

DOCUMENTACIÓN

- Descripción general
- Guías**
- Referencia
- Ejemplos
- Diseño y calidad

- Descripción general
- Descripción general del almacenamiento
- Cómo guardar contenido en el almacenamiento específico de la app
- ▶ Cómo guardar contenido en el almacenamiento compartido
- Cómo administrar todos los archivos de un dispositivo de almacenamiento
- Cómo guardar datos de pares clave-valor
- ▼ Cómo guardar contenido en una base de datos local
- Descripción general

Cómo definir datos mediante entidades

Cómo acceder a datos mediante DAO

Cómo definir relaciones entre objetos

Escribe consultas DAO asíncronas

Cómo implementar vistas en una base de datos

Cómo autocompletar el contenido de tu base de datos

Cómo migrar tu base de datos
- ▶ Cómo guardar contenido en el almacenamiento compartido
- Cómo administrar todos los archivos de un dispositivo de almacenamiento
- Cómo guardar datos de pares clave-valor
- ▼ Cómo guardar contenido en una base de datos local
- Descripción general

Cómo definir datos mediante entidades

Cómo acceder a datos mediante DAO

Cómo definir relaciones entre objetos

Escribe consultas DAO asíncronas

Cómo implementar vistas en una base de datos

Cómo autocompletar el contenido de tu base de datos

Cómo migrar tu base de datos

Cómo probar y depurar tu base de datos

Cómo hacer referencia a datos complejos
- ▶ Cómo guardar contenido en el almacenamiento compartido
- Cómo administrar todos los archivos de un dispositivo de almacenamiento
- Cómo guardar datos de pares clave-valor
- ▼ Cómo guardar contenido en una base de datos local
- Descripción general

Cómo definir datos mediante entidades

Cómo acceder a datos mediante DAO

Cómo definir relaciones entre objetos

Escribe consultas DAO asíncronas

Cómo implementar vistas en una base de datos

Cómo autocompletar el contenido de tu base de datos

Cómo migrar tu base de datos


Cómo probar y depurar tu base de datos

Cómo hacer referencia a datos complejos
- ▶ Cómo guardar contenido en el almacenamiento compartido
- Cómo administrar todos los archivos de un dispositivo de almacenamiento
- Cómo guardar datos de pares

Desarrolladores de Android > Documentos > Guías

¿Te resultó útil?  

Cómo guardar contenido en una base de datos local con Room

 Parte de **Android Jetpack** 

Las apps que controlan grandes cantidades de datos estructurados pueden beneficiarse con la posibilidad de conservar esos datos localmente. El caso de uso más común consiste en almacenar en caché datos relevantes para que el dispositivo no pueda acceder a la red, de modo que el usuario pueda explorar ese contenido mientras está sin conexión.

La biblioteca de persistencias Room brinda una capa de abstracción para SQLite que permite acceder a la base de datos sin problemas y, al mismo tiempo, aprovechar toda la tecnología de SQLite. En particular, Room brinda los siguientes beneficios:

- Verificación del tiempo de compilación de las consultas en SQL
- Anotaciones de conveniencia que minimizan el código estándar repetitivo y propenso a errores
- Rutas de migración de bases de datos optimizadas

Debido a estas consideraciones, te recomendamos que uses Room en lugar de **usar las API de SQLite directamente**.

Configuración

Para usar Room en tu app, agrega las siguientes dependencias al archivo `build.gradle` de la app:

GroovyKotlin

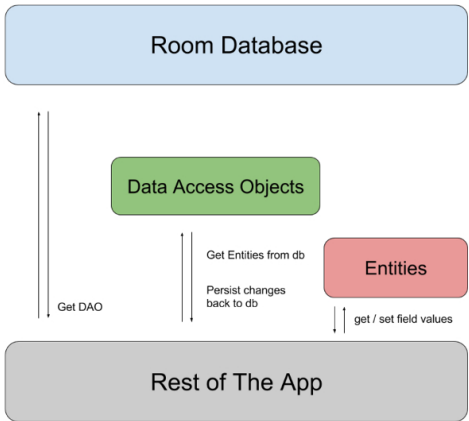
```
dependencies {  
    val room_version = "2.4.3"  
  
    implementation("androidx.room:room-runtime:$room_version")  
    annotationProcessor("androidx.room:room-compiler:$room_version")  
  
    // To use Kotlin annotation processing tool (kapt)  
    kapt("androidx.room:room-compiler:$room_version")  
    // To use Kotlin Symbol Processing (KSP)  
    ksp("androidx.room:room-compiler:$room_version")  
  
    // optional - Kotlin Extensions and Coroutines support for Room  
    implementation("androidx.room:room-ktx:$room_version")  
  
    // optional - RxJava2 support for Room  
    implementation("androidx.room:room-rxjava2:$room_version")  
  
    // optional - RxJava3 support for Room  
    implementation("androidx.room:room-rxjava3:$room_version")  
  
    // optional - Guava support for Room, including Optional and ListenableFuture  
    implementation("androidx.room:room-guava:$room_version")  
  
    // optional - Test helpers  
    testImplementation("androidx.room:room-testing:$room_version")  
  
    // optional - Paging 3 Integration  
    implementation("androidx.room:room-paging:2.5.0-alpha03")  
}
```

Componentes principales

Estos son los tres componentes principales de Room:

- La **clase de la base de datos** que contiene la base de datos y sirve como punto de acceso principal para la conexión subyacente a los datos persistentes de la app.
- Las **entidades de datos** que representan tablas de la base de datos de tu app.
- **Objetos de acceso a datos (DAOs)** que proporcionan métodos que tu app puede usar para consultar, actualizar, insertar y borrar datos en la base de datos.

La clase de base de datos proporciona a tu app instancias de los DAOs asociados con esa base de datos. A su vez, la app puede usar los DAOs para recuperar datos de la base de datos como instancias de objetos de entidad de datos asociados. La app también puede usar las entidades de datos definidas a fin de actualizar filas de las tablas correspondientes o crear filas nuevas para su inserción. En la figura 1, se muestran las relaciones entre los diferentes componentes de Room.



En esta página

Configuración

Componentes principales

Ejemplo de implementación

- Entidad de datos
- Objeto de acceso a datos (DAO)
- Base de datos
- Uso

Recursos adicionales

- Ejemplo
- Codelabs
- Blogs

clave-valor

▼

Cómo guardar contenido en una base de datos local

Descripción general

Cómo definir datos mediante entidades

Cómo acceder a datos mediante DAO

Cómo definir relaciones entre objetos

Escribe consultas DAO asincrónicas

Cómo implementar vistas en una base de datos

Cómo autocompletar el contenido de tu base de datos

Cómo migrar tu base de datos

Cómo probar y depurar tu base de datos

Cómo hacer referencia a datos complejos

▶

Cómo guardar contenido en el almacenamiento compartido

Cómo administrar todos los archivos de un dispositivo de almacenamiento

Cómo guardar datos de pares clave-valor

▼

Cómo guardar contenido en una base de datos local

Descripción general

Cómo definir datos mediante entidades

Cómo acceder a datos mediante DAO

Cómo definir relaciones entre objetos

Escribe consultas DAO asincrónicas

Cómo implementar vistas en una base de datos

Cómo autocompletar el contenido de tu base de datos

Cómo migrar tu base de datos

Cómo probar y depurar tu base de datos

Cómo hacer referencia a datos complejos

▶

Cómo guardar contenido en el almacenamiento compartido

Cómo administrar todos los archivos de un dispositivo de almacenamiento

Cómo guardar datos de pares clave-valor

▼

Cómo guardar contenido en una base de datos local

Descripción general

Cómo definir datos mediante entidades

Cómo acceder a datos mediante DAO

Cómo definir relaciones entre objetos

Escribe consultas DAO asincrónicas

Cómo implementar vistas en una base de datos

Cómo autocompletar el contenido de tu base de datos

Cómo migrar tu base de datos

Cómo probar y depurar tu base de datos

Cómo hacer referencia a datos complejos

▶

Cómo guardar contenido en el almacenamiento compartido

Cómo administrar todos los archivos de un dispositivo de almacenamiento

Cómo guardar datos de pares clave-valor

▼

Cómo guardar contenido en una base de datos local

Descripción general

Cómo definir datos mediante entidades

Cómo acceder a datos mediante DAO

Cómo definir relaciones entre objetos

Escribe consultas DAO asincrónicas

Cómo implementar vistas en una base de datos

Cómo autocompletar el contenido de tu base de datos

Cómo migrar tu base de datos

Cómo probar y depurar tu base de datos

Cómo hacer referencia a datos complejos

▶

Cómo guardar contenido en el almacenamiento compartido

Cómo administrar todos los archivos de un dispositivo de almacenamiento

Cómo guardar datos de pares clave-valor

▼

Cómo guardar contenido en una base de datos local

Figura 1. Diagrama de la arquitectura de la biblioteca de Room

Ejemplo de implementación

En esta sección, se presenta un ejemplo de implementación de una base de datos de Room con una sola entidad de datos y un DAO único.

Entidad de datos

El siguiente código define una entidad de datos `User`. Cada instancia de `User` representa una fila en una tabla de `user` en la base de datos de la app.

KotlinJava

```
@Entity
data class User(
    @PrimaryKey val uid: Int,
    @ColumnInfo(name = "first_name") val firstName: String?,
    @ColumnInfo(name = "last_name") val lastName: String?
)
```

Para obtener más información sobre las entidades de datos de Room, consulta [Cómo definir datos con entidades de Room](#).

Objeto de acceso a datos (DAO)

El siguiente código define un DAO llamado `UserDao`. `UserDao` proporciona los métodos que el resto de la app usa para interactuar con los datos de la tabla `user`.

KotlinJava

```
@Dao
interface UserDao {
    @Query("SELECT * FROM user")
    fun getAll(): List<User>

    @Query("SELECT * FROM user WHERE uid IN (:userIds)")
    fun loadAllByIds(userIds: IntArray): List<User>

    @Query("SELECT * FROM user WHERE first_name LIKE :first AND " +
        "last_name LIKE :last LIMIT 1")
    fun findByName(first: String, last: String): User

    @Insert
    fun insertAll(vararg users: User)

    @Delete
    fun delete(user: User)
}
```

Para obtener más información sobre los DAOs, consulta [Cómo acceder a los datos con DAO de Room](#).

Database

Con el siguiente código, se define una clase `AppDatabase` para contener la base de datos. `AppDatabase` define la configuración de la base de datos y sirve como el punto de acceso principal de la app a los datos persistentes. La clase de la base de datos debe cumplir con las siguientes condiciones:

- La clase debe tener una anotación `@Database` que incluya un array `entities` que enumere todas las entidades de datos asociados con la base de datos.
- Debe ser una clase abstracta que extienda `RoomDatabase`.
- Para cada clase DAO que se asoció con la base de datos, esta base de datos debe definir un método abstracto que tenga cero argumentos y muestre una instancia de la clase DAO.

KotlinJava

```
@Database(entities = [User::class], version = 1)
abstract class AppDatabase : RoomDatabase() {
    abstract fun userDao(): UserDao
}
```

★ **Nota:** Si tu app se ejecuta en un solo proceso, debes seguir el patrón de diseño singleton cuando crees una instancia de un objeto `AppDatabase`. Cada instancia `RoomDatabase` es bastante costosa y rara vez necesitas acceder a varias instancias en un mismo proceso.

Si tu app se ejecuta en varios procesos, incluye `enableMultiInstanceInvalidation()` en tu invocación del creador de bases de datos. De esa manera, cuando tienes una instancia de `AppDatabase` en cada proceso, puedes invalidar el archivo de base de datos compartido en un proceso y esta invalidación se propaga automáticamente a las instancias de `AppDatabase` dentro de otros procesos.

Uso

Después de definir la entidad de datos, el DAO y el objeto de base de datos, puedes usar el siguiente código para crear una instancia de la base de datos:



KotlinJava

```
val db = Room.databaseBuilder(
    applicationContext,
    AppDatabase::class.java, "database-name"
).build()
```

Luego, puedes usar los métodos abstractos de `AppDatabase` para obtener una instancia del DAO. A su vez, puedes usar los métodos de la instancia del DAO para interactuar con la base de datos:

KotlinJava

```
val userDao = db.userDao()
val users: List<User> = userDao.getAll()
```



Recursos adicionales

Si deseas obtener más información sobre Room, consulta los siguientes recursos adicionales:

Ejemplo



- Android Sunflower, una app de jardinería que ilustra las prácticas recomendadas de desarrollo de Android con Android Jetpack.
- Tivi, una app de seguimiento de programas de TV que usa las bibliotecas y las herramientas más recientes.

Codelabs

- Android Room con una vista (Java) (Kotlin)

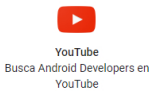
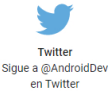
Blogs

- 7 sugerencias profesionales para Room
- Migración incremental de SQLite a Room

¿Te resultó útil?


Content and code samples on this page are subject to the licenses described in the [Content License](#). Java and OpenJDK are trademarks or registered trademarks of Oracle and/or its affiliates.

Last updated 2022-09-05 UTC.



MÁS ANDROID	DESCUBRE	DISPOSITIVOS ANDROID	VERSIONES	DOCUMENTACIÓN Y DESCARGAS	ASISTENCIA
Android	Videjuegos	Pantallas grandes	Android 11	Guía de Android Studio	Informar sobre un error en la plataforma
Android para empresas	Aprendizaje automático	Wear OS	Android 10	Guías para desarrolladores	Informar sobre un error en la documentación
Seguridad	Privacidad	Android TV	Pie	Referencia de API	Google Play support
Código abierto	5G	Android para vehículos	Oreo	Descargar Studio	Participar en los estudios de investigación
Noticias		Android Things	Nougat	NDK de Android	
Blog		Dispositivos con Chrome OS	Marshmallow		
Podcasts			Lollipop		
			KitKat		

Google Developers Android Chrome Firebase Google Cloud Platform Todos los productos