SKRIPSI

PEMBANGKIT TIMELAPSE PENGEMBANGAN PROYEK PERANGKAT LUNAK BERBASIS WEB

Billy Adiwijaya

NPM: 2015730053

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI DAN SAINS
UNIVERSITAS KATOLIK PARAHYANGAN
2019

UNDERGRADUATE THESIS

TIMELAPSE GENERATOR FOR WEB-BASED SOFTWARE PROJECT DEVELOPMENT

Billy Adiwijaya

NPM: 2015730053

DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY AND SCIENCES
PARAHYANGAN CATHOLIC UNIVERSITY
2019

LEMBAR PENGESAHAN

PEMBANGKIT TIMELAPSE PENGEMBANGAN PROYEK PERANGKAT LUNAK BERBASIS WEB

Billy Adiwijaya

NPM: 2015730053

Bandung, «tanggal» «bulan» 2019

Menyetujui,

Pembimbing

Pascal Alfadian, M.Comp.

Ketua Tim Penguji	Anggota Tim Penguji
«penguji 1»	«penguji 2»

Mengetahui,

Ketua Program Studi

Mariskha Tri Adithia, P.D.Eng

PERNYATAAN

Dengan ini saya yang bertandatangan di bawah ini menyatakan bahwa skripsi dengan judul:

PEMBANGKIT TIMELAPSE PENGEMBANGAN PROYEK PERANGKAT LUNAK BERBASIS WEB

adalah benar-benar karya saya sendiri, dan saya tidak melakukan penjiplakan atau pengutipan dengan cara-cara yang tidak sesuai dengan etika keilmuan yang berlaku dalam masyarakat keilmuan.

Atas pernyataan ini, saya siap menanggung segala risiko dan sanksi yang dijatuhkan kepada saya, apabila di kemudian hari ditemukan adanya pelanggaran terhadap etika keilmuan dalam karya saya, atau jika ada tuntutan formal atau non-formal dari pihak lain berkaitan dengan keaslian karya saya ini.

Dinyatakan di Bandung,
Tanggal «tanggal» «bulan» 2019

Meterai
Rp. 6000

Billy Adiwijaya
NPM: 2015730053

ABSTRAK

«Tuliskan abstrak anda di sini, dalam bahasa Indonesia»

Kata-kata kunci: «Tuliskan di sini kata-kata kunci yang anda gunakan, dalam bahasa Indonesia»

ABSTRACT

«Tuliskan abstrak anda di sini, dalam bahasa Inggris»

Keywords: «Tuliskan di sini kata-kata kunci yang anda gunakan, dalam bahasa Inggris»

«kepada siapa anda mempersembahkan skripsi ini. . . ?»

KATA PENGANTAR

«Tuliskan kata pengantar dari anda di sini . . . »

Bandung, «bulan» 2019

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB 1

PENDAHULUAN

1.1 Latar Belakang

Git merupakan perangkat lunak *Version Control Systems*[1]. *Version control* adalah sistem yang merekam perubahan pada *file* atau sekumpulan *file* dari waktu ke waktu. Perubahan yang terjadi pada *repository* dicatat oleh Git dalam bentuk histori *commit*. Setiap *commit* mengandung informasi mengenai perubahan yang terjadi pada *repository*, waktu perubahan, dan orang yang melakukan perubahan. *Database* pada *git* tidak bersifat terpusat, melainkan terdistribusi. Setiap orang yang terlibat mempunyai *database* lokal pada masing-masing komputer, sehingga pengelolaan perangkat lunak dapat dilakukan secara *online* dan *offline*.

JGit adalah *library* Java murni yang mengimplementasikan Git *version control systems*[2]. JGit dikembangkan oleh Eclipse Foundation. JGit bersifat *open source*. Dengan menggunakan JGit, fitur-fitur dalam Git dapat diakses melalui program Java.

Selenium adalah seperangkat alat yang secara khusus digunakan untuk mengotomatisasi *web browsers*[3]. Dengan menggunakan Selenium WebDriver, pengguna dapat memasukkan *script* bahasa pemrograman tertentu untuk melakukan pengujian. Bahasa pemrograman yang didukung yaitu C#, Java, Perl, PHP, Python, Ruby, dan JavaScript. Selenium WebDriver dapat melakukan pengujian pada Google Chrome *browser*, Firefox *browser*, Opera *browser*, Internet Explorer *browser*, dan Microsoft Edge *browser*.

Pada skripsi ini, akan dibuat sebuah perangkat lunak yang dapat membangun animasi *timelapse* dari pengembangan proyek perangkat lunak berbasis *web*. Yang akan dibuat animasinya adalah halaman *web* dari perangkat lunak. Jumlah halaman web bisa lebih dari satu, tergantung pada masukan. Tujuan dari dibuatnya animasi atau visualisasi adalah melihat progres dari perangkat lunak. Sehingga *programmer* bisa belajar dari proses perkembangan perangkat lunak sebelumnya dan bisa membuat perkembangan perangkat lunak menjadi lebih efisien. Perangkat lunak ini dibangun menggunakan bahasa Java. Perangkat lunak ini menggunakan tampilan terminal/konsol. Dalam pembuatan animasi timelapse, dibutuhkan perangkat lunak Selenium WebDriver dan JGit.

1.2 Rumusan Masalah

Rumusan masalah dari skripsi ini adalah sebagai berikut:

1. Bagaimana cara membangkitkan animasi *timelapse* pada pengembangan proyek perangkat lunak berbasis web?

2. Bagaimana cara mengimplementasikan aplikasi untuk membangkitkan *timelapse* pada pengembangan proyek perangkat lunak berbasis web?

1.3 Tujuan

Tujuan dari skripsi ini adalah sebagai berikut:

1. Mengetahui cara untuk membangkitkan animasi *timelapse* pada pengembangan proyek perangkat lunak berbasis web.
2. Mengetahui cara untuk mengimplementasikan aplikasi untuk membangkitkan *timelapse* pada pengembangan proyek perangkat lunak berbasis web.

1.4 Batasan Masalah

Batasan masalah pada skripsi ini adalah sebagai berikut:

1. Perangkat lunak ini hanya membangkitkan animasi *timelapse* untuk perangkat lunak berbasis *web*.
2. Masukan perangkat lunak berupa alamat direktori proyek perangkat lunak yang terekam oleh Git.
3. Jumlah maksimal halaman *web* pada hasil animasi adalah empat halaman.
4. *Setup* perangkat lunak dilakukan secara otomatis melalui *script* PHP.

1.5 Metodologi

Metodologi penelitian yang digunakan dalam skripsi ini adalah sebagai berikut:

1. Melakukan studi literatur tentang Git, Selenium WebDriver, Git, dan JGit.
2. Melakukan analisis penggunaan Selenium WebDriver dan JGit untuk membangkitkan animasi timelapse.
3. Merancang perangkat lunak.
4. Membangun perangkat lunak.
5. Melakukan eksperimen dan pengujian pada perangkat lunak.

1.6 Sistematika Pembahasan

Setiap bab dalam penelitian ini memiliki sistematika penulisan yang dijelaskan ke dalam poin-poin sebagai berikut:

1. Bab 1: Pendahuluan, yaitu membahas mengenai gambaran umum penelitian ini. Berisi tentang latar belakang, rumusan masalah, tujuan, batasan masalah, metode penelitian, dan sistematika penulisan.
2. Bab 2: Dasar Teori, yaitu membahas mengenai teori-teori yang mendukung berjalannya penelitian ini. Berisi tentang teori Git, JGit, Selenium WebDriver, dan Apache Commons CLI.
3. Bab 3: Analisis, yaitu membahas mengenai analisa masalah. Berisi tentang analisis aplikasi sejenis, analisis penggunaan JGit dan Selenium WebDriver untuk membangkitkan animasi timelapse, prapengujian, dan analisis fitur aplikasi yang dibangun.
4. Bab 4: Perancangan, yaitu membahas mengenai perancangan yang dilakukan sebelum melakukan tahapan implementasi. Berisi tentang perancangan perangkat lunak pembangkit *timelapse* proyek pengembangan perangkat lunak.

5. Bab 5: Implementasi dan Pengujian, yaitu membahas mengenai implementasi dan pengujian aplikasi yang telah dilakukan. Berisi tentang implementasi dan hasil pengujian aplikasi.

6. Bab 6: Kesimpulan dan Saran, yaitu membahas hasil kesimpulan dari keseluruhan penelitian ini dan saran-saran yang dapat diberikan untuk penelitian berikutnya.

BAB 2

LANDASAN TEORI

Pada bab ini dibahas dasar teori yang mendukung berjalannya skripsi ini. Dasar teori yang dibahas yaitu Git, JGit, Selenium WebDriver, dan Apache Commons CLI.

2.1 Git

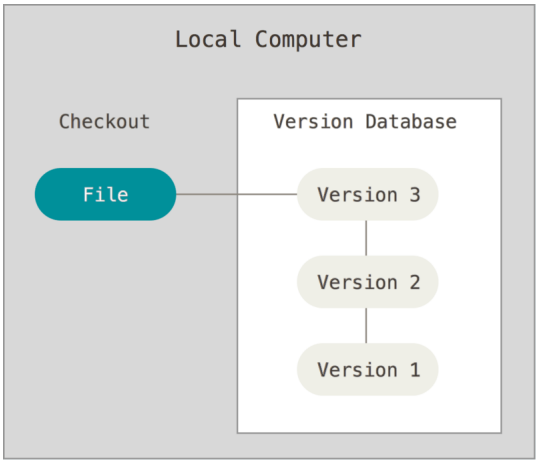
Git merupakan perangkat lunak *Version Control Systems*. Pada subbab ini, dijelaskan mengenai *Version Control Systems*, cara kerja Git, Git *checkout*, dan operasi-operasi dasar pada Git. Subbab ini mengacu pada [1].

2.1.1 Version Control Systems

Version Control Systems adalah sistem yang merekam perubahan pada *file* atau sekumpulan *file* dari waktu ke waktu. *Version Control Systems* biasanya digunakan untuk merekam file yang berisi *source code program*, tetapi pada kenyataannya *Version Control Systems* dapat merekam hampir semua jenis file dalam komputer. Terdapat tiga jenis *Version Control Systems*, yaitu: *Local Version Control Systems*, *Centralized Version Control Systems*, dan *Distributed Version Control Systems*.

Local Version Control Systems

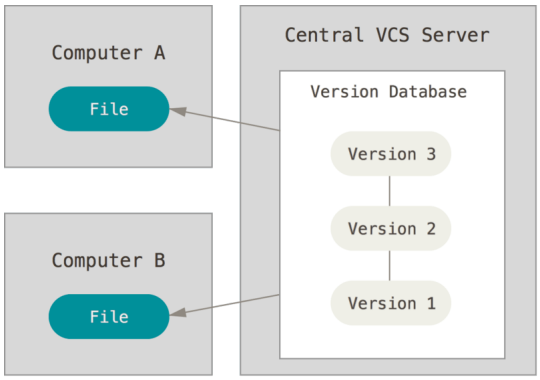
Metode *version-controlled* yang banyak digunakan orang adalah dengan cara menyalin sekumpulan *file* ke direktori lain. Namun cara tersebut rentan terhadap *error*. Misalnya, terdapat direktori A dan B, pengguna ingin mengubah *file* yang terdapat pada direktori B, tetapi pengguna lupa kalau dia sedang berada di direktori A, maka pengguna mengubah *file* pada direktori yang salah. Untuk mengatasi masalah tersebut, *programmer* mengembangkan *Local Version Control Systems*.



Gambar 2.1: Local version control[1].

Gambar 2.1 merupakan struktur dari *Local Version Control Systems*. *Database local Version Control Systems* ini tersimpan pada *local directory* di komputer. *Database* ini menyimpan perubahan *file* ke dalam beberapa versi atau *state*.

Centralized Version Control Systems



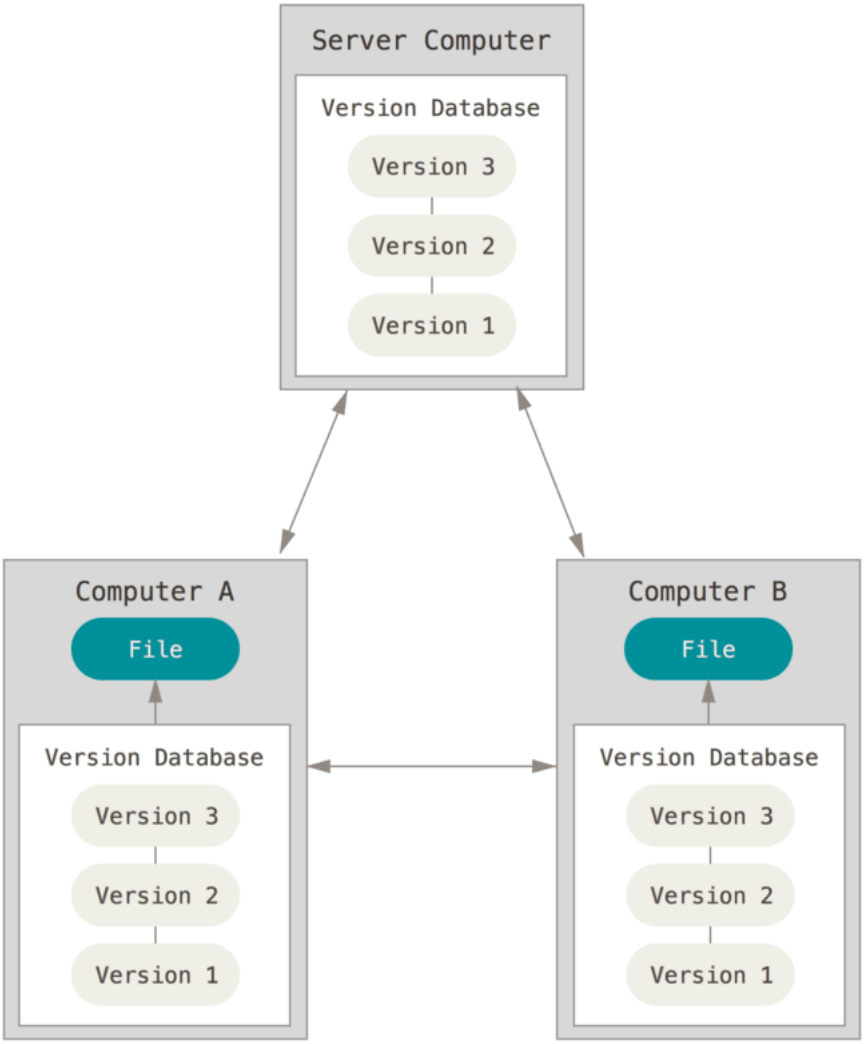
Gambar 2.2: Centralized version control[1].

Local Version Control hanya menyimpan *file* pada satu komputer saja. Muncul masalah baru ketika *user* ingin berkolaborasi dengan *user* lain. Untuk mengatasi masalah ini dikembangkan *Centralized version control*. Gambar 2.2 merupakan struktur dari *Centralized Version Control Systems*. Dalam *Centralized Control Version Systems* terdapat sebuah *server* yang menyimpan setiap versi *file*, dan klien yang dapat melakukan *checkout file*.

Sistem *Centralized Version Control Systems* memiliki beberapa kelebihan. Setiap *user* dapat mengetahui pekerjaan yang dilakukan oleh *user* lain. Administrator dapat lebih mudah mengontrol *database Centralized Version Control Systems* dibandingkan dengan *database Local Version Control Systems* dari setiap klien.

Sistem *Centralized Version Control Systems* memiliki kelemahan. Jika *server* pusat *Centralized Version Control Systems* mati , maka perubahan pada *file* tidak bisa disimpan. Klien juga tidak dapat melakukan kolaborasi dengan klien lain. Jika *harddisk* pada server rusak, maka semua versi *file* akan hilang.

Distributed Version Control Systems

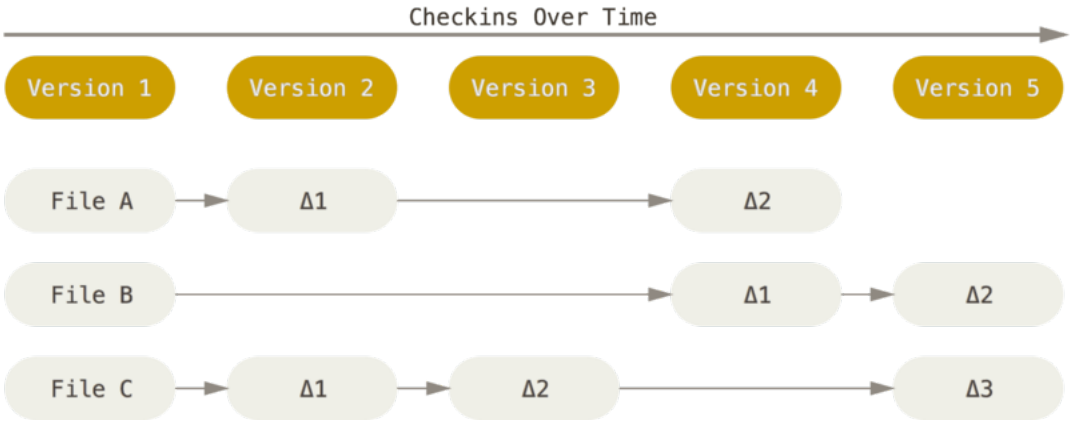


Gambar 2.3: Distributed version control[1].

Gambar 2.3 merupakan struktur dari *Distributed Version Control Systems*. Dalam sebuah DVCS (seperti Git, Mercurial, Bazaar atau Darcs), klien tidak hanya melakukan *checkout* untuk *snapshot* terakhir setiap *file*, namun klien juga memiliki salinan dari repositori tersebut. Dengan kata lain setiap klien memiliki *version database local* pada komputernya. Jika server pusat mati, klien masih bisa melakukan kolaborasi dan klien manapun dapat mengirimkan kembali salinan repositori ke *server*.

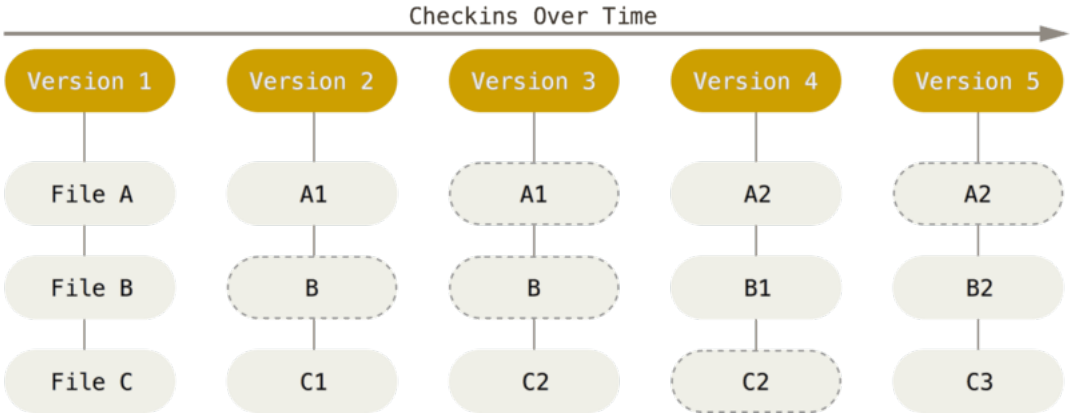
2.1.2 Cara Kerja Git

Salah satu perbedaan antara Git dengan VCS lainnya adalah dalam cara Git memperlakukan datanya. Kebanyakan sistem *Version Control Systems* lain menyimpan informasi sebagai daftar perubahan *file*. Pada Gambar 2.4, terdapat tiga *file*.*Version Control Systems* menyimpan *file* A, B, dan C pada versi pertama saja. Untuk versi kedua dan seterusnya yang disimpan adalah perubahan pada setiap *file*. Sistem ini disebut juga sebagai *delta-based Version Control Systems*.



Gambar 2.4: Menyimpan data sebagai *snapshots* dari *project*[1].

Berbeda dengan *Version Control Systems* lainnya, Git memperlakukan datanya sebagai sebuah kumpulan *snapshot* dari sebuah miniatur *file system*. Setiap kali dilakukan *commit*, git merekam *state* dari sekumpulan *file* dan menyimpannya sebagai *snapshot*. Gambar 2.5, menunjukkan *snapshots* dari *file* A, B, dan C. Pada versi kedua, *file* B tidak mengalami perubahan, sehingga yang disimpan adalah *reference* dari *file* B pada versi sebelumnya.



Gambar 2.5: Menyimpan data sebagai perubahan terhadap versi dasar dari setiap *file*[1].

State pada Git

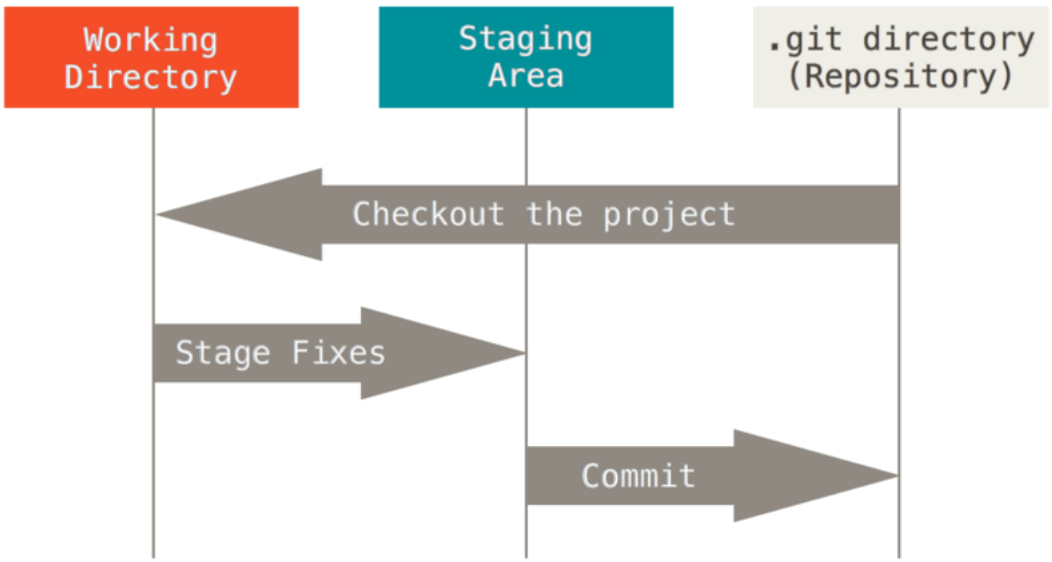
Terdapat tiga *state* pada Git yaitu *committed*, *modified*, and *staged*.*Committed* adalah *state* dimana data sudah disimpan di *local database*. *Modified* adalah *state* dimana terdapat perubahan pada *file*, namun *file* tersebut belum di *commit* ke *database*. *Staged* adalah *state* dimana *file* telah ditandai untuk kemudian dilakukan *commit*.

Terdapat tiga bagian utama dari sebuah *project* Git yaitu direktori Git, *working directory*, dan *staging area*. Direktori Git merupakan tempat dimana Git menyimpan *metadata* dan *object database* dari *project*. *Working tree* adalah suatu *snapshot* dari *project*. Sekumpulan *file* ini diambil dari *database* di direktori Git dan ditempatkan pada *disk* untuk digunakan dan dimodifikasi. *Staging* area adalah suatu *file*, dimana *file* ini menyimpan daftar *file* yang telah ditandai untuk kemudian dilakukan *commit*. *File staging area* terdapat pada direktori Git. Untuk lebih jelasnya, lihat Gambar 2.6.

Alur kerja dari Git adalah sebagai berikut:

1. Melakukan modifikasi pada *file*.
2. Menandai perubahan pada *file* dan memindahkannya ke *staging area*.

3. Mengambil *file* dari *staging area* dan menyimpan *snapshot* ke direktori Git. Proses ini disebut dengan *commit*.



Gambar 2.6: *Working tree*, *Staging area*, dan Git direktori[1].

Commit

Commit merupakan sebuah *snapshot* dari suatu *file* atau direktori. *Commit* menggambarkan *state* dari *working directory*. Gambar 2.5 menunjukkan terdapat tiga *file* pada *version* 4. Dimana terdapat *file* A1, B1, dan C1 pada *working directory*. *File* A1, B1, dan C2 merupakan *state file* A, B, dan C pada *version* 4.

Listing 2.1: Contoh histori commit dalam pengembangan perangkat lunak

```
C:\Users\user\Documents\GitHub\train-tracker-ellena-angelica>git log

commit b8aeacbd4743619b7b2d790d45bde26b899641e0 (HEAD -> master, origin/master,
    origin/HEAD)
Author: adamadamadamadamadam <adamnurmishwari@gmail.com>
Date:   Thu May 3 01:15:31 2018 +0700

    commitan terakhir. mastiin g buang memory sm batre

commit f836cc65bf6d50e274df54aa06c6fb529667aa06
Author: Evelyn Wijaya <evelynwijaya777@gmail.com>
Date:   Wed May 2 22:03:25 2018 +0700

    Update README.md

commit 2e1ce9a03a1f417326c3c6586503303cf6daf6b8
Author: Evelyn Wijaya <evelynwijaya777@gmail.com>
Date:   Wed May 2 22:01:10 2018 +0700

    Create README.md

commit 2f04488f9008745e8e6f67da33ffb2f6c2c9e747
Author: Evelyn Wijaya <evelynwijaya777@gmail.com>
Date:   Wed May 2 14:08:02 2018 +0700

    fix stasiun double
```

```
26
27 commit 7d8b66a9c6500de2753cdeac1084dc049c0c9f20
28 Author: Evelyn Wijaya <evelynwijaya777@gmail.com>
29 Date:   Wed May 2 13:32:21 2018 +0700
30
31     fix stasiun double
```

Seperti yang diperlihatkan pada Listing 2.1, setiap *commit* memiliki beberapa informasi. Baris pertama menunjukkan *commit* ID berupa *SHA-1 hash value*, dengan panjang empat puluh karakter heksadesimal. *SHA-1 hash value* ini digunakan untuk pengecekan integritas pada *file*. Pada baris pertama, *master* menunjukkan *branch* yang sedang aktif, *master* juga merupakan *reference* ke *commit* terakhir. *HEAD* merupakan *reference* ke *master*. *Origin/master* dan *origin/HEAD* merupakan *master* dan *HEAD* pada *remote repository*. Baris kedua menunjukkan orang yang melakukan *commit* dan alamat emailnya. Baris ketiga menunjukkan waktu *commit*. Baris terakhir berisi deskripsi dari *commit* tersebut.

2.1.3 Operasi Dasar pada Git

Pada subbab ini dijelaskan mengenai operasi dasar dalam Git dan sintaks-sintaksnya. Sintaks-sintaksnya ini dimasukkan pada Git *command line*. Berikut ini adalah operasi-operasi dasar dalam Git:

1. Init
   Operasi ini digunakan untuk membuat repositori lokal baru dengan nama tertentu. Bisa juga digunakan untuk merekam direktori yang sudah ada. Berikut adalah sintaks untuk melakukan operasi *init*:
   ```
   $ git init [project-name]
   ```

2. Add
   Operasi ini digunakan untuk menandai perubahan pada *file* dan memindahkan *file* tersebut ke *staging area*. Operasi ini juga digunakan untuk menambahkan *file* yang dipantau perubahannya. Berikut adalah sintaks untuk melakukan operasi add:
   ```
   $ git add [file]
   ```

3. Commit
   Operasi ini digunakan untuk merekam *snapshot* atau *state file* atau sekumpulan *file*. Operasi ini juga digunakan untuk memindahkan *file* yang berada di *stagging area* ke direktori Git. Berikut adalah sintaks untuk melakukan operasi *commit*:
   ```
   $ git commit -m "[descriptive message]"
   ```

4. Branch
   Operasi ini digunakan untuk menampilkan semua *branch* yang ada pada repositori Git, membuat *branch* baru, dan menghapus *branch*. Berikut adalah sintaks-sintaks untuk melakukan operasi *branch*:
   ```
   $ git branch
   $ git branch [branch-name]
   $ git branch -d [branch-name]
   $ git branch -D [branch-name]
   ```

5. Diff
   Operasi ini digunakan untuk menampilkan perbedaan pada *file* yang belum masuk *staging area*, menampilkan perbedaan pada *file* yang berada di *staging area* dengan *file* di *commit* sebelumnya, dan perbedaan *file* antara dua *branch*. Berikut adalah sintaks-sintaks untuk melakukan operasi *diff*:
   ```
   $ git diff
   ```

```
$ git diff -staged
$ git diff [first-branch]...[second-branch]
```

6. Clone
   Operasi ini digunakan untuk menyalin repositori Git yang berada di komputer lain atau suatu *server*. Berikut adalah sintaks untuk melakukan operasi *clone*:
   `$ git clone [url]`

7. Fetch
   Operasi ini digunakan untuk mengambil data dari *remote* repositori ke repositori lokal. Berikut adalah sintaks untuk melakukan operasi *fetch*:
   `$ git fetch [bookmark]`

8. Merge
   Operasi ini digunakan untuk menggabungkan *branch* tertentu dengan *branch* yang sedang aktif. Operasi ini juga digunakan untuk menggabungkan data yang diambil dari *remote* repositori dengan data pada *working directory*. Berikut adalah sintaks untuk melakukan operasi *merge*:
   `$ git merge [branch]/[bookmark]`

9. Pull
   Operasi ini adalah gabungan dari operasi *fetch* dan *merge*. Berikut adalah sintaks untuk melakukan operasi *pull*:
   `$ git pull`

10. Push
    Operasi ini digunakan untuk mengirim data pada reposipori Git lokal ke *remote repository*. Berikut adalah sintaks untuk melakukan operasi *push*:
    `$ git push [bookmark] [branch]`

11. Checkout
    Operasi ini digunakan untuk berpindah ke *branch* atau *commit* tertentu, setelah itu memperbarui *file* pada *working directory* berdasarkan *branch* atau *commit* tersebut. Berikut ini adalah sintaks-sintaks untuk operasi *checkout*:
    ```
    $ git checkout [commit ID]
    $ git checkout [branch-name]
    ```

12. Log
    Operasi ini digunakan untuk menampilkan semua histori *commit* pada *branch* yang sedang aktif. Berikut ini adalah sintaks untuk melakukan operasi *log*:
    `$ git log`

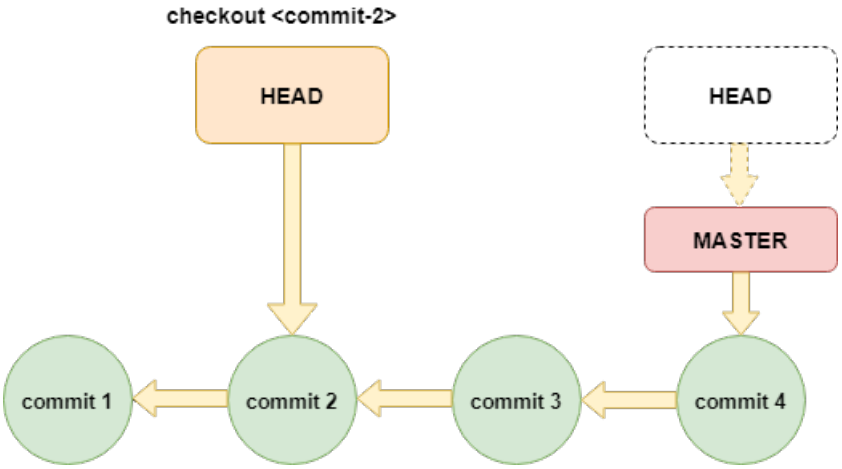
13. Reset
    Operasi ini digunakan untuk memindahkan posisi *HEAD* ke *commit* tertentu, selain itu secara opsional melakukan *reset* pada *staging area* dan *working tree* berdasarkan tipe *reset*. Terdapat tiga tipe *reset* yaitu, *hard*, *soft*, dan *mixed*. Pada tipe *hard*, dilakukan *reset* pada *staging area* dan *working tree*, selain itu, posisi *HEAD* dipindah sehingga menunjuk ke *commit* tertentu. Pada tipe *soft*, hanya posisi *HEAD* saja yang berpindah. Pada tipe *mixed*, dilakukan *reset* pada *staging area* dan posisi *HEAD* dipindah sehingga menunjuk ke *commit* tertentu.

    Berikut ini adalah sintaks untuk melakukan operasi *reset*:
    ```
    $ git reset -hard [commit]
    $ git reset -mixed [commit]
    $ git reset -soft [commit]
    ```

2.1.4 Git Checkout

Seperti yang sudah dijelaskan pada subbab 2.1.3, *checkout* dapat digunakan untuk berpindah ke *branch* atau *commit* tertentu. Operasi *checkout* dapat dilakukan menggunakan sintaks `$ git checkout` diikuti dengan nama *branch* atau *commit ID*. Gambar 2.7 menunjukkan contoh *checkout* pada *commit*. Posisi awal *HEAD* menunjuk pada *branch master*, setelah dilakukan *checkout* ke *commit* 2, posisi *HEAD* menunjuk pada *commit* 2. *Working directory* diperbarui berdasarkan *state* pada *commit* 2.

HEAD yang menunjuk langsung ke suatu *commit* disebut dengan *detached HEAD*. Perubahan yang terjadi pada *detached HEAD* tidak akan terekam oleh Git. Jika terdapat perubahan, kemudian dilakukan *checkout commit* atau *branch*, perubahan tersebut akan hilang. *HEAD* dapat dipindahkan ke posisi semula(menunjuk pada *branch master*) dengan menggunakan sintaks `$ git checkout master`.



Gambar 2.7: *Checkout* pada *commit*

2.2 JGit

JGit adalah *library* Java murni yang mengimplementasikan Git *version control systems*[2]. Dengan menggunakan JGit, operasi-operasi dalam Git bisa dilakukan melalui program Java. Pada subbab berikut dijelaskan beberapa kelas dari *library* JGit. Subbab ini mengacu pada [4].

2.2.1 Kelas Repository

Kelas ini merepresentasikan repositori Git. Berikut ini adalah beberapa *method* dalam kelas ini:

- public void create()
  Berfungsi untuk membuat repositori Git baru.

- public String getBranch()
  Berfungsi untuk mendapatkan nama *branch* yang ditunjuk oleh *HEAD*.
  Kembalian: nama dari *branch* yang sedang aktif, contohnya *master*.

- public Ref getRef(String name)
  Berfungsi untuk mendapatkan *reference* berdasarkan nama yang diberikan.
  Parameter: nama dari *reference*.
  Kembalian: *object* bertipe Ref. Ref merupakan variabel yang menyimpan ID dari Git object. Salah satu contoh dari Git Object adalah *commit*.

2.2.2 Kelas FileRepository

Kelas ini merupakan turunan dari kelas *Repository*. Berikut ini adalah *construtor* dari kelas ini:

- public FileRepository(String gitDir) throws IOException
  Constructor ini membuat representasi dari repositori Git. *Constructor* ini melempar IOException jika repositori tidak bisa diakses.
  Parameter: *path* dari suatu repositori Git.

2.2.3 Kelas Git

Kelas ini menyediakan API yang mirip Git Command Line untuk berinteraksi dengan repositori Git. Berikut ini adalah *constructor* dan beberapa *method* dalam kelas ini:

- public Git(Repository repo)
  Constructor ini membuat objek Git yang digunakan untuk berinteraksi dengan repositori Git.
  Parameter: objek *Repository* yang digunakan untuk berinteraksi. Parameter tidak boleh bernilai *null*.

- public static InitCommand init()
  Method ini mengembalikan objek *command* untuk mengeksekusi operasi *init*.
  Kembalian: objek *InitCommand* yang berfungsi untuk mengumpulkan parameter opsional dan akhirnya mengeksekusi operasi *init*.

- public AddCommand add()
  Method ini mengembalikan objek *command* untuk mengeksekusi operasi *add*.
  Kembalian: objek *AddCommand* yang berfungsi untuk mengumpulkan parameter opsional dan akhirnya mengeksekusi operasi *add*.

- public LogCommand log()
  Method ini mengembalikan objek *command* untuk mengeksekusi operasi *log*.
  Kembalian: objek *LogCommand* yang berfungsi untuk mengumpulkan parameter opsional dan akhirnya mengeksekusi operasi *log*.

- public CheckoutCommand checkout()
  Method ini mengembalikan objek *command* untuk mengeksekusi operasi *checkout*.
  Kembalian: objek *CheckoutCommand* yang berfungsi untuk mengumpulkan parameter opsional dan akhirnya mengeksekusi operasi *checkout*.

- public CommitCommand commit()
  Method ini mengembalikan objek *command* untuk mengeksekusi operasi *commit*.
  Kembalian: objek *CommitCommand* yang berfungsi untuk mengumpulkan parameter opsional dan akhirnya mengeksekusi operasi *commit*.

- public FetchCommand fetch()
  Method ini mengembalikan objek *command* untuk mengeksekusi operasi *fetch*.
  Kembalian: objek *FetchCommand* yang berfungsi untuk mengumpulkan parameter opsional dan akhirnya mengeksekusi operasi *fetch*.

- public PushCommand push()
  Method ini mengembalikan objek *command* untuk mengeksekusi operasi *push*.
  Kembalian: objek *PushCommand* yang berfungsi untuk mengumpulkan parameter opsional dan akhirnya mengeksekusi operasi *push*.

- public DiffCommand diff()
  Method ini mengembalikan objek *command* untuk mengeksekusi operasi *diff*.

Kembalian: objek *DiffCommand* yang berfungsi untuk mengumpulkan parameter opsional dan akhirnya mengeksekusi operasi *diff*.

- public static CloneCommand cloneRepository()
  Method ini mengembalikan objek *command* untuk mengeksekusi operasi *clone*.
  Kembalian: objek *CloneCommand* yang berfungsi untuk mengumpulkan parameter opsional dan akhirnya mengeksekusi operasi *clone*.

- public MergeCommand merge()
  Method ini mengembalikan objek *command* untuk mengeksekusi operasi *merge*.
  Kembalian: objek *MergeCommand* yang berfungsi untuk mengumpulkan parameter opsional dan akhirnya mengeksekusi operasi *merge*.

- public PullCommand pull()
  Method ini mengembalikan objek *command* untuk mengeksekusi operasi *pull*.
  Kembalian: objek *PullCommand*.

- public CreateBranchCommand branchCreate()
  Method ini mengembalikan objek *command* untuk membuat *branch* baru.
  Kembalian: objek *CreateBranchCommand*.

- public public ListBranchCommand branchList()
  Method ini mengembalikan objek *command* untuk menampilkan daftar *branch*.
  Kembalian: objek *ListBranchCommand*.

- public DeleteBranchCommand branchDelete()
  Method ini mengembalikan objek *command* untuk menghapus *branch*.
  Kembalian: objek *DeleteBranchCommand*.

- public ResetCommand reset()
  Method ini mengembalikan objek *command* untuk mengeksekusi operasi *reset*.
  Kembalian: objek *ResetCommand*.

2.2.4 Kelas Checkout Command

Kelas yang digunakan untuk melakukan operasi *checkout*. Berikut adalah beberapa *method* yang terdapat pada kelas ini:

- public CheckoutCommand setName(String name)
  Menentukan nama *branch* atau *commit* untuk melakukan checkout.
  Parameter: nama dari *branch* atau *commit*.
  Kembalian: *object* CheckoutCommand.

- public Ref call()
  Berfungsi untuk menjalankan operasi *checkout*.

2.2.5 Kelas Log Command

Kelas yang digunakan untuk melakukan operasi *log*. Berikut adalah *method* yang terdapat pada kelas ini:

- public Iterable<RevCommit> call()
  Berfungsi untuk menjalankan operasi *log*.
  Kembalian: histori *commit* pada *branch* yang sedang aktif.

2.2.6 Kelas Reset Command

Kelas yang digunakan untuk melakukan operasi *reset*. Berikut adalah beberapa *method* yang terdapat pada kelas ini:

- public ResetCommand setMode(ResetCommand.ResetType mode)
  Menentukan tipe *reset*.
  Parameter: tipe *reset*.
  Kembalian: *object* ResetCommand.
- public ResetCommand setRef(String ref)
  Mengatur nama *reference* tujuan dalam operasi *reset*. Secara default, *reference* yang digunakan adalah *HEAD*.
  Parameter: nama dari *reference*.
  Kembalian: *object* ResetCommand.
- public Ref call()
  Berfungsi untuk menjalankan operasi *reset*.

2.2.7 Kelas ResetCommand.ResetType

Merupakan *enumeration* yang menentukan tipe *reset* yang digunakan pada operasi *reset*. *Enumeration* tersebut adalah sebagai berikut:

- public static final ResetCommand.ResetType SOFT
  Hanya mengubah posisi *HEAD*.
- public static final ResetCommand.ResetType HARD
  Mengubah posisi *HEAD* ke *reference* tujuan. Selain itu, mengubah *staging area* dan *working directory* berdasarkan *reference* tersebut.
- public static final ResetCommand.ResetType MIXED
  Mengubah posisi *HEAD* ke *reference* tujuan dan mengubah *staging area* berdasarkan *reference* tersebut.

2.2.8 Kelas RevCommit

Kelas ini merupakan *reference* ke *commit* yang ada di *Directed Acyclic Graph*. Berikut ini adalah beberapa *method* dari kelas ini:

- public final String getFullMessage()
  Berfungsi untuk mendapatkan deskripsi dari suatu *commit*.
  Kembalian: deskripsi dari suatu *commit* dalam bentuk String.
- public final String getShortMessage()
  Berfungsi untuk mendapatkan deskripsi dari suatu *commit*. Jika panjang deskripsi *commit* lebih dari satu baris, maka hanya baris pertama yang dikembalikan.
  Kembalian: baris pertama dari deskripsi *commit* dalam bentuk String.
- public final String getName()
  Method ini mengembalikan *commit ID*.
  Kembalian: *commit ID* berupa SHA-1 hash value dalam bentuk String.
- public final PersonIdent getAuthorIdent()
  Berfungsi untuk mendapatkan informasi mengenai *author* yang melakukan *commit*.
  Kembalian: objek *PersonIdent* yang memuat informasi tentang *author*(nama dan *email*) dan waktu dilakukannya *commit*.

2.2.9 Kelas PersonIdent

Kelas ini memberikan informasi mengenai *author* dari suatu *commit*. Berikut ini adalah beberapa *method* dari kelas ini:

- public String getName()
  Berfungsi untuk mengembalikan nama dari *author* yang melakukan *commit*.
  Kembalian: nama dari *author*.
- public String getEmailAddress()
  Berfungsi untuk mengembalikan alamat *email* dari *author* yang melakukan *commit*.
  Kembalian: alamat *email* dari *author*.
- public Date getWhen()
  Berfungsi mengembalikan waktu dilakukannya suatu *commit* oleh *author*.
  Kembalian: waktu dilakukannya suatu *commit* berupa objek Date.

2.3 Selenium WebDriver

Selenium adalah seperangkat alat yang secara khusus digunakan untuk mengotomatisasi *web browsers*[3]. Selenium mendukung bahasa pemrograman C#, Java, Perl, PHP, Python, Ruby, dan JavaScript. Selenium terdiri dari beberapa kakas, yaitu Selenium 1(Selenium RC), Selenium 2(Selenium WebDriver), dan Selenium IDE. Selenium RC merupakan proyek utama *Selenium* untuk waktu yang lama, sebelum akhirnya bergabung dengan *WebDriver* menjadi Selenium 2. Selenium RC melakukan *automation test* dengan cara menginjeksi kode JavaScript ke *browser*. Selenium RC sekarang sudah *deprecated* dan tidak digunakan lagi. Selenium Webdriver merupakan gabungan dari Selenium RC dan WebDriver. Selenium IDE merupakan *extension* dari*browser* Chrome dan Firefox yang digunakan untuk melakukan *record and playback test* pada *browser*.

WebDriver merupakan kakas untuk mengotomatisasi pengujian pada aplikasi *web*[5]. WebDriver dapat berkomunikasi secara langsung dengan *browser* menggunakan *native support* pada *browser* tanpa melakukan injeksi kode JavaScript. WebDriver yang terdapat pada Selenium WebDriver antara lain ChromeDriver, FirefoxDriver, OperaDriver, InternetExplorerDriver, EdgeDriver, dan HtmlUnitDriver.

ChromeDriver merupakan implementasi WebDriver yang mengontrol Chrome *browser*. FirefoxDriver merupakan implementasi WebDriver yang mengontrol Firefox *browser*. OperaDriver merupakan implementasi WebDriver yang mengontrol Opera *browser*. InternetExplorerDriver merupakan implementasi WebDriver yang mengontrol Internet Explorer *browser*. EdgeDriver merupakan implementasi WebDriver yang mengontrol Microsoft Edge *browser*. HtmlUnitDriver merupakan implementasi WebDriver yang mengontrol HTMLUnit *browser* yang bersifat *headless*(tidak mempunyai GUI).

Pada skripsi ini *tools* Selenium yang digunakan hanya Selenium WebDriver. Bahasa pemrograman yang digunakan adalah java. WebDriver yang digunakan adalah ChromeDriver. Pemilihan WebDriver ini hanya berdasarkan preferensi penulis saja. Penulis tidak memilih InternetExplorerDriver, EdgeDriver, dan HtmlUnitDriver karena InternetExplorerDriver dan EdgeDriver hanya bisa berjalan pada sistem operasi Windows, selain itu HtmlUnitDriver tidak mempunyai GUI sehingga tidak bisa dilakukan pengambilan *screenshot*. Pada subbab berikut dijelaskan beberapa kelas dan *interface* dari *library* Selenium WebDriver. Subbab ini mengacu pada [6].

2.3.1 Interface WebDriver

Merupakan *interface* utama yang digunakan untuk pengujian, *interface* ini merepresentasikan *web browser* yang ideal. Berikut ini adalah beberapa *method* dalam *interface* ini:

- void close()
  Berfungsi untuk menutup *window* pada *browser*, jika *window* yang sekarang merupakan satu-satunya *window* yang terbuka maka *browser* akan ditutup.

- void quit()
  Berfungsi untuk menutup *browser* dan semua *window* yang sedang terbuka.

- void get(String url)
  Berfungsi untuk memuat halaman *web* pada *window* yang sedang aktif. *Method* ini mengirim *HTPP GET Request* untuk memuat halaman, dan *method* ini akan melakukan *blocking* sampai halaman *web* selesai dimuat.
  Parameter: alamat *url* untuk memuat halaman *web*.

- String getTitle()
  Berfungsi untuk mengembalikan judul dari halaman *web* yang sedang aktif.
  Kembalian: judul dari halaman *web*.

- String getCurrentUrl()
  Berfungsi untuk mendapatkan URL yang sedang aktif di *browser*.
  Kembalian: URL dari halaman *web* yang sedang dimuat di *browser*.

- WebDriver.Options manage()
  Berfungsi untuk mendapatkan interface WebDriver.Options.
  Kembalian: interface dengan tipe WebDriver.Options. Di dalam kelas ini terdapat *method* window() yang mengembalikan *interface* dengan tipe WebDriver.Window.

2.3.2 Interface WebDriver.Window

Interface yang digunakan untuk mengatur *window* pada *browser*. Berikut adalah *method* yang dimiliki oleh *interface* ini:

- void maximize()
  Berfungsi untuk membuat ukuran *window* pada *browser* menjadi maksimal.

- public setSize(Dimension targetSize)
  Berfungsi untuk mengatur ukuran *window* pada *browser* sesuai parameter.
  Parameter: ukuran dari *browser* berupa objek Dimension. Objek Dimension mempunyai atribut height dan weight yang menyatakan ukuran panjang dan lebar dalam satuan piksel.

2.3.3 Kelas ChromeDriver

Merupakan merupakan implementasi WebDriver yang mengontrol Chrome *browser*. Berikut adalah *constructor* yang terdapat pada kelas ini:

- public ChromeDriver()
  Constructor ini membuat instans dengan tipe ChromeDriver.

2.3.4 Kelas FirefoxDriver

Merupakan merupakan implementasi WebDriver yang mengontrol Firefox *browser*. Berikut adalah *constructor* yang terdapat pada kelas ini:

- public FirefoxDriver()
  Constructor ini membuat instans dengan tipe FirefoxDriver.

2.3.5 Kelas OperaDriver

Merupakan merupakan implementasi WebDriver yang mengontrol Opera *browser*. Berikut adalah *constructor* yang terdapat pada kelas ini:

- public OperaDriver()
  Constructor ini membuat instans dengan tipe OperaDriver.

2.3.6 Kelas InternetExplorerDriver

Merupakan merupakan implementasi WebDriver yang mengontrol Internet Explorer *browser*. Berikut adalah *constructor* yang terdapat pada kelas ini:

- public InternetExplorerDriver()
  Constructor ini membuat instans dengan tipe InternetExplorerDriver.

2.3.7 Kelas EdgeDriver

Merupakan merupakan implementasi WebDriver yang mengontrol Microsoft Edge *browser*. Berikut adalah *constructor* yang terdapat pada kelas ini:

- public EdgeDriver()
  Constructor ini membuat instans dengan tipe EdgeDriver.

2.3.8 Kelas HtmlUnitDriver

Merupakan merupakan implementasi WebDriver yang mengontrol HtmlUnit *browser*. Berikut adalah *constructor* yang terdapat pada kelas ini:

- public HtmlUnitDriver()
  Constructor ini membuat instans dengan tipe HtmlUnitDriver.

2.3.9 Interface OutputType

Merupakan *interface* yang menentukan tipe *output* pada *screenshot*. Terdapat tiga konstanta untuk menentukan tipe *output* pada *screenshot*. Konstanta tersebut adalah sebagai berikut:

- static final OutputType<String> BASE64
  Berfungsi untuk mendapatkan *screenshot* dalam bentuk *base64 data*.
- static final OutputType<byte[]> BYTES
  Berfungsi untuk mendapatkan *screenshot* dalam bentuk *raw bytes*.
- static final OutputType<java.io.File> FILE
  Berfungsi untuk mendapatkan *screenshot* dalam bentuk *temprorary file* yang akan dihapus setelah program keluar dari *Java Virtual Machine*.

2.3.10 Interface TakesScreenshot

Merupakan *interface* yang digunakan untuk mengambil *screenshot*. WebDriver yang mengimplementasikan *interface* ini yaitu ChromeDriver, FirefoxDriver, OperaDriver, InternetExplorerDriver, dan EdgeDriver. HtmlUnitDriver tidak mengimplentasikan *interface* ini. Berikut adalah *method* yang terdapat dalam *interface* ini:

- <X> X getScreenshotAs(OutputType<X> target) throws WebDriverException
  Method ini berfungsi untuk mengambil *screenshot* dan mengembalikan hasilnya.
  Kembalian: objek hasil *screenshot*
  Parameter: tipe *output* yang diinginkan(lihat subbab 2.3.9).

2.4 Apache Commons CLI

Library Apache Commons CLI menyediakan API untuk melakukan *parsing* argumen Command Line yang dikirimkan ke program[7]. Apache Commons CLI termasuk ke dalam salah satu *project* Apache Commons. Tujuan utama dari *project* Apache Commons adalah membuat dan melakukan *maintain* pada komponen Java yang *reusable*. Pada subbab berikut dijelaskan beberapa kelas dan *interface* dari *library* Apache Commons CLI. Subbab ini mengacu pada [8].

2.4.1 CommandLineParser

Merupakan sebuah *interface*. Kelas yang mengimplementasikan *interface* ini dapat melakukan *parsing* argumen CommandLine berdasarkan pada *option* yang telah ditentukan. Berikut ini adalah beberapa *method* yang dimiliki *interface* ini:

- CommandLine parse(Options options, String[] arguments) throws ParseException
  Berfungsi untuk melakukan *parsing* argumen Command Line berdasarkan pada *option* yang telah ditentukan. *Method* ini melempar *ParseException* jika terjadi masalah saat melakukan *parsing*.
  Parameter: *option* yang telah ditentukan, argumen Command Line.
  Kembalian: objek *CommandLine*.

- CommandLine parse(Options options, String[] arguments,boolean stopAtNonOption) throws ParseException
  Berfungsi untuk melakukan *parsing* argumen Command Line berdasarkan pada *option* yang telah ditentukan.
  Parameter: *option* yang telah ditentukan, argumen Command Line, dan suatu *boolean* yang menentukan apakah *parsing* dihentikan jika terdapat argumen yang tidak valid. Jika bernilai *true*, *parsing* akan dihentikan dan semua argumen yang sudah diuraikan akan ditambahkan ke objek *CommandLine*. Jika bernilai *false*, akan dilempar *ParseException* bila terdapat Option yang tidak valid.
  Kembalian: objek *CommandLine*.

2.4.2 DefaultParser

Merupakan kelas yang mengimplementasikan *interface* CommandLineParser. Berikut adalah *method* yang dimiliki kelas ini:

- CommandLine parse(Options options, String[] arguments) throws ParseException
  Berfungsi untuk melakukan *parsing* argumen Command Line berdasarkan pada *option* yang telah ditentukan. *Method* ini melempar *ParseException* jika terjadi masalah saat melakukan *parsing*.
  Parameter: *option* yang telah ditentukan, argumen Command Line.
  Kembalian: objek *CommandLine*.

- CommandLine parse(Options options, String[] arguments,boolean stopAtNonOption) throws ParseException
  Berfungsi untuk melakukan *parsing* argumen Command Line berdasarkan pada *option* yang telah ditentukan.
  Parameter: *option* yang telah ditentukan, argumen Command Line, dan suatu *boolean* yang menentukan apakah *parsing* dihentikan jika terdapat argumen yang tidak valid. Jika bernilai *true*, *parsing* akan dihentikan dan semua argumen yang sudah diuraikan akan ditambahkan ke objek *CommandLine*. Jika bernilai *false*, akan dilempar *ParseException* bila terdapat Option yang tidak valid.
  Kembalian: objek *CommandLine*.

2.4.3 CommandLine

Kelas ini merepresentasikan kumpulan Option hasil *parsing*. Berikut ini adalah beberapa *method* yang dimiliki kelas ini:

- public String getOptionValue(String opt)
  Berfungsi untuk mendapatkan nilai dari suatu Option sesuai dengan namanya.
  Parameter: nama dari Option.
  Kembalian: nilai dari suatu Option. Jika Option tidak ditemukan, akan mengembalikan *null*.

- public String[] getOptionValues(Option option)
  Berfungsi untuk mendapatkan kumpulan nilai dari suatu Option. Parameter: nama dari Option. Kembalian: kumpulan nilai dari suatu Option dengan tipe *array of String*. Jika Option tidak ditemukan, akan mengembalikan *null*.

- public Option[] getOptions()
  Berfungsi untuk mengembalikan Option-option hasil *parsing*.
  Kembalian: *array* dari Option hasil *parsing*.

2.4.4 Options

Kelas ini merepresentasikan kumpulan dari objek *Option*, yang mendeskripsikan kemungkinan *option* pada *command line*. Berikut ini adalah *method* yang dimiliki kelas ini:

- public Options addOption(Option opt)
  Berfungsi untuk menambahkan Option.
  Parameter: Option yang akan ditambahkan.
  Kembalian: hasil dari Option yang ditambahkan.

2.4.5 Option

Kelas ini merepresentasikan sebuah Command Line Option. Berikut ini adalah beberapa *method* yang dimiliki kelas ini:

- public String getLongOpt()
  Berfungsi untuk mendapatkan nama panjang dari suatu *option*.
  Kembalian: nama panjang dari suatu *option*.

- public String getValue()
  Berfungsi untuk mendapatkan nilai dari Option.
  Kembalian: nilai dari Option. Jika terdapat lebih dari satu kembalian, hanya nilai pertama yang dikembalikan. Jika tidak ada nilai, akan mengembalikan *null*.

- public String[] getValues()
  Berfungsi untuk mendapatkan nilai-nilai dari Option.
  Kembalian: nilai-nilai dari Option berupa *array of String*. Jika tidak ada nilai, akan mengembalikan *null*.

- public boolean hasArg()
  Berfungsi untuk mengetahui apakah suatu *option* membutuhkan argumen.
  Kembalian: *true* jika *option* ini membutuhkan argumen , *false* jika *option* ini tidak membutuhkan argumen.

- public boolean hasArg()
  Berfungsi untuk mengetahui apakah suatu *option* dapat menerima lebih dari satu argumen.
  Kembalian: *true* jika *option* ini dapat menerima lebih dari satu argumen , *false* jika tidak.

- public String getDescription()
  Berfungsi untuk mendapatkan deskripsi dari suatu *option*.
  Kembalian: deskripsi dari *option* ini.

- public String getArgName()
  Berfungsi untuk mendapatkan nama dari suatu *option*.
  Kembalian: nama dari argumen suatu *option*

2.4.6 Option.Builder

Kelas ini merupakan *nested class* dari kelas *Option*. Kelas ini digunakan untuk membuat objek Option berdasarkan parameter yang diberikan. Berikut ini adalah beberapa *method* yang dimiliki kelas ini:

- public Option.Builder desc(String description)
  Berfungsi untuk memberikan deskripsi pada *option*.
  Parameter: deskripsi dari *option*.
  Kembalian: objek *Option.Builder* yang bisa digunakan untuk *method chaining*.

- public Option.Builder longOpt(String longOpt)
  Berfungsi untuk memberikan nama panjang pada *option*.
  Parameter: nama panjang *option*.
  Kembalian: objek *Option.Builder* yang bisa digunakan untuk *method chaining*.

- public public Option.Builder hasArg()
  Berfungsi untuk menyatakan bahwa *option* ini membutuhkan argumen.
  Kembalian: objek *Option.Builder* yang bisa digunakan untuk *method chaining*.

- public public Option.Builder hasArgs()
  Berfungsi untuk menyatakan bahwa *option* ini membutuhkan argumen, dimana jumlah argumen bisa lebih dari satu.
  Kembalian: objek *Option.Builder* yang bisa digunakan untuk *method chaining*.

- public Option.Builder argName(String argName)
  Berfungsi untuk memberi nama pada argumen.
  Parameter: nama argumen.
  Kembalian: objek *Option.Builder* yang bisa digunakan untuk *method chaining*.

- public Option.Builder required(boolean required)
  Berfungsi untuk menyatakan bahwa *option* ini wajib ada.
  Parameter: variabel bertipe *boolean* yang menentukan apakah *option* ini wajib ada.
  Kembalian: objek *Option.Builder* yang bisa digunakan untuk *method chaining*.

- public Option build()
  Berfungi untuk membuat objek *Option* berdasarkan nilai pada *Option.Builder*.
  Kembalian: objek *Option*.

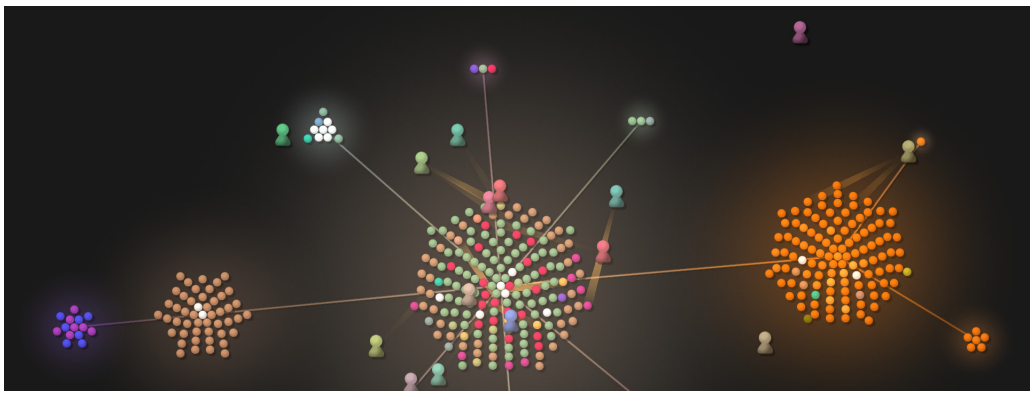
BAB 3

ANALISIS

Pada bab ini dijelaskan mengenai analisis aplikasi sejenis, analisis penggunaan JGit dan Selenium WebDriver untuk membangkitkan animasi timelapse, prapengujian, dan analisis fitur aplikasi yang dibangun.

3.1 Analisis Aplikasi Sejenis

Saat skripsi ini dibuat, aplikasi sejenis yang digunakan untuk membangkitkan animasi adalah Gource. Proyek perangkat lunak ditampilkan oleh Gource sebagai animasi pohon, dimana pusatnya adalah *root directory* dari proyek perangkat lunak[9]. Direktori ditampilkan sebagai *branch*, sedangkan *file* ditampilkan sebagai *leaf*. Developer dapat terlihat di *working tree* pada saat mereka berkontribusi untuk proyek.



Gambar 3.1: Visualisasi proyek perangkat lunak menggunakan Gource.

Gambar 3.1 menunjukkan contoh visualisasi proyek perangkat lunak menggunakan Gource. Efek cahaya yang terdapat pada Gambar 3.1 disebut dengan *bloom*. Pada awalnya ukuran *working tree* tidak terlalu besar. Setiap kali ditambahkan *file* dan *folder* baru, akan dibuat *branch* dan *leaf* baru pada *working tree*.

Gource memiliki beberapa fitur. Fitur-fitur tersebut dapat diatur melalui *command line options*. Berikut ini adalah beberapa *command line options* yang terdapat pada Gource:

1. `gource -[WIDTH]x[HEIGHT]`
   Opsi ini berfungsi untuk mengatur resolusi layar dari animasi. Parameter dari opsi ini adalah lebar dan panjang layar dalam satuan piksel.

2. `gource -camera-mode [MODE]`
   Opsi ini berfungsi untuk mengatur mode kamera pada Gource. Parameter dari opsi ini adalah mode dari kamera. Terdapat dua mode yaitu *overview* dan *track*. Dalam mode *track*, kamera

bergerak mengikuti *user* yang sedang aktif. Dalam mode *overview*, kamera menampilkan seluruh repositori.

3. `gource -path [PATH]`
   Opsi ini berfungsi untuk mengatur *path* dari direktori yang akan dibuat animasinya. Opsi dari parameter ini adalah *path* dari direktori.

4. `gource -start-date [YYYY-MM-DD hh:mm:ss +tz] -stop-date [YYYY-MM-DD hh:mm:ss +tz]`
   Opsi ini berfungsi untuk mengatur periode waktu dalam menampilkan animasi. Parameter dari opsi ini adalah waktu mulai dan waktu akhir dalam format "YYYY-MM-DD hh:mm:ss +tz". Dimana YYYY adalah tahun, MM adalah bulan, DD adalah tanggal, hh adalah jam, mm adalah menit, ss adalah detik, dan +tz adalah zona waktu. Parameter jam, menit, detik, dan zona waktu bersifat opsional.

5. `gource -bloom-multiplier [FLOAT]`
   Opsi ini berfungsi untuk mengatur radius dari efek *bloom*. Parameter dari opsi ini adalah radius dalam format bilangan riil.

6. `gource -bloom-intensity [FLOAT]`
   Opsi ini berfungsi untuk mengatur intensitas dari efek *bloom*. Parameter dari opsi ini adalah intensitas *bloom* dalam format bilangan riil.

7. `gource -disable-bloom`
   Opsi ini berfungsi untuk menonaktifkan animasi *bloom*.

8. `gource -date-format [FORMAT]`
   Opsi untuk mengatur format waktu yang ditampilkan pada bagian tengah atas. Opsi dari parameter ini adalah format waktu dalam bentuk *string*.

9. `gource -background [FFFFFF]`
   Opsi ini berfungsi untuk mengatur warna *background*. Parameter dari opsi ini adalah warna dalam format heksadesimal.

10. `gource -background-image [IMAGE]`
    Opsi ini berfungsi untuk mengatur gambar *background*. Parameter dari opsi ini adalah nama *file* dari gambar.

11. `gource -font-size [SIZE]`
    Opsi ini digunakan untuk mengatur ukuran *font* pada tulisan *title* dan tanggal. Parameter dari opsi ini adalah ukuran *font*.

12. `gource -font-colour [FFFFFF]`
    Opsi ini digunakan untuk mengatur warna *font* pada tulisan *title* dan tanggal. Parameter dari opsi ini adalah warna *font* dalam format heksadesimal.

13. `gource -logo [IMAGE]`
    Opsi ini berfungsi untuk memasukkan logo. Parameter dari opsi ini adalah nama *file* dari gambar.

14. `gource -logo-offset [X]x[Y]`
    Opsi ini berfungsi untuk mengatur posisi dari logo. Parameter dari opsi ini adalah posisi x dan posisi y dari logo.

15. `gource -title [TITLE]`
    Opsi ini berfungsi untuk memberi judul. Dimana judul tersebut ditampilkan pada pojok kiri bawah layar.

16. `gource -output-framerate [FPS]`
    Opsi ini berfungsi untuk mengatur jumlah *frame* per detik pada video animasi. Parameter dari opsi ini adalah jumlah *frame* per detik.

17. `gource -seconds-per-day [SECONDS]`
    Opsi ini berfungsi untuk mengatur kecepatan visualisasi. Parameter dari opsi ini adalah waktu yang dibutuhkan untuk menampilkan satu hari.

18. `gource -hide [DISPLAY-ELEMENT]`
    Opsi ini berfungsi untuk menyembunyikan satu atau lebih *display element*. Parameter dari opsi ini adalah elemen yang akan disembunyikan. *Display element* yang dapat disembunyikan yaitu:

    - *bloom*: efek *bloom*.
    - *date*: waktu.
    - *dirnames*: nama direktori.
    - *files*: ikon dari berkas.
    - *filenames*: nama berkas.
    - *root*: *root directory*.
    - *users*: ikon dari *user*.
    - *usernames*: nama dari *user*.

    Parameter yang berjumlah lebih dari satu dipisahkan dengan koma, contoh: *bloom*,*root*,*users*.

Gource dapat digunakan untuk berbagai macam proyek perangkat lunak. Program pada skripsi ini hanya akan berfokus untuk proyek perangkat lunak berbasis *web*. Tidak seperti Gource yang menampilkan direktori dan *file* pada animasi, program pada skripsi ini menampilkan *screenshot* dari halaman suatu *website*.

3.2 Analisis Penggunaan JGit dan Selenium WebDriver untuk membangkitkan animasi timelapse

Pada subbab ini dijelaskan mengenai analisis penggunaan JGit, analisis penggunaan Selenium WebDriver, dan langkah-langkah dalam membangkitkan animasi timelapse.

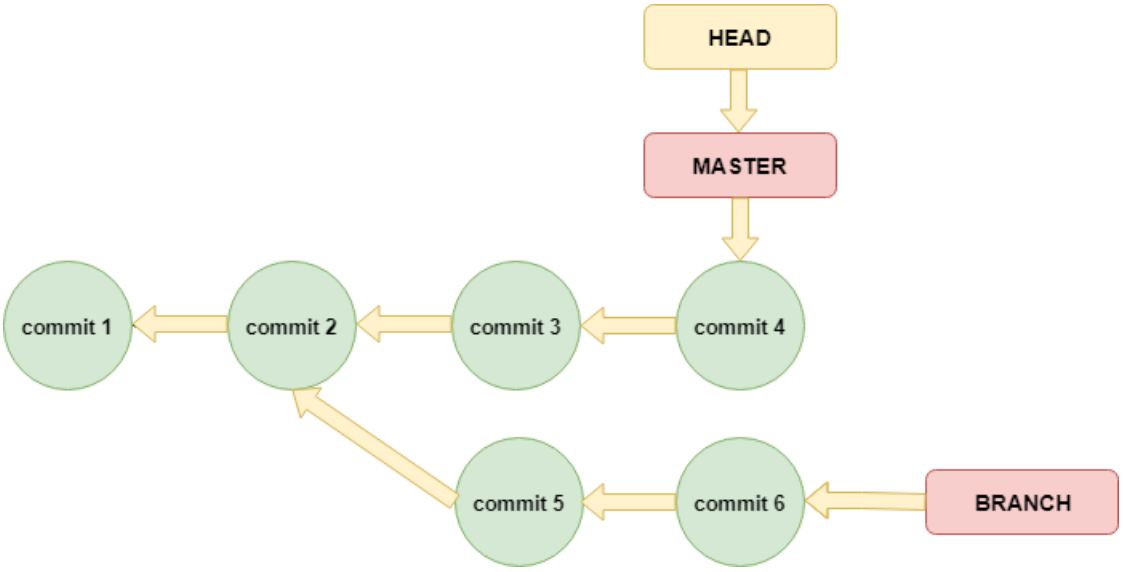
3.2.1 Analisis Penggunaan JGit

JGit dapat digunakan untuk berinteraksi dengan repositori yang terekam oleh Git. Dimana interaksi ini dapat dilakukan melalui program java. Pada analisis ini dibahas kelas-kelas pada *library* JGit yang digunakan untuk berinteraksi dengan suatu repositori Git. Kelas-kelas yang dipakai yaitu Git, Repository, FileRepository, dan RevCommit.

Untuk dapat berinteraksi dengan suatu repositori Git diperlukan kelas Repository dan Git. Kelas Repository dan FileRepository merepresentasikan suatu repositori Git. Kelas FileRepository merupakan turunan dari kelas Repository. Operasi-operasi pada Git dapat dilakukan dengan menggunakan kelas Git. *Constructor* dari kelas Git menerima parameter bertipe Repository. Repository bersifat abstrak, karena itu tidak bisa diinisialisasi secara langsung. Repository dapat diinisialisasi menggunakan *object* yang bertipe FileRepository. Dimana *constructor* FileRepository menerima parameter berupa alamat dari direktori Git. Gambar **??** merupakan contoh suatu repositori Git.

Untuk mendapatkan histori *commit* pada suatu repositori Git digunakan operasi Git Log. Sintaks untuk melakukan operasi Git Log adalah `git.log().all().call();`. Dengan sintaks tersebut,

akan didapatkan seluruh histori *commit* pada *branch* yang sedang aktif. Seluruh histori *commit* akan disimpan dalam bentuk Iterable<RevCommit>. Histori *commit* pada Git direpresentasikan dalam bentuk Directed Acyclic Graph seperti pada Gambar 3.2.



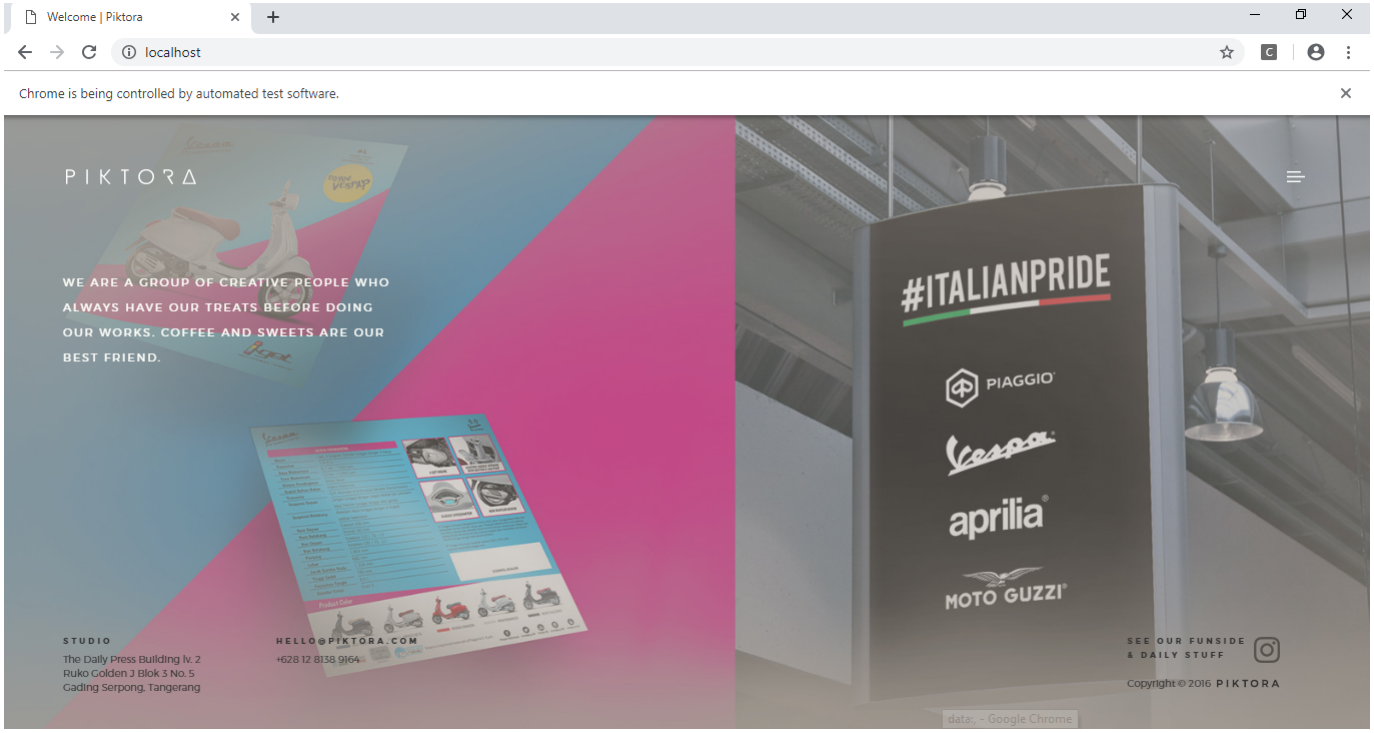
Gambar 3.2: Histori *commmit* direpresentasikan sebagai Directed Acyclic Graph.

Operasi *checkout* pada Git dapat digunakan mengubah *working directory* sesuai dengan *commit* tertentu. Operasi ini dilakukan dengan menggunakan kelas Git. Sintaks untuk melakukan operasi *checkout* adalah `git.checkout().setName(commitID).call();`. *Method* checkout() mengembalikan *object* CheckoutCommand. Pada *object* CheckoutCommand terdapat *method* setName() dengan parameter berupa *commit* ID. Untuk menjalankan operasi *checkout*, *method* call() milik CheckoutCommand dipanggil.

3.2.2 Analisis Penggunaan Selenium WebDriver

Selenium WebDriver dapat digunakan untuk mengotomatisasi *web browser*. Pada analisis ini dibahas kelas-kelas pada *library* Selenium WebDriver yang digunakan untuk mengotomatisasi *web browser*. Kelas-kelas yang dipakai yaitu WebDriver, ChromeDriver, dan TakesScreenshot.

WebDriver merupakan representasi dari *browser* yang digunakan untuk *automation testing*. WebDriver merupakan suatu *interface*. WebDriver dapat diinisialisasi menggunakan *object* dengan tipe ChromeDriver. Setelah melakukan inisialisasi pada WebDriver, *browser* akan dijalankan. Untuk membuka suatu halaman *web*, digunakan *method* get() dengan parameter alamat URL. Gambar 3.3 menunjukkan sebuah *browser* yang dikontrol oleh ChromeDriver.



Gambar 3.3: *Browser* yang dikontrol oleh ChromeDriver.

TakesScreenshot merupakan *interface* yang digunakan untuk menangkap *screenshot* halaman *web* pada WebDriver. Kelas ChromeDriver mengimplementasikan *interface* ini. *Method* yang digunakan untuk menangkap *screenshot* adalah getScreenshotAs(), diikuti dengan parameter bertipe OutputType. OutputType ini merupakan *interface* yang digunakan untuk menentukan tipe *output* dari hasil *screenshot*. OutputType yang umum digunakan adalah File.

3.2.3 Langkah-Langkah dalam Membangkitkan Animasi Timelapse

Setelah melakukan analisis pada subbab 3.2.1 sampai subbab 3.2.2, dapat dibuat langkah-langkah untuk membangkitkan animasi *timelapse*. Langkah-langkah untuk membangkitkan animasi *timelapse* adalah sebagai berikut:

1. Program membaca argumen Command Line menggunakan Apache Commons CLI.
2. Program mendapatkan seluruh *commit* histori dari proyek perangkat lunak berbasis web menggunakan JGit.
3. Program membuka semua *browser* menggunakan Selenium WebDriver. Jumlah *browser* bergantung pada jumlah argumen dari Option `-capture-url`.
4. Program melakukan *checkout* dalam suatu rentang *commit* tertentu. Jika tidak terdapat Option `-start-commit` dan Option `-stop-commit`, akan dilakukan *checkout* ke semua *commit*.
5. Program melakukan *checkout* ke suatu *commit*.
6. Program menjalankan *script* PHP jika terdapat Option `-before-capture`.
7. Program membuka setiap URL yang didapatkan dari Option `-capture-url` menggunakan Selenium WebDriver. Setiap *browser* membuka URL yang berbeda.
8. Selenium WebDriver kemudian mengambil *screenshot* pada semua *browser*.

9. Jika saat ini sedang berada pada *commit* terakhir, lanjut ke langkah berikutnya. Jika tidak, ulangi langkah 5-8 untuk *commit* selanjutnya.
10. Jika terdapat lebih dari satu *browser*, hasil *screenshot* dari setiap *browser* digabungkan menjadi satu gambar baru.
11. Program menambahkan judul di pojok kiri bawah jika terdapat Option `-title`.
12. Program menambahkan logo di pojok kanan bawah jika terdapat Option `-logo`.
13. Menggabungkan semua *file* gambar menjadi satu *file* bertipe GIF.

Penulis menggunakan *library* yang didapatkan dari internet[1] untuk menggabungkan gambar menjadi *file* GIF.

3.3 Prapengujian Website Piktora

Pengujian dilakukan dengan proyek Piktora sebagai input dari program. Piktora merupakan *website* yang menawarkan layanan *creative design* dan *branding*. Layanan yang ditawarkan berupa *graphic design* untuk poster, *banner*, *website*, dan aplikasi *mobile*. Alamat dari *website* ini adalah[2]. Alamat repositori dari perangkat lunak ini adalah[3].

Prapengujian ini hanya menggunakan dua input. Input dimasukkan melalui argumen Command Line. Input dari program ini adalah alamat direktori proyek Piktora dan satu URL untuk membuka halaman *web*. Langkah pada pengujian menggunakan langkah yang terdapat pada subbab 3.2.3.

Pada proyek Piktora terdapat 58 *commit*. Listing 3.1 menunjukkan histori *commit* pada proyek Piktora. Setelah dilakukan pengujian terdapat beberapa masalah. Masalah tersebut yaitu perbedaan letak *file*, migrasi *database*, dan konfigurasi *database*. Masalah-masalah tersebut dibahas pada subbab 3.3.1 sampai 3.3.3.

Listing 3.1: Histori *commit* pada proyek Piktora

```
1  315d374 - Oct 31 16:52:46 2016 : Basic CI files + htaccess & webconfig + database.
       php ignore
2
3  27ce3d4 - Nov 5 13:12:43 2016 : setup environment for piktora
4  65f0c9c - Nov 5 19:22:58 2016 : * create structure for all pages * add dummy
       images
5
6  bffbae1 - Nov 8 18:00:32 2016 : - basic structure (navbar semi complete) - add
       fonts
7
8  5c59916 - Nov 8 19:51:18 2016 : implement navbar, footer, and projects/ page
9  7738380 - Nov 8 20:05:27 2016 : fix pc and ipad navbar fontsize
10 26bdbee - Nov 8 20:16:33 2016 : fix position image for desktop /projects
11 3db3ce8 - Nov 9 00:28:27 2016 : - implements project details page - fix some minor
       issue - add some project image
12
13 5ef34fa - Nov 13 13:01:06 2016 : implement about us (raw version)
14 3caf535 - Nov 15 11:55:15 2016 : fix minor issues view/about: - background-image
       position. Make it to the center position - slick.js img need to set to inline
15
16 c5eb3b6 - Nov 15 13:02:42 2016 : implement /welcome page
17 ade9216 - Nov 15 13:12:08 2016 : fix minor issues: - move style footer to global -
       add space before PIKTORA
18
19 c77b5b3 - Nov 18 18:18:25 2016 : implement /contact
```

[1] http://elliot.kroo.net/software/java/GifSequenceWriter
[2] http://www.piktora.com
[3] https://gitlab.com/PNDevworks/Piktora

```
20 b42b819 - Nov 18 21:22:22 2016 : cange a href to style cursor: pointer
21 3eb7af8 - Nov 21 16:09:40 2016 : .htaccess to be compatible with cloud kilat
22 e87e84b - Nov 22 14:53:45 2016 : - change vw to 100% - add captcha
23 ff8d829 - Nov 22 15:22:53 2016 : Solved captcha font load: use otf instead of ttf
       Also: create directory assets/img/captcha and ignore everything inside
24
25 dc87342 - Nov 22 15:49:03 2016 : - implement captcha code - remove wrong css
26 9ebe433 - Nov 22 16:12:44 2016 : add scroll feature in project/
27 f0f7270 - Nov 23 15:17:42 2016 : Added Google PHP Client v2 See https://github.com
       /google/google-api-php-client
28
29 57a239b - Nov 23 15:19:57 2016 : Merge origin/master
30 e2dfebe - Nov 27 14:42:24 2016 : - remove blue outline when click with slick -
       change background image in about to newest one
31
32 a19e7f2 - Nov 27 15:24:42 2016 : Detailing dari Edina: 1. Hal. Project Detail,
       font coba diperkecil saja mungkin ya. 3. Beberapa ukuran font dan spacing ada
       yang kurang pas sedikit, terlampir detail revisinya ya (file pdf) 4. Footer
       dibuat selalu stay terus di bagian bawah dengan posisi yang selalu sama. Di home
        & contact sudah sama, namun di hal. product posisinya agak lebih naik.
33
34 3d79d0a - Nov 27 15:29:58 2016 : fix minor issue
35 0fcd958 - Nov 28 10:11:06 2016 : add raw admin contents
36 add3974 - Nov 28 12:03:21 2016 : add summernote, implement read project
37 0680488 - Nov 29 12:38:23 2016 : update admin for projects
38 fbe7639 - Nov 29 13:10:30 2016 : implement admin for home
39 db0cedd - Nov 29 13:38:29 2016 : - implementasi database bagian user - upload 9
       gambar contoh project
40
41 0fe9aaf - Nov 29 14:17:20 2016 : ubah warna garis captcha
42 f2326dd - Nov 29 14:44:51 2016 : - lewati proses otentikasi sementara
43 f78cdb4 - Dec 2 12:10:47 2016 : (trying to) fix issue #2
44 ef9b62b - Dec 2 17:09:58 2016 : revisi dari edina ke-2
45 c689aa8 - Dec 2 17:11:13 2016 : perbaikan admin sedikit
46 c4e9576 - Dec 2 17:14:06 2016 : perbaikan di /contact, kelewat
47 02d04f1 - Dec 5 14:55:20 2016 : tambah wording
48 a4e4858 - Dec 5 15:08:59 2016 : perbaikan kata2 sedikit
49 bbd82c2 - Dec 6 10:41:40 2016 : implementasi email
50 f8c64fc - Dec 6 11:03:08 2016 : change to httpdocs
51 eb49c2b - Dec 6 11:35:20 2016 : hapus migrasi script di admin
52 ace1988 - Dec 6 11:39:00 2016 : change overflow to auto
53 627e65b - Dec 6 11:45:26 2016 : modify database back to local
54 0896f81 - Dec 7 16:08:30 2016 : update home versi mobile jadi baru (revisi dari
       Edina)
55
56 5cf1292 - Dec 7 16:21:01 2016 : ubah background di about menjadi tidak pecah
57 c83f4aa - Thu Dec 15 15:04:30 2016 : remove piktora secrets
58 57f5ea4 - Thu Dec 15 15:09:43 2016 : remove unimportant data
59 7931c21 - Dec 24 18:40:41 2016 : edit wording
60 9b0a302 - Dec 25 06:03:50 2016 : Another wording fix
61 f1ea410 - Thu Jan 5 15:23:32 2017 : fix instagram link
62 1880a88 - Thu Jan 5 15:24:12 2017 : Merge branch 'master' of https://github.com/
       pascalalfadian/Piktora
63
64 286aa78 - Jan 16 12:48:45 2017 : Perbaikan wording di admin edit project
65 33702c2 - Feb 21 13:31:08 2017 : change email sender to piktora@mailgun.dnartworks.
       com.au
66
67 18c39ef - Thu Apr 13 15:21:49 2017 : Test commit (in gitlab). Nothing much important
68 9bfde3c - Apr 17 15:09:54 2017 : add ignore sftp-config.json
69 38711f0 - Apr 17 15:15:03 2017 : fix bug ugly display when projects too high
70 9f041ef - May 15 10:40:16 2017 : set insta url to https://www.instagram.com/
       piktorastudio/
```

```
71
72 6a085c1 - Dec 12 14:38:38 2017 : Update company address
73 89000be - Jan 12 12:25:30 2018 : Update new company address
```

3.3.1 Perbedaan Letak *File*

Pada *commit* 315d374(31 Oktober 2016) s.d. bbd82c2(6 Desember 2016) halaman *web* proyek Piktora tidak bisa dibuka. Hal ini disebabkan oleh perbedaan letak *file* "index.php". Pada *commit* 315d374(31 Oktober 2016) s.d. bbd82c2(6 Desember 2016), *file* "index.php" berada pada *folder* "www", sedangkan pada *commit* f8c64fc(6 Desember 2016) s.d. 89000be(12 Januari 2018) *file* "index.php" berada pada *folder* "httpdocs".

Solusi untuk mengatasi masalah ini adalah dengan menambahkan *command line options* yang menerima sebuah *script* PHP. *Script* PHP ini akan mengecek letak *file* "index.php" pada *folder* "www" dan "httpdocs". *Script* kemudian akan mengecek *directory root* XAMPP pada *file* "httpd.conf". Jika *directory root* sudah mengarah ke *folder* tempat "index.php" berada, maka *script* tidak akan mengubah isi *file* "httpd.conf". Jika *directory root* tidak mengarah ke *folder* tempat "index.php" berada, maka *script* akan mengubah *directory root* pada *file* "httpd.conf". Setelah itu, *script* akan menjalankan perintah "httpd -k restart" untuk melakukan *restart* pada XAMPP. *Script* PHP ini akan dijalankan sebelum melakukan migrasi *database*.

3.3.2 Permasalahan Konfigurasi *Database*

Pada *commit* f8c64fc(6 Desember 2016) s.d. ace1988(6 Desember 2016), halaman *web* tidak bisa dibuka. Hal ini disebabkan karena perbedaan konfigurasi pada *file* "database.php". Pada *commit* f8c64fc(6 Desember 2016) s.d. 57f5ea4(15 Desember 2016) dan *commit* f1ea410(5 Januari 2017), di dalam *file* "database.php" terdapat *password* . *Commit* lainnya tidak terdapat *password* pada *file* "database.php". Penulis menggunakan *password* `"piktora"` pada konfigurasi *database* di XAMPP.

Listing 3.2: Isi *file* "database.php" pada *commit* f8c64fc(6 Desember 2016)

```
1  $active_group = 'default';
2  $query_builder = TRUE;
3
4  $db['default'] = array(
5   'dsn' => '',
6   'hostname' => 'localhost',
7   'username' => 'piktora',
8   'password' => 'dmHx64%6',
9   'database' => 'piktora',
10  'dbdriver' => 'mysqli',
11  'dbprefix' => '',
12  'pconnect' => FALSE,
13  'db_debug' => (ENVIRONMENT !== 'production'),
14  'cache_on' => FALSE,
15  'cachedir' => '',
16  'char_set' => 'utf8',
17  'dbcollat' => 'utf8_general_ci',
18  'swap_pre' => '',
19  'encrypt' => FALSE,
20  'compress' => FALSE,
21  'stricton' => FALSE,
22  'failover' => array(),
23  'save_queries' => TRUE
24 );
```

Listing 3.2 merupakan isi dari *file* "database.php" pada *commit* f8c64fc(6 Desember 2016). Dapat dilihat bahwa *password* yang terdapat pada *file* "database.php" adalah `dmHx64%6`. *Commit* f8c64fc(6 Desember 2016) s.d. ace1988(6 Desember 2016) menggunakan *password* `dmHx64%6`, sedangkan *commit* 627e65b(6 Desember 2016) s.d. 57f5ea4(15 Desember 2016) dan f1ea410(5 Januari 2017)

menggunakan *password* `piktora`. Karena konfigurasi *password* pada *file* "database.php" dan XAMPP berbeda, halaman *website* pada *commit* f8c64fc(6 Desember 2016) s.d. ace1988(6 Desember 2016) tidak bisa dibuka.

Sama seperti subbab 3.3.1, solusi untuk mengatasi masalah ini adalah dengan menambahkan *command line options* yang menerima sebuah *script* PHP. *Script* ini akan mengecek *password* yang terdapat pada *file* "database.php". Jika tidak ditemukan *password* atau ditemukan *password* berupa `piktora`, maka *script* tidak akan mengubah isi *file* "database.php". Jika ditemukan *password* berupa `dmHx64%6`, maka *script* akan mengubah *password* menjadi `piktora`.

3.3.3 Permasalahan Migrasi *Database*

Pada *commit* 3d79d0a(27 Nov 2016), terjadi *error* saat melakukan migrasi database. Pada *commit* a19e7f2(27 November 2016), terdapat satu *file* untuk melakukan migrasi yaitu "20161122150000_Structure.php". Pada *commit* 3d79d0a(27 Nov 2016) terdapat dua *file* untuk melakukan migrasi yaitu "20161122150000_Structure.php" dan "20161122150001_InitialData.php". Pada *commit* a19e7f2(27 November 2016) *file* "20161122150001_InitialData.php" dijalankan saat melakukan migrasi. Versi migrasi *database* menjadi "20161122150000". Pada *commit* 3d79d0a(27 Nov 2016), *file* "20161122150000_Structure.php" tidak dijalankan karena dianggap sama dengan versi migrasi *database* saat ini. Hanya *file* "20161122150001_InitialData.php" yang dijalankan pada *commit* 3d79d0a(27 Nov 2016). Isi *file* "20161122150001_InitialData.php" pada *commit* a19e7f2(27 November 2016) dan 3d79d0a(27 Nov 2016) berbeda. Hal ini yang menyebabkan terjadinya *error* saat melakukan migrasi database.

Sama seperti subbab 3.3.1, solusi untuk mengatasi masalah ini adalah dengan menambahkan *command line options* yang menerima sebuah *script* PHP. *Script* ini akan melakukan dua pekerjaan. Pertama, *script* akan menghapus *database* piktora dengan menggunakan *query* "DROP DATABASE piktora". Setelah itu *script* akan membuat *database* piktora dengan menggunakan *query* "CREATE DATABASE piktora". *Script* akan dijalankan sebelum melakukan migrasi *database*.

3.4 Analisis Fitur Aplikasi yang Dibangun

Pada skripsi ini, dibuat sebuah perangkat lunak yang dapat membangun animasi *timelapse* dari pengembangan proyek perangkat lunak berbasis web. Yang akan dibuat animasinya adalah halaman web dari perangkat lunak. Jumlah halaman berkisar antara satu sampai empat halaman, tergantung pada masukan dari *user*. *Output* dari program adalah *file* hasil animasi yang bertipe GIF. Program dapat menerima masukan dan konfigurasi dari Command Line Options.

Berikut ini adalah *command line options* yang akan diimplementasikan pada skripsi ini:

1. `-project-path [PATH]`
   Opsi ini berfungsi untuk mengatur *path* dari direktori yang akan dibuat animasinya. Parameter dari opsi ini adalah *path* dari proyek perangkat lunak web yang terekam oleh Git. Opsi ini wajib ada.

2. `-before-capture [PHP-SCRIPT]`
   Opsi ini berfungsi untuk menjalankan *script* PHP. *Script* ini dijalankan sebelum melakukan *screenshot*. Parameternya adalah *path* dari *script* PHP. Opsi ini bersifat opsional.

3. `-capture-url [URL 1, URL 2, URL 3, URL 4]`
   Opsi ini berfungsi untuk mengatur alamat *url* dari halaman *web*, dimana dilakukan pengambilan *screenshot* pada halaman ini. Jumlah halaman yang dicapture bisa lebih dari satu, dengan jumlah maksimal empat halaman. Parameter dari opsi ini adalah alamat *url* dari halaman *web*, dengan jumlah maksimal empat alamat *url*. Opsi ini wajib ada.

4. `-seconds-per-commit [SECONDS]`
   Opsi ini berfungsi untuk mengatur durasi munculnya satu *commit* pada animasi. Parameter dari opsi ini adalah durasi munculnya satu *commit* dalam satuan detik. Parameter harus berupa bilangan bulat atau riil, dimana nilainya lebih besar dari nol. Opsi ini bersifat opsional.

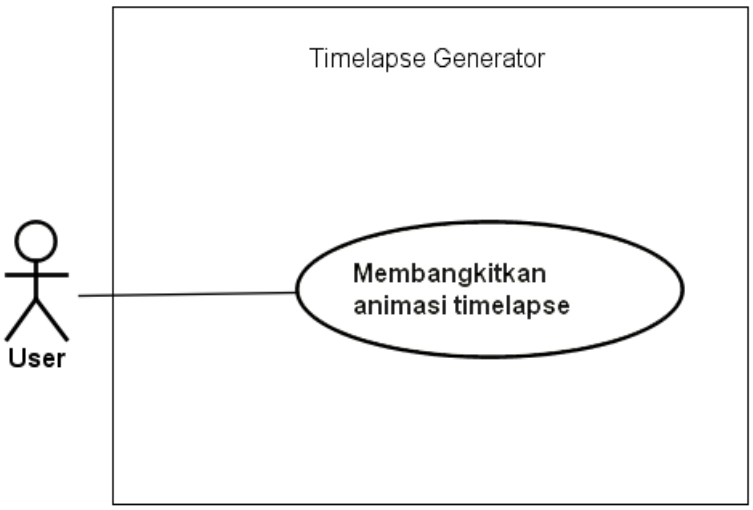
5. `-title [TITLE]`
   Opsi ini berfungsi untuk memberi judul. Dimana judul tersebut ditampilkan pada pojok kiri bawah layar. Opsi ini bersifat opsional.

6. `-logo [IMAGE]`
   Opsi ini berfungsi untuk memasukkan logo. Dimana logo tersebut ditampilkan pada pojok kanan bawah layar. Parameternya adalah *path* dari *file* gambar. Opsi ini bersifat opsional.

7. `-start-commit [ID] -stop-commit [ID]`
   Opsi ini berfungsi untuk mengatur rentang *commit* yang akan dibuat animasinya. Parameter dari opsi ini adalah *commit ID* awal dan *commit ID* akhir, dimana panjang *commit ID* adalah tujuh karakter. Opsi ini bersifat opsional.

Opsi `-path`, `-title`, dan `-image` mengacu pada opsi yang terdapat pada Gource. Opsi `-start-commit -stop-commit` dan `-seconds-per-commit` mengacu pada Gource, dengan sedikit penyesuaian. Opsi `-seconds-per-day` pada Gource menyatakan durasi munculnya satu hari, sedangkan Opsi `-seconds-per-commit` menyatakan durasi munculnya satu *commit*. Opsi `-start-date -stop-date` pada Gource dan opsi `-start-commit -stop-commit` mengatur periode dalam menampilkan animasi. Pada Gource, periode yang digunakan adalah rentang waktu berupa tanggal, bulan, dan tahun. Sedangkan pada opsi `-start-commit -stop-commit`, rentangnya berupa *commit* ID. Opsi `-before-capture` dan `-capture-url` tidak mengacu pada Gource. Kedua opsi tersebut secara khusus dibuat karena program pada skripsi ini membuat animasi *timelapse* dari kumpulan halaman *web*, sedangkan program Gource membuat animasi dari struktur *folder* atau *file*.

Program pada skripsi ini memiliki satu fitur. Fitur tersebut adalah membangkitkan animasi *timelapse*. Penjelasan fitur dapat dilihat pada Gambar 3.4 dan Tabel 3.1.



Gambar 3.4: *Use case diagram* perangkat lunak.

Nama	Membangkitkan animasi *timelapse*
Deskripsi	Fitur untuk membangkitkan animasi timelapse dari pengembangan proyek perangkat lunak berbasis web
Aktor	*User*
Pos-kondisi	File animasi bertipe GIF berhasil dibuat
Skenario	1. Sistem membaca masukan *input* dari Command Line Options. 2. Sistem menelusuri histori perkembangan perangkat lunak berbasis *web* dengan fitur Git. Saat menelusuri histori perkembangan perangkat lunak, sistem mengambil *screenshot* dari halaman web menggunakan SeleniumWebDriver. 3. Sistem menggabungkan file *screenshot* menjadi satu file bertipe GIF.

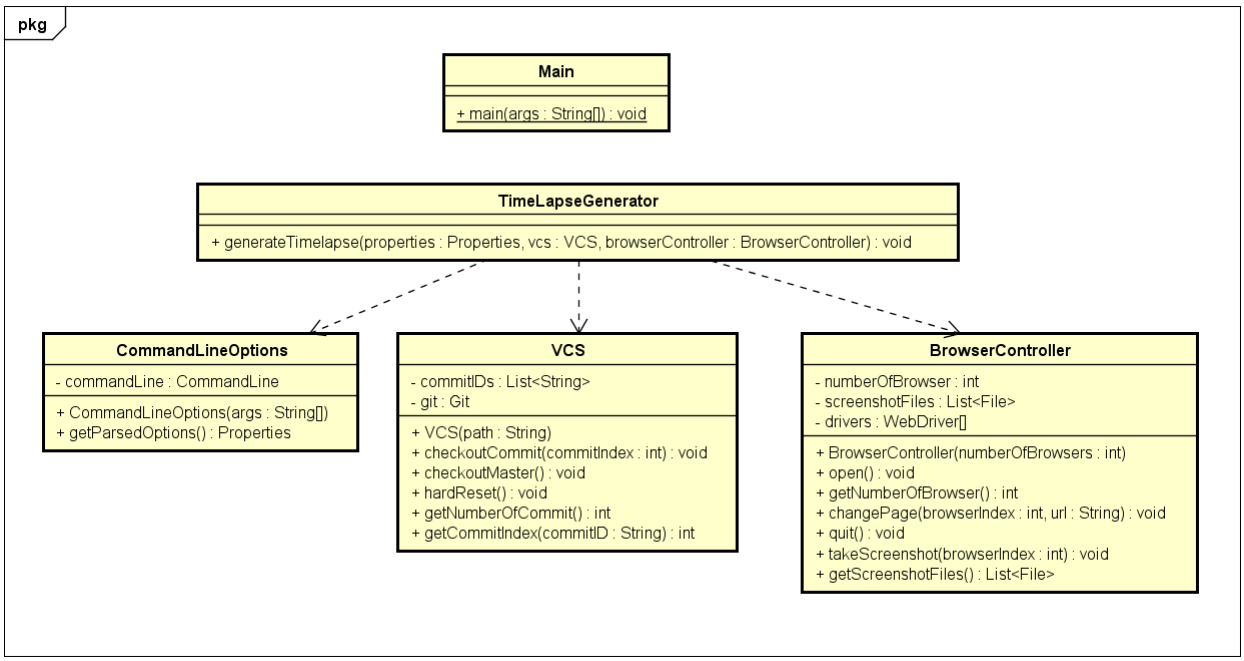
Tabel 3.1: *Scenario case* membangkitkan animasi *timelapse*

BAB 4

PERANCANGAN

Pada bab ini dijelaskan mengenai perancangan perangkat lunak yang dibangun, meliputi perancangan kelas dan perancangan antarmuka.

4.1 Perancangan Kelas



Gambar 4.1: Diagram kelas.

Program pada skripsi ini memiliki lima kelas. Diagram kelas pada program ini dapat dilihat pada Gambar 4.1. Berikut adalah rincian kelas yang terdapat pada program ini:

- BrowserController
  Kelas ini digunakan untuk mengatur *browser*. Operasi-operasi yang dilakukan terhadap *browser* yaitu membuka *browser*, mengambil *screenshot*, membuat *browser window* menjadi maksimal, dan menutup *browser*. Berikut adalah atribut yang terdapat pada kelas ini:
  - private final WebDriver[] drivers
    Atribut ini adalah kumpulan *browser* yang digunakan untuk keperluan *automation testing*.
  - private final List<File> screenshotFiles
    Atribut ini berfungsi untuk menyimpan *file* hasil *screenshot*.

  - private final int numberOfBrowser
    Atribut ini menyatakan jumlah *browser* yang dimiliki oleh kelas ini. Jumlah maksimal dari *browser* adalah empat.

  Berikut adalah *method* yang terdapat pada kelas ini:

  - public BrowserController(int numberOfBrowsers)
    Constructor dari kelas ini. Berfungsi untuk menginisialisasi atribut yang dimiliki oleh kelas ini. Parameternya adalah jumlah *browser* yang dapat dimiliki oleh kelas ini.
  - public void open()
    Berfungsi untuk membuka semua *browser*, kemudian mengatur ukuran *browser window* menjadi maksimal.
  - public int getNumberOfBrowser()
    Berfungsi untuk mengembalikan jumlah browser yang dimiliki kelas ini.
  - public void changePage(int browserIndex, String url)
    Berfungsi untuk berpindah halaman pada *browser* tertentu. Parameternya adalah alamat URL untuk berpindah halaman dan indeks *browser* yang akan diubah halamannya.
  - public void quit()
    Berfungsi untuk menutup semua *browser*.
  - public void takeScreenshot(int browserIndex)
    Berfungsi untuk mengambil *screenshot* pada *browser* tertentu dan menyimpannya ke atribut screenshotFiles. Parameternya adalah indeks *browser* yang akan diambil screenshotnya.

- CommandLineOptions
  Kelas ini berfungsi untuk menyimpan semua Option yang terdapat dalam program ini, dan melakukan *parsing* argumen Command Line Options yang dimasukkan oleh *user*.

  Berikut adalah atribut yang terdapat pada kelas ini:

  - private final CommandLine commandLine
    Atribut ini berfungsi untuk melakukan *parsing* argumen Command Line Options dan menampung hasilnya.

  Berikut adalah *method* yang terdapat pada kelas ini:

  - public CommandLineOptions(String[] args)
    Merupakan Constructor dari kelas ini. Berfungsi untuk menentukan Option yang terdapat pada program dan melakukan parsing argumen Command Line. Parameternya adalah argumen Command Line Option yang didapatkan dari kelas Main.
  - public Properties getParsedOptions()
    Berfungsi untuk mengembalikan Command Line Option yang sudah diparsing.

- VCS
  Kelas ini digunakan untuk berinteraksi pada proyek perangkat lunak yang terekam oleh Git. Berikut adalah atribut yang terdapat pada kelas ini:

  - private final Git git
    Atribut ini digunakan untuk melakukan interaksi pada proyek perangkat lunak yang terekam oleh Git.
  - private final List<String> commitIDs
    Atribut ini digunakan untuk menampung seluruh *commit* ID dari hasil penelusuran histori.

Berikut ini adalah *method* yang terdapat dalam kelas ini:

  - public VCS(String path)
    Constructor dari kelas ini. Berfungsi untuk menginisialisasi variabel git dan mendapatkan seluruh histori commit pada proyek perangkat lunak berbasis web. Parameternya adalah *path* dari proyek perangkat lunak berbasis web.
  - public void checkoutCommit(int commitIndex)
    Berfungsi untuk melakukan *checkout* ke *commit* tertentu. Parameter dari *method* ini adalah indeks dari variabel commitIDs.
  - public void checkoutMaster()
    Berfungsi untuk melakukan *checkout* ke *commit* terakir.
  - public void hardReset()
    Berfungsi untuk melakukan operasi Git Reset. Operasi ini menghapus perubahan pada *working tree* di *commit* tertentu.
  - public int getNumberOfCommit()
    Berfungi untuk mendapatkan jumlah *commit*.
  - public int getCommitIndex(String commitID)
    Berfungsi untuk mendapatkan indeks dari variabel commitIDs. Parameternya adalah *Commit* ID yang akan dicari indeksnya.

- TimeLapseGenerator
  Kelas ini digunakan untuk membangkitkan animasi *timelapse*. Berikut adalah *method* yang dimiliki oleh kelas ini:

  - public void generateTimelapse(Properties properties, VCS vcs, BrowserController browserController)
    Berfungsi untuk membangkitkan animasi *timelapse* berdasarkan langkah-langkah pada Bab 3.2.3. Hasil animasi berupa *file* bertipe GIF. Parameternya adalah objek yang bertipe VCS, BrowserController, dan Properties. Parameter properties menampung key dan value dari Option yang sudah diparsing.

4.2 Perancangan Antarmuka

Program dalam skripsi ini menggunakan dengan terminal sebagai antarmuka, dengan kata lain menggunakan Command Line Interface. *Input* dari program ini dimasukkan melalui argumen Command Line. *Option* yang dapat dimasukkan ke program ini dapat dilihat pada subbab 3.4. *Output* dari program ini berupa status pada terminal dan *file* hasil animasi bertipe GIF.

Listing 4.1 menunjukkan status yang terdapat pada terminal saat program sedang dijalankan dan berhasil membangkitkan animasi *timelapse*. Baris pertama menunjukkan bahwa program sedang membuat animasi *timelapse*. Status pada baris ke-2 sampai baris ke-5 muncul saat program sedang menjalankan WebDriver. Baris terakhir menunjukkan bahwa animasi *timelapse* berhasil dibuat.

Listing 4.1: Status pesan pada terminal saat program sedang dijalankan dan berhasil membangkitkan animasi *timelapse*.

```
1 Membuat animasi timelapse
2 Starting ChromeDriver 2.42.591088 (7b2b2dca23cca0862f674758c9a3933e685c27d5) on port
      16446
3 Only local connections are allowed.
4 Feb 24, 2019 3:26:25 PM org.openqa.selenium.remote.ProtocolHandshake createSession
5 INFO: Detected dialect: OSS
6 Animasi timelapse berhasil dibuat
```

Listing 4.2: Status pesan pada terminal saat program gagal membangkitkan animasi *timelapse*.

```
1 Animasi timelapse gagal dibuat
2 <Error Message>
```

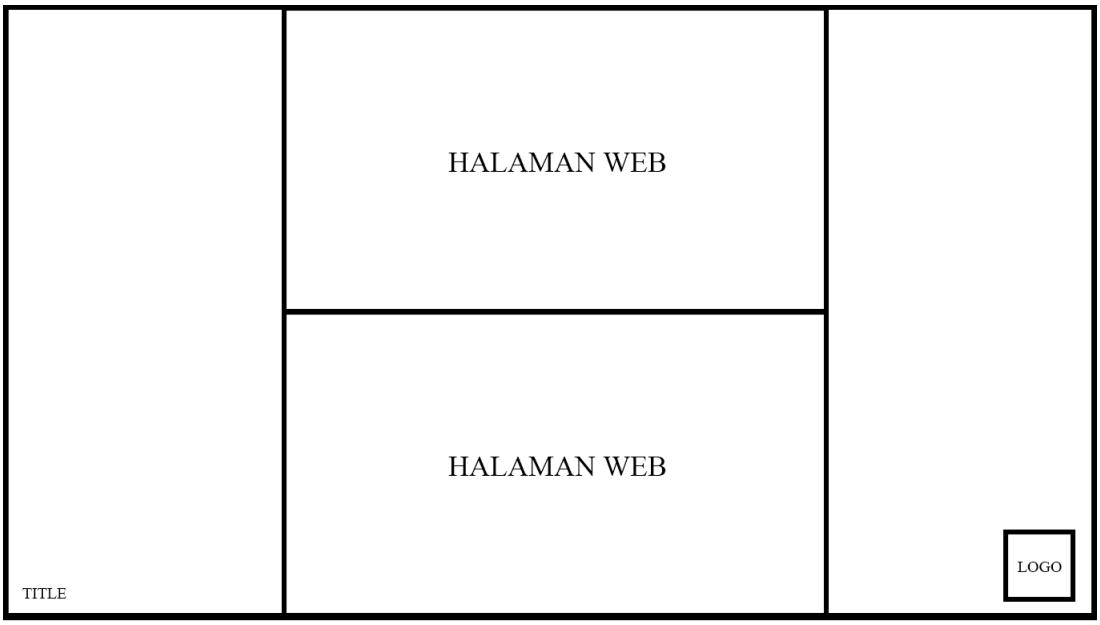
Listing 4.2 menunjukkan status yang terdapat pada terminal saat program gagal membangkitkan animasi *timelapse*. Baris pertama menunjukkan bahwa program gagal membangkitkan animasi *timelapse*. Baris kedua menyatakan *error message* dari program. Program akan gagal membangkitkan animasi *timelapse*, jika *user* memasukkan argumen Command Line yang tidak valid. Berikut ini adalah *error message* yang ditampilkan saat *user* memasukkan *input* yang tidak valid:

- `Missing required option: [OPTION NAME]`
  Pesan ini muncul jika *user* tidak memasukkan Option yang wajib ada. Option yang wajib dimasukkan adalah `seconds-per-commit`, `project-path`, dan `capture-url`.
- `Missing argument for option: [OPTION NAME]`
  Pesan ini muncul jika *user* memasukkan Option tanpa diikuti dengan argumennya. Semua Option yang terdapat pada program ini harus memiliki argumen.
- `Jumlah url yang akan dicapture maksimal 4`
  Pesan ini muncul jika jumlah argumen pada Option `capture-url` lebih dari 4.
- `Seconds per commit harus lebih besar dari 0`
  Pesan ini muncul jika argumen dari `seconds-per-commit` bernilai lebih kecil dari 0.
- `Seconds per commit harus berupa bilangan riil atau bilangan bulat`
  Pesan ini muncul jika argumen dari `seconds-per-commit` bukan bertipe bilangan riil atau bilangan bulat.
- `Capture url tidak valid`
  Pesan ini muncul jika argumen `capture-url` merupakan alamat URL yang tidak valid.
- `Path script PHP tidak valid`
  Pesan ini muncul jika *path* pada argumen Option `before-capture` tidak ditemukan atau `path` bukan merupakan *file* PHP.
- `Path gambar tidak valid`
  Pesan ini muncul jika *path* gambar pada argumen Option `logo` tidak ditemukan atau `path` gambar bukan merupakan *file* gambar.
- `Panjang commit ID awal harus 7 karakter`
  Pesan ini muncul jika panjang Commit ID pada argumen Option `start-commit` tidak sama dengan tujuh karakter.
- `Panjang commit ID akhir harus 7 karakter`
  Pesan ini muncul jika panjang Commit ID pada argumen Option `stop-commit` tidak sama dengan tujuh karakter.
- `Commit ID awal tidak ditemukan`
  Pesan ini muncul jika Commit ID pada argumen Option `start-commit` tidak ditemukan.
- `Commit ID akhir tidak ditemukan`
  Pesan ini muncul jika Commit ID pada argumen Option `stop-commit` tidak ditemukan.
- `Commit ID awal dan akhir tidak boleh sama`
  Pesan ini muncul jika argumen pada Option `start-commit` dan Option `stop-commit` bernilai sama.

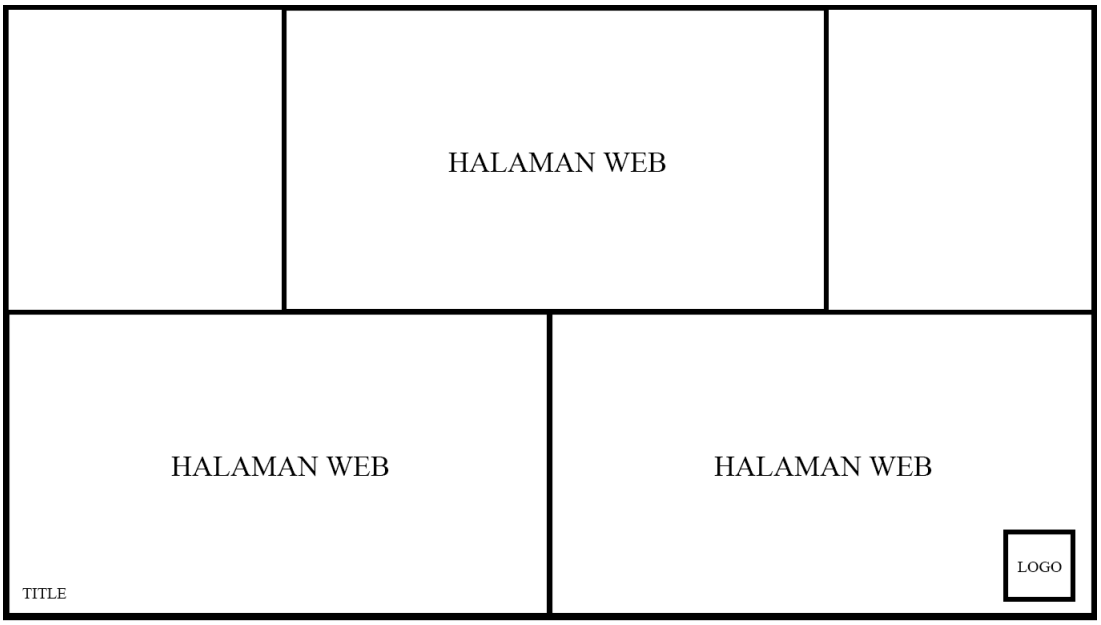
- `Commit ID awal dan akhir terbalik`
  Pesan ini muncul jika nilai argumen pada Option `start-commit` dan Option `stop-commit` tertukar.

Rancangan *output* dari hasil *file* animasi *timelapse* dapat dilihat pada Gambar 4.2 sampai dengan Gambar 4.5. Gambar 4.2 menunjukkan rancangan *output* jika terdapat satu halaman *web*. Gambar 4.3 menunjukkan rancangan *output* jika terdapat dua halaman *web*. Gambar 4.4 menunjukkan rancangan *output* jika terdapat tiga halaman *web*. Gambar 4.5 menunjukkan rancangan *output* jika terdapat empat halaman *web*.

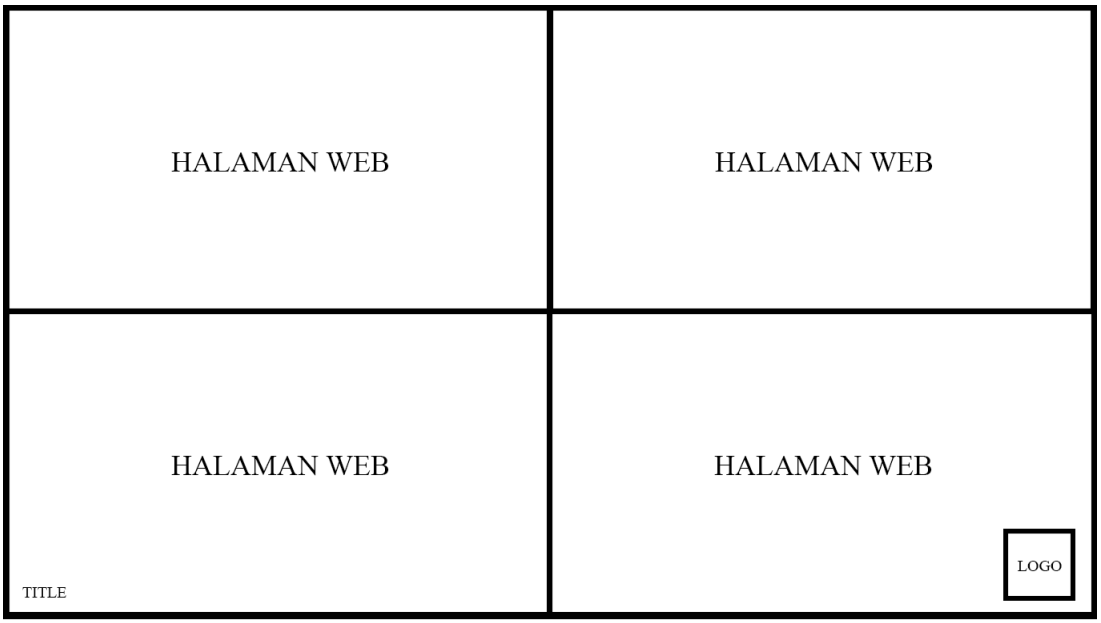
Gambar 4.2: Rancangan *output* satu halaman *web*.



Gambar 4.3: Rancangan *output* dua halaman *web*.



Gambar 4.4: Rancangan *output* tiga halaman *web*.



Gambar 4.5: Rancangan *output* empat halaman *web*.

DAFTAR REFERENSI

[1] Chacon, S. dan Straub, B. (2014) *Pro Git* The expert's voice. Apress.

[2] Jgit | the eclipse foundation. https://www.eclipse.org/jgit/. [Online; diakses 2-September-2018].

[3] Selenium webdriver. https://www.seleniumhq.org/about/. [Online; diakses 2-September-2018].

[4] Jgit - parent 5.0.3.201809091024-r api. http://download.eclipse.org/jgit/site/5.0.3.201809091024-r/apidocs/index.html. [Online; diakses 17-September-2018].

[5] Selenium documentation. https://www.seleniumhq.org/docs/. [Online; diakses 17-September-2018].

[6] Generated documentation. https://seleniumhq.github.io/selenium/docs/api/java/. [Online; diakses 17-September-2018].

[7] Commons - home. https://commons.apache.org/proper/commons-cli/index.html. [Online; diakses 11-Oktober-2018].

[8] Apache commons cli 1.3.1 api. https://commons.apache.org/proper/commons-cli/javadocs/api-release/index.html. [Online; diakses 11-Oktober-2018].

[9] Gource - a software version control visualization tool. https://https://gource.io/. [Online; diakses 29-Oktober-2018].

LAMPIRAN A

KODE PROGRAM

Listing A.1: MyCode.c

```c

// This does not make algorithmic sense,
// but it shows off significant programming characters.

#include<stdio.h>

void myFunction( int input, float* output ) {
    switch ( array[i] ) {
        case 1: // This is silly code
            if ( a >= 0 || b <= 3 && c != x )
                *output += 0.005 + 20050;
            char = 'g';
            b = 2^n + ~right_size - leftSize * MAX_SIZE;
            c = (--aaa + &daa) / (bbb++ - ccc % 2 );
            strcpy(a,"hello $@?");
    }
    count = ~mask | 0x00FF00AA;
}

// Fonts for Displaying Program Code in LATEX
// Adrian P. Robson, nepsweb.co.uk
// 8 October 2012
// http://nepsweb.co.uk/docs/progfonts.pdf
```

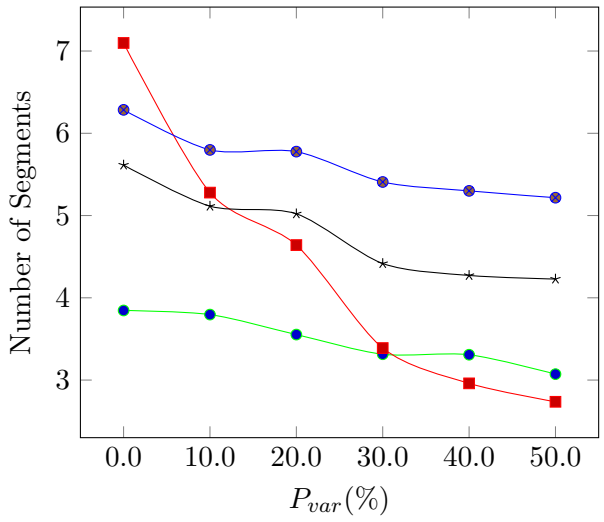
Listing A.2: MyCode.java

```java
import java.util.ArrayList;
import java.util.Collections;
import java.util.HashSet;

//class for set of vertices close to furthest edge
public class MyFurSet {
    protected int id;                                   //id of the set
    protected MyEdge FurthestEdge;                      //the furthest edge
    protected HashSet<MyVertex> set;                    //set of vertices close to furthest edge
    protected ArrayList<ArrayList<Integer>> ordered;    //list of all vertices in the set for each trajectory
    protected ArrayList<Integer> closeID;               //store the ID of all vertices
    protected ArrayList<Double> closeDist;              //store the distance of all vertices
    protected int totaltrj;                             //total trajectories in the set

    /*
     * Constructor
     * @param id : id of the set
     * @param totaltrj : total number of trajectories in the set
     * @param FurthestEdge : the furthest edge
     */
    public MyFurSet(int id,int totaltrj,MyEdge FurthestEdge) {
        this.id = id;
        this.totaltrj = totaltrj;
        this.FurthestEdge = FurthestEdge;
        set = new HashSet<MyVertex>();
        ordered = new ArrayList<ArrayList<Integer>>();
        for (int i=0;i<totaltrj;i++) ordered.add(new ArrayList<Integer>());
        closeID = new ArrayList<Integer>(totaltrj);
        closeDist = new ArrayList<Double>(totaltrj);
        for (int i = 0;i <totaltrj;i++) {
            closeID.add(-1);
            closeDist.add(Double.MAX_VALUE);
        }
    }

}
```

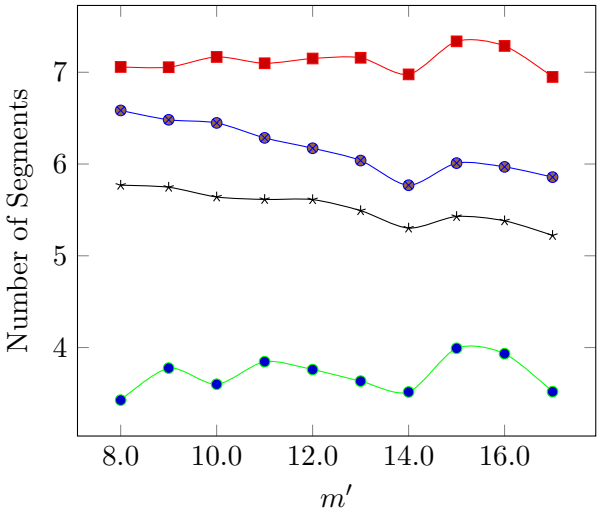
LAMPIRAN B

HASIL EKSPERIMEN

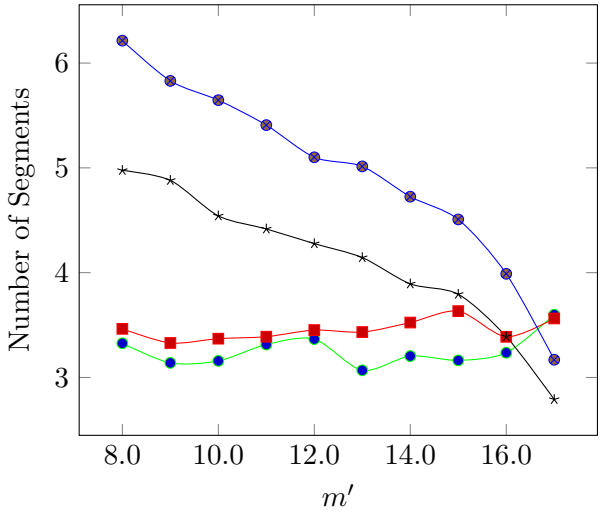
Hasil eksperimen berikut dibuat dengan menggunakan TIKZPICTURE (bukan hasil excel yg diubah ke file bitmap). Sangat berguna jika ingin menampilkan tabel (yang kuantitasnya sangat banyak) yang datanya dihasilkan dari program komputer.



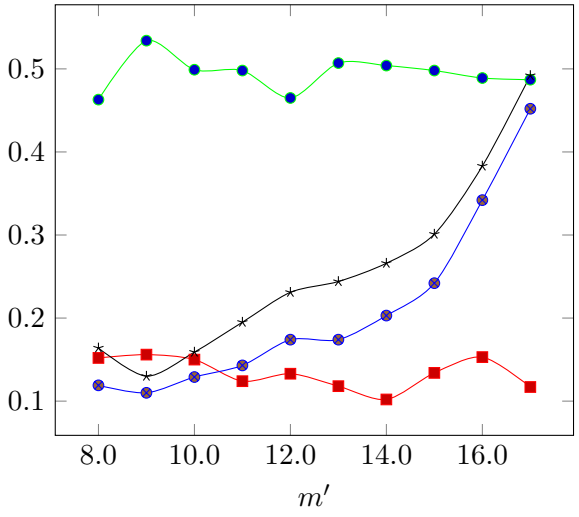
Gambar B.1: Hasil 1



Gambar B.2: Hasil 2



Gambar B.3: Hasil 3



Gambar B.4: Hasil 4