## Segunda Iteracion

### Requerimientos:

Nuevos requerimientos funcionales:

* El usuario debe poder crearse una cuenta para el gestor de gastos con su email y crear una contraseña.
* EL usuario debe poder ingresar a su cuenta con el email y contraseña anteriormente creada.
* El sistema debe verificar que no existan dos cuentas con el mismo email.
* El sistema debe permitir al usuario cambiar la contraseña si se la olvida.

### Diagrama BPMN:

Diagrama, Esquemático

El contenido generado por IA puede ser incorrecto.

<https://modeler.camunda.io/share/24e32cb1-4b6f-4057-a2fb-7b920527c106>

### Diagrama casos de uso:

Diagrama

El contenido generado por IA puede ser incorrecto.

### Prototipo Iniciar sesion:

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

Interfaz de usuario gráfica, Aplicación, Sitio web

El contenido generado por IA puede ser incorrecto.

### Descripcion de casos de uso:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Caso de Uso**: Ver gráficos | | | | |
| **ID: 1** | | | **Fecha:** | |
| **Descripción:** Se le muestran los gráficos al usuario en la sección de gráficos | | | | |
| **Actores Principales**: Usuario | | **Actores Secundarios**: | | |
| **Observaciones:** | | | | |
| **Precondiciones**: - | | | | |
| **Post- Condiciones** | **Éxito:** Se muestran correctamente los gráficos al usuario. | | | |
| **Fracaso:** | | | |
| Flujo principal | | | | Flujo Alternativo |
| 1. **El caso de uso comienza cuando** el usuario ingresa al centro de control y presiona en el botón de gráficos. | | | |  |
| 1. El sistema obtiene con éxito de la base de datos los movimientos correspondientes al periodo corriente | | | | * 1. El sistema obtiene sin éxito los movimientos de la base de datos.   2. Se notifica al usuario   3. Fin CU. |
| 1. El sistema genera con éxito los gráficos y se los muestra al usuario. | | | | * 1. Ocurre un error y no se generan los gráficos.   2. Se notifica al usuario.   3. Fin CU. |
| 1. El usuario selecciona otro periodo para ver los gráficos | | | | * 1. El usuario no desea ver los gráficos de otro periodo   2. Fin CU. |
| 1. EL sistema obtiene con éxito de la base de datos los movimientos correspondientes al periodo seleccionado por el usuario. | | | | * 1. El sistema obtiene sin éxito los movimientos de la base de datos.   2. Se notifica al usuario   5.3 Fin CU. |
| 1. El sistema genera con éxito los gráficos y se los muestra al usuario. | | | | * 1. Ocurre un error y no se generan los gráficos.   2. Se notifica al usuario.   6.3Fin CU. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Caso de Uso**: Crear Presupuesto | | | | |
| **ID: 1** | | | **Fecha:** | |
| **Descripción:** El usuario crea un nuevo presupuesto para el periodo corriente | | | | |
| **Actores Principales**: Usuario | | **Actores Secundarios**: | | |
| **Observaciones:** | | | | |
| **Precondiciones**: - | | | | |
| **Post- Condiciones** | **Éxito:** Se crea con éxito un nuevo presupuesto | | | |
| **Fracaso:** | | | |
| Flujo principal | | | | Flujo Alternativo |
| 1. **El caso de uso comienza cuando** el usuario desea determinar un presupuesto y presiona en el centro de control crear presupuesto. | | | |  |
| 1. El usuario ingresa el monto del presupuesto. | | | |  |
| 1. Se verifica el monto y es correcto. | | | | * 1. Se verifica el monto ingresado y no es correcto.   2. Se notifica al usuario y se le solicita volver a ingresar el monto. |
| 1. Se carga el nuevo presupuesto con éxito en la base de datos y se notifica al usuario. | | | | * 1. Ocurre un error en la carga del nuevo presupuesto en la base de dato y se lo notifica al usuario.   2. Fin CU |
| 1. El sistema inicia el seguimiento del presupuesto. | | | |  |

### Diagramas de secuencia:

Diagrama

El contenido generado por IA puede ser incorrecto.

Tabla

El contenido generado por IA puede ser incorrecto.

Tabla, Calendario

El contenido generado por IA puede ser incorrecto.

Diagrama

El contenido generado por IA puede ser incorrecto.

Diagrama

El contenido generado por IA puede ser incorrecto.

## Arquitectura

Para este proyecto se decidió utilizar una arquitectura cliente/servidor porque permite separar la parte visual del sistema (la interfaz) de la lógica y del manejo de los datos. El cliente, desarrollado con React, se encarga de mostrar la información y recibir las acciones del usuario. Por otro lado, el backend se desarrollará usando Node.js y Express para que el servidor procese esas acciones, realice las validaciones necesarias y guarde los datos . Para almacenar la información se va a usar PostgreSQL que es un sistema de gestión de bases de datos relacional, la cual permite guardar, organizar y consultar datos. Esta arquitectura ayuda a que el sistema sea más organizado, fácil de mantener y escalar si en el futuro quiero agregar nuevas funciones.

### Diagrama de la arquitectura

Imagen que contiene Diagrama

El contenido generado por IA puede ser incorrecto.

**Componentes principales de la arquitectura:**

Cliente (Frontend):

* Desarrollado en React.js
* Se encarga de la presentación de la información y la interacción con el usuario.
* Consume la API REST del backend para obtener y enviar datos.

Servidor (Backend)

* Desarrollado con Node.js y Express.
* Maneja la lógica de negocio, incluyendo la autenticación de usuarios, el registro de movimientos, la gestión de categorías y la creación de presupuestos.
* Expone una API REST, compuesta por múltiples endpoints que permiten el acceso a la información del sistema.

Base de datos (persistencia de datos)

* Implementada en PostgreSQL.
* Estructurada con entidades como usuarios, movimientos, categorías y presupuestos.
* Relaciona los datos mediante claves foráneas, para garantizar la integridad de los datos.

Explicación del flujo de interacción:

1. El usuario accede a la interfaz web desde su navegador y realiza acciones como iniciar sesión, registrar gastos, definir presupuestos y visualizar gráficos.
2. El frontend envía una solicitud HTTP al backend, utilizando el método correspondiente. La solicitud incluye información necesaria en formato JSON.
3. El backend procesa la solicitud, validando los datos recibidos.
4. El backend consulta la base de datos en PostgreSQL para obtener o almacenar datos.
5. El backend genera una respuesta en formato JSON, conteniendo la información solicitada o el resultado de la operación, incluyendo los códigos de estado HTPP para indicar el resultado de la solicitud y lo envía al frontend.
6. El frontend recibe la respuesta y actualiza la interfaz.

### Listado de endpoints de la API REST:

|  |  |  |
| --- | --- | --- |
| Método | URL | Descripción |
| POST | |  | | --- | |  | | /api/auth/register | | Registrar un nuevo usuario en el sistema |
| POST | /api/auth/login | |  | | --- | | Iniciar sesión con email y contraseña | |  | |
| POST | /api/auth/reset-password | Solicitar el cambio de contraseña |
| GET | /api/movements | Obtener la lista de movimientos del usuario |
| POST | /api/movements | Registrar un nuevo movimiento |
| PUT | /api/movements/{id} | Editar un movimiento existente |
| DELETE | /api/movements/{id} | Eliminar un movimiento |
| GET | /api/budget | Obtener el presupuesto mensual del usuario |
| POST | /api/budget | Crear un nuevo presupuesto mensual |
| GET | /api/statistics/income-vs-expense | Obtener datos para grafico de ingresos y gastos |
| GET | /api/statistics/category-distribution | Obtener datos para grafico de distribución de gastos |
| GET | /api/statistics/monthly-balance | Obtener datos para grafico de evolución del balance mensual |
| GET | /api/categories | Obtener lista de categorías |
| POST | /api/categories | Crear nueva categoría |
| PUT | /api/categories/{id} | Editar una categoría existente |
| DELETE | /api/categories/{id} | Eliminar una categoría |

Requerimientos realizados en esta implementación:

RF7. El sistema muestra una lista de todos los movimientos en orden cronológico del mostrando el más reciente primero.

RF9: El sistema debe mostrar en el centro de control, para un mes determinado, el total de ingresos, gastos y el balance.