1. Pendahuluan

1.1. Latar Belakang

Kemampuan manusia berkembang dan beradaptasi begitu cepat, terutama dalam hal kehidupan, teknologi, dan budaya. Hal ini terjadi karena gaya hidup dan tuntutan untuk menjadi terdepan dan terbaik, karenanya manusia berlomba untuk menyelesaikan masalah dan tantangan hidup dengan mudah, cepat dan praktis. Dalam hal ini kehadiran teknologi sangat penting untuk membantu dan mempermudah kehidupan manusia sehari-hari. Terutama di bidang teknologi informasi, tentunya kehadiran dunia *Information Technology (IT)* atau biasa disebut sebagai Teknologi Informasi (TI) mengubah era manusia di masa sekarang, bisa terlihat di berbagai sektor seperti pendidikan, industri budaya dan masih banyak lagi. Karenanya kehadiran di anggap mengubah dan membantu sistem yang sudah ada sebelumnya, dan karenanya diharapkan kehadiran TI sendiri dapat membantu, mempermudah dan menyederhanakan kegiatan yang ada dalam kehidupan masyarakat itu sendiri.

Alasan tersebut yang kemudian mendasari ide untuk membuat sebuah *website* dengan *server based springboot* yang dapat dikendalikan oleh pihak UMKM sebagai sumber akses informasi, pemasaran dan juga barang yang dijual oleh *clothing brand* dari UMKM yang bernama Eey.Id, dengan begitu konsumen dapat menjadi lebih dekat dan juga memiliki standar kepercayaan masyarakat serta konsumen sebagai salah satu *clothing brand* yang sedang bertumbuh dengan menawarkan produk *clothing* yang memiliki desain dan juga nilai seni di dalamnya. Kendala yang saat ini dihadapi oleh UMKM Eey.Id ini sendiri adalah kesulitan untuk mengakses segala informasi dan juga kegiatan sebelumnya yang telah dilakukan, pemasaran yang sudah berjalan dan juga produk-produk yang akan di hadirkan. Karena itu menggunakan *website* Eey.id yang akan diolah akan mempermudah mengakses dan juga menyimpan informasi yang sebelumnya sudah dimiliki dan juga menghadirkan kembali informasi yang diperlukan sebagai kebutuhan pemasaran dan juga daya tarik terhadap konsumen dan juga masyarakat. Karena itu pembuatan *website* ini sendiri menggunakan *Spring boot* karena dikenal sebagai sebuah *framework* pembuatan *website* yang lebih menekankan terhadap penggunaan dan kontrol dari *server*, dalam kasus ini *server* itu sendiri adalah pihak UMKM Eey.Id. Sehingga informasi dan segala macam kegiatan serta konten yang di hadirkan memiliki persetujuan dari pihak Eey.Id dan menguntungkan untuk pihak konsumen serta masyarakat sendiri.

1.2. Rumusan Masalah

Dari latar belakang masalah yang telah disimpulkan maka, tersusun beberapa rumusan masalah yang ada dari kasus tersebut, yaitu :

1. Bagaimana menyampaikan informasi terkait produk dari Eey.Id sendiri kepada konsumen.
2. Bagaimana membuat sistem informasi dan tempat penyimpanan detail produk yang ada, agar mudah dihadirkan dan dikemas untuk disajikan kepada konsumen.
3. Bagaimana membuat sistem informasi yang menarik dan mudah untuk diakses oleh konsumen.

1.3. Tujuan dan Manfaat

Tujuan dari pembuatan *website* sistem informasi untuk UMKM Eey.Id sendiri untuk kemudahan dan juga meningkatkan daya tarik dari brand lokal Semarang yang sedang berusaha meningkatkan kepercayaan dan juga kredibilitas dari brand ini sendiri.

Karena itu dibutuhkan sistem informasi yang mudah diakses seperti *website*, karena kemudahan untuk diakses dan juga memiliki nilai estetika dalam penyajian terhadap konsumen dan juga masyarakat.

Diharapkan manfaat yang diinginkan dapat tercapai dari pembangunan *website* ini sendiri, manfaat tersebut bagi pihak pemilik sendiri dapat memberi kemudahan untuk mempromosikan *brand* Eey.Id ini sendiri, selain itu dapat memudahkan dalam proses penjualan dan pembelian barang-barang dari brand Eey.Id ini.

1.4. Batasan Masalah

Agar perancangan dan pembuatan sistem informasi serta penelitian ini lebih fokus pada masalah yang ada, maka dibuatlah beberapa batasan masalah yang ada, yaitu:

1. Perancangan website hanya berupa sistem informasi dimana penjualan diarahkan ke platform website yang bernama Shopee.
2. Tidak diperlukan akun untuk mengakses *website*, karena tidak ada penjualan dan pembelian melalui *website* Eey.Id sendiri, melainkan pembelian diarahkan ke *website* Shopee.

2. Tinjauan Pustaka

a. Penelitian Terdahulu

Penelitian terdahulu ada pada jurnal SISFOKOM yang berjudul "Perancangan Mengenai Perancangan Sistem Informasi Penjualan Pakaian Berbasis *Web* Pada Toko Uj Outlet". Uj Outlet adalah toko yang menjual pakaian pria.

Website tersebut dibuat untuk meningkatkan promosi dan kesadaran masyarakat tentang Uj Outlet, sama halnya seperti yang dilakukan untuk brand Eey.Id ini sendiri.

Dengan memiliki website sendiri untuk promosi dan penjualan maka akan mempermudah konsumen secara luas untuk menjangkau pakaian Uj Outlet, tanpa harus datang ke outlet atau toko pakaian dan *brand* itu sendiri.

b. Bahasa Pemrograman *Java*

Java adalah sebuah bahasa pemrograman komputer berbasis *Object Oriented Programming (OOP)*. Java diciptakan setelah C++ dan memiliki desain sedemikian rupa, sehingga *portable,* ukurannya kecil, dan sederhana (dapat dipindah-pindahkan di antara bermacam-macam platform dan sistem operasi). Program yang dihasilkan dapat berupa APPLET (aplikasi kecil yang dijalankan melalui *web*

browser), MIDLET (aplikasi yang berjalan melalui ponsel), maupun aplikasi mandiri yang dijalankan dengan program java *Interpreter*.

Java memiliki keunggulan yaitu sifatnya yang *platform independence*, yang artinya setiap *source code* yang dihasilkan dan dijalankan dan juga kompilasinya tidak bergantung pada sistem operasi dan juga *platform* yang digunakan, contohnya adalah *source code* yang dibuat dan dikompilasi di Windows NT dapat dijalankan pada sistem operasi Linux.

Saat ini generasi yang sedang berkembang dari *platform* Java adalah Java 2. Sebuah program Java akan dikompilasi menjadi *file bytecode*, hal ini dilakukan agar sebuah program Java dapat dijalankan. Untuk melakukan hal ini diperlukan *Java Runtime Environment* (JRE) yang memungkinkan sebuah program Java dapat digunakan tanpa menulis atau membuat ulang. Diperlukan JRE berisi JVM dan *library* Java.

c. *Hibernate ORM Framework*

Object Relational Mapping (ORM) adalah suatu cara untuk mengubah data dari *object oriented model* ke dalam bentuk relasi *database model*. Dimana *Object Oriented Programming* (OOP) berbasis entitas pada *database*, sehingga *Relational Database Management System* (RDBMS) di sortir atau di pilah bagaimana relasi dan kolom pada *database* digunakan untuk menyimpan data.

Data disimpan pada MySQL menggunakan *Structured Query Language* (SQL), kemudian pada SQL tersebut telah dipetakan bagaimana hubungan antar entitas dan kolom. Melalui *framework* ORM, entitas tersebut akan di konversi menjadi sebuah objek dalam struktur proyek bahasa pemrograman yang berbasis objek. Sehingga akan memudahkan untuk memetakan dan memanggil isi dari *database* yang telah dihubungkan dalam sebuah *project*.

d. Springboot Framework

Spring Boot memiliki kemampuan yang memungkinkan untuk pengembangan aplikasi berbasis *Spring* yang dapat berdiri sendiri dan memiliki kualitas produksi yang dapat langsung digunakan atau dijalankan menggunakan server Tomcat tertanam. Hal ini dapat menghindari pembuatan *file* WAR yang tidak perlu untuk hal-hal dalam pembuatan. *Spring Boot* memiliki banyak konfigurasi *default* dan konfigurasi *default* ini juga dapat dilakukan beberapa adaptasi dan perubahan sesuai dengan kebutuhan aplikasi yang sedang dibangun. Ini juga mencakup penggunaan beberapa JAR pihak ketiga, default yang umum untuk semua aplikasi Spring.

Spring Framework didasarkan pada bahasa JVM dan memberikan dukungan infrastruktur lengkap untuk menulis aplikasi *website* yang efektif. Spring menyediakan beberapa keunggulan seperti lebih banyak waktu yang dapat digunakan untuk pengembangan aplikasi dan mengurangi penggunaan waktu yang tidak efisien dan berlebihan pada konfigurasi.

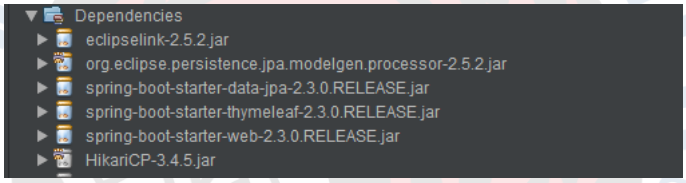
Aplikasi *Spring Boot* dapat dengan mudah untuk dibuat dan diakses menggunakan Spring Initializr, yang merupakan alat bootstrap yang disediakan oleh tim Pivotal. Kebutuhan yang ingin digunakan oleh aplikasi web dapat dipilih saat membuat proyek. *Spring boot* tidak menyediakan web.xml untuk melakukan konfigurasi aplikasi dan menghindari dapat menghindari *overheat* yang dapat

ditimbulkan dalam mengelola dan mengatur *file* XML. Sehingga, untuk menyelesaikan masalah ini *Spring Boot* memberikan fitur-fitur untuk mengubah konfigurasi aplikasi menggunakan anotasi, seperti menambahkan anotasi *@EnableAutoConfiguration* atau *@SpringBootApplication* di dalam *main class*, dan secara otomatis akan melakukan konfigurasi aplikasi berdasarkan dependensi JAR.

e. *Dependency Injection*

Dependency Injection merupakan metode yang digunakan untuk mengurangi ketergantungan pada aplikasi dengan basis objek. Penerapan pada *dependency injection* erat kaitannya dengan pemisahan perancangan aplikasi dengan *design pattern*, pemisahan ini bertujuan mengurangi ketergantungan antar bagian di dalam aplikasi.

Dependency merupakan sebuah *library code*, yang digunakan untuk memasukkan kebutuhan apa saja yang diperlukan dan akan digunakan oleh aplikasi tersebut. Kemudian *library* tersebut akan di-*inject* atau dimasukkan kedalam susunan *library* pada aplikasi tersebut. Sehingga *library* dapat digunakan untuk keperluan aplikasi yang dibuat. Berikut contohnya.



Gambar 1 Dependencies

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-thymeleaf</artifactId>
</dependency>
```

Gambar 2 Dependency Thymeleaf

Dalam *dependencies* ini, dimasukan *dependency* atau *library* Bernama *thymeleaf* untuk dapat menggunakan fungsi yang dapat digunakan jika menggunakan *dependency* ini, fungsi *thymeleaf* dapat digunakan sebagai *code* dan *syntax* untuk mengakses isi dari *database* yang ingin ditampilkan dalam aplikasi *web* yang dibuat.

f. *Design-Pattern*

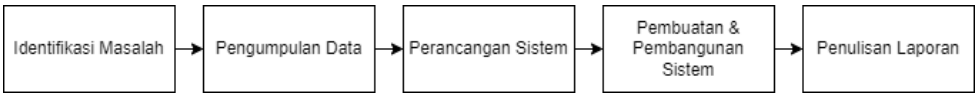
Design Pattern merupakan desain sebuah proyek dengan *data-flow* yang dibuat dan diatur sedemikian rupa agar proses data dan tersusunnya data yang digunakan sesuai dengan kebutuhan penggunaan dan pembuatan.

Design Pattern dibuat sesuai dengan kebutuhan programmer untuk menyelesaikan permasalahan yang berulang-ulang.

3. Metode Perancangan/Penelitian

a. Tahapan Penelitian

Tahapan yang dilakukan dapat dilihat pada gambar 3.



Gambar 3 Tahapan Penelitian

Tahapan penelitian dari gambar 3, dijelaskan bahwa pada tahap pertama diperlukan adanya identifikasi masalah dari studi kasus yang ada, setelah masalah dapat teridentifikasi maka dapat dilakukan tahap kedua, yaitu pengumpulan data dari masalah yang sudah teridentifikasi tersebut, kemudian dilanjutkan dengan tahap ketiga, yaitu perancangan sistem yang dapat menjawab permasalahan yang ada, setelah sistem dirancang maka dilanjutkan dengan tahap keempat, yaitu sistem yang sudah dirancangkan akan dibangun dan dibuat sesuai dengan permasalahan yang ada, sehingga dapat menjawab dan menjadi solusi dari permasalahan yang sudah teridentifikasi tersebut, pada tahap kelima disusun laporan dari seluruh kegiatan yang sudah dilakukan dan dikerjakan sesuai permasalahan dan metode yang sudah dipilih.

b. Rencana Kerja

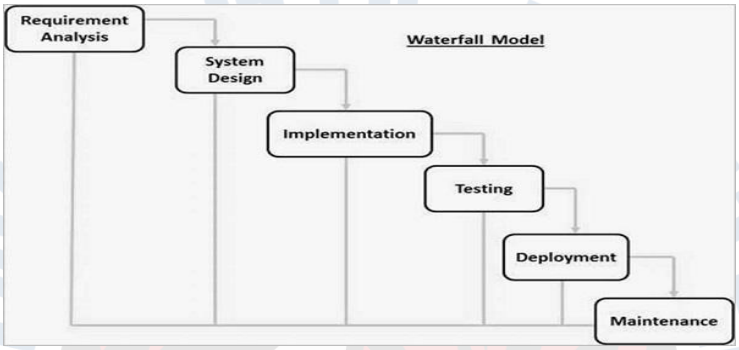
Kegiatan	Minggu ke-						
	1-2	3-4	5-6	7-8	9-10	11-12	13-14
Identifikasi masalah	v						
Pengumpulan data	v						
Perancangan sistem	v	v					
Pembuatan sistem		v	v	v	v		
Pengujian					v	v	
Penulisan laporan						v	v

Gambar 4 Tabel Rencana Kerja

c. Metode Penelitian

Metode yang digunakan untuk merancang dan membuat *website* sistem informasi UMKM Eey.Id sendiri menggunakan *Software Development Life Cycle* (SDLC) *Waterfall*, metode ini menggambarkan pendekatan yang sistematis dan juga berurutan pada pengembangan perangkat lunak (*software*), dimana siklus ini dimulai dengan spesifikasi kebutuhan pengguna, dan juga berlanjut kepada tahapan-tahapan perencanaan (*planning*), permodelan (*modeling*), konstruksi (*construction*), serta penyerahaan sistem kepada klien atau pengguna (*deployment*), yang diakhiri dengan dukungan terhadap perangkat lunak (*software*) yang telah dihasilkan atau dibuat.

Tahapan metode *waterfall* dapat dilihat pada gambar 5.



Gambar 5 Tahapan Metode Waterfall

Dalam *Software Development Life Cycle (SDLC),* terdapat urutan tahapan dalam prosesnya. Dimulai dari analisa kebutuhan (*requirement*), desain *(design)*, implementasi *(implementation), testing*, penyerahan kepada klien *(deployment)*, dan perawatan *(maintenance)*. Berikut penjelasan dari tahapan-tahapan tersebut :

a). *Requirement Analysis*

Dilakukan 6 nalisa kebutuhan terhadap perangkat lunak *(software)* oleh pengembang *(developer)*, kebutuhan yang diperoleh didapatkan dari hasil wawancara, survey atau diskusi. Hal ini dilakukan untuk mendapatkan keperluan pengguna dan juga Batasan perangkat lunak *(software)* itu sendiri.

b). *Design*

Analisis kebutuhan yang telah dikumpulkan kemudian akan dirancang sebelum dilakukan proses pembuatan atau pengkodingan *(code)*. Proses ini berfokus pada struktur data, arsitektural perangkat lunak, representasi *interface*, dan detail algoritma procedural.

c). *Implementasi/Code*

Desain atau rancangan mengenai perangkat lunak *(software)*, yang akan dibuat dituangkan melalui tahap implementasi, dimana rancangan yang telah disusun tadi dibuat menjadi bentuk kode *(code)*, menjadi bahasa yang dimengerti oleh mesin. Kode program yang telah dibuat biasanya masih menjadi bentuk modul-modul kecil yang nantinya akan digabungkan pada tahap berikutnya.

d). *Testing*

Pada tahap ini dilakukan penggabungan dari modul-modul yang telah dibuat, dan dilakukan pengujian apakah sesuai dengan desainnya dan fungsi pada perangkat lunak *(software)* terdapat kesalahan atau tidak.

e). *Deployment*

Setelah dilakukan penggabungan dan pengujian, maka perangkat lunak *(software)* yang telah selesai tahapan-tahapan diatas akan dijalankan atau diberikan kepada klien.

f). *Maintenance*

Tentunya perangkat lunak (*software*) yang telah dibuat tidak ditinggalkan begitu saja, tetapi dilakukan tahap pemeliharaan, termasuk perbaikan kesalahan yang tidak ditemukan pada tahap sebelumnya, dan perbaikan implementasi unit sistem serta peningkatan jasa sistem sebagai kebutuhan baru.

Untuk metode penelitian demi mengumpulkan data bagi pembangunan *website* ini sendiri menggunakan metode kualitatif yaitu wawancara, terhadap pihak pemilik dan beberapa orang untuk menguji pengalaman pengguna *(user experience)* terhadap aplikasi *website* ini.

Beberapa pihak yang terlibat dalam proses wawancara ini sendiri disebutkan sebagai berikut ini :

a) Alexander Paryto Buyung Erang : Founder
b) Irana Selistyowati : Founder

Kedua orang yang disebutkan adalah pemilik dari Brand Eey.Id ini sendiri, mereka melakukan kolaborasi dalam mendirikan dan mengembangkan bisnis ini. Berikut beberapa wawancara yang dicatat dan sebutkan :

> "Untuk promosi via medsos sendiri sudah dilakukan, diharapkan ini akan menaikan minat buat orang biar tertarik sama kaos-kaos kita. Berikutnya kami akan melakukan pergerakan ke sektor digital. Ya.. seperti website buat tempat promosi sekalian jualan gitu"

Disebutkan oleh Alex sendiri bahwa mereka memiliki program dalam bisnis untuk mengembangkan Eey.Id, dan jelas dikatakan bahwa sebuah *website* dibutuhkan dalam menjalankan bisnis ini seterusnya untuk promosi dan penjualan. Kemudian keinginan untuk menaikkan *engagement* dari pelanggan dapat dipecahkan dengan kehadiran *website* resmi milik mereka, hal ini disebabkan oleh kemampuan digital dimana produk mereka bisa diakses kapan saja dan dimana saja, selain itu media sosial yang lain bertujuan sebagai wadah untuk melakukan promosi untuk meningkatkan daya tarik, sementara untuk tahap *closing* atau penjualan dilakukan melalui *website* yang terhubung ke situs resmi Shoppee.

> "Kalau instagram berbeda buat penjualan, soalnya instagram cuma buat konten menarik minat masyarakat aja. Kalau ada *website* milik kami

sendiri akan lebih nyaman. Karena *wesbite* sendiri bisa untuk ngejual produk kami. Di mana nanti diarahkan ke Shoppee aja, soalnya kami juga lagi naikin traffic di olshop Shoppee kami".

Begitu informasi yang didapatkan melalui metode kualitatif yaitu wawancara langsung terhadap pihak owner, kemudian hal tersebut menjadi sebuah data dimana diketahui apa saja yang diinginkan untuk tercapai dari *website* ini sendiri. Sehingga dapat diketahui acuan bahwa *website* ini dianggap berhasil dari tujuan awalnya.

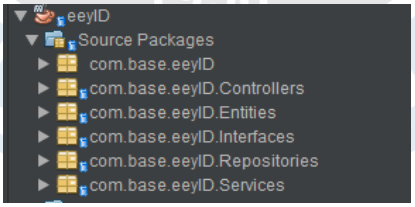
4. Hasil dan Pembahasan

a. MVC (*Model, View & Controller*)

Berdasarkan kebiasaan dan pengaruh sebuah *website* untuk sebuah *brand* khususnya dalam taraf UMKM diperlukan keabsahan dalam informasi yang disajikan, hal ini bertujuan untuk meningkatkan kredibilitas dan kepercayaan *customer* terhadap sebuah *brand* tersebut. Khususnya untuk Eey.Id ini sendiri, karena merupakan fokus dari penelitian saya sendiri. Berikut dipaparkan bagian-bagian dari *website* yang telah di buat.

Website dibuat menggunakan *framework Spring Boot*, sebelumnya telah dibahas bahwa *Spring Boot* merupakan sebuah *Framework Java* yang dibuat untuk mempermudah *programmer* untuk membuat aplikasi *Java* dengan menerapkan *design-pattern* yaitu *dependency-injection.*

Design-pattern menggunakan MVC (*Model View and Controller*), yaitu dimana project dipisahkan menjadi 3 *packages* utama yaitu *model, view* dan *controller,* hal ini bertujuan untuk memudahkan membuat sebuah *data flow* dari aplikasi mulai dari membuat *object* dan kemudian diteruskan dan digunakan di beberapa *class* yang membutuhkan. Berikut MVC yang telah diterapkan dalam aplikasi *web* ini.



Gambar 6 MVC *website* Eey.Id

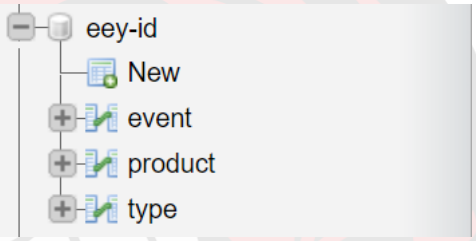
b. *Model*

Dalam *design-pattern* pada penerapan *model, view* dan *controller* tidak hanya terletak dalam satu *package*. Seperti *model* itu sendiri, bagian *model* dimulai dari *packages Entities, Interfaces,* dan *Repositories*, kemudian bagian *View* terdiri dari *Packages Services* dan terakhir bagian *Controller* terdiri dari *PackagesController*. Di bagian ini akan dijelaskan isi dari *design-pattern* ini sendiri dan fungsinya.

Gambar 7 Bagian *Model*

Package Entities merupakan sebuah *class* yang dibuat dari penggambaran table di dalam *database* yang kemudian diubah kedalam *project* menjadi sebuah *class*, sehingga segala atributnya dapat diakses dengan menggunakan *object* yang di buat dari *class*nya tersebut.

Dalam hal ini ada tiga *class*, yaitu *Class Event*, *Class Product*, dan *Class Type*. Semua kelas ini berasal dari *database* yang di *import* ke dalam *project* ini, dan keunggulannya sudah dilengkapi dengan *framework Hibernate ORM (Object Relational Mapping)*. Hal ini memudahkan *developer*, karena seluruh table di dalam *database* sudah saling terhubung di dalam *project* ini tanpa perlu membuat *query*. Berikut isi dari *database* Eey.Id.



Gambar 8 *Database* Eey.Id

4.3. *Entities*

Isi dari *package entities* merupakan penerapan langsung dari *database* menggunakan nama tabel yang ada didalam *database* dan digunakan sebagai *class* dalam *project* aplikasi.

Sehingga *function* dan atribut yang ada di dalam kelas-kelas tersebut dapat digunakan melalui metode pemanggilan objek dalam *class-class* yang memerlukan penggunaan dalam data yang ada di dalam kelas sebagai duplikasi dari tabel dalam *database*. Berikut bagian kecil sebagai contoh dari *code* isi database diimplementasikan ke dalam *class*.

Kode Program 1 Entitas *Database* Menjadi *Class*

```
@Entity
@Table(name = ”Product”)
....
Public class Product implements Serializable {
        @Id
        @GeneratedValue(strategy = GenerationType.IDENTITY)
        @Basic(Optional = false)
        @Column(name = ”id”);
....
}
```

Nama tabel *product* digunakan sebagai nama *class*, dimana setiap atribut yang ada dalam tabel tersebut, akan dijadikan *method* dan *function* agar dapat diakses, dengan membuat *class* tersebut sebagai *object.*

4.4. *Repository*

Dalam *Repository* disusun sebuah *class* dengan menggunakan pewarisan *(extends)* dari *library* atau *dependency* yang telah dimasukkan dalam aplikasi.

Class Repository ini menggunakan anotasi (*@Repository*) hal ini bertujuan agar *kelas* ini dapat menggunakan hal-hal khusus yang ada di dalam *repository.*

Pewarisan yang digunakan berasal dari *library* yaitu merupakan *class* dengan nama *CrudRepository*, *class* ini berisi DAO *(Data Array Object)* dan *query* yang dapat digunakan untuk mengakses data yang ada di dalam *database* sesuai dengan kegunaan yang telah disediakan, hanya perlu dilakukan pemanggilan melalui *class* yang telah diubah menjadi *object* untuk setiap *function* atau *method* yang ada. Berikut gambar dari *class repository*.

```
@Repository
public interface ProductRepo extends CrudRepository<Product, String> {

    @Query(value = "SELECT * FROM `product` WHERE `id`= ?", nativeQuery = true)
    public Optional<Product> findById(int id);

//    @Query(value = "SELECT `image` FROM `product` WHERE `id`= ?", nativeQuery = true)
//    public byte[] findImage(int id);
}
```

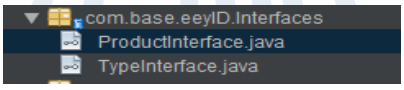
Gambar 9 *Class Repository*

Dalam *repository* ini terdapat *custom query*, yaitu *query sql* dengan pembuatan yang berbeda dari yang telah *library* siapkan atau sediakan dan dimasukkan dalam sebuah *method* yang dibuat.

Hal ini bertujuan untuk penggunaan yang khusus dan *query* tersebut tidak tersedia di dalam *library* atau *class CrudRepository*.

4.5. *Interfaces*

Interfaces digunakan layaknya *interface* pada *Java*, yaitu sebuah *class abstract* dengan *method-method* tanpa isi. *Method* ini kemudian akan digunakan dan diisi dengan *function* dari *library* dalam *Repository*. Berikut gambar dari *Packages Interfaces.*



Gambar 10 *Packages Interfaces*

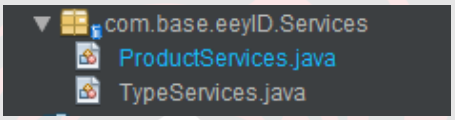
Dalam *Packages* ini ada *class* dengan nama *Product Interface*, didalamnya terdapat *method-method* tanpa isi. Hanya mendeklarasikan tipe data, *class* dimana data akan diakses dan nama dari *method* tersebut. Berikut gambar dari isi *class* tersebut.

```
package com.base.eeyID.Interfaces;

import com.base.eeyID.Entities.Product;
import java.util.Optional;
import javax.servlet.http.HttpServletResponse;

/**
 *
 * @author Dhanuaji Pratama
 */
public interface ProductInterface {

    public Iterable<Product> getAll();

//    public byte[] findImage(int id);

    public Optional<Product> findById(int id);

}
```

Gambar 11 *Class Product Interface*

4.6. *View*

Bagian *View* berisikan *packages Services*, *packages* ini berisikan *services* atau fitur yang dibuat sesuai dengan kebutuhan yang diperlukan dalam aplikasi *web* yang dibuat. Berikut gambar dari isi *Packages Services*



Gambar 12 *Packages Services*

Dalam *Packages Services* terdapat *class-class* yang akan digunakan sesuai dengan pengkhususan dan penggunaan masing-masing, disini terdapat *Class ProductServices* dimana di dalam *class* tersebut terdapat fitur untuk menghadirkan semua data di dalam *database* menggunakan salah satu *method* yang ada didalam *CrudRepository*, yaitu *method GetAll* untuk menampilkan seluruh isi data yang didalam tabel.

Penamaan *method* dan juga tipe datanya harus sesuai dengan *method* dan tipe data yang sebelumnya sudah dibuat di dalam *Class Interfaces.* Berikut *code* dari isi *Class ProductServices.Java.*

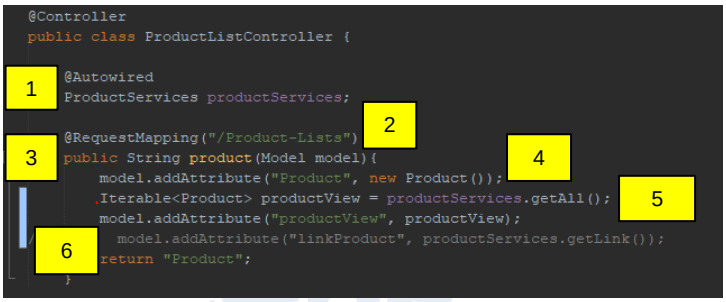
Kode Program 2 *Product Services*

```
@Service
Public class ProductService implements ProductInterface
@Autowired
ProductRepo productRepo;

@Override
Public Iterable<Product> getAll() {
       Return productRepo.findAll();
```

4.7. *Controllers*

Packages Controller berfungsi untuk mengatur *mapping* atau pemetaan data yang akan ditampilkan dan diletakkan dimana saja yang dibutuhkan, *class-class* yang ada pada *Controllers* sarat akan pengalamatan halaman *website* dan juga

nama-nama *object* yang digunakan. Berikut gambar dari isi *Class ProductListController.Java.*



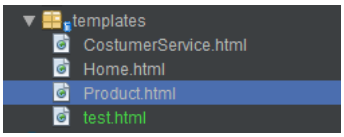
Gambar 13 *Class ProductListController*

Akan dijelaskan isi dari *kode* diatas, berikut penjelasannya :

1. Jika dalam OOP baris *code* ini, merupakan penginialisasian sebuah *class* menjadi *object* agar setiap *method* yang ada dalam *class* tersebut dapat diakses dan digunakan untuk kebutuhan yang ada dalam *class* tersebut.
2. *Annotation RequestMapping* (@RequestMapping) digunakan untuk mengatur nama *mapping* (pemetaan) untuk sebuah alamat url sebuah halaman *website*, halaman ini akan dipanggil dan dikenali dengan nama *Product-Lists*.
3. Nama *method*-nya adalah *Product* dan tipe datanya merupakan *String*, kemudian dilakukan pemanggilan *Library* berupa *Class* dengan nama *Model* dan diinisialisasikan sebagai *Object model*. Hal ini dilakukan agar *function* yang ada di dalam *Library* tersebut dapat digunakan.
4. Menginisialisasikan nama dari *class Product* menjadi *object Product*, yang nantinya akan dipanggil dan digunakan untuk mengambil nilai di dalam *Class HTML*.
5. Tahap ini kemudian memanggil dan menggunakan *function* yang ada di dalam *model*, dan menggunakan *method* atau *function* yang ada di dalam *Class ProductServices* yang telah diubah menjadi *object*.
6. Kemudian setiap isi serta *method-method* yang digunakan akan dikembalikan nilainya dalam hal ini akan digunakan dan ditampilkan dalam sebuah halaman *HTML* dengan nama halaman "*Product*".

4.8. *Templates*

Packages Templates berisi halaman-halaman *HTML*, yang dibuat sesuai dengan apa yang akan ditampilkan di halaman *website*. Dalam hal ini seluruh desain *Front-End* fungsi *Back-End*, dan nilai serta data-data yang akan dipanggil nantinya. Berikut isi dari *Packages Templates*.



Gambar 14 *Packages Templates*

Packages ini berisi 4 *class HTML* yang berbeda setiap halamannya dan menampilkan isi dari data-data yang telah diatur dari *Controller* setiap *Class HTML*. Berikut isi dari *Class Product.html.*

```
<html lang="en" xmlns:th="http://www.thymeleaf.org">

    <head>
        <title>All Products</title>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <link rel="icon" type="image/png" th:href="@{/img/Logo EEY Dasar Hitam.png}">

        <link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Open+Sans:300,400,600">

        <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
        <link rel="stylesheet" href="css/bootstrap.min.css">

        <link rel="stylesheet" href="slick/slick.css">
        <link rel="stylesheet" href="slick/slick-theme.css">
        <link rel="stylesheet" href="css/magnific-popup.css">
        <link rel="stylesheet" href="css/tooplate-style.css">
        <style>
```

Gambar 15 *Class Product.html*

```
<div class="row">
        <table>
        <thead>
            <tr>
                <td th:style="'background-image:url('+ ${productView.image}+');'"></td>
            </tr>
        </thead>
    </table>-->
    <div class="column" th:each="productView : ${productView}">
        <div class="card card-body tm-bg-dark" style="color: #F3F3F3">
            <h3 th:text="${productView.name}"></h3>
            <div>

            </div>
            <img th:src="@{'image/display/' + @{productView.image}}">
            <!--<img alt="img" th:src="*{''+{productView.image}}">-->
            <p th:text="${productView.id}" hidden=""></p>
            <p th:text="${productView.name}" hidden=""></p>
            <p th:text="${productView.stock}" hidden=""></p>
            <p th:text="${productView.event.name}" hidden=""></p>
            <h5 th:text="${productView.type.name}"></h5>
            <a th:href="@{'detail/' + ${productView.id}}">
                <!--<button>check</button>-->
            </a>
            <h6>
                <a th:href="@{https://shopee.co.id} + ${productView.link}">Shop Now</a>
            </h6>
        </div>
    </div>
```

Gambar 16 *Data* yang akan ditampilkan di *Product.Html*

Pada gambar 15 merupakan pengaturan awal *website*, pada bagian ini ditentukan *xml* yang digunakan, untuk dapat menggunakan *thymeleaf* maka xml standar dimodifikasi dan ditambahkan *thymeleaf* di dalamnya. Lebar halaman ditentukan sesuai dengan perangkat yang digunakan untuk mengakses halaman ini, dimasukkan juga *icon* sebagai logo dari brand Eey.Id, dan kemudian ditambahkan *library* baik secara online (mengambil *library* dengan mengakses *web* yang menyediakan *library* tersebut) atau offline (mendownload *library* dan diakses langsung dari aplikasi ini).

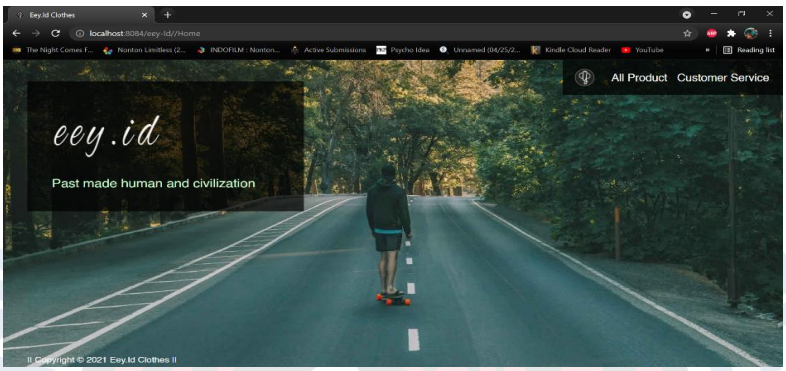
Pada gambar 16 ditentukan desain dari *website*, dalam hal ini yang ditentukan adalah desain *style* dari kolom, baris dan juga fitur *card* (kartu) yang digunakan sebagai wadah dari isi konten.

Pada gambar 16 pengaturan tabel, baris serta kolom dan pemanggilan data-data dari yang sudah ditentukan dari *Class Controller*. Disini *object* serta *method* yang sudah ada dalam *controller* dimasukkan dan dipanggil untuk digunakan kembali.

Serta dibuat tombol *link* untuk menghubungkan ke *website* Shopee sebagai *platform* untuk pembelian dan pembayaran. Hal ini sengaja dilakukan karena pemilik bisnis menginginkan peningkatan penjualan pada akun Shopee brand Eey.Id tersebut.

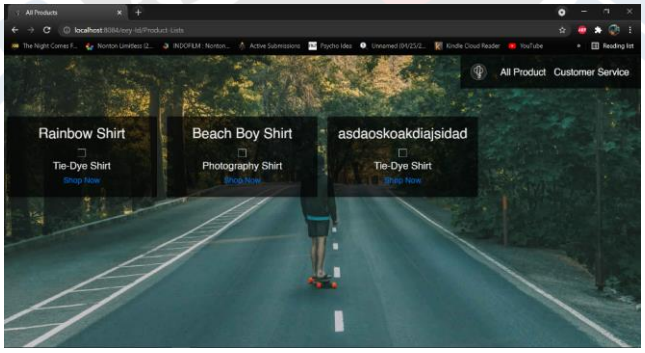
4.9. Tampilan *Website* Melalui Desain HTML

Tampilan awal atau *Home* dialamatkan melalui *Class Controller*, isi *code* dari *Class HTML* yang kemudian akan menentukan desain dan juga bentuk serta tatanan dari *website*. Berikut gambar dari *website Home* Eey.id.



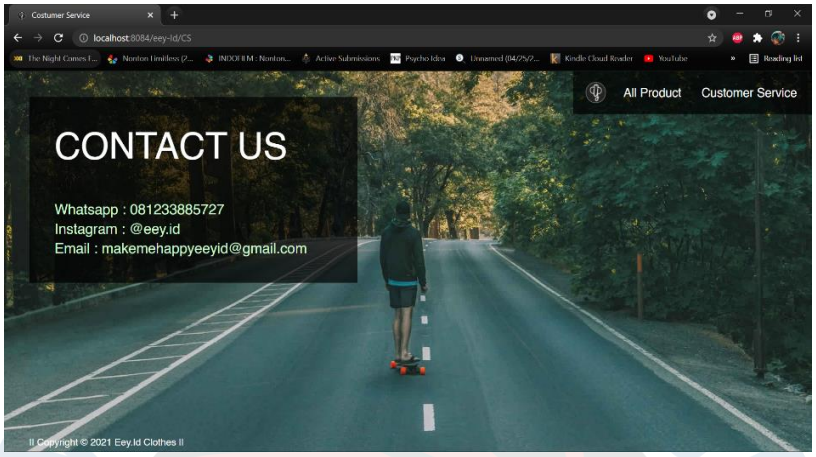
Gambar 17 Tampilan pada *Home* Eey.Id

Kemudian pada tampilan "*All Product*" ditampilkan jenis produk apa saja yang ada, nama produk, serta jenis eventnya. Kemudian di bagian *footer* dari *card* tersebut terdapat sebuah *link* untuk masuk ke *website* Shopee sebagai tempat untuk pemesanan dan pembelian, sehingga *website* ini hanya sebagai pusat informasi dan ragam produk.



Gambar 18 Tampilan pada halaman *Product*

Pada halaman terakhir berisi bagaimana cara *customer* dapat menjangkau yang bertanggung jawab terhadap produk-produk Eey.Id sehingga dicantumkan kontak dan *platform* yang dapat dijangkau oleh para *customer*. Berikut gambar dari halaman *Customer Service.*



Gambar 19 Halaman *Customer Service*

6. Simpulan

Dari penelitian yang dilakukan, ditemukan sebuah kesimpulan dimana suatu bentuk keabsahan produk dari segi merk atau *brand* menjadi salah satu faktor penjualan yang repetitif, hal ini juga dapat dibantuk dengan program pemasaran dan promosi yang efektif.

Selain faktor-faktor penunjang tersebut sebuah kemasan yang menarik untuk dihadirkan kepada masyarakat untuk dapat mengetahui sebuah nilai dari produk yang sudah ada.

Hal ini dapat dihadirkan melalui sebuah *website* yang dapat menampung semua secara keseluruhan dengan tingkat keefektifan yang cukup baik, karena *platfrom web* yang tidak hanya berjualan produknya tetapi sekaligus mengenalkan nilai-nilai dari sebuah produk akan menjadi lebih dihargai dan dikenal.