EVALUACIÓN PRÁCTICA DE CONOCIMIENTOS PARA EL PERFIL FULLSTACK

Objetivo

Evaluar las capacidades del candidato para diseñar e implementar un sistema de blogging que permita la creación, edición y gestión de publicaciones, comentarios, seguidores, y notificaciones, utilizando una arquitectura basada en .NET para el back-end y Angular para el front-end.

Contexto del Proyecto

Estás trabajando en el desarrollo de un sistema de gestión de contenido para una plataforma de blogging. El sistema debe permitir a los usuarios crear y editar publicaciones, comentar en las publicaciones, seguir a otros usuarios y recibir notificaciones sobre nuevas publicaciones y comentarios.

Back-end (.NET):

Entidades (posiblemente):

1. **User (Usuario):**
    - Propiedades: UserId, Username, Email, PasswordHash.
2. **Post (Publicación):**
    - Propiedades: PostId, Title, Content, DatePublished, UserId (autor de la publicación).
3. **Comment (Comentario):**
    - Propiedades: CommentId, Text, DatePosted, UserId (autor del comentario), PostId.
4. **Follower (Seguidor):**
    - Propiedades: FollowerId, UserId (seguidor), FollowedUserId (usuario seguido).
5. **Notification (Notificación):**
    - Propiedades: NotificationId, Text, DateSent, UserId (destinatario de la notificación), RelatedPostId, RelatedCommentId.

Controladores (API):

1. **UserController:**
    - Métodos:
        - GET /api/users: Obtener la lista de usuarios.
        - GET /api/users/{id}: Obtener detalles de un usuario por su ID.

- POST /api/users: Crear un nuevo usuario.
- PUT /api/users/{id}: Actualizar un usuario existente.
- DELETE /api/users/{id}: Eliminar un usuario.

2. **PostController:**
   - Métodos:
     - GET /api/posts: Obtener la lista de publicaciones.
     - GET /api/posts/{id}: Obtener detalles de una publicación por su ID.
     - POST /api/posts: Crear una nueva publicación.
     - PUT /api/posts/{id}: Actualizar una publicación existente.
     - DELETE /api/posts/{id}: Eliminar una publicación.
3. **CommentController:**
   - Métodos:
     - GET /api/comments: Obtener la lista de comentarios.
     - GET /api/comments/{id}: Obtener detalles de un comentario por su ID.
     - POST /api/comments: Crear un nuevo comentario.
     - PUT /api/comments/{id}: Actualizar un comentario existente.
     - DELETE /api/comments/{id}: Eliminar un comentario.
4. **FollowerController:**
   - Métodos:
     - GET /api/followers/{userId}: Obtener la lista de seguidores de un usuario.
     - GET /api/following/{userId}: Obtener la lista de usuarios que un usuario sigue.
     - POST /api/followers: Seguir a un usuario.
     - DELETE /api/followers/{followerId}: Dejar de seguir a un usuario.
5. **NotificationController:**
   - Métodos:
     - GET /api/notifications/{userId}: Obtener la lista de notificaciones para un usuario.
     - PUT /api/notifications/{id}: Marcar una notificación como leída.
     - DELETE /api/notifications/{id}: Eliminar una notificación.

Servicios:

1. **UserService:**
    - Métodos:
        - GetAllUsers(): Obtener todos los usuarios.
        - GetUserById(id): Obtener detalles de un usuario por su ID.
        - CreateUser(user): Crear un nuevo usuario.
        - UpdateUser(id, user): Actualizar un usuario existente.
        - DeleteUser(id): Eliminar un usuario.
2. **PostService:**
    - Métodos:
        - GetAllPosts(): Obtener todas las publicaciones.
        - GetPostById(id): Obtener detalles de una publicación por su ID.
        - CreatePost(post): Crear una nueva publicación.
        - UpdatePost(id, post): Actualizar una publicación existente.
        - DeletePost(id): Eliminar una publicación.
3. **CommentService:**
    - Métodos:
        - GetAllComments(): Obtener todos los comentarios.
        - GetCommentById(id): Obtener detalles de un comentario por su ID.
        - CreateComment(comment): Crear un nuevo comentario.
        - UpdateComment(id, comment): Actualizar un comentario existente.
        - DeleteComment(id): Eliminar un comentario.
4. **FollowerService:**
    - Métodos:
        - GetFollowers(userId): Obtener la lista de seguidores de un usuario.
        - GetFollowing(userId): Obtener la lista de usuarios que un usuario sigue.
        - FollowUser(followedUserId): Seguir a un usuario.
        - UnfollowUser(followerId): Dejar de seguir a un usuario.
5. **NotificationService:**

- Métodos:
  - GetNotifications(userId): Obtener la lista de notificaciones para un usuario.
  - MarkNotificationAsRead(id): Marcar una notificación como leída.
  - DeleteNotification(id): Eliminar una notificación.

Front-end (Angular):

Componentes:

1. **UserListComponent:**
   - Mostrará la lista de usuarios.
   - Permitirá la navegación a la vista detallada de un usuario.
2. **UserDetailsComponent:**
   - Mostrará los detalles de un usuario, incluyendo la lista de seguidores y seguidos.
   - Permitirá la actualización y eliminación de un usuario (solo para el usuario actual).
3. **UserFormComponent:**
   - Permitirá la creación y edición de usuarios.
4. **PostListComponent:**
   - Mostrará la lista de publicaciones, con información sobre el autor y la cantidad de comentarios.
   - Permitirá la navegación a la vista detallada de una publicación.
5. **PostDetailsComponent:**
   - Mostrará los detalles de una publicación, incluyendo la lista de comentarios.
   - Permitirá la actualización y eliminación de una publicación (solo para el autor).
6. **PostFormComponent:**
   - Permitirá la creación y edición de publicaciones.
7. **CommentListComponent:**
   - Mostrará la lista de comentarios en una publicación.
   - Permitirá la navegación a la vista detallada de un comentario.
8. **CommentDetailsComponent:**
   - Mostrará los detalles de un comentario.

- Permitirá la actualización y eliminación de un comentario (solo para el autor).

9. **CommentFormComponent:**
    - Permitirá la creación y edición de comentarios.
10. **NotificationListComponent:**
    - Mostrará la lista de notificaciones para el usuario actual.
    - Permitirá marcar una notificación como leída y eliminar notificaciones.

Servicios:

1. **UserService:**
    - Se comunicará con el back-end para realizar operaciones CRUD de usuarios.
2. **PostService:**
    - Se comunicará con el back-end para realizar operaciones CRUD de publicaciones.
3. **CommentService:**
    - Se comunicará con el back-end para realizar operaciones CRUD de comentarios.
4. **FollowerService:**
    - Se comunicará con el back-end para realizar operaciones relacionadas con seguir y dejar de seguir usuarios.
5. **NotificationService:**
    - Se comunicará con el back-end para obtener notificaciones y realizar operaciones relacionadas con su gestión.

Core del Negocio:

- Implementar la lógica para gestionar la relación entre usuarios, publicaciones, comentarios y seguidores.
- Utilizar tokens JWT para la autenticación y protección de rutas.
- Definir roles y permisos para ciertas acciones, como la eliminación de publicaciones.
- Garantizar que los usuarios solo puedan actualizar o eliminar sus propias publicaciones y comentarios.
- Desarrollar una interfaz de usuario que permita a los usuarios explorar publicaciones, interactuar mediante comentarios y seguir a otros usuarios.

Entregables

- Proyecto Web App (WebAPP_Nombre_Apellido_añomesdia)
- Proyecto Web API (WebAPI_Nombre_Apellido_ añomesdia) (Agregar una carpeta “SCRIPTS_BD” con los scripts de creación de BD, tablas)