Tarea #4 - (4%)

Fecha de entrega Viernes 26 de Julio

Filtros, Extensiones y Excepciones

> *Terminar el escenario para crear los filtros de autenticacion y autorizacion asi como un filtro de excepciones para que atrape todas las excepciones no manejadas de la aplicacion (investigacion). Usuarios no registrados no podran accesar los controllers ni las acciones.*

Pasos

Parte 1 - Completar Escenario (1%)

1. Dentro de la libreria de Service cree un servicio donde exista un metodo que verifique si el usuario esta logueado. IsUserLoggedIn puede recibir dentro de su firma un entero o un int.
2. Crear el repositorio de UserRepository un metodo Exists que verifique si el usuario existe en la base de datos.
3. Debe implementar / inyectar el servicio de IUserRepository como dependencia en el SecurityService. Tome en cuenta que estos (todos los servicios a llamar / inyectar) deben estar implementados en el Program.cs como servicios "Scoped".
4. Posteriormente debe crear el AuthorizationFilter y agregar el codigo para que llame a los metodos del servicio de SecurityService para autenticar al usuario. Tome en cuenta de que el IAuthorizationFilter es un contrato propio de NET Core por lo que no puede inyectarse ninguna otra dependencia. Debe llamar la dependencia utiliando el `context.HttpContext.RequestServices.GetService<ISecurityService>();`
5. Debe agregar el codigo para accesar mediante autorizacion a todos los controllers menos al de Home y al metodo de Privacy en Home debe autorizarlo tambien.

Parte 2 - Filtro de Exceptiones Globales (1.5%)

1. Mediante investigacion debe buscar la forma de implementar un Filter para agarrar todas las excepciones no manejadas en la aplicacion, con el fin de evitar que la aplicacion dispare un error y redireccione a la pantalla de error por defecto de ASP.NET Core
2. Debe crear el filtro y agregar un metodo que atrape la excepcion, y evite un manejo no-controlado.
3. Debe hacer que este filtro corra para cualquier o todos los controllers.
4. Probablemente va a necesitar agregar algo al Program.cs para que sea inicializado a nivel del `middleware` asi como ejecutado a nivel de runtime.

Parte 3 - Metodo Extension y Validacion (1.5%)

1. Mediante investigacion debe buscar crear un metodo de extension (`Extension Methods`) Extension Methods MSDN este metodo de extension debe validar si el usuario esta autorizado.
2. Para autorizar el usuario debe crear una clase User o ApplicationUser y esta debe tener una propiedad `public IsAuthorized {get; set; }` que debe ser asignado en el repository cuando se valida si el usuario existe. (Para este paso debe usar la conexion a una base de datos mediante EF).

3. Para accesar el Usuario debe crear una `interfaz` `IApplicationUser` y registrarla dentro del `Program.cs` tal como se registran los demas servicios y contratos. Esta interfaz debe definir las propiedades del usuario incluyendo IsAuthorized. En la clase de implementacion, o sea, el ApplicationUser debe implementar el contrato `ApplicationUser : IApplicationUser`.
4. En la base de datos debe guardar el valor de IsAuthorized como una propiedad bool. Esta debe ser leida cada vez que se quiera validar que el usuario esta autorizado.
5. Debe crear un controller Base que se llame ControllerBase esta debe ser una clase `abstract` Abstract Classes y debe implementar un metodo que inyecte dentro del constructor el `IApplicationUser` asi como el `ISecurityService` y asignarlos a dos miembros privados `readonly` que sean asignados unicamente en el `ctor` (constructor).
6. Esta clase abstracta debe crear un metodo que se llame: `GetApplicationUser(string id /*Email*/)`, puede usar el email tambien en lugar del id. Debera llamar al servicio para que valide el usuario y asignar a la propiedad del ApplicationUser llamada IsAuthorized el valor del resultado del metodo de GetApplicationUser de manera que si el usuario existe entonces el valor es true e.g. `IsAuthorized= GetApplicationUser("1")!=null`, como no existe proceso de logueo en la app debe simular/quemar el id por ahora.
7. Volviendo al punto #1, el metodo de extension debe devolver interpolar el nombre y el apellido o el nombre y el email del usuario (lo que prefiera) e.g. `$"{appUser.Name} - {appUser.LastName}"` este metodo de extension puede ser utilizado en el FE para mostrar el usuario logueado (simulado).