Penerapan Test Driven Development (TDD) pada Laravel Menggunakan PHPUnit

Sigit Wijanarko[1], Subhiyanto[2]

***Abstract*—Test Driven Development (TDD) is a software development approach that prioritizes writing tests before functional code. Laravel, as one of the popular PHP frameworks, provides strong support for TDD through tools like PHPUnit and Laravel Dusk. This journal discusses the concept of TDD, its benefits and challenges, and its application in developing applications with Laravel. A simple case study will also be provided to illustrate the implementation of TDD in Laravel.**

***Intisari*—Test Driven Development (TDD) adalah pendekatan pengembangan perangkat lunak yang mendahulukan penulisan tes sebelum kode fungsional. Laravel, sebagai salah satu framework PHP yang populer, menyediakan dukungan yang kuat untuk TDD melalui alat-alat seperti PHPUnit dan Laravel Dusk. Jurnal ini membahas konsep TDD, keuntungan dan tantangannya, serta penerapannya dalam pengembangan aplikasi dengan Laravel. Studi kasus sederhana juga akan diberikan untuk mengilustrasikan penerapan TDD dalam Laravel.**

***Kata Kunci*— perangkat lunak; pengujian; TDD; aplikasi web; Laravel; PHPUnit.**

I. PENDAHULUAN

Test Driven Development (TDD) adalah metodologi pengembangan perangkat lunak yang berfokus pada penulisan tes otomatis sebelum menulis kode aplikasi. "TDD sebagai salah satu praktik yang umum dalam inti pengembangan metode"[1]. "TDD merupakan sebuah strategi pengembangan perangkat lunak yang Langkah pengerjaannya adalah dengan membuat *test case* terlebih dahulu baru kemudian dilakukan *producing code* dan *refactoring*"[2].

TDD digunakan untuk mengembangkan perangkat lunak gabungan antara desain dan pengujian logika program. Oleh karena itu dianggap sebagai penyatuan dari *Test First Development* dimana unit test dilakukan terlebih dahulu sebelum *writing code*. "Refactoring juga terkait dengan proses TDD dan ini berperan penting dalam menstruktur ulang potongan kode. Selanjutnya, test harus berhasil untuk mengurangi kompleksitas, dan meningkatkan pemahanan, pemeliharaan, dan kejelasan kode"[3]–[6]

"Satu poin penting dalam metode TDD adalah pengembangan perangkat lunak focus pada spesifikasi bukan pada validasinya"[7]. "Dengan demikian perangkat lunak membutuhkan proses pengujian yang mendetail pada komponen-komponen yang ada sebelum komponen tersebut diintegrasikan satu sama lain hingga menjadi suatu system untuk digunakan"[8]. Hal ini membantu mengurangi jumlah bug dan mempercepat proses pengembangan.

[1,2] *STMIK Antar Bangsa, Kawasan Bisnis CBD Ciledug, Jl. HOS Cokroaminoto No.29-35, RT.001/RW.001, Karang Tengah, Kec. Ciledug, Kota Tangerang, Banten 15157 (tlp: 5098-6099;sgtwijanarko23@gmail.com, subhiyanto.bian@gmail.com,)*

"Sistem yang mempunyai bug atau error tentu saja akan mempengaruhi kinerja perangkat lunak seperti kegagalan dalam pemrosesan data, kegagalan dalam mengirim data dan kegagalan pada hasil yang diingingkan"[9], [10]

II. KAJUAN LITERATUR

A. *Test Driven Development*

Test Driven Development (TDD) adalah metodologi pengembangan perangkat lunak yang berfokus pada penulisan tes otomatis sebelum menulis kode aplikasi. Metode ini pertama kali diperkenalkan oleh Kent Beck dalam bukunya "Test Driven Development: By Example". Konsep utama TDD adalah "Red, Green, Refactor" yang menggambarkan tiga fase utama dalam siklus pengembangan TDD.

B. *Konsep dan Prinsip TDD*

Penulisan Tes Terlebih Dahulu: Dalam TDD, pengembang menulis tes sebelum menulis kode fungsional. Tes ini menggambarkan spesifikasi atau persyaratan dari fungsi yang akan dibuat.

Siklus *Red-Green-Refactor*:

1) *Red*: Menulis tes yang akan gagal karena kode fungsional belum ada.
2) *Green*: Menulis kode minimum yang diperlukan agar tes tersebut lulus.
3) *Refactor*: Memperbaiki kode agar lebih bersih dan efisien tanpa mengubah perilaku yang diuji.

Continuous Testing: Tes harus dijalankan secara terus menerus selama pengembangan untuk memastikan bahwa perubahan tidak merusak fungsionalitas yang sudah ada.

C. *Keuntungan dan Tantangan TDD*

Keuntungan:

1) Kode Lebih Andal: TDD membantu memastikan bahwa setiap bagian kode telah diuji dengan baik sebelum digabungkan ke dalam sistem.
2) Dokumentasi Otomatis: Tes yang ditulis dapat berfungsi sebagai dokumentasi hidup dari sistem.
3) Desain yang Lebih Baik: TDD mendorong desain kode yang modular dan dapat diuji.

Tantangan:

1) Kurva Belajar: Memulai dengan TDD bisa menjadi sulit, terutama bagi pengembang yang belum terbiasa dengan pengujian otomatis.
2) Waktu Pengembangan Awal yang Lebih Lama: Menulis tes dan kode fungsional dapat memakan waktu lebih lama di awal, meskipun ini biasanya terbayar dalam jangka panjang dengan kualitas kode yang lebih tinggi.

III. METODE PENELITIAN

Penelitian ini menggunakan pendekatan kualitatif dan kuantitatif untuk menganalisis penerapan TDD dalam pengembangan aplikasi dengan Laravel. Pendekatan kualitatif digunakan untuk memahami persepsi dan pengalaman pengembang yang menggunakan TDD, sedangkan pendekatan kuantitatif digunakan untuk mengukur efektivitas TDD dalam hal kualitas kode dan waktu pengembangan. Laravel adalah framework PHP yang sangat cocok untuk penerapan TDD karena dukungan yang kuat dari komunitasnya dan integrasi alat pengujian seperti PHPUnit dan Laravel Dusk.

A. Pengumpulan Data

Metode pengumpulan data pada penelitian ini dilakukan dengan melakukan studi literatur untuk memahami konsep dasar TDD, manfaat dan tantangannya, serta bagaimana Laravel mendukung TDD. Sumber literatur mencakup buku, artikel jurnal, dokumentasi resmi Laravel, dan blog pengembang. Mengimplementasikan studi kasus penerapan TDD dalam proyek Laravel nyata. Studi kasus ini mencakup seluruh siklus pengembangan dari penulisan tes, pengembangan fitur, hingga refactoring. Data yang dikumpulkan mencakup waktu pengembangan, jumlah bug yang ditemukan, dan perubahan kualitas kode.

B. Implementasi

Implementasi program dilakukan dengan mempersiapkan lingkungan pengujian: Laravel menggunakan PHPUnit sebagai kerangka kerja pengujian default.

Pertama adalah membuat proyek Laravel untuk mengelola sumber daya posting.

```
composer create-project --prefer-dist laravel/laravel laravel-crud
```

Konfigurasi pengaturan database di file .env yang terletak di root proyek Laravel untuk menggunakan SQLite atau MySql, lalu buat *model*, *migration*, dan *controller* untuk mengelola sumber daya posting.

```
php artisan make:model Post -mcr
```

Lakukan modifikasi di file migrasi untuk memasukkan bidang yang diperlukan.

```
public function up()
{
    Schema::create('posts', function (Blueprint $table) {
        $table->id();
        $table->string('title');
        $table->text('content');
        $table->timestamps();
    });
}
```

Jalankan migrasi untuk membuat tabel di database.

```
php artisan migrate
```

Ubah file Post.php di direktori app/models untuk menambahkan properti $fillable, yaitu mengaktifkan mass

```
namespace Tests\Feature;

use App\Models\User;
use Illuminate\Foundation\Testing\RefreshDatabase;
use Tests\TestCase;

class PostTest extends TestCase
{
    use RefreshDatabase;

    /** @test */
    public function a_user_can_create_a_post()
    {
        $user = User::factory()->create();

        $this->actingAs($user);

        $response = $this->post('/posts', [
            'title' => 'Test Post',
            'content' => 'This is a test post content.',
        ]);

        $response->assertStatus(302);
        $this->assertDatabaseHas('posts', [
            'title' => 'Test Post',
            'content' => 'This is a test post content.',
        ]);
    }
}
```

assignment untuk atribut yang aman dengan menandai mereka sebagai "fillable".

```
namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Post extends Model
{
    use HasFactory;

    protected $fillable = ['title', 'content', 'user_id'];

    public function user()
    {
        return $this->belongsTo(User::class);
    }
}
```

Selanjutnya adalah menulis tes atau pengujian, kita akan menambahkan fitur untuk mengelola postingan dalam aplikasi Laravel. Lalu, implementasikan fungsionalitas untuk membuat pengujian berhasil.

C. Perapihan Kode

Dalam pendekatan Test First Development (TFD), kode yang dihasilkan biasanya belum rapi. Oleh karena itu, diperlukan proses perapihan kode untuk meningkatkan kemudahan pemeliharaan.

IV. HASIL DAN PEMBAHASAN

Penelitian ini menghasilkan sebuah aplikasi web untuk mengelola sumber daya posting. Pengembangan aplikasi menggunakan metode *Test Driven Development*, yang dimulai dengan pengujian, dilanjutkan dengan implementasi program, dan diakhiri dengan proses perapihan kode.

A. *Menulis Pengujian*

Proses pengujian unit dibuat dengan bantuan PHPUnit. dengan perintah berikut:

```
composer require --dev phpunit/phpunit
```

Tulis pengujian untuk memverifikasi bahwa pengguna dapat membuat postingan. Buat file pengujian jika belum ada, menggunakan perintah berikut:

```
php artisan make:test PostTest
```

Edit file pengujian untuk memasukkan pengujian pembuatan postingan.

B. *Implementasi Fungsionalitas*

Selanjutnya, implementasikan fungsionalitas untuk membuat pengujian berhasil.

Buka file routes/web.php dan tentukan rute untuk operasi CRUD aplikasi.

```
use App\Http\Controllers\PostController;

Route::resource('posts', PostController::class);
```

Buka app/Http/Controllers/PostController.php dan implementasikan metode controller untuk CRUD pengelolaan posting.

```
namespace App\Http\Controllers;

use App\Models\Post;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;

class PostController extends Controller
{
    public function __construct()
    {
        $this->middleware('auth');
    }

    public function index()
    {
        $posts = Post::where('user_id', Auth::id())->get();
        return view('posts.index', compact('posts'));
    }

    public function create()
    {
        return view('posts.create');
    }

    public function store(Request $request)
    {
        $request->validate([
            'title' => 'required',
            'content' => 'required',
        ]);

        Auth::user()->posts()->create($request->all());

        return redirect()->route('posts.index')
                         ->with('success', 'Post created successfully.');
    }

    // Metode CRUD lainnya...
}
```

Jalankan pengujian dengan perintah berikut:

```
php artisan test

C:\Users\ASUS A407M\WPII\laravel-crud
λ php artisan test --filter=PostTest::a_user_can_create_a_post

   PASS  Tests\Feature\PostTest
✓ a user can create a post

  Tests:    1 passed (2 assertions)
  Duration: 1.26s
```

Posts

Create New Post

Title	Content	Actions
Lorem	Lorem ipsum	Show Edit Delete
dolor	dolor sit amet	Show Edit Delete

Gambar 1. Hasil Pengujian Unit

Tabel 1. Hasil Pengujian Unit

Unit	Hasil Pengujian
Tambah Posting	Berhasil

C. *Perapihan Kode*

Dengan berhasilnya pengujian, Kode dapat dirapihkan untuk meningkatkan strukturnya dan keterbacaan sambil memastikan pengujian tetap berhasil.

```
use App\Models\Post;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;

class PostController extends Controller
{
    public function __construct()
    {
        $this->middleware('auth');
    }

    public function index()
    {
        $posts = Auth::user()->posts;
        return view('posts.index', compact('posts'));
    }

    public function create()
    {
        return view('posts.create');
    }

    public function store(Request $request)
    {
        $validatedData = $request->validate([
            'title' => 'required',
            'content' => 'required',
        ]);

        $post = Auth::user()->posts()->create($validatedData);

        return redirect()->route('posts.index')
                         ->with('success', 'Post created successfully.');
    }

    // Metode CRUD lainnya...
}
```

V. KESIMPULAN

Penerapan TDD dalam pengembangan aplikasi menggunakan Laravel menawarkan banyak keuntungan, termasuk kode yang lebih andal, desain yang lebih baik, dan

dokumentasi otomatis. Meskipun ada tantangan dalam mengadopsi TDD, manfaat jangka panjangnya sering kali sepadan. Dengan alat pengujian yang kuat seperti PHPUnit, Laravel menyediakan lingkungan yang sangat baik untuk penerapan TDD. Implementasi studi kasus sederhana menunjukkan bagaimana TDD dapat digunakan untuk mengembangkan fitur-fitur baru secara iteratif dan teruji dengan baik.

UCAPAN TERIMA KASIH

Penulis bersyukur kepada Tuhan Yang Maha Esa, Pencipta alam semesta, dan ingin mengungkapkan rasa terima kasih kepada semua individu dan kelompok yang telah memberikan bantuan, dukungan, dan arahan berharga selama pelaksanaan penelitian ini.

REFERENSI

[1] Z. Khanam dan M. N. Ahsan, "Evaluating the Effectiveness of Test Driven Development: Advantages and pitfalls," *International Journal of Applied Engineering Research*, vol. 12, no. 18, hlm. 7705–7716, 2017.

[2] W. Bissi, A. G. Serra Seca Neto, dan M. C. F. P. Emer, "The effects of test driven development on internal quality, external quality and productivity: A systematic review," *Inf Softw Technol*, vol. 74, hlm. 45–54, Jun 2016, doi: 10.1016/j.infsof.2016.02.004.

[3] K. Buffardi dan S. H. Edwards, "Impact of Teaching Test-Driven Development to Novice Programmers," *International Journal of Information and Computer Science IJICS*, vol. 1, no. 6, hlm. 135–143, 2012.

[4] A. A. S. Ivo, E. M. Guerra, S. M. Porto, J. Choma, dan M. G. Quiles, "An approach for applying Test-Driven Development (TDD) in the development of randomized algorithms," *Journal of Software Engineering Research and Development*, vol. 6, no. 1, hlm. 9, Des 2018, doi: 10.1186/s40411-018-0053-5.

[5] M. M. Moe, "Comparative Study of Test-Driven Development TDD, Behavior-Driven Development BDD and Acceptance Test–Driven Development ATDD," *International Journal of Trend in Scientific Research and Development*, vol. 3, no. 4, hlm. 231–234, Jun 2019, doi: 10.31142/ijtsrd23698.

[6] K. Naveed, S. K. Muhammad, A. J. Muhammad, dan A. K. Muhammad, "Reducing Testing Effort in the Test Driven Development," *Global Journal of Computer Science and Technology Software & Data Engineering*, vol. 13, no. 7, hlm. 0–4, 2013.

[7] R. C. Martin, *Clean Code: A Handbook of Agile Software Craftsmanship*, 1 ed. Boston: Pearson, 2008.

[8] Z. Khanam, "Barriers to Refactoring: Issues and Solutions," *International Journal on Future Revolution in Computer Science & Communication Engineering*, vol. 4, no. 2, hlm. 232–235, 2018.

[9] A. Bulajic, S. Sambasivam, dan R. Stojic, "Overview of the Test Driven Development Research Projects and Experiments," 2012, hlm. 165–187. doi: 10.28945/1647.

[10] Raquelita Ros Aguilar, "Using Test-Driven Development to Improve Software Development Practices," *School of Computer Science*. Reykjavik University, Reykjavík, 2016.

Sigit Wijanarko lahir di Sintang pada tanggal 23 Juni 1980. Lulus S1 Program Studi Teknik Informatika Fakultas Ilmu Komputer Universitas Pembangunan Nasional – Jakarta pada tahun 2004. Lulus Magister Ilmu Komputer Program Pascasarjana Universitas Budi Luhur dengan konsentrasi Rekayasa Komputasi Terapan pada tahun 2019. Saat ini aktif sebagai Dosen tetap di STMIK Antar Bangsa dan praktisi IT di perusahaan swasta.

Subhiyanto, lahir di Brebes pada tanggal 10 Maret 1984.Tahun 2012 Lulus Sarjana Komputer Jurusan Teknik Informatika di STMIK Nusa Mandiri. Tahun 2020 lulus program Pasca Sarjana Ilmu Komputer dengan konsentrasi Rekayasa Komputasi Terapan di Universitas Budi Luhur. Saat ini aktif mengajar sebagai dosen tetap dan sebagai Kepala Biro Kemahasiswaan di STMIK Antar Bangsa