Приложение для обмена фотографиями(Предложения)

*Документ подробного планирования*

*28.11.2022*

Автор: *Андрей Козырин*

# Введение

# Автор изучил первоначальный следственный документ. Основываясь на вариантах использования, технических требованиях и другом содержании в этом документе, автор создал подробные планы ниже. Правление уже согласовало, что приложение для обмена фотографиями будет создано как веб-сайт на основе технологии Microsoft ASP.NET MVC. Поэтому детали, представленные здесь, включают имена и свойства классов моделей и контроллеров, которые должны создать разработчики. Также были определены представления и включены каркасные схемы, помогающие представить пользовательский интерфейс для важных частей сайта.

# Дизайн приложения, вероятно, будет меняться в процессе разработки по мере изменения требований. Команда разработчиков будет применять методы Agile, чтобы такие изменения отражались в конечном продукте. Поэтому этот документ не следует рассматривать как полное определение конечного приложения.

# MVC Model

Разработчики создадут модель со следующими классами моделей. Для каждого класса моделей перечислены свойства и даны описания.

Table 1: MVC Model

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Model Class | Description | Properties | Data Types |
| 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. | CreatedDate | Date |
| 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 |

# MVC Controllers

Разработчики создадут следующие контроллеры. Для каждого контроллера перечислены действия и даны описания.

Table 2: MVC Controllers

|  |  |  |  |
| --- | --- | --- | --- |
|  | Controller | Action | Description |
| 1. | 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. |
| 2. | 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. |
| 3. | 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. |
| 4. | AddPhoto (GET) | This action runs when the user clicks an “Add a Photo” link. The action creates a new instance of the Photo model class and passes it to the AddPhoto view. |
| 5. | 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. |
| 6. | 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. |
| 7. | 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. |
| 8. | 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. |
| 9. | 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. |
| 10. | 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. |

# MVC Views

Разработчики создадут следующие представления. Каждое представление указано вместе с контроллером, с которым оно связано.

Table 3: MVC Views

|  |  |  |  |
| --- | --- | --- | --- |
|  | Controller | View | Description |
| 1. | 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. |
| 2. | 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. |
| 3. | AddPhoto | This view displays a form that the user can use to upload and describe a new photo |
| 4. | 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 |
| 5. | CommentController | DisplayComments | This partial view, which is used on the DisplayPhoto form, displays all the comments associated with the current photo |
| 6. | AddComment | This view displays a form that the user can use to create a new comment for a photo |

# Рекомендации по хостингу

## Поскольку приложение для обмена фотографиями будет разработано в ASP.NET Core MVC, оно должно быть размещено на веб-сервере Microsoft. Автор рекомендует следующую конфигурацию хостинга:

## Web Server

## Автор рекомендует использовать Microsoft Azure для размещения приложения для обмена фотографиями. Microsoft Azure может разместить любой веб-сайт ASP.NET, включая приложение ASP.NET Core MVC, предложенное в этом документе. Масштабирование очень просто, потому что Microsoft, а не Adventure Works, отвечает за добавление серверных ресурсов в периоды высокого трафика. Затраты минимальны: они зависят от объема данных, предоставляемых посетителям, но не требуют обслуживания собственного оборудования.

## Database

Автор рекомендует использовать базу данных SQL в Microsoft Azure для размещения базовой базы данных приложения для обмена фотографиями. Что касается веб-сервера, эта рекомендация обеспечивает высоко доступный хостинг для базы данных с хорошим соотношением цены и качества. Это имеет особое значение, если веб-сайт размещен в Microsoft Azure.