Documentación de Pruebas de Calidad

**Módulo de creación de Barberos - Componente GestionarBarberos**

1. **Introducción**

Este documento presenta el plan de pruebas de calidad para el componente GestionarBarberos, que permite la creación de cuentas de los roles barberos. Se detallan los casos de prueba, criterios de aceptación, y escenarios de prueba para asegurar la correcta funcionalidad del módulo.

**Componente Analizado:** GestionarBarberos

**Tecnologías utilizadas:** React, Axios, SweetAlert2 **Fecha de evaluación:** 19 de mayo de 2025

1. **Descripción del Componente**

El componente GestionarBarberos es responsable de la creación de nuevas cuentas para el rol barbero. Permite a los (administradores) ingresar la información necesaria para crear una nueva cuenta, incluyendo:

Nombre Barbero



 Email

 Contraseña

Descripcion



* Imagen del barbero

El componente utiliza el token de autenticación almacenado en localStorage para identificar al administrador creador de la cuenta.

1. **Funcionalidades a Probar**

## 1. Crear Cuenta Barbero

 Ingreso de datos en el formulario  Validación de campos obligatorios

 Carga y previsualización de imágenes  Envío de datos al servidor

Manejo de respuestas exitosas y errores

****

* 1. **Casos de Pruebas Funcionales – Crear Cuenta de barberos**

****

| **ID** | **Caso de Prueba** | **Datos de Entrada** | **Resultado Esperado** | **Criterio de**  **Aceptación** |
| --- | --- | --- | --- | --- |
| CP- 01 | Creación exitosa de cuenta con todos los datos incluida imagen | Nombre Barbero: "Cristian Rueda"  Email: [cris@gmail.com](mailto:cris@gmail.com)  Contraseña: cris12345  Descripcion: Cortes Perfilados  Imagen del barbero: archivo.jpg | - El sistema envía los datos correctamente - Se muestra mensaje de éxito - Se limpian los campos del formulario | La campaña se crea en la base de datos y se muestra mensaje de éxito |
| CP- 02 | Creación exitosa de cuenta con todos los datos incluida imagen | Nombre Barbero: "Fide"  Email: [fide@gmail.com](mailto:fide@gmail.com)  Contraseña: fide12345  Descripción: Cortes Lindos  Imagen del barbero: archivo2.jpg | - El sistema envía los datos correctamente - Se muestra mensaje de éxito - Se limpian los campos del formulario | La campaña se crea en la base de datos y se muestra mensaje de éxito |
| CP- 03 | Intentar crear cuenta sin completar campos  obligatorios | Omitir alguno o todos los campos requeridos | El formulario bloquea el envío y muestra indicaciones de campos requeridos | No se envía el formulario hasta completar los campos  requeridos |
| CP- 04 | Intentar crear cuenta con usuario ya creado | Nombre Barbero: "Cristian Rueda"  Email: [cris@gmail.com](mailto:cris@gmail.com)  Contraseña: cris12345  Descripcion: Cortes Perfilados  Imagen del barbero: archivo.jpg | El sistema debe validar que este usuario ya está creado y que no se puede crear | El sistema muestra un mensaje de error indicando que el usuario ya fue creado |
| CP- 05 | Simular error del servidor al crear campaña | Datos válidos, pero servidor no disponible | El sistema maneja el error y muestra mensaje adecuado | Se muestra mensaje de error indicando problema de  conexión |

|  |  |  |  | C |
| --- | --- | --- | --- | --- |

* 1. **Pruebas de Interfaz de Usuario**

| **ID** | **Caso de Prueba** | **Pasos** | **Resultado Esperado** | **Criterio de Aceptación** |
| --- | --- | --- | --- | --- |
| UI- 01 | Previsualización de imagen funciona  correctamente | 1. Seleccionar archivo de imagen2. Observar  área de previsualización | La imagen seleccionada se muestra en la  previsualización | La previsualización muestra correctamente la  imagen seleccionada |
| UI- 03 | Response en dispositivos móviles | Acceder al formulario desde un dispositivo móvil | El formulario se adapta  correctamente al tamaño de pantalla | Todos los elementos son  visibles y utilizables en pantalla pequeña |
| UI- 04 | Retroalimentación visual en campos obligatorios | Intentar enviar formulario  sin completar campos obligatorios | Se resaltan visualmente  los campos obligatorios no completados | Los campos requeridos  muestran indicación visual clara |
| UI- 05 | Visualización correcta de mensajes de alerta | Completar diferentes acciones que generan  alertas | Las alertas de SweetAlert2 se muestran  correctamente | Las alertas son legibles y contienen la información  adecuada |
| C |  |  |  | C |

1. **Pruebas de Seguridad**

| **ID** | **Caso de Prueba** | **Descripción** | **Resultado Esperado** | **Criterio de Aceptación** |
| --- | --- | --- | --- | --- |
| SEC- 01 | Validación de token de usuario | Manipular o eliminar el token de localStorage | El sistema detecta token  inválido y maneja adecuadamente el error | El usuario es redirigido a la  página de login o se muestra mensaje de error |
| SEC- 02 | Inyección de código en campos  de texto | Intentar ingresar código HTML/JavaScript en  campos de texto | El sistema sanitiza la entrada o rechaza  contenido malicioso | El contenido malicioso no se ejecuta ni almacena |
| SEC- 03 | Intento de acceso sin autenticación | Acceder directamente a la  ruta sin tener un token válido | El sistema restringe el acceso | Usuario no autenticado no  puede acceder al componente |
| C |  |  |  | C |

1. **Pruebas de Rendimiento**

| **ID** | **Caso de Prueba** | **Descripción** | **Resultado Esperado** | **Criterio de**  **Aceptación** |
| --- | --- | --- | --- | --- |
| PERF-  01 | Tiempo de carga del  componente | Medir tiempo de carga  inicial del componente | El componente se carga en  menos de 1 segundos | Tiempo de carga <  1ms |
| PERF- 02 | Tiempo de respuesta al enviar formulario | Medir tiempo desde envío  hasta recepción de respuesta | La respuesta se recibe en menos de 1 segundos | Tiempo de respuesta  < 1s |
| PERF- 03 | Tiempo de  previsualización de imagen | Medir tiempo desde  selección hasta visualización | La previsualización aparece en menos de 1 segundo | Tiempo de previsualización < 1s |
| C |  |  |  | C |

1. **Pruebas de Accesibilidad**

| **ID** | **Caso de Prueba** | **Descripción** | **Resultado Esperado** | **Criterio de Aceptación** |
| --- | --- | --- | --- | --- |
| ACC- 01 | Navegación con teclado | Navegar por el  formulario usando solo el teclado | Todos los campos son accesibles mediante teclado | Navegación completa sin ratón es posible |
| ACC- 02 | Compatibilidad con lectores de pantalla | Probar componente con lector de pantalla | El lector de pantalla identifica correctamente todos los  elementos | Todos los elementos tienen etiquetas  accesibles |
| ACC- 03 | Contraste de colores | Revisar contraste entre texto y fondo | El contraste cumple con estándares WCAG | Proporción de contraste según WCAG AA o  superior |
| C |  |  |  | C |

1. **Pruebas de Integración**

| **ID** | **Caso de Prueba** | **Descripción** | **Resultado Esperado** | **Criterio de Aceptación** |
| --- | --- | --- | --- | --- |
| INT- 01 | Integración con API backend | Verificar que los datos enviados son recibidos correctamente por el  backend | La API procesa correctamente los datos | La API confirma recepción correcta de datos |
| INT- 02 | Integración con sistema de autenticación | Verificar que el token de  autenticación es utilizado correctamente | El backend valida  correctamente el token | No hay errores de autenticación |
| INT- 03 | Integración con sistema de almacenamiento de  imágenes | Verificar que las imágenes se almacenan correctamente | Las imágenes son almacenadas y  accesibles | Las imágenes se pueden recuperar  posteriormente |
| C |  |  |  | C |

1. **Conclusiones**

El componente GestionarBarberos cumple con su función ya que creara correctamente dependiendo el caso la Cuenta del BARBERO.

1. **Anexos**

# Diagrama de Flujo del Proceso

I INICIO

INGRESAR FORMULARIO

INGRESAR CAMPOS REQUERIDOS

CORRECTOS LOS CAMPOS

CAMPOS INCOMPLETO SI HACE FALTA

AGREGAR IMAGEN, PREVISUALIZACION DE IMAGEN CORRECTAMENTE

AÑADIR EL BARBERO

VISUALIZACION DEL BARBERO CORRECTAMENTE

# Entorno de Pruebas Recomendado

## Navegadores a probar:



Google Chrome (última versión)



Mozilla Firefox (última versión)  Microsoft Edge (última versión)  Brave (última versión)



## Dispositivos a probar:

 Desktop (resoluciones 1920x1080, 1366x768)  Tablet (iPad, resolución 768x1024)

 Móvil (iPhone, Galaxy Alcatel, resoluciones 375x667, 414x896)

## Herramientas recomendadas:

 React Testing Library para pruebas unitarias  Cypress para pruebas end-to-end

 Lighthouse para evaluación de rendimiento  WAVE o axe para pruebas de accesibilidad

# Código Fuente del Componente

import NavbarAdmin from '../../Components/NavbarAdmin';

import SidebarAdmin from '../../Components/SidebarAdmin';

import Swal from 'sweetalert2';

import React, { useEffect, useState } from 'react';

import axios from 'axios';

export default function GestionarBarberos() {

  const [imagePreview, setImagePreview] = useState('');

  const [imagePreviewEdit, setImagePreviewEdit] = useState('');

  const [barberos, setBarberos] = useState([]);

  const [isPasswordVisible, setIsPasswordVisible] = useState(false);

  const [barbero, setBarbero] = useState({

    nombre: "",

    email: "",

    contrasena: "",

    descripcion: "",

    foto: null,

  });

  const [barberoEdit, setBarberoEdit] = useState({

    id\_usuario: "",

    nombre\_usuario: "",

    email: "",

    descripcion: "",

    foto: null,

  });

  // Crear barberos

  const handleSubmit = async (*e*) => {

*e*.preventDefault();

    try {

      const formData = **new** *FormData*();

      formData.append('nombre', barbero.nombre);

      formData.append('email', barbero.email);

      formData.append('contrasena', barbero.contrasena);

      formData.append('descripcion', barbero.descripcion);

      formData.append('foto', barbero.foto);

      const res = await axios.post(`http://localhost:8080/CreateBarberos`, formData);

      if (res.status === 200) {

*Swal*.fire({

          icon: 'success',

          title: res.data,

          timer: 1000,

          customClass: {

            popup: "dark-theme-popup bg-dark antonparabackend ",

          },

        }).then(() => {

          window.location.reload(0);

        });

      }

    } catch (err) {

      console.log(err);

*Swal*.fire({

        icon: 'error',

        title: 'Error',

        text: err.response.data,

        customClass: {

          popup: "dark-theme-popup bg-dark antonparabackend ",

        },

      });

    }

  };

  // Editar barberos

  const handleSubmitEdit = async (*e*) => {

    e.preventDefault();

    try {

      const formData = **new** FormData();

      formData.append('nombre', barberoEdit.nombre\_usuario);

      formData.append('email', barberoEdit.email);

      formData.append('descripcion', barberoEdit.descripcion);

      formData.append('foto', barberoEdit.foto);

      const res = await axios.put(`http://localhost:8080/UpdateBarberos/${barberoEdit.id\_usuario}`, formData);

      if (res.status === 200) {

        Swal.fire({

          icon: 'success',

          title: res.data,

          timer: 1000,

          customClass: {

            popup: "dark-theme-popup bg-dark antonparabackend ",

          },

        }).then(() => {

          window.location.reload(0);

        });

      }

    } catch (err) {

      console.log(err);

      Swal.fire({

        icon: 'error',

        title: 'Error',

        text: err.response.data,

        customClass: {

          popup: "dark-theme-popup bg-dark antonparabackend ",

        },

      });

    }

  };

  // Handle change para añadir

  const handleChange = (*e*) => {

    setBarbero(*prev* => ({ ...prev, [e.target.name]: e.target.value }));

  };

  // Handle change para editar

  const handleChangeEdit = (*e*) => {

    setBarberoEdit(*prev* => ({ ...prev, [e.target.name]: e.target.value }));

  };

  // Handle change para añadir foto

  const handleFileChange = (*e*) => {

    const selectedFile = e.target.files[0];

    if (selectedFile && selectedFile.type.startsWith('image/')) {

      setBarbero({ ...barbero, [e.target.name]: selectedFile });

      setImagePreview(URL.createObjectURL(selectedFile));

    } else {

      Swal.fire({

        icon: 'error',

        title: 'Error',

        text: 'Por favor, seleccione un archivo de imagen válido.',

        customClass: {

          popup: "dark-theme-popup bg-dark antonparabackend ",

        },

      });

    }

  };

  // Handle change para editar foto

  const handleFileChangeEdit = (*e*) => {

    const selectedFile = e.target.files[0];

    if (selectedFile && selectedFile.type.startsWith('image/')) {

      setBarberoEdit({ ...barberoEdit, [e.target.name]: selectedFile });

      setImagePreviewEdit(URL.createObjectURL(selectedFile));

    } else {

      Swal.fire({

        icon: 'error',

        title: 'Error',

        text: 'Por favor, seleccione un archivo de imagen válido.',

        customClass: {

          popup: "dark-theme-popup bg-dark antonparabackend ",

        },

      });

    }

  };

  const DeleteBarberos = async (*id*) => {

    try {

      const confirm = await Swal.fire({

        title: '¿Estas seguro de borrar este barbero?',

        text: "No podrás revertir esta acción",

        icon: 'question',

        showCancelButton: true,

        confirmButtonColor: '#3085d6',

        cancelButtonColor: '#d33',

        confirmButtonText: 'Sí, borrar',

        customClass: {

          popup: "dark-theme-popup bg-dark antonparabackend ",

        },

      });

      if (!confirm.isConfirmed) {

        return;

      }

      const res = await axios.delete(`http://localhost:8080/DeleteBarberos/${id}`);

      console.log(res);

      if (res.status === 200) {

        Swal.fire({

          icon: 'success',

          title: res.data,

          customClass: {

            popup: "dark-theme-popup bg-dark antonparabackend ",

          },

        })

          .then(() => {

            window.location.reload(0);

          });

      }

    } catch (error) {

      console.log(error);

      Swal.fire({

        icon: 'error',

        title: 'Error al borrar',

        text: error.response.data,

        customClass: {

          popup: "dark-theme-popup bg-dark antonparabackend ",

        },

      });

    }

  };

  const fetchBarberos = async () => {

    try {

      const res = await axios.get("http://localhost:8080/GetBarberos");

      setBarberos(res.data);

    } catch (err) {

      console.log("Error al obtener los datos:", err);

    }

  };

  useEffect(() => {

    fetchBarberos();

  }, []);

  const openEditModal = (*barbero*) => {

    setBarberoEdit(barbero);

    setImagePreviewEdit(`http://localhost:8080/imagesBarbero/${barbero.Foto}`);

  };

  return (

    <div>

      <*NavbarAdmin* />

      <*SidebarAdmin* />

      <div *className*='contenido2' *id*='GestionarBarberos'>

        <p *className*='text-center  mt-5 text-white display-6 bebas col-sm-12 col'>HOLA, <span *className*='text-danger'>ADMINISTRADOR</span> |AQUI PODRAS EDITAR, AÑADIR Y ELIMINAR BARBEROS</p>

        <div *className*="d-flex justify-content-end mt-3 p-5 mx-5">

          <button *type*="button" *className*="btn btn-danger" *data-bs-toggle*="modal" *data-bs-target*="#AñadirModal" *data-bs-whatever*="@mdo" >Añadir</button>

        </div>

        <div *className*=' text-center row col col-sm-12 justify-content-end'>

          <div *className*="table-responsive">

            <table *className*="table  table-bordered table-responsive table-dark table-hover mt-4 container p-5">

              <thead>

                <tr *className*='bg-white'>

                  <th *scope*="col" *className*='display-6 bebas p-4 text-danger'>Nombre</th>

                  <th *scope*="col" *className*='display-6 bebas p-4 '>Email</th>

                  <th *scope*="col" *className*='display-6 bebas p-4 '>Descripcion</th>

                  <th *scope*="col" *className*='display-6 bebas p-4'>imagen</th>

                  <th *scope*="col" *className*='text-warning display-6 bebas p-4'>Acciones</th>

                </tr>

              </thead>

              <tbody *className*='table-group-divider'>

                {barberos.map((*barbero*) => (

                  <tr *key*={barbero.id\_usuario}>

                    <td *className*='w-25 text-center p-5'>{barbero.nombre\_usuario}</td>

                    <td *className*='w-25 text-center p-5'>{barbero.email}</td>

                    <td *className*='w-25 text-center p-5'>{barbero.descripcion}</td>

                    <td><img *src*={`http://localhost:8080/imagesBarbero/${barbero.Foto}`} *className*='img-fluid zoomhover2' *alt*="" /></td>

                    <td>

                      <div *className*="d-flex justify-content-center mt-5 mx-5">

                        <button *type*="button" *className*="btn btn-outline-warning me-5" *onClick*={() => openEditModal(barbero)} *data-bs-toggle*="modal" *data-bs-target*="#exampleModal" *data-bs-whatever*="@mdo">

                          <i *className*='bi bi-pencil-fill text-white'></i>

                        </button>

                        <button *className*='btn btn-outline-danger' *onClick*={() => DeleteBarberos(barbero.id\_usuario)}>

                          <i *className*="bi bi-trash-fill"  ></i>

                        </button>

                      </div>

                    </td>

                  </tr>

                ))}

              </tbody>

            </table>

          </div>

        </div>

        {/\* MODAL EDIT \*/}

        <div *className*="modal fade" *id*="exampleModal" *tabIndex*="-1" *aria-labelledby*="exampleModalLabel" *aria-hidden*="true">

          <div *className*="modal-dialog">

            <div *className*="modal-content bg-dark">

              <form *onSubmit*={handleSubmitEdit}>

                <div *className*="modal-header">

                  <h1 *className*="modal-title fs-5 text-white" *id*="exampleModalLabel">EDITAR</h1>

                  <button *type*="button" *className*="btn-close bg-white" *data-bs-dismiss*="modal" *aria-label*="Close"></button>

                </div>

                <div *className*="modal-body d-flex justify-content-center ">

                  <div *className*="card bg-dark" *style*={{ width: '10rem' }}>

                    <img *src*={imagePreviewEdit || `http://localhost:8080/imagesBarbero/${barberoEdit.foto}`} *style*={{ width: '100%', height: '100%', objectFit: 'cover' }} *className*='img-fluid text-white rounded' *alt*="Imagen Barbero" />

                  </div>

                </div>

                <div *className*="modal-body">

                  <div *className*="mb-3">

                    <label *htmlFor*="recipient-name" *className*="col-form-label text-white">Nombre Barbero</label>

                    <input *required* *type*="text" *className*="form-control bg-dark text-white" *pattern*='^[A-Za-z\s]+$' *id*="recipient-name" *value*={barberoEdit.nombre\_usuario} *name*='nombre\_usuario' *onChange*={handleChangeEdit} *placeholder*='Escriba un nombre' />

                  </div>

                  <div *className*="mb-3">

                    <label *htmlFor*="recipient-name" *className*="col-form-label text-white">Email</label>

                    <input *required* *type*="email" *className*="form-control bg-dark text-white" *id*="recipient-name" *value*={barberoEdit.email} *name*='email' *onChange*={handleChangeEdit} *placeholder*='Email' />

                  </div>

                  <div *className*="mb-3">

                    <label *htmlFor*="recipient-name" *className*="col-form-label text-white">Descripcion</label>

                    <input *required* *type*="text" *className*="form-control bg-dark text-white" *id*="recipient-name" *value*={barberoEdit.descripcion} *name*='descripcion' *onChange*={handleChangeEdit} *placeholder*='Escriba una Descripcion' />

                  </div>

                  <p *className*="text-white antonparabackend"> Imagen Del Barbero</p>

                  <div *className*="input-group">

                    <input

*name*="foto"

*accept*="image/\*"

*type*="file"

*className*="form-control bg-dark text-white"

*id*="inputGroupFile04"

*onChange*={handleFileChangeEdit}

                    />

                  </div>

                </div>

                <div *className*="modal-footer">

                  <button *type*="button" *className*="btn btn-secondary" *data-bs-dismiss*="modal" >Close</button>

                  <button *type*="submit" *className*="btn btn-danger" >Editar</button>

                </div>

              </form>

            </div>

          </div>

        </div>

        {/\* MODAL AÑADIR \*/}

        <div *className*="modal fade" *id*="AñadirModal" *tabIndex*="-1" *aria-labelledby*="exampleModalLabel" *aria-hidden*="true">

          <div *className*="modal-dialog">

            <div *className*="modal-content bg-dark row">

              <form *onSubmit*={handleSubmit}>

                <div *className*="modal-header">

                  <h1 *className*="modal-title fs-5 text-white" *id*="exampleModalLabel">Añadir</h1>

                  <button *type*="button" *className*="btn-close bg-white" *data-bs-dismiss*="modal" *aria-label*="Close"></button>

                </div>

                <div *className*="modal-body d-flex justify-content-center ">

                  <div *className*="card bg-dark" *style*={{ width: '10rem' }}>

                    <img *src*={imagePreview || ''} *style*={{ width: '100%', height: '100%', objectFit: 'cover' }} *className*='img-fluid text-white rounded' *alt*="Imagen Barbero" />

                  </div>

                </div>

                <div *className*="mb-3">

                  <label *htmlFor*="recipient-name" *className*="col-form-label text-white">Nombre Barbero</label>

                  <input *required* *type*="text" *className*="form-control bg-dark text-white" *id*="recipient-name" *name*='nombre' *onChange*={handleChange} *placeholder*='Escriba un Nombre' />

                </div>

                <div *className*="mb-3">

                  <label *htmlFor*="recipient-name" *className*="col-form-label text-white">Email</label>

                  <input *required* *type*="email" *className*="form-control bg-dark text-white" *id*="recipient-name" *name*='email' *onChange*={handleChange} *placeholder*='Escriba un Email' />

                </div>

                <label *htmlFor*="recipient-name" *className*="col-form-label text-white">Contraseña</label>

                <div *className*="mb-3 input-group">

                  <input *required* *className*="form-control bg-dark text-white" *id*="recipient-name" *name*='contrasena' *onChange*={handleChange} *placeholder*='Escriba una Contraseña' *type*={isPasswordVisible ? "text" : "password"} />

                  <button

*type*="button"

*onClick*={() => setIsPasswordVisible((*prev*) => !prev)}

*className*=" bg-dark rounded-end"

                  >

                    {isPasswordVisible ? " 🙉" : "🙈"}

                  </button>

                </div>

                <div *className*="mb-3">

                  <label *htmlFor*="recipient-name" *className*="col-form-label text-white">Descripcion</label>

                  <input *required* *type*="text" *className*="form-control bg-dark text-white" *id*="recipient-name" *name*='descripcion' *onChange*={handleChange} *placeholder*='Escriba una Descripcion' />

                </div>

                <p *className*="text-white antonparabackend"> Imagen Del Barbero</p>

                <div *className*="input-group">

                  <input

*required*

*name*="foto"

*accept*="image/\*"

*type*="file"

*className*="form-control bg-dark text-white "

*id*="inputGroupFile04"

*onChange*={handleFileChange}

                  />

                </div>

                <div *className*="modal-footer mt-4">

                  <button *type*="button" *className*="btn btn-secondary" *data-bs-dismiss*="modal">Close</button>

                  <button *type*="submit" *className*="btn btn-danger">Añadir</button>

                </div>

              </form>

            </div>

          </div>

        </div>

      </div>

    </div>

  );

}