Guía para la Capa de Datos: Room + Repository + Hilt

¿Qué implementé y cómo funciona?

Esta estructura sigue las mejores prácticas modernas de Android (2025): Clean Architecture + MVVM + Inyección con Hilt.

1. Estructura General

- **Base de Datos Room**
  - Archivo: `AppDatabase.kt`
  - Patrón: Singleton
  - Expone el único acceso a la base local de gastos (tabla `expenses`).
- **Entidad de Gasto**
  - Archivo: `ExpenseEntity.kt`
  - Define los campos que se guardan: id, title, amount, category, dateTimestamp, isSynced.
- **DAO (Data Access Object)**

- Archivo: `ExpenseDao.kt`
- Métodos para insertar, borrar, actualizar y obtener gastos, todo reactivo usando `Flow`.

- **Repositorio**
  - Archivo: `ExpenseRepository.kt`
  - Expone funciones limpias para interactuar con los datos. Nunca accedas al DAO DIRECTAMENTE desde UI o ViewModel; usa el repositorio.
- **Inyección de Dependencias (Hilt)**
  - Carpeta: `di/DatabaseModule.kt`
  - Permite que todo (base de datos, DAO, repositorio) se cree automáticamente sin hacer `new` manual.

¿Cómo debe llamarse el código desde otro módulo?

– Para obtener la lista de gastos o el total gastado:

```
// En cualquier ViewModel que tenga inyectado el repositorio:
@HiltViewModel
class ExpenseViewModel @Inject constructor(
    private val repository: ExpenseRepository
) : ViewModel() {
    val allExpenses = repository.allExpenses
    val totalSpending = repository.totalSpending
}
```

– Para agregar o borrar gastos:

```
// Desde ViewModel o una clase con acceso al repositorio:
viewModelScope.launch {
    repository.addExpense(
        ExpenseEntity(
            title = "Cena",
            amount = 50.0,
            category = "Comida",
            date = System.currentTimeMillis()
        )
    )
}

viewModelScope.launch {
    repository.deleteExpense(expense)
}
```

Nota técnica importante

- **Nunca Crear Instancias Manualmente**
  Siempre usa la inyección de dependencias de Hilt. Por ejemplo, si necesitas el repositorio en tu UI o ViewModel, declare `@Inject` en el constructor o usa `by viewModels()` si estás en una Activity/Fragment con Hilt.
- **El código está listo para migrar a nube (Fase 2)**
  Solo hay que cambiar la implementación interna del Repository. La UI no cambiará nada.

Ejemplo de cómo usar el Repository en un ViewModel (para el desarrollador UI)

```
@HiltViewModel
class ExpenseViewModel @Inject constructor(
    private val repository: ExpenseRepository
) : ViewModel() {
    val allExpenses = repository.allExpenses
    val totalSpending = repository.totalSpending
}
```

Por favor, **no accedas a la base de datos ni al DAO directo desde la UI**. Usa siempre el repositorio y la inyección vía Hilt.