TUGAS AKHIR

PENGEMBANGAN MULTI ARCHITECTURE PATTERN CODE GENERATOR UNTUK APLIKASI TYPESCRIPT DENGAN FRAMEWORK EXPRESS.JS

DISUSUN OLEH:

FARID ANWAR WAHDIE

21106050049

PROGRAM STUDI INFORMATIKA

FAKULTAS SAINS DAN TEKNOLOGI

UNIVERSITAS ISLAM NEGERI SUNAN KALIJAGA

YOGYAKARTA

2025

LEMBAR PENGESAHAN

KEMENTERIAN AGAMA
UNIVERSITAS ISLAM NEGERI SUNAN KALIJAGA
FAKULTAS SAINS DAN TEKNOLOGI
Jl. Marsda Adisucipto Telp. (0274) 540971 Fax. (0274) 519739 Yogyakarta 55281

PENGESAHAN TUGAS AKHIR

Nomor : B-1869/Un.02/DST/PP.00.9/08/2025

Tugas Akhir dengan judul : PENGEMBANGAN MULTI ARCHITECTURE PATTERN CODE GENERATOR UNTUK APLIKASI TYPESCRIPT DENGAN FRAMEWORK EXPRESS.JS

yang dipersiapkan dan disusun oleh:

Nama : FARID ANWAR WAHDIE
Nomor Induk Mahasiswa : 21106050049
Telah diujikan pada : Kamis, 14 Agustus 2025
Nilai ujian Tugas Akhir : A

dinyatakan telah diterima oleh Fakultas Sains dan Teknologi UIN Sunan Kalijaga Yogyakarta

TIM UJIAN TUGAS AKHIR

Ketua Sidang

Dr. Ir. Sumarsono, S.T., M.Kom.
SIGNED

Valid ID: 68a5555a04cc7

Penguji I

Dr. Agung Fatwanto, S.Si., M.Kom.
SIGNED

Valid ID: 68a46eff127fb

Penguji II

Ir. Maria Ulfah Siregar, S.Kom., MIT., Ph.D.
SIGNED

Valid ID: 68a6375205207

Yogyakarta, 14 Agustus 2025
UIN Sunan Kalijaga
Dekan Fakultas Sains dan Teknologi

Prof. Dr. Dra. Hj. Khurul Wardati, M.Si.
SIGNED

Valid ID: 68a765b9576d2

LEMBAR PERNYATAAN

SURAT PERNYATAAN KEASLIAN SKRIPSI

Yang bertanda tangan di bawah ini:

Nama	: Farid Anwar Wahdie
NIM	: 21106050049
Program Studi	: Informatika
Fakultas	: Sains dan Teknologi

Menyatakan dengan sesungguhnya, bahwa skripsi saya yang berjudul "*PENGEMBANGAN MULTI ARCHITECTURE PATTERN CODE GENERATOR UNTUK APLIKASI TYPESCRIPT DENGAN FRAMEWORK EXPRESS.JS*" merupakan penelitian saya sendiri, tidak terdapat karya yang pernah diajukan untuk memperoleh gelar kesarjanaan di suatu perguruan tinggi dan bukan plagiasi karya orang lain kecuali yang secara tertulis diacu dalam naskah ini dan disebutkan dalam daftar pustaka. Sebagai salah satu syarat untuk memperoleh gelar Sarjana Program Studi Informatika pada Fakultas Sains dan Teknologi, Universitas Islam Negeri Sunan Kalijaga.

Yogyakrta, 4 Agustus 2025
Yang membuat pernyataan,

Farid Anwar Wahdie
21106050049

LEMBAR PERSETUJUAN TUGAS AKHIR

Universitas Islam Negeri Sunan Kalijaga FM-UINSK-BM-05-03/R0

SURAT PERSETUJUAN SKRIPSI/TUGAS AKHIR

Hal : Persetujuan Skripsi / Tugas Akhir

Lamp : -

Kepada

Yth. Dekan Fakultas Sains dan Teknologi

UIN Sunan Kalijaga Yogyakarta

Di Yogyakarta

Assalammu'alaikum wr. wb.

Setelah membaca, meneliti, memberikan petunjuk dan mengoreksi serta mengadakan perbaikan seperlunya, maka saya selaku pembimbing berpendapat bahwa skripsi Saudari:

Nama	:	Farid Anwar Wahdie
NIM	:	21106050049
Judul Skripsi	:	PENGEMBANGAN MULTI ARCHITECTURE PATTERN CODE GENERATOR UNTUK APLIKASI TYPESCRIPT DENGAN FRAMEWORK EXPRESS.JS

Sudah dapat diajukan kembali kepada Program Studi Informatika Fakultas Sains dan Teknologi UIN Sunan Kalijaga Yogyakarta sebagai salah satu syarat untuk memperoleh gelar Sarjana Strata Satu dalam Program Studi Informatika.

Dengan ini kami mengharap agar skripsi/tugas akhir Saudari dapat segera dimunaqasyahkan. Atas perhatiannya saya ucapkan terima kasih.

Wassalammu'alaikum wr. wb.

Yogyakarta, 5 Agustus 2025

Pembimbing

Dr. Ir. Sumarsono, S.T., M.Kom.,

19710209 200501 1 003

INTISARI

Perkembangan teknologi, terutama dalam perangkat lunak berbasis web tumbuh secara pesat selaras dengan kebutuhan. Kecepatan dan ketepatan dalam pembangunan aplikasi web menjadi tantangan tersendiri bagi para pengembang. Pengaturan *setup* awal projek dan pembuatan REST API backend, terutama yang dibuat dengan framework Express.js melibatkan banyak pengkodean dan konfigurasi berulang yang terbukti memakan waktu dan berpotensi menimbulkan inkonsistensi arsitektur. Untuk mengatasi masalah ini, penelitian ini menyajikan pengembangan kode generator bernama *generator-express-ts-antistress* yang dirancang untuk mengotomatisasi setup aplikasi dan pembuatan kerangka aplikasi backend. Generator yang dibangun berbasis *Command-Line Interface* (CLI) dan dikembangkan menggunakan *framework* Yeoman. Dengan menerapkan pendekatan berbasis template, generator akan menerjemahkan skema JSON menjadi kerangka kode REST API yang dapat langsung dikembangkan. Pengembangan generator ini mengikuti metodologi Extreme Programming (XP) dengan dua iterasi. Iterasi pertama berfokus pada implementasi inti, yaitu konversi skema JSON menjadi kode scaffold REST API beserta kode unit testing. Iterasi kedua berfokus pada perbaikan-perbaikan berdasar pengujian iterasi pertama. Alat yang dihasilkan berhasil membuat proyek REST API yang fungsional dengan dukungan untuk pola arsitektur *Model-View-Controller* (MVC) dan *Layered*.

Kata Kunci: Generator Kode, Express.js, TypeScript, MVC, Layered Architecture

ABSTRACT

The development of technology, especially in web-based software, has grown rapidly in line with increasing demands. Speed and accuracy in web application development have become a particular challenge for developers. Initial project setup and the creation of a REST API backend—especially when built with the Express.js framework—involve extensive coding and repetitive configurations, which can be time-consuming and may lead to architectural inconsistencies. To address this issue, this study presents the development of a code generator named generator-express-ts-antistress, designed to automate application setup and the creation of backend application scaffolding. The generator is built as a Command-Line Interface (CLI) tool and developed using the Yeoman framework. By applying a template-based approach, the generator translates a JSON schema into a REST API code scaffold that can be immediately further developed. The development of the generator follows the Extreme Programming (XP) methodology with two iterations. The first iteration focuses on core implementation, namely converting a JSON schema into a REST API scaffold along with unit testing code. The second iteration focuses on improvements based on testing from the first iteration. The resulting tool successfully generates a functional REST API project with support for both Model-View-Controller (MVC) and Layered Architectural patterns.

Keywords: *Code Generation, Express.js, TypeScript, MVC Architecture, Layered Architecture.*

MOTTO

لا تكن جلدا وتظهر الجزع

“Jangan mengaku kuat kalau masih sambat”

بجد لا بجد كل مجد, فهل جد بلا جد بمجد

“Sukses dari usahamu bukan dari kakek nenekmu. Bukankah mereka juga tak sukses tanpa perjuangan?”

HALAMAN PERSEMBAHAN

Dengan segala kerendahan hati, karya sederhana ini penulis persembahkan kepada Ayah dan Ibu tercinta, yang dengan tulus ikhlas memberikan doa, kasih sayang, dukungan, serta pengorbanan yang tiada henti. Semoga karya ini menjadi wujud kecil dari rasa hormat dan terima kasih penulis atas segala jerih payah dan cinta yang telah diberikan sepanjang hayat.

KATA PENGANTAR

Segala bentuk puji milik Allah Subḥānahu wata‘ālā, atas segala rahmat dan karunia-Nya penulis dapat menyelesaikan tugas akhir yang berjudul *“Pengembangan Multi Architecture Pattern Code Generator untuk Aplikasi Typescript dengan Framework Express.Js”*. Shalawat serta salam senantiasa tercurah kepada junjungan Nabi Muhammad SAW, yang menjadi teladan bagi umat manusia hingga akhir zaman.

Penyusunan tugas akhir ini tidak terlepas dari bantuan, dukungan, serta doa dari berbagai pihak. Oleh karena itu, pada kesempatan ini penulis menyampaikan ucapan terima kasih yang sebesar-besarnya kepada:

1. Prof. Noorhaidi Hasan, S.Ag., MA., M.Phil., Ph.D., selaku Rektor UIN Sunan Kalijaga Yogyakarta.
2. Prof. Dr. Dra. Hj. Khurul Wardati, M.Si., selaku Dekan Fakultas Sains dan Teknologi UIN Sunan Kalijaga Yogyakarta.
3. Bapak Muhammad Mustakim, S.T. M.T., selaku Ketua Program Studi Informatika Fakultas Sains dan Teknologi UIN Sunan Kalijaga Yogyakarta.
4. Bapak Nurochman, S.Kom., M.Kom., selaku dosen penasehat akademik penulis selama menempuh studi di UIN Sunan Kalijaga.
5. Bapak Dr. Ir. Sumarsono, S.T., M.Kom., selaku dosen pembimbing yang dengan sabar memberikan bimbingan, arahan, serta masukan berharga dalam penyusunan tugas akhir ini.
6. Seluruh dosen dan karyawan Program Studi Informatika UIN Sunan Kalijaga atas ilmu, arahan, dan bantuan yang telah diberikan.
7. Kedua orang tua penulis, Bapak Kholid Wahyudi dan Ibu Mudrikah, atas segala doa, kasih sayang, serta dukungan moral maupun material yang tidak ternilai.
8. Seluruh teman angkatan Informatika 2021 UIN Sunan Kalijaga yang senantiasa memberikan semangat, kebersamaan, dan dukungan selama perkuliahan.

Penulis menyadari bahwa tugas akhir ini masih jauh dari sempurna, baik dari segi isi maupun penyajiannya. Oleh karena itu, penulis sangat mengharapkan kritik dan saran yang bersifat membangun demi perbaikan di masa mendatang. Semoga tugas akhir ini dapat memberikan manfaat serta menjadi kontribusi bagi pengembangan ilmu pengetahuan.

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR LAMPIRAN

BAB I
PENDAHULUAN

1.1 Latar Belakang

Kebutuhan terhadap aplikasi yang tinggi menuntut kecepatan dalam pengembangan perangkat lunak. Namun, saat ini masih banyak tantangan yang sering ditemui, salah satunya proses pembuatan kode secara manual. Proses ini tak hanya memakan waktu yang lama, namun juga rentan terhadap kesalahan. Selain itu, proses ini sulit beradaptasi jika terjadi perubahan pada kebutuhan arsitektur [4]. Tak hanya itu, pembuatan struktur kode, *file-file* program, dan konfigurasi database juga menjadi pekerjaan berulang yang harus dikerjakan pengembang. Pengembang harus menuliskan kode dan membuat struktur *file-file* yang mirip setiap membuat aplikasi [5]. Seperti, developer *backend* harus membuat controller dasar operasi CRUD untuk beberapa entitas, mendefinisikan model, dan membuat validasi *form create* atau *update* yang merupakan pekerjaan yang berulang [6]. Seharusnya pekerjaan yang mirip dan berulang ini bisa dilakukan hanya sekali [5].

Pendekatan manual tentu sangat mempengaruhi produktivitas pengembang. Tak hanya produktivitas, pendekatan ini juga berpotensi memberikan dampak negatif pada kualitas software. Hal ini dijelaskan oleh Robert C. Martin pada bukunya yang berjudul *Clean Code: A Handbook of Agile Software Craftsmanship*. Konsistensi dan penerapan *clean architecture* merupakan aspek fundamental untuk mengurangi adanya bug sistem dan mempermudah pemeliharaan kode software [4]. Kini, semakin banyak penelitian yang memepelajari tentang metode-metode baru dalam merancang dan mengembagkan aplikasi web. Penelitian-penelitian ini memiliki beberapa fokus, seperti memudahkan perancangan dan perkembangan aplikasi, mempercepat pengembangan dan mengurangi biaya produksi, serta mengurangi bug dalam perangkat lunak [5].

Dalam mengatasi permasalahan ini *Automatic Code Generation* hadir sebagai Solusi yang inovatif dalam software development. Menurut Charles Rich dan Richard Waters pada tulisan mereka yang berjudul *Automatic Programming: Myths and Prospects*, pemrograman otomatis merupakan tujuan yang usianya setua dengan pemrograman itu sendiri. Konsep ini pertama kali dikenalkan pada tahun 1950, yang bermula dari teknologi *compiler* [7]. Sejak saat itu, Perusahaan-perusahaan software mulai mengadopsi teknik kode sintesis dalam pembuatan software. Tujuannya mengurangi waktu pengembangan dan meningkatkan produktivitas [8]. Penggunaan alat generator kode memberikan keefisienan, karena generator yang sama dapat digunakan kembali untuk menghasilkan aplikasi-aplikasi yang berbeda sesuai dengan input yang diterima [9].

Perlu dipahami, code generator yang dimaksud disini adalah istilah yang mengacu pada pemrograman otomatis. Karena istilah ini juga digunakan pada konteks penelitian kompiler yang bukan dari fokus penelitian ini. *Code generator* pada pemrograman otomatis mengacu pada proses pembuatan software secara otomatis yang memanfaatkan input dari pengguna. Input bisa berupa skema database, parameter, diagram uml, model, atau yang lainnya. Adapun keluaran yang dihasilkan bisa berupa *file* source code, *file* konfigurasi, struktur *file* kode, serta bisa juga object dan record basis data [5].

Pada saat tulisan ini dibuat, telah banyak teknik yang digunakan dalam *Code generator*, salah satunya adalah *Template-Based Code Generation* (TBCG). Pendekatan ini telah ada sejak pertengahan tahun 1990. Pendekatan ini memberikan kemudahan bagi pengembangnya, karena membutuhkan lebih sedikit upaya dalam pengembangannya. Code Generation berbasis template juga lebih disukai karena mengikuti prinsip *write once, produce many* [9]. Sudah banyak program yang menggunakan konsep ini, seperti Dreamweaver milik adobe yang menghasilkan halaman web. Ada juga alat perangkat lunak Computer Aided Software Engineering (CASE) yang bisa menghasilkan kode hanya dari diagram UML saja [9].

Lebih lanjut, *code generator* berbasis template merupakan bagian penting dari Model-Driven Architecture (MDA). Bisa dikatakan code generator menjadi teknik populer yang digunakan dalam MDA. Hal ini dikarenakan keduanya memberikan

penekanan pada abstraksi dan otomatisasi [9]. MDA merupakan pendekatan dalam perancangan, pengembangan, dan implemetasi perangkat lunak yang diusung oleh OMG (*Object Management Group*). MDA memberikan pedoman dalam menyusun spesifikasi perangkat lunak yang diekspresikan sebagai model [3]. Paradigma utama teknik ini ialah mentransformasikan model menjadi baris-baris kode (*model-to-text*) [12]. MDA memungkinkan pengembangan perangkat lunak dengan menjadikan model sebagai pondasi utamanya. MDA juga menekankan bahwa dengan mendefinisikan model secara tepat, transformasi kode dapat dilakukan secara otomatis untuk menghasilkan implementasi yang konsisten dan efisien [3].

Pada pengembangan aplikasi berbasis web modern, bagian pengembangannya dipisahkan menjadi *server side* dan *client side*. *Client side* merujuk pada semua hal yang ditampilkan oleh aplikasi kepada *client* atau end user, termasuk teks, Gambar, grafik, dan bagian User *Interface* lainnya. Selain itu, *client side* juga mencakup setiap aksi yang dapat dilakukan oleh user pada aplikasi yang ditampilkan melalui web browser [13]. Pengembang aplikasi web bagian *client side* sering disebut dengan pengembang *frontend* [14]. Sedangkan *server side* merujuk pada setiap hal yang terjadi pada *server*, meliputi interasksi dengan database, mengelola data yang akan ditampilkan ke klien, serta memproses setiap permintaan yang dikirimkan oleh klien. Sisi *server* ini sering disebut dengan *backend*, walaupun sebenarnya keduanya tidak menunjukkan makna yang sama. *Server side* berarti lokasi setiap proses dijalankan, sedangkan *backend* merupakan tipe dan proses itu sendiri [13].

Komunikasi antara *server* dengan *client* bisa terjadi dengan bantuan API (*Application Programming Interface)* [15]. API adalah antarmuka yang berbasis *server-side* yang memungkinkan dua komponen perangkat lunak (*server* dan *client*) atau antar perangkat lunak melakukan komunikasi dengan serangkaian definisi dan *protocol*. Makna dari singkatan API, Application berarti perangkat lunak yang memiliki fungsi berbeda. Sedangkan *interface* dapat diartikan sebagai layanan kontrak antara dua aplikasi. Kontrak ini mengatur tentang cara dua aplikasi berbeda berkomunikasi menggunakan *request* dan *response* sebagai metodenya [16].

Setiap aplikasi berbasis web merupakan API, tapi tak semua API berupa aplikasi website. Hal ini didasarkan pada aplikasi berbasis web memiliki dua

komponen perangkat lunak yang saling berkomunikasi, yaitu *server side* dan *client side*. API yang digunakan dalam aplikasi web disebut Web API [16]. Web API menjadi antarmuka yang mengatur proses pertukaran informasi antara web *server* dengan browser pengguna. Web API mendukung fungsi Create, Read, Update, dan Delete (CRUD) yang menjadi fungsi dasar pada aplikasi. Fungsi ini bekerja melalui HTTP *protocol* dengan method, seperti *GET*, *POST*, *PUT*, dan *DELETE*. *Response* pada HTTP *protocol* memiliki *response* Accept Header, *response* status code, *response* data yang berupa JSON dan XML, dan sebagainya [18].

Berdasarkan arsitekturnya, API memiliki beberapa jenis, salah satunya adalah REST API. REST API menjadi istilah yang merujuk pada Web API Modern. REST (Representational State Transfer) merupakan arsitektur yang diaplikasikan untuk membuat layanan yang digunakan pada berbagai environment dan platform yang mendukung sistem berbeda berbagi informasi dan mendukung WWW (World Wide Web) [19]. Fitur utama yang dimiliki REST ialah *stateless* dan *cross-platform*. Stateless berarti *server* tidak menyimpan informasi klien diantara *request* [16]. Proses yang terjadi pada REST API ialah klien mengirimkan *request* ke *server* yang disebut data. Lalu, *server* menerima data tersebut untuk menjalankan fungsi internal dan mengembalikan hasil olahan data (*output*) ke klien lagi. Saat ini, REST API menjadi arsitektur API yang paling popular dan paling banyak digunakan [5].

Pemilihan bahasa pemrograman dan teknologi yang digunakan untuk membuat REST sangatlah penting, karena dapat mempengaruhi kinerja *server*. Perngaruh ini termasuk waktu *response*, penggunaan memori, *CPU*, dan *resource* lainnya [24]. Banyak penelitian yang sudah dilakukan untuk mengetahui teknologi mana yang lebih baik digunakan dalam pengembangan REST. Tentunya hasil penelitian ini hanya menilai dari beberapa aspek, artinya setiap teknologi *framework* memiliki kelebihan masing-masing dalam beberapa aspek. Dari berbagai hasil penelitian, Express.js menjadi teknologi yang paling baik dalam segi performa, termasuk kecepatan *response*, penggunaan memori, dan *CPU* [20][21][24]. Dengan begitu, tak ayal jika Express.js menjadi salah satu *framework* yang paling sering digunakan untuk membuat REST API [24].

Express.js merupakan *framework* JavaScript yang berjalan di environment Node.js. *Framework* ini dirilis pada tahun 2010 oleh TJ Holowaychuk [25]. Express.js terkenal sebagai *framework* yang ringan, karena tidak terlalu membutuhkan banyak dependensi tambahan [24]. Pada *framework* ini terdapat banyak *middleware* yang dapat dimanfaatkan dalam pengembangan web. Middleware Express berupa fungsi-fungsi yang mempunyai akses ke objek *request*, *response* dan *next* dalam siklus proses yang berlangsung di aplikasi. Middleware yang ada dapat digunakan untuk otentikasi, pengolahan data, *logging, throw error,* dan lainnya [25]. Keunggulan lain yang dimiliki Express.js adalah bersifat *unopinionated*. *Unopinionated* berarti tidak ada aturan baku dalam penggunaan Express.js [25]. Hal ini membuat pengembang memiliki kebebasan untuk menentukan struktur aplikasi, struktur *folder*, *workflows*, *tools* dan lainnya [26].

Unopinionated memang memberikan keleluasaan bagi programmer. Mereka bebas berkreasi dalam menentukan cara dan kebutuhannya. Tapi kelebihan ini bisa menjadi kelemahan Express.js. Karena tidak ada panduan pasti, programmer pemula pasti akan kebingungan dalam mengunakan express. Mereka tidak memiliki standar kebenaran dalam mengambangkan aplikasi dengan Express.js [27]. Selain itu, Ketika menggunakan Express.js, pengembang tidak diberikan struktur *folder* aplikasi. Sehingga pengembang harus membuat *folder* dan *file-file* yang diperlukan secara manual. Dengan kata lain mereka harus membuat setup awal proyek secara manual [23]. Selain itu, mereka juga harus membuat konfigurasi database, model entitas, validasi data, *routing*, *error handling*, autentikasi dan otorisasi, *logging*, dan *testing* secara manual. Ini juga merupakan masalah yang sering ditemui oleh pengembang aplikasi web.

Untuk mengatasi masalah yang tersebut, perlu dikembangkan *code generator* berbasis template yang mampu mengonversi *JSON Schema* sebagai input (model) menjadi kode-kode Express.js. Hal ini sejalan dengan metode *Model-Driven Architecture* dan *Code generator* berbasis template yang terbukti memberikan kemudahan dan kecepatan bagi pengembang dalam mengembangkan aplikasi web [5][9]. *JSON Schema* akan dijadikan dasar dalam pembuatan model, controller, routing, validation, dan lainnya. Pendekatan ini juga memungkinkan pembuatan

struktur aplikasi secara otomatis. Kode yang dihasilkan akan berupa kode-kode *boilerplate*, sehingga pengembang bisa langsung fokus ke logika bisnis dan fitur-fitur unik aplikasi.

Penelitian ini akan mengembangkan *Code Generator* yang memanfaatkan *JSON Schema* yang mendefinisikan model-model entitas sebagai input dan mengonversinya menjadi kode TypeScript yang menggunakan teknologi Express.js. Walaupun Express.js merupakan *framework* milik JavaScript, banyak developer justru memilih menggunakan TypeScript dalam penulisan kode programnya. Pemilihan ini bukan tanpa alasan. Dikutip dari halaman resmi TypeScript, TypeScript merupakan bahasa JavaScript yang memiliki sintaks tipe data. TypeScript juga dikembangkan oleh Microsoft dari bahasa JavaScript. Selain itu, pada level *production*, kode Typescript juga akan dikompilasi ke JavaScript menggunakan TSC (Typescript Compiler). Bisa dikatakan sintaksis pada TypeScript merupakan perkembangan dari sintaksis JavaScript [28]. Pengunaan TypeScript akan memberikan peningkatan keandalan dan maintainability kode. Sistem pengetikan statis yang diterapkan TypeScript akan mendeteksi kesalahan-kesalahan pada tahap kompilasi. Pendeteksian kesalahan ini akan mengurangi jumlah bug pada aplikasi [29].

Selain itu, *Code Generator* yang dibuat akan mendukung dua *Architecture Pattern* yang sering digunakan dalam pengembangan aplikasi, yakni *MVC* dan *Layered Architecture*. Software Architecture Pattern merupakan struktur Tingkat tinggi dari sistem perangkat lunak yang mencakup serangkaian aturan, pola, dan pedoman yang mengatur interaksi antar komponen. Pola Arsitektur berfungsi seagai *blueprint* yang memastikan sistem memenuhi prasyaratnya dan mudah untuk dipelihara dan dikembangkan. Pola arsitektur mengatur bagaimana antar komponenn aplikasi saling terhubung dan berinteraksi [30]. Manfaat dari penerapan pola arsitektur dalam apilkasi memiliki beberapa manfaat seperti, meningkatkan efisiensi, efktivitas, produktivitas, kecepatan pengembangan aplikasi, menekan biaya pengembangan, mudah dipelihara, menjaga konsistensi struktur aplikasi, dan lainnya [31].

MVC dipilih karena sudah teruji meningkatkan produktivitas dalam pengembangan aplikasi website [32]. Hal ini disebabkan pola *MVC* mendukung pemisahan tanggung jawab pengembangan website menjadi frontend dan backend.

Arsitektur MVC membagi komponen aplikasi menjadi tiga bagian utama, *Model*, *Controller*, dan *View*. *Model* memiliki tangung jawab terhadap logika dan pengolahan database. Controller bertugas untuk mengatur alur data yang dikirimkan oleh klien melalui *View* dan *server* melalui Model. Controller juga menangani interkasi yang dilakukan oleh user. Sedangkan View mengatur tampilan yang ditampilkan pada sisi klien. Dengan terbaginya komponen aplikasi, pengembang bisa fokus terhadap masing-masing komponen tanpa mengganggu komponen lain [33].

Sedangkan *Layered Architecture* dipilih karena kemampuannya memisahkan logika bisnis dari detail implementasi penyimpanan data. Hal ini memungkinkan pengembang untuk menggunakan API yang konsisten tanpa harus khawatir tentang sumber data, baik dari database, API pihak ketiga, ataupun media penyimpanan lainnya [34]. *Layered Architecture* mengabstraksikan akses data dan mengimplementasikan cara terpusat untuk mengelola operasi data. Pola ini memisahkan antar *business logic* dengan *query logic* sehingga keduanya tidak langsung ditangani oleh *controller* [35]. Dengan pola ini, komponen aplikasi menjadi mudah untuk dilakukan pengujian, pemeliharaan, dan pengembangan skala besar, karena perubahan pada data store dapat dilakukan tanpa mengganggu logika bisnis yang ada [34].

Penulis mengusulkan menggunakan dua *Architecture Pattern* berbeda agar tidak menghilangkan kelebihan Express.js, yakni bersifat *Unopinionated* yang memberikan kebebasan penggunanya dalam menentukan struktur programnya. Dengan adanya dua *Architecture Pattern* yang berbeda, diharapkan pengembang bisa memilih diantara keduanya. Secara keseluruhan, alat yang dibangun merupakan aplikasi berbasis CLI yang menerapkan konsep *Code Generator* berbasis template untuk menghasilkan kode yang ditulis menggunakan *Typescript* dengan teknologi Express.js. Kode yang dihasilkan didasarkan input *JSON Schema* yang mendefiniskan entitas, mulai dari nama, atribut, tipe data atribut, dan lain-lain. *Output* program yang dihasilkan adalah struktur folder apilkasi dan kode *REST API* yang terdiri dari kode *boilerplate* dan hasil rendering dari template yang telah disesuaikan. Kode yang dihasilkan juga akan menggunakan pola arsitektur *MVC* atau *Layered Architecture*.

Kode *boilerplate* dan template memiliki banyak kemiripan, akan tetapi keduanya memiliki perbedaan yang mendasar. Kode boilerplate merupakan kode yang digunakan berulang kali pada konteks yang berbeda tanpa atau hanya dengan sedikit perubahan. Dengan kata lain, boilerplate dapat digunakan apa adanya tanpa penyesuaian. Sedangkan template merujuk pada kode yang dapat digunakan Kembali dengan beberapa penyesuaian. Dengan begitu, template dapat digunakan untuk membuat beberapa code dengan tujuan berbeda namun memiliki kerangka yang sama [36].

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah dituturkan diatas, masalah dapat dirumuskan sebagai berikut:

1. Bagaimana merancang aplikasi berbasis CLI yang merubah input *JSON Schema* menjadi kode REST API TypeScript dengan framework Express.js?
2. Bagaimana menerapkan struktur aplikasi Express.js menggunakan *Architecture Pattern* MVC dan *Layered Architecture*?

1.3 Tujuan

Adapun tujuan tugas akhir ini yaitu:

1. Membangun alat konversi *JSON Schema* menjadi kode *backend* aplikasi website dengan pendekatan *multi-model architecture pattern* (*MVC* dan *Layered Architecture*)
2. Mengotomatisasi pengembangan aplikasi website yang ditulis dengan bahasa Typescript menggunakan teknologi *framework* Express.js.
3. Mengurangi waktu setup awal proyek aplikasi web dengan menyediakan struktur dan template yang sudah jadi.

1.4 Manfaat

Hasil dari pengembangan perangkat lunak *code generator* ini diharapkan memeberikan beberapa manfaat sebagai berikut:

1. Membantu pengembang yang mengunakan bahasa Typescript dan *framework* Express.js dalam proses pembangunan aplikasi website.
2. Mengurangi potensi *human error* dalam penulisan kode *backend*.
3. Meningkatkan konsistensi dan standarisasi kode dalam pengembangan perangkat lunak.
4. Memberikan studi kasus pada pengembangan pengembangan *code generator*, sehingga membuka peluang penelitian optimasi *code generator* berbasis template dengan bahasa pemrograman yang sama atau yang lain.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil perancangan dan implementasi sistem generator yang diberi nama generator-express-ts-antistress, dapat ditarik kesimpulan yang menjawab rumusan masalah. Perancangan aplikasi genetator berbasis CLI yang mengubah input skema model JSON menjadi kode REST API dilakukan dengan beberapa tahapan inti. Tahapan yang pertama, mendefinisikan struktur skema model JSON yang mencakup nama model, field, dan relasi. Kedua, membangun validator skema dengan menggunakan Ajv dan logika buatan untuk memastikan integritas skema yang dibuat. Ketiga, memanfaatkan package generator yang memiliki beberapa function khusus yang dapat digunakan dalam generator. Selain itu, dengan membuat template dengan memanfaatkan *template engine* EJS dan diparsing dengan model json dengan kelas-kelas dalam generator. Hasil parsing dari *template* adalah file-file kode TypeScript, termasuk model, konfigurasi, dan struktur dasar aplikasi Express.js.

Penerapan *architecture pattern* MVC dan Layered Architecture diimplementasikan dengan menggunakan logika kondisional di dalam generator. Sebelum generator membuat file-file aplikasi, pengguna diwajibkan untuk memilih nama, input skema, database yang digunakan, dan pola arstitektur yang digunakan. Generator akan memilih template file dari setiap komponen berdasarkan input yang diberikan pengguna. Pada MVC, generator akan menghasilkan file models, routes, middleware dan lainya. Controller pada MVC yang dibuat akan menangani logika bisnis dan akses data secara langsung. Pada Layered Architecture, generator akan menambahkan lapisan abstraksi tambahan dengan menghasilkan direktori services dan repositories. Dengan begitu pola Layered akan memisahkan tanggung jawab controller. Pada layered, controller bertugas untuk menerima *request* dan mengirim *response*, service bertugas untuk mengatur logika bisnis, dan repository untuk akses data dari database.

5.2 Saran

Meskipun sistem generator yang dibangun telah berhasil memenuhi tujuannya, terdapat beberapa potensi pengembangan di masa mendatang untuk meningkatkan fungsionalitas dan cakupannya:

1. Dukungan Arsitektur Tambahan: Generator dapat dikembangkan lebih lanjut untuk mendukung arsitektur yang lebih modern seperti Clean Architecture atau Hexagonal Architecture, memberikan pilihan yang lebih luas kepada developer.
2. Integrasi Database Lain: Selain database SQL (melalui Sequelize) dan MongoDB (melalui Mongoose), dukungan untuk database NoSQL lain seperti Redis atau Firebase Firestore dapat ditambahkan.
3. Fitur Otomatisasi Lanjutan: Generator dapat ditingkatkan untuk menghasilkan fitur-fitur tambahan secara otomatis, seperti skrip database seeder, konfigurasi Docker (Dockerfile), dan pipeline CI/CD dasar untuk mempermudah proses deployment.
4. Ekspansi ke Frontend: Membangun generator pendamping yang dapat menghasilkan kode frontend (misalnya, React atau Vue) berdasarkan skema JSON yang sama, sehingga dapat menciptakan full-stack application generator.

DAFTAR PUSTAKA

[1] Agus Tri Haryanto, "APJII Jumlah Pengguna Internet Indonesia Tembus 221 Juta Orang," Detik, Accessed: Mar. 10, 2025. [Online]. Available: https://inet.detik.com/cyberlife/d-7816040/jumlah-pengguna-internet-indonesia-tembus-212-juta-di-2025

[2] D. Maharani, F. Helmiah, and N. Rahmadani, "Penyuluhan Manfaat Menggunakan Internet dan Website Pada Masa Pandemi Covid-19," *Abdiformatika: Jurnal Pengabdian Masyarakat Informatika*, vol. 1, no. 1, pp. 1–7, May 2021, doi: 10.25008/abdiformatika.v1i1.130.

[3] G. Paolone, M. Marinelli, R. Paesani, and P. Di Felice, "Automatic code generation of mvc web applications," *Computers*, vol. 9, no. 3, pp. 1–29, Sep. 2020, doi: 10.3390/computers9030056.

[4] R. C. Martin, "Clean Code," Boston, Jul. 2008.

[5] B. Uyanik and V. H. Şahin, "A template-based code generator for web applications," *Turkish Journal of Electrical Engineering and Computer Sciences*, vol. 28, no. 3, pp. 1747–1762, 2020, doi: 10.3906/elk-1910-44.

[6] P. Egli and K. Ibrahim, "A New Code Generation Tool for Rapid Application Development," Zurich, Jun. 2023.

[7] C. Lewis, "Automatic Programming and Education," in *ACM International Conference Proceeding Series*, Association for Computing Machinery, Mar. 2022, pp. 70–80. doi: 10.1145/3532512.3539664.

[8] S. Kelly, J.-P. Tolvanen, and J. Wiley, "DOMAIN-SPECIFIC MODELING Enabling Full Code Generation A Wiley-Interscience Publication," 2008.

[9] E. Syriani, L. Luhunu, and H. Sahraoui, "Systematic Mapping Study of Template-based Code Generation," Mar. 2017, doi: 10.1016/j.cl.2017.11.003.

[10] M. A. Possatto and D. Lucrédio, "Automatically propagating changes from reference implementations to code generation templates," *Inf Softw Technol*, vol. 67, pp. 65–78, 2015.

[11] A. Akbulut and S. Toprak, "Code Generator Framework for Smart TV Platforms," *IET Software*, vol. 13, no. 4, pp. 268–279, 2019.

[12] A. C. Bock and U. Frank, "Low-Code Platform," *Business and Information Systems Engineering*, vol. 63, no. 6, pp. 733–740, Dec. 2021, doi: 10.1007/s12599-021-00726-8.

[13] "What do client side and server side mean? | Client side vs. server side," Cloudflare. Accessed: Mar. 10, 2025. [Online]. Available: https://www.cloudflare.com/learning/serverless/glossary/client-side-vs-server-side/

[14] "What's the Difference Between Frontend and Backend in Application Development?," Amazon Web Services. Accessed: Mar. 10, 2025. [Online]. Available: https://aws.amazon.com/compare/the-difference-between-frontend-and-backend

[15] A. G. Kleppe, J. B. Warmer, and W. Bast, *MDA explained: the model driven architecture: practice and promise*. Addison-Wesley Professional, 2003.

[16] "What is an API (Application Programming Interface)?," Amazon Web Services. Accessed: Mar. 08, 2025. [Online]. Available: https://aws.amazon.com/what-is/api

[17] I. R. D. Muhammad and I. V. Paputungan, "Development of Backend Server Based on REST API Architecture in E-Wallet Transfer System," *Jurnal Sains, Nalar, dan Aplikasi Teknologi Informasi*, vol. 3, no. 2, pp. 79–87, Jan. 2024, doi: 10.20885/snati.v3.i2.35.

[18] S. N. Yanti and E. Rihyanti, "Penerapan Rest API untuk Sistem Informasi Film Secara Daring," *Jurnal Informatika Universitas Pamulang*, vol. 6, no. 1, p. 195, Mar. 2021, doi: 10.32493/informatika.v6i1.10033.

[19] A. Ehsan, M. A. M. E. Abuhaliqa, C. Catal, and D. Mishra, "RESTful API Testing Methodologies: Rationale, Challenges, and Solution Directions," May 01, 2022, *MDPI*. doi: 10.3390/app12094369.

[20] T. Purwanto, "Analisa Perbandingan Kinerja Rest Api Dengan Framework Flask, Laravel, Dan Express Js" 2023. [Online]. Available:

https://www.academia.edu/122821758/Analisis_Perbandingan_Performa_Restfull_Api_Antara_EXPRESS_JS_Dengan_Laravel_Framework

[21] L. Mulana, K. Prihandani, A. Rizal, U. Singaperbanga, and K. Abstract, "Analisis Perbandingan Kinerja Framework Codeigniter Dengan Express.Js Pada Server RESTful Api," *Jurnal Ilmiah Wahana Pendidikan*, vol. 8, no. 16, pp. 316–326, 2022, doi: 10.5281/zenodo.7067707.

[22] "10 Best Frontend JavaScript Frameworks to Use in 2025," Geeksforgeeks. Accessed: Mar. 09, 2025. [Online]. Available: https://www.geeksforgeeks.org/best-frontend-javascript-frameworks/

[23] Prasatya, "Framework JavaScript untuk Back-end Developer," CODEPOLITAN. Accessed: Mar. 10, 2025. [Online]. Available: https://www.codepolitan.com/blog/framework-javascript-untuk-backend-developer/

[24] B. Sutara and S. S. Gunawan, "Comparative Analysis of Rest Api Performance Between Express.Js Framework And Hapi.Js Using Apache Jmeter," 2024.

[25] Prasetya, "Apa Itu Express JS: Pengertian, Manfaat, Contoh Syntax," CODEPOLITAN. Accessed: Mar. 11, 2025. [Online]. Available: https://www.codepolitan.com/blog/apa-itu-express-js-pengertian-manfaat-contoh-syntax/

[26] M. Medhat, "Opinionated vs. Non-Opinionated Frameworks: Understanding the Difference," DEV Community. Accessed: Mar. 12, 2025. [Online]. Available: https://dev.to/muhammadmedhat/opinionated-vs-non-opinionated-frameworks-understanding-the-difference-2379

[27] S. Galih, Indonesia. *Belajar NodeJS | 14. Apa itu Express?*, (Apr. 21, 2021). Accessed: Mar. 12, 2025. [Online Video]. Available: https://www.youtube.com/watch?v=TecGUz4bPFA

[28] A. Haapalainen, "A Comparative Analysis Of Converting Javascript to Typescript in Advanced Web Applications Course," 2024.

[29] "Advantages and Disadvantages of TypeScript over JavaScript," Geeksforgeeks. Accessed: Mar. 13, 2025. [Online]. Available:

https://www.geeksforgeeks.org/advantages-and-disadvantages-of-typescript-over-javascript/

[30] J. Satyabrata, "Types of Software Architecture Patterns," geeksforgeeks. Accessed: Aug. 02, 2025. [Online]. Available: https://www.geeksforgeeks.org/software-engineering/types-of-software-architecture-patterns/

[31] V. Walker, "14 software architecture design patterns to know," Redhat. Accessed: Aug. 02, 2025. [Online]. Available: https://www.redhat.com/en/blog/14-software-architecture-patterns

[32] S. Bhatia and H. Singh, "Analyzing and Improving Web Application Quality Using Design Patterns," Apr. 2012. [Online]. Available: www.ijctonline.com

[33] L. Rahmawati and S. Sumarsono, "Desain Pengembangan Website dengan Arsitektur Model View Controller pada Framework Laravel," *Jurnal Teknologi Dan Sistem Informasi Bisnis*, vol. 6, no. 4, pp. 785–790, Oct. 2024, doi: 10.47233/jteksis.v6i4.1497.

[34] T. McFarlin, "Repository Pattern Benefits: Why We Should Consider It," Tom McFarlin. Accessed: Mar. 15, 2025. [Online]. Available: https://tommcfarlin.com/repository-pattern-benefits/

[35] J. D. Waspada, "Apa itu Repository Pattern?," Medium. Accessed: Mar. 15, 2025. [Online]. Available: https://medium.com/@Dewey92/repository-pattern-what-e47ddee3364d

[36] M. Zaveri, "What is boilerplate and why do we use it? Necessity of coding style guide," FreeCodeCamp. Accessed: Jul. 09, 2025. [Online]. Available: http://freecodecamp.org/news/whats-boilerplate-and-why-do-we-use-it-let-s-check-out-the-coding-style-guide-ac2b6c814ee7/

[37] I. Wayan Sugiartana, P. Studi Administrasi Bisnis Denpasar, I. I. Sekolah Tinggi Ilmu Sosial dan Politik Wira Bhakti Denpasar Wayan Candra Winetra, P. Studi Teknik Elektro Denpasar, and I. Politeknik Negeri Bali Ida Ayu Putu Febri Imawati, "EFEKTIFITAS PENGGUNAAN CODE GENERATOR (UNDAGI CODE CREATOR) DALAM MEMBANGUN

SISTEM INFORMASI MODUL ADMIN," Online, 2022. [Online]. Available: https://ojs.mahadewa.ac.id/index.php/jmti

[38] Aflah Taqiu Sondha, Umi Sa'adah, Fadilah Fahrul Hardiansyah, and Maulidan Bagus Afridian Rasyid, "Framework dan Code Generator Pengembangan Aplikasi Android dengan Menerapkan Prinsip Clean Architecture," *Jurnal Nasional Teknik Elektro dan Teknologi Informasi*, vol. 9, no. 4, pp. 327–335, Dec. 2020, doi: 10.22146/jnteti.v9i4.572.

[39] E. A. Laksana, "PHP CRUD Generator Menggunakan Twig Template Engine dan Framework Codeigniter."

[40] I. A. Baltariu and C. Mironeanu, "Component generator for the development of RESTful APIs," in *2022 26th International Conference on System Theory, Control and Computing, ICSTCC 2022 - Proceedings*, Institute of Electrical and Electronics Engineers Inc., 2022, pp. 123–128. doi: 10.1109/ICSTCC55426.2022.9931876.

[41] A. Bochenek, "Can ChatGPT Replace a Template-based Code Generator?," in *Communication Papers of the 18th Conference on Computer Science and Intelligence Systems*, PTI, Oct. 2023, pp. 43–50. doi: 10.15439/2023f5691.

[42] I. Ullah and I. Inayat, " Template-based Automatic code generation for Web application and APIs Using Class Diagram," in *International Conference on Frontiers of Information Technology (FIT)*, 2022, pp. 332-337. doi: https://doi.org/10.1109/FIT57066.2022.00067.

[43] M. Vuković, V. Vujović, S. Milinković, Z. Štaka, and S. Čavoški, "From Visual Model to Source Code: Automated Fluent API Development Using Template-Based Code Generation in DSL Tools," in *2025 24th International Symposium INFOTEH-JAHORINA (INFOTEH)*, 2025, pp. 1–6. doi: 10.1109/INFOTEH64129.2025.10959264.

[44] X. She and B. Bian, "An Automatic Front-End Code Generation Method Based on Data and Template Integration," in *Proceedings of the 2024 Guangdong-Hong Kong-Macao Greater Bay Area International Conference on Digital Economy and Artificial Intelligence*, in DEAI '24. New York, NY,

USA: Association for Computing Machinery, 2024, pp. 463–467. doi: 10.1145/3675417.3675494.

[45] P. Gregory, C. Lassenius, X. Wang, and P. Kruchten, "Agile Processes in Software Engineering and Extreme Programming." [Online]. Available: http://www.springer.com/series/7911

[46] C. Puspitasari, "Metode pada Agile Development: Extreme Programming (Bagian 1)," Binus University. Accessed: Apr. 23, 2025. [Online]. Available: http://binus.ac.id/malang/2022/05/metode-pada-agile-development-extreme-programming-bagian-1/

[47] "Extreme Programming" Agile Alliance. Accessed: Apr. 25, 2025. [Online]. Available: https://www.agilealliance.org/glossary/xp/

[48] "Metode Agile: Kelebihan dan Tahapannya dalam Product Development," Digital Skola. Accessed: Apr. 23, 2025. [Online]. Available: https://digitalskola.com/blog/product-management/metode-agile

[49] D. Andriansyah and L. Nulhakim, "Extreme Programming Dalam Perancangan Sistem Informasi Jasa Fotografi."

[50] L. F. Andriani, "Extreme Programming: Tahapan, Kelebihan, dan Kekurangannya," Sekawan Media. Accessed: Apr. 26, 2025. [Online]. Available: https://www.sekawanmedia.co.id/blog/extreme-programming-adalah/

[51] R. Syaifuddin, "Implementasi Framework Express.js dan Flutter Pada Aplikasi Android 'ID Karier' Menggunakan Metode Prototyping," UIN Syarif Hidayatullah Jakarta, Jakarta, 2024.

[52] Prasatya, "Apa Itu Refactoring Code? Pengertian, Manfaat, dan Tekniknya," CODEPOLITAN. Accessed: Apr. 26, 2025. [Online]. Available: https://www.codepolitan.com/blog/apa-itu-refactoring-code-pengertian-manfaat-dan-tekniknya/

[53] R. Juliarto, "Apa itu UML? Beserta Pengertian dan Contohnya," Dcoding. Accessed: Jun. 19, 2025. [Online]. Available: https://www.dicoding.com/blog/apa-itu-uml/

[54] S. Ambler, “Class Responsibility Collaborator (CRC) Cards: An Agile Introduction,” Agile Modeling. Accessed: Jun. 19, 2025. [Online]. Available: https://agilemodeling.com/artifacts/crcModel.htm