Model Validation

Goals

During the Upload process, the model was not validated at all. We gave for granted that the data sent by the user was correct. Of course, we should not trust users, because they may not know what the rules are. Our goal for this lab is to validate the data, reject it if it's not valid, show error messages and give the users the chance to fix their errors once they know what they did wrong.

We're going to: - Add rules for validation - Add checks in the page methods before submitting to the service - Add messages for validation

Add rules for validation

Instead of using `Validation Attributes`, included in the `System.ComponentModel.DataAnnotations` namespace, we're going to use `FluntValidation`. We'll do this in order to maintain our concerns neatly separated. We'll build a `PhotoValidator` class which will contain the `Rules` to apply during validation.

- Add a Reference to `FluentValidation` in the `PhotoSharingApplication.Core` project
- In the same project, add a `Validators` folder
- In the `Validators` folder, add a `PhotoValidator` class
    - Let the class extend `AbstractValidator<Photo>`
    - Add a Constructor
    - Add a Rule for the Title so that it's not Empty and its maximum length is 100
    - Add a Rule for the Description so that it's not Empty and its maximum length is 250
    - Add a Rule for the PhotoFile so that it's not Empty

The code becomes:

```
using FluentValidation;
using PhotoSharingApplication.Core.Entities;

namespace PhotoSharingApplication.Core.Validators;

public class PhotoValidator : AbstractValidator<Photo> {
    public PhotoValidator() {
        RuleFor(photo => photo.Title).NotEmpty().MaximumLength(100);
        RuleFor(photo => photo.Description).NotEmpty().MaximumLength(250);
        RuleFor(photo => photo.PhotoFile).NotEmpty();
    }
}
```

- In the `PhotosService` class:
    - Add a `PhotoValidator` field
    - In the constructor, accept a `PhotoValidator` parameter and initialize the `PhotoValidator` field with its value
    - In the `AddPhotoAsync` method, before sending the photo to the repository, invoke the `ValidateAndThrow` method on the `PhotoValidator` field
- In the `PhotoSharingApplication.Web` project, add the `FluentValidation.AspNetCore` NuGet Package
- In the `ServiceCollectionExtensions.cs` file, add the following code:

```
services.AddFluentValidation(fv => fv.RegisterValidatorsFromAssemblyContaining<PhotoValidator>());
```

Add checks in the page methods before submitting to the service

Right now, if we try to upload a photo without inserting a title, the validation will fail. We need to add code to the OnPost of the `Upload` page to check and show the error messages if we get an exception from the validator.

ASP.NET gives us a `ModelState` property in our page which is set by the `ModelBinder` during the binding phase. Right now we defined the `PhotoFile` property as a non nullable array of byte. The `ModelState.IsValid` is therefore set to `false` because during the `post` our array of byte is still empty. So the first thing we want to do is to remove the `PhotoFile` error. Then we can check and eventually return the Page if it's still `false`.
When we invoke the service, the `ValidateAndThrow` method will throw an exception if the validation fails. That's why we want to wrap it in a try catch block, returning a `Page` if we catch the exception.

```
public async Task<IActionResult> OnPostAsync() {
    ModelState.Remove("Photo.PhotoFile");

    if (!ModelState.IsValid) {
        return Page();
    }
    using (var memoryStream = new MemoryStream()) {
        await FormFile.CopyToAsync(memoryStream);
        Photo.PhotoFile = memoryStream.ToArray();
        Photo.ContentType = FormFile.ContentType;
    }

    try {
        await photosService.AddPhotoAsync(Photo);
    } catch (FluentValidation.ValidationException) {
        return Page();
    }
    return RedirectToPage("./Index");
}
```

So far, if we try to upload a picture without a title, we should see the upload page again. But if we insert a title, the validation will pass and we can go to the `Index` page. We don't see any explanation for the error message, though. Let's fix that.

Add messages for validation

We need to add a couple of `Tag Helpers` to the `Upload.cshtml` file so that the error messages are shown when necessary.
The code becomes:

```
@page
@model PhotoSharingApplication.Web.Pages.Photos.UploadModel
<form method="post" enctype="multipart/form-data">
    <div asp-validation-summary="ModelOnly" class="text-danger"></div>
    <div>
        <input asp-for="Photo.Title" />
        <span asp-validation-for="Photo.Title" class="text-danger"></span>
    </div>
    <div>
        <textarea asp-for="Photo.Description"></textarea>
        <span asp-validation-for="Photo.Description" class="text-danger"></span>
    </div>
    <div>
        <input asp-for="FormFile" type="file">
        <span asp-validation-for="FormFile" class="text-danger"></span>
    </div>
    <div>
        <input type="submit" />
    </div>
</form>
```

By running the application now and submitting the form without a title, we should see the error message for the title.

Resources

- https://docs.microsoft.com/en-us/aspnet/core/tutorials/razor-pages/validation?view=aspnetcore-6.0&tabs=visual-studio
- https://fluentvalidation.net/
- https://docs.fluentvalidation.net/en/latest/aspnet.html