if(model == null)

{

return View("NotFound");

}

return View(model);

}

public ActionResult Create ()

{

return View();

}

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Create(Counsellor counsellor)

{

if (ModelState.IsValid)

{

db.Add(counsellor);

return RedirectToAction("Details", new { id = counsellor.CounsellorID });

}

return View();

}

[HttpGet]

public ActionResult Edit(int id)

{

var model = db.Get(id);

if(model == null)

{

{

return HttpNotFound();

}

}

return View(model);

}

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Edit(Counsellor counsellor)

{

if (ModelState.IsValid)

{

db.Update(counsellor);

TempData["Message"] = "You have succesfully saved the Counsellor!";

return RedirectToAction("Details", new { id = counsellor.CounsellorID });

}

return View(counsellor);

}

[HttpGet]

public ActionResult Delete(int id)

{

var model = db.Get(id);

if (model == null)

{

{

return HttpNotFound();

}

}

return View(model);

}

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Delete(int id, FormCollection form)

{

db.Delete(id);

return RedirectToAction("Index");

}

}

}

See the Client Controller code below:

using ChangeCounselling.Data.Models;

using ChangeCounselling.Data.Services;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

namespace ChangeCounselling.Web.Controllers

{

public class ClientController : Controller

{

private readonly IClientData db;

public ClientController(IClientData db)

{

this.db = db;

}

//Get Clients

[HttpGet]

public ActionResult Index()

{

var model = db.GetAll();

return View(model);

}

[HttpGet]

public ActionResult Details(int id)

{

var model = db.Get(id);

if(model == null)

{

return HttpNotFound();

}

return View(model);

}

public ActionResult Create()

{

return View();

}

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Create(Client client)

{

if(ModelState.IsValid)

{

db.Add(client);

return RedirectToAction("Details", new { id = client.ClientID });

}

return View(client);

}

[HttpGet]

public ActionResult Edit(int id)

{

var model = db.Get(id);

if(model == null)

{

{

return HttpNotFound();

}

}

return View(model);

}

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Edit (Client client)

{

if (ModelState.IsValid)

{

db.Update(client);

TempData["Message"] = "You have succesfully saved the Client!";

return RedirectToAction("Details", new { id = client.ClientID });

}

return View(client);

}

[HttpGet]

public ActionResult Delete (int id)

{

var model = db.Get(id);

if (model == null)

{

{

return HttpNotFound();

}

}

return View(model);

}

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Delete (int id, FormCollection form)

{

db.Delete(id);

return RedirectToAction ("Index");

}

}

}

See the Book Controller code below:

using ChangeCounselling.Data.Models;

using ChangeCounselling.Data.Services;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

namespace ChangeCounselling.Web.Controllers

{

public class BookController : Controller

{

private readonly IBookData db;

private readonly IClientData dbClient;

private readonly ICounsellorData dbCounsellor;

private readonly IBillData dbBill;

public BookController(IBookData db, IClientData dbClient, ICounsellorData dbCounsellor, IBillData dbBill)

{

this.db = db;

this.dbClient = dbClient;

this.dbCounsellor = dbCounsellor;

this.dbBill = dbBill;

}

// GET: Book

[HttpGet]

public ActionResult Index()

{

var model = db.GetAll();

return View(model);

}

// GET: Book/Details/5

public ActionResult Details(int id)

{

var model = db.Get(id);

if (model == null)

{

return View("NotFound");

}

return View(model);

}

// GET: Book/Create

public ActionResult Create()

{

var db = new SqlClientData(new CounsellorDbContext());

var result = db.GetAll().ToList();

var db1 = new SqlCounsellorData(new CounsellorDbContext());

var result1 = db1.GetAll().ToList();

ViewBag.data = result;

ViewBag.data1 = result1;

return View();

}

public ActionResult CreateBill(int id)

{

var book = this.db.Get(id);

Bill bill = new Bill()

{

DateTime = DateTime.Now,

BookID = id,

ClientFirstName = dbClient.Get(book.ClientID).ClientFirstName,

ClientLastName = dbClient.Get(book.ClientID).ClientLastName,

ClientEmail = dbClient.Get(book.ClientID).ClientEmail,

CounsellorFirstName = dbCounsellor.Get(book.CounsellorID).CouncellorFirstName,

CounsellorLastName = dbCounsellor.Get(book.CounsellorID).CouncellorLastName,

CounsellorEmail = dbCounsellor.Get(book.CounsellorID).CouncellorEmail

};

dbBill.Add(bill);

return RedirectToAction("Index", "Bill");

}

// POST: Book/Create

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Create(Book book)

{

if (ModelState.IsValid)

{

db.Add(book);

return RedirectToAction("Details", new { id = book.BookID });

}

var dbC = new SqlClientData(new CounsellorDbContext());

var result = dbC.GetAll().ToList();

var db1 = new SqlCounsellorData(new CounsellorDbContext());

var result1 = db1.GetAll().ToList();

ViewBag.data = result;

ViewBag.data1 = result1;

return View();

}

// GET: Book/Edit/5

[HttpGet]

public ActionResult Edit(int id)

{

var db = new SqlClientData(new CounsellorDbContext());

var result = db.GetAll().ToList();

var db1 = new SqlCounsellorData(new CounsellorDbContext());

var result1 = db1.GetAll().ToList();

ViewBag.data = result;

ViewBag.data1 = result1;

var model =this.db.Get(id);

if(model == null)

{

return View("NotFound");

}

return View(model);

}

// POST: Book/Edit/5

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Edit(Book book)

{

if (ModelState.IsValid)

{

db.Update(book);

TempData["Message"] = "You have succesfully updated your booking!";

return RedirectToAction("Details", new { id = book.BookID});

}

var dbC = new SqlClientData(new CounsellorDbContext());

var result = dbC.GetAll().ToList();

var db1 = new SqlCounsellorData(new CounsellorDbContext());

var result1 = db1.GetAll().ToList();

ViewBag.data = result;

ViewBag.data1 = result1;

return View(book);

}

// GET: Book/Delete/5

[HttpGet]

public ActionResult Delete(int id)

{

var model = db.Get(id);

if (model == null)

{

return View("NotFound");

}

return View(model);

}

// POST: Book/Delete/5

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Delete(int id, Book book)

{

db.Delete(id);

return RedirectToAction("Index");

}

}

}

See the Bill Controller code below:

using ChangeCounselling.Data.Models;

using ChangeCounselling.Data.Services;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

namespace ChangeCounselling.Web.Controllers

{

public class BillController : Controller

{

private readonly IBillData db;

private readonly IBookData dbBook;

private readonly IClientData dbClient;

private readonly ICounsellorData dbCounsellor;

public BillController(IBillData db, IClientData dbClient, ICounsellorData dbCounsellor,IBookData dbBook)

{

this.db = db;

this.dbClient = dbClient;

this.dbCounsellor = dbCounsellor;

this.dbBook = dbBook;

}

// GET: Bill

public ActionResult Index()

{

var model = db.GetAll();

return View(model);

}

// GET: Bill/Details/5

[HttpGet]

public ActionResult Details(int id)

{

var model = db.Get(id);

if (model == null)

{

return View("NotFound");

}

return View(model);

}

// GET: Bill/Create

public ActionResult Create()

{

return View();

}

// POST: Bill/Create

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Create(Bill bill)

{

if (ModelState.IsValid)

{

db.Add(bill);

return RedirectToAction("Details", new { id = bill.BillID });

}

return View();

}

// GET: Bill/Edit/5

[HttpGet]

public ActionResult Edit(int id)

{

var dbBook = new SqlBookData(new CounsellorDbContext());

var result = dbBook.GetAll().ToList();

ViewBag.data = result;

var model = db.Get(id);

if (model == null)

{

{

return HttpNotFound();

}

}

return View(model);

}

// POST: Bill/Edit/5

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Edit(int id, Bill bill)

{

var book = this.dbBook.Get(bill.BookID);

bill.DateTime = DateTime.Now;

bill.ClientFirstName = dbClient.Get(book.ClientID).ClientFirstName;

bill.ClientLastName = dbClient.Get(book.ClientID).ClientLastName;

bill.ClientEmail = dbClient.Get(book.ClientID).ClientEmail;

bill.CounsellorFirstName = dbCounsellor.Get(book.CounsellorID).CouncellorFirstName;

bill.CounsellorLastName = dbCounsellor.Get(book.CounsellorID).CouncellorLastName;

bill.CounsellorEmail = dbCounsellor.Get(book.CounsellorID).CouncellorEmail;

db.Update(bill);

TempData["Message"] = "You have succesfully saved the Bill!";

return RedirectToAction("Details", new { id = bill.BillID });

}

// GET: Bill/Delete/5

[HttpGet]

public ActionResult Delete(int id)

{

var model = db.Get(id);

if (model == null)

{

{

return HttpNotFound();

}

}

return View(model);

}

// POST: Bill/Delete/5

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Delete(int id, FormCollection form)

{

db.Delete(id);

return RedirectToAction("Index");

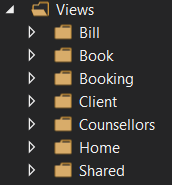
}

}

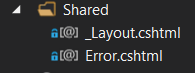
}

**1.6 Views**

Views display objects within an application. For example displaying a window, buttons, backgorund, text within a window, etc. It includes anything that the user can see. The views we used are as follows:



For the Shared folder I used the following code to present our data:



For \_Layout code is as follows:

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>@ViewBag.Title - Change Counselling Home</title>

@Styles.Render("~/Content/css")

@Scripts.Render("~/bundles/modernizr")

@Scripts.Render("~/bundles/jquery")

@Scripts.Render("~/bundles/bootstrap")

</head>

<body>

<style>

.jumbotron h1 {padding-left: 20px; display: inline; color: darkgreen; font-family: 'Times New Roman', Times, serif; text-shadow: 3px 2px lightgrey; font-size: 75px;}

.jumbotron h2 {padding-left: 20px; display: inline; color: forestgreen; font-family: Arial, Helvetica, sans-serif; font-style: italic; font-size: 30px; }

.jumbotron h3 {padding-left: 20px;}

.jumbotron li {display: inline; text-align:center; float: left; padding-left: 0px; margin: 35px; font-size: 20px; font-weight: bold; font-family: Calibri, Times, serif;}

.myactionlink {color: darkgreen;}

.jumbotron {background-color: ghostwhite;}

</style>

<div class="jumbotron">

<h1><a href="@Url.Action("Index","Home")"> <strong>CHANGE COUNSELLING</strong></a> </h1>

<div>

<h2>Mental Wellbeing through change</h2>

</div>

<div class="navbar-header">

<button type="button" class="navbar-toggle" data-toggle="collapse" data-target=".navbar-collapse">

<span class="icon-bar"></span>

<span class="icon-bar"></span>

<span class="icon-bar"></span>

</button>

</div>

<div class="container">

<div class="navbar-collapse collapse">

<ul class="nav navbar-nav">

<li>@Html.ActionLink("Home", "Index", "Home")</li>

<li>@Html.ActionLink("About", "About", "Home")</li>

<li>@Html.ActionLink("Counsellors", "Index", "Counsellors")</li>

<li>@Html.ActionLink("Clients", "Index", "Client")</li>

<li>@Html.ActionLink("Book", "Index", "Book")</li>

<li>@Html.ActionLink("Bills", "Index", "Bill")</li>

<li>@Html.ActionLink("Contact", "Contact", "Home")</li>

</ul>

</div>

</div>

</div>

<div class="container body-content">

@if(TempData["Message"] != null)

{

<div class="alert alert-info">

@TempData["Message"]

</div>

}

@RenderBody()

<hr />

<footer>

<p>&copy; @DateTime.Now.Year - Change Counselling</p>

</footer>

</div>

@RenderSection("scripts", required: false)

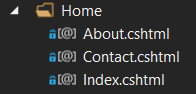
</body>

</html>

For Error View:



For the Home folder I used the following viewsto present our data:



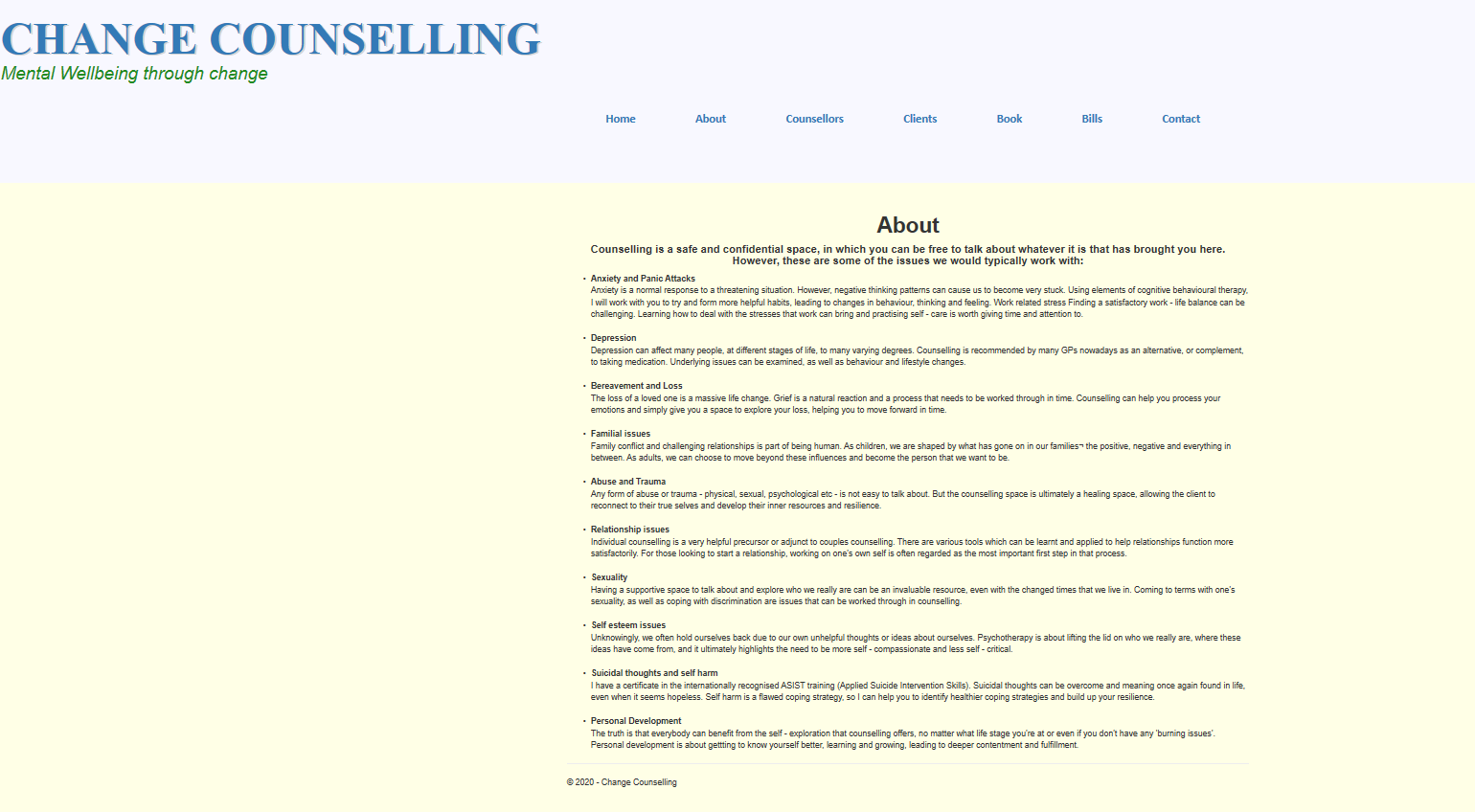
For the Index view:



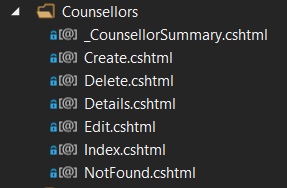
The Contact page View is:



The view for the About page is:



For the Counsellors folder I used the following views to present our data:



I used the following code to make a \_CounsellorSummary using panels:

@model ChangeCounselling.Data.Models.Counsellor

<div class="panel panel-default">

<div class="panel-heading">

<h3 class="panel-title">@Model.CouncellorFirstName, @Model.CouncellorLastName</h3>

</div>

<div class="panell-body">

@Html.ActionLink("Edit", "Edit", new { id = Model.CounsellorID }) |

@Html.ActionLink("Details", "Details", new { id = Model.CounsellorID }) |

@Html.ActionLink("Delete", "Delete", new { id = Model.CounsellorID })

</div>

</div>

The Index view:



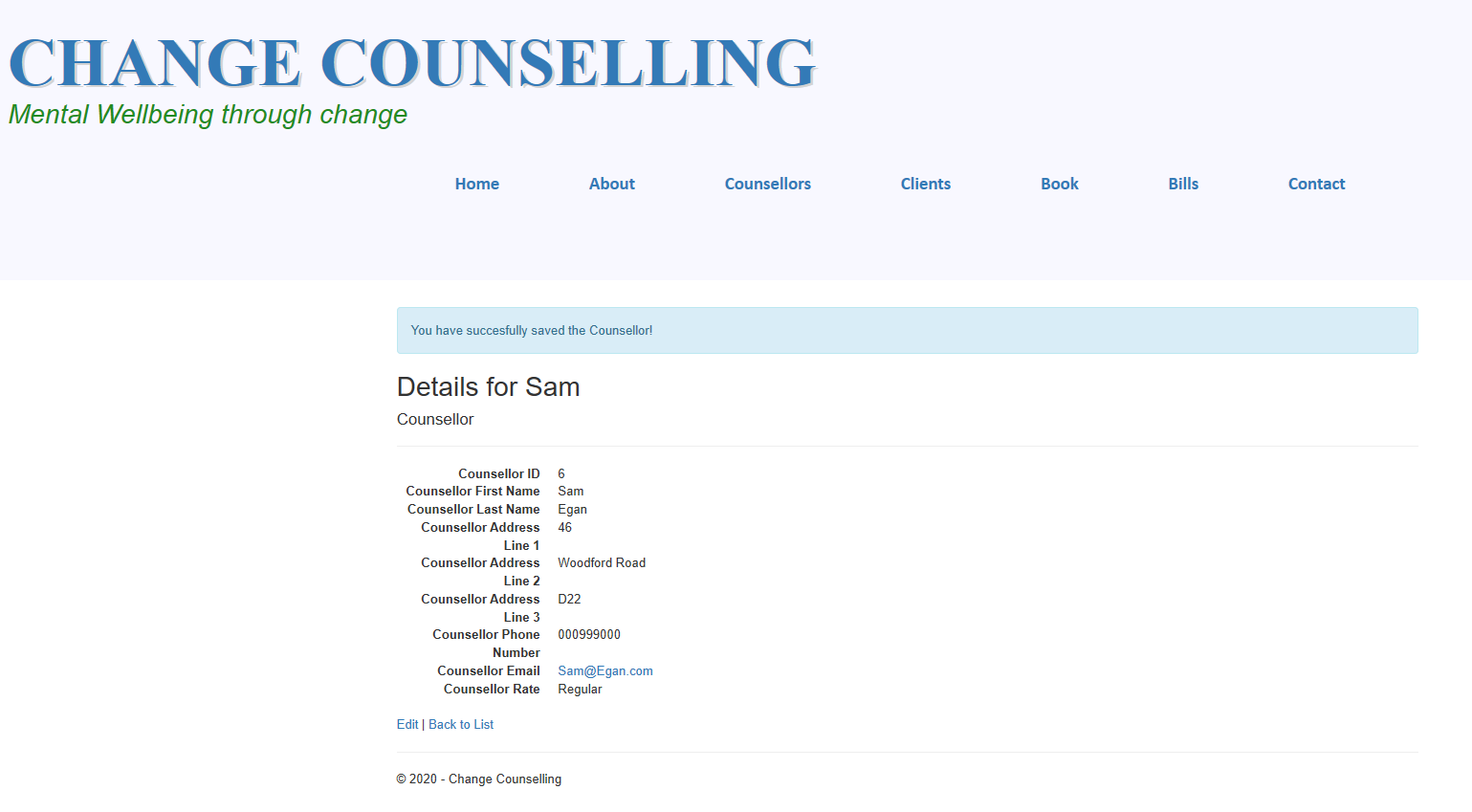
Create view:



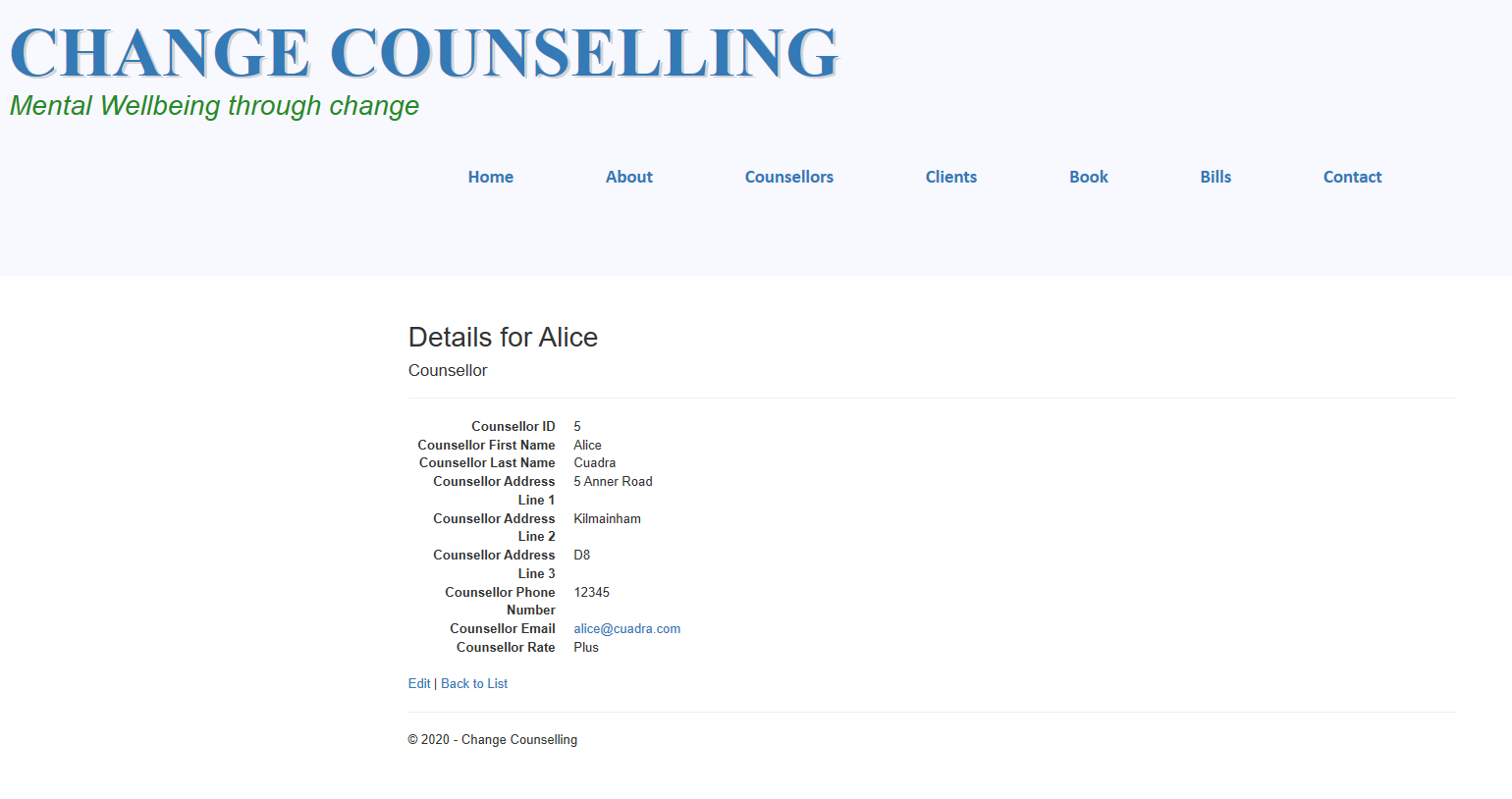
Edit view:



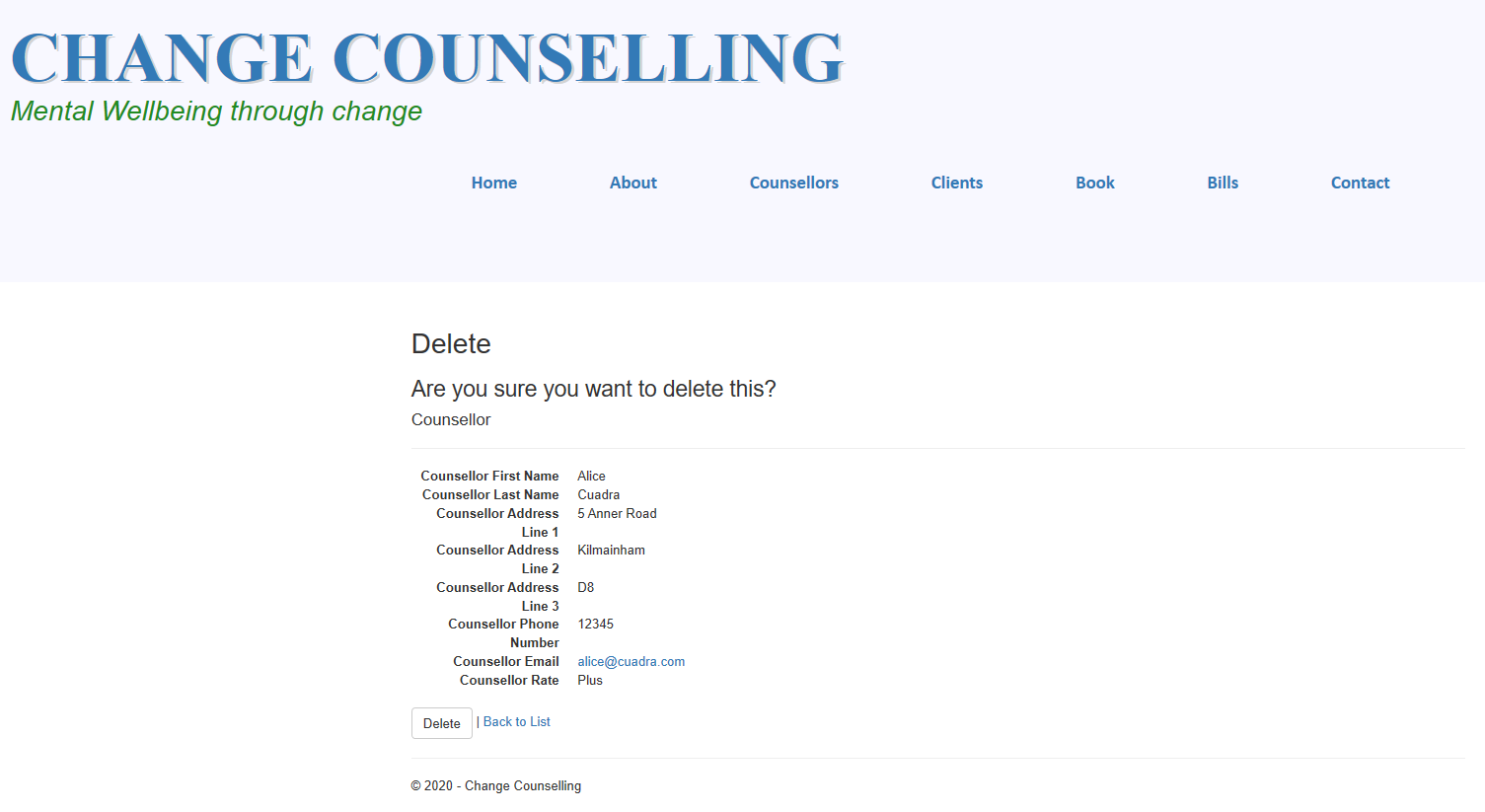
Succesfull Edit view:



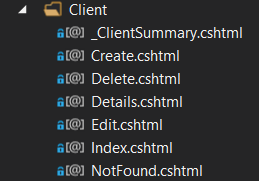
Details view:



Delete view:



For the Client folder I used the following views to present our data:



I used the following code to make a \_ClientSummary using panels:

@model ChangeCounselling.Data.Models.Client

<div class="panel panel-default">

<div class="panel-heading">

<h3 class="panel-title"> @Model.ClientFirstName @Model.ClientLastName</h3>

</div>

<div class="panell-body">

@Html.ActionLink("Edit", "Edit", new { id = Model.ClientID }) |

@Html.ActionLink("Details", "Details", new { id = Model.ClientID }) |

@Html.ActionLink("Delete", "Delete", new { id = Model.ClientID })

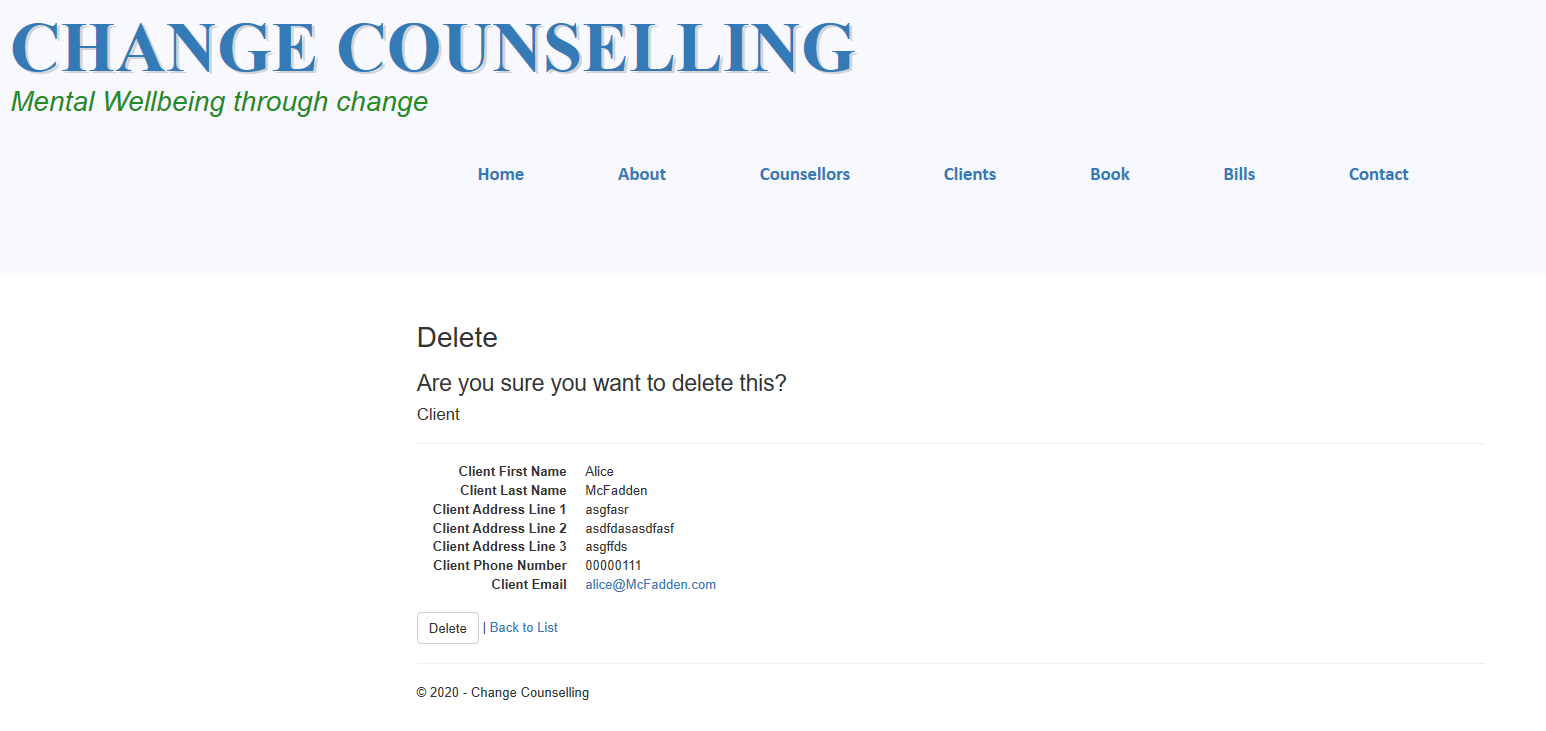
</div>

</div>

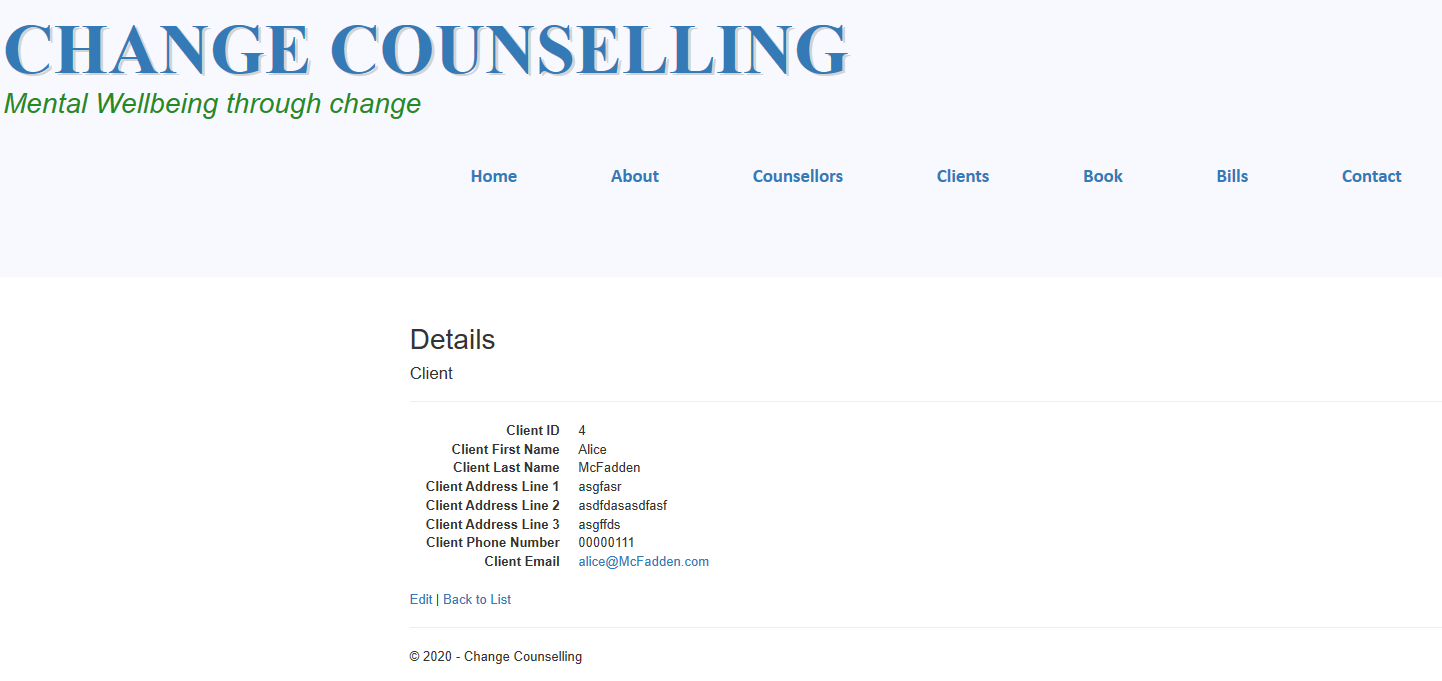
Create view:



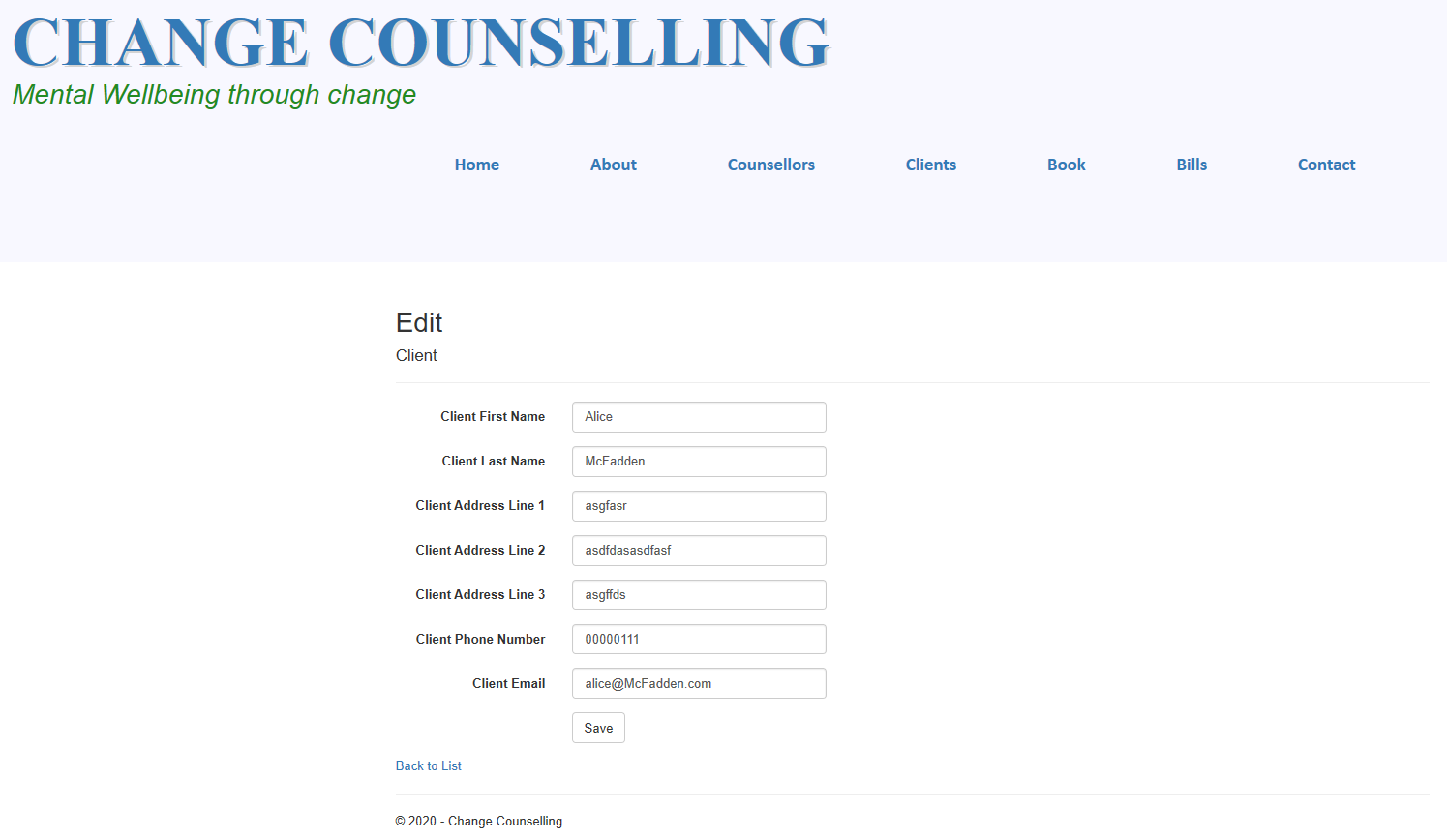
Delete view:



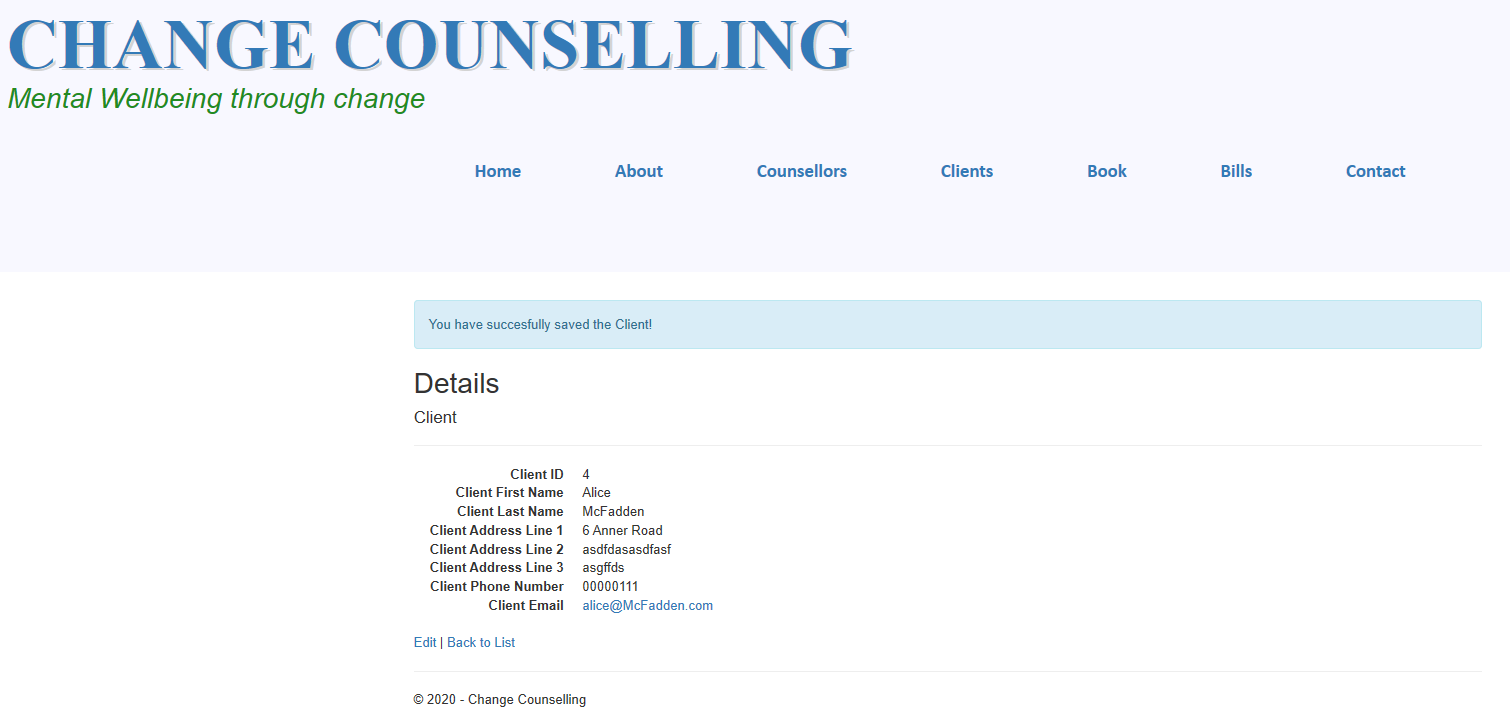
Details view:



Edit view:



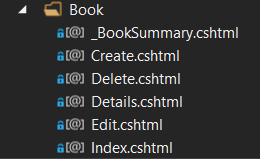
Successfull Edit view:



Index view:



For the Book folder I used the following views to present our data:



I used the \_BookSummary view to make panel presentation, see the code below:

@model ChangeCounselling.Data.Models.Book

<div class="panel panel-default">

<div class="panel-heading">

<p>Booking ID</p>

<h3 class="panel-title">@Model.BookID</h3>

</div>

<div class="panell-body">

@Html.ActionLink("Edit", "Edit", new { id = Model.BookID }) |

@Html.ActionLink("Details", "Details", new { id = Model.BookID }) |

@Html.ActionLink("Delete", "Delete", new { id = Model.BookID })|

@if (Model.Bills.Count == 0)

{

@Html.ActionLink("Create Bill", "CreateBill", new { id = Model.BookID })

}

else

{

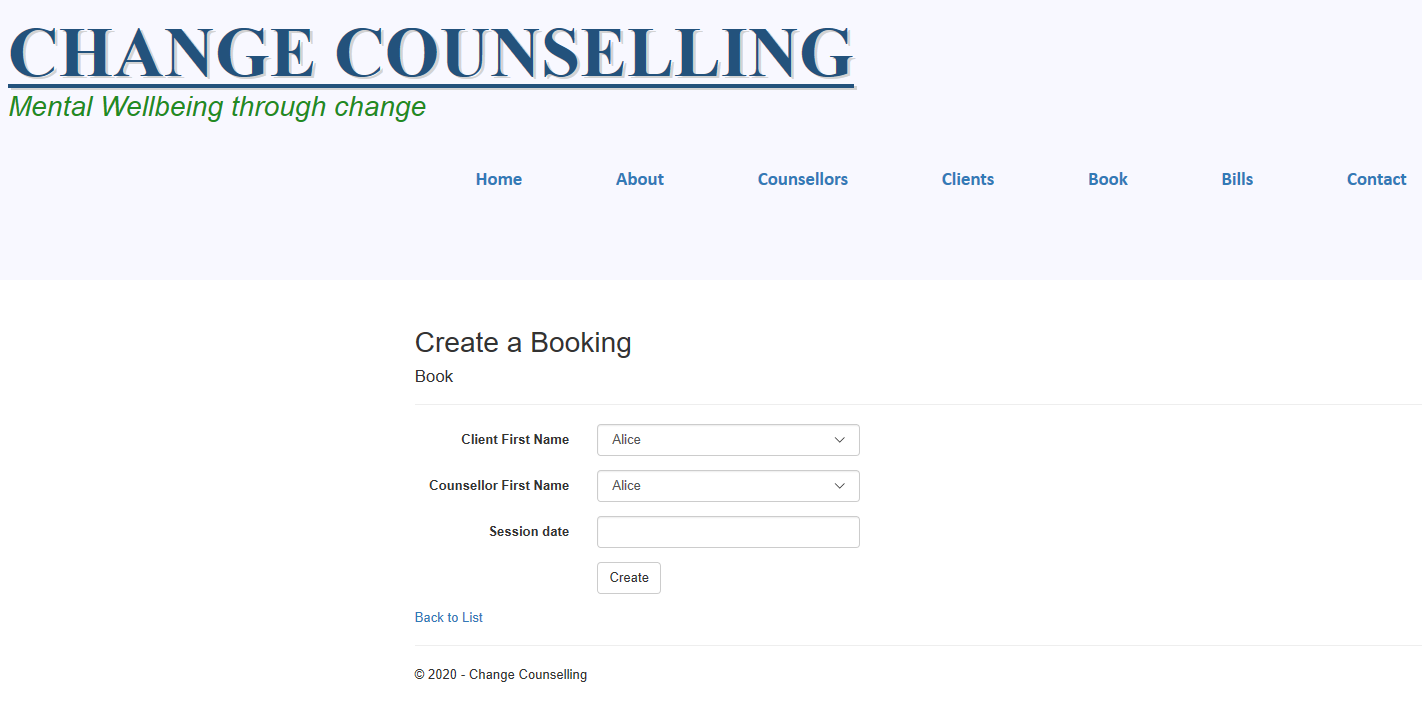
@Html.ActionLink("Go to Bill", "Details","Bill", new { id = Model.Bills.First().BillID },null)

}

</div>

</div>

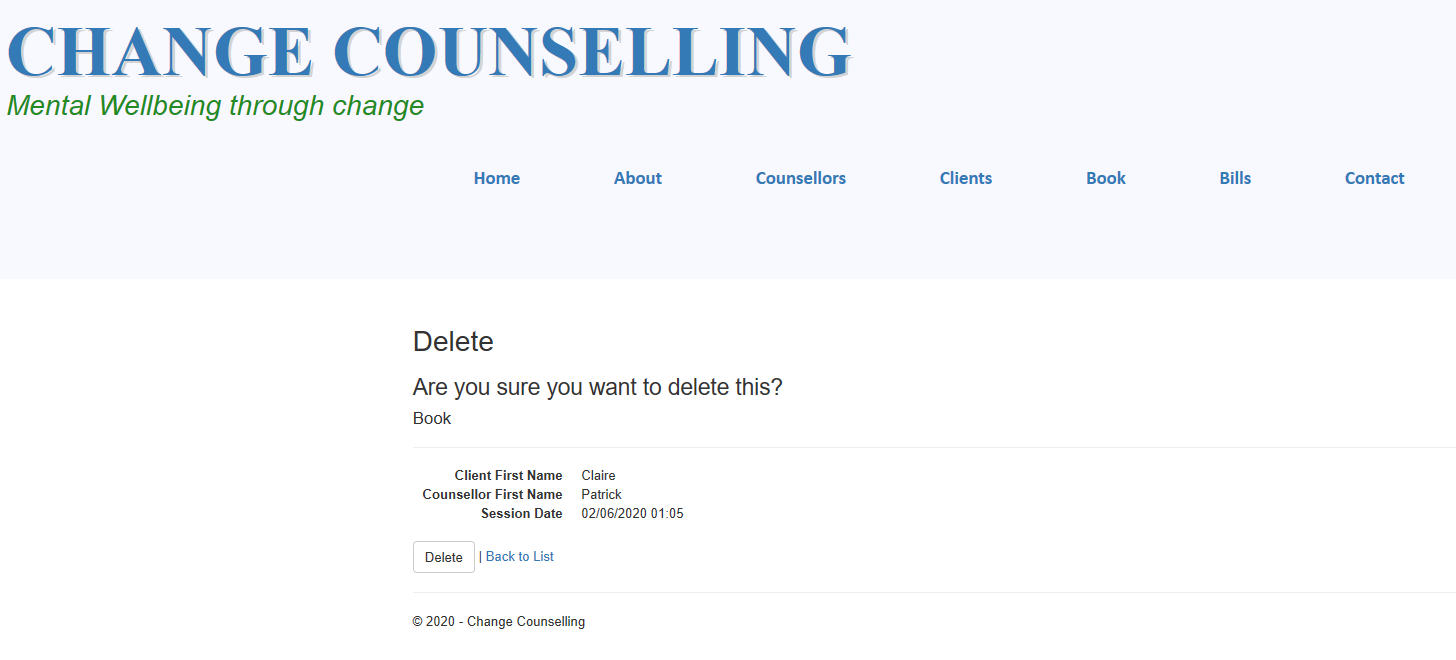
Create view:



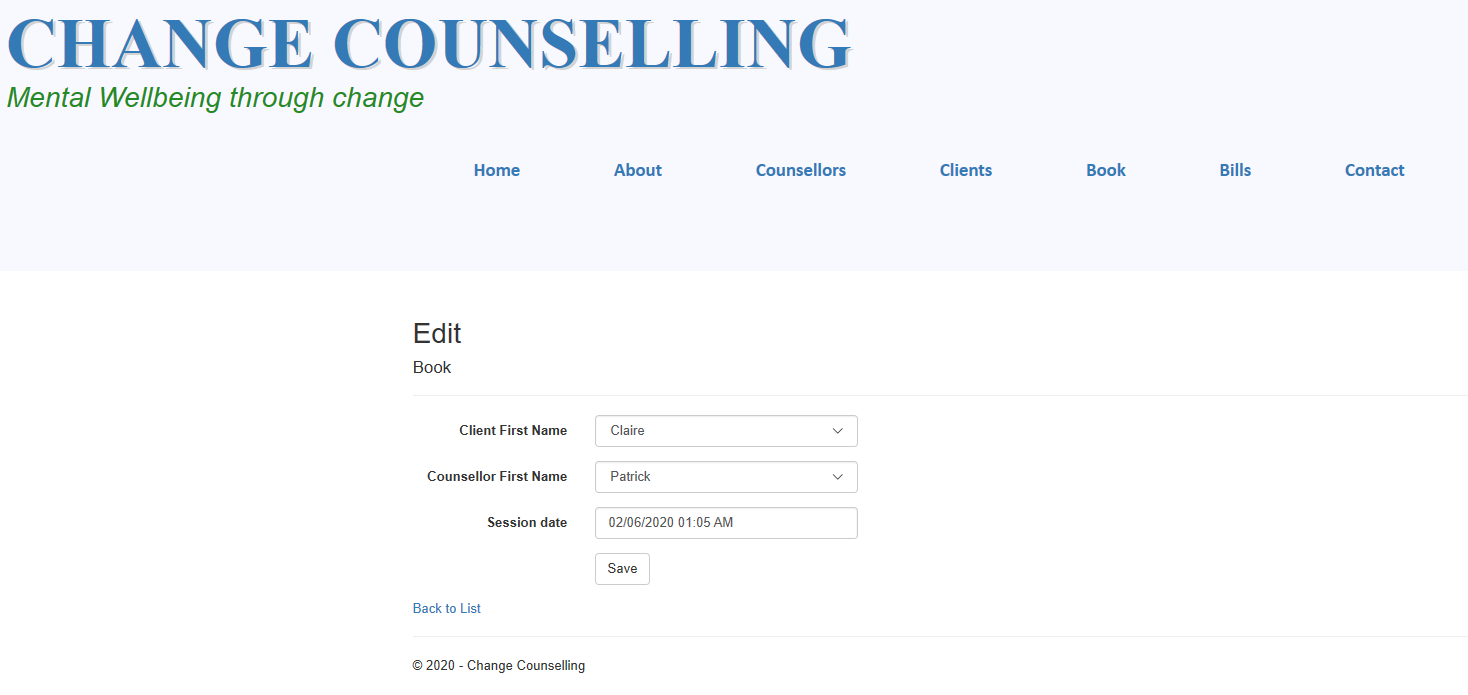
Details view:



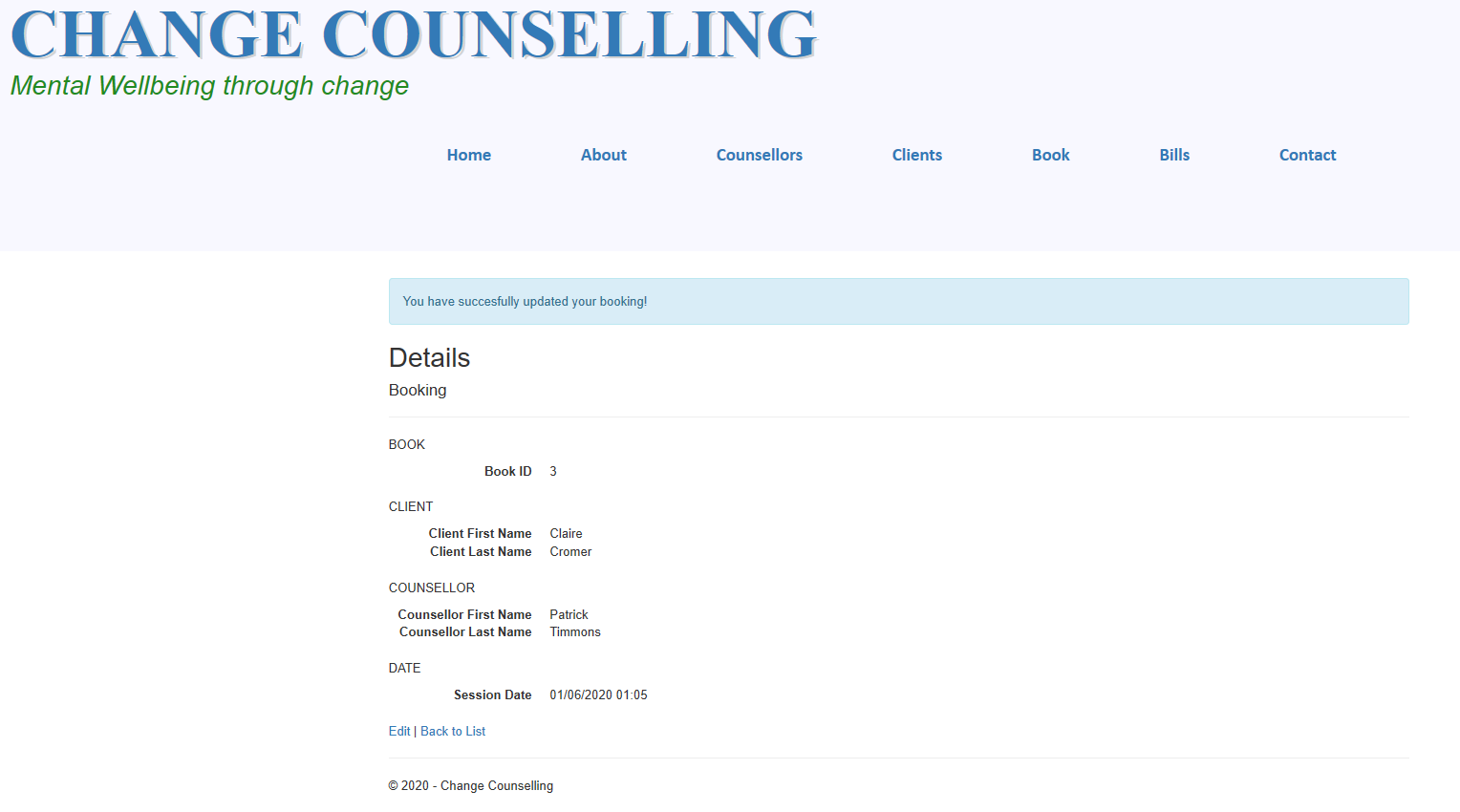
Delete view:



Edit view:



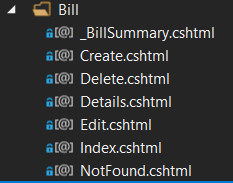
Successfull Edit view:



Index view:



For the Bill folder initialy I used the following views to present data, however, last minuet desicion was to leave it as only an Index and Details view:



I used the \_BillSummary view to make panel presentation, see the code below:

@model ChangeCounselling.Data.Models.Bill

<div class="panel panel-default">

<div class="panel-heading">

<h1 class="panel-title"><p>Bill ID</p>@Model.BillID</h1>

<h5 class="panel-footer"><p>Book ID</p>@Model.BookID</h5>

</div>

<div class="panell-body">

@\*@Html.ActionLink("Edit", "Edit", new { id = Model.BillID }) |\*@

@Html.ActionLink("Details", "Details", new { id = Model.BillID }) |

@\*@Html.ActionLink("Delete", "Delete", new { id = Model.BillID })\*@

</div>

</div>

Details view:



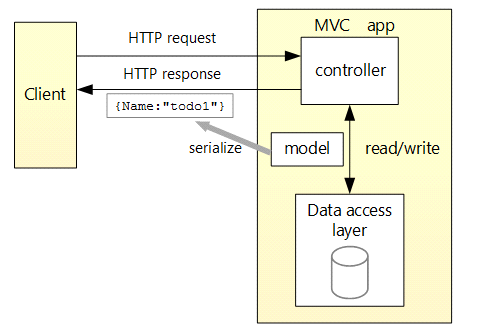
Index view:



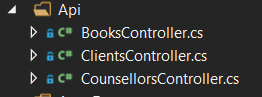
**1.7 Web API**

ASP.NET Web API is a framework that makes it easy to build HTTP services that reach a broad range of clients, including browsers and mobile devices. ASP.NET Web API is an ideal platform for building RESTful applications on the .NET Framework.

The verbs themselves are included in the API and each of these endpoints end up being a separate URI.



The API Controllers were created for 3 Model intances:



The API code for BooksContoller is:

using ChangeCounselling.Data.Models;

using ChangeCounselling.Data.Services;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Http;

using System.Web.Mvc;

namespace ChangeCounselling.Web.Api

{

public class BooksController : ApiController

{

private readonly IBookData db;

public BooksController(IBookData db)

{

this.db = db;

}

public IEnumerable<Book> Get()

{

var result = db.GetAll().ToList();

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

foreach(var item in result)

{

books.Add(new Book {

BookID=item.BookID,

CounsellorID=item.CounsellorID,

ClientID=item.ClientID,

DateTime=item.DateTime

});

}

return books;

}

}

}

The API code for ClientsContoller is:

using ChangeCounselling.Data.Models;

using ChangeCounselling.Data.Services;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Net;

using System.Net.Http;

using System.Web.Http;

namespace ChangeCounselling.Web.Api

{

public class ClientsController : ApiController

{

private readonly IClientData db;

public ClientsController(IClientData db)

{

this.db = db;

}

public IEnumerable<Client> Get()

{

var result = db.GetAll().ToList();

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

foreach (var item in result)

{

clients.Add(new Client

{

ClientID = item.ClientID,

ClientFirstName = item.ClientFirstName,

ClientLastName = item.ClientLastName,

ClientAddressLine1 = item.ClientAddressLine1,

ClientAddressLine2 = item.ClientAddressLine2,

ClientAddressLine3 = item.ClientAddressLine3,

ClientEmail = item.ClientEmail,

ClientPhone = item.ClientPhone

});

}

return clients;

}

}

}

The API code for CounsellorsContoller is:

using ChangeCounselling.Data.Models;

using ChangeCounselling.Data.Services;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Net;

using System.Net.Http;

using System.Web.Http;

namespace ChangeCounselling.Web.Api

{

public class CounsellorsController : ApiController

{

private readonly ICounsellorData db;

public CounsellorsController(ICounsellorData db)

{

this.db = db;

}

public IEnumerable<Counsellor> Get()

{

var result = db.GetAll().ToList();

List<Counsellor> counsellors = new List<Counsellor>();

foreach (var item in result)

{

counsellors.Add(new Counsellor

{

CounsellorID = item.CounsellorID,

CouncellorFirstName = item.CouncellorFirstName,

CouncellorLastName = item.CouncellorLastName,

CouncellorAddressLine1 = item.CouncellorAddressLine1,

CouncellorAddressLine2 = item.CouncellorAddressLine2,

CouncellorAddressLine3 = item.CouncellorAddressLine3,

CouncellorEmail = item.CouncellorEmail,

CouncellorPhone = item.CouncellorPhone,

CouncellorRate = item.CouncellorRate

});

}

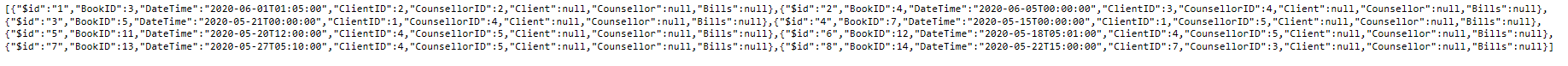
return counsellors;

}

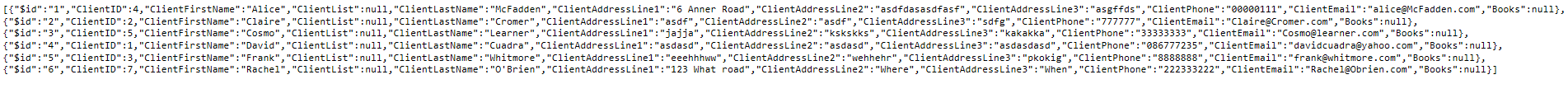
}

}

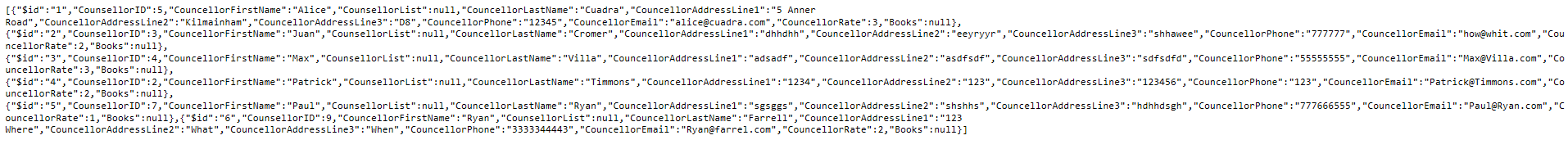
**API to get all the Bookings:**



**API to get all the Clients:**

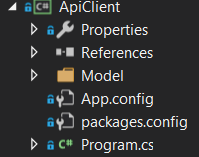


**API to get all the Counsellors:**



**1.7.2 API Client**

The APIClient console program created is as a proyect within the solution as follows:



The Program.cs code is below:

using ApiClient.Model;

using Newtonsoft.Json;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Net.Http;

using System.Net.Http.Headers;

using System.Text;

using System.Threading.Tasks;

namespace ApiClient

{

class Program

{

public static async Task GetAsync()

{

try

{

using (HttpClient client = new HttpClient())

{

client.BaseAddress = new Uri("http://localhost:50547/");

//add an Accept header for JSON

client.DefaultRequestHeaders.

Accept.Add(new MediaTypeWithQualityHeaderValue("application/json"));

//Get/property/all

HttpResponseMessage response = await client.GetAsync("Api/Books");

if (response.IsSuccessStatusCode)

{

//read results

var stringJson = await response.Content.ReadAsStringAsync();

var booking = JsonConvert.DeserializeObject<List<Book>>(stringJson);

foreach (var book in booking)

{

Console.WriteLine("Booking ID is" + book.BookID);

Console.WriteLine("Client ID is" + book.ClientID);

Console.WriteLine("Date is" + book.DateTime);

Console.WriteLine("Counsellor ID is" + book.CounsellorID);

}

}

else

{

Console.WriteLine(response.StatusCode + " " + response.ReasonPhrase);

}

}

}

catch (Exception e)

{

Console.WriteLine(e.ToString());

}

}

static void Main(string[] args)

{

GetAsync().Wait();

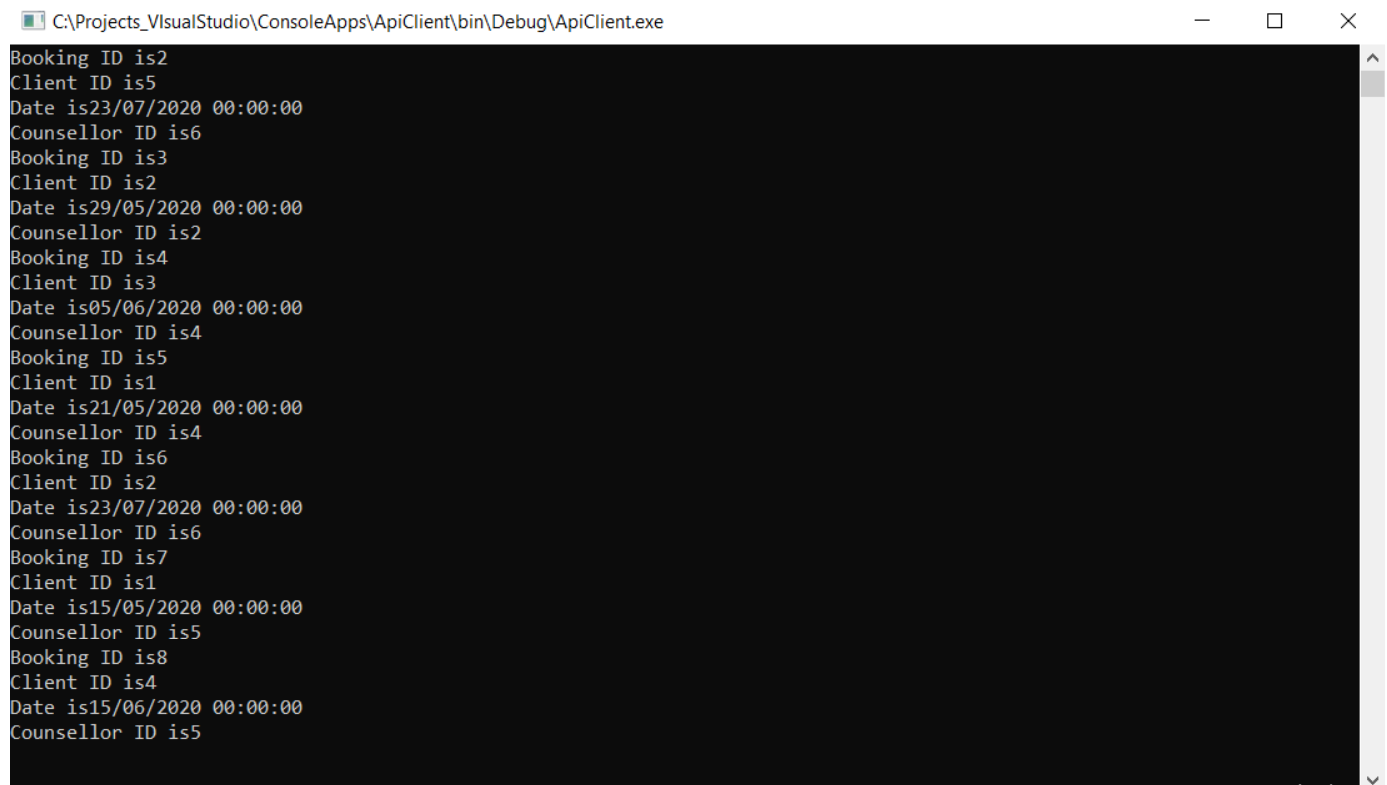
Console.ReadLine();

}

}

}

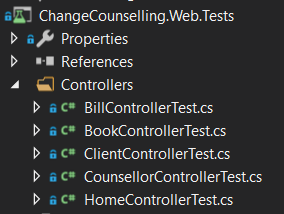
**API Client view**



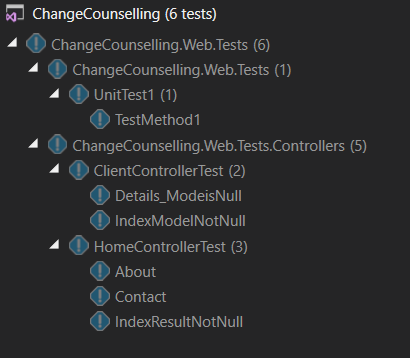
**1.8 Unit Tests**

Unit testing takes the smallest piece of testable software in the application and determines whether it behaves exactly as you expect. Each unit is tested separately before integrating them into modules to test the interfaces between modules.

In our project we tested the all 5 Controllers :



Tests did not show falier but didnt show passed. I ran tests as per URI addresssing Scheme below:



|  |  |  |  |
| --- | --- | --- | --- |
| **URI Addressing Scheme** | | | |
| **Action Type** | **Route** | **Purpose** |
| Get | <https://changecounsellingweb12345.azurewebsites.net/Book> | Get all the bookings |
| Get | <https://changecounsellingweb12345.azurewebsites.net/Home/About> | Access the About Page |
| Post | <https://changecounsellingweb12345.azurewebsites.net/Client/Create> | Create Client POST |
| Delete | <https://changecounsellingweb12345.azurewebsites.net/Client/Delete/1> | Delete BOOKING with Identifier 1 |