UBudget - Documentación de la Base de Datos

1. Principios de Diseño

Se tienen en cuenta cuatro principios fundamentales:

1. **Separación de Responsabilidades** – Cada tabla representa una sola repsonsabilidad dentro de las funcionalidades de la aplicación.
2. **Normalización** – La base de datos debe estar normalizada a la forma 3FN.
3. **Extensibilidad** – Se soportan subsistemas opcionales sin forzarlos en el núcleo del esquema.
4. **Auditabilidad y Transparencia** – Todas las operaciones financieras son rastreables mediante historiales de transacciones, snapshots, y registros de procedimientos.

---

2. Entidades

2.1 `auth_user`

Propósito:
Almacena los identificadores de la aplicación. Es la entidad raiz que une datos de todas las funcionalidades.

Atributo	Explicación
`email` (único)	Es el método estándar de autentificación
`password_hash`	Almacenamiento seguro de contraseñas con hashing (responsabilidad de la capa de aplicación)
`role`	Permite diferenciar privilegios
`metadata` (JSON)	Extensión de almacenamiento que no requiere de modificaciones en la base de datos (por si acaso)

2.2 `account`

Propósito:
Representar un contenedor financiero (cuenta bancaria, cartera, tarjeta, etc.).

Atributo	Explicación
`account_type`	Posibles ventajas para el desarrollo de la interfaz gráfica
`currency`	Almacenar la divisa explícitamente previene posibles errores de cálculo
`cached_balance`	Optimización opcional de rendimiento. El balance en sí es calculado mediante una vista

Relaciones:
Cada `account` le pertenece a un `auth_user`.

2.3 `category`

Propósito:
Clasificar las transacciones para evitar hard-coding de strings en las tablas.

Atributo	Explicación
`type`	Distingue la dirección financiera (ingreso o gasto)
`parent_category_id`	Agrupa jerárquicamente (por ejemplo, "Cena" y "Mercado" serían subcategorías de "Comida")

2.4 `transaction_tbl`

Propósito:
Registra todos los movimientos financieros.

Atributo	Explicación
`related_account_id`	Soporta el modelamiento de transferencia automática
`metadata`	Extensión de almacenamiento que no requiere de modificaciones en la base de datos (por si acaso)
`is_reconciled`	Permite revisar flujos financieros manual o automáticamente

Relaciones:
Con `account` y `category`.

2.5 `obligation`

Propósito:
Registra pagos pendientes y recurrentes (deudas, suscripciones, facturas futuras).

Atributo	Explicación
`amount_remaining`	Útil para la amortización o registros adicionales
`frequency`	Permite automatizar recordatorios

2.6 `budget`

Propósito:
Define restricciones de gastos por categoría.

Atributo	Explicación
`period`	Define explicitamente el periodo de alcance del presupuesto
`start_date` / `end_date`	Permite la evolución historica del presupuesto sin eliminar datos

---

2.7 `balance_history`

Propósito:
Auditoría basada en "Snapshots" para reportes financieros.

Atributo	Explicación
`payload` JSON	Almacena información para cada cuenta que permita reproducir o simular actividad pasada

---

2.8 `reminder`

Propósito:
Agenda acciones futuras (notificaciones, pagos).

Atributo	Explicación
`channel_set`	Permite alertas por varios medios sin proliferación de tablas
`obligation_id` / `account_id`	Relacionar alertas a entidades financieras

2.9 `notification_log`

Propósito:
Registra las notificaciones que el sistema lanza.

Atributo	Explicación
`status`	Permite implementar reenvíos de mensajes si no se entregan correctamente
`payload`	Contexto completo del mensaje en formato JSON

2.10 `noticia`

Propósito:
Almacena la fuente de la información de noticias y finanzas.

Atributo	Explicación
`category`	Permite filtrado y personalización
`raw_payload`	Retiene la respuesta de la API para poderla reutilizar si se requiere

---

2.11 `preferencia_noticia`

Propósito:
Permite la personalización de noticias por usuario.

Atributo	Explicación
`priority`	Permite jerarquizar, quizá también implementar lógica de las recomendaciones

---

2.12 `simulacion_financiera`

Propósito:
Registra el uso de calculadoras.

Atributo	Explicación
`parameters` / `result` as JSON	Historial de entrada y salida que es totalmente rastreable sin complicar el esquema de la base de datos

---

3. Vistas y Procedimientos

3.1 `v_account_balance`

Propósito:
Ofrece un balance en tiempo real para cada cuenta mediante la agregación de transacciones, evitando estados duplicados sin sacrificar rendimiento en las consultas.

3.2 `sp_snapshot_user_balance`

Propósito:
Automatiza la inserción a `balance_history`. Produce trazos de historial para analíticas, gráficos de tendencia, etc.

3.3 `trg_transaction_after_insert`

Propósito:
Permite alertas y notificaciónes automáticas directamente desde SQL.
