

# TALLER DE DESEMPEÑO PROFESIONAL

Semana 12

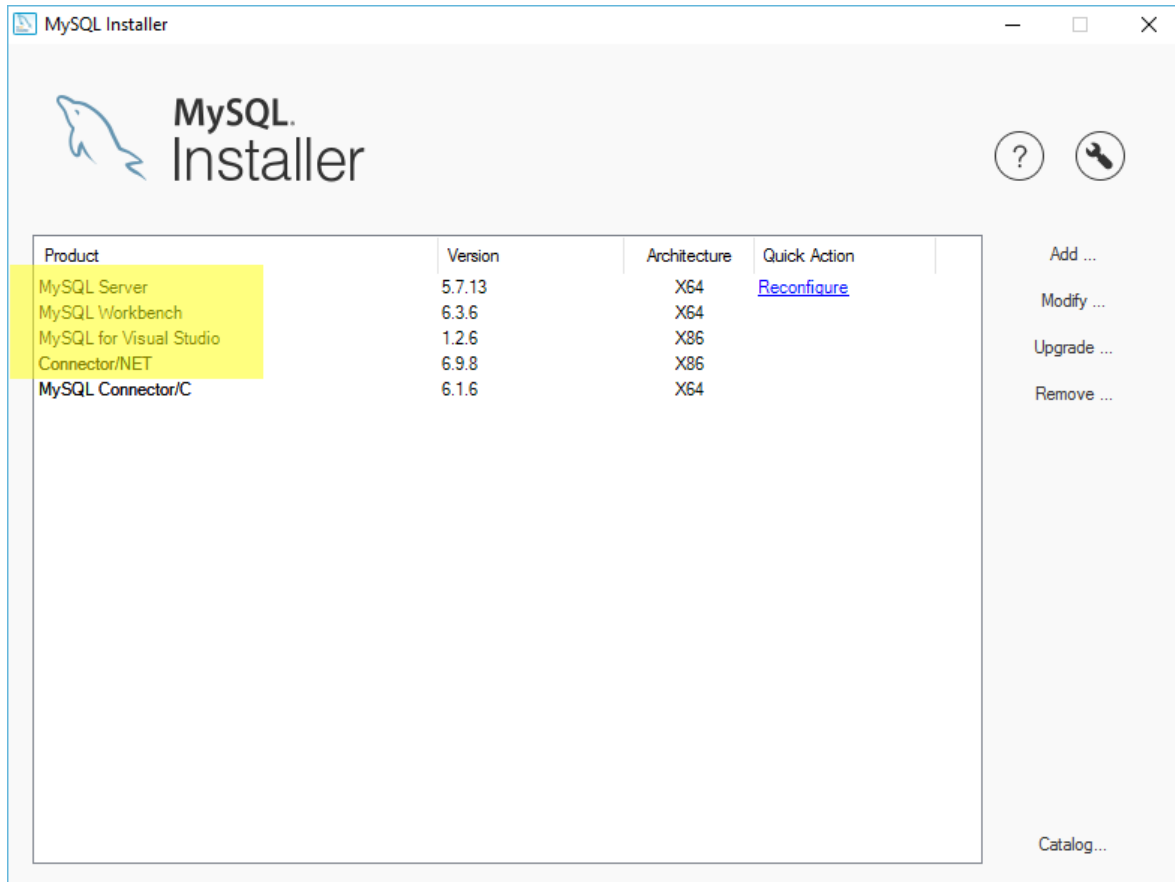
## MySQL con Entity Framework

### Preparación de MySQL

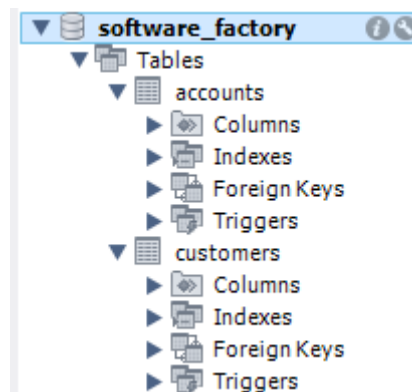
Debemos tener instalado MySQL, descargamos el instalador:

✓ <https://dev.mysql.com/downloads/installer/>

Debemos contar con lo siguiente:



Creamos una base de datos denominada "Software Factory" y a continuación se indican los Script de las tablas "Accounts" y "Customers":



```
CREATE TABLE customers(
customer_id INT NOT NULL AUTO_INCREMENT,
full_name VARCHAR(20) NOT NULL,
PRIMARY KEY ( customer_id )
)
```

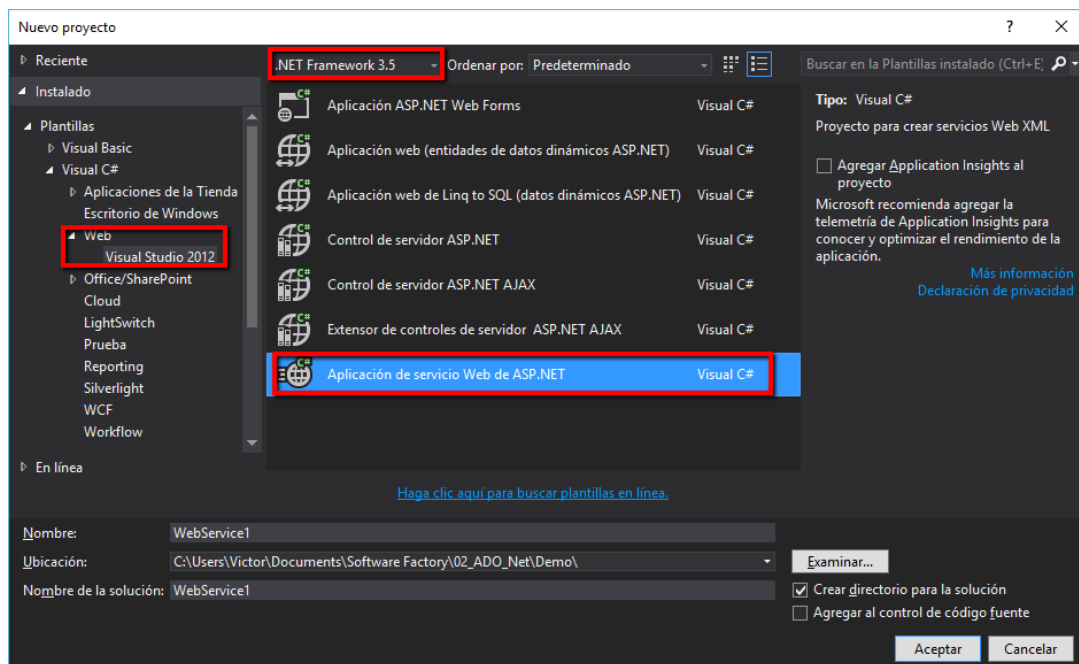
```
CREATE TABLE accounts(
account_id INT NOT NULL AUTO_INCREMENT,
customer_id INT NOT NULL ,
balance FLOAT NOT NULL,
PRIMARY KEY ( account_id ), FOREIGN KEY (customer_id) REFERENCES customers(customer_id)
) ENGINE=INNODB;
```

El flag “derived\_merge” controla cuando se fusionan tablas y vistas derivadas en el bloque de consulta externa. Esto puede causar error al usar una base de datos MySQL en Entity Framework. Por lo cual debemos ejecutar el siguiente query sobre la base de datos a usar:

```
use software_factory;
set global optimizer_switch='derived_merge=OFF';
```

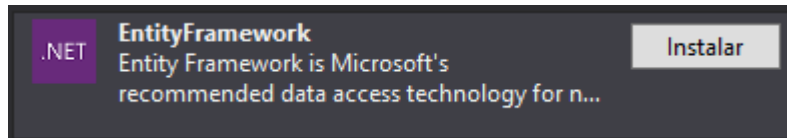
## Creación del proyecto en Visual Studio

Creamos el proyecto de la siguiente manera:

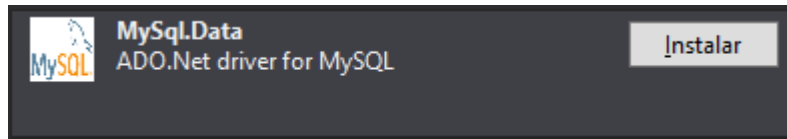


Escogemos un proyecto del tipo “MVC”. Y cambiamos el tipo de autenticación a “Sin autenticación”. Sobre el proyecto seleccionamos “Administrar paquetes NuGet” y debemos asegurarnos que tenemos instalado las últimas versiones de lo siguiente:

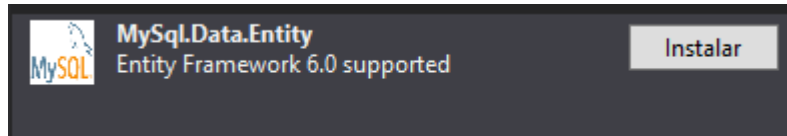
- ✓ Entity Framework:
  - Es un asignador objeto-relacional que permite a los desarrolladores de .NET trabajar con datos relacionales



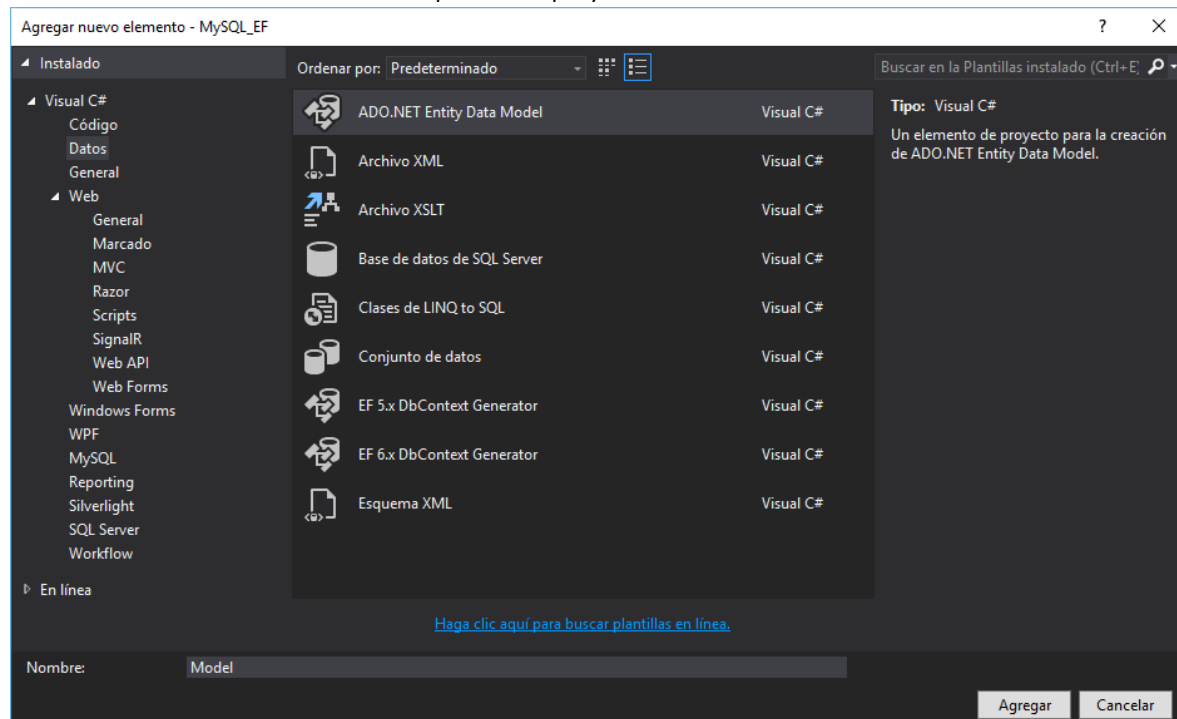
- ✓ MySql.Data:
  - Driver ADO.NET para MySQL




- ✓ MySql.Data.Entity:



Finalizando con las instalaciones recompilamos el proyecto.




Asistente para Entity Data Model




Elegir contenido del modelo


¿Qué debería contener el modelo?




EF Designer desde base de datos



Modelo vacío de EF Designer



Modelo vacío de Code First



Code First desde base de datos

Crea un modelo de Code First basado en una base de datos existente. Puede elegir la conexión de base de datos, la configuración del modelo y los objetos de base de datos que se incluirán en el modelo.

< Anterior

Siguiente >

Finalizar

Cancelar

Elegir origen de datos

?

×

Origen de datos:

Archivo de base de datos de Microsoft SQL Server

Microsoft SQL Server

MySQL Database

<otro>

Descripción

Use this selection to connect to MySQL Server using the .NET Framework Data Provider for MySQL

Proveedor de datos:

.NET Framework Data Provider for MySQL

▼

☒ Utilizar siempre esta selección

Continuar

Cancelar

Propiedades de la conexión?×

Especifique la información para establecer conexión con el origen de datos seleccionado o haga clic en "Cambiar" para elegir un origen y/o un proveedor de datos diferente.

Origen de datos:

MySQL Database (MySQL Data Provider)Cambiar...

Server name:

User name:

Password:

☐ Save my password

Database name:

Avanzadas...

Probar conexiónAceptarCancelar



## Elegir la conexión de datos

¿Qué conexión de datos debe usar la aplicación para conectarse a la base de datos?

localhost(software\_factory) ▼

Nueva conexión...

Esta cadena de conexión parece contener datos confidenciales (por ejemplo, una contraseña) que son necesarios para conectarse con la base de datos. Almacenar datos confidenciales en la cadena de conexión puede suponer un riesgo para la seguridad. ¿Desea incluir estos datos en la cadena de conexión?

- ☐ No, excluir datos confidenciales de la cadena de conexión. Los estableceré en el código de mi aplicación.
- ☒ Sí, incluir datos confidenciales en la cadena de conexión.

Cadena de conexión:

server=localhost;user id=root;database=software\_factory

☒ Guardar configuración de conexión en Web.Config como:

MySQL

< Anterior

Siguiente >

Finalizar

Cancelar

Asistente para Entity Data Model

Elegir los objetos y la configuración de la base de datos

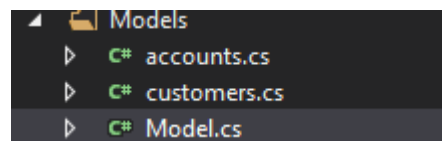
¿Qué objetos de la base de datos desea incluir en su modelo?

- ☒ Tablas
  - ☒ software\_factory
    - ☒ person
    - ☒ polling\_place
  - ☐ Vistas

☐ Poner en plural o en singular los nombres de objeto generados  
☒ Incluir columnas de clave externa en el modelo  
☐ Importar procedimientos almacenados y funciones seleccionados en Entity Model

< Anterior    Siguiente >    Finalizar    Cancelar

Se generan 2 clases que representan las tablas en MySQL y la clase "Model" que nos permitirá interactuar con la tablas:



El archivo "Web.config" podemos notar lo siguiente:

En la sección de `entityFramework` notamos el nuevo provider que nos permitirá la conexión a MySQL

```
<providers>
  <provider invariantName="System.Data.SqlClient"
type="System.Data.Entity.SqlServer.SqlProviderServices, EntityFramework.SqlServer"
/>
  <provider invariantName="MySQL.Data.MySqlClient"
type="MySQL.Data.MySqlClient.MySqlProviderServices, MySQL.Data.Entity.EF6,
Version=6.9.8.0, Culture=neutral, PublicKeyToken=c5687fc88969c44d"></provider>
</providers>
```

En la sección de `system.data` notamos cambios igualmente.

```
<DbProviderFactories>
  <remove invariant="MySQL.Data.MySqlClient" />
```



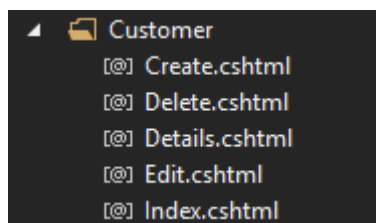
```

<add name="MySQL Data Provider" invariant="MySql.Data.MySqlClient"
description=".Net Framework Data Provider for MySQL"
type="MySql.Data.MySqlClient.MySqlClientFactory, MySql.Data, Version=6.9.8.0,
Culture=neutral, PublicKeyToken=c5687fc88969c44d" />
</DbProviderFactories>
<connectionStrings>
<add name="MySQL" connectionString="server=localhost;user
id=root;password=root;database=software_factory"
providerName="MySql.Data.MySqlClient" />
</connectionStrings>

```

Procedemos a recompilar el proyecto.

Teniendo el modelo de datos. Procedemos a crear los controladores que permitan el CRUD de ambas tablas.



Habiendo creado las vistas en la vista Views/Shared/\_Layout.cshtml agregamos lo siguiente:

```

<ul class="nav navbar-nav">
<li>@Html.ActionLink("Inicio", "Index", "Home")</li>
<li>@Html.ActionLink("Acerca de", "About", "Home")</li>
<li>@Html.ActionLink("Contacto", "Contact", "Home")</li>
<li>@Html.ActionLink("Clientes", "Index", "Customer")</li>
<li>@Html.ActionLink("Cuentas", "Index", "Account")</li>

```

</ul>

Con esto abremos finalizado el proyecto.

Demo

La vista de clientes:

## Index

[Create New](#)

**full\_name**

Víctor Daniel

[Edit](#) | [Details](#) | [Delete](#)

Daniel Kross

[Edit](#) | [Details](#) | [Delete](#)

© 2016 - Mi aplicación ASP.NET

The screenshot shows a database management interface. At the top, there are tabs for 'Query 1', 'accounts', and 'customers'. The 'customers' tab is active. Below the tabs is a toolbar with various icons for file operations, search, and execution. A text box contains the SQL query: `SELECT * FROM software_factory.customers;`. Below the query is a 'Result Grid' section. It has a 'Filter Rows' input field and an 'Edit' button. The grid displays the following data:

customer_id	full_name
1	Víctor Daniel
3	Daniel Kross
NULL	NULL

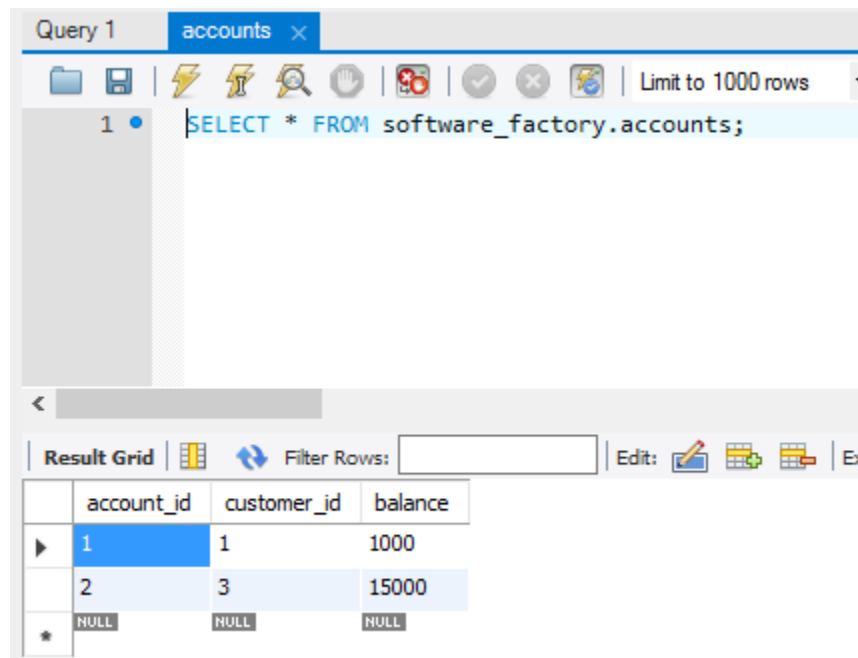
La vista de cuentas:

# Index

[Create New](#)

full_name	balance	
Víctor Daniel	1000	<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>
Daniel Kross	15000	<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>

© 2016 - Mi aplicación ASP.NET



## Referencias:

- ✓ Definición de "derived\_merge"
  - Link: <https://dev.mysql.com/doc/refman/5.7/en/switchable-optimizations.html>
- ✓ Corrección del error al conectar con MySQL
  - Link: <http://stackoverflow.com/questions/33575109/mysql-entity-the-value-for-column-isprimarykey-in-table-tabledetails-is>