Rancang Bangun Sistem MyITS Dorm Menggunakan Metode *Domain Driven Design* dan *Onion Architecture*

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

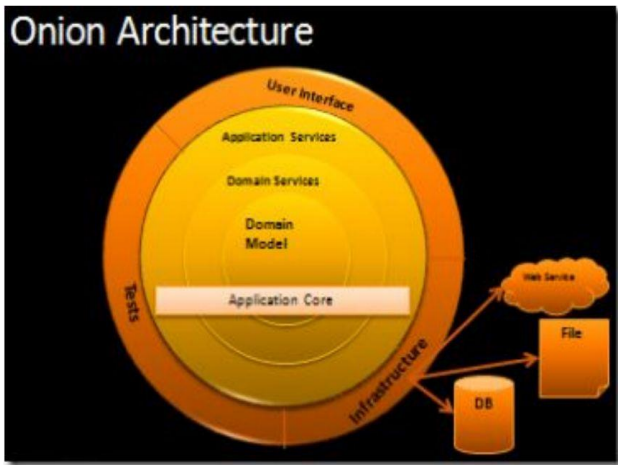
***Abstrak*—MyITS Marketplace merupakan sebuah platform yang berfungsi sebagai integrasi dari banyaknya layanan yang tersedia di Institut Teknologi Sepuluh Nopember. Selain sebagai integrasi dari banyaknya layanan, sistem ini juga akan membantu pihak ITS dalam mempromosikan layanan ITS di luar sivitas akademik ITS. Salah satu layanan yang terdapat pada sistem myITS Marketplace adalah layanan Asrama Mahasiswa. Layanan Asrama Mahasiswa menawarkan fasilitas hunian sewa yang disediakan bagi mahasiswa untuk tinggal selama masa studi di ITS Surabaya. Pengelolaan layanan Asrama Mahasiswa tersebut akan dikelola oleh sebuah sistem yang dirancang pada penelitian ini, bernama myITS Dorm. MyITS Dorm mempunyai tiga fungsional utama, yaitu pemesanan kamar oleh mahasiswa ITS, pengelolaan fasilitas hunian oleh pengurus asrama ITS, dan pengelolaan kontrak antara mahasiswa dan pengurus asrama ITS. Tiap fungsional tersebut akan mempunyai sebuah proses bisnis yang cukup rumit, sehingga perancangan sistem myITS Dorm akan menggunakan metode *Domain Driven Design*, dimana melakukan analisis terhadap proses bisnisnya. Melalui rancangan yang telah dibuat, sistem akan diimplementasikan menggunakan *Onion Architecture*. Penggunaan *Onion Architecture* pada implementasi sistem ini akan membagi sistem myITS Dorm menjadi beberapa layer untuk mengurangi kompleksitas dari sistem myITS Dorm. Sistem myITS Dorm yang telah dibuat pada penelitian ini akan dievaluasi untuk setiap fungsionalnya, terutama fungsional yang berkaitan dengan fungsional utamanya. Hasil dari penelitian ini adalah menghasilkan sebuah sistem myITS Dorm yang berjalan dengan baik sesuai dengan proses bisnisnya, terstruktur dan terkelola dengan baik, sehingga dalam pemeliharaan aplikasi maupun dalam pengembangan aplikasi lebih lanjut, dapat dilakukan dengan lebih efektif dan efisien.**

Kata Kunci*—*Domain-Driven Design, Onion Architecture.

I. PENDAHULUAN

BANYAK sekali layanan yang disediakan oleh Institut Teknologi Sepuluh Nopember, seperti layanan Asrama Mahasiswa, Laboratorium, ITS Training Center, UPT Bahasa dan Budaya, Medical Center, Fasilitas Olahraga, Perpustakaan dan ITS Press. Layanan-layanan tersebut merupakan layanan yang disediakan dan ditujukan untuk sivitas akademik ITS ataupun untuk kalangan masyarakat umum. Banyaknya layanan tersebut masih dikelola secara terpisah, dan tidak terintegrasi satu sama lain, sehingga banyak sivitas akademik ITS dan masyarakat umum yang masih kesulitan dalam mengakses layanan-layanan tersebut. Melihat masalah tersebut, muncul sebuah ide untuk membuat sebuah sistem yang mengelola layanan yang disediakan oleh Institut Teknologi Sepuluh Nopember secara terintegrasi, yaitu sistem myITS Marketplace.

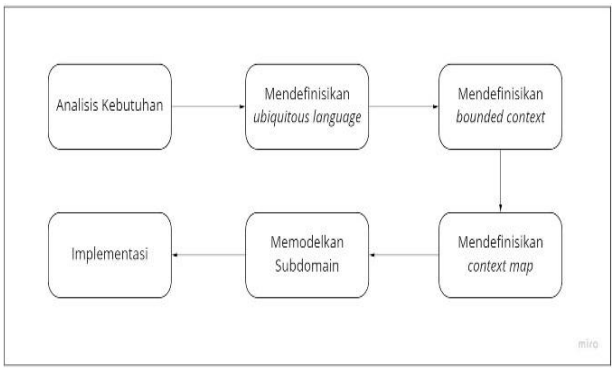
Salah satu layanan yang terdapat pada sistem myITS Marketplace adalah layanan Asrama Mahasiswa, dimana layanan tersebut ditujukan kepada mahasiswa, terutama mahasiswa yang berasal dari luar kota Surabaya. Layanan yang ditawarkan adalah fasilitas hunian sewa yang disediakan bagi mahasiswa untuk tinggal selama masa studi di ITS Surabaya. Pengelolaan layanan Asrama Mahasiswa tersebut akan dikelola oleh sebuah sistem yang akan dirancang bernama myITS Dorm.



Gambar 1. Lapisan-lapisan di *onion architecture*.

Dari sisi kebutuhan, terdapat tiga fungsional utama yang terdapat pada sistem myITS Dorm, yaitu pemesanan kamar oleh mahasiswa ITS, pengelolaan fasilitas hunian oleh pengurus asrama ITS, dan pengelolaan kontrak antara mahasiswa dan pengurus asrama ITS. Tiap fungsional tersebut akan mempunyai sebuah proses bisnis yang cukup rumit dikarenakan berkaitan dengan pembelian, penyewaan, dan pembuatan kontrak. Dikarenakan rumitnya proses bisnis pada sistem myITS Dorm, pembuatan sistem myITS Dorm akan menggunakan metode *Domain Driven Design* dalam melakukan analisis terhadap proses bisnis terkait. Metode ini akan memecah suatu aplikasi menjadi beberapa domain, dimana pemecahan ini dilakukan berdasarkan proses bisnis dari sistem yang akan dibuat. Dengan pemecahan aplikasi menjadi domain-domain ini, maka kerumitan dari sebuah aplikasi akan berkurang, dan fungsi bisnis dari sebuah aplikasi dapat terklasifikasi lebih baik. Selain itu, sistem akan diimplementasikan menggunakan *Onion Architecture.* Penggunaan *Onion Architecture* pada implementasi sistem ini adalah berkaitan dengan *Separation of Concern* (pemisahan urusan), dimana dengan menggunakan arsitektur tersebut pada pembuatan sistem myITS Dorm berfungsi membagi aplikasi tersebut menjadi beberapa *layer*.

Hasil yang diharapkan adalah dapat menghasilkan sebuah sistem myITS Dorm yang berjalan dengan baik sesuai dengan proses bisnisnya, terstruktur dan terkelola dengan baik, sehingga dalam pemeliharaan aplikasi maupun dalam



Gambar 2. Lapisan-lapisan di *onion architecture.*

pengembangan aplikasi lebih lanjut, dapat dilakukan dengan lebih efektif dan efisien.

II. TINJAUAN PUSTAKA

A. Asrama Mahasiswa ITS

Asrama Mahasiswa ITS merupakan fasilitas hunian sewa yang disediakan bagi mahasiswa untuk tinggal selama masa studi di ITS Surabaya. Di bawah pengelolaan Direktorat Kerjasama dan Pengelolaan Usaha, asrama didesain sebagai suatu wahana dalam mendukung proses pengembangan diri dan pembangunan karakter pribadi mahasiswa. Berlokasi di dalam area kampus ITS Sukolilo dan didukung fasilitas yang memadai, menjadikan asrama mahasiswa adalah suatu pilihan kenyamanan dan keamanan dalam mengiringi kesuksesan studi di ITS Surabaya [1].

B. Domain Driven Design

Domain Driven Design adalah sebuah teknik dalam pengembangan sebuah perangkat lunak yang berfokus pada kolaborasi antara *technical experts* dan *domain experts* [2]. Dimana dalam pengembangan perangkat lunak, *technical experts* adalah orang yang berpengetahuan dalam pembuatan dan pengembangan sebuah perangkat lunak secara teknis, sedangkan *domain experts* adalah orang yang memiliki pengetahuan dan pengalaman terkait apa saja yang diperlukan dalam pembuatan sebuah perangkat lunak tersebut, biasanya terkait dengan proses bisnis dari perangkat lunak tersebut.

Karena perbedaan latar belakang, terkadang kolaborasi antara *technical experts* dan *domain experts* menjadi sulit. Oleh karena itu, dibutuhkan sebuah *ubiquitous language*. *Ubiquitous language* adalah sebuah bahasa yang sudah ditentukan dan digunakan dalam melakukan kolaborasi tersebut, sehingga semua pihak yang ikut dalam pengembangan tersebut, entah seorang *technical experts* ataupun seorang *domain experts* akan mengerti tentang proses pengembangan perangkat lunak tersebut [2].

Dengan *Domain Driven Design*, sebuah aplikasi akan dibagi menjadi beberapa *domain*, dimana tiap *domain* ini akan bersifat independen dan tidak berhubungan dengan *domain* lainnya. Penggambaran dari *domain* sendiri pada konsep *Domain Driven Design* adalah pada *Domain Model*. *Domain model* menjelaskan abstraksi pengetahuan yang disusun secara teliti dan selektif. *Domain model* dapat digunakan sebagai gambaran umum untuk mengomunikasikan ide antar *technical experts*, *domain expert*, dan pemangku kepentingan serta pihak-pihak lain yang terkait [2].

Tabel 1.
Aktor dan Perannya

No	Aktor	Peran
1	Administrator	Mengelola data kamar asrama Mengelola data jenis kamar asrama Mengelola data blok / gedung asrama Mengelola data pemesanan Mengelola data penghuni Mengelola laporan pemesanan Mencari jenis kamar asrama
2	Mahasiswa	Melakukan pemesanan reservasi kamar asrama Melakukan pembayaran pemesanan melalui sistem

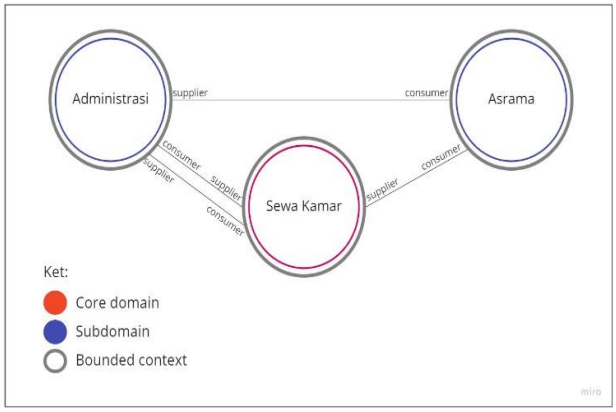
Tabel 2.
Kebutuhan Fungsional sistem myITS Dorm

Kode	Kebutuhan Fungsional	Deskripsi
F01	Mengelola data terkait fasilitas asrama	Mengelola data fasilitas asrama dalam bentuk penambahan, pengubahan dan penghapusan data-data seperti data blok / gedung, jenis kamar, kamar, pemesanan, penghuni, dan laporan pemesanan
F02	Melayani pencarian jenis kamar asrama, pengelolaan notifikasi, dan pengelolaan pemesanan yang telah dilakukan	Melayani pencarian jenis kamar yang disewakan oleh pihak asrama ITS. Melayani juga pencarian jenis kamar berdasarkan spesifikasinya. Melayani pengelolaan notifikasi yang diterima oleh pengguna dan pengelolaan pemesanan yang telah dilakukan oleh pengguna.
F03	Melayani proses pemesanan asrama ITS	Melayani proses pemesanan jenis kamar asrama ITS, dari awal melakukan pemesanan terhadap satu jenis kamar, memberikan spesifikasi pemesanan yang diinginkan, melakukan pembayaran, dan melakukan monitor pemesanan yang telah dilakukan.

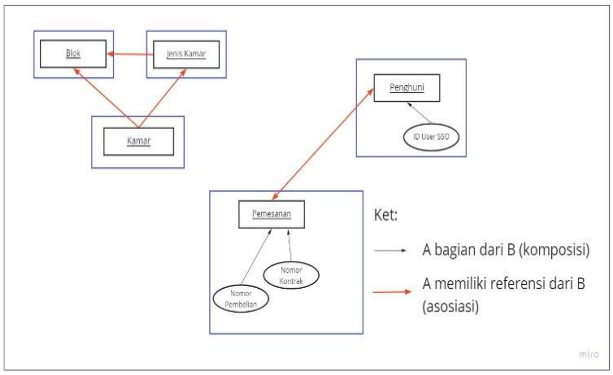
C. Onion Architecture

Onion Architecture adalah sebuah arsitektur perangkat lunak yang membagi sebuah perangkat lunak menjadi beberapa layer. Dimana pembagian layer ini bertujuan untuk *Separation of Concern* (Pemisahan urusan), sehingga tiap *layer* akan mempunyai urusannya tersendiri. Meskipun dipisahkan, namun antara *layer* luar dengan *layer* di dalamnya akan saling berhubungan, namun tidak sebaliknya, *layer* dalam tidak dapat berhubungan dengan *layer* luarnya. Contoh dari *Onion Architecture* ditunjukkan pada Gambar 1.

Dapat dilihat pada Gambar 1, bahwasannya sebuah perangkat lunak akan dibagi menjadi beberapa *layer*, dengan tiga *layer* yang paling dalam adalah *Application Core* layer, dimana berisi inti atau *core* dari sistemnya, dimana *core* tersebut terdiri dari *layer Domain Model*, *layer Domain Services*, dan *layer Application Services*. *Layer Domain Model* berisi model-model yang akan digunakan dalam sistemnya. *Layer Domain Services* adalah *layer* di atas *Domain Model*, berisi beragam *services* dari *domain* yang akan dijalankan oleh *layer Application Services*. *Layer* selanjutnya adalah *Application Services*, *layer* ini berisi penggunaan *method* dari *Domain Services*, dan pembuatan *instance* menggunakan *Domain Model* yang telah ditentukan sebelumnya. Pada *layer* ini nantinya akan menjalankan alur proses suatu *service*. *Layer* ini biasanya diakses oleh *layer* terluar, yaitu *layer User Interfaces*, *Infrastructure*, dan *Tests* [3].



Gambar 3. *Context Map* sistem myITS Dorm.



Gambar 4. *Domain model* dalam konteks Administrasi.

D. PHP

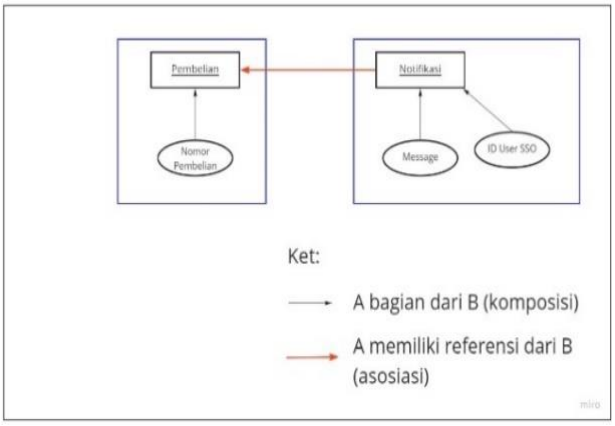
PHP adalah sebuah bahasa pemrograman di sisi *server* (*server side scripting*) yang bersifat *open source*. Sebagai sebuah *scripting language*, PHP menjalankan instruksi pemrograman saat proses *runtime*. Hasil dari instruksi tentu akan berbeda tergantung data yang diproses. PHP merupakan bahasa pemrograman *server-side*, sehingga *script* dari PHP nantinya akan diproses di sisi server.

Keunggulan menggunakan bahasa pemrograman PHP adalah bahasa PHP merupakan bahasa yang bersifat *open source*, sehingga pengguna bebas menggunakan dan mengembangkan sesuai dengan kebutuhan mereka. Selain itu, banyak pengembang web yang menggunakan bahasa PHP, sehingga banyak panduan yang tersebar di internet dan mempelajari bahasa PHP ini menjadi tidak sulit. Bahasa PHP juga mempunyai kecepatan yang lebih cepat dibandingkan dengan bahasa lain. Bahasa ini juga mempunyai banyak *library*, yang dapat memudahkan pengembangan dan mempercepat waktu pengembangan aplikasi. Bahasa ini juga mempunyai kompabilitas yang baik dengan HTML, sehingga cocok digunakan dalam mengembangkan sebuah aplikasi web.

E. Laravel

Laravel adalah sebuah *framework* aplikasi web yang ekspresif dan mempunyai sintaks yang elegan [4]. Laravel merupakan *framework* yang menggunakan bahasa pemrograman PHP. Versi terbaru dari *framework* laravel adalah versi 8 (Laravel 8).

Beberapa fitur unggulan dalam *framework* laravel yang dapat memudahkan proses pengembangan sebuah aplikasi, yaitu *Template Engine*, *Routing*, dan *Modularity*. Pada salah satu fitur unggulan, terdapat fitur *Modularity*, dimana *framework* laravel dapat dikembangkan secara modular.



Gambar 5. *Domain model* dalam konteks Sewa Kamar.

III. PERANCANGAN DAN IMPLEMENTASI

A. Analisis Kebutuhan

Terdapat beberapa tahapan dalam melakukan perancangan menggunakan konsep *domain driven design*, tahapan tersebut dapat dilihat pada Gambar 2.

Berdasarkan tahapan tersebut, tahapan pertama pada perancangannya adalah analisis kebutuhan. Analisis kebutuhan dilakukan berdasarkan spesifikasi dari sistem yang akan dibuat. Spesifikasi dari sistem yang akan dibuat seperti yang ditunjukkan Tabel 1.

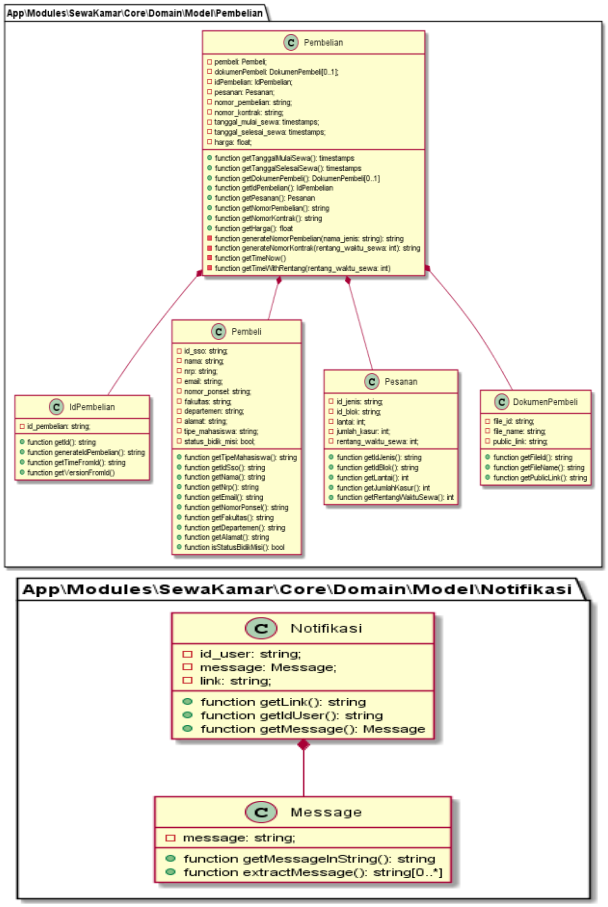
Berdasarkan spesifikasi sistem pada Tabel 1 dapat dianalisis kebutuhan fungsionalnya. Kebutuhan fungsional dari sistem myITS Dorm yang dapat penulis analisis adalah seperti yang ditunjukkan Tabel 2.

B. Mendefinisikan Ubiquitous Language

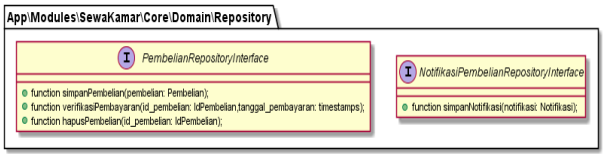
Setelah mendapatkan kebutuhan fungsional dari sistem myITS Dorm, langkah selanjutnya adalah mendefinisikan *ubiquitous language*. *Ubiquitous language* dari sistem yang akan didefinisikan harus didasarkan pada proses bisnis sistemnya. Dengan mendefinisikan *ubiquitous language*, maka pengembangan sistem akan didiskusikan dengan menggunakan *ubiquitous language*, dan implementasi kode dari sistem yang dibuat akan disesuaikan dengan ubiquitous language yang didefinisikan. Hal ini bertujuan agar implementasi sistem yang dibuat dapat dipahami semua pihak yang ikut mengembangkan sistem yang dibuat, dan tidak memerlukan dokumentasi formal terkait sistem yang dibuat. Beberapa *ubiquitous language*-nya adalah sebagai berikut:

1. Penyewa adalah mahasiswa ITS
2. Pengurus asrama adalah pengguna yang dapat melakukan pengelolaan data
3. Blok adalah jenis gedung yang mempunyai banyak jenis kamar
4. Jenis kamar merupakan jenis kamar asrama ITS. Satu jenis kamar bisa mencirikan banyak kamar.
5. Kamar mempunyai nomor kamar sebagai identitas dari kamar tersebut.
6. Pemesanan merupakan kegiatan yang dapat dilakukan penyewa pada sistem myITS Dorm, berkaitan dengan pemilihan jenis kamar oleh penyewa dan memberikan spesifikasi tertentu

Penjelasan terkait *ubiquitous language* yang lengkap dan detail akan dijelaskan pada saat mendefinisikan *context map*.



Gambar 6. Diagram kelas *aggregate* Notifikasi modul Sewa Kamar.



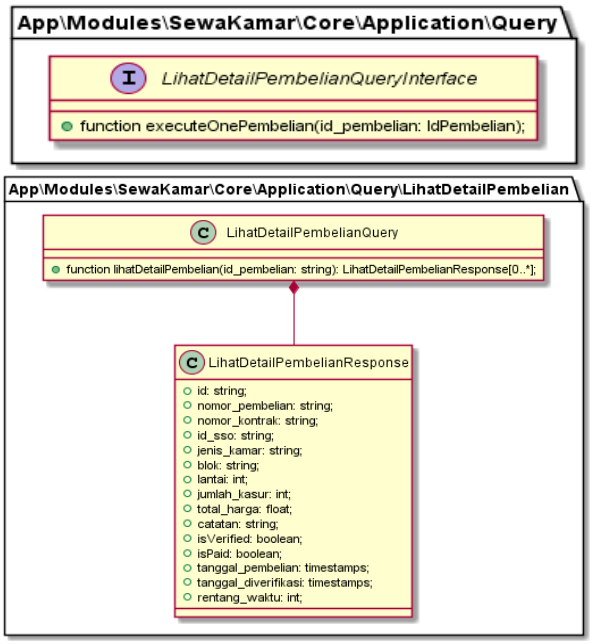
Gambar 7. Diagram kelas *repository* modul Sewa Kamar.

C. *Mendefinisikan Bounded Context*

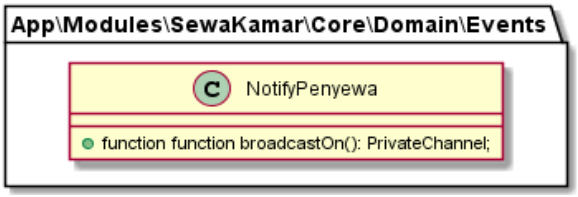
Setelah mendefinisikan *ubiquitous language*, selanjutnya adalah mendefinisikan *bounded context* dari sistem yang akan dibuat. Pada kasus ini, penulis melakukan tahap kedua dan tahap ketiga secara bersamaan, karena pendefinisian *bounded context* didasarkan pada *ubiquitous language* yang telah didefinisikan. *Bounded context* bertujuan untuk membatasi definisi *ubiquitous language* yang masih bersifat ambigu, atau bermakna lebih dari satu, dan atau masih mempunyai definisi dengan cakupan yang luas serta belum memiliki definisi yang detail dan jelas. *Bounded context* pada sistem ini terdapat tiga, yaitu Administrasi, Sewa Kamar, dan Asrama. Penjelasan terkait ketiga *bounded context* ini akan dijelaskan pada saat melakukan pendefinisian *context map*-nya.

D. *Mendefinisikan Context Map*

Context map merupakan sebuah gambar yang berfungsi untuk memahami konteks-konteks yang ada dan relasinya. Seperti yang dijelaskan sebelumnya, bahwasannya terdapat tiga *bounded context* pada sistem ini, yaitu Administrasi, Sewa Kamar, dan Asrama. *Bounded context* Administrasi akan mengelola fungsional F01, *bounded context* Asrama akan mewakili fungsional F02, dan *bounded context* Sewa Kamar akan mewakili fungsional F03. Pendefinisian



Gambar 8. *Interface Query Object*, *Class* dan DTO *Query Object* modul Sewa Kamar.



Gambar 9. Diagram kelas *domain event* modul Sewa Kamar.

bounded context yang dijelaskan sebelumnya, masih belum menjelaskan terkait interaksi antar *bounded context*-nya. Penjelasan terkait interaksi tersebut akan dijelaskan pada sebuah *context map*. *Context map* tidak memiliki bentuk yang baku, namun sebisanya dapat dipahami oleh seluruh pihak yang ikut dalam mengembangkan sistemnya. *Context map* dari sistem myITS Dorm seperti yang ditunjukkan Gambar 3.

E. *Memodelkan Subdomain*

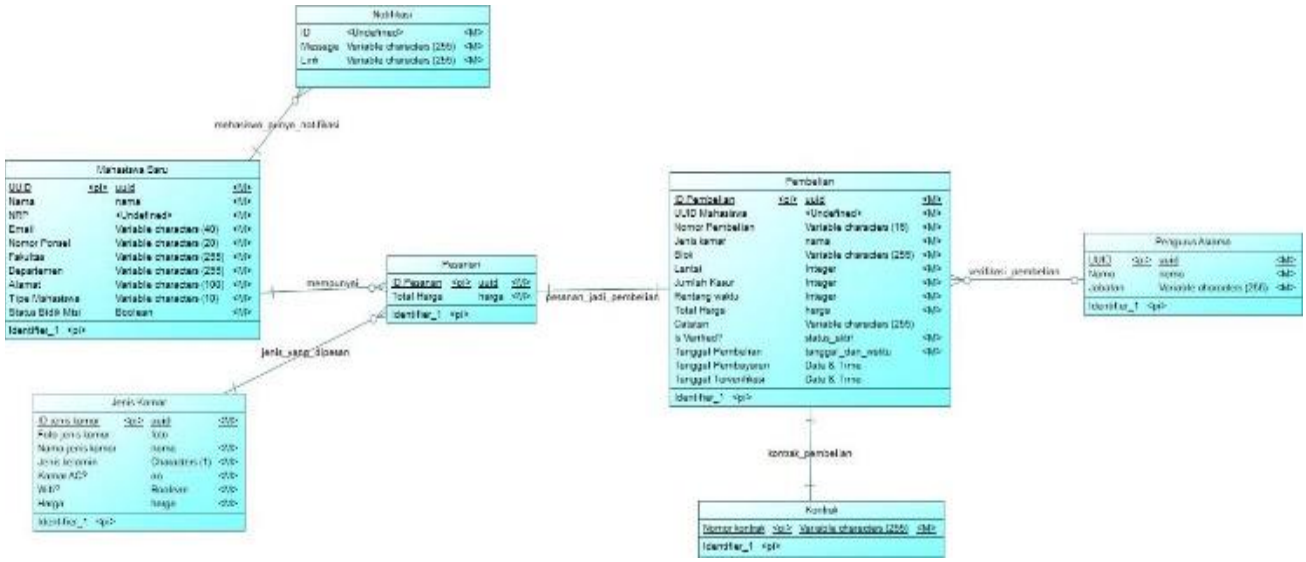
Pada Gambar 3, dapat dilihat bahwa 1 *subdomain* mempunyai 1 *bounded context*, serta relasi antar *bounded context* adalah *Supplier-Customer*, dimana *supplier* berarti pemberi informasi, sedangkan *customer* merupakan penerima informasi.

Penjelasan mengenai *ubiquitous language* dan *domain model* pada setiap *bounded context* pada *context map* Gambar 3 adalah sebagai berikut:

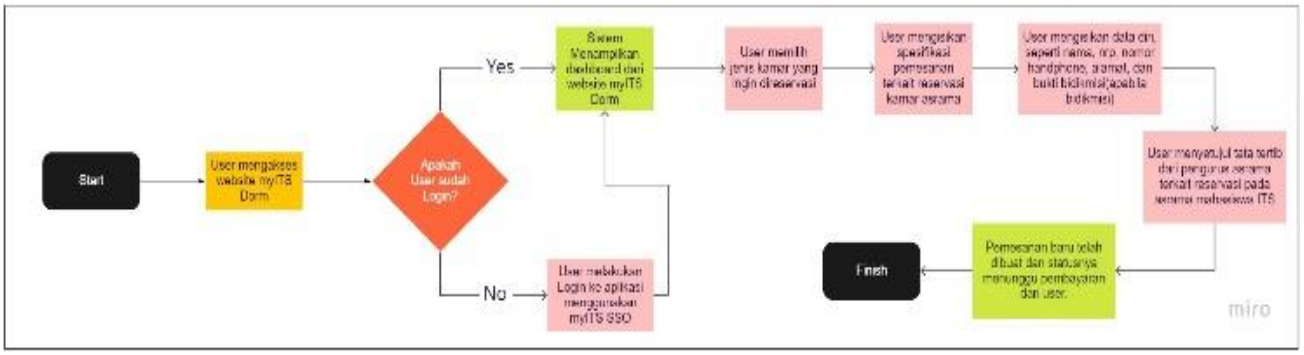
1) *Administrasi*

Ubiquitous language dalam *bounded context* ini dijelaskan sebagai berikut:

1. Pengurus asrama adalah pengguna yang dapat melakukan pengelolaan data
2. Pengurus asrama dapat menambah blok, jenis kamar, dan kamar asrama baru
3. Blok adalah jenis gedung yang mempunyai banyak jenis kamar
4. Jenis kamar merupakan jenis kamar asrama ITS. Satu jenis kamar bisa mencirikan banyak kamar.
5. Kamar mempunyai nomor kamar sebagai identitas dari kamar tersebut.
6. Pengurus asrama dapat melihat pemesanan dan penghuni



Gambar 10. CDM modul Sewa kamar.



Gambar 11. Alur Kerja pemesanan sewa kamar.

7. Pengurus asrama dapat mengatur blok dan kamar yang tersedia bagi penyewa
8. Pengurus asrama dapat melakukan verifikasi pemesanan dan memilihkan kamar untuk penyewa
9. Pengurus asrama dapat melihat laporan pemesanan dan penghasilan bulanan

Gambar 4 merupakan *domain model* dari konteks Administrasi.

2) Sewa Kamar

Ubiquitous language dalam *bounded context* ini dijelaskan sebagai berikut:

a. Pemesanan merupakan kegiatan yang dapat dilakukan penyewa pada sistem myITS Dorm, berkaitan dengan pemilihan jenis kamar oleh penyewa dan memberikan spesifikasi tertentu
b. Penyewa melakukan sewa jenis kamar asrama
c. Penyewa mengisikan spesifikasi pembelian dan data diri pada saat melakukan sewa kamar asrama
d. Penyewa melakukan pembayaran melalui virtual account
e. Penyewa yang sudah terverifikasi oleh pengurus asrama akan menjadi Penghuni Asrama ITS
f. Penyewa dan pengurus asrama melihat detail pemesanan
g. Pembuatan kontrak setelah pembelian telah usai

Gambar 5 merupakan *domain model* dari konteks Sewa Kamar.

3) Asrama

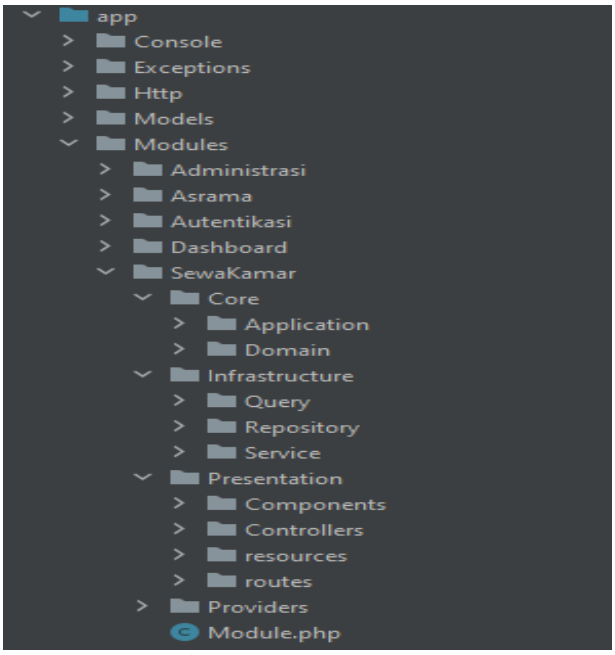
Bounded context Asrama memuat *subdomain* Asrama. *Ubiquitous language* dalam *bounded context* ini dijelaskan sebagai berikut:

a) Penyewa melihat jenis kamar yang ditawarkan
b) Penyewa mendapatkan notifikasi terkait pemesanan dan pembayarannya
c) Penyewa dapat melihat semua pemesanan yang telah dilakukan

Pada *bounded context* Asrama penulis tidak mendesain *domain model*, karena *business logic* yang terdapat pada *bounded context* Asrama yang sederhana, sehingga hanya menggunakan *query object* untuk menampilkan katalog jenis kamar yang ditawarkan, notifikasi, dan pemesanan mahasiswa.

F. Implementasi

Tahapan terakhir dari perancangan dengan menggunakan konsep *domain driven design* adalah implementasi dari sistemnya. Implementasi sistem myITS Dorm akan menggunakan *onion architecture*. Sebelum membahas terkait hal tersebut, penulis akan menjelaskan penggambaran *class diagram* yang dibuat berdasarkan *domain model* yang telah dibuat. Salah satu contoh kelas diagram yang penulis ambil sebagai contoh adalah kelas diagram pada modul Sewa Kamar. Pada modul ini akan dibuat diagram kelas dari *domain model*, *repository*, *events*, dan *query object*. *Domain model* yang dibuat akan berdasarkan *domain model* diagram yang telah digambarkan sebelumnya, sedangkan *repository* akan menunjukan metode apa saja yang akan digunakan pada *service* modul Sewa Kamar, *Events* merupakan sebuah kejadian yang di-*publish* dan akan melakukan *update* pada *domain model* yang mendengarkan *event* tersebut, dan *query object* merupakan kelas yang mengelola tampilan dari modul



Gambar 12. Struktur *folder* proyek perangkat lunak.

```
class Pembelian
{
    private Pembeli $pembeli;
    private ?DokumenPembeli $dokumenPembeli;
    private IdPembelian $idPembelian;
    private Pesanan $pesanan;
    private NomorPembelian $nomor_pembelian;
    private $tanggal_pembayaran;
    private bool $isPaid;
    private float $harga;
public function __construct(...)
    {
        ...
    }
...
public function setTanggalPembayaran($tanggal_pembayaran): void
    {
        $this->tanggal_pembayaran = $tanggal_pembayaran;
    }
 public function setIsPaid(bool $isPaid): void
    {
        $this->isPaid = $isPaid;
    }
...
```

Gambar 13. Implementasi *Class* Pembelian.

Sewa Kamar. Visualisasi diagram kelas tersebut seperti yang ditunjukkan Gambar 6, Gambar 7, Gambar 8, dan Gambar 9.

Setelah melakukan visualisasi, selanjutnya adalah melakukan perancangan basis data dari modul tersebut. Dimana perancangan tersebut divisualisasikan dalam sebuah *Conceptual Data Model*. *Conceptual Data Model* dari modul Sewa Kamar seperti yang ditunjukkan Gambar 10.

Tahapan terakhir dari implementasi adalah membuat implementasi kode nya sesuai dengan alur kerja dari proses bisnis modul tersebut. Salah satu alur kerja dari modul Sewa Kamar adalah pemesanan sewa kamar oleh mahasiswa. Alur kerja tersebut dapat dilihat pada Gambar 11.

Implementasi pada sistem myITS Dorm meliputi implementasi struktur proyek dalam Laravel versi 8, implementasi pengetahuan *domain*, implementasi kasus penggunaan, dan implementasi antarmuka.

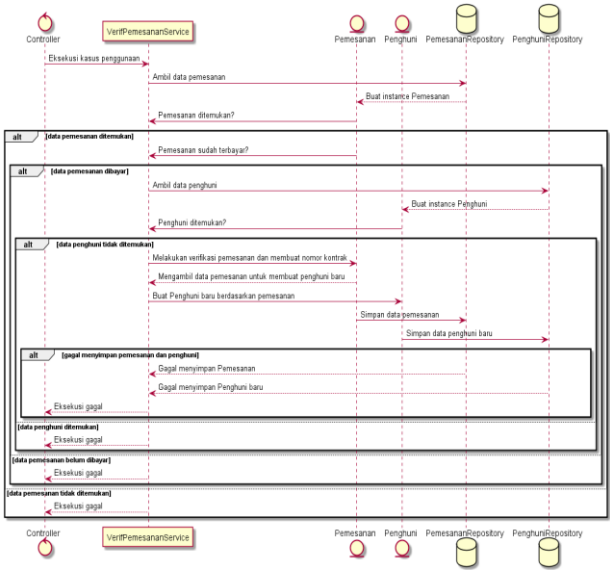
Implementasi struktur proyek dalam Laravel akan mengubah struktur *folder* pada struktur umum laravel sendiri. Dimana pengubahan ini akan berkaitan dengan implementasi dari sistem myITS Dorm menggunakan *onion architecture.*

```
class VerifPemesananService
{
....
public function execute(VerifPemesananRequest $request)
    {
        ....
        $pemesanan = $this->pemesananRepository->findPemesanan(new
IdPemesanan($request->getId()));

        if(!$pemesanan)
            throw new \InvalidArgumentException("Pemesanan tidak
ditemukan");
        if(!$pemesanan->isPaid())
            throw new \InvalidArgumentException("Pemesanan belum
dibayar oleh user, tidak bisa melakukan verifikasi");

        $penghuni = $this->penghuniRepository->findPenghuni(new
IdPenghuni($pemesanan->getPemesan()->getIdSso()));
        if($penghuni)
            throw new \InvalidArgumentException("User telah menjadi
penghuni dan sudah terikat kontrak, mohon menghapus user dari
penghuni terlebih dahulu");
        ....
    }
```

Gambar 14. Implementasi kasus penggunaan verifikasi Pemesanan modul Administrasi.



Gambar 15. Contoh *sequence diagram* yang dapat divisualisasikan dari kode verifikasi Pemesanan.

Struktur *project* laravel dirubah menjadi seperti yang ditunjukkan Gambar 12.

Berdasarkan Gambar 12, modifikasi dilakukan pada folder *app*, dimana terdapat tambahan folder baru yaitu *modules*. Pada folder *modules* ini nantinya terdapat modul-modul yang sudah dirancang sebelumnya, dimana berfungsi untuk membagi sistem agar kompleksitasnya menjadi lebih kecil. Nantinya tiap modul akan mengimplementasikan *onion architecture*, dimana membagi modul menjadi beberapa *layer*, yaitu *application core layer*, yang terdiri dari *Application* dan *Domain*, dimana *Application* berisi *Application Service*, dan *Domain* berisi *Domain Model* dan *Domain Service*. Lalu terdapat dua *layer* terluar dari *onion architecture*, yaitu *Infrastructure* dan *Presentation*. *Folder infrastructure* akan menggambarkan layer terluar *onion architecture*, yaitu *layer infrastructure*. *Folder Presentation* akan menggambarkan *layer* terluar onion architecture, yaitu *layer user interface*.

Implementasi pengetahuan *domain* adalah bagaimana implementasi kode pada suatu kelas digambarkan berdasarkan *ubiquitous language*-nya. Contoh yang akan

Tabel 3.
Lingkungan Pengujian

Spesifikasi	Deskripsi
Jenis Perangkat	*Personal Computer*
CPU	Intel Core i3-9100f
RAM	8GB
Sistem Operasi	Windows 10

Tabel 4.
Hasil Pengujian Kasus Penggunaan Modul Administrasi

No.	Kasus Penggunaan	Terpenuhi	
		Ya	Tidak
1	melihat blok	√	
2	membuat blok baru	√	
3	menghapus satu blok	√	
4	melihat jenis kamar	√	
5	menambah jenis kamar baru	√	
6	menghapus satu jenis kamar	√	
7	melihat kamar	√	
8	menambah kamar baru	√	
9	menghapus satu kamar	√	
10	melihat pemesanan	√	
11	melihat detail pemesanan	√	
12	melihat penghuni	√	
13	melihat detail penghuni	√	
14	melihat laporan pemesanan	√	
15	melihat laporan penghasilan	√	
16	melakukan verifikasi pemesanan	√	
17	menghapus satu penghuni	√	

Tabel 5.
Hasil Pengujian Kasus Penggunaan Modul Asrama

No.	Kasus Penggunaan	Terpenuhi	
		Ya	Tidak
1	melihat menu jenis kamar	√	
2	melihat menu jenis kamar dengan filter	√	
3	melihat detail menu jenis	√	
4	melihat semua pembelian	√	
5	melihat notifikasi	√	

penulis gunakan adalah *class* Pembelian yang merupakan *domain model* dari modul Sewa Kamar.

Pada kode dalam Gambar 13, terdapat beberapa fungsional yang berkaitan dengan *ubiquitous language* yang ditentukan. Seperti pada pembelian, terdapat pembeli, pesanan, dokumen pembelian yang bersifat opsional, nomor pembelian, dan harga dari pembeliannya sendiri. Selain itu, terdapat fungsi untuk memberikan nilai, yaitu setTanggalPembayaran dan setIsPaid, dimana fungsi tersebut digunakan pada saat melakukan verifikasi pembayaran. Apabila nilai dari tanggal pembayaran dan *is paid* terisi, maka pembelian tersebut telah terbayar oleh pengguna.

Implementasi dari kasus penggunaan dalam kode yang dibuat pada saat menjalankan suatu *service* mengikuti *ubiquitous language* yang telah didefinisikan sebelumnya. Contoh yang akan penulis ambil adalah pada saat melakukan verifikasi pemesanan pada modul Administrasi seperti yang ditunjukkan pada Gambar 14.

Gambar 15 merupakan contoh *sequence diagram* yang dapat divisualisasikan dari runtutan kode tersebut.

Pada visualisasi Gambar 15, dapat dilihat bahwa dalam melakukan verifikasi pemesanan, maka perlu membuat *instance* pemesanan dengan mencari berdasarkan idnya melalui *repository*. Setelah itu, melakukan pengecekan apakah pemesanan ada, dan apakah pemesanan sudah dibayar atau belum. Apabila terdapat pemesanan dan sudah dibayar,

Tabel 6.
Hasil Pengujian Kasus Penggunaan Modul Sewa Kamar

No.	Kasus Penggunaan	Terpenuhi	
		Ya	Tidak
1	melakukan pembelian baru	√	
2	melakukan verifikasi pembayaran	√	

selanjutnya adalah mengecek apakah pemesan sudah menjadi penghuni atau belum, dengan melakukan pencarian penghuni menggunakan id pemesan pada pemesanan. Apabila belum menjadi penghuni, maka pemesanan dapat diverifikasi, dan pemesan akan menjadi penghuni serta melakukan *generate* nomor kontrak yang menandakan kontrak sewa antara pemesan dengan pengurus asrama ITS.

IV. UJI COBA DAN EVALUASI

Uji coba yang dilakukan pada sistem myITS Dorm akan dilakukan pengujian setiap fungsionalitas dari myITS Dorm. Pengujian menggunakan lingkungan pengujian dengan spesifikasi seperti yang ditunjukkan Tabel 3.

Hasil pengujian untuk setiap kasus penggunaan pada modul ditunjukkan pada Tabel 4, Tabel 5, dan Tabel 6.

V. KESIMPULAN DAN SARAN

A. Kesimpulan

Kesimpulan yang dapat diambil berdasarkan penjelasan diatas adalah sebagai berikut: (1) Pembentukan *domain model* dilakukan berdasarkan proses bisnis yang telah didapatkan dari *domain expert* sebagai bagian dari proses konsep berfikir dengan pendekatan *domain driven design.* Proses perancangan juga sudah dilakukan sesuai dengan tahapannya, dimulai dari analisis kebutuhan, menentukan *ubiquitous language*, mendefinisikan *bounded* context, merancang *context map*, memodelkan *subdomain* (modul), dan melakukan implementasi berdasarkan rancangannya; (2) Implementasi *onion architecture* diterapkan dalam pembuatan sistem myITS Dorm, dimana penerapan didasarkan pada hasil perancangan menggunakan pendekatan *domain driven design*. Proses implementasi diawali dengan membuat *Domain Model layer*, dimana dibuat berdasarkan *domain model* hasil perancangannya, lalu dilanjutkan dengan pembuatan *Domain Service layer* dan *Application Service layer*.

B. Saran

Saran dari penjelasan yang telah dijelaskan sebelumnya adalah sebagai berikut: (1) Implementasi mekanisme autentikasi masih menggunakan myITS SSO, dan belum menggunakan implementasi autentikasi biasa, sehingga user umum belum bisa mengakses sistemnya; (2) Sistem masih terpisah dari sistem myITS Marketplace, kedepannya dapat diintegrasikan dengan layanan lain yang terdapat pada myITS Marketplace.

DAFTAR PUSTAKA

[1] I. T. S. Nopember, "Tempat Tinggal dan Tempat Makan - Institut Teknologi Sepuluh Nopember," *Institut Teknologi Sepuluh Nopember*. https://www.its.ac.id/id/kehidupan-kampus/eksplor-its/tempat-tinggal-dan-tempat-makan/.

[2] E. Evans and E. J. Evans, *Domain-Driven Design: Tackling Complexity In The Heart Of Software*. Bolyston Street, Boston: Addison-Wesley

Professional, 2004.
[3] M. E. Khalil, K. Ghani, and W. Khalil, “Onion Architecture: A New Approach For Xaas (Every-Thing-As-A Service) Based Virtual Collaborations,” in *2016 13th Learning and Technology Conference (L\&T)*, 2016, pp. 1–7.
[4] Laravel, “Laravel - The PHP Framework For Web Artisans,” *Laravel*, 2011. https://laravel.com/.