UNIVERSIDAD DE LAS FUERZAS ARMADAS ESPE

Departamento de Ciencias de la Computación

Proyecto Final: Plan de Aseguramiento de la Calidad (SQAP)

Sistema TravelBrain

Informe 2: Diseño de Casos de Prueba y Matrices de Rastreabilidad

ACTUALIZADO CON RESULTADOS DE EJECUCIÓN

(Estado al 21 de enero de 2026)

Asignatura:	Desarrollo De software Seguro
NRC:	27886
Estudiantes:	Cáceres Germán (Scrum Master & Tech Lead)
	Ponce Diego (Development Team)
Docente:	Ing. Angel Cudco, Mgs.
Fecha:	12 de febrero de 2026

Sangolquí, Ecuador
2026

Índice

Índice de cuadros

Resumen Ejecutivo

El presente documento constituye el **Diseño Detallado de Casos de Prueba** para el sistema TravelBrain, complementando el Plan Maestro de Pruebas establecido en el Informe 1.

Este informe proporciona especificaciones técnicas completas de los casos de prueba diseñados para cada nivel (unitario, integración, funcional y seguridad), matrices de rastreabilidad que vinculan requisitos con casos de prueba, y scripts automatizados listos para ejecución.

Estado Actual de Ejecución

Actualización basada en resultados reales obtenidos:

- **Pruebas E2E (Cypress):** 9 casos ejecutados exitosamente
- **Pruebas Unitarias Frontend (Jest + RTL):** 14 componentes y 7 servicios probados
- **!Pruebas Unitarias Backend (Jest):** 10 conjuntos ejecutados con fallos
- **!Pruebas Business Rules:** 5 conjuntos ejecutados con fallos
- **...Pruebas de Seguridad (OWASP ZAP):** Pendientes de ejecución

Métricas Clave Actuales:

- **Cobertura Total:** 15.3 % (SonarQube)
- **Issues Abiertos:** 381 (1 Seguridad, 94 Confiabilidad, 286 Mantenibilidad)
- **Tasa de Éxito Pruebas:** 34.5 % de 87 casos diseñados
- **Pruebas E2E Completadas:** 9/16 casos diseñados

Total de Casos Diseñados:

87 casos de prueba distribuidos en 4 niveles.

Próximos Pasos Críticos:

1. Corregir pruebas unitarias fallidas en backend
2. Mejorar cobertura de código al menos al 70 %
3. Ejecutar pruebas de integración y seguridad
4. Resolver issues de confiabilidad (94 abiertos)

Palabras clave: Casos de Prueba, Matriz de Rastreabilidad, Test Design, Cypress, Jest, Postman, OWASP ZAP, Ejecución Real, SonarQube.

1. Introducción

1.1. Propósito del Documento

Este documento detalla el diseño completo de casos de prueba para el sistema TravelBrain, proporcionando especificaciones técnicas que servirán como guía durante la fase de ejecución del Sprint de Calidad.

1.2. Alcance del Diseño

El diseño cubre los siguientes niveles de prueba:

1. **Pruebas Unitarias:** Componentes individuales (Frontend + Backend)
2. **Pruebas de Integración:** APIs RESTful y comunicación entre servicios
3. **Pruebas Funcionales E2E:** Flujos completos de usuario
4. **Pruebas de Seguridad:** Vulnerabilidades OWASP Top 10

1.3. Referencias

Este documento se basa en:

- **Informe 1:** Plan Maestro de Pruebas (SQAP)
- **ARCHITECTURE.md:** Documentación técnica del sistema
- **IEEE 829:** Estándar para documentación de pruebas **ieee829**
- **ISO/IEC/IEEE 29119:** Estándar de pruebas de software **iso29119**
- **Resultados reales de ejecución:** Evidencias de Cypress, Jest y SonarQube

1.4. Estado Actual de Ejecución

Basado en las evidencias obtenidas, se presenta el siguiente estado de ejecución:

Cuadro 1: Resumen del Estado de Ejecución

Tipo de Prueba	**Diseñados**	**Ejecutados**	**Estado**
Unitarias (Frontend)	15	21+	Completado
Unitarias (Backend)	18	10+	!Con fallos
Pruebas E2E	16	9	Completado
Integración	28	0	...Pendiente
Seguridad	10	0	...Pendiente
TOTAL	**87**	**40+**	**34.5 % completado**

2. Requisitos Funcionales del Sistema

2.1. Módulo de Autenticación

Cuadro 2: Requisitos Funcionales - Autenticación

ID	Requisito	Prioridad
RF-AUTH-01	El sistema debe permitir registro de usuarios con email, username y password	Alta
RF-AUTH-02	El sistema debe validar formato de email (RFC 5322)	Alta
RF-AUTH-03	El sistema debe requerir contraseñas con mínimo 8 caracteres	Alta
RF-AUTH-04	El sistema debe hashear contraseñas con bcrypt antes de almacenar	Alta
RF-AUTH-05	El sistema debe permitir login con email y password	Alta
RF-AUTH-06	El sistema debe generar token JWT con expiración de 24h tras login exitoso	Alta
RF-AUTH-07	El sistema debe rechazar credenciales inválidas con mensaje apropiado	Alta
RF-AUTH-08	El sistema debe permitir logout y invalidar token JWT	Media

2.2. Módulo de Autenticación Biométrica

Cuadro 3: Requisitos Funcionales - Biometría

ID	Requisito	Prioridad
RF-BIO-01	El sistema debe permitir registro de datos biométricos faciales	Alta
RF-BIO-02	El sistema debe generar challenge token temporal (TTL 2 min) para verificación	Alta
RF-BIO-03	El sistema debe extraer encoding 128D mediante microservicio facial	Alta
RF-BIO-04	El sistema debe cifrar encodings con AES-256-CBC antes de almacenar	Alta
RF-BIO-05	El sistema debe validar liveness score ¿= 0.6	Alta
RF-BIO-06	El sistema debe validar quality score ¿= 0.6	Alta
RF-BIO-07	El sistema debe comparar encodings con threshold 0.6 para autenticación	Alta
RF-BIO-08	El sistema debe registrar auditoría de intentos biométricos	Media

2.3. Módulo de Gestión de Viajes

Cuadro 4: Requisitos Funcionales - Viajes

ID	Requisito	Prioridad
RF-TRIP-01	El sistema debe permitir crear viajes con título, destino, fechas y presupuesto	Alta
RF-TRIP-02	El sistema debe validar que startDate ¡endDate	Alta
RF-TRIP-03	El sistema debe permitir listar viajes del usuario autenticado	Alta
RF-TRIP-04	El sistema debe permitir actualizar datos de un viaje propio	Media
RF-TRIP-05	El sistema debe permitir eliminar un viaje propio	Media
RF-TRIP-06	El sistema debe impedir acceso a viajes de otros usuarios (IDOR protection)	Alta

2.4. Módulo de Clima

Cuadro 5: Requisitos Funcionales - Clima

ID	Requisito	Prioridad
RF-WEAT-01	El sistema debe buscar clima actual mediante OpenWeather API	Media
RF-WEAT-02	El sistema debe almacenar historial de búsquedas de clima	Media
RF-WEAT-03	El sistema debe mostrar temperatura, condiciones e ícono	Media
RF-WEAT-04	El sistema debe permitir listar historial de búsquedas	Baja

2.5. Módulo de Administración

Cuadro 6: Requisitos Funcionales - Admin

ID	Requisito	Prioridad
RF-ADM-01	El sistema debe permitir a ADMIN listar todos los usuarios	Alta
RF-ADM-02	El sistema debe permitir a ADMIN cambiar roles de usuarios	Alta
RF-ADM-03	El sistema debe impedir a usuarios regulares acceder a rutas admin	Alta

3. Matriz de Rastreabilidad: Requisitos vs Casos de Prueba

Cuadro 7: Matriz de Rastreabilidad - Requisitos vs Casos de Prueba

ID Req.	ID Caso	Nombre del Caso de Prueba	Nivel	Herramienta	Estado Ejecución
RF-AUTH-01	TC-AUTH-001	Registro exitoso con datos válidos	E2E	Cypress	...
RF-AUTH-01	TC-AUTH-002	Registro falla con email duplicado	E2E	Cypress	...
RF-AUTH-02	TC-AUTH-003	Registro rechaza email inválido	Integración	Postman	...
RF-AUTH-03	TC-AUTH-004	Registro rechaza password corto	Integración	Postman	...
RF-AUTH-04	TC-AUTH-U01	bcrypt hashea contraseñas correctamente	Unitaria	Jest	!
RF-AUTH-05	TC-AUTH-005	Login exitoso con credenciales válidas	E2E	Cypress	
RF-AUTH-06	TC-AUTH-U02	JWT generado contiene userId y expira en 24h	Unitaria	Jest	!
RF-AUTH-07	TC-AUTH-006	Login fallido con credenciales inválidas	E2E	Cypress	
RF-AUTH-08	TC-AUTH-007	Logout exitoso elimina token del cliente	E2E	Cypress	...
RF-BIO-01	TC-BIO-001	Registro biométrico exitoso con imagen válida	E2E	Cypress	...
RF-BIO-02	TC-BIO-002	Challenge token generado expira en 2 minutos	Integración	Postman	...
RF-BIO-03	TC-BIO-003	Microservicio extrae encoding 128D correctamente	Integración	Postman	...
RF-BIO-04	TC-BIO-U01	Encoding cifrado con AES-256 puede descifrarse	Unitaria	Jest	!
RF-BIO-05	TC-BIO-004	Verificación rechaza imagen con liveness bajo	Integración	Postman	...

Continúa en la siguiente página...

Cuadro 7 – Continuación

ID Req.	ID Caso	Nombre del Caso de Prueba	Nivel	Herramienta	Estado Ejecución
RF-BIO-06	TC-BIO-005	Verificación rechaza imagen de baja calidad	Integración	Postman	...
RF-BIO-07	TC-BIO-006	Login biométrico exitoso con rostro registrado	E2E	Cypress	...
RF-BIO-08	TC-BIO-U02	Auditoría registra timestamp e IP del intento	Unitaria	Jest	!
RF-TRIP-01	TC-TRIP-001	Crear viaje exitosamente con datos completos	E2E	Cypress	
RF-TRIP-02	TC-TRIP-002	Creación rechaza startDate posterior a endDate	Integración	Postman	...
RF-TRIP-03	TC-TRIP-003	Listar viajes retorna solo viajes propios	Integración	Postman	...
RF-TRIP-04	TC-TRIP-004	Actualizar viaje propio exitosamente	E2E	Cypress	...
RF-TRIP-05	TC-TRIP-005	Eliminar viaje propio exitosamente	E2E	Cypress	...
RF-TRIP-06	TC-TRIP-006	Acceso a viaje ajeno retorna 403 Forbidden	Integración	Postman	...
RF-TRIP-06	TC-SEC-001	Prevención de IDOR en endpoint /trips/:id	Seguridad	OWASP ZAP	...
RF-WEAT-01	TC-WEAT-001	Búsqueda de clima retorna datos válidos	Integración	Postman	...
RF-WEAT-02	TC-WEAT-002	Búsqueda almacena registro en BD	Unitaria	Jest	!
RF-WEAT-03	TC-WEAT-003	Respuesta incluye temp, description e icon	Integración	Postman	...

Continúa en la siguiente página...

Cuadro 7 – Continuación

ID Req.	ID Caso	Nombre del Caso de Prueba	Nivel	Herramienta	Estado Ejecución
RF-WEAT-04	TC-WEAT-004	Listar historial retorna búsquedas ordenadas	E2E	Cypress	...
RF-ADM-01	TC-ADM-001	Admin lista todos los usuarios	Integración	Postman	...
RF-ADM-02	TC-ADM-002	Admin cambia rol de usuario exitosamente	Integración	Postman	...
RF-ADM-03	TC-ADM-003	Usuario regular no accede a /api/users	Integración	Postman	...
RF-ADM-03	TC-SEC-002	Protección de rutas admin sin bypass	Seguridad	OWASP ZAP	...
-	TC-SEC-003	Detección de vulnerabilidades XSS en formularios	Seguridad	OWASP ZAP	...
-	TC-SEC-004	Detección de SQL Injection en parámetros	Seguridad	OWASP ZAP	...
-	TC-SEC-005	Headers de seguridad presentes (CSP, HSTS)	Seguridad	OWASP ZAP	...
-	TC-SEC-006	Sin exposición de información sensible en errores	Seguridad	OWASP ZAP	...
-	TC-SEC-007	JWT no puede ser manipulado sin invalidar firma	Seguridad	Manual	...
-	TC-LAND-001	Carga la página principal	E2E	Cypress	
-	TC-NAV-001	Navegar al dashboard después del login	E2E	Cypress	
-	TC-NAV-002	Acceder a la página de destinos	E2E	Cypress	
-	TC-NAV-003	Acceder a la página del clima	E2E	Cypress	
-	TC-NAV-004	Acceder a la página de perfil	E2E	Cypress	
-	TC-SEC-002A	Verificar protección de rutas sin autenticación	E2E	Cypress	

4. Diseño de Casos de Prueba Unitarias

4.1. Casos de Prueba Unitaria - Backend (Jest)

4.1.1. TC-AUTH-U01: Hashing de Contraseñas con bcrypt

Cuadro 8: TC-AUTH-U01: Hashing de Contraseñas

ID	TC-AUTH-U01
Nombre	Verificar hashing correcto de contraseñas con bcrypt
Módulo	authController.js - register()
Prioridad	Alta
Precondiciones	Módulo bcrypt importado. Función register() disponible.
Datos de Entrada	password: "Test123!"
Pasos	1. Mock de User.findOne() retornando null 2. Mock de bcrypt.hash() retornando hash predeterminado 3. Llamar a register() con datos de prueba 4. Verificar que bcrypt.hash fue llamado con password
Resultado Esperado	1. bcrypt.hash() llamado exactamente 1 vez 2. Password no almacenado en texto plano 3. Hash almacenado en BD es string de 60 caracteres
Resultado Obtenido	**!No ejecutado exitosamente** - Suite de pruebas fallida
Estado	Fallido - Requiere corrección

Script de Prueba:

```
const { register } = require("../src/controllers/authController");
const User = require("../src/models/User");
const bcrypt = require("bcrypt");

jest.mock("../src/models/User");
jest.mock("bcrypt");

describe("TC-AUTH-U01: Password Hashing with bcrypt", () => {
let req, res;

beforeEach(() => {
    req = {
    body: {
        email: "test@mail.com",
        password: "Test123!",
        username: "testuser"
    }
    };
    res = {
    status: jest.fn().mockReturnThis(),
    json: jest.fn()
    };
});

afterEach(() => {
    jest.clearAllMocks();
```

```
});

it("should hash password with bcrypt before storing", async () => {
    // Arrange
    const hashedPassword = "$2b$10$abcdefghijklmnopqrstuvwxyz123456";
    User.findOne.mockResolvedValue(null);
    bcrypt.hash.mockResolvedValue(hashedPassword);
    User.prototype.save = jest.fn().mockResolvedValue({
    _id: "123",
    email: "test@mail.com",
    password: hashedPassword
    });

    // Act
    await register(req, res);

    // Assert
    expect(bcrypt.hash).toHaveBeenCalledTimes(1);
    expect(bcrypt.hash).toHaveBeenCalledWith("Test123!", 10);
    expect(User.prototype.save).toHaveBeenCalled();
    const savedUser = await User.prototype.save.mock.results[0].value;
    expect(savedUser.password).toBe(hashedPassword);
    expect(savedUser.password).toHaveLength(60);
});

it("should not store password in plain text", async () => {
    User.findOne.mockResolvedValue(null);
    bcrypt.hash.mockResolvedValue("hashedPassword");
    User.prototype.save = jest.fn().mockResolvedValue({
    password: "hashedPassword"
    });

    await register(req, res);

    const savedUser = await User.prototype.save.mock.results[0].value;
    expect(savedUser.password).not.toBe("Test123!");
});
});
```

Listing 1: TC-AUTH-U01: Jest Script

4.1.2. TC-AUTH-U02: Generación y Expiración de JWT

Cuadro 9: TC-AUTH-U02: Generación de JWT

ID	TC-AUTH-U02
Nombre	Verificar generación correcta de JWT con userId y expiración 24h
Módulo	authController.js - login()
Prioridad	Alta
Precondiciones	Módulo jsonwebtoken importado. JWT_SECRET configurado.
Datos de Entrada	userId: "507f1f77bcf86cd799439011"
Pasos	1. Mock de User.findOne() retornando usuario válido 2. Mock de bcrypt.compare() retornando true 3. Llamar a login() con credenciales válidas 4. Decodificar JWT retornado 5. Verificar payload y expiración
Resultado Esperado	1. JWT contiene userId en payload 2. exp (expiration) = iat + 24 horas (86400 seg) 3. JWT firma válida con JWT_SECRET
Resultado Obtenido	!**No ejecutado exitosamente** - Parte de suite fallida
Estado	Fallido - Requiere corrección

Script de Prueba:

```
const { login } = require("../src/controllers/authController");
const User = require("../src/models/User");
const bcrypt = require("bcrypt");
const jwt = require("jsonwebtoken");

jest.mock("../src/models/User");
jest.mock("bcrypt");

describe("TC-AUTH-U02: JWT Generation and Expiration", () => {
const JWT_SECRET = "test-secret-key";
process.env.JWT_SECRET = JWT_SECRET;

let req, res;

beforeEach(() => {
    req = {
    body: {
        email: "test@mail.com",
        password: "Test123!"
    }
    };
    res = {
    status: jest.fn().mockReturnThis(),
    json: jest.fn()
    };
});

it("should generate JWT with userId and 24h expiration", async () => {
    // Arrange
```

```
    const mockUser = {
    _id: "507f1f77bcf86cd799439011",
    email: "test@mail.com",
    password: "hashedPassword",
    username: "testuser"
    };
    User.findOne.mockResolvedValue(mockUser);
    bcrypt.compare.mockResolvedValue(true);

    // Act
    await login(req, res);

    // Assert
    expect(res.status).toHaveBeenCalledWith(200);
    const response = res.json.mock.calls[0][0];
    const token = response.data.token;

    // Decode JWT
    const decoded = jwt.verify(token, JWT_SECRET);

    expect(decoded.userId).toBe(mockUser._id);

    // Verify 24h expiration (86400 seconds)
    const expirationTime = decoded.exp - decoded.iat;
    expect(expirationTime).toBe(86400);
});

it("should sign JWT with JWT_SECRET", async () => {
    const mockUser = {
    _id: "123",
    email: "test@mail.com",
    password: "hashedPassword"
    };
    User.findOne.mockResolvedValue(mockUser);
    bcrypt.compare.mockResolvedValue(true);

    await login(req, res);

    const response = res.json.mock.calls[0][0];
    const token = response.data.token;

    // Should not throw error
    expect(() => {
    jwt.verify(token, JWT_SECRET);
    }).not.toThrow();

    // Should throw with wrong secret
    expect(() => {
    jwt.verify(token, "wrong-secret");
    }).toThrow();
});
});
```

Listing 2: TC-AUTH-U02: Jest Script

4.2. Casos de Prueba Unitaria - Frontend (Jest + RTL)

4.2.1. TC-FE-U01: Componente BiometricLogin Renderiza Correctamente

Cuadro 10: TC-FE-U01: Renderizado de BiometricLogin

ID	TC-FE-U01
Nombre	Verificar renderizado correcto de BiometricLogin component
Componente	BiometricLoginAdvanced.jsx
Prioridad	Media
Precondiciones	React Testing Library configurado. face-api.js mockeado.
Pasos	1. Renderizar componente BiometricLoginAdvanced 2. Buscar elementos clave del DOM 3. Verificar visibilidad de botones
Resultado Esperado	1. Input de email visible 2. Botón "Start Face Detection"visible 3. Canvas de video presente en DOM
Resultado Obtenido	**Ejecutado exitosamente** - Parte de pruebas frontend completadas
Estado	Completado

Script de Prueba:

```
import { render, screen } from "@testing-library/react";
import { BrowserRouter } from "react-router-dom";
import BiometricLoginAdvanced from "../src/components/
   BiometricLoginAdvanced";

// Mock face-api.js
jest.mock("face-api.js", () => ({
nets: {
    tinyFaceDetector: { loadFromUri: jest.fn() },
    faceLandmark68Net: { loadFromUri: jest.fn() },
    faceExpressionNet: { loadFromUri: jest.fn() }
},
detectSingleFace: jest.fn(),
TinyFaceDetectorOptions: jest.fn()
}));

describe("TC-FE-U01: BiometricLogin Component Rendering", () => {
it("should render email input field", () => {
    render(
    <BrowserRouter>
        <BiometricLoginAdvanced />
    </BrowserRouter>
    );

    const emailInput = screen.getByLabelText(/email/i);
    expect(emailInput).toBeInTheDocument();
    expect(emailInput).toHaveAttribute("type", "email");
});

it("should render start detection button", () => {
    render(
```

```
    <BrowserRouter>
        <BiometricLoginAdvanced />
    </BrowserRouter>
    );

    const startButton = screen.getByText(/start face detection/i);
    expect(startButton).toBeInTheDocument();
    expect(startButton).toBeEnabled();
});

it("should render video canvas element", () => {
    const { container } = render(
    <BrowserRouter>
        <BiometricLoginAdvanced />
    </BrowserRouter>
    );

    const canvas = container.querySelector("canvas");
    expect(canvas).toBeInTheDocument();
});
});
```

Listing 3: TC-FE-U01: RTL Script

5. Diseño de Casos de Prueba de Integración

5.1. Casos de Prueba de Integración - APIs (Postman)

5.1.1. TC-AUTH-003: Validación de Email Inválido

Cuadro 11: TC-AUTH-003: Email Inválido

ID	TC-AUTH-003
Nombre	Registro rechaza email con formato inválido
Endpoint	POST /api/auth/register
Prioridad	Alta
Precondiciones	Backend desplegado en http://localhost:4000. Base de datos accesible.
Headers	Content-Type: application/json

Body (JSON)	{
	.emailï”: ïnvalid-email-format”,
	”password”: ”Test123!”,
	üsername”: ”testuser”
	}

Resultado Esperado	1. Status Code: 400 Bad Request 2. Response: {”success”: false, ”message”: Ïnvalid email format”} 3. No se crea usuario en BD

Test Script (Postman)	pm.test(”Status is 400”, () =¿{
	pm.response.to.have.status(400);
	});
	
	pm.test(.Error message present”, () =¿{
	const json = pm.response.json();
	pm.expect(json.success).to.be.false;
	pm.expect(json.message).to.include(.emailï”);
	});

Resultado Obtenido	**...Pendiente de ejecución**
Estado	No Ejecutado

5.1.2. TC-BIO-003: Extracción de Encoding 128D

Cuadro 12: TC-BIO-003: Extracción de Encoding

ID	TC-BIO-003
Nombre	Microservicio extrae encoding 128D correctamente de imagen facial
Endpoint	POST http://localhost:8001/extract-features (Interno)
Prioridad	Alta
Precondiciones	Facial service desplegado. Backend tiene acceso a red interna. INTERNAL_SERVICE_TOKEN configurado.
Headers	X-Internal-Token: {{internal_service_token}}
Body (multipart)	face: [archivo face.jpg válido]
Resultado Esperado	1. Status Code: 200 OK 2. Response contiene encoding: array de 128 floats 3. liveness score: 0.0-1.0 4. quality score: 0.0-1.0 5. confidence: 0.0-1.0

Test Script	pm.test(”Status is 200”, () =¿{
	pm.response.to.have.status(200);
	});
	
	pm.test(.Encoding is 128D array”, () =¿{
	const json = pm.response.json();
	pm.expect(json.encoding).to.be.an(.array”);
	pm.expect(json.encoding).to.have.lengthOf(128);
	});
	
	pm.test(”Liveness score present”, () =¿{
	const json = pm.response.json();
	pm.expect(json.liveness_score).to.be.a(”number”);
	pm.expect(json.liveness_score).to.be.within(0, 1);
	});

Resultado Obtenido	**...Pendiente de ejecución**
Estado	No Ejecutado

5.1.3. TC-TRIP-006: Protección IDOR en Viajes

Cuadro 13: TC-TRIP-006: Protección IDOR

ID	TC-TRIP-006
Nombre	Usuario no puede acceder a viajes de otros usuarios (IDOR prevention)
Endpoint	GET /trips/:tripId
Prioridad	Alta
Precondiciones	Usuario A autenticado (JWT token en variable). Usuario B tiene viaje con _id conocido.
Headers	Authorization: Bearer {{jwt_token_userA}}
URL	GET http://localhost:4000/trips/{{trip_id_userB}}
Resultado Esperado	1. Status Code: 403 Forbidden 2. Response: {"success": false, "message": .Access denied"} 3. No se retornan datos del viaje ajeno

Test Script	pm.test("Status is 403", () =¿{
	pm.response.to.have.status(403);
	});
	
	pm.test(.Access denied message", () =¿{
	const json = pm.response.json();
	pm.expect(json.success).to.be.false;
	pm.expect(json.message).to.include("denied");
	});

Resultado Obtenido	**...Pendiente de ejecución**
Estado	No Ejecutado

6. Diseño de Casos de Prueba End-to-End

6.1. Casos de Prueba E2E - Frontend (Cypress)

6.1.1. TC-AUTH-005: Login Tradicional Exitoso

Cuadro 14: TC-AUTH-005: Login Exitoso

ID	TC-AUTH-005
Nombre	Login tradicional exitoso con credenciales válidas
Flujo	Autenticación
Prioridad	Alta
Precondiciones	Usuario registrado: test@test.com / admin1234. Frontend accesible en http://localhost:3001. Backend funcional.
Pasos	1. Navegar a http://travelbrain.ddns.net/login 2. Ingresar email: test@test.com 3. Ingresar password: admin1234 4. Hacer clic en botón "Sign in" 5. Esperar redirección
Resultado Esperado	1. Redirección a /dashboard 2. Mensaje de bienvenida visible 3. Token JWT almacenado en localStorage 4. Navbar muestra nombre del usuario
Resultado Obtenido	✓**Ejecutado exitosamente**
Estado	Completado

Script Cypress Ejecutado:

```
describe('Login de usuario', () => {
  it('Login exitoso con credenciales válidas', () => {
    cy.visit('https://travelbrain.ddns.net/login')

    // Ingresar credenciales válidas
    cy.get('input[name="email"]').type('test@test.com')
    cy.get('input[type="password"]').type('admin1234')

    // Hacer clic en el botón de login
    cy.get('button[type="submit"]').click()

    // Verificar redirección al dashboard
    cy.url().should('include', '/dashboard')

    // Verificar elementos del dashboard
    cy.contains('Welcome back').should('be.visible')
    cy.contains('Total Trips').should('be.visible')
  })
})
```

Listing 4: TC-AUTH-005: Cypress Script Ejecutado

6.1.2. TC-AUTH-006: Login Fallido con Credenciales Inválidas

Cuadro 15: TC-AUTH-006: Login Fallido

ID	TC-AUTH-006
Nombre	Login fallido con credenciales inválidas
Flujo	Autenticación
Prioridad	Alta
Precondiciones	Frontend accesible. Usuario no registrado.
Pasos	1. Navegar a https://travelbrain.ddns.net/login 2. Ingresar email: ihopc@gmail.com 3. Ingresar password: incorrecta123 4. Hacer clic en botón "Sign in" 5. Esperar respuesta
Resultado Esperado	1. Permanece en página /login 2. Mensaje de error visible 3. No se redirige al dashboard 4. Token no almacenado
Resultado Obtenido	✓**Ejecutado exitosamente** - Mensaje "Login failed. Please try again."
Estado	Completado

6.1.3. TC-TRIP-001: Crear Viaje Exitosamente

Cuadro 16: TC-TRIP-001: Crear Viaje

ID	TC-TRIP-001
Nombre	Crear viaje exitosamente con datos completos
Flujo	Gestión de Viajes
Prioridad	Alta
Precondiciones	Usuario autenticado. En página /trips.
Datos de Prueba	Título: "Viaje E2E Cypress". Destino: "Paris, France". Fecha inicio: 2026-06-01. Fecha fin: 2026-06-15. Presupuesto: 5000.
Pasos	1. Login con credenciales válidas 2. Navegar a /trips 3. Hacer clic en "New Trip" 4. Completar formulario con datos de prueba 5. Hacer clic en "Save" 6. Esperar respuesta
Resultado Esperado	1. Mensaje de éxito visible 2. Viaje aparece en lista 3. Datos coinciden con ingresados
Resultado Obtenido	✓**Ejecutado exitosamente**
Estado	Completado

Script Cypress Ejecutado:

```
describe('Crear un viaje', () => {
  it('Crear viaje exitosamente', () => {
    // Login primero
    cy.visit('https://travelbrain.ddns.net/login')
    cy.get('input[name="email"]').type('test@test.com')
    cy.get('input[type="password"]').type('admin1234')
    cy.get('button[type="submit"]').click()

    // Navegar a trips
    cy.visit('https://travelbrain.ddns.net/trips')

    // Abrir modal de crear viaje
    cy.contains('New Trip').click()

    // Completar formulario
    cy.get('input[name="title"]').type('Viaje E2E Cypress')
    cy.get('input[name="destination"]').type('Paris, France')
    cy.get('input[name="startDate"]').type('2026-06-01')
    cy.get('input[name="endDate"]').type('2026-06-15')
    cy.get('input[name="budget"]').type('5000')

    // Enviar formulario
    cy.get('button[type="submit"]').click()

    // Verificar éxito
    cy.contains('Trip created successfully').should('be.visible')
    cy.contains('Viaje E2E Cypress').should('be.visible')
  })
})
```

Listing 5: TC-TRIP-001: Cypress Script Ejecutado

6.1.4. TC-LAND-001: Carga de Página Principal

Cuadro 17: TC-LAND-001: Carga Página Principal

ID	TC-LAND-001
Nombre	Carga exitosa de la página principal
Flujo	Landing
Prioridad	Alta
Precondiciones	Servidor web funcionando.
Pasos	1. Navegar a https://travelbrain.ddns.net/ 2. Esperar carga completa
Resultado Esperado	1. Página carga sin errores 2. Texto "TRAVELBRAIN"visible 3. Botones de login/registro presentes
Resultado Obtenido	✓**Ejecutado exitosamente**
Estado	Completado

7. Diseño de Casos de Prueba de Seguridad

7.1. Casos de Prueba de Seguridad - OWASP ZAP

7.1.1. TC-SEC-001: Protección IDOR

Cuadro 18: TC-SEC-001: IDOR Prevention

ID	TC-SEC-001
Nombre	Prevención de IDOR (Insecure Direct Object Reference) en /trips/:id
Vulnerabilidad	OWASP A01:2021 - Broken Access Control
Prioridad	Alta
Método	Escaneo Activo + Manual
Precondiciones	OWASP ZAP configurado como proxy. 2 usuarios con sesiones activas (UserA, UserB). UserB tiene viaje con _id conocido.
Pasos	1. Autenticar como UserA en ZAP 2. Interceptar GET /trips/:id con token de UserA 3. Modificar :id por ID de viaje de UserB 4. Enviar request 5. Analizar respuesta
Resultado Esperado	1. HTTP 403 Forbidden 2. No se retornan datos del viaje ajeno 3. No hay information leakage
Criterio de Éxito	Sin vulnerabilidad IDOR detectada
Resultado Obtenido	**...Pendiente de ejecución**
Estado	No Ejecutado

7.1.2. TC-SEC-003: Cross-Site Scripting (XSS)

Cuadro 19: TC-SEC-003: XSS Detection

ID	TC-SEC-003
Nombre	Detección de vulnerabilidades XSS en formularios de entrada
Vulnerabilidad	OWASP A03:2021 - Injection
Prioridad	Alta
Método	Escaneo Activo (ZAP Spider + Active Scan)
Targets	/register (username, email). /trips (title, description, destination). /weather (city search).

Payloads	`<script>alert("XSS")</script>`
	`<img src=x onerror=alert("XSS")>`
	»`<script>alert(String.fromCharCode(88,83,83))</script>`

Pasos	1. Configurar contexto en ZAP 2. Ejecutar Spider en http://localhost:3001 3. Ejecutar Active Scan con XSS rules 4. Revisar alertas generadas 5. Verificar manualmente payloads sospechosos
Resultado Esperado	1. 0 alertas de XSS de severidad Alta/Crítica 2. Inputs sanitizados correctamente 3. React escapa HTML automáticamente
Criterio de Éxito	Sin XSS ejecutable detectado
Resultado Obtenido	**...Pendiente de ejecución**
Estado	No Ejecutado

7.1.3. TC-SEC-005: Headers de Seguridad

Cuadro 20: TC-SEC-005: Security Headers

ID	TC-SEC-005
Nombre	Verificación de headers de seguridad HTTP
Categoría	Security Misconfiguration
Prioridad	Media
Método	Passive Scan (ZAP)
Targets	Todas las respuestas HTTP del backend
Headers Requeridos	Strict-Transport-Security. X-Content-Type-Options: nosniff. X-Frame-Options: DENY. Content-Security-Policy. X-XSS-Protection: 1; mode=block.
Pasos	1. Navegar aplicación con ZAP proxy activo 2. ZAP registra headers automáticamente 3. Revisar alertas de "Missing Security Headers" 4. Verificar cada header manualmente
Resultado Esperado	1. Todos los headers requeridos presentes 2. HSTS con maxAge ¿= 31536000 3. CSP configurado correctamente
Criterio de Éxito	Sin alertas de headers faltantes
Resultado Obtenido	**...Pendiente de ejecución**
Estado	No Ejecutado

8. Resultados de Ejecución y Análisis

8.1. Estado Actual de Ejecución

Cuadro 21: Resumen de Ejecución por Nivel

Nivel de Prueba	Diseñados	Ejecutados	Exitosa	Estado
Unitarias (Backend)	18	10+	0/10	!Requiere corrección
Unitarias (Frontend)	15	21+	21/21	✓Completado
Integración (APIs)	28	0	0/28	...Pendiente
E2E (Cypress)	16	9	9/9	✓Completado
Seguridad (ZAP)	10	0	0/10	...Pendiente
TOTAL	**87**	**40+**	**30/87**	**34.5 % completado**

8.2. Resultados de SonarQube

Cuadro 22: Análisis de Calidad - SonarQube

Métrica	Valor Actual	Calificación	Estado
Cobertura de Código	15.3 %	D (Deficiente)	!
Issues de Seguridad	1	A (Aceptable)	!
Issues de Confiabilidad	94	C (Crítico)	X
Issues de Mantenibilidad	286	A (Aceptable)	!
Duplicación de Código	2.6 %	C (Crítico)	!
Líneas de Código	19,001	-	-
Líneas por Cubrir	3,700	-	!
QUALITY GATE	**PASSED (con advertencias)**	-	!

8.3. Análisis de Cobertura por Módulo

Cuadro 23: Cobertura por Módulo Backend

Módulo	Cobertura	Líneas Cubiertas	Estado
Total Backend	15.3 %	3.7k/23k	X
backend-project/src	17.2 %	3,064 líneas	!
business-rules-backend/src	7.5 %	1,505 líneas	X
frontend-react/src	17.1 %	14,432 líneas	!
Controllers	22.64 %	-	!
Services	1.12 %	-	X
Middlewares	24.29 %	-	!
Models	72 %	-	✓
Routes	43.62 %	-	!
Utils	13.46 %	-	X

8.4. Pruebas Unitarias Ejecutadas

8.4.1. Frontend React - Pruebas Completadas

Cuadro 24: Componentes Frontend Probados

Componente/Servicio	Tests	Estado
AuthSuccess	Múltiples	✓
Admin	Múltiples	✓
App	Múltiples	✓
CurrencySelector	Múltiples	✓
Dashboard	Múltiples	✓
GoogleLoginButton	Múltiples	✓
Itineraries	Múltiples	✓
Landing	Múltiples	✓
Login	Múltiples	✓
Navbar	Múltiples	✓
Profile	Múltiples	✓
Register	Múltiples	✓
Trips	Múltiples	✓
Weather	Múltiples	✓
api (servicio)	Múltiples	✓
config	Múltiples	✓
currencyService	Múltiples	✓
itineraryService	Múltiples	✓
tripService	Múltiples	✓
useAuth (hook)	Múltiples	✓
weatherService	Múltiples	✓

8.4.2. Backend - Pruebas Ejecutadas con Fallos

Cuadro 25: Pruebas Backend Ejecutadas

Archivo de Prueba	Resultado	Detalles
auth.test.js	X	Suite fallida
performance.test.js	X	Suite fallida
security.test.js	X	Suite fallida
trips.test.js	X	Suite fallida
trips.coverage.test.js	X	Suite fallida
users.test.js	X	Suite fallida
destinations.test.js	X	Suite fallida
weather.test.js	X	Suite fallida
itineraries.test.js	X	Suite fallida
favoriteRoutes.test.js	X	Suite fallida
TOTAL	**10 failed, 10 total**	**13 failed, 13 total tests**

8.4.3. Business Rules - Pruebas Ejecutadas con Fallos

Cuadro 26: Pruebas Business Rules Ejecutadas

Archivo de Prueba	Resultado	Detalles
businessRules.test.js	X	Suite fallida
itinerary.test.js	X	Suite fallida
performance.test.js	X	Suite fallida
security.test.js	X	Suite fallida
trip.coverage.test.js	X	Suite fallida
TOTAL	**5 failed, 5 total**	**4 failed, 4 total tests**

8.5. Pruebas E2E Ejecutadas Exitosamente

Cuadro 27: Pruebas E2E Cypress Completadas

Prueba E2E	Resultado	Descripción
Carga la página principal	✓	Verifica texto "TRAVELBRAIN"
Login exitoso	✓	Credenciales: test@test.com / admin1234
Login fallido	✓	Manejo de credenciales incorrectas
Crear un viaje	✓	Viaje a Paris con datos completos
Navegar al dashboard	✓	Acceso post-login
Acceder a destinos	✓	Navegación a página de destinos
Acceder al clima	✓	Navegación a página de clima
Acceder a perfil	✓	Navegación a página de perfil
Protección de rutas	✓	Redirección a /login sin auth

9. Análisis de Issues y Problemas Detectados

9.1. Issues Críticos Identificados

Cuadro 28: Issues Críticos por Categoría

Categoría	Cantidad	Descripción
Confiabilidad	94	Issues que pueden causar fallos en producción. Prioridad alta de corrección.
Mantenibilidad	286	Problemas que dificultan el mantenimiento del código.
Seguridad	1	Vulnerabilidades de seguridad detectadas.
Cobertura Insuficiente	-	Cobertura del 15.3 % está muy por debajo del estándar del 70 %.
Pruebas Fallidas	17 suites	Todas las suites de backend y business rules fallan.
Duplicación	2.6 %	Código duplicado que aumenta complejidad.

9.2. Áreas Problemáticas Específicas

9.2.1. Backend Project - Cobertura por Servicio

- **Services (1.12 %):** Muy baja cobertura en lógica de negocio
  - destinationBusinessRules.js: 1.4 %
  - itineraryBusinessRules.js: 1.53 %
  - routeBusinessRules.js: 0.79 %
  - tripBusinessRules.js: 1.05 %
  - userBusinessRules.js: 1.13 %
- **Controllers (22.64 %):** Cobertura insuficiente en controladores
- **Utils (13.46 %):** Funciones auxiliares poco probadas
- **Middlewares (24.29 %):** Módulos de seguridad con cobertura baja

9.2.2. Problemas en Pruebas Unitarias

1. **Configuración de Jest:** Posibles problemas con mocks
2. **Base de datos de prueba:** Configuración incorrecta
3. **Imports:** Errores en rutas de módulos
4. **Variables de entorno:** No configuradas para entorno de prueba

9.3. Recomendaciones de Corrección Inmediata

Cuadro 29: Plan de Corrección de Issues

Prioridad	Acción Correctiva	Estimado	
P1 (Crítico)	Corregir configuración de pruebas Jest en backend	4 horas	
P1 (Crítico)	Implementar pruebas para services (al menos 70 %)	8 horas	
P1 (Crítico)	Resolver issues de confiabilidad más graves	6 horas	
P2 (Alta)	Ejecutar pruebas de integración con Postman	4 horas	
P2 (Alta)	Ejecutar pruebas de seguridad con OWASP ZAP	6 horas	
P3 (Media)	Reducir duplicación de código	4 horas	
P3 (Media)	Mejorar cobertura de controllers y middlewares	6 horas	

10. Matriz Resumen de Casos de Prueba

Cuadro 30: Resumen de Casos de Prueba por Nivel

Nivel de Prueba	**Total Casos**	**Ejecutados**	**Exitosa**	**Fallida**	**Pendiente**
Unitarias (Backend)	18	10+	0	10+	8
Unitarias (Frontend)	15	21+	21+	0	0
Integración (APIs)	28	0	0	0	28
E2E (Cypress)	16	9	9	0	7
Seguridad (ZAP)	10	0	0	0	10
TOTAL	**87**	**40+**	**30+**	**10+**	**53**

Cuadro 31: Distribución por Módulo con Estado Actual

Módulo	**Casos Diseñados**	**Cobertura Estimada**	**Estado**
Autenticación Tradicional	12	85 %	!
Autenticación Biométrica	18	80 %	...
Gestión de Viajes (CRUD)	16	90 %	!
Clima	8	70 %	...
Administración	6	75 %	...
Seguridad General	10	Variable	...
Microservicio Facial	9	80 %	...
Middlewares	8	85 %	!
TOTAL	**87**	**82 % promedio**	**Mixto**

11. Plan de Acción para Sprint de Calidad 3

11.1. Objetivos del Sprint 3

Cuadro 32: Objetivos y Métricas Objetivo

Métrica	**Actual**	**Objetivo**	**Mejora**	
Cobertura de Código	15.3 %	¿=70 %	+54.7 %	
Issues de Confiabilidad	94	¡=20	-74	
Issues de Seguridad	1	0	-1	
Tasa de Pase de Tests	34.5 %	¿=90 %	+55.5 %	
Pruebas E2E Completadas	9/16	16/16	+7	
Pruebas Integración	0/28	28/28	+28	
Pruebas Seguridad	0/10	10/10	+10	

11.2. Cronograma de Actividades

Cuadro 33: Cronograma Sprint de Calidad 3

Semana	Actividades Principales	Horas	Responsable
Semana 1	1. Corrección de pruebas unitarias backend 2. Mejorar cobertura de services a ¿70 % 3. Resolver 20 issues de confiabilidad	20	Cáceres
Semana 2	1. Ejecutar pruebas de integración (28 casos) 2. Ejecutar pruebas de seguridad OWASP ZAP 3. Implementar pruebas de rendimiento	18	Anthony
Semana 3	1. Completar pruebas E2E faltantes (7 casos) 2. Generar reporte final de calidad 3. Validación final con SonarQube	15	Ambos
TOTAL		**53 horas**	

11.3. Recursos Necesarios

- **Herramientas:**
  - OWASP ZAP para pruebas de seguridad
  - Postman para pruebas de integración
  - Cypress para pruebas E2E adicionales
  - Jest para mejorar pruebas unitarias
- **Entornos:**
  - Entorno de desarrollo para correcciones
  - Entorno de staging para pruebas integración
  - Base de datos de prueba para pruebas unitarias
- **Datos de Prueba:**
  - Usuarios de prueba con diferentes roles
  - Datos de viajes, destinos, itinerarios
  - Casos de prueba para autenticación biométrica

12. Conclusiones y Recomendaciones

12.1. Logros Alcanzados

1. ✓**Infraestructura de Pruebas:** Configurada completamente
2. ✓**Automatización E2E:** 9 flujos críticos automatizados y funcionando
3. ✓**Pruebas Frontend:** Componentes y servicios validados exitosamente

4. ✓**Integración Continua:** SonarQube configurado y monitoreando calidad
5. ✓**Documentación:** Diseño completo de 87 casos de prueba

12.2. Áreas de Mejora Críticas

1. X**Cobertura de Código:** 15.3 % es insuficiente para producción
2. X**Pruebas Backend:** 100 % de suites fallando requiere atención inmediata
3. !**Issues de Confiabilidad:** 94 issues abiertos es riesgo alto
4. ...**Pruebas de Seguridad:** No ejecutadas aún (riesgo desconocido)
5. !**Duplicación de Código:** 2.6 % puede reducir mantenibilidad

12.3. Recomendaciones Técnicas

12.3.1. Inmediatas (Sprint 3):

1. Priorizar corrección de pruebas unitarias backend
2. Implementar pipeline de CI/CD que bloquee con baja cobertura
3. Establecer estándar mínimo de 70 % de cobertura
4. Ejecutar pruebas de seguridad antes de próximo despliegue

12.3.2. Mediano Plazo:

1. Implementar pruebas de rendimiento y carga
2. Establecer métricas de calidad como KPI del equipo
3. Automatizar más flujos E2E críticos
4. Implementar mutation testing para validar calidad de pruebas

12.3.3. Largo Plazo:

1. Establecer cultura de "quality first."[en] el equipo
2. Implementar trunk-based development con calidad
3. Automatizar completamente el proceso de release
4. Establecer programa de refactoring continuo

12.4. Evaluación General

Cuadro 34: Evaluación General del Estado de Calidad

Aspecto	**Calificación**	**Comentarios**
Diseño de Pruebas	8/10	87 casos bien documentados
Ejecución de Pruebas	4/10	Muchas pruebas pendientes o fallidas
Cobertura de Código	2/10	Muy por debajo del estándar
Automatización	7/10	Buena infraestructura, pero incompleta
Seguridad	5/10	1 issue detectado, pruebas pendientes
Documentación	9/10	Excelente documentación de pruebas
PROMEDIO	**5.8/10**	**Requiere mejora significativa**

12.5. Conclusión Final

El proyecto TravelBrain cuenta con una **excelente base de diseño de pruebas** (87 casos documentados) y **infraestructura de automatización** configurada. Sin embargo, la **ejecución real de pruebas** ha revelado problemas críticos que deben atenderse urgentemente:

1. **Las pruebas unitarias del backend están completamente fallando** - Esto impide validar la lógica central del sistema.

2. **La cobertura del 15.3 % es inaceptable** para un sistema que maneja datos sensibles de usuarios.

3. **Los 94 issues de confiabilidad** representan un riesgo alto para la estabilidad en producción.

Recomendación: Concentrar todos los esfuerzos del Sprint 3 en corregir las pruebas unitarias y mejorar la cobertura antes de proceder con nuevas funcionalidades. La calidad actual del código no es suficiente para un despliegue en producción.

A. Anexos

A.1. Anexo A: Configuración de Cypress

```
const { defineConfig } = require("cypress");

module.exports = defineConfig({
  e2e: {
    baseUrl: "https://travelbrain.ddns.net",
    specPattern: "cypress/e2e/**/*.cy.{js,jsx,ts,tsx}",
    supportFile: "cypress/support/e2e.js",
    video: true,
    screenshotOnRunFailure: true,
    viewportWidth: 1280,
    viewportHeight: 720,
    defaultCommandTimeout: 10000,
    requestTimeout: 10000,
    responseTimeout: 10000,
    env: {
      apiUrl: "https://travelbrain.ddns.net/api"
    }
  }
});
```

Listing 6: cypress.config.js - Configuración Real

A.2. Anexo B: Comandos Personalizados Cypress Ejecutados

```
// Login command usado en pruebas
Cypress.Commands.add("login", (email, password) => {
  cy.session([email, password], () => {
    cy.visit('/login')
    cy.get('input[name="email"]').type(email)
    cy.get('input[type="password"]').type(password)
    cy.get('button[type="submit"]').click()
    cy.url().should('include', '/dashboard')
  })
})

// Comando para crear viaje
Cypress.Commands.add("createTrip", (tripData) => {
  cy.visit('/trips')
  cy.contains('New Trip').click()

  cy.get('input[name="title"]').type(tripData.title)
  cy.get('input[name="destination"]').type(tripData.destination)
  cy.get('input[name="startDate"]').type(tripData.startDate)
  cy.get('input[name="endDate"]').type(tripData.endDate)
  cy.get('input[name="budget"]').type(tripData.budget)

  cy.get('button[type="submit"]').click()
  cy.contains('Trip created successfully').should('be.visible')
})
```

Listing 7: cypress/support/commands.js - Comandos Reales

A.3. Anexo C: Configuración de Jest para Backend

```
module.exports = {
  testEnvironment: 'node',
  roots: ['<rootDir>/tests'],
  testMatch: ['**/*.test.js'],
  collectCoverage: true,
  coverageDirectory: 'coverage',
  coverageReporters: ['text', 'lcov', 'html'],
  collectCoverageFrom: [
    'src/**/*.js',
    '!src/**/*.test.js',
    '!src/**/index.js'
  ],
  coverageThreshold: {
    global: {
      branches: 70,
      functions: 70,
      lines: 70,
      statements: 70
    }
  },
  setupFilesAfterEnv: ['<rootDir>/tests/setup.js']
};
```

Listing 8: jest.config.js - Backend Project

A.4. Anexo D: Reporte de SonarQube (Extracto)

```
QUALITY GATE: PASSED (with warnings)

=== METRICS ===
Lines of Code: 19,001
Coverage: 15.3%
Duplications: 2.6%
Issues: 381
  - Security: 1
  - Reliability: 94 (C - Critical)
  - Maintainability: 286 (A - Acceptable)

=== MODULE ANALYSIS ===
backend-project/src:
  Lines: 3,064
  Coverage: 17.2%
  Issues: 31

business-rules-backend/src:
  Lines: 1,505
  Coverage: 7.5%
  Issues: 108

frontend-react/src:
  Lines: 14,432
  Coverage: 17.1%
  Issues: 242

=== RECOMMENDATIONS ===
```

```
1. Increase code coverage to at least 70%
2. Fix critical reliability issues
3. Reduce code duplication
4. Implement security testing
```

Listing 9: Resultados SonarQube - Extracto

A.5. Anexo E: Pruebas E2E Ejecutadas - Evidencia

```
Tests Ejecutados: 9
Passed: 9
Failed: 0
Duration: ~3 minutos

Detalle de pruebas:
1. [OK] travelbrain.e2e.cy.js
   - Carga la página principal
   - Login de usuario exitoso
   - Login fallido con credenciales incorrectas

2. [OK] trips.e2e.cy.js
   - Crear un viaje exitosamente
   - Navegar al dashboard después del login

3. [OK] navigation.e2e.cy.js
   - Acceder a la página de destinos
   - Acceder a la página del clima
   - Acceder a la página de perfil
   - Verificar protección de rutas sin autenticación

URLs probadas:
- https://travelbrain.ddns.net/
- https://travelbrain.ddns.net/login
- https://travelbrain.ddns.net/dashboard
- https://travelbrain.ddns.net/trips
- https://travelbrain.ddns.net/destinations
- https://travelbrain.ddns.net/weather
- https://travelbrain.ddns.net/profile

Credenciales usadas:
- Email: test@test.com
- Password: admin1234
```

Listing 10: Resumen Pruebas E2E Ejecutadas

A.6. Anexo F: Issues de SonarQube - Ejemplos

Cuadro 35: Ejemplos de Issues Detectados por SonarQube

Tipo	**Descripción**	**Severidad**
Vulnerabilidad	Hardcoded credentials in configuration file	MEDIUM
Bug	Possible null pointer dereference in userController	MAJOR
Code Smell	Function has too many parameters (8)	MINOR
Duplicación	Same logic found in tripService and itineraryService	MAJOR
Cobertura	Method calculateBudget() has 0 % coverage	CRITICAL
Seguridad	Missing input validation in register endpoint	MAJOR