Jurnal Informatika: Jurnal pengembangan IT
Vol. 10, No. 2, 2025
ISSN : 2477-5126, DOI:10.30591/jpit.v9ix.xxx

Integrasi Backend Golang-Echo pada Aplikasi Greenly sebagai Solusi Teknologi Pengelolaan Sampah Digital

Mutia Dwi Anggraeni[1], Fandy Setyo Utomo[2], Hendra Marcos[3]
[1]Universitas Amikom Purwokerto, Jl. Letjen Pol. Soemarto, Purwokerto, 53127, Indonesia

Info Artikel

Riwayat Artikel:

Received 2025-01-07
Revised 2025-05-02
Accepted 2025-05-15

Corresponding Author:
Mutia Dwi Anggraeni
Email: mutiaangg09@gmail.com

Abstract *– The waste problem in Indonesia is increasingly pressing, with 35.7% of the 31.9 million tons of national waste in 2023 not being managed properly. This study develops the Greenly web application backend as a digital solution to support waste reporting, recycling education, and increasing community participation through a gamification system. The methodology used is the Waterfall model, including needs analysis, design with Entity Relationship Diagram (ERD), implementation, and testing. The backend is built using the Golang and Echo frameworks, then packaged in Docker and deployed on the AWS EC2 service. The Continuous Integration/Deployment (CI/CD) process is carried out using GitHub Actions, with Nginx as a reverse proxy. Testing is carried out through Integration Test to ensure the reliability of key features such as CRUD data, waste reporting, and gamification. The results show that the backend system runs stably, safely, and efficiently, with an automatic CI/CD flow that is successfully executed without errors. The main contribution of this study is the provision of an adaptive and reliable backend as the foundation for a digital waste management system based on community participation.*

Keywords: *Backend, Golang, Integration Test, Docker, CI/CD*

Abstrak *– Masalah sampah di Indonesia semakin mendesak, dengan 35,7% dari 31,9 juta ton sampah nasional pada 2023 tidak terkelola dengan baik. Penelitian ini mengembangkan backend aplikasi web Greenly sebagai solusi digital untuk mendukung pelaporan sampah, edukasi daur ulang, dan peningkatan partisipasi masyarakat melalui sistem gamifikasi. Metodologi yang dig unakan adalah model Waterfall, meliputi analisis kebutuhan, perancangan dengan Entity Relationship Diagram (ERD), implementasi, serta pengujian. Backend dibangun menggunakan Golang dan Echo framework, kemudian dikemas dalam Docker dan di-deploy pada layanan AWS EC2. Proses Continuous Integration/Deployment (CI/CD) dilakukan menggunakan GitHub Actions, dengan Nginx sebagai reverse proxy. Pengujian dilakukan melalui Integration Test untuk memastikan keandalan fitur utama seperti CRUD data, pelaporan sampah, dan gamifikasi. Hasil menunjukkan sistem backend berjalan stabil, aman, dan efisien, dengan alur CI/CD otomatis yang berhasil dijalankan tanpa error. Kontribusi utama dari penelitian ini adalah penyediaan backend yang adaptif dan andal sebagai fondasi sistem pengelolaan sampah digital berbasis partisipasi masyarakat.*

Kata Kunci*: Backend, Golang, Integration Test, Docker, CI/CD*

I. PENDAHULUAN

Permasalahan sampah di Indonesia masih menjadi permasalahan nasional yang mendesak. Berdasarkan data Sistem Pengelolaan Sampah Nasional (SIPSN) Kementerian Lingkungan Hidup dan Kehutanan (KLHK) tahun 2023, per 24 Juli 2024, jumlah sampah nasional dari 290 kabupaten/kota mencapai 31,9 juta ton, dimana 35,7% atau 11,4juta ton tidak dapat dikelola Hal ini dicatat sebagai sampah tidak dikelola. Sampah dapat dipahami sebagai residu yang tidak diinginkan setelah selesainya suatu proses yang mencerminkan dampak aktivitas manusia [1]. Kesadaran masyarakat terhadap pengelolaan sampah masih tergolong rendah. Masih banyak masyarakat yang membakar sampah atau membuangnya ke sungai tanpa mempertimbangkan dampaknya, seperti penumpukan sampah di hilir dan pencemaran lingkungan. Selain itu, kemungkinan dampak negatif dari penggunaan lahan terlantar sebagai tempat pembuangan akhir, seperti bau tidak sedap, sarang nyamuk, dan risiko lingkungan yang tidak sehat, tidak diperhitungkan.

Situasi mengkhawatirkan terkait pengelolaan sampah di Indonesia dapat diatasi melalui peningkatan kesadaran masyarakat terhadap pentingnya membuang dan memilah sampah, sejalan dengan prinsip 3R (Reduce, Reuse, Recycle) [2]. Sayangnya, rendahnya kesadaran masyarakat dan kurangnya infrastruktur pengolahan sampah memperburuk kondisi ini. Seiring perkembangan era digital, teknologi menjadi alternatif solusi yang potensial dalam menjawab tantangan tersebut. Teknologi digital, khususnya dalam bentuk aplikasi pembelajaran interaktif dan sistem informasi, telah terbukti mendorong partisipasi aktif masyarakat melalui visualisasi informasi, edukasi digital, dan keterlibatan berbasis data [3].

Penelitian ini menawarkan solusi berupa pengembangan backend dari aplikasi Greenly, yang dirancang untuk mendukung sistem pelaporan sampah secara langsung, memberikan edukasi daur ulang yang terstruktur,

serta menerapkan mekanisme gamifikasi untuk meningkatkan keterlibatan pengguna. Backend dikembangkan menggunakan Golang dan Echo, serta di-deploy menggunakan teknologi modern seperti Docker, AWS EC2, dan GitHub Actions untuk efisiensi dan keandalan sistem.

Echo dipilih karena keunggulannya dalam efisiensi dan routing dibandingkan framework seperti Gin yang lebih kompleks, serta Fiber yang lebih cocok untuk proyek berperforma tinggi namun belum sepopuler Echo dalam dokumentasi komunitas di Indonesia.

Beberapa penelitian sebelumnya telah mengembangkan aplikasi terkait pengelolaan sampah, seperti sistem monitoring volume sampah berbasis IoT atau aplikasi edukasi lingkungan berbasis mobile. Namun, mayoritas penelitian tersebut lebih berfokus pada sisi frontend, sensor, atau kampanye sosial, dan belum banyak yang secara mendalam mengkaji aspek backend development sebagai tulang punggung integrasi data, skalabilitas sistem, serta automasi CI/CD dalam konteks pengelolaan sampah.

Dengan demikian, posisi penelitian ini mentikberatkan pada pengembangan infrastruktur digital (backend system) dengan automasi CI/CD dan deployment berbasis container untuk mendukung ekosistem edukatif dan partisipatif pengelolaan sampah. Kontribusi utama penelitian ini adalah menghadirkan solusi backend yang secure, scalable, dan maintainable, sekaligus membuktikan bahwa integrasi teknologi modern dapat memperkuat gerakan sosial dalam isu lingkungan melalui pendekatan teknis yang terukur dan efisien.

Back-end adalah sistem informasi atau area aplikasi tempat terjadinya proses, termasuk pengelolaan data yang ditambahkan, diubah, atau dihapus, dan mengacu pada server dan database [4]. Front-end, di sisi lain, bertanggung jawab untuk membuat antarmuka pengguna suatu aplikasi atau situs web [5]. Pengalaman pengguna merupakan persepsi dan reaksi seseorang yang dihasilkan dari suatu produk atau sistem yang berinteraksi dengannya [6]. Antarmuka Pengguna (User Interface/UI) adalah antarmuka pengguna teknologi informasi yang dimaksudkan untuk membantu pengguna menggunakan teknologi tersebut [7] .

Untuk memenuhi kebutuhan kesadaran dan pengelolaan sampah, aplikasi Greenly dilengkapi dengan berbagai fitur edukasi terkait daur ulang, pelaporan sampah, dan sistem gamifikasi. Bahasa pemrograman Go (Golang) dan kerangka Echo dipilih untuk mengembangkan backend aplikasi yang efisien. Golang menawarkan sejumlah fitur hebat, termasuk kesederhanaan, sintaksis yang mudah, manajemen memori yang baik, kecepatan proses kompilasi, dan dukungan untuk konkurensi dan pengetikan statis [8]. Selain itu, Go adalah bahasa pemrograman sumber terbuka gratis dengan dokumentasi lengkap yang mendukung kompilasi silang. Di sisi lain, framework Echo menawarkan beberapa keunggulan seperti router sederhana, skalabilitas, pengikatan data, rendering data, dan middleware yang didukung oleh teknologi HTTP/2 [9].

Permasalahan pengelolaan sampah di Indonesia semakin mendesak, dengan lebih dari sepertiga total sampah nasional tidak terkelola secara optimal. Rendahnya kesadaran masyarakat serta keterbatasan infrastruktur menjadi penyebab utama. Penelitian ini penting dilakukan untuk menjawab kebutuhan akan solusi digital yang dapat mendorong perubahan perilaku masyarakat secara partisipatif dan terstruktur.

Penelitian ini berkontribusi melalui pengembangan backend aplikasi Greenly menggunakan Golang dan framework Echo, yang dirancang untuk mendukung fitur pelaporan sampah, edukasi daur ulang, dan sistem gamifikasi. Sistem didukung oleh deployment berbasis Docker, CI/CD dengan GitHub Actions, serta layanan hosting AWS EC2 untuk memastikan efisiensi dan skalabilitas.

Berbeda dari penelitian terdahulu yang cenderung berfokus pada aplikasi edukasi atau pengelolaan sampah berbasis IoT, penelitian ini menitikberatkan pada penguatan sisi backend sebagai fondasi utama sistem digital. Dengan pendekatan ini, penelitian memberikan kontribusi signifikan terhadap pengembangan solusi pengelolaan sampah yang lebih terintegrasi, efisien, dan siap diimplementasikan secara luas.

II. METODE

Metodologi yang digunakan dalam penelitian ini adalah rekayasa perangkat lunak, dengan fokus pada tahapan perancangan, pengembangan, dan pengujian backend aplikasi Greenly menggunakan bahasa pemrograman Go (Golang) dan framework Echo. Pendekatan yang digunakan mengikuti model Waterfall, yang terdiri dari tahapan analisis kebutuhan, perancangan sistem, implementasi, pengujian, serta dokumentasi. Model ini dipilih karena alurnya yang sistematis dan sesuai untuk proyek pengembangan dengan kebutuhan yang telah didefinisikan secara jelas sejak awal. Metodologi ini dirancang secara terstruktur untuk menjawab rumusan masalah dan tujuan penelitian yang telah dikemukakan dalam bagian pendahuluan.

A. Rancangan Penelitian

Penelitian diawali dengan identifikasi permasalahan pengelolaan sampah di Indonesia melalui tiga pendekatan utama: (1) kajian literatur terhadap laporan resmi, jurnal, dan kebijakan pengelolaan sampah nasional; (2) analisis data sekunder dari Sistem Informasi Pengelolaan Sampah Nasional (SIPSN) milik Kementerian Lingkungan Hidup dan Kehutanan (KLHK) untuk mengetahui pola dan sebaran volume sampah

yang tidak terkelola; serta (3) observasi kebutuhan pengguna melalui studi awal terhadap perilaku masyarakat dalam membuang sampah, memahami edukasi daur ulang, dan penggunaan aplikasi digital lingkungan.

Berdasarkan hasil identifikasi, dikembangkan proses bisnis sistem yang menggambarkan alur interaksi antara pengguna dan sistem. Proses bisnis ini mencakup pelaporan sampah oleh pengguna, verifikasi data oleh sistem, pemberian poin sebagai bagian dari sistem gamifikasi, serta penyediaan konten edukasi mengenai daur ulang.

1. *Spesifikasi Kebutuhan Sistem*
   a. Kebutuhan Fungsional:
      - Pengguna dapat membuat akun dan melakukan login/logout.
      - Pengguna dapat mengirimkan laporan sampah beserta lokasi dan foto.
      - Sistem menyimpan laporan ke dalam basis data dan memproses verifikasi.
      - Pengguna dapat mengakses konten edukasi daur ulang.
      - Sistem memberikan poin setiap kali pengguna melakukan pelaporan.
      - Admin dapat mengelola data laporan, konten edukasi, dan pengguna.
   b. Kebutuhan Non-Fungsional:
      - Sistem harus memiliki waktu respons maksimal 2 detik per permintaan API.
      - Backend harus mendukung skalabilitas horizontal menggunakan Docker.
      - Sistem harus aman dari serangan umum seperti SQL Injection dan CSRF.
      - Harus tersedia dokumentasi API menggunakan format Swagger/OpenAPI.
      - Sistem harus kompatibel dengan frontend berbasis React atau mobile hybrid.
2. *Deskripsi Aktor Pengguna*
   a. **Pengguna Umum (User)**: Masyarakat umum yang mendaftar untuk mengakses fitur edukasi, mengirim laporan sampah, dan mengikuti sistem gamifikasi.
   b. **Administrator (Admin)**: Pihak pengelola sistem yang bertanggung jawab memverifikasi laporan, mengelola konten edukasi, dan melakukan pengawasan terhadap sistem.

B. Prosedur Penelitian

Proses penelitian meliputi tahapan berikut:

1. *Perancangan Sistem:*
   a. Mengadopsi pendekatan model *Entity Relationship Diagram* (ERD) dalam merancang basis data aplikasi Greenly.
   b. Merancang arsitektur *backend* berbasis REST menggunakan Golang dan *framework* Echo untuk mengelola data dan komunikasi dengan *frontend.*
2. *Pengembangan Backend:*
   a. *Backend* dibangun menggunakan Golang karena keunggulannya dalam performa, keamanan, dan dukungan terhadap konkurensi.
   b. *Framework* Echo digunakan untuk mempermudah routing, middleware, dan data binding.
   c. *Backend* dan komponen terkait dikemas dalam container *Docker* untuk memastikan portabilitas dan efisiensi deployment. *Docker* sangat ringan dan mempunyai mekanisme yang lebih maju karena adanya efektivitas lebih pada *Docker* dalam hal penggunaan sumber daya mesin *host*. Karena dalam proses *deployment, docker* akan menjalankan sebuah *container* menggunakan *base image* dengan metode *file system as a layer* yang berarti *docker* hanya akan menyalin lapisan perubahannya saja untuk dijalankan sebagai duplikasi *container* yang berbeda dengan *base image* yang sama[10].
   d. *Docker* beroperasi pada level OS, maka *Docker* masih bisa berjalan di dalam mesin virtual[11]. Teknologi ini dipilih karena mendukung portabilitas, isolasi lingkungan, serta efisiensi sumber daya melalui pendekatan container yang ringan. Dengan sistem berlapis (layered file system), Docker memungkinkan pembaruan hanya pada bagian yang berubah tanpa membangun ulang seluruh sistem. Keunggulan ini membantu menghindari masalah perbedaan lingkungan dan mempercepat proses deployment.

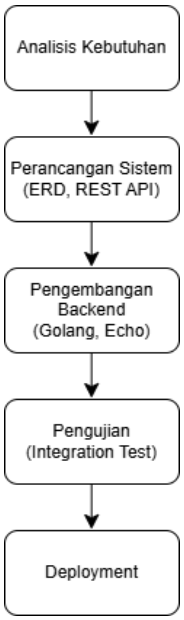
3. Pengujian:

a. ***Integration Testing*:** Dilakukan untuk memastikan fungsi *backend* berjalan sesuai dengan spesifikasi, termasuk fitur edukasi, pelaporan sampah, dan gamifikasi.
b. ***Performance Evaluation*:** *Backend* diuji menggunakan data simulasi untuk mengevaluasi kecepatan respons, penggunaan sumber daya, dan stabilitas sistem.

4. Deployment:

a. Sistem di-*deploy* menggunakan layanan cloud AWS EC2 untuk memastikan ketersediaan dan skalabilitas aplikasi. Digunakan sebagai infrastruktur (kapasitas pemrosesan, memori, dan ruang hardisk) yang menyediakan layanan (*service*) yang dibutuhkan oleh para pengguna[12]. EC2 memberikan kontrol penuh atas konfigurasi sistem, cocok untuk integrasi dengan Docker dan CI/CD. Selain itu, skema biaya berbasis pemakaian menjadikannya efisien untuk skala penelitian.
b. Proses *deployment* didukung oleh GitHub *Actions* untuk *continuous integration/continuous deployment* (CI/CD) dan Nginx sebagai *web server*. *Continuous Integration* dan *Continuous Deployment* (atau *Continuous Delivery*) merupakan praktek pengembangan perangkat lunak yang bertujuan untuk meningkatkan efisiensi dan kualitas pengembangan perangkat lunak dengan cara otomatisasi dan pengujian berkelanjutan[13]. GitHub Actions digunakan untuk otomatisasi CI/CD karena terintegrasi langsung dengan GitHub, mendukung konfigurasi workflow YAML, serta menyediakan notifikasi dan log real-time. Layanan ini gratis untuk proyek skala kecil dan efektif mempercepat proses build, test, dan deploy tanpa alat tambahan.

Berikut adalah ilustrasi diagram alur metode Waterfall yang digunakan:



Gambar 1. *Alur metode*

C. *Pengujian dan Eksperimen*

1. Integration Testing

Pengujian integrasi dilakukan untuk memastikan bahwa seluruh komponen backend saling berinteraksi secara benar dan fitur-fitur yang dikembangkan berjalan sesuai dengan spesifikasi sistem. Validasi dilakukan dengan pendekatan *black-box testing,* menggunakan skenario data simulasi yang merepresentasikan perilaku pengguna nyata. Adapun langkah-langkah validasi meliputi:

a. **Pengujian Endpoint REST API:** Setiap endpoint diuji untuk memastikan menghasilkan response sesuai HTTP status yang diharapkan (200, 201, 400, atau 401), termasuk pengujian autentikasi, otorisasi, dan proteksi middleware.
b. **Fungsi CRUD**: Proses *Create, Read, Update, Delete* untuk entitas seperti artikel edukasi, laporan sampah, dan pengguna diuji dengan input valid dan tidak valid untuk memastikan logika backend berjalan konsisten.

c. **Fitur Gamifikasi**: Validasi dilakukan terhadap sistem pemberian poin dan log aktivitas berdasarkan aksi pengguna (misalnya saat mengirim laporan atau membaca artikel), untuk memastikan bahwa skema reward bekerja sesuai aturan yang ditentukan.
d. **Pengukuran Respons & Stabilitas**: Setiap pengujian mencatat waktu respons dan resource usage (memori/CPU) menggunakan tools log dan observasi manual, guna memastikan backend tetap efisien dan stabil dalam kondisi simulasi beban ringan hingga sedang.
e. **Verifikasi Otomatis & Manual**: Selain pengujian terotomatisasi melalui skrip (misalnya unit test ArticleController), hasil response juga divalidasi secara manual menggunakan tools seperti Postman untuk cross-check struktur dan isi data.

2. Hasil Integration Test

Pengujian dilakukan terhadap berbagai *controller* pada sistem *backend*. Salah satu contoh adalah *ArticleController* yang diuji untuk memastikan proses pengelolaan artikel edukasi berjalan baik. Gambar 6 memperlihatkan hasil pengujian integrasi yang menunjukkan bahwa seluruh *endpoint* berhasil memberikan *response* sesuai harapan. Gambar 7 menampilkan cuplikan kode *unit test* yang digunakan untuk menguji logika masing-masing fungsi secara independen.

3. Deployment dan CI/CD

Sistem backend di-deploy ke layanan cloud AWS EC2 untuk menjamin ketersediaan dan skalabilitas. Proses deployment didukung oleh GitHub Actions untuk automasi CI/CD dan Nginx sebagai reverse proxy. Proses ini meliputi:

a. Pembuatan workflow CI/CD untuk membangun, menguji, dan men-deploy backend secara otomatis.
b. Penanganan error log dan rollback otomatis jika terjadi kegagalan build.
c. Optimasi performa dan penggunaan sumber daya dengan memanfaatkan container Docker.

Gambar 8 hingga 9 menampilkan hasil dari eksekusi GitHub Actions serta alur otomatisasi deployment secara keseluruhan.

4. Dokumentasi API

Untuk memudahkan integrasi dengan frontend, dokumentasi API dibuat menggunakan format Swagger/OpenAPI. Hal ini bertujuan memberikan transparansi dan kemudahan bagi pengembang frontend dalam mengakses berbagai endpoint sistem. Gambar 10 menyajikan dokumentasi untuk endpoint login yang mengilustrasikan struktur request dan response standar.

III. HASIL DAN PEMBAHASAN

A. Perancangan Backend

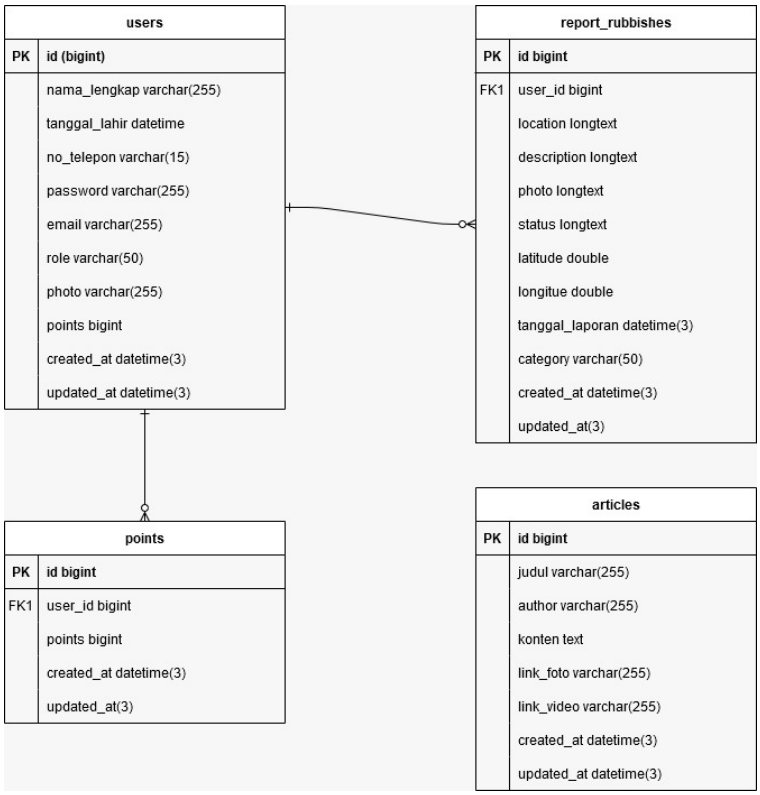
Penelitian ini menghasilkan sebuah backend untuk aplikasi web Greenly yang dibangun menggunakan bahasa pemrograman Golang dan framework Echo. Backend yang dirancang memiliki kapabilitas untuk mendukung fitur utama aplikasi Greenly yaitu, menyediakan panduan edukatif mengenai daur ulang sampah dengan penyajian data yang responsif. Fitur unggulan lainnya mencakup sistem pelaporan sampah secara langsung, termasuk informasi lokasi tempat sampah yang penuh, serta implementasi sistem gamifikasi yang memberikan poin sebagai insentif untuk meningkatkan partisipasi pengguna.

Entity Relationship Diagram(ERD) merupakan diagram yang menggunakan notasi grafis untuk merancang pembuatan database dengan menghubungkan antara data satu dan data yang lain, relasi yang dapat dimiliki oleh ERD ada beberapa macam, yaitu: One to One (Satu anggota entitas dapat berelasi dengan satu anggota entitas lain), One to Many(Satu anggota entitas dapat berelasi dengan beberapa anggota entitas lain), Many to Many(Beberapa anggota entitas dapat berelasi dengan beberapa anggota entitas lain) [14].

Untuk menunjang pengelolaan data yang efisien dan terstruktur, dilakukan perancangan basis data dengan pendekatan Entity Relationship Diagram (ERD). ERD ini merepresentasikan kebutuhan sistem secara konseptual dengan menunjukkan entitas utama dan relasinya. Entitas inti yang dikembangkan meliputi:

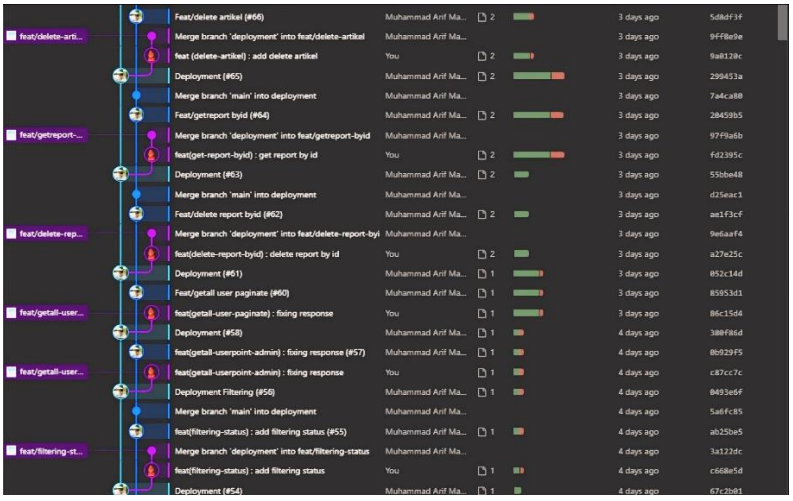
1. **User**: Menyimpan informasi pengguna, seperti nama, email, dan kata sandi. Satu pengguna dapat memiliki banyak laporan dan riwayat poin.
2. **Report**: Menyimpan data pelaporan sampah seperti lokasi, deskripsi, foto, dan status verifikasi. Berelasi dengan satu pengguna.
3. **Article**: Menyediakan konten edukatif yang dapat diakses semua pengguna.
4. **PointLog**: Menyimpan catatan aktivitas pengguna yang menghasilkan poin, seperti pelaporan atau pembacaan artikel.

Hubungan antar entitas ditunjukkan dengan kardinalitas 1:N pada relasi User–Report dan User–PointLog, yang mencerminkan bahwa satu pengguna dapat membuat banyak laporan maupun memperoleh banyak poin.



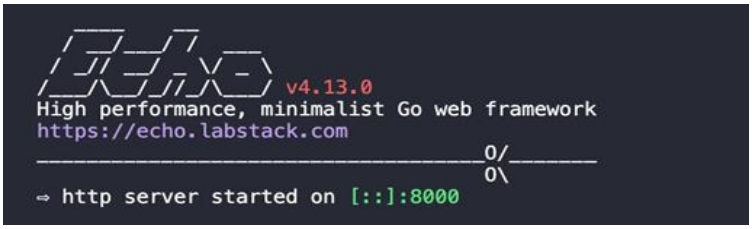
Gambar 2. *Entity Relationship Diagram* (ERD) *Greenly*.

Gambar 3 menggambarkan riwayat *commit* dari *repository* GitHub yang berfungsi untuk menunjukkan alur pengembangan proyek, termasuk proses *branching*, *merge*, dan *deployment*. Proyek ini dikelola dengan baik melalui penggunaan cabang-cabang terpisah untuk mengembangan fitur tertentu secara terpisah, yang kemudian digabungkan ke branch utama atau *deployment* untuk memastikan stabilitas dan keteraturan.



Gambar 3. Riwayat commit repository

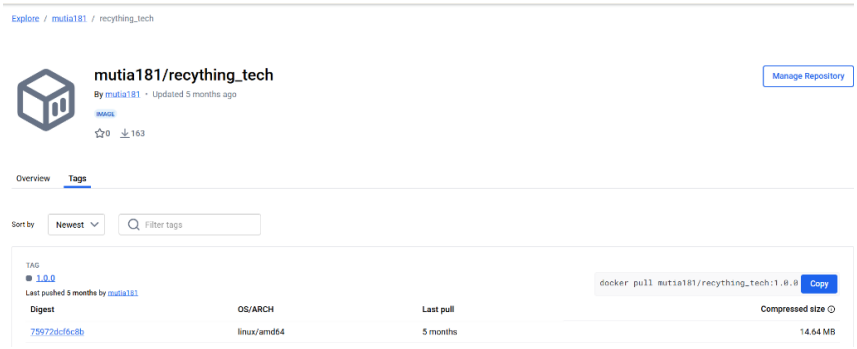
Gambar 4 menunjukan Echo *Framework* untuk mempermudah implementasi arsitektur RESTfull API.



Gambar 4. Pengembangan sistem menggunakan Echo Framework

B. *Implementasi Docker*

Penggunaan *Docker* dalam penelitian ini menawarkan keuntungan signifikan dalam proses *deployment* terutama untuk kebutuhan skala besar, memungkinkan efisiensi dalam pembaruan *container* hanya pada lapisan yang berubah, serta memberikan isolasi lingkungan, yang memastikan kompatibilitas antara pengembangan lokal dan produksi



Gambar 5. Penerapan Docker

C. *Pengujian dan Deployment*

Pengujian dilakukan melalui *Integration Test* untuk memverifikasi bahwa seluruh fitur pada *backend* berfungsi sesuai dengan spesifikasiny. Proses pengujian mencangkup validasi *endpoint API* serta pengujian proses CRUD (*Create, Read, Update, Delete*) terhadap data yang relevan, termasuk pengujian fitur pelaporan sampah dan sistem gamifikasi. Gambar 6 menunjukan hasil *integration test* terhadap salah satu *controller*, yaitu *ArticleController*.

```
PS D:\Studi Independen\CAPSTONE\Backend-Recything> go test -cover ./controllers
ok      Backend-Recything/controllers   (cached)        coverage: 30.8% of statements
PS D:\Studi Independen\CAPSTONE\Backend-Recything>
```

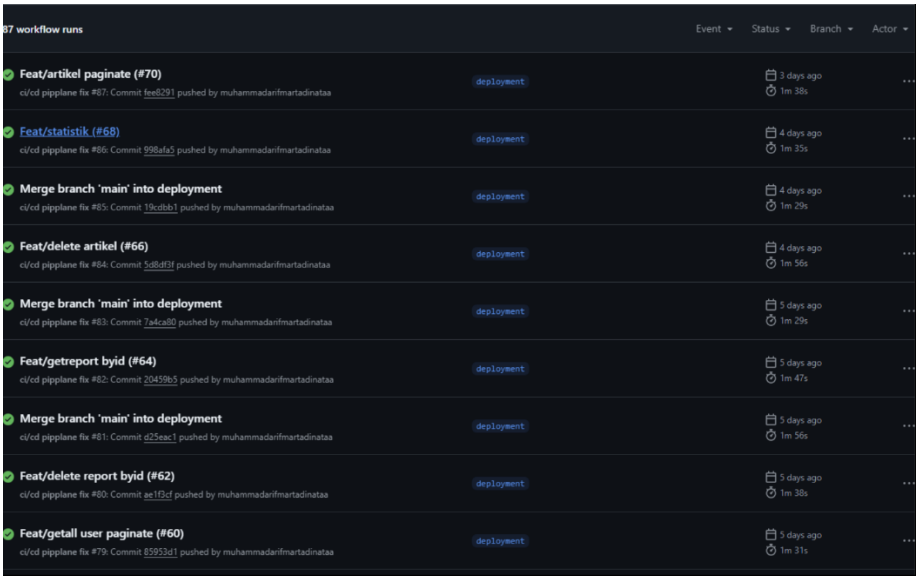
Gambar 6. Hasil Integration Test dari *ArticleController*.

Selanjutnya, Gambar 7 menyajikan cuplikan kode *Unit Test* yang digunakan untuk menguji logika dari ArticleController secara terpisah.

```go
func TestBikinArtikel(t *testing.T) {
    e := echo.New()

    input := models.Article{

        Judul:     "Test Judu 1",
        Author:    "Test Author",
        Konten:    "Ini adalah konten test",
        LinkFoto:  "https://example.com/image.jpg",
        LinkVideo: "https://example.com/video.mp4",
    }

    requestBody, _ := json.Marshal(input)
    req := httptest.NewRequest(http.MethodPost, "/api/v1/articles", bytes.NewReader(requestBody))
    req.Header.Set(echo.HeaderContentType, echo.MIMEApplicationJSON)
    rec := httptest.NewRecorder()
    c := e.NewContext(req, rec)

    // Panggil controller
    if err := BikinArtikel(c); err != nil {
        t.Fatalf("Gagal membuat artikel: %v", err)
    }
}
```

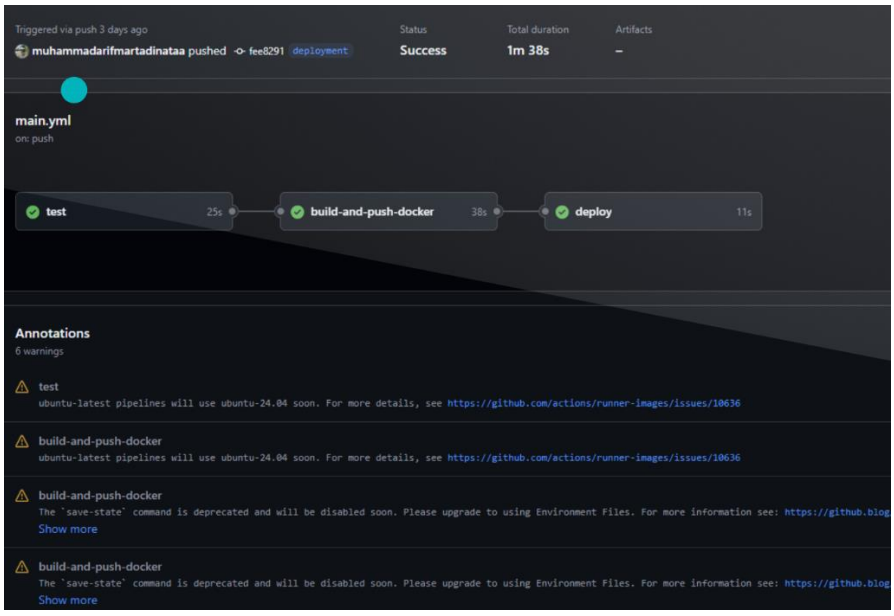
Gambar 7. Cuplikan kode Unit Test pada *ArticleController*

Backend berhasil di-*deploy* menggunakan kombinasi teknologi AWS EC2 sebagai layanan *hosting*, GitHub Actions untuk mendukung proses *Continuous Integration/Continuous Deployment* (CI/CD), dan Nginx sebagai *reverse proxy*. Gambar 8 menampilkan *workflow runs* sebagai hasil dari eksekusi otomatis pada GitHub Actions.



Gambar 8. Hasil Workflow Runs dari proses CI/CD

Sedangkan Gambar 9 memperlihatkan keseluruhan alur CI/CD beserta proses *deployment*-nya.



Gambar 9. Alur CI/CD dan Deployment menggunakan GitHub Actions

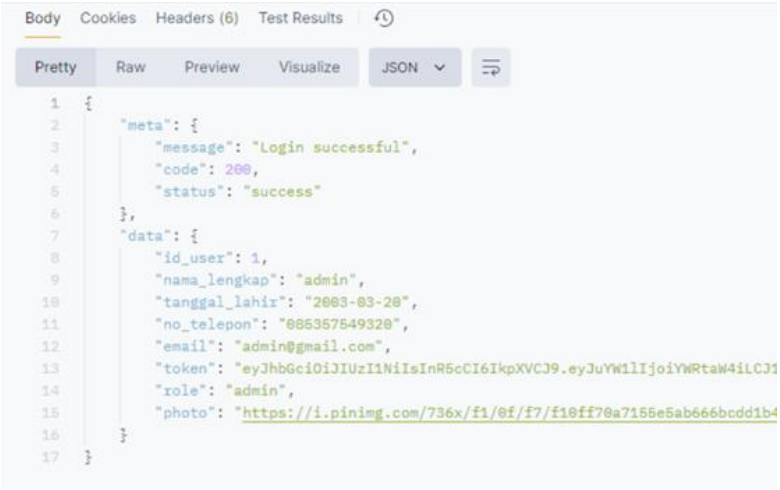
Sistem yang dikembangkan dalam studi kasus ini menggunakan pendekatan arsitektur berbasis Application Programming Interface (API), di mana backend dirancang sebagai penyedia layanan data yang dapat diakses oleh berbagai klien, seperti antarmuka web atau aplikasi mobile. *Application Programming Interface (API)* merupakan antarmuka yang dkembangkan oleh pengembang sistem agar beberapa atau semua fungsi sistem dapat diakses secara terprogram[15]. Setiap fitur utama seperti autentikasi pengguna, pengelolaan data sampah, serta pencatatan transaksi diimplementasikan sebagai *endpoint* API yang dapat diakses secara *stateless* dan terprogram melalui HTTP. Arsitektur ini mendukung integrasi yang fleksibel, serta memudahkan proses pengujian dan deployment otomatis menggunakan layanan seperti GitHub Actions.

Setelah backend berhasil di-deploy, dilakukan pengujian performa menggunakan skenario simulasi beban ringan hingga sedang. Hasil pengujian menunjukkan bahwa waktu respons rata-rata API berada di kisaran **0,6–1,2 detik**, dengan tingkat keberhasilan request mencapai **100%** untuk fitur utama seperti login, pelaporan, dan

pemberian poin. Penggunaan container Docker dan Nginx sebagai reverse proxy turut mempercepat waktu respon dan meningkatkan efisiensi penggunaan sumber daya, terutama dalam hal alokasi memori dan pemrosesan request paralel.

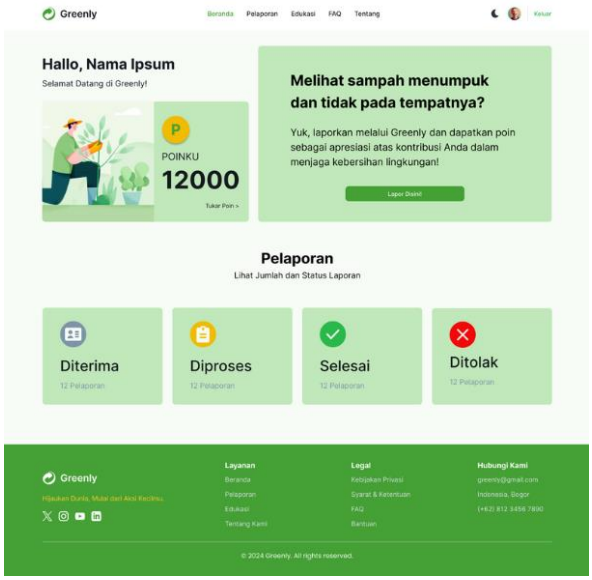
Monitoring dilakukan melalui log server dan hasil output dari workflow GitHub Actions, yang memperlihatkan bahwa proses build dan deploy berjalan stabil, dengan rata-rata waktu build sekitar **40–50 detik**. Hal ini menunjukkan bahwa pipeline CI/CD yang diterapkan mendukung proses iterasi pengembangan yang cepat dan andal.

Gambar 10 menampilkan bagian dari dokumentasi API yang menggambarkan *response* dari sistem pada proses login, yang menjadi penghubung antara frontend dan backend.



Gambar 10. Dokumentasi API pada sistem login

Hasil dari pengintegrasian backend ke dalam website Greenly dapat dilihat pada Gambar 11.



Gambar 11. Hasil integrasi backend pada website Greenly

Perkembangan teknologi yang pesat dapat juga dijadikan solusi untuk mengatasi permasalahan dalam pembelajaran yang memerlukan media untuk mempermudah dan memvisualisaikan objek yang bersifat abstrak dalam proses pembelajaran [16]

IV. SIMPULAN

Penelitian ini berhasil merancang dan mengembangkan *backend* aplikasi web Greenly menggunakan bahasa pemrograman Golang dan framework Echo sebagai solusi atas permasalahan pengelolaan sampah di Indonesia. *Backend* yang dihasilkan mampu mendukung fitur utama seperti panduan edukatif tentang daur ulang

sampah, pelaporan sampah secara langsung (termasuk lokasi tempat sampah yang penuh), serta sistem gamifikasi berbasis pemberian poin guna meningkatkan partisipasi pengguna.

Proses *deployment* menggunakan Docker terbukti memberikan efisiensi yang tinggi dengan isolasi lingkungan yang baik serta kompatibilitas antara lingkungan pengembangan lokal dan produksi. Pengujian melalui *Integration Test* menunjukkan bahwa seluruh fitur *backend* berfungsi dengan baik sesuai dengan yang diharapkan. Teknologi seperti AWS EC2, GitHub Actions, dan Nginx mendukung proses *hosting* dan *Continuous Integration/Continuous Deployment* (CI/CD) secara andal dan terotomatisasi.

Keberhasilan sistem diukur berdasarkan jumlah permintaan (requests) yang berhasil ditangani oleh API tanpa error, serta hasil pengujian integrasi yang menunjukkan 100% keberhasilan dalam mengakses fitur utama. Namun, karena belum dilakukan uji coba langsung kepada masyarakat, efektivitas sosial dari aplikasi ini masih belum dapat disimpulkan secara pasti.

Penelitian ini bertujuan untuk merancang sistem backend yang mampu mendukung pelaporan dan edukasi pengelolaan sampah secara digital. Berdasarkan hasil implementasi dan pengujian, sistem telah berhasil menyediakan API yang stabil dan fungsional untuk fitur-fitur tersebut, sehingga secara aplikatif memenuhi tujuan penelitian dalam menyediakan solusi teknologis terhadap isu pengelolaan sampah di Indonesia

Penelitian ini memberikan kontribusi signifikan dalam menyediakan solusi digital yang edukatif, partisipatif, dan praktis untuk pengelolaan sampah serta dalam meningkatkan kesadaran masyarakat mengenai pentingnya daur ulang.

Saran untuk penelitian selanjutnya adalah melakukan integrasi antara *backend* dan *mobile application* agar jangkauan pengguna menjadi lebih luas dan lebih fleksibel. Selain itu, implementasi teknologi *machine learning* dapat dipertimbangkan untuk mengoptimalkan sistem rekomendasi pengelolaan sampah atau klasifikasi otomatis jenis sampah berdasarkan gambar. Penelitian lanjutan juga disarankan untuk melibatkan uji coba lapangan (field testing) secara langsung dengan masyarakat untuk mengukur dampak sosial dan efektivitas penggunaan aplikasi secara nyata.

.

UCAPAN TERIMAKASIH

Penulis mengucapkan terima kasih yang sebesar-besarnya kepada Universitas Amikom Purwokerto atas dukungan penuh yang diberikan dalam bentuk fasilitas, sumber daya, serta pendanaan yang memungkinkan penelitian ini dapat dilaksanakan dengan baik. Dukungan dari institusi ini sangat berperan penting dalam setiap tahapan pelaksanaan penelitian, mulai dari perencanaan, pengembangan sistem, hingga penyusunan artikel ilmiah. Penelitian ini tidak akan terselesaikan tanpa adanya bantuan dan kepercayaan dari pihak universitas yang senantiasa mendukung kegiatan riset dan inovasi teknologi, khususnya dalam bidang pengembangan aplikasi berbasis teknologi informasi. Penulis berharap hasil dari penelitian ini dapat memberikan kontribusi positif, baik bagi institusi maupun masyarakat secara luas.

DAFTAR PUSTAKA

[1] J. Inovasi, “Sosialisasi sampah organik dan non organik serta pelatihan kreasi sampah,” vol. 4, no. 1, pp. 68–73, 2015.
[2] A. Kahfi, “Tinjauan terhadap pengelolaan sampah,” vol. 4, pp. 12–25.
[3] N. Hidayat and H. Khotimah, “PEMANFAATAN TEKNOLOGI DIGITAL DALAM KEGIATAN,” vol. 02, pp. 10–15, 2019.
[4] R. Annisa, R. A. Ananda, W. E. Sulistiono, T. Informatika, and U. Lampung, “IMPLEMENTASI GOLANG CLEAN ARCHITECTURE PADA PERANCANGAN BACKEND POINT OF SALES,” vol. 12, no. 2, 2024.
[5] A. Hadi, Andik Prakasa; Nugroho, Setiyo Adi; Priyadi, *MENGENAL FRONTEND DEVELOPMENT*. Semarang: Yayasan Prima Agus Teknik Redaksi, 2024.
[6] A. R. Setiadi, “Perancangan UI / UX menggunakan pendekatan HCD (Human-Centered design) pada website Thriftdoor”.
[7] P. S. Rosiana, A. Voutama, and A. A. Ridha, “PERANCANGAN UI / UX SISTEM PEMBELIAN HASIL TANI BERBASIS MOBILE DENGAN METODE DESIGN THINKING,” vol. 11, no. 3, pp. 246–253, 2023.
[8] N. Kadek, D. Sabrina, D. Pramana, and T. M. Kusuma, “Implementation of Golang and ReactJS in the COVID- 19 Vaccination Reservation System,” vol. 5, no. 1, pp. 1–12, 2023.
[9] K. Laurence, “Pengembangan API Perusahaan HiColleagues pada Modul Master Data dengan Metode Monolitik,” vol. 6, pp. 231–240, 2024.
[10] S. Dwiyatno, E. Rakhmat, and O. Gustiawan, “Implementasi virtualisasi server berbasis docker container,” vol. 7, no. 2, pp. 165–175, 2020.
[11] S. Apridayanti, R. A. Saputra, J. T. Informatika, F. Teknik, and U. H. Oleo, “Desain dan implementasi virtualisasi berbasis docker untuk deployment aplkasi web,” vol. 4, no. 2, pp. 37–46, 2018.
[12] G. Ramadani, C. Prabowo, and D. Prayama, “Implementasi Cloud Computing Pada Sistem Penyiraman Tanaman Tomat Otomatis Pada Kebun Tomat,” vol. 2, no. 3, pp. 97–102, 2021.
[13] G. F. Tumewu, “Implementasi layanan cloud computing ci/cd pada aplikasi pendeteksi kepiting soka,” 2023.
[14] J. Ekonomi *et al.*, “Analisis Teknik Entity-Relationship Diagram Dalam Perancangan Database,” vol. 1, no. 2, pp. 143–147, 2022.
[15] B. H. Hasanuddin, Hari Asgar, “RANCANG BANGUN REST API APLIKASI WESHARE SEBAGAI UPAYA MEMPERMUDAH PELAYANAN DONASI KEMANUSIAAN,” *Inform. Teknol. dan Sains*, vol. Vol. 4 No., 2022.
[16] Q. J. Adrian, A. T. Prastowo, M. S. Kuswara, and R. D. Anggita, “Perancangan Visualisasi Elektronik Pencegahan Jantung Koroner Berbasis Teknologi Augmented Reality,” vol. 10, no. 1, pp. 86–94, 2025, doi: 10.30591/jpit.v10i1.8137.