Librería virtual

EJERCICIO DE ENTRENAMIENTO

ASP.NET Core | Entity Framework Core

Aspectos generales

Para la realización de este ejercicio se debe crear un proyecto de tipo **ASP.NET Core Web API,** donde se implemente un conjunto de apis que serán descritas posteriormente. Usted debe usar **SQL Server** como Sistema Gestor de Bases de Datos y **Entity Framework Core** como **ORM** (Object-Relational Mapping en inglés). La BD debe ser generada a partir de las clases/modelos, es decir, modelada a través de clases de C# (Code Fisrt). Los objetos devueltos por las apis, así como los recibidos desde los clientes, estarán en formato JSON.

Entidades del sistema

El backend a implementar consiste en una librería virtual en la cual se almacenan libros y se pueden registrar usuarios que podrán suscribirse a las publicaciones de sus autores preferidos, para ser notificados cuando hayan nuevos libros disponibles. Además los usuarios pueden hacer reviews sobre los libros en el cual puede describir mediante un texto su criterio sobre el libro asi como especificar la cantidad de estrellas que le otorga entre 1 y 5 (son numeros enteros, no se aceptan decimales como por ejemplo 1.5)

De los libros almacenados se conoce:

- Título
- Autor
- Nombre de la editorial
- Cantidad de páginas
- Fecha de publicación
- Enlace de descarga (URL)
- ISBN (International Standard Book Number)
- Calificación (representa el valor promedio de los reviews realizados por los usuarios, este valor es manejado internamente por la aplicacion)

De los autores se conoce:

- Id
- Nombre
- Nacionalidad
- Fecha de nacimiento

De los usuarios registrados se conoce:

- Id (Guid)
- Nombre
- Email
- URL de la imagen de su perfil

De los reviews de los libros de conoce:

- Id

- Usuario
- Fecha (en la que se realizó el review)
- Opinion (texto del usuario, es opcional)
- Calificación (valor en el rango del 1 – 5, se recomienda usar un enum)

Nota: Puede crear las entidades que desee y/o agregar nuevos campos a las anteriores si lo considera conveniente.

Apis

Los aspectos no detallados en las descripciones de las apis, quedan a su consideración. Por ejemplo, los verbos http (GET, POST, etc), los tipos de respuestas (200, 204, 400, 404, etc) y cualquier otro no mencionado. Debe implementar un middleware que capture cualquier excepción no manejada en las apis y devuelva una respuesta con código 500 y que el cuerpo sea un objeto como el siguiente: **{ "message": <some text>, "description": <exception message> }**. El valor de "message" será definido en la configuración de la aplicación (archivo appsettings.json)

1) Registrar un nuevo usuario
   - URL: /api/v1.0/library/users
   - Recibe en el body un json con: id, nombre, email, imagen
   - Excepto la imagen, los demás campos son requeridos

2) Actualizar imagen del usuario
   - URL: /api/v1.0/library/users/{**userId**}
   - Cambia la imagen del usuario con id **userId** por una recibida en el body. Para esto se pasará un json con una propiedad que contenga la nueva URL

3) Eliminar un usuario
   - URL: /api/v1.0/library/users/{**userId**}
   - **userId** es el id del usuario a eliminar

4) Suscribirse a un autor
   - URL: /api/v1.0/library/users/{**userId**}/subscribe-to-author/{**authorId**}
   - Crea una suscripción del usuario con id **userId** al autor con id **authorId**

5) Eliminar suscripción
   - URL: /api/v1.0/library/users/{**userId**}/subscribe-to-author/{**authorId**}

6) Listado de usuarios
   - URL: /api/v1.0/library/users?**offset**=0&**limit**=50

- Devuelve un json que representa una página de usuarios. El json debe tener una propiedad que indique el total de usuarios y otra propiedad con una lista de usuarios. Para tomar los usuarios se deben ignorar los primeros "**offset**" y entonces tomar los siguientes "**limit**"
- De cada usuario se debe devolver: id, nombre, email, imagen, fecha de registro en el sistema, cantidad de autores a los que está suscrito

7) Ingresar un nuevo autor
   - URL: /api/v1.0/library/authors
   - Recibe en el body un json con: nombre, nacionalidad, fecha de nacimiento. Todos los campos son requeridos.

8) Obtener detalles de un autor
   - URL: /api/v1.0/library/authors/{**authorId**}
   - Se debe devolver un json con: nombre, nacionalidad, fecha de nacimiento, cantidad de usuarios suscritos, colección de libros publicados.
   - De los libros publicados solo se incluye: isbn, título y fecha de publicación

9) Buscar libros
   - URL: /api/v1.0/library/books?**authorId**=null&**editorialName**=null&**before**=null&**after**=null&**offset**=0&**limit**=50&sort=(null|true|false)
   - Obtiene la colección de libros que resultan de aplicar los filtros siguientes:
   - **authorId:** libros cuyo autor tenga el id especificado.
   - **editorialName:** libros cuya editorial tenga el nombre especificado.
   - **before:** libros publicados antes de la fecha especificada.
   - **after:** libros publicados después de la fecha especificada.
   - **offset** y **limit** para paginación.
   - sort si es igual a null no aplica, si es true entonces retorna los libros ascendentemente por su calificación, en caso contrario descendente
   - De cada libro que pase todos los filtros, se debe devolver: título, nombre del autor, nombre de la editorial, isbn.
   - Cada uno de los filtros es opcional. Si alguno no es especificado, vale null, entonces no se aplica.

10) Ingresar nuevo libro
    - URL: /api/v1.0/library/authors/{**authorId**}/books
    - Añade un nuevo libro cuyo autor tiene como id **authorId**
    - Recibe en el body un json con: título, nombre de editorial, cantidad de páginas, fecha de publicación, isbn, enlace de descarga. Todos los campos son requeridos.
    - Se debe notificar, vía email, acerca del nuevo libro a todos los usuarios suscritos al autor del mismo. Para el envío de emails se proporciona un servicio, que en realidad simula el envío.
11) Buscar reviews de un libro

- URL:/api/v1.0/library/books/{**bookId**}/reviews?**reviewType**=(null|1-5)&**sort**=(null|true|false) **offset**=0&**limit**=50
- A la hora de filtar si **reviewType** es null entonces se toman en cuenta todos los reviews sin importar el valor de la estrella, en caso contrario se toman los reviews cuya calificacion sea igual a **reviewType**. En el caso de **sort** si el valor es null no se aplica este filtro, si es true entonces se retornan los reviews de manera ascendente por la fecha creacion del mismo, en caso contrario de manera descendente. Los valores **offset** y **limit** son utilizados para el mecanismo de paginación.
- Retorna los reviews que cumplen las condiciones previamentes descritas

12) Agregar review de un libro
- URL: /api/v1.0/library/books/{**bookId**}/reviews/from/users/{**userId**}
- Recibe en el body el texto del review del usuario y la calificación que le otorga el usuario al libro
- Tomar en cuenta que en este proceso se debe actualizar el valor de la calificación de la entidad libro cuyo id sea **bookId**

Servicio que simula el envío de emails

```
public interface IEmailSender
{
    Task SendEmailAsync(string fromAddress,
        string destinationAddress,
        string subject,
        string textMessage);
}

public class EmailSender : IEmailSender
{
    public async Task SendEmailAsync(string fromAddress,
        string destinationAddress,
```

```
        string subject,
        string textMessage)
    {
        //Sending email...
        await Task.CompletedTask;
    }
}
```

Debe usar el servicio anterior para simular los envíos de emails. La dirección del remitente será virtual.library@email.com y el asunto, “Nuevo libro de interés para usted”. El cuerpo del email lo define a su gusto. Tanto el asunto como la dirección del remitente deben ser establecidos en las configuraciones de la aplicación (archivo appsettings.json)

Importante

Es importante que tenga en cuenta el uso de buenas prácticas, principalmente en la realización de consultas a la BD, creación de índices adecuados y en el uso de la programación asíncrona (async/await). También son de importancia el manejo de errores, validación de los datos y el uso de Data Transfer Objects.

https://docs.microsoft.com/en-us/ef/core/performance/

https://docs.microsoft.com/en-us/azure/architecture/best-practices/index-best-practices

https://docs.microsoft.com/en-us/aspnet/core/performance/performance-best-practices?view=aspnetcore-5.0