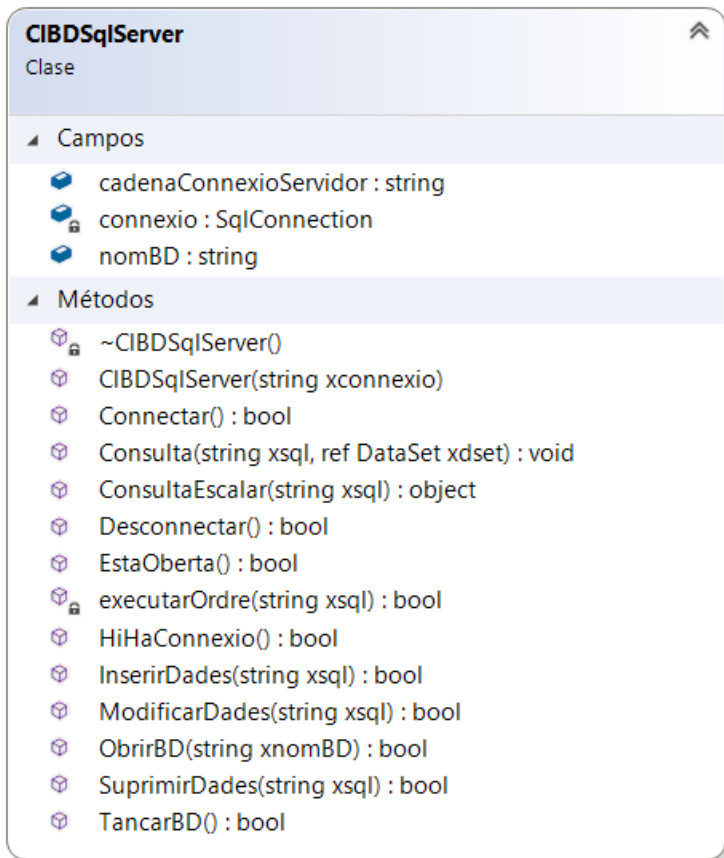


CICLE	DESENVOLUPAMENT D'APLICACIONS MULTIPLATAFORMA	
MÒDUL	M3 - PROGRAMACIÓ	
U.FORMATIVA	UF6 – INTRODUCCIÓ A LA PERSISTÈNCIA EN BASES DE DADES	
PROFESSOR	JAUME FADÓ	CURS 2022/23

CIBDSqlServer



Propietats

- **cadenaConnexioServidor** Conté la cadena que permet connectar-nos al servidor SQLServer. Aquesta cadena no porta el nom de la BD, només les dades de connexió amb el servidor. La raó és perquè volem que ens permeti accedir a bases de dades diferents del mateix servidor.
- **nomBD** El nom de la base de dades a la que volem accedir del servidor SQLServer.
- **connexió** La instància de la classe **SqlConnection** de .NET amb la que ens connectarem al servidor.

CICLE	DESENVOLUPAMENT D'APLICACIONS MULTIPLATAFORMA	
MÒDUL	M3 - PROGRAMACIÓ	
U.FORMATIVA	UF6 – INTRODUCCIÓ A LA PERSISTÈNCIA EN BASES DE DADES	
PROFESSOR	JAUME FADÓ	CURS 2022/23

Mètodes

- **Constructor** Se li passa la cadena de connexió que no tindrà el nom de la base de dades tal com s'ha explicat en les propietats. Aquesta cadena es guardarà en la propietat corresponent. El constructor no realitzarà la connexió perquè tenim un mètode per a fer-ho. Fent-ho així tindrem la possibilitat de connectar-nos a diferents bases de dades del mateix servidor SQL Server.
- **Connectar** Fa la instància **SqlConnection** associada a la propietat **connexió** mitjançant la **cadenaConnexioServidor** i retorna cert o fals segons si ha pogut fer la instància o no.
- **HiHaConnexio** Indica si disposem de la instància **SqlConnection**, és a dir, si la propietat **connexió** és nul o no.
- **Disconnectar** Posa la propietat **connexió** a NULL. Retorna cert o fals segons si s'ha pogut fer aquesta operació amb normalitat.
- **ObrirBD** Se li passa el nom de la BD que es vol obrir. Aquest nom es guardarà a la propietat **nomBD** per a poder saber amb quina BD s'està treballant. En SQL Server entenem que obrir la BD correspon a assignar-la a la instància de **SqlConnection** (pot ser d'utilitat el mètode **ChangeDatabase** de la classe **SqlConnection**). Si la connexió no està oberta caldrà obrir-la per a poder assignar la nova BD. Aquest mètode retornarà un booleà segons si tot ha anat bé o no.
- **TancarBD** Posa la cadena nul·la a la propietat **nomBD** i manté la connexió sense canvis. Aquest mètode retornarà un booleà segons si tot ha anat bé o no.
- **EstaOberta** Retorna un booleà que indica si la connexió està o no oberta.
- **Consulta** Se li passa una instrucció de consulta i un **DataSet per referència** que és on es deixaran les files obtingudes. Cal verificar que la instrucció comença per "SELECT", si no hi comença es mostrarà un missatge d'error de sintaxi.

CICLE	DESENVOLUPAMENT D'APLICACIONS MULTIPLATAFORMA	
MÒDUL	M3 - PROGRAMACIÓ	
U.FORMATIVA	UF6 – INTRODUCCIÓ A LA PERSISTÈNCIA EN BASES DE DADES	
PROFESSOR	JAUME FADÓ	CURS 2022/23

- **ConsultaEscalar** Se li passa una instrucció de consulta i retorna l'objecte resultant en la primera columna de la primera fila o NULL. Cal verificar que la instrucció comença per "SELECT", si no hi comença es mostrarà un missatge d'error de sintaxi.
- **executarOrdre** Executa la instrucció SQL que se li passa. Està orientat a instruccions que no són de consulta. És un mètode privat. Retorna un booleà indicant si s'ha pogut realitzar.
- **InserirDades** Se li passa una instrucció d'inserció i retorna un booleà indicant si s'ha pogut fer la inserció o no. Cal verificar que la instrucció comença per "INSERT INTO", si no hi comença es mostrarà un missatge d'error de sintaxi.
- **ModificarDades** Se li passa una instrucció de modificació de dades i retorna un booleà indicant si s'ha pogut fer la inserció o no. Cal verificar que la instrucció comença per "UPDATE" , si no hi comença es mostrarà un missatge d'error de sintaxi.
- **SuprimirDades** Se li passa una instrucció de supressió i retorna un booleà indicant si s'ha pogut fer la inserció o no. Cal verificar que la instrucció comença per "DELETE" , si no hi comença es mostrarà un missatge d'error de sintaxi.

CONSIDERACIONS GENERALS A TENIR EN COMPTE

1. Cal mirar el diagrama de classes per a esbrinar quins elements són **privats** i quins **públics**. Cal recordar que Visual Studio anomena **Camps** a les variables que no tenen **Get** i **Set** associat però que nosaltres les tractem com si fossin propietats.
2. En C#, si un mètode **public** el fem també **static** el podrem cridar com si fos una funció sense haver de tenir una instància d'aquella classe. Per exemple, si tenim una classe **CIPassword** on hi definim el mètode **Encriptar** com `publicstatic String Encriptar(string xs)` podrem fer crides com `CIPassword.Encriptar("kepassatronku");`
Si no posem static només podrem fer aquesta crida **a partir d'una instància** de CIPassword.

CICLE	DESENVOLUPAMENT D'APLICACIONS MULTIPLATAFORMA	
MÒDUL	M3 - PROGRAMACIÓ	
U.FORMATIVA	UF6 – INTRODUCCIÓ A LA PERSISTÈNCIA EN BASES DE DADES	
PROFESSOR	JAUME FADÓ	CURS 2022/23

3. S'han de programar les classes de manera que el codi sàpiga gestionar tota mena d'errors que es puguin produir perquè les dades no són correctes, perquè en falten, perquè s'intenta fer una operació que no es pot realitzar, etc. Si s'utilitzen operacions `try` cal que les excepcions mostrin missatges de tipus `MessageBox` amb les causes de l'excepció i tota la informació que sembli convenient.