JURNAL INSTEK
INFORMATIKA SAINS DAN TEKNOLOGI
Volume 9 Nomor. 1 April 2024
P –ISSN : 2541-1179, E-ISSN : 2581-1711
Ojs :http://journal.uin-alauddin.ac.id/index.php/instek/index
Email : instek@uin-alauddin.ac.id

IMPLEMENTASI CLEAN ARCHITECTURE PADA PEMBUATAN API MENGGUNAKAN GOLANG

FADEL PAMUNGKAS[1], HARI SETIAJI[2]
[1,2] Program Studi Informatika, Fakultas Teknologi Industri, Universitas Islam Indonesia
Email: [1]18523048@students.uii.ac.id, [2]hari.setiaji@uii.ac.id

ABSTRAK

Penggunaan teknologi saat ini sangat pesat mengakibatkan individu maupun perusahaan dituntut untuk mengikuti perkembangan teknologi yang ada. Dalam pembuatan sistem yang berskala besar dan memiliki lebih dari satu pengembang biasanya membutuhkan penggunaan arsitektur-arsitektur yang telah disepakati agar penulisan kode dapat tertata rapi dan mudah dipahami untuk dapat berkolaborasi dengan pengembang lain dengan mudah. Pemanfaatan *Clean Architecture* ini merupakan salah satu dari beberapa arsitektur lainnya yang sering digunakan dikarenakan memiliki struktur lapisan yang terpisah dengan lapisan lainnya sesuai dengan tugasnya masing-masing.

Kata Kunci: *Clean Architecture, REST API, Golang, HTTP endpoint*

I.PENDAHULUAN

Pada tahun 2021-2022, tingkat penetrasi internet di Indonesia mencapai 77,02% dari total populasi penduduk di Indonesia. Ini berarti sekitar 210 juta dari 272 juta total penduduk di Indonesia sudah menggunakan internet (APJII, 2022). Dengan angka yang tinggi ini membuat perusahan-perusahaan mulai mengadopsi teknologi digital untuk dapat terus berkembang mengikuti zaman. Ini mengakibatkan perusahaan membuka lowongan pekerjaan untuk para pengembang aplikasi maupun website agar dapat memenuhi kebutuhan pasar. Semakin banyak pengguna internet, semakin banyak juga lowongan pekerjaan yang dibutuhkan oleh perusahaan untuk para pengembang aplikasi maupun website.

Saat pembuatan aplikasi yang cukup kompleks dan skala menengah ke atas biasanya membutuhkan jumlah pengembang yang tidak sedikit. Para pengembang dituntut untuk berkolaborasi dengan pengembang lainnya untuk membuat aplikasi yang sesuai dengan yang diharapkan. Oleh karena itu, para pengembang tentu ingin kode yang dibuat dapat dibaca dan dipahami oleh pengembang lain untuk dapat berkolaborasi dengan cepat dan mudah.

JURNAL INSTEK
INFORMATIKA SAINS DAN TEKNOLOGI
Volume 9 Nomor. 1 April 2024
P –ISSN : 2541-1179, E-ISSN : 2581-1711
Ojs :http://journal.uin-alauddin.ac.id/index.php/instek/index
Email : instek@uin-alauddin.ac.id

Dari permasalahan tersebut, muncullah arsitektur-arsitektur diluar sana yang dapat membantu pada pengembang sesuai dengan permasalahan, fungsi, maupun kesepakatan tim. Arsitektur ini disepakati bersama tim maupun pengembang lain untuk membuat aplikasi dengan struktur yang rapi, dan mudah dimengerti oleh pengembang lain.

Salah satu arsitektur yang sangat membantu para pengembang dan dapat digunakan pada semua proyek yaitu Clean Architecture. Tujuan dari penggunaan Clean Architecture ini yaitu agar struktur proyek menjadi terorganisir dengan baik, mudah dipahami oleh para pengembang, dan mudah untuk dirawat jika kode perlu ada perubahan.

II. METODE PENELITIAN

A.Clean architecture

Clean Architecture merupakan konsep perancangan perangkat lunak seperti ditunjukkan pada Gambar 1 yang dipopulerkan oleh Robert C. Martin atau Uncle Bob dalam judul bukunya Clean Architecture: A Craftsman's Guide to Software Structure and Design. Konsep ini membuat pengembangan perangkat lunak untuk berfokus pada algoritma dan struktur data yang dibutuhkan untuk implementasi kebutuhan bisnis, bukan hal-hal yang tidak berhubungan langsung dengan kebutuhan bisnis seperti framework, database, I/O device sehingga hal tersebut dapat ditunda sesuai kebutuhan (Prananta, 2018).

Gambar 1. Clean Architecture (Martin, 2012)

B.Golang

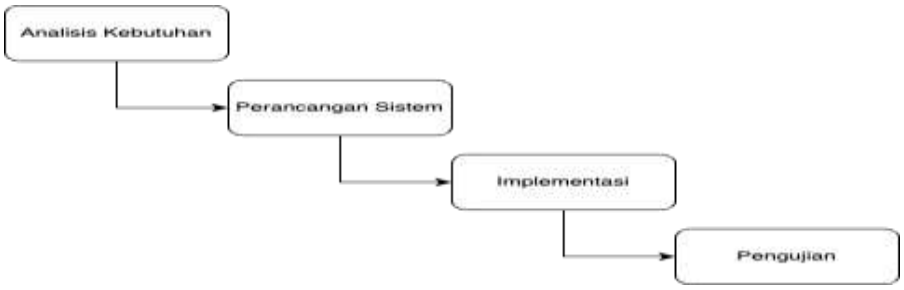
Golang (Go Language) merupakan bahasa pemrograman yang bersifat open source yang dikembangkan di Google oleh Rob Pike, Robert Griesemer, Ken

JURNAL INSTEK
INFORMATIKA SAINS DAN TEKNOLOGI
Volume 9 Nomor. 1 April 2024
P –ISSN : 2541-1179, E-ISSN : 2581-1711
Ojs :http://journal.uin-alauddin.ac.id/index.php/instek/index
Email : instek@uin-alauddin.ac.id

Thompson, dan kontributor lainnya dalam komunitas pengembang open source. Bahasa pemrograman ini memiliki beberapa kelebihan, seperti: (1) Merupakan bahasa pemrograman yang bersifat open source. (12) Mendukung konkurensi yang sangat baik dengan pengaplikasiannya sendiri yang cukup mudah. (3) Memiliki sistem *garbage collection* yang baik dengan memanfaatkan bantuan *built-in garbage collector process* (Goroutines). (5) Bahasa pemrograman yang reliable dan cepat dalam skala besar. (5) Memiliki *syntax* yang simpel dan bersih sehingga tidak mengotori sistem terlalu berlebihan (Tumorang, 2017). Dengan kelebihan tersebut, Golang menjadi pilihan yang tepat untuk digunakan pada aplikasi skala besar dan mendukung *clean architecture* dengan baik.

C.Metode

Penelitian ini menggunakan metode yang ditunjukkan pada gambar 2, seperti: (1) Analisis - Pada tahapan ini melakukan analisis dari sistem yang akan dibuat, seperti fitur yang akan dibuat maupun fungsi yang terdapat pada sistem. (2) Perancangan - Tahapan ini melanjutkan perancangan sistem yang sudah dibuat garis besar sebelumnya pada tahap analisis, seperti menentukan use case sistem. (3) Implementasi - Tahapan ini yaitu mengimplementasikan rancangan yang sudah dibuat sebelumnya untuk menghasilkan sebuah sistem yang sesuai dengan tujuan penelitian ini.



Gambar 2. Metode

III. HASIL DAN PEMBAHASAN

A.Analisis Kebutuhan

Sistem Golang Server yang dikembangkan merupakan sebuah REST API menggunakan bahasa pemrograman Go seperti yang ditunjukkan pada Gambar 3.

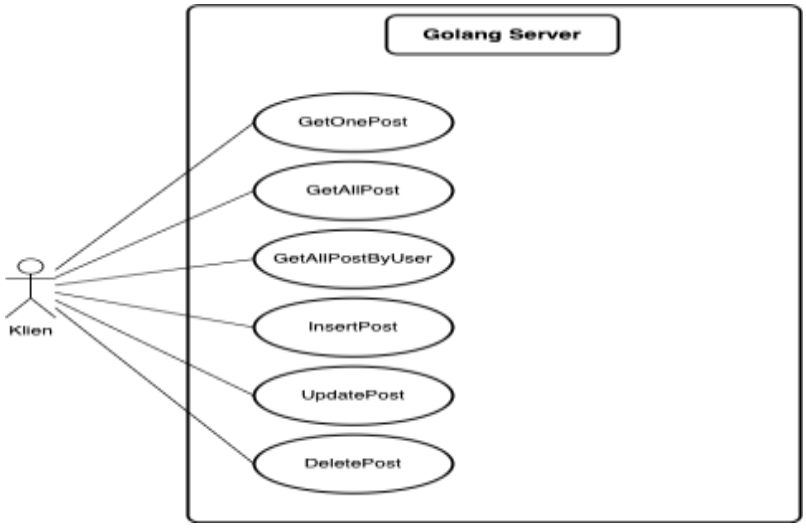
JURNAL INSTEK
INFORMATIKA SAINS DAN TEKNOLOGI
Volume 9 Nomor. 1 April 2024
P –ISSN : 2541-1179, E-ISSN : 2581-1711
Ojs :http://journal.uin-alauddin.ac.id/index.php/instek/index
Email : instek@uin-alauddin.ac.id

Website Snapwork dapat melakukan komunikasi secara langsung ke Golang Server dengan cara mengirimkan *request* ke *endpoint* yang telah disediakan oleh Golang Server, lalu mendapat *response* berupa data yang dapat diproses lebih lanjut oleh *Snapwork Website.*

Gambar 3. Analisis Sistem

B.Perancangan Sistem

Setelah menganalisis sistem secara garis besar, kemudian membuat rancangan diagram *use case* untuk mengetahui semua fungsi yang dibutuhkan pada sistem Golang Server ini. Fungsi sistem ini dapat ditunjukkan pada Gambar 4 dimana pada kasus ini klien merupakan Snapwork Website. Pada penelitian ini, Golang Server memiliki 6 endpoint yang dapat digunakan oleh klien, seperti: (1) *GetAllPost (2) GetAllPostByUser (3) GetOnePost (4) InsertPost (5) UpdatePost (6) DeletePost.*



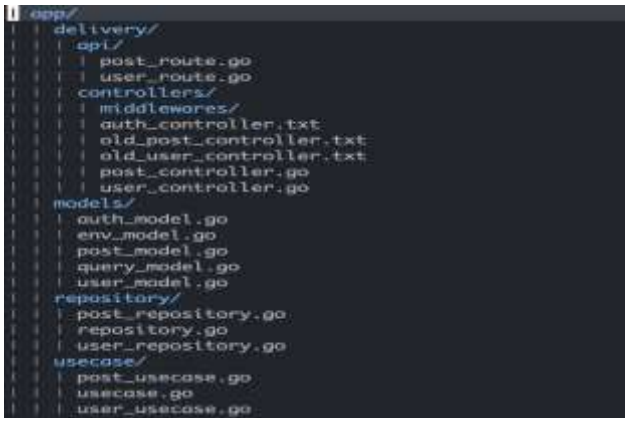
Gambar 4. Perancangan Sistem

C. Implementasi

Setelah perancangan *use case*, kemudian menerapkan *Clean Architecture* pada struktur sistem yang memiliki Lapisan *Delivery*, *Use Case*, dan *Entity*. Lapisan *Delivery* merupakan lapisan terluar yang menerima request dari client

JURNAL INSTEK
INFORMATIKA SAINS DAN TEKNOLOGI
Volume 9 Nomor. 1 April 2024
P –ISSN : 2541-1179, E-ISSN : 2581-1711
Ojs :http://journal.uin-alauddin.ac.id/index.php/instek/index
Email : instek@uin-alauddin.ac.id

untuk diteruskan ke Lapisan *Use Case* dan mengembalikan hasil *response* yang sudah diproses oleh Lapisan *Use Case*. Lapisan *Use Case* yaitu lapisan yang berisi logika bisnis dimana semua pemrosesan data terjadi di lapisan ini. Lapisan *Entity* yaitu lapisan yang berisi struktur sebuah objek yang digunakan oleh lapisan lainnya. Struktur sistem dapat ditunjukkan pada Gambar 5.



Gambar 5. Struktur Sistem

Setiap lapisan memiliki *folder* dan *file*-nya masing-masing, seperti: (1) *folder delivery* mengimplementasikan Lapisan *Delivery* yang berisi *endpoint* untuk menerima *request* dari klien yang ditunjukkan pada Gambar 6.

```
func PostRoute(app *fiber.App, u usecase.PostUsecaseI) {
  c := controllers.NewPostController(app, u)

  api := app.Group("/api")

  api.Get("/posts", c.GetAllPost)                    // Get all posts
  api.Get("/posts/user/:userId", c.GetAllPostByUser) // Get all posts by user id
  api.Get("/post/:postId", c.GetOnePost)             // Get a single post
  api.Post("/post", c.InsertPost)                    // Create a new post
  api.Put("/post", c.UpdatePost)                     // Update an existing post
  api.Delete("/post/:postId", c.DeletePost)          // Delete post
}
```

Gambar 6. Lapisan *Delivery*

(2) *Folder use case* mengimplementasikan Lapisan *Use Case* yang berisi logika bisnis seperti yang ditunjukkan pada Gambar 7.

```
type (
  PostUsecaseI interface {
    GetAllUC(ctx context.Context, query models.Query) (res models.PostResponse, err error)
    GetAllByUserUC(ctx context.Context, id string) (res models.PostResponse, err error)
    GetOneUC(ctx context.Context, id string) (res models.PostResponse, err error)
    InsertUC(ctx context.Context, req models.PostRequest) (res int, err error)
    UpdateUC(ctx context.Context, req models.PostRequest) (res int, err error)
    DeleteUC(ctx context.Context, id string) (res int, err error)
  }
)
```

Gambar 7. Lapisan *Use Case*

JURNAL INSTEK
INFORMATIKA SAINS DAN TEKNOLOGI
Volume 9 Nomor. 1 April 2024
P –ISSN : 2541-1179, E-ISSN : 2581-1711
Ojs :http://journal.uin-alauddin.ac.id/index.php/instek/index
Email : instek@uin-alauddin.ac.id

(3) *Folder models* mengimplementasikan Lapisan *Entity* berisi struktur suatu objek seperti yang ditunjukkan pada Gambar 8.

Gambar 8. Lapisan *Entity*

Dari perancangan yang sudah dibuat, sistem Golang Server berbasis *REST API* ini dapat dijalankan di *terminal* dengan perintah:

go run main.go

Kemudian Golang Server ini dapat diakses melalui protokol HTTP dengan *port* 8080. Untuk mendapatkan data yang diinginkan, dapat menggunakan aplikasi seperti Postman sebagai *API platform* untuk mengirim *request* ke *endpoint* yang disediakan dan mendapatkan *response* yang berupa JSON. Setiap *endpoint* memiliki data *response* yang berbeda-beda sesuai yang disediakan oleh Golang Server. *Response* yang ditunjukkan pada Gambar 9 merupakan hasil *request* dari klien dengan *endpoint* /api/post/{postId}

Gambar 9. *Response* JSON

Sebagai detail dari pengimplementasian *Clean Architecture*, dilakukan pengujian pada beberapa *case* ukuran dokumen untuk mendapatkan hasil *response time* rata-rata.

Tabel 1. Hasil *Response Time*

No.	Kriteria	Jumlah Dokumen	Total Ukuran Dokumen	*average response time*
1.	Kecil	1	430 B	22 ms
2.	Menengah	19	8,6 KB	34 ms
3.	Besar	50	28,79 KB	79 ms

JURNAL INSTEK
INFORMATIKA SAINS DAN TEKNOLOGI
Volume 9 Nomor. 1 April 2024
P –ISSN : 2541-1179, E-ISSN : 2581-1711
Ojs :http://journal.uin-alauddin.ac.id/index.php/instek/index
Email : instek@uin-alauddin.ac.id

IV. KESIMPULAN

Dari penelitian yang telah dilakukan terdapat kesimpulan, seperti: (1) Penggunaan *Clean Architecture* dapat membuat struktur sistem menjadi lebih terorganisir karena setiap lapisan memiliki tugasnya masing-masing. (2) Kode menjadi rapi dan mudah untuk dipahami sehingga jika terjadi perubahan atau perawatan kode menjadi lebih mudah.

DAFTAR PUSTAKA

APJII. (2022, June). *Survei APJII (Asosiasi Penyelenggara Jasa Internet Indonesia) 2022*.

Ivanics, P.́. (2017). An Introduction to Clean Software Architecture.

Kristanto, A. A., Harjoseputro, Y., & Samodra, J. E. (2020). Golang and New Simple Queue Implementation on Third Party Sandbox System Based on REST API. *Jurnal RESTI (Rekayasa Sistem Dan Teknologi Informasi)*, *4*(4), 745 - 750. https://doi.org/10.29207/resti.v4i4.2218

Martin, R. C. (2012, August 13). *The Clean Architecture*. Clean Coder Blog. https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html

Prananta, Y. C. (2018, November 5). *Clean Architecture in Android?. Sedikit cerita tentang bagaimana…*. https://medium.com/style-theory-engineering/android-clean-architecture-using-kotlin-48306644ada7

Ramdan, R. (2020, February 22). *Manfaat Pragmatis Clean Architecture*. https://medium.com/@riza.ramadan/manfaat-pragmatis-clean-architecture-f5164c923ada

Rizkifar, M. A., Insan Purnama, R. F., & Rosmiati, M. (2021). Aplikasi untuk pemasaran dan penjualan produk di baker's corner berbasis android. *7*(5), 1923 - 1931.

Tumorang, I. (2017, Jul 7). *Trying Clean Architecture on Golang*. https://medium.com/easyread/golang-clean-archithecture-efd6d7c43047