



# Acceso a datos en Android

La librería ROOM (Parte I)

***Implementar capa de acceso  
a datos en un aplicativo  
móvil utilizando la librería  
ROOM para otorgar  
persistencia de estados  
resolviendo el problema  
planteado***

**{desafío}**  
**latam\_**

- Unidad 1:  
Acceso a datos en Android
- Unidad 2:  
Consumo de API REST
- Unidad 3:  
Testing
- Unidad 4:  
Distribución del aplicativo Android



Te encuentras aquí



## ¿Qué aprenderás en esta sesión?

- *Describe el rol de la capa de abstracción Room y sus principales elementos tales como Entidad y DAO*

¿Podrías nombrar 2  
ventajas de utilizar  
Room?



**`/* Contexto de uso de Room */`**

# Contexto de uso de Room

- En general, Room está diseñado para facilitar a los desarrolladores el trabajo con bases de datos en Android al proporcionar una forma simple y eficiente de interactuar con los datos en una base de datos SQLite.
- Su objetivo es simplificar el proceso de creación, lectura, actualización y eliminación de datos en una base de datos SQLite y mejorar el rendimiento de las aplicaciones de Android al descargar las operaciones de la base de datos a subprocesos en segundo plano.

# Room

*Algunas de las características clave de Room incluyen:*

- **Entidades anotadas:** Room utiliza anotaciones para definir el esquema de la base de datos y las entidades que representan los datos.
- **SQL con seguridad de tipos:** Room proporciona una API con seguridad de tipos para consultar la base de datos, lo que elimina la necesidad de usar declaraciones de SQL sin formato.
- **Compatibilidad con LiveData y StateFlow:** Room se integra con LiveData, una biblioteca que le permite observar los datos en la base de datos y actualizar automáticamente la interfaz de usuario cuando los datos cambian.
- **Seguridad de subprocesos:** Room proporciona compatibilidad integrada para ejecutar operaciones de base de datos en subprocesos en segundo plano, lo que facilita la realización de operaciones de base de datos sin bloquear el subproceso de la interfaz de usuario.

**`/* Integrar Room en un proyecto */`**



# Integrar Room en un proyecto

Para integrar Room en un proyecto de Android, deberá realizar los siguientes pasos:

- Agrega la biblioteca Room al archivo build.gradle de tu aplicación:

```
dependencies {  
    implementation "androidx.room:room-runtime:<version>"  
    annotationProcessor "androidx.room:room-compiler:<version>"  
    testImplementation "androidx.room:room-testing:<version>"  
}
```

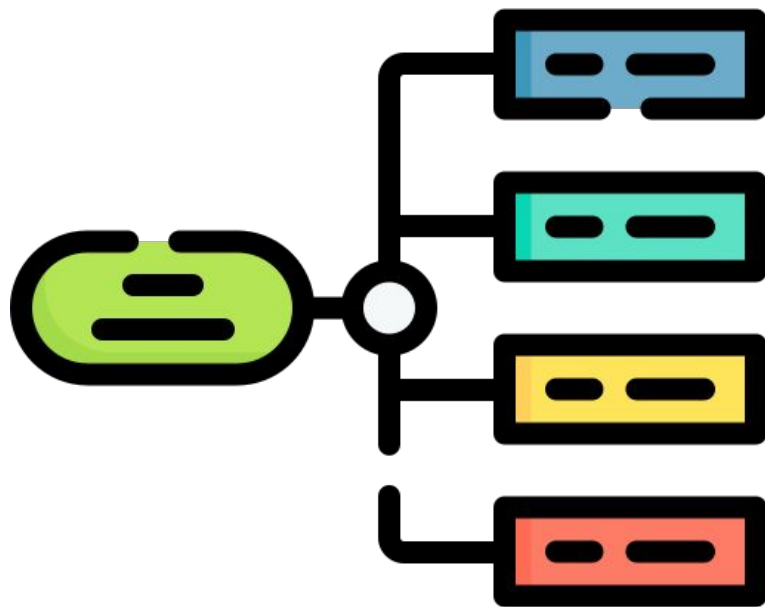
Es necesario crear una clase abstracta que extienda de **RoomDatabase**, pero antes se deben definir las entidades y las DAOs.

**/\* Definiendo datos  
mediante entidades \*/**

# ¿Qué es una entidad o Entity?

En Room, una entidad es una clase que representa una tabla en la base de datos. Se utiliza para definir el esquema de la base de datos y la estructura de los datos que se almacenarán en ella.

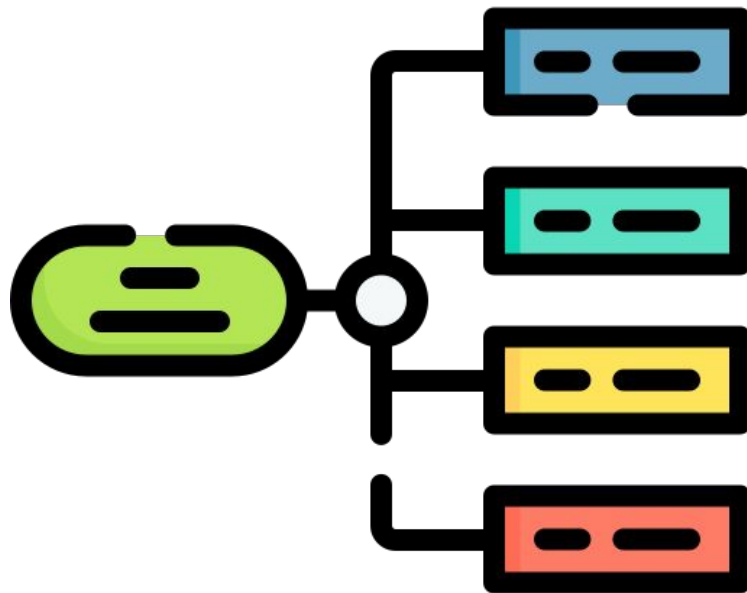
Una clase de entidad se anota como `@Entity` y sus campos corresponden a las columnas de la tabla en la base de datos.



# ¿Qué es una entidad o Entity?

Una clase de entidad define el esquema de la tabla de la base de datos, incluidas las columnas y sus tipos.

Room utiliza las anotaciones en los campos y la clase para crear automáticamente la tabla en la base de datos. Los campos de clase deben representar las columnas de la tabla y la clase debe tener definida una clave principal.



# Definiendo datos mediante entidades

Define las entidades de tu base de datos creando data class que se anotan con **@Entity**. Estas clases representarán las tablas de tu base de datos y los campos de la clase representarán las columnas de la tabla.

Por ejemplo:

```
@Entity(tableName = "users")
data class User(
    @PrimaryKey
    val email: String,
    val name: String,
    val age: Int
)
```



Es importante tener en cuenta que Room solo admite un subconjunto de tipos de Java, por lo que es posible que debas usar tipos proporcionados por Room, como TypeConverters, para campos que no son reconocidos

**/\* Accesando datos mediante DAO \*/**

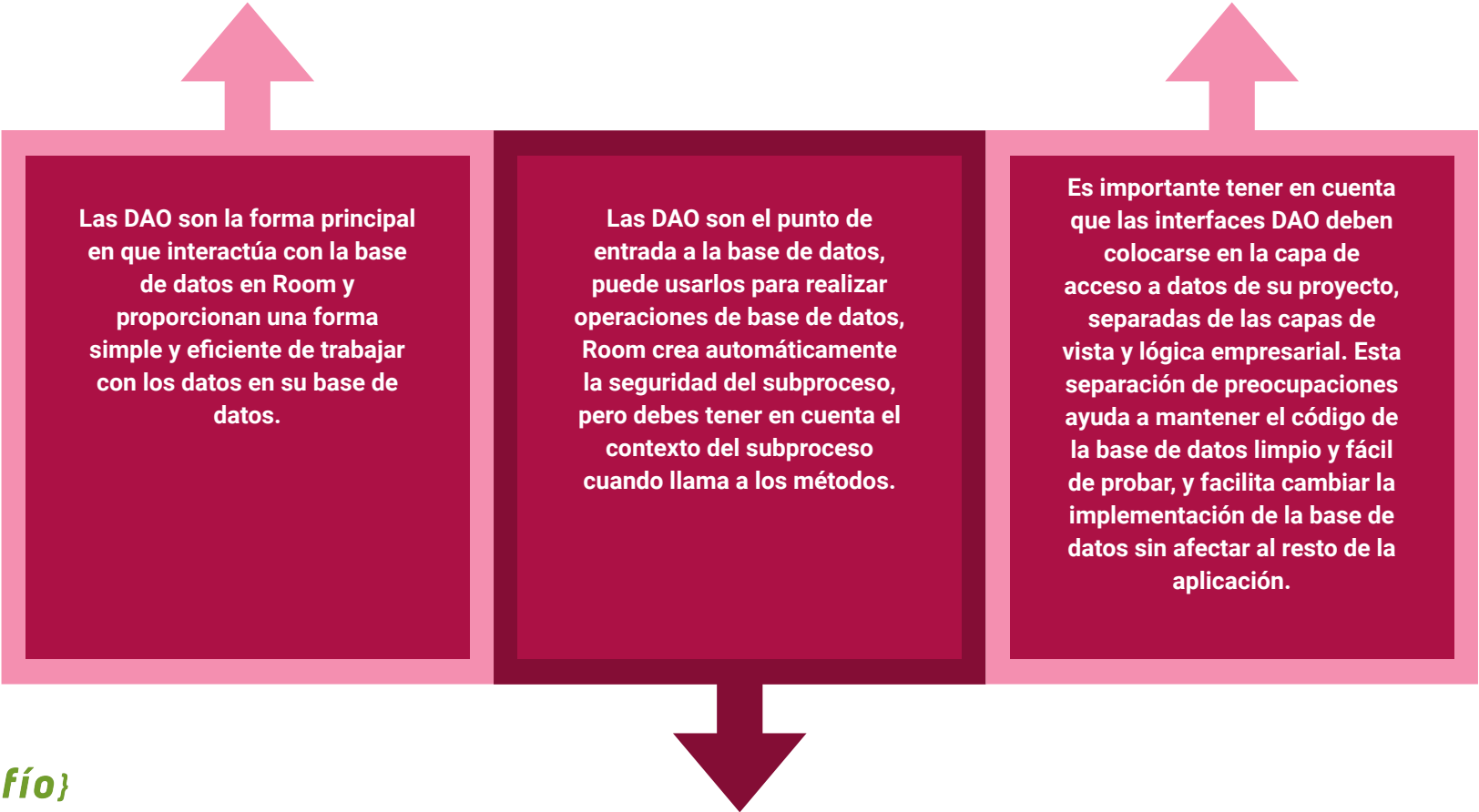
# ¿Qué es DAO?

**DAO** significa Data Access Object u Objeto de acceso a datos, es un patrón que se usa para abstraer los detalles de cómo se accede y almacena los datos. En Room, una DAO es una interfaz que define los métodos que se pueden usar para interactuar con la base de datos.

Una DAO define los métodos que se pueden usar para realizar operaciones CRUD (Crear, Leer, Actualizar y Eliminar) en las entidades de la base de datos. Room proporciona anotaciones que puede usar para indicar el tipo de operación que realizará un método (por ejemplo, @Insert, @Update, @Delete) y la consulta que ejecutará el método (por ejemplo, @Query).



Room generará el código necesario para implementar la DAO en función de las anotaciones y los métodos de la interfaz.



The diagram consists of three rectangular boxes arranged horizontally. The left and right boxes have a light pink border, while the middle box has a dark red border. Above the left and right boxes are light pink upward-pointing arrows. Below the middle box is a dark red downward-pointing arrow.

Las DAO son la forma principal en que interactúa con la base de datos en Room y proporcionan una forma simple y eficiente de trabajar con los datos en su base de datos.

Las DAO son el punto de entrada a la base de datos, puede usarlos para realizar operaciones de base de datos, Room crea automáticamente la seguridad del subproceso, pero debes tener en cuenta el contexto del subproceso cuando llamas a los métodos.

Es importante tener en cuenta que las interfaces DAO deben colocarse en la capa de acceso a datos de su proyecto, separadas de las capas de vista y lógica empresarial. Esta separación de preocupaciones ayuda a mantener el código de la base de datos limpio y fácil de probar, y facilita cambiar la implementación de la base de datos sin afectar al resto de la aplicación.



# Accesando datos mediante DAO

A continuación veremos un ejemplo de una interfaz DAO que se puede usar para interactuar con el ejemplo de del usuario (User) que proporcionamos antes:

```
@Dao
interface UserDao {

    @Insert
    fun insert(user: User)

    @Update
    fun update(user: User)

    @Delete
    fun delete(user: User)

    @Query("SELECT * FROM users")
    fun getAllUsers(): List<User>

    @Query("SELECT * FROM users WHERE email = :email")
    fun findUserByEmail(email: String): User
}
```

**/\* Accediendo a la base de datos \*/**

# Accediendo a la base de datos

Crea una clase que amplíe **RoomDatabase** y anótala con **@Database**. Esta clase definirá la versión de la base de datos y la lista de entidades. Esta clase también proporcionará acceso a las DAO.

Ejemplo:

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

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

Ten en cuenta que el número de versión de la base de datos debe aumentar cada vez que realices cambios en el esquema de la base de datos, como agregar o eliminar tablas o columnas. Además, Room usa el complemento kapt gradle en lugar del procesador de anotaciones, esto es específico para los proyectos de kotlin.

# Resumiendo

Los pasos necesarios para implementar y empezar a usar Room en tu proyecto son los siguientes:

- Define las entidades de tu base de datos creando clases que se anotan con `@Entity`. Estas clases representarán las tablas de su base de datos y las propiedades de la clase representarán las columnas de la tabla.
- Cree una clase que extienda de `RoomDatabase` y anótala con `@Database`. Esta clase definirá la versión de la base de datos y la lista de entidades. Esta clase también proporcionará acceso a los DAO.
- Define objetos de acceso a datos (DAO) mediante la creación de interfaces que se anotan con `@Dao`. Estas interfaces definirán los métodos que puede usar para interactuar con la base de datos.

# Resumiendo

Los pasos necesarios para implementar y empezar a usar Room en tu proyecto son los siguientes:

- En tu clase de aplicación o cualquier otra clase en la que desee crear la base de datos, obtenga una instancia de RoomDatabase llamando a Room.databaseBuilder() y pasando el contexto, la clase de base de datos y el nombre de la base de datos.
- Utiliza los DAO para realizar operaciones de base de datos.



Se recomienda utilizar la arquitectura Room junto con los componentes de la arquitectura de Android, como ViewModel, LiveData o StateFlow y Coroutines, para lograr una solución robusta y mantenible.

¿Te ha parecido de utilidad el  
uso de la librería ROOM?

¿Has tenido alguna dificultad en  
lo visto en clases? ¡Hablémoslo!





## Próxima sesión...

- *Describe el rol de la capa de abstracción Room y sus principales elementos tales como Entidad y DAO*

**{desafío}**  
**latam\_**

*Academia de  
talentos digitales*

