**Gestión de Gastos**

**BPMN**

Para comenzar el desarrollo del sistema de gestión de gastos personales, decidí modelar el proceso general mediante un diagrama BPMN, ya que es una herramienta visual y estandarizada que permite representar de forma clara y precisa el flujo de actividades que realizará el usuario. El modelo parte desde el momento en que el usuario ingresa al sistema, lo cual se representa con un evento de inicio. Esto marca el punto donde comienza la interacción con la aplicación, un paso necesario en cualquier sistema que tenga una interfaz de usuario.

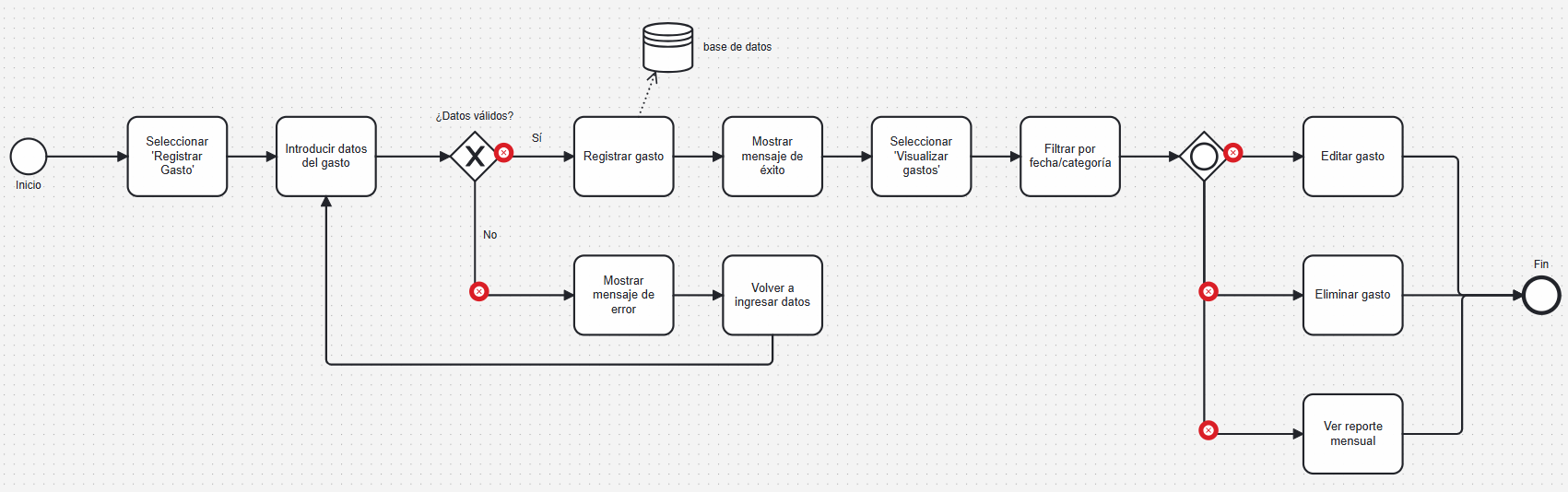
La primera actividad que planteé fue la opción de registrar un gasto. Esta funcionalidad es la base del sistema, ya que el objetivo principal del usuario es poder llevar un control de sus egresos. Por eso, diseñé una tarea que refleja la selección de esa función en el menú o interfaz. A continuación, se incorpora la tarea de ingresar los datos del gasto, que representa el formulario donde el usuario completará información como fecha, monto, categoría y descripción del gasto. Consideré importante validar esos datos, por lo que incluí un gateway de decisión que evalúa si los datos introducidos son válidos. Si hay errores, el sistema muestra un mensaje de advertencia y le da al usuario la opción de corregir los datos, lo cual es esencial para mantener la calidad de la información almacenada.

Cuando los datos son correctos, el sistema procede a registrar el gasto en la base de datos. Esta tarea representa el proceso de guardar la información y, una vez que se completa correctamente, el sistema muestra un mensaje de confirmación. Esto mejora la experiencia del usuario al brindarle feedback claro sobre el éxito de su acción.

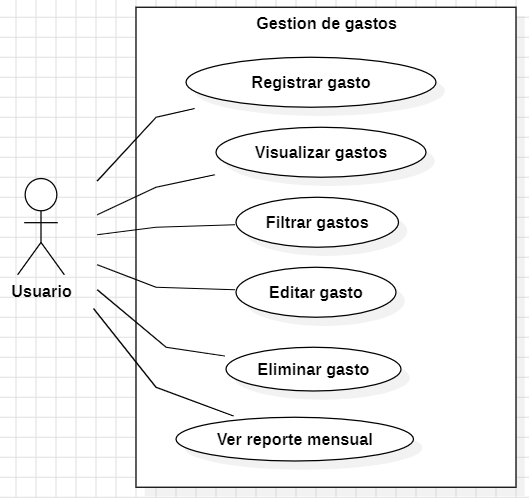
Luego de registrar un gasto, es natural que el usuario quiera revisar su historial. Por eso incorporé la opción de visualizar los gastos. Esta parte del proceso permite al usuario acceder a sus registros anteriores y aplicar filtros, como por fecha o categoría, para encontrar información específica. Después del filtrado, planteé una nueva puerta de decisión que permite al usuario elegir entre editar un gasto, eliminarlo o bien generar un reporte mensual. Esta ramificación se justifica en que, al visualizar sus datos, el usuario puede detectar errores y querer corregirlos, o simplemente desear eliminar entradas innecesarias o duplicadas.

Las tareas de editar y eliminar reflejan funcionalidades necesarias para dar flexibilidad y control al usuario sobre su información. Por otro lado, la visualización de reportes mensuales se diseñó para brindar una visión global del estado financiero del usuario, con análisis resumido por categorías o totales, algo muy útil para la toma de decisiones. Finalmente, todas estas posibles acciones convergen hacia el evento de fin del proceso, lo que representa que la interacción del usuario con el sistema ha concluido satisfactoriamente.

Elegí este enfoque porque cubre las funciones clave que se esperan de un sistema de este tipo: registrar, consultar, modificar y analizar. Cada paso tiene sentido dentro del flujo de uso real que tendría una persona que quiera administrar sus gastos personales, y el uso de BPMN permite visualizar claramente cómo se conectan esas acciones. Además, este modelo servirá como base sólida para continuar con el prototipado de pantallas y la posterior implementación.

****

**CASO DE USO Y DESCRIPCIONES**



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Caso de Uso**: Registrar Gasto | | | | |
| **ID: 1** | | | **Fecha: --** | |
| **Descripción:** Permite al usuario ingresar un nuevo gasto al sistema incluyendo monto, categoría, fecha y descripción. | | | | |
| **Actores Principales**: Usuario. | | **Actores Secundarios**: -- | | |
| **Observaciones: --** | | | | |
| **Precondiciones**: El usuario debe estar autenticado. | | | | |
| **Post- Condiciones** | **Éxito:** El gasto se registra correctamente en el sistema. | | | |
| **Fracaso:** Error en la validación de los datos o fallo en el sistema. | | | |
| Flujo principal | | | | Flujo Alternativo |
| 1. El caso de uso comienza cuando el **usuario** accede a la opción **registrar gasto**. | | | |  |
| 2. El **sistema** solicita los datos del **nuevo gasto**. | | | |  |
| 3. El **usuario** completa los campos requeridos. | | | |  |
| 4. El **sistema** valida la información. | | | |  |
| 5. Los **datos** son válidos, el **sistema** guarda el **nuevo gasto** y muestra un mensaje de confirmación. | | | | 5.1 Los datos **no** son válidos.  5.2 Se muestra mensaje de error y se solicita la corrección de los mismos. |
| **Fin de CU.** | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Caso de Uso**: Visualizar gastos | | | | |
| **ID: 2** | | | **Fecha: --** | |
| **Descripción:** Muestra todo los gastos registrados, permitiendo su ordenación por fecha o categoría. | | | | |
| **Actores Principales**: Usuario. | | **Actores Secundarios**: -- | | |
| **Observaciones: --** | | | | |
| **Precondiciones**: El usuario debe haber registrado al menos un gasto. | | | | |
| **Post- Condiciones** | **Éxito:** El sistema despliega la lista de gastos correctamente. | | | |
| **Fracaso:** No se pueden recuperar los datos por error del sistema. | | | |
| Flujo principal | | | | Flujo Alternativo |
| 1. El caso de uso comienza cuando el **usuario** accede al listado de gastos. | | | |  |
| 2. El **sistema** recupera los datos de **gastos**. | | | | 2.1 No se recuperan los datos.  2.2 El sistema muestra un mensaje indicando el motivo. |
| 3. Se presentan los datos al usuario en forma de llista ordenada. | | | |  |
| **Fin de CU.** | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Caso de Uso**: Filtrar gastos | | | | |
| **ID: 3** | | | **Fecha: --** | |
| **Descripción:** Permite buscar gastos por fecha, categoría o monto | | | | |
| **Actores Principales**: Usuario. | | **Actores Secundarios**: -- | | |
| **Observaciones: --** | | | | |
| **Precondiciones**: Deben existir gastos registrados. | | | | |
| **Post- Condiciones** | **Éxito:** Se muestran los resultados filtrados. | | | |
| **Fracaso:** No se encuentran resultados. | | | |
| Flujo principal | | | | Flujo Alternativo |
| 1. El caso de uso comienza cuando el **usuario** ingresa la o las características deseadas. | | | |  |
| 2. El **sistema** aplica los filtros. | | | |  |
| 3. Se muestra la lista de gastos que coinciden. | | | | 3.1 Si no se encuentran coincidencias, se muestra mensaje adecuado. |
| **Fin de CU.** | | | | |

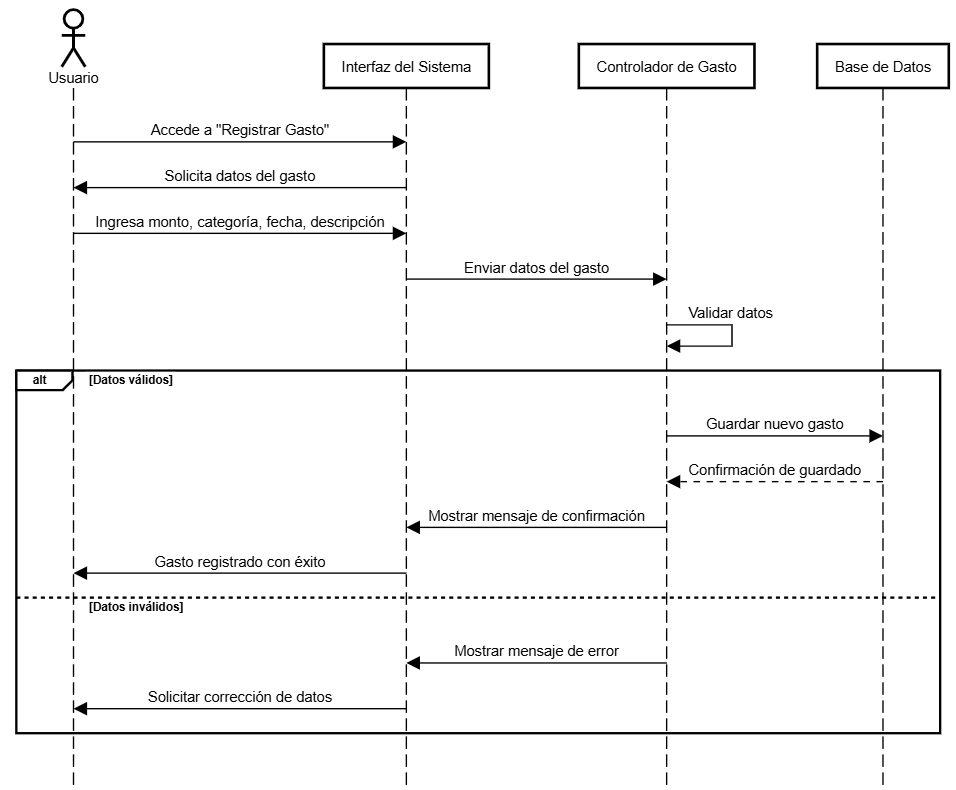
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Caso de Uso**: Editar gasto | | | | |
| **ID: 4** | | | **Fecha: --** | |
| **Descripción:** Permite modificar la información de un gasto existente. | | | | |
| **Actores Principales**: Usuario. | | **Actores Secundarios**: -- | | |
| **Observaciones: --** | | | | |
| **Precondiciones**: El usuario debe haber registrado gastos previamente. | | | | |
| **Post- Condiciones** | **Éxito:** El gasto se actualiza correctamente. | | | |
| **Fracaso:** El gasto no se encuentra o falla la validación. | | | |
| Flujo principal | | | | Flujo Alternativo |
| 1. El caso de uso comienza cuando el **usuario** selecciona un gasto para editar. | | | |  |
| 2. El **sistema** muestra los datos actuales | | | |  |
| 3. El **usuario** modifica la información. | | | |  |
| 4. El **sistema** valida y guarda los cambios. | | | |  |
| **Fin de CU.** | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Caso de Uso**: Eliminar gasto | | | | |
| **ID: 5** | | | **Fecha: --** | |
| **Descripción:** Permite eliminar un gasto específico del sistema. | | | | |
| **Actores Principales**: Usuario. | | **Actores Secundarios**: -- | | |
| **Observaciones: --** | | | | |
| **Precondiciones**: Debe existir al menos un gasto. | | | | |
| **Post- Condiciones** | **Éxito:** El gasto es eliminado del sistema. | | | |
| **Fracaso:** El gasto no se encuentra o falla la operación. | | | |
| Flujo principal | | | | Flujo Alternativo |
| 1. El caso de uso comienza cuando el **usuario** selecciona un gasto para eliminar. | | | |  |
| 2. El **sistema** solicita confirmación. | | | |  |
| 3. El **usuario** confirma la eliminación. | | | | 3.1 El usuario cancela la operación antes de confirmar. |
| 4. El **sistema** elimina el gasto. | | | |  |
| **Fin de CU.** | | | | |

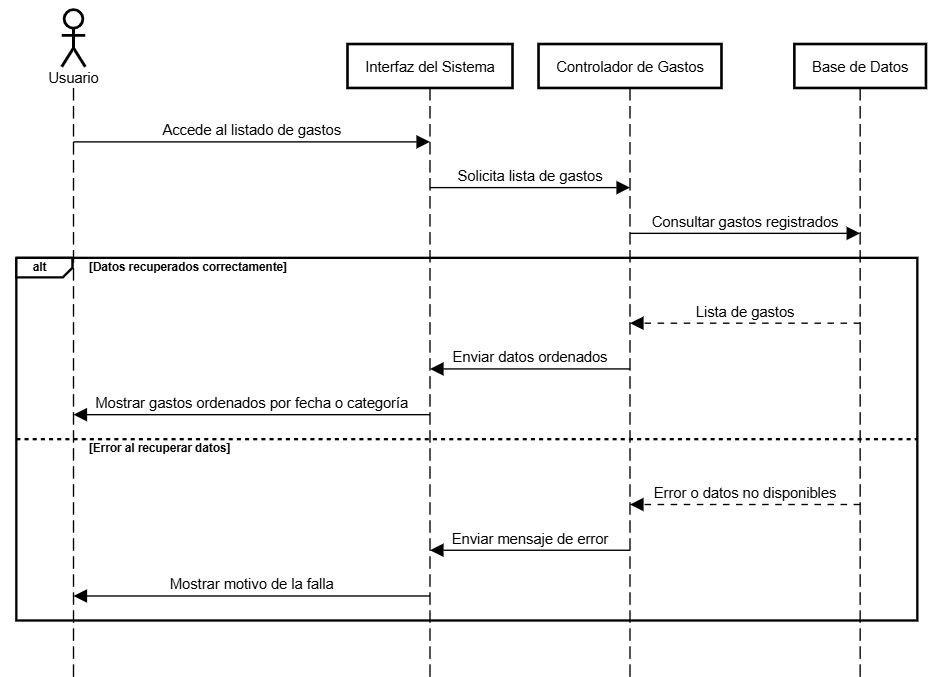
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Caso de Uso**: Ver reporte mensual | | | | |
| **ID: 6** | | | **Fecha: --** | |
| **Descripción:** Genera un resumen de los gastos del mes deseado agrupados por categoría. | | | | |
| **Actores Principales**: Usuario. | | **Actores Secundarios**: -- | | |
| **Observaciones: --** | | | | |
| **Precondiciones**: Debe existir gastos en el mes indicado. | | | | |
| **Post- Condiciones** | **Éxito:** Se muestra el reporte mensual al usuario. | | | |
| **Fracaso:** No hay datos para mostrar o falla la generación. | | | |
| Flujo principal | | | | Flujo Alternativo |
| 1. El caso de uso comienza cuando el **usuario** solicita el reporte mensual indicando el mes deseado. | | | |  |
| 2. El **sistema** filtra los gastos del mes deseado. | | | |  |
| 3. El **sistema** agrupa y calcula totales. | | | |  |
| 4. Se muestra el reporte al usuario. | | | |  |
| **Fin de CU.** | | | | |

**DIAGRAMAS DE SECUENCIA**

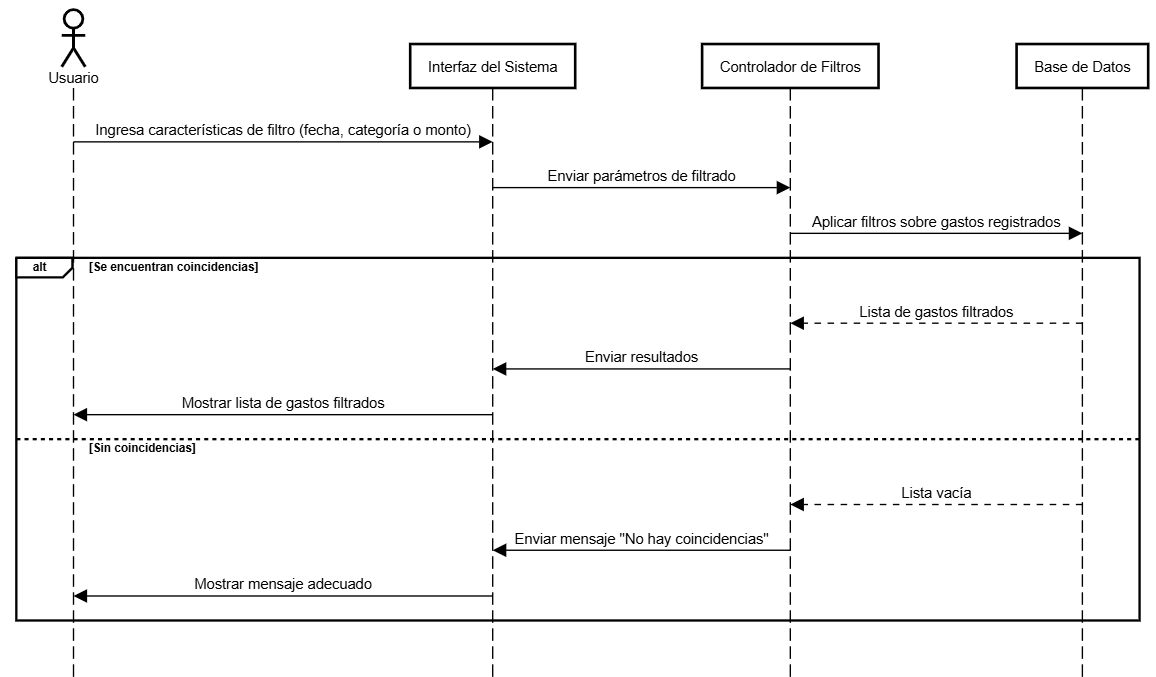
**Registra gasto**



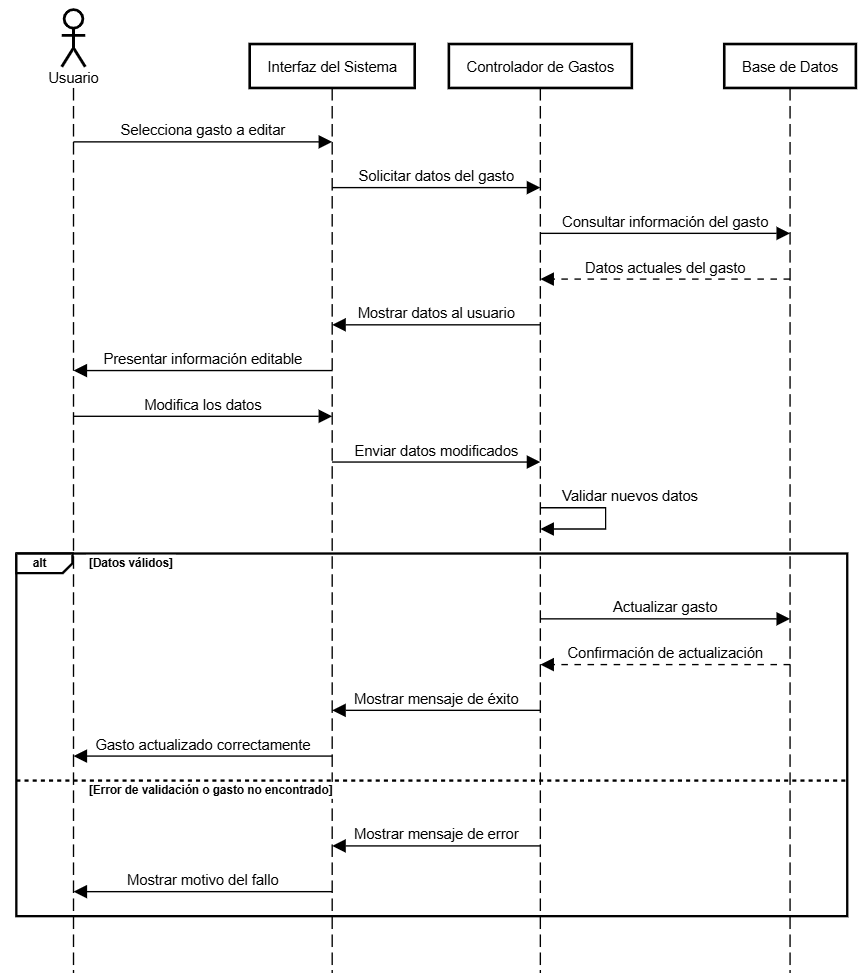
**Visualizar gasto**



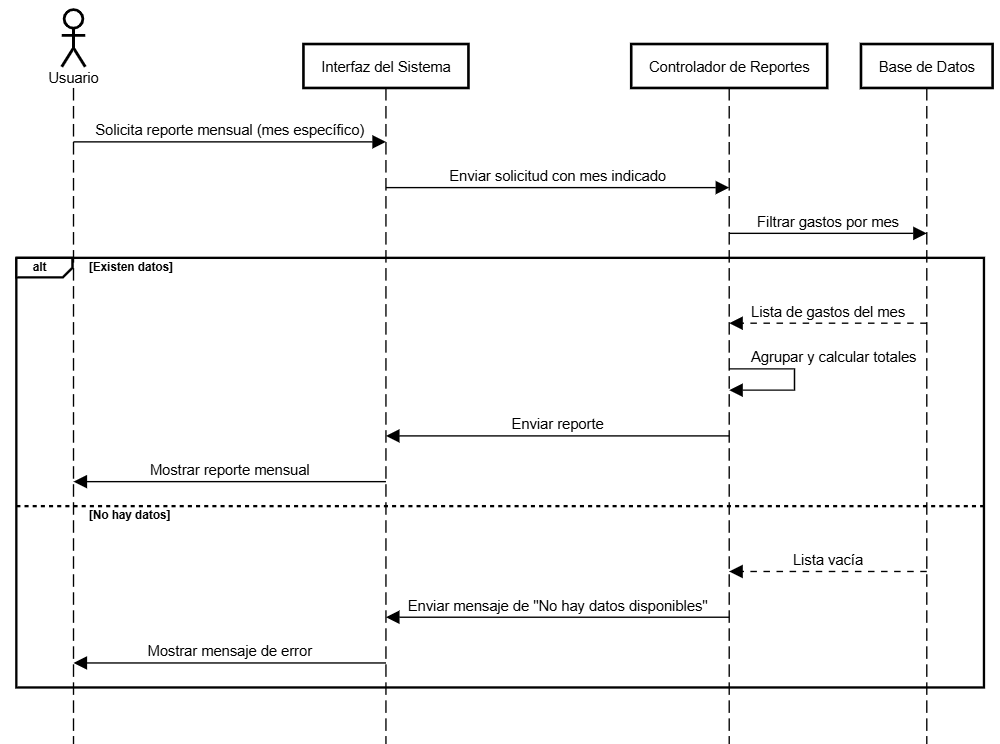
**Filtrar gastos**



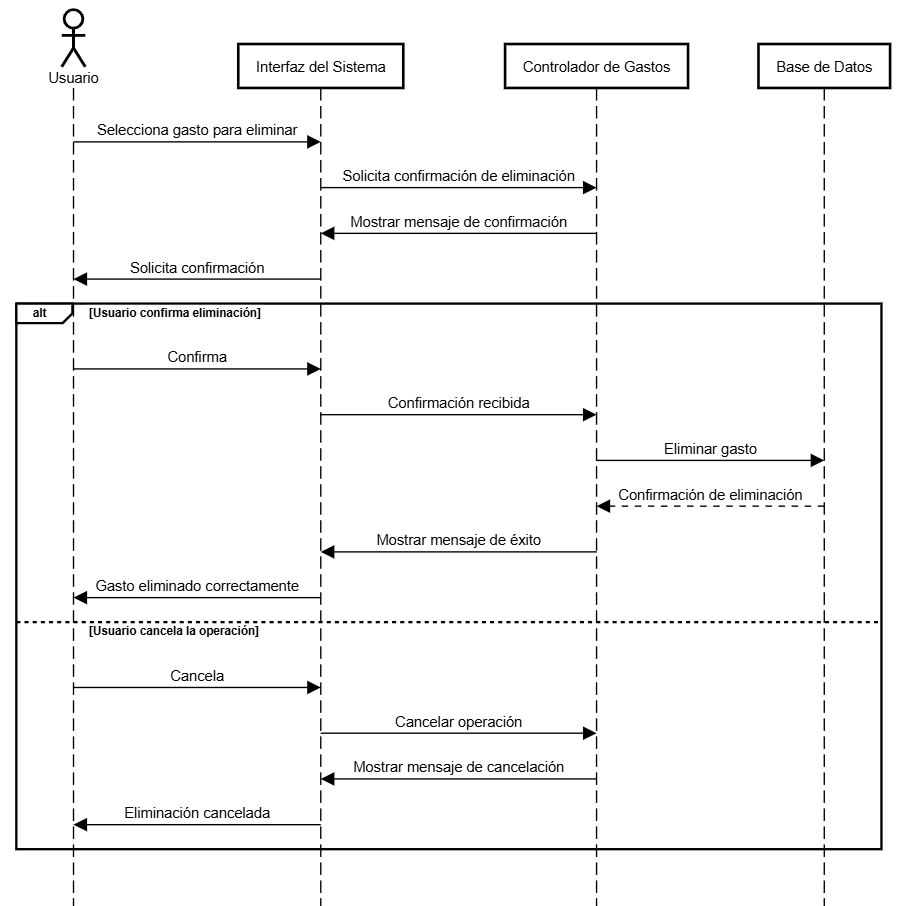
**Editar gastos**



**Reporte mensual**

****

**Eliminar gasto**

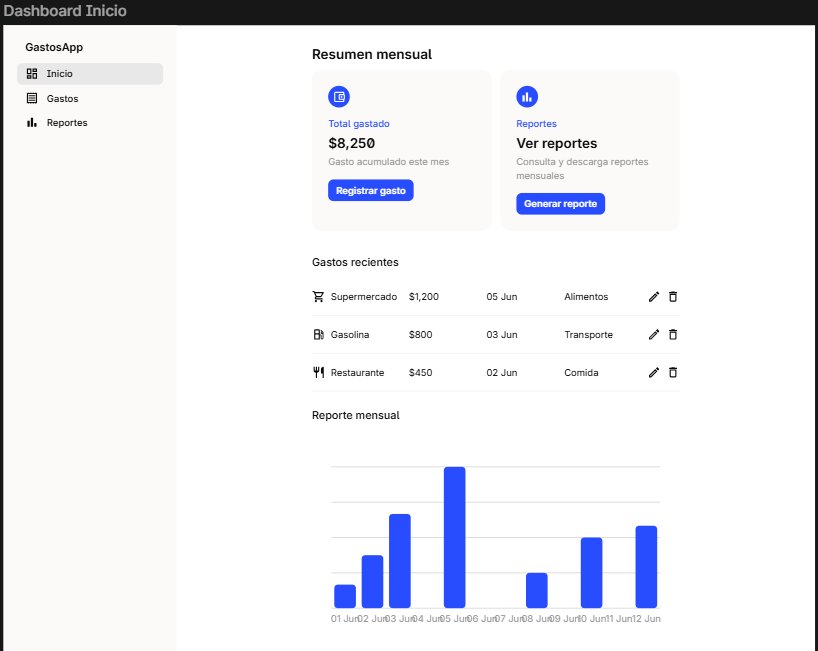


**PROTOTIPADO**

**Pantalla de inicio (Dashboard principal)**

La pantalla inicial del sistema debe ofrecer una visión general de lo gastado en el mes actual (el cual se actualiza con cada gasto del mes y se actualiza al comenzar un nuevo mes) y acceso rápido a las funciones principales, como registrar un nuevo gasto, visualizar los gastos existentes (con opción de editar o eliminar gastos), y generar reportes de un mes preciso.

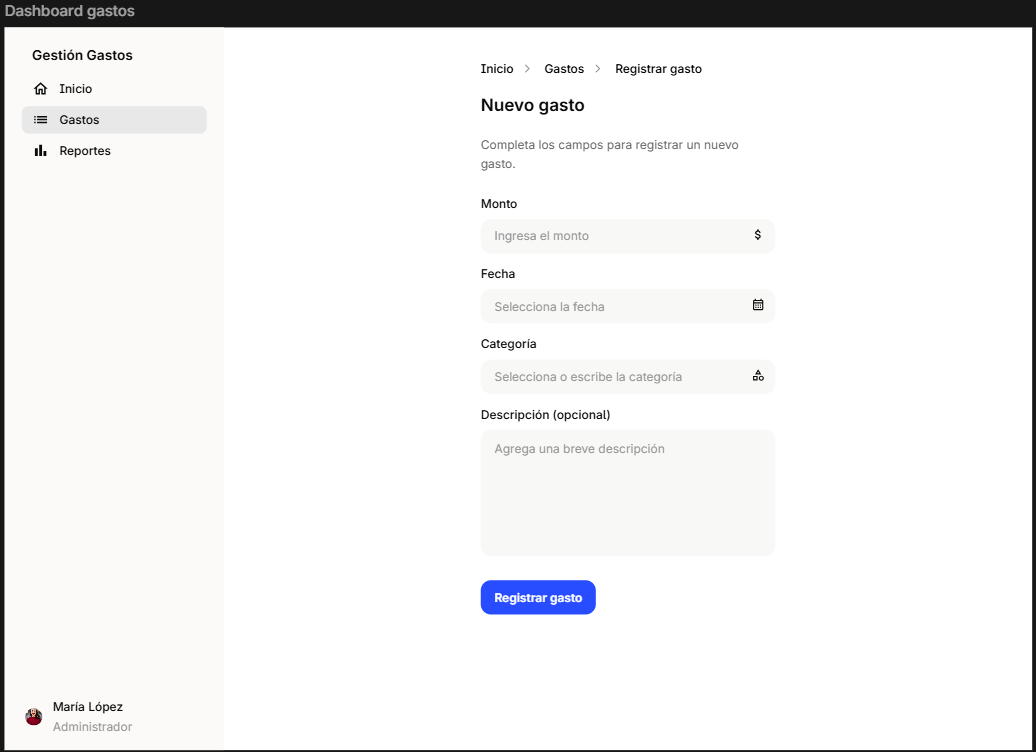
El dashboard debe ser claro y fácil de navegar. Los botones principales deberían estar destacados y ubicados en una posición visible para que el usuario pueda acceder rápidamente a las funciones más importantes.



**Pantalla de "Registrar Gasto"**

Esta pantalla debe presentar un formulario para que el usuario ingrese los detalles de un gasto: monto (solo números), fecha (indicar la fecha en el calendario), categoría y una breve descripción (único criterio no obligatorio).

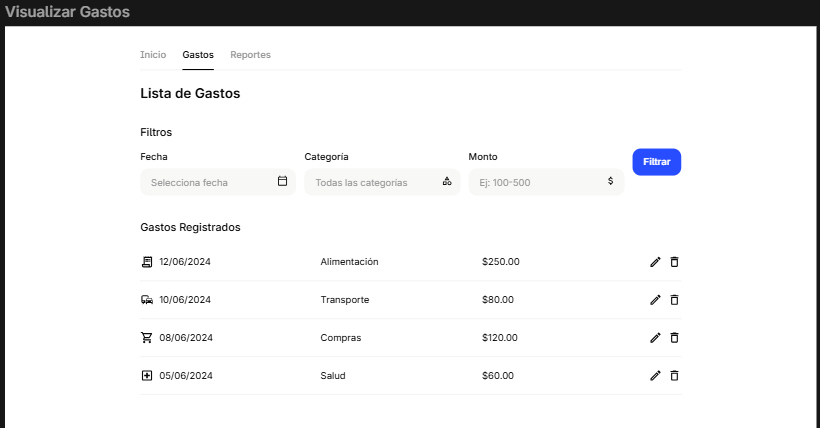
Es esencial que el formulario sea simple pero detallado, y que cada campo esté bien etiquetado. Esto facilitará la entrada de datos por parte del usuario y evitará errores. El sistema también debe validar los datos antes de permitir el registro, asegurándose de que la información ingresada sea coherente y no esté incompleta.



**Pantalla de "Visualizar Gastos"**

Aquí, el usuario podrá ver una lista de todos los gastos registrados, con detalles como la fecha, la categoría y el monto. Debería incluir opciones para filtrar los gastos, ya sea por fecha, categoría o monto. Esto permitirá al usuario revisar sus gastos más fácilmente y encontrar información específica rápidamente. También al visualizar un gasto tener la opción de editar o eliminar el mismo.

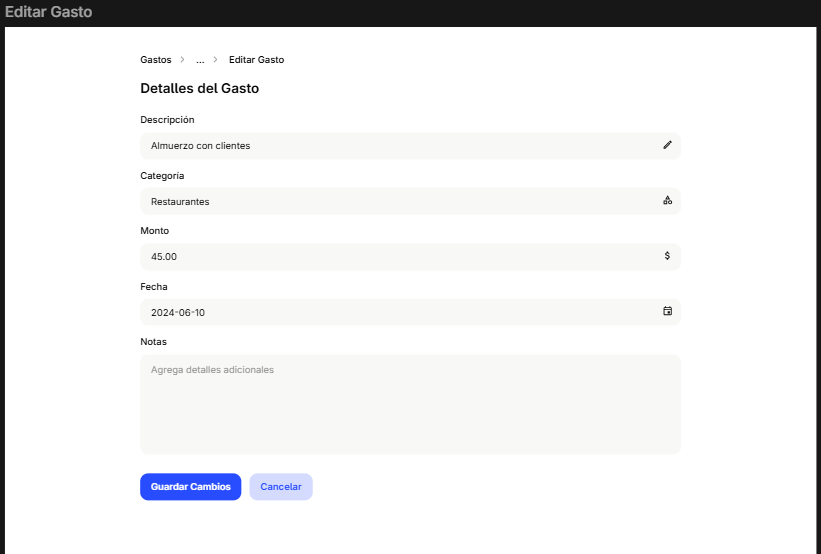
Esta pantalla facilita la revisión de los registros pasados. Los filtros permiten que el usuario busque información específica sin tener que navegar por toda la lista, haciendo la experiencia más eficiente. La interfaz debe ser limpia y ordenada, y los datos deben presentarse de manera clara y fácil de leer.



**Pantalla de "Editar Gasto"**

Cuando un usuario decide editar un gasto, esta pantalla debería mostrar los detalles del gasto seleccionado, permitiendo la modificación de los campos necesarios. Los cambios pueden ser hechos en los mismos campos que en la pantalla de registro, y el sistema debe validar nuevamente la información antes de permitir que se guarde.

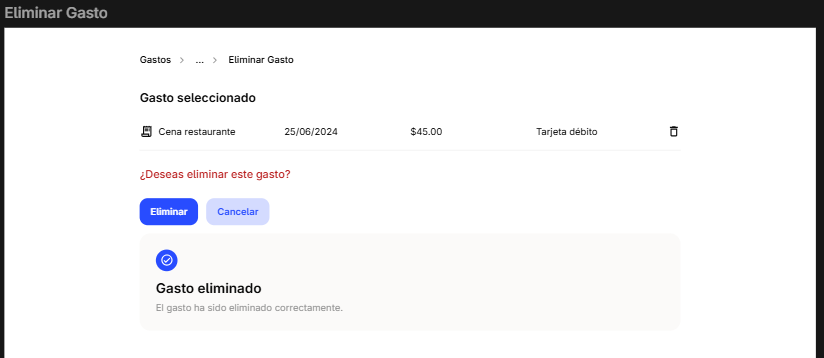
La capacidad de editar un gasto es crucial, ya que los usuarios pueden cometer errores al ingresar los datos. Al permitirles modificar cualquier campo del gasto registrado, el sistema ofrece flexibilidad y control sobre los datos. Debe asegurarse de que cualquier cambio sea validado para evitar la entrada de datos erróneos.



**Pantalla de "Eliminar Gasto"**

La opción para eliminar un gasto debe presentarse de forma clara, posiblemente con un botón de "Eliminar" junto al gasto específico. Al hacer clic en el botón, el sistema debería pedir una confirmación para asegurarse de que el usuario realmente desea eliminar ese gasto.

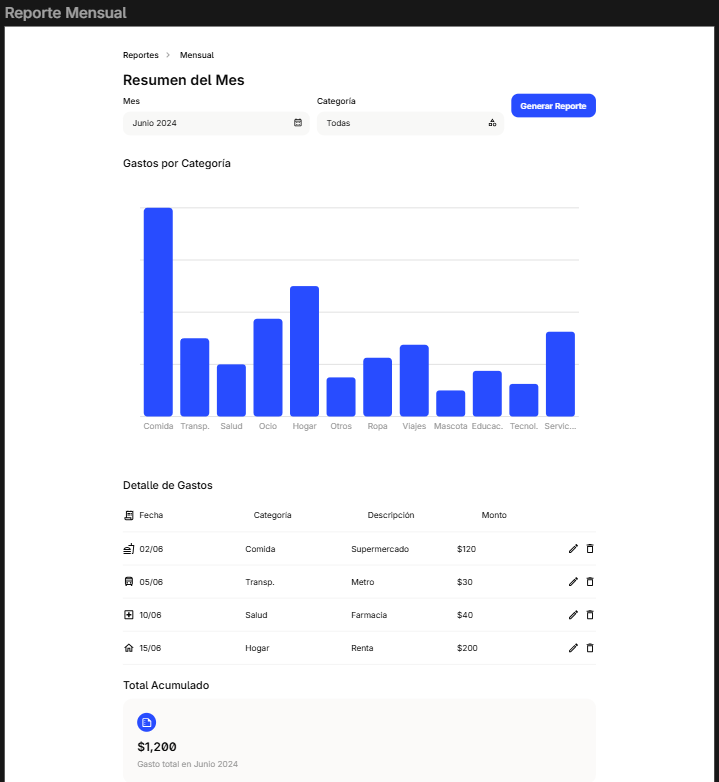
La eliminación de datos es una operación crítica, por lo que siempre debe pedirse confirmación antes de llevarla a cabo. Esto evita que los usuarios eliminen accidentalmente registros importantes. También, una vez que se elimina el gasto, debe mostrarse un mensaje de confirmación.



**Pantalla de "Generar Reporte Mensual"**

Esta pantalla debe permitir al usuario ver un resumen de sus gastos durante el mes seleccionado. El reporte puede incluir gráficos o tablas que desglosen los gastos por categoría, total acumulado, etc.

Los reportes son fundamentales para que el usuario pueda analizar sus finanzas y tomar decisiones informadas. Mostrar los datos de manera visual (por ejemplo, con gráficos) facilita la comprensión de los patrones de gasto. Además, debe ser fácil para el usuario generar estos reportes, y el sistema debe permitir la opción de filtrar por mes o por categoría.



**IMPLEMENTACIÓN**

https://gastos-app-ecru.vercel.app/

**CASOS DE PRUEBA**

|  |  |  |
| --- | --- | --- |
| **Título**: Registrar válido | | |
| **Descripción**: Verifica que el usuario pueda registrar un gasto correctamente. | | |
| **Precondición:** El usuario accede a la pantalla registro. | | |
| **Pasos para reproducir**:   1. Completar todos los campos del formulario. 2. Presionar “Guardar gasto”. | | |
| **Resultado esperado**: El gasto se guarda en localStorage y se redirige al inicio. | | **Resultado real**: El gasto fue guardado y redirigió correctamente. |
| **Estado**: Éxito. | **Nota**: -- | |
| **Justificación:** Es el flujo central del sistema: sin registro, no hay funcionalidad real. | | |

|  |  |  |
| --- | --- | --- |
| **Título**: Registro con campos vacíos | | |
| **Descripción**: Verifica que el sistema bloquee formularios incompletos. | | |
| **Precondición:** El usuario accede a la pantalla registro y no completa ningún campo. | | |
| **Pasos para reproducir**:   1. Presionar “Guardar gasto” sin completar el formulario. | | |
| **Resultado esperado**: El sistema muestra mensajes de validación ('Campo requerido'). | | **Resultado real**: Los mensajes se mostraron correctamente. |
| **Estado**: Éxito. | **Nota**: -- | |
| **Justificación:** Garantiza la integridad de los datos e impide entradas inválidas. | | |

|  |  |  |
| --- | --- | --- |
| **Título**: Visualización de gastos | | |
| **Descripción**: Verifica que los gastos se muestren correctamente en el dashboard. | | |
| **Precondición:** Hay al menos un gasto guardado previamente. | | |
| **Pasos para reproducir**:   1. Ir a la pantalla de inicio/dashboard. | | |
| **Resultado esperado**: El gasto aparece en la lista. | | **Resultado real**: El gasto fue mostrado correctamente. |
| **Estado**: Éxito. | **Nota**: -- | |
| **Justificación:** Confirma la persistencia y que el usuario ve lo que registra. | | |

|  |  |  |
| --- | --- | --- |
| **Título**: Edición de gasto. | | |
| **Descripción**: Verifica que el usuario pueda modificar un gasto existente. | | |
| **Precondición:** Existe un gasto previamente guardado. | | |
| **Pasos para reproducir**:   1. Ir a “Editar” del gasto elegido. 2. Modificar campo/s. 3. Presionar “Guardar”. | | |
| **Resultado esperado**: El gasto se actualiza correctamente en localStorage. | | **Resultado real**: El gasto fue actualizado y se muestra correctamente. |
| **Estado**: Éxito. | **Nota**: -- | |
| **Justificación:** Permite corregir errores y mejora la confiabilidad del sistema. | | |

|  |  |  |
| --- | --- | --- |
| **Título**: Eliminación de gasto | | |
| **Descripción**: Verifica que el usuario pueda eliminar un gasto. | | |
| **Precondición:** Existe un gasto listado en el dashboard. | | |
| **Pasos para reproducir**:   1. Hacer clic en “Eliminar” en el gasto. 2. Confirmar eliminación. | | |
| **Resultado esperado**: El gasto desaparece de la lista y de localStorage. | | **Resultado real**: El gasto fue eliminado exitosamente. |
| **Estado**: Éxito. | **Nota**: -- | |
| **Justificación:** Permite mantener limpio el registro de gastos y evitar duplicados. | | |

|  |  |  |
| --- | --- | --- |
| **Título**: Reporte mensual | | |
| **Descripción**: Verifica que el sistema agrupe y sume los gastos por mes. | | |
| **Precondición:** Existen múltiples gastos registrados con fechas distintas. | | |
| **Pasos para reproducir**:   1. Ir a la pantalla “Ver reporte mensual”. 2. Elegir una fecha y generar el reporte. | | |
| **Resultado esperado**: Se muestran totales mensuales correctamente agrupados. | | **Resultado real**: Los totales se calcularon y mostraron correctamente. |
| **Estado**: Éxito. | **Nota**: -- | |
| **Justificación:** Permite analizar el gasto mensual y tomar decisiones personales. | | |

**ARQUITECTURA DEL SISTEMA**

El sistema de gestión de gastos desarrollado sigue una arquitectura cliente–servidor basada en una SPA (Single Page Application), implementada con React + Vite en el lado del cliente y utilizando localStorage del navegador como persistencia simulada.

**Componentes principales:**

Cliente (Frontend - React)

* Aplicación desarrollada en React con Vite, donde se estructura el sistema a través de rutas (React Router) y componentes reutilizables.
* Cada funcionalidad (registro, edición, eliminación, visualización y reporte de gastos) está encapsulada en páginas y componentes bien definidos, respetando el patrón de diseño modular.

LocalStorage (Simulación de Backend)

* Se utiliza el localStorage del navegador para almacenar y recuperar los datos sin necesidad de un servidor. Esta aproximación permite simular la lógica de backend y mantener persistencia entre sesiones mientras se simplifica el despliegue y testing.

React Router

* Administra la navegación dentro de la SPA, evitando recargas completas de página. Esto mejora la experiencia del usuario y organiza el código en rutas como /registro, /dashboard, /editar/:id, y /reporte.

Bootstrap

* Se incorpora para mantener un diseño visual consistente, responsive y profesional sin necesidad de construir todos los estilos desde cero.

**Justificación de la arquitectura:**

* Simplicidad y rapidez de desarrollo: Al evitar una API real y utilizar localStorage, se reduce el tiempo de desarrollo y la complejidad técnica, permitiendo concentrarse en la lógica de negocio y usabilidad.
* Escalabilidad futura: La arquitectura modular permite sustituir fácilmente localStorage por un backend real (API REST con Node.js, por ejemplo) sin necesidad de reescribir la interfaz de usuario.
* Separación de responsabilidades: El sistema divide claramente la lógica de interfaz (frontend) y la persistencia (aunque simulada), respetando el principio de separación de preocupaciones.
* Bajo acoplamiento y alta cohesión: Cada componente de React está pensado para cumplir una única función, facilitando el mantenimiento y la reutilización de código.

**Endpoints RESTful**

Recurso: /expenses

| **Método** | **Endpoint** | **Descripción** |
| --- | --- | --- |
| GET | /api/expenses | Obtiene todos los gastos registrados. |
| GET | /api/expenses/{id} | Obtiene un gasto específico por su ID. |
| POST | /api/expenses | Crea un nuevo gasto. |
| PUT | /api/expenses/{id} | Actualiza un gasto existente. |
| DELETE | /api/expenses/{id} | Elimina un gasto por su ID. |

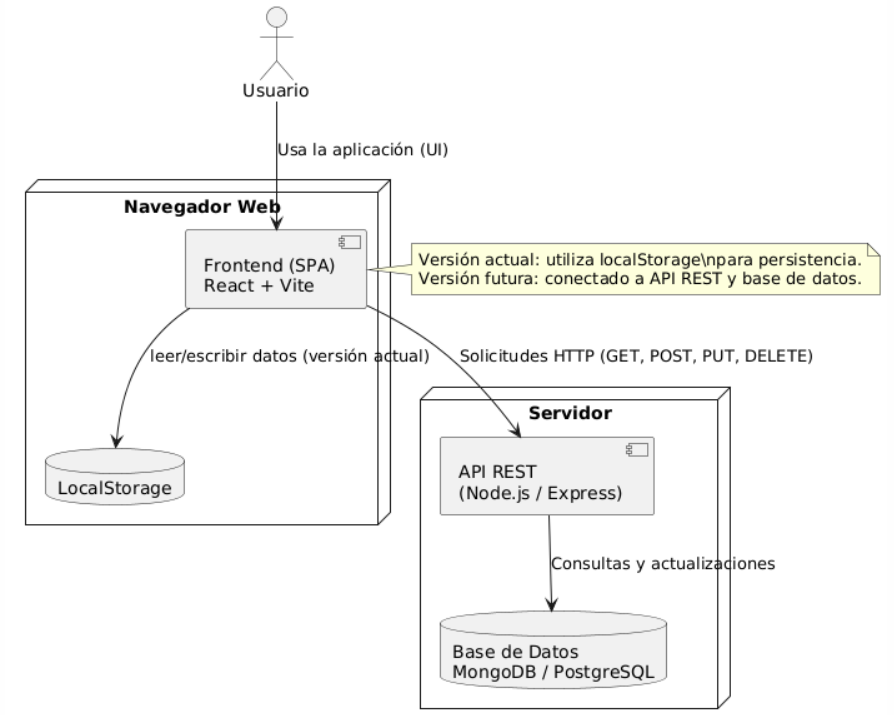
Filtros:

| **Método** | **Endpoint** | **Descripción** |
| --- | --- | --- |
| GET | /api/expenses?category=transporte | Filtra los gastos por categoría. |
| GET | /api/expenses?month=2025-05 | Filtra los gastos por mes (formato AAAA-MM). |

Reportes:

| **Método** | **Endpoint** | **Descripción** |
| --- | --- | --- |
| GET | /api/reports/monthly?month=2025-05 | Obtiene un reporte mensual agrupado por categoría y total. |

**DIAGRAMA DE DESPLIEGUE**



El diagrama de despliegue representa cómo se distribuyen los componentes del sistema de gestión de gastos en una arquitectura basada en cliente-servidor. Se identifican tres actores principales:

* Usuario: Interactúa con la aplicación a través de un navegador web.
* Cliente (Frontend): Implementado como una SPA (Single Page Application) en React + Vite, se ejecuta en el navegador del usuario.
* Servidor Backend: Es una proyección futura del sistema, donde las operaciones de lectura y escritura se realizarán a través de una API REST, alojada en un servidor, conectada a una base de datos como MongoDB o PostgreSQL.

En la versión actual, el sistema utiliza localStorage en el navegador para simular persistencia. El diagrama contempla tanto esta realidad como una posible evolución hacia una arquitectura con backend real, donde los datos se gestionan en una base de datos mediante llamadas HTTP.