



Desenvolvimento de Aplicações Web

Trabalho Prático

Hugo Gonçalves (PG38928)

Luis Bouça (PG38933)

janeiro, 2019

Área de Aplicação: Desenvolvimento de Aplicações Web

Palavras-chave: *MongoDB, NodeJs, JQuery, AJAX, Pug, Express*

Conteúdo

Introdução	1
1 <i>MongoDB</i>	2
1.1 Esquema da base de dados	2
2 Demonstração de operabilidade	3
2.1 Schmocial	3
2.2 Login e Registo	4
2.2.1 Registo	4
2.2.2 Login	5
2.3 Página inicial	6
2.4 Hashtags	7
2.5 Perfil de Utilizador	8
2.6 Amigos	9
2.7 Eventos	10
2.8 Chat	13
2.9 Exportação	14
3 Conclusão	15
4 Anexos	15
4.1 Autenticação	16
4.1.1 Estratégia Local	16
4.1.2 Estratégia facebook	17
4.2 Models	18
4.2.1 User.js	18
4.2.2 Post.js	19
4.2.3 Message.js	20
4.2.4 Events.js	21
4.3 <i>Controllers</i>	22
4.3.1 Users.js	22
4.3.2 Posts.js	24
4.3.3 Messages.js	27
4.3.4 Events.js	28
4.4 Routes	30
4.4.1 Api de Dados	30
4.4.2 Index Route	37
4.4.3 Users Route	45
4.5 Jquery	48
4.5.1 posts.js	48
4.5.2 events.js	52

4.5.3	connections.js	56
4.5.4	chat.js	62

Lista de Figuras

1	Esquema geral das coleções em <i>MongoDB</i> .	2
2	Homepage	3
3	Registo de um utilizador	4
4	Login de um utilizador	5
5	Página inicial	6
6	Navegação Por Hashtags	7
7	Página de Perfil	8
8	Envio Pedido de Amizade	9
9	Pedido de Amizade aceite	9
10	Visualização de todos os eventos	10
11	Adicionar Evento	11
12	Informação do Evento	12
13	Envio de Mensagem	13
14	Receção de Mensagem	13
15	Exportação do conteúdo	14

Introdução

Como parte da avaliação da cadeira de **Desenvolvimento de aplicações web** tínhamos como proposta desenvolver uma aplicação que tivesse como objetivo ser parecido a uma rede social.

A ideia era aplicar todo o conhecimento adquirido ao longo das aulas na realização deste trabalho. Para isso recorreremos a um interpretador de javascript chamado de *NodeJs* e a um sistemas de base de dados *MongoDB*.

Ao longo deste relatório iremos abordar as várias funcionalidades que foram implementadas, explicar como é que elas funcionam na prática e no final teremos uma secção de anexos com todo o código usado para a realização deste trabalho.

1 *MongoDB*

MongoDB é uma base de dados documental que foi desenvolvida pela *MongoDB, Inc*, esta usa *Binary JavaScript Object Notation* () para armazenar os seus documentos. Uma documental utiliza coleções, as quais podem ser comparadas a tabelas em bases de dados relacionais. Para armazenar documentos, equiparáveis a entradas numa tabela, os quais não necessitam de conter todos os mesmos atributos, podendo novos atributos ser adicionados a qualquer momento, além disso esta não contém o conceito de relações entre coleções.

1.1 Esquema da base de dados

Collection Name ^	Documents	Avg. Document Size	Total Document Size
events	2	235.0 B	470.0 B
message	2	528.0 B	1.0 KB
posts	4	222.3 B	889.0 B
users	3	587.7 B	1.7 KB

Figura 1: Esquema geral das coleções em *MongoDB*.

Para a realização do trabalho criamos uma base de dados com as 4 coleções que se encontram na figura 1. O objetivo destas coleções era permitirem-nos uma flexibilidade grande na construção da nossa aplicação.

Na nossa coleção *events* vamos ter toda a informação dos eventos que existem na aplicação, desde os utilizadores que vão participar, à data de realização, local, etc.

Na coleção de *message* irá guardar toda a troca de mensagens entre os utilizadores, esta coleção permite-nos realizar uma das *features* extras que queríamos que era um sistema de chat em tempo real.

Na coleção de *posts* iremos ter associada toda a informação dos posts de cada utilizador desde as hashtags, os *likes* de cada post, os comentários, etc.

E por fim, na coleção *users* é onde temos todos os utilizadores presentes no sistema.

2 Demonstração de operabilidade

Vamos agora proceder à demonstração de todas as *features* presentes na nossa aplicação. Iremos apresentar imagens sobre cada funcionalidade e explicar o que se pretende. No final iremos ter um anexo com todas as imagens do código da aplicação que servirão para mostrar como cada uma destas *features* foi realizada.

2.1 Schmocial

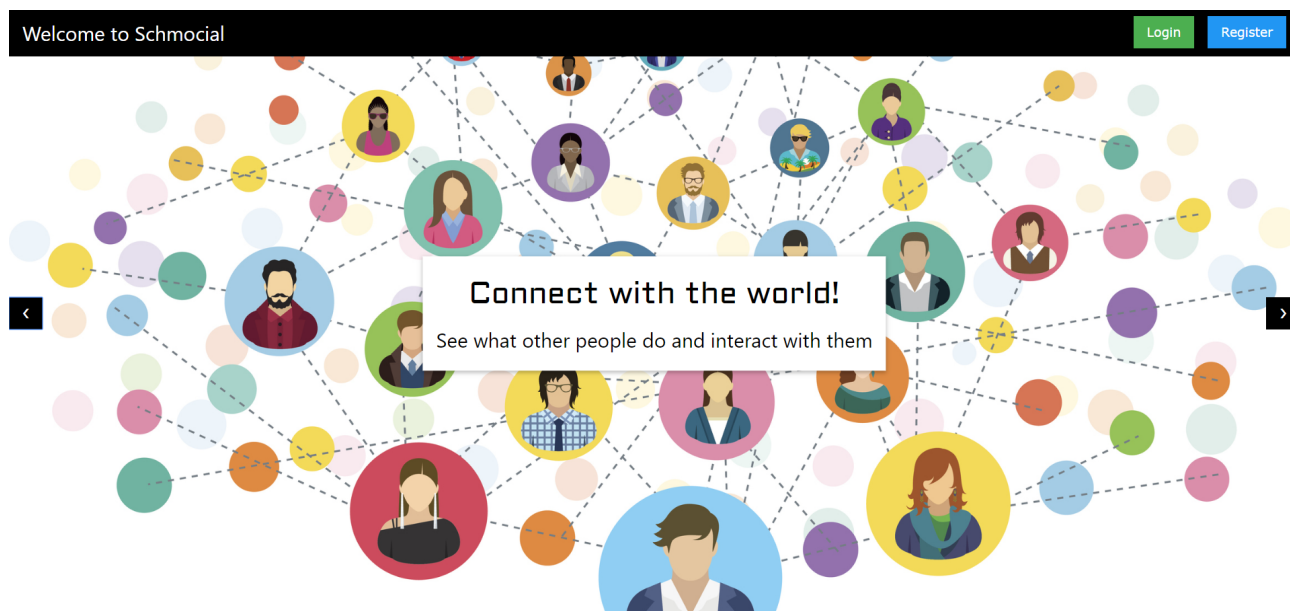


Figura 2: Homepage

O que está representado na figura 2 é a página de acesso à aplicação. É esta a página que o utilizador primeiro encontra quando quer aceder à sua página principal. Nesta página são então apresentadas duas opções. Se o utilizador já tiver conta pode aceder através do botão de login, ou caso seja a primeira vez em contacto com a nossa aplicação têm ainda a opção de registo. Caso o utilizador queira, pode também conectar-se através do facebook.

2.2 Login e Registo

2.2.1 Registo

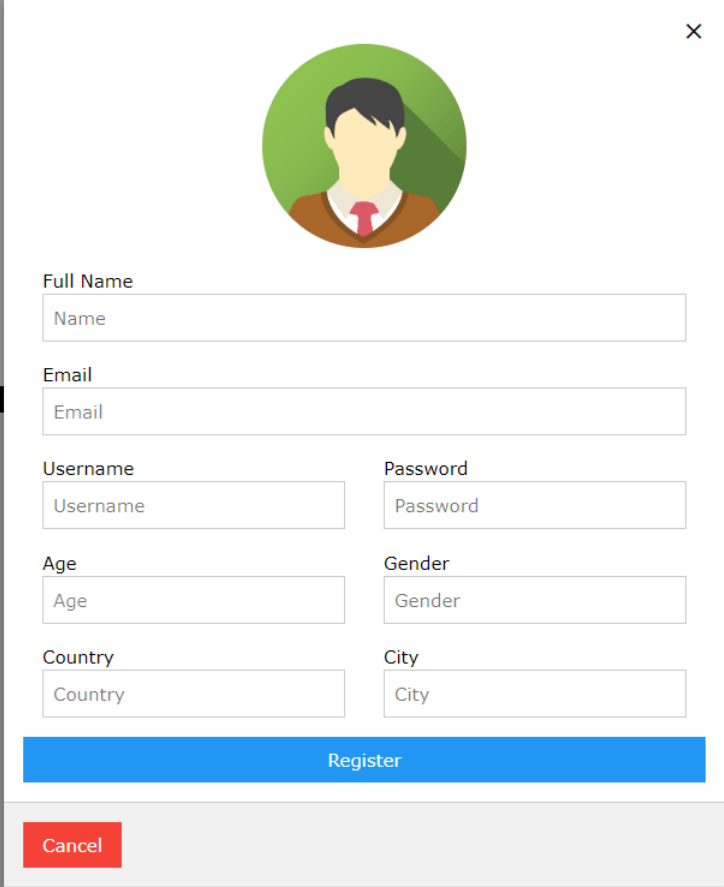
A user registration form displayed in a modal window. At the top center is a circular profile picture placeholder with a green background and a brown shirt. Below it are several input fields: 'Full Name' (placeholder 'Name'), 'Email' (placeholder 'Email'), 'Username' (placeholder 'Username'), 'Password' (placeholder 'Password'), 'Age' (placeholder 'Age'), 'Gender' (placeholder 'Gender'), 'Country' (placeholder 'Country'), and 'City' (placeholder 'City'). At the bottom are two buttons: a blue 'Register' button and a red 'Cancel' button. A close button (X) is in the top right corner of the modal.

Figura 3: Registo de um utilizador

Para realizar-mos a função de registo, sempre que o utilizador clica no botão *Register* é lhe apresentado um formulário como o que está na figura 3 para preencher os seus dados. Se quiser pode colocar uma foto de perfil mas não é obrigatório.

Para isto usamos o modulo *formidable* para obter as variáveis vindas do formulário. Devido ao facto de que podemos inserir imagens não podemos usar simplesmente o *req.body*.

2.2.2 Login

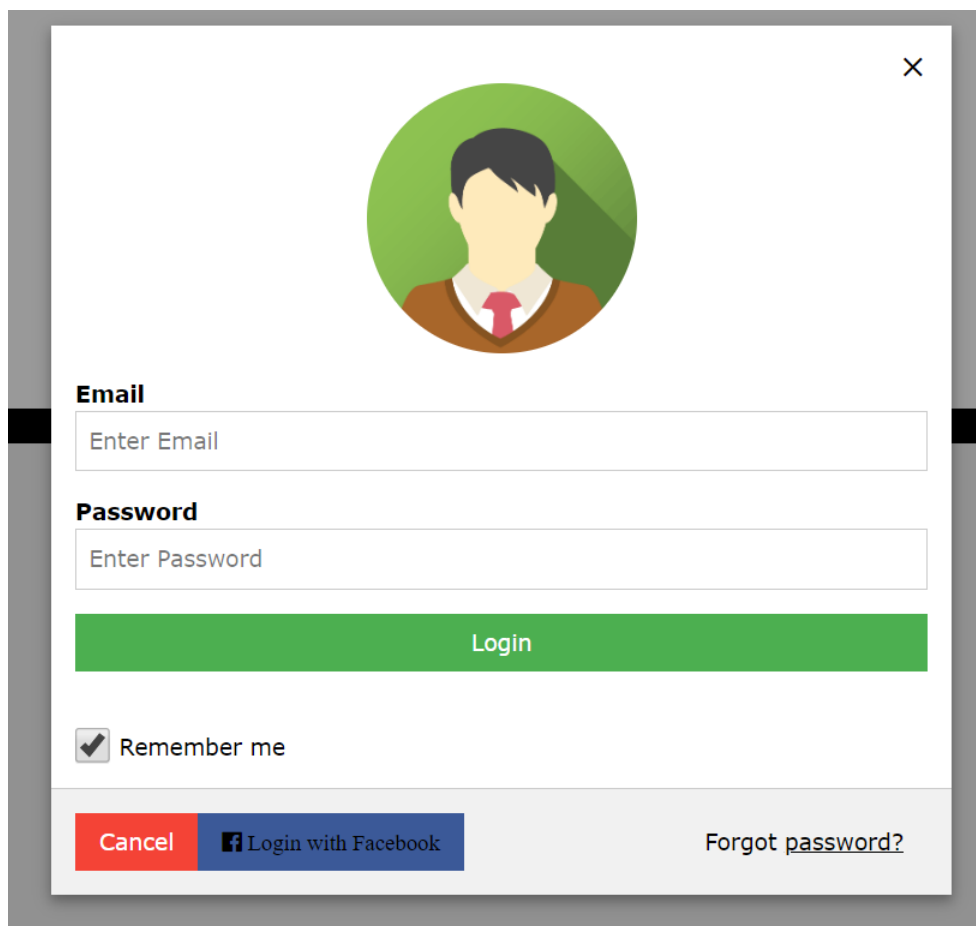
A login form with a white background and a gray border. At the top center is a circular profile picture placeholder with a green background and a brown shirt with a red tie. Below the profile picture are two input fields: 'Email' with the placeholder text 'Enter Email' and 'Password' with the placeholder text 'Enter Password'. Below the password field is a green 'Login' button. Underneath the button is a checkbox labeled 'Remember me' which is checked. At the bottom left are two buttons: a red 'Cancel' button and a blue 'Login with Facebook' button with the Facebook logo. At the bottom right is a link that says 'Forgot password?'. A close button (X) is in the top right corner.

Figura 4: Login de um utilizador

O login apresentado na figura 4 usa o mesmo tipo de layout que usamos no registo 3 porém aqui fazemos uso de vários *módulos* para gerir todo o processo de login de um utilizador com a nossa aplicação. Como podemos ver, temos dois tipos de *login*, o normal que requer o email e password e o login através do *facebook*. Para o *login* normal fizemos uso do modulo *passport*(Anexo 4.1.1) tal como foi dado nas aulas, do modulo *sessions* e *session-file-store* para guardar a sessão de cada utilizador.

Para o login através do *facebook*, primeiro tivemos que registar a nossa aplicação na consola do *Facebook Developer* para podermos ter acesso a usar a api deles. Para completar o login através do facebook também tivemos que usar o modulo *passport* mas com a estratégia do *facebook*(Anexo 4.1.2), ou seja usamos o modulo *passport-facebook*.

2.3 Página inicial

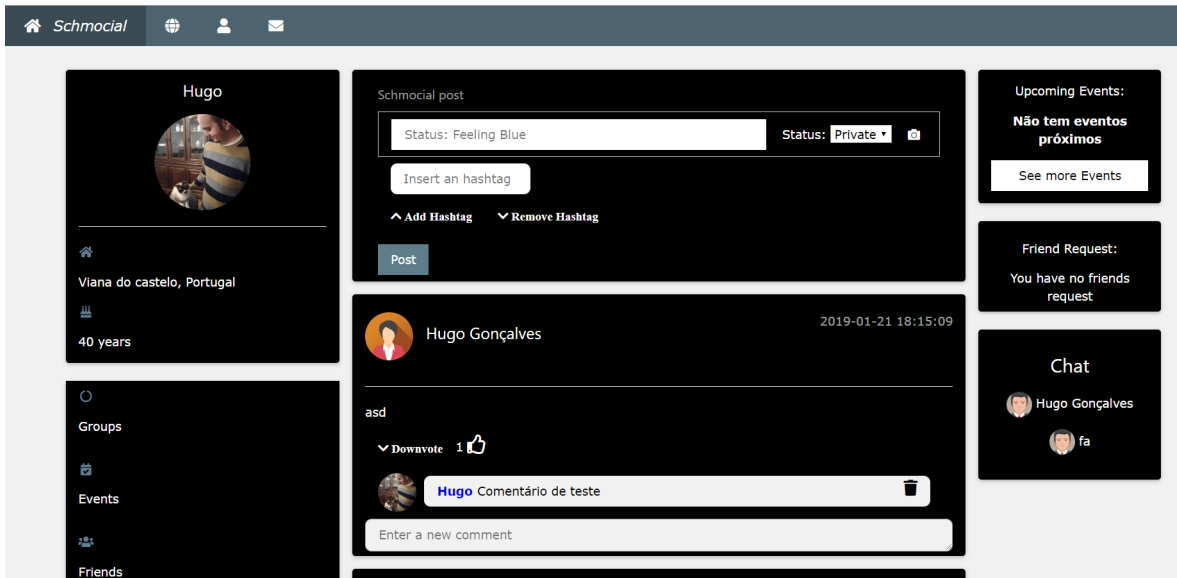


Figura 5: Página inicial

Na figura 5 temos representado o que cada utilizador, ao fazer *login* encontra. Para começar temos uma *navbar* que irá servir de suporte de navegação na aplicação. Com ela o utilizador pode ver o seu perfil, ver todas as mensagens ou simplesmente voltar para a página inicial.

Na coluna à esquerda, o utilizador consegue visualizar a sua informação básica tal como a sua localidade, idade e nome. Caso o utilizador tenha colocado foto de perfil a mesma irá aparecer na página tal como mostra a figura senão irá aparecer uma foto predefinida.

Continuando na mesma coluna, temos outro suporte à navegação do utilizador na sua página, ele pode se entender visualizar todos os seus amigos virtuais, ou se pretender pode também encontrar quais os eventos que estão a ocorrer. Passando agora para a coluna central e talvez a mais importante, começamos por ter uma secção de inserção de um *Post*. O utilizador aqui além do texto do *Post*, tem a opção de colocar uma foto que detalhe melhor o seu *Post*. Como opção pré definida, o *Post* irá ser privado porém o utilizador através da *drop-box* pode torna-lo publico. Por fim, como um dos requisitos pedidos era a atribuição de hashtags, o utilizador se quiser pode inserir nenhuma ou quantas ele quiser.

Imediatamente a seguir à secção de inserção temos a secção de todos os *Posts* existentes na aplicação. Cada *Post* é composto pelo nome de que o inseriu, a imagem caso tenha, o texto descritivo, os *likes* que teve e todos os comentários associados a esse *Post*. Isto é tudo dinâmico no sentido em que cada utilizador só pode dar *Like* apenas uma vez, pode remover os seus comentários se assim desejar como podemos ver na figura através do ícone do caixote do lixo.

Passando agora para a coluna à direita começamos pela secção dos eventos. Aqui irá aparecer o evento mais próximo(data) que o utilizador vai participar, Neste caso ele não participa em nenhum por isso não tem eventos próximos. Seguindo para a secção de baixo, temos os pedidos de amizade. Isto foi uma *feature* nova que implementamos para se complementar com o chat. Ou seja cada utilizador pode enviar um pedido de amizade a outro presente no sistema e caso o pedido seja aceite, a secção de chat irá ser atualizada com o novo utilizador.

2.4 Hashtags

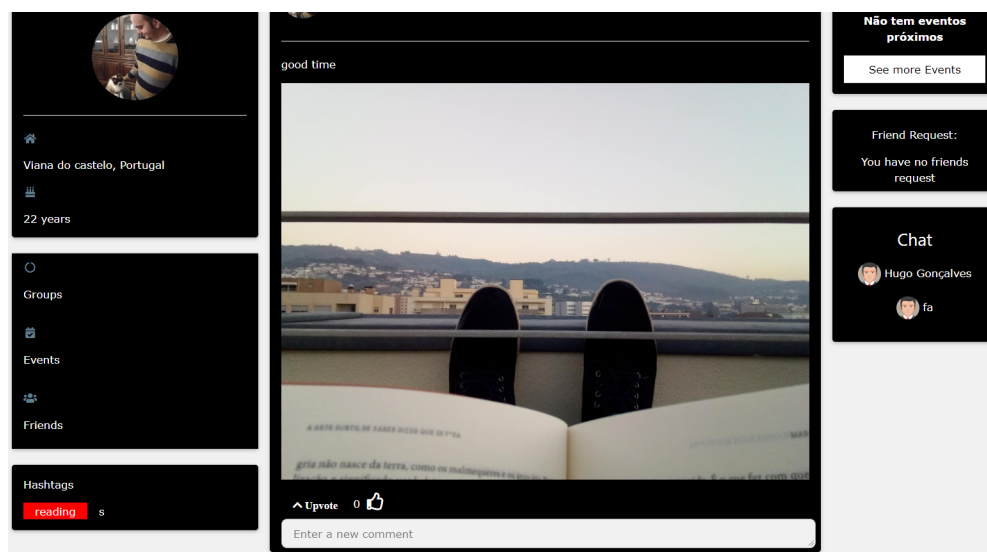


Figura 6: Navegação Por Hashtags

Uma das alternativas à forma predefinida como os utilizadores viam os *Posts* ou seja pela data, era a utilização de *Hashs* que permitissem o filtro dos posts. Para isso existe uma secção na coluna da esquerda onde irão ser apresentadas as 5 *Hashtags* com maior utilização. Na figura 6 temos o exemplo da demonstração do uso deste tipo de navegação em que o utilizador escolhe que tipo de *Posts* quer ver de acordo com a/as *hashs* escolhidas.

Esta seleção das hashtags despoleta uma função javascript que irá manipular os dados que estão presentes no ecrã para assim a página não ser obrigada a recarregar.

2.5 Perfil de Utilizador

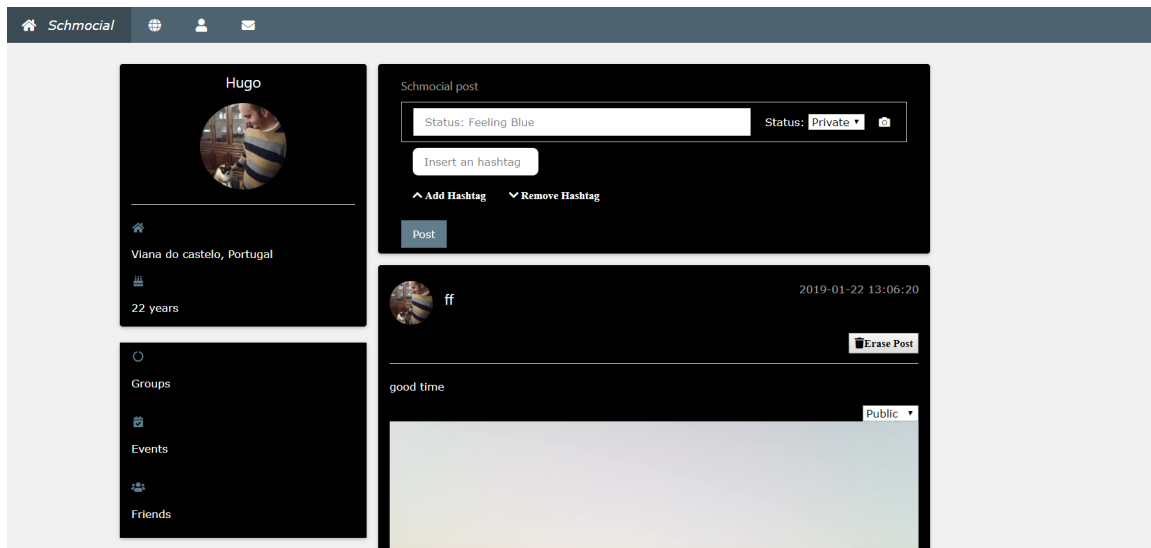


Figura 7: Página de Perfil

O utilizador quando faz *login* na aplicação é redirecionado para a página pública, ou seja, a página onde irão aparecer todos os *Posts* de todos os utilizadores, porém é preciso haver uma página pessoal onde o utilizador vai poder conforme o desejado, apagar algum *Post* que tenha criado ou simplesmente alterar o seu estado de publico para privado e vice-versa. Como podemos ver na figura 7 a página apresentada é relativamente igual à página inicial porém aqui o utilizador apenas visualiza os seus *Posts* e de acordo com as suas intenções se quiser apaga ou altera o estado como podemos verificar com a *drop-down* para alterar o estado e o botão para apagar. Caso também deseje o utilizador pode criar um novo post, inserir um comentário, colocar *like* tal como faria na página principal.

2.6 Amigos

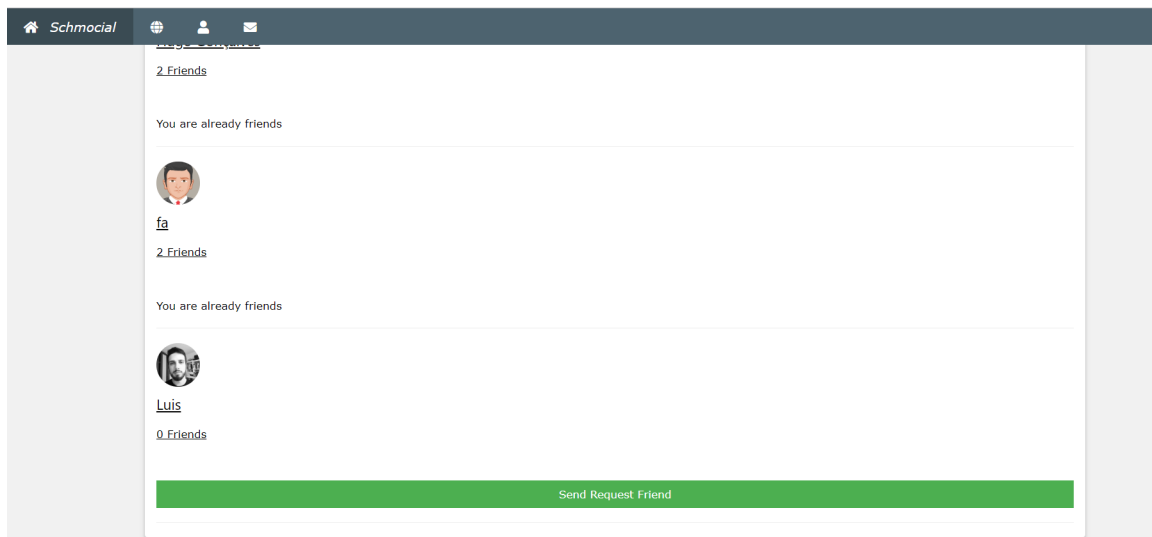


Figura 8: Envio Pedido de Amizade

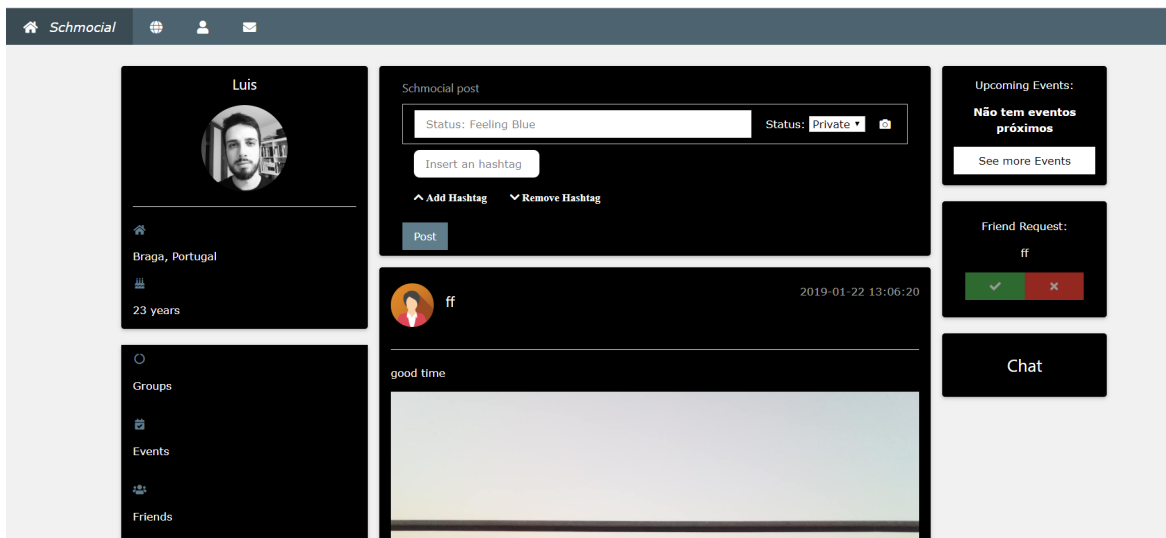


Figura 9: Pedido de Amizade aceite

Uma das funcionalidades extra que nós adicionamos como mencionado no início do relatório foi a introdução de um sistema de amigos virtuais. Como podemos ver na figura 8, nesta página nós temos acesso a todos os utilizadores que existem no nosso sistema, Para enviar um pedido de amizade basta encontrar o utilizador que nós queremos e clicar no botão que irá enviar um pedido. Neste caso ainda não somos amigos do utilizador Luís por isso vamos lhe enviar um pedido.

Quando o pedido é enviado irá aparecer no perfil do Luís na secção dos *Friends Request* o pedido de amizade que chegou como observamos na figura 9. Aqui o Luís tem duas opções, ou aceita o nosso pedido ou rejeita. Se aceitar a base de dados irá ser atualizada e a secção de chat irá ser atualizada senão fica tudo na mesma.

Esta funcionalidade serve também como um prototipo para uma versão melhorada da nossa aplicação. Nesta fase simplesmente serve para conseguirmos falar com o utilizador e ver o seu perfil, porém existem muitas mais funcionalidades que poderíamos implementar com este sistema tal como, apenas partilhar posts para os nossos amigos, apenas ver posts dos nossos amigos, criar conversas de grupo, etc.

2.7 Eventos

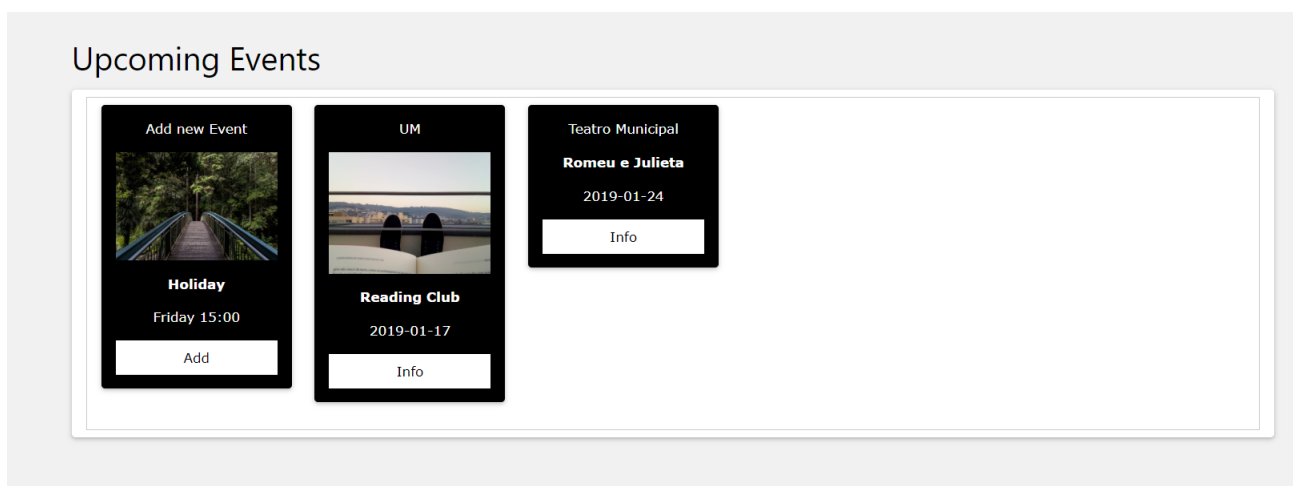







Figura 10: Visualização de todos os eventos

 Data

 Local

 Titulo

 Descrição

 Image

Escolher ficheiro

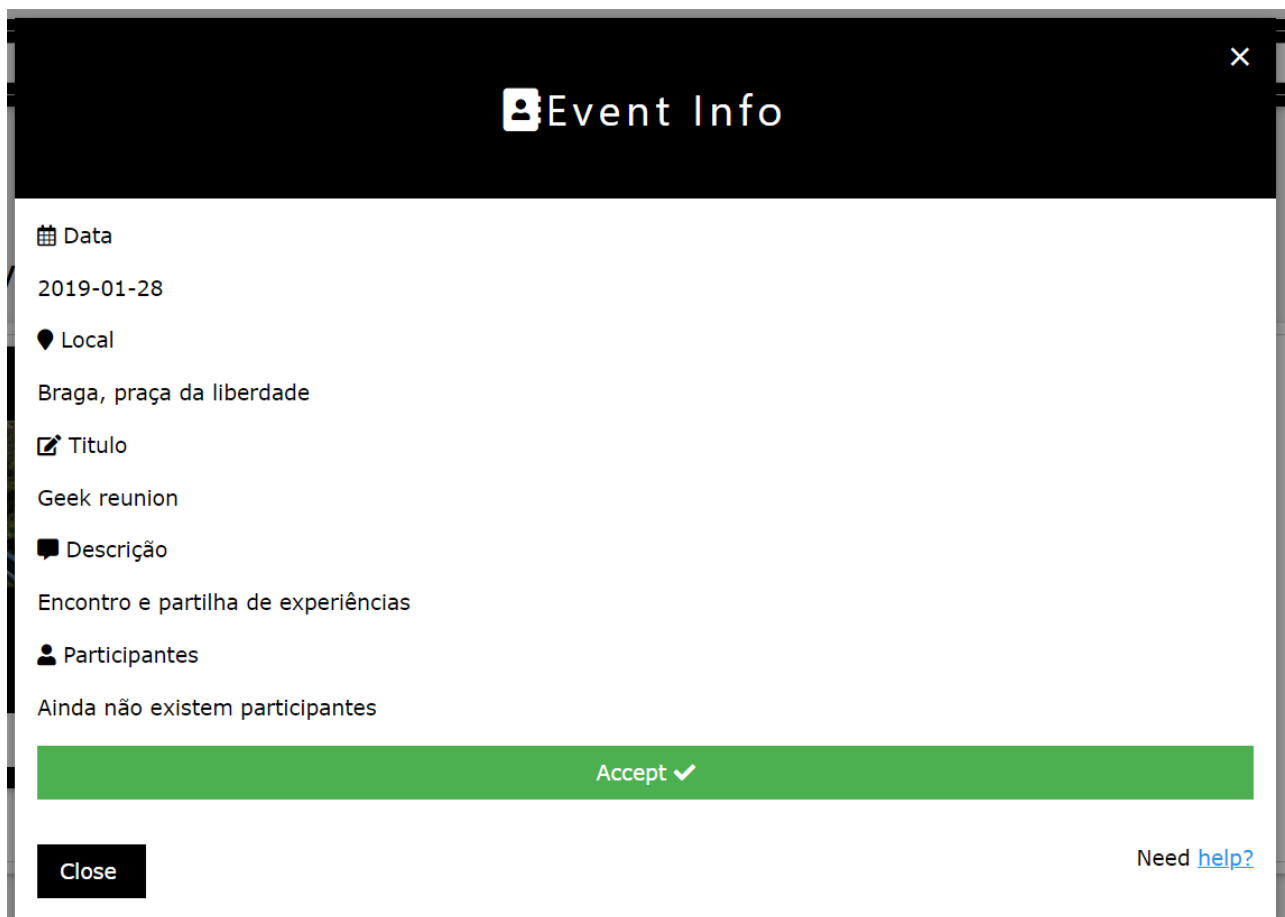
Nenhum ficheiro selecionado

Submeter

Close

Need [help?](#)

Figura 11: Adicionar Evento

A screenshot of a web application's 'Event Info' modal window. The modal has a dark header with a close button (X) in the top right corner. The main content area is white and contains several sections: 'Data' with a calendar icon and the date '2019-01-28'; 'Local' with a location pin icon and the text 'Braga, praça da liberdade'; 'Titulo' with a pencil icon and the text 'Geek reunion'; 'Descrição' with a speech bubble icon and the text 'Encontro e partilha de experiências'; and 'Participantes' with a person icon and the text 'Ainda não existem participantes'. At the bottom, there is a green bar with the text 'Accept ✓' and a black button labeled 'Close'. In the bottom right corner, there is a link that says 'Need [help?](#)'.

×

Event Info

📅 Data

2019-01-28

📍 Local

Braga, praça da liberdade

✎ Titulo

Geek reunion

💬 Descrição

Encontro e partilha de experiências

👤 Participantes

Ainda não existem participantes

Accept ✓

Close

Need [help?](#)

Figura 12: Informação do Evento

Para realizar a funcionalidade dos eventos pensamos que seria útil ter uma página própria com todos os eventos que vão decorrer. Quando o utilizador clica no botão de eventos que se encontra na página inicial ele é redirecionado para uma nova página em que lhe vai ser apresentado todos os eventos que existem no sistema como podemos ver na figura 10.

O utilizador tem depois duas opções, ou adiciona um evento, ou então indica que vai participar num evento(também pode cancelar a participação). Para adicionar 11 o evento fizemos uso do mesmo tipo de layout que usamos no login e registo, 3, 4. Aqui o utilizador preenche a informação básica sobre o evento que quer criar, como o local, a data, o assunto do evento, etc. Clicando em submeter o evento fica registado e disponível a todos os utilizadores para caso queiram aderir.

Na parte da informação do evento, é bastante similar ao adicionar ou seja, o utilizador pode clicar para ver mais informação sobre o evento que lhe interessa. Esta informação irá ser, tal como na inserção do evento, a data, o local, o assunto, os participantes, etc. O utilizador pode então aderir ao evento ou caso já tenha aderido e queira cancelar a participação também pode. Por fim na pagina inicial de cada

utilizador, na secção dos eventos irá aparecer o evento mais próximo, se o utilizador aderiu a 4 eventos, apenas irá aparecer o evento mais próximo.

2.8 Chat

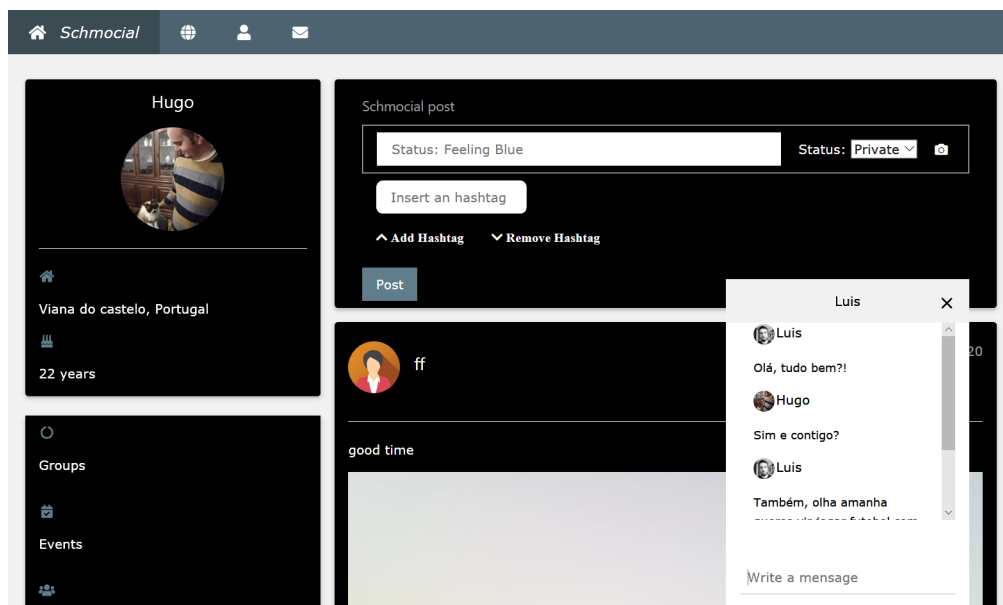


Figura 13: Envio de Mensagem

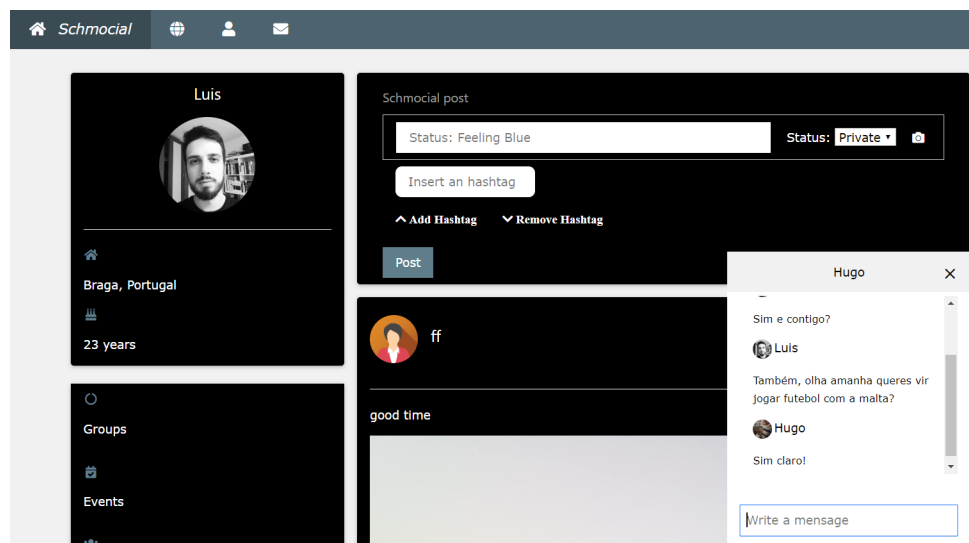


Figura 14: Receção de Mensagem

A ultima feature extra que nós adicionamos foi a troca de mensagens em tempo real. Nós queríamos que a nossa aplicação sentisse como uma verdadeira rede social e

para isso precisava de um chat. Para realizar esta funcionalidade tivemos que recorrer ao modulo *socket.io* que é o que nos vai permitir enviar e receber informação em tempo real. É um modulo relativamente simples de usar e que nos permite trocar mensagens com outros utilizadores presentes no sistema. Podemos observar um pouco mais do seu funcionamento no Anexo [4.5.4](#).

2.9 Exportação

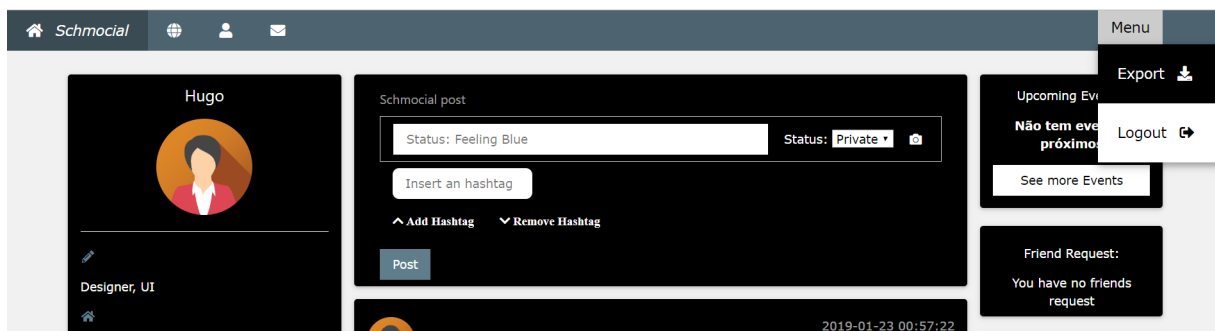


Figura 15: Exportação do conteúdo

A ultima funcionalidade presente na nossa aplicação é a exportação dos dados. Esta funcionalidade funciona de maneira bastante simples, se o utilizador que estiver com sessão iniciada no sistema tiver permissões de administrador então ele pode efetuar a exportação como podemos ver na figura [15](#), se não, então o utilizador não possui essa opção. A exportação de dados funciona de maneira bastante simples, sempre que o utilizador clica no botão de exportação a nossa aplicação vai através do uso de *Promises* recolher e guardar toda a informação das coleções em variáveis. Depois de termos toda a informação guardamos em ficheiros json de acordo com cada coleção para mais tarde quem quiser usar a nossa aplicação importar os dados.

3 Conclusão

Neste trabalho abordamos o desenvolvimento de aplicações web recorrendo a um interpretador de javascript, mais concretamente *NodeJs* em conjunto com um sistema de base de dados *MongoDB*. A maioria dos requisitos propostos foram atingidos assim como também foram implementadas funcionalidades extra que deram uma maior qualidade à aplicação.

Para uma melhor fluidez e utilização da aplicação recorreremos bastante ao uso de *javascripts* que implementassem técnicas de desenvolvimento web, neste caso *AJAX*. Isto permitiu-nos que fizéssemos uma aplicação bastante mais fluida e interativa porque evitamos que sempre que o utilizador realizasse uma ação a página inteira tivesse que ser recarregada. Esta técnica foi usada por exemplo na funcionalidade do chat, na inserção de comentários e *likes*, na filtragem de posts, etc. Os scripts podem ser consultados a partir do Anexo 4.5

Como melhoria futura, ficou por fazer a partilha de posts nas redes sociais, porém este requisito só não foi feito agora porque o *Facebook* possui regras bastante restritas para quem pretende utilizar certas funcionalidades da api pelo que foi-nos negado o acesso a essas funcionalidades. De resto achamos que todas as funcionalidades que nós temos podem ser melhoradas de modo a permitir outro tipo de experiência tais como, conversas de grupo, envio de ficheiros, imagens, vídeos através do chat, Diferentes tipos de posts, etc.

4 Anexos

Em anexo vamos deixar todos os códigos usados na realização do projeto, desde os modelos, controladores, javascripts. Devido ao nosso código ser bastante extenso achamos melhor explicar numa parte inicial o funcionamento da nossa aplicação, como já foi feito através de prints da aplicação em funcionamento e em anexo colocamos toda a parte de código que faz a aplicação funcionar. Tudo se encontra minimamente comentado pelo qual a compreensão de cada comando é relativamente simples.

4.1 Autenticação

4.1.1 Estratégia Local

Auth.js - Autenticação usando a estratégia local com JWT

```
var passport = require('passport')
var localStrategy = require('passport-local').Strategy
var UserModel = require('../models/user')
// Login de utilizadores
passport.use('login', new localStrategy({
  usernameField: 'email',
  passwordField: 'password'
}, async (email, password, done) => {
  try{var user = await UserModel.findOne({email:email})
    if(!user)
      return done(null, false, {message: 'Utilizador não existe!'})
    var valid = await user.isValidPassword(password)
    if(!valid)
      return done(null, false, {message: 'Password inválida!'})
    return done(null, user, {message: 'Login feito com sucesso.'})
  }
  catch(error){ return done(error)}
}))
// Autenticação com JWT
var JWTStrategy = require('passport-jwt').Strategy
var ExtractJWT = require('passport-jwt').ExtractJwt
var extractFromSession = function(req){
  var token = null
  if(req && req.session) token = req.session.token
  return token
}
passport.use(new JWTStrategy({
  secretOrKey: 'dweb2018',
  jwtFromRequest: ExtractJWT.fromExtractors([extractFromSession])
}, async (token, done) => {
  try{ return done(null, token.user) }
  catch(error){ return done(error) }
}))
```

4.1.2 Estratégia facebook

Facebook.js - Autenticação usando a estratégia do facebook

```
var passport = require('passport') , FacebookStrategy =
require('passport-facebook').Strategy;
var User = require('../models/user');
passport.use(new FacebookStrategy({
  clientID: "*****",
  clientSecret: "*****",
  callbackURL: "http://localhost:3000/facebook/callback",
  profileFields: ['id', 'displayName', 'emails', 'name'] },
function(accessToken, refreshToken, profile, done) {
  User.findOne({
    'facebook.id': profile.id
  }, function(err, user) {
    if (err) { return done(err); }
    if (!user) {
      var facebook = {
        id: profile.id,
        token: accessToken,
        email: profile.emails[0].value,
        name: profile._json.first_name
      }
      user = new User({
        name: profile.displayName,
        email: profile.emails[0].value,
        username: profile._json.first_name,
        gender: profile.gender,
        provider: 'facebook',
        password: "123",
        age: "23",
        address:{
          country:"England",
          city: "Porto" },
        facebook: facebook});
      user.save(function(err) {
        if (err) console.log(err);
        return done(err, user); });
    } else { return done(err, user); });
  });
module.exports = passport;
```

4.2 Models

4.2.1 User.js

Models-User.js - Criação do modelo User

```
var mongoose = require('mongoose')
var bcrypt = require('bcryptjs')
var md5 = require('md5')
var schema = mongoose.Schema
var addressSchema = new schema({
  street: {type: String},
  country: {type: String},
  city: {type: String}})
var friendSchema = new schema({
  id: {type: String},
  name:{type:String},
  state: {type: String}})
var UserSchema = new schema({
  name: {type: String, required:true},
  username: {type: String, required:true},
  password: {type: String, required:true},
  email: {type: String, required:true},
  age: {type: String},
  gender: {type: String},
  friends:[friendSchema],
  address: addressSchema,
  picture:{type:String},
  facebook: {
    id: {type: String},
    token: {type: String},
    email: {type: String},
    name: {type: String}
  },})
UserSchema.pre('save', async function(next){
  var hash = await bcrypt.hash(md5(this.password), 10)
  this.password = hash
  next()})
UserSchema.methods.isValidPassword = async function(password){
  var user = this
  var compare = await bcrypt.compare(md5(password), user.password)
  return compare}
module.exports = mongoose.model('Users', UserSchema, 'users')
```

4.2.2 Post.js

Models-Post.js - Criação do modelo Post

```
var mongoose = require('mongoose')

var schema = mongoose.Schema

var commentSchema = new schema({
  message: {type: String, required:true},
  user: {type: String, required:true},
  date: {type: String},
  picture: {type:String}
})

var voteSchema = new schema({
  type: {type: String, required:true},
  user: {type: String, required:true}
})

var postSchema = new schema({
  owner: {type: String, required:true},
  title: {type: String, required:true},
  date: {type: String, required:true},
  content: {type: String},
  picture:{type: String},
  file:{type:String},
  state:{type:String},
  hashtags:[{type:String}],
  comments: [commentSchema],
  votes: [{type:String}]
})

module.exports = mongoose.model('Post', postSchema, 'posts')
```


4.2.3 Message.js

Models-Messages.js - Criação do modelo Message

```
var mongoose = require('mongoose')

var schema = mongoose.Schema

var mensagem = new schema({
  nome:{type:String},
  foto:{type:String},
  texto : {type:String}
})

var MessageSchema = new schema({
  idUser1 : {type:String},
  idUser2 : {type:String},
  mensagens: [mensagem]
})

module.exports = mongoose.model('Message', MessageSchema, 'message')
```

4.2.4 Events.js

Models-Event.js - Criação do modelo Events

```
var mongoose = require('mongoose')

var schema = mongoose.Schema

var participantSchema = new schema({
  id : {type:String},
  name : {type: String}
})

var EventSchema = new schema({
  date : {type: String,required:true},
  local : {type: String,required:true},
  title : {type: String,required:true},
  description : {type: String},
  picture : {type:String},
  participants : [participantSchema]
})

module.exports = mongoose.model('Events', EventSchema,'events')
```

4.3 *Controllers*

4.3.1 Users.js

Controllers-users.js - Criação do Controller Users

```
var User = require('../models/user')
const Users = module.exports

//Returns Users list
Users.list = ()=>{
  return User
    .find({})
    .exec()
}

//Returns User by name
Users.getByName = name=>{
  return User
    .find({name: name})
    .exec()
}

//Returns User by id
Users.getById = id=>{
  return User
    .findOne({_id: id})
    .exec()
}

//Insert New Friend
Users.insertNewFriend = newFriend=>{
  return User
    .updateOne({_id:newFriend.userDest}, {
      $push:{friends:newFriend.userOrigin}})
    .exec()
}

//Export Users list
Users.export = ()=>{
  return User
    .find({})
    .exec()
}
```

Controllers-users.js - Continuação do Controller Users

```
//Get Friend Request
Users.getFriends = id=>{
  return User
    .find({_id: id, "friends.state":"request"},
    {_id:0,"friends.$":1})
    .exec()
}

//Get User Friends
Users.getUserFriends = id=>{
  return User
    .find({_id: id, "friends.state":"accepted"},
    {_id:0,friends:1})
    .exec()
}

//Accept Friend
Users.acceptFriend = info=>{
  User
    .updateOne({_id:info.user}, {$push:{friends:info.friend}})
    .exec()

  return User
    .updateOne({_id:info.friend.id, "friends.id":info.user}, {
    $set:{"friends.$.state":"accepted"}})
    .exec()
}

//Decline Friend
Users.deleteFriend = info=>{
  console.log("OLAAAAAAAA " +info.user)
  return User
    .update({_id: info.user},
    { $pull: { "friends" : { id: info.friend } } })
    .exec()
}

//Create new User
Users.insertNew = newDoc=>{
  return User
    .create(newDoc)
}
```

4.3.2 Posts.js

Controllers-Posts.js - Criação do Controller Posts

```
var Post = require('../models/post')

const Posts = module.exports

//Returns Posts list
Posts.list = ()=>{
  return Post
    .find({})
    .sort({_id:-1})
    .exec()}

//Returns Posts by hashtags
Posts.listByHashtags = hashtags =>{
  console.log("CHEGUEI" + hashtags.length)

  return Post
    .find({hashtags:{$in:hashtags}})
    .sort({_id:-1})
    .exec()

  return Post
    .find({})
    .sort({_id:-1})
    .exec()}

//Returns Comment list for post by id
Posts.listComments = id=>{
  return Post
    .findOne({_id:id},{comments:1, _id:0})
    .exec()}

//Returns Posts by title
Posts.getByTitle = title=>{
  return Post
    .findOne({title: title})
    .exec()}
```

Controllers-Posts.js - Continuação do Controller Posts

```
//Returns Posts by user
Posts.getByUser = user=>{
  return Post
    .find({owner: user})
    .sort({_id:-1})
    .exec()
}

//Insert comment in Post
Posts.insertNewComent = (id,newDoc)=>{
  console.log("\n"+newDoc.user+"\n")
  return Post
    .update({_id:id}, {$push:{comments:newDoc}})
    .exec()
}

//Remove comment in Post
Posts.removeComment = (info)=>{
  console.log("Nao aqui")
  return Post
    .update({_id: info.post},
      {$pull:{ comments: {_id:info.comment} }})
    .exec()
}

//Insert vote in Post
Posts.insertNewVote = (user,post)=>{
  return Post
    .update({_id:post}, {$push:{votes:user}})
    .exec()
}

//Remove vote in Post
Posts.removeVote = (user,post)=>{
  return Post
    .update({_id: post},
      { $pull: { "votes" : user } })
    .exec()
}
```

Controllers-Posts.js - Continuação do Controller Posts

```
//List all Post Hashtags
Posts.listHashtags = ()=>{
  return Post
    .aggregate([{$unwind:"$hashtags"},
      {$group: {_id:"$hashtags", count:{$sum:1}}},
      {$sort:{count:-1}},
      {$limit:5},
      {$project: {_id:0, hashtags :"$_id", count:"$count"}}
    ])
    .exec()
}

//Update status Post
Posts.updatePost = info=>{
  return Post
    .update({_id:info.id},
      {$set:{state:info.estado}})
    .exec()
}

//Remove Post
Posts.removePost = id=>{
  return Post
    .remove({_id:id})
    .exec()
}

//Create new Posts
Posts.insertNew = newDoc=>{
  return Post
    .create(newDoc)
}

//Exports Posts list
Posts.export = ()=>{
  return Post
    .find({})
    .exec()
}
```

4.3.3 Messages.js

Controllers-Messages.js - Criação do Controller Messages

```
var Message = require('../models/messages')
var moment = require('moment');
const Messages = module.exports

//Get Messages between users
Messages.listBetweenUsers = (user1,user2)=>{
  return Message
    .find(
      { $or : [ { idUser1 : user1, idUser2:user2 },
        {idUser1 : user2, idUser2 :user1 } ] }
      ,{messages:0})
    .exec()
}

//Add Messages between users
Messages.addMessages = info=>{
  console.log("Chego ao controller")
  for(var key in info.mensagens){
    console.log("Mensagem "+key)
  }
  return Message
    .update({ $or : [ { idUser1 : info.idUser1,
      idUser2:info.idUser2 },
      {idUser1 : info.idUser2, idUser2 :info.idUser1 } ] },
      {$push:{mensagens:info.mensagens}})
    .exec()
}

//Create new Message
Messages.insertNew = newDoc=>{
  return Message
    .create(newDoc)
}

//Export all Messages
Messages.export = ()=>{
  return Message
    .find({})
    .exec()
}
```


4.3.4 Events.js

Controllers-Events.js - Criação do Controller Events

```
var Event = require('../models/event')
var moment = require('moment');
const Events = module.exports

//Returns Events list
Events.list = ()=>{
  return Event
    .find({})
    .sort({date:1})
    .exec()
}

//Returns Events after today date
Events.getByDate = ()=>{
  return Event
    .find({date: {$gte: moment().format("YYYY-MM-DD")}})
    .sort({date:1})
    .exec()
}

//Returns Events by title
Events.getByTitle = title=>{
  return Event
    .findOne({title: title})
    .exec()
}
```

Controllers-Events.js - Continuação do Controller Events

```
//Insert Participants
Events.insertNewParticipant = newParticipant=>{
  return Event
    .update({_id:newParticipant.idPost}, {
      $push:{participants:newParticipant.participant}})
    .exec()
}

//Remove Participants
Events.removeParticipant = id=>{
  console.log("Participante: "+id.participant +
    " Evento: "+id.idPost)
  return Event
    .updateOne({_id:id.idPost},
      {$pull:{ participants: {id:id.participant} }})
    .exec()
}

//Get Participants
Events.getParticipants = id=>{
  return Event
    .find({_id:id},{_id:0,participants:1})
    .exec()
}

//Create new Events
Events.insertNew = newDoc=>{
  return Event
    .create(newDoc)
}

//Export Events list
Events.export = ()=>{
  return Event
    .find({})
    .exec()
}
```

4.4 Routes

4.4.1 Api de Dados

Routes-api.js - Criação da Api de dados

```
var express = require('express');
var router = express.Router();
var Users = require('../controllers/users')
var Events = require('../controllers/events')
var Posts = require('../controllers/posts')
var Messages = require('../controllers/messages')
var passport = require('passport')
var jwt = require('jsonwebtoken')

/* GET home page. */
router.get('/', function(req, res, next) {
  res.render('index', { title: 'Express' });
});

//USER API ROUTES

//Get all users
router.get('/users', (req, res)=>{
  Users.list()
  .then(dados => res.jsonp(dados))
  .catch(erro => res.status(500).send('DEU ERRO NA LISTAGEM'))
})
//Get user by name
router.get('/users/:id', (req, res)=>{
  Users.getById(req.params.id)
  .then(dados => res.jsonp(dados))
  .catch(erro => res.status(500).send('DEU ERRO NA LISTAGEM'))
})
//Get user by name
router.get('/users/name/:name', (req, res)=>{
  Users.getByName(req.params.name)
  .then(dados => res.jsonp(dados))
  .catch(erro => res.status(500).send('DEU ERRO NA LISTAGEM'))
})
```

Routes-api.js - Continuação Api de dados

```
//Insert User
router.post('/users/signup', (req, res)=>{
  Users.insertNew(req.body)
  .then(dados => res.jsonp(dados))
  .catch(erro => res.status(500).send('DEU ERRO NA INSERÇÃO'))
})

//EVENT API ROUTES

//get all events
router.get('/events', (req, res)=>{
  Events.list()
  .then(dados => res.jsonp(dados))
  .catch(erro => res.status(500).send('DEU ERRO NA LISTAGEM'))
})

//get all events after today date
router.get('/events/byDate', (req, res)=>{
  Events.getByDate()
  .then(dados => res.jsonp(dados))
  .catch(erro => res.status(500).send('DEU ERRO NA LISTAGEM'))
})

//get events by name
router.get('/events/:name', (req, res)=>{
  Events.getByTitle(req.params.name)
  .then(dados => res.jsonp(dados))
  .catch(erro => res.status(500).send('DEU ERRO NA LISTAGEM'))
})

//insert new Event
router.post('/events', (req, res)=>{
  Events.insertNew(req.body)
  .then(dados => res.jsonp(dados))
  .catch(erro => res.status(500).send('DEU ERRO NA LISTAGEM'))
})

//insert participant in Event
router.post('/events/participant', (req, res)=>{
  Events.insertNewParticipant(req.body)
  .then(dados => res.jsonp(dados))
  .catch(erro => res.status(500).send('DEU ERRO NA LISTAGEM'))
})
```

Routes-api.js - Continuação Api de dados

```
//remove participant in Event
router.post('/events/participant/remove', (req, res)=>{
  Events.removeParticipant(req.body)
  .then(dados => res.jsonp(dados))
  .catch(erro => res.status(500).send('DEU ERRO NA LISTAGEM'))
})

//gets participants from Event
router.get('/events/participants/:id', (req, res)=>{
  Events.getParticipants(req.params.id)
  .then(dados => res.jsonp(dados))
  .catch(erro => res.status(500).send('DEU ERRO NA LISTAGEM'))
})

//POSTS API ROUTES

//get all posts
router.get('/posts', (req, res)=>{
  console.log("Chegueii POST")
  Posts.list()
  .then(dados => res.jsonp(dados))
  .catch(erro => res.status(500).send('DEU ERRO NA LISTAGEM'))
})

//get all hashtags
router.get('/posts/hashtags', (req, res)=>{
  Posts.listHashtags()
  .then(dados => res.jsonp(dados))
  .catch(erro => res.status(500).send('DEU ERRO NA LISTAGEM'))
})

//get posts by tittle
router.get('/posts/:tittle', (req, res)=>{
  Posts.getByTitle(req.params.tittle)
  .then(dados => res.jsonp(dados))
  .catch(erro => res.status(500).send('DEU ERRO NA LISTAGEM'))
})
```

Routes-api.js - Continuação Api de dados

```
//get user posts
router.get('/posts/user/:id', (req, res)=>{
  Posts.getByUser(req.params.id)
  .then(dados => res.jsonp(dados))
  .catch(err => res.status(500).send('DEU ERRO NA LISTAGEM'))
})

//get post comment
router.get('/posts/comments/:id', (req, res)=>{
  console.log("COMENTARIO")
  Posts.listComments(req.params.id)
  .then(dados => res.jsonp(dados))
  .catch(err => res.status(500).send('DEU ERRO NA LISTAGEM'))
})

//update status post
router.post('/posts/update/', (req, res)=>{
  Posts.updatePost(req.body)
  .then(dados => res.jsonp(dados))
  .catch(err => res.status(500).send('DEU ERRO NA LISTAGEM'))
})

//remove post by id
router.post('/posts/remove/', (req, res)=>{
  Posts.removePost(req.body.id)
  .then(dados => res.jsonp(dados))
  .catch(err => res.status(500).send('DEU ERRO NA LISTAGEM'))
})

//insert new Post
router.post('/posts', (req, res)=>{
  console.log("API "+req.body.hashtags)
  Posts.insertNew(req.body)
  .then(dados => res.jsonp(dados))
  .catch(err => res.status(500).send('DEU ERRO NA LISTAGEM'))
})

//Remove comment on Post
router.post('/posts/comment/remove', (req, res)=>{
  Posts.removeComment(req.body)
  .then(dados => res.jsonp(dados))
  .catch(err => res.status(500).send('DEU ERRO NA LISTAGEM'))
})
```

Routes-api.js - Continuação Api de dados

```
//insert new comment on Post
router.post('/posts/comment/:id', (req, res)=>{
  Posts.insertNewComent(req.params.id, req.body)
  .then(dados => res.jsonp(dados))
  .catch(erro => res.status(500).send('DEU ERRO NA LISTAGEM'))
})

//Filter by hashtags
router.post('/posts/filter/', (req, res)=>{
  Posts.listByHashtags(req.body.hashtags)
  .then(dados => res.jsonp(dados))
  .catch(erro => res.status(500).send('DEU ERRO NA LISTAGEM'))
})

//Add new vote
router.post('/posts/vote/', (req, res)=>{
  Posts.insertNewVote(req.body.user, req.body.post)
  .then(dados => res.jsonp(dados))
  .catch(erro => res.status(500).send('DEU ERRO NA LISTAGEM'))
})

//Remove vote
router.post('/posts/vote/remove', (req, res)=>{
  Posts.removeVote(req.body.user, req.body.post)
  .then(dados => res.jsonp(dados))
  .catch(erro => res.status(500).send('DEU ERRO NA LISTAGEM'))
})

//New Friend Request
router.post('/friends/new/', (req, res)=>{
  Users.insertNewFriend(req.body)
  .then(dados => res.jsonp(dados))
})

//Accept friend
router.post('/friends/accept/', (req, res)=>{
  Users.acceptFriend(req.body)
  .then(dados => res.jsonp(dados))
})
```

Routes-api.js - Continuação Api de dados

```
//Delete friend
router.post('/friends/delete/', (req, res)=>{
  Users.deleteFriend(req.body)
  .then(dados => res.jsonp(dados))
})

//Get Friends
router.get('/friends/get/:id', (req, res)=>{
  Users.getFriends(req.params.id)
  .then(dados => res.jsonp(dados))
})

//Get User Friends
router.get('/friends/get/user/:id', (req, res)=>{
  Users.getUserFriends(req.params.id)
  .then(dados => res.jsonp(dados))
})

//Create new messages beetween users
router.post('/messages/', (req, res)=>{
  var mensagens = {
    nome: req.body.nome,
    foto: req.body.foto,
    texto:req.body.mensagem
  }
  var mensagem = {
    idUser1:req.body.idUser1,
    idUser2:req.body.idUser2,
    mensagens:mensagens
  }
  Messages.insertNew(mensagem)
  .then(dados => res.jsonp(dados))
})
```


Routes-api.js - Continuação Api de dados

```
//Update messages beetween users
router.post('/messages/update', (req, res)=>{
  var mensagens = {
    nome: req.body.nome,
    foto: req.body.foto,
    texto:req.body.mensagem
  }
  var mensagem = {
    idUser1:req.body.idUser1,
    idUser2:req.body.idUser2,
    mensagens:mensagens
  }
  Messages.addMessages(mensagem)
    .then(dados => res.jsonp(dados))
})

//Get messages beetween users
router.get('/messages/:id/:id2', (req, res)=>{
  Messages.listBetweenUsers(req.params.id,req.params.id2)
    .then(dados => res.jsonp(dados))
})

/* Export Data. */
router.get('/export', async function(req, res, next) {
  let data
  //guarda toda a informação das coleções na variavel data
  try {
    data = await Promise.all([
      Posts.export(),
      Events.export(),
      Messages.export(),
      Users.export()
    ]);
    //data[0] = Posts.list result, data[1] = Events.list result etc..
    res.status(200).json(data)
  } catch (e) {
    res.status(500).send('error');
  }
});

module.exports = router;
```

4.4.2 Index Route

Routes-index.js - Criação da Route Index

```
var express = require('express');
var router = express.Router();
var axios = require('axios')
var fs = require('fs')
var formidable = require('formidable')
var moment = require('moment');
var FB = require('fb');
var passportFacebook = require('../auth/facebook');

/* FACEBOOK ROUTER */
router.get('/facebook',
  passportFacebook.authenticate('facebook', {scope:['email'] }));

router.get('/facebook/callback',
  passportFacebook.authenticate('facebook',
    {failureRedirect:'/login'}),
  function(req, res) {
    // Successful authentication, redirect home.
    res.redirect('/');
  });

/* Home page. */
router.get('/', function (req, res) {

  if (req.isAuthenticated()) {
    res.redirect('/home')
  } else {
    res.render('index', { title: 'Welcome to Schmocial' });
  }
});

/* Login page.*/
router.get('/signin', function (req, res) {
  res.render('login', { title: 'Welcome to Schmocial' });
});
```

Routes-index.js - Continuação da Route Index

```
/* Home page */
router.get('/home', verifyAuth, function (req, res) {
  axios.get('http://localhost:3000/api/posts')
    .then(resposta => {
      res.render('home',
        { posts: resposta.data, username:req.user._id,
          utilizador:req.user })
    })
    .catch(erro => {
      console.log('Erro ao carregar da bd')
      res.render('error',
        { error: erro, message: 'Erro ao carregar da bd' })
    })
});

/* Profile page */
router.get('/profile/:id', verifyAuth, function (req, res) {
  axios.get('http://localhost:3000/api/posts/user/'+req.params.id)
    .then(resposta => {
      res.render('profile',
        { posts: resposta.data, username:req.user._id,
          utilizador:req.user })
    })
    .catch(erro => {
      console.log('Erro ao carregar da bd')
      res.render('error',
        { error: erro, message: 'Erro ao carregar da bd' })
    })
});

/* Event page */
router.get('/events', verifyAuth, function (req, res) {
  axios.get('http://localhost:3000/api/events/')
    .then(resposta => {
      res.render('events',
        { events: resposta.data, utilizador:req.user })
    })
    .catch(erro => {
      console.log('Erro ao carregar da bd')
      res.render('error',
        { error: erro, message: 'Erro ao carregar da bd' })
    })
});
```

```
//Insert new Event

router.post('/events', verifyAuth, function (req, res) {
  console.log("Chego")
  var event;
  var form = new formidable.IncomingForm()
  form.parse(req, (erro, fields, files) => {
    if (!files.inputImage.name) {
      event = {
        date: fields.inputDate,
        local: fields.inputLocation,
        title : fields.inputTitle,
        description : fields.inputDescription,
      }
      axios.post('http://localhost:3000/api/events/', event)
        .then(() => res.redirect('http://localhost:3000/'))
        .catch(erro => {
          console.log('Erro na inserção da bd')
          //res.redirect('http://localhost:3000/')
        })
    } else {
      var fenviado = files.inputImage.path
      var fnovo = '../Schmocial/public/images/eventos/' +
        files.inputImage.name
      fs.rename(fenviado, fnovo, erro => {
        if (!erro) {
          event = {
            date: fields.inputDate,
            local: fields.inputLocation,
            title : fields.inputTitle,
            description : fields.inputDescription,
            picture : files.inputImage.name,
          }
          axios.post('http://localhost:3000/api/events/', event)
            .then(() => res.redirect('http://localhost:3000/'))
            .catch(erro => {
              console.log('Erro na inserção da bd')
              //res.redirect('http://localhost:3000/')
            })
        } else {
          //res.render(error)
          console.log("ERRO +" + erro)
        }
      })
    }
  })
})
});
```

Routes-index.js - Continuação da Route Index

```
//Connects Routes-Show all users in the system
router.get('/friends', verifyAuth, function (req, res) {
  res.render('connections', { username_id:req.user._id,
    username_name:req.user.name })
});

//Friends Profile
router.get('/friends/:id', verifyAuth, function (req, res) {
  axios.get('http://localhost:3000/api/posts/user/'+req.params.id)
    .then(resposta => {
      res.render('friend', { posts: resposta.data,
        username:req.params.id.split(":")[1], utilizador:req.user})
    })
    .catch(erro => {
      console.log('Erro ao carregar da bd')
      res.render('error',
        { error: erro, message: 'Erro ao carregar da bd' })
    })
});

//Request friendship
router.post('/friends/new', verifyAuth, function (req, res) {
  axios.get('http://localhost:3000/api/friends/new/')
    .then(resposta => {
      res.render('home', { posts: resposta.data })
    })
    .catch(erro => {
      console.log('Erro ao carregar da bd asdasd'+erro)
    })
});

//Post routs

//Get hashtags
router.get('/posts/hashtags', verifyAuth, function (req, res) {
  axios.get('http://localhost:3000/api/posts/hashtags')
    .then(resposta => {
      res.render('hashtags', {lista:resposta.data})
    })
    .catch(erro => {
      console.log('Erro ao carregar da bd asdasd'+erro)
    })
});
```

Routes-index.js - Continuação da Route Index

```
//Insert a new Post
router.post('/insertPost', verifyAuth, function (req, res) {
  var poste;
  var hashtags = new Array()
  var form = new formidable.IncomingForm()
  form.on('field', function(field, value) {
    if(field=='hashtags[]' && value!=""){
      hashtags.push(value)
    }
  }).on('end', function() {});
  form.parse(req, (erro, fields, files) => {
    if (!files.ficheiro.name) {
      poste = {
        owner: req.user._id+": "+req.user.name,
        title: "teste",
        date: moment().format("YYYY-MM-DD HH:mm:ss"),
        content: fields.descricao,
        state: fields.state,
        hashtags:hashtags
      }
      console.log(poste)
      axios.post('http://localhost:3000/api/posts', poste)
        .then(() => res.redirect('http://localhost:3000/'))
        .catch(erro => {
          //res.redirect('http://localhost:3000/')
        })
    } else {
      var fenviado = files.ficheiro.path
      var fnovo = '../Schmocial/public/images/'
        + files.ficheiro.name
      fs.rename(fenviado, fnovo, erro => {
        if (!erro) {
          poste = {
            owner: req.user._id+": "+req.user.name,
            title: "teste",
            date: moment().format("YYYY-MM-DD HH:mm:ss"),
            content: fields.descricao,
            picture: files.ficheiro.name,
            file: fnovo,
            state: fields.state,
            hashtags:hashtags
          }
          axios.post('http://localhost:3000/api/posts', poste)
            .then(() => res.redirect('http://localhost:3000/'))
            .catch(erro => {
              console.log('Erro na inserção da bd')
            })
        }
      })
    }
  })
})
```

Routes-index.js - Continuação da Route Index

```
//Post filter
router.post('/posts/filter', verifyAuth, function (req, res) {
  axios.get('http://localhost:3000/api/posts/filter/'
    +req.body.hashtags)
    .then(resposta => {
      res.render('home', { posts: resposta.data })
    })
    .catch(erro => {
      console.log('Erro ao carregar da bd asdasd'+erro)
    })
});

//Insere comentario
router.post('/posts/comment/', verifyAuth, function (req, res) {
  var comment
  if(req.user.picture){
    comment = {
      message:req.body.comment,
      user:req.user._id+": "+req.user.name,
      date:moment().format("YYYY-MM-DD HH:mm:ss"),
      picture:req.user.picture
    }
  }else{
    comment = {
      message:req.body.comment,
      user:req.user._id+": "+req.user.name,
      date:moment().format("YYYY-MM-DD HH:mm:ss")
    }
  }
  axios.post('http://localhost:3000/api/posts/comment/'
    +req.body.id, comment)
    .then(resposta => {
      console.log(resposta.data)
    })
    .catch(erro => {
      console.log('Erro ao carregar da bd asdasd'+erro)
    })
});
```

Routes-index.js - Continuação da Route Index

```
//Exportação da base de dados
router.get('/export', function (req, res) {
  axios.get('http://localhost:3000/api/export')
    .then(resposta => {
//tratamos o campo _id para no mongo aparecer ObjectId
      for(i=0;i<resposta.data[0].length;i++){
        resposta.data[0][i]['_id'] = {'$oid': ""+
          resposta.data[0][i]['_id']+""}
      }
      for(i=0;i<resposta.data[1].length;i++){
        resposta.data[1][i]['_id'] = {'$oid': ""+
          resposta.data[1][i]['_id']+""}
      }
      for(i=0;i<resposta.data[2].length;i++){
        resposta.data[2][i]['_id'] = {'$oid': ""+
          resposta.data[2][i]['_id']+""}
      }
      for(i=0;i<resposta.data[3].length;i++){
        resposta.data[3][i]['_id'] = {'$oid': ""+
          resposta.data[3][i]['_id']+""}
      }
      var postData=JSON.stringify(resposta.data[0])
      var eventData=JSON.stringify(resposta.data[1])
      var messageData=JSON.stringify(resposta.data[2])
      var userData=JSON.stringify(resposta.data[3])

//Guardamos nos ficheiros json toda a informação de cada coleção
      var arr = [{ 'filename': './backup/posts.json',
        'content':postData},{ 'filename': './backup/events.json',
        'content':eventData}
      ,{ 'filename': './backup/message.json',
        'content':messageData},{ 'filename': './backup/users.json',
        'content':userData}];
      async.map(arr, getInfo, function (e, r) {
        console.log(r);
      });

      function getInfo(obj, callback) {
        fs.writeFile(obj.filename, obj.content, callback);
      }
    })
    .catch(erro => {console.log('Erro ao carregar da bd '+erro)})
  });
```


Routes-index.js - Continuação da Route Index

```
//Insere voto
router.post('/posts/vote/', verifyAuth, function (req, res) {
  axios.post('http://localhost:3000/api/posts/vote/', req.body)
    .then(resposta => {
      console.log(resposta.data)
    })
    .catch(erro => {
      console.log('Erro ao carregar da bd asdasd'+erro)
    })
});

module.exports = router;
```

4.4.3 Users Route

Routes-users.js - Criação da Route Users

```
var express = require('express');
var router = express.Router();
var passport = require('passport')
var axios = require('axios')
var jwt = require('jsonwebtoken')
var fs = require('fs')
var formidable = require('formidable')

/* GET users listing. */
router.get('/', function(req, res, next) {
  res.send('respond with a resource');
});

//Get Login Page
router.get('/signin', (req, res)=>{
  Users.list()
  .then(dados => res.jsonp(dados))
  .catch(erro => res.status(500).send('DEU ERRO NA LISTAGEMMMMM'))
})

//Logout
router.get('/logout', (req, res)=>{
  req.session.destroy(function (err) {
    res.redirect('/'); //Inside a callback... bulletproof!
  });
})
```

Routes-users.js - Continuação da Route Users

```
//Post Login
router.post('/signin', async (req, res, next) => {
  passport.authenticate('login', async (err, user, info) => {
    try {
      if(err || !user){
        if(err){
          return next(err);
        }else{
          return next(new Error(info.message))//TODO
        }
      }
      req.login(user, async (error) => {
        if( error ) return next(error)
        var myuser = { _id : user._id, email : user.email };
        // Geração do token
        var token = jwt.sign({ user : myuser }, 'dweb2018');
        req.session.token = token
        res.redirect('/home')
      });
    }
    catch (error) {
      return next(error);
    }
  })(req, res, next);
});
```

Routes-users.js - Continuação da Route Users

```
router.post('/signup', function(req, res) {
  var user
  var form = new formidable.IncomingForm()
  form.parse(req, (erro, fields, files) => {
    if (!files.profilePic.name) {
      user = {
        name: fields.name,
        username: fields.username,
        password : fields.password,
        email : fields.email,
        age: fields.age,
        gender: fields.gender,
        address:{
          country:fields.country,
          city: fields.city
        }
      }
      axios.post('http://localhost:3000/api/users/signup', user)
        .then(() => res.redirect('http://localhost:3000/'))
        .catch(erro => {
        })
    } else {
      var fenviado = files.profilePic.path
      var fnovo = '../Schmocial/public/images/profile/' +
        files.profilePic.name
      fs.rename(fenviado, fnovo, erro => {
        if (!erro) {
          user = {
            name: fields.name,
            username: fields.username,
            password : fields.password,
            email : fields.email,
            age: fields.age,
            gender: fields.gender,
            address:{
              country:fields.country,
              city: fields.city
            },
            picture:files.profilePic.name
          }
          axios.post('http://localhost:3000/api/users/signup', user)
            .then(() => res.redirect('http://localhost:3000/'))
            .catch(erro => {
            })
        } else {
          //res.render(error)
        }
      })
    }
  })
})
```

4.5 JQuery

4.5.1 posts.js

Jquery-posts.js - Criação do script que vai manipular os posts

```
//Iserção de voto
function voteFunc(user,post){
    var result = $('#'+''+post+'').text()
    $.ajax({
        type: "POST",
        contentType: "application/json",
        url: "http://localhost:3000/api/posts/vote",
        data: JSON.stringify({ user: user, post: post }),
        dataType: "json",
        success: function () {
            $("#upVote"+post).text(' Downvote')
            $("#upVoteBtn"+post).attr("onclick",
                "voteDown('"+user+"', '"+post+"')");
            $("#upVoteBtn"+post).attr("id","downVoteBtn"+post);
            $('#upVote'+post).attr({
                id: 'downVote'+post,
                class: 'fa fa-chevron-down'
            });
        }
    })
    $('#'+''+post+'').text((parseInt(result)+1))
}
```

Jquery-posts.js - Continuação do script que vai manipular os posts

```
//Remoção de voto
function voteDown(user,post){
  var result = $('#'+''+post+'').text()
  $.ajax({
    type: "POST",
    contentType: "application/json",
    url: "http://localhost:3000/api/posts/vote/remove",
    data: JSON.stringify({ user: user, post: post }),
    dataType: "json",
    success: function () {
      $("#downVote"+post).text(' Upvote')
      $("#downVoteBtn"+post).attr("onclick",
        "voteFunc('"+user+"','"+post+"')");
      $("#downVoteBtn"+post).attr("id","upVoteBtn"+post);
      $('#downVote'+post).attr({
        id: 'upVote'+post,
        class: 'fa fa-chevron-up'
      });
    }
  })
  $('#'+''+post+'').text((parseInt(result)-1))
}
```

Jquery-posts.js - Continuação do script que vai manipular os posts

```
//Adiciona os amigos à janela do chat
$.ajax({
  type: "GET",
  contentType: "application/json",
  url: "http://localhost:3000/api/friends/get/user/"+
    $('#userId').attr("name"),
  dataType: "json",
  success: function (data) {
    $("#chatUsers").html('')
    $("#chatUsers").append("<h3>Chat</h3>")
    if(data.length!=0){
      for(i=0; i<data[0].friends.length; i++){
        var myvar = '<p><span class="w3-hover-white"
        onclick="loadUser('+'\'+data[0].friends[i].id+'\'+data[0].friends[i].name+'\'+data[0].friends[i].name+'</span></p>';
        $("#chatUsers").append(myvar)
      }
    }
  }
})

//Remoção de comentario
function removeComent(idComment, idPost){
  $.ajax({
    type: "POST",
    contentType: "application/json",
    url: "http://localhost:3000/api/posts/comment/remove",
    data: JSON.stringify({ post: idPost, comment:idComment}),
    dataType: "json",
    success: function () {
      $('#comment'+idComment).remove()
      $('#image'+idComment).remove()
    }
  })
}
```

Jquery-posts.js - Continuação do script que vai manipular os posts

```
$(() => {
    $('#hashtags').load('http://localhost:3000/posts/hashtags');
    $('#addHashtag').click(function () {
        $('#hashtagContainer').append('<input class="w3-border-0 w3-padding w3-round-large" style="width:26%; margin-left:1%" type="text" name="hashtags[]" placeholder="Insert an hashtag" />');
    });
    $('#removeHashtag').click(function () {
        $('#hashtagContainer input:last-child').remove()
    });
    //Inserir comentario
    $(document).on('keydown', function (e) {
        var targetInput = $(e.target);
        if (targetInput.is('textarea')) {
            if (e.which == 13) {
                e.preventDefault();
                $('#commentSection').each(function () {
                    if ($(this).attr("name") == targetInput.attr('name')) {
                        var myvar = '<p class="w3-padding w3-round-large w3-light-grey"
                        style="display:inline-block;width:90%;"> <span style="color:blue;
                        font-weight: bold;">' + $('#userId').text() + '</span><span
                        style="margin-left:1%;">' + targetInput.val() + '</span><i class="fa fa-trash
                        fa-lg" style="float:right;"></i></p>';
                        $(this).append(myvar);
                    }
                });
                // $("#commentSection").append('<p>' + targetInput.val() + '</p>');
                ajaxPost(targetInput.val(), targetInput.attr('name'),
                    p => alert(JSON.stringify(p)),
                    e => {
                        console.log("AJAX ERRO:" + JSON.stringify(e));
                    });
                targetInput.val('');
            }
        }
    });
    function ajaxPost(texto, id, success, error) {
        $.ajax({
            type: "POST",
            contentType: "application/json",
            url: "http://localhost:3000/posts/comment",
            data: JSON.stringify({ comment: texto, id: id }),
            dataType: "json",
            success: success,
            error: error
        });
    }
});
```


4.5.2 events.js

Jquery-events.js - Criação do script que vai manipular os eventos

```
$((() => { //Carrega o evento mais proximo que participas
    var eventos,pic
    var contador = 0
    $.ajax({
        type: "GET",
        contentType: "application/json",
        url: "http://localhost:3000/api/events/byDate/",
        dataType: "json",
        success: function (data) {
            $("#conteudoEventos").html('')
            eventos = data
            if (eventos.length == 0) {
                var myvar = '<div class="w3-container">' +
                    '<p>Upcoming Events:</p>' +
                    '<p><strong>Não tem eventos próximos</strong></p>' +
                    '<p><a class="w3-button w3-block w3-white" href="/events">See more Events</a></p>' +
                    '</div>';
            } else {
                for (i = 0; i < eventos.length; i++) {
                    if(contador==1){
                        break;
                    }else{
                        for (j = 0; j < eventos[i].participants.length; j++) {
                            if (eventos[i].participants[j].id ==
                                $("#userId").attr('name')) {
                                    contador=1
                                    if (eventos[i].picture) {
                                        pic = ''
                                    } else { pic = ""}
                                    var myvar = '<div class="w3-container">' +
                                        '<p>Upcoming Events:</p>'+pic+' ' +
                                        '<p><strong>' + eventos[i].title + '</strong></p>' +
                                        '<p>' + eventos[i].date + '</p>' +
                                        '<p><a class="w3-button w3-block w3-white" href="/events">' +
                                        'See more Events</a></p>' + '</div>';
                                    break;}}}
```

Jquery-events.js - Continuação do script que vai manipular os eventos

```
if(contador==0 && i==eventos.length-1){
    var myvar = '<div class="w3-container">' +
        '    <p>Upcoming Events:</p>' +
        '    <p><strong>Não tem eventos próximos</strong></p>' +
        '    <p><a class="w3-button w3-block w3-white"' +
        'href="/events">See more Events</a></p>' +
        '    </div>';
        }
        }

    $("#conteudoEventos").append(myvar)
}
})
})
```

Jquery-events.js - Continuação do script que vai manipular os eventos

```
//Mostra a informação de cada evento
function showEventInfo(id,data,local,titulo,descricao,utilizador){
    $("#idDoEvento").text(id)
    $("#eventoData").text(data)
    $("#eventoLocal").text(local)
    $("#eventoTitle").text(titulo)
    $("#eventoDescription").text(descricao)

    var participantes
    $.ajax({
        type: "GET",
        contentType: "application/json",
        url: "http://localhost:3000/api/events/participants/" + id,
        dataType: "json",
        success: function (data) {
            $("#eventoParticipants").html('')
            participantes = data[0].participants
            if (participantes.length != 0) {
                for (i = 0; i < participantes.length; i++) {
                    $("#eventoParticipants").append(participantes[i].name)
                    if (participantes[i].id == utilizador) {
                        $("#acceptEvent").hide();
                        $("#cancelEvent").show();
                        break;
                    }else{
                        $("#acceptEvent").show();
                        $("#cancelEvent").hide();
                    }
                }
            } else {
                $("#acceptEvent").show();
                $("#cancelEvent").hide();
                $("#eventoParticipants").text("Ainda não existem participantes")
            }
            $("#eventInfo").css("display", "block");
        }
    })
}
```

Jquery-events.js - Criação do script que vai manipular os eventos

```
//Submete participação
function submeteParticipacao(idUtilizador, nameUtilizador) {
    var idPost = $("#idDoEvento").text()
    var participante = {
        id: idUtilizador,
        name: nameUtilizador
    }
    $.ajax({
        type: "POST",
        contentType: "application/json",
        url: "http://localhost:3000/api/events/participant",
        data: JSON.stringify({ idPost: idPost,
            participant: participante }),
        dataType: "json",
        success: function () {
            window.location.href = "http://localhost:3000/home"
        }
    })
}

//cancela participação
function cancelParticipacao(idUtilizador) {
    var idPost = $("#idDoEvento").text()
    $.ajax({
        type: "POST",
        contentType: "application/json",
        url: "http://localhost:3000/api/events/participant/remove",
        data: JSON.stringify({ idPost: idPost,
            participant: idUtilizador }),
        dataType: "json",
        success: function () {
            window.location.href = "http://localhost:3000/home"
        }
    })
}
```

4.5.3 connections.js

Jquery-connections.js - Criação do script que vai manipular a informação da página dos pedidos de amizade

```
//get friends request and display in page
var friends
$.ajax({
  type: "GET",
  contentType: "application/json",
  url: "http://localhost:3000/api/friends/get/"+$("#userId")
    .attr('name'),
  dataType: "json",
  success: function (data) {
    $("#friendsContainer").html('')
    if(data.length!=0){
      friends = data[0].friends
      var myvar = '<p>Friend Request:</p><span>'+data[0].friends[0]
        .name+'</span>'+
      '<div class="w3-row w3-opacity">'+
      '  <div class="w3-half"><button class="w3-button w3-block w3-green
w3-section decision" name="accept"
value="'+data[0].friends[0].id+'':'+data[0].friends[0].name+' "
title="Accept"><i class="fa fa-check"></i></button></div>'+
      '  <div class="w3-half"><button class="w3-button w3-block w3-red
w3-section decision" name="dennied"
value="'+data[0].friends[0].id+'':'+data[0].friends[0].name+' "
title="Decline"><i class="fa fa-times"></i></button></div>'+
      '</div>';
    }else{
      var myvar = '<p>Friend Request:</p><span>You have no friends
request</span>'
    }
    $("#friendsContainer").append(myvar)
  }
})
```

Jquery-connections.js - Continuação do script que vai manipular a informação da página dos pedidos de amizade

```
//load all users into the connections page
$.ajax({
  type: "GET",
  contentType: "application/json",
  url: "http://localhost:3000/api/users/",
  dataType: "json",
  success: function (data) {
    var res = $("#search").attr('name').split(":");
    for(i=0;i<data.length;i++){
      var newHtml = '<button class="close w3-button w3-block w3-green w3-section" name='+data[i]._id+'>Send Request Friend</button>'
      if(data[i]._id!=res[0]){
        for(j=0;j<data[i].friends.length;j++){
          if(data[i].friends[j].id==res[0]){
            newHtml = '<p>You are already friends</p>'
          }
        }
        if(!data[i].picture){
          pic=''
        }else{
          pic='<img src='+images/profile/"+data[i].picture+' class="w3-circle" width="60px">'
        }
      }
    }
    var myvar =
    ' <div class="row">' +
    '' +
    ' <div class="shadow">' +
    ' <div class="col-sm-12">' +
    ' <div class="col-sm-2">' +
    pic +
    ' </div>' +
    ' <div class="col-sm-8">' +
    ' <h4><a href="#">' + data[i].name + '</a></h4>' +
    ' <p><a href="#">' + data[i].friends.length + ' Friends</a></p>' +
    ' </div>' +
    ' <div class="col-sm-2">' +
    ' <br>' +
    newHtml +
    ' </div>' +
    ' </div>' +
    ' <div class="clearfix"></div>' +
    '<hr>'
  }
});
```

Jquery-connections.js - Criação do script que vai manipular a informação da página dos pedidos de amizade

```
//Filter Users by name
$("#search").keyup(function(){
    $.ajax({
        type: "GET",
        contentType: "application/json",
        url: "http://localhost:3000/api/users/name/"+$("#search").val(),
        success: function (data) {
            $("#listConnections").html('')
            for(i=0;i<data.length;i++){
                var myvar = '<div class="container">'+
                    ' <div class="top">'+
                    ' <h2>Friend\'s List</h2>'+
                    ' </div>'+
                    ' <div class="row">'+
                    ''+
                    ' <div class="shadow">'+
                    ' <div class="col-sm-12">'+
                    ' <div class="col-sm-2">'+
                    ' <img'
src="https://www.infrascan.net/demo/assets/img/avatar5.png"
class="img-circle" width="60px">'+
                    ' </div>'+
                    ' <div class="col-sm-8">'+
                    ' <h4><a href="#">'+data[i].name+'</a></h4>'+
                    ' <p><a href="#">'+data[i].friends.length+'
Friends</a></p>'+
                    ' </div>'+
                    ' <div class="col-sm-2">'+
                    ' <br>'+
                    ' <button class="close" name='+data[i]._id+'>Send
Request Friend</button>'+
                    ' </div>'+
                    ' </div>'+
                    ' <div class="clearfix"></div>'+
                    '<hr>'
                    ' </div>'+
                    ' </div>'+
                    '</div>';
                $("#listConnections").append(myvar)
            }
        })
    });
```

Jquery-connections.js - Criação do script que vai manipular a informação da página dos pedidos de amizade

```
//sends friendship request
$('body').on('click', 'button.close', function() {
  var res = $("#search").attr('name').split(":");
  var amigo = {
    id : res[0], //user logado
    name : res[1], //nome user logado
    state : "request"
  }

  $.ajax({
    type: "POST",
    contentType: "application/json",
    url: "http://localhost:3000/api/friends/new/",
    data: JSON.stringify({ userDest: $(this).attr('name'),
    userOrigin: amigo}),
    dataType: "json",
    success: function (data) {
      alert("Pedido enviado com sucesso!!")
      window.location.href = "http://localhost:3000/home"
    }
  })
});
```


Jquery-connections.js - Criação do script que vai manipular a informação da página dos pedidos de amizade

```
//accepts/dennies friendship request
$('body').on('click', 'button.decision', function() {
    var op = $(this).attr("name")
    var userToUpdate
    var infoFriend
    var url
    var user = $("#userId").attr('name')
    var res = $(this).val().split(":")
    if(op=="accept"){
        url = "http://localhost:3000/api/friends/accept/"
        userToUpdate = res[0]//id do user que queremos aceitar o pedido
        infoFriend = {
            id : user,//user logado
            name : $("#userId").text(),//nome user logado
            state : "accepted"
        }
    }else{
        userToUpdate = user
        infoFriend = res[0]
        url = "http://localhost:3000/api/friends/delete/"
    }
    alert(infoFriend.id)
    $.ajax({
        type: "POST",
        contentType: "application/json",
        url: url,
        data: JSON.stringify({user:userToUpdate,
            friend:infoFriend}),
        dataType: "json",
        success: function (data) {
            window.location.href = "http://localhost:3000/home"
        }
    })
});
```

Jquery-connections.js - Criação do script que vai manipular a informação da página dos pedidos de amizade

```
//Mostra apenas os amigos
$('#friends').click(function () {
  if($("#friends").attr('name')== 'stand-by'){
    var user = $("#search").attr('name').split(":");
    $("#listConnections").html('')
    for(j=0; j<users.length;j++){
      for(i=0;i<users[j].friends.length;i++){
        if(users[j].friends[i].id==user[0]){
          if(users[j].friends[i].state=="accepted"){
            var myvar = '<div class="container">' +
              '<div class="top">' +
              '<h2>Friend\'s List</h2>' +
              '</div>' +
              '<div class="row">' +
              '<div class="shadow">' +
              '<div class="col-sm-12">' +
              '<div class="col-sm-2">' +
              '<img'
src="https://www.infrascan.net/demo/assets/img/avatar5.png"
class="img-circle" width="60px">' +
              '</div>' +
              '<div class="col-sm-8">' +
              '<h4><a'
href="friends/' + users[j]._id + ':' + users[j].name + '>' + users[j].name
+ '</a></h4>' +
              '<p><a href="#">' + users[j].friends.length +
Friends</a></p>' +
              '</div>' +
              '<div class="col-sm-2">' +
              '<br>' +
              '</div>' +
              '</div>' +
              '<div class="clearfix"></div>' +
              '<hr>' +
              '</div>' +
              '</div>' +
              '</div>';
            $("#listConnections").append(myvar)
          }
        }
      }
      $("#friends").attr('name', 'checked')
      $('#friends').css("background-color", "red");
    }
  }else{window.location.href = "http://localhost:3000/friends"}});
```

4.5.4 chat.js

Jquery-chat.js - Criação do script que vai tratar da troca de mensagens

```
var socket = io.connect('http://localhost:3000');
socket.emit('join', {email: "user1@example.com"});
//Envio de mensagem para o utilizador em questão
$('#message').keypress(function(e){
    if(e.which == 13){//Enter key pressed
        if($("#message").val()!=""){
            var foto = $('#fotoPerfil').attr('src')
            var nome = $('#userId').text()
            var url
            var userOrigem = $('#userId').attr('name')
            if(empty==0){
                url="http://localhost:3000/api/messages/"
            }else{
                url="http://localhost:3000/api/messages/update"
            }
            $.ajax({
                type: "POST",
                contentType: "application/json",
                url: url,
                data: JSON.stringify({ mensagem: $("#message").val(),
                    idUser1:userOrigem, idUser2:personTalkingTo,
                    nome:nome, foto:foto}),
                dataType: "json",
                success: function () {
                }
            })
            socket.emit("news", { message: $("#message").val(),
                foto:foto,nome:nome, idUser1:userOrigem,
                idUser2:personTalkingTo
            })
            empty=1
        }
    }
});
})
```

Jquery-chat.js - Criação do script que vai tratar da troca de mensagens

```
socket.on('news', function (data) {
  if(data.idUser2==$('#userId').attr('name')||data.idUser1==$('#userId'
).attr('name')){
    var myvar = '<span
class="time-right" style="margin-left:1%;">'+data.nome+'</span>'+
'<p><font size="2">'+data.message+'</font></p>';
    $("#chatId").append(myvar)
    $("#message").val('')
    $("#chatId").animate({ scrollTop:
$('#chatId').prop("scrollHeight")}, 0);
    $("#message").attr("autofocus", true)
    $('#chatPop').css('display', 'block')
    personTalkingTo=data.idUser1
  }
});
```

Jquery-chat.js - Criação do script que vai tratar da troca de mensagens

```
//Carrega toda a informação sobre o user e respetiva popup
var personTalkingTo
var nomeUserDestino
var empty
function loadUser(id, name) {
    $("#chatId").html('')
    nomeUserDestino = name
    loadDataFromDB(id)

    personTalkingTo=id
    $('#chatUserName').text(name)
    $('#chatPop').css('display', 'block')
}
//Carrega as mensagens entre os utilizadores da BD
function loadDataFromDB(id) {
    var userOrigem = $('#userId').attr('name')
    $.ajax({
        type: "GET",
        contentType: "application/json",
        url: "http://localhost:3000/api/messages/"+userOrigem+'/'+id,
        dataType: "json",
        success: function (data) {
            if(data.length!=0){
                empty = 1
                for(i=0; i<data[0].mensagens.length;i++){
                    var myvar = '<span
class="time-right"
style="margin-left:1%;">'+data[0].mensagens[i].nome+'</span>'+
'<p><font
size="2">'+data[0].mensagens[i].texto+'</font></p>';
                    $("#chatId").append(myvar)
                }
                $("#chatId").animate({ scrollTop:
                $('#chatId').prop("scrollHeight")}, 0);
            }else{
                empty=0
            }
        }
    })
}
```