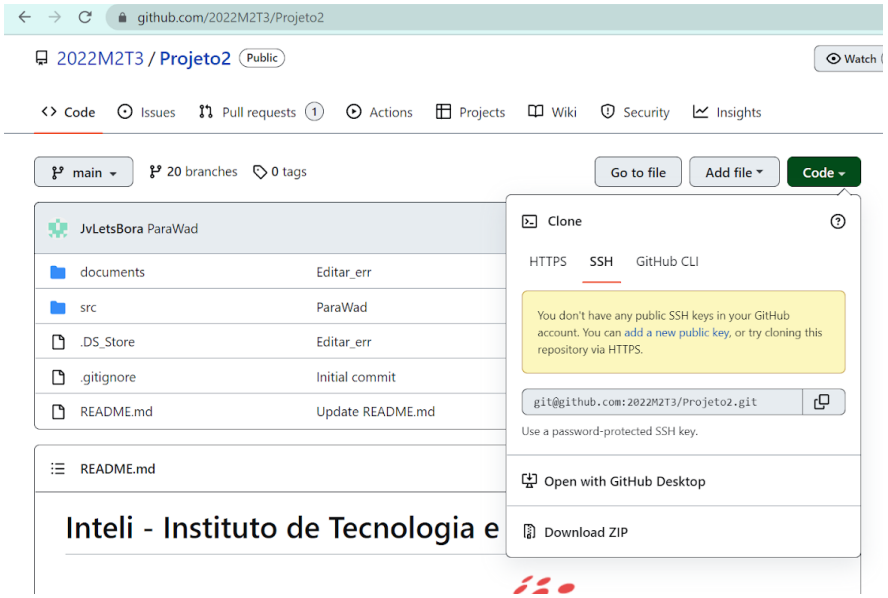
Manual de Manutenção do Código

Yamaha Planejamento de Capacidade

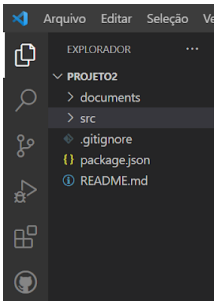
Grupo cinquentinha

Instalação do repositório

Ao entrar no link do projeto (https://github.com/2022M2T3/Projeto2), há a página de arquivos. Ao clicar em cima de **code,** instale o projeto clicando em **Download ZIP.** Depois de instalado, é só descompactar a página, para descompactá-la clique com o botão direito do mouse em cima da pasta e escolha a opção **extrair tudo**.

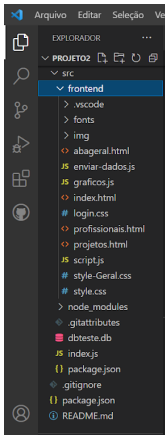


Pastas do código



Dentro do repositório do nosso projeto estão duas páginas uma com todos os documentos da solução, WAD, Manual do Usuário e manual do administrador, já dentro da página **src** se encontram todas as nossas páginas executáveis do projeto**.**

Elas então divididas pela pasta **frontend.** Dentro do **frontend** estão todas as pastas que interferem na interface, como imagens, fontes e o próprio código, e tudo que está fora do frontend faz parte do back-end, o “dbteste.db” é o nosso banco de dados e “index.js” tem as nossas rotas, faz a conexão entre o código e o banco de dados.



Front-end:

Header (cabeçalho): Parte estática que fica no começo das páginas, está presente em todas as abas(exceto login) e contém a barra de pesquisas.

```
<header> <!-- nosso HEADER estático -->
    <div class="barra-pesqusia">
        <button class="btn-meu"><span class="material-symbols-outlined">search</span></button>

        <input class="barra-texto" type="text" name="" placeholder="Pesquisar projeto">

        <span class="material-symbols-outlined centro" id="fitro-posicao">filter_list</span>

    </div>
    <nav class="bar">
```

Menu: Local principal para a alteração de profissionais e projetos. A aba fica no canto direito das páginas, presente na maioria das páginas, usado para passar da aba geral para a visualização dos funcionários, por exemplo.

```
<div id="menu">  <!-- Menu lateral. Aqui são colocados todos os botões e a logo da Yamaha -->
    <div class="centro"><img id="logo" src="img/logo2.png" alt=""></div> <!-- logo -->
       <nav id="nav-menu"> <!-- botões -->
           <ul class="centro">
           <div class="profi">
               <li class="ordem"><a class="centra ordem" href="profissionais.html"><span
                           class="material-symbols-outlined">
                           person_search
                       </span>Profissionais</a></li>
           </div>
           <div class="ative-projeto">
               <li class="ordem"><span class="material-symbols-outlined">
                       task
                   </span>Projetos</li>
           </div>
           </ul>
        </nav>
</div>
```

Calendário dinâmico: Fica dentro da aba geral, na parte inferior, ele mostra a duração do projeto e consegue fornecer uma visão mais detalhada da

quantidade de projetos e a duração dele(ele age em conjunto com uma função do gráfico)

```
<!--calendário dinâmico-->
<div id="calendario"> <!-- segunda parte do conteúdo não estatico da aplicação, responsável pelo calendário dinâmico -->
    <div class="month-div" id="add">
        <ul class="month">
            <li><h3> jan </h3></li>
        <li><h3> fev </h3></li>
        <li><h3> mar </h3></li>
        <li><h3> abr </h3></li>
        <li><h3> mai </h3></li>
        <li><h3> jun </h3></li>
        <li><h3> jul </h3></li>
        <li><h3> ago </h3></li>
        <li><h3> set </h3></li>
        <li><h3> out </h3></li>
        <li><h3> nov </h3></li>
        <li><h3> dez </h3></li>
        </ul>
    </div>

    <u id="c-todos"> <!-- todos os projetos exibidos no calendário são alocados aqui -->
    </u>
</div>
```

Modais: Os modais são as telas pequenas que sobrepõe as páginas, todos modais foram feitos com o “Bootstrap” e funcionam de maneira similar, “modal-header” mostra a divisão superior do modal, “modal-body” representa a parte onde o conteúdo em si está.

```
<!-- modal dos gráficos -->
<div class="modal fade" id="modalgraphs" role="dialog">
    <div class="modal-dialog">
        <div class="modal-content">
        <div class="modal-header">
        <button type="button" class="close" data-dismiss="modal">&times;</button>
        </div>
        <div class="modal-body">
            <button class="hand_hover" type="submit" data-dismiss="modal">Alocar</button>
            <div id="alocacao">
                <label for="alocacaos">Alocar: </label>
                <select name="alocacaos" id="alocacaos">
                </select>
                <div id="nfuncionarios">
                    <label for="nfunciona">Horas Totais a Alocar: </label>
                    <input type="number" placeholder="Por Funcionario" name="nfunciona">
                </div>
            </div>
```

“Modal-footer” representa a parte inferior do modal

```
            <div class="modal-footer">
              <button type="submit" data-dismiss="modal" action="/profissionais" onclick="enviarDados();"> Criar </button>
            </div>
```

Back-end:

Rotas:

Método POST: utilizado para adicionar os funcionários na tabela do banco de dados

```
// Insere um registro
app.post('/profissionais/adicionar', urlencodedParser, (req, res) => {
    res.statusCode = 200;
    res.setHeader('Access-Control-Allow-Origin', '*'); // Isso é importante para evitar o erro de CORS

    sql = `INSERT INTO PROFISSIONAIS (nome, area, tipo, estado) VALUES ('${req.body.nome}', '${req.body.area}', '${req.body.tipo}', '${req.body.
    estado}')`;
    var db = new sqlite3.Database(DBPATH); // Abre o banco
    console.log(sql);
    db.run(sql, [],  err => {
        if (err) {
            throw err;
        }
        else console.log(sql);
    });
    db.close(); // Fecha o banco
    res.end();
});
```

Método GET: Utilizado para pegar as informações fornecidas pelo banco de dados.

```
// Retorna todos registros
app.get('/profissionais', (req, res) => {
    res.statusCode = 200;
    res.setHeader('Access-Control-Allow-Origin', '*'); // Isso é importante para evitar o erro de CORS

    var db = new sqlite3.Database(DBPATH); // Abre o ban    co
  var sql = 'SELECT * FROM PROFISSIONAIS ORDER BY idFunc';
    db.all(sql, [],  (err, rows ) => {
        if (err) {
            throw err;

        }
    res.json(rows)
    });
    db.close(); // Fecha o banco
});
```

Método PATCH: Utilizado para editar um funcionário que já está presente dentro do banco de dados.

```
// Atualiza um registro
app.patch('/profissionais/atualizar', urlencodedParser, (req, res) => {
    res.statusCode = 200;
    res.setHeader('Access-Control-Allow-Origin', '*'); // Isso é importante para evitar o erros

    sql = `UPDATE PROFISSIONAIS SET nome = '${req.body.nome}', area = '${req.body.area}', tipo = '${req.body.tipo}', estado = '${req.body.estado}'
    WHERE idFunc = ${req.body.idFunc}`;
    var db = new sqlite3.Database(DBPATH); // Abre o banco
    db.run(sql, [],  err => {
        if (err) {
            throw err;
        }
        res.end();
    });
    db.close(); // Fecha o banco
});
```

Os métodos utilizados para os projetos e para a alocação são semelhantes àqueles mostrados acima, mudando apenas as informações do banco de dados

Ajax:

Método POST usado para adicionar o projeto direto do front-end para o banco de dados:

```
//POST PROJETOS
function enviaProjeto(){
    const nomeProj = document.getElementById("nomeProj").value
    const unidadeProj = document.getElementById("unidadeProj").value
    const diaInicioProj = document.getElementById("diaInicioProj").value
    const mesInicioProj = document.getElementById("mesInicioProj").value
    const anoInicioProj = document.getElementById("anoInicioProj").value
    const mesFimProj = document.getElementById("mesFimProj").value
    const anoFimProj = document.getElementById("anoFimProj").value
    const areaProj = document.getElementById("areaProj").value
    url = "/projetos/adicionar"
    $.ajax({
        type: "POST",
        url: url,
        contentType: "application/json; charset=utf-8",
        dataType: "json",
        data: JSON.stringify(
            {
                "nome": nomeProj,
                "area": areaProj,
                "mesInicio": mesInicioProj,
                "anoInicio": anoInicioProj,
                "mesFim": mesFimProj,
                "anoFim": anoFimProj,
                "unidade": unidadeProj
            }
        )
    });
}
```

Método PATCH para editar um funcionário previamente adicionado no banco de dados diretamente pelo front-end:

```
function atualizarProfissional(idP){
        let idNP = code(idP,"_F_NP") // Nome
        let idAI = code(idP,"_F_AI") // Area
        let idMF = code(idP,"_F_MF") // CLT
        let idUP = code(idP,"_F_UP") // Estado



    const nomeP = document.getElementById(idNP).value;
    const area = document.getElementById(idAI).value;
    const tipo = document.getElementById(idMF).value;
    const estado = document.getElementById(idUP).value;

    url = "/profissionais/atualizar"
    $.ajax({
        type: "PATCH",
        url: url,
        contentType: "application/json; charset=utf-8",
        dataType: "json",
        data: JSON.stringify(
            {
                "nome": nomeP,
                "area": area,
                "tipo": tipo,
                "estado": estado,
                "idFunc": idP,
            }
            )
        });
        window.location.reload(); //essa função mágica faz com que atualize a página, sem haver a necessidade de recarregar manualmente para ver a
        adição do profissional na tabela
}
```

Função para permitir a edição do projeto

```
// Edita os projetos
function editar(idP,inicio,fim,anoI,anoF){
    let idNP = code(idP,"NP")
    let idMI = code(idP,"MI")
    let idAI = code(idP,"AI")
    let idMF = code(idP,"MF")
    let idAF = code(idP,"AF")
    let idUP = code(idP,"UP")
    let anoFim = anoF
    let anoInicio = anoI
    let idProjeto = "projeto_"+idP
    let mesInicio = inicio.toString()
    let mesFim = fim.toString()
    let lista = nomesP.get("nP"+idP) // indice [0] reprensta o nome do projeto; indice [1] reprensta a cidade do projeto
    let selecao =""
    mesInicio = eval("meses.n"+mesInicio)
    mesFim = eval("meses.n"+mesFim)
    if(lista[1] == "am"){
        selecao = "<select id=\""+idUP+"\"><option value=\"am\">AM</option><option value=\"sp\">SP</option><option value=\"ambos\">Ambos</
        option> </select>"
    }else{selecao = "<select id=\""+idUP+"\"><option value=\"sp\">SP</option><option value=\"am\">AM</option><option value=\"ambos\">Ambos</
    option> </select>"}
    document.getElementById(idProjeto).innerHTML = "<form method=\"post\"><td id=\"coldata\" class=\"aba\"><input value=\""+lista[0]+"\"
    class=\"limitador\" id=\""+idNP+"\" type=\"text\" placeholder=\""+lista[0]+"\"></td>  <td id=\"coldata\" class=\"aba\"><input value=\""
    +mesInicio+"\" class=\"limitador\" id=\""+idMI+"\" type=\"text\" placeholder=\""+mesInicio+"\"><input value=\""+mesFim+"\"
    class=\"limitador\" id=\""+idMF+"\" type=\"text\" placeholder=\""+mesFim+"\"></td>  <td id=\"coldata\">Auto</td> <td>"+selecao+"</td>
    <td><input value=\""+anoInicio+"\" class=\"limitador\" id=\""+idAI+"\" type=\"text\" placeholder=\""+anoInicio+"\"> a <input value=\""+anoFim
    +"\" class=\"limitador\" id=\""+idAF+"\" type=\"text\" placeholder=\""+anoFim+"\"></td> <td><div class=\"linha\"> <button
    class=\"hand_hover\" type=\"submit\" onclick=\"atualizar("+idP+");\" >Confirmar</button> <a href=\"/projetos.html\">Voltar</a> </div></td></
    form>"

}
```

Método PATCH de projetos:

```
// Atualiza os projetos
function atualizar(idP){
    let idNP = code(idP,"NP")
    let idMI = code(idP,"MI")
    let idAI = code(idP,"AI")
    let idMF = code(idP,"MF")
    let idAF = code(idP,"AF")
    let idUP = code(idP,"UP")



    const nomeP = document.getElementById(idNP).value;
    const mesInicio = document.getElementById(idMI).value;
    const mesFim =  document.getElementById(idMF).value;
    const anoInicio = parseInt(document.getElementById(idAI).value);
    const anoFim = parseInt(document.getElementById(idAF).value);
    const unidade = document.getElementById(idUP).value

    url = "/projetos/atualizar"
    $.ajax({
        type: "PATCH",
        url: url,
        contentType: "application/json; charset=utf-8",
        dataType: "json",
        data: JSON.stringify(
            {
                "nome": nomeP,
                "area": "Teste",
                "mesInicio": mesInicio,
                "anoInicio": anoInicio,
                "mesFim": mesFim,
                "anoFim": anoFim,
                "unidade": unidade,
                "idProject": idP,
            }
            )
```

Método DELETE de projetos:

```
// Exclui os projetos
    function excluirProjeto(idP){
        url = "/projetos/deletar"
        $.ajax({
            type: "DELETE",
            url: url,
            contentType: "application/json; charset=utf-8",
            dataType: "json",
            data: JSON.stringify(
                {
                    "idProject": idP
                }
            )
        });
    }
```

Gráficos:

Função necessária para que a geração de gráficos seja feita utilizando o banco de dados.

```
function generateGraphics(){
    let requestGraph = new XMLHttpRequest();
    /* informações que serão coletadas do banco de dados para gerar o gráfico*/

    requestGraph.onload = function(){
        let dados = JSON.parse(this.response)
        let tamanhoDados = dados.length
        let arrayHoras = []
        let arrayNomes = []
        for(let i = 0; i < tamanhoDados; i++){
            arrayHoras.push(dados[i].somaHoras)
            arrayNomes.push(dados[i].nome)
        }
        graficos(arrayNomes, arrayHoras);
    }

    /*rota que o gráfico será exibidp*/
```

Função responsável pelas cores do gráfico:

```
function graficos(nomeProj, horasProj){
    new Chart(document.getElementById("bar-chart"), {
        type: 'bar',
        data: {
            labels: nomeProj,
            datasets: [
                {
                    label: "Horas",
                    backgroundColor: ["#2d3870", "rgb(82, 111, 255)", "#8d8d8d", "#ef0404", "#39cfd75f"],
                    data: horasProj
                }
            ]
        },
        options: {
            legend: { display: false },
            title: {
                display: true,
                text: 'Número de horas/ projeto'
            }
        }
    });
}
```

Explicado nos comentários do código:

```
function generateLines(){ // Essa Função é chamada no momento do carregamento da tag body na aba visão geral, relacionando a duração com o tamanho
do grafico
    let requestLines = new XMLHttpRequest();
    requestLines.onload = function(){
        let dados = JSON.parse(this.responseText) // Essa linha representa a devolutiva da consulta ao banco de dados amarzenada em um array com
        varios dicionários.
        let tamanhoDados = dados.length
        let anos = [] // Guarda os anos dos projetos. Ela é passada como argumento na funsao criarCalendario()
        for(let i = 0; i < tamanhoDados; i++){
            anos.push(dados[i].anoFim)
            criarProjeto(dados[i].nome, dados[i].anoInicio, dados[i].anoFim, eval("meses."+dados[i].mesInicio), eval("meses."+dados[i].mesFim),i)
        }
        var novaArr = anos.filter(function(este, i) {
            return anos.indexOf(este) === i;
        });
        for (let i = 1; i < novaArr.length; i++) {
            criarCalendario(novaArr[i],i)

        }
    }
    /*rota que será exibida*/

    url = "/projetos/timeline"
    requestLines.open("GET", url, true);
    requestLines.send();
}
```

Criação do calendário

Cria um novo item no calendário, faz com que a duração do projeto respeite os seus respectivos meses no gráfico e envia pro banco de dados.

```
function criarCalendario(fim,anos){
    var largura = window.screen.height
    var larguraAjuste = largura - (largura*0.15)
    var percentual = 1/larguraAjuste
    let y = (50/(anos+1*(percentual+100)))
    window.document.getElementById("add").innerHTML += "<div class=\"step-c\">"+"|----"+String(fim)+"---->"+"</div><ul class=\"month\"><li><h3> jan </h3></li><li><h3> fev </h3></
    li><li><h3> mar </h3></li><li><h3> abr </h3></li><li><h3> mai </h3></li><li><h3> jun </h3></li><li><h3> jul </h3></li><li><h3> ago </h3></li><li><h3> set </h3></li><li><h3> out </h3></
    li><li><h3> nov </h3></li><li><h3> dez </h3></li></ul>"
    $(".month").css("min-width",(y)+"%"+"!important")
}
```

Função do calendário responsável pelas barras do calendário descreve a duração dos projetos e mostra quando o período é superior a um ano.

```
function criarProjeto(nomedb, anoIniciodb, anoFimdb, mesIniciodb, mesFinaldb,i){
    var n = i +1
    var largura = window.screen.height
    var larguraAjuste = largura - (largura*0.15)
    var nP = nomedb; // np = Nome do projeto
    var anoInicio = parseInt(anoIniciodb);
    var anoFim = parseInt(anoFimdb);
    var anoRes = (anoFim - anoInicio);
    var mesInicio = parseInt(mesIniciodb)
    var mesFinal = parseInt(mesFinaldb)
    var tamanho = ((mesFinal - mesInicio)+1)
    var percentual = 1/larguraAjuste
    console.log("Projeto: "+nP+ " - "+n)
    console.log("AnoResto: "+anoRes)
    if( anoRes == 0 ){
        tamanho = ((percentual+7)*tamanho)
        }
    else{
        console.log("ELSE AQUI ESTOU")
        tamanho = (anoRes*((percentual+7)*11) + ((percentual+7)*tamanho))
        }
    mesInicio+=1
    console.log("Antes: "+mesInicio)
    mesInicio = ((percentual+7)*(mesInicio-1))
    console.log("Depois: "+mesInicio)
    document.getElementById("c-todos").innerHTML += "<li><div>"+nP+"</div></li>";
    console.log("Tamanho:"+tamanho)
    if(tamanho>80){
        if(btnAtivado == false){
            gerarBtn()
            btnAtivado = true
        }
        tamanho = tamanho + (percentual+19) * anoRes
        $("#c-todos > li:nth-child("+n+")").css("width",(tamanho)+"%");
        $("#c-todos > li:nth-child("+n+")").css("margin-left",parseInt(mesInicio)+"%")
    }else{ $("#c-todos > li:nth-child("+n+")").css("width",(tamanho)+"%");
    $("#c-todos > li:nth-child("+n+")").css("margin-left",parseInt(mesInicio)+"%")}
}
```