JURNAL SISTEM INFORMASI DAN TEKNOLOGI

http://jurnal.umk.ac.id/index.php/sitech

IMPLEMENTASI TEST DRIVEN DEVELOPMENT DALAM PENGEMBANGAN APLIKASI BERBASIS WEB

[1]Afandi Nur Aziz Thohari, [2]Andika Elok Amalia

[1] Program Studi S1 RPL, Fakultas Teknologi Industri dan Informatika Institut Teknologi Telkom Purwokerto
[2] Program Studi S1 RPL, Fakultas Teknologi Industri dan Informatika Institut Teknologi Telkom Purwokerto

Article Info:
Dikirim: Mei 2018
Diterima: Mei 2018
Tersedia Online: Juni 208

Penulis Korespondensi:
Afandi Nur Aziz Thohari
Fakultas Teknologi Industri dan Informatika, Program Studi S1 Rekayasa Perangkat Lunak, Institut Teknologi Telkom Purwokerto
Email: afandi@ittelkom-pwt.ac.id

Abstrak: Sebuah perangkat lunak dikatakan siap untuk dipakai apabila sudah melalui tahap pengujian. Pada era pengembangan perangkat lunak dengan metodologi tradisional, pengujian dilakukan dengan cara mencoba satu persatu menu aplikasi ketika aplikasi yang dikembangkan sudah jadi. Cara pengujian tersebut akan membutuhkan waktu yang lama apabila *developer* mengerjakan proyek perangkat lunak dalam skala besar. Selain itu, cara tersebut juga tidak dapat menguji logika dan *method* dari suatu kelas. Salah satu metode pengembangan perangkat lunak yang dapat menghemat waktu pengujian, namun fungsionalitasnya tetap terjaga adalah *test driven development* (TDD). Pada metode TDD, pengembangan perangkat lunak dilakukan dengan membuat *test case* terlebih dahulu baru kemudian melakukan *producing code*. Pada penelitian ini, dikembangkan sebuah aplikasi web menggunakan TDD. Aplikasi web yang dikembangkan adalah berupa sistem informasi mengenai ulasan film lokal indonesia atau disebut *Indonesia Movie Database* (IMDB). Bahasa pemrograman web yang dipakai adalah ruby dengan menggunakan *framework* rails. Sedangkan alat yang dipakai untuk pengujian adalah Rspec. Hasil implementasi TDD membuktikan bahwa fungsi-fungsi dari aplikasi web yang dibangun dapat berkerja dengan baik. Selain itu kode program yang dihasilkan juga menjadi rapi dan mudah dibaca oleh pengembang lain karena menerapkan *refactoring*. Pengujian *unit test* menggunakan Rspec membantu pengembang dalam menangani kesalahan dan memudahkan menambah fitur baru dari aplikasi web.

Kata kunci: perangkat lunak; pengujian; TDD; aplikasi web.

Abstract: *A Software is said ready to use when it has been through the testing phase. In the era of software development with traditional methodology, testing is done by trying one by one menu on the application when developed application is finished. Those testing method will take a long time if developer work on large scale software project. Beside that, the prosess also can't test logic and method from the class. One method of software development that can save testing time, but it's functionality still maintained is test driven development (TDD). In the TDD method, software development is done by creating a test case first then do producing code. This research developed a web application using TDD. Web application that had been developed is an information system about local films in Indonesia or called Indonesia Movie Database (IMDB). Programming Language that used is ruby using rails framework. Tool that used for tersting is Rspec. The result of TDD implementation proves that the functions of web application can work well. In additon, program code that generated also become tidy and easy to read by other developers because implementating refactoring program. Unit test testing using Rspec helps developer in handling errors and makes it easy to add new features of web apps.*

Keywords: *software; testing; TDD; web application.*

1. PENDAHULUAN

Pengembangan perangkat lunak dewasa ini telah bergeser dari yang mulanya menggunakan metodologi tradisional menjadi metodologi baru yang disebut agile. Metode agile memungkinkan pengembang menyelesaikan perangkat lunak dalam jangka waktu singkat, namun tetap dapat beradaptasi terhadap adanya perubahan dalam bentuk apapun. Salah satu model dalam metode agile adalah *Test Driven Development* (TDD). TDD merupakan sebuah strategi pengembangan perangkat lunak yang langkah pengerjaannya adalah dengan membuat *test case* terlebih dahulu baru kemudian dilakukan *producing code* dan *refactoring* [1].

Metode TDD mengharuskan pengembang untuk menentukan terlebih dahulu kebutuhan dan desain dari perangkat lunak yang akan dibuat baru kemudian melakukan implementasi kode. Satu poin penting dalam metode TDD adalah pengembangan perangkat lunak fokus pada spesifikasi bukan pada validasinya [2]. Sebab pada TTD validitas dari kode program sudah pasti terjamin saat sistem selesai dibuat. Sehingga akan mempercepat waktu pengembangan program ketika mengerjakan proyek dalam skala besar.

Proses pengembangan perangkat lunak akan menghasilkan suatu produk yang berkualitas apabila terjadi sinkronisasi antar analis dan pemrogram[3]. Sebuah perangkat lunak dikatakan berkualitas apabila perangkat lunak dapat bekerja sesuai apa yang diinginkan pengguna (*user requirement* dan *bussines proces*). Selain itu juga dapat memudahkan pengguna dalam memodifikasi perubahan, serta memudahkan dalam menemukan *bug*. Semua spesifikasi tersebut dapat diakomodai oleh metode TTD. Sebab pada metode TDD terdapat *unit test* yang berperan untuk menjaga agar pengembang tetap *stay on track* ketika membuat sebuah perangkat lunak. Sehingga memungkinkan hasil perangkat lunak yang dibuat akan sesuai dengan kebutuhan dan proses bisnis pengguna.

Sebagai contoh, terdapat beberapa penelitian yang membahas mengenai implementasi metode TDD dalam pembuatan aplikasi. Salah satu penelitian tersebut adalah penelitian yang dilakukan oleh [4] yang melakukan implementasi TDD untuk pengembangan sisten informasi rawat jalan. Kemudian implementasi TDD juga dilakukan dalam pembuatan perangkat lunak pengajuan tugas akhir [5]. Hasil yang diperoleh menunjukan bahwa dengan menerapakan TDD pengecekan *error* lebih mudah dilakukan. Selain itu pengembang dapat bekerja lebih cepat karena simulasi manual untuk pengujian fungsi perangkat lunak menjadi berkurang.

Pada penelitian ini dibahas tentang implementasi metode TDD dalam pengembangan aplikasi web. Aplikasi web yang dibangun adalah aplikasi yang beisi informasi mengenai film-film lokal di Indonesia atau disebut *Indonesia Movie Database* (IMDB). Aplikasi IMDB dibangun menggunakan suatu web *framework* bernama Rails dan kodenya ditulis dengan bahasa Ruby.

Sebelum melakukan implementasi sistem, terlebih dahulu ditentukan fitur-fitur dari aplikasi IMDB ke dalam *unit test*. Kemudian setiap *unit test* akan diselesaikan dan diuji validitasnya. Pengujian validitas *unit test* dilakukan dengan *unit test framework* bawaan Ruby yaitu Rspec. Setelah berhasil diuji, langkah selanjutnya adalah *Refactoring* program agar tidak terjadi *redudancy code* sehingga lebih mudah untuk melakukan perawatan program. Penggunaan TTD ini dapat menjadi alternatif pengganti metode tradisional, ditinjau dari kecepatan pengembangan yang lebih cepat.

2. METODOLOGI PENELITIAN

Metodologi yang digunakan dalam penelitian ini adalah metodologi kualitatif. Dimana menurut [6] metodologi penelitian kualitatif adalah suatu metode yang digunakan untuk memahami fenomena tentang yang dialami oleh objek penelitian. Pada metodologi kualitatif tidak terdapat perhitungan matematika dan statistik.

2.1 Pengumpulan Data

Metode pengumpulan data pada penelitian ini dilakukan dengan melakukan pengamatan langsung terhadap website-website resmi seperti broadway.com dan imdb.com. Tujuan pengamatan tersebut adalah untuk melakukan kajian tentang kebutuhan pengguna dan proses bisnis yang berjalan pada sebuah sistem rating dan review film.

2.2 Analisis Data

Berdasarkan hasil pengamatan pada tahap pengumpulan data, dilakukan analisa untuk menentukan kebutuhan pengguna dan proses bisnis dari aplikasi IMDB yang dibuat. Agar lebih memudahkan pengembang dalam bekerja, analisis kebutuhan sistem tersebut diimplementasikan dalam bentuk *user stories.* Secara mendasar *user stories* tidak memiliki rumus baku dalam pembuatannya. Namun demikian,

terdapat rumus yang banyak diadopsi oleh tim Agile, yaitu dengan menggunakan rumus **As-I-So** sebagai berikut [7].

As [an actor] I want to [action] so that [benefit]

Selanjutnya berdasarkan *user stories* tersebut, maka dirancanglah sebuah *rule* untuk membuat *unit test*. Setiap *unit test* akan diuji validitasnya dengan menggunakan *tools* yang berjalan pada *framework* rails yaitu Rspec. Adapun *user stories* dari aplikasi IMDB, ditunjukan pada Tabel 1.

Tabel 1. *User stories* dari aplikasi IMDB

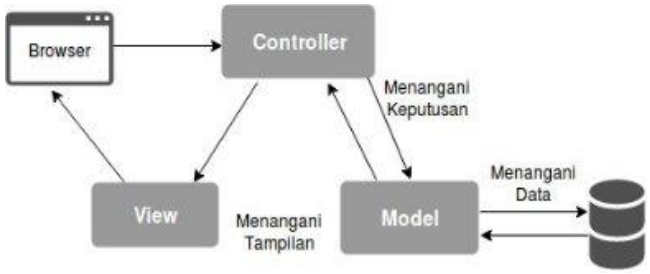
No	*As {Actor)*	*I Want to {Action}*	*So that {Object)*
MV-001	Admin, member	Menambah daftar film (*create post*)	Admin dan member dapat menambah daftar film.
MV-002	Admin	Mengubah dan menghapus daftar film yang dibuat oleh member	Admin dapat mengedit dan menghapus postingan film yang dibuat oleh semua member
MV-003	Admin, member	Memberikan komentar dan rating dari sebuah film	Admin dan member dapat memberikan komentar dan rating dari sebuah film yang telah di posting member lain
MV-004	Admin	Mengubah dan menghapus komentar dari member	Admin dapat menertipkan komentar dari member yang mengganggu
MV-005	Admin, member	Melihat daftar film sesuai kategori	Admin dan member dapat memilih kategori film yang ingin di review
MV-006	Member	Mengubah dan menghapus daftar film yang dibuat	Member dapat mengubah dan menghapus daftar film yang telah member buat sendiri
MV-007	Member	Mengubah dan menghapus komentar yang dibuat	Member dapat mengubah dan menghapus komentar yang telah dibuat
MV-008	Pengunjung	Mendaftar aplikasi IMDB	Pengunjung dapat mendaftar menjadi member dari IMDB

2.3 Desain Sistem

Pada tahap ini akan dilakukan penentuan arsitektur sistem, desain model proses dan model data. Arsitektur sistem yang dikembangkan pada aplikasi web ini adalah MVC. Desain model proses dari aplikasi web digambarkan menggunakan *United Modelling Language* (UML). Sedangkan untuk model data digambarkan dengan relasi antar tabel dalam basis data. Masing-masing tahap pada desain sistem, dijelaskan sebagai berikut

2.3.1 Perancangan Arsitektur Sistem

Seperti pada umumnya framework pengembangan web, Rails mengabdopsi model arsitektur MVC. Cara kerja dari model MVC ini yaitu memisahkan data (*Model*) dari tampilan (*View*) dan cara mengaksesnya (*Controller*). Arsitektur dari MVC pada sebuah pengembangan web, ditunjukan pada Gambar 1.



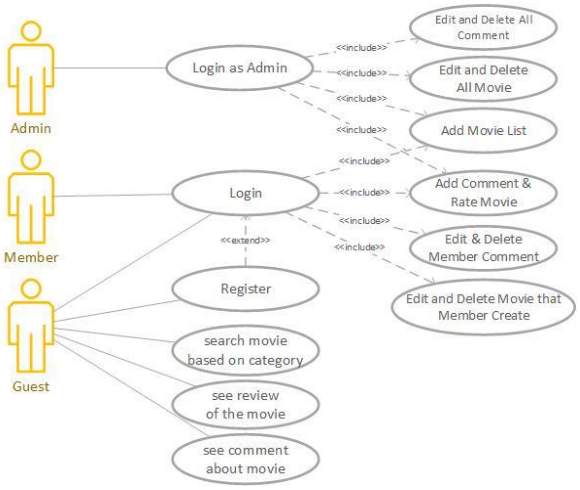
Gambar 1. Arsitektur MVC Pada Aplikasi Web [8]

Penjelasan mengenai arsitektur MVC pada Gambar 1 adalah sebagai berikut

a. Pengguna melakukan permintaan untuk mengakses aplikasi web ke browser.
b. Setiap permintaan yang diberikan oleh pengguna akan ditangani oleh Controller.
c. Kemudian dari Controller akan dilanjutkan ke Model.
d. Saat mengakses Model, dilakukan pengecekan mengenai kebutuhan basis data.
e. Jika permintaan membutuhkan basis data, maka Model akan mengambil data dari basis data. Namun jika tidak maka dari Controller akan langsung ke proses View.
f. Informasi dari Model dikembalikan ke Controller, kemudian Controller melanjutkan ke *state* View untuk ditampilkan di browser.
g. View menampilkan antarmuka pengguna dan juga informasi yang telah diperoleh dari basis data.

2.3.2 Perancangan Diagram Use Case

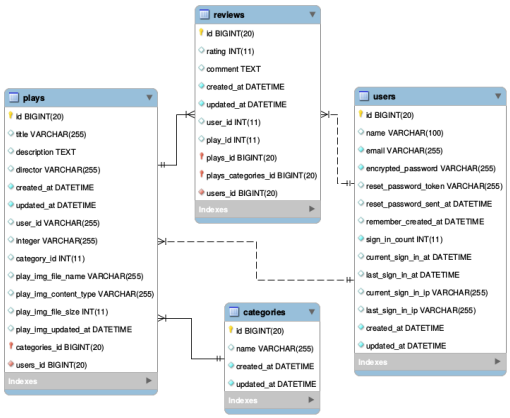
Pada tahap ini, ditentukan sebuah alur proses dari masing-masing pengguna. Setiap pengguna memiliki fitur-fitur tertentu dalam mengakses aplikasi IMDB yang dibuat. Terdapat tiga pengguna yang berperan dalam menggunakan aplikasi IMDB ini. Adapun diagram *use case* dari aplikasi IMDB, ditunjukan pada gambar 2.



Gambar 2. Diagram *Use Case* Aplikasi IMDB

Gambar 2 menunjukan bahwa terdapat tiga pengguna yang dapat mengakses aplikasi IMBD. Setiap pengguna memiliki hak akses yang berbeda-beda, tergantung dari levelnya. Pada level pengunjung, pengguna hanya memiliki akses untuk melihat daftar film, melihat review film dan membaca komentar dari member. Ketika pengunjung mendaftar sistem, maka status nya naik menjadi member. Pada level member, pengguna dapat menambahkan daftar film, menambahkan komentar dan memberikan *rating* film. Sedangkan pada level yang paling tinggi yaitu admin, pengguna dapat mengedit dan menghapus data film serta komentar-komentar yang dibuat oleh member.

2.3.3 Perancangan Model Data



Gambar 3. Relasi Antar Tabel Pada Aplikasi IMDB

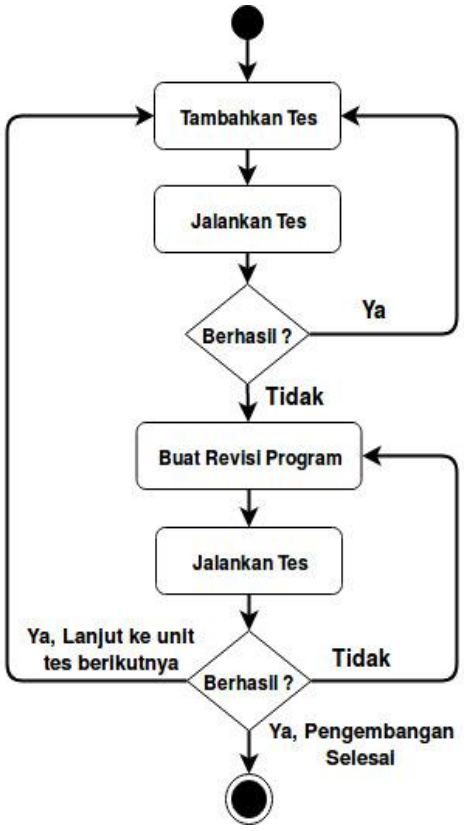
Relasi antar tabel pada aplikasi IMDB ditunjukan pada gambar 3. Aplikasi IMDB yang dibangun memiliki 4 buah tabel yaitu tabel play yang berisi deskripsi dari film, tabel users yang berisi deskripsi dari pengguna, tabel categories berisi kategori (*genre*) dari film, dan tabel reviews berisi keterangan review yang diberikan oleh pengguna. Masing-masing tabel tersebut saling terintegrasi satu sama lain. Relasi antar tabel ini juga diimplementasikan ke dalam kode program, sebagai contoh pada berkas play.rb yang isinya sebagai berikut.

```
class Play < ApplicationRecord
      belongs_to :category
      belongs_to :user
      has_many :review
end
```

Maksud pada kode diatas adalah bahwa kelas plays berelasi dengan kelas category, user, dan review. Kelas-kelas tersebut berada di direktori model yang digunakan untuk mengatur hubungan relasi antar tabel dalam basis data. Sebagai contoh relasi yang terjadi antara tabel plays dan tabel categories adalah *many to one* yang artinya satu kategori dapat memiliki banyak daftar film. Kemudian relasi antara tabel plays dan users yang terjalin adalah *many to one* karena satu pengguna dapat membuat lebih dari satu daftar film. Terakhir relasi antar tabel plays dengan tabel reviews yaitu *one to many*, sebab satu film dapat memiliki lebih dari satu review.

2.4 Implementasi

Implementasi program dilakukan dengan menerapkan metode *Test First Development* (TFD). Adapun langkah-langkah pada metode TFD ditunjukan seperti gambar 4.

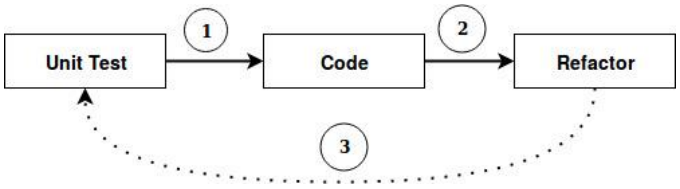


Gambar 4. Alur Kerja TFD

Berdasarkan gambar 4, hal pertama yang dilakukan ketika menerapkan metode TFD adalah membuat *unit test* hasil turunan dari *user requirement*. Saat pertama kali tes dijalankan hasilnya akan *error,* sebab belum terdapat kode apapun yang membuat tes menjadi terpenuhi. Oleh sebab itu, tugas pengembang adalah membuat kode sampai tes yang dibuat berhasil terpenuhi. Pengembang akan terus melakukan revisi kode program, dan apabila tes pertama sudah berhasil terpenuhi maka akan berganti ke tes-tes berikutnya sesuai fungsinya masing-masing. Apabila sumua unit test sudah berhasil dijalankan, maka pengembangan sistem selesai dibangun.

2.5 Refactoring

Pada metode TFD, kode yang dihasilkan biasanya acak-acakan atau kurang rapi. Oleh sebab itu dilakukan proses *refactoring* agar kode lebih mudah di *maintenace* dan dipahami oleh pengembang lain. Proses *refactoring* dilakukan dengan memperhitungkan hasil pengujian agar tetap *passed* artinya fitur tetap dapat bekerja sebagaimana mestinya (sesuai spesifikasi). Sehingga dapat dikatakan bahwa TTD adalah gabungan antara TFD dengan *refactoring*, sebagaimana ditunjukan pada gambar 5.



Gambar 5. Alur Kerja TDD

Penjelasan gambar 5 adalah sebagai berikut

a. Pengembang menuliskan *unit test* terlebih dahulu, sesuai spesifikasi dan bisnis proses dari sistem yang diingikan oleh pengguna.
b. Setelah membuat *unit test*, pengembang menuliskan kode program untuk menyelesaikan *unit test* sampai *pass* (tidak terjadi *error*).
c. Pengembang melakukan *refactoring* kode program yaitu mengubah struktur program agar lebih mudah dipahami dan dimodifikasi, tanpa harus mengubah dari *behaviour* program.

3. HASIL DAN PEMBAHASAN

Luaran dari penelitian ini menghasilkan sebuah aplikasi web yang digunakan untuk memberikan rating dan review terhadap film-film Indonesia. Model pengembangan yang digunakan pada penelitian ini menggunakan metode *Test Driven Development.* Siklus pengembangan aplikasi yang dibangun dimulai dengan melakukan pengujian terlebih dahulu, kemudian melakukan implementasi program, dan terakhir melakukan *refactoring*.

3.1 Implementasi Pengujian Unit

Berdasarkan *user stories* yang telah dibuat, maka dirancang beberapa *unit test* dari aplikasi IMDB. Validasi dari *unit test* dilakukan dengan menggunakan sebuah *tools* bernama Rspec. Pemasangan Rspec pada *framework* Rails dilakukan dengan menambahkan sebuah sebuah kode pada berkas Gemfile sebagai berikut.

```
group :development, :test do
  gem 'rspec-rails'
end
```

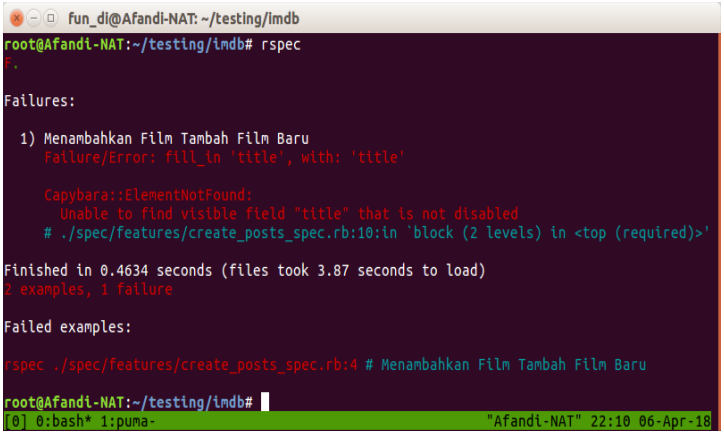
Kemudian pada teminal, lakukan pemasangan Rspec dengan mengetikan perintah `rails generate rspec:install`. Apabila pemasangan Rspec berhasil, maka akan muncul direktori bernama ‘spec’. Selanjutnya di dalam direktori spec, tambahkan direktori baru bernama ‘features’. Pembuatan *unit test* dilakukan di dalam direktori ‘features’ dengan cara menambahkan berkas berisi kode dalam bahasa ruby. Adapun contoh *unit test* yang ditulis menggunakan bahasa ruby, ditunjukan seperti kode berikut.

```
require ‘rails_helper.rb’
feature ‘Membuat Daftar Film’ do
    scenario ‘Dapat Menambahkan Daftar Film’ do
       visit ‘/’
       click_link ‘Tambahkan Film’
       fill_in ‘Judul’, with: ‘judul’
       fill_in ‘Deskripsi’, with: ‘deskripsi’
       click_button ‘Tambah Film’
       expect(page).to have_content(‘judul’)
       expect(page).to have_content(‘deskripsi’)
    end
end
```

Kode diatas adalah salah satu penerapan *unit test* dengan Rspec. Penulisan *unit test* pada Rspec dilakukan dengan cara menentukan terlebih dahulu fitur apa yang akan dibuat. Setelah itu menentukan skenario (langkah kerja) dari sebuah fitur. Kode diatas digunakan untuk menambahkan judul dan deskripsi film, dimana skenarionya adalah sebagai berikut.

a. Pada halaman beranda akan muncul tautan 'Tambah Film'
b. Ketika tautan 'Tambahkan Film' ditekan, maka akan diarahkan ke halaman tertentu yang berisi form untuk menambahkan judul dan deskripsi film.
c. Isikan judul dan deskripsi film, kemudian tekan tombol 'Tambah Film'.
d. Setelah tombol 'Tambah Film' ditekan, maka judul dan deskripsi yang telah *submit* akan ditampilkan.

Unit test yang telah dibuat, kemudian diuji validitasnya. Pengujian validitas dilakukan dengan mengetikan 'rspec' pada lokasi projek melalui terminal. Gambar 6 menunjukan contoh pengujian *unit test* yang gagal.



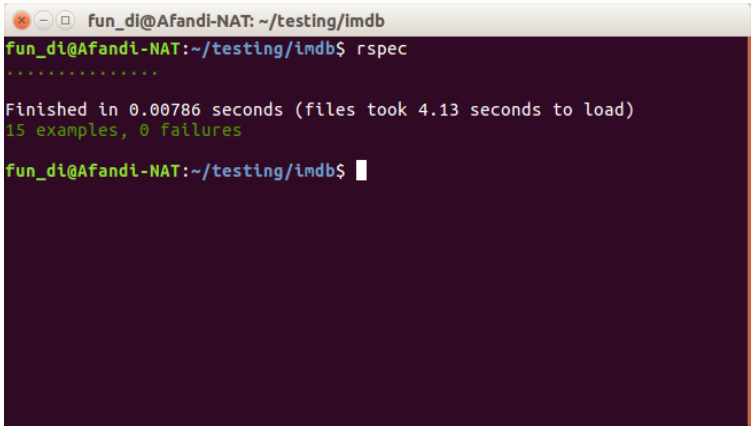
Gambar 6. Pengujian Unit dengan Rspec

Pengujian yang gagal akan ditunjukan dengan tulisan merah disertai pesan kesalahannya. Apabila pengujian gagal maka secara otomatis Rspec akan menunjukan bagian mana yang terjadi *error*. Rspec menunjukan penyebab kesalahan, nama berkas yang salah, serta letak baris kode yang salah. Selanjutnya pengembang tinggal memperbaiki sesuai pesan kesalahan yang ditampilkan, sampai kode berhasil dieksekusi.

Berdasarkan *user stories* yang telah dibuat pada Tabel 1, dihasilkan 15 *unit test* pada aplikasi IMDB ini. Fitur-fitur tiap *unit test* ditunjukan pada Tabel 2. Setiap *unit test* telah berhasil diuji menggunakan Rpsec, tampilkan pengujian *unit test* yang berhasil di eksekusi ditunjukan pada Gambar 7.

Tabel 2. Fitur dari masing-masing *unit test*

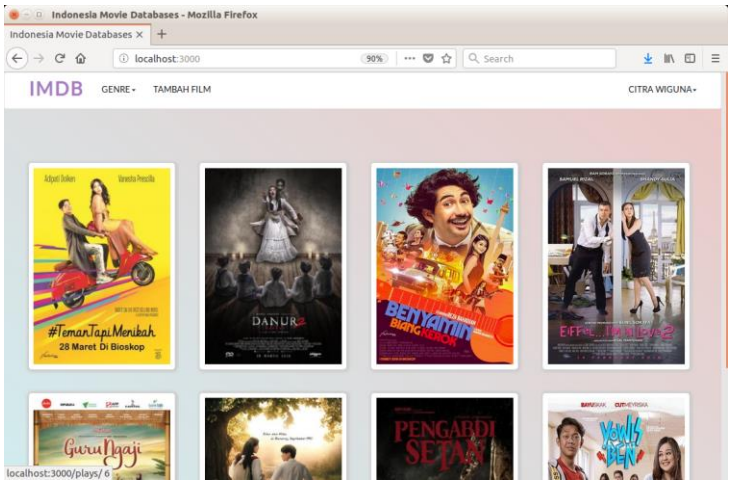
No.	***Fitur***	***Hasil***
1	*Create post*	Berhasil
2	*Display post*	Berhasil
3	*Edit and delete post*	Berhasil
4	*Add user to App*	Berhasil
5	*Navigation and model association*	Berhasil
6	*Categories for the movie*	Berhasil
7	*Edit page dropdown and navbar dropdown*	Berhasil
8	*Filtering with category*	Berhasil
9	*Image uploading*	Berhasil
10	*Display uploaded image*	Berhasil
11	*Add reviews*	Berhasil
12	*Display reviews*	Berhasil
13	*Edit and delete reviews*	Berhasil
14	*Add 5 star rating system*	Berhasil
15	*Average rating*	Berhasil



Tabel 7. Total Pengujian Unit dari Aplikasi IMDB

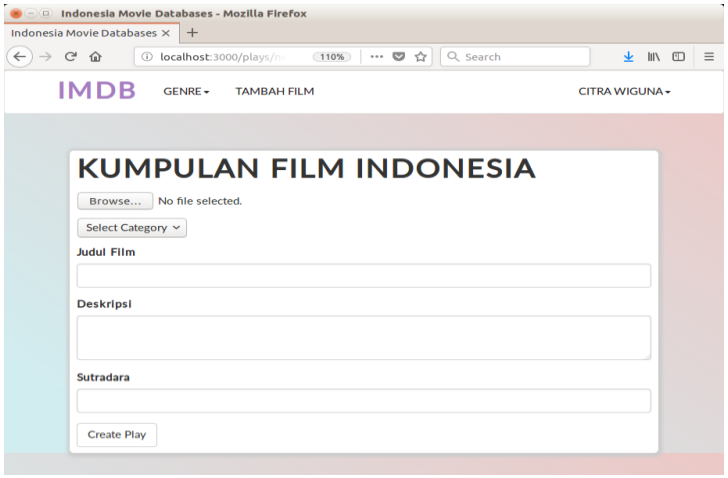
3.2 Tampilan Antarmuka Sistem

Tampilan antarmuka sistem setelah melalui serangkaian tes dan *production code*, ditunjukan pada Gambar 8 – 10. Ketika tes berhasil dijalankan (*pass*) maka dapat dipastikan bahwa fitur yang dikehendaki dapat bekerja dengan baik. Agar lebih memudahkan pengguna dalam menggunakan aplikasi (*user friendly*) digunakan bootstrap dan css untuk mempercantik tampilan.



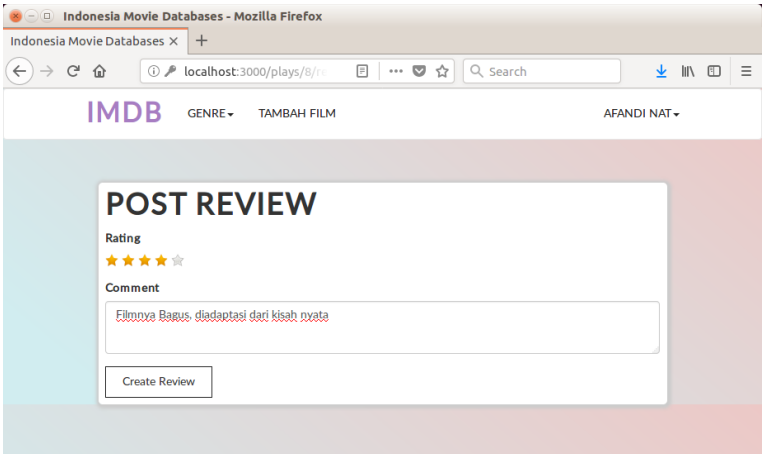
Gambar 8. Tampilan Antarmuka Halaman *Home*

Gambar 8 menunjukan tampilan antarmuka pada halaman beranda aplikasi IMDB. Pada halaman beranda, pengguna disuguhkan daftar film-film terbaru yang telah dimasukan admin atau member. Pengguna dapat mengelompokan film berdasarkan pada *genre* atau kategori.



Gambar 9. Tampilan Antarmuka Halaman *Add Movie*

Ketika pengunjung melakukan registrasi dan tercatat sebagai member IMDB, maka pengunjung memiliki hak akses untuk menambah daftar film. Gambar 9 merupakan tampilan antarmuka pada halaman tambah daftar film.

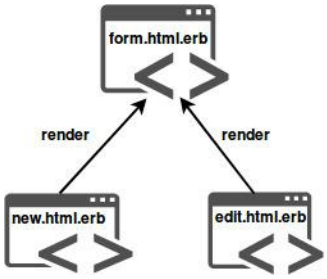


Gambar 10. Tampilan Antarmuka Halaman Tambah Komentar Dan Rating

Salah satu keuntungan menjadi member adalah dapat memberikan komentar dan rating terhadap sebuah film. gambar 10 menunjukan bahwa member dapat menuliskan komentar dan rating berupa pemberian bintang. Nilai maksimal rating memiliki bobot 5. Pada tampilan review film semua bobot rating yang diberikan oleh pengguna akan di rata-rata dan disajikan dalam tanda bintang.

3.3 Refactoring Program

Refactoring bertujuan untuk menghasilkan *clean code* yang mudah dibaca, tidak mengandung duplikasi, dan mudah dilakukan *maintenance.* Menurut [9] dalam bukunya menerangkan bahwa cara untuk memenuhi *deadline* dan bekerja dengan cepat adalah sebisa mungkin menjaga agar tercipta *clean code.* Oleh karena itu, pada aplikasi IMDB ini dilakukan *refactoring* kode program seperti ditunjukan pada gambar 11.



Gambar 11. Penerapan R*efactoring* Pada Modul

gambar 11 menunjukan bahwa berkas form.html.erb berisi *template* yang digunakan bersama oleh new.html.erb dan edit.html.erb. Sehingga isi dari berkas new.html.erb dan edit.html.erb hanya berisi satu baris kode yaitu render ‘form’. Penghematan listing penulisan kode ini dapat meningkatkan efisiensi dari pengembang dalam membangun aplikasi IMDB.

4. KESIMPULAN

Berdasarkan hasil perancangan dan implementasi terhadap metode TDD dapat disimpulkan bahwa fungsi-fungsi dari aplikasi web yang dibangun dapat berjalan dengan baik. Pengembang tidak perlu melakukan pengujian aplikasi dengan cara mencoba menu satu persatu. Penerapan metode TDD membantu pengembang dalam menerapkan proses bisnis dan kebutuhan dari pengguna aplikasi IMDB. Selain itu juga memudahkan pengembang menemukan kesalahan ketika mengimplementasikan kode program.

Adanya *refactoring* pada metode TDD membuat kode program yang dihasilkan lebih rapi. Pengembang yang bekerja secara tim dapat dengan mudah memahami dan memodifikasi kode setelah

melalui tahap *refactoring*. Pengujian *unit test* menggunakan Rspec membantu pengembang dalam menangani kesalahan dan memudahkan menambah fitur baru dari aplikasi web.

Selain beberapa kemudahan yang telah diuraikan, penerapan aplikasi web dengan TDD ini juga memiliki beberapa kesulitan yang terjadi selama *development*. Salah satu kesulitan dalam penerapan metode TDD ini adalah ketika mengkonversikan kebutuhan pengguna ke dalam *unit test.* Terkadang *unit test* yang telah dibuat tidak sesuai dengan proses bisnis, sebab tidak ada ukuran yang menunjukan bahwa *unit test* yang dibuat telah benar. Kemudian dikarenakan TDD hanya menguji fungsionalitas (*back-end*) dari aplikasi web, maka masalah tampilan (*front-end*) dari aplikasi web dilakukan secara manual.

DAFTAR PUSTAKA

[1] Beck, K. (2003). *Test-Driven Development: By Example*. Boston : Addison-Wesley.
[2] Bissi, W., Neto, A.G.S.S., Emer, M.C.F.P. 2016. "The Effect of Test Driven Development on Internal Quality, External Quality and Production : A Systematic Review", in *Information and Software Technology*, pp. 2 – 35.
[3] Setiawan, A., Satoto, K.I., Isnanto, R.R. 2013. "Implementasi Automated Unit dan Integration Testing Pada Pengujian Perangkat Lunak". *TRANSIENT*. Vol.2, No.3, 708-713.
[4] Rachmadi, G. 2015. "Implementasi Test Driven Development Dalam Studi Kasus Pengembangan Sistem Informasi Rawat Jalan Di Rumah Sakit Hewan Universitas Airlangga". *Skipsi*. Universitas Airlangga, Surabaya.
[5] Firmansyah, R. 2016. "Pengembangan Perangkat Lunak Pengajuan Sidang Tugas Akhir Menggunakan Meodologi Test Driven Development". *Skripsi*. Universitas Pasundan, Bandung.
[6] O'Rourke, C., and Fishman, N. (2003). *Enterprise Architecture Using the Zachman Framework*.
[7] Chon, M. 2004. *User Stories Applied: For Agile Software Development*. Boston : Addison-Wesley.
[8] Verma, A. 2014. "MVC Architecture : A Comparative Study Between Ruby On Rails and Laravel." *Indian Journal of Computer Science and Engineering (IJCSE)*, Vol.5, No.5, 196 -198.
[9] Martin, R. C. (2008). *Clean Code: A Handbook of Agile Software Craftmanship*. Boston : Pearson..