|  |
| --- |
| **Práctica microservicios** |
|  |
| **Spring Boot** |
|  |

# Microservicio de Gestión de Usuarios y seguridad (User Service). (Yuri)

**Responsabilidad:** Manejar el registro, autenticación y roles de los usuarios del sistema (administradores, usuarios regulares, etc.).

**Base de datos:** Postgres SQL

**Funcionalidades:**

**Registro y Autenticación de Usuarios**

* **Descripción**: Implementar endpoints para que los nuevos usuarios se registren en la plataforma, proporcionando su email, contraseña, y otros datos relevantes. La autenticación debe permitir que los usuarios ingresen a la aplicación validando sus credenciales (email y contraseña) y generando un token de acceso (como JWT).
* **Tareas**:
  + Crear un endpoint POST /users/register que valide los datos de registro y almacene el nuevo usuario en la base de datos.
  + Crear un endpoint POST /users/login para validar credenciales y generar un token JWT para autenticación.
  + Asegurar que las contraseñas se guarden cifradas (ej. bcrypt).

**Recuperación de contraseñas.**

* **Descripción**: Proporcionar un mecanismo seguro para que los usuarios puedan recuperar el acceso a su cuenta en caso de olvidar su contraseña. Esto incluye generar un token temporal para la recuperación de contraseñas y permitir al usuario establecer una nueva contraseña.
* **Tareas**:
  + Crear un endpoint POST /users/forgot-password que genere un token de recuperación y envíe un email con un enlace para restablecer la contraseña.
  + Implementar un endpoint POST /users/reset-password que permita cambiar la contraseña usando el token de recuperación.

**Perfil de usuario.**

* + Crear un endpoint GET /users/{id}/profile que devuelva la información del perfil del usuario autenticado.
  + Implementar un endpoint PUT /users/{id}/profile que permita actualizar los datos del perfil, asegurando la validación de campos.

**Ejemplo de API**:

* POST /users/register
* POST /users/login
* GET /users/{id}/profile

# 2. Microservicio de Categorías (Category Service). (Raul) Responsabilidad: Gestión de categorías de ingresos y gastos (e.g., comida, transporte, entretenimiento).

**Base de datos:** MongoDB

**Funcionalidades**:

**Crear, Editar, Eliminar y Listar Categorías**

* **Descripción**: Implementar un sistema para la gestión de categorías. Los usuarios deben poder crear nuevas categorías, editar las existentes, eliminarlas (si no están asociadas a transacciones) y listarlas para poder asociarlas a sus transacciones.
* **Tareas**:
  + Crear un endpoint POST /categories que permita a los usuarios crear nuevas categorías, validando que el nombre de la categoría no esté duplicado.
  + Implementar un endpoint PUT /categories/{id} que permita a los usuarios editar el nombre de la categoría.
  + Validar que las categorías asociadas existan y pertenezcan al usuario autenticado.
  + Crear un endpoint DELETE /categories/{id} que elimine una categoría, asegurándose de que no esté asociada a ninguna transacción o implementando una política de eliminación suave (soft delete).
  + Implementar un endpoint GET /categories que liste todas las categorías creadas por el usuario, con opciones de paginación y filtrado.

**Asociar Categorías a Transacciones**

* **Descripción**: Implementar la funcionalidad para que los usuarios puedan asociar una o más categorías a cada transacción (ingresos o gastos). Las categorías ayudarán a clasificar las transacciones y facilitarán la creación de reportes y resúmenes financieros.
* **Tareas**:
  + Crear un endpoint POST /transactions/{id}/categories que permita a los usuarios asociar una categoría existente a una transacción.
  + Asegurar que las categorías puedan ser eliminadas o reasignadas si una transacción es modificada.

**Ejemplo de API**:

* POST /categories
* GET /categories
* PUT /categories/{id}
* DELETE /categories/{id}

1. **Microservicio de Etiquetas (Tag Service)**

**Responsabilidad**: Permitir a los usuarios etiquetar sus transacciones para mayor organización y flexibilidad (e.g., viajes, trabajo).

**Base de datos:** MongoDB

**Funcionalidades:**

**Crear, Editar, Eliminar y Listar Etiquetas**

* **Descripción**: Implementar un sistema para la gestión de etiquetas. Las etiquetas ayudarán a los usuarios a categorizar y organizar mejor sus transacciones. Los usuarios deben poder crear nuevas etiquetas, editarlas, eliminarlas y listarlas.
* **Tareas**:
  + Crear un endpoint POST /tags para permitir a los usuarios crear nuevas etiquetas, asegurándose de que el nombre de la etiqueta sea único para cada usuario.
  + Validar que las etiquetas asociadas existan y pertenezcan al usuario autenticado.
  + Implementar un endpoint PUT /tags/{id} para editar el nombre de una etiqueta existente.
  + Crear un endpoint DELETE /tags/{id} que elimine una etiqueta, verificando que no esté asociada a ninguna transacción activa. También puedes implementar una eliminación suave (soft delete).
  + Implementar un endpoint GET /tags para listar todas las etiquetas creadas por el usuario, con opciones de paginación y filtrado.

**Asociar Etiquetas a Transacciones**

* **Descripción**: Permitir que los usuarios puedan asociar una o varias etiquetas a cada transacción. Las etiquetas permitirán una categorización flexible y no estructurada, para complementar las categorías. Esto facilitará el análisis y los resúmenes de las transacciones según las preferencias del usuario.
* **Tareas**:
  + Crear un endpoint POST /transactions/{id}/tags que permita a los usuarios asociar una o más etiquetas a una transacción específica.
  + Permitir la eliminación o reasignación de etiquetas si una transacción es editada o eliminada.

**Ejemplo de API:**

* POST /tags
* GET /tags
* PUT /tags/{id}
* DELETE /tags/{id}

1. **Microservicio de Transacciones (Transaction Service) (Mancha)**

**Responsabilidad:** Manejar la creación, consulta y modificación de ingresos y gastos.

**Base de datos:** MySQL

**Funcionalidades:**

**Crear Ingresos y Gastos**

* **Descripción**: Implementar la funcionalidad para que los usuarios puedan registrar sus ingresos y gastos en el sistema, especificando detalles como la categoría, etiquetas, monto, fecha, y una descripción opcional.
* **Tareas**:
  + Crear un endpoint POST /transactions para permitir a los usuarios crear una nueva transacción. El cuerpo de la solicitud debe incluir el tipo de transacción (ingreso o gasto), categoría, etiquetas, monto, fecha, y descripción.
  + Validar que las categorías y etiquetas asociadas existan y pertenezcan al usuario autenticado.
  + Implementar validaciones para garantizar que los montos sean válidos (positivos para ingresos y negativos para gastos).
  + Permitir la inserción opcional de múltiples etiquetas en el mismo ingreso o gasto, para mejorar la organización de las transacciones.

**Modificar o Eliminar Transacciones**

* **Descripción**: Permitir que los usuarios puedan modificar o eliminar sus transacciones en caso de errores o cambios. Esto incluye la edición de detalles como categoría, etiquetas, monto, y fecha.
* **Tareas**:
  + Crear un endpoint PUT /transactions/{id} para permitir la edición de una transacción existente. Los usuarios podrán modificar la categoría, etiquetas, monto, fecha, y descripción de la transacción.
  + Implementar un endpoint DELETE /transactions/{id} que elimine una transacción, asegurando que solo el propietario de la transacción pueda eliminarla.
  + Validar que, al modificar una transacción, las nuevas categorías o etiquetas existan y sean válidas.

**Filtrar Transacciones por Categoría, Etiqueta o Fecha**

* **Descripción**: Implementar filtros avanzados para que los usuarios puedan buscar y visualizar transacciones según sus necesidades. Los usuarios deben poder filtrar por categoría, etiqueta o un rango de fechas.
* **Tareas**:
  + Crear un endpoint GET /transactions que permita filtrar las transacciones con parámetros opcionales, como category, tag, y dateRange (rango de fechas).
  + Implementar paginación y ordenamiento en los resultados de las transacciones para optimizar la búsqueda y mejorar la experiencia de usuario.
  + Asegurar que los filtros puedan combinarse entre sí, por ejemplo, permitiendo filtrar por una categoría específica dentro de un rango de fechas, o filtrar por etiquetas dentro de una categoría.
  + Garantizar que los usuarios solo puedan ver sus propias transacciones, respetando los permisos adecuados.

**Ejemplo de API:**

* POST /transactions
* GET /transactions
* GET /transactions/{id}
* PUT /transactions/{id}
* DELETE /transactions/{id}

1. **Microservicio de Resúmenes Financieros (Summary Service) (Carlos)**

**Responsabilidad:** Generar resúmenes financieros basados en las transacciones (e.g., resumen mensual, por categoría, etc.).

**Base de datos:** MongoDB

**Funcionalidades:**

**Resumen de Ingresos y Gastos por Mes, Trimestre, Año, etc.**

* **Descripción**: Permitir a los usuarios ver un resumen de sus ingresos y gastos agrupados por diferentes periodos de tiempo, como mes, trimestre o año. Estos resúmenes proporcionan una visión clara de las finanzas en intervalos específicos.
* **Tareas**:
  + Crear un endpoint GET /transactions/summary que permita al usuario obtener resúmenes financieros agregados por periodos. El usuario podrá pasar parámetros como period (mes, trimestre, año) para definir el rango temporal.
  + Incluir detalles como ingresos totales, gastos totales, y balance neto (ingresos menos gastos) en cada periodo.
  + Ofrecer la opción de seleccionar diferentes periodos personalizados para que el usuario pueda comparar, por ejemplo, los primeros tres meses del año con el segundo trimestre.
  + Implementar gráficos simples (barras o líneas) para mostrar visualmente los cambios financieros a lo largo del tiempo.

**Resumen por Categoría y Etiqueta**

* **Descripción**: Proporcionar resúmenes de los ingresos y gastos organizados por categoría y etiqueta, lo que permite a los usuarios ver cómo se distribuyen sus transacciones. Este resumen ayuda a identificar en qué áreas se está gastando más y dónde se reciben más ingresos.
* **Tareas**:
  + Crear un endpoint GET /transactions/summary-by-category que devuelva un desglose de los ingresos y gastos por cada categoría. Incluir tanto el monto total como el porcentaje del total de gastos o ingresos que representa cada categoría.
  + Desarrollar un endpoint similar GET /transactions/summary-by-tag para permitir a los usuarios ver un desglose de transacciones por etiquetas.
  + Incluir la posibilidad de combinar el resumen de categoría y etiqueta con filtros por periodos (mes, trimestre, año).
  + Implementar gráficos circulares (o "pie charts") para visualizar la proporción de gastos o ingresos en cada categoría o etiqueta.

**Reportes Financieros Exportables (PDF, Excel)**

* **Descripción**: Dar a los usuarios la posibilidad de generar reportes financieros completos en formatos exportables, como PDF y Excel, para compartir o archivar sus datos financieros.
* **Tareas**:
  + Crear un endpoint GET /transactions/export que permita al usuario generar un reporte financiero en formato PDF o Excel. El reporte debe incluir resúmenes por periodo, categoría, y etiqueta, así como los gráficos que correspondan.
  + Utilizar bibliotecas como iText (para PDF) y Apache POI (para Excel) para la generación de estos documentos.
  + Ofrecer la opción de personalizar los reportes, permitiendo al usuario seleccionar el periodo y las categorías o etiquetas que desea incluir.
  + Incluir un resumen general de ingresos, gastos, y balance, junto con las listas de transacciones individuales filtradas en el periodo seleccionado.
  + Implementar una opción de exportar automáticamente los reportes a un almacenamiento en la nube (por ejemplo, Google Drive o Dropbox) si el usuario lo desea.

**Ejemplo de API:**

* GET /summaries/monthly
* GET /summaries/yearly
* GET /summaries/category/{categoryId}

1. **Microservicio de Notificaciones (Notification Service)**

**Responsabilidad:** Notificar a los usuarios sobre ciertos eventos, como cuando una transacción excede un presupuesto o un resumen está disponible.

**Base de datos:** MongoDB

**Funcionalidades:**

**Enviar Notificaciones por Correo Electrónico, SMS o Push**

* **Descripción**: Esta funcionalidad permite a la aplicación enviar notificaciones a los usuarios en diferentes formatos (correo electrónico, SMS o notificaciones push) según eventos relevantes, como la confirmación de una transacción, recordatorios de facturas, alertas de presupuesto, etc.
* **Tareas**:
  + Implementar un servicio que permita enviar notificaciones en tres formatos distintos:
    - **Correo electrónico**: Utilizar servicios como **JavaMail** o **SendGrid**.
    - **SMS**: Utilizar servicios de terceros como **Twilio**.
    - **Push notifications**: Para aplicaciones móviles o web, usar servicios como **Firebase Cloud Messaging (FCM)** o **OneSignal**.
  + Crear un endpoint POST /notifications/send que reciba los detalles de la notificación (tipo de notificación, destinatario, mensaje) y lo procese.
  + Incluir lógica para manejar plantillas de notificación personalizables para cada tipo (correo, SMS, push), lo que facilita el envío de mensajes consistentes y fáciles de editar.
  + Añadir soporte para el envío de notificaciones automáticas basadas en eventos de usuario, por ejemplo, enviar un correo cuando se realice una nueva transacción.

**Obtener Historial de Notificaciones de un Usuario**

* **Descripción**: Permitir a los usuarios acceder a un registro de todas las notificaciones que han recibido en el pasado, brindando transparencia y facilitando la búsqueda de notificaciones antiguas.
* **Tareas**:
  + Crear un endpoint GET /notifications/history/{userId} que devuelva el historial de notificaciones de un usuario en particular.
  + Guardar las notificaciones enviadas en una base de datos para que se puedan consultar posteriormente, incluyendo detalles como la fecha y hora del envío, el tipo de notificación y el contenido.
  + Implementar filtros para que el usuario pueda buscar notificaciones por rango de fechas, tipo (correo, SMS, push), o categoría (transacciones, alertas, recordatorios).
  + Asegurar que el historial solo es accesible para el propio usuario o administradores con los permisos adecuados.

**Gestión de Preferencias de Notificación de Usuarios**

* **Descripción**: Permitir a los usuarios gestionar sus preferencias de notificación, como decidir qué tipo de notificaciones quieren recibir y en qué formato (correo, SMS, push).
* **Tareas**:
  + Crear un endpoint PUT /notifications/preferences/{userId} para que el usuario pueda actualizar sus preferencias de notificación.
  + Desarrollar un modelo de **preferencias de usuario** que incluya opciones como:
    - Qué tipo de notificaciones desea recibir (transacciones, recordatorios, alertas presupuestarias).
    - En qué formato prefiere recibirlas (correo, SMS, push).
    - Frecuencia de las notificaciones (inmediato, diario, semanal).
  + Implementar un mecanismo que respete estas preferencias cuando se envíen las notificaciones, evitando que los usuarios reciban notificaciones que no desean.
  + Ofrecer una interfaz amigable (en web o móvil) para que los usuarios puedan gestionar estas preferencias fácilmente.

**Ejemplo de API:**

* POST /notifications/send – Enviar una notificación.
* GET /notifications/{userId} – Obtener historial de notificaciones de un usuario.
* POST /notifications/settings – Configurar preferencias de notificación.

1. **Microservicio de Presupuestos (Budget Service)( masó)**

**Responsabilidad:** Permitir a los usuarios establecer y gestionar presupuestos para diferentes categorías o periodos.

**Base de datos:** PostgresSQL

**Funcionalidades:**

**Crear, Editar, Eliminar y Consultar Presupuestos**

* **Descripción**: Permitir a los usuarios establecer presupuestos personalizados, actualizarlos según sea necesario, eliminarlos cuando ya no sean útiles, y consultarlos en cualquier momento para ver el progreso.
* **Tareas**:
  + Implementar un endpoint POST /budgets para crear nuevos presupuestos, donde los usuarios puedan definir:
    - **Monto total** del presupuesto.
    - **Periodo de tiempo** (mensual, trimestral, anual).
  + Desarrollar un endpoint PUT /budgets/{id} para editar un presupuesto existente, permitiendo ajustes en el monto o el periodo de tiempo.
  + Crear un endpoint DELETE /budgets/{id} para eliminar un presupuesto cuando ya no sea necesario.
  + Implementar un endpoint GET /budgets/{id} para que los usuarios puedan consultar un presupuesto en detalle, viendo el progreso y el monto restante.
  + Añadir validaciones para asegurar que los presupuestos no pueden tener montos negativos y que el periodo de tiempo sea válido.
  + Posibilidad de asignar un nombre o descripción para identificar fácilmente el presupuesto en la lista general.

**Asociar Presupuestos a Categorías y Monitorizar Gastos**

* **Descripción**: Permitir que los presupuestos se asocien a categorías específicas de gasto, lo que facilita a los usuarios monitorizar cómo se alinean sus gastos con los límites que establecieron.
* **Tareas**:
  + Implementar la lógica para que un presupuesto pueda estar asociado a una o varias **categorías de gasto**. Esto permitirá al usuario tener control específico sobre ciertos tipos de gastos (e.g., alimentación, ocio, transporte).
  + Crear un mecanismo para **monitorizar automáticamente los gastos** de esas categorías, calculando el total gastado en relación al presupuesto asignado.
  + Mostrar alertas o notificaciones automáticas cuando los gastos en una categoría excedan un porcentaje predefinido del presupuesto (e.g., 80%).
  + Desarrollar un dashboard o vista que muestre visualmente el progreso del presupuesto, indicando cuánto se ha gastado, cuánto queda disponible, y cuánto falta para exceder el presupuesto.
  + Implementar un sistema de reportes que permita a los usuarios ver cómo sus gastos se distribuyen dentro de cada presupuesto y qué categorías están afectando más su balance financiero.

**Ejemplo de API:**

* POST /budgets
* GET /budgets
* PUT /budgets/{id}
* DELETE /budgets/{id}