Rancang Bangun Modul Komunitas di Aplikasi MyITS Connect Berdasarkan Onion Architecture dengan Paradigma Domain Driven Design

Ersad Ahmad Ishlahuddin, Rizky Januar Akbar, dan Hadziq Fabroyir
Departemen Teknik Informatika, Institut Teknologi Sepuluh Nopember (ITS)
e-mail: rizky@if.its.ac.id

***Abstrak*—Hampir satu dari dua orang di dunia atau 3.8 miliar orang adalah pengguna media sosial, dimana sebanyak 84% didominasi oleh rentang usia 18-29 tahun yang merupakan mahasiswa. Hal tersebut akan memudahkan rencana dari implementasi Modul Komunitas dari myITS. Lembaga PPK-SAC (Pusat Pengembangan Karir) yang menaungi kegiatan konseling, pemetaan, hingga perencanaan karir bagi mahasiswa ITS belum memiliki wadah yang dapat menghubungkan antara mahasiswa, alumni dan stakeholder ITS sebagai sarana pengembangan karir mahasiswa sesuai passion yang dimiliki. Dengan Modul Komunitas myITS Connect pengguna dapat saling berinteraksi atau berdiskusi dengan pengguna lain. Dengan demikian tercipta suatu ekosistem antara mahasiswa, alumni dan stakeholder ITS. Berdasarkan paradigma domain driven design, Modul Komunitas myITS Connect akan dibagi menjadi beberapa modul diantaranya Modul Komunitas, Modul Profil, Modul Post dan Modul Notifikasi. Masing-masing modul akan memiliki interaksi dan dibangun dengan menggunakan onion architecture di atas framework laravel yang dimodifikasi. Paradigma dan arsitektur tersebut dipilih karena keduanya mendukung maintainability yang tinggi. Untuk memastikan aplikasi dapat berjalan dengan baik maka dilakukan ujicoba untuk setiap kasus penggunaan.**

***Kata Kunci*—Domain Driven Design, Komunitas, Media Sosial, MyITS Connect, Onion Architecture**

I. PENDAHULUAN

LEBIH dari 4.5 miliar orang sekarang telah menggunakan internet, dimana pengguna media sosial mencapai 3,8 miliar. Dengan populasi penduduk dunia sekarang yang berjumlah 7.8 miliar menandakan bahwa hampir satu per dua penduduk dunia telah menggunakan media sosial, di mana rata-rata orang menghabiskan 3,7 jam perhari untuk menggunakan aplikasi media sosial dan komunikasi. Kegiatan dalam berselancar di dunia maya yang menjadi proporsi terbesar adalah penggunaan media sosial, di mana media sosial dapat membentuk komunitas yang besar di internet.

Pengguna media sosial saat ini didominasi oleh pengguna dengan rentang usia 18-29 tahun, diikuti dengan pengguna dengan rentang usia 30-49 tahun. Di mana mahasiswa termasuk dalam rentang usia 18-29 tahun. Dengan jumlah pengguna media sosial dan komunitas di dalamnya yang besar pada generasi yang saat ini menjadi mahasiswa, akan memudahkan rencana dari implementasi Modul Komunitas dari myITS untuk meningkatkan jaringan komunikasi mahasiswa di lingkungan ITS. Terlebih lagi saat ini Indonesia juga merupakan negara dengan pengguna *social network* untuk kegiatan terkait bisnis terbesar di dunia, dengan persentase sebesar 65% dari seluruh pengguna media sosial di Indonesia. Ini merupakan jumlah yang besar jika dibandingkan negara seperti Amerika Serikat yang hanya 27% pengguna media sosialnya memanfaatkan sebagai media kegiatan professional.

Networking selama menjadi mahasiswa menjadi kebutuhan utama dalam menopang karir ke depannya. Dengan kekuatan jaringan yang luas, tentunya dapat memudahkan langkah dalam mengembangkan kemampuan pada bidang masing-masing mahasiswa. Dalam mengakomodir pengembangan karir mahasiswa, lembaga PPK-SAC ITS (Pusat Pengembangan Karir) yang menaungi kegiatan konseling, pemetaan, hingga perencanaan karir bagi mahasiswa ITS belum memiliki wadah yang dapat menghubungkan antara mahasiswa, alumni dan *stakeholder* ITS [1]. Hal tersebut dapat menjembatani *passion* mahasiswa maupun pengembangan karir mahasiswa. Maka melalui Modul Komunitas myITS Connect, diharapkan dapat menjadi wadah dalam pengembangan minat mahasiswa melalui komunitas di dalam media sosial yang dirancang untuk saling menghubungkan pengguna untuk saling berinteraksi atau berdiskusi.

Pembuatan Modul Komunitas myITS Connect ini digunakan paradigma *Domain Driven Design* dan *Onion Architecture*. Paradigma *Domain Driven Design* dan *Onion Architecture* digunakan karena keduanya mendukung modularitas aplikasi. Konsep modularitas menjadikan suatu aplikasi yang besar dipecah menjadi modul-modul kecil yang dapat saling berinteraksi melalui antarmuka modul. Dengan memecah aplikasi menjadi modul-modul kecil, kesalahan pada suatu modul tidak memberikan dampak yang besar kepada modul lain sehingga proses perbaikan dan pengembangan dapat lebih mudah dilakukan. Konsep modularitas juga sejalan dengan atribut kualitas *maintainability*. Tingginya nilai atribut kualitas *maintainability* menandakan bahwa aplikasi tersebut mudah untuk diperluas fungsinya atau diperbaiki galatnya. Alasan lain Modul Komunitas myITS Connect dibangun dengan mengedepankan modularitas karena modul-modul lain yang ada pada myITS Connect juga mendukung konsep modularitas. Penerapan konsep modularitas memungkinkan jika suatu saat modul dilepas menjadi suatu aplikasi tersendiri, dan tidak mempengaruhi aplikasi yang lama.

II. TINJAUAN PUSTAKA

A. Onion Architecture

Onion architecture merupakan sebuah pola pendekatan arsitektur perangkat lunak yang dibangun pada beberapa

Tabel 1.
Kebutuhan Fungsional pada Sistem

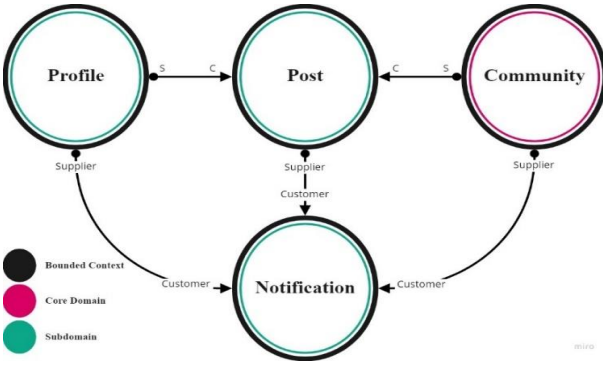
Kode	Kebutuhan Fungsional	Deskripsi
F01	Menangani pengelolaan *post*	Memfasilitasi pembuatan *post* untuk penyampaian informasi, meyunting *post* jika terdapat kesalahan, menghapus *post*, melihat *post* pada beranda pengguna maupun perusahaan, melihat *post* yang ada di komunitas.
F02	Menangani reaksi yang diberikan pada *post*	Memfasilitasi reaksi yang ingin diberikan pengguna pada *post*, reaksi yang diberikan bisa dalam bentuk menyukai *post* atau menyimpan *post*. Reaksi juga dapat diberikan dengan memberikan komentar, dan komentar tersebut dapat dibalas.
F03	Menangani koneksi antar pengguna	Memfasilitasi koneksi yang dikirim maupun diterima oleh pengguna, koneksi yang dikirim dapat dibatalkan, koneksi yang diterima dapat diterima atau ditolak, dan koneksi yang sudah terjadi bisa dihapus.
F04	Menangani pengelolaan komunitas	Memfasilitasi pembuatan komunitas, menyunting komunitas yang telah dibuat, menerima atau menolak anggota yang ingin masuk komunitas, mengeluarkan anggota dari komunitas. Pengguna lain dapat mengirim permintaan untuk bergabung di komunitas dan juga dapat keluar dari komunitas jika sudah bergabung.
F05	Menangani pengelolaan perusahaan	Memfasilitasi pembuatan perusahaan oleh pengguna eksternal, menyunting perusahaan, mengelola *post* perusahaan. Pengguna lain dapat mengikuti perusahaan.

lapisan. Semua kode implementasi hanya boleh mempunyai ketergantungan terhadap lapisan yang lebih dalam [2]. *Onion architecture* menerapkan prinsip yang disebut dengan *dependency inversion* artinya modul yang lebih tinggi tidak boleh bergantung secara langsung terhadap komponen yang lebih rendah. Lapisan pada *onion architecture* terdiri dari *Domain Service*, *Application Service*, *Infrastructure Layer*, dan *Presentation Layer.*

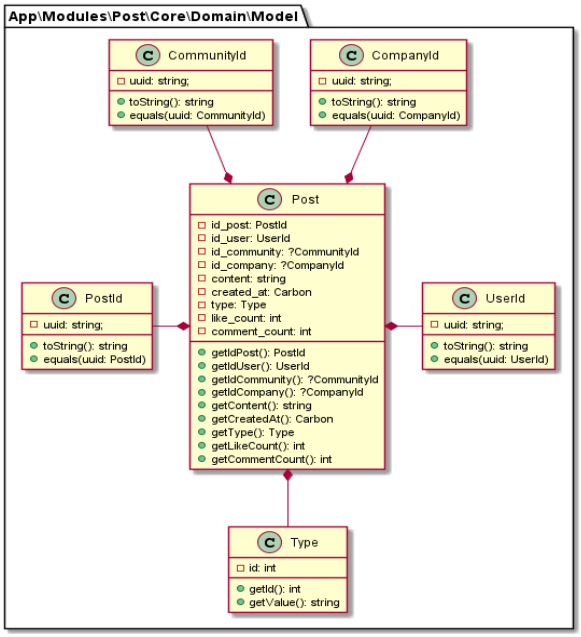
B. Domain Driven Design

Domain Driven Design (DDD) adalah suatu metode pengembangan perangkat lunak yang memfokuskan implementasi sistem pada penyelesaian masalah-masalah yang dihadapi pada domain bisnis dalam bentuk *domain model*. Seluruh hal yang dikembangkan harus berdasar pada cara kerja sebuah proses bisnis yang sesuai dengan *domain model* tersebut. Dalam penerapan paradigma DDD ini, terdapat dua tahap desain yakni *strategic design* dan *tactical design.*

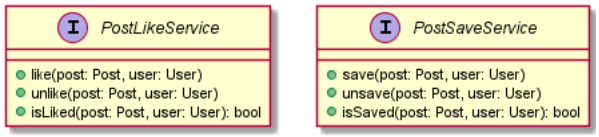
Tujuan dari *strategic design* ini adalah untuk menentukan *ubiquitous language*, *bounded context*, dan *context map*. *Ubiquiotous language* adalah kesepakatan yang dikembangkan dan disetujui oleh seluruh tim. *Bounded context* adalah batasan konseptual yang terdapat dalam dalam suatu domain bisnis, tim, ataupun kode. *Context map* membantu tim dalam memahami keseluruhan proses pengembangan, khususnya dalam mengenali hubungan (*relationship*) antara *bounded context* yang berbeda.



Gambar 1. Context map dari modul komunitas.

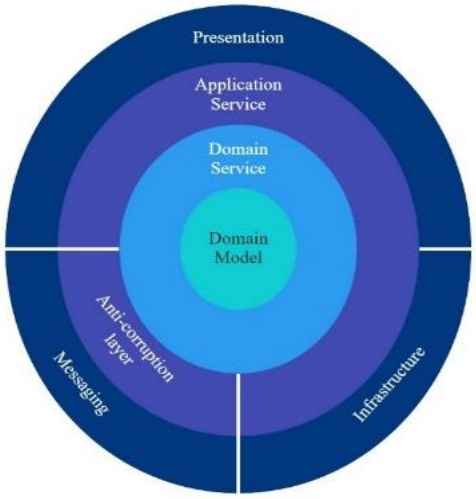


Gambar 2. Diagram kelas aggregate Post modul Post.

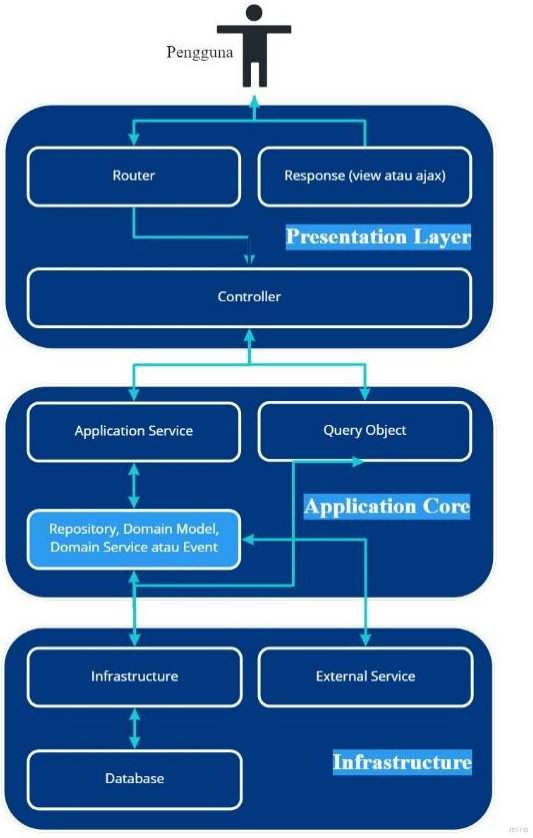


Gambar 3. Diagram kelas domain service modul Post.

Tactical design bertujuan agar domain model yang dibangun memang mencerminkan proses bisnis dari domain aplikasi yang sedang dibangun. Terdapat beberapa komponen yang digunakan dalam melakukan *tactical design* di antaranya adalah *entity*, *value object*, *aggregate*, *service*, *event*, *repository dan exception*. *Entity* adalah suatu objek dalam domain yang pengidentifikasiannya menggunakan suatu *identifier* (identitas), bukan berdasarkan atributnya. *Value object* merupakan objek dalam domain yang merepresentasikan kombinasi dari nilai atribut-atributnya. *Aggregate* merupakan kumpulan objek baik itu *entity* ataupun *value object* yang menjadi satu kesatuan dalam perubahan data. *Domain service* merupakan sebuah perantara yang memuat tingkah laku yang tidak dapat diletakkan dalam salah satu *entity*. *Domain event* merupakan sebuah kejadian. *Application service* bertugas untuk mengkoordinasikan pekerjaan-pekerjaan (*task*) yang dilakukan dalam sebuah kasus penggunaan (*use case*) dengan mengambil ataupun mengubah beberapa *aggregate* dan mengoperasikan *domain service*.



Gambar 6. Rancangan arsitektur.



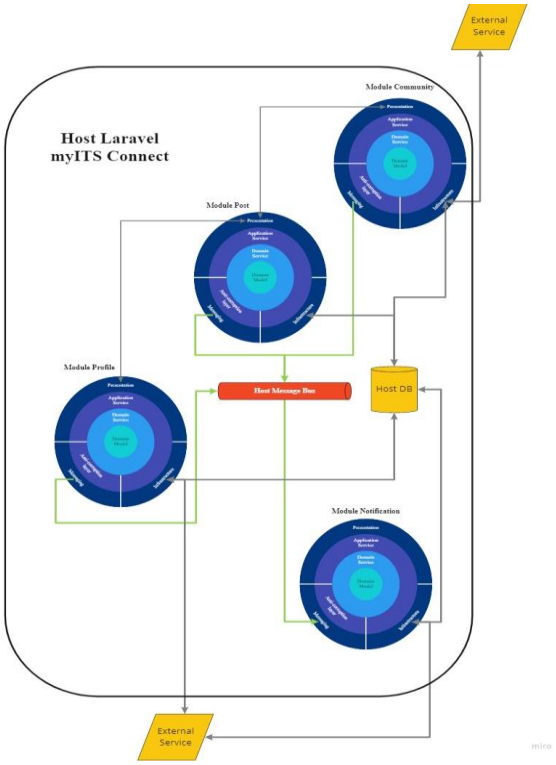
Gambar 7. Rancangan arsitektur di dalam modul.

C. PHP

PHP merupakan bahasa pemrograman *open-source* multifungsi yang dirancang khusus untuk digunakan pada pengembangan aplikasi web.

D. Laravel

Laravel merupakan sebuah kerangka kerja (*framework*) yang dibangun berdasarkan bahasa pemrograman PHP untuk pengembangan aplikasi web. Laravel menyediakan berbagai fitur dan kemudahan untuk membantu pengembang dalam mengembangkan aplikasi web dan menyediakan dokumentasi yang lengkap dan komprehensif sehingga mudah dipelajari.



Gambar 4. Rancangan arsitektur antar modul.

Gambar 5. Rancangan algoritman like post.

D. Black Box Testing

Black Box Testing adalah salah satu metode ujicoba dalam perangkat lunak yang berfokus pada masukan dan keluaran suatu proses, biasanya dilakukan berdasarkan kasus penggunaan atau spesifikasi kebutuhan.

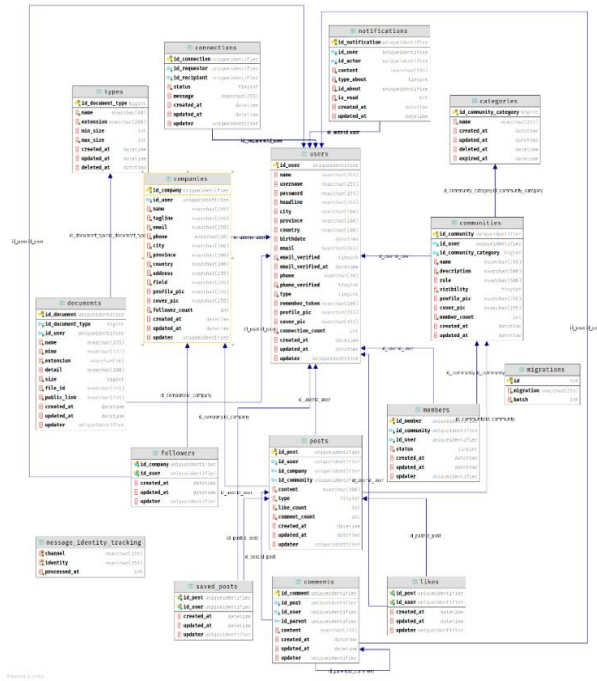
E. Query Object Pattern

Query Object Pattern adalah pola dalam pengembangan perangkat lunak dengan membuat suatu objek untuk melakukan *query* tertentu berdasarkan keluaran yang diharapkan, hal ini dilakukan agar dapat memisahkan operasi pembacaan basis data.

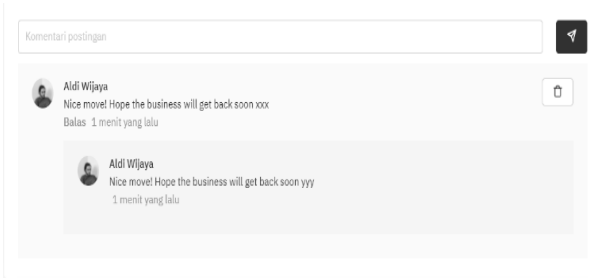
III. ANALISIS DAN PERANCANGAN SISTEM

A. Analisis

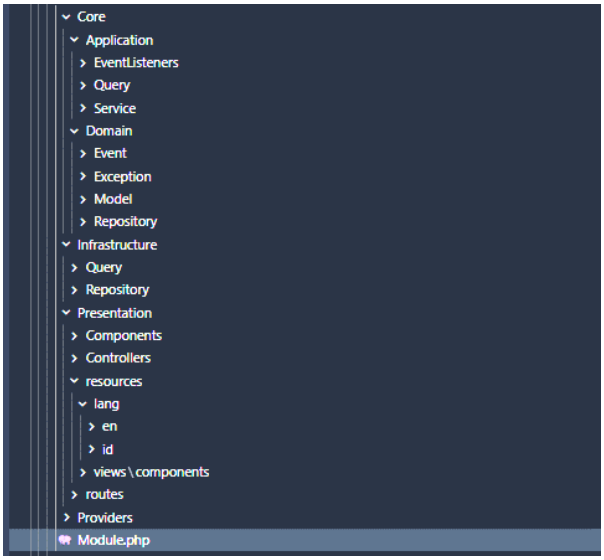
Permasalahan utama yang diangkat dalam penelitian ini adalah bagaimana membuat wadah yang menghubungkan antara mahasiswa, alumni dan *stakeholder* ITS. Sehingga



Gambar 8. Rancangan basis data.



Gambar 9. Rancangan antarmuka buat komentar.



Gambar 10. Implementasi struktur directory.

```
class Post {
  private PostId $id_post;
  private UserId $id_user;
  private ?CommunityId $id_community;
  private ?CompanyId $id_company;
  private string $content;
  private Carbon $created_at;
  private Type $type;
  private int $like_count;
  private int $comment_count;
}
```

Gambar 11. Kode sumber 1 implementasi domain model post.

dapat tercipta interaksi ataupun diskusi antar pengguna. Dengan menggunakan Modul Komunitas myITS Connect, mahasiswa, alumni dan *stakeholder* ITS akan mempunyai wadah untuk saling berinteraksi karena dapat mengakomodir akses dari 2 sisi tersebut (pengguna internal dan pengguna eksternal). Secara garis besar kedua pengguna tersebut dapat melakukan interaksi dalam bentuk yang sama yaitu membuat *post*, memberikan reaksi, memberikan komentar, membalas komentar, mengirim koneksi ke pengguna lain membuat komunitas dan bergabung di komunitas. Perbedaannya terletak pada perusahaan, dimana yang bisa membuat perusahaan hanya akun eksternal, kemudian pengguna lain dapat mengikuti perusahaan tersebut untuk mengetahui *post* yang dibuat. Kebutuhan fungsional pada system tertera pada Tabel 1.

B. Strategic Design

Strategic design dilakukan agar mendapatkan *ubiquitous language* dan *bounded context* dari perangkat lunak yang akan dibuat. Cara untuk mendapatkan hal tersebut adalah dengan melakukan diskusi dengan *domain expert* yang dalam penelitian adalah pihak PPK-SAC ITS. Terdapat kesepakatan konvensi di mana pada untuk implementasinya *ubiquitous language* menggunakan bahasa inggris, karena pada komunitas terdapat istilah yang memang lebih familiar jika disebutkan dalam bahasa inggris seperti *Post*, *Like* dan *Comment*. *Domain* pada Modul Community myITS Connect dapat dibagi menjadi beberapa *bounded context*, yaitu *Community*, *Profile*, *Post*, dan *Notification*. Ilustrasi mengenai *context map* pada aplikasi dilihat pada Gambar 1.

Bounded context yang berperan sebagai *upstream* memberikan informasi yang diperlukan kepada *bounded context* yang berperan sebagai *downstream*. Berikut adalah simplifikasi *ubiquitous language* dalam bentuk *user story*. Modul *Community* sebagai pengguna saya dapat membuat suatu komunitas, sehingga bisa mengumpulkan orang-orang yang mempunyai ketertarikan atau tujuan sama, sebagai pengguna saya dapat bergabung dalam suatu komunitas, sehingga bisa berkumpul dan berdiskusi anggota lain, sebagai anggota komunitas saya dapat membuat *post* di grup, sehingga bisa memberikan informasi atau diskusi.

Modul *Profile* sebagai pengguna saya dapat mengirimkan koneksi pada pengguna lain, sehingga bisa terhubung dengan pengguna lain. Sebagai pengguna eksternal saya dapat membuat perusahaan, sehingga dapat mepresentasikan perusahaan saya. Sebagai pengguna saya dapat mengikuti perusahaan yang diminati, sehingga bisa mendapat informasi terbaru mengenai perusahaan tersebut. Sebagai pengguna atau perusahaan saya dapat membuat *post* di beranda atau halaman profil, sehingga bisa memberikan informasi atau diskusi.

Modul *Post* sebagai pengguna saya dapat memberikan *like* pada *post* yang dibuat, sehingga terdapat interaksi. Sebagai pengguna saya dapat memberikan komentar pada *post* yang dibuat, sehingga bisa memberikan umpan balik atau diskusi.

Tabel 2.
Penjelasan rancangan antarmuka buat komentar

No	Nama Atribut Antarmuka	Jenis	I/O	Keterangan
1	Input komentar	Input field	Input: string	Memasukkan komentar yang inginkan pada post.
2	Button kirim	Button	Input: click	Menyimpan data komentar yang telah dibuat.
3	Button hapus	Button	Input: click	Menghapus data komentar yang dipilih.
4	Section komentar	Section	-	Menampilkan daftar komentar yang ada pada post.

Tabel 3.
Tabel lingkungan uji coba

Spesifikasi	Deskripsi
Jenis Perangkat	Laptop
CPU	Intel Core i7-7700HQ
RAM	16GB
Sistem Operasi	Windows 10

Sebagai pengguna saya dapat menyimpan *post* yang diinginkan, sehingga dapat membacanya lagi di lain waktu.

Modul *Notification* sebagai pengguna saya dapat menerima notifikasi dari *post* yang saya buat, sehingga dapat mengetahui pembaruan dari *post* tersebut.

C. Tactical Design

Tactical design dilakukan dengan fokus pada *context map* dan *bounded context* yang telah dibuat. Untuk mengurangi kompleksitas pada *domain model*, maka kasus penggunaan akan dibagi menjadi 2 jenis yaitu *command* dan *query*. Beberapa modul akan menggunakan pendekatan *disconnected domain model* untuk membuat sebuah *aggregate*, dimana suatu *aggregate* tidak secara langsung berkomposisi dengan *aggregate* lain akan tetapi menggunakan referensi atau identitas *aggreagte* tersebut. Hal ini dilakukan karena *command* dan *query* yang telah dipisah, mempercepat proses penulisan pada basis data dan memperkecil aggregate. *Tactical design* yang digunakan sebagai contoh adalah pada Modul *Post*.

Pada Gambar 2 *Post* hanya berkomposisi dengan referensi dari masing-masing *User*, *Community* dan *Company* karena proses pengambilan data *Post* yang dimiliki oleh *Entity* tersebut dilakukan oleh *query object*, sehingga *Post* belum membutuhkan komposisi dengan *aggregate* tersebut. Dan untuk *aggregate Like* dan *Save* jika ditambahkan maka hanya akan memperbesar *aggregate* yang sebenarnya tidak dibutuhkan, sehingga untuk *Like* dan *Save* akan digantikan dengan *Domain Service* seperti yang ditunjukkan pada Gambar 3.

D. Perancangan Arsitektur

Setiap modul akan menggunakan *onion architecture* yang di dalamnya terdapat beberapa lapisan yang memiliki tanggung jawab masing-masing. Detail mengenai arsitektur yang digunakan akan diilustrasikan pada Gambar 4. Kemudian untuk interaksi yang terjadi pada setiap modul akan diilustrasikan pada Gambar 5.

Setiap modul yang ada tidak boleh saling akses secara langsung ataupun memanggil komponen yang dimiliki modul lain. Oleh karena itu diperlukan mekanisme komunikasi antar modul yang tepat. Detail mengenai interaksi dan komunikasi antar modul akan diilustrasikan pada Gambar 6.

```
interface PostLikeService {
  public function like(Post $post, User $user);
  public function unlike(Post $post, User $user);
  public function isLiked(Post $post, User $user): bool;
}
```

Gambar 12. Kode sumber 2 implementasi domain service postlike.

```
interface PostSaveService {
  public function save(Post $post, User $user);
  public function unsave(Post $post, User $user);
  public function isSaved(Post $post, User $user): bool;
}
```

Gambar 13. Kode sumber 3 implementasi domain service postsave.

Untuk melakukan interaksi antar modul sekaligus mengurangi ketergantungan antar modul, setiap modul dapat mengakses sebuah *message bus* dengan mekanisme *publish-subscribe*. Untuk menerjemahkan pesan dari modul lain dengan *ubiquitous language* yang berbeda, digunakan sebuah *anti-corruption layer*.

E. Perancangan Algoritma

Pada subbab ini akan menjelaskan mengenai rancangan dari salah satu algoritma yang akan digunakan, dan yang digunakan untuk contoh adalah *Like Post*, karena operasi ini akan dilakukan hampir di semua modul. Rancangan algoritma akan diilustrasikan dalam bentuk diagram alur yang ditunjukkan pada Gambar 7.

F. Perancangan Basis Data

Pada subbab ini akan menjelaskan mengenai rancangan basis data yang akan digunakan. Rancangan basis data diilustrasikan dalam bentuk PDM (*Physical Data Model*) yang ditunjukkan pada Gambar 8.

G. Perancangan Antarmuka

Pada subbab ini akan menjelaskan mengenai rancangan antarmuka yang akan digunakan. Rancangan antarmuka akan diilustrasikan dengan merepsentasikan setiap halaman pada aplikasi. Kasus penggunaan saat membuat komentar akan dijadikan sebagai contoh. Rancangan dan penjelasan antarmuka diilustrasikan dalam Gambar 9 dan Tabel 2.

IV. IMPLEMENTASI

A. Implementasi Arsitektur

Implementasi arsitektur pada penelitian menggunakan *onion architecture* dengan menggunakan *package* laravel-modular yang juga dikembangkan selama pengerjaan penelitian ini. Dengan menggunakan *package* ini maka secara otomatis dapat membuat sebuah modul dengan menggunakan *skeleton directory* dalam bentuk *onion architecture* yang ditunjukkan pada Gambar 10.

Lapisan *application core* pada *onion architecture* diletakkan pada folder Core di setiap modul. Pada folder Core/Application terdapat *event listeners*, *query object* dan *application service*. Kemudian pada folder Core/Domain terdapat *domain model*, *domain service*, *event* dan *exception*.

```
class DeleteCommentService
{
  // ...
  // Repository Constructor
  // ...
  public function execute(DeleteCommentRequest $request)
  {
    $comment = $this->comment_repo-
>find(new CommentId($request->getIdComment()));

    if (!$comment)
      throw new NotFoundException("Comment");

    if (!$comment->getIdUser()-
>equals(new UserId($request->getIdUser())))
      throw new NotAllowedDeleteData("Comment");

    $post = $this->post_repo->find($comment-
>getIdPost());

    if (!$post)
      throw new NotFoundException("Post");

    $user = $this->user_repo->find(new UserId($request-
>getIdUser()));

    if (!$user)
      throw new NotFoundException("User");

    if (!$comment->hasReply()) {
      $this->comment_repo->delete($comment);
      Event::publish(new CommentDeleted($post, $user));
    }
  }
}
```

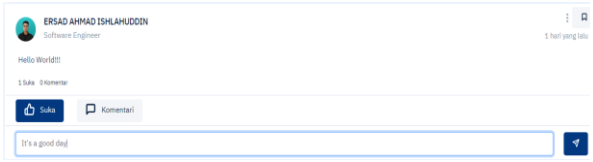
Gambar 14. Kode sumber 4 implementasi delete comment service.

Pada folder Infrastructure terdapat implementasi dari *interface* yang ada pada *application core*. Untuk lapisan antarmuka, struktur *directory* pada aplikasi ini serupa dengan struktur bawaan dari Laravel. Seluruh *file* yang berkaitan dengan antarmuka diletakkan dalam folder Presentation.

B. Implementasi Pengetahuan Domain

Pada subbab ini akan dijelaskan bagaimana mengimplementasikan pengetahuan domain yang digambarkan dengan acuan *ubiquitous language* dalam sebuah kode. Beberapa kelas yang berhubungan dengan reaksi yang diberikan pada *post* akan digunakan sebagai contoh. *Post* memiliki peran untuk menangani reaksi yang diberikan pada sebuah Post, reaksi tersebut adalah *like*, *comment*, dan *save*. Akan tetapi pada implementasi pengetahuan domain ada beberapa penyesuaian yang dilakukan. *Comment* dipisah menjadi *aggregate* sendiri agar *aggregate Post* tidak menjadi terlalu besar, begitu juga dengan *like* dan *save*. Seperti yang ditunjukkan pada Gambar 11. Kode sumber 1 yang tertera pada Gambar 11.

Ketika *like* dan *save* dipisah, *aggregate* tersebut menjadi terlalu kecil. Oleh karena itu dibuat *domain service* masing-masing untuk menangani *like* dan *save* yang ditunjukkan pada Gambar 12. Kode sumber 2 yang tertera pada Gambar 12. dan Gambar 13. Kode sumber 3 yang tertera pada Gambar 13. Hal ini juga dilakukan untuk memperjelas dan mempermudah



Gambar 115. Implementasi antarmuka buat komentar.

Tabel 4.
Rekapitulasi pengujian kasus penggunaan modul community

Kode	Kasus Penggunaan	Terpenuhi	
		Ya	Tidak
UJ-001	Membuat komunitas	✓	
UJ-002	Melihat detail komunitas	✓	
UJ-003	Menyunting komunitas	✓	
UJ-004	Mengajukan permintaan bergubung di komunitas	✓	
UJ-005	Membatalkan permintaan bergabung di komunitas	✓	
UJ-006	Menerima permintaan bergabung	✓	
UJ-007	Menolak permintaan bergabung	✓	
UJ-008	Keluar dari komunitas	✓	
UJ-009	Membuat post	✓	
UJ-010	Menyunting post	✓	
UJ-011	Menghapus post	✓	

Tabel 5.
Rekapitulasi pengujian kasus penggunaan modul profile

Kode	Kasus Penggunaan	Terpenuhi	
		Ya	Tidak
UJ-012	Melihat profil pengguna	✓	
UJ-013	Menyunting profil pengguna	✓	
UJ-014	Mengirim koneksi ke pengguna lain	✓	
UJ-015	Membatalkan permintaan koneksi	✓	
UJ-016	Menerima koneksi	✓	
UJ-017	Menolak koneksi	✓	
UJ-018	Menghapus koneksi	✓	
UJ-019	Membuat perusahaan	✓	
UJ-020	Mengikuti perusahaan	✓	
UJ-021	Batal mengikuti perusahaan	✓	
UJ-022	Menyunting perusahaan	✓	
UJ-023	Menghapus perusahaan	✓	

alur program, ketika ada *service* yang membutuhkan reaksi tersebut maka dapat melakukan pemanggilan langsung.

C. Implementasi Kasus Penggunaan

Pada subbab ini akan dijelaskan bagaimana mengimplementasikan kasus penggunaan dengan acuan *ubiquitous language* dalam sebuah kode. Kasus penggunaan hapus komentar akan digunakan sebagai contoh untuk implementasi. Detail implementasi akan dijelaskan pada Kode sumber 4 yang tertera pada Gambar 14.

Proses hapus komentar dimulai dengan sistem memeriksa apakah komentar ada di dalam sistem. Jika komentar ada, sistem akan memeriksa apakah pengguna berhak untuk menghapus komentar. Jika kedua syarat terpenuhi sistem akan melakukan pengecekan apakah komentar memiliki balasasn. Terakhir, sistem akan menghapus data komentar melalui *repository*. Setelah operasi berhasil dilakukan maka sistem akan mengirim *event*. Langkah – langkah tersebut tergambar dengan jelas pada implementasi sehingga dokumentasi formal dapat dikurangi.

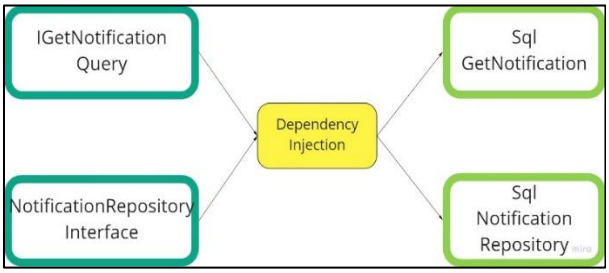
D. Implementasi Antarmuka

Pada subbab ini akan dijelaskan bagaimana implementasi antarmuka berdasarkan rancangan. Implementasi antarmuka pengguna berbasis web menggunakan HTML, CSS,

JavaScript, dan dibangun menggunakan bantuan *framework* Laravel. Implementasi antarmuka saat membuat komentar akan dijadikan sebagai contoh dan ditunjukkan pada Gambar

Tabel 6.
Rekapitulasi Pengujian Kasus Penggunaan Modul Post

Kode	Kasus Penggunaan	Terpenuhi	
		Ya	Tidak
UJ-024	Menyukai post	✓	
UJ-025	Batal menyukai post	✓	
UJ-026	Melihat pengguna yang menyukai post	✓	
UJ-027	Menyimpan post	✓	
UJ-028	Batal menyimpan post	✓	
UJ-029	Memberikan komentar pada post	✓	
UJ-030	Memberikan balasan pada komentar	✓	
UJ-031	Menghapus komentar	✓	



Gambar 16. Skenario modularitas awal.

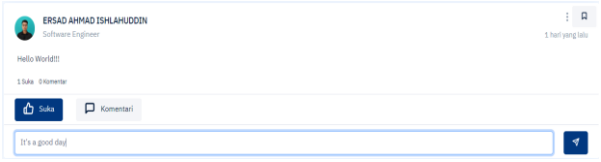
15.

V. UJI COBA DAN EVALUASI

Pada bab ini dijelaskan tentang uji coba dan evaluasi dari implementasi yang telah dilakukan pada penelitian ini, uji coba dilakukan dengan skenario berdasarkan kasus penggunaan dan skenario modularitas sistem.

Tabel 3 menunjukkan lingkungan pengujian yang digunakan.

Skenario ujicoba pertama dilakukan berdasarkan kasus penggunaan dengan menggunakan metode *Blackbox Testing*, daftar rekapitulasi pengujian yang dilakukan akan dibagi berdasarkan kasus penggunaan pada setiap modul.



Gambar 15. Implementasi antarmuka buat komentar.

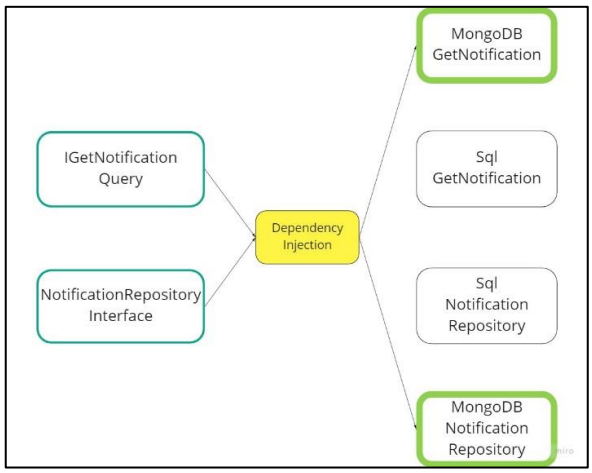
Tabel 4 menunjukkan rekapitulasi Modul *Community*,

Tabel 5 menunjukkan rekapitulasi Modul *Profile* dan Tabel 6 menunjukkan rekapitulasi Modul *Post*.

Skenario ujicoba dilakukan untuk menguji modularitas, pengujian dilakukan dengan mengubah basis data yang digunakan pada Modul *Notification*. Sebelumnya basis data yang digunakan adalah SQL Server, yang pada ujicoba ini akan diubah menjadi MongoDB untuk memastikan bahwa aplikasi bersifat modular.

Gambar menunjukkan alur awal sistem, di mana penyimpanan basis data menggunakan SQL Server. Dengan menggunakan *Dependency Injection*, interface tersebut akan memanggil kelas konkritnya yang menggunakan SQL Server sebagai basis datanya.

Gambar 1 menunjukkan implementasi kelas konkrit baru yang menggunakan MongoDB sebagai basis datanya. Setelah kelas konkrit baru terbuat, maka Dependency Injection yang ada perlu sedikit diubah agar interface tersebut memanggil kelas konkrit yang menggunakan MongoDB sebagai basis



Gambar 127. Implementasi kelas konkrit baru.

datanya yang ditunjukkan pada.

Berdasarkan ujicoba kasus penggunaan, Modul Komunitas myITS Connect dapat menjalankan semua fungsionalitas. Dan dengan menggunakan *Onion Architecture* modularitas sistem menjadi lebih tinggi, yang dibuktikan dengan kemudahan ketika ingin mengubah kelas konkrit dari suatu *interface*.

VI. KESIMPULAN DAN SARAN

Berdasarkan penjabaran di subbab-subbab sebelumnya dari hasil uji coba yang dilakukan, dapat disimpulkan beberapa poin. Proses perancangan menggunakan paradigma *domain driven design* dilakukan berdasarkan proses bisnis hasil diskusi dengan domain *expert*. Hasil diskusi digunakan sebagai *strategic design* dan dilanjutkan dengan *tactical design*. Proses implementasi *onion architecture* dengan *framework laravel* berdasarkan rancangan proses bisnis berhasil dilakukan. Implementasi dilakukan dengan modifikasi *laravel* menggunakan *package laravel modular*.

Beberapa saran yang dapat diberikan kepada penulis terkait penelitian ini adalah meliputi sebagai berikut modul Komunitas merupakan salah satu Modul di myITS Connect, sehingga diperlukan integrasi dengan Modul *Job* agar sistem dapat berjalan secara utuh. Fungsionalitas dari sisi adminitrator untuk verifikasi tidak dibahas di penelitian ini, disarankan untuk mengembangkan Modul Administrator. Data sistem berasal dari masukan internal sistem, akan lebih baik jika terdapat pertukaran data dengan sistem yang sudah ada di ITS.

DAFTAR PUSTAKA

[1] ITS, "ITS Career & Student Entrepreneurship Center," *Selayang Pandang*, 2021. https://sac.its.ac.id/content/view?id=123&t=profil-career-center.

[2] M. E. Khalil, K. Ghani, and W. Khalil, "Onion architecture: a new

approach for xaas (every-thing-as-a service) based virtual collaborations," in *13th Learning and Technology Conference*, 2016, pp. 1--7.