Materia:	Programación IV
Nivel:	4º Cuatrimestre
Tipo de Examen:	Trabajo Práctico. Corresponde a Segundo parcial

TP #2 - Red social

Enunciado/s:

- Se debe realizar la aplicación “Red social”
- La forma de corrección será por sprint de una semana.
- La aplicación debe permitir a nuestros usuarios hacer publicaciones y reaccionar, y comentar las mismas.
- Cada usuario debe poder ver y editar su perfil.
- Solo se puede acceder a la aplicación estando registrado o logueado.
- La aplicación debe tener un diseño trabajado, no HTML plano. No utilizar alert(), sinó modales.

Contenido de la aplicación

La “Red social” será una aplicación dividida en dos proyectos, uno frontend y otro backend. Deberá contar con los siguientes puntos:

1. Aplicación frontend en Angular.
2. Servidor en NestJS.
3. Base de datos en MongoDB.
4. Login y registro de usuarios contra el servidor en Nest propio. Se utilizará JWT.
5. Respuestas de la api con status correctos. 400 bad request, 401 Unauthorized, 201 created, etc. No debe haber status 200 en errores ni 400 en respuestas correctas.
6. Publicaciones:
    a. Deben contener título, mensaje e imagen opcional.
    b. Deben permitir que los usuarios den “Me gusta”.
    c. Debe permitir que los usuarios dejen comentarios.
7. Experiencia de usuario:
    a. Las pantallas deben contar con diseño trabajado y uniforme a lo largo de la aplicación.
    b. Navegación entre pantallas.
    c. Información clara y completa al mostrar mensajes o realizar acciones.

Entregas por sprint:

Sprint #1

Front:

- Creación del proyecto front - Angular.
- Pantallas:
    - Registro
    - Login
    - Publicaciones
    - MiPerfil
- Deploy en hosting
- Navegación entre componentes. Sin límites de accesibilidad.
- Implementar un favicon propio.
- Componente login:
    - Debe poseer un formulario con validaciones y mensajes acordes.
    - Puede ser por correo o por nombre de usuario, pero cualquiera que sea elegido debe ser único en la base de datos.
    - La contraseña debe poseer al menos 8 caracteres, una mayúscula y un número.
- Componente Registro:
    - Debe poseer un formulario con validaciones y mensajes acordes.
    - Debe poseer los campos: nombre, apellido, correo, nombre de usuario, contraseña, repetir contraseña, fecha de nacimiento, descripción breve.
    - Debe poseer un campo de tipo file para la imagen de perfil.
    - Los usuarios deben poseer un atributo perfil. Por defecto poseen el perfil “usuario” pero se puede modificar para que su perfil sea “administrador”.

Back:

- Creación del proyecto back - NestJS.
- Creación de módulos:
    - Publicaciones
    - Autenticación
    - Usuarios
- Módulo Autenticación:
    - Ruta registro:
        - Por **POST**: debe recibir todos los datos de un usuario, validarlos y guardarlo en la base de datos.

        - La contraseña debe quedar encriptada.
        - Debe recibir la imágen de perfil, guardarla apropiadamente y guardar la URL en la base de datos.
    - Ruta login:
        - Por **POST**: debe recibir el usuario / correo y contraseña sin encriptar.
        - Debe encriptar la contraseña recibida para confirmar el login.
        - Debe devolver todos los datos del usuario.

Sprint #2

Front:

- Página publicaciones.
    - Debe mostrar el listado de publicaciones, ordenado por fecha por defecto.
    - Debe permitir cambiar el ordenamiento a por cantidad de me gusta
    - Debe traer una cantidad limitada de publicaciones, permitiendo paginarlas.
    - Cada publicación debe ser un componente.
    - Se debe poder dar y quitar me gusta de una publicación siempre que sea el caso.
    - Debo ser capaz de eliminar mis propias publicaciones.
- Componente Mi Perfil:
    - Debe mostrar todos los datos del usuario, así como su foto de perfil.
    - Debe mostrar mis últimas 3 publicaciones y sus comentarios.

Back:

- Módulo publicaciones - publicaciones controller:
    - Debe permitir dar de alta, listar y dar de baja publicaciones (baja lógica).
    - Por **POST**: debe guardar una publicación relacionada a un usuario. Título, descripción, url de la imagen si es que tiene. La imagen debe ser guardada.
    - Por **DELETE**: baja lógica, solo realizada por el usuario que la creó o un administrador.
    - Por **GET**: debe permitir listar las últimas publicaciones. Debe poder recibir un parámetro para cambiar el ordenamiento: por fecha/ por cantidad de me gusta. Debe poder filtrar los posteos de un usuario particular. Debe poder recibir un parámetro offset y limit para paginar los resultados.
    - Por **POST**: debe permitir que un usuario le dé me gusta a la publicación que elija. Un usuario puede darle un solo me gusta a cada publicación.
    - Por **DELETE**: debe permitir eliminar un me gusta de una publicación, solo si el usuario previamente lo había realizado.

Sprint #3

Front:

- Página publicación.
    - Debe permitir ver grande en la pantalla la publicación realizada, junto con sus respectivos comentarios.
    - Una publicación debe permitir escribir comentarios.
    - Los comentarios deben mostrarse ordenados, uno debajo del otro. En la primera carga debe llegar una cantidad de comentarios limitados, dónde sólo si el usuario presiona un botón “cargar más” se seguirán trayendo, sin dejar de mostrar los anteriores.
    - El usuario que escribió un comentario debe poder editarlo. Un comentario editado debe anunciar que fue editado.
- Pantalla login y registro:
    - Debe tomar el token que devuelve la petición indicada y guardarlo localmente en el navegador. (O utilizar cookies que lo hacen por defecto).
- Página cargando:
    - Al iniciar la aplicación, debe mostrarse una pantalla de cargando y un spinner.
    - En este tiempo, debe validarse frente a la ruta autorizar si el token es válido. En caso de que lo sea, se redirige a la pantalla de publicaciones. En caso de que no lo sea, redirige al login.
- A nivel aplicación:
    - Al iniciar sesión, iniciar un contador de 10 minutos. Al finalizar, deberá aparecer un modal avisando que quedan 5 minutos de sesión y preguntando al usuario si desea extender su sesión (ya que el token vence a los 15 minutos). En caso de que la respuesta sea afirmativa, refrescar el token.
    - Si una petición devuelve un error 401, redirigir al login para rehacer el token.

Back:

- Módulo publicaciones - comentarios controller:
    - Debe permitir traer los comentarios de una publicación, agregar nuevos y modificarlos.
    - Por **POST**: agrega un comentario a una publicación junto con el usuario que lo realizó y la fecha.
    - Por **PUT**: modifica el mensaje del comentario. Agrega el atributo modificado: true para marcar que el comentario se modificó.
    - Por **GET**: trae los comentarios de una publicación específica. Debe permitir paginar los resultados. Debe ordenar los resultados, los más recientes primero.
- Módulo autenticación:

- Rutas Login y registro: debe generar un token JWT que valide quien es el usuario (uuid/correo/nombre de usuario) y su rol (usuario o administrador).
- El token debe ser devuelto en la respuesta o por cookies.
- El token debe vencer a los 15 minutos.
- Ruta autorizar por **POST**: debe validar si un token es válido y no está vencido. Devuelve el status 401 si hubo algún error con el token. En caso de que el token sea válido, devolver los datos del usuario en la respuesta.
- Ruta refrescar por **POST**: debe validar si un token es válido y no está vencido. Devuelve un nuevo token con la misma payload y un vencimiento de 15 minutos.

Sprint #4

Front:

- Página publicación y página publicaciones:
  - Si un usuario con perfil de administrador se logueó, se deben habilitar los botones para dar de baja cualquier publicación. Esto hará que dejen de estar disponibles, tanto la publicación como sus comentarios.
- Página dashboard/usuarios:
  - Solo disponible para usuarios con perfil administrador.
  - Debe permitir ver el listado de los usuarios.
  - Debe permitir crear nuevos usuarios (mismos datos que el registro), permitiendo a través de dos radio buttons elegir si el nuevo usuario es “usuario” o “administrador”.
  - Debe permitir realizar las acciones de alta y baja lógica como corresponda. (habilitar y deshabilitar usuarios. Un usuario deshabilitado no podrá ingresar a la aplicación y será notificado de eso al intentar el login).
  - Debe poseer un formulario de registro para nuevos usuarios.

- Página dashboard/estadísticas:
    - Se deben crear gráficos que representen las siguientes estadísticas:
        - Cantidad de publicaciones realizadas por cada usuario en un lapso de tiempo. El lapso de tiempo debe poder ser elegido.
        - Cantidad de comentarios realizados en un lapso de tiempo. El lapso de tiempo debe poder ser elegido.
        - Cantidad de comentarios en cada publicación en un lapso de tiempo. El lapso de tiempo debe poder ser elegido.
    - Los gráficos pueden ser de cualquier tipo, pero deben variar en tipos (torta, barras, lineas, etc).

- A nivel aplicación:
    - Implementar PWA.
    - Agregar 3 pipes PROPIAS.
    - Agregar 3 directivas PROPIAS.

Back:

- Módulo usuarios - usuarios controller:
    - Debe poseer la lógica para que un administrador pueda listar a los usuarios, dar de alta uno nuevo y realizar bajas y altas lógicas.
    - Se debe validar que el token pertenezca a un administrador.
    - Por **GET**: listado de usuarios
    - Por **POST**: alta de un nuevo usuario. Se puede definir si su perfil es administrador o usuario.
    - Por **DELETE**: permite deshabilitar a un usuario. Cuando dicho usuario quiera ingresar, deberá ser notificado que no está autorizado.

    - Por **POST**: alta lógica, rehabilita a un usuario previamente deshabilitado, permitiéndole utilizar la aplicación.
- Módulo publicaciones - Estadísticas controller.
    - Realizar todas las rutas necesarias por GET para que los gráficos funcionen.
    - Se debe validar que el token pertenezca a un administrador.

Sprint #5 - Recuperatorio

Front:

- Pantalla Perfil:
  - Ahora el usuario puede clickear el nombre o la foto de otro usuario para acceder a ver su perfil. Aquí podrá ver sus datos y sus últimas 3 publicaciones.
  - Se debe mantener en esta pantalla las acciones que puede realizar el administrador (baja lógica de publicaciones)
- Pantalla publicaciones:
  - Se debe reemplazar la paginación tradicional por *scroll infinito*. Las nuevas publicaciones deben aparecer automáticamente sin necesidad de pulsar el botón de página siguiente.
- Página dashboard/estadisticas:
  - Se deben crear gráficos que representen las siguientes estadísticas:
    - Cantidad de ingresos (log in) por usuario
    - Cantidad de visitas a mi perfil (por parte de usuarios que no sean uno mismo).
    - Cantidad de me gusta otorgados por día

Back:

- Realizar la lógica necesaria para cumplir con los requisitos del front.

Sprint #6 - Fecha de final

Front:

- Pantalla publicación: El usuario deberá poder guardar una publicación o quitarla de sus publicaciones guardadas. Se deberá indicar si la publicación está guardada o no.
- Pantalla guardados: El usuario deberá poder ver e interactuar con todas las publicaciones que guardó.
- Pantalla publicación: Botón compartir. Al presionarlo, se debe poder elegir un usuario al que se le compartirá la publicación.
- Pantalla compartidos: En esta pantalla, cada usuario podrá ver las publicaciones que fueron compartidas consigo, además de qué usuario realizó la acción de compartir cada publicación.
- Se deben agregar dos nuevos ordenamientos:
    - Por cantidad de veces que se guardó la publicación.
    - Por cantidad de veces que se compartió la publicación.

Back:

- Módulo publicaciones
    - Por **POST:** Guardar una publicación en los guardados de un usuario (usando el token).
    - Por **POST:** Quitar una publicación de los guardados de un usuario (usando el token).
    - Por **POST**: Se debe guardar que un usuario (usando el token) le comparte a otro usuario (en el body) una publicación.
    - Por **GET**: Debe listar las publicaciones que se compartieron con el usuario que hace la petición (token) junto con los usuarios que compartieron dichas publicaciones.
    - Agregar la lógica necesaria para realizar los nuevos ordenamientos.