PENERAPAN DESIGN PATTERN DALAM PERANCANGAN WEB ORDER

SANDY KOSASI, DAVID
[1]Program Studi Sistem Informasi, [2]Program Studi Teknik Informatika
Sekolah Tinggi Manajemen Informatika dan Komputer Pontianak
Jln. Merdeka No. 372 Pontianak, Kalimantan Barat
[1]E-mail: sandykosasi@yahoo.co.id dan sandykosasi@stmikpontianak.ac.id
[2]E-mail: David_Liauw@yahoo.com dan David_Liauw@stmikpontianak.ac.id

ABSTRACT

This research using design pattern applied in resolving the issues raised in the booking process which is in the process of recording the menu order and payment information resulting in less accurate. The analysis tools are BPMN, Use Case Diagram, Sequence Diagram, User Interface Diagram, Class Diagram, Component Diagram, and Deployment Diagram. This research produces web-based presentation layer using HTML, JavaScript, JQuery, JQueryUI, JSPX, Apache Tiles and Spring Web MVC. Business logic layer (backend) is developed using Java technology and the Spring Core. The data access layer was developed using Spring Data JPA, Hibernate JPA Provider, and the MySQL Server. Deployment performed on the Cloud Foundry using cloud infrastructure. The proposed web order system facilitates precise, accurate, booking and fast payment.

***Keywords*: Design pattern, Web Order, Java, MVC, BPNM, OOP**

INTISARI

Penelitian ini menerapkan Design pattern dalam menyelesaikan permasalahan yang terjadi di dalam proses pemesanan yaitu pada proses pencatatan menu pesanan dan pembayaran yang menghasilkan informasi yang kurang akurat. Alat analisis yang digunakan antara lain BPMN, Use Case Diagram, Sequence Diagram, User Interface Diagram, Class Diagram, Component Diagram, dan Deployment Diagram. Penelitian menghasilkan presentation layer berbasis web dengan menggunakan HTML, JavaScript, JQuery, JQueryUI, JSPX, Apache Tiles, dan Spring Web MVC. Business logic layer (backend) dikembangkan menggunakan teknologi Java dan Spring Core. Data access layer dikembangkan dengan Spring Data JPA, JPA Hibernate Provider, dan MySQL Server. Deployment dilakukan pada infrastruktur cloud yaitu Cloud Foundry. Sistem web order yang diusulkan memudahkan dalam pemesanan dan pembayaran yang cepat, tepat dan akurat.

***Kata Kuncl* : Design pattern, Web Order, Java, MVC, BPNM, OOP**

PENDAHULUAN

Dalam mengembangkan suatu perangkat lunak berbasis web terkadang memunculkan permasalahan yaitu diperlukannya pemisahan halaman logika bisnis dan halaman tampilan presentasi. Myer (2008) menjelaskan bahwa pengembangan perangkat lunak berbasis web mungkin kurang disadari kelemahan penggabungan halaman logika dan halaman presentasi tampilan, karena memang logika yang digunakan masih sederhana dan biasanya hanya dikerjakan oleh satu orang. Pada pengembangan perangkat lunak berbasis web, penyatuan ini akan berakibat fatal ketika ada perubahan logika yang berdampak pula pada perubahan tampilan. Atau sebaliknya perubahan secara radikal yang memaksa programmer untuk mengubah keseluruhan halaman, termasuk logika pemrograman yang telah dibuat menyatu dengan tampilan.

Berkembangnya penggunaan pemrograman berorientasi objek juga menstimulasi timbulnya berbagai ragam pola pemecahan masalah dengan tetap berbasis Object Oriented. Pola pemecahan masalah yang dibuat tentu tidak dapat menyelesaikan semua masalah yang di hadapi, namun lebih berperan sebagai solusi untuk masalah yang spesifik. Pola pemecahan untuk masalah yang spesifik ini seringkali digunakan secara berulang-ulang oleh programmer untuk menyelesaikan permasalahan yang sama pada waktu yang berbeda. Pola pemecahan masalah ini yang kemudian dalam OOP dinamakan Design Pattern.

Design Pattern dibuat sesuai dengan kebutuhan programmer terhadap sekumpulan pustaka pemrograman dan komponen-komponen lain untuk menyelesaikan masalah pemrograman yang dihadapi secara berulang-ulang. Biasanya sebuah framework dibangun berbasis Design Pattern tertentu. Salah satu pola perancangan yang dianggap paling sesuai dengan arsitektur perangkat lunak berbasis web yaitu pola perancangan MVC yang memisahkan perangkat lunak ke

dalam tiga bagian yaitu Model, View dan Controller (MVC).

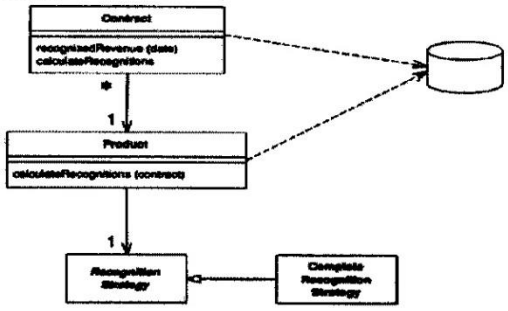
Restoran Fajar merupakan usaha dagang yang bergerak di bidang jasa pelayanan dan penyajian makanan. Dengan penerapan Design pattern dalam perancangan sistem pemesanan berbasis web, diharapkan Restoran Fajar dapat meningkatkan jumlah pendapatan dari makan di tempat dan memberikan kelebihan-kelebihan, seperti pencatatan transaksi, pencarian data maupun pembuatan laporan. Adapun sistem pemesanan berbasis web dikembangkan dengan menggunakan bahasa pemrograman Java. *Hosting* pengujian aplikasi dilakukan melalui Google App Engine yang menggunakan infrastruktur Cloud Computing. Dengan demikian, database dan infrastruktur lain yang dipakai dalam pengujian adalah yang disediakan oleh Google App Engine (GAE) untuk pengembangan aplikasi di platformnya, yaitu penggunaan GQL untuk menggantikan SQL.

Landasan Teori

Nock (2003:2) mengatakan bahwa "Pola perancangan (*design pattern*) tidak mendefinisikan kode program dan tidak spesifik pada domain pemograman tertentu." Pola perancangan (*design pattern*) menyediakan rancangan serupa yang terbukti dan teruji menyelesaikan permasalahan yang mirip. Pola perancangan juga menyediakan istilah umum yang dapat dipakai dalam perancangan untuk mempermudah dokumentasi dan pemahaman.

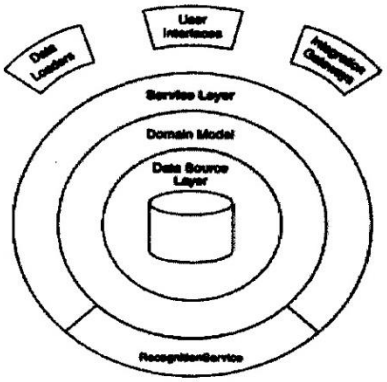
Buschmann (1996:31) mengatakan bahwa "Pola arsitektural *layer* membantu menstrukturisasi aplikasi sehingga aplikasi dapat dibagi ke dalam kumpulan sub-kelompok tugas dimana setiap kelompok tugas berada dalam level abstraksi tersendiri."

Fowler (2002:128) menyatakan bahwa "Domain Model membuat sekumpulan objek yang saling berhubungan, dimana setiap objek mewakili individu yang berarti, baik sebesar sebuah perusahaan ataupun sekecil sebuah item tunggal dalam form pemesanan." Sebuah domain model berorientasi objek umumnya terlihat mirip seperti model database, tetapi memiliki banyak perbedaan. Sebuah Domain Model menggabungkan data dan proses, memiliki atribut dengan banyak nilai dan asosiasi yang kompleks, dan memakai *inheritance* (pewarisan). Fowler (2002:130) lebih lanjut menambahkan bahwa hampir semua buku tentang perancangan berorientasi objek membahas Domain Model, karena apa yang umumnya dirujuk oleh orang-orang sebagai pengembangan berorientasi objek berpusat pada penggunaan Domain Model. Gambar 1 memperlihatkan sebuah contoh *domain model*.



Gambar 1. Domain Object Pattern

Fowler (2002:143) menyatakan bahwa "*Service layer pattern* mendefinisikan batasan aplikasi dengan sebuah lapisan layanan yang mencakup operasi yang tersedia dan mengkoordinasi respon aplikasi dalam setiap operasi." Gambar 2 memperlihatkan struktur *service layer pattern*.Sebuah *service layer* mencakup *business logic* aplikasi, mengendalikan transaksi dan koordinasi reponse dalam implementasi operasinya. Fowler (2002:144) lebih lanjut menyatakan bahwa *business logic* secara garis besar dapat dibedakan menjadi dua, yaitu *domain logic* (murni hanya berhubungan dengan domain permasalahan) dan *application logic* (berkaitan dengan tugas aplikasi).*Application logic* umumnya disebut sebagai *workflow logic*.Manfaat *service layer* adalah terdapatnya kumpulan operasi aplikasi yang umum dan dapat dipakai oleh lebih dari satu *client*.



Gambar 2. Service Layer Pattern

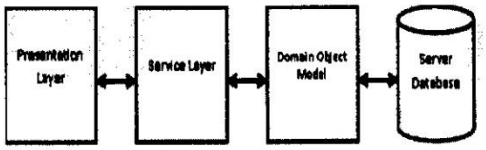
Menurut Fowler (2002:321), m*odel view controller pattern* adalah sebuah pola perancangan antar-muka dimana terdapat tiga peran yaitu model, view, dan controller.

Model adalah objek non-visual yang mewakili informasi mengenai domain. Umumnya, model adalah sebuah *domain model*.View mewakili tampilan atas model dalam UI. View hanya menampilkan informasi saja. Controller bertanggung jawab atas penanganan input dari user, manipulasi model, dan memperbaharui view.

PEMBAHASAN

Gambar 3 memperlihatkan *layer* yang ada pada sistem usulan. Setiap layer dalam sistem usulan berada pada infrastruktur fisik yang sama di platform Cloud Foundry. Pola perancangan (*design pattern*) yang dipakai dalam sistem usulan antara lain:

1. Untuk arsitektur secara umum, sistem usulan memakai pola rancangan arsitektur berlapis (*layered architecturedesign pattern).*
2. Untuk *domain logic*, sistem usulan memakai pola rancangan *domain model* dan pola rancangan *service layer* seperti yang dikemukakan oleh Martin Fowler.
3. Untuk pola arsitektur akses data (*data source architectural pattern*), sistem usulan memakai pola rancangan *data mapper.*
4. Untuk pola presentasi web (*web presentation pattern*), sistem usulan memakai pola perancangan Model View Controller (MVC).

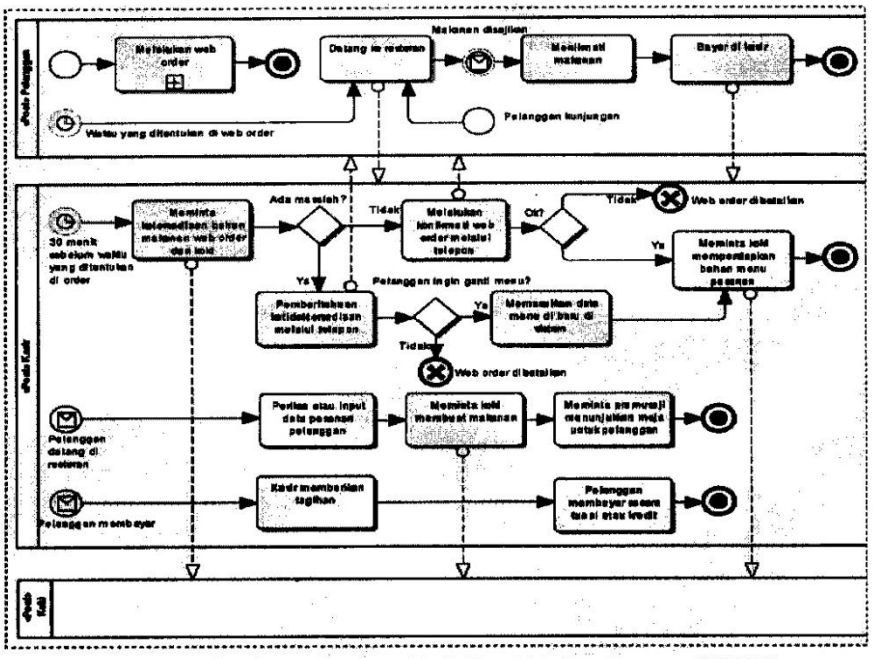


Gambar 3. *Layer Architectural Pattern* Untuk Sistem Usulan

Sistem usulan memiliki lapisan sebagai berikut: 1) *Presentation layer* menyediakan tampilan antar-muka aplikasi. Implementasi *presentation layer* dilakukan dengan menerapkan pola perancangan MVC melalui penggunaan teknologi JSP dan Servlet; 2) *Service layer* menyediakan layanan yang bertanggung jawab atas operasi *business logic*; dan 3) *Domain object model* terdiri atas class-class yang mewakili permasalahan bisnis.

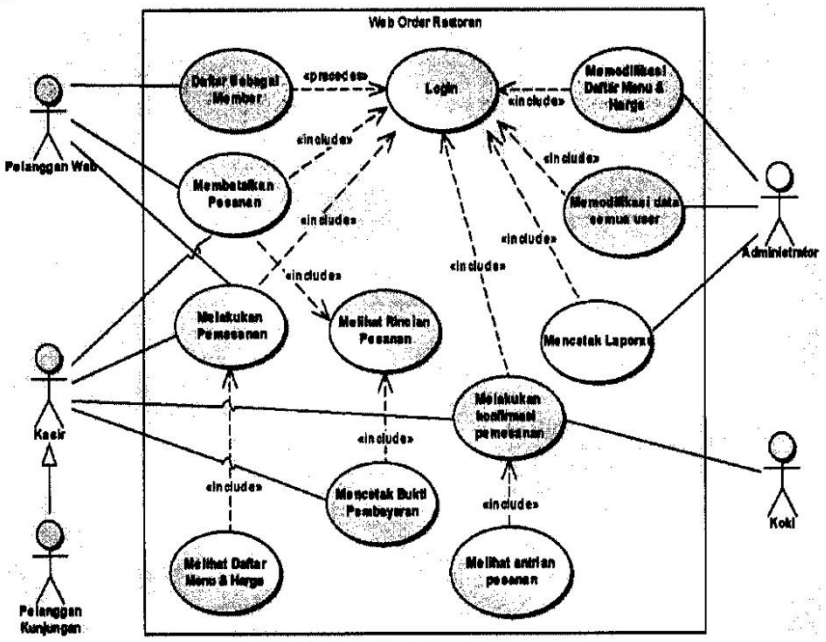
Implementasi *service layer* dilakukan dengan membuat *interface* serta implementasinya (*service by interface*). Alasan penggunaan arsitektur seperti ini adalah 1) Mempermudah pengujian *unit testing* dengan JUnit karena *service by interface* memungkinkan membuat implementasi *prototype* khusus untuk pengujian (*mocking*). Sebagai contoh, controller memakai UserService (sebuah *interface*) dengan implementasi berupa UserServiceImpl pada saat penggunaan sehari-hari yang menyimpan data secara langsung di database. Tetapi pada saat pengujian, *controller* memakai UserService dengan implementasi berupa UserServiceTestImpl yang berupa prototipe dan tidak mengakses database secara langsung sehingga database tidak berubah/rusak pada saat pengujian *controller*; dan 2) Mengurangi keterikatan antara *service layer* dan *presentation layer*, sehingga sebisa mungkin perubahan pada *service layer* tidak banyak berdampak pada *presentation layer.*

Langkah-langkah analisa yang diterapkan dalam perancangan sistem usulan antara lain: mengembangkan proses bisnis baru yang sudah terkomputerisasi untuk Restoran Fajar Pontianak berdasarkan proses bisnis yang sudah ada dan analisa permasalahan. Proses bisnis ini dituangkan dalam bentuk diagram BPMN (*Business Process Modelling and Notation*) yang dapat dilihat pada gambar 4.



Gambar 4. *Business Process Modelling NotationDiagram* (BPMN)

Berdasarkan hasil analisa proses bisnis, penulis membuat *UML Use Case Diagram* yang dapat dilihat di Gambar 5.



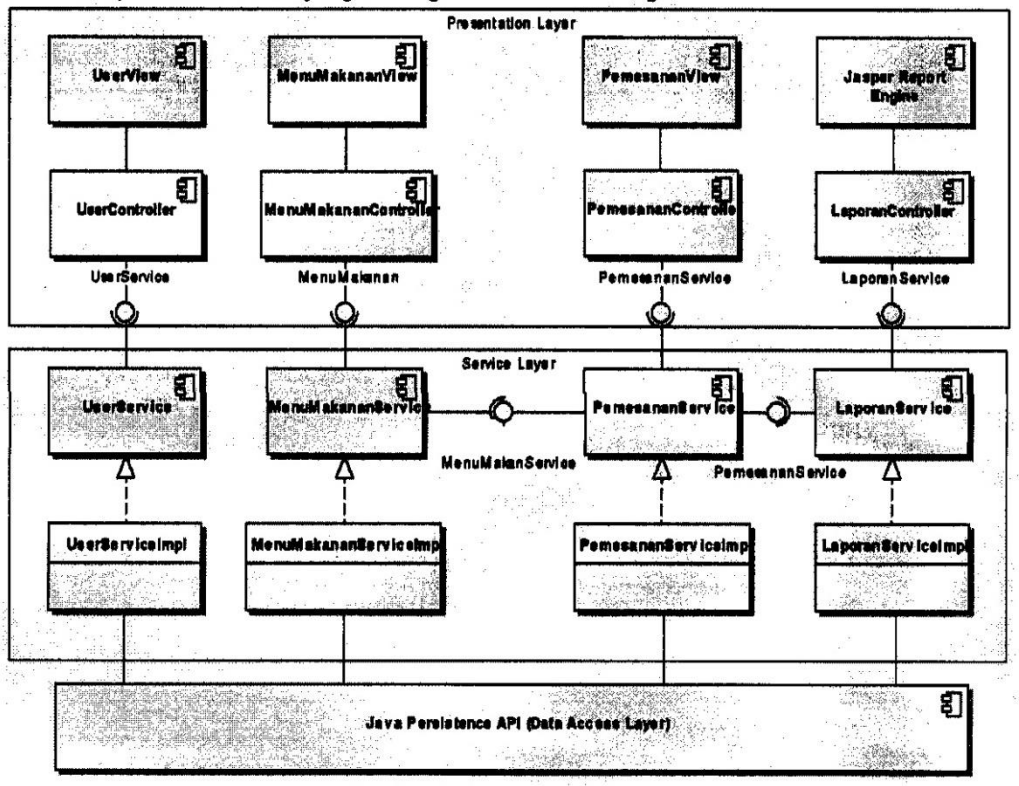
Gambar 5. Use Case Diagram

Berdasarkan *UML Use Case Diagram*, dilakukan analisis untuk menemukan *domain model* yang mewakili objek yang terlibat dalam sistem. Domain model ini dituangkan dalam bentuk *UML Class Diagram*. *UML Class Diagram* tersebut perlu dikembangkan untuk mencakup atribut dan operasi setelah melakukan analisa lebih mendalam. Dalam

penelitian ini dipisahkan operasi untuk setiap *domain model* ke dalam sebuah layer tersendiri yang disebut sebagai *service layer*. Hal ini dilakukan untuk meningkatkan *reusability* operasi. Hasil analisa operasi pada *service layer* dituangkan dalam bentuk *UML Sequence Diagram*.

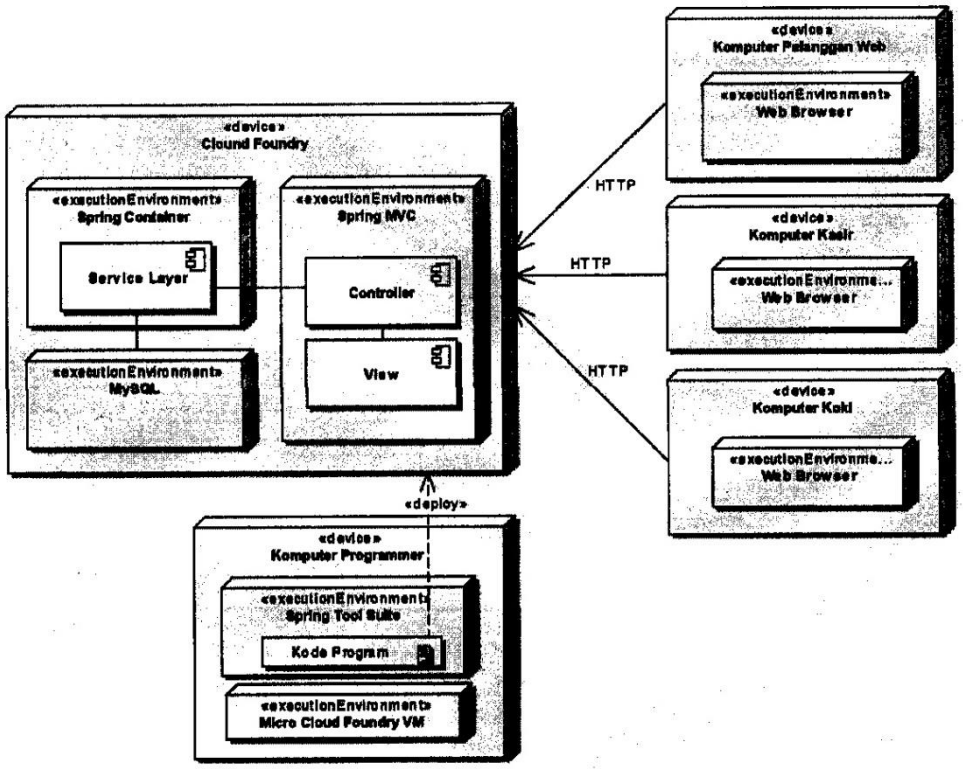
Penulis melakukan analisa lebih lanjut pada bagian *presentation layer* yang ada di *UML Sequence Diagram*. Hasil analisa ini dalam bentuk tampilan antar-muka yang dituangkan dalam bentuk *User Interface Diagram* (salah satu *extension* dari UML sebagai bagian dari *EA's Core Extension*).

Setelah mendapatkan rancangan detail pada *UML Class Diagram* dilakukan analisis bagaimana mengimplementasikan setiap *class* yang ada ke dalam komponen *POJO (Plain Old Java Object)* atau *JavaBean*. Hasil analisa ini dituangkan dalam bentuk *UML Component Diagram* seperti yang terlihat di gambar berikut.



Gambar 6. *Component Diagram*

Penulis melakukan analisa kebutuhan perangkat keras yang hasilnya dalam bentuk *UML Deployment Diagram* yang dapat dilihat di Gambar 5.28.



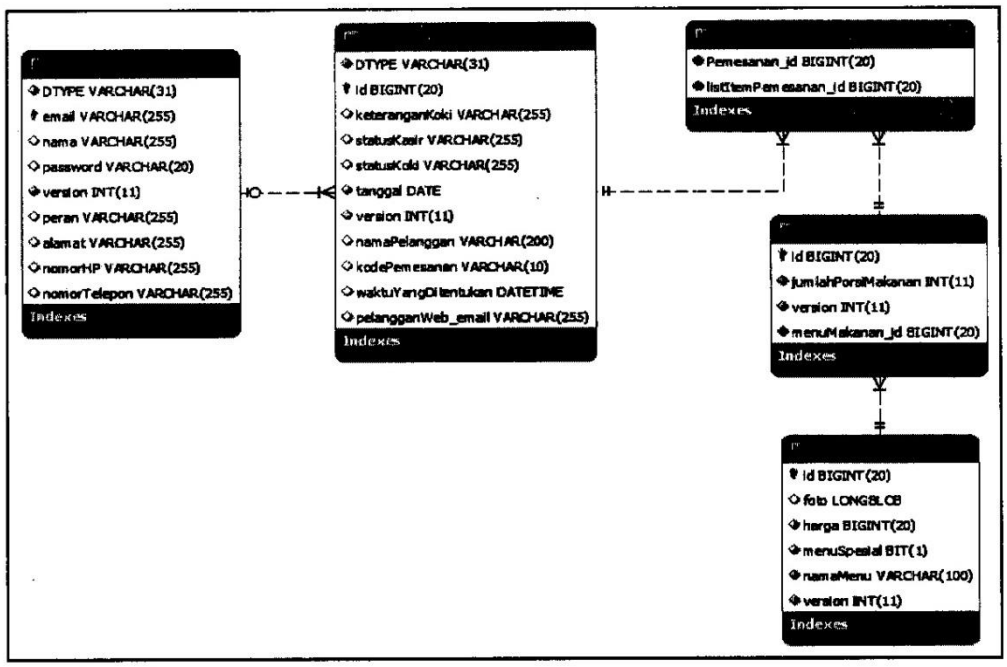
Gambar 7. *Deployment Diagram*

Penelitian ini menggunakan database relasional MySQL Server sebagai media untuk menyimpan setiap objek *domain model*secara permanen. MySQL Server tidak dapat menyimpan objek secara langsung melainkan melakukan penyimpanan dalam bentuk tabel relasional. Untuk menjembatani perbedaan ini, penulis menggunakan kerangka kerja Object-Relational Mapping (ORM) Hibernate. Pemetaan (*mapping*) dari objek ke tabel relasional di Hibernate dapat dilakukan dengan menggunakan XML atau *annotation*. Penulis memilih untuk menggunakan *annotation* karena lebih mudah dipakai (misalnya, tidak perlu membuat file XML terpisah dan dapat langsung disertakan di *class*).

Pemetaan di Hibernate dapat dilakukan dengan mengikuti salah satu dari filosofi berikut ini, yaitu 1) Mulai dengan membuat skema database (seperti tabel, index, dan sebagainya), baru kemudian membuat *class* sesuai dengan skema database yang sudah dibuat; dan 2) Mulai dengan membuat *class* dan membiarkan Hibernate menghasilkan skema database secara otomatis.

Penelitian ini menggunakan rekomendasi Fisher & Murphy (2010:71) yang menyarankan untuk mendefinisikan pemetaan di class terlebih dahulu. Dengan berkonsentrasi pada class, penulis dapat berfokus pada pemograman berorientasi objek (OOP) tanpa perlu mengkhawatirkan perancangan tabel relasional yang pada dasarnya tidak sesuai untuk OOP.

Gambar 8 memperlihatkan skema yang dihasilkan oleh Hibernate dalam bentuk diagram hubungan entitas. Skema dihasilkan dengan menggunakan penamaan standar dari Hibernate. Untuk menyimpan relasi pewarisan (inheritance) OOP di domain model, penulis menggunakan strategi Single Table Per Class Hierarchy dimana terdapat sebuah kolom DTYPE yang berisi nilai discriminator untuk membedakan hierarki class. Hierarki pewarisan yang terdiri atas dua class atau lebih akan disimpan dalam sebuah tabel yang sama. Tujuannya adalah untuk meningkatkan kinerja persistensi data dimana Hibernate tidak perlu melakukan operasi join pada saat memakai class yang diturunkan.



Gambar 8. *Entity Relationship Diagram*)

Dalam implementasi tampilan sistem pemesanan web di Restoran Fajar, menggunakan kerangka kerja (*framework*) berikut ini : 1) Spring Web MVC untuk menerapkan pola perancangan Model View Controller (MVC) pada tampilan tunggal yang mewakili satu *use case*. Penulis memilih untuk menerapkan pola perancangan MVC dengan tujuan agar mempermudah strukturisasi *presentation layer* dan modifikasi di masa depan; 2) Spring Web Flow dipakai untuk lebih dari satu *view* yang mewakili satu *use case* seperti pada *use case* pemesanan makanan yang merupakan sebuah *flow* yang terdiri atas *view* berurut mulai dari *view* lihat menu, *view* pesan menu, *view* penentuan jadwal, dan *view* konfirmasi. Penulis memakai Spring Web Flow untuk menangani bila pengguna menekan tombol kembali (*back*) di *browser* selama dalam *flow*, pengguna tiba-tiba membuka halaman lain secara tidak berurut saat berada dalam *flow*, dan sebagainya; 3) JQuery dan JQuery UI untuk mempermudah dalam pengkodean JavaScript dan penanganan AJAX; 4) jqGrid untuk menampilkan data dalam bentuk tabel yang dilengkapi dengan fitur pembuatan halaman dan dukungan AJAX (misalnya, memperbaharui tabel tanpa harus memperbaharui seluruh halaman); dan 4) Jackson JSON Processor yang dipakai bersamaan dengan Spring MVC untuk menghasilkan data JSON yang dipakai oleh jqGrid.

Pada view tunggal yang mewakili sebuah *use case*, penulis berusaha menggunakan AJAX supaya setiap proses CRUD (Create, Read, Update, Delete) terhadap entitas tunggal tidak perlu memperbaharui keseluruhan halaman. Hal ini akan membuat tampilan lebih *responsive* dan tidak terlalu berat dalam koneksi ke server web. Gambar berikut ini memperlihatkan implementasi tampilan Sistem Web Order pada Restoran Fajar Pontianak:

Gambar 9. Tampilan *HomepPage*

Untuk meningkatkan keamanan pada sistem pemesanan web di Restoran Fajar, digunakan *framework* Spring Security. Fasilitas Spring Security yang digunakan antara lain: 1) *Authentication* yang terdiri atas fitur login dan logout; 2) *Authorization* untuk membatasi akses *resources* tertentu hanya pada user yang berwenang saja; 3) Fitur *remember-me* yang menyimpan informasi *authentication* user dalam bentuk cookie yang ter-enkripsi. Dengan fitur *remember-me*, pengguna dapat login secara otomatis pada komputer yang dipercaya tanpa perlu memasukkan password. Penulis mengatur fitur *remember-me* untuk melakukan *authentication* secara otomatis selama 1 minggu; 4) Perlindungan terhadap serangan *session fixation* dimana penyerang berusaha mencuri *session* dari user yang akan login; dan 5) *Conccurency control* dimana penulis melakukan konfigurasi pada Spring Security agar hanya membolehkan 1 user yang login secara bersamaan. Bila penyerang memperoleh password pengguna dan login pada saat yang bersamaan dengan pengguna, maka penyerang akan logout secara otomatis. Demikian juga, bila seseorang telah mencuri password pengguna dan sedang login, maka pengguna akan logout secara otomatis pada saat tersebut juga. Hal ini dapat dipakai sebagai dasar bagi pengguna untuk mencurigai seseorang sedang memakai akun-nya.

Penulis juga memakai Hibernate Validator yang mengimplementasikan JSR-303 Bean Validation (spesifikasi resmi Java) sehingga penulis dapat melakukan validasi data secara konsisten mulai dari *presentation layer* hingga ke *service layer* secara otomatis. Peraturan validasi (*validation rule*) hanya perlu diterapkan dalam bentuk *annotation* di *domain model*. Tanpa Hibernate Validator, untuk mencapai tingkat keamanan yang wajar, penulis harus membuat beberapa kode program terpisah yaitu kode program yang melakukan validasi di view (JavaScript), kode program validasi saat view memberikan nilai ke *controller* (Java), kode program validasi saat *controller* memberikan nilai ke *service layer*, dankode program validasi saat akan menyimpan domain model ke database.Penulis juga membuat implementasi *validator* untuk *validation rule* sesuai keperluan Sistem Web Order di Restoran Fajar.

Setiap catatan yang berhubungan dengan keamanan disimpan, seperti user yang login dengan password yang salah, dengan Apache Commons Logging dan Log4J. Penulis dapat menyimpan hasil catatan ke dalam file sehingga administrator dapat mengkaji ulang catatan tersebut bila diperlukan.

Penggunaan fitur *remember-me* yang menyimpan data *authentication* pada browser pengguna dapat menimbulkan celah keamanan tersendiri bila cookie berisi data dicuri oleh pihak tidak bertanggung jawab. Oleh sebab itu, untuk menyeimbangkan antara kenyamanan dan keamanan, penulis mensyaratkan pengguna untuk wajib memasukkan *password* (login secara manual) pada saat pengguna mengubah identitas diri (profile) dan melakukan pemesanan.

Pada kasus yang ideal, penulis dapat menerapkan perlindungan dengan memakai IP, dimana hanya komputer dengan IP tertentu saja yang dapat login sebagai administrator. Hal ini akan meningkatkan keamanan sistem informasi. Akan tetapi, fitur ini hanya dapat diaktifkan tergantung pada provider yang dipakai oleh Restoran Fajar. Bila Restoran Fajar memakai Internet Service Provider (ISP) yang menyediakan IP publik, maka penulis akan melakukan konfigurasi sehingga hanya komputer dengan IP tersebut yang dapat login sebagai administrator. Dengan demikian, bila penyerang berhasil mengetahui password administrator, penyerang tetap tidak dapat login selama ia tidak memakai IP yang sama dengan IP yang dimiliki oleh jaringan Restoran Fajar.

Untuk mencegah agar password tidak dapat dilihat oleh penyerang, penulis menerapkan algoritma enkripsi SHA1 pada setiap password yang tersimpan di database. SHA1 adalah algoritma *one-way encoding* sehingga akan sulit untuk membalikkan password yang telah terenkripsi ke dalam bentuk aslinya. Algoritma SHA1 dipakai karena algoritma SHA1 dianggap sebagai algoritma yang effisien dan sering dipakai dalam penyimpanan data. Untuk menghindari serangan *brute-force* pada password yang ter-enkripsi, penulis menambahkan *salt* berupa email pengguna sehingga biarpun dua pengguna memakai *password* yang sama, hasil enkripsi-nya akan selalu berbeda.

Penulis tidak menambahkan keamanan dengan menggunakan SSL karena Restoran Fajar harus membayar untuk memperoleh sertifikat digital. Selain itu, Restoran Fajar harus membuktikan diri dengan menyertakan identitas seperti nama penanggung jawab perusahaan, alamat perusahaan, dan sebagainya. Akan tetapi, bila Restoran Fajar telah memiliki sertifikat

digital, sistem pemesanan web order dapat memakainya tanpa banyak perubahan.

Proses authorization yang dipakai pada sistem web order Restoran Fajar adalah *fine-grained authorization*.

Untuk mencegah hal yang tidak diinginkan dimana penyerang berhasil melewati *presentation layer* dan menembus *service layer*, maka penulis juga menerapkan prinsip keamanan sistem informasi pada *service layer*. Spring Security menyediakan pengamanan pada *service layer* dengan menggunakan annotation @PreAuthorize pada setiap method di *service layer* yang akan diamankan.

Penulis melakukan pengujian *logic unit test* dengan menggunakan JUnit. Tujuan dari *unit test* adalah menguji sebuah *unit* tunggal secara terisolasi dari unit lainnya. Pengujian *unit test* pada *domain object layer* hanya dilakukan pada *domain object* yang memiliki perilaku (*behaviour*).

Penulis melakukan *integration test* pada *service layer*. Pengujian ini menggunakan database *in-memory* H2 *Database Engine* yang telah ter-integrasi dalam Spring. Tujuan pemisahan antara database yang dipakai dalam pengembangan perangkat lunak dan dalam pengujian adalah untuk menghindari ketidak-konsistenan database yang dapat terjadi bila terdapat kesalahan pada unit yang diuji.

KESIMPULAN

Penelitian menghasilkan presentation layer berbasis web dengan menggunakan *HTML, JavaScript, JQuery, JQueryUI, JSPX, Apache Tiles*, dan *Spring Web MVC. Business logi layer (backend)* dikembang dengan teknologi *Java* dan *Spring Core*. Data access layerdikembangkan dengan Spring Data JPA, JPA Hibernate Provider, dan MySQL Server.Deployment dilakukan pada infrastruktur cloud yaitu CloudFoundry. Sistem web order yang diusulkan memudahkan dalam pemesanan dan pembayaran yang cepat, tepat dan akurat. Rancangan sistem yang diusulkan masih sebatas pada perancangan prototipe dan memerlukan pengembangan lebih lanjut sebelum diterapkan. Sistem usulan ini hanya menitikberatkan pada masalah aplikasi pemesanan dan pembayaran sehingga perlu dikembangkan lebih lanjut sesuai kebutuhan di masa yang akan datang. Misalnya, sistem dapat dipadukan dengan pengendalian bahan baku sehingga pengecekan manual atas ketersediaan bahan makanan dapat dihindari. Penerapan sistem usulan ini diharapkan dapat mengatasi dan meminimalkan kendala-kendala yang dihadapi pada Restoran Fajar Pontianak sehingga dapat menyajikan informasi kepada pengelola serta pihak yang membutuhkan.

DAFTAR PUSTAKA

Buschmann, Frank, et al., 1996, *Pattern-Oriented Software Architecture*, Vol 1, New York, Wiley & Sons.

Fisher, Paul Tepper dan Murphy, Brian D, 2010, *Spring Persistence With Hibernate*, New York, Apress.

Fowler, Martin, et al., 2002, *Patterns of Enterprise Application Architecture*, Addison Wesley.

Ho, Clarence, dan Harrop, Rob, 2012, *Pro Spring 3*, New York, Apress.

Myer, Thomas., 2008,*Professional CodeIgniter (Wrox Professional Guides)*, Wrox

Nock, Clifton, 2003, *Data Access Patterns: Database Interactions in Object-Oriented Applications*, Addison Wesley.