WebApi

Goals

- Use an actual table of our database to store and retrieve the comments.
- Expose the data through a Web Service by implementing a Web Api controller.
- Call the API from the Blazor Component.

We're going to: - Change the DB - Create Service and Repository for the Server Side - Expose the DB through a WebApi - Connect the BlazorComponent with the API

Change the DB

- In the `PhotoSharingApplication.Shared` project, under the `Entities` folder, add a `Comment` class with an `Id`, `Title`, `Body` and `PhotoId`

```
namespace PhotoSharingApplication.Shared.Entities;

public class Comment {
    public int Id { get; set; }
    public string Title { get; set; } = string.Empty;
    public string Body { get; set; } = string.Empty;
    public int PhotoId { get; set; }
}
```

- Add a property to the `Photo` Entity to establish the one to many relationship between the two entities.

```
namespace PhotoSharingApplication.Shared.Entities;

public class Photo {
    public int Id { get; set; }
    public string Title { get; set; } = string.Empty;
    public string Description { get; set; } = string.Empty;
    public byte[] PhotoFile { get; set; }
    public string ContentType { get; set; } = string.Empty;
    public List<Comment>? Comments { get; set; }
}
```

- Add the `DbSet<Comment> Comments` property on the `PhotoSharingDbContext` located under the `Data` folder of the `PhotoSharingApplication.Infrastructure` project.

```
public DbSet<Comment> Comments { get; set; }
```

- Configure the `Comment` Entity of the `DbContext` to ensure that the `Title` has a maximum length of 100 characters and the `Body` has a maximum length of 250 characters.

```
protected override void OnModelCreating(ModelBuilder modelBuilder) {
    modelBuilder.Entity<Photo>()
        .Property(b => b.Title)
        .HasMaxLength(100);
    modelBuilder.Entity<Photo>()
        .Property(b => b.Description)
        .HasMaxLength(250);
    modelBuilder.Entity<Photo>()
        .Property(b => b.ContentType)
        .HasMaxLength(30);

    modelBuilder.Entity<Comment>()
        .Property(b => b.Title)
        .HasMaxLength(100);
    modelBuilder.Entity<Comment>()
        .Property(b => b.Body)
        .HasMaxLength(250);
}
```

- Add the migration to the database by running the following command:

```
Add-Migration CommentsTable
```

- Update the DB by running the following command

```
Update-Database
```

Create Service and Repository for the Server Side

Add the interfaces and classes for the `CommentsService` and `CommentsRepository`. Let the `CommentsRepository` work with the `PhotoSharingDbContext`. - In the `PhotoSharingApplication.Core` project, under the `Interfaces` folder, add an `ICommentsService` interface. Add methods to add a comment, get a comment given its id and get a list of comments given a photo id.

```
 using PhotoSharingApplication.Shared.Entities;

namespace PhotoSharingApplication.Core.Interfaces;

public interface ICommentsService {
    Task<IEnumerable<Comment>> GetCommentsForPhotoAsync(int photoId);
    Task<Comment?> GetCommentByIdAsync(int id);
    Task<Comment> AddCommentAsync(Comment comment);
}
```

- In the `PhotoSharingApplication.Core` project, under the `Interfaces` folder, add an `ICommentsRRepository` interface. Add methods to add a comment, get a comment given its id and get a list of comments given a photo id.

```
 using PhotoSharingApplication.Shared.Entities;

namespace PhotoSharingApplication.Core.Interfaces;

public interface ICommentsRepository {
    Task<IEnumerable<Comment>> GetCommentsForPhotoAsync(int photoId);
    Task<Comment?> GetCommentByIdAsync(int id);
    Task<Comment> AddCommentAsync(Comment comment);
}
```

- In the `PhotoSharingApplication.Core` project, under the `Services` folder, add a `CommentsService` class that implements the `ICommentsService` interface.

```
 using FluentValidation;
using PhotoSharingApplication.Shared.Entities;
using PhotoSharingApplication.Core.Interfaces;
using PhotoSharingApplication.Shared.Validators;

namespace PhotoSharingApplication.Core.Services;

public class CommentsService : ICommentsService {
    private readonly ICommentsRepository repository;
    private readonly CommentValidator validator;

    public CommentsService(ICommentsRepository repository, CommentValidator validator) {
        this.repository = repository;
        this.validator = validator;
    }
    public Task<Comment> AddCommentAsync(Comment comment) {
        validator.ValidateAndThrow(comment);
        return repository.AddCommentAsync(comment);
    }

    public Task<Comment?> GetCommentByIdAsync(int id) => repository.GetCommentByIdAsync(id);

    public Task<IEnumerable<Comment>> GetCommentsForPhotoAsync(int photoId) => repository.GetCommentsForPhotoAsync(photoId);
}
```

- In the `PhotoSharingApplication.Infrastructure` project, under the `Repositories` folder, add a `CommentsRepository` class that implements the

`ICommentsRepository` interface.

```
using Microsoft.EntityFrameworkCore;
using PhotoSharingApplication.Shared.Entities;
using PhotoSharingApplication.Core.Interfaces;
using PhotoSharingApplication.Infrastructure.Data;

namespace PhotoSharingApplication.Infrastructure.Repositories;

public class CommentsRepositoryEF : ICommentsRepository {
    private readonly PhotoSharingDbContext dbContext;

    public CommentsRepositoryEF(PhotoSharingDbContext dbContext) {
        this.dbContext = dbContext;
    }
    public async Task<Comment> AddCommentAsync(Comment comment) {
        dbContext.Comments.Add(comment);
        await dbContext.SaveChangesAsync();
        return comment;
    }

    public async Task<Comment?> GetCommentByIdAsync(int id) => await dbContext.Comments.FirstOrDefaultAsync(c => c.Id == id);

    public async Task<IEnumerable<Comment>> GetCommentsForPhotoAsync(int photoId) => await dbContext.Comments.Where(c => c.PhotoId =
}
```

- Register them in the `ServiceCollectionExtensions.cs` file of the `PhotoSharingApplication.Web` project.

```
services.AddScoped<Core.Interfaces.ICommentsRepository, Infrastructure.Repositories.CommentsRepositoryEF>();
services.AddScoped<Core.Interfaces.ICommentsService, Core.Services.CommentsService>();
```

Expose the DB through a WebApi

- Add a `CommentsController` Controller to the `PhotoSharingApplication.Web` project, under a new `Controllers` folder and create actions to retrieve and add comments by using an `ICommentsService` .
  - Add a constructor to the `CommentsController` that accepts an `ICommentsService` as a parameter and save the parameter in a private readonly field.
  - Add a `[HttpGet]` action to the `CommentsController` that accepts a `int` as a parameter and returns a `Task<ActionResult<IEnumerable<Comment>>>` that calls the `GetCommentsForPhotoAsync` method of the `ICommentsService` and passes the `int` as a parameter. Map the route to `/api/Photos/{photoId}/Comments`
  - Add a `[HttpPost]` action to the `CommentsController` that accepts a `Comment` as a parameter and returns a `Task<ActionResult<Comment>>` that calls the `AddCommentAsync` method of the `ICommentsService` and passes the `Comment` as a parameter. Return a `CreatedAtAction` result with the `Comment` as a parameter.
  - Add a `[HttpGet]` action to the `CommentsController` that accepts a `int` as a parameter and returns a `Task<ActionResult<Comment>>` that calls the `GetCommentByIdAsync` method of the `ICommentsService` and passes the `int` as a parameter. Map the route to `/api/Comments/{id}`

```
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using PhotoSharingApplication.Shared.Entities;
using PhotoSharingApplication.Core.Interfaces;

namespace PhotoSharingApplication.Web.Controllers;

[Route("api/[controller]")]
[ApiController]
public class CommentsController : ControllerBase {
    private readonly ICommentsService service;

    public CommentsController(ICommentsService service) => this.service = service;

    [HttpGet("/api/Photos/{photoId}/Comments")]
    public async Task<ActionResult<IEnumerable<Comment>>> GetCommentsForPhoto(int photoId) => (await service.GetCommentsForPhotoAsyn

    [HttpGet("{id:int}")]
    public async Task<ActionResult<Comment>> GetCommentById(int id) {
        Comment? comment = await service.GetCommentByIdAsync(id);
        if (comment is null) return NotFound();
        return comment;
    }

    [HttpPost]
    public async Task<ActionResult<Comment>> AddComment(Comment comment) {
        await service.AddCommentAsync(comment);
        return CreatedAtAction(nameof(GetCommentById), new { id = comment.Id }, comment);
    }
}
```

- Map the Controller routes in the `Program.cs` file of the `PhotoSharingApplication.Web` project, between the `MinimalApi` and the `MapRazorPages`.

```
services.MapControllers();
```

- Optional: add some comments on the DB and test if your controller works by navigating to the route to get the comments for one photo.
- Optional: add OpenAPI support to the project and try the Api by navigating to the swagger ui.

```
services.AddControllers();
services.AddEndpointsApiExplorer();
services.AddSwaggerGen();
```

At this point, you should be able to navigate to the `/api/Photos/{photoId}/Comments` route and see the comments for the photo.

Connect the BlazorComponent with the API

- In the `PhotoSharingApplication.Blazor.Infrastructure.Repositories` folder, create a `CommentsRepositoryHttp.cs` file with a `CommentsRepositoryHttp` class that implements the `PhotoSharingApplication.Blazor.Core.Interfaces.ICommentsRepository` interface.
    - Accept an `HttpClient` parameter in the constructor and use it in each method.

```
using PhotoSharingApplication.Blazor.Core.Interfaces;
using PhotoSharingApplication.Shared.Entities;
using System.Net.Http.Json;
using System.Text;
using System.Text.Json;
using static System.Net.Mime.MediaTypeNames;

namespace PhotoSharingApplication.Blazor.Infrastructure.Repositories;

public class CommentsRepositoryHttp : ICommentsRepository {
    private readonly HttpClient httpClient;

    public CommentsRepositoryHttp(HttpClient httpClient) {
        this.httpClient = httpClient;
    }
    public async Task<Comment> AddCommentAsync(Comment comment) {
        var commentJson = new StringContent(JsonSerializer.Serialize(comment), Encoding.UTF8, Application.Json);

        using var httpResponseMessage = await httpClient.PostAsync("/api/Comments", commentJson);

        return await httpResponseMessage.Content.ReadFromJsonAsync<Comment>();
    }

    public async Task<Comment?> GetCommentByIdAsync(int id) => await httpClient.GetFromJsonAsync<Comment>($"/api/Comments/{id}");

    public async Task<IEnumerable<Comment>> GetCommentsForPhotoAsync(int photoId) => await httpClient.GetFromJsonAsync<IEnumerable<C
}
```

- Register the `CommentsRepositoryHttp` class as a Scoped service in the `Program.cs` file of the Blazor client application, instead of the `CommentsRepositoryList`.

```
builder.Services.AddScoped<ICommentsRepository, CommentsRepositoryHttp>();
```

- In the `PhotoSharingApplication.Web.Controllers` folder add a new `CommentsRepositoryApi` class that implements the `PhotoSharingApplication.Blazor.Core.Interfaces.ICommentsRepository` interface.
  - Accept a `CommentsController` parameter in the constructor and use it in each method.

```
using PhotoSharingApplication.Blazor.Core.Interfaces;
using PhotoSharingApplication.Shared.Entities;

namespace PhotoSharingApplication.Web.Controllers;

public class CommentsRepositoryApi : ICommentsRepository {
    private readonly CommentsController controller;

    public CommentsRepositoryApi(CommentsController controller) {
        this.controller = controller;
    }
    public async Task<Comment> AddCommentAsync(Comment comment) => (await controller.AddComment(comment)).Value;

    public async Task<Comment?> GetCommentByIdAsync(int id) => (await controller.GetCommentById(id)).Value;

    public async Task<IEnumerable<Comment>> GetCommentsForPhotoAsync(int photoId) => (await controller.GetCommentsForPhoto(photoId))
}
```

- Register the `CommentsRepositoryApi` class as a Scoped service in the `ServiceCollectionExtensions.cs` file of the Web project.
- Register the `CommentsController` as a Scoped Service in the `ServiceCollectionExtensions.cs` file of the Web project.

```
services.AddScoped<CommentsController>();
services.AddScoped<PhotoSharingApplication.Blazor.Core.Interfaces.ICommentsRepository, PhotoSharingApplication.Web.Controllers.Comme
services.AddScoped<PhotoSharingApplication.Blazor.Core.Interfaces.ICommentsService, PhotoSharingApplication.Core.Services.Client.Com
```

At this point, running the application and navigating to the details of a photo should show the comments for that photo. Adding a new comment should work and the new comment should be added to the database.

Resources

- https://docs.microsoft.com/en-us/aspnet/core/tutorials/first-web-api?view=aspnetcore-6.0&tabs=visual-studio
- https://docs.microsoft.com/en-us/aspnet/core/fundamentals/http-requests?view=aspnetcore-6.0
- https://jonhilton.net/blazor-prerendering-net6/
- https://jonhilton.net/blazor-wasm-prerendering/
- https://jonhilton.net/blazor-wasm-prerendering-missing-http-client/
- https://docs.microsoft.com/en-us/aspnet/core/tutorials/getting-started-with-swashbuckle?view=aspnetcore-6.0&tabs=visual-studio