04-02-Performing-Insert-Operation-on-Room

On se place dans le projet Step2

PART 2 : Performing INSERT Operation

1 - Create the NewBookActivity

Créer une nouvelle EmptyActivity : `NewBookActivity` en cochant "Generate a layout file"

2

Dans acttivity_new_book.xml :

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".NewBookActivity"
    android:paddingTop="75dp">

    <EditText
        android:id="@+id/etAuthorName"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:ems="15"
        android:hint="@string/author"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <EditText
        android:id="@+id/etBookName"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_marginTop="46dp"
        android:ems="15"
        android:hint="@string/book"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/etAuthorName" />

    <Button
        android:id="@+id/bAdd"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_marginTop="16dp"
        android:text="@string/save"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.498"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/etBookName" />


</androidx.constraintlayout.widget.ConstraintLayout>
```

3 - Moving from the MainActivity to NewBookActivity

Dans le MainActivity.kt :

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)
    setSupportActionBar(toolbar)

    val bookDB = BookRoomDatabase.getDatabase(this);

    fab.setOnClickListener { view ->
        val intent = Intent(this, NewBookActivity::class.java)
        startActivityForResult(intent, NEW_BOOK_REQUEST_CODE)
    }
}

companion object {
    const val NEW_BOOK_REQUEST_CODE = 1
}
```

4

Dans le NewBookActivity :

```
bAdd.setOnClickListener {
    val resultIntent = Intent()
    if (TextUtils.isEmpty(etAuthorName.text) ||
TextUtils.isEmpty(etBookName.text)) {
        setResult(Activity.RESULT_CANCELED)
    }
    else {
        val authorName = etAuthorName.text.toString()
        val bookName = etBookName.text.toString()
        resultIntent.putExtra(NEW_AUTHOR, authorName)
        resultIntent.putExtra(NEW_BOOK, bookName)
        setResult(Activity.RESULT_OK, resultIntent)
    }
    finish()
}

...

companion object {
        const val NEW_AUTHOR: String = "new_author"
        const val NEW_BOOK: String = "new_book"
    }
```

5 - Main Activity handles Data intent sent by NewBookActivity

```
override fun onActivityResult(requestCode: Int, resultCode: Int, data:
Intent?) {
    super.onActivityResult(requestCode, resultCode, intent)
    if (resultCode == Activity.RESULT_OK && requestCode ==
MainActivity.NEW_BOOK_REQUEST_CODE) {
        insertNewBook(data)
    }
    else {
        Toast.makeText(applicationContext, R.string.not_saved,
Toast.LENGTH_LONG).show()
    }
}
```

6

Créer un nouveau ViewModel : `BookViewModel` :

```
class BookViewModel (application :Application) :
.AndroidViewModel(application) {
    private val bookDao : BookDao
    private val myExecutor = Executors.newSingleThreadExectutor()

    init {
        val bookDB = BookRoomDatabase.getDatabase(application)
        bookDao = bookDB!!.bookDao()
    }
    fun insert(book: Book) {
        myExecutor.execute{
            bookDao.insert(book)
        }
    }
}
```

Dans le main actvity :

```
private lateinit var bookViewModel: BookViewModel

...

bookViewModel = ViewModelProviders.of(this).get(BookViewModel::class.java)
```

Implémentation de la fonction `insertNewBook` :

```
private fun insertNewBook(data: Intent?) {
    val id = UUID.randomUUID().toString()
    val authorName: String? =
data?.getStringExtra(NewBookActivity.NEW_AUTHOR)
    val bookName : String? = data?.getStringExtra(NewBookActivity.NEW_BOOK)
    val book = Book(id, authorName!!, bookName!!)

    bookViewModel.insert(book)

    Toast.makeText(applicationContext, "Saved", Toast.LENGTH_LONG).show()
}
```

EXECUTER !!

Dans le DeviceFileExplorer : `/data/data/iutinfo.lp.devmob.bookkeeper/databases`

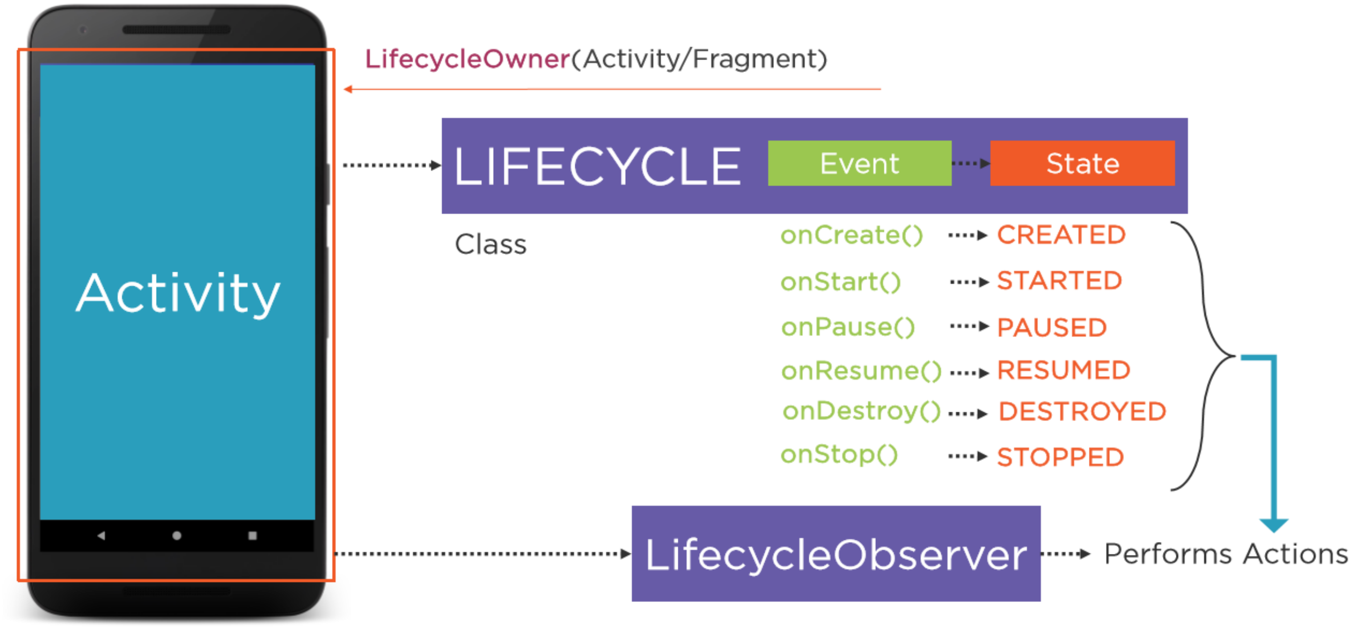
Récupérer les trois fichier et lancer sqlite sur le fichier book_room_db.sqlite

.tables :

```
sqlite> .tables
android_metadata    book                  room_master_table
```

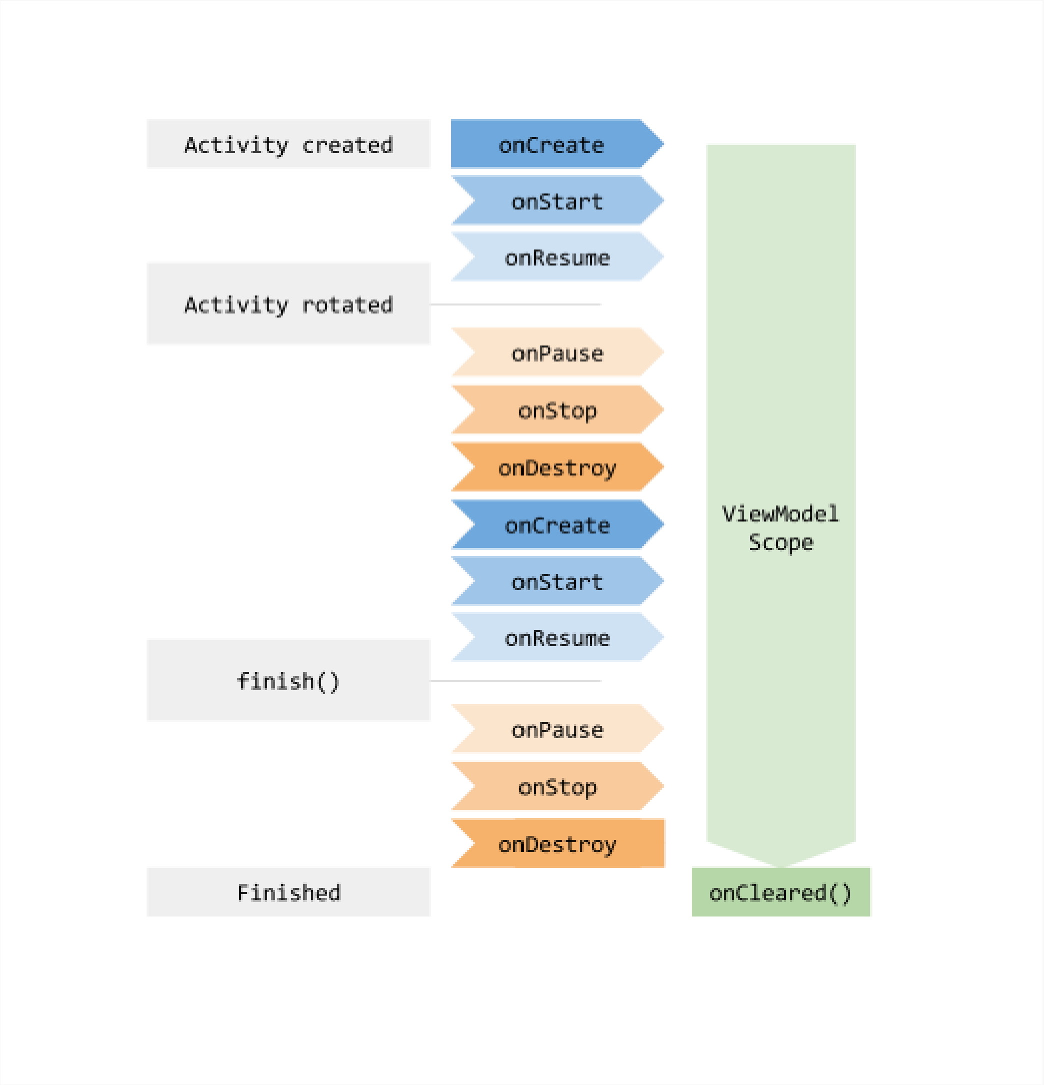
Getting Familiar with Android Architecture Components

Lifecyle Aware Components

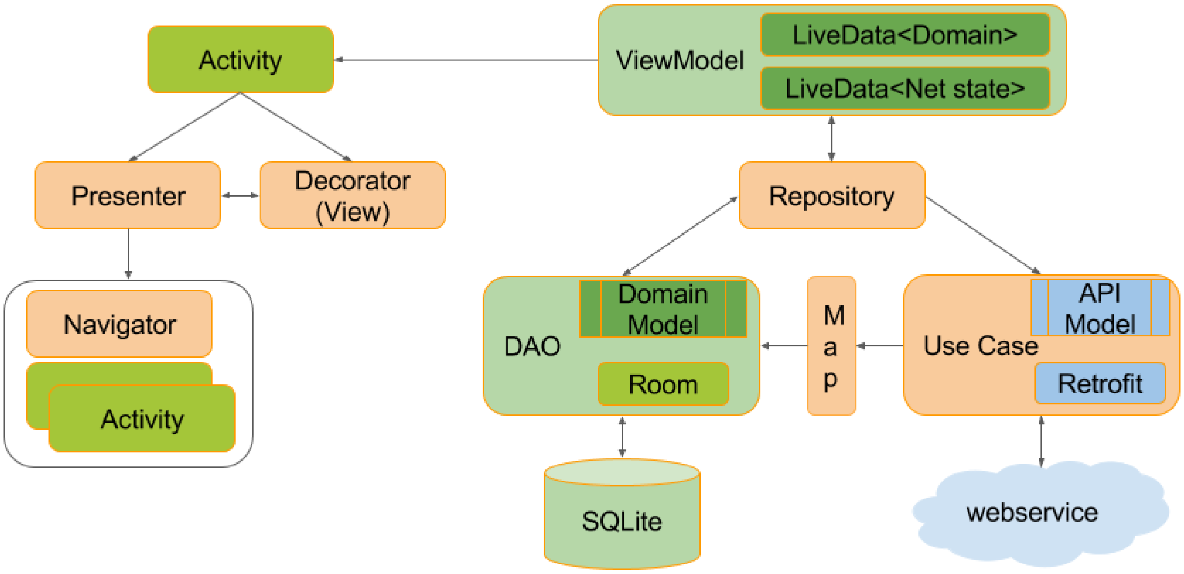


Why ViewModel ?

Les données dans le ViewModel persistent peut importe l'état de l'activité



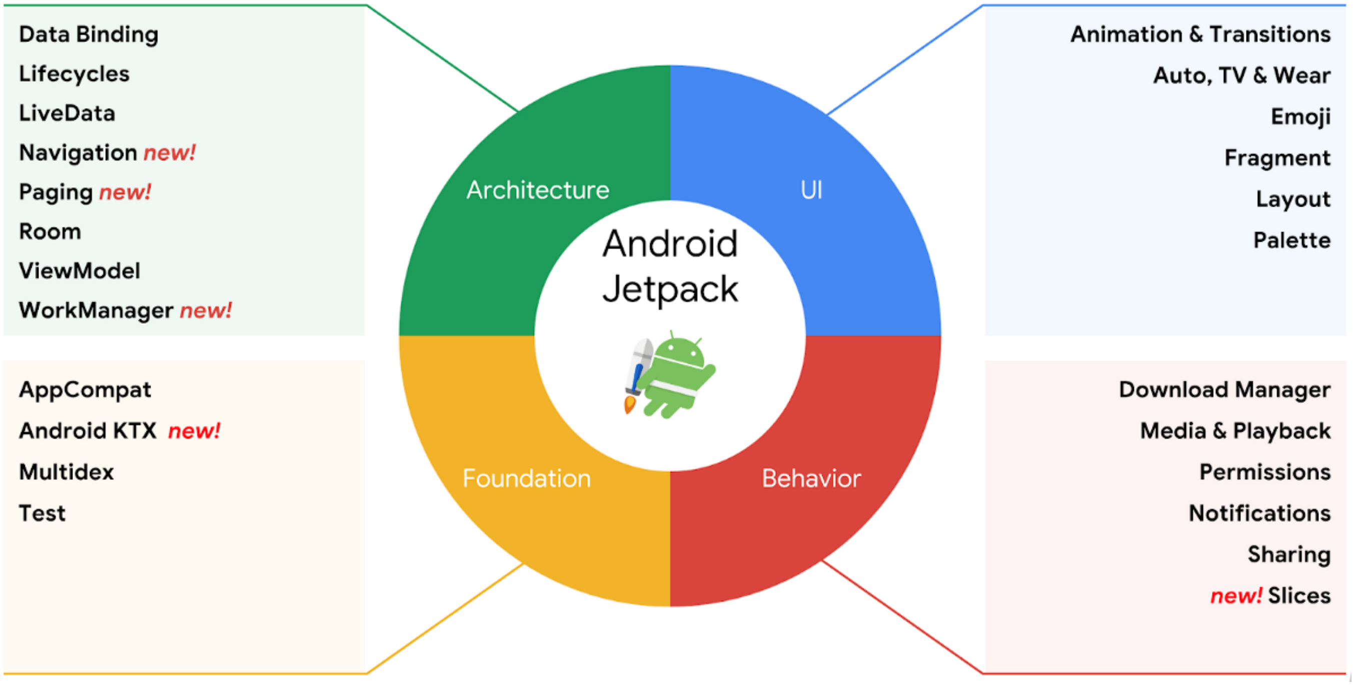
- Display UI Data
- React to user actions
- Handle OS Communication
- Load data from network/DB



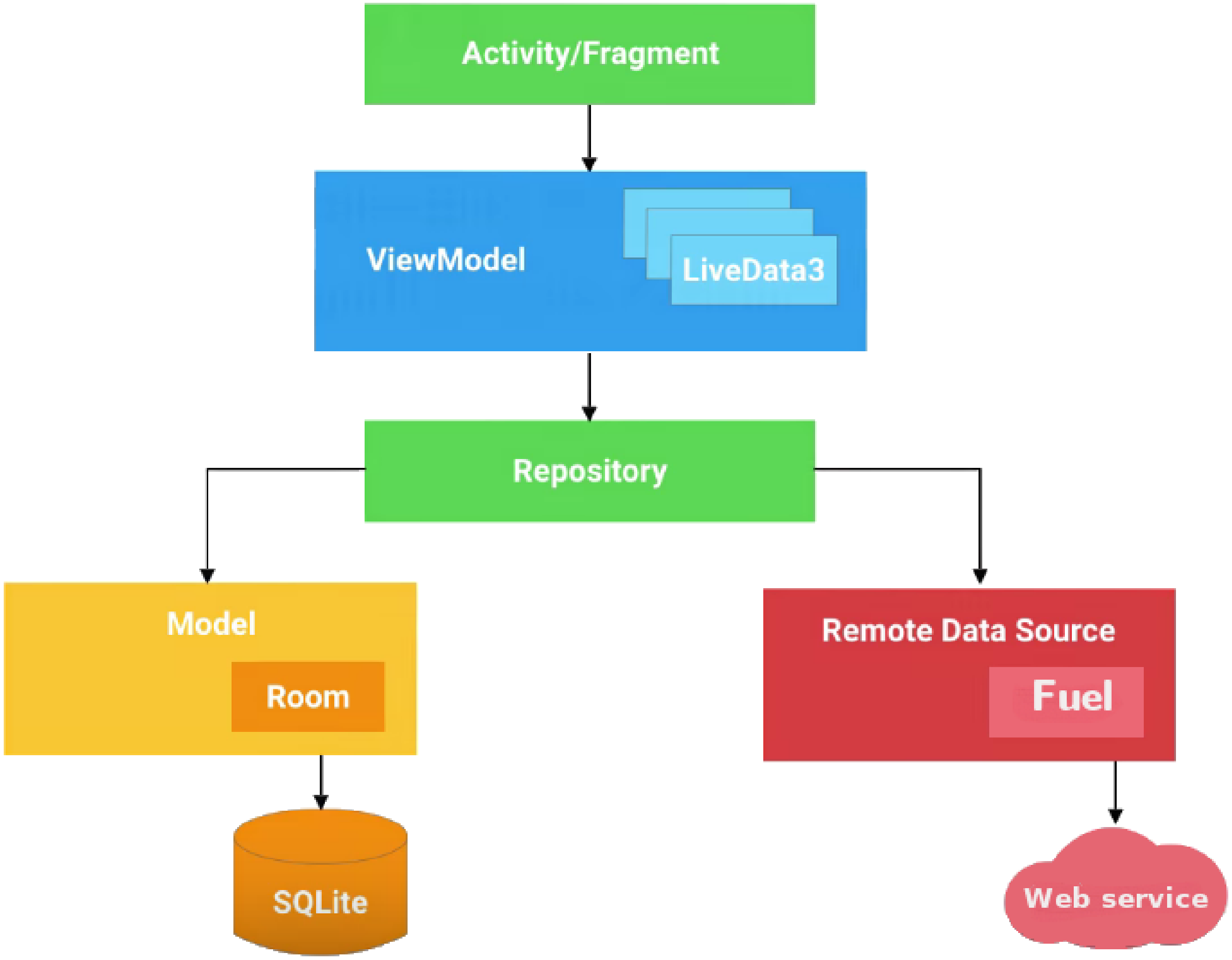
Android Room Fundamentals

What are Android Architecture Components ?

- Libraries for different tasks
- Robust, testable, maintainable, less boilerplate code
- Architecture components are a part of [Android Jetpack]



Components



Options for Storage

- Realm DataBase
- SQLite
- Room Database

Working with Room

Components of Room :

- Entities : schema de notre base de données (annotées vec @Entity)
- DataAccessObject : contient les méthodes pour accéder à la DB (annoté avec @Dao)
- Database : contient les données (annotée avec @Database)

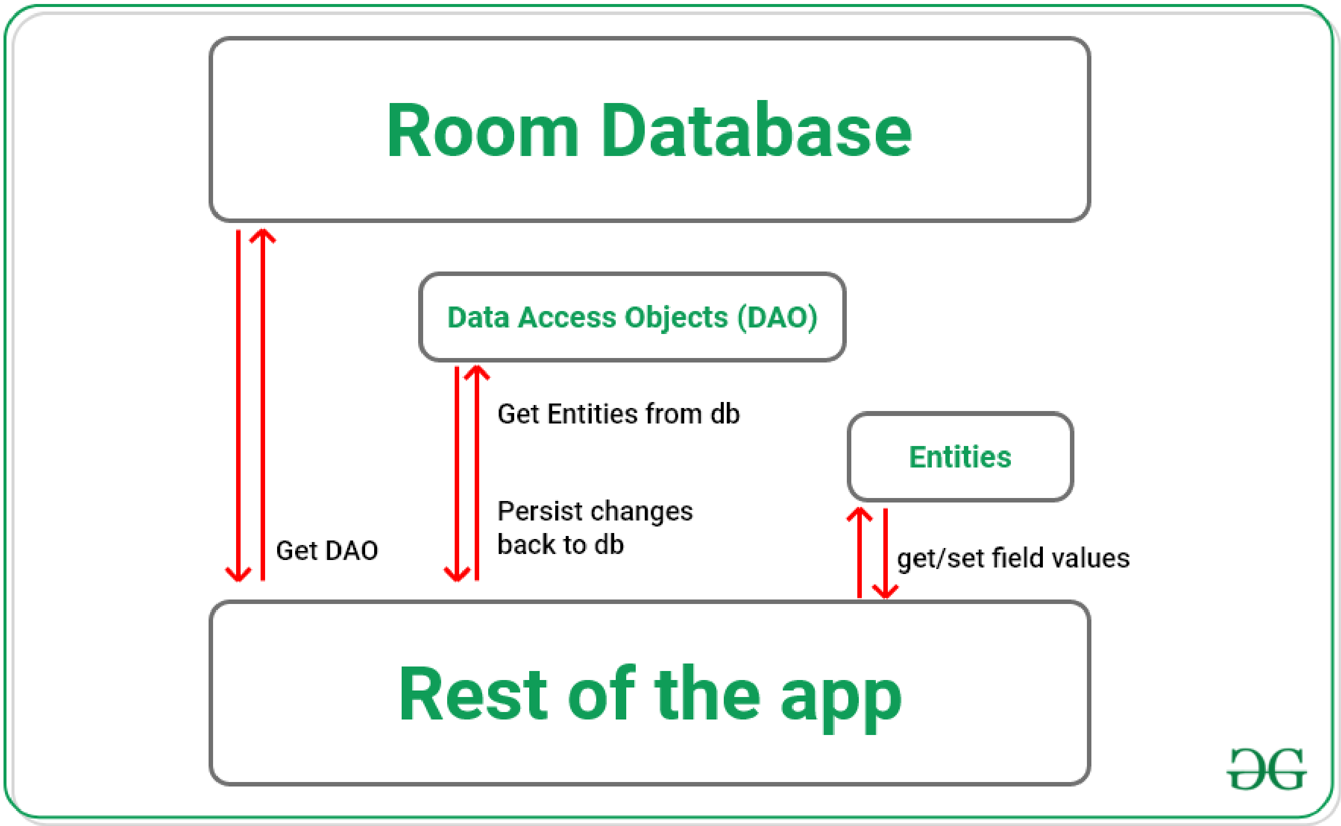
Annotations for Entity :

- @Entity
- @PrimaryKey

- @ColumnInfo
- @Ignore
- @ForeignKey

Annotations for Dao :

- @Dao
- @Insert
- @Update
- @Delete
- @Query



Threads asynchrones pour ne pas que l'UI attende (Loader / Barres de chargement)