

Membangun

APLIKASI CHAT

dengan

Meteor Js & MongoDB



Muhammad Iskandar Dzulkornain

 [Fb.com/M.IskandarDzulkornain](https://fb.com/M.IskandarDzulkornain)

 id.linkedin.com/in/isdzulqor

 midzulqornain@gmail.com

Apa itu Meteor ?

Meteor adalah sebuah platform yang dibangun di atas Node.js untuk membuat aplikasi web real-time. Meteor bekerja di antara database aplikasi Anda dan antarmuka pengguna dan memastikan bahwa keduanya senantiasa selaras.

Karena dibangun di atas Node.js, Meteor menggunakan JavaScript baik pada sisi client maupun server. Terlebih lagi, Meteor juga dapat men-share kode antar-environment.

Hasil dari semua ini adalah sebuah platform yang sangat sederhana namun kapabel, yang menangani secara otomatis kerumitan-kerumitan dan kesulitan umum pengembangan aplikasi web.

<http://id.discovermeteor.com/chapters/introduction/>

<https://www.meteor.com/>

Apa itu MongoDB ?

MongoDB (dari kata “humongous”) adalah sebuah document oriented database yang bersifat open source. MongoDB merupakan salah satu database noSQL, yaitu sebuah konsep penyimpanan data non-relational. Istilah noSQL merupakan kepanjangan dari “Not Only SQL” yaitu sistem manajemen database yang berbeda dari sistem manajemen database relasional dalam beberapa cara. Penyimpanan data tanpa perlu adanya tabel schema dan tidak ada bahasa sql yang terlibat dalam pemakaian database.

MongoDB tidak mengenal adanya tabel, kolom dan baris jadi tidak ada schema dalam MongoDB (schema-less). Unit paling kecil dari MongoDB adalah document, sedangkan kumpulan dari document adalah collection. Seperti halnya dalam database relasional document, ibarat record dan collection sebuah tabel. document dalam MongoDB dapat memiliki atribut yang berbeda-beda dengan document yang lainnya walaupun dalam satu collection. Dalam MongoDB data tidak ditulis/dibaca dari database dengan menggunakan bahasa SQL, tetapi menggunakan metode object-oriented selain itu adanya banyak dukungan tipe index yang berbeda beda untuk lookups terhadap data tertentu dan memiliki kemampuan clustering secara default.

<http://litawismaayu.blog.ugm.ac.id/2013/09/17/mongodb/>

<https://www.mongodb.org/>

Hubungan antara Meteor Js dan MongoDB

Database MongoDB merupakan database bawaan alias database *default* yang sudah ter*include* ketika kita menginstall Meteor Js.

Kelebihan menggunakan Meteor Js

BLAZE

Blaze ini adalah salah satu JavaScript library yang membuat suatu template menjadi live-updating.

DPP

DPP atau singkatan dari Distributed Data Protocol merupakan protocol untuk mengambil data dari database dari server, dan menerima live-update ketika data diubah.

LIVEQUERY

Meteor Livequery masih merupakan keluarga dari live database connectors. Konektor ini memungkinkan kalian untuk melakukan “live query” pada database favorit kalian.

Persiapan

Berdoa agar bermanfaat pembelajaran yang kita lakukan :D.

INSTALASI METEOR JS DAN MONGODB

Untuk menginstall Meteor Js anda dapat mengunjungi link berikut

<https://www.meteor.com/install>

dan download installernya untuk Windows. Sedangkan untuk Linux anda dapat mengetikkan di terminal

```
curl https://install.meteor.com/ | sh
```

INSTALASI ROBOMONGO

Robomongo merupakan management tool untuk MongoDB berbasis GUI. Anda dapat mendownloadnya di <https://robomongo.org/download>.

INTEGRATED DEVELOPMENT ENVIRONMENT (IDE)

Sebenarnya terserah anda ingin menggunakan IDE apa, tapi di sini saya menggunakan PHP Storm karena saya suka dengan *autocompletenya*. Hehehe... :D. Di sini saya menggunakan PHP Storm for Student dengan masa berlaku 1 tahun. Untuk teman-teman yang ingin tau lebih lanjut mengenai product JetBrains for Education bisa ke link ini <https://www.jetbrains.com/student/>.

Let's Get Started !

Jika anda ingin melihat demo project yang akan kita buat. Anda bisa mengakses di url <https://youtu.be/2M0KuewNh5Q>.

Buat project anda dengan menggunakan cmd atau terminal. Di sini saya menggunakan Ubuntu. Masuk ke direktori yang anda inginkan dan ketikkan perintah di bawah ini.

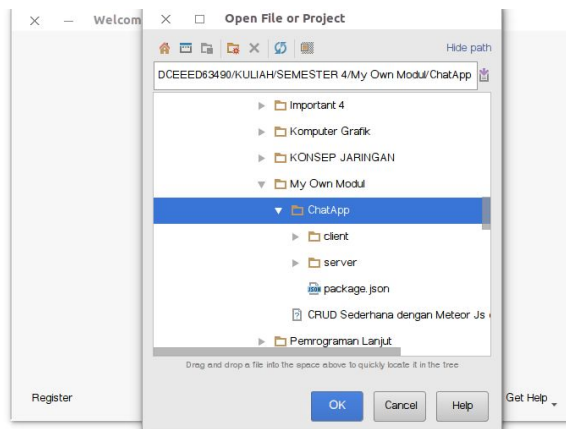
1. meteor create ChatApp

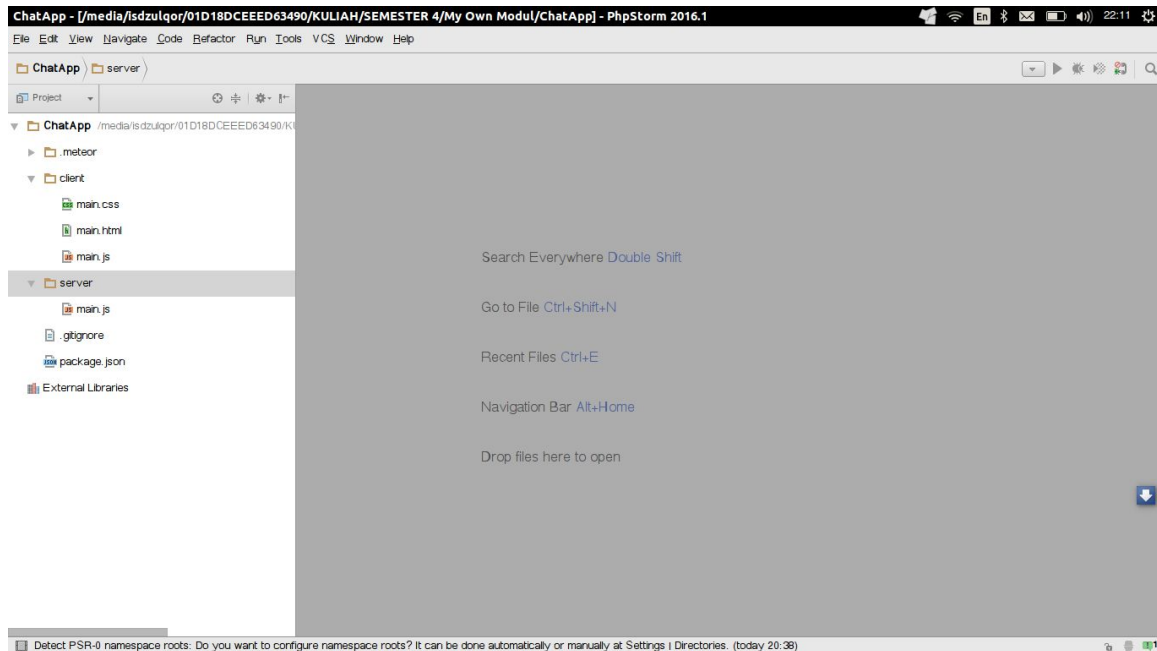
Maka otomatis akan tergenerate project dengan nama direktori ChatApp.

Langsung buka PHP Stormnya



Dan open project ChatApp yang telah kita buat dengan PHPStorm.





Jika terminal/cmd anda tidak muncul maka tekan

1. alt + f12

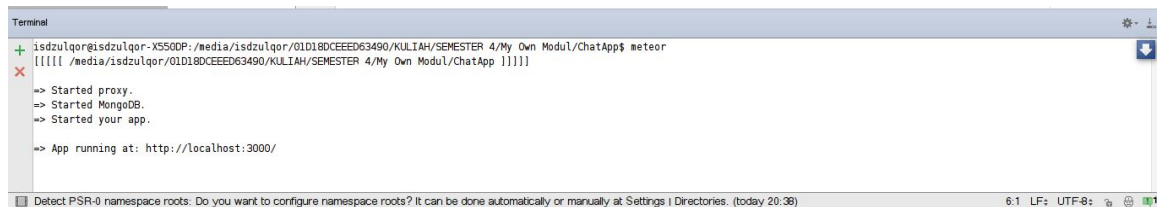
karena kita akan banyak bekerja dengan terminal/cmd ini.

Perhatikan struktur folder yang tergenerate tersebut terdapat tiga direktori, yaitu .meteor, server dan client.

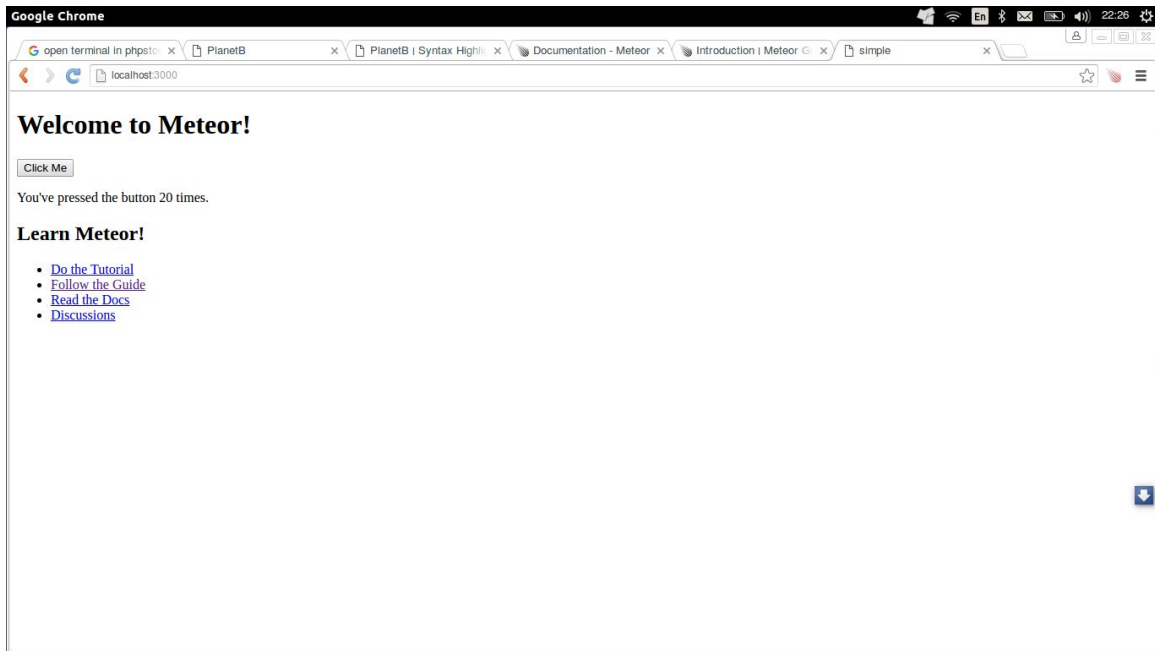
Anda dapat mencoba merunning project anda dengan mengetikkan di terminal/cmd.

1. meteor

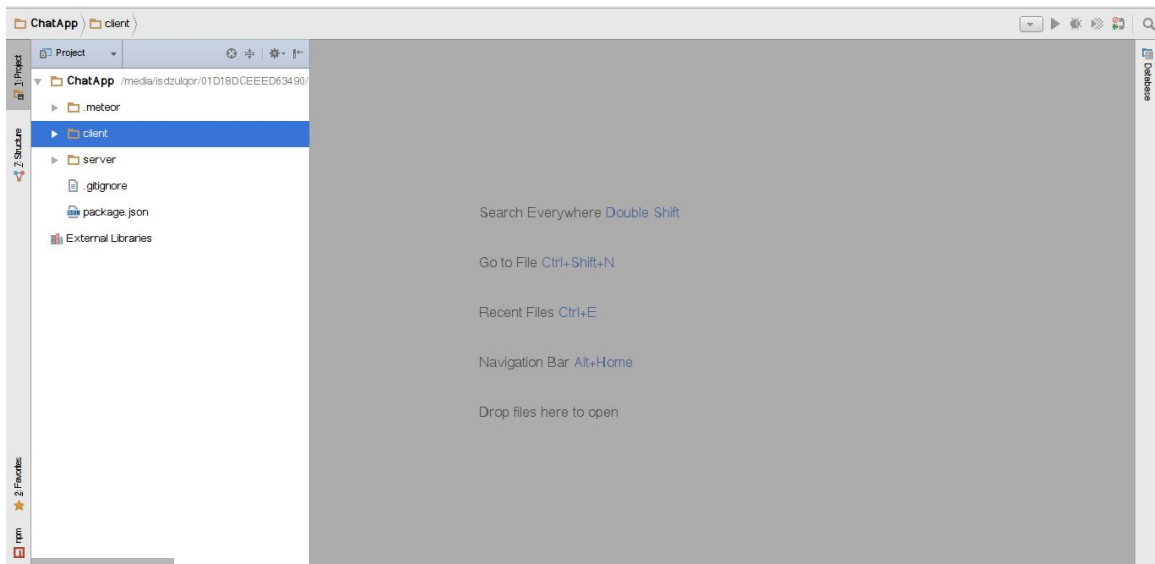
Pastikan cmd/terminal anda berada pada direktori root project anda.

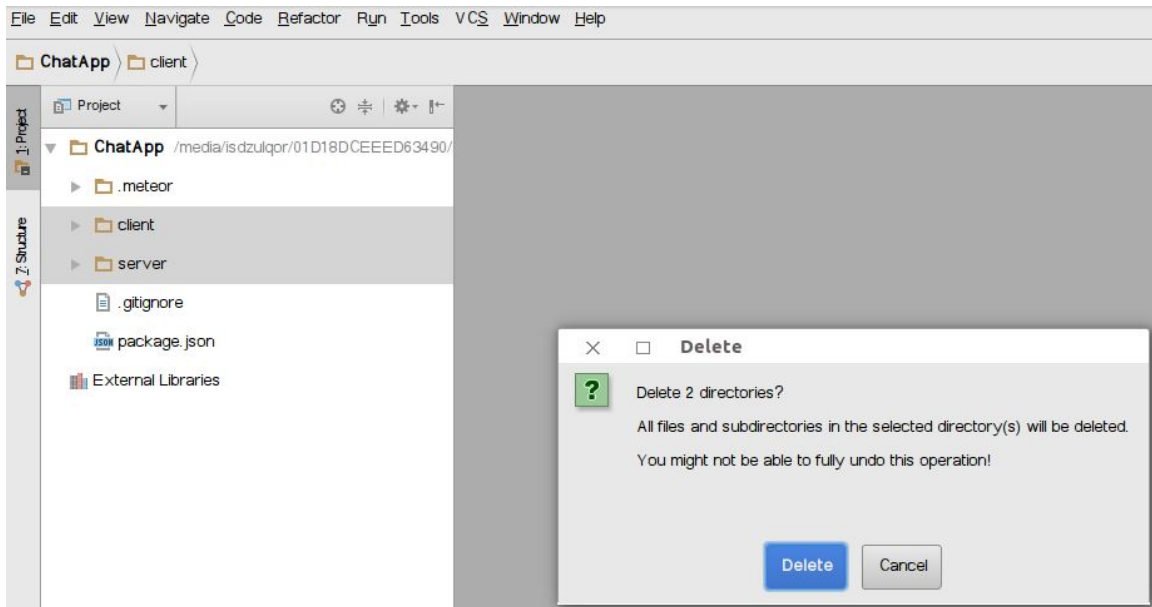


Dan akses browser anda dengan url <http://localhost:3000/>. Maka halaman yang ditampilkan akan menjadi seperti di bawah ini.

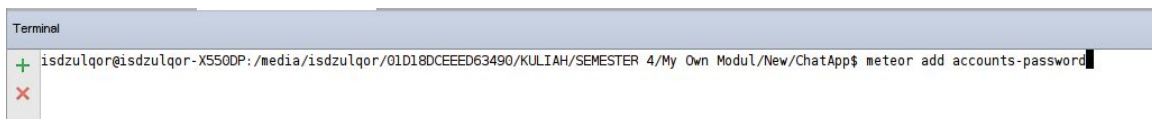


Kembali ke project kita yang ada di PHPStorm.





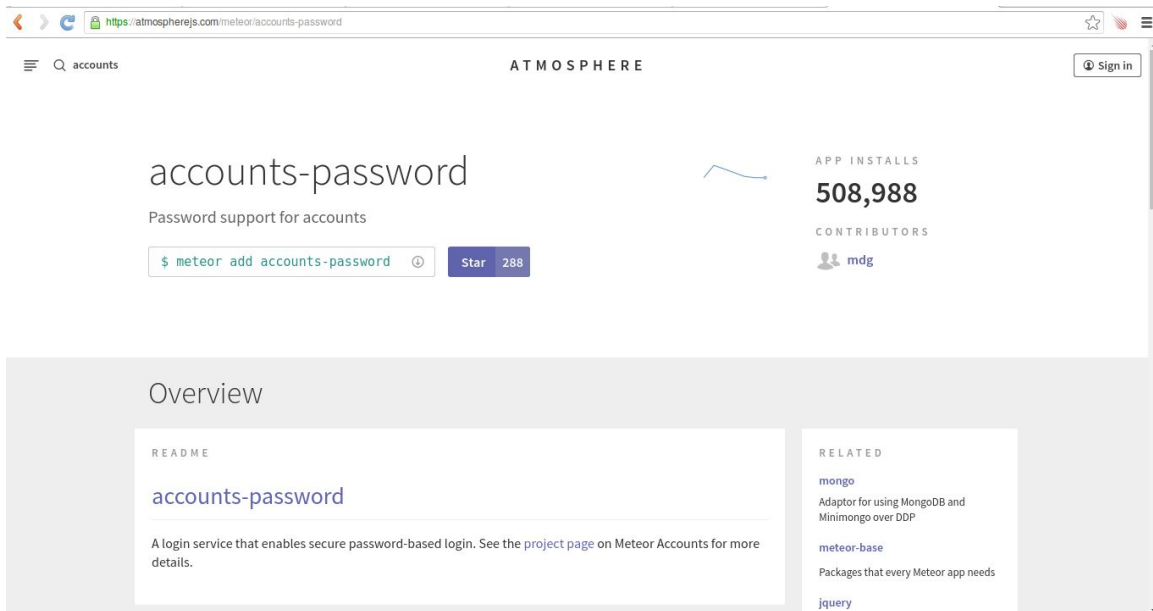
Hapus direktori server dan client, karena di sini kita akan membuat project baru yang benar-benar bersih.



Di terminal tambahkan dependency atau library accounts-password dengan cara mengetikkan

1. meteor add accounts/ password

Dan Enter. Komputer anda terkoneksi dengan internet. Accounts-password merupakan package atau dependency daripada meteor yang membantu kita dalam membuat form registasi. Untuk mengetahui lebih lanjut anda dapat mengunjungi link ini <https://atmospherejs.com/meteor/accounts-password>.



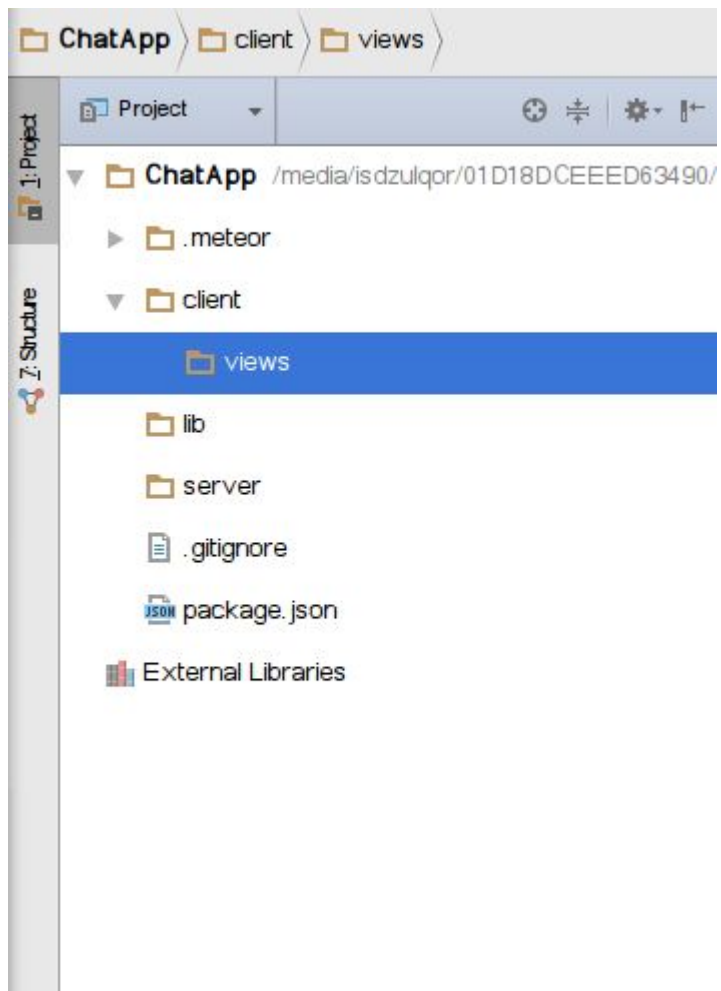
<https://atmospherejs.com> merupakan situs sumber package atau dependency dari meteor js.

Kemudian tambahkan juga dependency bootstrap dengan cara

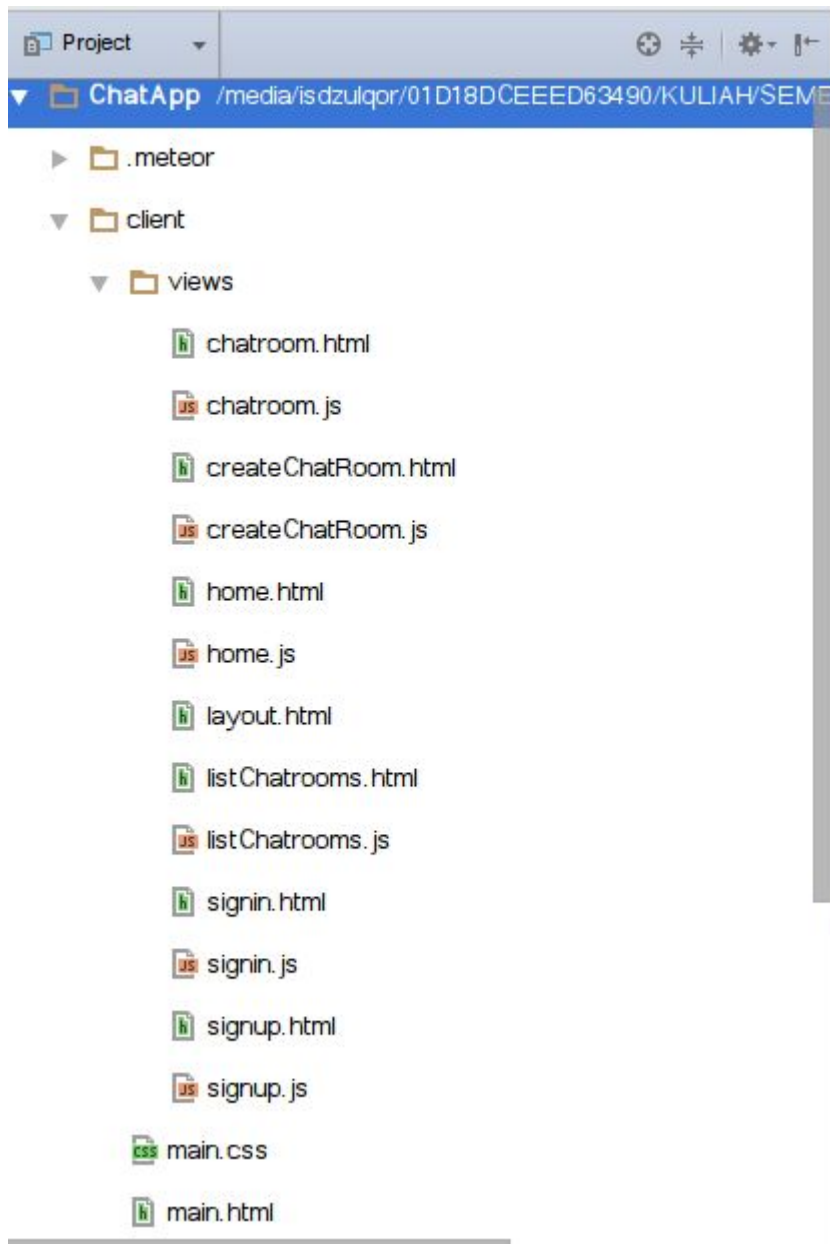
1. `meteor add twbs:bootstrap`

Ketikkan perintah di atas di terminal.

Setelah package `accounts-password` dan `twbs:bootstrap` tertambahkan di project anda, langkah selanjutnya adalah membuat tiga direktory utama.st



Tambahkan direktori server, lib dan client. Dan direktori views di bawah direktori client seperti gambar di atas.





Setelah itu anda dapat membuat file html dan javascript sesuai gambar di atas. Biarkan semua file masih kosong.

Penjelasan :

chatroom.html : Tampilan dari halaman chatroom
Chatroom.js : Javascript handler untuk tampilan chatroom.html

createChatRoom.html : Tampilan dialog pop up untuk penambahan chatroom
createChatRoom.js : Javascript handler untuk tampilan
CreateChatRoom.html

home.html : Tampilan dari halaman home
home.js : Javascript handler untuk tampilan **home.html**

layout.html : Verifikasi apakah user sudah login, jika sudah login diarahkan ke tampilan home jika belum ke tampilan sign in.

listChatrooms.html : Tampilan dialog pop up untuk daftar chatroom yang ada
listChatrooms.js : Javascript handler untuk tampilan ***listChatrooms.html***

signin.html : Tampilan dari halaman sign in
signin.js : Javascript handler untuk tampilan ***signin.html***

signup.html : Tampilan dari halaman sign up atau registrasi
signup.js : Javascript handler untuk tampilan ***signup.html***

main.css : Css tambahan yang ingin kita gunakan
main.html : Tampilan atau template utama dari aplikasi chat yang akan kita buat.
main.js : Javascript handler utama untuk semua template dan tampilan yang ada.

router.js : Javascript handle routing
Collection.js : Definsi dokumen database yang ada di project kita selain

dokumen user dan juga pendefinisian untuk skema dari dokumen tersebut.

Di dalam Meteor terdapat 3 cara untuk templating yaitu :

1. Menggunakan Angular Js
2. Menggunakan React Js
3. Menggunakan blaze

Templating dengan menggunakan blaze merupakan templating bawaan dari Meteor Js. Dan templating yang kita gunakan adalah menggunakan blaze.

Penambahan package-package yang dibutuhkan :

Sebelumnya kita sudah menambahkan package twbs:bootstrap dan accounts-password. Masih ada beberapa package tambahan yang diperlukan. Yaitu :

1. Meteor add Session

Pada Meteor di bawah versi 1.3 package session sudah terinstall di Meteor Js. Tapi untuk versi 1.3 ke atas anda harus menambahkannya. Fungsi dari package ini adalah untuk membuat data session sementara. Data session tersebut akan terset ke default jika web kita reload/refresh.

2. Meteor add iron:router

Iron:router berfungsi untuk handling routing dari website kita.

3. Meteor add aldeed:collection2

Package ini berfungsi untuk membuat skema dan susunan dari dokumen(tabel dalam sql) database kita. Karena pada dasarnya database nosql tidak terdapat susunan alias skemanya. Jadi kita dapat menambahkan dengan seenaknya. Tapi dengan package aldeed:collection2 ini kita bisa mengkonfigurasi field-field apa saja yang harus ada dalam skema kita, dan field apa saja yang nilainya tidak boleh null dan lain sebagainya.

START TO CODE

FIRST STEP : MEMBUAT TAMPILAN SIGN IN

Tuliskan kodingan berikut ke dalam file **signin.html**

```
<template name="signin">
  <div class="top-content">
    <div class="inner-bg">
      <div class="container">
        <div class="row">
          <div class="col-sm-8 col-sm-offset-2 text">
            <h1><strong>Chat</strong>App</h1>
            <div class="description">
              <p>
                Aplikasi Chatting dengan Meteor Js dan MongoDB
              </p>
            </div>
          </div>
        </div>
        <div class="row">
          <div class="col-sm-6 col-sm-offset-3 form-box">
            <div class="form-top">
              <div class="form-top-left">
                <h3>Masuk</h3>
                <p>Masukkan username dan password anda</p>
              </div>
              <div class="form-top-right">
                <i class="fa fa-key"></i>
              </div>
            </div>
            <div class="form-bottom">
              <form role="form" action="" method="post"
class="login-form">
                <div class="form-group">
                  <label class="sr-only"
for="form-username">Username</label>
                  <input type="text" name="form-username"
placeholder="Username..." class="form-username form-control" id="form-username">
                </div>
                <div class="form-group">
                  <label class="sr-only"
for="form-password">Password</label>
                  <input type="password" name="form-password"
placeholder="Password..." class="form-password form-control" id="form-password">
                </div>
                <button type="submit" class="btn">Sign in!</button>
                <h6>Don't have account?</h6>
                <a href="{{pathFor 'signup'}}">Create account</a>
              </form>
            </div>
          </div>
        </div>
      </div>
    </div>
  </div>
</template>
```

Dan tuliskan kode berikut ke dalam file **signin.js**

```
Template.signin.events({
  'submit form':function (e, tpl) {
    e.preventDefault();
    var usernameVar = tpl.find('#form-username').value;
    var passwordVar = tpl.find('#form-password').value;
    Meteor.loginWithPassword(usernameVar, passwordVar, function (e) {
      if(e){
        alert("Login Error \n" + e);
      }else{
        Router.go('/');
      }
    });
  });
});
```

Yang perlu diperhatikan :

```
1 Template.signin.events({
2   'submit form':function
```

```
1 <template name="signin">
2   <div class="container">
```

File signin harus sesuai dengan nama template yang kita buat di html. Di file javascript di atas saya mendefinisikan event-event yang ada di file signin.html. Tetapi untuk event di atas cuma terdapat satu event. Yaitu event **submit** untuk tag form yang ada di html file signin.html.

```
var passwordVar = tpl.find('#form-password').value;
Meteor.loginWithPassword(usernameVar, passwordVar, function (e) {
  if(e){
```

Meteor.loginWithPassword merupakan method atau fungsi dari package accounts-password yang berfungsi untuk mengecek data inputan user apakah sudah terdaftar sebagai user atau belum.

Pendefinisian event seperti di atas akan juga terjadi di template-template lain yang kita buat.

SECOND STEP : MEMBUAT TAMPILAN SIGN UP

Tuliskan kodingan berikut ke dalam file **signup.html**

```
<template name="signup">
  <div class="top-content">
    <div class="inner-bg">
      <div class="container">
        <div class="row">
          <div class="col-sm-8 col-sm-offset-2 text">
            <h1><strong>Chat</strong>App</h1>
            <div class="description">
              <p>
                Aplikasi Chatting dengan Meteor Js dan MongoDB
              </p>
            </div>
          </div>
        </div>
      </div>
    </div>
  </div>
</template>
```



```

        window.alert("Isikan data dengan benar");
    }
    Accounts.createUser({
        email: emailVar,
        username: usernameVar,
        password: passwordVar
    }, function (err) {
        if(err){
            window.alert("Error \n"+err);
            console.log(err);
        }
        else{
            console.log("success");
            window.alert("Sukses");
            Router.go('/home');
        }
    });
}

```

Accounts.createUser merupakan method untuk melakukan penambahan user ke database MongoDB. Method ini merupakan bawaan dari package accounts-password.

Tuliskan kodingan berikut ke dalam file **home.html**

Aplikasi Chat dengan Meteor Js & MongoDB
Muhammad Iskandar Dzulqornain


```

    {{/if}}
    {{#if showListDialog}}
      {{> listChatrooms}}
    {{/if}}
  <div>
    <div class="portlet portlet-default">
      <div class="portlet-heading">
        <div class="portlet-title">
          <ul class="nav nav-pills" style="float: right">
            <li role="presentation"><a href="#"
id="createChatRoom">Create Chatroom</a></li>
            <li role="presentation"><a href="#"
id="listChatRoom">List Chatroom</a></li>
          </ul>
          <h2>Aplikasi Chatting</h2>
          <h3>{{ currentUser.username }}</h3>
          <a href="#" id="logout">Log Out</a>
        </div>
        <div class="clearfix"></div>
      </div>
      <div id="chat" class="panel-collapse collapse in">
        <div>
          <div id="always-scoll" class="portlet-body chat-widget"
style="overflow-y: auto; overflow-x: hidden; width: auto; height: 420px">
            <div class="row">
              <div class="col-lg-12">
                <p class="text-center text-muted small"
style="float: right"></p>
              </div>
            </div>
            {{#each chat}}
              <div class="row">
                <div class="col-lg-12">
                  <div class="media">
                    <div class="media-body">
                      <h4 class="media-heading">{{username}}
pull-right">{{formatDate timestamp}}</span>
                    </h4>
                    <p>
                      {{text}}
                    </p>
                    {{#if canDelete username}}
                      <a href="#"
class="delete"><h6>Delete</h6></a>
                    </if>
                  </div>
                </div>
              </div>
            </div>
          </div>
          <hr>
          {{/each}}
        </div>
      </div>
      <div class="portlet-footer">
        <form role="form add-chat">
          <div class="form-group">
            <textarea class="form-control" placeholder="Enter
message..." id="text-msg"></textarea>
          </div>
        </form>
      </div>
    </div>
  </div>

```



```

    },
    chat: function () {
        return Messages.find({
            chatRoomId: { $exists : false }
        }, { sort: { timestamp: 1 }});
    },
    formatDate: function(timestamp) {
        var date = new Date(timestamp);
        return date.toLocaleString();
    },
    canDelete: function (username) {
        if(username == Meteor.user().username)
            return true;
        else
            return false;
    }
});

var addChat = function (text) {
    var id = Messages.insert({
        username: Meteor.user().username,
        text: text,
        timestamp: Date.now()
    });
    if(id==0){
        alert("insert error");
    }
};

var deleteChat = function (_id) {
    Messages.remove(_id);
};

Template.home.onRendered(function () {
    var objDiv = document.getElementById("always-scoll");
    objDiv.scrollTop = objDiv.scrollHeight;
});

```

Yang perlu diperhatikan :

```

<template name="home">
  {{#if currentUser}}
  <br>
  </if>
</template>

```

Tulisan `{{#if currentUser}}` juga merupakan bawaan dari package `accounts-password`. Dimana **currentUser** merupakan pengecekan apakah halaman ini di akses oleh user yang terdaftar.

Ketika ada **if** pembuka maka pasti ada **if** penutup bahkan juga ada **else**.

```

</div>
{{else}}
  {{ toLayout }}
{{/if}}

```

Else dan If penutup berada di bawah. Dimana jikalau user belum terdaftar maka akan di panggil {{ toLayout }}.

Apa itu {{ to Layout }} ???

Meteor Js ini menggunakan handlebar.js sebagai templatingnya. Jadi penulisan dengan {{ }} merupakan penulisan handlebar.js. Dan pemrosesannya berada di sisi javascript.

Dan kita bisa lihat di file **home.js** terdapat seperti ini

```

Template.home.helpers({
  toLayout: function () {
    Router.go('/');
  },
  showCreateDialog: function () {
    return Session.get('showCreateDialog');
  },
  showListDialog: function () {
    return Session.get('showListDialog');
  }
});

```

Bisa kita lihat gambar di atas, perhatikan bahwa koding di atas merupakan pendefinisian helpers untuk file template home. Dan terdapat

```

toLayout : function () {
  .....
}

```

ToLayout di atas merupakan fungsi atau method yang dipanggil ketika kita mendefinisikan {{ toLayout }} di file **home.html**. Dan bisa kita lihat di dalam method atau helper tersebut dia memanggil **Router.go('/')** ;

Router.go('/') merupakan fungsi bawaan dari iron:router. Dimana dia akan redirect ke url root dengan memanggil ('/').

Dan di bawahnya terdapat helper

```

showCreateDialog: function(){
  .....
}

```

Yang mana dia mengembalikan value yang ada di session dengan key showCreateDialog.

```

{{/if}}
{{#if showListDialog}}
  {{> listChatRooms}}
{{/if}}
</div>
<div class="portlet portlet-default">
  <div class="portlet-heading">

```

Dapat kita lihat potongan kodingan di atas terdapat handlebar dengan penulisan `{{> listChatRooms}}`.

Apa itu `{{> listChatRooms}}` ???

Penulisan tersebut merupakan pemanggilan template dengan nama `listChatRooms`. Jadi kalo digantikan dengan template maka akan tertulis seperti di bawah ini.

```

<template name="listChatRooms">
  .....
</template>

```

Helper atau method dengan mengirimkan parameter

```

</p>
{{#if canDelete username}}
<a href="#"
right">{{formatDate timestamp}}</span>

```

Dapat di lihat gambar di atas merupakan pemanggilan helper dengan parameter yang satu **canDelete** dengan parameter **username** Dan **formatDate** dengan parameter **timestamp**

Bagaimana cara menangkap parameter tersebut di javascript ????

Lihat file **home.js** terdapat koding seperti ini

```

},
formatDate: function(timestamp) {
  var date = new Date(timestamp);
  return date.toLocaleString();
},
canDelete: function (username) {
  if (username == Meteor.user().username)
    return true;
  else
    return false;
}

```

Lihat gambar di atas parameter dia tangkap sedemikan rupa kemudian dilakukan pemrosesan data.

Interaksi dengan Database

```

var addChat = function (text) {
  var id = Messages.insert({
    username: Meteor.user().username,
    text: text,
    timestamp: Date.now()
  });
  if(id==0){
    alert("insert error");
  }
};

var deleteChat = function (_id) {
  Messages.remove(_id);
};

```

Perhatikan file **home.js** terdapat kodingan seperti di atas. Koding di atas merupakan salah satu cara pendefinisian fungsi atau method di javascript. Dimana pada method addChat dia membutuhkan satu parameter yaitu parameter text yang diinput user. Kemudian dilakukan insert data ke dalam dokumen **Messages** yang ada di database.

Dan di method deleteChat juga membutuhkan satu parameter, yaitu id dari dokumen **Messages** yang ingin kita hapus.

Deklarasi **Messages** akan kita definisikan file collection.js di bawah direktori lib.

Retrieving data dari Database

```

{{#each chat}}
  <div class="row">
    <div class="col-lg-12">
      <div class="media">
        <div class="media-body">
          <h4 class="media-heading">{{username}}
            <span class="small
pull-right">{{formatDate timestamp}}</span>
          </h4>
          <p>
            {{text}}
          </p>
          {{#if canDelete username}}
            <a href="#"
class="delete"><h6>Delete</h6></a>
          {{/if}}
        </div>
      </div>
    </div>
  </div>
</div>
<hr>
{{/each}}

```

Perhatikan kodingan di atas. Terdapat beberapa handlebar. Di atas sendiri kita dapat tulisan `{{#each chat}}` .

Apa itu `{{#each chat}}` ????

Handlebar tersebut berarti looping sebanyak data yang ada di method chat. Kita lihat di file **home.js** mengenai kodingan chat ini.

```

},
chat:function () {
  return Messages.find({
    chatRoomId: { $exists : false }
  }, { sort: { timestamp: 1 }});
}

```

Perhatikan kodingan di atas. Return data dari method chat merupakan data yang di dapat dari database. Bisa di lihat data tersebut di dapat dari dokumen **Messages**.

Perhatikan lagi gambar kodingan ke dua sebelah atas dari kodingan gambar di atas. Terdapat tulisan {{username}}.

{{username}} merupakan value field yang ada di dokumen **Messages**. Begitu juga dengan timestamp yang di tulis seperti ini {{ formatDate timestamp }} . Itu berarti timestamp di jadikan parameter untuk method formatDate. Dimana timestamp merupakan value yang ada di dokumen **Messages**.

FOURTH STEP : MEMBUAT TAMPILAN DIALOG POP UP CREATECHATROOM

Tuliskan kodingan berikut ke dalam file **createChatRoom.html**

```

<template name="createChatRoom">
  <div id="modal-id">
    <div class="modal-dialog">
      <div class="modal-content">
        <div class="modal-header">
          <button type="button" class="close exit" data-dismiss="modal"
aria-hidden="true">&times;</button>
          <h4 class="modal-title">Create Chatroom</h4>
        </div>
        <div class="modal-body">
          <input type="text" name="name" class="form-control" id="name"
placeholder="Chatroom name">
        </div>
        <div class="modal-footer">
          <button type="button" class="btn btn-success add
create">Create</button>
          <button type="button" class="btn cancel exit"
data-dismiss="modal">Close</button>
        </div>
      </div><!-- /.modal-content -->
    </div><!-- /.modal-dialog -->
  </div><!-- /.modal -->
</template>

```

Dan tuliskan kodingan berikut ke dalam file **createChatRoom.js**

```

Template.createChatRoom.events({
  'click .exit':function (e) {
    e.preventDefault();
    Session.set('showCreateDialog', false);
  },
  'click .create':function (e, tpl) {
    e.preventDefault();
    var name = tpl.find('#name').value;
    addChatroom(name);
    tpl.find('#name').value = '';
  }
});

```

```

        Session.set('showCreateDialog', false);
    }
});

var addChatroom = function (name) {
    var id = ChatRooms.insert({
        userId: Meteor.user().username,
        name: name,
        timestamp: Date.now()
    });
    if(id==0){
        alert("insert error");
    }else{
        alert("insert success");
    }
};

```

FIFTH STEP : MEMBUAT TAMPILAN DIALOG POP UP LISTCHATROOMS

Tuliskan kodingan berikut ke dalam file **listChatrooms.html**

```

<template name="listChatrooms">
    <div id="modal-id">
        <div class="modal-dialog">
            <div class="modal-content">
                <div class="modal-header">
                    <button type="button" class="close exit" data-dismiss="modal"
aria-hidden="true">&times;</button>
                    <h4 class="modal-title">List Available Chatrooms</h4>
                </div>
                <div class="modal-body">
                    <div class="list-group">
                        {{#each chatrooms}}
                            <a href="{{pathFor route='chatroom' _id=id
chatroom_name=name}}" class="list-group-item">{{name}}</a>
                        {{/each}}
                    </div>
                </div>
                <div class="modal-footer">
                    <!--<button type="button" class="btn btn-success
add">Join</button-->
                    <button type="button" class="btn cancel exit"
data-dismiss="modal">Close</button>
                </div>
            </div><!-- /.modal-content -->
        </div><!-- /.modal-dialog -->
    </div><!-- /.modal -->
</template>

```

Dan tuliskan kodingan berikut ke dalam file **listChatrooms.js**

```

Template.listChatrooms.events({
    'click .exit':function (e) {
        e.preventDefault();
        Session.set('showListDialog', false);
    }
});

```



```

Template.listChatrooms.helpers({
  chatrooms:function () {
    return ChatRooms.find({}, { sort: { timestamp: -1 }});
  }
});

```

SIXTH STEP : MEMBUAT TAMPILAN CHATROOM

Tuliskan kodingan berikut ke dalam file **chatroom.html**

```

<template name="chatroom">
  {{#if currentUser}}
  <div class="container bootstrap snippet">
    <div class="row">
      {{#if showCreateDialog}}
      {{> createChatRoom}}
      {{/if}}
      {{#if showListDialog}}
      {{> listChatrooms}}
      {{/if}}
    <div>
      <div class="portlet portlet-default">
        <div class="portlet-heading">
          <div class="portlet-title">
            <ul class="nav nav-pills" style="float: right">
              <li role="presentation"><a href="#"
id="createChatRoom">Create Chatroom</a></li>
              <li role="presentation"><a href="#"
id="listChatRoom">List Chatroom</a></li>
            </ul>
            <a href="{{pathFor route='home'}}" ><h4>Back to
Home</h4></a>
          <h2>{{chatroomName}}</h2>
          <a href="#" id="logout">Log Out</a>
        </div>
        <div class="clearfix"></div>
      </div>
      <div id="chat" class="panel-collapse collapse in">
        <div>
          <div id="always-scoll" class="portlet-body chat-widget"
style="overflow-y: auto; overflow-x: hidden; width: auto; height: 420px">
            <div class="row">
              <div class="col-lg-12">
                <p class="text-center text-muted small"
style="float: right"></p>
              </div>
            </div>
            {{#each chat}}
            <div class="row">
              <div class="col-lg-12">
                <div class="media">
                  <div class="media-body">
                    <h4 class="media-heading">{{username}}
                    <span class="small
pull-right">{{formatDate timestamp}}</span>
                  </h4>
                  <p>
                    {{text}}

```

```

class="delete"><h6>Delete</h6></a>
    </p>
    {{#if canDelete username}}
    <a href="#"
    {{/if}}
  </div>
</div>
</div>
</div>
<hr>
{{/each}}
</div>
</div>
<div class="portlet-footer">
  <form role="form add-chat">
    <div class="form-group">
      <textarea class="form-control" placeholder="Enter
message..." id="text-msg"></textarea>
    </div>
    <div class="form-group">
      <button type="button" class="btn btn-default
pull-right add-chat ">Send</button>
      <div class="clearfix"></div>
    </div>
  </form>
</div>
</div>
</div>
</div>
</div>
</div>
{{else}}
{{ tolayout }}
{{/if}}
</template>

```

Dan tuliskan kodingan berikut ke dalam file **chatroom.js**

```

Template.chatroom.helpers({
  tolayout: function () {
    Router.go('/');
  },
  showCreateDialog: function () {
    return Session.get('showCreateDialog');
  },
  showListDialog: function () {
    return Session.get('showListDialog');
  },
  formatDate: function(timestamp) {
    var date = new Date(timestamp);
    return date.toLocaleString();
  },
  canDelete: function (username) {
    if(username == Meteor.user().username)
      return true;
    else
      return false;
  }
});

```

```

Template.chatroom.events({
  'click #logout':function (e) {
    e.preventDefault();
    Meteor.logout();
    Router.go('/');
  },
  'click #createChatRoom':function (e) {
    e.preventDefault();
    Session.set('showCreateDialog', true);
  },
  'click #listChatRoom':function (e) {
    e.preventDefault();
    Session.set('showListDialog', true);
  },
  'click .add-chat':function (e, tmpl) {
    e.preventDefault();
    var text = tmpl.find('#text-msg').value;
    var _id = Router.current().params._id;
    addChatInChatroom(text, _id);
    var objDiv = document.getElementById("always-scoll");
    objDiv.scrollTop = objDiv.scrollHeight;
    tmpl.find('#text-msg').value = '';
  },
  'click .delete':function (e, tmpl) {
    e.preventDefault();
    deleteChat(this._id);
  }
});

var addChatInChatroom = function (text, chatroom_id) {
  var id = Messages.insert({
    username: Meteor.user().username,
    text: text,
    timestamp: Date.now(),
    chatRoomId: chatroom_id
  });
  if(id==0){
    alert("insert error");
  }
};

var deleteChat = function (_id) {
  Messages.remove(_id);
};

Template.chatroom.onRendered(function () {
  var objDiv = document.getElementById("always-scoll");
  objDiv.scrollTop = objDiv.scrollHeight;
});

```

SEVENTH STEP : MEMBUAT TAMPILAN LAYOUT

Tuliskan kodingan berikut ke dalam file **layout.html**

```
<template name="layout">
  {{#if currentUser}}
    {{> home}}
  {{else}}
    {{> signin}}
  {{/if}}
</template>
```

Menambahkan sedikit css Tuliskan koding d bawah ini. Buka **main.css**

```
#modal-id{
  position: fixed;
  top: 0;
  right: 0;
  bottom: 0;
  left: 0;
  z-index: 1050;
}
```

Tuliskan koding seperti di bawah ini di file main.html

```
<head>
  <title>Chat App</title>
</head>
<body>

</body>

<template name="main">
  {{> yield}}
</template>
```

{{ >yield }} merupakan pendefinisian untuk tumpuan layout karena template main merupakan layout template dari aplikasi kita yang kita definisikan di dalam file router.js.

Inisialisai Dokumen database dan skemanya

Tuliskan koding di bawah ini untuk file collection.js

```
Messages = new Mongo.Collection('messages');
var Schemas = {};
Schemas.Messages = new SimpleSchema({
  text: {
    type: String
  },
  timestamp: {
    type: Number
  },
  username: {
    type: String
  },
  chatRoomId: {
    type: String,
```

```

        optional: true
    }
});
Messages.attachSchema(Schemas.Messages);

ChatRooms = new Mongo.Collection('chatRooms');
Schemas.ChatRooms = new SimpleSchema({
    name: {
        type: String
    },
    userId: {
        type: String
    },
    timestamp: {
        type: Number
    }
});
ChatRooms.attachSchema(Schemas.ChatRooms);

```

Kodingan di atas merupakan inisialisasi dokumen database kita. Terdapat 2 dokumen yaitu Messages dan ChatRooms. Sebenarnya terdapat satu dokumen lagi yaitu dokumen Users. Tapi dokumen Users sudah otomatis ada pada project Meteor kita tanpa kita definisikan. Yang dapat digunakan dengan package accounts-password.

EIGHTH STEP : HANDLING ROUTING DENGAN IRON:ROUTER

Tuliskan kodingan berikut ke dalam file **router.js**

```

/**
 * Created by isdzulqor on 17/04/16.
 */
Router.configure({
    layoutTemplate: 'main'
});
Router.route('/', {
    name: 'layout'
});
Router.route('/signup', {
    name: 'signup'
});
Router.route('/home', {
    name: 'home'
});

Router.route('/chatroom/:_id/name/:chatroom_name', {
    name: 'chatroom',
    data: function() {
        return {
            chatroomName: this.params.chatroom_name,
            chat: Messages.find({
                chatRoomId: this.params._id
            },
            { sort:
              { timestamp: 1 }}
            )
        };
    }
});

```

```
});
```

Yang perlu diperhatikan :
Terdapat kodingan seperti ini

```
Router.route('/chatroom/:_id/name/:chatroom_name', {
  name: 'chatroom',
  data: function() {
    return {
      chatroomName: this.params.chatroom_name,
      chat: Messages.find({
        chatRoomId: this.params._id
      }, {
        sort: {
          timestamp: 1
        }
      })
    };
  }
});
```

Ini berarti ketika user mengakses halaman chatroom diperlukan 2 parameter. Yaitu parameter `_id` dan parameter `chatroom_name`.

Dan pada template chatroom terdapat return 2 data yaitu **chatroomName** dan data **chat**. Perhatikan file chatroom.html terdapat kodingan seperti di bawah ini

```
st Chatroom</a></li>
</ul>
<a href="{{pathFor route='home'}}" >
<h2>{{chatroomName}}</h2>
<a href="#" id="logout">Log Out</a>
</div>
```

Perhatikan `{{chatroomName}}` merupakan data return yang kita definisikan di `router.js`

```
{{#each chat}}
<div class="row">
  <div class="col-lg-12">
    <div class="media">
      <div class="media-body">
        <h4 class="media-heading">{{username}}
        <span class="small
pull-right">{{formatDate timestamp}}</span>
      </h4>
      <p>
        {{text}}
      </p>
    </div>
  </div>
</div>
```

Begitu juga dengan `{{ #each chat }}` merupakan data return yang kita definisikan di `router.js` yang kita dapat data tersebut dari database dengan query berikut

```

chatroomName: this.params.chatroom name,
chat: Messages.find({
  chatRoomId: this.params._id
}),
{ sort:
{ timestamp: 1 }}
)

```

Building Aplikasi

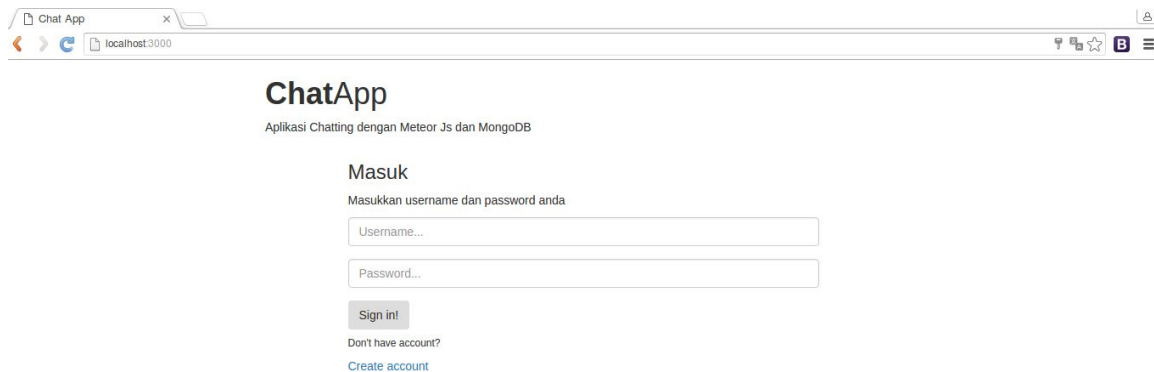
Sekarang anda bisa mencoba membuild aplikasi kita dengan cara seperti ini

```

Terminal
+ isdzulqor@isdzulqor-X550DP:/media/isdzulqor/01D18DCEED63490/KULIAH/SEMESTER 4/My Own Modul/New/ChatApp$ meteor
X [|||| /media/isdzulqor/01D18DCEED63490/KULIAH/SEMESTER 4/My Own Modul/New/ChatApp ]||||
=> Started proxy.
=> Started MongoDB.
█ Selecting package versions |

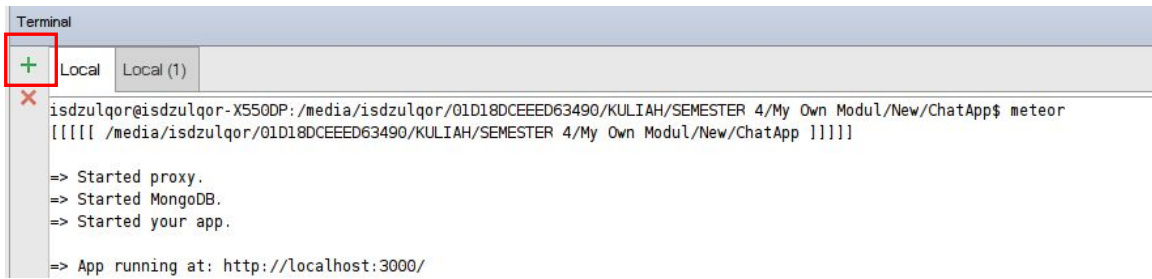
```

Ketikkan meteor di terminal. Setelah proses buid selesai. Akses browser anda dengan alamat localhost:3000. Maka akan tampil seperti di bawah ini.



Aplikasi kita siap digunakan.

Untuk dapat melihat kumpulan dari dokumen yang kita inisialisasikan di aplikasi kita. Kita bisa menggunakan robomongo. Caranya seperti di bawah ini



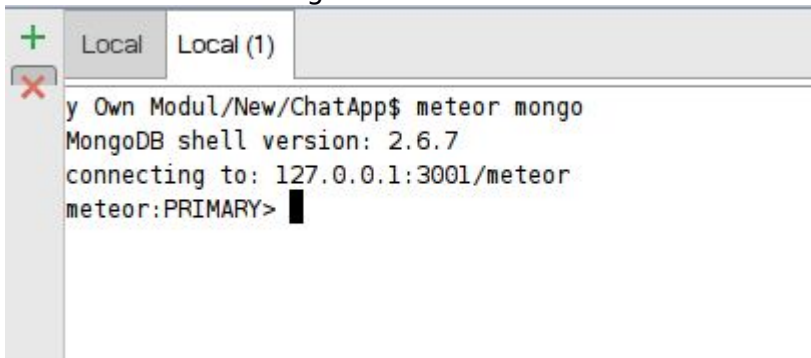
```
Terminal
+ Local Local (1)
isdzulqor@isdzulqor-X550DP:/media/isdzulqor/01D18DCEED63490/KULIAH/SEMESTER 4/My Own Modul/New/ChatApp$ meteor
[[[[[ /media/isdzulqor/01D18DCEED63490/KULIAH/SEMESTER 4/My Own Modul/New/ChatApp ]]]]]

=> Started proxy.
=> Started MongoDB.
=> Started your app.

=> App running at: http://localhost:3000/
```

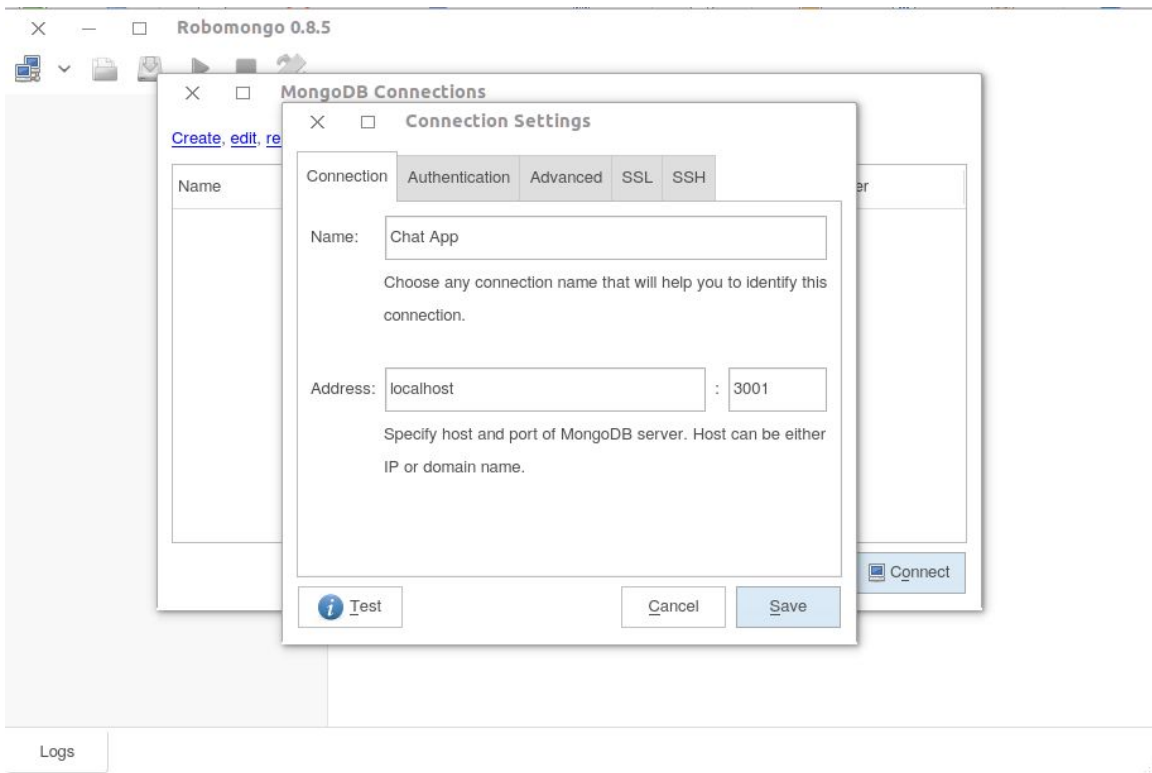
Tekan icon + untuk menambah terminal baru. Pastikan untuk membuka robomongo ini projek aplikasi kita juga harus running.

Ketikkan meteor mongo dan Enter.

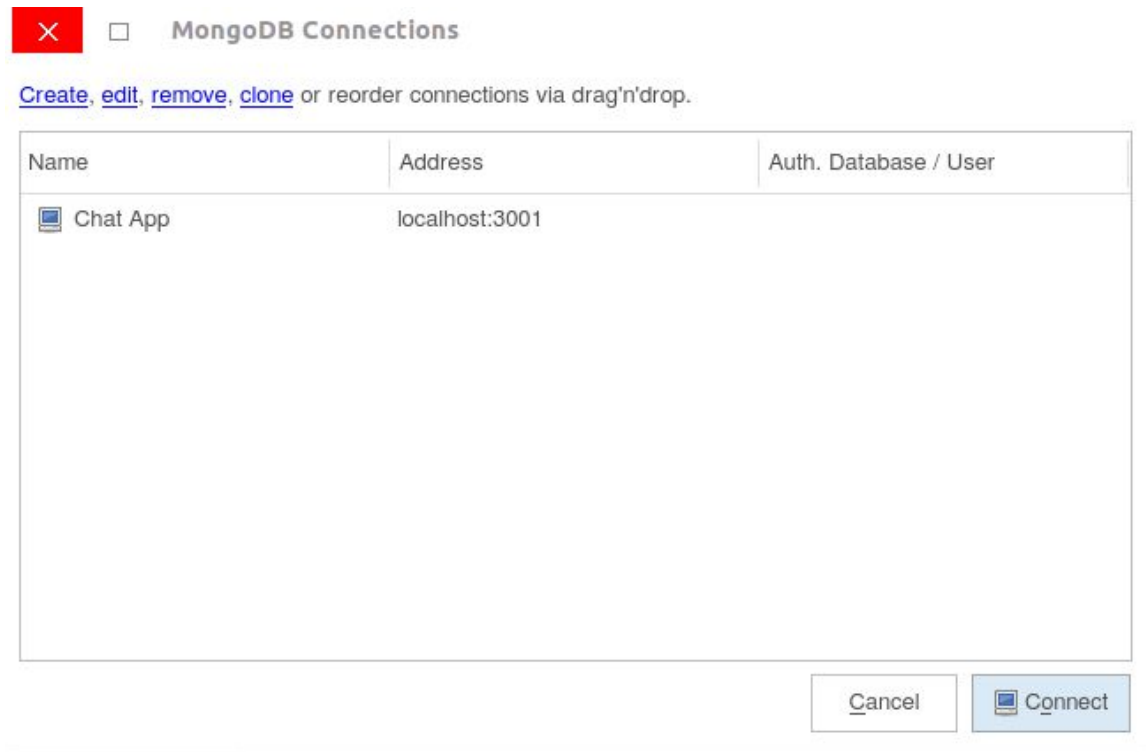


```
+ Local Local (1)
y Own Modul/New/ChatApp$ meteor mongo
MongoDB shell version: 2.6.7
connecting to: 127.0.0.1:3001/meteor
meteor:PRIMARY>
```

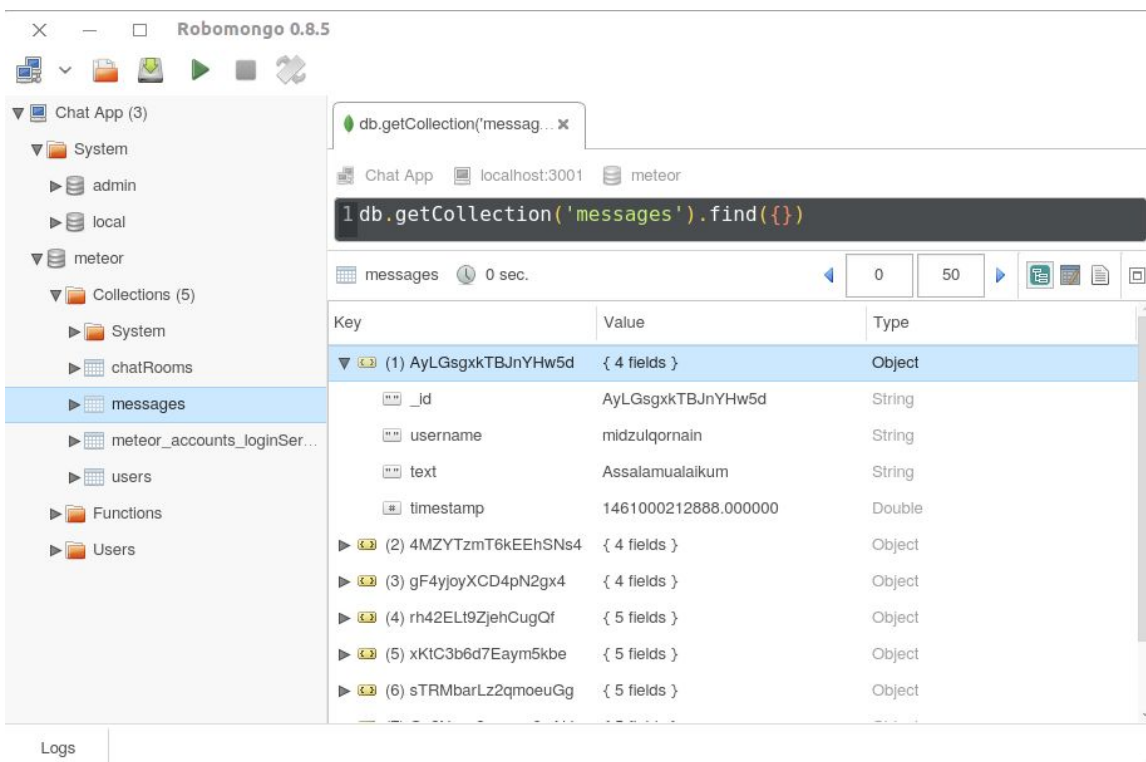
Jalankan Robomongo anda



Create New Connection. Dan Isi field nama koneksi anda, address atau alamat dan portnya. Lalu Save.



Langsung saja pilih koneksi anda dan klik connect.



Anda sudah dapat melihat dokumen database anda.

Demikian tutorial singkat mengenai pembuatan aplikasi chat dengan menggunakan Meteor Js dan MongoDB database. Semoga dapat bermanfaat.

Untuk demo aplikasinya bisa menuju ke link ini <https://youtu.be/2M0KuewNh5Q>.

Dan untuk source code yang sudah jadi anda dapat ke ke sini.
<https://github.com/isdzulqor/Chat-App-with-Meteor-Js>

Sekian, Terima kasih.

-