INFORME DE TEST UNITARIOS FRONTEND – SPRINT IV

Los tests unitarios en frontend se están realizando con base en el framework Jest. Para este sprint se inició con su implementación en los Components que se han creado hasta el momento, por lo que a continuación detallaré lo realizado.

- **Reservas**

  En esta parte se realizará la validación del componente CompletedBookings. El objetivo es verificar los diferentes estados de reserva acuerdo con los datos que elegidos del usuario.

Nombre y descripción	Código del Test	Resultado
Renders CompletedBookings componente: Valida el renderizado inicial verificando si los datos se obtienen correctamente y se muestran en la tabla	jest.mock('../config', () => ({ axiosInstance: { get: jest.fn(), delete: jest.fn(), }, })); describe('CurrentBooking', () => { test('fetches data and renders table', async () => { const mockData = [{ id: 1, nombreCancha: 'Cancha 1', nombreUser: 'Usuario 1', fecha: '2023-07-01', completado: true }, { id: 2, nombreCancha: 'Cancha 2', nombreUser: 'Usuario 2', fecha: '2023-07-02', completado: false },]; axiosInstance.get.mockResolvedValueOnce({ data: mockData }); render(<CurrentBooking />); await waitFor(() => { expect(screen.getByText('Cancha 1')).toBeInTheDocument(); expect(screen.getByText('Usuario 1')).toBeInTheDocument(); expect(screen.getByText('2023-07-01')).toBeInTheDocument(); expect(screen.getByText('Completado')).toBeInTheDocument(); }); });	Correcta ejecución

Renders delete action : valida la acción de eliminación haciendo clic en el botón "Eliminar". Verificamos que se haya llamado a la función con el ID correcto y que se haya llamado para volver a obtener los datos actualizados.	`test('handles delete action', async () => {` `  const mockData = [` `  { id: 1, nombreCancha: 'Cancha 1', nombreUser: 'Usuario 1', fecha: '2023-07-01', completado: true },` `  ];` `  axiosInstance.get.mockResolvedValueOnce({ data: mockData });` `  axiosInstance.delete.mockResolvedValueOnce({ data: 'Delete successful' });` `  render(<CurrentBooking />);` `  await waitFor(() => {` `  expect(screen.getByText('Cancha 1')).toBeInTheDocument();` `  });` `  fireEvent.click(screen.getByText('Eliminar'));` `  await waitFor(() => {` `expect(axiosInstance.delete).toHaveBeenCalledWith('/all/deleteturno/1');` `  expect(axiosInstance.get).toHaveBeenCalledTimes(2);` `  });` `  });` `});`	Correcta ejecución

- **CurrentBookings**

  En esta parte se realizará la validación del componente CurrentBookings. El objetivo es verificar el renderizado de la reserva que está predeterminada por el usuario:

Nombre y descripción	Código del Test	Resultado
renders CurrentBookings component with the correct data: valida si los datos se	**`jest.mock('../config', () => ({`** **`  axiosInstance: {`** **`  get: jest.fn(),`**	Correcta ejecución

obtienen correctamente y se muestran en la tabla. Luego, verificamos que los elementos esperados se encuentren en la pantalla después de esperar a que se resuelva la promesa.	**delete: jest.fn(),** **},** **}));** **describe('CurrentBooking', () => {** **test('fetches data and renders table', async () => {** **const mockData = [** **{ id: 1, nombreCancha: 'Cancha 1', nombreUser: 'Usuario 1', fecha: '2023-07-01', completado: true },** **{ id: 2, nombreCancha: 'Cancha 2', nombreUser: 'Usuario 2', fecha: '2023-07-02', completado: false },** **];** **axiosInstance.get.mockResolvedValueOnce({ data: mockData });** **render(<CurrentBooking />);** **await waitFor(() => {** **expect(screen.getByText('Cancha 1')).toBeInTheDocument();** **expect(screen.getByText('Usuario 1')).toBeInTheDocument();** **expect(screen.getByText('2023-07-01')).toBeInTheDocument();** **expect(screen.getByText('Completado')).toBeInTheDocument();** **});** **});**	
renders CurrentBookings component with the delete data: valida si la acción de eliminación se maneja correctamente. Luego, hacemos clic en el botón "Eliminar" y verificamos que se haya llamado con el ID correcto y que se haya actualizado los datos.	**test('handles delete action', async () => {** **const mockData = [** **{ id: 1, nombreCancha: 'Cancha 1', nombreUser: 'Usuario 1', fecha: '2023-07-01', completado: true },** **];** **axiosInstance.get.mockResolvedValueOnce({ data: mockData });** **axiosInstance.delete.mockResolvedValueOnce({ data: 'Delete successful' });** **render(<CurrentBooking />);**	Correcta ejecución

	await waitFor(() => { **expect(screen.getByText('Cancha 1')).toBeInTheDocument();** **});** **fireEvent.click(screen.getByText('Eliminar'));** **await waitFor(() => {** **expect(axiosInstance.delete).toHaveBeenCalledWith('/all/deleteturno/1');** **expect(axiosInstance.get).toHaveBeenCalledTimes(2);** **});** **});** **});**	

- **Book**

  En esta parte se realizará la validación del componente Book. El objetivo es verificar el renderizado del componente, incluyendo los datos que debe contener de acuerdo con lo requerido del usuario.

Nombre y descripción	Código del Test	Resultado
renders Book component with user data : valida el renderizado de los datos del usuario se obtengan correctamente y se muestren en el formulario.	`jest.mock('../config', () => ({` `  axiosInstance: {` `  get: jest.fn(),` `  post: jest.fn(),` `  },` `}));` `describe('Book', () => {` `  test('fetches user data and renders form', async () => {` `  const mockUser = {` `  nombre: 'John',` `  apellido: 'Doe',` `  email: 'john.doe@example.com',` `  };` `  axiosInstance.get.mockResolvedValueOnce({ data: mockUser });`	Correcta ejecución

	`` render(<Book detail={{ canchaDTO: {} }} selectedDate={{ date: '2023-07-01' }} />); await waitFor(() => { expect(screen.getByText('Nombre: John')).toBeInTheDocument(); expect(screen.getByText('Apellido: Doe')).toBeInTheDocument(); expect(screen.getByDisplayValue('john.doe@example.com')).toBeInTheDocument(); expect(screen.getByText('Fecha y hora de la reserva')).toBeInTheDocument(); expect(screen.getByText('Datos de la cancha')).toBeInTheDocument(); expect(screen.getByText('Reservar')).toBeInTheDocument(); }); }); ```	
renders Book component when the reserved is correct : verifica la funcionalidad de confirmación de reserva haciendo clic en el botón "Reservar" y asegura que se realice la reserva correcta	``` test('handles book confirmation', async () => { const mockUser = { nombre: 'John', apellido: 'Doe', email: 'john.doe@example.com', }; axiosInstance.get.mockResolvedValueOnce({ data: mockUser }); axiosInstance.post.mockResolvedValueOnce({}); render(<Book detail={{ canchaDTO: {} }} selectedDate={{ date: '2023-07-01' }} />); await waitFor(() => { expect(screen.getByText('Nombre: John')).toBeInTheDocument(); }); fireEvent.click(screen.getByText('Reservar')); await waitFor(() => { expect(axiosInstance.post).toHaveBeenCalledWith('/user/createturno', ```	Correcta ejecución

```
        {
          horas: undefined, // Set the appropriate value here
          fecha: '2023-07-01',
          idCancha: '', // Set the appropriate value here
        },
        {
          params: {
            token: localStorage.getItem('jwt'),
          },
        }
      );
      expect(screen.getByText('Reserva creada con
éxito')).toBeInTheDocument();
    });
  });
});
```