educación IT
CODE YOUR FUTURE

Programación Web .NET Core

Módulo 4

Diseñar modelos

Diseñar modelos

Cada clase de modelo representa un tipo de objeto que su aplicación gestiona:

- **Identificación de clases de modelo y propiedades**.
- **Utilización de diagramas**.
- **Identificación de relaciones**.
- ***EntityFramework***.
- **Diseño en programación ágil y extrema**.

Identificación de clases de modelo y propiedades

Los **casos de uso**, **escenarios de uso** o **historias de usuarios** que se recopilaron, durante la fase de análisis del proyecto, deben permitir determinar las ***clases de modelo*** que es necesario crear. Cada clase de modelo posee una variedad de propiedades.

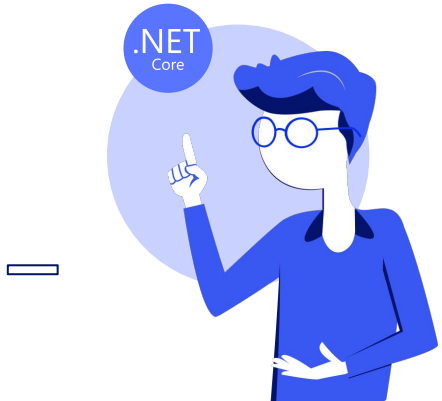
Considerar el siguiente caso de uso compartido anteriormente:

1. El **usuario** hace clic en el enlace llamado ***Agregar foto***, en el menú principal del sitio.
2. Si el **usuario** es **anónimo**, aparece la página de inicio de sesión y el **usuario** proporciona las credenciales.
3. Si las credenciales son correctas, aparecerá la vista ***Crear foto***.
4. El usuario escribe un **título**.

5. El usuario especifica el **archivo de foto** a cargar.
6. El usuario opcionalmente escribe una **descripción** para la foto.
7. El usuario hace clic en el botón ***Cargar***.
8. La aplicación web almacena la nueva foto y muestra la galería de fotos al usuario.

Este ejemplo incluye los siguientes objetos, cada uno de los cuales requiere una clase de modelo:

- **Usuario:** Debe incluir propiedades como el **nombre de usuario y la contraseña**.
- **Foto:** Debe incluir las propiedades de **título, archivo y descripción**.



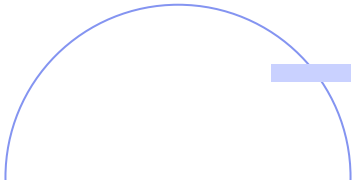
Utilizar diagramas

Los diagramas sirven para analizar la información que gestiona el sitio web y sugerir los datos desde el modelo físico de la base de datos. También para planificar clases de modelos.

Cada objeto en su diagrama debe implementarse como una clase de modelo **MVC**.

Relaciones

Una vez identificadas las clases de modelo que se implementarán en el sitio web, también se deben considerar las relaciones entre los objetos. Por ejemplo, en el caso de uso de la aplicación ***PhotoSharingApplication*** que se vió antes, una foto está asociada a un solo usuario. Esto se conoce como ***relación uno a uno***. Cada usuario, sin embargo, puede crear varias fotos. Esto se conoce como ***relación de uno a varios***.



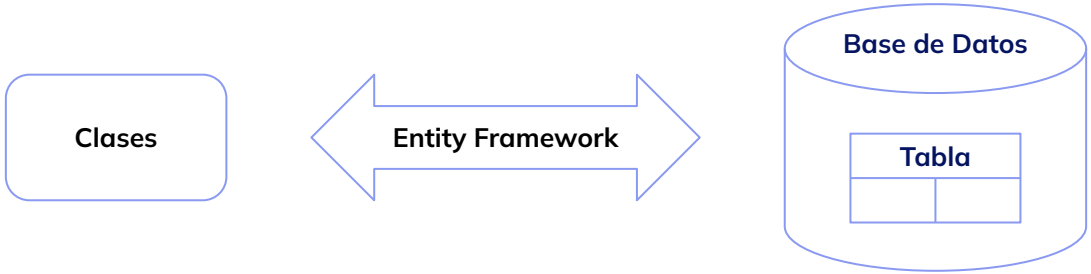
Los diagramas incluyen relaciones tales como **vínculos entre objetos.** Los números al final de cada enlace muestran si la relación es de uno a uno, de uno a varios o de varios a varios.

Entity Framework

Es un mapa de asignación de objetos relacionales (ORM) para .NET Framework. Un marco ORM vincula tablas de bases de datos y las clases de modelo de la aplicación.

No es necesario escribir código SQL para consultar o actualizar las tablas de la base de datos porque Entity Framework ya lo hace.

Entity Framework trabaja con ***Lenguaje de consulta integrado* (*LINQ*)**.


Clases
Entity Framework
Base de Datos
Tabla

Diseño de controladores

Diseño de controladores

Controller	Action
Photo	AddPhoto (GET)
	AddPhoto (POST)
	DisplayGallery (GET)
User	Logon (GET)
	Logon (POST)

En una aplicación web **ASP.NET Core MVC**, los controladores son clases basadas en **.NET Framework** que heredan de la clase base de **`Microsoft.AspNetCore.Mvc.Controller`**. Implementan la lógica de entrada, esto significa que reciben información del usuario en el formulario de solicitudes **HTTP**, y seleccionan el modelo correcto y la vista correcta a utilizar, para poder formular una respuesta.

Identificar controladores y acciones

En una aplicación web **ASP.NET Core MVC**, suele haber un controlador para cada clase de modelo.

Es convención que si la clase de modelo se llama **`Foto`**, el controlador se llamará **`PhotoController`**. Si se sigue esta convención, en el diseño, se puede usar el comportamiento de enrutamiento predeterminado de **MVC** para seleccionar el controlador correcto para una solicitud.

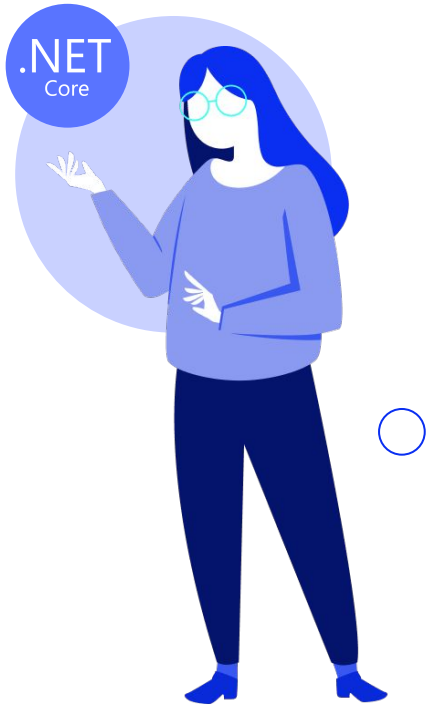
En un controlador puede haber muchas acciones; cada acción se implementa como un método y generalmente devuelve una vista. Muchas veces, se necesitan acciones separadas para **HTTP GET** y **HTTP POST**.

Por ejemplo, en el caso de uso `Add Photo`, compartido anteriormente, ya se han identificado las clases modelo de **foto** y **usuario** de ese caso de uso. En **MVC** por convención, se debe crear un **`PhotoController`** y un **`UserController`**.

El caso de uso muestra las siguientes acciones para cada controlador:

- **`Photo AddPhoto (GET):`** La acción **`AddPhoto`** para las solicitudes **`GET`** crea una nueva instancia de la clase modelo **`Photo`**, establece valores predeterminados como la fecha de creación y la pasa a la vista.
- **`AddPhoto (POST):`** La acción **`AddPhoto`** por **`POST`** guarda los valores de la foto en la base de datos y redirige el navegador a la Acción **`DisplayGallery`**.
- **`DisplayGallery(GET):`** La acción llamada **`DisplayGallery`**, para solicitudes **`GET`**, muestra todas las fotos almacenadas en la base de datos.
- **`Inicio de sesión de usuario (GET):`** La acción de inicio de sesión, para solicitudes **`GET`**, muestra la vista ***Login*** a un usuario anónimo para ingresar sus credenciales (Nombre y Contraseña).

- **`Inicio de sesión (POST):`** La acción de inicio de sesión, para solicitudes **`POST`**, comprueba las credenciales del usuario contra la base de datos de usuarios. Si las credenciales son correctas, la acción llamada **`Login`** autentica y redirige al usuario a la página solicitada originalmente.



Diseño de vistas

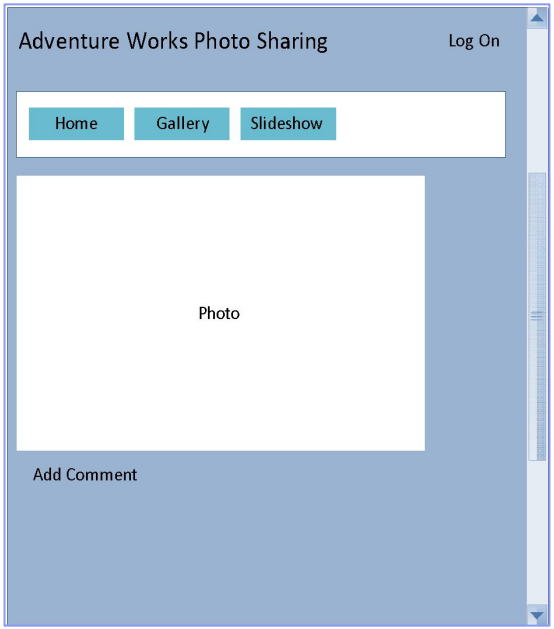
Diseño de vistas

- **Vistas.**
- **Layout.**
- **Vistas parciales y componentes de vista.**

Diseño de vistas

En una aplicación web ASP.NET Core MVC, la interfaz gráfica del usuario se crea construyendo *vistas*. La relación entre los controladores **MVC** y las vistas es ***de varios a uno***. Por ejemplo, un controlador puede usar una vista para mostrar una sola foto, otra para mostrar varias fotos y una tercera vista para habilitar usuarios para cargar nuevas fotos. Cada vista corresponde a una página web que la aplicación muestra al usuario, aunque puede mostrar diferentes datos. Por ejemplo, la vista **`PhotoDetails`** puede exponer diferentes fotos, dependiendo del parámetro de ID que reciba.

Al planificar las vistas, también se deben considerar las partes de la interfaz gráfica de usuario que aparecen en todas las páginas. Por ejemplo, el logotipo de la empresa, el menú principal del sitio, los enlaces a información legal y los controles de inicio de sesión pueden aparecer en todas las páginas mediante un ***Layout***.

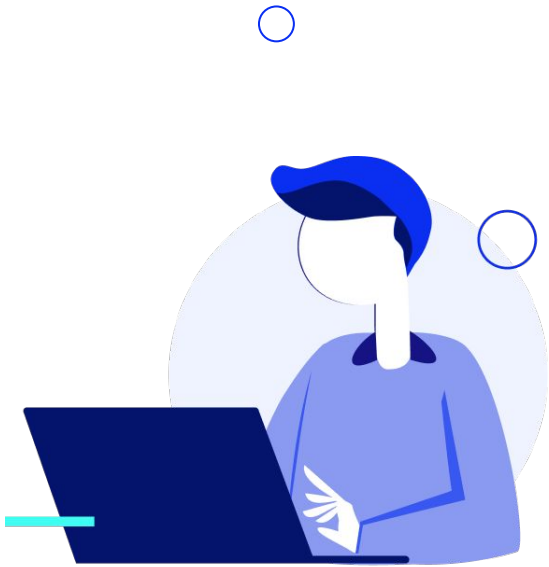

Adventure Works Photo Sharing
Log On
Home
Gallery
Slideshow
Photo
Add Comment

Arquitectura informacional

Information Architecture

La Arquitectura de la Información se aplica mediante:

- Planificación de una jerarquía **lógica.**
- Presentación de una jerarquía en los **controles de navegación.**
- Presentación de una jerarquía en la **URL.**



Arquitectura informacional

La arquitectura de la información es una estruc-
tura lógica para los objetos que administra la aplicación web. Se debe diseñar de manera que los usuarios puedan **encontrar contenido de manera rápida,** sin tener que comprender ningún aspecto de la aplicación.

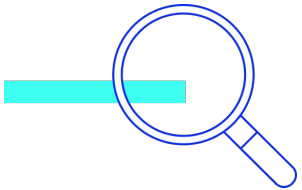
Los tipos de controles de navegación que se crean, dependen de cómo se espera que los usuarios encuentren la información. Por ejemplo, los usuarios deberían poder encontrar una respuesta técnica sobre su producto sin comprender la estructura de la base de datos o las clases que constituyen al modelo.

En el siguiente ejemplo, una empresa fabrica hornos de calefacción domésticos, que son instalados en los hogares por ingenieros de calefacción calificados. La aplicación web está destinada a permitir a los clientes **encontrar instrucciones, sugerencias y consejos.** Esta web también está destinada a permitir a los ingenieros **obtener la documentación técnica sobre la instalación y mantenimiento de hornos.**

El equipo de desarrollo ha identificado las siguientes historias de usuarios simples:

- Los **clientes** tienen cierto problema con sus calderas. Quieren encontrar una respuesta específica en las preguntas frecuentes. Conocen el nombre del producto de la caldera y el combustible, pero no el número de producto. Visitan la aplicación web y navegan hasta el nombre de la caldera. Hacen clic en el enlace de preguntas frecuentes para la caldera
y localizan la respuesta.

- Los **ingenieros** necesitan el último manual de instalación para una caldera. Saben el número de producto de la caldera, el nombre del producto y tipo de combustible. Visitan el sitio y navegan hasta el nombre de la caldera. Ellos hacen clic en el enlace de manuales y localizan el manual de instalación.



Presentar una jerarquía en URL

En las aplicaciones web de **MVC**, la configuración predeterminada es simple, se basa en **controladores, acciones y parámetros**.

A continuación, se muestra un ejemplo de **URL** que siguen este patrón predeterminado:

```
Controller/Action/Parameter
```

- *http://site/**InstallationManual/Details/3654***

Esta **URL** enlaza con el controlador llamado ***InstallationManual*** / acción **Detalles** / y muestra el manual con el parámetro ID **3654**.

En su lugar, se puede configurar **rutas más cortas. URL más fáciles de entender**, como la siguiente, que **refleja la jerarquía de información:**

- *http://site/**OilFired/HotBurner2000/***

Esta **URL** enlaza con el manual de instalación especificando el nombre del horno al que se refiere el manual.

educación IT
CODE YOUR FUTURE

¡Sigamos trabajando!