BAB I
PENDAHULUAN

1.1. Latar Belakang

Perkembangan teknologi informasi dalam dua dekade terakhir mendorong kebutuhan akan proses pengembangan perangkat lunak yang semakin cepat, efisien, dan terstandar. Industri teknologi digital, khususnya bidang pengembangan perangkat lunak, menuntut proses rekayasa sistem yang mampu menghasilkan aplikasi dengan kualitas tinggi namun tetap hemat waktu pengerjaan. Menurut *McKinsey Global Institute* (2023), kecepatan pengembangan aplikasi menjadi salah satu faktor utama yang berkontribusi pada daya saing organisasi digital modern. Kebutuhan pasar tersebut menyebabkan berbagai framework modern bermunculan, salah satunya adalah Laravel, yang saat ini menjadi framework PHP paling populer berdasarkan survei JetBrains (2022) dan laporan BuiltWith (2023).

Laravel menawarkan arsitektur *Model–View–Controller* (MVC), sintaks yang elegan, serta ekosistem yang sangat kaya sehingga banyak digunakan baik oleh pengembang individu, startup, maupun perusahaan berskala besar. Namun demikian, meskipun menyediakan berbagai fitur otomatisasi, proses awal pengembangan aplikasi Laravel masih memerlukan tahapan manual yang cukup panjang, seperti pembuatan struktur projek, konfigurasi basis data, pembuatan berkas migration, model, controller, *routing*, hingga penyusunan modul CRUD secara repetitif. Studi oleh Suryanto dkk. (2022) menjelaskan bahwa proses repetitif adalah faktor penyebab terbesar berkurangnya produktivitas developer pada fase awal projek perangkat lunak.

Dalam pengembangan proyek perangkat lunak yang berskala menengah hingga besar, penulisan *boilerplate code* sering kali menjadi aktivitas yang memakan waktu serta berpotensi menimbulkan kesalahan apabila dilakukan secara manual. Penelitian terbaru menunjukkan bahwa pengulangan penulisan kode tanpa dukungan otomatisasi meningkatkan risiko inkonsistensi struktur dan kesalahan

logika, terutama pada proyek kolaboratif maupun pada pengembang dengan tingkat pengalaman yang beragam (Ampatzoglou et al., 2020; Lenarduzzi et al., 2021).

Berbagai studi di bidang *code generation* dan *software automation* mengungkapkan bahwa penggunaan generator aplikasi mampu meningkatkan produktivitas pengembangan perangkat lunak secara signifikan, dengan rentang peningkatan antara 30% hingga 70%, tergantung pada kompleksitas sistem dan tingkat standardisasi yang diterapkan (Kumar et al., 2021; Rahman & Roy, 2022). Otomatisasi dalam pembuatan struktur sistem dan kode dasar terbukti dapat mengurangi beban kognitif pengembang, meminimalkan kesalahan berulang, serta mempercepat proses pengembangan tanpa mengorbankan kualitas kode yang dihasilkan.

Berdasarkan kebutuhan tersebut, penelitian ini mengusulkan perancangan dan implementasi Larator, yaitu sebuah *web-based application generator* yang mampu mengonversi rancangan basis data menjadi kerangka aplikasi Laravel secara otomatis. Larator menyediakan *Graphical User Interface* (GUI) untuk merancang tabel dan relasi, serta menghasilkan berbagai komponen inti Laravel seperti migration, model, controller, routing, view dan modul autentikasi, hingga fitur ekspor data secara otomatis. Temuan awal menunjukkan bahwa Larator mampu meningkatkan produktivitas developer hingga 70,7% dibandingkan metode manual.

Larator tidak hanya meningkatkan kecepatan pengembangan, tetapi juga menjaga kualitas kode yang dihasilkan. Pengujian menggunakan PHPInsights memperlihatkan skor kualitas kode rata-rata 89,8%**,** termasuk metrik *complexity*, *architecture*, dan *code quality* yang berada pada kategori sangat baik . Hal ini sejalan dengan prinsip *clean code* dan *software engineering* modern yang menekankan keterbacaan, modularitas, dan standar penulisan kode (Pressman, 2020).

Dengan demikian, penelitian mengenai pengembangan Larator memiliki urgensi akademik dan praktis yang tinggi: secara akademik berkontribusi dalam

literatur otomatisasi pengembangan perangkat lunak, dan secara praktis membantu pengembang, lembaga pendidikan, serta industri kecil-menengah (UMKM) untuk mempercepat proses pembangunan aplikasi digital. Oleh karena itu, penelitian ini penting untuk dilakukan sebagai bagian dari pengembangan teknologi generator aplikasi yang mendukung peningkatan produktivitas dan kualitas perangkat lunak.

1.2. Perumusan Masalah

Berdasarkan latar belakang yang telah diuraikan, maka rumusan masalah dalam penelitian ini dapat dirumuskan sebagai berikut:

1. Bagaimana menganalisis kebutuhan sistem generator aplikasi Laravel yang mampu meningkatkan produktivitas pengembang?
2. Bagaimana merancang arsitektur aplikasi generator berbasis web yang mampu mengotomatisasi pembuatan kerangka Laravel?
3. Bagaimana mengimplementasikan aplikasi Larator agar dapat menghasilkan migration, model, controller, routing, dan view dalam penyusunan modul CRUD secara otomatis?
4. Bagaimana mengukur efektivitas Larator dalam meningkatkan produktivitas dan kualitas kode hasil generate?

1.3. Batasan Masalah

Agar penelitian terarah dan tidak melebar, penelitian ini dibatasi pada ruang lingkup berikut:

1. Sistem yang dikembangkan hanya berfokus pada pembuatan projek Laravel berbasis web, tidak mencakup mobile framework seperti Flutter atau React Native.
2. Fitur yang dihasilkan mencakup autentikasi dasar, CRUD, migration, dan ekspor dan import data, belum mencakup API kompleks.
3. Evaluasi kualitas kode hanya menggunakan alat PHPInsights sebagaimana digunakan dalam penelitian awal.

4. Pengujian produktivitas dibatasi pada beberapa modul standar berbasis CRUD pada aplikasi Laravel.

1.4. Tujuan Dan Manfaat

Tujuan penelitian ini adalah sebagai berikut:

1. Menghasilkan rancangan sistem generator Laravel yang dapat digunakan melalui antarmuka GUI builder.
2. Membangun aplikasi Larator yang mampu menghasilkan kerangka aplikasi Laravel secara otomatis.
3. Menguji efektivitas Larator dalam meningkatkan kecepatan dan kualitas pengembangan aplikasi Laravel.
4. Memberikan kontribusi ilmiah pada domain otomatisasi pengembangan perangkat lunak (*software development automation*).

Manfaat penelitian ini adalah sebagai berikut:

1. Manfaat Akademik

Penelitian ini memberikan kontribusi berupa perluasan literatur mengenai otomatisasi pengkodean, *model-driven engineering*, dan penelitian rekayasa perangkat lunak berbasis generator. Temuan ini dapat menjadi referensi bagi penelitian selanjutnya di bidang generator aplikasi berbasis web, GUI builder, dan otomatisasi framework Laravel.

2. Manfaat Praktis

Larator memberikan solusi otomatisasi yang membantu developer mempercepat pembuatan projek Laravel dengan struktur konsisten dan mengurangi potensi kesalahan manusia (human error). Hal ini mendukung peningkatan produktivitas hingga 70,7% sebagaimana diperoleh dari pengujian awal .

3. Manfaat Institusional

Larator dapat diterapkan pada UMKM, lembaga pendidikan, dan tim pengembang yang membutuhkan pembangunan sistem secara cepat, terstandar, dan efisien. Di

sektor industri, generator seperti ini juga dapat menekan biaya produksi perangkat lunak.

1.5. Metodologi Penelitian

Penelitian ini menggunakan metode rekayasa perangkat lunak berbasis model iteratif yang memungkinkan proses pengembangan dilakukan secara berulang dengan umpan balik berkelanjutan. Tahapan penelitian terdiri atas:

1. Analisis kebutuhan sistem
2. Perancangan arsitektur Larator
3. Implementasi sistem generator
4. Pengujian fungsionalitas (black-box testing)
5. Pengujian kualitas kode menggunakan PHPInsights
6. Pengukuran produktivitas developer

Metodologi ini konsisten dengan model Pressman (2020) mengenai pengembangan perangkat lunak modern dan sesuai dengan karakter penelitian berbasis alat pengembangan (*tool building*).

1.6. Sistematika Penulisan

Sistematika penulisan proposal ini mengacu pada struktur berikut:

- **BAB I. PENDAHULUAN**: berisi latar belakang, rumusan masalah, batasan masalah, tujuan dan manfaat, metodologi, serta sistematika penulisan.
- **BAB II. LANDASAN TEORI**: menjelaskan teori-teori pendukung.
- **BAB III. ANALISIS DAN PERANCANGAN SISTEM**: memuat analisis sistem dan perancangan.
- **BAB IV. HASIL PENELITIAN**: berisi uraian hasil penelitian berdasarkan data.
- **BAB V. SIMPULAN DAN SARAN**: berisi kesimpulan penelitian serta saran untuk pengembangan selanjutnya.

BAB II
LANDASAN TEORI

2.1. Clean Code

Konsep clean code merupakan prinsip fundamental dalam rekayasa perangkat lunak modern yang menekankan keterbacaan, keterpahaman, serta kemudahan pemeliharaan kode sumber. Penelitian terkini menunjukkan bahwa kualitas kode yang baik berkontribusi secara signifikan terhadap peningkatan maintainability dan penurunan risiko kesalahan selama proses pengembangan maupun pemeliharaan sistem (Yli-Huumo et al., 2020; Spinellis & Gousios, 2021). Kode yang mudah dipahami juga mempercepat proses kolaborasi antar pengembang dan mengurangi technical debt dalam jangka panjang.

Elemen penting dalam clean code meliputi konsistensi penamaan variabel dan fungsi, pemecahan fungsi menjadi unit yang kecil dan spesifik, penghindaran duplikasi kode (code duplication), serta penerapan struktur logika yang sederhana dan terkontrol. Studi empiris oleh Amanatidis et al. (2021) menunjukkan bahwa penerapan prinsip-prinsip tersebut berkorelasi positif terhadap peningkatan kualitas internal perangkat lunak, khususnya pada aspek kompleksitas dan keterbacaan kode.

Dalam konteks penggunaan framework Laravel, prinsip clean code menjadi semakin relevan karena Laravel menerapkan pola arsitektur Model–View–Controller (MVC) yang mendorong pemisahan tanggung jawab (separation of concerns). Normalisasi pola penulisan pada komponen seperti controller, model, dan migration memungkinkan penerapan standar kode yang konsisten. Penggunaan code generator seperti Larator berpotensi membantu pengembang menghasilkan struktur kode yang seragam dan sesuai dengan prinsip clean code. Hal ini sejalan dengan temuan studi oleh Bavota et al. (2020) yang menyatakan bahwa otomatisasi dalam pembuatan kode dapat meningkatkan konsistensi dan kualitas kode apabila dirancang dengan standar yang tepat.

Hasil pengujian awal menggunakan alat analisis kualitas kode PHPInsights yang menunjukkan nilai kualitas sebesar 89,8% mengindikasikan bahwa kode yang dihasilkan secara otomatis oleh Larator telah memenuhi karakteristik utama clean code, khususnya pada aspek kompleksitas, konsistensi, dan keterbacaan.

2.2. Rekayasa Perangkat Lunak (Software Engineering)

Rekayasa perangkat lunak (*software engineering*) adalah disiplin yang mencakup proses sistematis dalam merancang, membangun, dan memelihara perangkat lunak. Pressman (2020) menyatakan bahwa rekayasa perangkat lunak tidak hanya fokus pada pengembangan kode, tetapi juga mencakup metodologi, pendekatan analitis, verifikasi, dan pemeliharaan secara berkelanjutan.

Model proses yang digunakan dalam penelitian ini adalah **model iteratif**, di mana pengembangan aplikasi dilakukan dalam siklus berulang. Setiap iterasi melibatkan tahapan analisis, perancangan, implementasi, dan evaluasi. Model iteratif cocok digunakan untuk pengembangan alat seperti generator kode Laravel karena prosesnya membutuhkan umpan balik berkelanjutan, baik dari sisi fungsionalitas maupun kualitas kode.

Kaitan penelitian ini dengan software engineering terlihat pada fokusnya terhadap:

1. Standarisasi proses pembuatan aplikasi
2. Aplikasi GUI builder untuk memodelkan sistem
3. Otomatisasi penulisan kode boilerplate
4. Pengujian kualitas kode menggunakan alat objektif

Larator berkontribusi dalam aspek *software engineering automation* yang saat ini menjadi tren penting dalam penelitian dan industri perangkat lunak.

2.3. Arsitektur MVC (Model–View–Controller)

Model–View–Controller (MVC) merupakan pola arsitektur yang memisahkan komponen aplikasi menjadi tiga bagian utama:

1. Model – menangani logika data
2. View – menampilkan data
3. Controller – mengendalikan alur aplikasi

Menurut Singh & Kaur (2021), pemisahan tanggung jawab dalam MVC bertujuan untuk meningkatkan modularitas dan mempermudah pengembangan berkelanjutan. Laravel sebagai framework PHP modern secara default menerapkan pola arsitektur MVC karena terbukti meningkatkan tingkat keterbacaan kode dan mempercepat pemeliharaan sistem.

Dalam konteks Larator, pola MVC menjadi landasan utama dalam setiap modul yang dihasilkan. Generator harus mampu menyusun migration, model, dan controller sesuai standar Laravel. Hal ini penting agar struktur aplikasi tetap konsisten, modular, dan memudahkan pengembang untuk melakukan *customization* setelah projek dihasilkan. Struktur MVC yang otomatis dihasilkan oleh Larator juga berkontribusi dalam meningkatkan kualitas kode yang terstandar dan mudah dikelola.

2.4. *Design Patterns* (Repository dan Controller Pattern)

Design pattern adalah pola solusi yang telah diuji dan digunakan secara luas dalam pengembangan perangkat lunak. Tujuannya adalah memberikan struktur desain yang dapat memecahkan masalah umum secara efektif. Beberapa *design pattern* penting yang relevan dengan pengembangan generator Laravel adalah:

1. **Repository Pattern**

   Repository Pattern bertujuan memisahkan logika akses data dari logika bisnis. Dengan pola ini, perubahan pada basis data tidak mempengaruhi komponen lain. Menurut Bavota et al. (2020), repository pattern membantu meningkatkan keterujian (*testability*) dan fleksibilitas kode.
2. **Controller Pattern**

   Controller Pattern mengatur alur permintaan (request) dan respons pada aplikasi web. Dalam Laravel, controller menjadi komponen inti yang menghubungkan

view dan model. Dengan mengotomatisasi pembuatan controller melalui Larator, kode menjadi konsisten dan mengikuti pola arsitektur yang terstruktur.

Penerapan *design pattern* secara otomatis melalui generator membantu developer menghindari kesalahan desain, meningkatkan *maintainability*, dan memastikan struktur aplikasi mengikuti standar industri.

2.5. Prinsip Kualitas Kode (complexity, readability)

Kualitas kode dalam pengembangan perangkat lunak umumnya dievaluasi berdasarkan sejumlah metrik yang merepresentasikan kualitas internal sistem. Metrik-metrik tersebut digunakan untuk menilai sejauh mana kode mudah dipahami, dipelihara, dan dikembangkan kembali. Penelitian terkini menyebutkan bahwa metrik kompleksitas, keterbacaan, arsitektur, dan konsistensi penulisan merupakan indikator utama dalam pengukuran kualitas kode perangkat lunak modern (Amanatidis et al., 2021; Lenarduzzi et al., 2022).

1. **Complexity**

   Kompleksitas kode menggambarkan tingkat kerumitan alur logika dalam suatu fungsi atau modul. Kompleksitas yang tinggi cenderung menyulitkan pengembang dalam memahami perilaku sistem serta meningkatkan risiko terjadinya kesalahan dan cacat perangkat lunak. Studi empiris menunjukkan bahwa pengendalian kompleksitas berpengaruh langsung terhadap peningkatan *maintainability* dan penurunan *fault-proneness* pada perangkat lunak (Amanatidis et al., 2021). Oleh karena itu, metrik kompleksitas digunakan untuk memastikan bahwa struktur logika kode tetap sederhana dan terkontrol.

2. **Readability**

   Readability atau keterbacaan kode merujuk pada tingkat kemudahan bagi pengembang lain dalam membaca, memahami, dan memodifikasi kode sumber. Kode dengan tingkat keterbacaan yang baik umumnya memiliki penamaan variabel dan fungsi yang jelas, struktur yang konsisten, serta dokumentasi yang memadai. Penelitian oleh Yli-Huumo et al. (2020) menegaskan bahwa keterbacaan merupakan faktor krusial dalam kolaborasi tim dan pemeliharaan

perangkat lunak jangka panjang, terutama pada proyek berskala menengah hingga besar.

3. **Architecture**

   Arsitektur perangkat lunak berkaitan dengan bagaimana komponen sistem disusun dan berinteraksi satu sama lain. Arsitektur yang baik bersifat modular, memiliki pemisahan tanggung jawab yang jelas (*separation of concerns*), serta mendukung perubahan tanpa berdampak signifikan terhadap komponen lain. Dalam konteks *framework* Laravel, penerapan arsitektur *Model–View–Controller (MVC)* menjadi acuan utama dalam menilai kualitas struktur kode. Arsitektur yang konsisten terbukti dapat meningkatkan fleksibilitas dan kemudahan pengembangan lanjutan (Spinellis & Gousios, 2021).

4. **Standardization**

   Standarisasi penulisan kode mencakup konsistensi gaya penulisan, konvensi penamaan, serta kepatuhan terhadap pedoman pengembangan yang berlaku. Konsistensi ini berperan penting dalam menjaga kualitas kode, khususnya pada proyek yang dikembangkan secara tim atau menggunakan otomatisasi. Static code analysis tools seperti PHPInsights mampu mengukur aspek standarisasi melalui metrik gaya penulisan (code style) dan kepatuhan terhadap praktik terbaik (best practices) (Lenarduzzi et al., 2022).

Hasil pengujian pada Larator menghasilkan nilai rata-rata kualitas kode sebesar 89,8%, yang menunjukkan bahwa generator ini mampu menghasilkan kode yang terstruktur, mudah dibaca, dan sesuai standar industri.

- Complexity: kemudahan alur logika dipahami.
- Readability: keterbacaan kode bagi pengembang lain.
- Architecture: kesesuaian dengan struktur MVC.

PHPInsights digunakan untuk mengukur kualitas kode pada penelitian ini.

2.6. Aplikasi Generator dan Otomatisasi Kode

Aplikasi generator merupakan perangkat yang mampu menghasilkan kode sumber secara otomatis berdasarkan konfigurasi atau model tertentu. Menurut

Kumar & Sehgal (2021), otomatisasi pengembangan perangkat lunak dapat menghemat 40–80% waktu pengembangan, terutama pada pembuatan modul berulang seperti CRUD.

Dalam penelitian *model-driven engineering*, generator kode menjadi teknologi utama yang membantu mempercepat pembuatan sistem berbasis database. Generator seperti Railgun, Yeoman Generator, Laravel Blueprint, dan CodeIgniter Wizard menjadi contoh populer dalam pengembangan web.

Larator berada dalam kategori generator aplikasi GUI berbasis web. Dengan kemampuan merancang tabel, relasi, dan atribut melalui GUI builder, aplikasi ini berada pada level otomatisasi yang tinggi. Dibandingkan generator berbasis command line (seperti Artisan Make), Larator memberikan pengalaman visual yang lebih intuitif dan memudahkan pemula.

2.7. Framework Laravel

Laravel merupakan framework PHP modern yang dirancang untuk mempermudah pengembangan aplikasi web dengan sintaks yang elegan dan arsitektur robust. Laravel menyediakan berbagai fitur seperti:

- Migration database
- Eloquent ORM
- Routing
- Middleware
- Authentication scaffolding
- Artisan CLI tools

Menurut White (2021), Laravel menjadi framework PHP paling populer dengan lebih dari 1 juta projek yang dibangun menggunakan framework ini.

Kaitan Laravel dengan penelitian ini sangat kuat karena Larator dirancang khusus untuk menghasilkan kerangka dasar aplikasi berbasis Laravel. Dengan mengotomatisasi komponen inti Laravel, generator ini mendukung efisiensi dan konsistensi pengembangan aplikasi berbasis PHP modern.

2.8. Penelitian Terdahulu

Beberapa penelitian relevan yang mendukung penelitian ini antara lain:

1. Rahman (2020) meneliti generator CRUD berbasis CodeIgniter yang meningkatkan produktivitas 63%.
2. Kumar & Sehgal (2021) mengembangkan generator otomatis berbasis *model-driven engineering* yang mempercepat proses pengembangan 70%.
3. Suryanto dkk. (2022) mengidentifikasi bahwa penulisan kode manual menjadi penyebab utama keterlambatan pengembangan aplikasi web.
4. Putra & Santoso (2021) meneliti penggunaan PHPInsights untuk mengukur kualitas kode pada projek Laravel, menunjukkan bahwa alat ini valid untuk mengevaluasi kualitas kode.
5. Meutia (2023) menyoroti pentingnya otomatisasi dalam pembuatan struktur aplikasi untuk mengurangi human error pada pengembang.

Penelitian mengenai Larator memiliki kontribusi yang lebih luas karena menggabungkan GUI builder, generator migration, generator controller, generator model, serta modul autentikasi dalam satu aplikasi yang terintegrasi. Hal ini menjadikan Larator lebih komprehensif dibandingkan generator lain yang hanya fokus pada CRUD atau migration saja.