**Technical University Dublin, Tallaght Campus, Department of Computing Higher Diploma in Computing in Science**

**2019**

|  |
| --- |
| **Enterprise Applications Development** |
| **CA2** |
|  |
|  |
|  |
| **Autors:**  **Aliona Mocanu X00134659**  **David Cuadra**  **X00143907** |
|  |
|  |

**Content**

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

**Introduction.........................................................................................................3**

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

**Controllers...........................................................................................................9**

**Views..................................................................................................................12**

**API......................................................................................................................16**

**APIClient...........................................................................................................18**

**UnitTests............................................................................................................20**

**URI Addressing Scheme...................................................................................21**

**Overview**

**Booking rooms in Knockmitten Youth and Community centre**

**Knockmitten Youth and Community** centre has 3 rooms and a foyer. These rooms are available to the public. The Client can book a room for different kind of activities.

System admin can create, update and delete the booking data.

**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/Knockmitten-Booking>
* For the Console Client Application: <https://github.com/DavidCuadra/Books>

**AZURE LINKS:**

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

* Azure app link: <https://bookingca2.azurewebsites.net/>
* Azure API link: <https://bookingca2.azurewebsites.net/api/BookingsAPI/GetBookingsAPI>

**Introduction**

**Room number**: We have 4 rooms including the foyer: room1, room2, room3 and the number for the foyer is Room4.

|  |  |  |
| --- | --- | --- |
| Room 1 | Room 2 | Room 3 |
| Foyer | | |

Clients can choose the number of rooms according to their requirements.

The noise level for each group is important when booking as the moving partition walls do not provide an adequate sound proofing from the other rooms.

**Noise levels**

|  |  |
| --- | --- |
| Noise levels are | |
| 1 | Need Silence |
| 2 | Normal |
| 3 | Noisy |

**Capacity per rooms is:**

|  |  |
| --- | --- |
| **Room Number** | **Capacity of people** |
| Room 1 | 15 |
| Room 2 | 15 |
| Room 3 | 12 |
| Foyer | 30 |

**Room ID is:**

|  |  |
| --- | --- |
| **Room ID** | **Room Number** |
| 1 | 1 |
| 2 | 2 |
| 3 | 3 |
| 4 | 4 |

**Teas/Coffees:**

The renters have the option to book teas and coffees for their members.

**The project will be created with an MVC Application**.

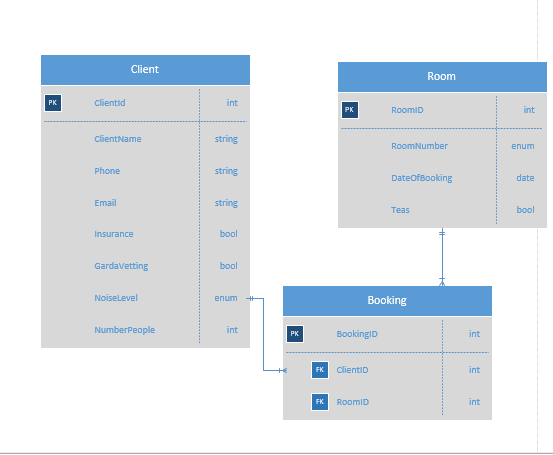
MVC is an [application](https://techterms.com/definition/application) design model comprised of three interconnected parts. They include the model ([data](https://techterms.com/definition/data)), the view ([user interface](https://techterms.com/definition/user_interface)), and the controller ([processes](https://techterms.com/definition/process) that handle input).

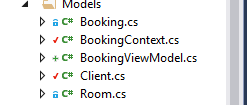
****

**1.2Models**

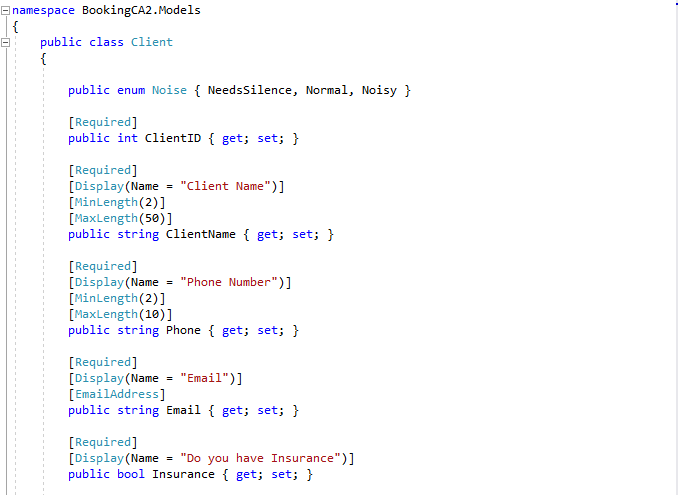
A model is data used by a program. This may be a [database](https://techterms.com/definition/database), [file](https://techterms.com/definition/file), or a simple object, such as an [icon](https://techterms.com/definition/icon) or a character.

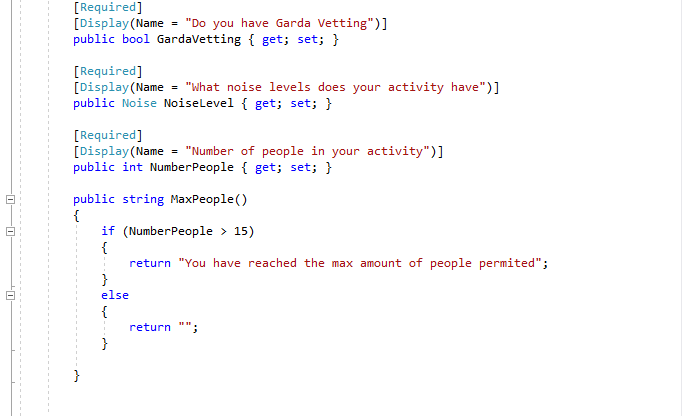
Our project will have 3 Models: The Client/s, Room/s and Booking.



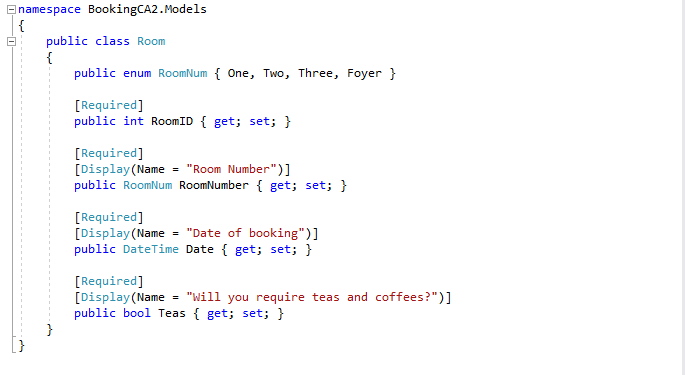


Our program will allow the client to enter the details like: client name, phone, email, insurance (if will have it), garda vetting (if will have one), to choose the noise level(need silence, normal or noisy) and to enter the number of people that will be in the group.

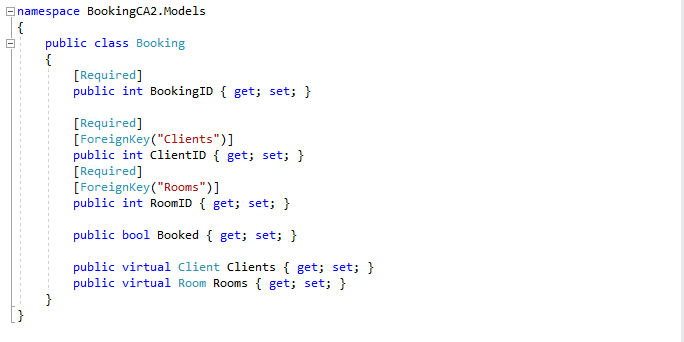




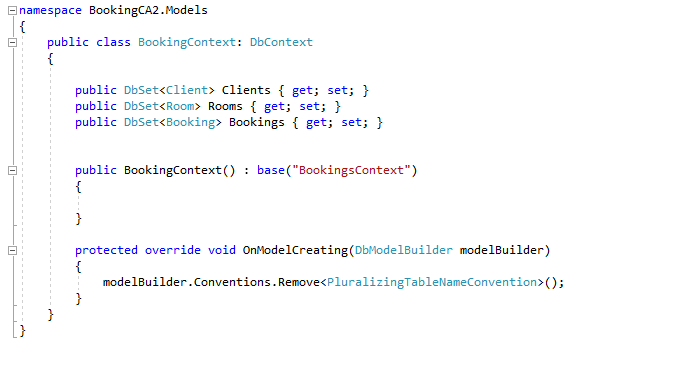
After this the client will be redirected to the second page that will be the Room, and will have to pick a room number, the date, and to choose if would like tea and coffee.



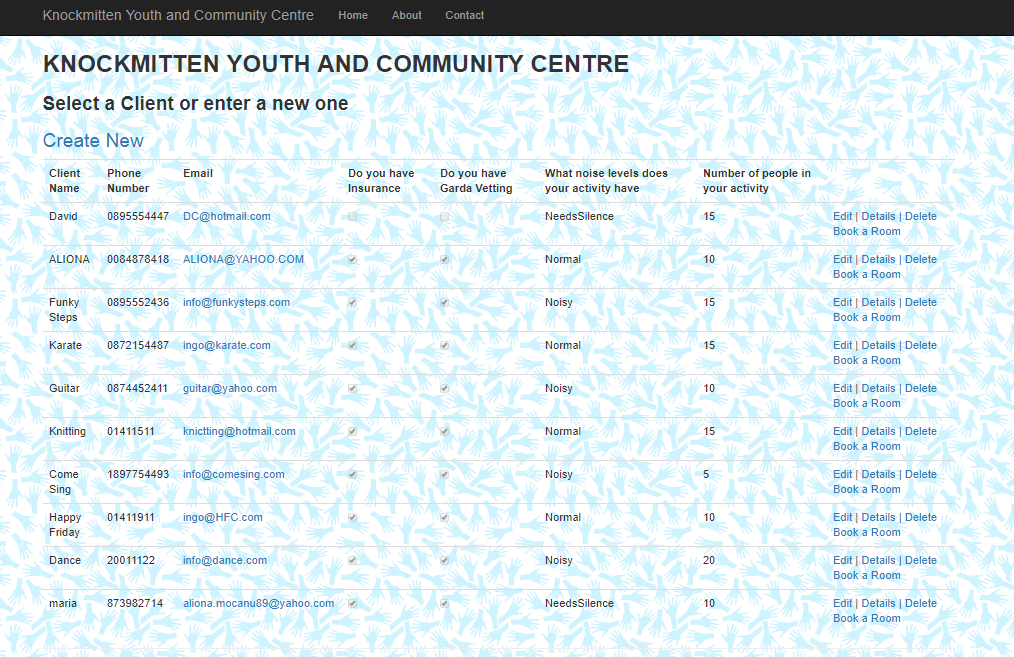
The 3rd Model is Booking. This is making the connection between the client and the room that was booked.



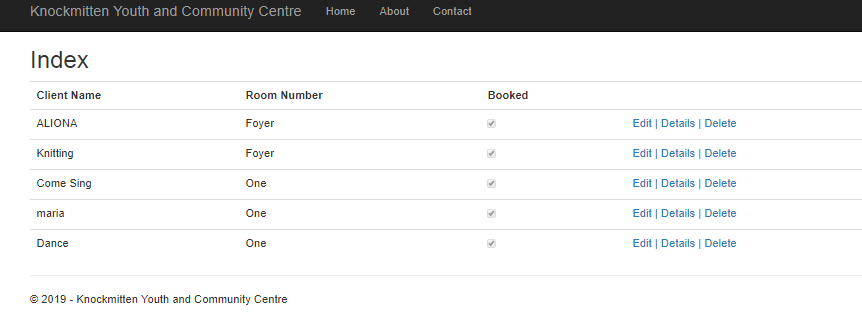
Our data is saved in the database, every time the client is entering the details:



The database for clients:

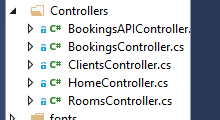


The database for rooms:



**1.3. Controllers**

A controller updates both models and views. It accepts [input](https://techterms.com/definition/input) 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.



**We have few Controllers. Bellow is one of them, for Room:**

namespace BookingCA2.Controllers

{

public class RoomsController : Controller

{

private BookingContext db = new BookingContext();

// GET: Rooms

public ActionResult Index(int? clientId)

{

ViewBag.clientId = clientId;

return View(db.Rooms.ToList());

}

// GET: Rooms/Details/5

public ActionResult Details(int? id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Room room = db.Rooms.Find(id);

if (room == null)

{

return HttpNotFound();

}

return View(room);

}

// GET: Rooms/Create

public ActionResult Create()

{

return View();

}

// POST: Rooms/Create

// To protect from overposting attacks, please enable the specific properties you want to bind to, for

//more details see https://go.microsoft.com/fwlink/?LinkId=317598.

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Create([Bind(Include = "RoomID,RoomNumber,Date,Teas")] Room room)

{

if (ModelState.IsValid)

{

// Booking b1 = new Booking() {ClientID= }

db.Rooms.Add(room);

db.SaveChanges();

return RedirectToAction("Index/", "Bookings");

}

return View();

}

// GET: Rooms/Edit/5

public ActionResult Edit(int? id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Room room = db.Rooms.Find(id);

if (room == null)

{

return HttpNotFound();

}

return View(room);

}

// POST: Rooms/Edit/5

// To protect from overposting attacks, please enable the specific properties you want to bind to, for

// more details see https://go.microsoft.com/fwlink/?LinkId=317598.

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Edit([Bind(Include = "RoomID,RoomNumber,Date,Teas")] Room room)

{

if (ModelState.IsValid)

{

db.Entry(room).State = EntityState.Modified;

db.SaveChanges();

return RedirectToAction("Index");

}

return View(room);

}

// GET: Rooms/Delete/5

public ActionResult Delete(int? id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Room room = db.Rooms.Find(id);

if (room == null)

{

return HttpNotFound();

}

return View(room);

}

// POST: Rooms/Delete/5

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

public ActionResult DeleteConfirmed(int id)

{

Room room = db.Rooms.Find(id);

db.Rooms.Remove(room);

db.SaveChanges();

return RedirectToAction("Index");

}

protected override void Dispose(bool disposing)

{

if (disposing)

{

db.Dispose();

}

base.Dispose(disposing);

}

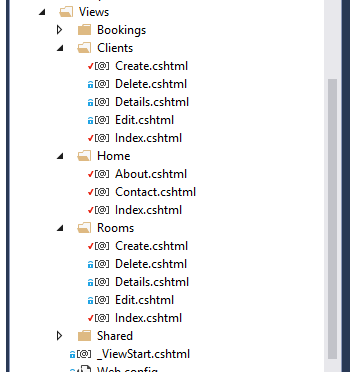
}

}

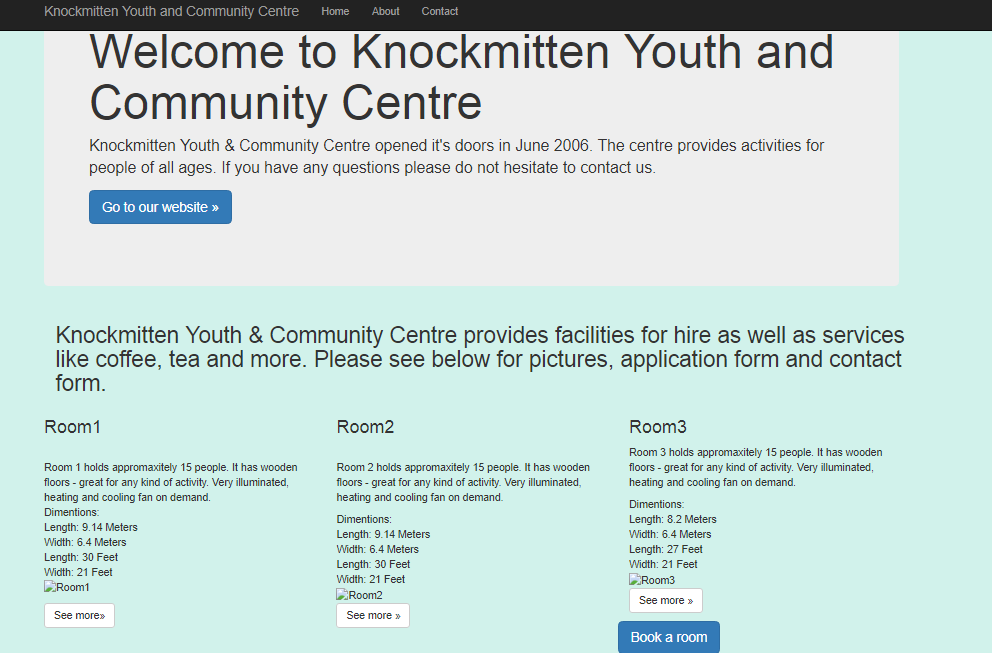
The three parts of MVC are interconnected . The view displays the model for the user. The controller accepts user input and updates the model and view accordingly.

**1.4 Views**

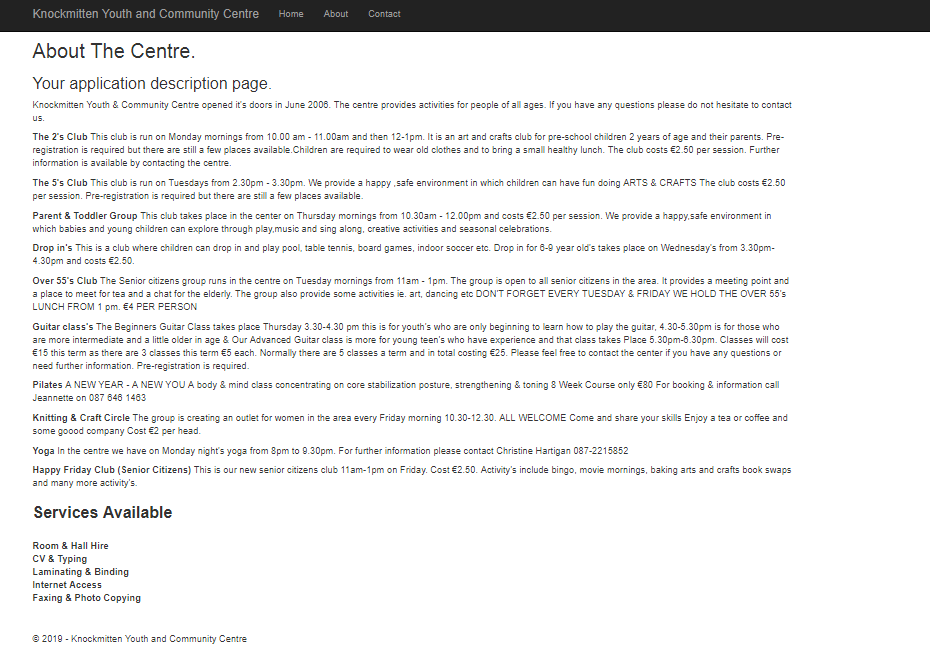
A view is the means of displaying objects within an application. Examples include displaying a [window](https://techterms.com/definition/window) or buttons or text within a window. It includes anything that the user can see.



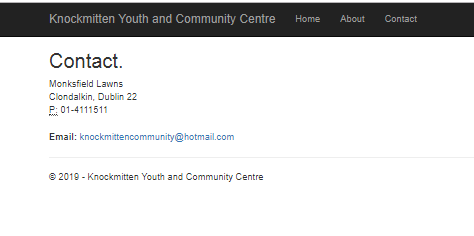
Home Page:



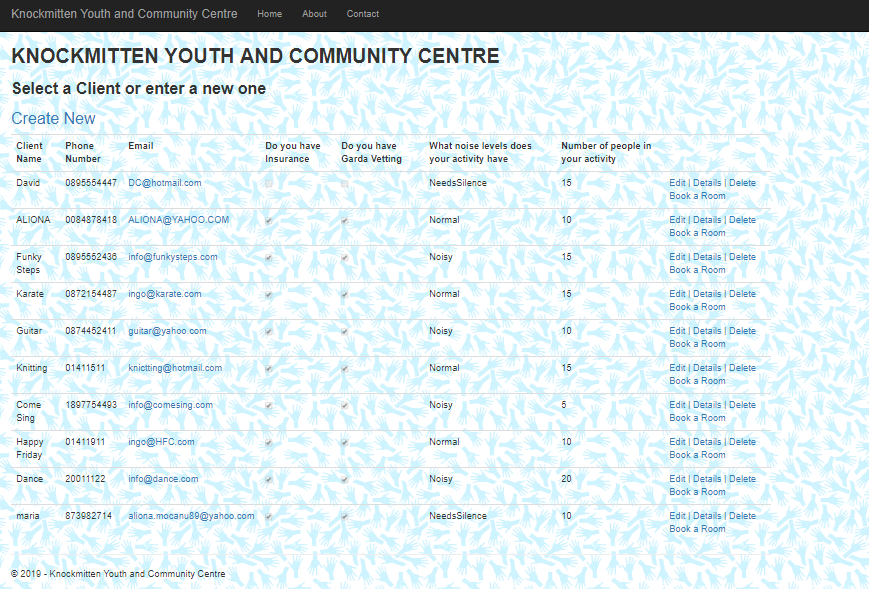
**About Page**

****

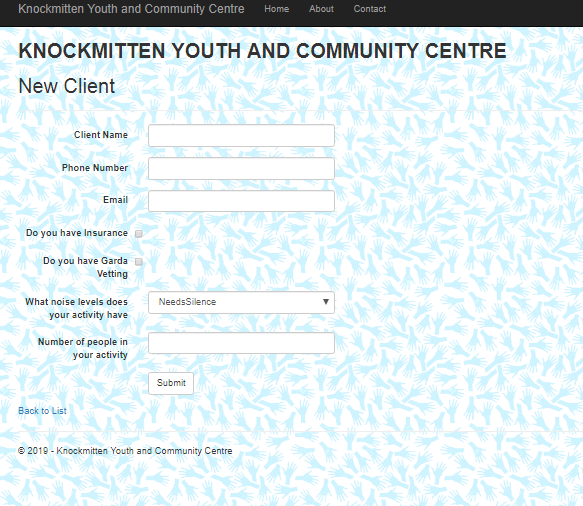
**Contact page:**

****

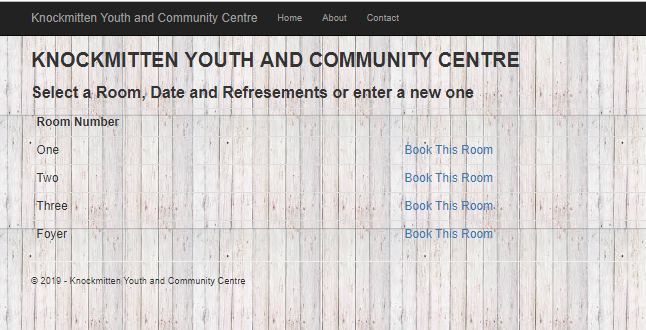
Our Client is able to create a new client, or if he is already in our database he can choose just the room that wants to rent:



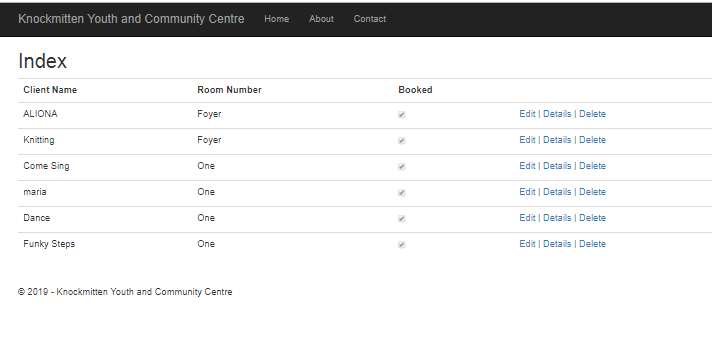
An existing client can rebook the room, or can create a new client.



After the client input all data, he will have to choose one of our four rooms available:



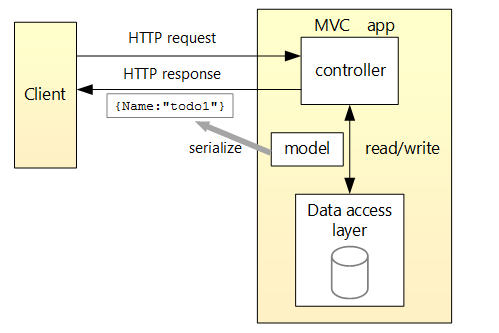
At the end we are displaying all the bookings:

The client can Edit, Delete or view the Details.

**1.5 ASP.NET 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 APIs, like Get Clients, Insert Room, Delete Booking, and that each of these endpoints end up being a separate URI.

****

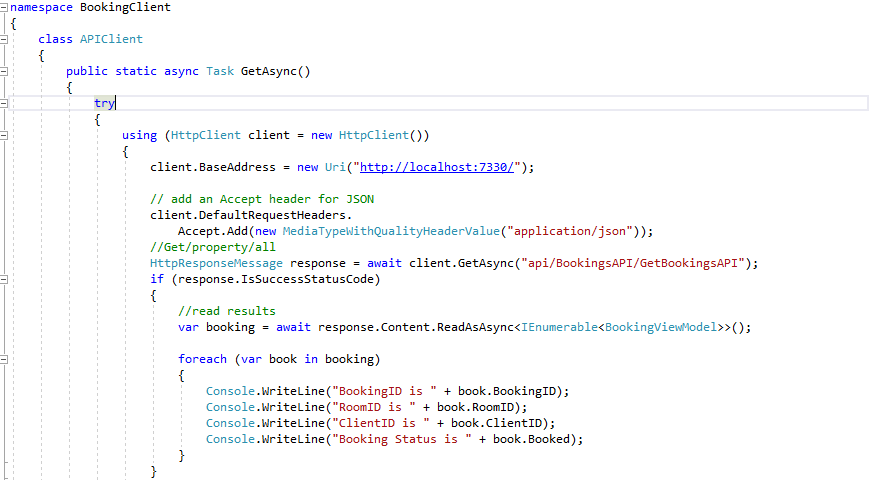
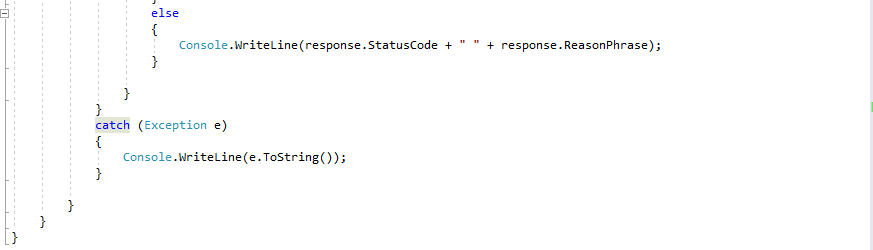
****

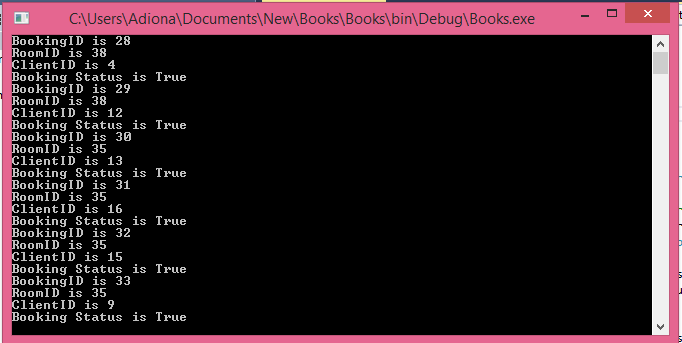
****

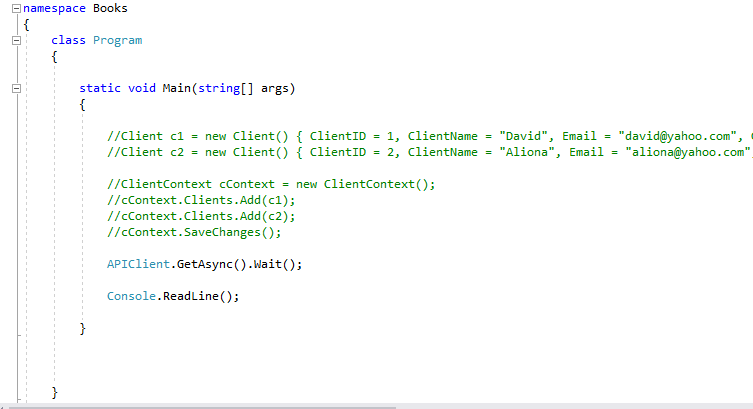
**API to get all the Bookings:**

****

**API/Client:**

** **

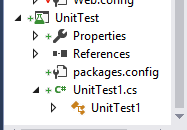
****

****

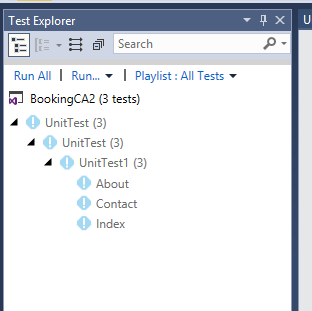
**1.6Unit Tests**

The primary goal of unit testing is to take the smallest piece of testable software in the application and determine 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 Home page, Contact page and About Page:



As result, our tests Failed:



|  |  |  |
| --- | --- | --- |
| **URI Addressing Scheme** | | |
| **Action Type** | **Route** | **Purpose** |
| Get | <https://bookingca2.azurewebsites.net/Bookings> | Get all the bookings |
| Get | <https://bookingca2.azurewebsites.net/Home/About> | Access the About Page |
| Post | <https://bookingca2.azurewebsites.net/Clients/Create> | Create Client POST |
| Delete | <https://bookingca2.azurewebsites.net/Bookings/Delete/32> | Delete BOOKING with Identifier 32 |