# Actividad 6. Proyectos con React Native

Aplicacion de Finanzas Personales y Ahorro

Descripcion:

En esta aplicación de finanzas personales, el usuario se registra proporcionando sus datos básicos: nombre, correo electrónico único, contraseña encriptada, su moneda preferida y la fecha de registro.

Una vez registrado, el usuario puede crear categorías para clasificar sus ingresos y gastos. Cada categoría pertenece únicamente a ese usuario y puede ser de tipo ingreso o gasto.

* Por ejemplo, si el usuario está planeando unas vacaciones, podría crear categorías como “Transporte”, “Hospedaje”, “Comida”, “Entretenimiento” y “Compras”.

El usuario también puede establecer presupuestos, plan financiero para un periodo específico (por ejemplo, una semana, un mes o las fechas exactas de las vacaciones) y define un monto total disponible.

* Ejemplo, el usuario podría crear un presupuesto para las vacaciones con un monto total de $5,000 y con fechas que cubran todo el viaje.

Dentro de ese presupuesto general, el usuario puede repartir el dinero en límites por categoría. Esto se hace en la relación entre presupuestos y categorías.

* Por ejemplo:

Transporte: $1,000

Hospedaje: $2,000

Comida: $1,000

Entretenimiento: $500

Compras: $500

* El usuario tiene $5,000 para todo el viaje, cada categoría tendrá un límite máximo que no debería sobrepasar.

Cuando el usuario realiza un gasto o ingreso, lo registra como una transacción. Cada transacción está asociada a una categoría y a un usuario.

* Por ejemplo, si paga $800 de hotel, registra una transacción de tipo gasto, monto $800, categoría “Hospedaje”. El sistema automáticamente descontará ese gasto del límite de la categoría “Hospedaje” y del total del presupuesto de vacaciones.

Si el gasto en alguna categoría se acerca a su límite (por ejemplo, alcanza el 80%), la aplicación puede enviar una alerta para que el usuario tenga cuidado. Si se supera el límite, también puede avisar o marcarlo como excedido.

Además de manejar presupuestos, el usuario puede definir metas de ahorro. Por ejemplo, podría crear una meta llamada “Ahorro para vacaciones” con un objetivo de $5,000 y un plazo de seis meses. Cada vez que aparta dinero, registra un aporte que se suma al monto actual de la meta, hasta alcanzar el objetivo.

## **usuarios**

📌 **Descripción:** Guarda la información básica de cada usuario que usa la app.  
📂 **Campos:**

* id (INT, PK, AI) → Identificador único del usuario.
* nombre (VARCHAR) → Nombre completo del usuario.
* email (VARCHAR, único) → Correo electrónico único para iniciar sesión.
* password (VARCHAR) → Contraseña encriptada.
* moneda\_preferida (VARCHAR) → Moneda que usa el usuario (MXN, USD, EUR, etc.).
* fecha\_registro (DATETIME) → Fecha y hora en que se creó la cuenta.

🎯 **Propósito:** Identificar y autenticar usuarios, y almacenar su configuración personal.

## **categorias**

📌 **Descripción:** Clasificación personalizada que cada usuario crea para organizar sus ingresos y gastos.  
📂 **Campos:**

* id (INT, PK, AI) → Identificador único de la categoría.
* usuario\_id (INT, FK → usuarios.id) → Usuario al que pertenece la categoría.
* nombre (VARCHAR) → Ejemplo: Alimentación, Transporte, Hospedaje, Comida, Entretenimiento.
* tipo (ENUM: ingreso, gasto) → Define si la categoría es para ingresos o gastos.
* color (VARCHAR) → Código HEX para identificar la categoría visualmente en reportes.

🎯 **Propósito:** Agrupar y filtrar transacciones para reportes y presupuestos.

## presupuestos ()

📌 **Descripción:** Plan financiero para un periodo específico (mensual, semanal o personalizado) con **un monto total** que se **divide entre categorías**.  
📂 **Campos (actualizado):**

* id (INT, PK, AI)
* usuario\_id (INT, FK → usuarios.id)
* nombre (VARCHAR) — ej. “Vacaciones 2025” o “Mes actual”
* monto\_total (DECIMAL) — **bolsa total** del periodo
* periodo (ENUM: mensual, semanal, personalizado)
* fecha\_inicio (DATE)
* fecha\_fin (DATE)
* estado (ENUM: activo, cerrado, cancelado) — opcional, para cierres de periodo

🧩 **Reglas clave:**

* **Un presupuesto por periodo** (o por ventana de fechas) por usuario y “nombre” (sugerido):
  + Índice único sugerido: UNIQUE (usuario\_id, fecha\_inicio, fecha\_fin, nombre)
* **La suma de los límites por categoría ≤ monto\_total** (ver tabla intermedia).
* Si no asignas límites por categoría, el presupuesto actúa como **bolsa global**.

🎯 **Propósito:** Controlar el gasto del periodo y enviar alertas si se alcanzan o superan límites globales y por categoría.

## presupuesto\_categorias (actualizado)

📌 **Descripción:** **Distribución interna** del presupuesto por categoría (el “sobre” de cada categoría).  
📂 **Campos (actualizado):**

* id (INT, PK, AI)
* presupuesto\_id (INT, FK → presupuestos.id)
* categoria\_id (INT, FK → categorias.id)
* monto\_limite (DECIMAL) — **tope** para esa categoría dentro del presupuesto
* monto\_gastado (DECIMAL, default 0) — opcional, para acelerar reportes
* alerta\_80 (BOOLEAN, default FALSE) — opcional, marcado cuando ≥80%

🔗 **Restricciones sugeridas:**

* UNIQUE (presupuesto\_id, categoria\_id) — una categoría no se duplica dentro del mismo presupuesto.
* **Check lógico** (a nivel app/trigger):  
  SUM(monto\_limite) POR presupuesto\_id ≤ presupuestos.monto\_total

📣 **Alertas (lógica de app):**

* Al registrar gasto:
  + Actualiza monto\_gastado de la categoría y el acumulado global del presupuesto.
  + Si monto\_gastado >= 0.8 \* monto\_limite ⇒ alerta “80%”.
  + Si monto\_gastado > monto\_limite ⇒ marcar “excedido”.

## **transacciones**

📌 **Descripción:** Registra cada ingreso o gasto que el usuario realiza, afectando el presupuesto y categoría correspondiente.  
📂 **Campos:**

* id (INT, PK, AI) → Identificador único de la transacción.
* usuario\_id (INT, FK → usuarios.id) → Usuario que realizó la transacción.
* tipo (ENUM: ingreso, gasto) → Tipo de movimiento.
* monto (DECIMAL) → Cantidad de dinero.
* fecha (DATETIME) → Fecha y hora del movimiento.
* categoria\_id (INT, FK → categorias.id) → Categoría a la que pertenece la transacción.
* presupuesto\_id (INT, FK → presupuestos.id, opcional) → Presupuesto asociado (si aplica).
* nota (TEXT) → Descripción opcional (ej. "Hotel 2 noches").

🎯 **Propósito:** Mantener un historial de movimientos para reportes, balances y control de presupuesto.

## **metas\_ahorro**

📌 **Descripción:** Objetivos financieros que el usuario quiere alcanzar en un plazo determinado.  
📂 **Campos:**

* id (INT, PK, AI) → Identificador único de la meta.
* usuario\_id (INT, FK → usuarios.id) → Usuario que creó la meta.
* nombre\_meta (VARCHAR) → Ejemplo: "Ahorro para vacaciones".
* monto\_objetivo (DECIMAL) → Monto que se quiere alcanzar.
* monto\_actual (DECIMAL) → Monto acumulado hasta ahora.
* fecha\_meta (DATE) → Fecha límite para alcanzar la meta.
* estado (ENUM: en\_progreso, cumplida, cancelada) → Estado actual.

🎯 **Propósito:** Medir el progreso hacia un objetivo financiero y registrar aportes.