Y falla, menos mal, porque no hemos escrito ni una palabra!!!!

Cambiar antes el appsettings.json

Colocar el appsettings.json: AUT02_05Users

A database operation failed while processing the request.

SqlException: Cannot open database "aspnet-AUT02_05_IdentityAutoMartinezInma-887AFF2F-2FC9-47DB-9106-9AB43F0EBF8A" requested by the login. The login failed. Login failed for user 'RTFG6GU---INMA\inmav'.

Applying existing migrations may resolve this issue

There are migrations that have not been applied to the following database(s):

ApplicationDbContext

- 00000000000000_CreateIdentitySchema

Apply Migrations

In Visual Studio, you can use the Package Manager Console to apply pending migrations to the database:

```
PM> Update-Database
```

Alternatively, you can apply pending migrations from a command prompt at your project directory:

```
> dotnet ef database update
```

Veamos ahora nuestro program.cs

- Se nos han añadido, la conexión que vamos a usar.
- Otro servicio para la confirmación del email (set to false, si queremos)
- Además de dos sentencias más.

```
app.UseAuthentication();
app.UseAuthorization();
```

```
builder.Services.AddDefaultIdentity<AppAutUser>(options =>
options.SignIn.RequireConfirmedAccount = false)
.AddEntityFrameworkStores<AppAutContext>();
```

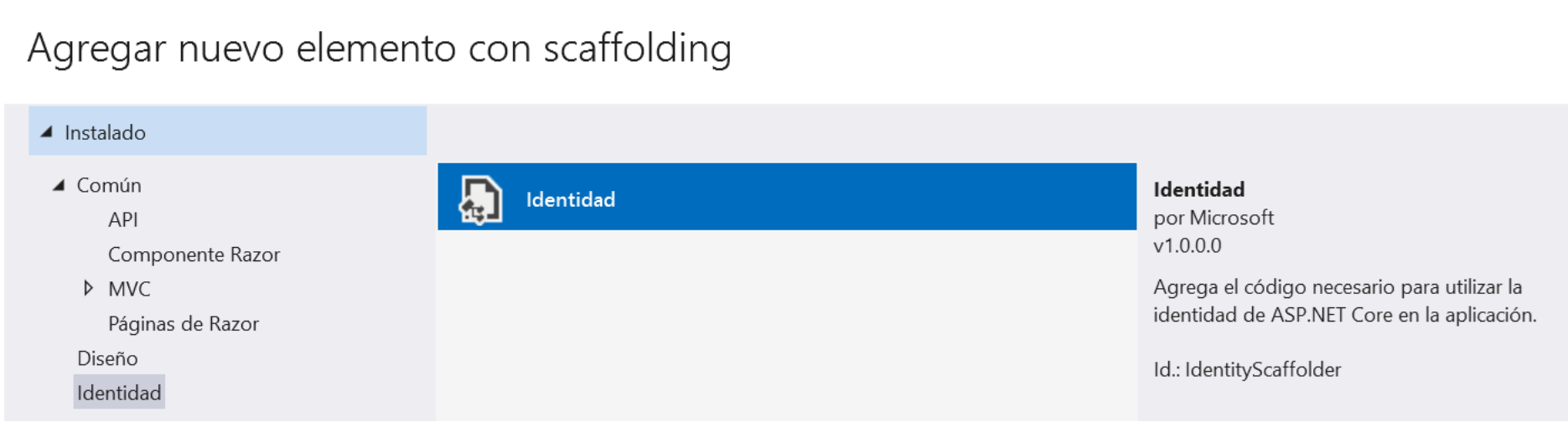
Y en las rutas una más:

```
app.MapRazorPages();
```

Y ahora probemos a añadir un usuario nuevo.

ASP .NET Identity en un proyecto existente

- Lo primero asegurarnos de que tenemos el paquete:
  - Web.CodeGeneration.Design
- En Explorador de soluciones, haga clic con el botón derecho en el proyecto >Agregar>nuevo elemento con scaffolding.
- En el panel izquierdo del cuadro de diálogo Agregar scaffolding , seleccione Identity>Agregar.



Agregar Identidad

Seleccione una página de diseño existente o especifique una nueva:

~/Views/Shared/_Layout.cshtml ...

(Dejar en blanco si se define en un archivo _viewstart de Razor)

☐ Reemplazar todos los archivos

Elija los archivos que quiere reemplazar

☐ Account\StatusMessage	☐ Account\AccessDenied	☐ Account\ConfirmEmail
☐ Account\ConfirmEmailChange	☐ Account\ExternalLogin	☐ Account\ForgotPassword
☐ Account\ForgotPasswordConfirmation	☐ Account\Lockout	☑ Account\Login
☐ Account\LoginWith2fa	☐ Account\LoginWithRecoveryCode	☑ Account\Logout
☐ Account\Manage\Layout	☐ Account\Manage\ManageNav	☐ Account\Manage\StatusMessage
☐ Account\Manage\ChangePassword	☐ Account\Manage\DeletePersonalData	☐ Account\Manage\Disable2fa
☐ Account\Manage\DownloadPersonalData	☐ Account\Manage\Email	☐ Account\Manage\EnableAuthenticator
☐ Account\Manage\ExternalLogins	☐ Account\Manage\GenerateRecoveryCodes	☐ Account\Manage\Index
☐ Account\Manage\PersonalData	☐ Account\Manage\ResetAuthenticator	☐ Account\Manage\SetPassword
☐ Account\Manage\ShowRecoveryCodes	☐ Account\Manage\TwoFactorAuthentication	☑ Account\Register
☐ Account\RegisterConfirmation	☐ Account\ResendEmailConfirmation	☐ Account\ResetPassword
☐ Account\ResetPasswordConfirmation		

Clase de contexto de datos: AppAutContext (AppAutContext) +

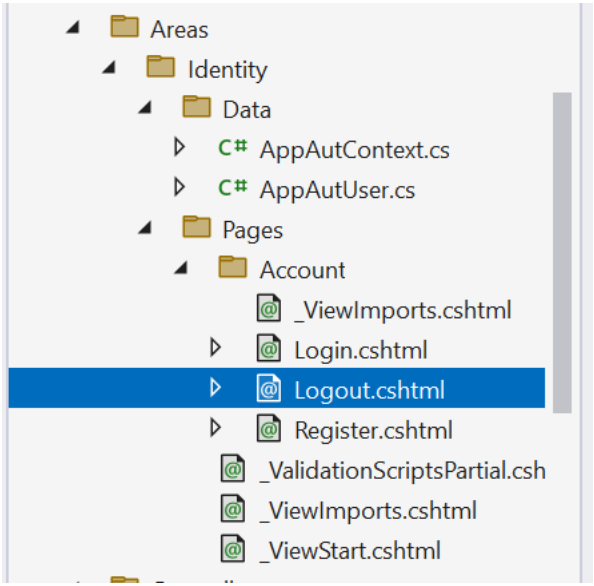
☐ Usar SQLite en lugar de SQL Server

Clase de usuario: AppAutUser (AppAutUser) +

Agregar Cancelar

Paso a paso

1)Comprobamos lo que se ha creado nuevo. Areas/Identity….



Además nos ha aparecido una vista parcial

Que curioso!!!

Ahora queremos usarla. Para ello en nuestra barra de navegación añadimos que queremos usarla en nuestro layout:

<partial name="_LoginPartial.cshtml"/>

PROBAMOS

Como no funciona tenemos que añadir que queremos usar Razor en nuestro program.cs

SEGUIMOS

Y ahora usamos la base de datos de usuarios y contraseñas que habíamos creado anteriormente. Para ello, tendremos que añadir la cadena de conexión de la base de datos que necesitamos.

Nombre base de datos

```
"AppAutContextConnection": "Server=(localdb)\\mssqllocaldb;Database=AUT02_05Users;Trusted_Connection=True;
MultipleActiveResultSets=true"
```

Y ahora, queremos traer nuestras tablas de la base de datos a entidades, como hicimos anteriormente con la conversión de tablas para usarlas a través del contexto.

Add-Migration InitApplicationUser -Context AppAutContext

2do esto

Preparamos un link para acceder a coches.

Añadimos nuestros Roles en el siguiente capítulo.

1ero esto

```
builder.Services.AddDbContext<AppAutContext>(options =>
{
    options.UseSqlServer(builder.Configuration.GetConnectionString("
AppAutContextConnection"));
});
```

Autorización Simple - En vistas

•Usando el método “SignInManager.IsSignedIn(User)” en las vistas de la siguiente manera:

•Se pueden usar estructuras if-else para mostrar elementos distintos en función de que esté autenticado o no el usuario

```
@if (SignInManager.IsSignedIn(User))
{
    <!-- Elementos HTML que se quiera mostrar
        a usuarios autenticados
    -->
}
```

Autorización Simple – En controladores

- Para ello usaremos el atributo [Authorize] sobre la clase del controlador, para restringir todas sus acciones, o sobre una acción en concreto.
- Con ello restringimos el acceso al controlador u acciones a usuarios autenticados.

```
[Authorize]
// GET: Shippers
3 referencias
public async Task<IActionResult> Index()
{
        return View(await _context.Shippers.ToListAsync());
}
```

Para Añadir Roles a nuestra base de datos

En este momento una vez que tenemos las tablas de la base de datos.

- Borramos la carpeta de migraciones
- Añadimos una nueva migración
- Add-Migration SeedRoles -Context AppAutContext
- Cambiamos el fichero creado por el que está en nuestro EVAGD.
- Actualizamos base de datos.

- Añadimos en el program.cs — .AddRoles<IdentityRole>(), en el servicio de Identity.
- Comprobamos que tenemos varios roles y usuarios nuevos.

Especificando roles en nuestra aplicación

Aplicamos roles a nuestro aplicación.

- Ejemplo: Para crear un coche, tendremos que ser “Manager”
- [Authorize(Roles="Manager")]
- Nota: Si todo ha ido bien, el usuario test3, es manager y administrador.
- Nota: El test2 es de tipo User.

- Ejercicio:

- Solo los usuarios que sean Admin pueden Editar y Borrar los discos.
- Solo los usuarios autenticados como Users pueden ver los detalles.