Aplicación para compartir fotos de Adventure Works (propuesta)

Documento de planificación detallado

*18 de noviembre 2020*

Autores: *Billy Vanegas*

# Introducción

El autor ha examinado el documento de investigación inicial. Según los casos de uso, los requisitos técnicos y otro contenido de ese documento, el autor ha creado los planes detallados a continuación. La junta ya acordó que la aplicación para compartir fotos se construirá como un sitio web basado en la tecnología ASP.NET MVC de Microsoft. Por lo tanto, los detalles presentados aquí incluyen los nombres y propiedades de las clases de modelo y los controladores que los desarrolladores deben crear. También se han identificado vistas y se han incluido diagramas de prototipo (wireframes [MockFlow](https://mockflow.com/)) para ayudar a visualizar la interfaz de usuario para partes importantes del sitio.

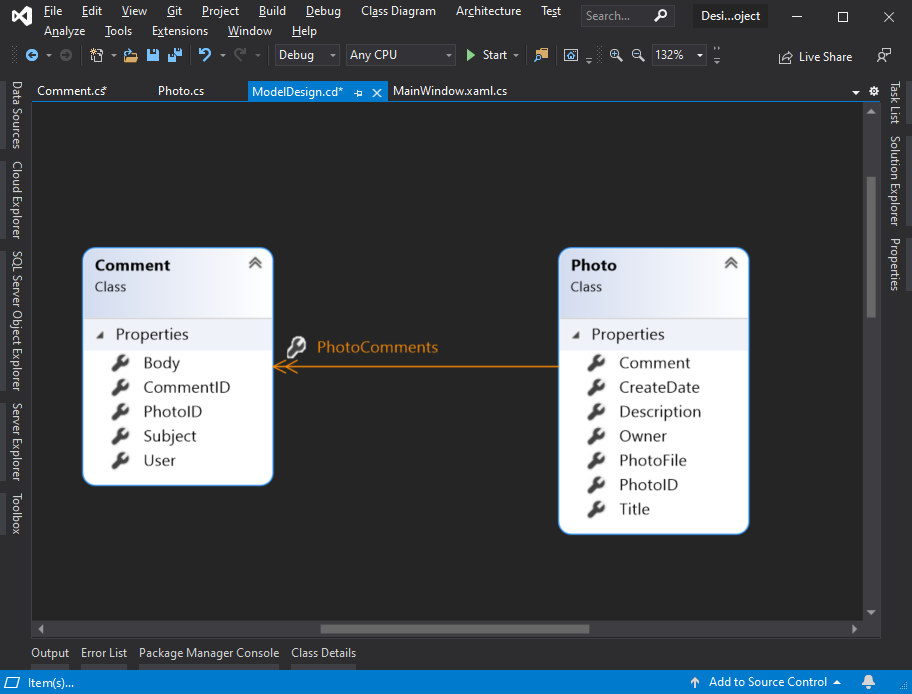
Es probable que el diseño de la aplicación evolucione a lo largo del proceso de desarrollo a medida que cambian los requisitos. El equipo de desarrollo adoptará prácticas ágiles para garantizar que dichos cambios se reflejen en el producto final. Por lo tanto, este documento no debe considerarse una definición completa de la aplicación final.

# Modelo MVC

Los desarrolladores crearán un modelo con las siguientes clases de modelo. Para cada clase de modelo, se han enumerado las propiedades y se han proporcionado descripciones.

This enables users to discuss

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Modelo de Clases | Descripción | Propiedades | Tipos de Datos |
| 1. | Photo | The photo model class represents  a photo that authenticated users  can upload to the website. | **PhotoID** | Integer |
| 2. |  | Title | String |
| 3. |  | PhotoFile | Binary |
| 4. |  | Description | String |
| 5. |  | CreateDate | DateTime |
| 6. |  | Owner | Integer |
| 7. | Comment | The comment model class represents  a comment that authenticated users  can add to photos This enables users  to discuss others’ photos Each  comment is associated with just one  photo. | **CommentID** | Integer |
| 8. |  | **User** | String |
| 9. |  | **Subject** | String |
| 10. |  | **Body** | String |
| 11. |  | **PhotoID** | Integer |
|  |  |  |  |



# Cotroladores MVC

Los desarrolladores crearán los siguientes controladores. Para cada controlador, se han enumerado las acciones y se han proporcionado descripciones.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Controlador | Acción | Descripción |
|  | **PhotoController** | DisplayGallery(GET) | The action runs when the user requests the Photo Gallery page. The action obtains all the photos from the database and passes them to the *DisplayGallery view.* |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  | DisplayRecent(GET) | This action is similar to the DisplayGallery action except that only the most recent photos are obtained from the database. This smaller collection of photos is passed to the *DisplayGallery view.* |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  | DisplayPhoto (GET) | This action runs when the user clicks a photo's “Details” link in a gallery. The action obtains full details of a single photo from the database and passes it to the *DisplayPhoto view.* |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  | AddPhoto (POST) | This action runs when the user clicks “Save” in the AddPhoto view. The action saves the file and details of the new photo to the database and redirects the user to the *DisplayGallery view.* |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  | AddPhoto (GET) | This action runs when the user clicks the “Add a Photo” link. The action creates a new instance of the Photo model class and passes it to the *AddPhoto view.* |
|  |  | DeletePhoto (GET) | This action runs when the user clicks a “Delete this Photo” link in the *DisplayPhoto view*. The action displays the *DeletePhoto view,* which requests confirmation for the deletion. |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  | DeletePhoto (POST) | This action runs when the user clicks “Delete” in the *DeletePhoto view*. The action deletes the current photo, with its associated comments from the database and redirects the user to the *DisplayGallery view*. |
|  |  |  |
|  |  |  |
|  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Controlador | Acción | Descripción |
|  | **CommentController** | DisplayComments (GET) | This action runs when the *DisplayPhoto view* is displayed. The action requires the current PhotoID as a parameter and uses it to get all the comments for the current photo from the database. The action returns the *DisplayComments partial view*. |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  | AddComment (GET) | This action runs when the user clicks the “Add a Comment” link in the *DisplayPhoto view*. The action creates a new instance of the Comment model class and sets its PhotoID to be the ID of the current photo. It passes this new comment to the *AddComment view*. |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  | AddComment (POST) | This action runs when the user clicks “Submit” in the *AddComment view*. The action saves the details of the new comment in the database and redirects the user to the *DisplayPhoto view.* |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |  |
|  |  |  |  |

# Vistas MVC

|  |  |  |  |
| --- | --- | --- | --- |
|  | Controlador | Vista | Descripción |
|  | PhotoController | DisplayGallery | This view displays a collection of photos in the thumbnail size. For each photo the Title, Owner, and Created Date values are displayed. |
|  |  |  |
|  |  |  |
|  | PhotoController | DisplayPhoto | This view displays a single photo in full size. The Title and Owner values appear above the photo. The Photo Name, Description, and other values appear beneath the photo. Under these details, all the comments for the current photo are listed with an “Add a Comment” link. |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  | PhotoController | AddPhoto | This view displays a form that the user can use to upload and describe a new photo. |
|  |  |  |
|  | PhotoController | DeletePhoto | This view displays a form that the user can use to confirm the deletion of a photo. The view displays details of the current photo such as its title and description. |
|  |  |  |
|  |  |  |
|  |  |  |
|  | CommentController | DisplayComments | This view displays a form that the user can use to confirm the deletion of a photo. The view displays details of the current photo such as its title and description. |
|  |  |  |
|  |  |  |
|  | CommentController | AddComment | This view displays a form that the user can use to create a new comment for a photo. |
|  |  |  |

|  |  |
| --- | --- |
| DisplayGalleryView | DisplayPhotoView |
|  |  |

# Recomendaciones de alojamiento

Dado que la aplicación para compartir fotos se desarrollará en ASP.NET Core MVC, debe estar alojada en un servidor web de Microsoft. El autor recomienda la siguiente configuración de alojamiento:

## Servidor web

*The author recommends using Microsoft Azure to host the Photo Sharing application. Microsoft Azure can host any ASP.NET website, including the ASP.NET Core MVC application proposed in this document. Scaling is very simple because Microsoft, not Adventure Works, is responsible for adding server resources at times of high traffic. Costs are minimal: they depend on the amount of data served to visitors but it is not necessary to maintain our own hardware.*

## Base de datos

*The author recommends using SQL Database, within Microsoft Azure, to host the Photo Sharing application underlying database. As for the web server, this recommendation ensures high-availability hosting for the database with good value for money. This makes particular sense if the web site is hosted in Microsoft Azure.*