Documentación de Pruebas de Calidad

Módulo de Eventos - Componentes EventoController y EventosUsuario

Introducción

Este documento presenta el plan de pruebas de calidad para los componentes relacionados con el manejo de eventos en el sistema, incluyendo el controlador EventoController (backend PHP) y el componente EventosUsuario (frontend React). Se detallan los casos de prueba, criterios de aceptación, y escenarios de prueba para asegurar la correcta funcionalidad del módulo.

Componentes Analizados:

- EventoController (Backend)
- EventosUsuario (Frontend)
- Modelo Evento

Tecnologías utilizadas:

- Backend: PHP, Laravel
- Frontend: React, Axios
- Base de datos: Relacional con relación muchos a muchos

Fecha de evaluación: 19 de mayo de 2025

Descripción de los Componentes

Backend - EventoController

El controlador EventoController es responsable de gestionar los eventos en el sistema. Permite realizar las siguientes operaciones:

- Listar todos los eventos disponibles
- Crear nuevos eventos
- Registrar un usuario en un evento específico

El controlador utiliza el modelo Evento para interactuar con la base de datos y maneja la relación muchos a muchos entre eventos y usuarios.

Frontend - EventosUsuario

El componente EventosUsuario es responsable de mostrar la lista de eventos disponibles al usuario y permitirle registrarse en ellos. Sus características incluyen:

- Visualización de eventos en formato de tarjetas
- Ver detalles ampliados de un evento en un modal
- Registrarse en un evento específico
- Manejo de estados de carga y errores

Funcionalidades a Probar

1. Backend - EventoController

- Listar eventos (GET /api/eventos)
- Crear evento (POST /api/eventos)
- Registrar usuario en evento (POST /api/eventos/{eventoId}/registrar-usuario)

2. Frontend - EventosUsuario

- Carga y visualización de eventos
- Registro de usuario en evento
- Visualización de detalles de evento en modal
- Manejo de estados (carga, éxito, error)

Casos de Prueba

Pruebas Funcionales - Backend (EventoController)

ID	Caso de Prueba	Datos de Entrada	Resultado Esperado	Criterio de Aceptación
BP-01	Listar todos los eventos	Petición GET a /api/eventos	- Respuesta con código 200 - Array con todos los eventos	La API devuelve correctamente los eventos almacenados en la base de datos
BP-02	Crear evento con datos válidos	Petición POST a /api/eventos Datos: nombre: "Feria Tecnológica" fecha: "2025-06-15" lugar: "Campus Principal"	- Respuesta con código 201 - Objeto JSON con datos del evento creado	El evento se crea correctamente en la base de datos
BP-03	Crear evento con datos inválidos	Petición POST a /api/eventos Datos incompletos: nombre: "Feria Tecnológica" (sin fecha ni lugar)	- Respuesta con código 422 - Objeto JSON con errores de validación	La API rechaza la creación y devuelve errores de validación
BP-04	Registrar usuario en evento	Petición POST a /api/eventos/1/registrar-usuario Usuario autenticado	- Respuesta con código 200 - Mensaje de registro exitoso	El usuario queda registrado en el evento en la tabla pivote
BP-05	Registrar usuario en evento ya registrado	Petición POST a /api/eventos/1/registrar-usuario Usuario ya registrado en ese evento	- Respuesta con código 409 - Mensaje indicando que ya está registrado	La API impide el registro duplicado
BP-06	Registrar usuario en evento inexistente	Petición POST a /api/eventos/999/registrar-usuario (ID no existente)	- Respuesta con código 404 - Mensaje de evento no encontrado	La API maneja correctamente recursos inexistentes

Pruebas Funcionales - Frontend (EventosUsuario)

ID	Caso de Prueba	Datos de Entrada	Resultado Esperado	Criterio de Aceptación
FP-01	Carga de eventos	Usuario accede a la página	- Indicador de carga mientras se obtienen los datos - Visualización de eventos en tarjetas al completar	Los eventos se muestran correctamente organizados en tarjetas
FP-02	Sin eventos disponibles	API devuelve array vacío	Mensaje indicando que no hay eventos disponibles	El componente maneja correctamente la ausencia de eventos
FP-03	Visualizar detalles de evento	Clic en botón "Ver más detalles"	- Modal se abre - Muestra información detallada del evento seleccionado	El modal muestra correctamente los detalles del evento seleccionado
FP-04	Cerrar modal de detalles	Clic en botón de cerrar o fuera del modal	Modal se cierra	El modal se cierra correctamente
FP-05	Registro exitoso en evento	Clic en botón "Registrarme" Usuario autenticado	- Botón muestra "Registrando..." durante el proceso - Alerta de éxito al completar	El usuario recibe confirmación visual de registro exitoso
FP-06	Registro en evento ya registrado	Clic en botón "Registrarme" Usuario ya registrado	Alerta indicando que ya está registrado en el evento	Se muestra correctamente el mensaje de error específico
FP-07	Registro sin autenticación	Clic en botón "Registrarme" Usuario no autenticado	Alerta indicando que debe iniciar sesión	Se muestra el mensaje de error de autenticación
FP-08	Error genérico al registrarse	Clic en botón "Registrarme" Error del servidor diferente a 401/409	Alerta genérica de error	Se muestra el mensaje de error genérico

Pruebas de Interfaz de Usuario (EventosUsuario)

ID	Caso de Prueba	Pasos	Resultado Esperado	Criterio de Aceptación
UI-01	Visualización responsive	Acceder al componente desde diferentes tamaños de pantalla	El diseño se adapta según el tamaño de pantalla (1, 2 o 3 columnas)	El componente es usable en dispositivos móviles, tablets y escritorio
UI-02	Estado de carga	Acceder al componente	Mensaje "Cargando eventos..." mientras se realiza la petición	Se muestra indicador de carga para proporcionar feedback al usuario
UI-03	Estado sin eventos	Acceder cuando no hay eventos	Mensaje "No hay eventos disponibles"	Se muestra mensaje informativo cuando no hay datos
UI-04	Feedback visual de registro	Clic en botón "Registrarme"	El botón cambia a "Registrando..." durante el proceso	El usuario recibe feedback visual durante la acción
UI-05	Modal de detalles	Clic en "Ver más detalles"	Modal con fondo oscuro y contenido centrado, estilo coherente	Modal visualmente agradable y funcional

Pruebas de Seguridad

ID	Caso de Prueba	Descripción	Resultado Esperado	Criterio de Aceptación
SEC-01	Autenticación en backend	Acceder a registrar usuario sin token	Respuesta con código 401 no autorizado	La API rechaza peticiones no autenticadas
SEC-02	Validación de token	Acceder con token manipulado o expirado	Respuesta con código 401 o 403	El sistema rechaza tokens inválidos
SEC-03	Inyección SQL en parámetros	Enviar datos maliciosos en los campos de creación de evento	Los datos son sanitizados antes de procesarse	La base de datos está protegida contra inyección SQL
SEC-04	Autorización para crear eventos	Verificar que solo usuarios autorizados pueden crear eventos	API rechaza peticiones de usuarios sin permisos	Control de acceso funciona correctamente

Pruebas de Integración

ID	Caso de Prueba	Descripción	Resultado Esperado	Criterio de Aceptación
INT-01	Flujo completo de creación y registro	Crear evento y luego registrar usuario	Evento creado y usuario registrado correctamente	La integración completa funciona sin problemas
INT-02	Integridad referencial	Verificar comportamiento al eliminar evento con usuarios registrados	El sistema maneja correctamente las relaciones en la base de datos	No se producen inconsistencias en la base de datos
INT-03	Integración entre frontend y backend	Realizar flujo completo desde interfaz de usuario	Los datos fluyen correctamente entre los componentes	El sistema funciona como un todo coherente

Pruebas de Rendimiento

ID	Caso de Prueba	Descripción	Resultado Esperado	Criterio de Aceptación
PERF-01	Carga de eventos con gran volumen	Realizar petición con 100+ eventos	Tiempo de respuesta aceptable (<3s)	La aplicación mantiene buen rendimiento con gran volumen de datos
PERF-02	Múltiples registros simultáneos	Simular múltiples usuarios registrándose simultáneamente	El sistema procesa todas las peticiones sin errores	No se producen condiciones de carrera o bloqueos
PERF-03	Tiempo de respuesta del backend	Medir tiempo de respuesta promedio de endpoints	Respuesta en menos de 300ms	Tiempo de respuesta dentro de parámetros aceptables
PERF-04	Carga inicial de frontend	Medir tiempo hasta que el componente es interactivo	Menos de 2 segundos	El componente se carga rápidamente

Pruebas de Accesibilidad

ID	Caso de Prueba	Descripción	Resultado Esperado	Criterio de Aceptación
ACC-01	Navegación con teclado	Navegar por el componente usando solo el teclado	Todos los elementos interactivos son accesibles	El componente es usable sin ratón
ACC-02	Compatibilidad con lectores de pantalla	Probar componente con lector de pantalla	Los elementos tienen etiquetas adecuadas	El componente es accesible para personas con discapacidad visual
ACC-03	Contraste de colores	Evaluar contraste entre texto y fondo	Cumple con estándares WCAG	Contraste suficiente para buena legibilidad
ACC-04	Estados interactivos visibles	Verificar que los elementos interactivos tienen estados visibles	Los botones y elementos interactivos indican su estado	Los estados son perceptibles por distintos medios

Defectos y Mejoras Sugeridas

Defectos Identificados

Backend (EventoController)

1. **Inconsistencia en rutas**: La ruta en el controlador es `/api/eventos/{eventoId}/registrar-usuario` pero en el frontend se usa `/api/eventos/{eventoId}/registrarse`
2. **Falta validación de fecha**: No se valida que la fecha del evento sea futura
3. **Falta paginación**: El endpoint de listar eventos devuelve todos sin paginación
4. **Falta documentación para algunos parámetros**: En la documentación OpenAPI faltan algunos parámetros

Frontend (EventosUsuario)

1. **Gestión de errores limitada**: Se manejan solo algunos códigos de error específicos
2. **No se actualiza la UI tras registro**: No hay actualización de interfaz después de un registro exitoso
3. **Alerta nativa**: Se usa `alert()` nativo en lugar de un componente de alerta más integrado estéticamente
4. **No se valida `null` en datos**: Podría haber errores si algún campo del evento es null

Mejoras Sugeridas

Backend

1. **Estandarizar nomenclatura de rutas**:

php

```php
// Modificar la ruta en la documentación a:
/**
 * @OA\Post(
 *     path="/api/eventos/{eventoId}/registrarse",
 *     tags={"Eventos"},
 *     //...
 * )
 */
public function registrarUsuario(Request $request, $eventoId)
{
    // ...
}
```

2. **Implementar validación de fecha**:

php

```php
$validator = Validator::make($request->all(), [
    'nombre' => 'required|string|max:255',
    'fecha' => 'required|date|after:today',
    'lugar' => 'required|string|max:255',
]);
```

3. **Añadir paginación**:

php

```php
public function index()
{
    return response()->json(Evento::paginate(10), 200);
}
```

Frontend

1. **Mejorar gestión de errores**:

javascript

```
try {
    await axios.post(`/api/eventos/${eventoId}/registrarse`);
    setAlerta({ tipo: 'exito', mensaje: 'Te registraste exitosamente al evento.' });
} catch (err) {
    if (err.response?.status === 409) {
        setAlerta({ tipo: 'advertencia', mensaje: 'Ya estás registrado en este evento.' });
    } else if (err.response?.status === 401) {
        setAlerta({ tipo: 'error', mensaje: 'Debes iniciar sesión para registrarte.' });
    } else {
        setAlerta({
            tipo: 'error',
            mensaje: err.response?.data?.message || 'Error al registrarte al evento.'
        });
    }
} finally {
    setRegistrando(false);
}
```

2. **Implementar actualización de UI post-registro**:

javascript

```
const registrarse = async (eventoId) => {
    setRegistrando(true);
    try {
        await axios.post(`/api/eventos/${eventoId}/registrarse`);
        // Actualizar estado local para reflejar el registro
        setEventos(eventos.map(event =>
            event.id === eventoId
                ? { ...event, registrado: true }
                : event
        ));
        setAlerta({ tipo: 'exito', mensaje: 'Te registraste exitosamente al evento.' });
    } catch (err) {
        // Manejo de errores...
    }
};
```

3. **Reemplazar alerts nativos por componente personalizado**:

jsx

```jsx
const [alerta, setAlerta] = useState(null);

// En el JSX
{alerta && (
    <div className={`p-3 rounded mb-4 ${
        alerta.tipo === 'exito' ? 'bg-green-100 text-green-800' :
        alerta.tipo === 'advertencia' ? 'bg-yellow-100 text-yellow-800' :
        'bg-red-100 text-red-800'
    }`}>
        {alerta.mensaje}
        <button
            className="float-right"
            onClick={() => setAlerta(null)}
        >
            &times;
        </button>
    </div>
)}
```

4. **Validación de datos nulos**:

jsx

```jsx
<h3 className="text-xl font-semibold mb-2">
    {evento.nombre || 'Sin nombre'}
</h3>
<p>🗓️ {evento.fecha || 'Fecha no especificada'}</p>
<p>📍 {evento.lugar || 'Lugar no especificado'}</p>
```

Conclusiones

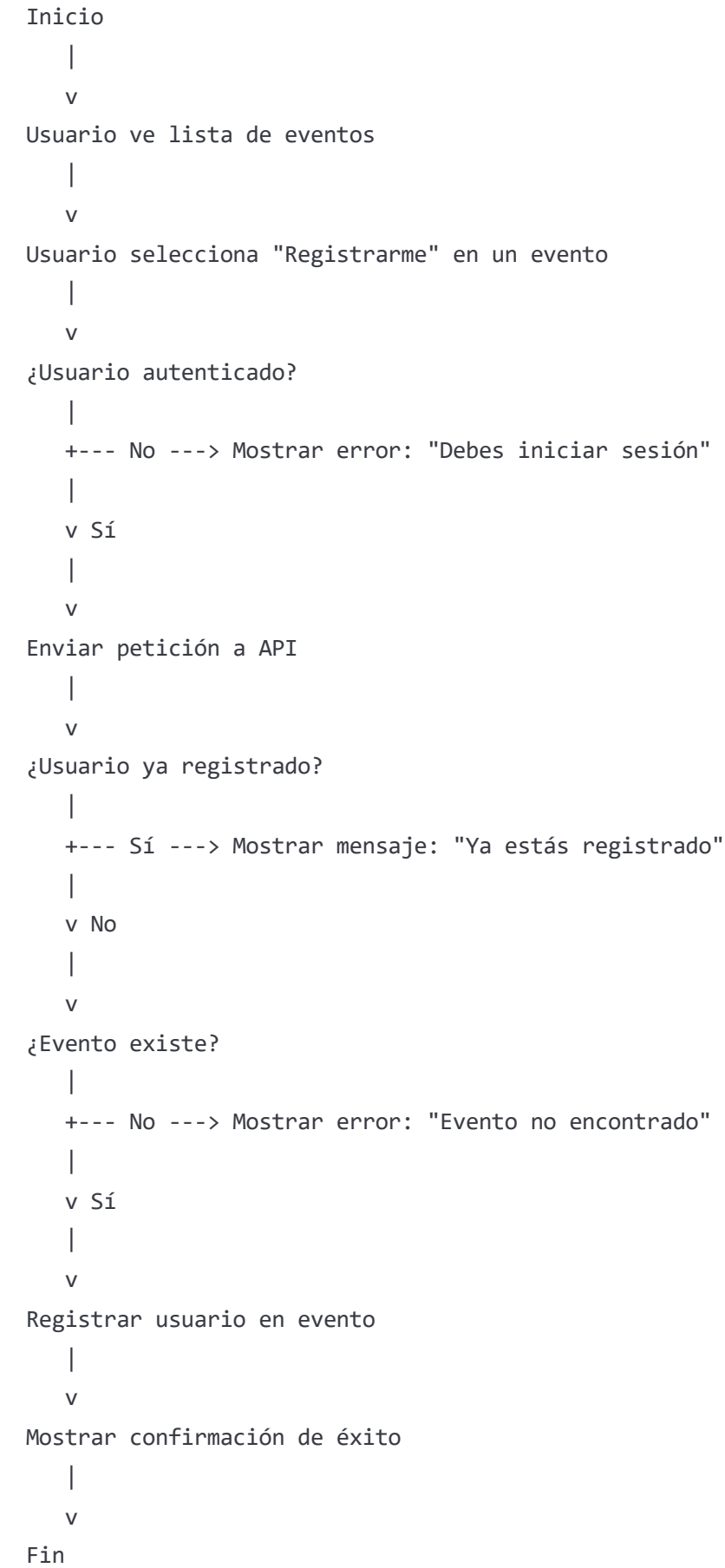
El módulo de Eventos tiene una arquitectura bien estructurada con separación clara entre backend y frontend. El controlador EventoController implementa correctamente las operaciones básicas CRUD y el registro de usuarios en eventos, mientras que el componente EventosUsuario proporciona una interfaz atractiva y funcional para interactuar con estos eventos.

Se han identificado algunos problemas de consistencia entre backend y frontend, principalmente en la nomenclatura de rutas, que deberían ser corregidos para evitar errores. También se han sugerido mejoras tanto en el backend (validación de fechas, paginación) como en el frontend (gestión de errores, componentes de alerta personalizados) para mejorar la experiencia del usuario y la robustez del sistema.

La implementación de las mejoras sugeridas aumentaría significativamente la calidad del módulo, asegurando una experiencia más coherente y sin errores para los usuarios.

Anexos

Diagrama de Flujo - Registro de Usuario en Evento



Entorno de Pruebas Recomendado

Navegadores a probar:

- Google Chrome (última versión)
- Mozilla Firefox (última versión)
- Microsoft Edge (última versión)
- Safari (última versión)

Dispositivos a probar:

- Desktop (resoluciones 1920x1080, 1366x768)
- Tablet (iPad, resolución 768x1024)
- Móvil (iPhone, Galaxy, resoluciones 375x667, 414x896)

Herramientas recomendadas:

- PHPUnit para pruebas unitarias de backend
- React Testing Library para pruebas unitarias de frontend
- Jest para testing de componentes React
- Postman para pruebas de API
- Lighthouse para evaluación de rendimiento
- axe para pruebas de accesibilidad