Práctico Unificado 2025 — Vue 3 (Directivas, Router y Login estilo Vuetify)

Objetivo general: consolidar los prácticos en una única guía coherente para los alumnos, articulando Directivas de Vue 3, Comunicación entre Componentes, Vue Router e **implementación de una pantalla de Login** con una librería de componentes estilo **Vuetify**.

---

1) Introducción y setup del proyecto

Crear un nuevo proyecto con **Vue 3 + Vite**.
Instalar una librería de componentes **estilo Vuetify** (por ejemplo, Vuetify 3) y configurarla: tema claro, tipografías legibles y espaciados consistentes.

---

2) Parte A — Directivas y estados (v-for, v-if, :class, :key)

Objetivo: practicar renderizado de listas, claves estables, estados vacíos, filtros simples y estilos condicionales.

Ejercicio 1 — Listado de productos con búsqueda

- Lista local `[{ id, nombre, precio, stock }]`.
- Input de búsqueda **case-insensitive** por `nombre`.
- Estado vacío: mostrar mensaje si no hay resultados.
- Estilo/flag visual para `stock === 0`.
- Botón “Agregar al carrito” que **emite un evento** con el `id`.

Criterios de aceptación:

- Uso correcto de `v-for` con `:key` estable.
- Manejo de estado vacío y estilos condicionales con `:class`.
- Búsqueda funciona y no rompe al limpiar el input.

---

3) Parte B — Comunicación y estado (carrito mínimo)

Objetivo: eventos entre componentes y `computed` para totales.

Requerimientos:

- Escuchar evento `add-to-cart(id)` y agregar ítems.
- Listar ítems del carrito con **cantidad, precio y subtotal**.
- Cálculo de **total general** con propiedad computada.
- Controles **+ / −** para cantidad; eliminar cuando llegue a 0.
- Mensaje si el carrito está vacío.

Criterios de aceptación:

- El total se actualiza reactivamente.
- Controles nunca llevan la cantidad a valores negativos.

---

4) Parte C — Formulario y validación (registro rápido)

Objetivo: trabajar con `v-model`, validaciones mínimas y feedback visual.

Requerimientos:

- Campo de texto con `v-model` para el **nombre del cliente**.
- Botón de registro: muestra **mensaje personalizado de bienvenida**.
- Simular guardado (console.log) y mostrar confirmación en pantalla solo si hay nombre.
- Estilo simple y usable para un mostrador real.

Criterios de aceptación:

- Validación básica: nombre no vacío (y ≥ 3 caracteres como mejora).
- Feedback claro de éxito/error.

---

5) Parte D — Vue Router (rutas, navegación y detalle)

Objetivo: dividir la app en vistas y practicar ruta dinámica.

Rutas mínimas:

- `/` (Inicio)
- `/productos` (lista)
- `/productos/:id` (detalle dinámico)
- `/clientes`

Comportamiento:

- Enlaces de menú con **enlace activo destacado**.
- Navegación programática desde la lista al detalle.
- “Volver” desde detalle a lista y **restaurar scroll**.
- Manejo de estado de error: “Producto no encontrado”.

Criterios de aceptación:

- Todas las rutas renderizan sus vistas.
- `:id` se recibe y muestra datos correctos del producto.
- No hay warnings de claves duplicadas.

---

6) Parte E — Autenticación (Login + guardas + persistencia)

Objetivo: implementar una pantalla de **Login** con librería tipo **Vuetify**, validaciones, llamadas a servicio (mock), **persistencia de sesión** y **guardas de navegación** para proteger rutas.

6.1 UI del Login (estilo Vuetify)

- **Campos**: email/usuario y contraseña (ambos requeridos).
- **Validación**: formato de email, contraseña de ≥ 6 caracteres.
- **Accesibilidad**: labels, `aria-*`, feedback de error legible.
- **UX**: botón “Ingresar” deshabilitado si el formulario es inválido; indicador de carga en envío; mensajes de error claros.
- **Estética**: card centrada, espaciados (paddings), sombras suaves, radius 2xl.

6.2 Guardia global de rutas (Router)

- Redirigir a `/login` si la ruta requiere auth y el usuario no está autenticado.
- Evitar acceder al login si ya hay sesión (redirigir a `/`).

6.4 Cierre de sesión y layout

- Agregar un botón “Salir” en el layout principal para llamar a `logout()`.
- Mostrar el email del usuario activo en la barra superior.

6.5 Criterios de aceptación (Login)

- Validaciones activas y mensajes claros.
- Redirección correcta post-login y al intentar ingresar a rutas protegidas.
- Persistencia en `localStorage` y restauración de estado al recargar.

- Manejo de errores de credenciales.

---

7) Criterios de evaluación (globales)

- Correcto uso de **directivas** y claves `:key`.
- **Comunicación** entre componentes y `computed` para totales.
- **Router** con rutas, detalle dinámico, restauración de scroll y estados de error controlados.
- **Login** funcional con validación, guardas y persistencia.
- Calidad de **UI/UX** (accesibilidad, consistencia visual, estados de carga/empty/error).
- **Código limpio**: composición, módulos `services/` y `composables/`, nombres claros.