Propuesta 2 — "ReservaLab: Plataforma de Reservación de Laboratorios y Préstamo de Equipo"

1. Descripción General

ReservaLab es una aplicación web para la **gestión integral de laboratorios y préstamo de equipo** en una institución académica.

El sistema permite consultar disponibilidad de laboratorios, realizar reservas por día y franja horaria, solicitar préstamo de equipo de laboratorio o aula, y controlar el estado de devolución. Incluye **autenticación y autorización por roles**, CRUDs completos, validaciones exhaustivas y uso de **ventanas emergentes** para confirmaciones de acciones críticas (reservas, préstamos, devoluciones, cancelaciones).

La aplicación inicia con una **Landing Page** informativa que explica los beneficios del sistema, muestra un resumen de laboratorios y equipos disponibles y botones para **Login** y **Registro**.

Objetivos del Cliente

- Digitalizar el proceso de reserva de laboratorios y préstamo de equipo.
- Evitar choques de horarios y sobreuso de recursos.
- Facilitar la trazabilidad de quién reservó o utilizó cada recurso.
- Generar reportes de ocupación y uso de equipo para toma de decisiones.

---

2. Módulos y Entidades Principales

2.1 Usuarios y Autenticación

- **Registro**: nombre, apellidos, matrícula/ID, correo, contraseña, confirmación, carrera/departamento.
- **Login**: correo + contraseña.
- **Perfil**: edición de datos personales y (opcional) fotografía.
- **Roles**: asignación de rol por Administrador (Administrador, Técnico/Laboratorista, Estudiante).
- **Sesiones**: uso de **JWT** con access y refresh tokens.
- **Autorización**: guards en frontend y permisos por vista en backend.

2.2 Catálogos

- **Laboratorios**: nombre, edificio, piso, capacidad, tipo (cómputo, electrónica, biología), estado (activo/inactivo).
- **Equipos**: nombre, descripción, número de inventario, cantidad total y disponible, estado (disponible/mantenimiento), laboratorio asignado (opcional).
- **Horarios de operación**: franja horaria, días de semana, estado.

2.3 Reservas

- **Campos obligatorios**: usuario solicitante, laboratorio, fecha, hora inicio, hora fin, motivo de uso.
- **Estado**: pendiente, aprobada, rechazada, cancelada.
- **Validaciones**:
    - No permitir choques de horario para el mismo laboratorio.
    - Validar que la reserva esté dentro del horario de operación.
    - Restringir reservas en laboratorios inactivos o en mantenimiento.

2.4 Préstamos de Equipo

- **Campos obligatorios**: usuario solicitante, equipo, cantidad solicitada, fecha de préstamo, fecha de devolución prevista.
- **Estado**: pendiente, aprobado, rechazado, devuelto, dañado.
- **Reglas**:
    - Validar disponibilidad de equipo antes de confirmar.
    - Registrar devoluciones y marcar daños desde el rol de Técnico/Laboratorista.
    - Alertar si hay retrasos en devolución.

2.5 Reportes

- **Administrador**:
    - Tasa de ocupación de laboratorios por periodo.
    - Equipos más solicitados.
    - Incidencias reportadas (daños o pérdidas).
- **Técnico/Laboratorista**:
    - Listado de préstamos activos, próximos a vencer y devueltos.
- **Estudiante**:
    - Historial de reservas y préstamos.

---

3. Flujo de Navegación (Frontend)

3.1 Estructura del Proyecto (Angular 19)

Mantener la organización estándar del curso:

- **/screens**
    - landing

- auth/login
- auth/register
- dashboard (con versión diferenciada por rol)
- labs/list, labs/form
- equipment/list, equipment/form
- reservations/list, reservations/form
- loans/list, loans/form
- admin/users
- reports
- profile

- **/partials**
  - navbar, sidebar, footer, breadcrumbs, toast/alerts
- **/modals**
  - confirm-reservation-modal
  - cancel-reservation-modal
  - confirm-loan-modal
  - return-equipment-modal
  - incident-report-modal
- **/services**
  - auth.service, users.service, labs.service, equipment.service, reservations.service, loans.service, reports.service
- **/shared**
  - Modelos/Interfaces TS (Usuario, Laboratorio, Equipo, Reserva, Préstamo, Estado).
  - Validadores (fechas, choques de horario, cantidad disponible).
  - Pipes (formato de fecha, estado).
  - Guards (AuthGuard, RoleGuard).
  - Interceptors (JWT, manejo de errores).

3.2 Rutas y Acceso

- **Públicas**: /, /auth/login, /auth/register
- **Autenticadas**: /dashboard, /profile, /reservations/list, /loans/list
- **Sólo Admin/Técnico**: /labs, /equipment, /reports, endpoints de aprobación y devoluciones.

3.3 Comportamientos UX Clave

- **Calendario interactivo** de disponibilidad de laboratorios.
- **Filtros** para laboratorios por tipo, capacidad y estado.
- **Notificaciones en tiempo real** (toast) para confirmaciones y rechazos.
- **Formularios reactivos** con validaciones en vivo.
- **Modales** para aprobar, rechazar, devolver y cancelar reservas/préstamos.
- **Diseño mobile-first**, con colapsado de menús en pantallas pequeñas.

---

4. Reglas de Negocio y Validaciones

4.1 Usuarios

- **Correo único** y matrícula obligatoria.
- **Contraseña**: mínimo 8 caracteres, con al menos una mayúscula, una minúscula y un número.
- Cambio de rol sólo permitido para Administrador.

4.2 Laboratorios

- Capacidad > 0.
- Estado **activo** para poder asignar reservas.
- Evitar solapamiento de horarios en reservas.

4.3 Equipos

- Cantidad total > 0.
- Estado debe ser “disponible” para que pueda prestarse.

4.4 Reservas

- No permitir reservar con fecha pasada.
- Validar choques de horario antes de aprobar.
- Cancelación requiere confirmación y (opcional) motivo.

4.5 Préstamos

- Validar disponibilidad antes de aprobar.
- Registrar fecha de devolución prevista.
- Marcar devolución o daño obligatoriamente al cierre.

---

5. Backend (Django 4.2 LTS + DRF + JWT)

5.1 Estructura de Apps

- accounts (usuarios, roles, autenticación)
- labs (CRUD laboratorios, horarios)
- equipment (CRUD equipo, inventario)
- reservations (reservas, validaciones, cancelaciones)
- loans (préstamos, devoluciones, incidencias)
- reports (métricas y estadísticas)

5.2 Modelos

- **User**: nombre, apellidos, correo, matrícula, rol.
- **Lab**: nombre, tipo, capacidad, estado.
- **Equipment**: nombre, número de inventario, cantidad, estado, laboratorio asignado.
- **Reservation**: usuario, laboratorio, fecha, hora inicio, hora fin, motivo, estado.
- **Loan**: usuario, equipo, cantidad, fecha préstamo, fecha devolución, estado.

5.3 Vistas/Endpoints

- CRUD de laboratorios y equipos (Admin).
- Endpoints de reservas: crear, aprobar/rechazar, cancelar.
- Endpoints de préstamos: crear, aprobar/rechazar, devolver, reportar daño.
- Reportes: reservas por fecha, uso de equipos, incidencias.

5.4 Paginación, Búsqueda y Orden

- Listados paginados de laboratorios, equipos, reservas y préstamos.
- Filtros por estado, fecha y usuario.
- Orden por fecha, nombre y capacidad.

---

6. Requisitos del Frontend

6.1 Pantallas Mínimas

- Landing con resumen de laboratorios y equipos.
- Login / Registro.
- Dashboard diferenciado por rol.
- Lista y formulario de reservas.

- Lista y formulario de préstamos.
- Administración de laboratorios y equipos.
- Reportes con gráficas (ng2-charts).
- Perfil de usuario editable.

6.2 Validaciones de Formularios

- **Reserva**: fechas coherentes, sin choques de horario, motivo obligatorio.
- **Préstamo**: cantidad ≤ disponible, fecha devolución obligatoria.
- **Laboratorios y equipos**: nombres únicos, capacidad/cantidad > 0.

6.3 Modales

- Confirmación para aprobar/rechazar reserva.
- Confirmación para devolver equipo y registrar daño.
- Cancelación de reserva con motivo.

---

7. Requisitos No Funcionales

- **Responsividad** total, diseño mobile-first.
- **Accesibilidad**: etiquetas ARIA, contraste adecuado, soporte teclado.
- **Seguridad**: JWT, permisos por rol en endpoints.
- **Mantenibilidad**: estructura de carpetas clara, separación de servicios.

---

8. Criterios de Aceptación

1. Landing Page informativa con CTA de login/registro.
2. Registro/Login funcionales con validaciones.
3. Roles activos (Admin, Técnico, Estudiante) con navegación diferenciada.
4. CRUD completo de laboratorios y equipos (solo Admin).
5. Reservas funcionales con control de horario y estados (pendiente/aprobada/rechazada).
6. Préstamos con flujo de solicitud → aprobación → devolución.
7. Reportes básicos por rol (ocupación, equipo más usado).
8. Modales para confirmaciones de acciones críticas.
9. App 100% responsiva y desplegada en Vercel + Render.

---

9. Despliegue

- **Frontend en Vercel** con variable API_BASE_URL.
- **Backend en Render** con MySQL configurado.
- Migraciones aplicadas y CORS habilitado.
- URLs de producción entregadas con usuarios de prueba por rol.

---

10. Entregables del Equipo

- **Modelo ER** de BD con descripción de entidades.
- **Manual de Usuario / Reporte formal** con capturas y flujos.
- **Video demostrativo** (5–10 min) evidenciando reservas y préstamos.
- **Repositorios GitHub** (frontend y backend) con README completo.
- **URLs de producción** (Vercel y Render) funcionando end-to-end.
- **Evidencia de responsividad** en móvil, tablet y escritorio.

---

11. Matriz de Permisos (resumen)

Funcionalidad / Recurso	Administrador	Técnico/Laboratorista	Estudiante
Ver landing page	✓	✓	✓
Registro/Login	✓	✓	✓
CRUD de laboratorios	✓	✕	✕
CRUD de equipos	✓	✕	✕
Consultar disponibilidad	✓	✓	✓
Crear reserva	✓	✕	✓
Aprobar/Cancelar reservas	✓	✓	✕
Crear solicitud de préstamo	✓	✕	✓
Aprobar/Cancelar préstamos	✓	✓	✕
Registrar devolución/incidencia	✓	✓	✕
Ver reportes de uso	✓ (globales)	✓ (su laboratorio/equipo)	✓ (historial propio)