Practica 3 – Gestor de Contactos

Una empresa del sector servicios nos ha encargado desarrollar una **aplicación interna de gestión de contactos** utilizando **Vue 3**, **PrimeVue**, **Pinia** y **vue-router**. El objetivo es que el equipo comercial pueda consultar, crear y actualizar rápidamente la información de sus clientes y contactos clave.

Objetivo del ejercicio

Desarrollar un módulo CRUD completo de contactos que incluya las funcionalidades de listado, creación, visualización de detalle, edición y eliminación de contactos, incorporando además una confirmación previa antes de realizar la eliminación.

Usando:

- **Vue 3** (Composition API o Options API, como prefieras)
- **PrimeVue** para los componentes de interfaz (tabla, formularios, botones, mensajes, etc.)
- **Pinia** para la gestión de estado global
- **vue-router** para la navegación entre vistas

Requisitos funcionales

1. **Rutas de la aplicación**
   Configura el router con las siguientes rutas:
   - `/contactos` → **Listado de contactos**
   - `/contactos/nuevo` → **Creación de un nuevo contacto**
   - `/contactos/:id` → **Detalle de un contacto**
   - `/contactos/:id/editar` → **Edición de un contacto existente**

   Tras cada operación (crear, actualizar, eliminar), la aplicación debe navegar programáticamente a la vista adecuada (por ejemplo, al listado o al detalle).

2. **Store de contactos (Pinia)**
   Crea un **store** `useContactosStore` que:
   - Mantenga en memoria un **array de contactos**.
   - Implemente acciones para:
     - `crearContacto(contacto)`
     - `actualizarContacto(id, datosActualizados)`
     - `eliminarContacto(id)`
     - `toggleFavorito(id)` para marcar/desmarcar como favorito.
   - Exponga **getters**, por ejemplo:
     - `totalContactos`

- `totalFavoritos`
- `contactosActivos`

Cada **contacto** tendrá, al menos, los siguientes campos:

- id
- nombre
- email
- telefono
- empresa
- favorito (booleano: true/false)
- estado ("Activo" / "Inactivo")

3. **Vistas (componentes) usando PrimeVue**

   Se puede partir del boilerplate que tiene configurado PrimeVue: https://github.com/salesmendesandre/daa_vue_examples/tree/main/99_utils/01_primevue_starter

   a) Listado de contactos (`/contactos`)

   - Muestra la lista de contactos en una tabla de **PrimeVue**.
   - Columnas mínimas: nombre, email, teléfono, estado, favorito.
   - Añade acciones por fila:
     - Botón "Ver" → navega a `/contactos/:id`
     - Botón "Editar" → navega a `/contactos/:id/editar`
     - Botón "Eliminar" → elimina el contacto (con confirmación previa).
   - Añade un botón "**Nuevo contacto**" que navegue a `/contactos/nuevo`.
   - Opcional:
     - Campo de búsqueda para filtrar por nombre o email.
     - Icono o etiqueta visual para indicar si un contacto es favorito.

   b) Creación de contacto (`/contactos/nuevo`)

   - Formulario construido con componentes de PrimeVue (InputText, InputTextarea, Dropdown, Checkbox, etc.).
   - Validación básica (por ejemplo, nombre y email obligatorios).
   - Al enviar el formulario:
     - Llama a la acción del store para **crear** el contacto (generando un id si hace falta).
     - Muestra un mensaje de éxito.
     - Navega programáticamente al listado (`/contactos`) o al detalle del nuevo contacto.

   c) Detalle de contacto (`/contactos/:id`)

- Muestra los datos del contacto en un diseño claro (Card de PrimeVue).
- Incluye:
  - Botón "Editar" → `/contactos/:id/editar`
  - Botón "Volver al listado" → `/contactos`
  - Botón para marcar/desmarcar como favorito usando el store.
  - Opcional: Mensaje de error si el id no existe.

d) Edición de contacto (/contactos/:id/editar)

- Formulario similar al de creación, pero precargado con los datos del contacto (obtenidos del store según el `id` de la ruta).
- Al enviar:
  - Llama a la acción del store para **actualizar** el contacto.
  - Muestra un mensaje de éxito.
  - Navega al detalle del contacto o al listado.

4. Confirmaciones y mensajes de estado

- Antes de eliminar un contacto:
  - Muestra una **confirmación**.
- Opcional:: Muestra **mensajes de estado**:
  - "Contacto creado correctamente"
  - "Contacto actualizado correctamente"
  - "Contacto eliminado correctamente"
  - "Contacto no encontrado"

Estos mensajes pueden mostrarse con componentes como Toast o Message de PrimeVue.

Aspectos relevantes

Dado que la aplicación se desarrollará utilizando **PrimeVue** como librería de componentes, es importante **no abusar de estilos CSS personalizados**. Siempre que sea posible, se deben aprovechar las clases, temas y props de estilo que ya ofrece la librería, limitando el uso de CSS propio a pequeños ajustes puntuales (márgenes, alineaciones específicas, etc.). El objetivo es mantener una interfaz homogénea, coherente con el diseño de PrimeVue, y evitar sobrecargar el código con hojas de estilo extensas y difíciles de mantener.

Entrega

Para la entrega se valorarán especialmente la claridad del código y la correcta estructuración de las vistas de la aplicación. Además, se deberá realizar un informe en PDF que explique, con capturas de pantalla, cada una de las funcionalidades implementadas y los aspectos más relevantes de la solución. Sin este informe, la práctica no será evaluable.

Cuando termines, comprime el proyecto completo, elimina la carpeta `node_modules` y súbelo con el nombre: **GestorContactos_NombreApellido1Apellido2**.zip