UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ARTES, CIÊNCIAS E HUMANIDADES

Documentação Técnica

Projeto 3

Gerson Revoredo Pereira - N°USP 8598728
Marco Modena Centeno - NºUSP 9377612
Rodrigo Kuba - N°USP 5364617
Thiago Bonfiglio Santos - N°USP 8598691

Sumário

1 – Projeto

O projeto de Licitação para as Micros e Pequenas empresas foi sugerido como um trabalho na disciplina “Governo Aberto”, da Escola de Artes, Ciências e Humanidades – EACH – USP, em parceria com o Observatório Social do Brasil.

Hoje, boa parte das Micro e Pequenas empresas brasileiras não participam de licitações de projetos públicos. E um dos principais motivos é a dificuldade no acesso às informações sobre licitações.

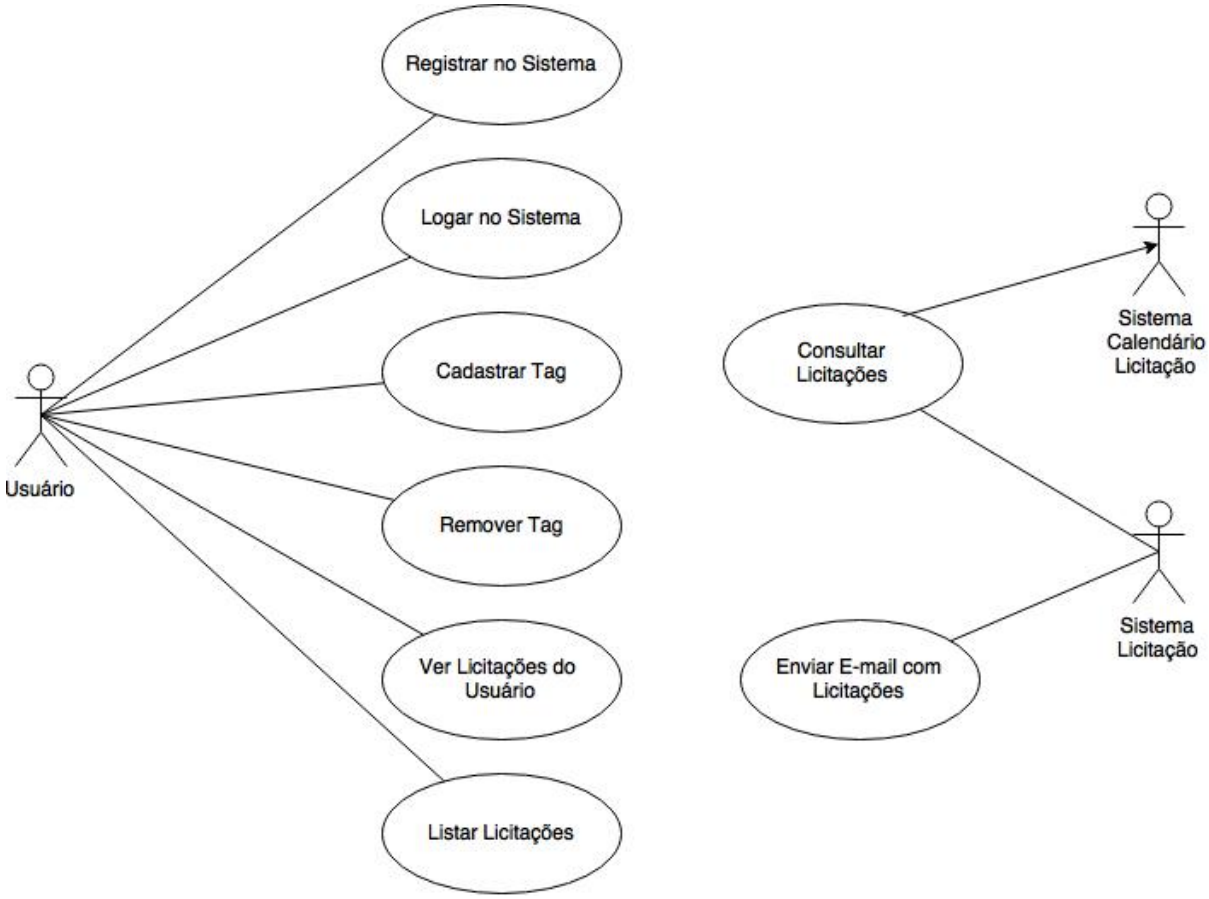
O projeto consiste em divulgar as licitações governamentais para as empresas cadastradas no sistema que queiram receber informações sobre novas licitações. Espera-se assim que as Micros e Pequenas Empresas passem a fazer parte desse importante mecanismo de utilização de recursos públicos.

2 – Requisitos

As informações necessárias para a criação do aplicativo de aviso de licitações para as Micros e Pequenas empresas foram levantadas através de reuniões periódicas com os especialistas do Observatório Social do Brasil: Gioia e Leonilda.

As informações sobre o calendário das licitações, assim como informações sobre elas, serão obtidas através de uma interface com o projeto 2.

2.1 - Diagrama de Caso de Uso



2.2 - Atores

Os atores que compõem o sistema são:

- **Usuário**: É a micro e pequena empresa que atuará como o usuário do sistema.
- **Sistema Calendário Licitação**: É o sistema responsável por fornecer as informações de licitações de diversas fontes,
- **Sistema Licitação**: É o sistema responsável realizar o cadastro do usuário, por buscar informações sobre licitações no Sistema Calendário Licitação e também por notificar os usuários cadastrados sobre novas licitações.

2.3 – Casos de Uso

2.3.1 – Registrar no Sistema

Descrição

Este Caso de Uso permite a um usuário se registrar no sistema.

Fluxo

1. O caso de uso inicia quando o usuário decide registrar no sistema.
2. O Sistema exibe os seguintes campos:
    a. Razão Social;
    b. Nome de usuário;
    c. E-mail do usuário;
    d. E-mail de confirmação do usuário;
    e. Senha;
    f. Confirmação da senha;
    g. Botão Registrar.
3. O usuário preenche os campos acima citados e clica no botão Enviar.
4. O Sistema verifica que todos os campos foram preenchidos. Caso contrário exibe a mensagem para preencher os campos.
5. O Sistema verifica que os valores dos campos E-mail e Confirmação do E-mail são iguais. Caso contrário exibe uma mensagem de E-mails diferentes.
6. O Sistema verifica que os valores dos campos Senha e Confirmação da Senha são iguais. Caso contrário exibe uma mensagem de Senhas diferentes.
7. O Sistema insere as informações do usuário em seu banco de dados.
8. Fim de caso de uso.

2.3.2 – Logar no Sistema

Descrição

Este Caso de Uso permite a um usuário logar no sistema.

Fluxo

1. O caso de uso inicia quando o usuário decide logar no sistema.
2. O Sistema exibe os seguintes campos:
    a. Nome de usuário;
    b. Senha do usuário;

c. Botão Entrar;

3. O usuário preenche os campos acima citados e clica no botão Entrar.
4. O Sistema verifica que todos os campos foram preenchidos. Caso contrário exibe a mensagem para preencher os campos.
5. O Sistema verifica se o usuário existe no banco de dados. Caso contrário exibe a mensagem de login inválido.
6. O Sistema verifica que o campo senha é válido para o login informado. Caso contrário exibe a mensagem de login inválido.
7. O Sistema gera um “token” para identificação do usuário em futuras requisições.
8. Fim de caso de uso.

2.3.3 – Cadastrar Tag

Descrição

Este Caso de Uso permite a um usuário cadastrar palavras-chaves (tags) para busca de licitações.

Fluxo

1. O caso de uso inicia quando o usuário decide cadastrar uma nova tag no sistema.
2. O Sistema verifica que o usuário está conectado no Sistema. Caso contrário, redireciona a requisição para a tela de login.
3. O Sistema exibe os seguintes campos:
    a. Tag;
    b. Botão Adicionar;
4. O usuário preenche os campos acima citados e clica no botão Adicionar.
5. O Sistema verifica que todos os campos foram preenchidos. Caso contrário exibe a mensagem para preencher os campos.
6. O Sistema verifica que a tag não existe para o usuário conectado no Sistema. Caso contrário, exibe a mensagem de tag já existente.
7. O Sistema insere a nova tag atrelada ao usuário conectado no Sistema.
8. Fim de caso de uso.

2.3.4 – Remover Tag

Descrição

Este Caso de Uso permite a um usuário remover palavras-chaves (tags.

Fluxo

1. O caso de uso inicia quando o usuário decide remover uma tag cadastrada no sistema.
2. O Sistema verifica que o usuário está conectado no Sistema. Caso contrário, redireciona a requisição para a tela de login.
3. O Sistema exibe os seguintes campos:
    a. Tag;
    b. Botão Remover;
4. O usuário preenche os campos acima citados e clica no botão Remover.
5. O Sistema verifica que todos os campos foram preenchidos. Caso contrário exibe a mensagem para preencher os campos.
6. O Sistema verifica que a tag existe para o usuário conectado no Sistema. Caso contrário, exibe a mensagem de tag inválida.
7. O Sistema remove tag atrelada ao usuário conectado no Sistema.
8. Fim de caso de uso.

2.3.5 – Ver Licitações do Usuário

Descrição

Este Caso de Uso permite a um usuário visualizar as suas licitações.

Fluxo

1. O caso de uso inicia quando o usuário decide visualizar as suas licitações.
2. O Sistema verifica que o usuário está conectado no Sistema. Caso contrário, redireciona a requisição para a tela de login.
3. O Sistema consulta as tags cadastradas pelo usuário.
4. O Sistema consulta as licitações que contenham em sua descrição pelo menos uma das tags consultadas no passo anterior.
5. O Sistema exibe as seguintes informações:
    a. Id;
    b. Título;
    c. Descrição;
    d. Data de Entrega;
6. Fim de caso de uso.

2.3.6 – Listar Licitações

Descrição

Este Caso de Uso permite a um usuário visualizar todas as licitações cadastradas no

Sistema.

Fluxo

1. O caso de uso inicia quando o usuário decide visualizar as licitações cadastradas no Sistema.
2. O Sistema consulta as licitações cadastradas no Sistema.
3. O Sistema exibe as seguintes informações:
    a. Id;
    b. Título;
    c. Descrição;
    d. Data de Entrega;
4. Fim de caso de uso.

2.3.7 – Consultar Licitações

Descrição

Este Caso de Uso permite ao Sistema de Licitações buscar novas licitações no Sistema Calendário Licitação.

Fluxo

1. O Sistema de Licitação solicita por novas licitações ao Sistema Calendário Licitação.
2. O Sistema envia como parâmetro de consulta a data corrente.
3. O Sistema Calendário Licitação retorna as novas licitações cadastradas para a data solicitada.
4. O Sistema insere as novas licitações em seu banco de dados.
5. Fim de caso de uso.

2.3.8 – Enviar E-mail com Licitações

Descrição

Este Caso de Uso permite ao Sistema de Licitações enviar e-mail aos usuários cadastrados no Sistema sobre as novas licitações.

Fluxo

1. O caso de uso inicia quando o Sistema de Licitação decide enviar e-mails notificando os usuários cadastrados no Sistema com as novas licitações.
2. O Sistema consulta todos os usuários cadastrados no Sistema.
3. Para cada usuário:

a. O Sistema consulta as tags associadas a esse usuário.
b. O Sistema consulta o e-mail.
c. O Sistema consulta as licitações com o parâmetro indicando que se trata de uma nova licitação.
d. O Sistema realiza o filtro das licitações consultadas anteriormente, buscando licitações que contenham no campo descrição pelo menos uma das tags do usuário.
e. O Sistema monta a mensagem de e-mail, com as informações das licitações filtradas no passo anterior.
f. O Sistema envia para o e-mail do usuário a mensagem construída no passo anterior.

4. O Sistema atualiza o status das novas licitações, informando que não são mais novas licitações.
5. Fim de caso de uso.

3 – Arquitetura

O Sistema de Licitação foi desenvolvido utilizando-se apenas ferramentas abertas na arquitetura e no desenvolvimento do projeto.

A arquitetura do projeto consiste em:

- Linguagens de Programação: JavaScript e HTML
- Camada de aplicação: Angular.js
- Servidor de aplicação: Node.js
- Banco de dados: Mongo.db
- Camada de integração: webservices RESTful com JSON.
- Hospedagem (Cloud IAAS): Amazon AWS
- Hospedagem (Cloud PAAS): Heroku (Node.js) e Mlab (MongoDB)

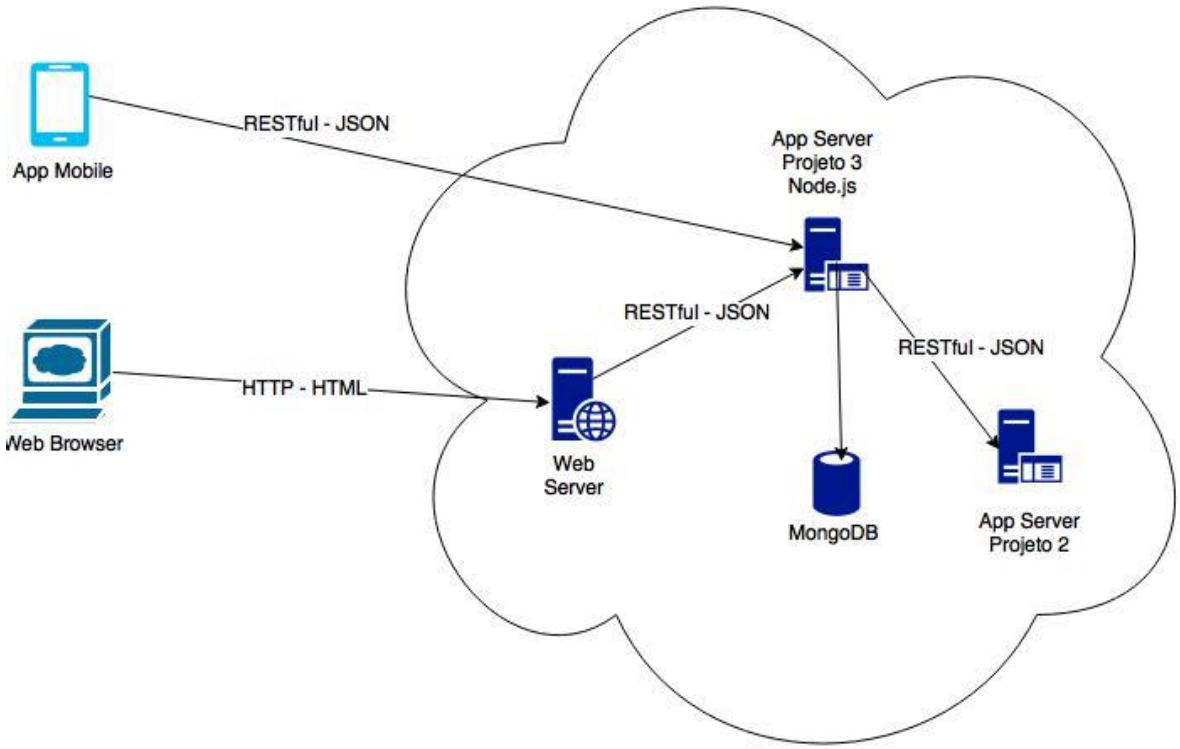


Figura 1: Diagrama de Arquitetura

4 – Desenvolvimento

O desenvolvimento do projeto consiste em duas partes. A parte da camada de apresentação (Front-End) e a parte de serviços (Back-End).

4.1 – Front-End

Nessa seção serão descritas as tecnologias utilizadas na parte front-end juntamente com algumas considerações.

4.1.1 – HTML

No HTML utilizamos o index.html para servir de home, com o menu fixado no topo. Com a função router do Angular, criamos um corpo do HTML (body) que possui a habilidade de se alterar baseado na escolha do menu.

As páginas HTML são segmentadas, facilitando a edição de terceiros sem conhecimento prévio e guardadas na pasta Panel.

Utilizamos também o Font Awesome, database de símbolos e marcas online, para achar símbolos, como o utilizado no botão de pesquisa no setor de tags.

4.1.2 – CSS

Utilizamos o css para mudanças visuais médias e pequenas, como cores e botões simples. A maior parte visual ficou sob responsabilidade do Bootstrap, devido a motivos discutidos na seção do Bootstrap.

Tivemos dois “elementos” principais criados no CSS, as bordas azuis e os botões.

As bordas azuis foram criadas para serem utilizadas na vertical e na horizontal, anexadas a uma div (na maioria dos casos), facilitando ao usuário a distinção de áreas e criando uma identidade visual em conjunto ao resto do site.

Por fim, os botões possuem uma função similar ao das bordas, criar uma identidade visual com animações próprias e facilidade de uso.

4.1.3 – Bootstrap

Foi utilizado o Bootstrap por dois motivos principais: a organização em grande escala e a portabilidade.

A organização de grande escala refere-se às classes como jumbotron e forms, que são classes de suporte de grande escala. Através do Bootstrap foi possível criar um menu mais elegante e funcional.

A portabilidade, onde o Bootstrap traz facilidades de construir um site responsivo, ou seja, o site se adapta de acordo com o aparelho utilizado pelo usuário.

4.1.4 – AngularJS

O AngularJS foi utilizado para facilitar o controle do conteúdo do site. Os módulos e funções utilizados foram:

- Router: utilizado para modificar e controlar o conteúdo da página, em conjunto com a modulação de todas os HTMLs. Ele cria um link no HTML baseado em um arquivo que foi associado a um controller angular, facilitando a customização de cada item.
- ng-repeat: possui a função de repetir, usando um padrão pré-definido (listas, grades, etc), até que todos os itens sejam processados, facilitando a exibição de um grande conjunto de informações de forma ordenada. Foi utilizado em todos as páginas com visualização de licitações e tags.

Foram criados também funções internas para ordenar, procurar e selecionar licitações e tags.

4.2 – Back-End

Nesta seção serão descritas as tecnologias utilizadas na parte de back-end, juntamente com algumas considerações.

4.2.1 – NodeJS

NodeJS é um interpretador de códigos escritos em JavaScript, utilizado no lado servidor da aplicação. Nesse projeto, o NodeJS foi utilizado para fornecer APIs WebServices Restful para serem utilizados/consumidos pelo lado "Front-end" da aplicação.

Com o NodeJS, podemos utilizar diversos pacotes/módulos "open-source", que facilita e agiliza o desenvolvimento de aplicações.

Nesse projeto, utilizamos os seguintes pacotes:

- body-parser: utilizado para realizar a conversão dos dados da tag "body" do HTML em forma de objeto.
- cookie-parser: utilizado para realizar a conversão dos dados do cookie em forma de objeto.
- debug: utilitário para ajudar na depuração dos códigos-fontes do projeto. Utilizado na fase de desenvolvimento da aplicação.
- express: um dos principais pacotes utilizado no projeto. É um arcabouço para utilização do protocolo HTTP e também para fornecer APIs "web-services" no formato RestFul. Faz parte do MEAN, acrônimo utilizado para identificar aplicações que utilizam o banco de dados, MongoDB, o Express, o arcabouço AngularJS e o servidor web NodeJS.
- jade: utilitário para escrever códigos em HTML.
- jsonwebtoken: utilitário utilizado para geração de "tokens". Foi utilizado na implementação do mecanismo de autenticação do projeto.
- moment: utilitário para o tratamento de datas e horas.
- mongoose: utilitário ODM ("Object Document Modeling") para o MongoDB.
- morgan: utilitário de log para NodeJS.
- passport: utilitário de autenticação para NodeJS. Utilizado juntamente com o jwt (jsonwebtoken), faz parte do mecanismo de autenticação da aplicação.
- cors (cross-origin resource sharing): utilitário de segurança. Permite ou não que navegadores ou servidores webs possam acessar recursos de domínios diferentes

4.2.2 – MongoDB

MongoDB é um banco de dados do tipo não-relacional (NoSQL - "Not Only SQL"), de alto desempenho e orientado a documentos do tipo JSON ("JavaScript Object Notation").

No projeto, o banco de dados "Licitation" é criado automaticamente pela aplicação.

Esse banco de dados possui dois tipos de documentos: "user" e "licitation":

```
var tagSchema = new Schema({
    tag:
    {
      type: String,
      required: true,
      unique: true
    }
});

var user = new Schema({
    username: String,
    password: String,
    email:
    {
      type: String,
      required: true
    },
    documentNumber: Number,
    admin:
    {
        type: Boolean,
        default: false
    },
    tags: [tagSchema]
});
```

Figura 2: Esquema do documento "user"

```
var licitation = new Schema({
    title: String,
    description: String,
    deliveryDate: Date,
    publishDate: Date
});
```

Figura 3: Esquema do documento "licitation"

Esses esquemas são utilizados pelo utilitário mongoose para validação dos dados a

serem inseridos no banco de dados e também para realizar a modelagem dos documentos.

4.3 – APIs RESTFul

As seguintes APIs RESTful são fornecidas pela aplicação, na parte de serviços (back-end):

1. **"/users"**
    a. verbo GET: Retorna os dados do usuário. Necessária autenticação.
        a. JSON de retorno:


```
{
  "_id": "594afb8b1232b81e4073e7b0",
  "username": "leonilda",
  "email": "leonilda@gmail.com",
  "documentNumber": 66788433,
  "__v": 2,
  "tags": [
    {
      "tag": "tomate",
      "_id": "594afbce1232b81e4073e7b1"
    },
    {
      "tag": "suco",
      "_id": "594afbd21232b81e4073e7b2"
    }
  ],
  "admin": false
}
```


2. **"/users/register"**
    a. verbo POST: Utilizado para criação de um novo usuário.
        a. JSON de entrada:


```
{
  "username": "usuario",
  "password": "senha",
  "email": "email@gmail.com.br",
  "documentNumber": 123456777
}
```


        b. JSON de retorno OK:


```
{
  "status": "Registration Successful!"
}
```


        c. JSON de retorno Erro:

```
{
  "err": {
    "name": "UserExistsError",
    "message": "A user with the given username is already registered"
  }
}
```


3. **"/users/login"**
   a. verbo POST: Utilizado para autenticação do usuário.
      a. JSON de entrada:


```
{
  "username": "usuario",
  "password": "senha"
}
```


      b. JSON de retorno OK:


```
{
  "status": "Login successful!",
  "success": true,
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9
      .eyIkX18iOnsic3RyaWN0TW9kZSI6dHJ1ZSwic2VsZWN0ZWQiOn
      iOiJpbml0IiwiYWRtaW4iOiJpbml0IiwidGFncyI6ImluaXQiLC
      iLCJzYWx0IjoiaW5pdCIsIl9pZCI6ImluaXQifSwic3RhdGVzIj
      ydWUsImRvY3VtZW50TnVtYmVyIjp0cnVlLCJlbWFpbCI6dHJ1ZS
      lcXVpcmUiOnt9fSwic3RhdGVOYW1lcyI6WyJyZXF1aXJlIiwibW
      zIjp7InNhdmUiOltudWxsLG51bGxdLCJpc05ldyI6W251bGwsbn
      kbWluIjpmYWxzZSwidGFncyI6W10sIl9fdiI6MCwiZG9jdW1lbn
      iLCJoYXNoIjoiOTRlNzRiYzI5M2YwMDU3YTAyMTViYzAxYTQ1MD
      mZTZkNTNkNGUzN2ZiNzMwNjU4OGE3YmY0YzgzZDQ4YTM2ZmEzNz
      wMzliN2Q3MWYwZGNjOTMwNjY2ZmNjNjVhNmVmNDNlM2RiMWEzOD
      5ZTA4OGM1OTE4ZmI2YjY4OWFkMTM0NTVjNGE4MTBlNTQ2ZDE2Yz
      1YWU2OTU0MGU2ZWQ4NTljN2M4MmQ3ZTU0NmVkODFhYmJiYmQ1YW
      2ZjM1NjM0NmY4Njg1NTMyY2I4NzJlOTZkNzA2ODEwYTY0NTc0MW
      iMmRlNWEwYzdhNDgxNTRhNmQxZWI4NDkzMWVmMWI5YTcxOThkMD
      jN2ZhZGU0YTJlNTQyZDhhNzYwYjEwNDI2YWJiNjk3YTY4NjU5Zm
      3MmFmMDA5M2FjNDUzZjEwNTY5YjFmMmRjMjg5Zjc4Zjc1MGYzYT
      iMWI1Y2VkMDE3MjU3YzI2YmI2NzI2MDk4OWYwOGE0ZmNhNjNhZW
      kNDA0NGZlNmE1YjgwNjhhZGNiYWJhMDJlYWY1ZDRhMGMxZTQ5OW
      lLCJpYXQiOjE0OTgxMDAxNjksImV4cCI6MTQ5ODEwMzc2OX0.QW
  "userId": "594b30f21232b81e4073e7b4"
}
```


      c. JSON de retorno Erro:

```
{
  "err": {
    "name": "IncorrectPasswordError",
    "message": "Password or username are incorrect"
  }
}
```

4. **"/users/tag"**
    a. verbo GET: Utilizado para recuperar as "tags" do usuário. Necessária autenticação.
        a. JSON de retorno:

```
[
  {
    "tag": "tomate",
    "_id": "594afbce1232b81e4073e7b1"
  },
  {
    "tag": "suco",
    "_id": "594afbd21232b81e4073e7b2"
  }
]
```

    b. verbo POST: Utilizado para incluir uma nova "tag" do usuário. Necessária autenticação.
        a. JSON de entrada:

```
{
  "tag": "tomate"
}
```

        b. JSON de retorno:

```
{
  "_id": "594afb8b1232b81e4073e7b0",
  "username": "leonilda",
  "email": "leonilda@gmail.com",
  "documentNumber": 66788433,
  "__v": 3,
  "tags": [
    {
      "tag": "tomate",
      "_id": "594afbce1232b81e4073e7b1"
    },
    {
      "tag": "suco",
      "_id": "594afbd21232b81e4073e7b2"
    },
    {
      "tag": "carro5",
      "_id": "594b34471232b81e4073e7b6"
    }
  ],
  "admin": false
}
```


5. **"/users/tag/:tagId"**
    a. verbo DELETE: Utilizado para deletar uma tag com o id "tagId". Necessária autenticação.
        a. JSON de retorno:


```
{
  "_id": "594afb8b1232b81e4073e7b0",
  "username": "leonilda",
  "email": "leonilda@gmail.com",
  "documentNumber": 66788433,
  "__v": 4,
  "tags": [
    {
      "tag": "tomate",
      "_id": "594afbce1232b81e4073e7b1"
    },
    {
      "tag": "suco",
      "_id": "594afbd21232b81e4073e7b2"
    }
  ],
  "admin": false
}
```

6. **"/licitations"**
    a. verbo GET: Utilizado para retornar as licitações cadastradas na aplicação.
        a. JSON de retorno:


```
{
  "licitations": [
    {
      "_id": "594744d9203eb91269f7e087",
      "title": "Licitacao Exemplo 1",
      "description": "Descrição exemplo de Licitação",
      "deliveryDate": "2017-06-19T03:28:25.000Z",
      "publishDate": "2017-06-19T03:28:25.000Z",
      "__v": 0
    },
    {
      "_id": "5947472d203eb91269f7e08b",
      "title": "Licitacao Exemplo 2",
      "description": "Descrição exemplo de Licitação 2",
      "deliveryDate": "2017-06-19T03:38:21.000Z",
      "publishDate": "2017-06-19T03:38:21.000Z",
      "__v": 0
    },
    {
      "_id": "59474916576aaf12866bd38b",
      "title": "Licitacao Exemplo 3",
      "description": "Descrição exemplo de Licitação 2",
      "deliveryDate": "2017-10-22T02:00:00.000Z",
      "publishDate": "2017-07-19T03:00:00.000Z",
      "__v": 0
    },
```


7. **"/licitations/user"**
    a. verbo POST: Utilizado para retornar as licitações cadastradas na aplicação e que contenham as "tags" cadastradas pelo usuário. Necessária autenticação.
        a. JSON de retorno:

```
{
  "licitations": [
    {
      "_id": "594abd771232b81e4073e7ac",
      "title": "Licitacao legumes, verduras e frutas",
      "description": "Descrição  de Licitação de tomate, alface, laranja e cebolas",
      "deliveryDate": "2017-12-01T02:00:00.000Z",
      "publishDate": "2017-06-19T03:00:00.000Z",
      "__v": 0
    },
    {
      "_id": "594afc251232b81e4073e7b3",
      "title": "Licitacao bebidas",
      "description": "Descrição  de Licitação de suco e água",
      "deliveryDate": "2017-12-01T02:00:00.000Z",
      "publishDate": "2017-06-19T03:00:00.000Z",
      "__v": 0
    }
  ]
}
```


8. **"/jobs/notification"**
    a. verbo POST: Utilizado para enviar e-mails aos usuários cadastrados, com as novas licitações que foram obtidas pelo Sistema.

5 – Deploy

O sistema de Licitação foi instalado nos serviços em Nuvem.

Para o front-end e o back-end, foi utilizado o heroku, que é um serviço PaaS (Plataform as a Service) para hospedar sites e aplicações NodeJS.

Para o banco de dados, foi utilizado o mLab, que é um serviço PaaS para bancos MongoDB.

Para a instalação dos serviços, é necessário possuir contas no heroku e no mLab.

5.1 – Banco de Dados

Para realizar o deploy do banco de dados MongoDB no mLab, seguir os seguintes passos:

1. No site do mLab, em “MongoDB Deployments”, clicar no botão “Create New”.
2. Escolher um Cloud Provider (Amazon AQW, Microsoft Azure ou Google Clooud Plataform).

3. Escolher um Plan Type.
4. Clicar em Continue.
5. Escolher uma Região e clicar em Continue.
6. Digitar um nome para o banco de dados. No exemplo, foi criado o banco licitation. Clicar em Continue.
7. Confirmar os dados e clicar em Submit order.
8. Após a criação do banco, será necessário criar um usuário de banco de dados.
9. No banco de dados, clicar em Users. E depois clicar em Add database user.
10. Digitar o nome e o usuário do banco de dados. E clicar em Create.
11. O seguinte endereço deverá ser utilizado na configuração do banco de dados na parte back-end:
    mongodb://<dbuser>:<dbpassword>@ds127063.mlab.com:27063/licitation

5.2 – Back-End

Para realizar o deploy da parte back-end da aplicação, é necessário seguir os seguintes passos:

1. Na pasta do projeto Back-End, inicializar um projeto git, conforme comando:
   git init
2. Criar uma aplicação no heroku, conforme comando:
   heroku create
3. O heroku criará um novo domínio para a aplicação. No exemplo a seguir, foi criada a aplicação infinite-refuge-17347.
4. Executar o comando abaixo para configurar o git:
   heroku git:remote -a infinite-refuge-17347
5. Alterar o arquivo de configuração cfg/config.js, atualizando a configuração do banco de dados MongoDB, fornecido pelo mLab:
   'mongoUrl' :
   'mongodb://govberto:90govberto91@ds127063.mlab.com:27063/licitation'
6. Adicionar o projeto no git, conforme comando abaixo:
   git add app.js bin/ cfg/ models/ package.json public/ routes/ views/
7. Realizar o commit do projeto no git:
   git commit -m "Deploy 1"
8. Subir o projeto para o heroku:
   git push heroku master

5.3 – Front-End

Para realizar o deploy da parte front-end da aplicação, é necessário seguir os seguintes passos:

1. Na pasta do projeto Front-End, inicializar um projeto git, conforme comando:
   git init
2. Criar uma aplicação no heroku, conforme comando:
   heroku create
3. O heroku criará um novo domínio para a aplicação. No exemplo a seguir, foi criada a aplicação desolate-ocean-33378.
4. Executar o comando abaixo para configurar o git:
   heroku git:remote -a desolate-ocean-33378
5. Alterar o arquivo js/licitationService.js, configurando a chave “url” com o endereço do serviço back-end. Exemplo:
   url: 'https://infinite-refuge-17347.herokuapp.com'
6. Adicionar o projeto no git:
   git add -A
7. Realizar o commit do projeto no git:
   git commit -m "Deploy 1"
8. Subir o projeto para o heroku:
   git push heroku master
9. Após o deploy, entrar na página web:
   http://desolate-ocean-33378.herokuapp.com