Examen Fullstack (Angular + NestJS) — Login simple + CRUD Usuarios

Objetivo

Construir una mini-app fullstack con:

1. **Login sencillo** (usuario/contraseña hardcodeados)
2. **Módulo de Usuarios** con **CRUD completo** (listado + crear + editar + eliminar)
3. Validación de **teléfono obligatorio y único**

> **Importante:** No usar JWT ni autenticación real. No base de datos obligatoria (puede ser in-memory).

Reglas

- Tiempo estimado recomendado: **3–5 horas**.
- Puedes usar Angular Material / Bootstrap / CSS propio (libre).
- Bd mongo
- No se permite copiar un proyecto completo ya hecho; sí se permite usar docs y buscar soluciones puntuales.

Stack obligatorio

- **Frontend:** Angular (ideal 15+)
- **Backend:** NestJS (ideal 9+)
- Comunicación por HTTP REST.

Parte A — Backend (NestJS)

1) Auth “fake”

Endpoint

- POST /auth/login
  - Body: { "username": string, "password": string }

- Credenciales válidas hardcode:
  - username: admin
  - password: admin123
- Respuesta si ok:

{ "ok": true, "token": "fake-token" }

- Respuesta si falla:
  - Status 401

{ "ok": false, "message": "Invalid credentials" }

> El “token” es un string fijo (ej: "fake-token"). No JWT.

2) Guard simple (fake)

Proteger el módulo de usuarios con un guard que valide:

- Header: Authorization: Bearer fake-token

Si no existe o es inválido → 401 Unauthorized.

3) CRUD de usuarios

Entidad User:

- id (string o number autogenerado)
- name (string, obligatorio)
- email (string, obligatorio, formato email deseable)
- phone (string, obligatorio, **no se puede repetir**)

Endpoints

- GET /users → lista
- GET /users/:id → detalle
- POST /users → crear
- PUT /users/:id → actualizar (puede ser full update)
- DELETE /users/:id → eliminar

Reglas de negocio

- phone es obligatorio.
- phone debe ser **único**:
  - Al crear: si existe, devolver 409 Conflict

- Al editar: si el phone cambia y ya existe en otro usuario, 409 Conflict

Respuestas de error esperadas

- 404 si id no existe
- 400 si faltan campos obligatorios
- 409 si phone duplicado

Parte B — Frontend (Angular)

1) Pantalla Login

- Form con username y password
- Si credenciales correctas:
  - Guardar el token "fake-token" (LocalStorage o SessionStorage)
  - Redirigir a /users
- Si falla:
  - Mostrar mensaje “Credenciales inválidas”

2) Guard de rutas

- Rutas protegidas: todo lo de /users
- Si no hay token en storage → redirigir a /login

3) Módulo Usuarios

Pantallas mínimas

1. **Listado** (tabla)
   - columnas: name, email, phone, acciones (editar/eliminar)
   - botón “Crear usuario”
2. **Crear usuario**
   - formulario con name/email/phone
3. **Editar usuario**
   - formulario con los datos del usuario
4. **Eliminar**
   - confirmación antes de borrar

Validaciones en Front

- name obligatorio

- email obligatorio + formato
- phone obligatorio
- si API responde 409 → mostrar error claro: “Teléfono ya existe”

4) Interceptor HTTP (recomendado)

- Inyectar automáticamente el header Authorization si hay token:
  - Authorization: Bearer fake-token

Criterios de evaluación (100 puntos)

Backend (45)

- (10) Endpoints CRUD completos y correctos
- (10) Validación phone único (create + update) con 409
- (10) Guard de token simple por header
- (10) Estructura Nest limpia (modules/controllers/services/dtos)
- (5) Manejo correcto de errores (status codes)

Frontend (45)

- (10) Login funcional + storage token + redirección
- (10) Guard de rutas funcionando
- (10) Listado + Create/Edit/Delete funcionando
- (10) Validaciones y mensajes de error claros (incluye 409)
- (5) Interceptor para Authorization

Calidad (10)

- (5) Código ordenado, nombres claros, commits decentes
- (5) README con pasos para correr y decisiones técnicas

Se valorara

- +5: persistencia real con DB Mongo (Obligatorio)
- +5: paginación / filtro por nombre/teléfono (opcional)
- +5: tests (unit o e2e) básicos (Obligatorio)
- +5: docker-compose para levantar todo fácil (opcional)

- +5: UI decente (Angular Material, etc.) (opcional)

Entregables

1. Repositorio Git con 2 carpetas:
   - /backend (Nest)
   - /frontend (Angular)
2. README que incluya:
   - requisitos (node version)
   - instalación
   - cómo correr backend y frontend
   - endpoints disponibles
   - credenciales hardcode