IMPLEMENTASI MODEL-VIEW-CONTROLLER UNTUK PERANCANGAN APLIKASI CHATROOM

Sansan Yusuf Alamsyah[1], Dhami Johar Dhamiri[2], Cepy Slamet[3]

Jurnal Algoritma
Sekolah Tinggi Teknologi Garut
Jl. Mayor Syamsu No. 1 Jayaraga Garut 44151 Indonesia
Email : jurnal@sttgarut.ac.id

[1]1006115@sttgarut.ac.id
[2]dhami_dj@sttgarut.ac.id
[3]cepy_lucky@uinsgd.ac.id

Abstrak **–** Chatroom merupkan bagain dari perkembangan teknologi komunikasi yang menggantikan komunikasi secara langsung melalui pada perangkat smartphone, ini membuat salah satu fungsi komputer sebagai media komunikasi, dapat sedikit tergantikan karena smartphone sudah mampu menjalankan berbagai macam aplikasi dengan dukungan dari hardware dan sistem operasi yang baik. Saat ini banyak sekali metode pengembangan perangkat lunak yang ada, begitu juga dengan banyaknya bahasa pemograma. Ini memberikan keleluasaan dalam proses pengembangannya, MVC (Model-View-Controller) merupakan sebuah konsep pemograman yang memiliki struktur tersendiri dalam proses pengkodean yang memisahkan antara data dengan tampilan dan mengontrolnya secara terpisah. Metode pengembangan sistem prototyping merupakan metode yang menggunakan pendekatan untuk membangun suatu program dengan cepat dan bertahap sehingga segera dapat dievaluasi oleh pemakai. Pendekatan MVC ini mudah dikenal dan yang paling banyak diterima. Keuntungan utama dalam MVC ini adalah penggunaan ulang (reusability) kode karena MVC memisahan tampilan pengguna dari kendali asupan pengguna dalam model informasi yang mendasarinya. Dengan menggunakan metode MVC, kode pemograman aplikasi menjadi lebih rapih, karena ada pembagian yang jelas antara masing-masing kelas, mulai dari kelas model, view, controller. Dan komponen pada aplikasi tersebut dapat digunakan oleh aplikasi lain khususnya komponen model.

Kata Kunci – *Model-View-Controller, Chatroom*.

I. PENDAHULUAN

Saat ini komputer tidak hanya berfungsi sebagai alat pengolah data saja, namun telah menjadi media komunikasi. Pengaruh dari komputer pun ikut ambil peran dalam mempengaruhi proses komunikasi, karena proses komunikasi yang dilakukan melalui komputer terdapat proses pengi-riman paket data yang dikirim dari satu tempat ke tempat lain.

Komunikasi yang dapat dilakukan dimana pun tidak selalu bersipat aman dan nyaman, aman apabila sebuah aplikasi memiliki tingkat perlindungan yang tinggi dalam memproteksi siapa saja yang masuk ke ruang obrolan dan nyaman apabila aplikasi tidak disisipkan *bots* yang mengakibatkan banyaknya iklan. Ini dibuktikan dengan ditutupnya fasilitas *chatroom* yang terdapat pada Yahoo!Messenger akhir tahun 2012 karena terlalu banyak dimasuki oleh *boots* yang mengakibatkan munculnya iklan biasa sampai dengan berbau pornografi. Permasalahan lainnya yang membuat Yahoo!Messenger di tutup adalah yahoo *chatroom* juga sering digunakan oleh tim *marketing* untuk mempromosikan dagangannya dengan megikuti obrolan yang sedang ramai dengan menyelipkan *link* yang merujuk pada produk[1].

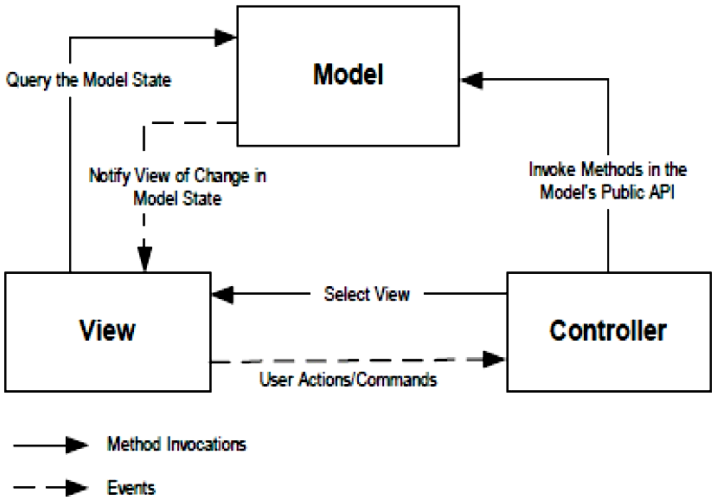
Untuk mengatasi kelemahan dari sebuah perangkat lunak, perlu adanya pengembangan sehingga meningkatkan kualitas dan kenyamanan, sementara pengembangan dan perbaikan dapat

dilakukan secara terus menerus apabila sebuah aplikasi dirancang dengan *design pattern atau* struktur pemograman yang baik sehingga dalam mengatasi *bug, error,* dan *maintenance* menjadi lebih mudah untuk dilakukan.

Maka untuk meningkatkan kualitas pengembangan perangkat lunak, diperlukan konsep MVC (*Model-View-Controller*), sehingga tercipta 3 layer yang berbeda yaitu *model, view* dan *controller*, maka proses *maintenance* suatu aplikasi akan semakin cepat karena penyederhanaan pemeliharaan kode dari setiap komponen dapat dikembangkan dan diperbaharui secara terpisah[2].

I. TINJAUAN PUSTAKA

MVC adalah pemisahan tampilan pengguna dari kendali asupan pengguna dalam model informasi yang mendasarinya [2]

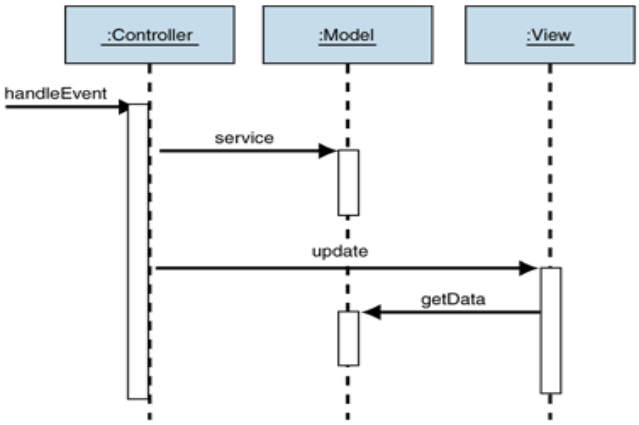


Gambar 2.3. Arsitektur MVC (Gulzar, 2003)

Definisi Dan fungsi teknis dari arsitektur MVC dibagi menjadi tiga lapisan yaitu *model view* dan *controller*[3]

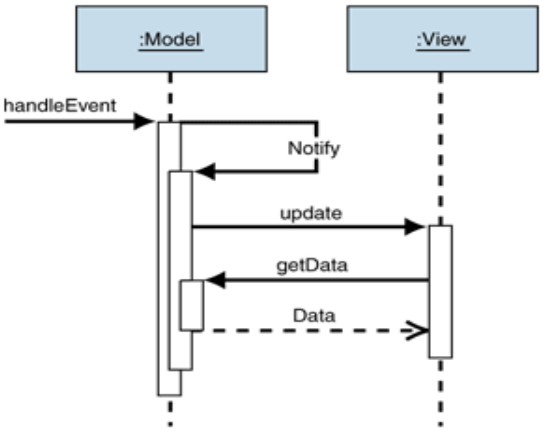
Model merupakan representasi data yang digunakan oleh aplikasi sebagai proses bisnis yang diasosiasiasikan terhadapnya. Tujuan dari MVC adalah untuk membuat model independen dari *view* dan *controller* yang bersama-sama membentuk *user interface* dari aplikasi. Model Dapat di bedakan berdasarkan sifatnya[4].

Model pasif digunakan ketika satu *controller* memanipulasi *model eksklusif*. *Controller* memodifikasi model dan kemudian menginformasikan ke *view* bahwa *model* telah diubah dan harus *refresh*.



Gambar 2.4 Behavior of the passive model[5]

Model aktif digunakan ketika model melakukan perubahan state tanpa melibatkan controller. ini dapat terjadi ketika object lain yang mengubah data dan perubahan akan terjadi pada *view*.



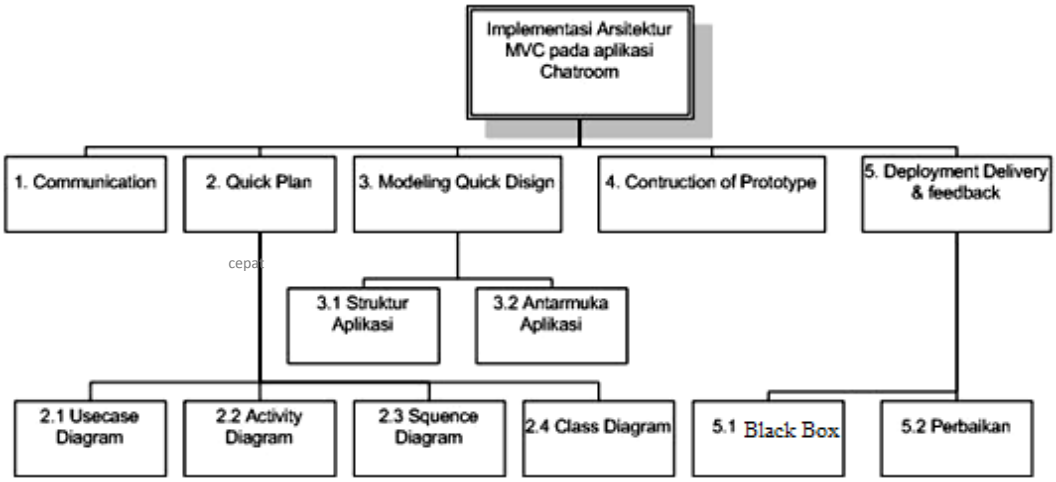
Gambar 2.5 Behavior of the active model[5]

Controller menerima dan menerjemahkan *input* dan *requests* pada *model* atau *view*. Controller biasanya bertanggung jawab untuk memanggil metode pada model yang mengubah keadaan model. Dalam model aktif, perubahan keadaan ini kemudian tercermin dalam view melalui mekanisme penyebaran perubahan. Dalam model pasif, *controller* bertanggung jawab untuk memberitahu *view* kapan harus memperbarui.

View memperoleh data dari model dan menampilkannya sebagai UI(user interface). Sehingga dapat dikatakan *view* sebagai output dari aplikasi. *View* umumnya merupakan implementasi dari model,

II. KERANGKA KERJA KONSEPTUAL

Pada proyek tugas akhir ini, dilakukan beberapa tahapan untuk mencapai tujuan yang direncanakan. Berikut merupakan alur kerja proyek pada penelitian berdasarkan Paradigma Prototypeing Pressman(2010)[6] digambarkan dalam *Work Breakdown Structure* (WBS) :



Gambar 1. Struktur Rincian Kerja (*Work Breakdown Structure*)

Berdasarkan pada *work breakdown structure* dan teori yang telah dikemukakan sebelumnya. *Communication* merupakan tahap pengumpulan kebutuhan perangkat lunak, dengan cara melakukan komunikasi dengan *stakeholder*. Sehingga dapat teridentifikasi aktifitas apa saja yang terlibat pada sistem yang akan dirancang, tujuan dari tahap ini adalah untuk mengidentifikasi

aktifitas apa saja yang terlibat maka akan dilakukan pemodelan *business process management.*

Quick plan merupakan tahap perancangan cepat, dengan memperhatikan aktifitas yang terjadi pada sistem, yang telah di komunikasikan pada tahap *communication*. Modeling quick design

Modeling quick design merupakan tahapan pembuatan rancangan cepat berdasarkan pada representasi aspek-aspek perangkat lunak yang akan terlihat oleh para *end user* seperti pembuatan *interface* dan struktur aplikasi sementara, sehingga aplikasi dapat terlihat, tahapan ini berdasarkan *sequence diagram* dan *aktivity diagram* dari tahap *quick plan*.

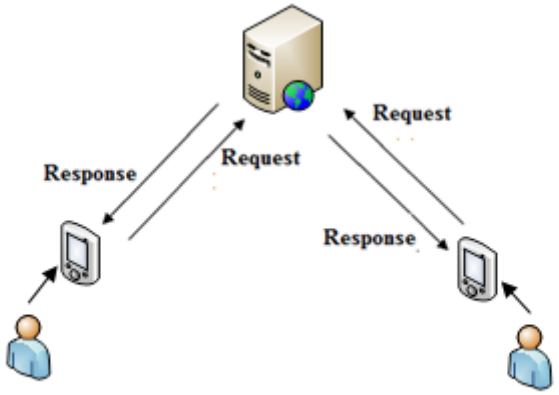
Contruction prototype yaitu tahapan penterjemahan pemodelan yang telah dibuat kedalam bahasa pemograman, tahapan ini meliputi pembuatan aplikasi dari sisi *client*, *server*, dan *basis data* berdasarkan model-model UML yang telah dibuat.

Deployment delivery and feedback adalah tahap melakukan pengujian sistem yang meliputi pengujian *black box* dan melakukan simulasi perbaikan dengan memperbaiki fasilitas obrolan *room* sehingga dapat mengatur siapa saja yang masuk dan melakukan setting *room* .

III. HASIL DAN PEMBAHASAN

Modeling quick design merupakan tahapan pembuatan arsitektur aplikasi, struktur menu aplikasi dan antarmuka aplikasi. Sehingga aplikasi dapat terlihat, tahapan ini berdasarkan *sequence diagram* dan *activity diagram* dari tahap *quick plan*.

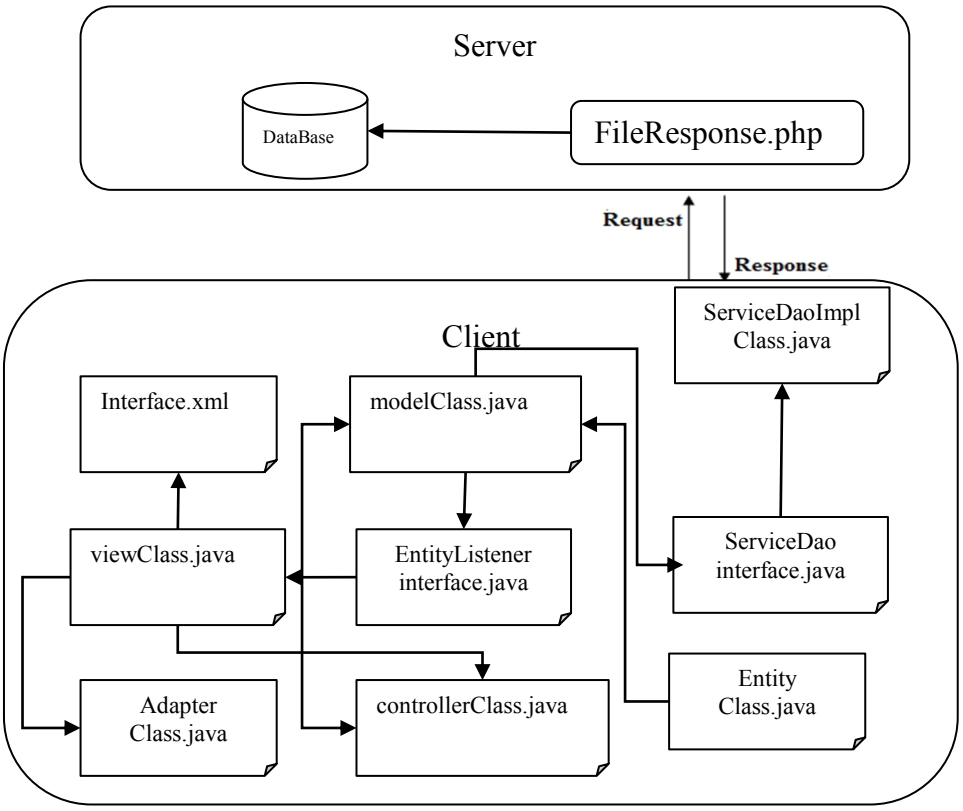
Arsitektur aplikasi yang akan dibuat yaitu *cliant server*, *client* yang dimaksudkan adalah *user* yang menggunakan *smartphone* untuk kemudian melakukan *request* ke *web server* sehingga *user* atau *cliant* mendapatkan layanan atau *response* dari *web server* yang kemudian *user* dapat menggunakan aplikasi secara penuh.



Gambar 4.19 Arsitektur Aplikasi Chatroom.

Tahapan *contruction prototype* yaitu tahapan penterjemahan pemodelan yang telah dibuat kedalam bahasa pemograman, tahapan ini meliputi pembuatan aplikasi dari sisi *client*, *server*, dan *basis data* berdasarkan model-model UML yang telah dibuat.

Tahapan selanjutnya merupakan *contruction prototype* yaitu tahapan penterjemahan pemodelan yang telah dibuat ke dalam bahasa pemograman, tahapan ini meliputi penerapan arsitektur MVC pada aplikasi yang akan dirancang, dan melakukan proses pengkodean berdasarkan model-model UML yang telah dibuat.



Pembuatan *Database*

a. Tabel *User*

Nama tabel : tbl_user

Tabel 4.11 Tabel *User*

No	Nama Field	Tipe Data	Status	Keterangan
1	Id	Int(10)auto_increment	Primary key	Id *user*
2	Nama	Varchar(25)	-	Nama *User*
3	Email	Varchar(25)	-	Email *User*
4	Password	Varchar(25)	-	Password *User*
5	Status	Varchar(7)	-	Status
6	Indata	Timestamp	-	Waktu Daftar

Pembuatan *Mapping Class*

No.	Kelas Implementasi *class entity*
1	com.sansan.mvcroom.entity.Messenger.java
2	com.sansan.mvcroom.entity.Room.java
3	com.sansan.mvcroom.entity.User.java

Pembuatan *Class Service*

No.	Kelas Implementasi *interface service*
1	com.sansan.mvcroom.service.MessengerDao.java
2	com.sansan.mvcroom.service.RoomDao.java
3	com.sansan.mvcroom.service.UserDao.java

Masing-masing interface ini kemudian diimplementasikan kedalam sebuah *class*:

Tabel 4.16 Implementasi DAO

No.	Kelas Implementasi *class service*
1	com.sansan.mvcroom.service.Imp.MessengerDaoImp.java
2	com.sansan.mvcroom.service.Imp.RoomDaoImp.java
3	com.sansan.mvcroom.service.Imp.UserDaoImp.java

Pembuatan *Class Adapter*

No.	Kelas Implementasi *class adapter*
1	com.sansan.mvcroom.adapter.ListAdapterMessenger.java
2	com.sansan.mvcroom.adapter.ListAdapterRoom.java

Pembuatan *Chatroom Connected file*

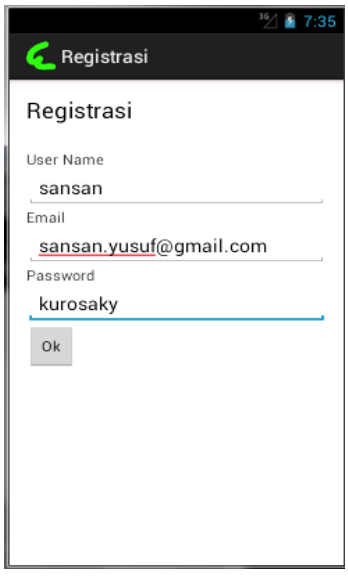
connection.php, *file* php ini berfungsi untuk mengkoneksikan antara *file* php dengan database sql yang ada di *server*.

Implementasi *Model-View-Controller* pada Aplikasi *Chatroom*

a. Implementasi View (Tampilan)

No.	Kelas Implementasi *view*
1	com.sansan.mvcroom.MenuAwal.java
2	com.sansan.mvcroom.LoginUser.java
3	com.sansan.mvcroom.RegistrasiUser.java
4	com.sansan.mvcroom.MenuUtama.java
5	com.sansan.mvcroom.BuatRoom.java
6	com.sansan.mvcroom.ListRoom.java
7	com.sansan.mvcroom.MessengerRoom.java

registrasi_user.xml merupakan *layout* untuk melakukan registrasi *user* yang belum meiliki akun, sehingga *user* bisa menggunakan aplikasi secara penuh. Pada *layout* ini *user* diminta mengisi form nama yang akan digunakan sebagai nick name, email, dan password sebagai pengunci akun. Berikut tampilan dari *layout* registrasi_view.xml.



Gambar 4.36 Registrasi_view.xml

Implementasi Model

No.	Kelas Implementasi *model*
1	com.sansan.mvcroom.model.MessengerModel.java
2	com.sansan.mvcroom.model.RoomModel.java
3	com.sansan.mvcroom.model.UserModel.java

Implementasi *Controller*

No.	Kelas Implementasi *view*
1	com.sansan.mvcroom.model.MessengerModel.java
2	com.sansan.mvcroom.model.RoomModel.java
3	com.sansan.mvcroom.model.UserModel.java

Pembuatan Class Interface Listener

No.	Kelas Implementasi *Class interface listener*
1	com.sansan.mvcroom.model.event.MessengerListener.java
2	com.sansan.mvcroom.model.event.RoomListener.java
3	com.sansan.mvcroom.model.event.UserListener.java

IV. KESIMPULAN

Dari penjelasan bab- bab sebelumnya maka dapat diambil kesimpulan sebagai berikut:

1. Dengan menggunakan metode MVC, kode pemograman aplikasi menjadi lebih rapih, karena ada pembagian yang jelas antara masing-masing kelas, mulai dari *model, view, controller.*
2. *Prototyping* merupakan metode pengembangan perangkat lunak yang cocok dalam pengembangan aplikasi yang tidak memiliki kepastian terhadap efisiensi algoritma, kemampuan penyesuaian harus dilakukan dengan cara berinteraksi dengan manusia dan mesin sehingga dapat dilakukan perbaika dengan cepat.

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada kedua orang tua yang telah membantu secara moril maupun materil. Penulis juga perkenankan untuk menyampaikan ucapan terima kasih kepada Bapak Dr. Dhami Johar Dhamiri M.Si selaku pembimbing I dan Bapak Cepy Slamet MT selaku pembimbing II yang telah memberikan arahan serta bimbingan selama penyelesaian laporan penelitian ini.

DAFTAR PUSTAKA

[1] Susanto, Andi (2010). Adobe Flash + XML = Rich Multimedia Application. Yogyakarta: Andi.

[2] Nugroho, Adi (2009). Rekayasa Perangkat Lunak Menggunakan UML dan Java. Yogyakarta: Andi.

[4] Phpwact (2014). Model-View-Controller (MVC). Posting date 2007/10/06, From online database Phpwact on the Word Wibe Web: http://www.phpwact.org/pattern/model_view_controller.

[5] Microsoft (2014). Model-View-Controller. Posting date 2014, From online database Microsoft on the Word Wibe Web: http://msdn.microsoft.com/en-us/library/ff649643.aspx.

[6] Presman, S. Roger (2010). Software Enggineering A Practitioner's Approach Sevent Edition. New York: McGraw-Hill.

