# F.3 – MVVM Testing

## Model

L’obbiettivo del test sarà di mostrare che le operazioni di MoviesRepository funzionano correttamente.

MoviesRepository dipende però da alcune data sources:

A picture containing text, screenshot, font

Description automatically generated

Rimuovi moviesDataSource, dato che ad ora è inutilizzata.

Per movieRemoteDataSource creeremo un **fake**, un tipo di test double che riproduce risultati stabili senza interrogare il server di TMDB.

### Fake

Crealo dentro un nuovo package *test* chiamato *data*

A screen shot of a computer

Description automatically generated with low confidence

A screen shot of a computer program

Description automatically generated with medium confidence

* *fakeData* farà le veci della risposta del servizio

### Test

Ora crea il setup per il test di MoviesRepository

A screenshot of a computer program

Description automatically generated with medium confidence

* All’inizio la annotation @ExperimentalCoroutinesApi, che ci permetterà di utilizzare strumenti per le coroutines.
* Creiamo il repository immettendo nel costruttore fakeDataSource. Avremo cosi’ una risposta controllata del “servizio”, permettendo di focalizzare i test sulle funzionalità di Repository.

A picture containing text, screenshot, font, line

Description automatically generated

getMoviesResponse è però una suspend function, che può essere lanciata solo in una coroutine o altre sus fun. Utilizziamo quindi il tool delle coroutines *runTest*, che ci permette di eseguire suspend function in maniera *sincrona*

A picture containing text, screenshot, font

Description automatically generated

## ViewModel

MovieListViewModel dipende da MoviesRepository



Andiamo a creare un Fake di MoviesRepository.

### Interfaccia

Quando abbiamo creato il fake di movieDataSource ci siamo appoggiati ad una interfaccia, faremo la stessa cosa per il repo.

Prima, rinomina il repo in **MoviesRepositoryImpl**

Poi estrai un’interfaccia dal repository:

A screenshot of a computer

Description automatically generated

Estraila in un nuovo file, dentro il package *data*, chiamato MoviesRepository. Fai attenzione a selezionare tutti i membri del repo.

A screenshot of a computer

Description automatically generated

Ora c’è una nuova interfaccia, che il nostro repo implementa:

A screenshot of a computer program

Description automatically generated with medium confidence

### Fake

Crea *FakeMoviesRepository* dentro test/data. Questo repo deve implementare la nuova interfaccia e tutti i relativi membri.

A screen shot of a computer

Description automatically generated with medium confidence

Copia e aggiungi fakeData da FakeMovieDataSource e usalo per il risultato di getMoviesResponse

A screen shot of a computer code

Description automatically generated with low confidence

Abbiamo copiato fakeData dentro il nostro fakeRepo perché cosi’ questo non sarà dipendente da FakeMovieDataSource.

### Test

Per prima cosa modifichiamo il costruttore il MovieListViewModel per integrare la nuova interfaccia:



Rendi movieCache pubblico, ci servirà per i test. Aggiungi l’annotazione @VisibleForTesting per lanciare warning se viene utilizzato da pubblico al di fuori dell’ambito dei test.



Poi crea un test, selezionando i metodi da testare:

Screens screenshot of a computer

Description automatically generated with medium confidence

Spostalo dentro un nuovo package *viewModels*

A picture containing text, screenshot, font

Description automatically generated

Effettua il setup, inserendo le annotations e i membri di test

A screenshot of a computer program

Description automatically generated with medium confidence

Ora puoi testare le funzioni come al solito. Se dovessi testare *suspend fun* utilizza *runTest,* ma ricordati di aggiungere l’annotazione *@ExperimentalCoroutinesApi*

## Esercizio

Completa i test di MovieListViewModelTest.

Per getMovies i test devono controllare:

* L’ordinazione della lista prima della pubblicazione. Aggiungi dati a fakeData.
  + Per controllare la lista film, assicurati di prendere come riferimento fakeRepo.fakeData, non movieCache.
* la corretta interazione con movieCache. Devono inoltre venir effettuati test sia in caso che la cache parta da una lista vuota, che da una lista piena.

Per getMovieById:

* Ci sono dei casi in cui questa funzione produce eccezione. Poniti come obbiettivo di fare in modo che non produca mai eccezione.
* **Lavora** **facendo** **prima i test** per i risultati che dovrebbe produrre questa funzione  
  **e** **dopo modifica la funzione** per fare in modo che abbia le funzionalità desiderate e passi i test.
  + Così starai lavorando con la filosofia **Test Driven Development (TDD)**
* Per testare correttamente questa funzione serviranno almeno 3 test.