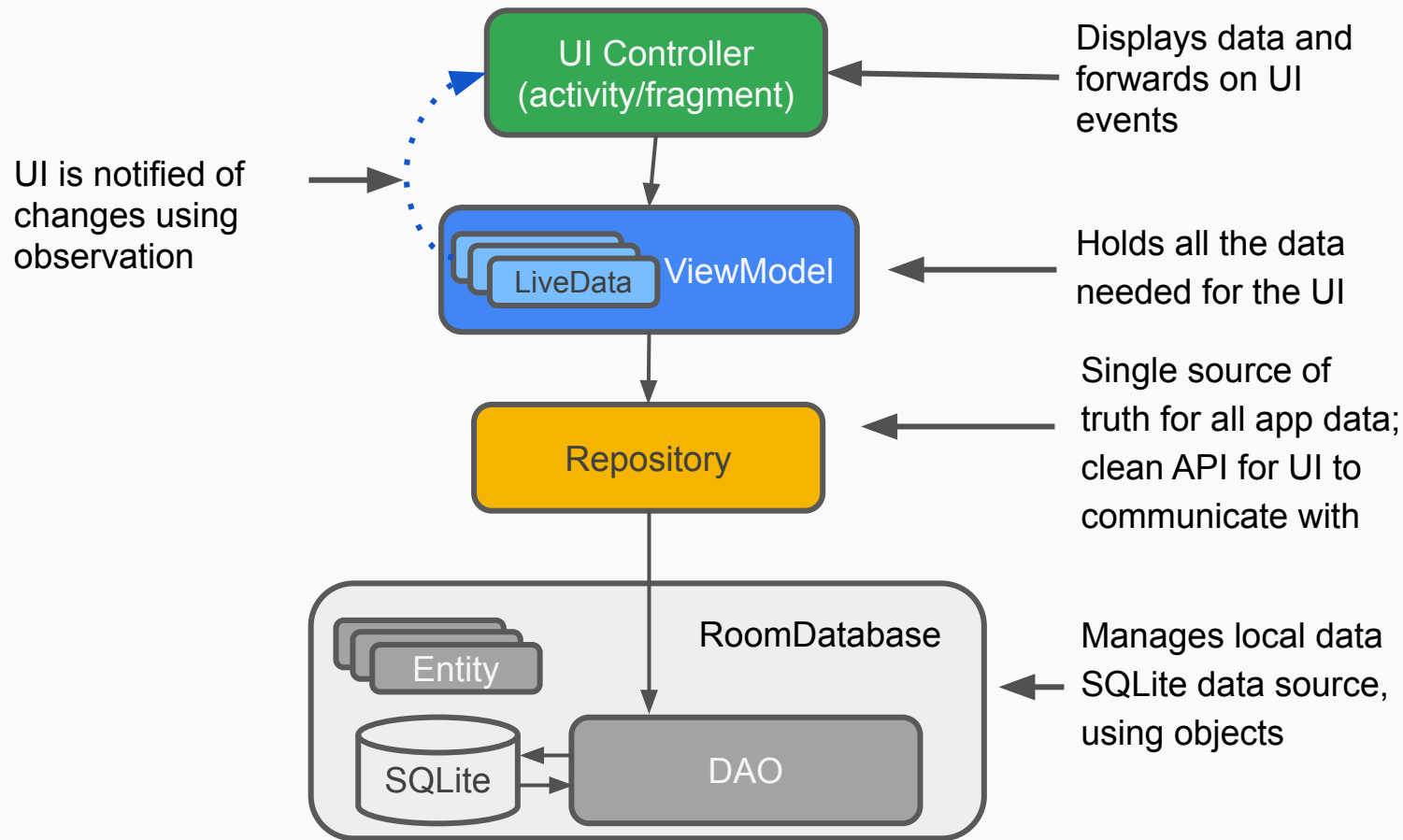


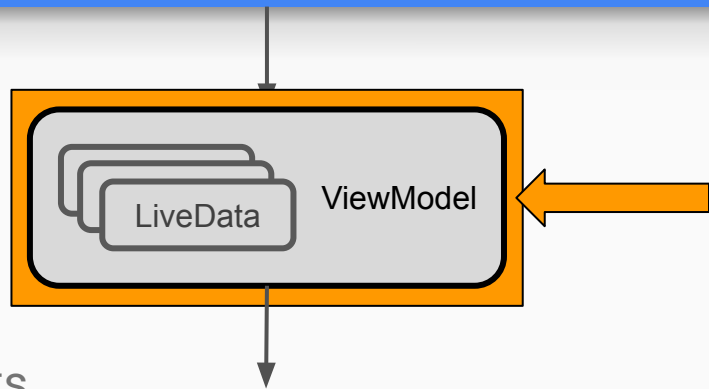
# ViewModel





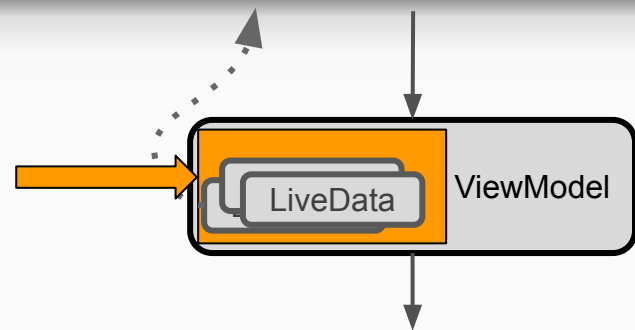
# ViewModel

- Fournis le données à l'UI
- Survis aux configuration changes
- Pas au redémarrage d'app
- Peut aussi partager des données entre Fragments
- Fait partie [lifecycle library](#)
- Ne pas passer de Context (si besoin, étendre [AndroidViewModel](#))
- Analogie: Serveur



# LiveData

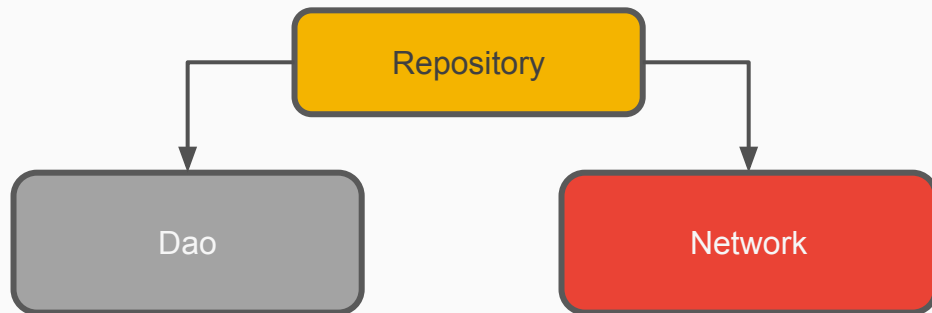
- Classe encapsulant de la donnée observable
- Permet de garder l'UI à jour dynamiquement
- Notifies observer when data changes
- lifecycle aware
- Généralement, c'est un composant ayant un lifecycle qui observe et met à jour l'UI en conséquence



```
// In fragment or Activity:  
viewModel.score.observe(this, Observer{ score ->  
    score_tv.text = score  
})
```

# Repository

- Pas un Architecture Components mais une bonne pratique
- Récupère la donnée en tâche de fond
- Peut choisir la source, les synchroniser, ...
- Analogie: Cuisine



# ViewModel example using repository

```
class WordViewModel : ViewModel(){
    private val repository = WordRepository(application)
    private var _words = MutableLiveData<Int>()
    val words: LiveData<List<Word>>
        get() = _words
    fun fetchWords() { viewModelScope.launch { _words.setValue(repository.getAll())}}
}

class WordRepository(application: Application) {
    val db = WordRoomDatabase.getDatabase(application)
    suspend fun getAll(): List<Word> { return db.wordDao().getAllWords()}
}

// in fragment or Activity:
viewModel.words.observe(this, Observer{ words ->
    // Update list
})
```