

# TP DISEÑO

## ANALIZADOR DE GASTOS

### **Análisis:**

El presente proyecto tiene como objetivo desarrollar un sistema que permita a los usuarios registrar, clasificar y analizar sus gastos personales o familiares. El sistema proporcionará herramientas visuales para la interpretación de la información, ayudando a mejorar la gestión financiera del usuario.

### Objetivo General:

Desarrollar un sistema de software que permita el registro, clasificación y análisis de gastos e ingresos personales de manera eficiente y visual.

### Requerimientos del Sistema

#### Requerimientos Funcionales:

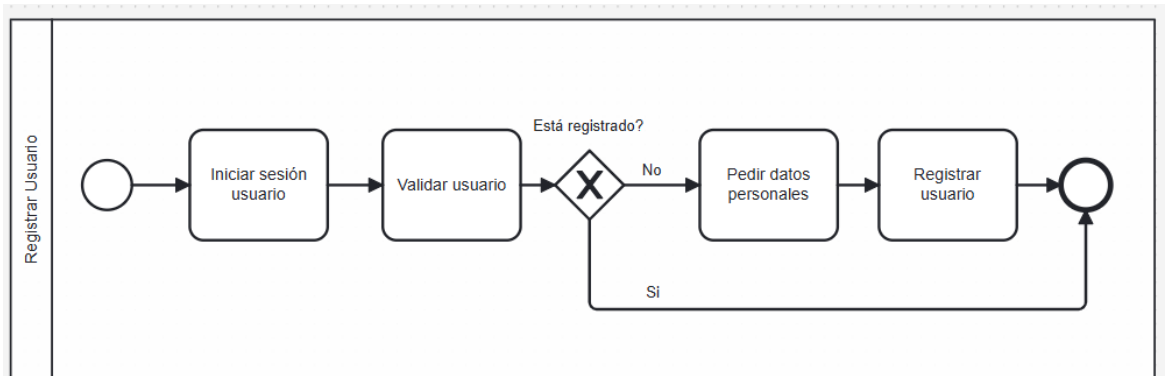
- El sistema debe permitir crear una cuenta de usuario.
- El usuario debe poder iniciar sesión.
- El sistema debe permitir registrar gastos e ingresos con fecha, monto y categoría.
- El usuario puede crear categorías personalizadas.
- El sistema debe permitir definir presupuestos por categoría.
- El usuario puede consultar su historial de transacciones y los presupuestos definidos para cada categoría.

#### Requerimientos No Funcionales:

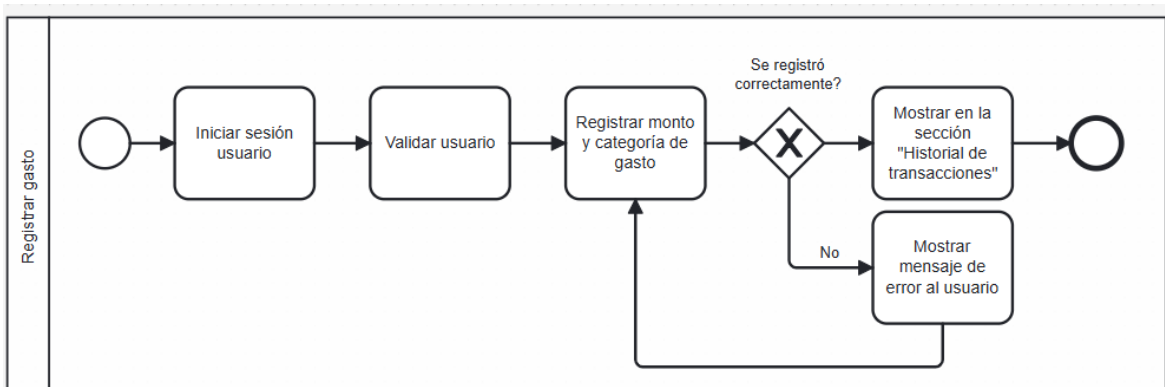
- La interfaz debe ser amigable e intuitiva.
- El sistema debe funcionar correctamente en dispositivos móviles y de escritorio.

## BPMN

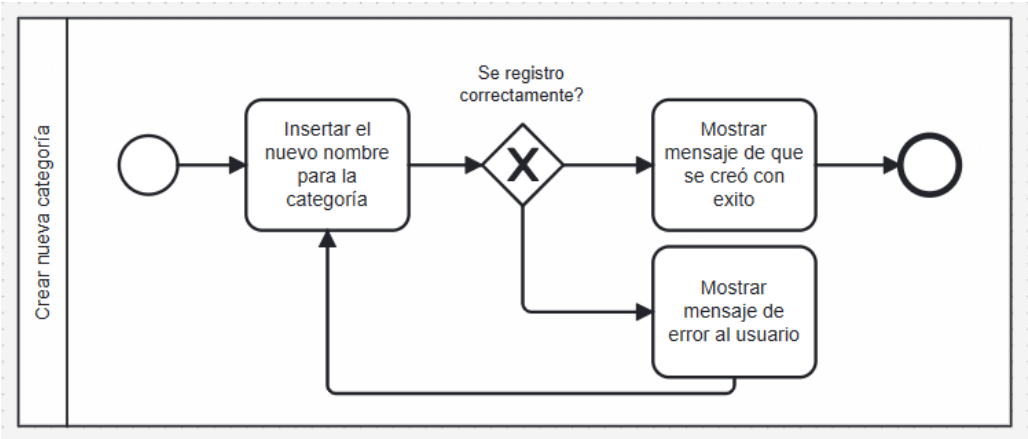
**Registrar Usuario:** este proceso permite que un nuevo usuario se registre en el sistema. Se inicia con el inicio de sesión, valida si el usuario ya está registrado y, en caso contrario, solicita sus datos personales, los guarda y finaliza.



**Registrar Gasto:** permite a un usuario logueado cargar un gasto indicando monto y categoría. Se valida la información ingresada, y si es correcta, se guarda el registro. Si hay error, se informa al usuario.



Crear nueva categoría:



Definir presupuesto por categoría:

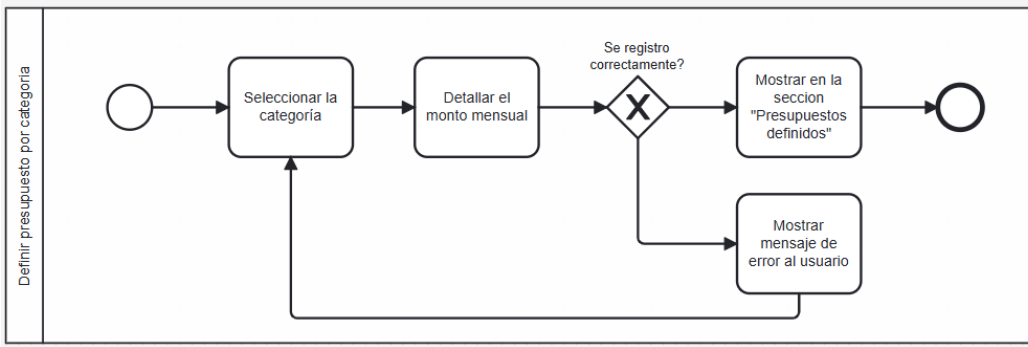
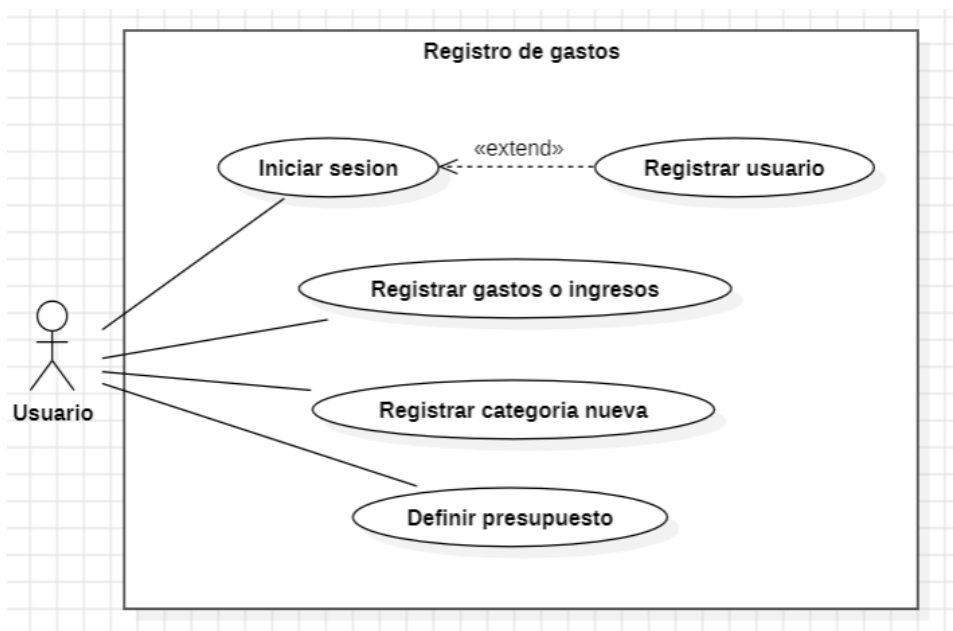


Diagrama de casos de uso



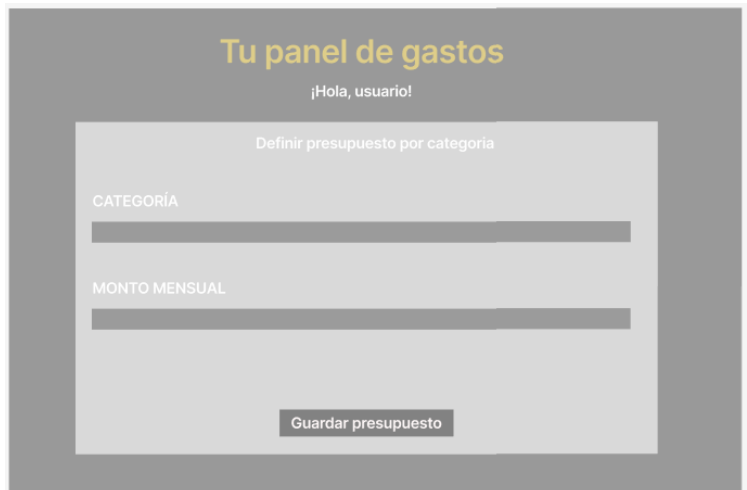
Caso de Uso: Iniciar sesión	
ID: 1	Fecha:
Descripción:	
Actores Principales: Usuario	Actores Secundarios:
Observaciones:	
Precondiciones: -	
Post- Condiciones	Éxito: Se inicia sesión correctamente.
	Fracaso:
Flujo PRINCIPAL	Flujo Alternativo
1. El caso de uso comienza cuando el usuario ingresa al analizador de gastos y quiere iniciar sesión	
2. Se le pide al usuario que ingrese su correo de email y contraseña si ya está registrado.	2.1. Si no está registrado, se le piden datos personales y se registra correctamente al usuario.
3. El usuario pudo ingresar exitosamente.	
4. Fin del CU	

Caso de Uso: Registrar gastos o ingresos	
ID: 1	Fecha:
Descripción:	
Actores Principales: Usuario	Actores Secundarios:
Observaciones:	
Precondiciones: -	
Post- Condiciones	Éxito: Se registra correctamente el gasto o el ingreso.
	Fracaso:
Flujo PRINCIPAL	Flujo Alternativo
1. El caso de uso comienza cuando el usuario ingresa al analizador de gastos.	
2. Se le pide al usuario que ingrese su correo de email y contraseña para iniciar sesión.	
3. El usuario elije si es gasto o ingreso, inserta la fecha, el monto y la categoría del gasto/ingreso.	
4. Se registra correctamente el gasto o ingreso.	4.1. Si el gasto/ingreso no se <u>registro</u> correctamente, mostrar mensaje de error.
5. Fin del CU.	

Caso de Uso: Registrar categoría nueva	
ID: 1	Fecha:
Descripción:	
Actores Principales: Usuario	Actores Secundarios:
Observaciones:	
Precondiciones: -	
Post- Condiciones	Éxito: Se registra la nueva categoría correctamente.
	Fracaso:
Flujo PRINCIPAL	Flujo Alternativo
1. El caso de uso comienza cuando el usuario ingresa al analizador de gastos.	
2. Se le pide al usuario que ingrese su correo de email y contraseña para iniciar sesión.	
3. El usuario se le pide el nombre para su nueva categoría	
4. Se registra correctamente la nueva categoría	4.1. Si la nueva categoría no se registró correctamente, mostrar mensaje de error.
5. Fin del CU	

Caso de Uso: Definir presupuesto		
ID: 1		Fecha:
Descripción:		
Actores Principales: Usuario		Actores Secundarios:
Observaciones:		
Precondiciones: -		
Post- Condiciones	Éxito: Se registra el presupuesto para la categoría correctamente.	
	Fracaso:	
Flujo PRINCIPAL		Flujo Alternativo
1. El caso de uso comienza cuando el usuario ingresa al analizador de gastos.		
2. Se le pide al usuario que ingrese su correo de email y contraseña para iniciar sesión.		
3. El usuario se le pide el nombre de la categoría y el presupuesto máximo que le quiera poner.		
4. Se registra correctamente el presupuesto.		4.1. Si el presupuesto no se registró correctamente, mostrar mensaje de error.
5. Fin del CU		

## Interfaz



Sistema:

Adjunto a este PDF.

Testing:

Test Case ID	AG_001	Test Case Description	Verificar que el usuario pueda iniciar sesión correctamente al ingresar email y contraseña válidos.		
Created By	Constanza	Reviewed By		Version	1.0
Tester's Name	Constanza	Date Tested	08-mayo-2025	Test Case (Pass/Fail/Not Executed)	Pass

S #	Prerequisites:
1	El usuario debe tener acceso al archivo iniciar-sesion.html.
2	El archivo panel.html debe existir y estar en la misma carpeta.
3	Se debe contar con un navegador actualizado (por ejemplo, Google Chrome).

S #	Test Data
1	Correo Electrónico: dksjds@sds.com
2	Contraseña: 123456

Test Scenario	Verificar que, al ingresar un correo y contraseña válidos, el sistema redirige correctamente al panel.
---------------	--

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Abrir iniciar-sesion.html en el navegador	Se muestra el formulario de login	Como se esperaba	Pass
2	Ingresar correo y contraseña válidos	Los datos pueden ingresarse	Como se esperaba	Pass
3	Hacer clic en el botón "Entrar"	Redirige a panel.html	Como se esperaba	Pass
4	Verificar que se muestre el título "Tu Panel de Gastos"	El usuario accede al panel correctamente	Como se esperaba	Pass

Test Case ID	AG_002	Test Case Description	Verificar que un nuevo usuario pueda registrarse correctamente con datos válidos.		
Created By	Constanza	Reviewed By	Constanza	Version	1.0
Tester's Name	Constanza	Date Tested	30-junio-2025	Test Case (Pass/Fail/Not Executed)	Pass

S #	Prerequisites:
1	Acceso a registro.html
2	No debe existir ya el correo nuevo@ejemplo.com en localStorage
3	Navegador actualizado (Chrome, Firefox)
4	

S #	Test Data
1	Nombre: Ana Pérez
2	Correo: nuevo@ejemplo.com
3	Contraseña: abc123
4	Confirmación: abc123

Test Scenario	Al ingresar todos los datos requeridos correctamente y hacer clic en "Registrarse", el sistema debe crear al usuario y redirigir al panel.
---------------	--

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Abrir registro.html en navegador	Se muestra el formulario de registro	Como se esperaba	Pass
2	Completar todos los campos válidamente	Se permiten ingresar los datos	Como se esperaba	Pass
3	Hacer clic en "Registrarse"	Se crea el usuario y se redirige a panel.html	Como se esperaba	Pass
4	Verificar panel	Se muestra el título "Tu Panel de Gastos"	Como se esperaba	Pass



Test Case ID	AG_003	Test Case Description	Verificar que al ingresar un gasto válido, este se registre correctamente y aparezca en el historial.		
Created By	Constanza	Reviewed By	Constanza	Version	1.0
Tester's Name	Constanza	Date Tested	30-junio-2025	Test Case (Pass/Fail/Not Executed)	Pass

S #	Prerequisites:
1	Usuario logueado y en panel.html
2	Existencia de categoría "comida"
3	
4	

S #	Test Data
1	Tipo: gasto
2	Monto: 2000
3	Fecha: 2025-06-30
4	Categoría: comida

**Test Scenario** Registrar un gasto con todos los campos completos correctamente y verificar que se agregue al historial de transacciones.

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Completar formulario de gasto	El formulario permite ingresar los datos	Como se esperaba	Pass
2	Hacer clic en "Registrar"	Se agrega al historial	Como se esperaba	Pass
3	Ver historial	Aparece gasto de \$2000 en categoría comida	Como se esperaba	Pass

Test Case ID	AG_004	Test Case Description	Verificar que el sistema no permita registrar si hay campos vacíos.		
Created By	Constanza	Reviewed By	Constanza	Version	1.0
Tester's Name	Constanza	Date Tested	30-junio-2025	Test Case (Pass/Fail/Not Executed)	Pass

S #	Prerequisites:
1	Acceso a registro.html
2	
3	
4	

S #	Test Data
1	Nombre: [vacío]
2	Correo: vacio@ejemplo.com
3	Contraseña: 123456
4	Confirmación: 123456

**Test Scenario** Intentar registrarse sin completar todos los campos.

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Dejar el campo "Nombre" vacío	Se bloquea el botón o muestra alerta	Como se esperaba	Pass
2	Intentar registrarse	No se permite registrar	Como se esperaba	Pass

Test Case ID	AG_005	Test Case Description	Verificar que no se pueda crear una categoría que ya existe.		
Created By	Constanza	Reviewed By	Constanza	Version	1.0
Tester's Name	Constanza	Date Tested	30-junio-2025	Test Case (Pass/Fail/Not Executed)	Pass

S #	Prerequisites:
1	Usuario logueado en panel.html
2	Ya existe la categoría "kiosco"
3	
4	

S #	Test Data
1	Nueva categoría: kiosco
2	
3	
4	

**Test Scenario** Intentar agregar una categoría duplicada.

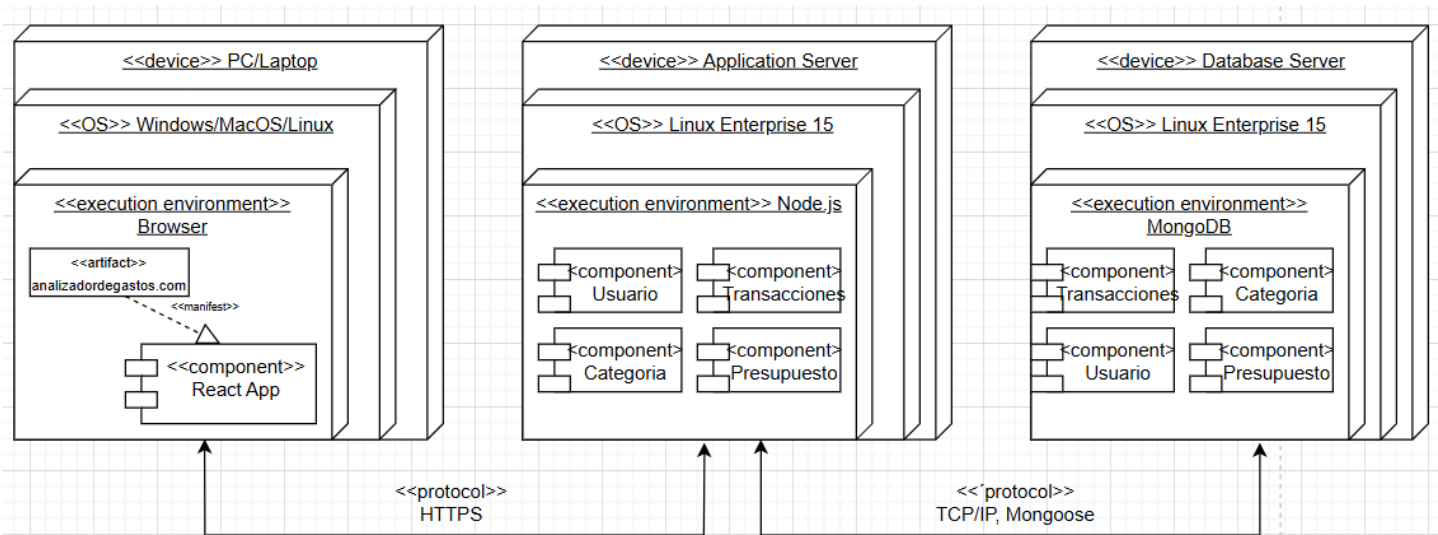
Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Ir a la sección de categorías	Se muestra el formulario de nueva categoría	Como se esperaba	Pass
2	Ingresar nombre "kiosco"	Se muestra alerta de categoría ya existente	Como se esperaba	Pass

# ENTREGA DE LA SEGUNDA ITERACION

## Arquitectura del Sistema

El sistema Analizador de Gastos utiliza una arquitectura cliente-servidor con una estructura en capas. La arquitectura actual está organizada en tres capas principales:

- **Capa de Presentación:** esta capa está desarrollada en React, junto con HTML y CSS. Se encarga de mostrar la interfaz al usuario, recolectar datos desde formularios (como login, registro, gastos, ingresos, presupuestos) y ofrecer retroalimentación visual. Cada pantalla (Inicio, Login, Registro, Panel) está representada por un componente React.
- **Capa de Lógica de Negocio:** la lógica del sistema está implementada con JavaScript dentro de los propios componentes React. Aquí se validan los datos ingresados, se controla el flujo de navegación, se manejan los errores, y se define cómo se actualiza la información en la interfaz. Se usan hooks como useState y useEffect para controlar los estados y efectos.
- **Capa de Datos:** se simula usando localStorage, que permite guardar información en el navegador sin necesidad de un servidor. Allí se almacenan las transacciones, categorías creadas por el usuario y los presupuestos definidos.



## **Flujo de datos entre capas**

- El usuario interactúa con la interfaz (por ejemplo, escribe un correo y contraseña, o registra un gasto).
- Esa información se envía a la lógica de negocio que la valida, determina si se debe mostrar un mensaje de error, agregar un elemento al historial o guardar algo nuevo.
- Si la validación es correcta, los datos se guardan en localStorage. Por ejemplo: se guarda una nueva transacción como un objeto dentro de un array en JSON.
- Cuando se vuelve a cargar la aplicación, React utiliza useEffect para leer los datos desde localStorage y mostrarlos automáticamente en pantalla.

Todo esto ocurre en el navegador del usuario, sin necesidad de un servidor real.

## **Clasificación de errores por gravedad**

### **Errores menores (Frontend)**

- Campos vacíos en formularios  
Ejemplo: dejar vacío el correo o la contraseña al iniciar sesión.  
Tratamiento: se bloquea el botón de enviar o se muestra un alert().
- Montos negativos o mal formateados  
Ejemplo: ingresar "-100" como monto de un gasto.  
Tratamiento: se valida el número y se bloquea el envío si es incorrecto.

### **Errores graves (Frontend)**

- Contraseñas no coinciden en el registro  
Ejemplo: escribir una contraseña y confirmarla con otra distinta.  
Tratamiento: se muestra un mensaje de error y no se crea el usuario.
- Categoría duplicada  
Ejemplo: intentar crear una categoría llamada "comida" cuando ya existe.  
Tratamiento: el sistema revisa si ya existe y alerta al usuario si se repite.
- Presupuesto duplicado por categoría  
Ejemplo: definir dos presupuestos distintos para "transporte".  
Tratamiento: el sistema valida que no exista antes de guardar.

## Advertencias (a futuro)

- Gasto que supera el presupuesto definido  
Ejemplo: gastar \$12000 en "comida" cuando el presupuesto era \$10000.  
Tratamiento: aún no se implementó, pero se puede mostrar una advertencia visual.

## Limitaciones actuales

- Como el sistema todavía no tiene base de datos, toda la información se guarda en el navegador con localStorage. Si el usuario borra los datos del navegador, cambia de dispositivo o usa modo incógnito, puede perderse la información. Esto no es un error del sistema, sino una limitación de cómo se guarda todo por ahora.
- Sin validación de sesión o usuarios reales  
Ejemplo: cualquiera puede acceder al panel sin estar autenticado.

## Simulación de API

- **GET /usuarios/{id}**  
**Devuelve los datos del usuario**  
{  
 "id": 1,  
 "nombre": "Constanza Garelo",  
 "correo": "coti@example.com"  
}
- **POST /usuarios**  
**Crea una cuenta nueva**  
{  
 "nombre": "Constanza Garelo",  
 "correo": "coti@example.com",  
 "contraseña": "123456"  
}
- **POST /login**  
**Inicia sesión con usuario y contraseña**  
{  
 "correo": "coti@example.com",  
 "contraseña": "123456"  
}  
**Respuesta:**  
{  
 "mensaje": "Login exitoso",  
 "token": "abc123"  
}
- **GET /transacciones**  
**Devuelve todas las transacciones del usuario**  
{

```

      "transacciones": [
    {
      "id": 1,
      "tipo": "gasto",
      "monto": 2500,
      "fecha": "2025-05-28",
      "categoria": "comida"
    },
    {
      "id": 2,
      "tipo": "ingreso",
      "monto": 80000,
      "fecha": "2025-05-01",
      "categoria": "salario"
    }
  ]
}

```

- **POST /transacciones**

**Registra una transacción nueva**

```

{
  "tipo": "gasto",
  "monto": 1000,
  "fecha": "2025-05-29",
  "categoria": "transporte"
}

```

- **PUT /transacciones/{id}**

**Edita una transacción existente**

```

{
  "id": 1,
  "tipo": "gasto",
  "monto": 1200,
  "fecha": "2025-05-29",
  "categoria": "comida"
}

```

- **DELETE /transacciones/{id}**

**Elimina una transacción**

- **GET /categorias**

**Devuelve las categorías creadas por el usuario**

```

{
  "categorias": ["comida", "transporte", "salud", "educación"]
}

```

- **POST /categorias**

**Agrega una nueva categoría**

```

{
  "nombre": "mascotas"
}

```

- ```
}
• GET /presupuestos
Lista los presupuestos por categoría
{
  "presupuestos": [
    {
      "categoria": "comida",
      "monto": 15000
    },
    {
      "categoria": "transporte",
      "monto": 6000
    }
  ]
}
```
- ```
• POST /presupuestos
Agrega un presupuesto por categoría
{
  "categoria": "comida",
  "monto": 10000
}
```

### Casos de prueba (tests como hipótesis)

- **Hipótesis 1:** Si un usuario intenta crear una categoría que ya existe, el sistema debe mostrar una alerta y no agregarla.
- **Hipótesis 2:** Si se dejan campos vacíos en cualquier formulario, no se debe permitir el envío.
- **Hipótesis 3:** Si se registra una transacción válida, debe aparecer en el historial.
- **Hipótesis 4:** Si se define un presupuesto para una categoría, no debe poder definirse otro duplicado.