

2024



**IES**  
**CO**  
**MER**  
**CIO**

MULWEB 2

## INDICE

1. Pruebas Sprint 2 .....	2
2. Pruebas Sprint 3 .....	5
3. Pruebas Sprint 4 .....	7

## 1. Pruebas Sprint 2

### Pruebas unitarias:

Primero hemos usado caja negra para diseñar las pruebas y después las hemos implementado en pruebas unitarias usando la clase de MSTEST (.NET) en el proyecto.

### Pruebas de integración:

- **Integración no incremental:** Cada uno hacía su parte y la probaba, luego lo juntamos todo y probamos todo a la vez de nuevo.

### Pruebas de validación:

- **Prueba alfa:** Iván le pidió a su padre que probase su aplicación y no se le dificultó ninguna parte.
- **Prueba beta:** En este caso, cada uno probaba la parte del otro y luego nos decíamos si había fallos o si había alguna recomendación.

### Pruebas de uso de recursos:

Como tal, la aplicación no es tan grande como para hacer pruebas de uso de recursos, pero la aplicación se ha probado en distintos dispositivos con diferentes capacidades y en todos ha rendido correctamente.

### Pruebas del sistema:

- **Pruebas de recuperación:** Hemos probado varias veces a hacer operaciones incorrectas sobre la base de datos o el propio programa y todos los casos han sido controlados para que la ejecución siga.
- **Prueba de seguridad:** Las pruebas que podríamos hacer aquí son comprobar si en la cadena de conexión a la base de datos salen datos confidenciales como el usuario y la contraseña, ya que esto le daría acceso a alguien a la base de datos.
- **Prueba de resistencia:** Como tal, la aplicación aún no necesita esto porque hace pocas cosas, pero hemos comprobado que cuando se listan las actividades (que son muchas consultas), no hay sobrecarga, incluso si hay muchas actividades.

**Pruebas Unitarias - Caja Negra:**

Método de Prueba	Entrada	Proceso	Resultado Esperado
comprobarLoginCorrecto	Email: pollino@gmail.com Contraseña: Root*root2	Llamada al método comprobarLogin con las credenciales proporcionadas.	El login es correcto, se devuelve un UsuarioDTO no nulo.
comprobarLoginIncorrecto	Email: gabriel@gmail.com Contraseña: a	Llamada al método comprobarLogin con las credenciales incorrectas.	El login es incorrecto, se devuelve null.
comprobarRegistroCorrecto	UsuarioDTO con DNI: 18372831b, Nombre: Gabriel, Apellido: Luezas, Email: unai@gmail.com, Contraseña: Root2/root.	Llamada al método altaUsuario para registrar un nuevo usuario.	Mensaje: "Usuario añadido con éxito".
comprobarRegistroIncorrectoEmail	UsuarioDTO con el mismo Email: unai@gmail.com y un DNI diferente.	Llamada al método altaUsuario para registrar un nuevo usuario con un email ya registrado.	Mensaje: "Ya hay un usuario registrado con ese email".
comprobarRegistroIncorrectoDNI	UsuarioDTO con el mismo DNI: 18372831b y un Email diferente.	Llamada al método altaUsuario para registrar un nuevo usuario con un DNI ya registrado.	Mensaje: "Ya hay un usuario registrado con ese DNI".

comprobarRegistroMonitorCorrecto	UsuarioDTO con DNI: 18079231C, Nombre: Test, Apellido: Monitor, Email: testMonitor@gmail.com, Contraseña: Root2/root.	Llamada al método altaMonitor para registrar un nuevo monitor.	Mensaje: "Monitor insertado correctamente".
comprobarRegistroIncorrectoMonitorEmail	UsuarioDTO con el mismo Email: testMonitor@gmail.com y un DNI diferente.	Llamada al método altaMonitor para registrar un nuevo monitor con un email ya registrado.	Mensaje: "Ya hay un monitor registrado con ese email".
comprobarRegistroIncorrectoMonitorDNI	UsuarioDTO con el mismo DNI: 18079231C y un Email diferente.	Llamada al método altaMonitor para registrar un nuevo monitor con un DNI ya registrado.	Mensaje: "Ya hay un monitor registrado con ese DNI".
comprobarRegistroActividadCorrecto	ActividadDTO con ID: 10, Nombre: Prueba, Descripción: Prueba, Monitor: 18079231C.	Llamada al método RegistrarActividad para registrar una nueva actividad.	Mensaje: "Actividad insertada con éxito".
comprobarRegistroActividadIncorrecto	ActividadDTO con el mismo ID: 10, Nombre: Prueba, Descripción: Prueba, Monitor: 18079231C.	Llamada al método RegistrarActividad para registrar una nueva actividad con un ID repetido.	Mensaje: "Error al guardar la actividad: An error occurred while updating the entries.".

## 2. Pruebas Sprint 3

### Pruebas unitarias:

Primero hemos usado caja negra para diseñar las pruebas y después las hemos implementado en pruebas unitarias usando la clase de MSTEST (.NET) en el proyecto.

### Pruebas de integración:

- **Integración no incremental:** Cada uno hacía su parte y la probaba, luego lo juntamos todo y probamos todo a la vez de nuevo.

### Pruebas de validación:

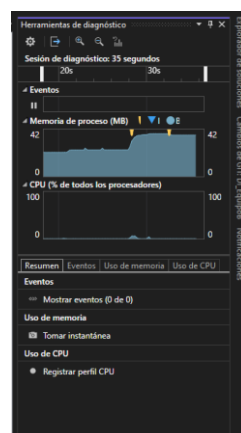
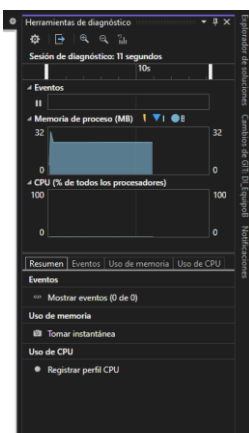
- **Prueba alfa:** Gabriel le pidió a su hermana que probase su aplicación y supo usar todas las funcionalidades de forma intuitiva, pero encontró un bug a la hora de mostrar las actividades del usuario y lo solucionamos.
- **Prueba beta:** En este caso, cada uno probaba la parte del otro y luego nos decíamos si había fallos o si había alguna recomendación.

### Pruebas de regresión:

En este sprint 3, se realizó un cambio en el DTO de actividades debido a modificaciones en la base de datos. Las pruebas previas utilizaban un constructor que ya no está disponible, por lo que hemos ajustado y corregido las pruebas para adaptarlas al nuevo DTO.

### Pruebas de uso de recursos:

Hemos comprobado como cambia al ejecutar la aplicación, que gasta pocos recursos, pero cuando listas actividades los recursos aumentan, pero siguen siendo pocos recursos los utilizados por la aplicación. Incluso cuando te apuntas a una actividad sube un poco mas pero siguen siendo pocos recursos



#### Pruebas del sistema:

- **Pruebas de recuperación:** Hemos probado varias veces a hacer operaciones incorrectas sobre la base de datos o el propio programa y todos los casos han sido controlados para que la ejecución siga.
- **Prueba de seguridad:** Las pruebas que podríamos hacer aquí son comprobar si en la cadena de conexión a la base de datos salen datos confidenciales como el usuario y la contraseña, ya que esto le daría acceso a alguien a la base de datos.
- **Prueba de resistencia:** Como tal, la aplicación aún no necesita esto porque hace pocas cosas, pero hemos comprobado que cuando se listan las actividades (que son muchas consultas), no hay sobrecarga, incluso si hay muchas actividades.

#### CAJA NEGRA

Caso de prueba	Entrada	Salida Esperada	Descripción
Registro exitoso	UsuarioActividadDTO("98939393C", 2)	true	Usuario válido y actividad válida. El registro debe ser exitoso.
Registro fallido (usuario ya inscrito en actividad)	UsuarioActividadDTO("98939393C", 2)	false	Usuario ya está inscrito en la actividad.
Registro fallido (actividad no existe)	UsuarioActividadDTO("98939393C", 100)	false	La actividad no existe. Violación de clave foránea.
Registro fallido (usuario no existe)	UsuarioActividadDTO("98939393V", 2)	false	El usuario no existe. Violación de clave foránea

### 3. Pruebas Sprint 4

#### CASOS DE USO

Código	Entrada	Salida Esperada	Descripción
CP2.1.1	Desea desapuntarse de una actividad válida.	true	Usuario válido y actividad válida. La acción de desapuntarse debe ser exitosa.
CP2.1.2	Desea desapuntarse de una actividad con un usuario no registrado.	false	El usuario no existe. La acción de desapuntarse debe fallar.
CP2.1.3	Desea desapuntarse de una actividad inexistente.	false	La actividad no existe. La acción de desapuntarse debe fallar.
CP2.2.1	Desea modificar sus datos de usuario.	"Usuario modificado con éxito"	Usuario válido con datos modificados correctamente. La edición debe ser exitosa.
CP2.2.2	Desea modificar los datos de un usuario inexistente.	"No existe el usuario"	El usuario no está registrado. La edición debe fallar.
CP2.3.1	Desea valorar una actividad en la que está registrado.	true	Usuario válido, actividad válida, y valoración dentro de rango aceptable. La valoración debe ser exitosa.
CP2.3.2	Desea valorar una actividad con un usuario no registrado.	false	El usuario no está registrado. La acción de valorar debe fallar.
CP2.3.3	Desea valorar una actividad inexistente.	false	La actividad no existe. La acción de valorar debe fallar.
CP2.3.4	Desea valorar una actividad con una puntuación inválida.	false ↓	La valoración está fuera del rango permitido. La acción de valorar debe fallar.



## CAJA NEGRA

Caso de prueba	Entrada	Salida Esperada
CP2.1.1 - Desapuntarse exitosamente	<code>UsuarioActividadDTO("99994444C", 6)</code>	<code>true</code>
CP2.1.2 - Desapuntarse fallido (usuario no existe)	<code>UsuarioActividadDTO("99994444B", 10)</code>	<code>false</code>
CP2.1.3 - Desapuntarse fallido (actividad no existe)	<code>UsuarioActividadDTO("99994444C", 100)</code>	<code>false</code>
CP2.2.1 - Editar usuario exitosamente	<code>UsuarioDTO("18079239L", Modificaciones)</code>	<code>"Usuario modificado con éxito"</code>
CP2.2.2 - Editar usuario fallido (usuario no existe)	<code>UsuarioDTO("18079239H", Modificaciones)</code>	<code>"No existe el usuario"</code>
CP2.3.1 - Valorar actividad exitosamente	<code>(6, "99994444C", 5)</code>	<code>true</code>
CP2.3.2 - Valorar actividad fallido (usuario no existe)	<code>(6, "99991444C", 5)</code>	<code>false</code>
CP2.3.3 - Valorar actividad fallido (actividad no existe)	<code>(30, "99994444C", 5)</code>	<code>false</code>
CP2.3.4 - Valorar actividad fallido (valoración inválida)	<code>(6, "99994444C", 30)</code>	<code>false</code>

## RESULTADO

Código	Entrada	Salida Esperada	Resultado Obtenido	Conclusión
CP2.1.1	Desea desapuntarse de una actividad válida.	true	El usuario fue desapuntado correctamente.	La acción se completó exitosamente.
CP2.1.2	Desea desapuntarse de una actividad con un usuario no registrado.	false	El sistema indicó que el usuario no existe, y no se desapuntó.	La acción falló, como era esperado.
CP2.1.3	Desea desapuntarse de una actividad inexistente.	false	El sistema indicó que la actividad no existe, y no se desapuntó.	La acción falló, como era esperado.
CP2.2.1	Desea modificar sus datos de usuario.	"Usuario modificado con éxito"	El sistema modificó exitosamente los datos del usuario.	La edición se realizó correctamente.
CP2.2.2	Desea modificar los datos de un usuario inexistente.	"No existe el usuario"	El sistema informó que el usuario no existe, y no realizó modificaciones.	La acción falló, como era esperado.
CP2.3.1	Desea valorar una actividad en la que está registrado.	true	El sistema registró la valoración correctamente.	La valoración se completó exitosamente.
CP2.3.2	Desea valorar una actividad con un usuario no registrado.	false	El sistema indicó que el usuario no existe, y no realizó la valoración.	La acción falló, como era esperado.
CP2.3.3	Desea valorar una actividad inexistente.	false	El sistema indicó que la actividad no existe, y no realizó la valoración.	La acción falló, como era esperado.
CP2.3.4	Desea valorar una actividad con una puntuación inválida.	false	El sistema indicó que la valoración está fuera de rango, y no la registró.	La acción falló, como era esperado.

**Pruebas unitarias:**

Primero hemos usado caja negra para diseñar las pruebas y después las hemos implementado

en pruebas unitarias usando la clase de MSTEST (.NET) en el proyecto.

**Pruebas de integración:**

Integración no incremental: Cada uno hacía su parte y la probaba, luego lo juntamos todo y probamos todo a la vez de nuevo.

**Pruebas de validación:**

- **Prueba alfa:** El padre de Gabriel ha probado la aplicación siguiendo las indicaciones para probar correctamente las nuevas funcionalidades de la aplicación

**Pasos a seguir**

- Registro de un usuario
  - Iniciar sesión
  - Ir al apartado de actividades
  - Apuntarse a una actividad
  - Desapuntarse de esa actividad
  - Volver a apuntarse a una actividad
  - Valorar actividad
  - Desapuntarse de esa actividad
  - Salir del apartado de actividades
  - Entrar al apartado de edición de usuario
  - Editar su usuario
  - Guardar cambios
  - Cerrar sesión
- **Prueba beta:** En este caso, cada uno probaba la parte del otro y luego nos decíamos si había fallos o si había alguna recomendación.

### Pruebas de regresión:

En este sprint 3, se realizó un cambio en el DTO de actividades otra vez debido a que añadimos un campo nuevo que es "Media Valoración"

```
public class ActividadDTO
{
    public int Id_Actividad { get; set; }
    public string Nombre { get; set; }
    public string Descripcion { get; set; }
    public string DNI_Monitor { get; set; }

    public double MediaValoracion { get; set; }

    public ActividadDTO()
    {
    }
}
```

Debido a este nuevo campo hemos alterado el constructor que por defecto pone en 1 este campo

```
3 referencias | 1/3 pasando
public ActividadDTO(int idActividad, string nombre, string descripcion, string dniMonitor)
{
    Id_Actividad = idActividad;
    Nombre = nombre;
    Descripcion = descripcion;
    DNI_Monitor = dniMonitor;
    MediaValoracion = 1;
}
```

Pero donde usamos la actividad para cualquier cosa hemos tenido que añadir la línea de la media valoración

```
public String bajaActividad(ActividadDTO actividadDTO)
{
    Actividad actividad = new Actividad();

    actividad.Id_Actividad = actividadDTO.Id_Actividad;
    actividad.Nombre = actividadDTO.Nombre;
    actividad.Descripcion = actividadDTO.Descripcion;
    actividad.DNI_Monitor = actividadDTO.DNI_Monitor;
    actividad.Media_Valoracion = (decimal?)actividadDTO.MediaValoracion;
```

Esto ha alterado sobre todo la clase ActividadMangment y también su repository.

Como no hemos alterado en constructor ya que siempre por defecto se crea con 1 en media valoración las pruebas unitarias que comprueban la creación de una actividad no han sido modificadas

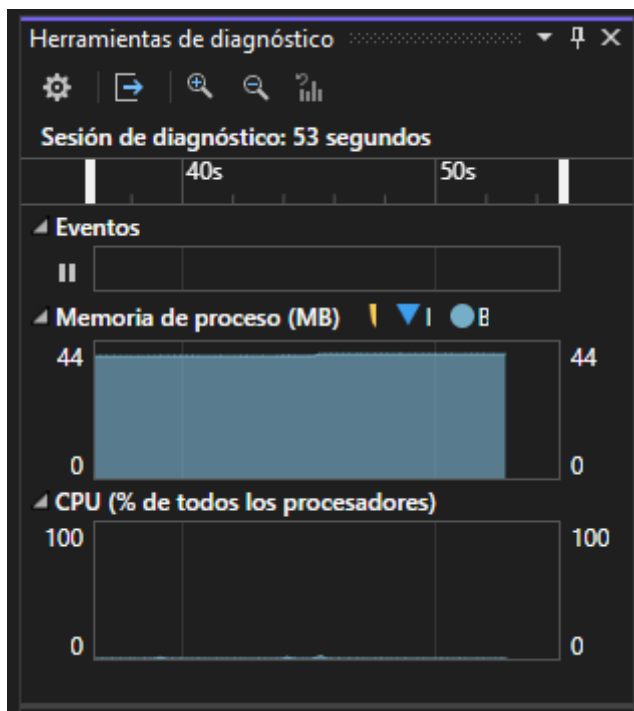
```
[TestClass]
0 referencias
public class TestsRegistroActividad
{
    [TestMethod]
    0 referencias
    public void comprobarRegistroActividadCorrecto()
    {
        ActividadDTO actividad = new ActividadDTO(10, "Prueba", "Prueba", "18879231C");
        string mensaje = new Negocio.Managment.ActividadManagment().RegistrarActividad(actividad);
        Assert.AreEqual(mensaje, "Actividad insertada con éxito.");
        System.Threading.Thread.Sleep(5000);
    }

    [TestMethod]
    0 referencias
    public void comprobarRegistroActividadIncorrectoIdRepetido()
    {
        ActividadDTO actividad = new ActividadDTO(10, "Prueba", "Prueba", "18879231C");
        string mensaje = new Negocio.Managment.ActividadManagment().RegistrarActividad(actividad);
        Assert.AreEqual(mensaje, "Error al guardar la actividad: An error occurred while updating the entries. See the inner exception for details.");
    }
}
```

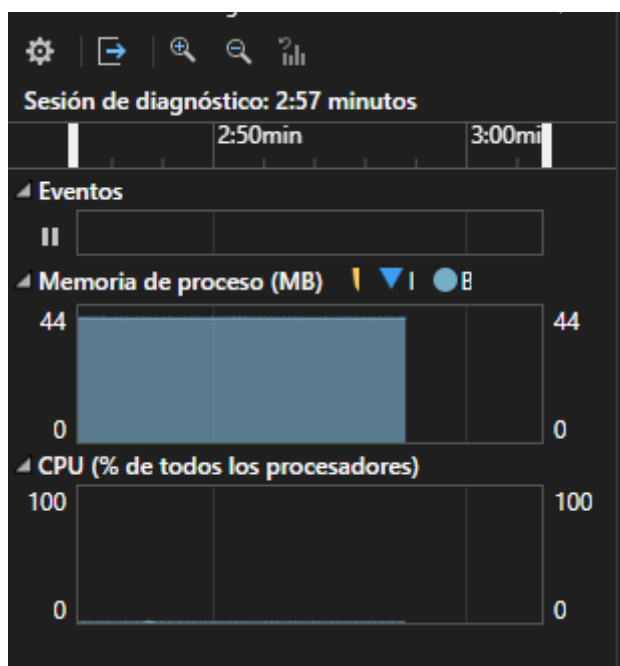
### Pruebas de uso de recursos:

Hemos probado a desapuntarnos de la actividad, valorar una actividad y editar el usuario.

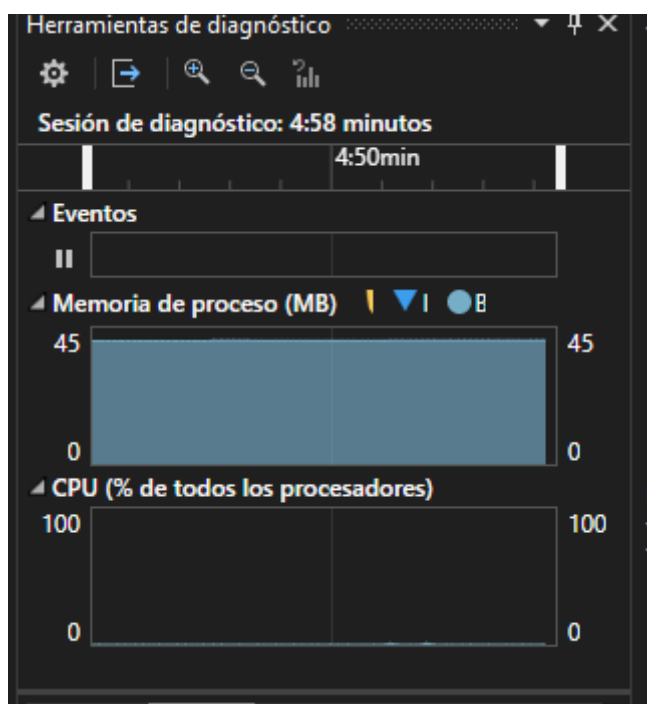
Desapuntarte de una actividad no consume recursos apenas ya que se queda en el mismo uso de memoria que cuando estamos en la aplicación normal.



Tampoco se altera al valorar una actividad



Sube un poco cuando haces la edición de los datos de un usuario, pero es insignificante



La aplicación consume pocos recursos ya que está bien optimizada.

**Pruebas del sistema:**

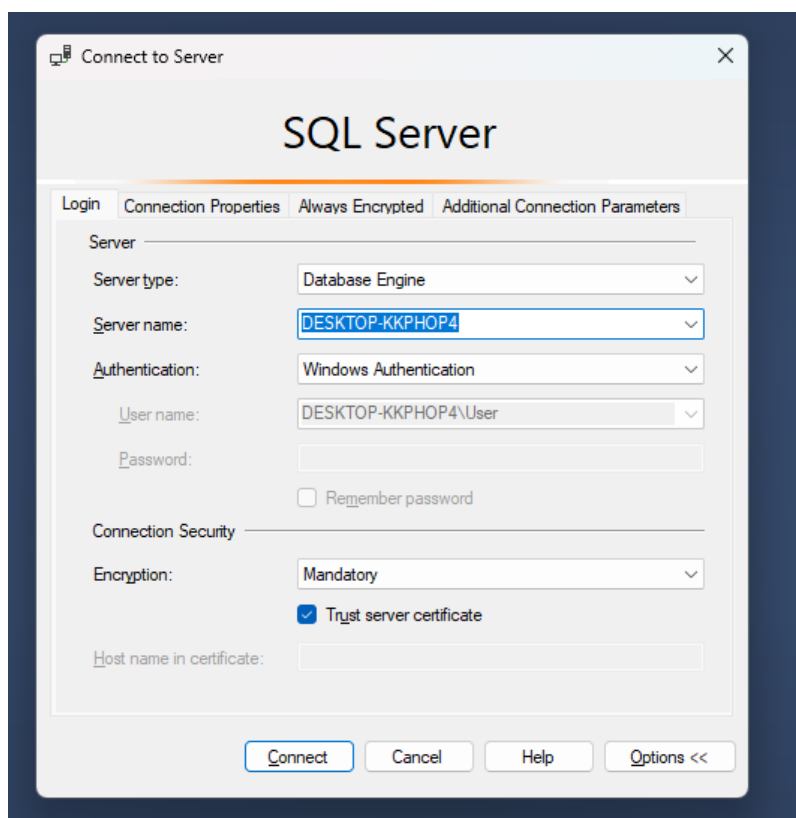
- **Pruebas de recuperación:** Hemos probado varias veces a hacer operaciones incorrectas sobre la base de datos o el propio programa y todos los casos han sido controlados para que la ejecución siga.
- **Prueba de seguridad:** Las pruebas que podríamos hacer aquí son comprobar si en la cadena de conexión a la base de datos salen datos confidenciales como el usuario y la contraseña, ya que esto le daría acceso a alguien a la base de datos.
- **Prueba de resistencia:** Como tal, la aplicación aún no necesita esto porque hace pocas cosas, pero hemos comprobado que cuando se listan las actividades (que son muchas consultas), no hay sobrecarga, incluso si hay muchas actividades.

### Prueba de seguridad:

Las pruebas que podríamos hacer aquí son comprobar si en la cadena de conexión a la base de datos salen datos confidenciales como el usuario y la contraseña, ya que esto le daría acceso a alguien a la base de datos.

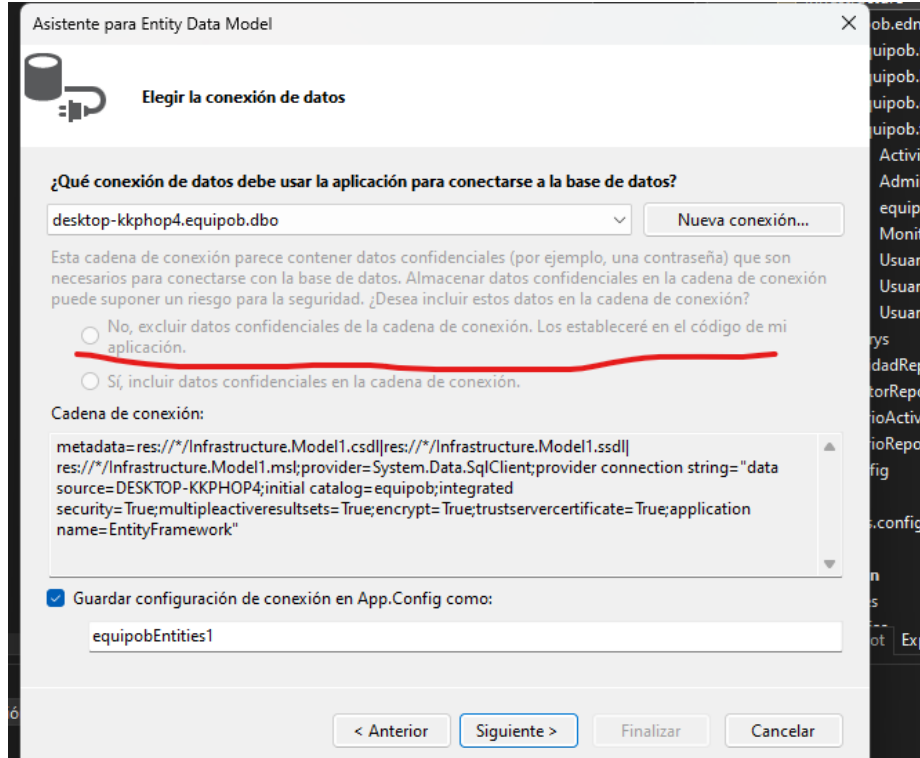
```
<connectionStrings> <add name="equipobEntities"
connectionString="metadata=res:///Infrastructure.equipob.csdl|res:///Infrastructure.e
quipob.ssdl|res:///*/Infrastructure.equipob.msl;provider=System.Data.SqlClient;provide
r connection string=&quot;data source=DESKTOP-KKPHOP4;initial
catalog=equipob;integrated
security=True;multipleactiveresultsets=True;encrypt=True;trustservercertificate=True;ap
plication name=EntityFramework&quot;;" providerName="System.Data.EntityClient" />
</connectionStrings>
```

Por ejemplo, en esta cadena de conexión salen datos, pero no sale una contraseña como tal porque la conexión a la base de datos está en local con Windows authentication.





En caso de que tengamos un servidor tendríamos que tener cuidado al crear el Entity data model y seleccionar esta opción



Lo mejor sería tener un fichero en local con los datos de la conexión, como la contraseña y que en el código de la aplicación no salga por ningún lado

## Desarrollo de la Interfaz: Entrevista Inicial con el Cliente

### Descripción General de la Aplicación

La aplicación está diseñada para gestionar usuarios (clientes, monitores, administradores) y las actividades dentro de un gimnasio. Su objetivo es facilitar la administración y mejorar la experiencia de los usuarios, integrando funciones clave:

- **Gestión de usuarios:** Registro y edición de información personal.
- **Gestión de actividades:** Creación, modificación y eliminación de actividades (solo administradores).
- **Inscripciones:** Permitir a los usuarios inscribirse en actividades disponibles.
- **Valoración de actividades:** Calificación de actividades (1-5) para retroalimentación.
- **Asignación de monitores:** Asignación de monitores a actividades (solo administradores).

### Preguntas para la Entrevista

1. **¿En qué dispositivos utilizarán principalmente la aplicación?**
  - Ordenadores, tablets, teléfonos móviles u otros dispositivos.
  - **Objetivo:** Determinar si la interfaz debe ser adaptable a diferentes plataformas y compatible con diseño responsivo.
2. **¿Utilizan alguna herramienta similar para gestionar estas tareas en el gimnasio?**
  - **Si la respuesta es sí:**
    - ¿Qué características les gustaría conservar?
    - ¿Qué aspectos cambiarían o mejorarían?
    - ¿Qué no les gusta de la herramienta actual?
  - **Si la respuesta es no:**
    - ¿Cómo gestionan actualmente las inscripciones y actividades en el gimnasio?
  - **Objetivo:** Identificar necesidades y áreas de mejora para ofrecer una solución eficiente.

### 3. ¿Quiénes serán los usuarios principales de la aplicación?

- Administradores, usuarios o ambos.
- Nivel de conocimiento tecnológico de los usuarios finales.
- **Objetivo:** Garantizar una interfaz accesible y fácil de usar según las habilidades tecnológicas de los usuarios.

### Pruebas de Usuario y Experto

#### Objetivo General

Evaluar la funcionalidad y facilidad de uso de la aplicación para la gestión de usuarios y actividades. Las pruebas se dividen en:

1. **Pruebas de experto:** Detectar errores técnicos y evaluar la eficiencia del diseño.
2. **Pruebas de usuario:** Medir la experiencia de los usuarios finales y evaluar la usabilidad.

#### Prueba de Experto

- **Alcance:**  
Evaluación de las siguientes tareas clave:
  - Registrarse en una actividad.
  - Valorar una actividad.
  - Desapuntarse de una actividad.
  - Editar información personal de un usuario.
- **Objetivos:**
  - Detectar fallos técnicos.
  - Medir el tiempo necesario para completar cada tarea.
  - Verificar la claridad de los flujos de trabajo.
- **Número de pruebas y lugar de ejecución:**
  - Cada miembro del equipo de desarrollo realizará una prueba.
  - Entorno controlado durante las primeras etapas del desarrollo.
- **Participantes:**
  - Miembros del equipo de desarrollo con conocimiento técnico de la aplicación.

## Prueba de Usuario

- **Alcance:**

Los usuarios finales deberán completar las siguientes tareas:

- Registrarse en una actividad.
- Valorar una actividad (1-5).
- Desapuntarse de una actividad inscrita.
- Editar su propia información personal.

- **Objetivos:**

- Medir el tiempo necesario para completar las tareas.
- Evaluar la intuición de los flujos de trabajo.
- Identificar problemas de navegación o comprensión de funciones.

- **Número de pruebas y lugar de ejecución:**

- Pruebas realizadas con usuarios finales en el gimnasio, usando dispositivos compatibles.
- **Fases:**
  - Primera visita: Evaluación inicial.
  - Segunda visita: Comparación del desempeño tras interacción previa con la interfaz.

- **Participantes:**

- Clientes y monitores que interactuarán regularmente con la aplicación.

## Preguntas al Usuario Después de la Prueba

1. ¿Te resultó fácil completar las tareas asignadas?
2. ¿Hubo algún paso que encontraste confuso o complicado?
3. ¿La interfaz fue intuitiva y fácil de navegar?
4. ¿Qué cambios o mejoras sugerirías para hacer la aplicación más fácil de usar?
5. ¿Consideras que el tiempo necesario para completar las tareas fue adecuado?