

# Manual do Administrador

---

Manual desenvolvido  
pelo grupo Gakki, em  
colaboração com o  
Inteli e a Yamaha.

# Introdução

---

Este manual foi desenvolvido com o objetivo de auxiliar os desenvolvedores que irão dar continuidade ao Yamaha Project Management ou que gostariam de entender o processo de lógica de programação utilizado na aplicação.

Portanto, foi dividido em 10 seções que explicam trechos centrais do código e dão direcionamento do que cada área vai executar.

# Sumário

---

- ❑ Src;
- ❑ Controllers;
- ❑ Routes;
- ❑ Database;
- ❑ Node\_modules;
- ❑ Public;
- ❑ Views;
- ❑ Index;
- ❑ Package .

# SRC

---

A pasta “src” (*System Resource Controller*) comporta os programas executáveis redigidos conforme a necessidade do projeto. Logo, todos os códigos e comandos necessários para o desenvolvimento da aplicação estão nessa pasta em 9 subdivisões.



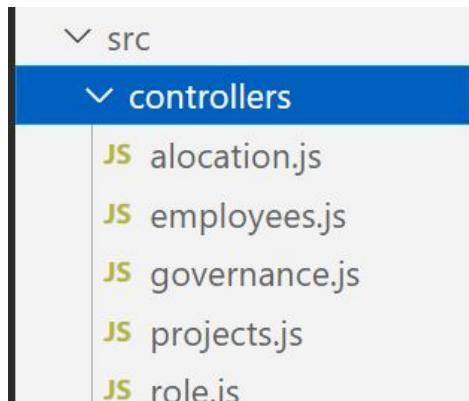
# Controllers

---

Sabe-se que a aplicação requer a utilização de dados armazenados no banco de dados.

Para isso, é necessário a criação de rotas (routes), que encaminham as requisições do cliente para as funções apropriadas do controlador (controller).

Primeiramente, cria-se todas as funções de retorno (controllers) que as rotas (routes) chamarão.



- Alocação;
- Funcionários;
- Governança;
- Projeto;
- Funções.

# Controllers

---

Observa-se a utilização do “Express” pois permite agrupar os manipuladores de rotas da aplicação e acessá-los usando um prefixo de rota comum.

Além das rotas para o sqlite3 e body-parser, módulo capaz de converter a informação enviada pelo usuário e transfere para o formato Json.

```
1 const express = require('express');
2 const sqlite3 = require('sqlite3').verbose();
3 const bodyParser = require('body-parser');
4
5 const dbPath = './database/yamaha.db';
6 const db = new sqlite3.Database(dbPath);
7 const app = express();
8
9 const getAllAllocations = (req, res) =>{
10
11 }
```

# Controllers

---

O “req” e “res” do Express, são os objetos de requisição e resposta, correspondente às horas alocadas nos meses do ano.

```
13 const createAlocation = (req, res) =>{
14     const sql = 'INSERT INTO Alocacao (HorasJaneiro, HorasFevereiro, HorasMarco, HorasAbril,
15     const { idProject } = req.query.id;
16     const beginDate = req.body.inicial;
17     const finalDate = req.body.final;
18     const hrJan = req.body.hrjan;
19     const hrFeb = req.body.hrfev;
20     const hrMar = req.body.hrmar;
21     const hrApr = req.body.hrabr;
22     const hrMay = req.body.hrmai;
23     const hrJun = req.body.hrjun;
24     const hrJul = req.body.hrjul;
25     const hrAug = req.body.hrago;
26     const hrSep = req.body.hrset;
27     const hrOct = req.body.hroutr;
28     const hrNov = req.body.hrnov;
29     const hrDec = req.body.hrdez;
30
31     db.run(sql, [hrJan, hrFeb, hrMar, hrApr, hrMay, hrJun, hrJul, hrAug, hrSep, hrOct, hrNov
32     if(err){
33         throw err;
34     } else {
35         res.render('novo');
36     }
37 });
38 }
39
40 module.exports = {
41     getAllAllocations,
42     createAlocation
43 }
```

# Controllers

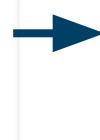
Utilizando o controller/employees.js como exemplo, pois, as outras áreas seguem a mesma construção com os métodos CRUD implementados na aplicação.

Método GET: é aplicado para a representação dos dados requeridos que estão hospedados no servidor.

```
12 const getAllEmployees = (req, res) =>{
13   const sql = 'SELECT * FROM Funcionario INNER JOIN Governanca ON Governanca.GovernancaID
14   db.all(sql, [], (err, rows) =>{
15     if(err){
16       throw err;
17     } else {
18       res.json(rows);
19     }
20   });
21 }
22
23 const getEmployeeById = (req, res) =>{
24   const { id } = req.params;
25   const sql = `SELECT * FROM Funcionario WHERE FuncionarioID = ${id}`;
26   db.get(sql, [], (err, row) =>{
27     if(err){
28       throw err;
29     } else {
30       res.json(row);
31     }
32   });
33 }
```



Método  
Get



Método  
Get

# Controllers

```
57 const updateEmployee = (req, res) =>{
58   const { id } = req.params;
59
60   const firstName = req.body.Nome;
61   const lastName = req.body.Sobrenome;
62   const employeeYamaha = req.body.FuncionarioYamaha;
63   const register = req.body.Registro;
64   const hoursProject = req.body.HorasProjetos;
65   const position = req.body.Funcao;
66   const employeesID = req.body.FuncionarioID;
67   const governaceID = req.body.GovernancaID;
68   const functionID = req.body.FuncaoID;
69
70   const sql = `UPDATE Funcionario SET Nome = ?, Sobrenome = ?, FuncionarioYamaha = ?, Registro = ?, HorasProjetos = ?, Funcao = ?, FuncionarioID = ?, GovernancaID = ?, FuncaoID = ?`;
71   db.run(sql, [firstName, lastName, employeeYamaha, register, hoursProject, position, empl
72     if(err){
73       throw err;
74     } else {
75       res.redirect('back');
76     }
77   });
--
```

## L Cláusula do SQLite: update(Comando DML)

src > controllers > JS employees.js > ...

```
35 const createEmployee = (req, res) =>{
36   const firstName = req.body.nome;
37   const lastName = req.body.sobrenome;
38   const funcYamaha = req.body.btnradio;
39   const register = req.body.registro;
40   const governace = req.body.governanca;
41   const company = req.body.empresa;
42   const durationContract = req.body.duracaocontrato;
43   const hoursProject = req.body.jornadaTrabalho;
44   const functionID = req.body.funcaoID;
45
46   const sql = 'INSERT INTO Funcionario (Nome, Sobrenome, FuncionarioYamaha, Registro, Empresa, Governanca, Funcao, FuncaoID)
47   console.log(funcYamaha)
48   db.run(sql, [firstName, lastName, funcYamaha, register, company, durationContract, hoursProject, governace, functionID]);
49     if(err){
50       throw err;
51     } else {
52       res.render('novoFuncionario');
53     }
54   );
55 }
```



## Cláusula do SQLite:create

# Controllers

```
79
80 const deleteEmployee = (req, res) =>{
81   const { id } = req.params;
82
83   const sql = `DELETE FROM Funcionario WHERE FuncionarioID = ${id}`;
84   db.run(sql, [], (err) =>{
85     if(err){
86       throw err;
87     } else {
88       res.send('Funcionário deletado com sucesso!');
89     }
90   });
91
92
93 module.exports = {
94   getAllEmployees,
95   getEmployeeById,
96   createEmployee,
97   updateEmployee,
98   deleteEmployee
99 }
```

→ Método  
DELETE

Método DELETE: pode deletar qualquer recurso especificado.

# Routes

---

## routes

<b>JS</b> alocation.js	→ Alocation
<b>JS</b> employees.js	→ Employees
<b>JS</b> governance.js	→ Governance
<b>JS</b> projects.js	→ Projects
<b>JS</b> role.js	→ Role

# Routes

---

## Alocation



Alocação dos funcionários nos projetos.

Routes: encaminham requerimentos para as funções do controller.

```
1 const express = require('express');
2 const alocationController = require('../controllers/alocation');
3 const router = express.Router();
4
5 router.get('/', alocationController.getAllAlocations);
6
7 router.post('/', alocationController.createAlocation);
8
9 module.exports = router;
```

Método POST: envia os parâmetros no corpo da requisição.

# Routes

---

## Employees/governance/role

Requerimento dos funcionários por meio do Id (Identify Document), nos métodos GET, POST e DELETE.

```
1 const express = require('express');
2 const employeesController = require('../controllers/employees');
3
4 const router = express.Router();
5
6 router.get('/', employeesController.getAllEmployees);
7
8 router.get('/:id', employeesController.getEmployeeById);
9
10 router.post('/', employeesController.createEmployee);
11
12 router.post('/:id', employeesController.updateEmployee);
13
14 router.delete('/:id', employeesController.deleteEmployee);
15
16 module.exports = router;
```

# Routes

---

## Projects

Caminhos criados para o requerimento, envio e apagamento da informação.

```
1 const express = require('express');
2 const projectsController = require('../controllers/projects');
3
4 const router = express.Router();
5
6 router.get('/', projectsController.getAllProjects);
7
8 //router.get('/alterar', projectsController.alterProject);
9
10 //router.get('/:id', projectsController.getProjectById);
11
12 router.post('/', projectsController.createProject);
13
14 //router.post('/update', projectsController.updateProject);
15
16 router.delete('/:id', projectsController.deleteProject);
17
18 module.exports = router;
```

# Database

---

Visto que as informações precisam estar contidas no banco de dados, o espaço “database” comporta os arquivos do DB browser para SQLite.

Conforme a necessidade do projeto, foram criadas 6 tabelas.

Nome	Tipo	Esquema
▼ Tabelas (6)		
> Alocacao		CREATE TABLE Alocacao(
> Funcao		CREATE TABLE "Funcao" (
> Funcionario		CREATE TABLE "Funcionari
> Governanca		CREATE TABLE "Governanc
> Projeto		CREATE TABLE "Projeto" (
> sqlite_sequence		CREATE TABLE sqlite_sequ
▼ Índices (0)		
▼ Vistas (0)		
▼ Gatilhos (0)		



Tabelas no DB browser para SQLite.

# Database

Nome	Tipo
▼ Tabelas (6)	
▼ Alocacao	
AlocacaoID	INTEGER
HorasJaneiro	INTEGER
HorasFevereiro	INTEGER
HorasMarco	INTEGER
HorasAbril	INTEGER
HorasMaio	INTEGER
HorasJunho	INTEGER
HorasJulho	INTEGER
HorasAgosto	INTEGER
HorasSetembro	INTEGER
HorasOutubro	INTEGER
HorasNovembro	INTEGER
HorasDezembro	INTEGER
DataInicialAlocacao	DATE
DataFinalAlocacao	DATE
ProjetoID	INTEGER
FuncionarioID	INTEGER
> ▼ Funcao	
FuncaoID	INTEGER
Titulo	TEXT
Area	TEXT
▼ Funcionario	
FuncionarioID	INTEGER
Nome	TEXT
Sobrenome	TEXT
FuncionarioYamaha	INTEGER
Registro	INTEGER
Empresa	TEXT
DuracaoContrato	INTEGER
HorasProjetos	INTEGER
GovernancaID	INTEGER
FuncaoID	INTEGER
▼ Governanca	

Entidades implementadas nas tabelas “Alocação”, “Função” e “Funcionário”.

# Database

---

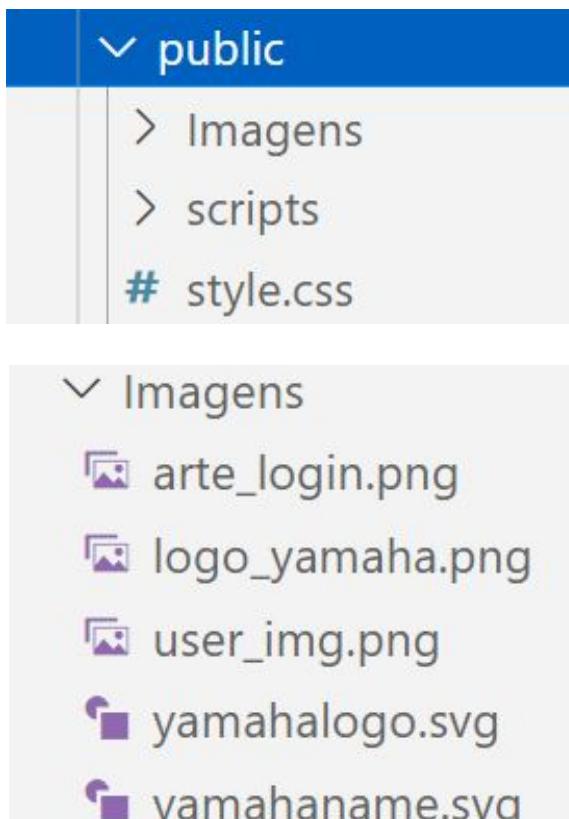
	GovernancaID	INTEGER
	País	TEXT
	Estado	TEXT
	Endereço	TEXT
	Projeto	
	ProjetoID	INTEGER
	NomeProjeto	TEXT
	Descrição	TEXT
	PrincipalResponsável	TEXT
	DataInicial	DATE
	DataFinal	DATE
	GovernancaID	INTEGER
	FuncionárioID	INTEGER
	sqlite_sequence	
	Índices (0)	
	Vistas (0)	
	Gatilhos (0)	

Entidades implementadas nas tabelas “Governança” e “Projeto”.

# Public

---

A pasta “Public” armazena documentos que fazem parte do Frontend, como as Imagens (imagens utilizadas durante a aplicação) e o arquivo CSS, e o script básico geral.



# Public

---

## ✓ scripts

JS alterar_funcionari...	Alterar_funcionário
JS alterar_projeto.js	Alterar_projeto
JS atribuir.js	Atribuir
JS graficos.js	Gráficos
JS home.js	Home
JS index.js	Index
JS nova_funcao.js	Nova_função
JS nova_governanca.js	Nova_governança
JS novo_funcionario.js	Novo_funcionário
JS novo_projeto.js	Novo_projeto
JS tabela_funcionari...	Tabela_funcionário

# Public

---

## Alterar\_funcionário

Ao fazer o cadastro de um novo funcionário ou a edição de informações específicas, o usuário precisa receber o feedback das ações.

```
1 if(localStorage.getItem('message')){  
2     if(localStorage.getItem('message') == 'updated employee'){  
3         toastShow();  
4         localStorage.removeItem('message');  
5     }  
6 }  
7  
8 function addToastUpdate(){  
9     localStorage.setItem('message', 'updated employee');  
10 }  
11  
12 function toastShow(){  
13     swal("Funcionário(a) atualizado com sucesso!", '', "success", {  
14         dangerMode: true,  
15     });  
16 }
```

# Public

---

## Alterar\_projeto

Ao fazer o cadastro de um novo projeto ou a edição de informações específicas, o usuário precisa receber o feedback das ações.

```
1 if(localStorage.getItem('message')){  
2     if(localStorage.getItem('message') == 'updated project'){  
3         toastShow();  
4         localStorage.removeItem('message');  
5     }  
6 }  
7  
8 function addToastUpdate(){  
9     localStorage.setItem('message', 'updated project');  
10 }  
11  
12 function toastShow(){  
13     swal("Projeto atualizado com sucesso!", '', "success", {  
14         dangerMode: true,  
15     });  
16 }
```

# Public

## Atribuir

XMLHttpRequest = é possível atualizar partes de uma página web, sem recarregar a página inteira

```
1 let ajax = new XMLHttpRequest();
2 ajax.open('GET', '/employees', true);
3 ajax.onreadystatechange = () =>{
4     let response = JSON.parse(ajax.responseText);
5     //let disponibile = document.getElementById('disponiveis');
6     for(let i = 0; i < response.length; i++){
7         $('#disponiveis').append(
8             <div class="todo" draggable="true">
9                 <button id="add_btn" data-target-modal="#todo_form" value="${response[i].Nome}">
10                <span class="close">&times;</span>
11            </div>;
12        }
13     const todos = document.querySelectorAll('.todo');
14     const all_status = document.querySelectorAll('.status');
15     let draggableTodo = null;
16
17     todos.forEach(
18         (todo) => {
19             todo.addEventListener('dragstart', dragStart);
20             todo.addEventListener('dragend', dragEnd);
21         });
22 }
```

Identificação do objeto sendo segurado (dragStart) e soltado do mouse(dragEnd).

```
22
23     function dragStart(){
24         draggableTodo = this;
25         console.log(this);
26         setTimeout(() => {
27             this.style.display = 'none';
28         }, 0);
29     }
30
31     function dragEnd(){
32         draggableTodo = null;
33         setTimeout(() => {
34             this.style.display = 'block';
35         }, 0);
36     }
37
38     all_status.forEach(status => {
39         status.addEventListener('dragover', dragOver);
40         status.addEventListener('dragenter', dragEnter);
41         status.addEventListener('dragleave', dragLeave);
42         status.addEventListener('drop', dragDrop);
43     });
44 }
```

# Public

## Atribuir

```
45     function dragOver(e){  
46         e.preventDefault();  
47     }  
48  
49     function dragEnter(){  
50         this.style.border = '1px dashed #ccc';  
51     }  
52  
53     function dragLeave(){  
54         this.style.border = 'none';  
55     }  
56  
57     function dragDrop(){  
58         this.style.border = 'none';  
59         this.appendChild(draggableTodo);  
60     }  
61
```

→ 1. DragOver

→ 2. DragEnter

→ 3. DragLeave

→ 4. DragDrop

1. Reconhece o elemento(atribuído ou disponível) no qual o objeto está em cima;
2. É a operação que confirma quando o objeto arrastado entrou em um elemento;
3. Confirma que o objeto saiu de um elemento;
4. Reconhece o objeto dentro do elemento após soltar o mouse.

# Public

---

## Atribuir

Modal que é acionado quando clicado no nome do funcionário, com as informações do seu nome completo, a data inicial e final, além das horas mensais destinadas à projeto.

```
 51
 52
 53
 54 // Modal
 55
 56 const btns = document.querySelectorAll("[data-target-modal]");
 57 const close_modals = document.querySelectorAll(".modal-btn");
 58 const overlay = document.querySelector("#overlay");
 59
 60
 61 btns.forEach(btn => {
 62   btn.addEventListener('click', () => {
 63     document.querySelector(btn.dataset.targetModal).classList.add('active');
 64     overlay.classList.add("active");
 65   })
 66 });
 67
 68
 69 close_modals.forEach(btn => {
 70   btn.addEventListener('click', () => {
 71     //Dicas maneiras de remover o Modal.
 72     document.querySelector(btn.dataset.targetModal).classList.remove('active');
 73     // btn.closest(".modal").classList.remove("active");
 74     overlay.classList.remove("active");
 75   })
 76 });
 77
 78
 79
 80
 81
 82
 83
 84
 85
 86
 87
 88
 89
 90
 91
 92
 93
```

O `window.onclick` representa qualquer lugar da tela possuir reação para fechar o modal.

```
83
84 window.onclick = (e) => {
85   if (e.target == overlay) {
86     const modals = document.querySelectorAll('.modal');
87     modals.forEach(modal => modal.classList.remove('active'));
88     overlay.classList.remove("active");
89   }
90 }
91
92
93 ajax.send();
```

# Public

---

## Gráficos

Código da biblioteca (Chart.js) para visualização em formato de gráficos.

```
1 //Gráfico bar
2 let barGrafico = document.getElementById("barGrafico");
3 const config = {
4   type: "line",
5   options: {
6     plugins: {
7       legend: {
8         display: true,
9         position: 'bottom'
10      },
11    },
12  },
13  data: {
14    labels: [
15      "Janeiro",
16      "Feverreiro",
17      "Marco",
18      "Abril",
19      "Maio",
20      "Junho",
21      "Julho",
22      "Agosto",
23      "Setembro",
24    ],
25    datasets: [
26      {
27        label: "humanResources",
28        data: [
29          1000, 6900, 7200, 7100, 5600, 6010, 5000, 6000, 5600, 5000, 5750,
30          5590,
31        ],
32        backgroundColor: ["#247BA0"],
33        borderColor: "#0A2463",
34        fill: true,
35        order: 2,
36      },
37    ],
38  },
39}
```



Especificação dos dados estáticos e layout do gráfico.

```
28 |   datasets: [
29 |   {
30 |     label: "humanResources",
31 |     data: [
32 |       1000, 6900, 7200, 7100, 5600, 6010, 5000, 6000, 5600, 5000, 5750,
33 |       5590,
34 |     ],
35 |     backgroundColor: ["#247BA0"],
36 |     borderColor: "#0A2463",
37 |     fill: true,
38 |     order: 2,
39 |   },
40 | }
```



# Public

---

## Gráficos

readyState:Mantém o status do XMLHttpRequest.

```
151 ajax2.onreadystatechange = () => {
152   if (ajax2.status === 200 && ajax2.readyState === 4) {
153     let response = JSON.parse(ajax2.responseText);
154     for (let i = 0; i < response.length; i++) {
155       let month = parseInt(response[i].DataInicial.slice(5, 7));
156       // percorre cada projeto e incrementa a variável de acordo com o mês do projeto
157       if (month === 1) {
158         january++;
159       }
160       if (month === 2) {
161         february++;
162       }
163
164       if(month === 3){
165         march++;
166       }
167
168       if(month === 4){
169         april++;
170       }
171     }
172   }
173 }
```

Define uma função a ser executada quando o readyState for alterado e as informações do gráfico forem percorridas.

# Public

---

## Home

Também define uma função a ser executada quando o readyState for alterado e as informações da home forem percorridas (utilizada na tabela\_funcionario, já que está na home).

```
1 // Requisição ajax que retorna todos os projetos cadastrados no banco de dados
2 let ajax = new XMLHttpRequest();
3 ajax.open('GET', '/projects', true);
4
5 ajax.onreadystatechange = () =>{
6   if(ajax.status === 200 && ajax.readyState === 4){
7     let response = JSON.parse(ajax.responseText);
8     console.log(response);
9     for(let i = 0; i < response.length; i++){
10       $('#tableProjects').append(
11         `<tr>
12           <td>${response[i].ProjetoID}</td>
13           <td>${response[i].NomeProjeto}</td>
14           <td>${response[i].PrincipalResponsavel}</td>
15           <td>${response[i].Estado}</td>
16           <td><a href="/alterarprojeto?id=${response[i].ProjetoID}" class="update">
17             </a>
18           </td>
19         `);
20     }
21   }
22
23 ajax.send();
```

# Public

## Index

Quando os botões forem selecionados, eventos são adicionados ao elemento referenciado.

# Public

---

Nova função/governança/funcionário /projeto

Ao passar a função para o método `getItem()`, o valor é retornado para a interface.

```
1 if(localStorage.getItem('message')){  
2     if(localStorage.getItem('message') == 'created role'){  
3         toastShow();  
4         localStorage.removeItem('message');  
5     }  
6 }  
7  
8 function addToast(){  
9     localStorage.setItem('message', 'created role');  
10 }  
11  
12 function toastShow(){  
13     swal("Função cadastrada com sucesso!", '', "success", {  
14         dangerMode: true,  
15     });  
16 }
```

# Public

---

## CSS

Indica-se no Guia de estilo da aplicação que a fonte de texto utilizada é a Roboto.

```
1 @import url('
2 https://fonts.googleapis.com/css2?family=Roboto:wght@100;300;400;500;700&display=swap');
3
```

Paleta de cores fixa da aplicação.

```
13 :root
14 {
15   --primaria: □#fff;
16   --secundaria: ■#0a0a6a;
17   --terciaria: ■#e53420;
18   --cinza: □#f2f2f2;
19   --preto: ■#000;
20 }
21
```

# Public

---

## CSS

Codificação de estilo da barra lateral e horizontal superior da aplicação.

```
40  .navbar
41  {
42    position: fixed;
43    width: 300px;
44    height: 100%;
45    background: var(--secundaria);
46    border-left: 10px solid var(--secundaria);
47    transition: 0.5s;
48    overflow: hidden;
49  }
50
51  .navbar.active
52  {
53    width: 80px;
54    transition: 0.5s;
55  }

57  .navbar ul
58  {
59    position: absolute;
60    top: 0;
61    left: 0;
62    width: 100%;
63  }
64
65  .navbar ul li
66  {
67    position: relative;
68    width: 100%;
69    list-style: none;
70    border-top-left-radius: 30px;
71    border-bottom-left-radius: 30px;
72  }

74  .navbar ul li:hover,
75  .navbar ul li:hovered
76  {
77    background: var(--primaria);
78  }
79
80  .navbar ul li:nth-child(1)
81  {
82    margin-bottom: 70px;
83    pointer-events: none;
84  }
85
86  .navbar ul li a
87  {
88    position: relative;
89    display: block;
90    width: 100%;
91    display: flex;
92    text-decoration: none;
93    color: var(--primaria);
94  }
```

# Public

---

## CSS

Estilização atribuídas ao modal.

```
1064     .modal
1065     {
1066         width: 450px;
1067         position: fixed;
1068         top: -50%;
1069         left: 50%;
1070         transform: translate(-50%, -50%);
1071         transition: top 0.5s ease-in-out;
1072         border: 1px solid #ccc;
1073         font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
1074         border-radius: 10px;
1075         z-index: 3;
1076         background-color: #fff;
1077     }
```

# Views

---

✓ views	
<> alterarfuncionario.ejs	Alterar Funcionário
<> alterarprojeto.ejs	Alterar Projeto
<> atribuir.ejs	Atribuir
<> graficos.ejs	Gráficos
<> home.ejs	Home
<> index.ejs	Index
<> novaFuncao.ejs	Nova Função
<> novaGovernanca.ejs	Nova Governança
<> novo.ejs	Novo
<> novoFuncionario.ejs	Novo Funcionário
<> novoProjeto.ejs	Novo Projeto
<> tabelaFuncionarios	Tabela Funcionários

# Views

---

O EJS é uma engine de visualização no qual é possível utilizar códigos em javascript no html de nossas páginas.

```
--> 22   <!--Menu vertical e horizontal-->
23   <div class="container">
24 >     <div class="navbar active">...
69     </div>
70     <!--Barra de pesquisa-->
71     <div class="main active">
72 >       <div class="topbar">...
85       </div>
86       <!--Entidade de adicionar novo funcionario-->
87 >         <div class="contents">...
158         </div>
159       </div>
160     </body>
161   </html>
```

Nota-se três “`<div>`” principais: “`navbar active`”, “`topbar`” e “`contents`”.

1. `Navbar`= configurações dos menus laterais e superiores;
2. `Topbar`= configuração da barra de pesquisa e estrutura da tela de perfil do usuário;
3. `Contents`= conteúdos que estão em cada formulário da aplicação.

# Index

---

Esse arquivo executa diversas funções na aplicação.

- Cria os servidores do node;
- Comporta os módulos e alguns métodos;
- Arquivos de rotas;
- Utiliza o EJS para gerar as páginas da aplicação;
- Utiliza os arquivos estáticos;
- Utiliza as rotas importadas dos arquivos de rotas (routes);
- Renderiza todas as páginas;
- Inicia o servidor.

```
25  app.use('/projects', projectsRoutes);
26  app.use('/employees', employeesRoutes);
27  app.use('/governance', governanceRoutes);
28  app.use('/role', roleRoutes);
29  app.use('/alocation', alocationRoutes);
30
```

Exemplo de uma das funções

# Packpage.json

---

O JSON tem a função de armazenar e transportar os dados do Backend.

```
1  {
2      "name": "backendgakki",
3      "version": "1.0.0",
4      "description": "",
5      "main": "index.js",
6      ▷ Depurar
7      "scripts": {
8          "test": "echo \\"$Error: no test specified\\" && exit 1",
9          "start": "nodemon index.js"
10     },
11     "keywords": [],
12     "author": "",
13     "license": "ISC",
14     "dependencies": {
15         "body-parser": "^1.20.0",
16         "cors": "^2.8.5",
17         "ejs": "^3.1.8",
18         "express": "^4.18.1",
19         "sqlite3": "^5.0.8",
20         "sweetalert": "^2.1.2"
21     }
}
```

# Desenvolvedores

---

Frederico  
Schur



Luiz  
Carlos



Luiz  
Augusto



Beatriz  
Hirasaki



Sérgio  
Lucas



Pedro  
Silva



Thainá Lima

# Agradecimentos finais

---

Agradecemos ao Professor Orientador Egon Daxbacher pela ajuda durante os dois primeiros módulos da faculdade, e os excelentes ensinamentos que agregaram imensamente para a construção profissional de cada integrante do time.

Além disso, reconhecemos também os professores que agregaram para a nossa aprendizagem, como a Professora Fabiana Martins de Oliveira, Professor Henrique Paiva, Professor Pedro Toberga, Professor Flávio Azevedo.