Pruebas de Integración – Sistema Tenis Master

Documento Completo y Detallado
Versión: 1.0
Fecha: 2025
Equipo: RedSoft – Proyecto Integrador

1. Introducción a las Pruebas de Integración

Las pruebas de integración verifican el correcto funcionamiento del sistema Tenis Master cuando múltiples componentes interactúan entre sí. Estas pruebas evalúan flujos completos que incluyen:

• Interacción entre vistas (Pug), rutas (Express) y controladores.
• Consulta e inserción en la base de datos MongoDB.
• Manejo de sesiones y roles.
• Validación de flujos reales realizados por un usuario.

Estas pruebas aseguran que el sistema funcione como un conjunto cohesivo y permita acciones consistentes desde la interfaz hasta la persistencia de datos.

2. Prueba de Integración Nº1: Creación de Reserva (Rol Cliente)

2.1 Objetivo

Validar que un usuario con rol Cliente pueda completar el formulario de reserva, y que dicha reserva sea correctamente registrada en la base de datos y mostrada en su panel de reservas.

2.2 Componentes Involucrados

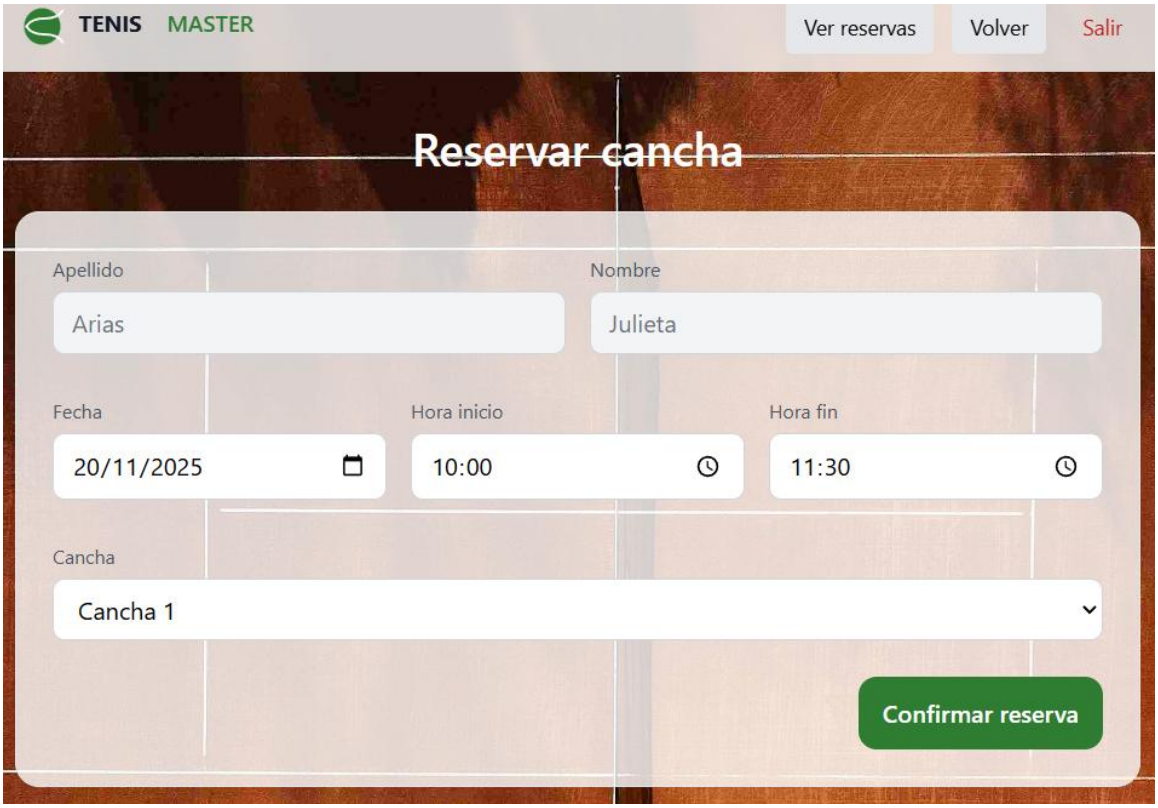
• Vista: reservar.pug
• Ruta: POST /reservar (reservaRoutes.js)
• Controlador: crearReserva() (reservaController.js)
• Modelo: Reserva.js
• Vista destino: panel-reservas-cliente.pug

2.3 Pasos Realizados

Paso 1 — Inicio de Sesión:
El usuario ingresa con rol Cliente (cliente@mail.com). El sistema lo redirige al panel de cliente (
/panel-reservas).

Paso 2 — Acceso a la Vista de Reserva:
El cliente selecciona 'Reservar', lo que dispara GET /reservar.

Paso 3 — Completar Formulario:
Se cargan los siguientes datos:
• Fecha: 2025-11-20
• Hora inicio: 10:00
• Hora fin: 11:30
• Cancha: Cancha 1
Los campos de nombre/apellido aparecen precargados y bloqueados.



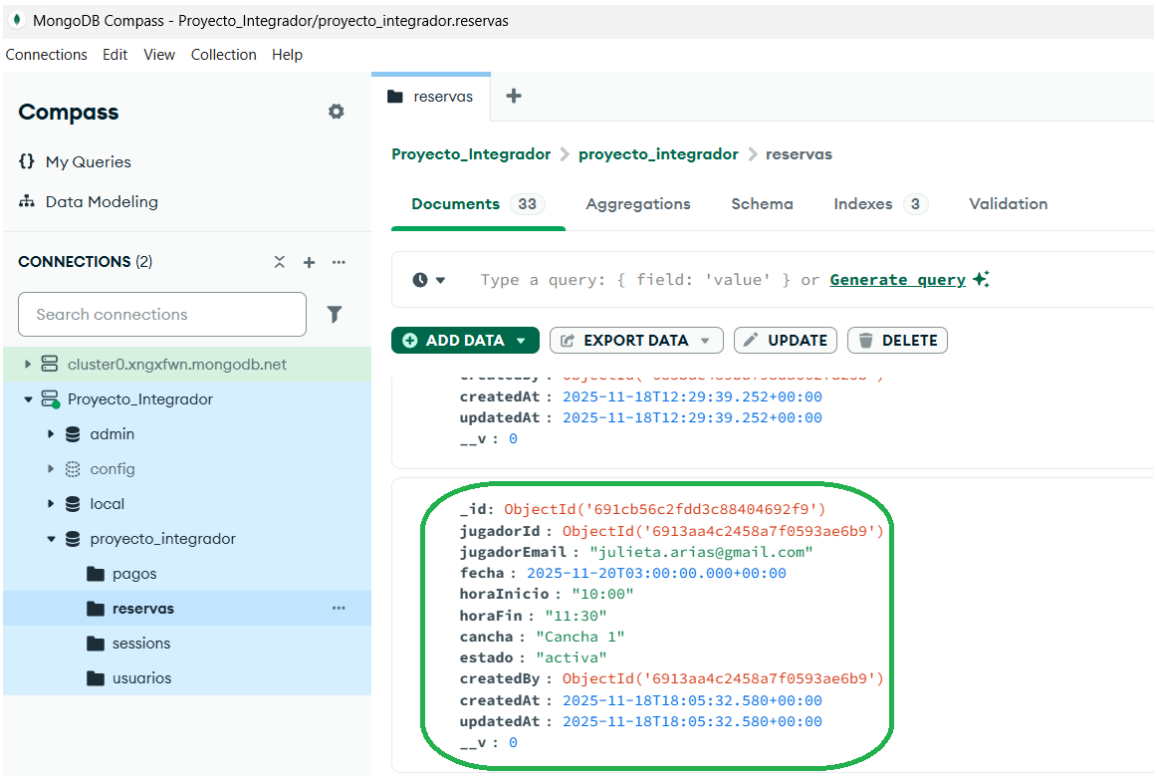
Paso 4 — Envío del Formulario:
El usuario confirma, activando la ruta POST /reservar.

Paso 5 — Lógica del Controlador (crearReserva):
• Valida campos obligatorios.
• Normaliza fecha.
• Verifica solapamientos.
• Inserta la reserva como documento en MongoDB.
• Redirige al panel del cliente.

Paso 6 — Visualización:
Panel-reservas-cliente.pug muestra la nueva reserva en la tabla de próximas reservas.

2.4 Resultado Esperado

• La reserva se inserta en la colección 'reservas'.
• Se visualiza en el panel del cliente como estado 'activa'.
• No ocurren errores en consola.



2.5 Resultado Obtenido

- La reserva fue creada correctamente.
- MongoDB muestra el nuevo documento.
- La interfaz refleja la reserva con los datos correctos.
- No se detectaron errores en el servidor.

3. Prueba de Integración Nº2: Pago de Reserva (Rol Admin/Empleado)

3.1 Objetivo

Confirmar que un usuario con rol Administrador o Empleado pueda ingresar al flujo de pago de una reserva activa, y que el sistema registre el pago, actualice la reserva y muestre el cambio en el panel administrativo.

3.2 Componentes Involucrados

- Vista origen: panel-reservas-admin.pug
- Ruta de acceso al pago: GET /reservas/:id/pago
- Ruta de confirmación: POST /reservas/:id/pagar
- Controladores: mostrarPago() y pagarReserva()
- Modelos: Reserva.js y Pago.js
- Vista final: panel-reservas-admin.pug

3.3 Pasos Realizados

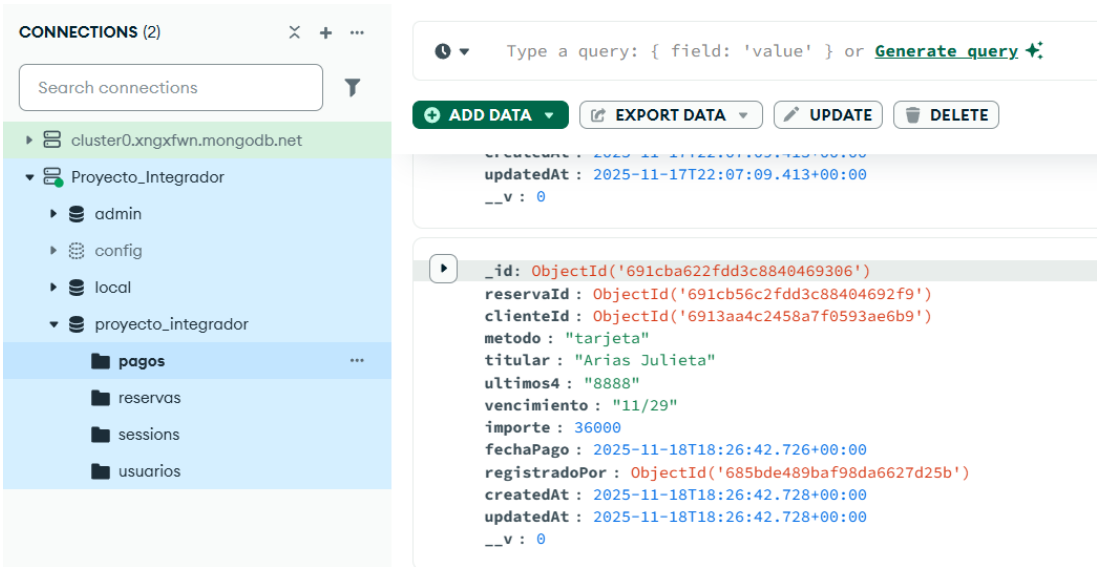
Paso 1 — Inicio de Sesión:
El administrador inicia sesión (admin@mail.com) y accede al panel de reservas.

Paso 2 — Selección de Reserva:
Se identifica una reserva activa con duración 10:00–11:30.

Paso 3 — Acceso a Pago:
Al presionar 'Pagar', se ejecuta GET /reservas/:id/pago.
La vista pago.pug muestra datos del cliente, cancha, fecha y total calculado.

Paso 4 — Confirmación de Pago:
Se elige método 'Tarjeta', se ingresan datos y se envía el formulario.

Paso 5 — Lógica del Controlador (pagarReserva):
• Obtiene la reserva.
• Valida que esté activa.
• Calcula importe:
10:00–11:30 = 90 minutos = 3 bloques de 30 min.
Precio por bloque = 24000 / 2 = 12000.
Total = 36000.
• Inserta documento en colección 'pagos'.



• Actualiza la reserva a estado 'pagado'.
• Redirige al panel administrativo.

3.4 Resultado Esperado

• Se crea un documento en la colección 'pagos' con importe = 36000.
• La reserva cambia su estado a 'pagado'.
• La interfaz no permite volver a pagar la misma reserva.

3.5 Resultado Obtenido

• El pago se registró con éxito.
• La reserva aparece como 'pagado' en panel-reservas-admin.
• No se generaron errores en consola ni en servidor.

4. Conclusiones Generales

Ambas pruebas de integración confirmaron que los módulos del sistema Tenis Master interactúan correctamente entre sí. Los flujos críticos —creación de reservas y pago— funcionan como un proceso sólido conectando vistas, rutas, controladores y modelos. El sistema responde adecuadamente a acciones reales de usuarios con roles distintos.