**I. Introduction.**

**1 Tạo tài khoản TFS**

**2. Giới thiệu về Project GigHub**

GigHub introduction.

Cho phép người yêu nhạc có thể track các buổi hòa nhạc của các nghệ sĩ mà họ thích.

Nghệ sỹ có thể đăng nhập để liệt kê các buổi hòa nhạc của họ, khi họ thêm một buổi hòa nhạc thì no có thời gian (ngày, giờ) và vị trí, và thể loại của buổi hòa nhạc đó.

Nghệ sỹ có một trang để gọi là các buổi hòa nhạc sắp tới. Từ trang này , họ có khả nặng chỉnh sửa và gỡ bỏ một buổi hòa nhạc đang có hoặc có thể thê một buổi hòa nhạc khác vào danh sách của họ.

Người sử dụng có quyền để xem các buổi hòa nhạc sắp tới và tìm kiếm nghệ sĩ , thể loại và vị trí của buổi hòa nhạc đó. Họ có thể xem chi tiết thông tin của một buổi hòa nhạc và thêm nó vào lịch biểu của họ.

Người dùng cũng có quyền theo dõi các nghệ sỹ yêu thích của họ, hi họ theo dõi một nghệ sỹ nào đó, khi mà họ theo dõi ai đó thì họ sẽ nhận được các thông báo về các buổi hòa nhạc sắp tởi của nghệ sỹ đó ở phần Gig Feed.

**3. Các chức năng chính.**

**Authentication:**

sign up

Login

Log out

Change password

Edit profile.

**Gigs**

Add a gig

My upcoming gigs (for artist)

Edit a gig

Remove a gig

View all upcoming gigs

Search

View gig details

**Gig Calendar**

Add a gig to calendar

Remove a gig from calendar

view gigs I’m attending

**Following**

Follow an artist

Unfollow an artist

Who I’m following

Gig feed.

**4. Dependencies.**

Tuân theo thứ tự.

**1. View Gigs => 2. Follow Artist => 3. Unfollow Artist.**

Thứ tự các hành động.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 6 |
| Add a gig | My upcoming gigs  All upcoming gigs | Edit a gig  Remove a gig  Add a gig to calendar  Follow an artist  Search  View gig details | View gigs I’m attending  Who I’m following | Remove a gig from calendar.  Unfollow an artist |

**II. Building a Model using code-first workflow.**

**1. Enable migration.**

From package manager console => gõ **enable-migrations**

Lệnh này chỉ chạy một lần duy nhất lúc vừa tạo project.

Ý tưởng của công nghệ này là mỗi lần ta có thay đổi ở Domain modal, thì ta tạo ra migration, và chạy nó bên csdl và do vậy csdl của ta cũng đươc cập nhật.

Và chú ý là mỗi lần có thay đổi nhỏ thì cũng nó cũng phải được migrations

**2. Thêm một migration.**

Gõ lệnh **add-migration + tên**

**3. Thực thi migration trên cơ sơ dữ liệu.**

Gõ lệnh **update-database**

**4. Create Domain classes.**

**Step 1: Tạo các classes**

|  |
| --- |
| public class Genre  {  public byte Id { get; set; }  public string Name { get; set; }  } |

|  |
| --- |
| public class Gig  {  public int Id { get; set; }  public ApplicationUser Artists { get; set; }  public DateTime DateTime { get; set; }  public string Venue { get; set; }  public Genre Genre { get; set; }  } |

**Step 2:**

Nếu bạn muốn thêm một domain class (như 2 classes ở trên) thì nó phải được tham chiếu đến nếu không EF không biết nó ở trong domain model.

Vì Gig nó có tham chiếu đến Genre nên lúc thực thi EF sẽ tìm đến và tạo Genre.

Nhưng ở đây, nếu bạn muốn lấy một list các Genre thì bạn phải khai báo như bên dưới.

Từ file IdentityModels.cs trong Models.

|  |
| --- |
| using System.Data.Entity;  using System.Security.Claims;  using System.Threading.Tasks;  using Microsoft.AspNet.Identity;  using Microsoft.AspNet.Identity.EntityFramework;  namespace GigHub.Models  {  // You can add profile data for the user by adding more properties to your ApplicationUser class, please visit http://go.microsoft.com/fwlink/?LinkID=317594 to learn more.  public class ApplicationUser : IdentityUser  {  public async Task<ClaimsIdentity> GenerateUserIdentityAsync(UserManager<ApplicationUser> manager)  {  // Note the authenticationType must match the one defined in CookieAuthenticationOptions.AuthenticationType  var userIdentity = await manager.CreateIdentityAsync(this, DefaultAuthenticationTypes.ApplicationCookie);  // Add custom user claims here  return userIdentity;  }  }  public class ApplicationDbContext : IdentityDbContext<ApplicationUser>  {  public ApplicationDbContext()  : base("DefaultConnection", throwIfV1Schema: false)  {  }  public static ApplicationDbContext Create()  {  return new ApplicationDbContext();  }  }  } |

Ta tách thành 2 files (2 lớp riêng biệt).

ApplicationDbContext.cs

|  |
| --- |
| public class ApplicationDbContext : IdentityDbContext<ApplicationUser>  {  public DbSet<Gig> Gigs { get; set; }  public DbSet<Genre> Genres { get; set; }  public ApplicationDbContext()  : base("DefaultConnection", throwIfV1Schema: false)  {  }  public static ApplicationDbContext Create()  {  return new ApplicationDbContext();  }  } |

ApplicationUser.cs

|  |
| --- |
| public class ApplicationUser : IdentityUser  {  public async Task<ClaimsIdentity> GenerateUserIdentityAsync(UserManager<ApplicationUser> manager)  {  // Note the authenticationType must match the one defined in CookieAuthenticationOptions.AuthenticationType  var userIdentity = await manager.CreateIdentityAsync(this, DefaultAuthenticationTypes.ApplicationCookie);  // Add custom user claims here  return userIdentity;  }  } |

**Step 3:**

Thực thi lệnh **add-migration + tên (nếu đó là lần đầu tiên)** còn nếu có sự thay đổi và generate lại thì gõ **add-migration CreateGigTable -force** để override nó.

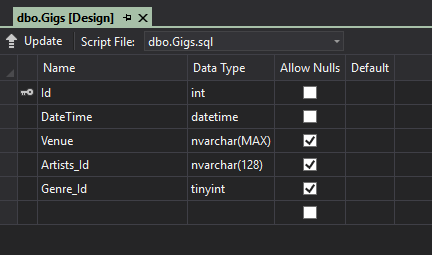
Step 4: nếu thành công thì ta sẽ thấy trong thư mục Migrations sẽ có một file cs có phần phía sau là CreateGigTable như nội dung sau:

|  |
| --- |
| namespace GigHub.Migrations  {  using System;  using System.Data.Entity.Migrations;    public partial class CreateGigTable : DbMigration  {  public override void Up()  {  CreateTable(  "dbo.Genres",  c => new  {  Id = c.Byte(nullable: false),  Name = c.String(),  })  .PrimaryKey(t => t.Id);    CreateTable(  "dbo.Gigs",  c => new  {  Id = c.Int(nullable: false, identity: true),  DateTime = c.DateTime(nullable: false),  Venue = c.String(),  Artists\_Id = c.String(maxLength: 128),  Genre\_Id = c.Byte(),  })  .PrimaryKey(t => t.Id)  .ForeignKey("dbo.AspNetUsers", t => t.Artists\_Id)  .ForeignKey("dbo.Genres", t => t.Genre\_Id)  .Index(t => t.Artists\_Id)  .Index(t => t.Genre\_Id);    }    public override void Down()  {  DropForeignKey("dbo.Gigs", "Genre\_Id", "dbo.Genres");  DropForeignKey("dbo.Gigs", "Artists\_Id", "dbo.AspNetUsers");  DropIndex("dbo.Gigs", new[] { "Genre\_Id" });  DropIndex("dbo.Gigs", new[] { "Artists\_Id" });  DropTable("dbo.Gigs");  DropTable("dbo.Genres");  }  }  } |

**Step 4** : Thực hiện lại việc cập nhật cơ sở dữ liệu.

Thực hiện gõ lệnh **Update-Database** trong **PM**

**5. Overriding code first conventions.**

Có một chút vấn đề ở đây.

- Như bạn thấy trường Venue là nvarchar(MAX) và được cho phép null, và ta không muốn điều này và chiều dài của nó thì nên là 255 là đủ

- Và bên dưới trường Artists\_Id cũng được cho phé null, và nó là kiểu nvarchar(128) chứ không phải kiểu Int, Lý do cho điều này xả ra là vì ở ApplicationUserTable thì khóa chính là kiểu nvarchar 128.

- Và tiếp theo là trường Genre\_Id nó cũng được cho phép null.

=> **Ta cần đến code first convertions.**

Và ta có 2 cách cho việc này là

+ **Data Annotations**

++ Easier

++ Has limiattions

+ **Fluent API**

++ More Powerful

++ A bit more complex.

Or đây ta sẽ sử dụng **Data Annotations.**

**Step 1**: Ta quay lại class Gig và thêm vào một số thứ để được như sau:

|  |
| --- |
| public class Gig  {  public int Id { get; set; }  [Required]  public ApplicationUser Artists { get; set; }  public DateTime DateTime { get; set; }  [Required]  [StringLength(255)]  public string Venue { get; set; }  [Required]  public Genre Genre { get; set; }  } |

Tương tự cho class Genre

|  |
| --- |
| public class Genre  {  public byte Id { get; set; }  [Required]  [StringLength(255)]  public string Name { get; set; }  } |

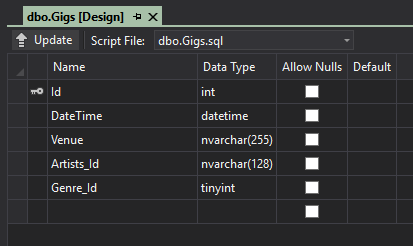
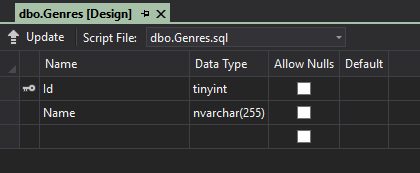
**Step 2:** Thực hiện cập nhật lại csdl.

**2.1** Trong màn hình PM ta gõ lệnh sau:

**add-migration OverrideCenventionsForGidsAndGenres**

**2.2** Gõ tiếp lệnh **Update-Database**

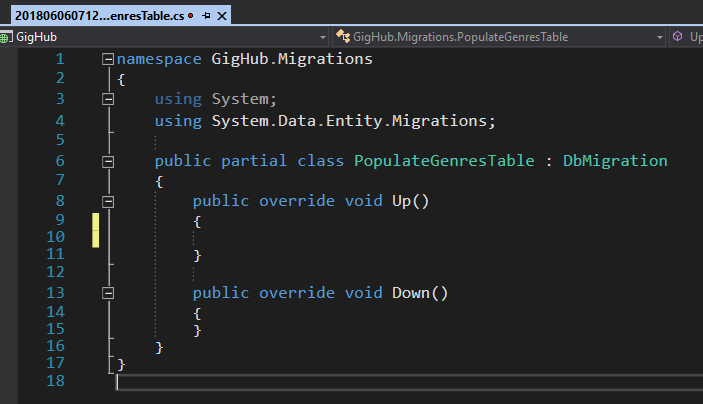
**Và giờ ta thấy kết quả đã như tà mong muốn**



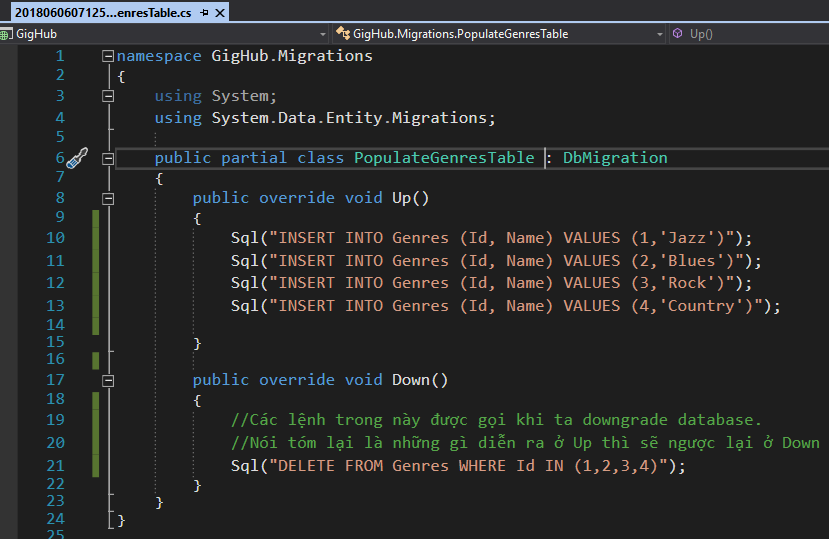
**6. Populating the database.**

Nếu ta thực thi một lệnh add-migration + ten\_gi\_do

thì sẽ được kết quả như sau:



Ta có thể thực thi các lệnh SQL như sau:



Và sau đó ta gọi lại lệnh **Update-database thì ta sẽ có kết quả như sau:**

