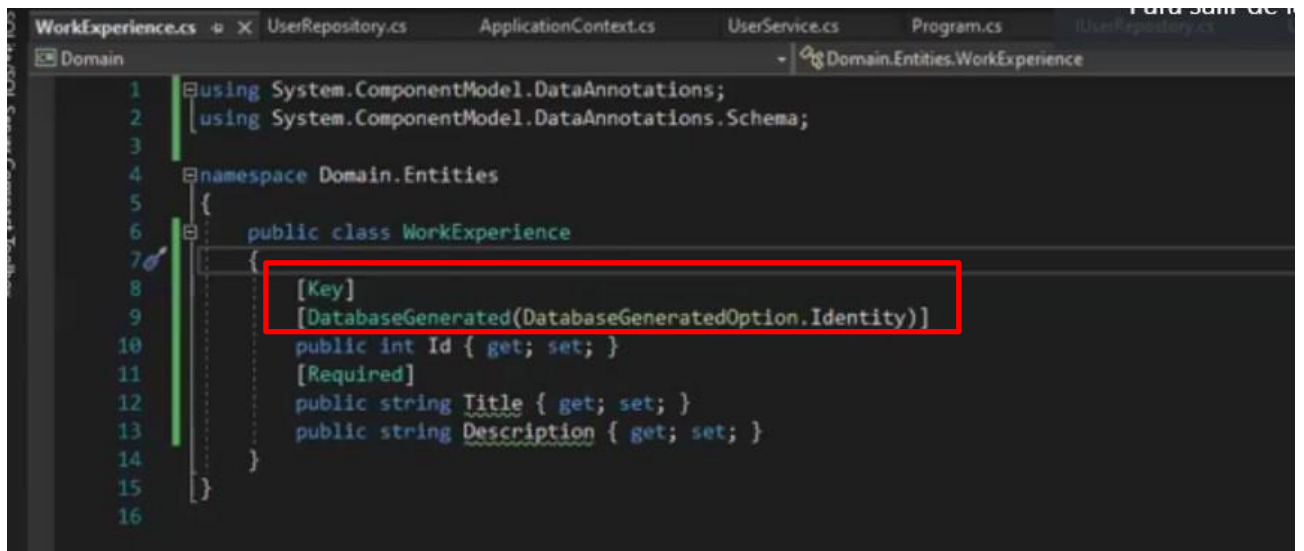


Patron repositorio

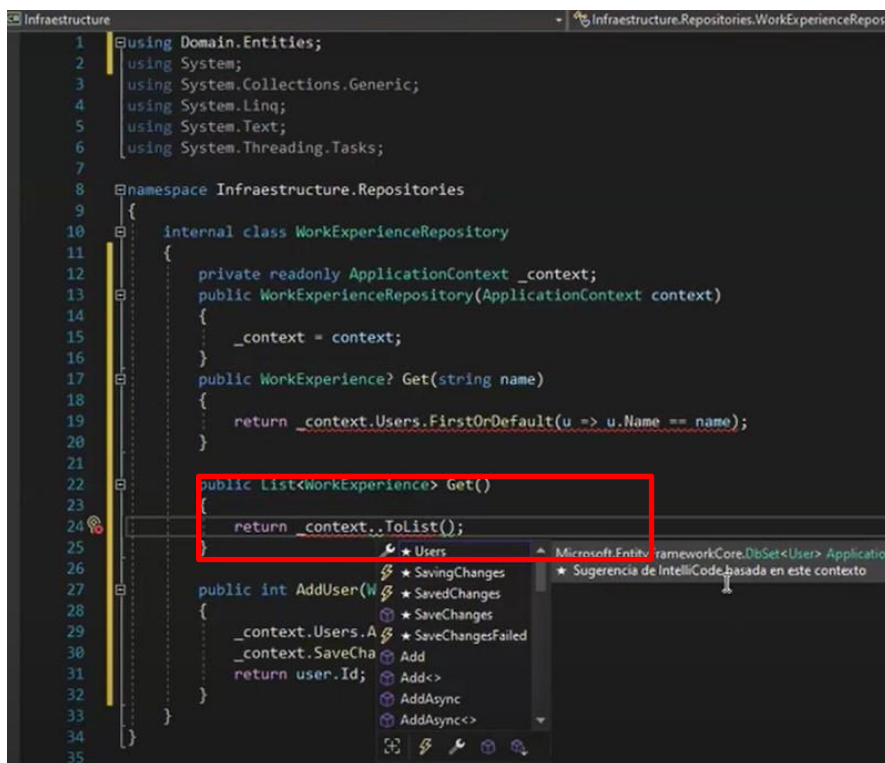


Key: decoradores para id

Databasegenerated: esto asigna un valor para la id en base a los que ya había en la db (eleva en 1 cada vez que pasa). Genera un id y lo devuelve cuando lo necesitamos

Hacemos el repositorio de work experience

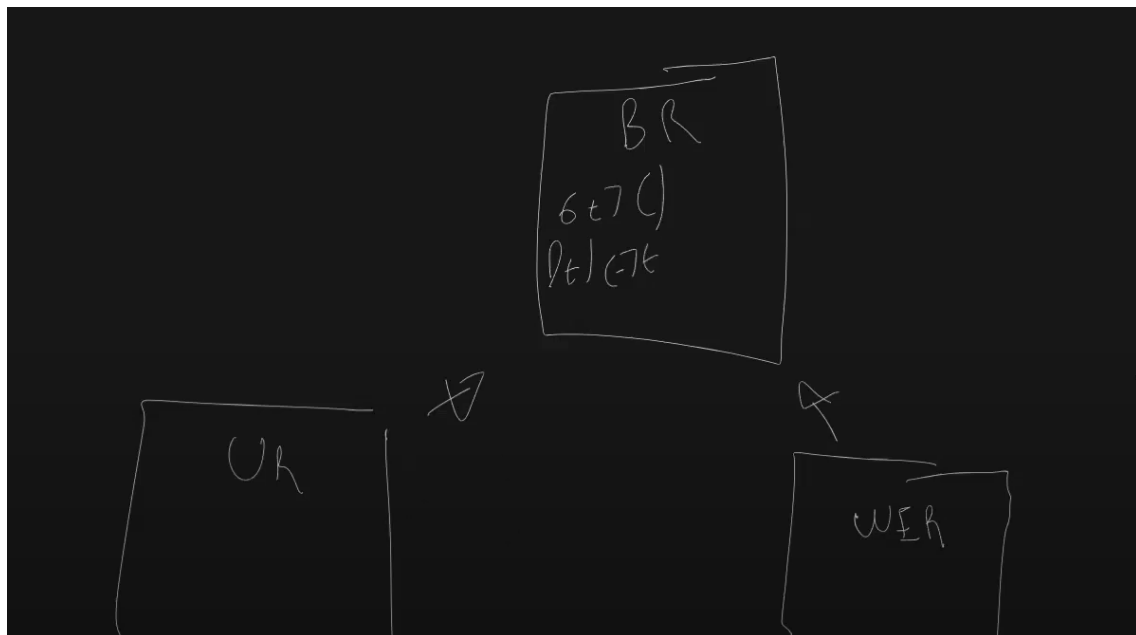
En este caso que tenemos que hacer para que tengamos disponible el conjunto de work experience y poder hacer linq?



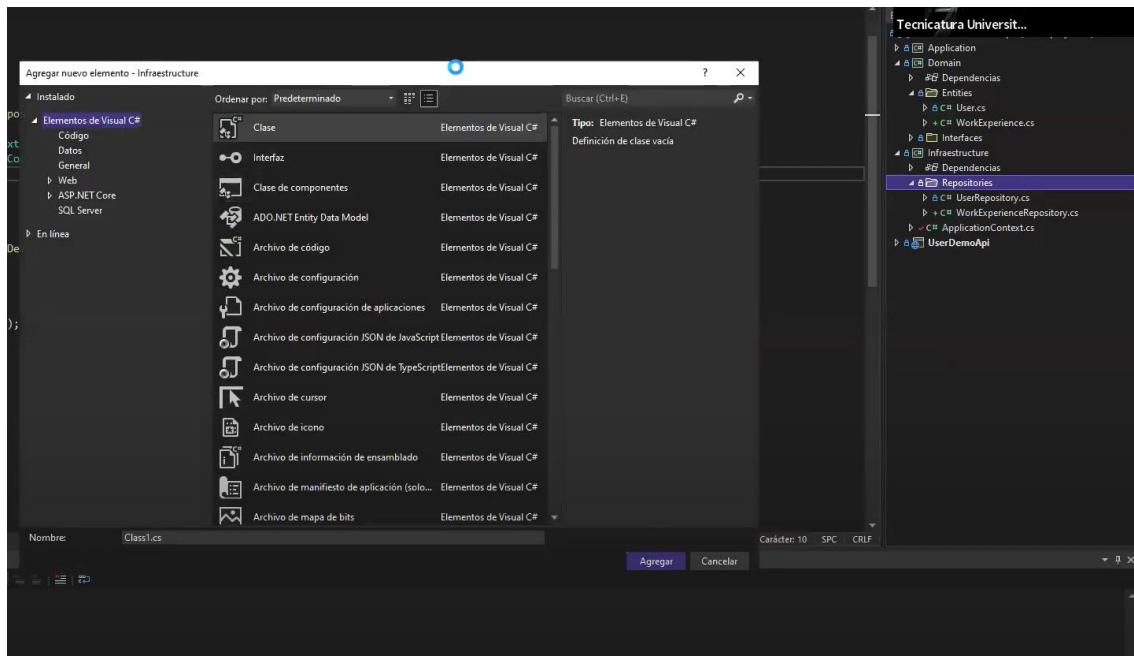
Tenes que ponerlo en context aplicación el DBSET

```
WorkExperienceRepository.cs*  WorkExperience.cs  UserRepository.cs  ApplicationContext.cs  X UserService.cs  Program.cs  IUserRepository.cs  User.cs
Infrastructure
1  using Domain.Entities;
2  using Microsoft.EntityFrameworkCore;
3
4  namespace Infrastructure
5  {
6      public class ApplicationDbContext : DbContext
7      {
8          public DbSet<User> Users { get; set; }
9
10         public DbSet<WorkExperience> WorkExperiences { get; set; }
11
12         private readonly bool isTestingEnvironment;
13
14         public ApplicationDbContext(DbContextOptions<ApplicationContext> options, bool isTestingEnvironment = false) : base(options)
15         {
16             this.isTestingEnvironment = isTestingEnvironment;
17         }
18     }
19 }
```

No se puede aplicar la herencia asi nomas pues los métodos de cada clase son distintos en userRepository por ejemplo el get te trae una lista de usuarios y el get de workexperience devuelve una lista de work experience entonces la herencia como tal no soluciona el problema de tener los métodos unificados en una clase, vamos a tener que seguir teniendo implementaciones distintas



Como se puede solucionar esto?

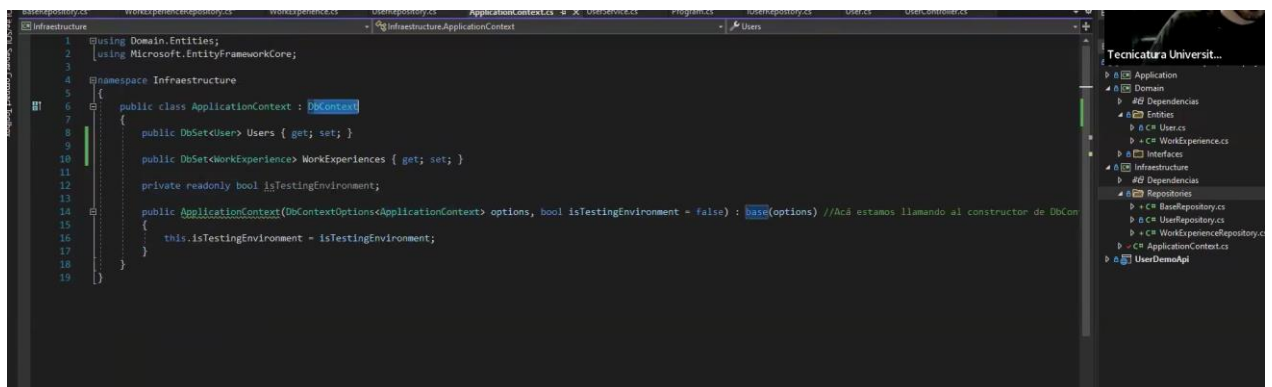


Creamos BaseRepository

Datos genéricos:

La clase como tal existe, la lista T existe, en este caso `BaseRepository<T>`, pero hasta que nadie la llama con lo que viene ahí dentro esa clase, no existe realmente.

Lo mismo pasa con las listas, si hay una lista y hasta que no pases lo que va ahí y definan la lista, la clase no existe



Db context es la clase q hereda

La necesitamos porque necesitamos db set y el dbSet de T no existe, lo que si existe es entity framework que nos disponibiliza un método que es `Set<T>` que nos devuelve el dbset si existe uno.

```
BaseRepository.cs*  X  WorkExperienceRepository.cs*  WorkExperience.cs  UserRepository.cs  ApplicationContext
Infrastructure
1  using Microsoft.EntityFrameworkCore;
2  using System;
3  using System.Collections.Generic;
4  using System.Linq;
5  using System.Text;
6  using System.Threading.Tasks;
7
8  namespace Infrastructure.Repositories
9  {
10     public class BaseRepository<T> where T : class
11     {
12         private readonly DbContext _context;
13         public BaseRepository(ApplicationContext context)
14         {
15             _context = context;
16         }
17         public T? Get(string name)
18         {
19             return _context.Set<T>().FirstOrDefault(u => u.Name == name);
20         }
21
22         public List<T> Get()
23         {
24             return _context.Set<T>().ToList();
25         }
26
27         public int AddUser(T user)
28         {
29             context.Set<T>().Add(user);
30             _context.SaveChanges();
31             return user.Id;
32         }
33     }
```

Con el set preguntas si existe un db de la entidad que me están pasando

```

using System.Text;
using System.Threading.Tasks;

namespace Infraestructure.Repositories
{
    public class BaseRepository<T> where T : class
    {
        private readonly DbContext _context;
        public BaseRepository(ApplicationDbContext context)
        {
            _context = context;
        }
        public T? Get(string name)
        {
            return _context.Set<T>().Find(id => );
        }

        public List<T> Get()
        {
            return _context.Set<T>().ToList();
        }

        public int AddUser(T user)
        {
            _context.Set<T>().Add(user);
            _context.SaveChanges();
            return user.Id;
        }
    }
}

```

Tienes que usar un TI porque sino fallaría porque si pones id sola no te va a entender porque es T entonces por eso ponemos Tid que va a ser el identificador de T y el tipo de dato que pasamos no es un entero sino Tid

```
BaseRepository.cs* x WorkExperienceRepository.cs WorkExperience.cs UserRepository.cs* Application
Infrastructure Infrastructure.Repositories.BaseRepository

1 using Microsoft.EntityFrameworkCore;
2 using System;
3 using System.Collections.Generic;
4 using System.Linq;
5 using System.Text;
6 using System.Threading.Tasks;
7
8 namespace Infrastructure.Repositories
9 {
10     public class BaseRepository<T> where T : class
11     {
12         private readonly DbContext _context;
13         public BaseRepository(DbContext context)
14         {
15             _context = context;
16         }
17         public T? Get<TId>(TId id)
18         {
19             return _context.Set<T>().Find(new object[] { id });
20         }
21
22         public List<T> Get()
23         {
24             return _context.Set<T>().ToList();
25         }
26
27         public T Add(T entity)
28         {
29             _context.Set<T>().Add(entity);
30             _context.SaveChanges();
31             return entity;
32         }
33     }
34 }
35
```

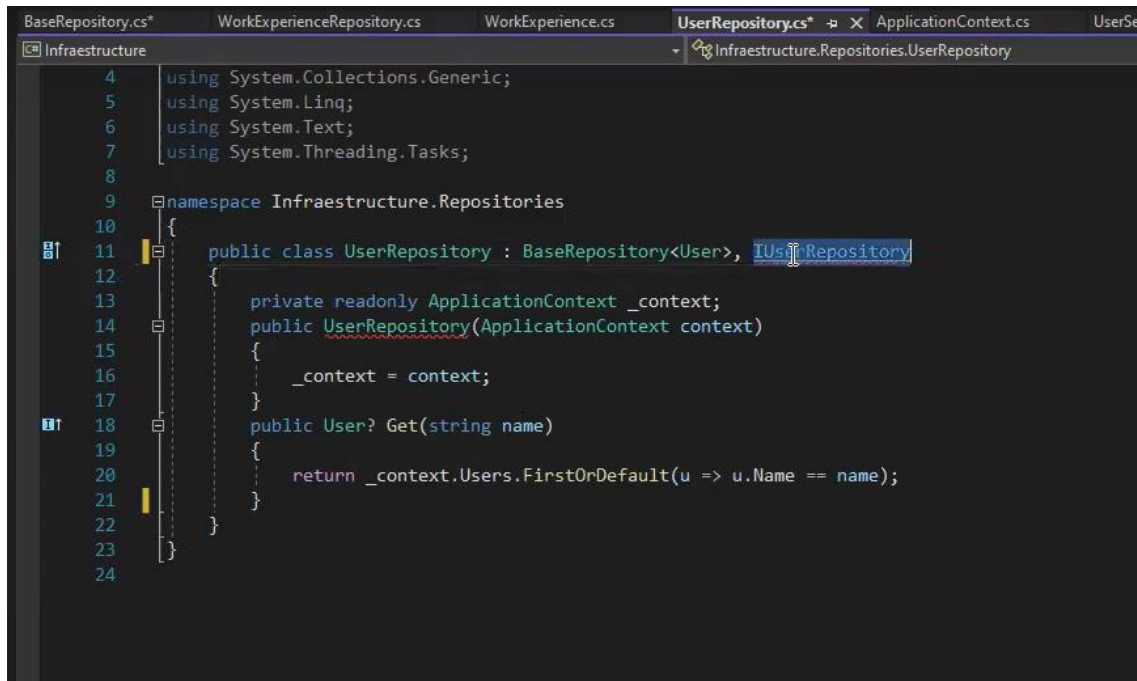


```
BaseRepository.cs  WorkExperienceRepository.cs  WorkExperience.cs  UserRepository.cs  Application
Infrastructure
4  using System.Linq;
5  using System.Text;
6  using System.Threading.Tasks;
7
8  namespace Infrastructure.Repositories
9  {
10     public class BaseRepository<T> where T : class
11     {
12         private readonly DbContext _context;
13         public BaseRepository(ApplicationDbContext context)
14         {
15             _context = context;
16         }
17         public T? Get<TId>(TId id)
18         {
19             return _context.Set<T>().Find(new object[] { id });
20         }
21
22         public List<T> Get()
23         {
24             return _context.Set<T>().ToList();
25         }
26
27         public T Add(T entity)
28         {
29             _context.Set<T>().Add(entity);
30             _context.SaveChanges();
31             return entity;
32         }
33     }
34 }
```

Ponemos que devuelva el objeto entero

```
BaseRepository.cs*  WorkExperienceRepository.cs  WorkExperience.cs  UserRepository.cs
Domain
1  using Domain.Entities;
2  using System;
3  using System.Collections.Generic;
4  using System.Linq;
5  using System.Text;
6  using System.Threading.Tasks;
7
8  namespace Domain.Interfaces
9  {
10     public interface IUserRepository
11     {
12         User? Get(string name);
13     }
14 }
15
```

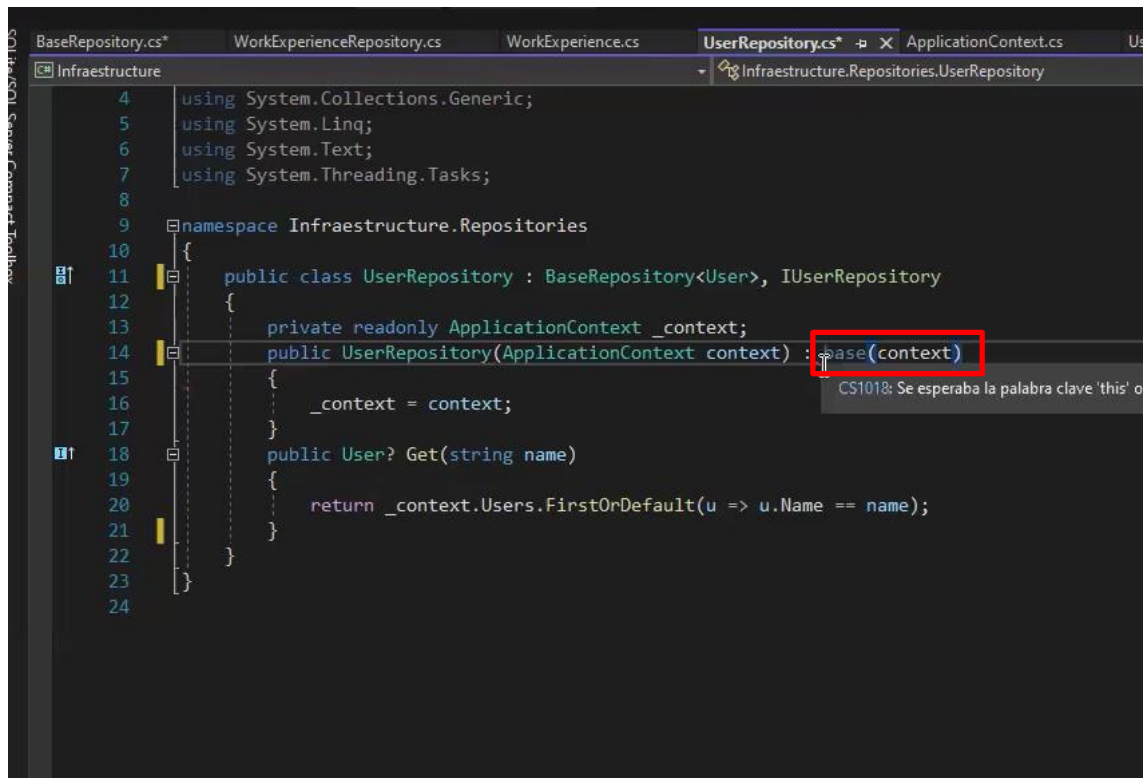
Aca usamos base repository user



```
BaseRepository.cs*  WorkExperienceRepository.cs  WorkExperience.cs  UserRepository.cs*  ApplicationContext.cs  UserSe
Infrastructure
4  using System.Collections.Generic;
5  using System.Linq;
6  using System.Text;
7  using System.Threading.Tasks;
8
9  namespace Infrastructure.Repositories
10 {
11     public class UserRepository : BaseRepository<User>, IUserRepository
12     {
13         private readonly ApplicationContext _context;
14         public UserRepository(ApplicationContext context)
15         {
16             _context = context;
17         }
18         public User? Get(string name)
19         {
20             return _context.Users.FirstOrDefault(u => u.Name == name);
21         }
22     }
23 }
24
```

Aquí debemos usar el base lo que haces es invocar la clase padre este base es palabra reservada, con esto que estas haciendo es invocar al constructor de la clase padre

Si vos tenes un método que tiene el nombre de la clase y paréntesis se llama constructor



```
BaseRepository.cs*  WorkExperienceRepository.cs  WorkExperience.cs  UserRepository.cs*  ApplicationContext.cs  Us
Infrastructure
4  using System.Collections.Generic;
5  using System.Linq;
6  using System.Text;
7  using System.Threading.Tasks;
8
9  namespace Infrastructure.Repositories
10 {
11     public class UserRepository : BaseRepository<User>, IUserRepository
12     {
13         private readonly ApplicationContext _context;
14         public UserRepository(ApplicationContext context) : base(context)
15         {
16             _context = context;
17         }
18         public User? Get(string name)
19         {
20             return _context.Users.FirstOrDefault(u => u.Name == name);
21         }
22     }
23 }
24
```

CS1018: Se esperaba la palabra clave 'this' o

- 1-Hacer un endpoint de autenticación - Chequear identidad del usuario y devolver el JWT
- 2-Configuraciones
- 3-Implementarlo en los controladores y endpoints que querramos.

Autenticación:

Quien es el usuario? Si es quien dice ser?

Objetivo de autenticación con JWT

- 1-Verificar identidad y credenciales la primera vez y conceder la llave.
- 2-Usar la api con esta JWT

```
CredentialsRequest.cs* AuthenticationController.cs* UserController.cs User.cs
UserDemoApi.Controllers.AuthenticationController Authenticate(CredentialsRequest credentials)
1 using Application.Dtos;
2 using Microsoft.AspNetCore.Http;
3 using Microsoft.AspNetCore.Mvc;
4
5 namespace UserDemoApi.Controllers
6 {
7     [Route("api/[controller]")]
8     [ApiController]
9     public class AuthenticationController : ControllerBase
10    {
11        [HttpPost]
12        public IActionResult Authenticate([FromBody] CredentialsRequest credentials)
13        {
14
15            return Ok("jwt");
16        }
17    }
18 }
```

```
0 references
public UserModel? CheckCredentials(CredentialsRequest credentials)
{
    User? user = Get(credentials.Username);
    if (user.Password == credentials.Password)
        return new UserModel()
        {
            Email = user.Email,
            Id = user.Id,
            Name = user.Name
        };
}
```

```
appsettings.json* CredentialsRequest.cs* AuthenticationController.cs* UserModel.cs
UserDemoApi.Controllers.AuthenticationController Authenticate(CredentialsRequest credentials)
var saltEncrypted = new SigningCredentials(securityPassword, SecurityAlgo);

var claimsForToken = new List<Claim>();
claimsForToken.Add(new Claim("sub", userLogged.Id.ToString()));
claimsForToken.Add(new Claim("role", userLogged.Role));

var jwtSecurityToken = new JwtSecurityToken( //agregar using System.IdentityModel.Tokens.Jwt
    _configuration["Authentication:Issuer"],
    _configuration["Authentication:Audience"],
    claimsForToken,
    DateTime.UtcNow,
    DateTime.UtcNow.AddHours(1),
    saltEncrypted);

var tokenToReturn = new JwtSecurityTokenHandler() //Pasamos el token a string
    .WriteToken(jwtSecurityToken);
```

```
UserDemoApi.Controllers.AuthenticationController Authenticate(CredentialsRequest credentials)
var claimsForToken = new List<Claim>();
claimsForToken.Add(new Claim("sub", userLogged.Id.ToString()));
claimsForToken.Add(new Claim("role", userLogged.Role));

var jwtSecurityToken = new JwtSecurityToken(
    _configuration["Authentication:Issuer"],
    _configuration["Authentication:Audience"],
    claimsForToken,
    DateTime.UtcNow,
    DateTime.UtcNow.AddHours(1),
    signature);

var tokenToReturn = new JwtSecurityTokenHandler() //Pasamos el token a
    .WriteToken(jwtSecurityToken);

}
```