# 10-Add-Repository

Créer une classe `BookRepository`

Copier tout le contenu de `BookViewModel` et le coller dans la classe `BookRepository`

Faire en sorte que `BookViewModel` va chercher les données dans `BookRepository` :

```
class BookViewModel(application: Application): AndroidViewModel(application)
{
    private val bookRepository = BookRepository(application)
    val allBooks: LiveData<List<Book>>

    private val myExecutor = Executors.newSingleThreadExecutor()

    init {
        allBooks = bookRepository.allBooks
    }

    fun insert(book: Book) {
        bookRepository.insert(book)
    }

    fun update(updatedBook: Book) {
        bookRepository.update(updatedBook)
    }

    fun delete(book:Book) {
        bookRepository.delete(book)
    }
}
```

Ajout dans le `BookRepository` :

```
fun getBooksByAuthorOrBook (searchString: String) : LiveData<List<Book>> {
    return bookDao.getBooksByAuthorOrBook(searchString)
}
```

On modifie la classe `SearchViewModel` pour intéragir avec le `BookRepository` :

```
class SearchViewModel(application: Application):
AndroidViewModel(application) {
    private val bookRepository = BookRepository(application)
    val allBooks: LiveData<List<Book>>

    private val myExecutor = Executors.newSingleThreadExecutor()

    init {
        allBooks = bookRepository.allBooks
    }

    fun update(updatedBook: Book) {
        bookRepository.update(updatedBook)
    }

    fun delete(book:Book) {
        bookRepository.delete(book)
    }

    fun getBooksByAuthorOrBook (searchString: String) : LiveData<List<Book>>
{
        return bookRepository.getBooksByAuthorOrBook(searchString)
    }
}
```

On exécute et on se rend compte que ca fonctionne bien !