Universidad Politécnica de Querétaro

Ingeniería en Sistemas computacionales

Materia: Ingeniería de requisitos

“Pruebas (Comprobación de cumplimiento de los requisitos de software)”

Docente: Iván Peredo Valderrama

Por: Diego García García, Andrés Joshua León Barranco, Alberto Luna Rufino, José Ángel Sánchez Linarez, Erik Narciso Bernardino, Eduardo Daniel Licea González y Alberto Adrián Muñiz López

Grupo: S-204

21/nov/2025

1. Pruebas de caja negra (funcionalidad externa)

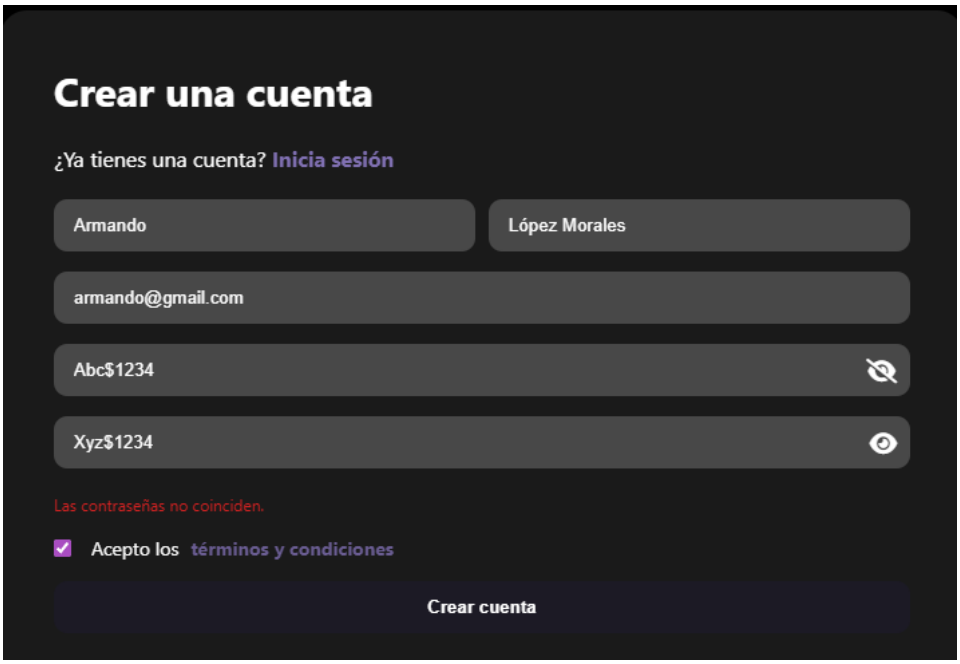
Objetivo: verificar que las funcionalidades del sistema responden correctamente a las entradas del usuario sin considerar la estructura interna del código.

Caso de prueba CN-01: Registro de usuario con validaciones

Descripción: verificar que el sistema impida el registro si las contraseñas no coinciden o si el formato del correo es inválido.

ID	Pasos a realizar	Datos de entrada	Resultado esperado	Estado
CN-01	1. Ir a la página de registro. 2. Llenar nombre y apellido. 3. Ingresar contraseñas diferentes. 4. Intentar enviar el formulario.	Nombre: Armando López Morales Pass 1: Abc$1234 Pass 2: Xyz$1234	El sistema debe mostrar una alerta indicando que las contraseñas no coinciden y no permitir el envío.	Exitoso

Resultado obtenido:



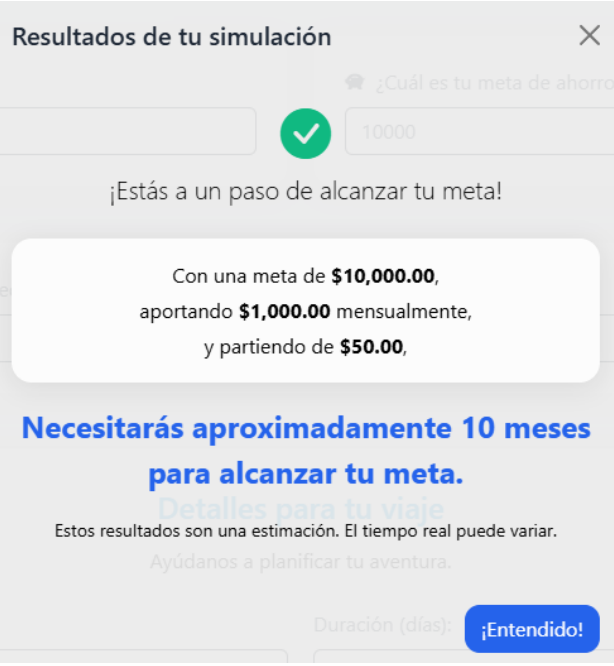
Nota de ejecución: al ingresar contraseñas distintas, el sistema bloqueó el envío y mostró el mensaje de error en rojo correspondiente.

Caso de prueba CN-02: Simulador de ahorro

Descripción: verificar que el simulador calcula correctamente el tiempo necesario para alcanzar una meta financiera.

ID	Pasos a realizar	Datos de entrada	Resultado esperado	Estado
CN-02	**1. Ingresar al simulador de ahorro.** **2. Seleccionar la meta "Viaje".** **3. Llenar los campos numéricos requeridos.**	**Meta: $10,000** **Ahorro actual: $0** **Aporte mensual: $1,000**	**El sistema debe mostrar un resultado de "10 meses" para alcanzar la meta.**	**Exitoso**

Resultado obtenido:



Nota de ejecución: el cálculo se realizó correctamente mostrando el tiempo estimado exacto según los datos ingresados.

2. Pruebas de caja gris (integración front-back-datos)

Objetivo: verificar que las acciones realizadas en la interfaz web se reflejan correctamente en la base de datos SQL Server y que el flujo de datos es íntegro.

Caso de prueba CG-01: Persistencia de datos en registro

Descripción: confirmar que al registrar un usuario desde la web, se crean los registros correspondientes en las tablas Usuarios y DatosP con la contraseña encriptada.

ID	Pasos a realizar	Validación en BD	Resultado esperado	Estado
CG-01	1. Registrar un usuario nuevo "PruebaGris" en la web. 2. Abrir SQL Server Management Studio. 3. Ejecutar una consulta de selección.	SELECT * FROM Usuarios WHERE correo_electronico = '...'	Debe existir un nuevo registro en la tabla. El campo contrasena debe contener un hash encriptado, no texto plano.	Exitoso

Resultado obtenido:

Resultados | Mensajes

	id	id_datosP	correo_electronico	contrasena	fecha_registro	ultima_sesion	test_completado
1	6	6	pruebagris@gmail.com	scrypt:32768:8:1$RWUTSG2qmKUf62aZ$783b5086aacc5c...	2025-11-21 21:10:07.110	NULL	0

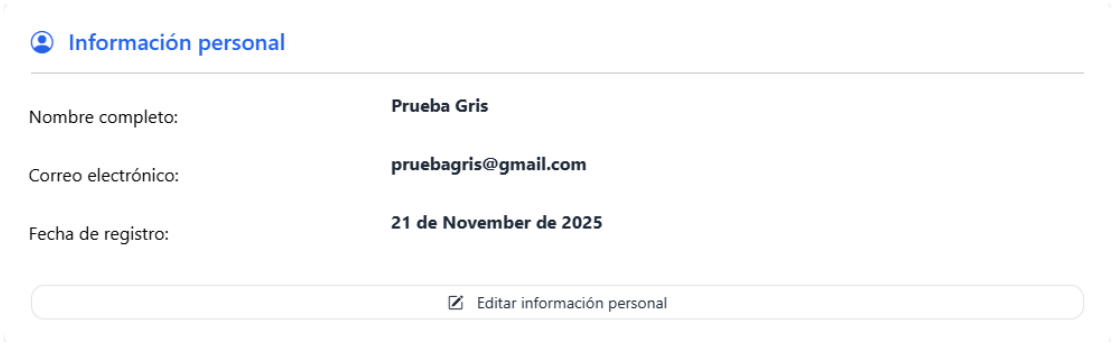
Nota de ejecución: se validó que el usuario fue insertado correctamente y la contraseña se almacenó de forma segura (hash).

Caso de prueba CG-02: Actualización de perfil

Descripción: verificar que el cambio de nombre realizado en el dashboard actualiza correctamente el registro en la tabla DatosP.

ID	Pasos a realizar	Validación en BD	Resultado esperado	Estado
CG-02	1. Iniciar sesión en el sistema. 2. Ir a "Mi perfil" -> "Editar". 3. Cambiar el nombre a "NombreNuevo" y guardar.	Consulta SQL a la tabla DatosP	El campo nombre en la tabla DatosP debe reflejar el valor "NombreNuevo" para el ID de usuario correspondiente.	Exitoso

Resultado obtenido:



3. Pruebas de caja blanca (lógica interna del código)

Objetivo: validar la lógica interna de las funciones, flujos de control (condicionales) y manejo de excepciones directamente en el código fuente.

Caso de prueba CB-01: Lógica de diagnóstico financiero (API Python)

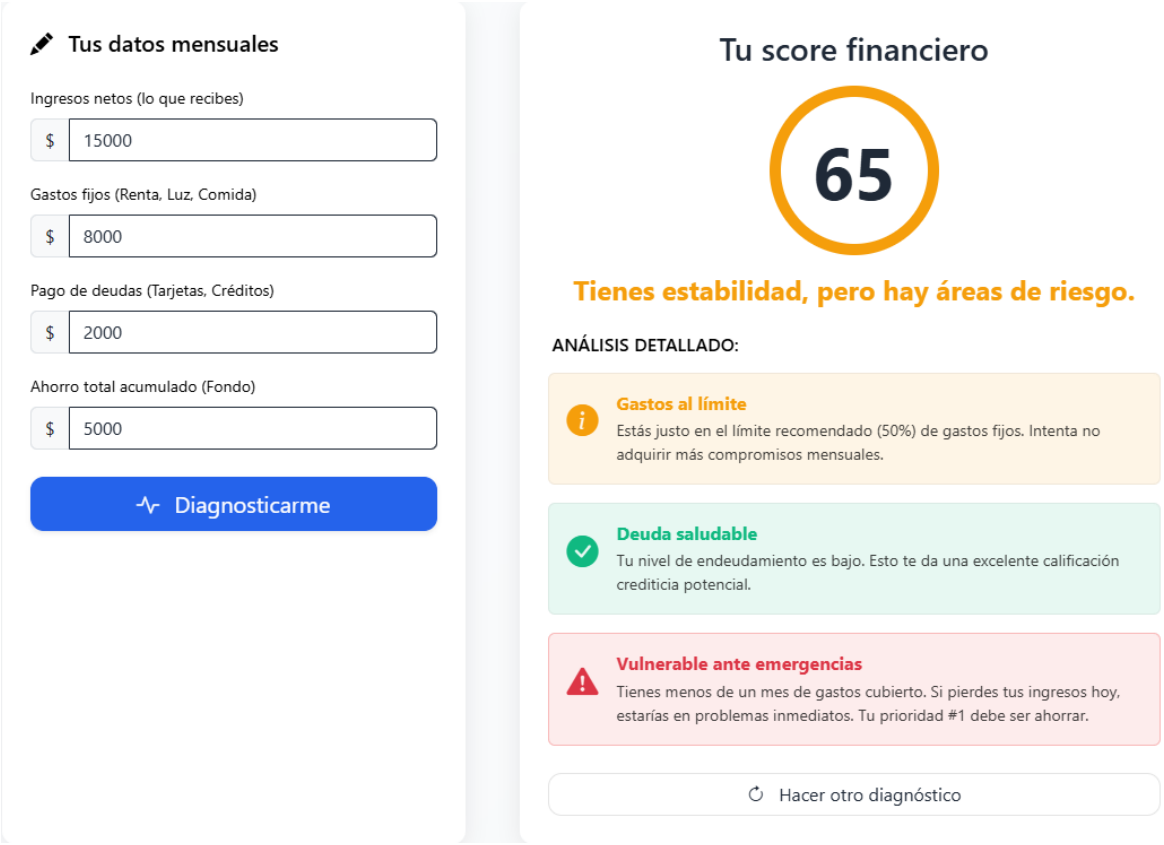
Descripción: probar la lógica condicional en la función calcular_salud del backend (app.py) que determina la penalización en el puntaje de salud financiera.

Código analizado:

```python
# 1. ANÁLISIS DE GASTOS FIJOS (Ideal: < 50% de ingresos)
ratio_gastos = (gastos / ingresos) * 100
if ratio_gastos > 60:
    puntaje -= 25
    analisis.append({
        "tipo": "gasto",
        "estado": "mal",
        "titulo": "Gastos fijos altos",
        "texto": f"Tus gastos fijos consumen el {ratio_gastos:.0f}% de tu ingreso. Lo ideal es mantenerlos bajo el 50% para tener margen de maniobra."
    })
elif ratio_gastos > 50:
    puntaje -= 10
    analisis.append({
        "tipo": "gasto",
        "estado": "regular",
        "titulo": "Gastos al límite",
        "texto": "Estás justo en el límite recomendado (50%) de gastos fijos. Intenta no adquirir más compromisos mensuales."
    })
```

ID	Pasos a realizar	Datos de entrada (JSON)	Resultado esperado (lógica)	Estado
CB-01	Enviar petición POST a /api/calcular_salud simulando gastos altos para forzar la condición.	Ingresos: 10000 Gastos: 7000 (Ratio: 70%)	El código debe entrar al primer bloque if (>60) y restar 25 puntos al puntaje inicial.	Exitoso

Resultado obtenido:



Nota de ejecución: la API respondió con un puntaje penalizado y la recomendación de alerta, confirmando que la lógica del if se ejecutó correctamente.

Caso de prueba CB-02: Lógica de cálculo de meses (JavaScript)

Descripción: validar el bloque condicional en simulacion_ahorro.js que maneja el cálculo de tiempo restante y evita divisiones por cero.

Código analizado:

```
if (currentAmount >= goalAmount) {
    document.getElementById('modalResultText').textContent = `¡Felicidades! Ya alcanzaste tu meta de ${formatCurrency(goalAmount)}.`;
} else if (monthlyContribution === 0) {
    document.getElementById('modalResultText').textContent = `Para alcanzar tu meta de ${formatCurrency(goalAmount)}, necesitas aportar más de ${formatCurrenc
} else {
    const remainingAmount = goalAmount - currentAmount;
    const monthsNeeded = Math.ceil(remainingAmount / monthlyContribution);

    let resultText = '';
    if (monthsNeeded === 1) {
        resultText = `Necesitarás aproximadamente 1 mes para alcanzar tu meta.`;
```

ID	Pasos a realizar	Variables JS	Resultado esperado (lógica)	Estado
CB-02	Ejecutar la función de cálculo ingresando un aporte mensual de cero.	goalAmount: 5000 monthlyContribution: 0	El código debe entrar al else if específico y mostrar un mensaje advirtiendo que se necesita aportar más.	Exitoso

Resultado obtenido: