Jurnal Algoritma
E-ISSN: 2302-7339 | P-ISSN: 1412-3622
INSTITUT TEKNOLOGI GARUT

Rancang Bangun Aplikasi Pengelolaan *Bug Tracker* Berbasis *Web* pada Pengembangan *Software*

Sendra Ilham Mandala[1], Leni Fitriani[2], Ade Sutedi[3]

Jurnal Algoritma
Institut Teknologi Garut
Jl. Mayor Syamsu No. 1 Jayaraga Garut 44151 Indonesia
Email : jurnal@itg.ac.id

[1]1706062@itg.ac.id
[2]leni.fitriani@itg.ac.id
[3]ade.sutedi@itg.ac.id

Abstrak – Dalam proses pengambangan perangkat lunak, Suatu tim pengembang berkemungkinan besar mendapatkan *bug* dalam pengembangan perangkat lunak. Untuk mengatasi masalah dibutuhkan suatu alat untuk mengelola *bug* yang ada pada suatu proyek pengembangan perangkat lunak. Tujuan dibuat rancang bangun suatu aplikasi *bug tracker* yaitu untuk memudahkan suatu tim proyek *software development* dalam mengevaluasi, mengoreksi, melacak, dan mengontrol *bug* dan masalah pada suatu proyek. metode yang digunakan adalah *Rational Unified Process* yang memiliki empat tahap, yaitu *Inception*, *Elaboration*, *Construction*, dan *Transition* dan menggunakan *framework* php yaitu laravel. Hasil penelitian ini berupa aplikasi *bug tracker* berbasis *web* untuk mengelola *bug* pada suatu *project* secara *online* yang terdapat fitur seperti *Upload* gambar untuk menambah informasi berupa file gambar, dan fitur *list step to reproduce* berupa informasi *list* berupa *text*.

Kata Kunci – Aplikasi; *Bug Tracker*; *Laravel*; *Rational Unified Process; Web*.

I. PENDAHULUAN

Dalam proses pengambangan perangkat lunak suatu *development team* ataupun seorang *developer* berkemungkinan besar mendapatkan *bug* dan *issue* (masalah). Sebuah *software bug* adalah suatu *flaw* (cacat), error, atau a fault (gagal) pada computer program yang menghasilkan sesuatu yang tidak diinginkan atau hasil yang tidak diprediksi. *bug* berasal dari kesalahan dan blunders *developer* yang membuat *software* design atau pada *source code*. maka, hal ini menjadi sangat penting bagi para *developer team* ataupun seorang *developer* dalam mengelola *bug* [1].

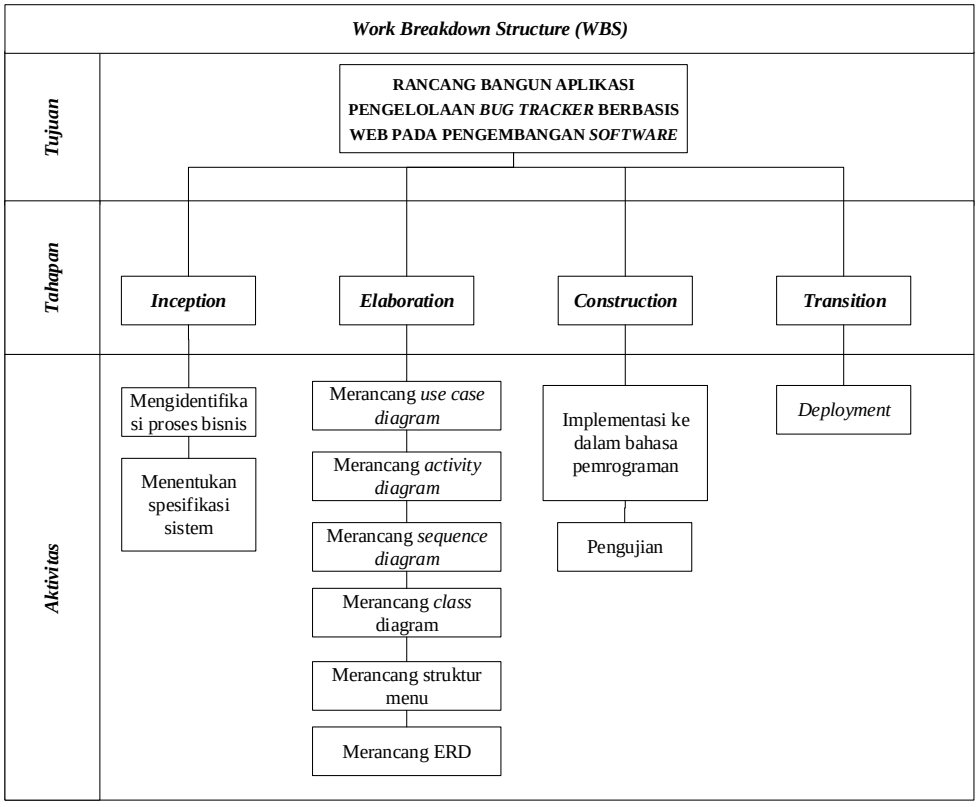
Sebelumnya ada beberapa penelitian tentang pengembangan sistem terkait dengan pengelolaan, pengembangan *software* dan berbasis *website*, penelitian pertama melakukan pengembangan sistem *bug tracking* usulan dan juga melakukan re*view* dan perbandingan terhadap teknologi yang digunakan dalam improvisasi *bug tracking system* dan pembuatan sistem *bug tracker* usulan [2]. Penelitian selanjutnya melakukan penelitian tentang *issue tracker* berbasis *web* untuk *software development* dimana sistem melakukan *maintain bug*, dan mengelola *bug* kedalam detail yang rinci. Aplikasi ini dibuat dengan bahasa pemrograman Java dan Mysql [3]. Penelitian berikutnya yang melakukan survey pada beberapa *bug tracking system* yang ada dimana beberapa *bug tracking system*s yang ada belum efektif dalam mengakumulasi semua informasi yang dibutuhkan oleh *developer* seperti belum adanya fitur informasi file berbentuk gambar, belum adanya *step to reproduce* [4]. Penelitian selanjutnya yang meneliti tentang desain kehadiran karyawan menggunakan teknologi identifikasi frekuensi radio dimana pada pengembangan aplikasi ini dibuat dengan

PHP dan menggunakan metodologi *Rational Unified Process* [5]. Penelitian berikutnya yang melakukan rancang bangun sistem pengelolaan *service level agreement* berbasis *online*, dimana pada penelitian ini perancangannya menggunakan metodologi *Rapid Application Development* [6]. Penelitian berikutnya yang melakukan pengembangan Sistem Informasi Geografis Penggalangan Danadan Donasi Berbasis *web* menggunakan metodologi *Rational Unified Process* [7].

Berdasarkan penelitian sebelumnya beberapa manajemen *bug* dan *issue* masih belum efektif dalam *member*i informasi yang dibutuhkan oleh developer yaitu *screenshot, steps to reproduce*. Oleh karena itu tujuan dari penelitian ini adalah merancang Aplikasi untuk mengelola *bug* yang ada pada suatu proyek pengembangan perangkat lunak dengan menambahkan fitur pemberian informasi yang dibutuhkan oleh user berdasarkan penelitian sebelumnya berupa tambah gambar atau *screenshot* dan *list step to reproduce*.

II. URAIAN PENELITIAN

Metodologi yang digunakan dalam perancangan aplikasi pengelolaan *bug tracker* berbasis *web* adalah *Rational Unified Process* (RUP) yang memiliki empat tahap, yaitu *Inception*, *Elaboration*, *Construction*, dan *Transition* [8]. Disajikan dalam diagram *Work Breakdown Structure* (WBS) seperti yang tampak pada Gambar 1.



Gambar 1: *Work Breakdown Structure*

Adapun penjelasan dari diagram WBS pada Gambar 3.1. yaitu sebagai berikut:

1. *Inception*, tahapan ini merupakan bagian dari persiapan yaitu megidentifikasi proses bisnis yang berjalan, dan Studi literature dengan membaca buku referensi atau dokumentasi yang berhubungan dengan penelitian tentang *bug tracker*.
2. *Elaboration*, pada tahap ini membuat diagram uml yang akan dijadikan desain untuk di implementasikan. UML [9] yang akan dibuat pada tahapan *Elaboration* diantaranya adalah *Usecase* diagram, *Activity* diagram, *Sequence* diagram, *Class* diagram, Merancang struktur menu, Merancang *interface*.
3. *Construction*, pada tahap ini pengimplementasian desain UML yang dibuat kedalam kode pada bahasa

programan PHP dengan menggunakan *framework* laravel [10] dan melakukan pengujian *blackbox* [11];
4. *Transition*, pada tahap ini dilakukan instalasi atau *deployment* sistem.

III. HASIL DAN DISKUSI

A. Hasil

Hasil penelitian adalah aplikasi pengelolaan *bug tracker* berbasis *web* yang di buat dengan menggunakan bahasa pemrograman PHP dan dengan menggunakan *framework* laravel.

B. Pembahasan

1. *Inception*

Pada tahap *Inception* melakukan identifikasi proses bisnis yang didapat dari referensi aplikasi serupa dan jurnal terkait yang ada sebelumnya. Kegiatan ini dilakukan untuk mengetahui segala sesuatu yang berhubungan dengan manajemen *bug tracker,* dan pengembangan perangkat lunak dengan metodologi RUP.

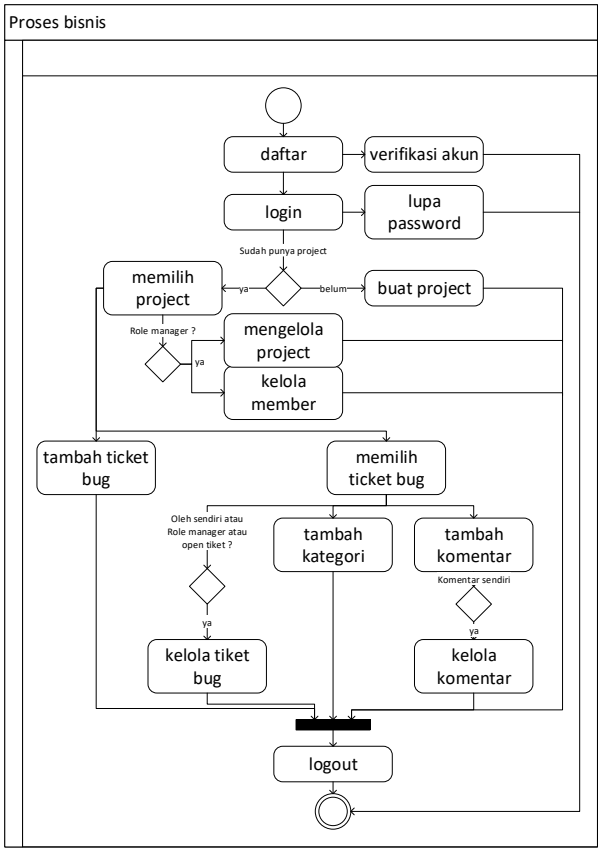
a. Studi literatur, merupakan metode pengumpulan data dengan membaca buku referansi atau dokumentasi yang berhubungan dengan penelitian tentang *bug tracker*. Dengan adanya pengumpulan data ini, sehingga menjadi salah satu penunjang dalam menyelesaikan aplikasi pengelolaan *bug tracker* berbasis *web*.
Studi literature pada penelitian "Analytical Survey on *bug* Tracking *System"*[4] yang *member*ikan tabel data hasil survey berikut:

Tabel 1: Fitur pada beberapa *bugtracker*

No	*Bug Tracker*	*Stack Trace*	*Step To Reproduce*	*Expected Behaviour*	*Test Case*	*Screenshot*	*Dependencies*
1.	*bugzilla*	*No*	*No*	*No*	*No*	*No*	*No*
2.	*Mantis*	*No*	*Yes*	*No*	*No*	*No*	*No*
3.	*bugtracket.net*	*No*	*No*	*No*	*No*	*Yes*	*Yes*
4.	*FlySpray*	*No*	*No*	*No*	*No*	*No*	*No*
5.	*Redmine*	*No*	*No*	*No*	*No*	*No*	*No*
6.	*bugzero*	*No*	*No*	*No*	*Yes*	*No*	*Yes*

Beberpa fitur yang belum terdapat pada sebagian *bug tracker* dan memutuskan untuk membuat fitur berupa tambah gambar atau *screenshot* dan *list step to reproduce.*

b. Identifikasi proses bisnis aplikasi pengelolaan *bug* berbasis *web* dapat digambarkan dengan membuat diagram aktivitas, sebagai tampak pada gambar berikut:



Gambar 2: Diagram aktivitas Proses Bisnis *bug tracker* Berbasis *web*

Proses pengelolaan *bug* dimulai dari user mendaftarkan akun , kemudian login , setelah itu user dapat membuat project , user dapat menambahkan *member* ke project yang user buat , setelah memiliki project user dapat menambahkan tiket *bug* pada project dan *member*ikan tugas tiket tersebut kepada user sendiri atau *member* , user dapat mengelola role pada *member*, tiket *bug* dapat dilihat oleh user dan *member* dari project tersebut , user yang membuat tiket atau *member* yang di beri tiket dapat mengubah tiket status tiket menjadi selesai jika *bug* sudah di perbaiki.

c. Membuat spesifikasi sistem.
   a. Spesifikasi tampilan, aplikasi ini harus memiliki tampilan awal yang mengandung bagian *login*, tampilan *dashboard*, tampilan *project* dan tampilan *ticket bug*, tampilan *settings*;
   b. Persyaratan sistem, yaitu Spesifikasi fungsional dan nonfungsional sistem;
   c. Persyaratan pengembangan yang Menggunakan *UML* sebagai rancangan sistem, dan diimplementasikan menggunakan *visual studio code, xampp , git , github, web browser*.

2. *Elaboration*

pada tahap *Elaboration* ini menentukan rancangan sistem. Yang akan di buat dalam diagram UML yaitu *use case* diagram, *activity* diagram, *sequence* diagram, dan *class* diagram.

a) Perancangan *Use Case* diagram

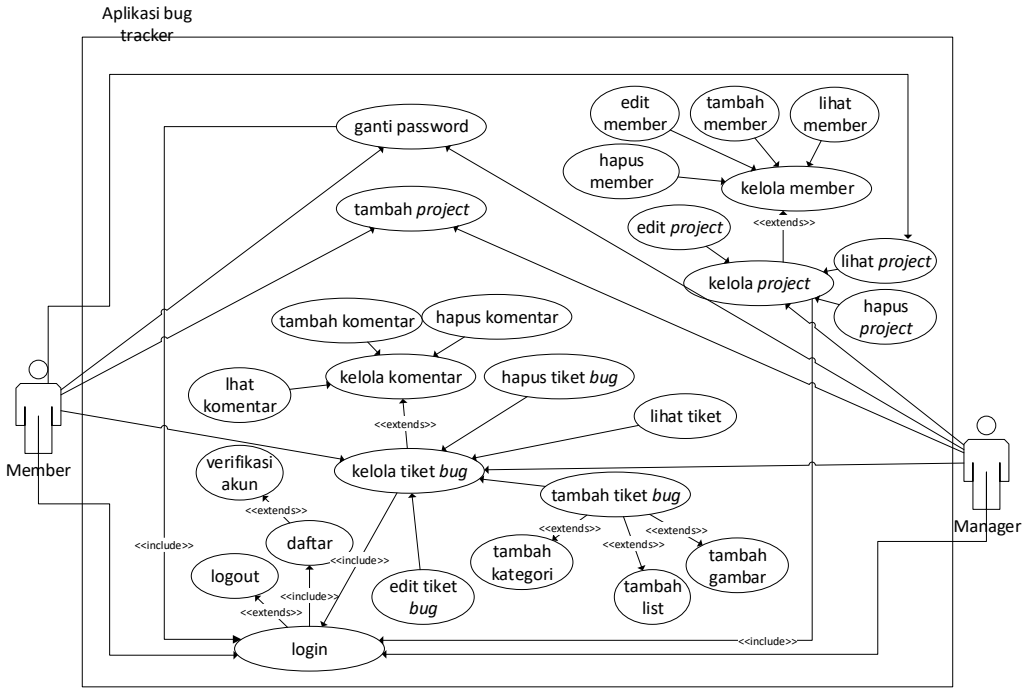
Melakukan identifikasi aktor sebelum membuat *use case* diagram.

1. Identifikasi aktor, seseorang yang melakukan interaksi dengan aplikasi. Dari hasil pengumpulan data terdapat identifikasi aktor yang berperan dalam proses aktivitas penggunaan aplikasi *bug tracker* yaitu *member* dan *manager* yaitu:

Tabel 2: Identifikasi Aktor

No	Aktor	Aktivitas
1	*Manager*	Orang yang dapat mengelola informasi *project, ticket bug*, dan *member*.
2	*Member*	Melihat informasi seputar *project* yang dimasuki sebagai *member*, menambah kan, mengedit, menghapus *ticket bug*.

2. *Use case*, digunakan untuk mengetahui fungsi-fungsi apa saja yang berada di dalam sebuah sistem dan siapa saja yang berhak menggunakan fungsi yang ada.



Gambar 3: *Use Case* diagram Aplikasi *bug tracker*

3. Skenario *use case*, adapun salahsatu skenario dari *use case* diagram akan dijelaskan sebagai berikut :

   a. Skenario *use case* tambah *project*

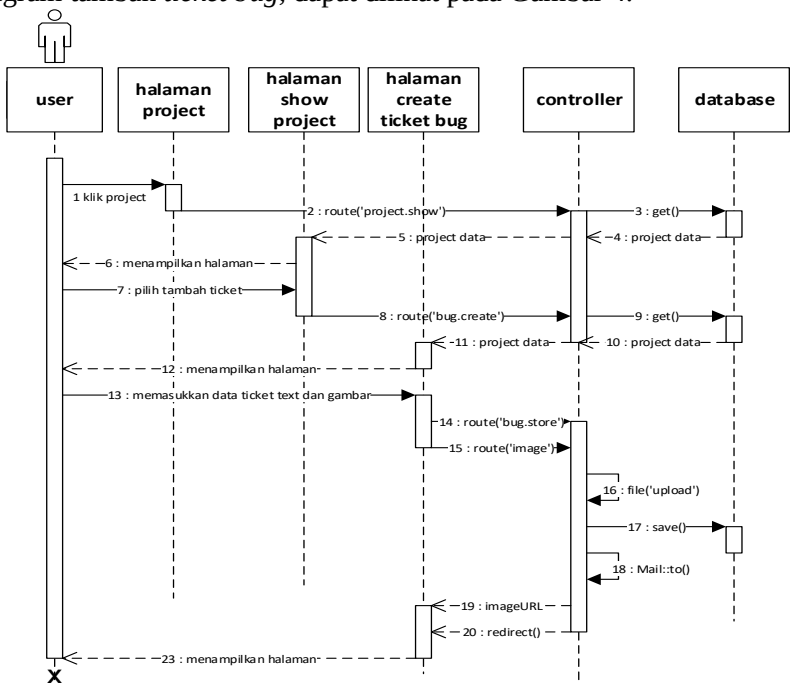
   Tabel 3: Skenario *Use Case* Tambah Tiket *bug*

Aksi aktor	Reaksi sistem
1. Melakukan *login*	2. menampilkan halaman *dashboard*
3. Menekan tombol *project* di bagian atas halaman	4. Menampilkan *project* pribadi dan *project* sebagai *member*
5. Memilih salah satu *project*	6. Menampilkan *ticket bug project*
7. Memilih tombol tambah *ticket*	8. Menampilkan form tambah *ticket bug*
9. Memasukkan *data ticket bug* dan menekan tombol tambah	9. Menyimpan data ke *database*

b) Perancangan *Sequence* diagram

Menggambarkan interaksi antara aktor dengan sistem. Adapun salahsatu *sequence* diagram yang telah dirancang adalah sebagai berikut:

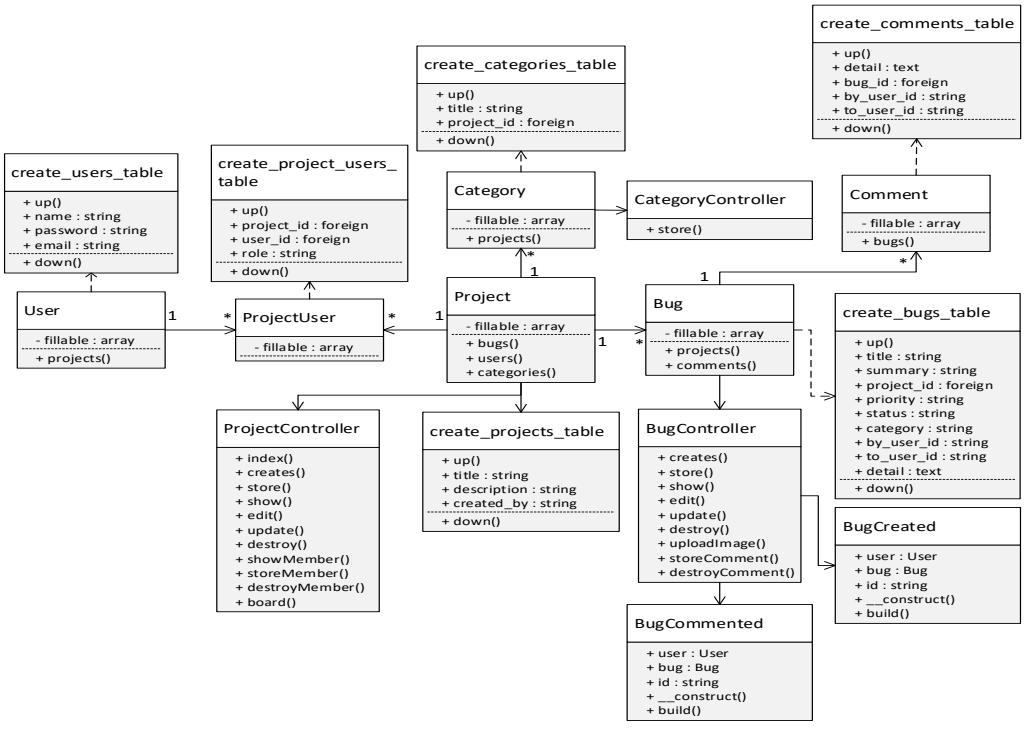
1. *Sequence* diagram tambah *ticket bug*, dapat dilihat pada Gambar 4.



Gambar 4: *Sequence* diagram Tambah *Ticket bug*

c) Perancangan *Class* diagram

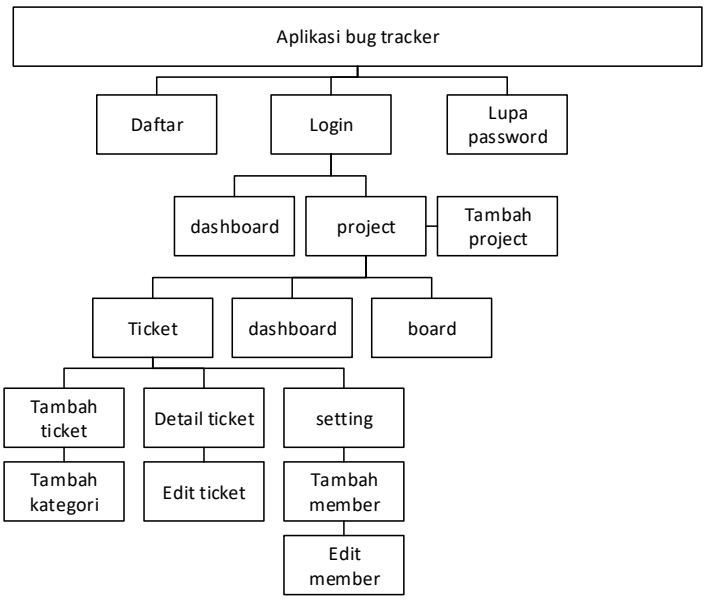
Sering disebut juga desain pemrograman. Di mana *class* diagram ini menggambarkan struktur dan deskripsi serta hubungan antar *class* diagram. Berikut ini merupakan *class* dari aplikasi *bug tracker* berbasis *web* pada Gambar 5.



Gambar 5: *Class* diagram Pada Aplikasi *bug tracker*

d) Perancangan struktur menu

Perancangan struktur menu dibuat untuk mempermudah pembuatan navigasi ketika menjalankan aplikasi. Rancangan struktur menu dapat dilihat pada gambar 6.

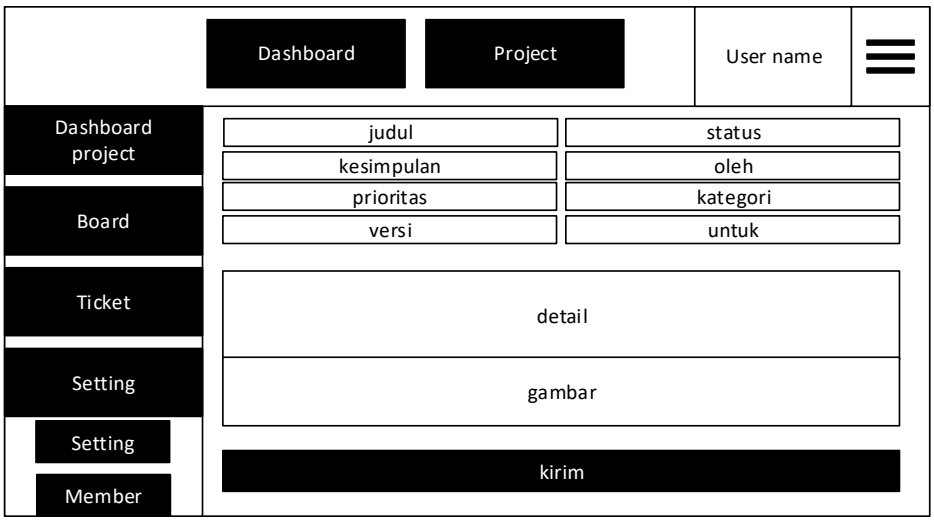


Gambar 6: Struktur Menu

e) Perancangan *interface*

Berikut ini adalah salah satu rancangan *interface* dari aplikasi *bug tracker* berbasis *web*.

1. *Interface* tambah *ticket bug*



Gambar 7: Rancangan *Interface* Tambah *Ticket bug*

3. *Construction*

Pada tahap *construction* ini pengaplikasian rancangan menjadi sebuah aplikasi berbasis *web* menggunakan bahasa pemrograman php dan javascript dengan menggunakan mysql sebagai database. Kode dibuat dengan menggunakan *Visual Studio Code,* xampp [12] dan *web browser opera*. dalam pengembangan sistem ini

menggunakan *framework* php yaitu Laravel. Laravel menggunakan arsitektur pattern MVC (Model *View Controller*) di dalamnya, maka pengembangan dibagi menjadi tiga bagian yaitu:

a) Pembuatan *view*

pembuatan *view* ini berdasarkan rancangan yang struktur menu, pembuatan *view* dibuat sebagai media *interface* dengan *user*. Pembuatan *view* ini menggunakan html dan *templating engine blade* yang tersedia dari laravel. Tampilan *blade* didapat dari rute atau *controller* yang menggunakan bantuan global *view* helper yang juga dapat mengirim data dari *view* dengan menggunakan argument kedua seperti yang dapat dilihat pada gambar 4.12.:

```
return view('projects.index', [
    'own_projects' => $own_projects,
    'other_projects' => $other_projects
]);
```

Gambar 8: *View* Function Dengan Dua *Argument*

Dari gambar diatas *view* menggunakan argument '*project.index*' sebagai *view* yang akan di kembalikan dan *array* sebagai data yang akan di kirim. Dalam *view* penerimaan data dari php menggunakan *syntax* dari *blade* yaitu "{{}}" yang bisa di lihat pada gambar :

```
<td>{{ $ticket->id }}</td>
<td>{{ $ticket->title }}</td>
<td>{{ $ticket->summary }}</td>
<td>{{ $ticket->priority }}</td>
<td>{{ $ticket->status }}</td>
<td>{{ $ticket->category }}</td>
<td>{{ $ticket->version }}</td>
```

Gambar 9: Penerimaan Data Dari Php Pada *View*

Pada pembuatan "*bug*s.*create*.*blade*.php" terdapat fitur menambah data berupa gambar dan *list* data yang dibuat dengan menggunakan *editor text* yaitu CK*Editor* 4. CK*Editor* 4 adalah *editor* WYSIWYG (*What You See Is What You Get*) untuk menulis berbagai konten langsung dalam *web*. Pada CKEditor4 terdapat *plugin* untuk *upload* gambar dan membuat *list* yang dapat digunakan untuk pembuatan fitur *screenshot* dan *step to reproduce*. Pada *view* di buat input berupa *text*area dan kode javascript seperti pada gambar

```
<div class="form-group">
    <label for="">detail</label>
    <textarea name="detail" id="content" ></textarea>
</div>

<script src="{{asset('ckeditor/ckeditor.js')}}"></script>
<script>

    CKEDITOR.replace('content', {
        filebrowserUploadUrl: "{{route('post2.image', ['_token' => csrf_token() ])}}",
        filebrowserUploadMethod: 'form'
    });
```

Gambar 10: Kode Pengimplementasian Fitur Dengan CKEditor4

b) Pembuatan *model*

Pembuatan *model* ini berdasarkan rancangan dari *class* diagram, pembuatan *model* dibuat sebagai data yang berhubungan antara *controller* dengan *database*.

Untuk model yang memiliki *relationship* maka ditambahkan fungsi masing-masing model yang memiliki *relationship* dengan model lainnya. Jika relasi nya *one to many* maka akan mengembalikan fungsi *hasMany (Class),* jika *many to many* akan mengembalikan fungsi *belongsToMany(Class)* yang dapat di lihat seperti pada gambar

```php
public function users()
{
    return $this->belongsToMany(User::class, 'project_users', 'project_id', 'user_id');
}
public function bugs()
{
    return $this->hasMany(Bug::class);
}

public function categories()
{
    return $this->hasMany(Category::class);
```

Gambar 11: Model *Project* Dengan *Relationship*

c) Pembuatan *controller*

Pembuatan *controller* ini berdasarkan dari *activity* diagram dan rancangan yang dibuat , pembuatan *controller* dibuat sebagai alat yang bertugas sebagai pengolah data.

Pembuatan *controller* dalam pengolahan data seperti *create, read, update,* dan *delete* menggunakan laravel eloquent, eloquent adalah suatu *object-relational mapper* yang mempermudah dalam berinteraksi dengan database. Pembuatan pengolahan data dapat dilihat pada daftar berikut:

a. *Read* atau baca dibuat dengan mengambil data dari database dengan memanggil model yang ingin di ambil. Dapat dilihat pada gambar :

```php
public function show($id, $projec_id)
{
    //
    $bug = Bug::where('id', $id)->firstOrFail();
    $comments = $bug->comments()->latest()->get();
```

Gambar 12: Kode Pengambilan Data Dengan Model

b. *Create* atau *update* dibuat dengan memanggil fungsi save() pada model yang ingin disimpan atau diubah. Yang dapat dilihat pada gambar:

```php
$bug = new Bug();
$bug->title = $request->title;
$bug->summary = $request->summary;
$bug->priority = $request->priority;
$bug->status = $request->status;
$bug->category = $request->category;
$bug->version = $request->version;
$bug->by_user_id = $request->by_user_id;
$bug->to_user_id = $request->to_user_id;
$bug->detail = $request->detail;

$bug->project()->associate($projectNew);
$bug->save();
```

Gambar 13: Kode Menyimpan Data Pada *Controller*

c. *Delete* dibuat dengan memanggil fungsi *delete*() pada model yang ingin dihapus. Dapat dilihat pada gambar:

```
$bugTitle = $bug->title;
$bug->delete();
```

Gambar 14: Kode Menghapus Data Pada *Controller*

d. Meng*upload file* gambar dibuat dengan membuat fungsi *upload* gambar yang mengambil *request* gambar dari CK*Editor* lalu di *upload* ke tempat penyimpanan *file* dan mengembalikan *response* kembali ke CK*Editor*, yang dapat dilihat pada gambar:

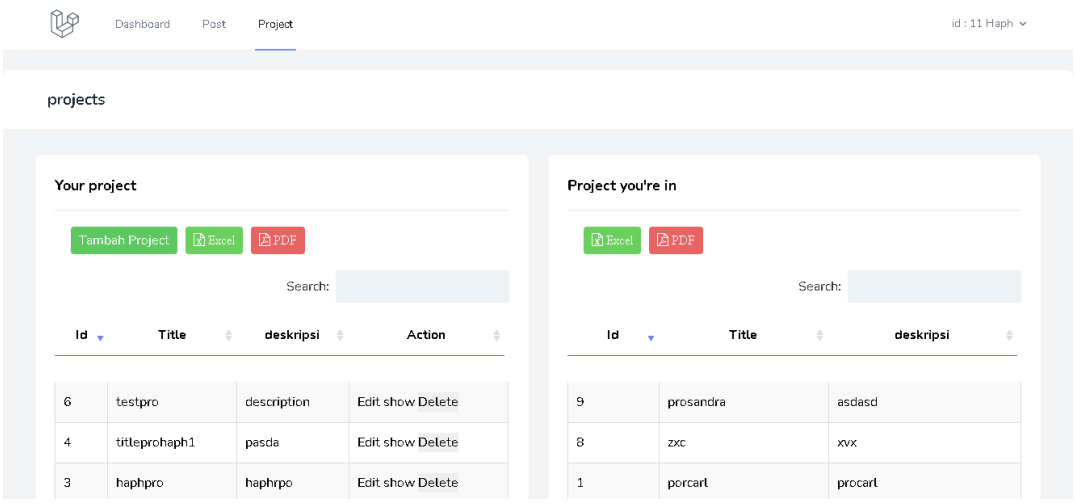
```
if ($request->hasFile('upload')) {
    $file = $request->file('upload'); //SIMPAN SEMENTARA FILENYA KE VARIABLE
    $fileName = pathinfo($file->getClientOriginalName(), PATHINFO_FILENAME); //KITA GET ORIGINAL NAME-NYA
    //KEMUDIAN GENERATE NAMA YANG BARU KOMBINASI NAMA FILE + TIME
    $fileName = $fileName . '_' . time() . '.' . $file->getClientOriginalExtension();
    // $file->move(public_path('uploads'), $fileName); //SIMPAN KE DALAM FOLDER PUBLIC/UPLOADS
    $request->file('upload')->storeAs(
        '',
        $fileName,
        'google'
    );
    //KEMUDIAN KITA BUAT RESPONSE KE CKEDITOR
    $ckeditor = $request->input('CKEditorFuncNum');
    $path = collect(Storage::disk('google')->listContents(''))->where('name', $fileName)->pluck('path')->get(0);
    $urlPath = $request->root() . '/image/' . $path;
    // $url = asset('uploads/' . $fileName);
    $msg = 'Image uploaded successfully';
    //DENGNA MENGIRIMKAN INFORMASI URL FILE DAN MESSAGE
    $response = "<script>window.parent.CKEDITOR.tools.callFunction($ckeditor, '$urlPath', '$msg')</script>";

    //SET HEADERNYA
    @header('Content-type: text/html; charset=utf-8');
    return $response;
```

Gambar 15: Kode *Upload* Gambar Pada *Controller*

d) Tampilan aplikasi

Adapun salah satu tampilan aplikasi yang sudah di buat:



Gambar 16: Halaman *Project*

e) Pengujian

Melakukan pengujian *blackbox*, pada pengujian ini hanya melakukan pengujian fungsional saja dan tidak melakukan pengujian yang lain. pengujian tersaji pada tabel dibawah ini:

Tabel 4: Functional *Testing*

No	Nama Kegiatan	Hasil Yang Diharapkan	Validitas	
			Y	T
1.	Daftar	Data *user* , pengiriman verifikasi email , redirect ke halaman verifikasi	Y	
2.	*Login*	Verifikasi data *login* , redirect ke *dashboard* , redirect ke halaman verifikasi	Y	
3.	Logout	Keluar akun	Y	
4.	Verifikasi akun	Akun terverifikasi	Y	
5.	Ganti *password*	Mengubah data *password*, mengirim email ganti *password*	Y	
6.	Tambah *project*	Data *project*	Y	
7.	Lihat *project*	Data *project*	Y	
8.	Hapus *project*	Data *project* terhapus	Y	
9.	Edit *project*	Data *project* terubah	Y	
10.	Lihat *ticket bug*	Data *ticket bug*	Y	
11.	Tambah *ticket bug*	Data *ticket bug*	Y	
12.	Edit *ticket bug*	Data data *ticket bug* terubah	Y	
13.	Hapus *ticket bug*	Data *ticket bug* terhapus	Y	
14.	Tambah kategori	Data kategori	Y	
15.	Tambah gambar	Data gambar	Y	
16.	Tambah *list*	Data *list*	Y	
17.	Tambah komentar	Data komentar	Y	
18.	Lihat komentar	Data komentar	Y	
19.	Hapus komentar	Data komentar terhapus	Y	
20.	Lihat *member*	Data *member*	Y	
21.	Tambah *member*	Data *member*	Y	
22.	Edit *member*	Data *member* terubah	Y	
23.	Hapus *member*	Data *member* terhapus	Y	

Pengujian dilakukan dengan menguji setiap fungsi yang dibuat secara manual dengan menggunakan desktop *web browser* yaitu Opera , Microsoft edge , Chrome , dan FireFox. dan menggunakan *mobile web browser* yaitu Chrome dan Opera.Pengujian pada 23 kegiatan menghasilkan persentase data yang valid sebesar 100% dengan jumlah data yang valid sebanyak 23.

4. *Transition*

Pada tahap transisi ini dilakukan *deployment* aplikasi ke *internet* dengan menggukanan heroku *free* dengan dyno yang akan *sleep* atau tertidur jika aplikasi tidak di akses selama 30 menit dan menggunakan *add-on database* heroku postgres, aplikasi *bug tracker ini* telah di *deploy* ke *web* dengan domain http://*bugtracker*sendra.herokuapp.com/. Aplikasi ini hanya di deploy ke internet dan tidak disebarluaskan.

5. Keterkaitan dengan penelitian

Telah berhasil membuat *bugtracker* berbasis *web* dengan dua fitur yang di sebutkan pada penelitian "*Analytical Survey on bug Tracking System"*[4]. Dimana beberapa *bugtracker* belum terdapat dua fitur tersebut. Dua fitur tersebut adalah fitur *Screenshot* dan *Step to Produce.*fitur *screenshot* berupa fitur penambahkan informasi berupa media gambar , dan *Step to produce* berupa *list text* .

IV. KESIMPULAN

Berdasarkan hasil penelitian dengan judul Rancang Bangun Aplikasi Pengelolaan *bug tracker* Berbasis *web* Pada Pengembangan Software, maka dapat diperoleh kesimpulan yaitu berhasil merancang dan membuat aplikasi *bug tracker* berbasis *web*. Berhasil membuat fitur yang belum ada pada beberapa *bug tracker*. fitur tersebut diantaranya: fitur *Upload* gambar untuk menambah informasi berupa file gambar, dan fitur *list step to reproduce* untuk menunjukkan bagaimana *bug* tersebut muncul atau dihasilkan. Fitur tersebut berhasil dibuat dengan menggunakan aplikasi CK*Editor* 4 yaitu sebuah *editor* WYSIWYG (*What You See Is What You Get*) dengan menggunakan plugin *Upload* image dan indent *list* untuk mengangani pembuatan fitur *Upload* gambar dan *list* informasi. Adapun saran untuk kedepannya adalah menambahkan fitur *daily reminder,* tenggat waktu, memperbaiki tampilan *user interface,* menambahkan fitur *Stack Trace, Expected behavior, Test Case,* dan *Dependencies.*

DAFTAR PUSTAKAs

[1] K. Pandey, "A Bug Tracking Tool for Efficient Penetration Testing," *Int. J. Educ. Manag. Eng.*, vol. 8, no. 3, pp. 14–20, 2018, doi: 10.5815/ijeme.2018.03.02.

[2] N. Zaware, P. Datir, M. Balwadkar, and V. W. Student, "Online Bug Tracking System," *Int. Res. J. Eng. Technol.*, vol. 9001, pp. 3–5, 2008.

[3] P. Rani, N. Srivastava, J. Soni, and R. Behl, "Web Based Issue Tracker for System Development," *Int. Res. J. Eng. Technol.*, vol. 07, no. 05, pp. 115–118, 2020.

[4] A. Sowjanya and B. P. Reddy, "Analytical Survey on Bug Tracking System," *Int. J. Sci. Eng. Technol. Res.*, vol. 5, no. 49, pp. 10011–10015, 2016.

[5] D. D. S. Fatimah, A. Sutedi, M. S. Hidayat, and L. Fitriani, "Design of employee presence system using Radio Frequency Identification technology," *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 1098, no. 3, p. 032105, 2021, doi: 10.1088/1757-899x/1098/3/032105.

[6] A. Kusumah and R. Cahyana, "Rancang Bangun Sistem Pengelolaan Service Level Agreement Berbasis Online," *J. Algoritm.*, vol. 13, no. 1, pp. 1–11, 2016, doi: 10.33364/algoritma/v.13-1.1.

[7] L. Fitriani and G. S. Sholihat, "Pengembangan Sistem Informasi Geografis Penggalangan Dana dan Donasi Berbasis Web," *J. Algoritm.*, vol. 17, no. 2, pp. 497–507, Feb. 2021, doi: 10.33364/algoritma/v.17-2.497.

[8] R. A S and M. Shalahuddin, *Rekayasa Perangkat Lunak (Terstruktur dan Berorientasi Objek)*. Bandung: Informatika Bandung, 2016.

[9] F. Fatmasari and S. Sauda, "Pemodelan Unified Modeling Language Sistem Informasi Enterprise Resource Planning," *J. Media Inform. Budidarma*, vol. 4, no. 2, p. 429, 2020, doi: 10.30865/mib.v4i2.2022.

[10] Aminudin, "Cara Efektif Belajar Framework Laravel," *Ilmu Teknol. Inf.*, vol. 1, no. 1, pp. 1–28, 2015.

[11] B. A. Priyaungga, D. B. Aji, M. Syahroni, N. T. S. Aji, and A. Saifudin, "Pengujian Black Box pada Aplikasi Perpustakaan Menggunakan Teknik Equivalence Partitions," *J. Teknol. Sist. Inf. dan Apl.*, vol. 3, no. 3, p. 150, 2020, doi: 10.32493/jtsi.v3i3.5343.

[12] Kumari and Nandal, "A Research Paper on Website Development Optimization using Xampp/PHP," *Int. J. Adv. Res. Comput. Sci.*, vol. 8, no. 5, pp. 1231–1235, 2017.