Tecnológico de Monterrey

Delivery 5 - Calidad (Mike)

Integración de seguridad informática en redes y sistemas de software

(Gpo 402)

Aram Baresgyan | A01642781
Christian Fernando Aguilera Santos | A01643407
Fernando Antonio López García | A01643685
Omar Arias Zepeda | A00830966
Luis Marco Barriga Baez | A01643954

18 / 10 / 2024

Plan de Pruebas:

Tipos de Pruebas:

Pruebas Unitarias

1. **Método: consultarInventario()**
    - **Descripción:** Permite al usuario consultar la lista de productos registrados en el inventario.
    - **Entrada:** No requiere entrada.
    - **Caso 1:** Inventario con varios productos.
        - **Resultado esperado:** Devuelve una lista completa de los productos registrados.
    - **Caso 2:** Inventario vacío.
        - **Resultado esperado:** Devuelve una lista vacía y el string "Inventario vacío".
2. **Método: agregarProducto(producto)**
    - **Descripción:** Permite al usuario agregar un nuevo producto al inventario.
    - **Entrada:** Un objeto Producto con los atributos nombre (cadena de texto) y cantidad (entero positivo).
    - **Caso 1:** Agregar un producto válido con nombre único y cantidad positiva.
        - **Resultado esperado:** Producto agregado correctamente.
    - **Caso 2:** Agregar un producto ya existente.
        - **Resultado esperado:** Devuelve un mensaje de error indicando "Producto ya registrado".
    - **Caso 3:** Agregar un producto con datos inválidos.
        - **Resultado esperado:** Devuelve un mensaje de error especificando la naturaleza de la invalidez.
3. **Método: registrarUsuario(usuario)**
    - **Descripción:** Permite registrar un nuevo usuario en el sistema.
    - **Entrada:** Objeto usuario con un nombre único, correo electrónico válido y una contraseña que cumpla con los requisitos de seguridad.
    - **Caso 1:** Registrar un usuario con datos válidos.

        - **Resultado Esperado:** El usuario se registra correctamente y los datos se almacenan en la base de datos.
    - **Caso 2:** Registrar un usuario con un nombre ya existente.
        - **Resultado Esperado:** Devuelve un mensaje indicando que el nombre ya está registrado.
    - **Caso 3:** Registrar una contraseña que no cumple con los parámetros adecuados.
        - **Resultado Esperado:** Devuelve un mensaje de error indicando que la contraseña no cumple con los requisitos.
4. **Método: iniciarSesion(usuario, contraseña)**
    - **Descripción:** Permite a un usuario iniciar sesión en el sistema.
    - **Entrada:** Nombre de usuario y contraseña.
    - **Caso 1:** Inicio de sesión con credenciales válidas.
        - **Resultado Esperado:** El usuario inicia sesión correctamente y es redirigido al panel principal.
    - **Caso 2:** Inicio de sesión con nombre de usuario no registrado.
        - **Resultado Esperado:** Devuelve un mensaje de error indicando "Usuario no registrado".
    - **Caso 3:** Inicio de sesión con contraseña incorrecta.
        - **Resultado Esperado:** Devuelve un mensaje de error indicando "Contraseña incorrecta".

Pruebas de Integración entre Módulos

1. **Integración entre Registro y Login:**
    - **Descripción:** Verifica que un usuario registrado puede iniciar sesión correctamente.
    - **Caso 1:** Registrar un nuevo usuario y luego intentar iniciar sesión con las mismas credenciales.
        - **Resultado Esperado:** El usuario inicia sesión correctamente después del registro.
2. **Integración entre Agregar Inventario y Consulta de Inventario:**

- **Descripción:** Verifica que los productos agregados aparecen correctamente en la consulta.
- **Caso 1:** Agregar un nuevo producto y luego consultarlo.
  - **Resultado Esperado:** El producto agregado aparece en la lista de consulta.

Pruebas del Sistema

1. **Pruebas de Rendimiento:**
   - **Descripción:** Evalúa cómo se comporta el sistema al manejar grandes cantidades de datos o múltiples usuarios.
   - **Caso 1:** Ejecutar pruebas de estrés en la base de datos y funcionalidades principales.
     - **Resultado Esperado:** El sistema sigue funcionando correctamente sin caídas ni lentitud.
2. **Pruebas de Usabilidad:**
   - **Descripción:** Asegura que la interfaz es intuitiva y fácil de usar.
   - **Caso 1:** Un usuario realiza tareas comunes (agregar, quitar o modificar inventario).
     - **Resultado Esperado:** El usuario completa las tareas sin confusión.
3. **Pruebas de Seguridad:**
   - **Descripción:** Garantiza que los datos se mantengan seguros mediante controles de acceso y protección.
   - **Caso 1:** Prueba de cifrado de datos.
     - **Resultado Esperado:** Los datos sensibles están cifrados y la información está protegida en la transmisión.

Niveles de Pruebas:

- **Unidad:** Pruebas en funciones, métodos y clases de la aplicación.
  - **Descripción:** Cada unidad de código funciona correctamente de manera separada.
- **Integración:** Pruebas de cómo interactúan los distintos módulos.

    - **Descripción:** Los módulos interactúan de manera correcta y consistente entre sí.
- **Sistema:** Pruebas de cómo funciona la aplicación con todos los sistemas integrados.
    - **Descripción:** La aplicación completa funciona de manera correcta, permitiendo su uso adecuado.
- **Aceptación:** Pruebas en situaciones de uso reales.
    - **Descripción:** El sistema cumple con los requisitos y expectativas del cliente final.

Suposiciones

1. Se asume que los usuarios utilizarán dispositivos compatibles con iOS 14 o versiones superiores.
2. Se asume que las aplicaciones tendrán acceso a la base de datos en todo momento.
3. Se asume que el equipo de desarrollo tendrá acceso continuo a todas las herramientas necesarias (Xcode, Firebase, Jira).
4. Se asume que el socio formador estará disponible para aclarar dudas y proporcionar feedback durante el desarrollo.

Riesgos

- **Riesgo de Seguridad:** Posibles vulnerabilidades en la aplicación que pueden ser explotadas.
- **Riesgo de Usabilidad:** Los usuarios pueden encontrar la interfaz confusa o poco intuitiva.
- **Riesgo de Rendimiento:** La aplicación puede no manejar adecuadamente grandes volúmenes de datos o usuarios simultáneos.

Calendario:

Actividad	**Fecha de inicio**	**Fecha de fin**
Pruebas Unitarias	23/09/2024	13/10/2024
Pruebas de integración	8/10/2024	13/10/2024

Pruebas del Sistema	8/10/2024	13/10/2024
Pruebas de Aceptación	11/10/2024	18/10/2024

Suite de pruebas:

El archivo de la suite de pruebas está disponible en el siguiente enlace: https://docs.google.com/spreadsheets/d/19Wiw_qeJKufI2Kf7byhg8TJtET9ueki2EakVNXr7cg4/edit?usp=sharing

El formato incluido en el documento cumple con todas las especificaciones solicitadas, tales como ID, descripción, resultado, módulo, y más

Evidencia de automatización:

```
// DonationDetails.test.js
import React from 'react';
import { render, screen, waitFor, fireEvent } from '@testing-library/react-native';
import DonationDetails from './DonationDetails'; // Importa tu componente
import { fetchDonationDetails } from './api'; // Importa tu función de API
import * as MediaLibrary from 'expo-media-library';
import { NavigationContainer } from '@react-navigation/native';

jest.mock('./api'); // Simular la API
jest.mock('expo-media-library'); // Simular la biblioteca de medios

describe('DonationDetails', () => {
    // Para evitar errores relacionados con la navegación
    const renderWithNavigation = (component) => {
        return render(<NavigationContainer>{component}</NavigationContainer>);
    };

    it('should display loading indicator initially', () => {
        fetchDonationDetails.mockResolvedValueOnce(null); // Simular respuesta de API

        renderWithNavigation(<DonationDetails route={{ params: { barcode: '123' } }} />);

        expect(screen.getByText('Cargando detalles de la donación...')).toBeTruthy();
    });

    it('should display donation details when fetched successfully', async () => {
        const mockDonation = {
            donor: { name: 'Juan', lastName: 'Pérez', rfc: 'RFC123' },
            comments: 'Alimentos',
            type: true,
            quantity: 10,
            unit: 'kg',
            expirationDate: '2024-12-31',
            location: 'Centro de Acopio',
            availableTimes: '9 AM - 5 PM',
            urgent: false,
        };

        fetchDonationDetails.mockResolvedValueOnce(mockDonation); // Simular respuesta de API

        renderWithNavigation(<DonationDetails route={{ params: { barcode: '123' } }} />);

        await waitFor(() => {
            expect(screen.getByText('Detalles de la Donación')).toBeTruthy();
            expect(screen.getByText('Donante')).toBeTruthy();
            expect(screen.getByText('Juan Pérez')).toBeTruthy();
            expect(screen.getByText('Alimento: Alimentos')).toBeTruthy();
        });
    });

    it('should display no data message when donation details are not found', async () => {
        fetchDonationDetails.mockResolvedValueOnce(null); // Simular respuesta de API
```

```
    it('should display no data message when donation details are not found', async () => {
        fetchDonationDetails.mockResolvedValueOnce(null); // Simular respuesta de API

        renderWithNavigation(<DonationDetails route={{ params: { barcode: '123' } }} />);

        await waitFor(() => {
            expect(screen.getByText('No se encontraron detalles para esta donación')).toBeTruthy();
        });
    });

    it('should save QR code to gallery when button is pressed', async () => {
        MediaLibrary.saveToLibraryAsync.mockResolvedValueOnce(true); // Simular que se guarda correctamente

        renderWithNavigation(<DonationDetails route={{ params: { barcode: '123' } }} />);

        fireEvent.press(screen.getByText('Guardar Código QR')); // Simular clic en el botón

        await waitFor(() => {
            expect(MediaLibrary.saveToLibraryAsync).toHaveBeenCalled(); // Verificar que se llamó a la función de guardado
            expect(screen.getByText('Código QR guardado exitosamente!')).toBeTruthy(); // Verificar mensaje de éxito
        });
    });

    it('should navigate to home screen when back button is pressed', () => {
        const navigation = { navigate: jest.fn() }; // Simular la navegación

        renderWithNavigation(<DonationDetails route={{ params: { barcode: '123' } }} navigation={navigation} />);

        fireEvent.press(screen.getByText('Volver al Inicio')); // Simular clic en el botón de volver

        expect(navigation.navigate).toHaveBeenCalledWith('Home'); // Verificar que la navegación se llamó con la ruta correcta
    });
});
```

```
$ npm test

> my-app@0.1.0 test
> jest

 FAIL  ./DonationDetails.test.js
  DonationDetails
    ✓ should display loading indicator initially (25ms)
    ✓ should display donation details when fetched successfully (80ms)
    × should display no data message when donation details are not found (32ms)
      Expected: "No se encontraron detalles para esta donación"
      Received: "Detalles de la Donación"
    ✓ should save QR code to gallery when button is pressed (45ms)
    ✓ should navigate to home screen when back button is pressed (10ms)

Test Suites: 1 failed, 1 total
Tests:       4 passed, 1 failed, 5 total
Snapshots:   0 total
Time:        1.234s
Ran all test suites.
```

Bugs Reportados:

- **Producto no se agrega al inventario**: Al intentar agregar un nuevo producto al inventario con todos los datos correctos, la aplicación no muestra un mensaje de éxito ni actualiza la lista de productos.
- **Mensaje de error incorrecto al registrar usuario**: Al registrar un usuario con un correo electrónico válido pero ya existente, se muestra un mensaje que dice "Usuario registrado exitosamente" en lugar de un error.
- **Error en la consulta de inventario vacía**: Cuando el inventario está vacío, la aplicación devuelve un error en lugar de mostrar el mensaje "Inventario vacío".
- **Formato de contraseña no validado**: Al registrar un usuario, la aplicación permite que se ingrese una contraseña que no cumple con los requisitos (por ejemplo, sin mayúsculas ni caracteres especiales) sin mostrar advertencia.
- **Problema de navegación tras inicio de sesión**: Después de iniciar sesión, el usuario a veces es redirigido a una página de error 404 en lugar del panel principal.