El examen tiene una duración de 1:30 horas.

## Ejercicio de Programación (6p)

Se desea implementar una aplicación web para gestionar una agenda telefónica. Esta aplicación web deberá tener las siguientes características:

- Funcionalidades:
  - Al entrar en la aplicación se mostrará el título "Contactos", un formulario para dar de alta un nuevo contacto y la lista de contactos existentes.
  - Los datos de un contacto son: nombre, número de teléfono y descripción (que es un campo en el que se podrá introducir un texto). La lista de contactos debe mostrar el nombre, número de teléfono y la descripción en una única linea.
  - Cuando el usuario complete los datos de un nuevo contacto y pulse el botón de "Crear", se enviarán los datos del nuevo contacto al servidor, que los guardará en la base de datos y este nuevo contacto aparecerá en la lista de contactos. Los campos de nombre y número de teléfono son obligatorios pudiendo dejar el campo descripción vacío.
  - Cuando un usuario quiere añadir un nuevo contacto, el servidor verificará que no hay ningún otro contacto con el mismo nombre. En caso de que exista, se le mostrará un error al usuario.
  - En la lista de contactos al lado de cada contacto aparecerá el botón "Borrar". Al pulsar dicho botón, el contacto será borrado de la base de datos y dejará de mostrarse en la lista de contactos.
- Cuestiones de implementación:
  - Se deberá implementar una única aplicación con interfaz web con arquitectura MVC tradicional (generando los HTML en el servidor) y con una interfaz SPA. Es decir, como hemos hecho en la práctica de la fase 4.
  - La aplicación web tradicional estará en la raíz de la URL (http://localhost/) y la aplicación SPA estará en (http://localhost/next/).
  - No es necesario usar ningún tipo de estilo CSS ni librería de componentes.
  - La forma de mostrar el error al usuario cuando se intenta crear un contacto con un nombre existente será diferente en la arquitectura web tradicional y en SPA.
    - En arquitectura web tradicional será una nueva página de error que muestre el mensaje "Contacto ya existente" y un botón o link para volver a la lista de contactos.
    - En SPA será una alerta de JavaScript que se mostrará con el código: `alert("Contacto ya existente");`.
  - El código deberá estar en inglés. Lo único que puede estar en castellano son los textos que aparecen en el interfaz de usuario.

**Se pide:**

- **A) Implementar la aplicación web con Spring MVC, SpringData y JPA. (2p)**
  - No es necesario escribir el fichero pom.xml, se puede asumir que tiene todas las dependencias correspondientes.
  - No es necesario escribir el fichero application.properties, se puede asumir que está conectada a una base de datos externa correctamente configurada.
  - No es necesario escribir la clase Application.
  - Es necesario escribir TODOS los demás ficheros de la aplicación.
  - No es necesario incluir los imports en los ficheros Java.
  - No es necesario implementar los getter y setter de las clases Java. Se pueden dejar indicados con un comentario.

- **B) API REST (0.5p)**
  - Se debe crear una API REST respetando las principios de diseño: formato de las URLs, uso de los códigos de estado, cabeceras, métodos, etc.
  - No debe haber lógica duplicada con la web MVC.
- **C) Implementar el frontend usando Angular (2p)**
  - Se puede asumir que se dispone de un proyecto Angular creado con "ng new" con todos los ficheros necesarios (index.html, angular.cli, package.json, tsconfig.json, app.module.ts, etc.).
  - Hay que escribir el resto de ficheros necesarios para implementar los componentes, servicios y configuración de rutas (en caso de que sean necesarias).
  - No es necesario incluir los imports en los ficheros TypeScript.
  - Se usará HTML plano en el template de los componentes (sin ninguna librería de componentes como ng-bootstrap o angular-material).
  - Se asumirá que el proxy está correctamente configurado y se pueden usar URLs relativas para acceder a la API REST del backend.
- **D) Empaquetar la aplicación en un contenedor Docker (1.5p)**
  - Escribe un fichero Dockerfile multistate que construya una imagen con la aplicación (tanto backend como frontend) empaquetada.
  - Crea un script (build.sh o build.ps1 o build.bat) de construcción y publicación de la imagen docker usando ese Dockerfile. El script deberá contener el comando necesario para construir la imagen Docker con el nombre daw/agenda:1.0.0 y el comando de publicación en DockerHub (se asume que el sistema está logueado en DockerHub con los permisos necesarios para publicar la imagen).
  - Se asumirá que el código Spring está alojado en una carpeta "backend" y el código Angular está alojado en una carpeta "frontend" y el Dockerfile y el script están en la carpeta raíz.
  - Se asumirá que el sistema donde se ejecute el script sólo tiene Docker instalado. No dispone de Node.js ni Maven.
  - Se asumirá que en DockerHub existen las imágenes maven:3.11.0, node:16.0.0 y openjdk-jre:17.0.0