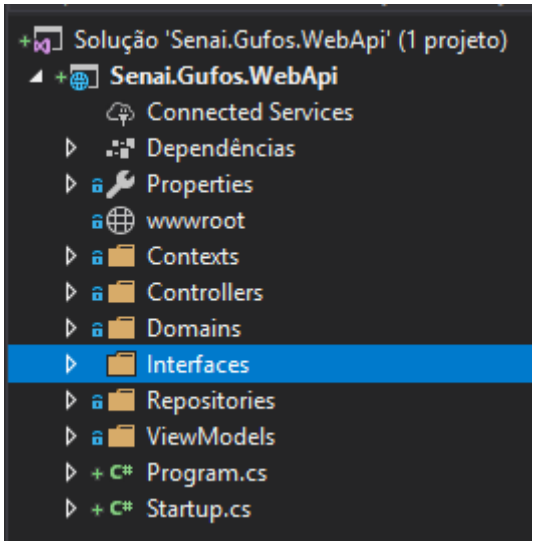
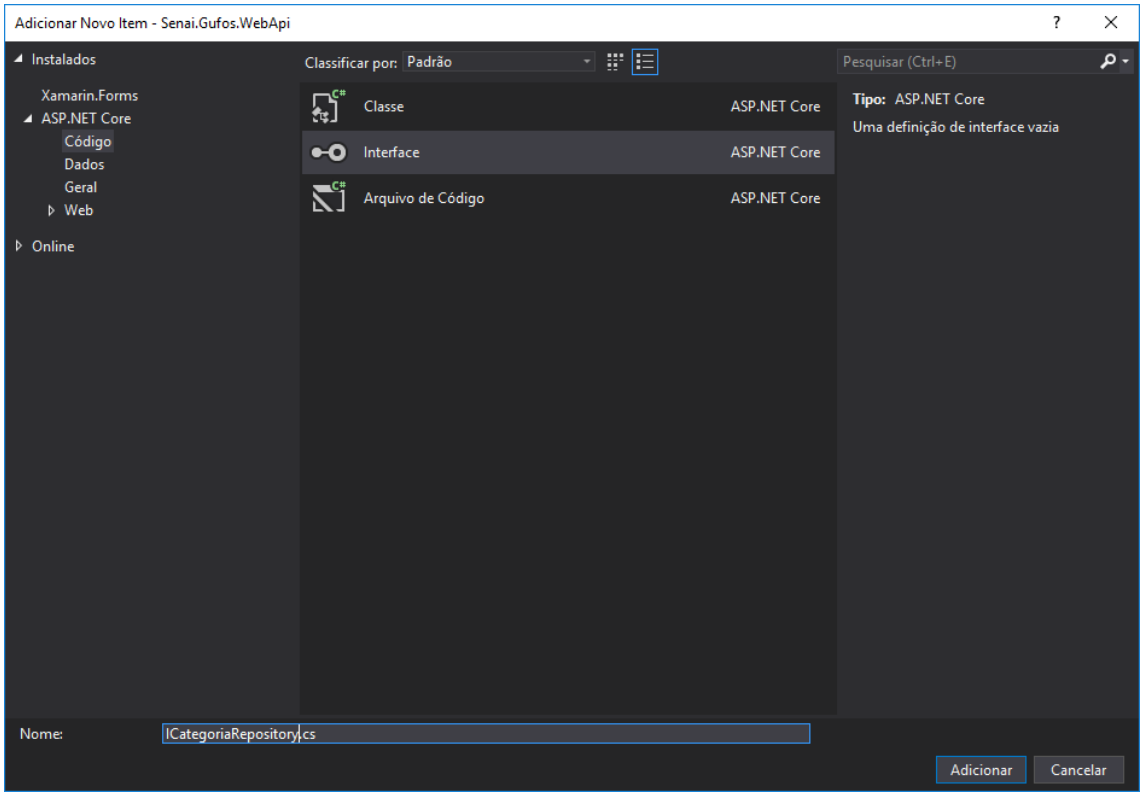
**Interface para os repositórios**

Criar uma nova pasta chamada Interfaces




Criar um arquivo chamado ICategoriaRepository.cs




Determinar as ações do que eu desejo realizar.

ICategoriaRepository.cs

```
namespace Senai.Gufos.WebApi.Interfaces
{
    public interface ICategoriaRepository
    {
        List<Categorias> Listar();

        Categorias BuscarPorId(int id);

        void Cadastrar(Categorias categoria);

        void Atualizar(Categorias categoria);

        void Deletar(int id);
    }
}
```

Realizar a alteração para que eu implemente a interface criada.

CategoriaRepository.cs

```
public class CategoriaRepository : ICategoriaRepository
{
```

CategoriasController.cs

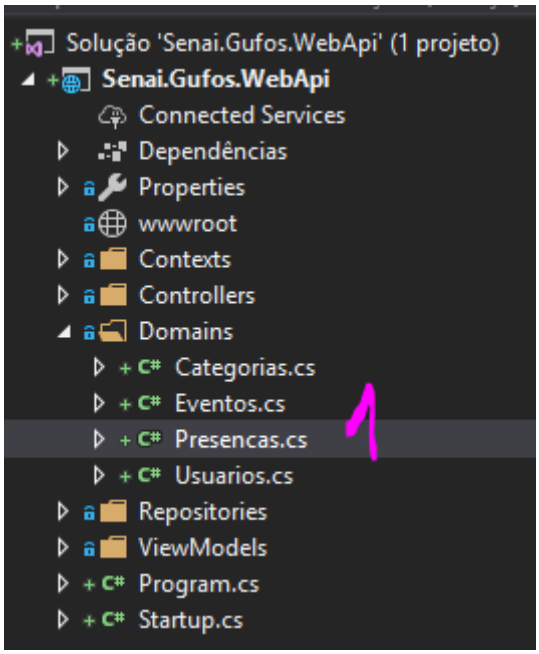
```
//CategoriaRepository CategoriaRepository = new CategoriaRepository();

private ICategoriaRepository CategoriaRepository { get; set; }

public CategoriasController()
{
    CategoriaRepository = new CategoriaRepository();
}
```

Comentar sobre as vantagens de trabalhar com interface.

**Many-To-Many**

Muitos para Muitos

Criar uma nova class na pasta domains chamada Presencas.cs




```cs
namespace Senai.Gufos.WebApi.Domains
{
    public class Presencas
    {
        public int IdUsuario { get; set; }
        public Usuarios Usuario { get; set; }

        public int IdEvento { get; set; }
        public Eventos Evento { get; set; }
    }
}
```

Alterar os usuários e os eventos para que possuam as presenças.

Usuarios.cs

```cs
public partial class Usuarios
{
    public int IdUsuario { get; set; }
    public string Nome { get; set; }
    public string Email { get; set; }
    public string Senha { get; set; }
    public string Permissao { get; set; }
    public List<Presencas> Presencas { get; set; }
}
```

Eventos.cs

```
public partial class Eventos
{
    public int IdEvento { get; set; }
    public string Titulo { get; set; }
    public string Descricao { get; set; }
    public DateTime DataEvento { get; set; }
    public bool? Ativo { get; set; }
    public string Localizacao { get; set; }
    public int? IdCategoria { get; set; }

    public Categorias IdCategoriaNavigation { get; set; }
    public List<Presencas> Presencas { get; set; }
}
```

Configurar no GufosContext a nova informação sobre Presenças.

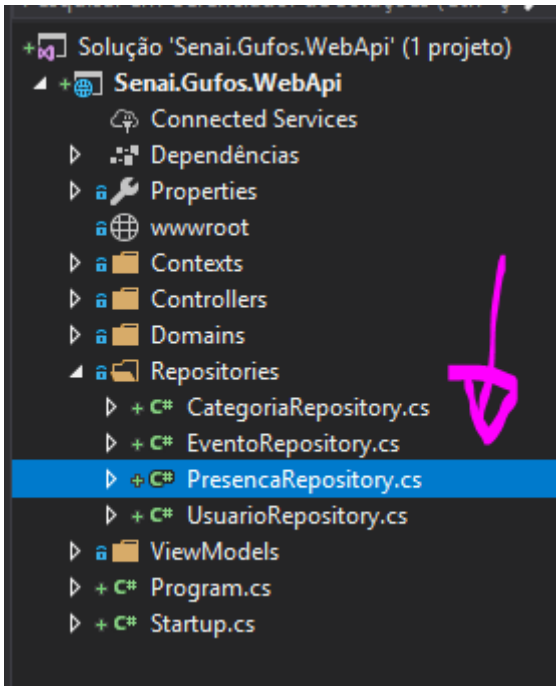
Adicionar o novo DbSet

```
public virtual DbSet<Categorias> Categorias { get; set; }
public virtual DbSet<Eventos> Eventos { get; set; }
public virtual DbSet<Usuarios> Usuarios { get; set; }

public virtual DbSet<Presencas> Presencas { get; set; }
```

Adicionar as configurações para realizar o join.

```
protected override void OnModelCreating(ModelBuilder modelBuilder)
{
    modelBuilder.Entity<Presencas>().HasKey(p => new { p.IdUsuario, p.IdEvento });

    modelBuilder.Entity<Presencas>()
    .HasOne<Usuarios>(sc => sc.Usuario)
    .WithMany(s => s.Presencas)
    .HasForeignKey(sc => sc.IdUsuario);


    modelBuilder.Entity<Presencas>()
        .HasOne<Eventos>(sc => sc.Evento)
        .WithMany(s => s.Presencas)
        .HasForeignKey(sc => sc.IdEvento);
```

Criar o Repositório




PresencaRepository.cs

```
public class PresencaRepository
{
    public List<Presencas> Listar()
    {
        using (GufosContext ctx = new GufosContext())
        {
            return ctx.Presencas.ToList();
        }
    }
}
```

Criar o controller de presenças.

```
[Route("api/[controller]")]
[Produces("application/json")]
[ApiController]
public class PresencasController : ControllerBase
{

    PresencaRepository PresencaRepository = new PresencaRepository();

    [HttpGet]
    public IActionResult Listar()
    {
        return Ok(PresencaRepository.Listar());
    }
}
```

Alterar o repository para também mostrar a lista de usuários e seus respectivos eventos.

```
public class PresencaRepository
{
    public List<Presencas> Listar()
    {
        using (GufosContext ctx = new GufosContext())
        {
            return ctx.Presencas.Include(x => x.Evento).Include(x => x.Usuario).ToList();
        }
    }
}
```

Cross-Origin Resource Sharing (Compartilhamento de recursos com origens diferentes).

ConfigureServices

```
services.AddCors(options =>
{
    options.AddPolicy("CorsPolicy", builder => builder.AllowAnyOrigin().AllowAnyMethod().AllowAnyHeader().AllowCredentials());
});
```

Configure

```
app.UseCors("CorsPolicy");
```