khazanah informatika

Jurnal Ilmu Komputer dan Informatika

Perancangan Aplikasi *Chatting* Berbasis *Web* di PT. Pura Barutama Kudus Menggunakan Socket.IO dan *Framework Foundation*

Ramos Somya*
Program Studi S1 Teknik Informatika, Fakultas Teknologi Informasi
Universitas Kristen Satya Wacana
Salatiga
*ramos.somya@uksw.edu

Abstrak-Aplikasi *chatting* merupakan aplikasi yang penting pada sebuah perusahaan besar karena dapat dipastikan antara divisi satu dengan divisi yang lain berada di lokasi yang berjauhan. EDP Keuangan PT. Pura Barutama Kudus bertugas dalam pembuatan program aplikasi yang terkait dengan pengolahan data keuangan dan faktur di semua unit di PT. Pura Barutama. Aplikasi keuangan yang telah ada akan selalu berkembang sesuai dengan kebutuhan *user*. Kendala yang dihadapi adalah aktivitas komunikasi dan penyebaran informasi pada PT. Pura Barutama dilakukan dengan menggunakan layanan *email* maupun aplikasi *messenger*. Untuk menggunakan layanan tersebut, perangkat yang digunakan harus selalu terkoneksi dengan internet. Apabila internet mati maka komunikasi tidak dapat dilakukan. Jalan satu-satunya untuk menyebarkan informasi adalah dengan menelepon satu per satu ke tiap-tiap unit. Hal ini menimbulkan ketidakefisienan pekerjaan dan produktivitas perusahaan menjadi terganggu. Pada penelitian ini dilakukan perancangan aplikasi *chatting* dengan menggunakan *Framework* CodeIgniter, Socket.IO dan *Framework* Foundation untuk mengatasi masalah yang ada. Hasil dari penelitian ini adalah aplikasi *chatting* berbasis *web* yang dibangun menggunakan *Framework* CodeIgniter, Socket.IO dan *Framework* Foundation terbukti dapat membantu *user* untuk saling berkomunikasi atau melakukan *attach file*, serta mempermudah melakukan penyebaran informasi atau pengumuman baik dalam kondisi terhubung internet maupun tidak.

Kata Kunci: Aplikasi Chatting, Web, Socket.IO, Framework Foundation, Framework CodeIgniter

1. Pendahuluan

Aplikasi *chatting* merupakan aplikasi yang penting pada sebuah perusahaan besar karena dapat dipastikan antara divisi satu dan divisi yang lainnya berada di tempat yang berjauhan. Divisi keuangan PT. Pura Barutama selalu mengontrol dan berkomunikasi dengan seluruh unit yang ada di PT. Pura Barutama. Penggunaan aplikasi *chatting* akan meningkatkan produktivitas dan efisiensi kerja. Misalnya sebagai sarana berkomunikasi, *attach file* dan sebagai media pengumuman.

EDP Keuangan adalah salah satu sub divisi dari divisi keuangan di PT. Pura Barutama selalu beinteraksi dengan berbagai macam divisi yang ada. Selain itu EDP Keuangan bertugas dalam pembuatan program yang terkait dengan pengolahan data keuangan, faktur di semua unit PT. Pura Barutama. Aplikasi keuangan yang telah ada akan selalu berkembang sesuai kebutuhan *user*.

Aktivitas komunikasi dan penyebaran informasi pada PT. Pura Barutama dilakukan dengan menggunakan layanan *email* maupun aplikasi *messenger*, untuk menggunakan layanan tersebut, perangkat yang digunakan harus selalu terkoneksi dengan internet. Apabila internet mati maka komunikasi tidak dapat dilakukan. Jalan satu-satunya untuk menyebarkan informasi adalah dengan menelepon satu per satu ke tiap-tiap unit. Hal ini menimbulkan ketidakefisienan pekerjaan dan produktivitas menjadi terganggu.

Berdasarkan permasalahan tersebut, maka dapat diketahui rumusan masalah dalam penelitian ini adalah bagaimana membuat aplikasi *chatting* berbasis *web* menggunakan *Framework* CodeIgniter, Socket.IO dan *Framework* Foundation dengan memanfaatkan jaringan komputer lokal yang ada di PT. Pura Barutama. Aplikasi *chatting* tersebut hanya membutuhkan sebuah *browser* yang digunakan untuk mengakses aplikasi *chatting* selama perangkat terkoneksi dengan jaringan lokal baik menggunakan kabel maupun *wireless* tanpa terkoneksi dengan internet. Selain itu, aplikasi berbasis *web* tergolong ringan, sehingga spesifikasi komputer yang digunakan oleh *user* tidak berat.

Aplikasi *chatting* dibangun menggunakan *Framework* CodeIgniter (CI) dan Socket.IO sebagai *back-end*. *Framework* CodeIgniter mendukung konsep *Model View Controller* (MVC) sehingga dalam pengembangan aplikasi *chatting* akan menjadi lebih terstruktur dan terorganisasi. Sedangkan untuk *interface* (*front-end*) aplikasi *chatting* menggunakan *Framework* Foundation untuk membuat aplikasi *chatting* ini menjadi *responsive* dan untuk penyimpanan data menggunakan *database* Oracle karena menyesuaikan dengan *database* yang digunakan PT. Pura Barutama.

Mengingat begitu luasnya ruang lingkup pembuatan aplikasi *chatting* dan terbatasnya kemampuan dan identifikasi maka diperlukan batasan agar sistem yang dibangun tidak menyimpang dari permasalahan dalam sistem aplikasi *chatting*. Adapun batasan masalah yang dilakukan adalah aplikasi hanya dapat dipergunakan untuk perorangan (*Private Chat*), dikarenakan aplikasi *chatting* ini bersifat rahasia. Selain digunakan untuk melakukan *private chat*, aplikasi ini juga dapat memberikan informasi berupa pengumuman kepada *user* maupun unit yang lain apabila ada pembaruan aplikasi yang dibuat oleh EDP Keuangan PT. Pura Barutama.

Diharapkan dengan dibangunnya aplikasi ini, dapat mempermudah para penggunanya untuk berkomunikasi dan bertukar informasi. Sehingga dapat meningkatkan produktivitas kerja dan mengefisienkan waktu

2. Kajian Pustaka

a. Penelitian Terdahulu

Terdapat beberapa penelitian terdahulu yang digunakan sebagai acuan dalam penelitian ini. Penelitian pertama menghasilkan aplikasi yang dapat melakukan pembuatan suatu *room* yang dapat dilakukan oleh semua pengguna baik itu pengguna telepon seluler maupun pengguna komputer. Apabila melakukan mengirim dengan ukuran *file* yang besar maka proses pengiriman akan menjadi lama [1].

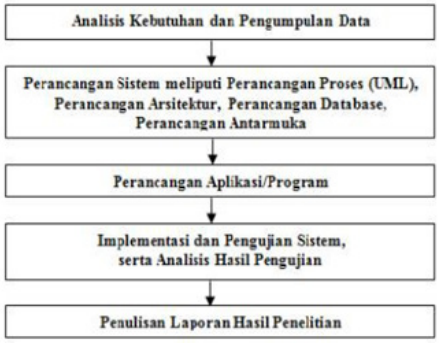
Penelitian kedua menghasilkan aplikasi yang dibangun dengan menggunakan Visual Basic 6.0. Aplikasi yang dilengkapi dengan menu *send file* untuk mengirimkan *text* yang mempermudah komunikasi antar pengguna yang berada dalam sebuah jaringan. Aplikasi *chatting* yang dikembangkan tidak adanya pemisahan form antara *private* dengan *group*. Model pengiriman *text* melalui menu *send file* harus memasukkan alamat IP tujuan pengguna sehingga dinilai tidak efektif dan efisien. Program aplikasi hanya dapat berjalan pada sistem operasi Windows [2].

Penelitian ketiga mengimplementasi Algoritma Rijndeal 128 pada aplikasi *chatting* berbasis HTML5 webSocket. Pada penelitian tersebut menghasilkan aplikasi *group chatting*. Pesan yang dikirim akan dienkripsi dengan menggunakan algoritma Rijndeal 128, sehingga pesan menjadi aman [3].

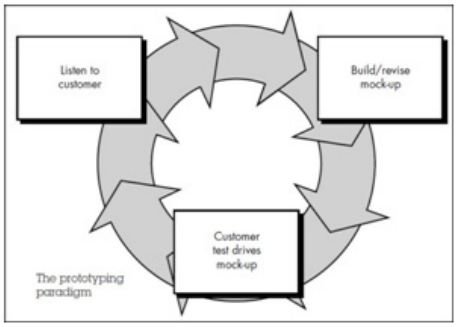
Berdasarkan penelitian yang telah dilakukan sebelumnya yang menjadi pembeda antara penelitian ini dan penelitian terdahulu adalah aplikasi *chatting* dibuat dengan menggunakan *Framework* CodeIgniter, Socket.IO dan *Framework* Foundation. Penggunaan *Framework* CodeIgniter penulisan kode akan menjadi terstruktur dan terorganisir karena *Framework* CodeIgniter menerapkan konsep MVC (*Model View Controller*). Socket.IO memungkinkan pembuatan aplikasi komunikasi *realtime* antara *client* dan *server*. Penggunaan *Framework* Foundation membuat halaman *web* menjadi *responsive* sehingga dapat menyesuaikan di perangkat manapun saat aplikasi diakses.

b. Chatting

Chatting adalah pecakapan dua orang atau lebih secara *real time* melalui komputer yang terhubung dengan jaringan. Layanan untuk *chatting* di internet antara lain Yahoo, Skype, mIRC, Windows Live Messenger. Adanya layanan *chat* memungkinkan untuk dapat berkomunikasi melalui internet dengan orang-orang yang berada di seluruh dunia [4].



Gambar 1. Tahapan Penelitian



Gambar 2. *Model prototyping* [10]

c. Framework

Framework adalah kumpulan perintah atau fungsi dasar yang membentuk aturan-aturan tertentu dan saling berinteraksi satu sama lain sehingga dalam pembuatan aplikasi *website*, *programmer* harus mengikuti aturan dari *framework* tersebut. CodeIgniter dilengkapi dengan berbagai pustaka siap pakai untuk berbagai kebutuhan. Misalnya saja koneksi *database*, email, *session* dan *cookies*, keamanan, manipulasi gambar dan banyak lagi sehingga mempermudah pekerjaan [5].

d. Socket.IO

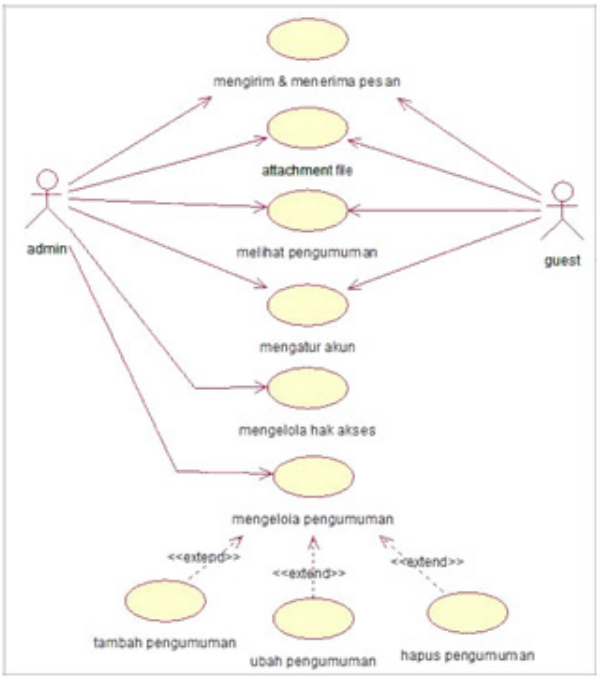
Socket.IO adalah lapisan komunikasi berbasis *event* untuk aplikasi *web realtime*, yang dibangun di atas Engine. IO. Hal ini memungkinkan pengembang untuk mengirim dan menerima data tanpa khawatir tentang kompatibilitas *browser* yang berbeda [6].

e. Framework Foundation

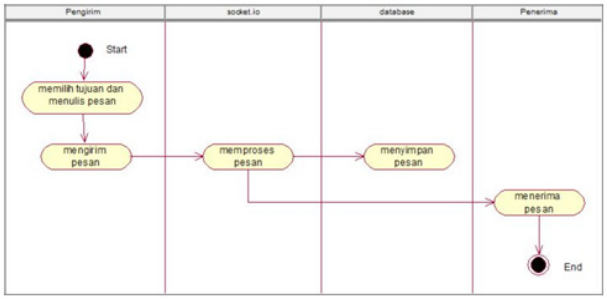
Framework Foundation adalah *framework* yang dibangun dengan HTML, CSS, dan Javascript, sebagai komponen utama dari *web*. *Framework* Foundation mengunakan teknologi JQuery, HTML 5 dan Normalizer [7]. *Framework* Foundation digunakan untuk membuat *web* dapat menyesuaikan resolusi layar yang digunakan.

f. Basis Data Oracle

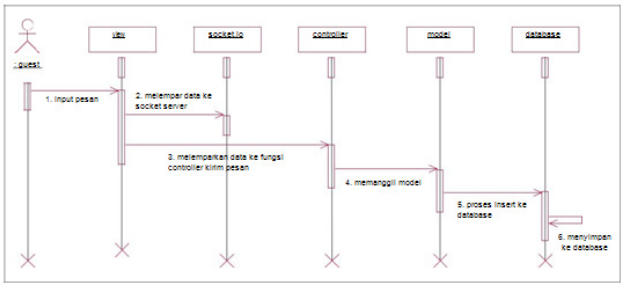
Oracle adalah sistem basis data yang memiliki banyak fitur yang memungkinkan administrator basis data dapat mengelola data secara lebih akurat sehingga Oracle lebih sesuai digunakan sebagai sistem basis data untuk aplikasi yang berkuran besar dan kompleks [8].



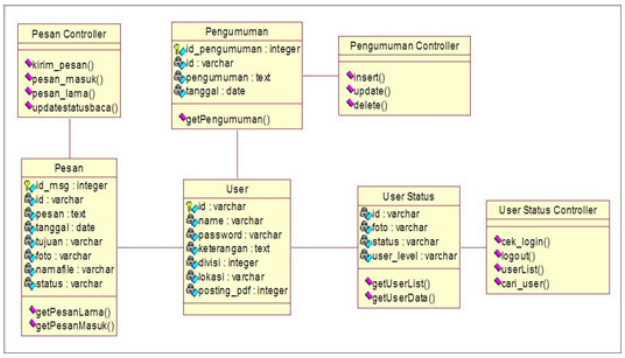
Gambar 3. Diagram *Use Case*



Gambar 4. Diagram *Activity* Mengirim Pesan



Gambar 5. Diagram *Sequence* Proses Kirim Pesan



Gambar 6. Diagram *Class*

3. Perancangan Sistem

Pada penelitian ini dilakukan beberapa tahapan yang saling berkaitan dengan tahapan selanjutnya, yaitu: 1) Analisis kebutuhan dan pengumpulan data yang diperlukan. 2) Perancangan sistem. 3) Perancangan aplikasi/program. 4) Implementasi dan pengujian sistem, serta analisis hasil pengujian. 5) Penulisan laporan hasil penelitian [9]. Tahapan-tahapan yang dilakukan dalam penelitian ini dapat dilihat pada Gambar 1.

Berdasarkan Gambar 1 dapat dijelaskan tahapan penelitian yang dilakukan adalah sebagai berikut: 1) Tahapan pertama: analisis dan pengumpulan data, di mana peneliti melakukan wawancara dengan karyawan EDP Keuangan tentang aplikasi yang akan dibuat. Berdasarkan hasil wawancara dengan karyawan EDP Keuangan bahwa selama ini belum ada media yang menghubungkan unit satu ke unit yang lainnya dalam menyebarkan informasi atau *attach file* tanpa menggunakan internet. Selama ini hanya menggunakan telepon dan *email* untuk berkomunikasi. *Email* sendiri memiliki kekurangan yaitu membutuhkan koneksi internet. Jika koneksi internet mati, maka tidak bisa terjadi komunikasi. Jika melalui telepon akan terasa tidak efisien karena karyawan EDP Keuangan harus menghubungi satu per satu unit yang banyak di PT. Pura Barutama jika terjadi perubahan program. Sehingga dibutuhkan aplikasi yang dapat menyediakan fasilitas *chatting*, *attach file,* dan penyebaran pengumuman dengan memanfaatkan jaringan lokal. Tahap kedua, ketiga, dan keempat dilakukan perancangan aplikasi *chatting* menggunakan metode Prototype. Sedangkan tahap kelima dilakukan penulisan artikel ilmiah atau laporan penelitian.

Metode perancangan yang dipakai dalam pembuatan aplikasi *chatting* adalah metode Prototyping. Metode Prototyping adalah metode dalam pengembangan rekayasa *software* yang bertahap dan berulang serta mementingkan sisi *user* sistem. Penggunaan metode Prototyping, pengembang dan karyawan EDP Keuangan dapat saling berinteraksi selama proses pembuatan sistem sampai aplikasi sesuai dengan kebutuhan pengguna.

Tahap pengumpulan kebutuhan dilakukan untuk mengetahui permasalahan dan kebutuhan sistem. Pada tahap ini juga dilakukan pencarian data yang dibutuhkan oleh sistem. Agar aplikasi *chatting* yang dibagun dapat memenuhi kebutuhan pengguna. Analisis kebutuhan sistem dilakukan bersama dengan karyawan EDP Keuangan. Berdasarkan analisis kebutuhan sistem yang dilakukan bahwa selain untuk melakukan *chatting* dengan user lain, dibutuhkan juga fasilitas untuk *attach file*, dan melakukan pengumuman. Pada aplikasi *chatting* yang dibangun dibagi menjadi dua hak akses yaitu administrator dan *user* biasa. Kebutuhan adminstrator mencakup: *chatting* dengan pengguna lain dengan fasilitas *attach file*, *posting* pengumuman, hapus pengumuman, mengubah pengumuman jika terjadi kesalahan, mengelola *user*, mengelola *user* di sini maksudnya adalah memberikan hak akses administrator kepada *user* biasa jika *user* tersebut ini mem*posting* pengumuman. Sedangkan kebutuhan *user* biasa hanya melakukan *chatting* dengan pengguna lain dengan fasilitas *attach file*. Data *user* sendiri menggunakan data yang sudah dimiliki oleh PT Pura Barutama, sehingga pada aplikasi *chatting* ini tidak perlu menambahkan data *user* baru, karena diambil dari *database* yang sudah ada.

Analisis kebutuhan *hardware* dan spesifikasi *software* yang digunakan dalam membangun aplikasi *chatting* ini yaitu: analisis perangkat keras yang akan digunakan adalah Prosesor Intel Core i3, 2.3 GHz, RAM 6 GB dan Harddisk 500 GB. Sedangkan perangkat lunak yang digunakan adalah sistem Operasi Windows 7 Ultimate, Sublime sebagai *text editor*, *web server* (Apache), *web browser* (dalam penelitian ini menggunakan Google Chrome), Oracle sebagai *database*, dan Rational Rose untuk membuat UML diagram sistem.

Sebelum dilakukan pengkodean, dilakukan perancangan UML diagram untuk memvisualisasikan alur proses dan kebutuhan data. UML dibuat dalam diagram Use Case, diagram Activity, diagram Sequence dan diagram Class yang akan dijelaskan satu per satu.

Gambar 3 menunjukkan diagram Use Case dari aplikasi *chatting*. Aplikasi yang dibuat dibedakan menjadi dua jenis *user* yaitu administrator dan *guest* (*user* biasa). Aktor dengan hak akses admin dapat mengirimkan dan menerima pesan, *attach file*, melihat pengumuman, mengelola data *user* dengan memberikan hak akses administrator kepada *user* biasa, mem*posting* pengumuman, menghapus pengumuman, mengubah pengumuman jika terjadi kesalahan. Sedangkan aktor dengan hak akses *guest* (*user* biasa) hanya dapat mengirimkan dan menerima pesan, *attach file*, melihat pengumuman, dan mengatur akunnya sendiri.

Gambar 4 menunjukkan diagram Activity ketika pengirim mengirimkan pesan. Hal pertama yang dilakukan adalah pengirim menentukan tujuan dan pesan yang akan dikirim. Kemudian data yang sudah diinputkan akan diproses melalui Socket kemudian ditampilkan pada panerima. Data pesan yang dikirim akan disimpan ke dalam *database*, sehingga histori percakapan masih bisa dilihat.

Gambar 5 merupakan diagram Sequence dari proses pengiriman pesan. Pertama *user* menginputkan pesan pada halaman *view*. Kemudian data diproses oleh *server* socket dan data menjadi parameter di *controller* kirim pesan. Setelah itu *controller* akan memanggil fungsi pada model dengan parameter yang dikirimkan *controller* kemudian model akan berhubungan dengan *database* untuk menyimpan data.

Gambar 6 merupakan diagram *Class* dari aplikasi *chatting* yang menunjukkan model dan *controller*. Pada aplikasi *chatting* dibuat dalam 4 model *class* yaitu *user class*, *user* status *class*, pesan *class* dan pengumuman *class*. Model *class* ini yang nantinya akan berhubungan dengan *database*. Sedangkan *controller class* akan menghubungkan antara *view* dan model *class*. Pada aplikasi *chatting* dibuat dalam 3 *controller class* yaitu *user* status *controller*, pesan *controller*, dan pengumuman *controller*.

Perulangan dari proses pada metode Prototyping terus berlangsung hingga semua kebutuhan terpenuhi. Proses evaluasi dilakukan sebanyak 3 kali, evaluasi pertama adalah menunjukkan kepada karyawan EDP Keuangan bagaimana aplikasi bekerja. Evaluasi kedua dilakukan penambahan fitur untuk *user* yang memiliki hak akses sebagai administrator agar dapat melakukan *posting* pengumuman, dan setiap pesan yang masuk mendapatkan notifikasi. Evaluasi ketiga dilakukan perbaikan beberapa fungsi yang masih memiliki *bug* di dalamnya.

Kode Program 1. Konfigurasi NodeJS

```
{
"name": "puramessenger",
"version": "1.0.0",
"description": "puramessenger",
"main": "server.js",
"scripts": {
"test": "echo \"Error: no test specified\" && exit 1"
},
"author": "ridvandani672012078",
"license": "ISC",
"dependencies":{
"Socket.IO":"*",
"express":"*"
}
}
```

Kode Program 2. Konfigurasi *Server*

```
var express   = require('express');
var app       = express();
var server    = require('http').createServer(app);
var io        = require('Socket.IO').listen(server);
var port      = process.env.PORT || 3001;
var users   = {};
server.listen(port, function(){
console.log("Server bejalan pada port %d", port);
});
```

Kode Program 3. Konfigurasi Koneksi *Database Framework* CodeIgniter

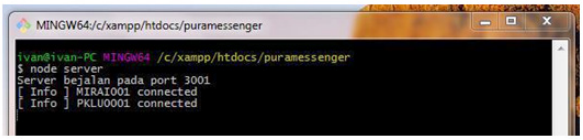
```
$db['default']['hostname'] = 'localhost/XE';
$db['default']['username'] = 'pura_m';
$db['default']['password'] = 'pura_m';
$db['default']['database'] = '';
$db['default']['dbdriver'] = 'oci8';
$db['default']['dbprefix'] = '';
$db['default']['pconnect'] = TRUE;
$db['default']['db_debug'] = TRUE;
$db['default']['cache_on'] = FALSE;
$db['default']['cachedir'] = '';
$db['default']['char_set'] = 'utf8';
$db['default']['dbcollat'] = 'utf8_general_ci';
$db['default']['swap_pre'] = '';
$db['default']['autoinit'] = TRUE;
$db['default']['stricton'] = FALSE;
```

Kode Program 4. Konfigurasi *Route Framework* CodeIgniter

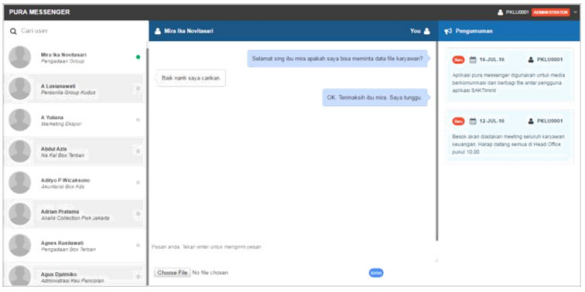
```
$route['default_controller'] = "user_con";
$route['404_override'] = '';
$route['translate_uri_dashes'] = FALSE;
```

Kode Program 5. Menampilkan *User* yang *Online* atau *Offline*

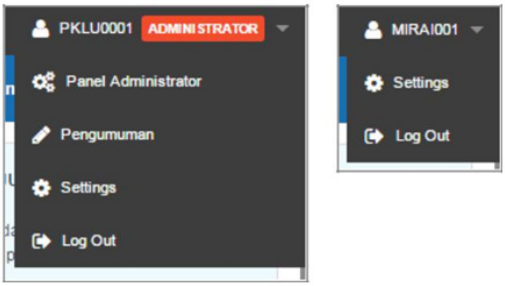
```
function cek_userlist() {
$.ajax({
url: "<?php echo base_url('index.php/user_con/userList');?>",
cache: false,
success: function(msg) {
$('#userList').load("<?php echo base_url('index.php/user_con/userList');?>");
}});
var waktu = setTimeout("cek_userlist()", 10000);
}
```



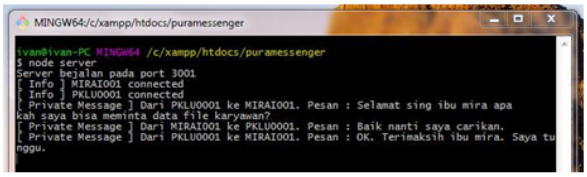
Gambar 7. Respon *Server* Ketika *User Log in*



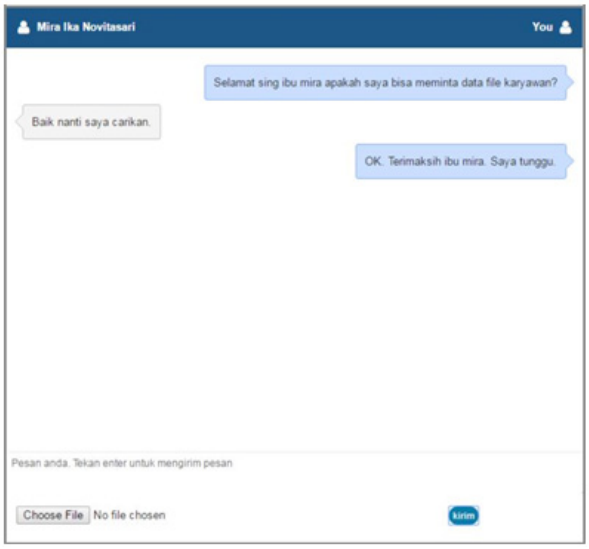
Gambar 8. Halaman Utama



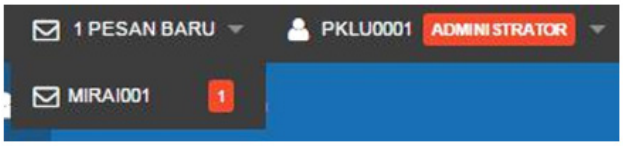
Gambar 9. Perbedaan Hak Akses Adminstrator dan *User* Biasa



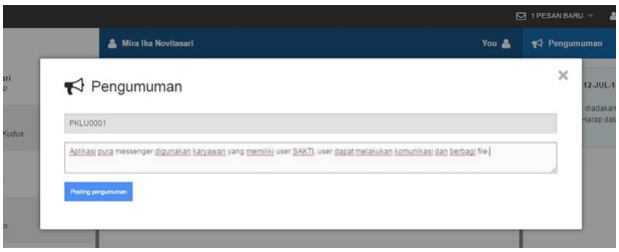
Gambar 10. Respon *Server* Ketika *User* Mengirim Pesan



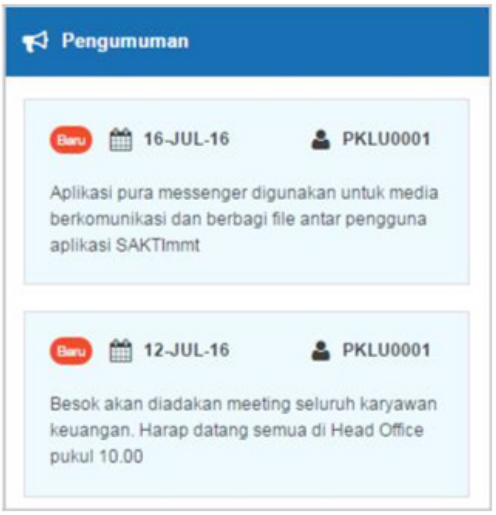
Gambar 11. Proses *Chatting*



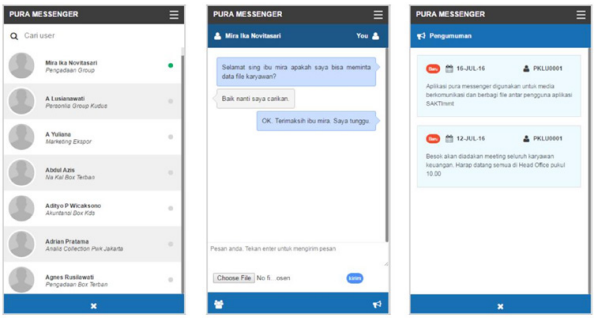
Gambar 12. Notifikasi Pesan yang Belum Dibaca



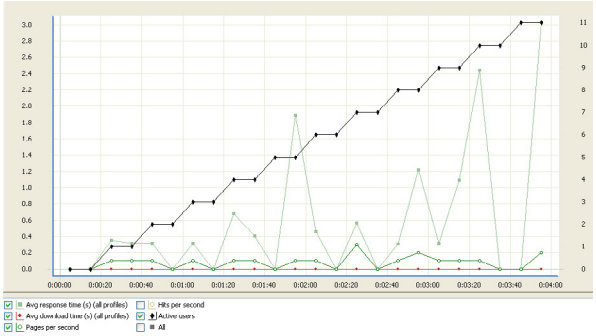
Gambar 13. Form *Posting* Pengumuman



Gambar 14. Daftar Pengumuman



Gambar 15. Tampilan *Responsive*



Gambar 16. Grafik *Performance* Aplikasi

Kode Program 6. Mengirim Pesan Melalui Socket

```
1.   $('#form-kirim-pesan').submit(function(e) {
2.   e.preventDefault();
3.   var url = $(this).attr('action');
4.   var data = new FormData(this);
5.   var pesan = '/w ' + $inp_tujuan.val() + " " +
     $inp_pesan.val();
6.   var filename =
     $('input[type=file]').val().replace(/C:\\fakep
     ath\\/i, '');
7.   var pData = {pesan: pesan, file: filename,
     gambar: $inp_gambar.val()}
8.
9.   Socket.emit('send message', pData);
10.    ...
11.  });
```

Kode Program 7. Media Query

```
1. @media (max-width: 700px){

2. .userListContainer {display:none;}

3. .announcementContainer {display: none;}

4. .chatbox{height: 67% !important;}

5.       }
```

4. Hasil dan Pembahasan

Sebelum melakukan pengkodean. Hal pertama yang harus dilakukan adalah meng-*install* NodeJS. Tujuan dari intalasi NodeJS karena Socket.IO berjalan di atas NodeJS sebagai modul. Kemudian dilakukan inisialisasi *project* NodeJS dengan perintah npm init. Perintah ini akan menghasilkan *file package*.json yang terlihat pada Kode Program 1.

Kode Program 1 menujukkan deskripsi mengenai aplikasi yang dibuat. Kemudian melakukan penambahan *dependencies* untuk menambahkan Socket.IO dan Express. Setelah konfigurasi *package*.json, selanjutnya penginstallan Socket.IO dan Express dengan menjalankan perintah npm *install* pada *command line*. Setelah proses instalasi, selanjutnya membuatan *server* untuk aplikasi *chatting* yang ditunjukkan pada Kode Program 2.

Kode Program 2 menunjukkan inisialisasi Socket dan *port*. Pada baris 5 diinisialisasikan bahwa aplikasi *chatting* akan berjalan pada *port* 3001. Selanjutnya dilakukan konfigurasi pada *Framework* CodeIgniter agar aplikasi dapat terhubung dengan *database* yang ditunjukkan pada Kode Program 3.

Kode Program 3 menunjukan konfigurasi *database* yang berisi *hostname*, *username*, *password*, dan *driver*. Pada penelitian ini *database* yang digunakan adalah Oracle, maka *driver* yang digunakan adalah OCI8. Sedangkan untuk pengaturan *route* terlihat pada Kode Program 4.

Kode Program 4 menunjukkan *controller* mana yang akan dipanggil pertama kali saat aplikasi dijalankan. Pengubahan dilakukan pada baris 1 yaitu saat aplikasi dijalankan akan memanggil kelas *controller* user_con yang mengarah pada halaman *log in*. Pada halaman *log in* terdapat dua *text field* dan satu tombol. *Text field* pertama adalah tempat *user* untuk memasukkan *user* id *user*. *Text field* kedua adalah tempat *user* memasukkan *password*, dan tombol *log in* yang digunakan untuk memproses inputan dari *user*. *Username* dan *password* kemudian diproses untuk dilakukan pengecekan ke *database*. Apabila *username* dan *password* tidak terdapat dalam *database*, akan kembali ke halaman *log in*. Jika cocok akan maka *user* berhasil *log in*. Gambar 7 adalah respons *server* ketika ada *user* yang *log in*. Gambar 8 merupakan halaman yang akan muncul ketika *user* berhasil *log in*.

Gambar 8 menunjukkan halaman yang ditampilkan ketika *user* berhasil *log in*. Pada bagian *header* sebelah kanan terdapat menu untuk *user*. *User* dibagi menjadi 2 hak akses dengan menu yang berbeda. Hak akses yang pertama adalah adminstrator dan hak akses yang kedua adalah *user* biasa. Perbedaan menu dapat dilihat pada Gambar 9.

Gambar 9 menunjukkan perbedaan hak akses pada tiap-tiap *user*. *User* yang mendapat hak akses sebagai adminstrator terdapat label "administrator" sedangkan untuk *user* yang tidak mendapat hak akses administrator tidak terdapat label "administrator". *User* yang mendapat hak administrator terdapat menu panel administrator, pengumuman, *settings,* dan *log out*. Sedangkan *user* biasa hanya terdapat menu *settings* dan *log out*. Kedua hak akses tersebut bisa melakukan *chatting* dengan sesama *user*. Pada bagian di bawah *header* dibagi lagi menjadi 3 bagian yaitu bagian sebelah kiri untuk menampilkan *user* yang *online* maupun yang sedang *offline*. Bagian tengah untuk melakukan *chatting*, dan bagian kanan untuk menampilkan pengumuman. Untuk menampilkan *user* yang sedang *online* maupun *offline* aplikasi akan me*reload* secara berkala halaman *userlist*.

Kode Program 5 menunjukkan fungsi untuk menampilkan *user* yang *online* maupun *offline* dengan memanfaatkan Jquery Ajax, setiap 10000 *millisecond class controller* userList akan di*reload* secara terus menerus tanpa me*refresh* keseluruhan halaman. Proses *chatting* pada penelitian ini memanfaatkan *library* Javascript Socket. IO. Penggunaan Socket.IO memungkinkan komunikasi secara *realtime* antara *client* dan *server*. Socket.IO berjalan pada *web browser* dan berjalan pada *server*. Kode Program 6 merupakan fungsi pada sisi *client* untuk mengirimkan data dari *client* ke *server*.

Kode Program 6 menunjukkan fungsi untuk mengirimkan data dari *client* ke *server*. Data yang diinputkan akan menjadi parameter pada fungsi '*send message*' pada komputer *client* dan diterima oleh *server*.

Gambar 10 menunjukkan aktivitas *server*. Pesan yang masuk akan ditampilkan ke dalam *console*. *Server* mengirim nilai pengembalian data pada *client* dalam fungsi '*whisper*' dan "kirim". Gambar 11 merupakan hasil data yang ditampilkan di *web browser*. Gambar 12 merupakan notifikasi ketika ada pesan yang belum dibaca.

Gambar 12 menunjukkan notifikasi pesan yang belum dibaca. Ketika ada pesan yang baru saja masuk atau pesan yang belum dibaca, pada *header* akan muncul notifikasi yang berisi jumlah pesan dan siapa pengirim pesan tersebut. *User* dengan hak akses administrator juga bisa melakukan *posting* pengumuman. *Posting* pengumuman juga dilakukan melalui Socket.

Gambar 13 menunjukkan *form* untuk menulis pengumuman. Data inputan akan dikirimkan ke *server*. Kemudian *server* mengolah data dan mengirimkan kembali ke *client* untuk ditampilkan yang terlihat pada Gambar 14.

Aplikasi *chatting* yang dibangun dapat menyesuaikan dengan resolusi layar ketika aplikasi diakses dengan memanfaatkan *Framework* Foundation. Pemanfaatan *Framework* Foundation aplikasi *web* akan menjadi *responsive*.

Grid dalam *framework* Foundation digunakan untuk mengaktifkan fitur *responsive*. Maksimum penggunaan grid pada *Framework* Foundation adalah 12 kolom. Grid dikelompokkan menjadi 3 properti yaitu *small*, *medium*, *large*. Properti *small* digunakan untuk menyesuaikan resolusi layar yang kecil, properti *medium* digunakan untuk menyesuaikan resolusi layar yang lebih besar dari properti *small* dan properti *large* untuk resolusi layar yang lebih besar lagi. Kode Program 7 menunjukkan pengaturan media *query* yang digunakan untuk menampilkan *layout* yang berbeda berdasarkan media yang digunakan.

Gambar 15 menunjukkan tampilan aplikasi ketika diakses pada resolusi layar yang lebih kecil. Dilakukan penambahan menu pada sisi bawah layar untuk mengakses menu melihat daftar *user* dan melihat pengumuman.

Pengujian sistem dilakukan untuk mencari kesalahan pada aplikasi *chatting* yang dibuat. Pengujian dilakukan untuk mengetahui apakah aplikasi yang sudah dibuat berjalan dengan baik dan sesuai dengan kebutuhan *user*. Pengujian alpha menggunakan metode *blackbox* yaitu pengujian fungsi-fungsi aplikasi secara langsung tanpa memperhatikan alur eksekusi program. Pengujian ini dilakukan dengan memeriksa setiap fitur aplikasi. Berdasarkan pengujian yang dilakukan pada aplikasi *chatting*, setiap fungsi dapat berjalan dengan baik, termasuk dapat mengirimkan dan menerima *file attacthment* dengan ukuran maksimum 25 MB. Berdasarkan hal ini, maka disimpulkan bahwa aplikasi ini berfungsi dengan baik dan sesuai yang diharapkan.

Pengujian lainnya juga dilakukan untuk menguji performa aplikasi *chatting* ini ketika digunakan oleh banyak *user*. Pengujian dilakukan dengan bantuan aplikasi Web Application Testing (WAPT), yaitu melibatkan 110 *user*. Hasil pengujian ditunjukkan oleh Gambar 16.

Berdasarkan grafik pada Gambar 16, dapat dijelaskan tentang *performance* aplikasi yang meliputi waktu *response* yang dibutuhkan oleh seorang *user* untuk menunggu *response* dari *server*, waktu *download* yang dibutuhkan oleh *user* dan banyaknya halaman *web* yang dapat dieksekusi tiap detik. Hasil pada grafik menunjukkan bahwa waktu *response* aplikasi cenderung tidak stabil. Karena pada interval waktu tertentu terjadi waktu *response* yang singkat yaitu 0 detik dan kadang terjadi waktu response yang cenderung lama yaitu 3.03 detik. Dalam analisis waktu *response* terdapat 3 (tiga) batasan, yaitu waktu *response* 0.1 detik merupakan batas ideal bagi *user* dalam menunggu *response* dari *server*, 1.0 detik merupakan batas yang masih bisa ditoleransi dan untuk batas 10 detik merupakan batas yang tidak dapat ditoleransi lagi, karena menurut *survey* waktu *response* yang memasuki detik ke 8.0 akan membuat *user* meninggalkan sistem. Dari grafik dapat disimpulkan bahwa waktu *response* masih dapat ditoleransi, karena berada dalam batas 0.0 - 3.0 detik.

Download time yang dimaksud di sini adalah waktu yang dibutuhkan pada saat *user* melihat *web title* (pada *browser title bar*) hingga *user* dapat melihat halaman *web* yang meliputi *frame*, *tabel* dan HTML *text*. Dari grafik pada Gambar 16 terlihat bahwa *download time* dapat dikatakan stabil (waktu *download* tidak berubah untuk tiap periode waktu) dan singkat (waktu *download* hanya 0 detik).

Pages per second yang dimaksud adalah banyaknya halaman *web* yang dapat dieksekusi tiap detik. Hal ini terlihat pada grafik pada Gambar 16 di mana untuk tiap interval waktu tertentu tidak terjadi perubahan *pages per second* yang besar. Didapatkan 0.1 halaman tiap detik (paling sedikit) dan 0.3 halaman tiap detik (paling banyak).

Pengujian beta dilakukan dengan melakukan presentasi dengan EDP Keuangan. Presentasi dilakukan dengan melakukan demo program yang telah di-*upload* ke *server* untuk mengetahui apakah aplikasi *chatting* berfungsi dengan baik dan sesuai kebutuhan pengguna. Berdasarkan hasil presentasi dengan karyawan EDP Keuangan, aplikasi berfungi dengan baik dan sesuai dengan kebutuhan pengguna.

5. Kesimpulan

Berdasarkan penelitian yang telah dilakukan maka dapat diambil kesimpulan bahwa pembuatan aplikasi *chatting* dapat dibuat dengan menggunakan *Framework* CodeIgniter, Socket.IO dan *Framework* Foundation. Pemanfaatan *Framework* CodeIgniter dapat diterapkan konsep MVC (*Model View Controller*) sehingga penulisan kode menjadi lebih terstruktur dan terorganisir. Pemanfaatan Socket.IO cocok untuk pembuatan *aplikasi realtime* seperti aplikasi *chatting* dan *Framework* Foundation membuat tampilan aplikasi menjadi *responsive* yang dapat menyesuaikan diberbagai resolusi layar.

Berdasarkan hasil pengujian, aplikasi *chatting* dapat membantu karyawan EDP Keuangan atau *user* lainnya untuk berkomunikasi, *attach file* dengan ukuran maksimum 25 MB, dan menyebarkan informasi jika sewaktu-waktu terjadi penambahan atau perubahan program yang dikembangkan oleh karyawan EDP Keuangan. Spesifikasi *hardware* maupun *software* aplikasi *chatting* ini tergolong aplikasi yang ringan karena merupakan aplikasi berbasis *web*, *user* hanya membutuhkan *web browser* kemudian mengakses aplikasi *chatting* yang sudah di*upload* ke *server* dan dengan memanfaatkan jaringan lokal maka untuk mengakses aplikasi *chatting* tidak membutuhkan koneksi internet.

Saran pengembangan yang dapat dilakukan adalah dengan menambahkan fitur keamanan pada aplikasi *chatting*. Fitur keamanan ini digunakan untuk mengenkripsi pesan *chat* yang dikirimkan. Aplikasi *chatting* juga bisa dikembangkan untuk versi aplikasi *mobile* yang tersinkronisasi ke aplikasi *chatting web* ini, sehingga *user experience* menjadi lebih baik.

6. Daftar Pustaka

[1] M. Z. Teddy, dan D. W. Surya, "Aplikasi Chat pada Handphone dan Komputer dengan Media Bluetooth (Bluetooth Chat)," Jurnal Teknologi Informasi - Aiti, vol. 4, no. 1, pp. 60-74, 2009.

[2] S. Roni, dan S. Edhy, "Membangun Aplikasi Chatting Berbasis Multiuser," Jurnal Dasi, vol. 10, no. 1, pp.3-22, 2009.

[3] E. Sularso, W. S. Raharjo, & Y. Lukito, "Implementsi Algoritma RijnDeal 128 pada Aplikasi Chatting Berbasis HTML WebSocket," Jurnal Infromatika, vol. 10, no. 2, pp. 66-79, 2014.

[4] Priyanto, D. Mahir Komputer Tanpa Kursus. Belajar Mudah Internet. Yogyakarta: MediaKom, 2009.

[5] Wardana. Menjadi Master PHP dengen Framework CodeIgniter. Jakarta: Elex Media Komputindo, 2010.

[6] Anonymous. Nodesource. [Online], https://nodesource.com/blog/understanding-socketio/, tanggal akses 6 Juli 2016.

[7] Anonymous. Foundation Zurb. [Online], http://foundation.zurb.com/learn/why-foundation.html/, tanggal akses 6 Juli 2016.

[8] Nugroho, A. Mengembangkan Aplikasi Basis Data Menggunakan C# dan SQL Server. Yogyakarta: Andi, 2010.

[9] Hasibuan, Z. A. Metodologi Penelitian Pada Bidang Ilmu Komputer dan Teknologi Informasi: Konsep, Teknik, dan Aplikasi. Jakarta: Ilmu Komputer Univesitas Indonesia, 2007.

[10] Pressman, R. S. Software Engineering: A Practitioner's Approach, 2001.