

| Plan de Pruebas |
| --- |
| *Proyecto: Gustalo APP* |
|  |
| **Revisión*: [99.99]*** |
| **[22/11/24]** |

| Responsable | Firma |
| --- | --- |
| Joaquin Carcamo | x |

Descripción:

Este documento garantiza la calidad del sistema desarrollado en el proyecto. Este sistema tiene como objetivo principal la gestión eficiente de inventarios y ventas para pequeños negocios, permitiendo la automatización de procesos, actualización en tiempo real del stock, y generación de alertas por productos en bajo inventario. El plan aborda las pruebas necesarias para validar la funcionalidad, integración, seguridad y desempeño del sistema antes de su implementación final.

#### **Objetivo del Plan**

El objetivo principal del plan de pruebas es:

* Identificar errores y defectos en el sistema antes de su puesta en marcha.
* Validar que todas las funcionalidades cumplen con los requisitos establecidos.
* Asegurar que el sistema opera correctamente bajo diferentes escenarios y condiciones.

#### **Alcance**

El alcance de este plan abarca las pruebas de todas las funcionalidades críticas del sistema, incluyendo:

* Registro y autenticación de usuarios (trabajadores y administradores).
* Gestión de inventarios y productos.
* Registro de ventas y actualización automática de stock.
* Alertas de inventario bajo.
* Generación de reportes en tiempo real.

### **3. Metodología**

* Descripción de la metodología utilizada para las pruebas (e.g., cascada, ágil, iterativa).
* Tipos de pruebas incluidas:
  + **Pruebas unitarias:** Evaluación de componentes individuales.
  + **Pruebas de integración:** Validación de interacción entre módulos
  + **Pruebas de estrés y carga:** Evaluar el desempeño bajo alta demanda.

**4.Casos de Prueba:**

Casos de Pruebas unitaria

| ID | P01 |
| --- | --- |
| Funcion/metodo | Login |
| Descripción | Al iniciar las pruebas comenzamos con un error que consiste que **NullInjectorError**: Este error significa que Angular no puede encontrar un proveedor para InjectionToken angularfire2.app.options, el cual es necesario para AngularFireAuth. |
| Precondiciones | Cambiaremos de injectionToken a AngularfireAuth con el que conecta a la base de datos. |
| Pasos de Prueba | Nos dirigimos a login.Page.spec.ts a cambiar el codigo y crea uno |
| Datos de entrada | import { TestBed } from '@angular/core/testing';  import { AngularFireModule } from '@angular/fire/compat';  import { AngularFireAuthModule } from '@angular/fire/compat/auth';  import { environment } from 'src/environments/environment';  import { LoginPage } from './login.page';  describe('LoginPage', () => {  beforeEach(async () => {  await TestBed.configureTestingModule({  imports: [  AngularFireModule.initializeApp(environment.firebaseConfig), // Configuración real de Firebase  AngularFireAuthModule, // Módulo de autenticación  ],  declarations: [LoginPage],  }).compileComponents();  });  it('should create', () => {  const fixture = TestBed.createComponent(LoginPage);  const app = fixture.componentInstance;  expect(app).toBeTruthy();  });  });  importamos (Angularfiremodule,AngularFiremoduleAuth) para permitir el uso de firebase que nuestra base de datos. Luego usamos Testbed lo cual nos proporciona los servicios de firebase, eliminando el error injectionToken |
| Resultado esperado | Jasmine debería quitar de 17 Failed(error) a 16 |
| Resultado Actual | de 17 errores cambio a 16 tenemos 5 success |
| Estado | Aprobado |

| ID | P02 |
| --- | --- |
| Funcion/metodo | * Gestionar categorías * Categorías Servicio, * Ver inventario Page, * Generar Reportes, * Gestionar Productos, * configuración de perfil, * gestionar productos, * Menupage, * forgot-password.page, * ver alertas, * notificaciones page, * Registrar Venta page, * Register Page. * Reporte.Service |
| Descripción | estas páginas de nuestra app tenian el mismo error que el loginPage |
| Precondiciones | Utilizaremos el mismo codigo pero con diferente rutas de cada page que ha estado relacionada con la base de datos |
| Pasos de Prueba | * Gestionar categorías * Categorías Servicio, * Ver inventario Page, * Generar Reportes, * Gestionar Productos, * configuración de perfil, * gestionar productos, * Menupage, * forgot-password.page, * ver alertas, * notificaciones page, * Registrar Venta page, * Register Page.   Estaremos en sus page pero .spec.ts donde realizaremos pruebas unitarias |
| Datos de entrada | import { TestBed } from '@angular/core/testing';  import { AngularFireModule } from '@angular/fire/compat';  import { AngularFireAuthModule } from '@angular/fire/compat/auth';  import { environment } from 'src/environments/environment';  import { LoginPage } from './login.page';  describe('LoginPage', () => {  beforeEach(async () => {  await TestBed.configureTestingModule({  imports: [  AngularFireModule.initializeApp(environment.firebaseConfig), // Configuración real de Firebase  AngularFireAuthModule, // Módulo de autenticación  ],  declarations: [LoginPage],  }).compileComponents();  });  it('should create', () => {  const fixture = TestBed.createComponent(LoginPage);  const app = fixture.componentInstance;  expect(app).toBeTruthy();  });  }); |
| Resultado esperado | 21 specs habian 17 errores de esos bajo a 3 errores al tener la importación de AngularFire AuthModule |
| Resultado Actual | Tenemos solo 3 errores con la plataforma de jasmine con karma |
| Estado | Aprobado |

| ID | P03 |
| --- | --- |
| Funcion/metodo | Editar Categoria |
| Descripción | El editar Categoria tenia un error en la introduccion de nombre, should create Failed: Cannot read properties of undefined (reading 'nombre') at templateFn (ng:///EditarCategoriaComponent.js:35:56) at executeTemplate (http://localhost:9876/\_karma\_webpack\_/webpack:/node\_modules/@angular/core/fesm2022/core.mjs:12159:9) |
| Precondiciones | El objeto categoria se inicializa antes de llamar a fixture.detectChanges(). Esto asegura que no sea undefined cuando la vista intente acceder a sus propiedades. |
| Pasos de Prueba | Editar.categoria.spect.ts |
| Datos de entrada | import { ComponentFixture, TestBed } from '@angular/core/testing';  import { EditarCategoriaComponent } from './editar-categoria.component';  import { FormsModule } from '@angular/forms';  import { IonicModule, ModalController } from '@ionic/angular';  describe('EditarCategoriaComponent', () => {  let component: EditarCategoriaComponent;  let fixture: ComponentFixture<EditarCategoriaComponent>;  beforeEach(async () => {  await TestBed.configureTestingModule({  declarations: [EditarCategoriaComponent],  imports: [  IonicModule.forRoot(), // Configura Ionic  FormsModule, // Importa FormsModule si se usa [(ngModel)]  ],  providers: [ModalController], // Proveedor del ModalController  }).compileComponents();  fixture = TestBed.createComponent(EditarCategoriaComponent);  component = fixture.componentInstance;  // Inicializa las propiedades necesarias  component.categoria = { nombre: 'Categoría de prueba' }; // Datos simulados  fixture.detectChanges();  });  it('should create', () => {  expect(component).toBeTruthy();  });  }); |
| Resultado esperado | El Error Cambio de Succes en la prueba de jasmine karma |
| Resultado Actual | de 5 errores Cambiaron a 4 |
| Estado | Aprobado |

| ID | P04 AdminMenuPage |
| --- | --- |
| Funcion/metodo |  |
| Descripción | El injector de firebase pero tuvbimos que importa el popovercontroller para poder ejecutar algunas funciones.  NullInjectorError: R3InjectorError(DynamicTestModule)[PopoverController -> PopoverController]: NullInjectorError: No provider for PopoverController! error properties: Object({ ngTempTokenPath: null, ngTokenPath: [ 'PopoverController', 'PopoverController' ] }) |
| Precondiciones | **PopoverController como proveedor:**   * El servicio PopoverController debe incluirse en los providers del entorno de pruebas para que pueda ser inyectado correctamente en el componente. |
| Pasos de Prueba | Estamos en Admin.page.spect.ts |
| Datos de entrada | import { TestBed } from '@angular/core/testing';  import { AdminMenuPage } from './admin-menu.page';  import { IonicModule, PopoverController } from '@ionic/angular';  describe('AdminMenuPage', () => {  let component: AdminMenuPage;  let fixture: any;  beforeEach(async () => {  await TestBed.configureTestingModule({  declarations: [AdminMenuPage],  imports: [IonicModule.forRoot()], // Configuración de Ionic  providers: [PopoverController], // Proveedor del PopoverController  }).compileComponents();  fixture = TestBed.createComponent(AdminMenuPage);  component = fixture.componentInstance;  fixture.detectChanges();  });  it('should create', () => {  expect(component).toBeTruthy();  });  }); |
| Resultado esperado | Cambiaron los resultado del ng test a 3 errores |
| Resultado Actual | Ng test jasmine karma ha informado que el error ha sido resuelto |
| Estado |  |

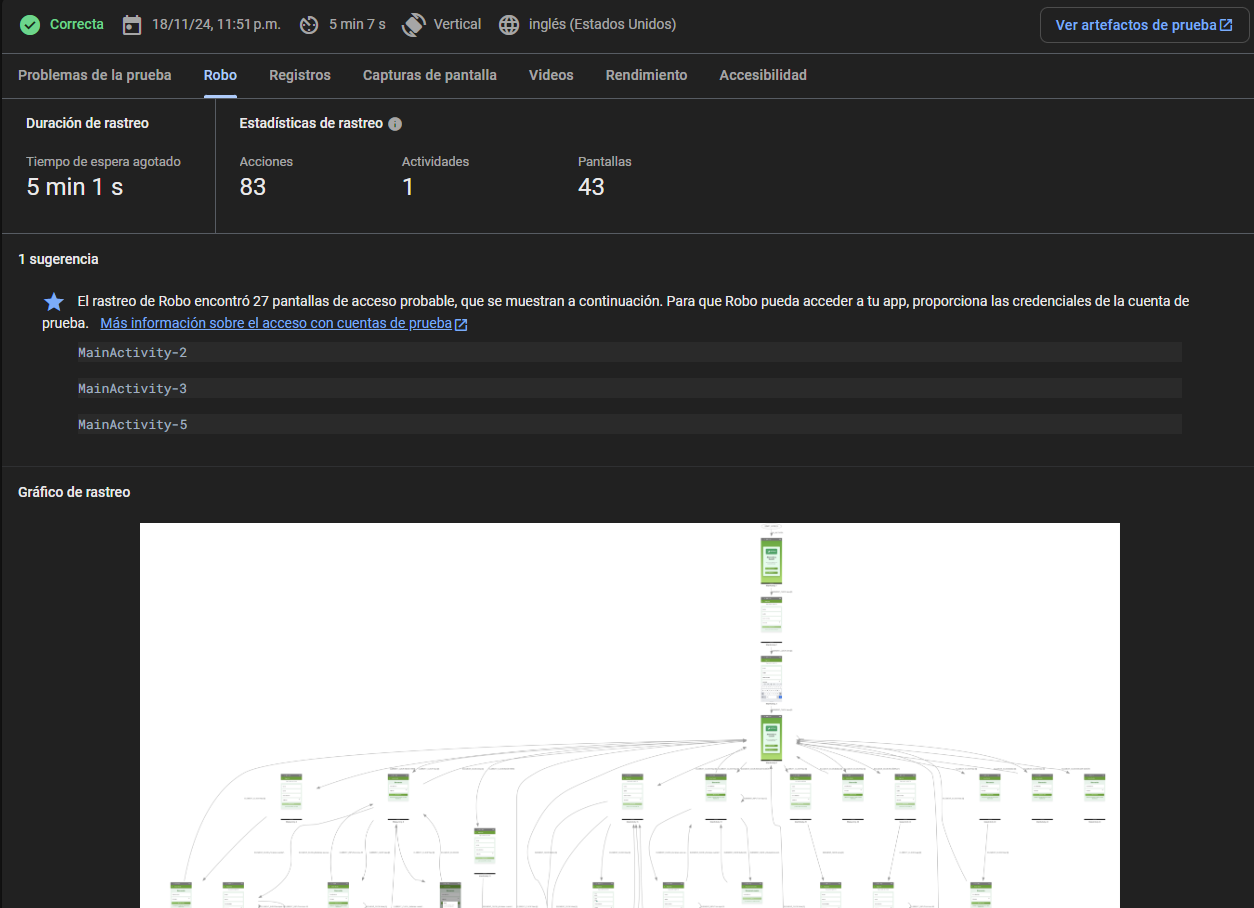
| ID | P05 MenuPage |
| --- | --- |
| Funcion/metodo | MenuPage |
| Descripción | El mismo error que de admin asi que solo importamos el popovercontroller y el Fireatuhmodule  MenuPage > should create NullInjectorError: R3InjectorError(DynamicTestModule)[PopoverController -> PopoverController]: NullInjectorError: No provider for PopoverController! error properties: Object({ ngTempTokenPath: null, ngTokenPath: [ 'PopoverController', 'PopoverController' ] }) at NullInjector.get (http://localhost:9876/\_karma\_webpack\_/webpack:/node\_modules/@angular/core/fesm2022/core.mjs:1663:27) |
| Precondiciones | **Incluir PopoverController en los providers:**   * Esto asegura que Angular pueda inyectar PopoverController en el componente durante las pruebas.   **Opción 1 (Firebase real):**   * Usa AngularFireModule.initializeApp y AngularFireAuthModule. * Proporciona la configuración real de Firebase desde environment.firebaseConfig. |
| Pasos de Prueba | Menu.page.Spect.ts |
| Datos de entrada | import { TestBed } from '@angular/core/testing';  import { IonicModule, PopoverController } from '@ionic/angular';  import { MenuPage } from './menu.page';  describe('MenuPage', () => {  let component: MenuPage;  let fixture: any;  beforeEach(async () => {  await TestBed.configureTestingModule({  declarations: [MenuPage],  imports: [IonicModule.forRoot()], // Configuración de Ionic  providers: [PopoverController], // Incluye el PopoverController  }).compileComponents();  fixture = TestBed.createComponent(MenuPage);  component = fixture.componentInstance;  fixture.detectChanges();  });  it('should create', () => {  expect(component).toBeTruthy();  });  }); |
| Resultado esperado | errores 0 de jasmine con karma |
| Resultado Actual | ya no arroja ningún error en jasmine karma |
| Estado | Aprobado |

**Pruebas de integracion**

### **4. Casos de Prueba**

#### **Caso de Prueba 1: Navegación entre pantallas**

* **ID del Caso:** TC01
* **Descripción:** Validar que la navegación entre MainActivity y SecondActivity funcione correctamente.
* **Precondiciones:**
  1. La aplicación debe estar instalada.
  2. La aplicación debe iniciarse correctamente.
* **Pasos:**
  1. Abrir la aplicación.
  2. Hacer clic en el botón "Ir a siguiente pantalla".
  3. Validar que la nueva pantalla se muestre correctamente.
* **Resultados esperados:**La pantalla SecondActivity se muestra y no hay errores.
* **Estado:** Pendiente / Pasado / Fallado



#### **Caso de Prueba 2: Robo de informacion**

* **ID del Caso:** TC02
* **Descripción:**Entrar a la app sin haber tenido sesion iniciada
* **Precondiciones:**
  1. El servicio API debe estar disponible.
  2. La aplicación debe estar conectada a Internet.
* **Pasos:**
  1. Iniciar sesión.
  2. Colocar datos erróneos
  3. La app no debería dejar entrar a cualquier usuario sin haberse registrado.
* **Resultados esperados: la App solo le permite a los usuarios registrados  
  Estado:** Pasado

#### 

#### 

#### **Caso de Prueba 2: Rendimiento de Sistema**

#### **ID del Caso:** TC02

#### **Descripción:**

Evaluar el consumo de CPU y memoria (RAM) durante la interacción de un usuario en un flujo principal de la aplicación, verificando que los recursos se mantengan dentro de límites aceptables.

#### **Precondiciones:**

1. La aplicación debe estar instalada en un dispositivo con las siguientes especificaciones:
   * **Modelo:** Google Pixel 5
   * **Versión de Android:** 11 (SDK 30)
   * **Configuración Regional:** en\_US
2. La red debe estar estable si la aplicación realiza solicitudes API.
3. Los datos necesarios (como usuarios o configuraciones) deben estar precargados en la aplicación.

#### **Pasos:**

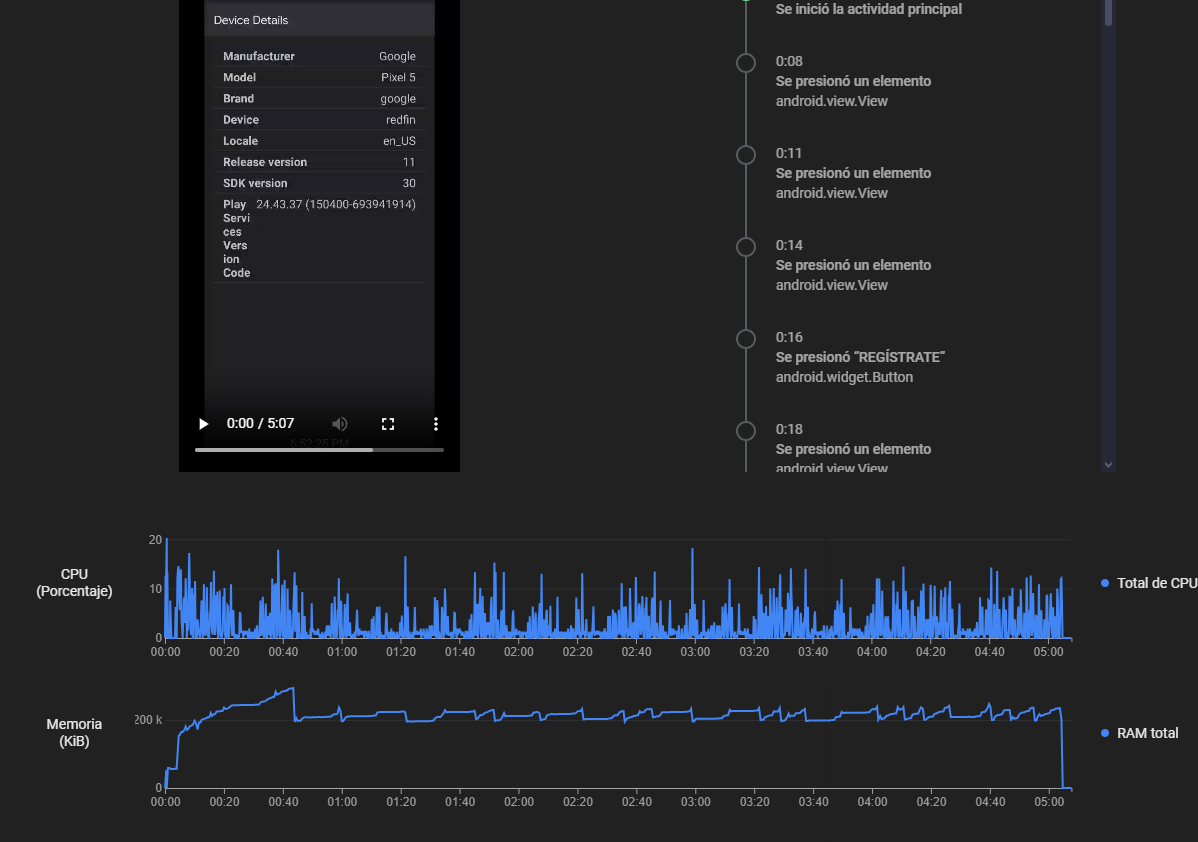
1. Inicia la aplicación y verifica que la actividad principal se cargue correctamente.
2. Realiza las siguientes interacciones:
   * Presiona los elementos identificados como android.view.View en las marcas de tiempo indicadas (0:08, 0:11, 0:14).
   * Presiona el botón etiquetado como "REGÍSTRATE" en la marca de tiempo 0:16.
3. Observa y registra el comportamiento de la aplicación hasta la marca de 5 minutos.
4. Mide y registra los siguientes indicadores:
   * **CPU:** Porcentaje de uso promedio y picos durante la interacción.
   * **RAM:** Memoria utilizada en KiB al inicio, durante y al final de la interacción.
5. Verifica que no haya aumentos inesperados en el uso de memoria (fugas de memoria) o picos altos de CPU.

#### **Resultados esperados:**

1. **CPU:**
   * El uso promedio no debe exceder el **15%** durante la interacción normal.
   * Los picos de uso no deben superar el **20%**.
2. **RAM:**
   * La memoria utilizada debe mantenerse dentro de los **200 KiB**.
   * No debe haber incrementos significativos o sostenidos de memoria que indiquen fugas.
3. **Fluidez del sistema:**
   * Las acciones realizadas deben ser rápidas, con un tiempo de respuesta menor a 1 segundo.
4. **Sin errores críticos:**
   * La aplicación no debe colapsar ni mostrar mensajes de error relacionados con el sistema o los recursos.

#### **Estado del Caso:**

* Pasado



### **Caso de Prueba 3: Validación de Accesibilidad en Elementos Interactivos**

#### **ID del Caso:** TC03

#### **Descripción:**

Evaluar la accesibilidad de la aplicación enfocándose en el tamaño de los objetivos táctiles, el contraste de elementos visuales y las etiquetas de contenido para garantizar que la aplicación sea usable por todos los usuarios, incluyendo aquellos con necesidades especiales.

#### **Precondiciones:**

1. La aplicación debe estar instalada y configurada en un dispositivo con Android 11 (SDK 30).
2. La funcionalidad principal de la aplicación debe estar operativa (por ejemplo, formulario de registro e inicio de sesión).
3. El dispositivo debe estar en modo vertical (portrait).

#### **Pasos:**

##### **A) Tamaño de Objetivos Táctiles**

1. Accede a la pantalla **"Regístrate"**.
2. Verifica que los botones y campos interactivos (como Nombre, Apellido, Correo Electrónico, Contraseña y el botón "Regístrate") tengan un tamaño mínimo de **48dp x 48dp**, según las recomendaciones de accesibilidad de Google.
3. Usa herramientas como **Layout Inspector** en Android Studio o el modo accesibilidad del sistema para inspeccionar las dimensiones.

##### **B) Contraste Bajo**

1. Inspecciona visualmente los textos y elementos importantes, como botones y campos de entrada.
2. Usa una herramienta como **Accessibility Scanner** o **Contrast Checker** para verificar que el contraste entre texto y fondo cumpla con los estándares WCAG AA (razón de contraste mínima: 4.5:1 para texto normal).

##### **C) Etiquetas de Contenido**

1. Revisa los campos del formulario y botones interactivos para asegurarte de que tengan descripciones accesibles (contentDescription).
2. Usa herramientas como **TalkBack** en Android para validar que los lectores de pantalla puedan describir correctamente los elementos.

#### **Resultados esperados:**

##### **A) Tamaño de Objetivos Táctiles**

* Todos los elementos interactivos deben cumplir con el tamaño mínimo recomendado de **48dp x 48dp**.
* No se debe encontrar ningún elemento fuera del estándar.

##### **B) Contraste Bajo**

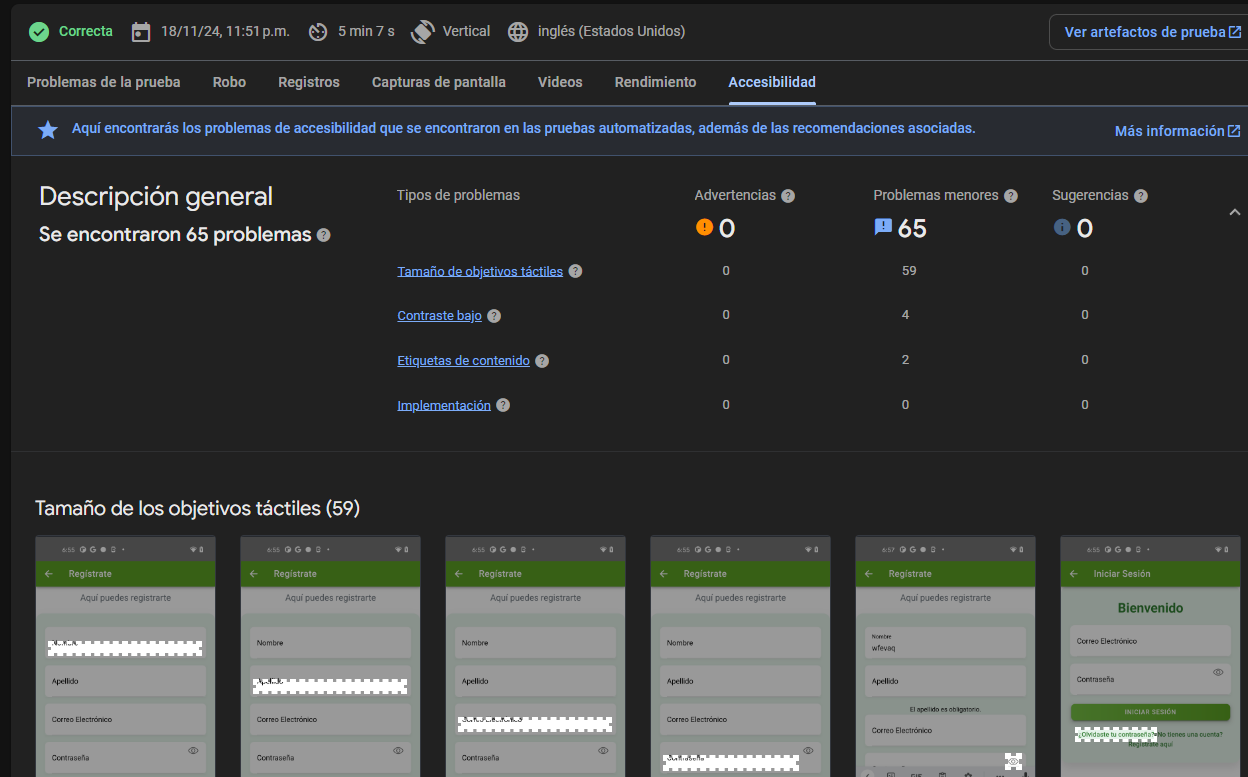
* El contraste entre el texto y el fondo debe cumplir con los estándares WCAG AA.
* El contraste debe ser suficiente incluso en modo oscuro, si la aplicación lo soporta.

##### **C) Etiquetas de Contenido**

* Todos los campos de entrada y botones deben tener etiquetas accesibles y descripciones claras.
* Los lectores de pantalla deben poder identificar correctamente cada elemento interactivo.

#### **Estado del Caso:**

* Pendiente / Pasado



**Cronogramas de Pruebas:**

| Actividad | Fecha de inicio | Fecha de termino | Responsable |
| --- | --- | --- | --- |
| Pruebas unitarias | 28/10/24 | 30/10/24 | Equipo de pruebas |
| Pruebas de integración | 31/10/24 | 04/11/24 | Equipo de pruebas |

### **7. Roles y Responsabilidades**:

* **Equipo de desarrollo:** Realiza pruebas unitarias.
* **Equipo de desarrollo:** Ejecuta pruebas funcionales e integrales.
* **Líder de proyecto:** Supervisa y valida los resultados.

### **8. Herramientas de Pruebas**

Lista las herramientas utilizadas en el proceso, como:

* **Gestión de pruebas:** Fire base Test Lab, Jasmine, Karma
* **Entorno de desarrollo:** Visual Studio Code, Android Studio.

### **Conclusión**

El presente **Plan de Pruebas** establece una estrategia integral para garantizar la calidad del sistema desarrollado, abarcando pruebas unitarias, de integración y de rendimiento. Este enfoque asegura que todas las funcionalidades del sistema operen correctamente, tanto de manera independiente como en conjunto, cumpliendo con los requerimientos definidos.

La implementación de las pruebas identificará posibles errores y áreas de mejora, permitiendo que sean solucionados antes de la puesta en producción. Además, el seguimiento estructurado y la documentación de los resultados contribuirán a mantener altos estándares de calidad durante todo el ciclo de vida del proyecto.